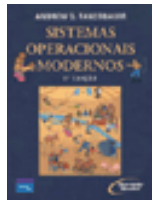
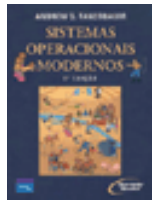


Capítulo 9

Segurança

9.1 O ambiente de segurança
9.2 Criptografia básica
9.3 Autenticação de usuário
9.4 Ataques de dentro do sistema
9.5 Ataques de fora do sistema
9.6 Mecanismos de proteção
9.7 Sistemas confiáveis

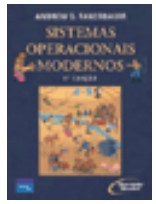
O Ambiente de Segurança
Ameaças



Objetivo	Ameaça
Confidencialidade dos dados	Exposição dos dados
Integridade dos dados	Adulteração dos dados
Disponibilidade do sistema	Recusa de serviço

Objetivos e ameaças à segurança

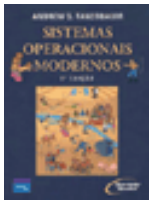
Invasores



Categorias comuns

2. Curiosidades casuais de usuários leigos
3. Espionagem por pessoas internas
4. Tentativas determinadas para ganhar dinheiro
5. Espionagem militar ou comercial

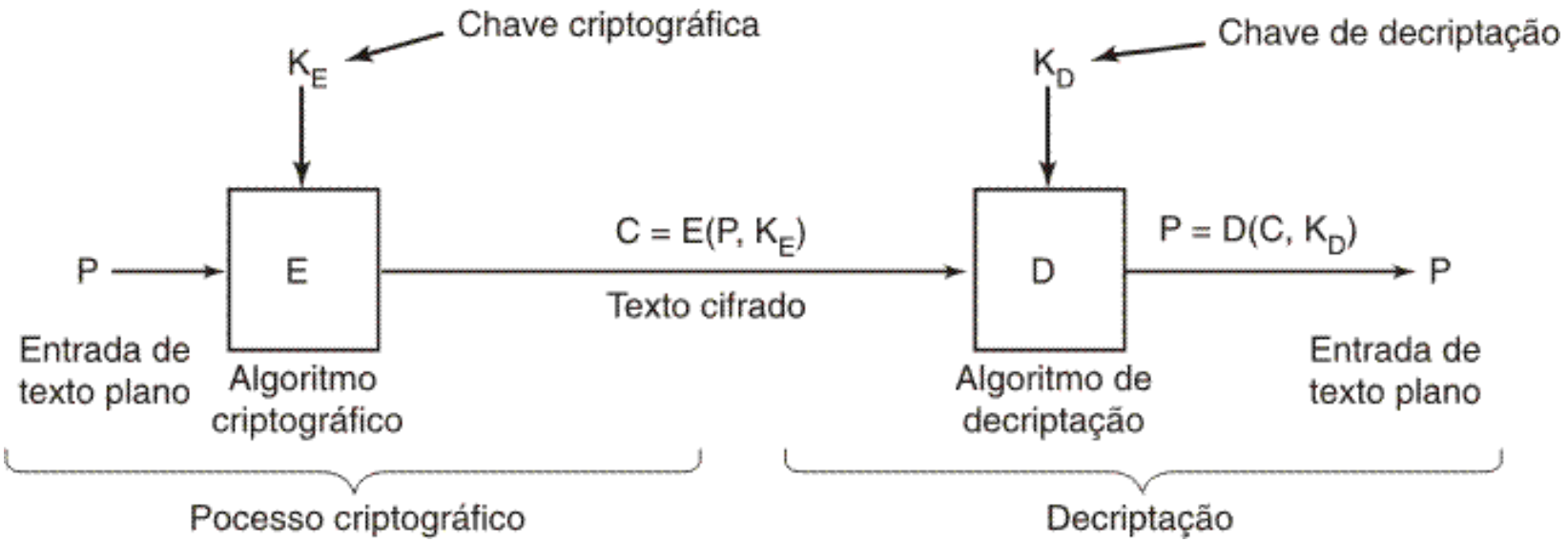
Perda Acidental de Dados



Causas comuns

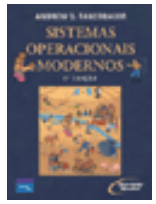
2. Atos de Deus
   - incêndios, enchentes, guerras
3. Erros de hardware ou software
   - defeitos na CPU, discos ruins, erros de programas
4. Erros humanos
   - entrada incorreta de dados, montagem errada da fita

Criptografia Básica



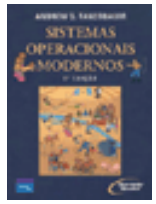
Relação entre o texto plano e o texto cifrado

Criptografia por Chave Secreta



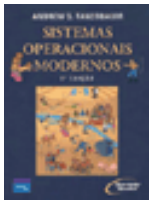
- Substituição monoalfabética
  - cada letra é substituída por letra diferente
- Dada a chave criptográfica,
  - fácil achar a chave de decriptação
- Criptografia de chave secreta ou criptografia de chave simétrica

