Texto

O conteúdo gerado por IA pode estar incorreto.

**BRENO MANOEL, PEDRO PESSOA, JOSÉ STENIO, DIEGO BRAGA, KAUÃ OLIVEIRA, PIETRO MORI, RAFAEL CASTRO, HENRIQUE LOPES.**

**TEMA 8 – SEGURANÇA**

**BRASÍLIA**

**2025**

**BRENO MANOEL, PEDRO PESSOA, JOSÉ STENIO, DIEGO BRAGA, KAUÃ OLIVEIRA , PIETRO MORI, RAFAEL CASTRO**

**TEMA 8 – SEGURANÇA**

Trabalho da disciplina

Introdução a computação do

curso ciência da computação no UNICEUB

**BRASÍLIA**

**2025**

**RESUMO**

A segurança em sistemas operacionais é fundamental para garantir a integridade, confidencialidade e disponibilidade dos dados. O ambiente de segurança engloba mecanismos, políticas e práticas que protegem os sistemas contra ameaças externas e internas, invasores e falhas acidentais. Com o aumento da conectividade, os riscos cresceram significativamente, exigindo estratégias eficazes de mitigação.

As principais ameaças incluem *malwares* (vírus, worms, ransomware), ataques de negação de serviço (DoS/DDoS), phishing e exploração de vulnerabilidades. Sistemas operacionais modernos combatem essas ameaças com atualizações automáticas, antivírus e políticas de acesso restrito.

Os invasores variam desde *hackers* maliciosos (chapéu preto) até agentes patrocinados por governos, utilizando métodos como força bruta, injeção de código e acesso remoto. Medidas como autenticação multifator e monitoramento de logs são essenciais para neutralizá-los. Além disso, perdas acidentais de dados—causadas por exclusões involuntárias, falhas de hardware ou desastres naturais—exigem backups frequentes e sistemas de redundância.

A criptografia surge como um pilar da defesa, dividindo-se em simétrica (AES) e assimétrica (RSA), garantindo confidencialidade e integridade. Mecanismos como DEP e ASLR complementam a proteção contra explorações de código.

Conclui-se que a segurança é um processo contínuo, dependente de atualizações, boas práticas e conscientização dos usuários. A combinação de técnicas avançadas e gestão de riscos é crucial para a proteção eficaz dos sistemas operacionais.

Palavras-chave: Segurança. Sistemas operacionais. Criptografia. Malwares. Invasores.

Sumário

[Introdução 4](#_Toc195200154)

[1.Ambiente de Segurança 5](#_Toc195200155)

[2.Criptografia básica 8](#_Toc195200156)

[3. Mecanismos de proteção 11](#_Toc195200157)

[4. Autenticação 13](#_Toc195200158)

[5.Ataques de dentro dos sistema 16](#_Toc195200159)

[6. Explorando Erros de Código: 18](#_Toc195200160)

[7.Malwares 22](#_Toc195200161)

[8. Explorando Criptografia Básica na Defesa da Segurança: 27](#_Toc195200162)

[Conclusão 31](#_Toc195200163)

[REFERÊNCIAS 33](#_Toc195200164)

# 

# Introdução

A segurança em sistemas operacionais configura-se como um dos pilares essenciais para a garantia da integridade, confidencialidade e disponibilidade de dados na era digital. Com o exponencial crescimento da conectividade e da dependência de sistemas computacionais, os riscos associados a ataques cibernéticos e falhas operacionais tornaram-se cada vez mais complexos e impactantes. Nesse contexto, compreender os mecanismos de proteção, as ameaças emergentes e as estratégias de mitigação transforma-se em uma necessidade crítica para profissionais e usuários de tecnologia.

Este trabalho aborda os fundamentos da segurança em sistemas operacionais, explorando desde as ameaças tradicionais — como *malwares* (vírus, worms, ransomware), ataques de negação de serviço (DoS/DDoS) e exploração de vulnerabilidades — até os perfis de invasores (hackers de chapéu preto, agentes patrocinados por governos) e suas táticas (engenharia social, injeção de código). Além disso, discute-se a importância de medidas preventivas contra perdas acidentais de dados, como backups e redundância, e o papel central da criptografia (simétrica e assimétrica) na proteção de informações.

Baseado no capítulo 9 do livro *Sistemas Operacionais Modernos* de Andrew Tanenbaum, o estudo também analisa mecanismos de defesa avançados, incluindo autenticação multifator, Prevenção de Execução de Dados (DEP) e Randomização do Layout do Espaço de Endereçamento (ASLR). A abordagem destaca a segurança como um processo dinâmico, que exige atualizações constantes, adoção de boas práticas e conscientização coletiva.

Por fim, o trabalho visa não apenas elucidar os desafios contemporâneos, mas também fornecer insights sobre como sistemas operacionais modernos podem ser fortalecidos para resistir a um cenário de ameaças em constante evolução.

Palavras-chave: Segurança cibernética; Sistemas operacionais; Criptografia; Malwares; Proteção de dados.

# 1.Ambiente de Segurança

A segurança em sistemas operacionais é um dos pilares fundamentais para garantir a integridade, confidencialidade e disponibilidade dos dados. O ambiente de segurança refere-se ao conjunto de mecanismos, políticas e práticas implementadas para proteger sistemas contra ameaças, invasores e falhas acidentais.

Com a crescente conectividade e dependência de sistemas computacionais, os riscos aumentaram significativamente, tornando essencial compreender as principais ameaças e estratégias para mitigá-las. Entre os aspectos mais críticos do ambiente de segurança estão as ameaças externas e internas, os invasores que exploram vulnerabilidades e os erros acidentais que podem comprometer dados.