Criptografia por Chave Pública



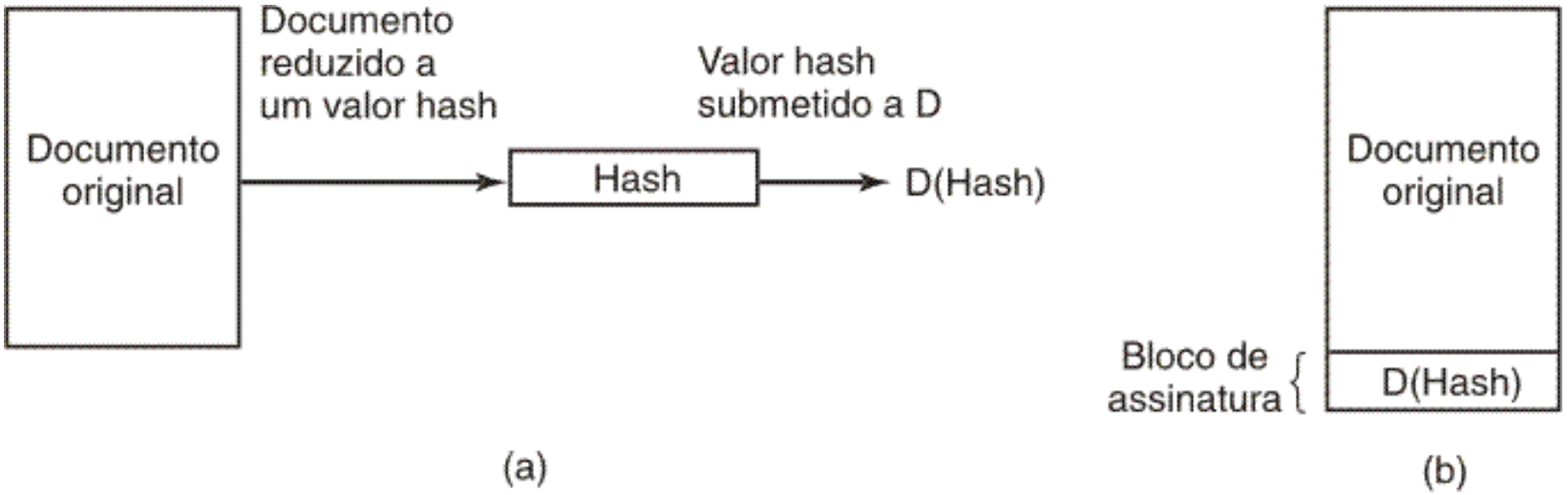
- Todos os usuários possuem um par de chaves pública/privada
  - publica a chave pública
  - chave privada não publicada
- Chave pública é a chave criptográfica
  - Chave privada é a chave de decriptação

Funções de Uma Via



- Função tal que dados f e seu parâmetro x
  - fácil calcular y = f(x)
- Mas dado y
  - computacionalmente inviável encontrar x

Assinaturas Digitais



- Calculando um bloco de assinatura
- O que o receptor recebe

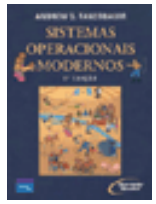
Autenticação de Usuário

Princípios Básicos. A autenticação deve identificar:

2. Alguma coisa que o usuário sabe
3. Alguma coisa que o usuário tem
4. Alguma coisa que o usuário é

Isto é feito antes do usuário poder usar o sistema

Autenticação Usando Senhas (1)



LOGIN: mauro
SENHA: qualquer
LOGIN COM SUCESSO

(a)

LOGIN: carolina
NOME INVÁLIDO
LOGIN:

(b)

LOGIN: carolina
SENHA: umdois
LOGIN INVÁLIDO
LOGIN:

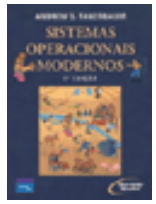
(c)

(a) Um acesso bem sucedido

(b) Acesso rejeitado depois da entrada de nome

(c) Acesso rejeitado depois da entrada de nome e senha

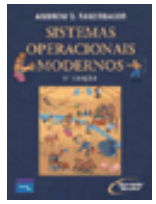
Autenticação Usando Senhas (2)



```
LBL> telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD: root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL
```

- Como um cracker invadiu o computador do LBL
  - um lab de pesquisa do Dep. de Energia dos EUA

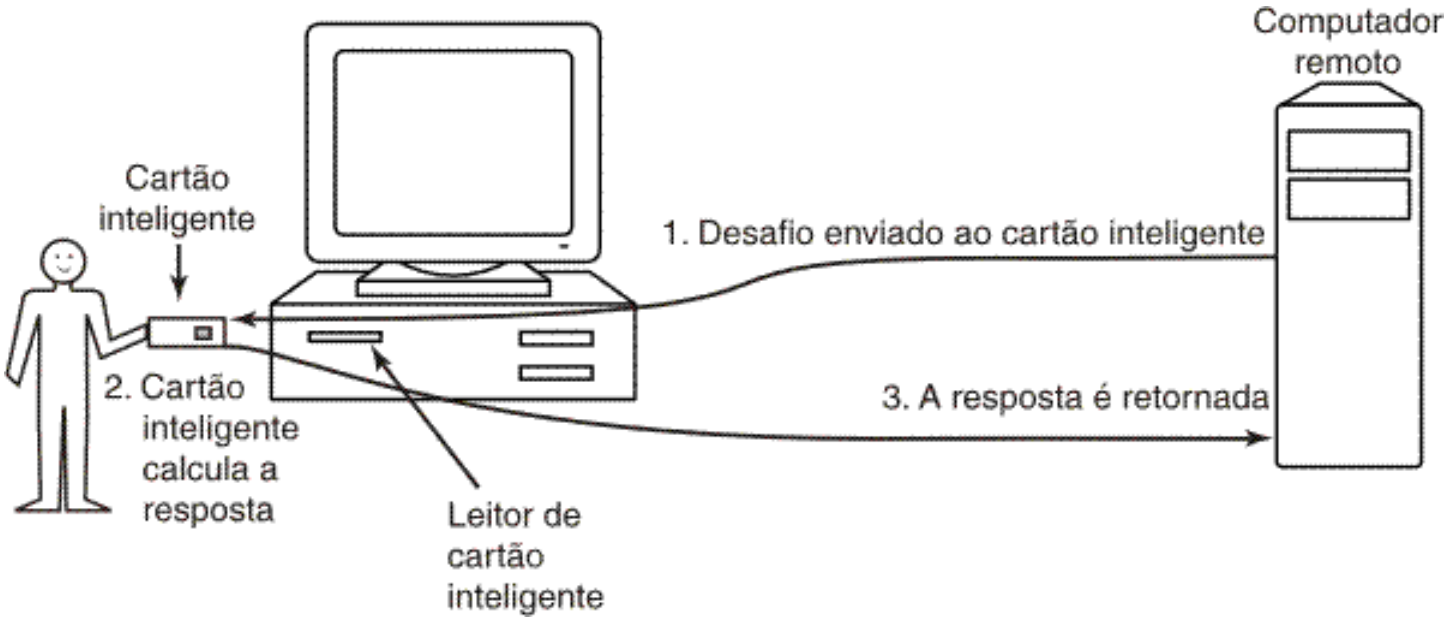
Autenticação Usando Senhas (3)



Barbara, 4238, e(Dog4238)
Tony, 2918, e(6%%TaeFF2918)
Laura, 6902, e(Shakespeare6902)
Mark, 1694, e(XaB@Bwcz1694)
Deborah, 1092, e(LordByron,1092)

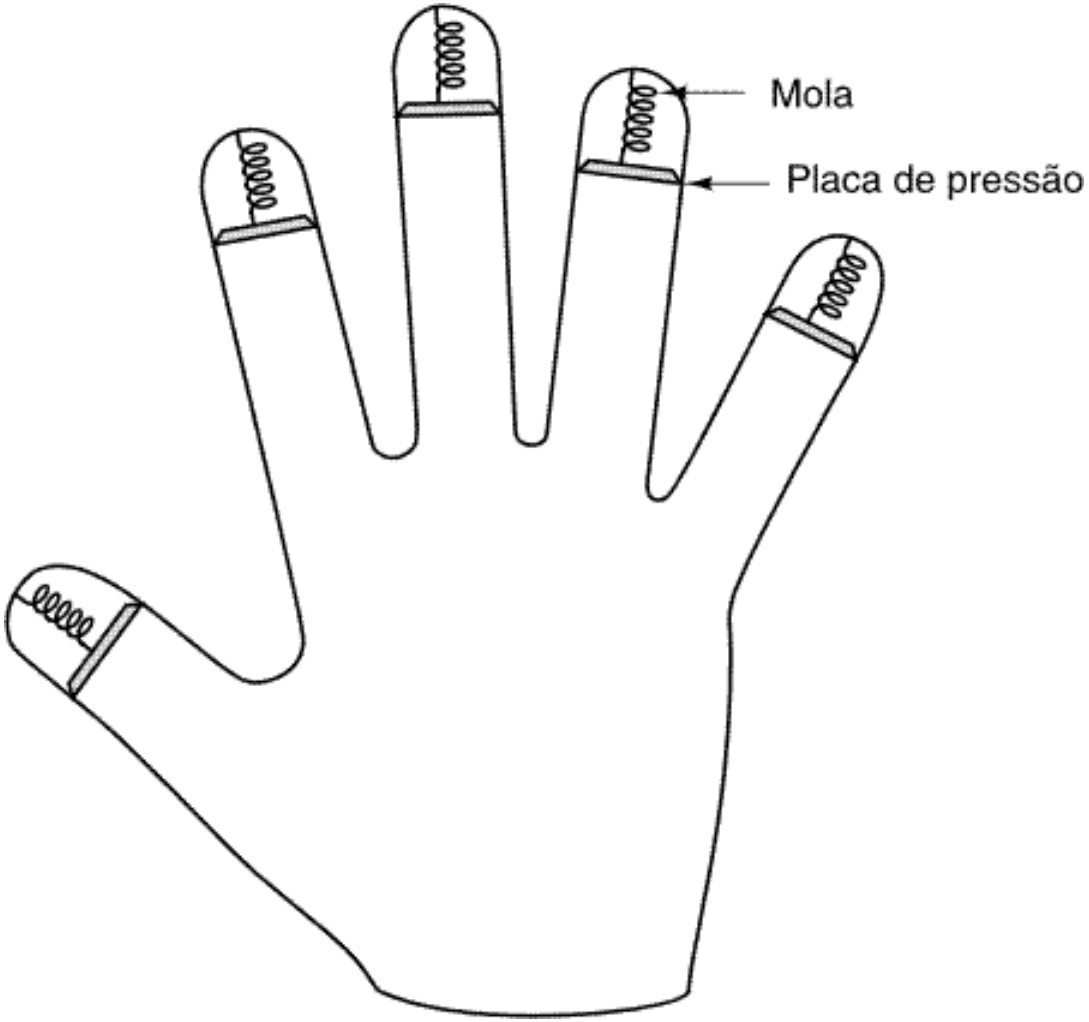
O uso do sal para atrapalhar a pré-computação de senhas criptografadas

Autenticação Usando um Objeto Físico



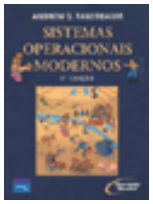
- Cartões de plástico
  - cartões de faixa magnética
  - cartões com processador: cartões com valores armazenados, cartões inteligentes

Autenticação Usando Biométrica



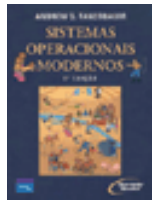
Um dispositivo para medir o comprimento do dedo