**1.2 Ameaças**

As ameaças em um ambiente de segurança podem ser classificadas em diferentes categorias, desde ataques intencionais até falhas não planejadas. As principais ameaças incluem:

* Malwares (Vírus, Worms, Trojans, Ransomware): Programas maliciosos que podem roubar dados, danificar arquivos ou comprometer a integridade do sistema.
* Ataques de Negação de Serviço (DoS e DDoS): Sobrecarga intencional de servidores para torná-los inacessíveis.
* Phishing e Engenharia Social: Tentativas de enganar usuários para obter credenciais ou informações sensíveis.
* Exploração de Vulnerabilidades: Uso de falhas no sistema para obter acesso não autorizado ou elevar privilégios.

Os sistemas operacionais modernos implementam diversas medidas para mitigar essas ameaças, como atualizações automáticas, antivírus integrados e políticas de acesso restrito.

**1.3 Invasores**

Os invasores são indivíduos ou grupos que buscam explorar falhas de segurança para obter vantagens ilícitas. Eles podem ser classificados em diferentes perfis:

* Hackers de chapéu preto: Atuam de forma ilegal, explorando vulnerabilidades para roubo de dados, fraudes ou sabotagem.
* Hackers de chapéu branco: Especialistas em segurança que identificam falhas para corrigi-las.
* Hackers de chapéu cinza: Podem atuar de forma ética ou não, dependendo da situação.
* Atores patrocinados por governos: Realizam espionagem cibernética ou ataques a infraestrutura de outros países.

Os métodos mais comuns utilizados pelos invasores incluem:

* Ataques de força bruta para quebrar senhas.
* Injeção de código malicioso (SQL Injection, Cross-Site Scripting).
* Acesso remoto via exploração de falhas (RCE - Remote Code Execution).

Para mitigar ações de invasores, os sistemas operacionais empregam técnicas como autenticação multifator, monitoramento de logs e restrições de acesso baseadas em privilégios mínimos.

**1.4 Perda Acidental de Dados**

Nem todas as ameaças ao ambiente de segurança vêm de invasores. Muitas vezes, erros humanos ou falhas técnicas podem levar à perda de dados críticos. Entre as principais causas estão:

* Exclusão acidental: Usuários podem apagar arquivos importantes sem querer.
* Falhas de hardware: Discos rígidos, SSDs e memórias podem falhar, causando perda irreversível de informações.
* Erros de software: Atualizações mal feitas ou bugs podem corromper arquivos e sistemas.
* Desastres naturais: Incêndios, enchentes ou quedas de energia podem afetar servidores e dispositivos de armazenamento.

Medidas preventivas incluem:

* Backups frequentes em dispositivos locais e na nuvem.
* Sistemas de redundância para evitar falhas únicas.
* Uso de sistemas de arquivos seguros, como NTFS e EXT4, que minimizam a corrupção de dados.
* Políticas de controle de acesso para restringir ações que possam levar à exclusão indevida de arquivos.

# 2.Criptografia básica

**2.1 Introdução à Criptografia**

A criptografia surgiu na **Antiguidade** como forma de proteger mensagens estratégicas. Um dos primeiros exemplos foi a **Cifra de César** (século I a.C.), que deslocava letras do alfabeto para codificar mensagens militares.

* **Definição moderna**: Técnica que usa algoritmos matemáticos para proteger dados.
* **Objetivos (CIA Triad)**:
  + **Confidencialidade**: Garantir que apenas destinatários autorizados acessem a informação.
  + **Integridade**: Assegurar que os dados não sejam alterados indevidamente.
  + **Autenticidade**: Verificar a origem legítima da informação.

**2.2 Tipos de Criptografia**

**A) Criptografia Simétrica**

* **Origem**: Utilizada desde a **Máquina Enigma** (1930–1945), que cifrava mensagens com rotores e uma chave compartilhada.
* **Características**:
  + Mesma chave para cifrar e decifrar.
  + Exemplos históricos: **DES** (1977), substituído pelo **AES** (2001).
* **Vantagens**: Rápida e eficiente para grandes volumes de dados.
* **Desvantagens**: Risco na distribuição segura da chave.

**B) Criptografia Assimétrica**

* **Marco histórico**: Proposta por **Diffie e Hellman (1976)** e implementada no **RSA (1977)**.
* **Características**:
  + Par de chaves (pública e privada).
  + Exemplos: **RSA** (usado em certificados digitais) e **ECC** (comum em dispositivos móveis).
* **Vantagens**: Elimina o problema da troca de chaves.
* **Desvantagens**: Mais lenta que a simétrica.

**2.3 Funções de Via Única (One-Way Functions)**

* **Contexto histórico**:
  + Os primeiros hashes criptográficos, como **MD5 (1991)**, foram criados para verificar integridade de arquivos.
  + **SHA-1 (1995)** e **SHA-256 (2001)** surgiram para substituir hashes vulneráveis.
* **Definição**: Fáceis de calcular, mas difíceis de inverter (ex: transformar uma senha em hash).
* **Aplicações**:
  + Armazenamento seguro de senhas (com *salting*).
  + Verificação de integridade em downloads (ex: checksums).

**2.4 Módulo de Plataforma Confiável (TPM)**

* **Surgimento**: Criado em **2009** pelo **Trusted Computing Group** para segurança baseada em hardware.
* **Funções**:
  + Armazenar chaves criptográficas de forma isolada.
  + Verificar integridade do sistema durante o *boot*.
* **Aplicações**:
  + **BitLocker** (Windows) e **LUKS** (Linux) usam TPM para criptografia de disco.
  + Autenticação biométrica (ex: Windows Hello).

**2.5 Ataques Comuns e Defesas ao Longo da História**

* **Força bruta**:
  + **Histórico**: Quebra do DES (56 bits) em 1998 mostrou a necessidade de chaves maiores.
  + **Defesa moderna**: Uso de AES-256 ou ChaCha20.
* **Ataques de dicionário**:
  + **Histórico**: Senhas em texto simples nos anos 1980 levou ao uso de *hashing* com *salting*.
  + **Defesa moderna**: Bcrypt ou Argon2.
* **MITM (Man-in-the-Middle)**:
  + **Histórico**: Ataques em redes Wi-Fi (WEP) nos anos 2000.
  + **Defesa moderna**: Certificados SSL/TLS e autenticação mútua.

# 3. Mecanismos de proteção

A proteção em sistemas operacionais é um tema essencial para garantir a integridade, confidencialidade e disponibilidade das informações. Os sistemas operacionais modernos implementam diversos mecanismos de proteção para controlar o acesso a recursos e evitar ações maliciosas ou acidentais que possam comprometer a segurança. Este trabalho discute os principais mecanismos de proteção, conforme apresentados no livro "Sistemas Operacionais Modernos" de Andrew Tanenbaum.

**3.1 Domínios de Proteção**

O conceito de domínio de proteção define um conjunto de objetos e as operações permitidas sobre eles. Cada processo opera dentro de um domínio específico, e a troca de domínios pode ser regulada pelo sistema operacional para garantir segurança.

**3.2 Matriz de Acesso**

A matriz de acesso é uma estrutura de dados que mapeia os direitos de acesso entre domínios e objetos. Cada linha representa um domínio e cada coluna representa um objeto, indicando as permissões concedidas.

**3.3 Lista de Controle de Acesso (ACL)**

A ACL associa a cada objeto uma lista que especifica quais usuários ou processos podem acessá-lo e quais operações podem ser realizadas. Esse mecanismo é amplamente utilizado em sistemas de arquivos.

**3.4 Lista de Capacidades**

Ao contrário da ACL, que é baseada nos objetos, a lista de capacidades é associada aos domínios, definindo os recursos que cada domínio pode acessar.

**3.5 Base Computacional Confiável (TCB)**

A TCB compreende os componentes críticos do sistema operacional responsáveis por garantir a segurança. Inclui o kernel, gerenciadores de processos e memória, entre outros elementos essenciais.

**3.6 Modelos de Segurança**

Vários modelos de segurança foram desenvolvidos para estruturar políticas de proteção. Os principais são:

* **Modelo Bell-LaPadula**: Foca na confidencialidade e impede a leitura não autorizada de informações.
* **Modelo Biba**: Prioriza a integridade e impede que dados de menor credibilidade alterem dados mais confiáveis.
* **Modelo Clark-Wilson**: Baseia-se na separação de deveres e transações bem-formadas.
* **Modelo Brewer-Nash (Chinese Wall)**: Evita conflitos de interesse ao restringir acessos baseados em interações anteriores.
* **Controle de Acesso Baseado em Papéis (RBAC)**: Define permissões com base em funções específicas dentro da organização.

**3.7 POLÍTICAS DE CONTROLE DE ACESSO**

Os sistemas operacionais empregam diferentes políticas de controle de acesso:

* **Controle de Acesso Discricionário (DAC)**: Permite que proprietários de objetos definam regras de acesso.
* **Controle de Acesso Obrigatório (MAC)**: O sistema determina as permissões com base em classificações de segurança.

A escolha entre essas políticas depende do nível de segurança exigido pelo ambiente computacional.

# 4. Autenticação

**4.1 O que é Autenticação?**

Autenticação é o processo de verificar a identidade de um usuário ou sistema antes de permitir o acesso.

Diferença entre autenticação e autorização:

* Autenticação: Confirma se a identidade do usuário é verdadeira.
* Autorização: Define quais recursos o usuário pode acessar após a autenticação.

**4.2 Métodos de Autenticação**

A autenticação pode ser feita de três formas principais:

Baseada em conhecimento (algo que o usuário sabe):

* Senhas
* PINs

Baseada em posse (algo que o usuário tem):

* Tokens físicos
* Cartões inteligentes

Baseada em características biométricas (algo que o usuário é):

* Impressão digital
* Reconhecimento facial
* Leitura de íris

**4.3 Fatores de Autenticação**

Os sistemas modernos usam múltiplos fatores para aumentar a segurança:

Autenticação de um fator (SFA):

* Apenas um método de autenticação (ex: senha).
* Risco: Se a senha for descoberta, o sistema fica vulnerável.

Autenticação de dois fatores (2FA):

* Combina dois métodos, como senha + código SMS.
* Vantagem: Se um fator for comprometido, o outro ainda protege o sistema.

Autenticação multifator (MFA):

* Utiliza três ou mais fatores (ex: senha + biometria + token físico).
* Alta segurança: Mesmo que um fator seja hackeado, os outros continuam protegendo o usuário.

**4.4 Mecanismos de Autenticação em Sistemas Operacionais**

Os sistemas operacionais utilizam diferentes mecanismos para autenticar usuários:

Login tradicional:

* Usuário e senha armazenados localmente.
* Vulnerável a ataques se senhas fracas forem usadas.