Medidas de Defesa



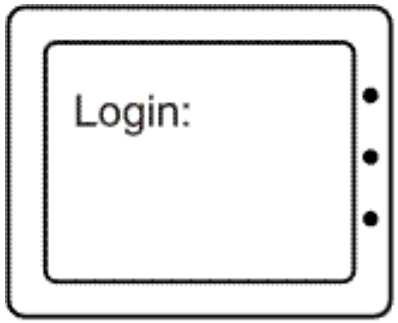
- Limitação do horário de acesso ao sistema
- Chamada automática de volta para um número pré-especificado
- Número limitado de tentativa de acessos
- Uma base de dados de todos os acessos ao sistema
- Nome e senha simples como isca
  - pessoal de segurança é notificado quando o intruso “morde a isca”

Segurança de Sistemas Operacionais
Cavalos de Tróia

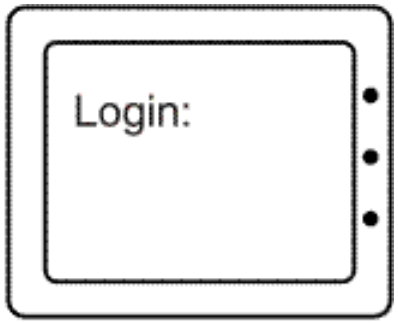


- Programa livre disponibilizado para usuários inocentes
  - contém na verdade código destrutivo
- Coloca versão adulterada de um programa utilitário no computador da vítima
  - leva o usuário a executar aquele programa

Conexão Impostora *(Spoofing)*



(a)

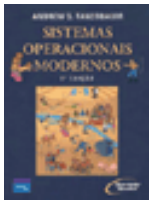


(b)

(a) Tela de conexão verdadeira

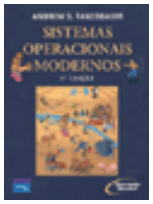
(b) Tela de conexão impostora

Bombas Lógicas



- Programador da empresa escreve programa
  - com potencial para causar danos (bomba lógica)
  - OK desde que ele/ela alimente o programa diariamente com uma senha
  - se programador é despedido, programa não é alimentado com senha, bomba explode

Alçapões



```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v) break;
}
execute_shell(name);
```

(a)

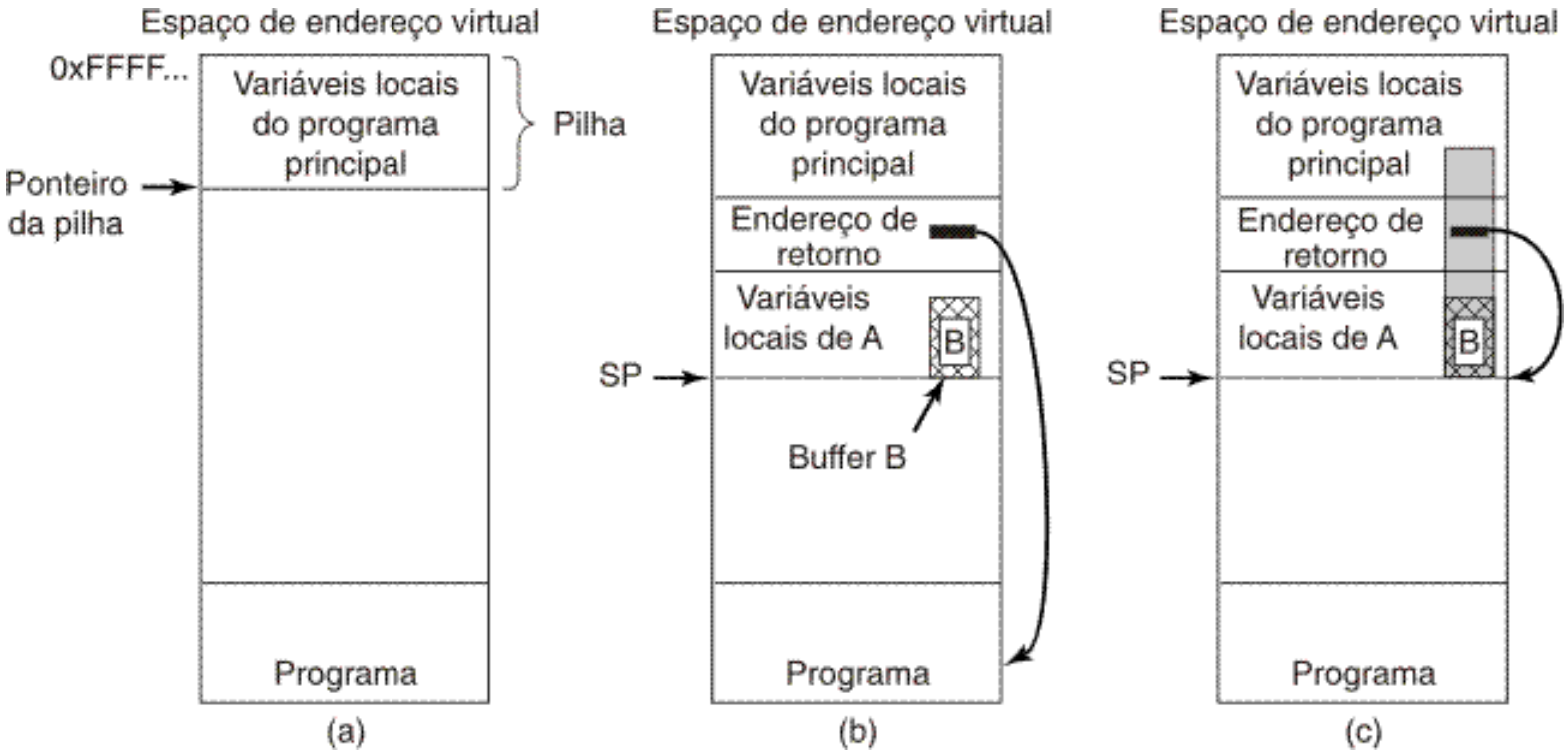
```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v || strcmp(name, "zzzzz") == 0) break;
}
execute_shell(name);
```