Autenticação baseada em redes:

* Kerberos: Protocolo seguro que usa criptografia para evitar o envio de senhas em texto simples.

LDAP e Active Directory:

* Usados para autenticação centralizada em redes empresariais.
* Permitem controle e gerenciamento de acessos dentro de grandes organizações.

**4.5 Principais Ataques e Vulnerabilidades**

Mesmo com autenticação, existem ameaças que podem comprometer a segurança:

Ataques de força bruta e dicionário:

* Hackers testam milhões de senhas até encontrar a correta.
* Prevenção: Uso de senhas longas e bloqueio de múltiplas tentativas erradas.

Phishing:

* Golpes que enganam usuários para roubarem suas credenciais.
* Prevenção: Educação do usuário e autenticação multifator.

Keyloggers:

* Programas que capturam tudo o que o usuário digita, incluindo senhas.
* Prevenção: Uso de antivírus e autenticação sem digitação (ex: biometria).

**4.6 Melhores Práticas de Segurança**

Para evitar falhas na autenticação, os usuários e sistemas devem seguir boas práticas:

1. Criar senhas fortes (longas, com letras, números e símbolos).
2. Não reutilizar senhas em vários sistemas.
3. Utilizar autenticação de dois fatores (2FA) sempre que disponível.
4. Monitorar acessos suspeitos e relatar tentativas de login incomuns.
5. Atualizar senhas regularmente e usar gerenciadores de senha.

# 5.Ataques de dentro dos sistema

**5.1 Principais Tipos de Ataques Internos**

**Privilege Escalation**

* Exploração de falhas para ganhar acesso de root/admin.
* Exemplo: Buffer overflow em programas com SUID (ex: passwd).

**Trojan Horses (Cavalos de Troia)**

* Código malicioso disfarçado (ex: um "jogo" que executa rm -rf /).

**Backdoors**

* Brechas deixadas por desenvolvedores ou invasores (ex: contas ocultas).

**Spyware/Keyloggers**

* Monitoramento de teclas ou screenshots para roubo de credenciais.

**Ataques de Race Conditions**

* Exploração de janelas de tempo entre verificação e execução (ex: TOCTTOU).

**5.2 Casos Reais (Exemplos do Livro)**

**MINIX/Unix/Linux:**

* Exploração de scripts Set-UID (programas com permissão elevada).

**Windows:**

* DLL hijacking (substituição de bibliotecas dinâmicas).

**5.3 Como o SO Pode se Proteger?**

1. Controle de Acesso Granular (DAC/MAC).
2. Princípio do Menor Privilégio (usuários rodam com permissões mínimas).
3. Isolamento de Processos (sandboxing).
4. Auditoria de Logs (monitoramento de atividades suspeitas).

**5.4 Exemplo Técnico - Buffer Overflow**

**Como acontece?**

* Um processo escreve além do espaço alocado na pilha.
* Permite injetar código malicioso (ex: shellcode).
* Pode sobrescrever endereços de retorno e assumir controle do fluxo de execução.

**Proteções:**

* ASLR (aleatorização de memória).
* Stack Canaries (detecção de overflow).
* DEP (Prevenção de Execução de Dados).

# 6. Explorando Erros de Código:

A seção 9.6 do livro "Sistemas Operacionais Modernos" aborda as diversas maneiras pelas quais erros presentes no código de sistemas operacionais e aplicações podem ser explorados por atacantes para comprometer a segurança. O foco principal reside na subversão do sistema operacional, permitindo a execução de código malicioso com privilégios elevados.

Inicialmente, um atacante pode realizar uma **varredura de portas** para identificar serviços de rede ativos e potencialmente vulneráveis, como o Telnet. Em seguida, pode tentar obter acesso não autorizado através de ataques de **força bruta** contra as credenciais de login. Uma vez com algum nível de acesso, o atacante pode buscar explorar vulnerabilidades específicas no código.

**6.1 Ataque por transbordamento do buffer (Buffer Overflow):**

Uma das classes mais comuns de vulnerabilidades resulta da ausência de verificação de limites em vetores (arrays) em linguagens como C. O exemplo clássico ilustra a tentativa de acessar um índice fora dos limites de um array, o que leva à sobrescrita de memória adjacente.

Em um cenário de ataque, considere um procedimento A que aloca um buffer de tamanho fixo B na pilha para armazenar, por exemplo, o caminho completo de um arquivo. A sequência de chamada padrão envolve a colocação do endereço de retorno na pilha antes da transferência de controle para A. Se o procedimento A copiar para o buffer B uma quantidade de dados que excede seu tamanho, os dados transbordarão para além dos limites do buffer, potencialmente sobrescrevendo o endereço de retorno armazenado na pilha.

Um atacante pode fornecer dados maliciosos que, ao transbordarem o buffer, sobrescrevem o endereço de retorno com o endereço de seu próprio código (shellcode), que também foi injetado no buffer. Quando o procedimento A tenta retornar, em vez de voltar para o seu chamador original, a execução é desviada para o código do atacante.

Se o programa vulnerável possuir permissões elevadas (como SETUID root em sistemas Unix ou privilégios de Administrador no Windows), o código injetado pelo atacante também será executado com essas permissões, permitindo ações como a modificação de arquivos de sistema, instalação de backdoors ou a criação de um shell com privilégios elevados. A função de biblioteca gets em C, que não realiza verificação de limites ao ler uma string para um buffer, é um exemplo clássico de função vulnerável a ataques de buffer overflow. Mecanismos de defesa modernos incluem a Prevenção de Execução de Dados (DEP) e a Randomização do Layout do Espaço de Endereçamento (ASLR) para dificultar a exploração de buffer overflows.