(b)

(a) Código normal

(b) Código com alçapão inserido

Transbordo de Buffer *(Overflow)*

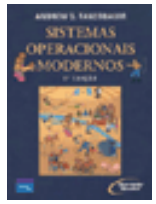


(a) Situação na qual programa principal está executando

(b) Depois que procedimento *A* foi chamado

(c) Transbordo de buffer mostrado em cinza

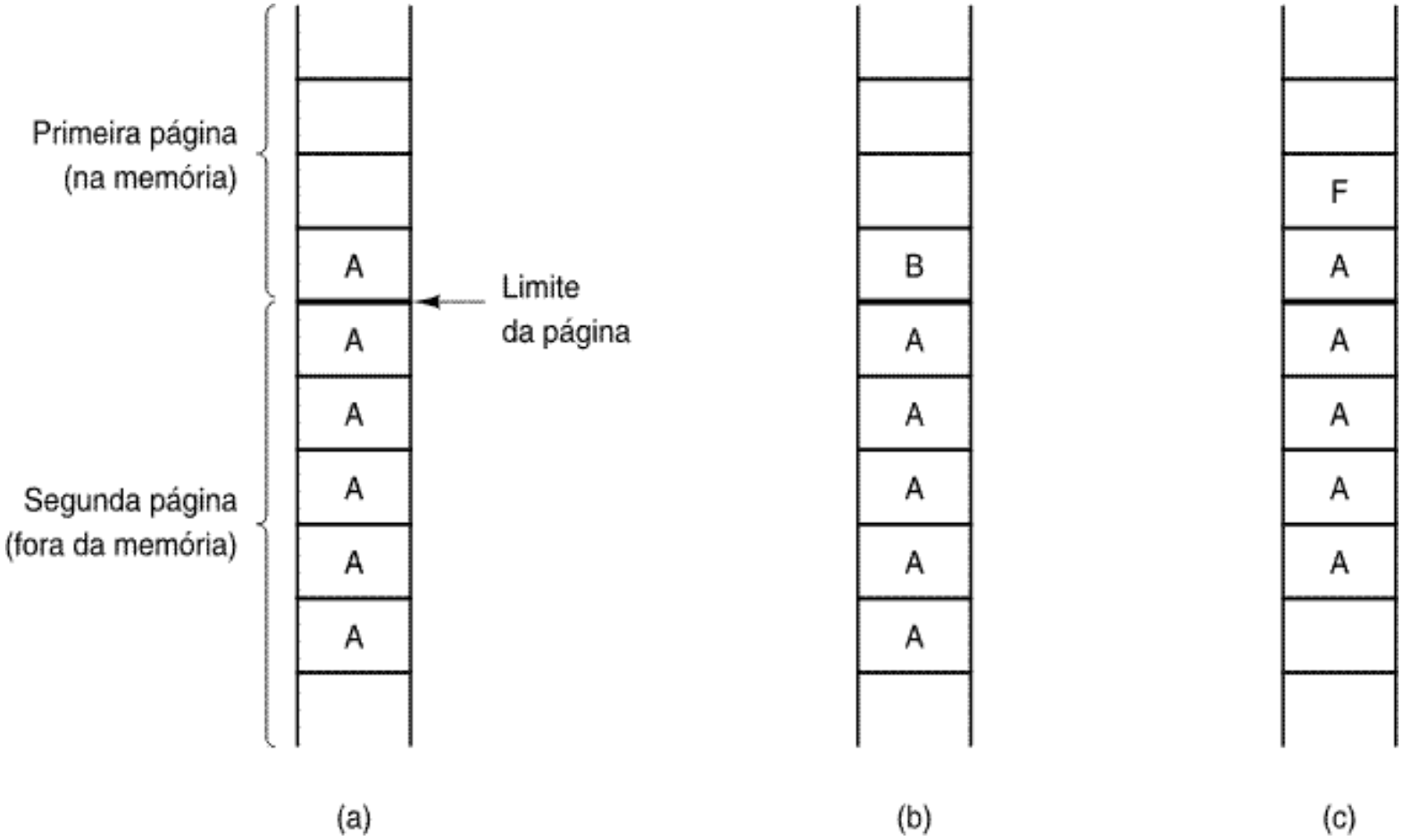
Ataques Genéricos à Segurança



Ataques típicos

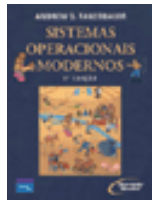
- Solicitar memória, espaço de disco, fitas e apenas ler
- Tentar chamadas ilegais ao sistema
- Iniciar o acesso ao sistema e pressionar as teclas DEL, RUBOUT, ou BREAK
- Tentar modificar estruturas complexas do SO
- Tentar todos os NÃO FAÇA especificados nos manuais
- Convencer um programador a introduzir um alçapão
- Implorar para a secretária do administrador do sistema para ajudar um pobre usuário que esqueceu a senha

Falhas Famosas de Segurança



TENEX – o problema da senha

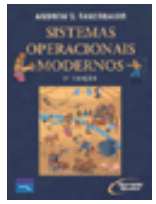
Princípios de Projeto de Segurança



1. O projeto do sistema deve ser público
2. Default deve ser "acesso negado"
3. Checar autoridade atual
4. Dar a cada processo o menor privilégio possível
5. Mecanismo de proteção deve ser
   - simples
   - uniforme
   - nas camadas mais inferiores do sistema
6. Esquema deve ser psicologicamente aceitável