**6.2 Ataques à cadeia de formato (Format String Attacks):**

A função printf em C permite a formatação de saída utilizando especificadores como %s para strings e %d para inteiros. Uma vulnerabilidade surge quando a cadeia de formato fornecida para a função printf é controlada pelo usuário e não é devidamente sanitizada.

Um especificador particularmente perigoso é o %n, que não produz saída, mas escreve o número de bytes já escritos pela função printf no endereço de memória apontado pelo argumento correspondente. Um atacante pode explorar essa funcionalidade para escrever valores arbitrários em locais específicos da memória.

No contexto de um ataque, se um programa vulnerável usa printf com uma cadeia de formato fornecida pelo usuário, o atacante pode inserir especificadores %n para sobrescrever o endereço de retorno da função printf na pilha. Ao fazer isso, o atacante pode redirecionar a execução do programa para um endereço de sua escolha, possivelmente contendo código malicioso. Essa técnica permite a execução de código arbitrário com os privilégios do programa vulnerável.

**6.3 Ataque de retorno à libc (Return-to-libc Attack):**

Esta técnica de ataque visa contornar proteções que impedem a execução de código diretamente na pilha. Em vez de injetar e executar código na pilha, o atacante manipula o endereço de retorno de uma função para apontar para uma função já existente em bibliotecas compartilhadas do sistema, como a libc em sistemas Unix.

O ataque geralmente envolve um "buffer overflow" ou um "ataque à cadeia de formato" para sobrescrever o endereço de retorno. Em seguida, o atacante configura a pilha de forma que os argumentos necessários para a função da biblioteca alvo (por exemplo, a função strcpy para copiar shellcode para o segmento de dados) estejam presentes. Ao retornar, a função da biblioteca é executada com os argumentos controlados pelo atacante.

O objetivo comum é utilizar funções da biblioteca para preparar o ambiente para a execução de código malicioso (shellcode) em uma região de memória executável, como o segmento de dados. Essa técnica é eficaz mesmo quando a execução de código na pilha é desabilitada.

**6.4 Ataque por transbordamento de números inteiros (Integer Overflow Attack):**

Computadores representam números inteiros com um número fixo de bits. Quando o resultado de uma operação aritmética (adição, multiplicação) excede a capacidade máxima de representação do tipo de dado inteiro, ocorre um "transbordamento de inteiro". Em linguagens como C, esses transbordamentos geralmente não são detectados pelo compilador ou em tempo de execução.

Atacantes podem explorar esses transbordamentos para manipular variáveis utilizadas em cálculos importantes, como o tamanho da memória a ser alocada. Por exemplo, em um programa que processa imagens, os parâmetros de altura e largura fornecidos pelo usuário podem ser multiplicados para determinar o tamanho do buffer necessário. Se os valores de altura e largura forem suficientemente grandes para causar um transbordamento no cálculo do tamanho, o programa pode alocar um buffer muito menor do que o necessário. Isso pode levar a uma condição de "buffer overflow" quando o programa tenta escrever a imagem no buffer alocado.

Outro cenário envolve transbordamentos em inteiros sinalizados, onde a soma de dois números positivos pode resultar em um número negativo, levando a comportamentos inesperados e potencialmente exploráveis.

**6.5 Ataques por injeção de código (Code Injection Attacks):**

Este tipo de ataque ocorre quando um atacante consegue inserir código malicioso em um sistema e fazer com que ele seja executado pelo programa alvo. Uma vulnerabilidade comum explorada é o uso de funções que invocam um shell de comandos para executar comandos externos, como a função system() em C.

Se um programa constroi um comando para ser executado pelo shell com base em dados fornecidos pelo usuário sem a devida validação ou sanitização, um atacante pode inserir comandos arbitrários junto com a entrada esperada. O exemplo clássico envolve um programa que copia arquivos usando o comando cp. Se um usuário malicioso fornecer como nome de arquivo de destino algo como "destino.txt ; rm -rf /", a função system() executará a seguinte sequência de comandos: cp origem.txt destino.txt ; rm -rf /. O comando rm -rf / tentará apagar recursivamente todos os arquivos do sistema, o que pode ter consequências desastrosas se o programa estiver sendo executado com privilégios elevados.

Outros exemplos incluem a injeção de comandos para enviar arquivos confidenciais por e-mail ou para executar outras ações maliciosas.

**6.6 Ataques de escalada de privilégio (Privilege Escalation Attacks):**

O objetivo de um ataque de escalada de privilégio é permitir que um atacante obtenha acesso a recursos ou execute ações que normalmente estão restritas a usuários com maiores privilégios (como o superusuário root ou um administrador).

Um exemplo clássico envolveu uma vulnerabilidade no daemon cron do Unix. O cron é um serviço que permite agendar a execução de tarefas em horários específicos e normalmente roda com privilégios de root para poder executar tarefas em nome de qualquer usuário. A vulnerabilidade permitiu que um atacante manipulasse o sistema para que o cron executasse um programa controlado pelo atacante com privilégios de root. Isso foi conseguido fazendo com que o programa do atacante definisse seu diretório de trabalho como o diretório do cron e, em seguida, causasse um despejo de memória nesse diretório. A imagem da memória do programa do atacante foi cuidadosamente construída para conter comandos válidos para o cron, incluindo comandos para alterar as permissões de um programa escolhido pelo atacante para root e, em seguida, executar esse programa.