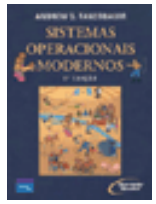
e … mantenha o projeto simples

Segurança de Rede



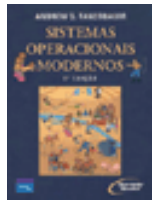
- Ameaça externa
  - código transmitido para máquina alvo
  - código executado lá, causando danos
- Objetivos do programador de vírus
  - espalhar rapidamente o vírus
  - difícil de detectar
  - difícil de se livrar
- Virus = programa capaz de se auto-reproduzir
  - anexa seu código a um outro programa
  - adicionalmente, causa danos

Cenários de Danos Causados por Vírus



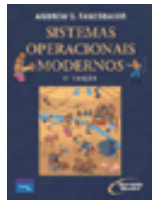
- Chantagem
- Recusa de serviço enquanto o vírus estiver executando
- Danificar o hardware permanentemente
- Lançar vírus no computador do concorrente
  - causar danos
  - espionagem
- Truques sujos intra-corporativo
  - sabotar arquivos de outros funcionários da corporação

Como Funcionam os Vírus (1)



- Vírus escritos em linguagem de montagem
- Inseridos em um outro programa
  - uso de ferramenta chamada conta-gotas *(dropper)*
  - vírus dormente até que programa executa
  - então infecta outros programas
  - eventualmente dispara sua carga explosiva

Como Funcionam os Vírus (2)