Embora essa falha específica tenha sido corrigida, ela ilustra como erros em programas que rodam com altas permissões podem ser explorados para conceder privilégios elevados a usuários não autorizados.

# 7.Malwares

**7.1Surgimento dos Malwares**

O primeiro registro de um programa com características de malware remonta a 1971, com o "Creeper". Desenvolvido por Bob Thomas na BBN Technologies, o Creeper infectava sistemas TENEX, exibindo a mensagem: "I'm the creeper, catch me if you can!". Embora não causasse danos, demonstrou a possibilidade de programas autorreplicantes.

Em 1982, Rich Skrenta, então com 15 anos, criou o "Elk Cloner", um vírus que infectava computadores Apple II por meio de disquetes. Após a 50ª inicialização do sistema infectado, o Elk Cloner exibia um poema na tela, indicando a infecção. Esses primeiros exemplos eram inofensivos, mas evidenciaram o potencial de programas autorreplicantes para disseminação não autorizada.

**7.2 Evolução e Contexto Histórico**

Nos anos 1980 e 1990, com a popularização dos computadores pessoais e o aumento da conectividade, os malwares tornaram-se mais prevalentes e destrutivos. Vírus como o "Brain" (1986), considerado o primeiro vírus de PC, e o "Michelangelo" (1991), que ameaçava apagar dados em uma data específica, exemplificam essa tendência.​

A virada do milênio marcou a transição dos malwares de simples experimentos para ferramentas de cibercrime organizado. O worm "ILOVEYOU" (2000) propagou-se via e-mail com o assunto "I LOVE YOU", infectando milhões de computadores e causando danos estimados entre US$ 5,5 e 8,7 bilhões. Esse incidente evidenciou a vulnerabilidade dos sistemas interconectados e a necessidade de medidas de segurança mais robustas. ​

A década de 2010 testemunhou a ascensão do ransomware, com destaque para o "WannaCry" (2017), que afetou sistemas em mais de 150 países, incluindo o Brasil, causando interrupções significativas em serviços públicos e privados. Esse ataque ressaltou a importância de manter sistemas atualizados e reforçou a necessidade de estratégias eficazes de resposta a incidentes.

Atualmente, os malwares continuam a evoluir em complexidade, explorando novas tecnologias e vulnerabilidades. A integração da inteligência artificial e o aumento de dispositivos conectados ampliam a superfície de ataque, exigindo abordagens inovadoras e colaborativas para a cibersegurança.​

**7.3 Worms**

Worms são programas autônomos que se replicam e se espalham por redes de computadores sem a necessidade de intervenção humana. Eles exploram vulnerabilidades em sistemas para se propagar e podem causar danos significativos.​

**Exemplo: Morris Worm (1988)**

Em 1988, o Morris Worm destacou-se por sua rápida disseminação, infectando aproximadamente 6.000 computadores em apenas 24 horas. Considerando que a internet da época contava com cerca de 60.000 máquinas conectadas, o impacto foi substancial. As estimativas de prejuízo variaram de US$ 100.000 a milhões de dólares, evidenciando a vulnerabilidade dos sistemas interconectados. ​

**Exemplo: Conficker (2008)**

O Conficker, detectado em 2008, é considerado um dos worms de computador que mais se espalhou pela internet desde o SQL Slammer, em 2003. A rápida disseminação inicial do worm foi atribuída ao grande número de computadores que utilizavam o sistema operacional Microsoft Windows sem as atualizações de segurança adequadas. Em janeiro de 2009, estimava-se que entre 9 e 15 milhões de computadores estavam infectados. O Conficker afetou diversas organizações, incluindo a rede de computadores da Marinha da França e o Ministério da Defesa do Reino Unido.

**7.4 Ransomware**

Ransomware é um tipo de malware que criptografa os dados do usuário e exige um pagamento (resgate) para restaurar o acesso. Esse tipo de ataque pode paralisar operações de empresas e causar perdas financeiras significativas.​

**Exemplo: WannaCry (2017)**

Em 2017, o WannaCry se destacou como um dos ataques de ransomware mais devastadores, afetando cerca de 200.000 computadores em 150 países. O malware explorou uma vulnerabilidade no sistema Windows para se espalhar rapidamente, causando interrupções em diversos setores, incluindo saúde, finanças e serviços públicos. As perdas financeiras foram estimadas em até US$ 4 bilhões. ​

**Exemplo: Petya/NotPetya (2017)**

O NotPetya, identificado em junho de 2017, causou danos estimados em mais de 10 bilhões de dólares. O ataque afetou sistemas de monitoramento de radiação na Central Nuclear de Chernobyl, além de ministérios, bancos e sistemas de metrô na Ucrânia. Empresas multinacionais como a Maersk Line e a Merck & Co. também foram impactadas.

**7.5 Cavalos de Troia (Trojans)**

Cavalos de Troia são programas maliciosos disfarçados de software legítimo. Uma vez instalados, podem conceder acesso remoto ao sistema infectado, permitindo atividades como roubo de dados ou instalação de outros malwares.​

**Exemplo: Shlayer (2018)**

O Shlayer, detectado em 2018, destacou-se como uma ameaça significativa para usuários de macOS. Disfarçado como atualizações ou downloads de software legítimos, o Shlayer infectou aproximadamente 10% dos Macs analisados, expondo os usuários a adware e outros riscos de segurança.