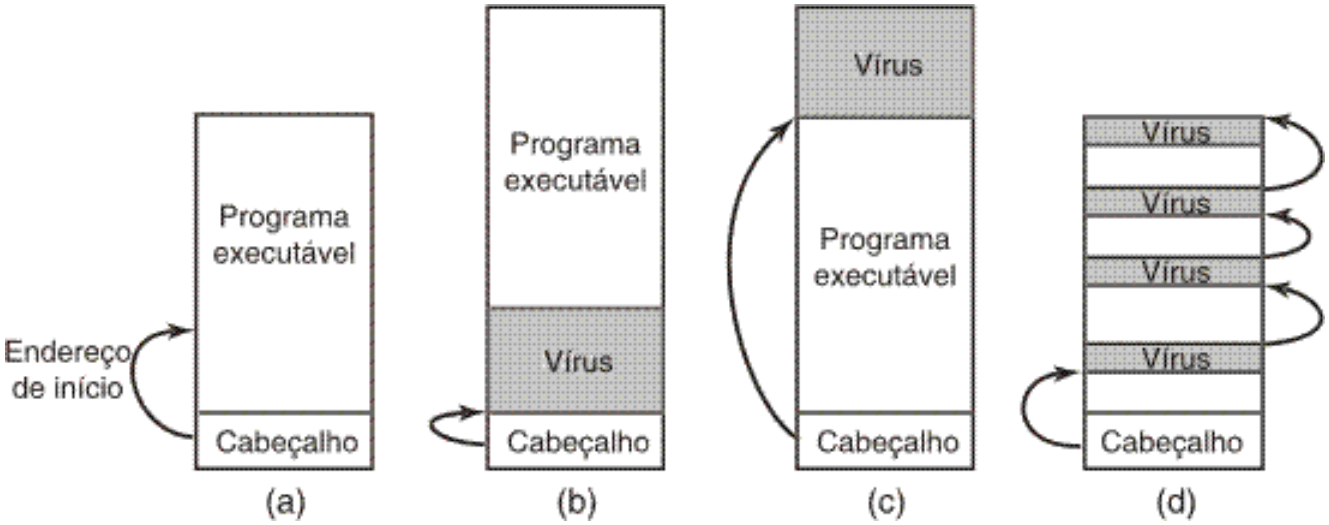


Procedimento recursivo que encontra arquivos executáveis em um sistema Unix

Vírus poderia infectá-los todos

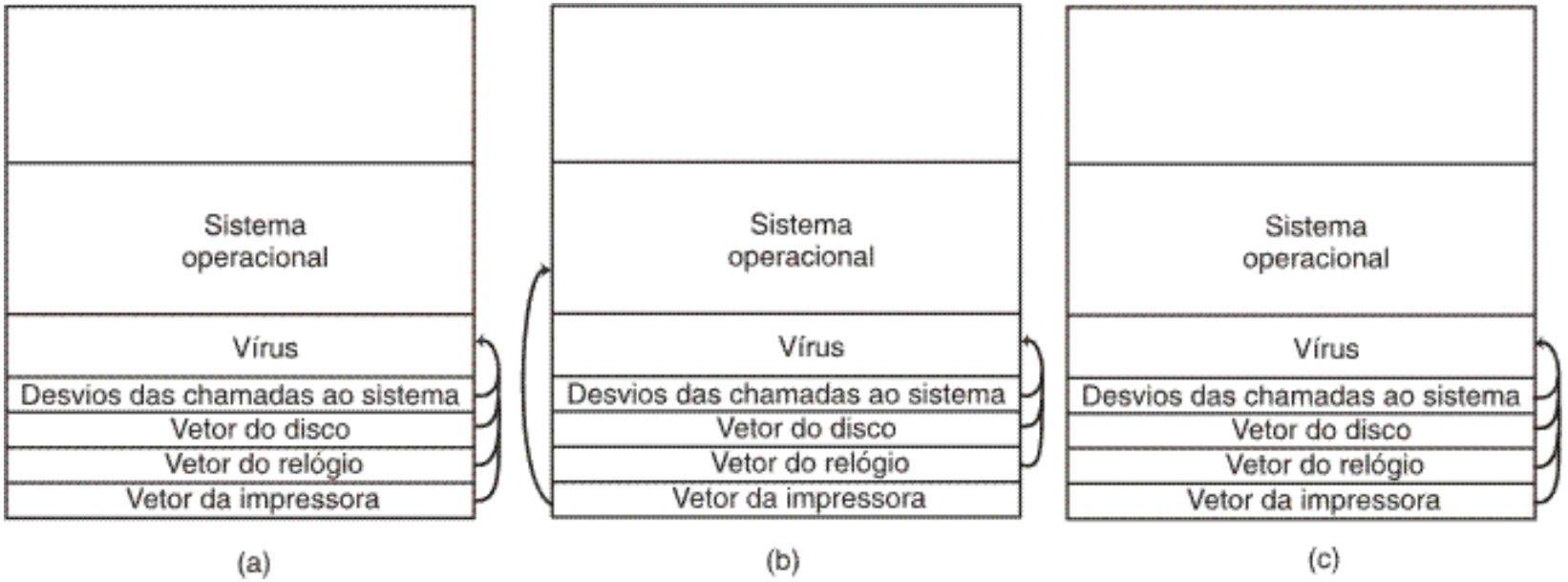
```
#include <sys/types.h>                          /* cabeçalhos-padrão POSIX */
#include <sys/stat.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
struct stat sbuf;                               /* para a chamada lstat veja se o arquivo é uma ligação simb. */

search(char *dir_name)
{                                               /* busca recursivamente por executáveis */
    DIR *dirp;                                  /* ponteiro para um fluxo aberto de diretório */
    struct dirent *dp;                          /* ponteiro para uma entrada de diretório */

    dirp = opendir(dir_name);                   /* abrir este diretório */
    if (dirp == NULL) return;                   /* se dir não puder ser aberto' esqueça-o */
    while (TRUE)  {
            dp = readdir(dirp);                 /* leia a próxima entrada de diretório */
            if (dp == NULL) {                   /* NULL significa que terminamos */
            chdir ("..");                       /* volte ao diretório-pai */
            break;                              /* sai do laço */
            }
            if (dp->d_name[0] == '.') continue; /* salte os diretórios . e .. */
            lstat(dp->d_name, &sbuf);           /* a entrada é uma ligação simbólica? */
            if (S_ISLNK(sbuf.st_mode)) continue;/* salte as ligações simbólicas */
            if(chdir(dp->d_name) == 0) {        /* se chdir tiver sucesso, deve ser um diretório */
                  search(".");                  /* sim, entre e busque-o */
            } else {                            /* não (arquivo), infecte-o */
                  if (access(dp->d_name, X_OK) == 0)    /* se for executável, infecte-o */
                        infect(dp->d_name);
            }
            closedir(dirp);                     /* diretório processado; feche e retorne */
    }
```

Como Funcionam os Vírus (3)



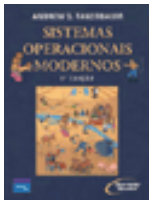
a) Um programa executável
b) Com um vírus à frente
c) Com um vírus no final
d) Com vírus espalhado pelos espaços livres dentro do programa

Como Funcionam os Vírus (4)



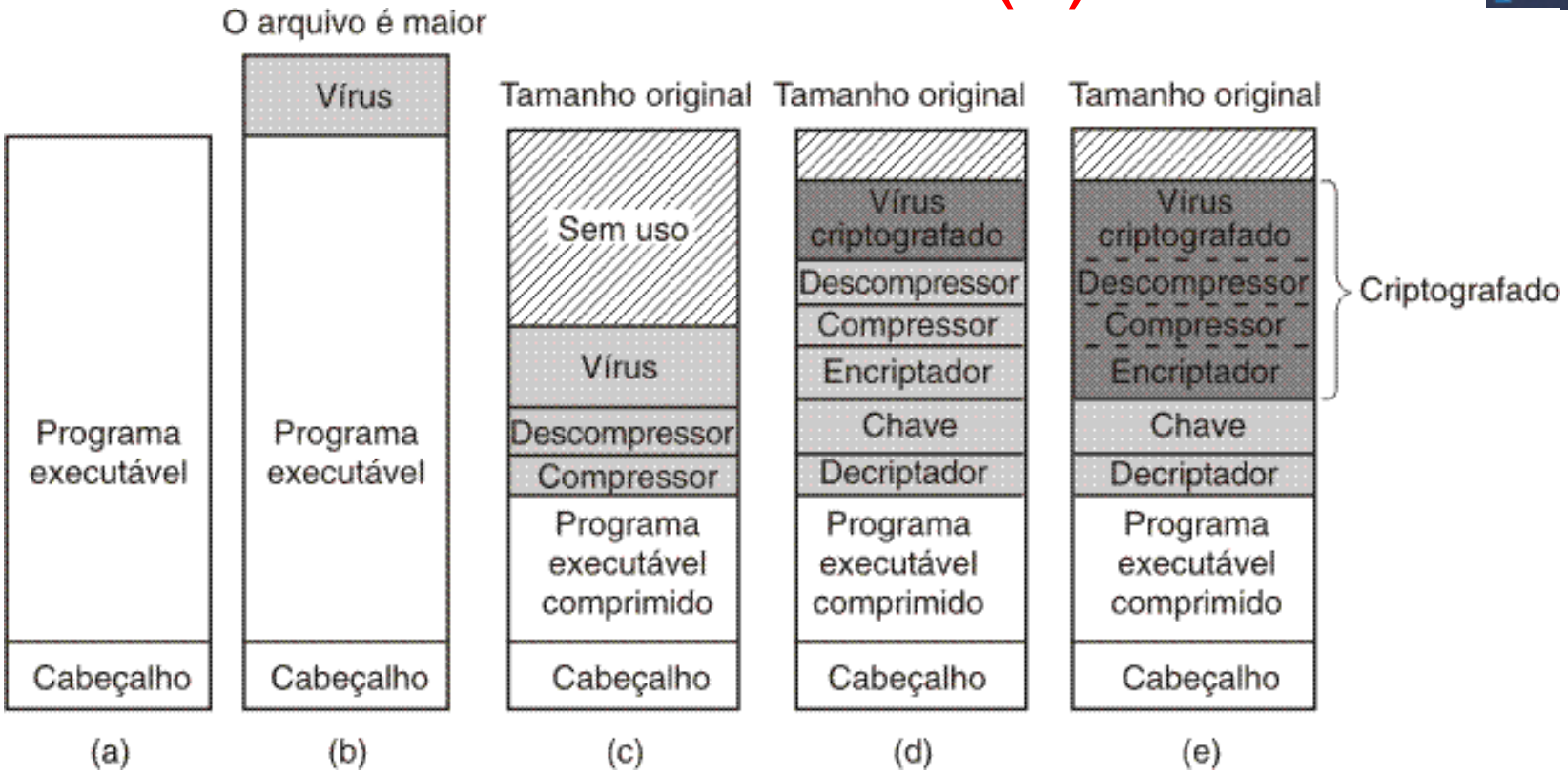
- Depois do vírus ter capturado os vetores de interrupção e de desvio de controle da CPU
- Depois do SO ter retomado o vetor de interrupção da impressora
- Depois do vírus ter percebido a perda do vetor de interrupção da impressora e tê-lo recuperado