**7.6 Botnets**

Botnets são redes de computadores infectados (bots) controlados remotamente por hackers. Essas redes podem ser utilizadas para lançar ataques distribuídos de negação de serviço (DDoS), enviar spam ou propagar outros tipos de malware.​

**Exemplo: Mirai (2016)**

Em 2016, a botnet Mirai foi responsável por um ataque DDoS massivo que afetou grande parte da costa leste dos EUA. O malware explorava vulnerabilidades em dispositivos da Internet das Coisas (IoT), recrutando-os para formar uma botnet capaz de gerar tráfego malicioso em larga escala.

**7.7 Programas Potencialmente Indesejáveis (PUPs)**

PUPs são softwares que podem incluir adware, barras de ferramentas e pop-ups não relacionados ao software originalmente baixado. Embora nem sempre sejam considerados malwares, podem degradar a experiência do usuário e introduzir vulnerabilidades.​

**Exemplo: Mindspark**

O Mindspark é um exemplo de PUP que se instala nos sistemas dos usuários sem consentimento explícito. Ele pode alterar configurações e desencadear comportamentos indesejados no dispositivo, sendo notoriamente difícil de remover.

**7.8 Worms Especializados**

Alguns worms são projetados para alvos específicos, visando infraestruturas críticas ou sistemas industriais.​

**Exemplo: Stuxnet (2010)**

O Stuxnet, descoberto em 2010, foi um worm projetado para atacar sistemas de controle industrial SCADA da Siemens, utilizados em centrífugas de enriquecimento de urânio no Irã. O malware reprogramava controladores lógicos programáveis (CLPs) e ocultava as alterações, danificando equipamentos sem que os operadores percebessem.

**7.9 Como os Malwares se Propagam**

A propagação de malwares ocorre por diversos vetores, conforme discutido por Tanenbaum:

Downloads e Anexos Maliciosos: Arquivos baixados da internet ou recebidos por e-mail que contêm código malicioso. Ao serem executados, infectam o sistema.

Engenharia Social e Phishing: Técnicas que enganam os usuários para que revelem informações confidenciais ou instalem malwares, geralmente através de e-mails ou sites falsos que imitam entidades legítimas.

Exploração de Vulnerabilidades: Ataques que tiram proveito de falhas de segurança em softwares ou sistemas operacionais para executar código malicioso.

Softwares Não Atualizados: Programas desatualizados podem conter vulnerabilidades conhecidas que são explorada por malwares para infectar o sistema.

# 8. Explorando Criptografia Básica na Defesa da Segurança:

(Seção sobre Defesa no Capítulo 12 de "Sistemas Operacionais Modernos", 3ª Edição)

A seção sobre defesa no livro *"Sistemas Operacionais Modernos"*, de Tanenbaum e Bos, aborda as diversas técnicas de segurança utilizadas para proteger sistemas operacionais contra ataques, sendo a **criptografia** um dos principais métodos de defesa. A criptografia tem um papel crucial na proteção de dados e comunicações, garantindo a confidencialidade e a integridade das informações. Neste contexto, vamos explorar como a criptografia é usada para fortalecer a segurança dos sistemas operacionais.

***8.1 Criptografia e Seus Tipos na Defesa de Sistemas:***

A criptografia é uma ferramenta essencial para a defesa da segurança em sistemas operacionais. Ela visa proteger dados contra acessos não autorizados e garantir que informações transmitidas ou armazenadas sejam mantidas seguras. O livro descreve dois tipos principais de criptografia, ambos usados para fins de defesa:

* **Criptografia Simétrica**: Nesse tipo de criptografia, a mesma chave é utilizada para criptografar e descriptografar os dados. Embora seja eficiente, a principal desvantagem é o gerenciamento da chave compartilhada. Mesmo assim, algoritmos como o **AES (Advanced Encryption Standard)** são amplamente empregados para proteger dados sensíveis em repouso ou em trânsito.
* **Criptografia Assimétrica**: Utiliza um par de chaves – uma pública e uma privada. A chave pública é usada para criptografar os dados, enquanto a chave privada é necessária para descriptografá-los. Este tipo de criptografia é fundamental para a defesa em autenticação e comunicação segura, com o **RSA** sendo um dos algoritmos mais comuns.

Ambos os tipos de criptografia desempenham papéis importantes na defesa de sistemas operacionais contra acessos não autorizados, espionagem e manipulação de dados.

***8.2 Criptografia na Defesa de Dados e Comunicações:***

Os sistemas operacionais modernos aplicam criptografia em diversas camadas para garantir que dados críticos e comunicações sejam protegidos. O livro detalha como essas aplicações de criptografia contribuem para a segurança geral do sistema operacional:

* **Proteção de Arquivos e Dispositivos**: Para defender dados armazenados, sistemas operacionais utilizam criptografia de discos e arquivos. Ferramentas como **BitLocker** (Windows) e **FileVault** (macOS) criptografam os dados do dispositivo, tornando-os inacessíveis mesmo em caso de roubo ou perda do dispositivo.
* **Segurança em Comunicações de Rede**: A criptografia também é fundamental para proteger dados transmitidos pela rede. Protocolos como **SSL/TLS** são usados para criptografar as conexões entre servidores e clientes, evitando que dados sensíveis, como senhas e informações bancárias, sejam interceptados durante a transmissão.
* **Autenticação e Defesa**: A autenticação de usuários é uma das áreas em que a criptografia assimétrica se destaca. O uso de **certificados digitais** e **assinaturas digitais** assegura que somente usuários autenticados possam acessar determinados recursos do sistema operacional, defendendo o sistema contra acessos não autorizados.

***8.4 Desafios e Defesas em Criptografia:***

Embora a criptografia seja um pilar essencial da defesa, ela também pode ser vulnerável a ataques. O livro menciona algumas das principais ameaças à criptografia e as defesas que podem ser usadas para mitigar esses riscos:

* **Ataques de Força Bruta**: Nos ataques de força bruta, os atacantes tentam todas as combinações possíveis de chaves até encontrar a correta. Para se proteger contra esses ataques, os sistemas devem utilizar algoritmos como o **AES**, que oferece chaves de tamanhos maiores, tornando a quebra mais difícil.
* **Análise Criptográfica**: Alguns ataques tentam explorar falhas nos algoritmos de criptografia, buscando padrões ou fragilidades para descriptografar dados. Para proteger os sistemas contra esses ataques, é essencial usar algoritmos robustos e constantemente atualizar os métodos criptográficos.
* **Gerenciamento de Chaves**: Um dos maiores desafios para a defesa da criptografia é o gerenciamento de chaves. Se uma chave privada for comprometida, toda a segurança do sistema pode ser comprometida. Para mitigar esse risco, os sistemas operacionais modernos adotam práticas rigorosas de **armazenamento seguro de chaves**, muitas vezes usando **módulos de segurança de hardware (HSM)**.

***8.5 Mecanismos de Defesa para Aumentar a Segurança:***

Além da criptografia, existem mecanismos adicionais que reforçam a defesa do sistema operacional e tornam a exploração de vulnerabilidades mais difícil:

* **Prevenção de Execução de Dados (DEP)**: Este mecanismo impede que código malicioso seja executado a partir de áreas de memória que não deveriam conter código, como buffers de dados. Isso dificulta a exploração de vulnerabilidades, como buffer overflows, que podem ser usadas por atacantes para executar código malicioso.
* **Randomização do Layout do Espaço de Endereçamento (ASLR)**: O ASLR dificulta para os atacantes a previsão de onde dados críticos ou código estão localizados na memória, tornando mais difícil explorar vulnerabilidades de buffer overflow e outras falhas de segurança.

Esses mecanismos de defesa complementam a criptografia e ajudam a fortalecer ainda mais a segurança dos sistemas operacionais.

***8.6 A Criptografia como Defensor da Segurança:***

A criptografia é um dos pilares fundamentais da defesa contra ataques em sistemas operacionais modernos. Ela protege dados e comunicações de interceptação e acesso não autorizado, garantindo a integridade e a confidencialidade das informações. O livro *"Sistemas Operacionais Modernos"* destaca como a criptografia é implementada para proteger arquivos, autenticar usuários e garantir a segurança das redes.

Com o aumento das ameaças cibernéticas, a criptografia continua a ser uma ferramenta crítica na defesa dos sistemas operacionais. O conhecimento de sua aplicação correta e das defesas associadas a ela é crucial para garantir a segurança dos sistemas modernos e a proteção dos dados sensíveis.

# Conclusão

O estudo abordado neste trabalho demonstra que a segurança em sistemas operacionais é um campo dinâmico e essencial para a proteção de dados e recursos computacionais. A análise das principais ameaças — como *malwares*, ataques de negação de serviço e exploração de vulnerabilidades — revela a complexidade dos riscos enfrentados por usuários e organizações. Além disso, a categorização dos invasores (desde *hackers* mal-intencionados até agentes patrocinados por governos) destaca a diversidade de motivações e métodos utilizados em ataques cibernéticos.

A autenticação multifator e a criptografia emergem como pilares fundamentais para a defesa, complementados por mecanismos como DEP (Prevenção de Execução de Dados) e ASLR (Randomização do Layout do Espaço de Endereçamento). No entanto, os ataques internos — como *buffer overflows* e cavalos de Troia — lembram que a segurança não depende apenas de tecnologias, mas também de boas práticas, como o princípio do menor privilégio e a auditoria constante de logs.

Em um cenário onde as ameaças evoluem rapidamente, a segurança deve ser entendida como um processo contínuo, que exige atualizações frequentes, educação dos usuários e adoção de políticas rigorosas. Sistemas operacionais modernos, quando bem configurados e monitorados, podem oferecer proteção robusta, mas a conscientização humana permanece como o elo crítico nessa cadeia.

Palavras-chave: Segurança cibernética; Sistemas operacionais; Autenticação; Criptografia; Mitigação de riscos.

# Referências

**Livro Principal (Base do Trabalho)**

TANENBAUM, Andrew S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson, 2016.

**Artigos Científicos**

SCHNEIER, Bruce. **Applied Cryptography**: Protocols, Algorithms, and Source Code in C. 2. ed. Wiley, 2015.

STALLINGS, William. **Cryptography and Network Security**: Principles and Practice. 7. ed. Pearson, 2017.

**Fontes Online (Atualizadas)**

CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil). **Cartilha de Segurança para Internet**. 2022. Disponível em: <https://cartilha.cert.br/>. Acesso em: 10 out. 2023.

MITRE ATT&CK®. **Tactics, Techniques, and Procedures (TTPs) for Cyber Threats**. 2023. Disponível em: <https://attack.mitre.org/>. Acesso em: 10 out. 2023.

**Relatórios Técnicos**

KASPERSKY LAB. **Relatório de Ameaças Cibernéticas 2023**. 2023. Disponível em: [https://www.kaspersky.com.br](https://www.kaspersky.com.br/). Acesso em: 10 out. 2023.

**Legislação**

BRASIL. **Lei Geral de Proteção de Dados (LGPD) – Lei nº 13.709/2018**. Diário Oficial da União, Brasília, 2018.