Como os Vírus se Disseminam



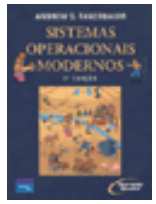
- Vírus colocados onde há chance de serem copiados
- Quando copiados
  - infectam programas no disco rígido, disquetes
  - podem tentar se disseminar na rede local
- Anexam-se à mensagens eletrônicas aparentemente inocentes
  - quando executados, usam listas de contatos para replicar

Técnicas Antivírus e Antiantivírus (1)



a) Um programa
b) Programa infectado
c) Programa infectado comprimido
d) Vírus criptografado
e) Vírus comprimido com o código de compressão criptografado

Técnicas Antivírus e Antiantivírus (2)



```
MOV A,R1
ADD B,R1
ADD C,R1
SUB #4,R1
MOV R1,X
```
(a)

```
MOV A,R1
NOP
ADD B,R1
NOP
ADD C,R1
NOP
SUB #4,R1
NOP
MOV R1,X
```
(b)

```
MOV A,R1
ADD #0,R1
ADD B,R1
OR R1,R1
ADD C,R1
SHL #0,R1
SUB #4,R1
JMP .+1
MOV R1,X
```
(c)

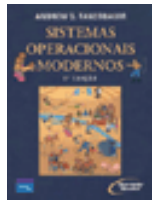
```
MOV A,R1
OR R1,R1
ADD B,R1
MOV R1,R5
ADD C,R1
SHL R1,0
SUB #4,R1
ADD R5,R5
MOV R1,X
MOV R5,Y
```
(d)

```
MOV A,R1
TST R1
ADD C,R1
MOV R1,R5
ADD B,R1
CMP R2,R5
SUB #4,R1
JMP .+1
MOV R1,X
MOV R5,Y
```
(e)

Exemplos de um vírus polimórfico

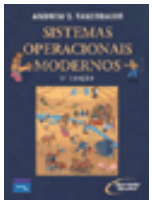
Todos esses exemplos fazem a mesma coisa

Técnicas Antivírus e Antiantivírus (3)



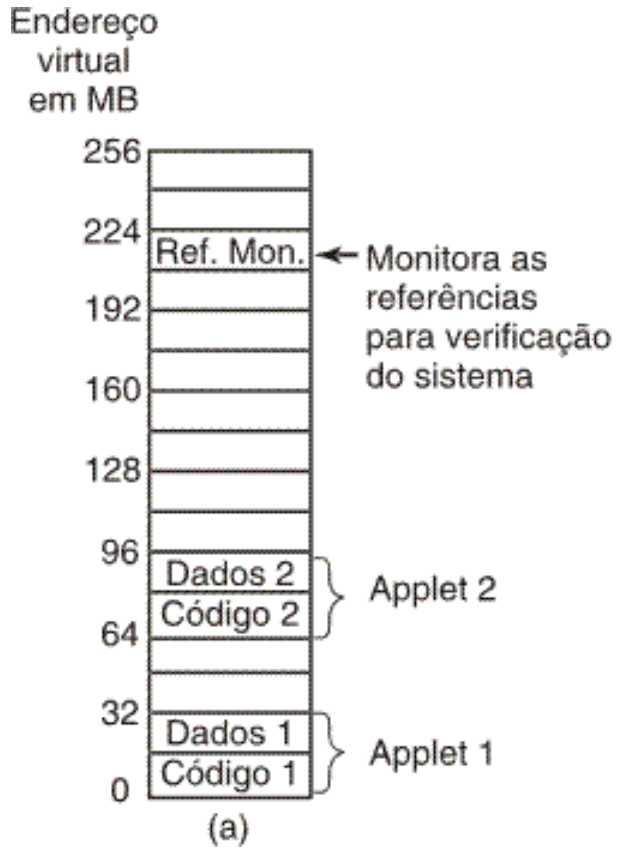
- Verificadores de integridade
- Verificadores de comportamento
- Prevenção contra vírus
  - um bom SO
  - instalar apenas softwares originais, de fabricante confiável
  - usar software antivírus
  - não clicar em anexos às mensagens eletrônicas
  - fazer cópias de segurança com frequência
- Recuperação de um ataque de vírus
  - parar o computador, reiniciar de disco seguro, executar antivírus

O Verme da Internet



- Consistia de dois programas
  - iniciador *(bootstrap)* para carregar o verme
  - o verme em si
- O verme primeiro esconde sua existência
- Em seguida se replica em novas máquinas

Código Móvel (1)
Caixa de Areia *(sandboxing)*



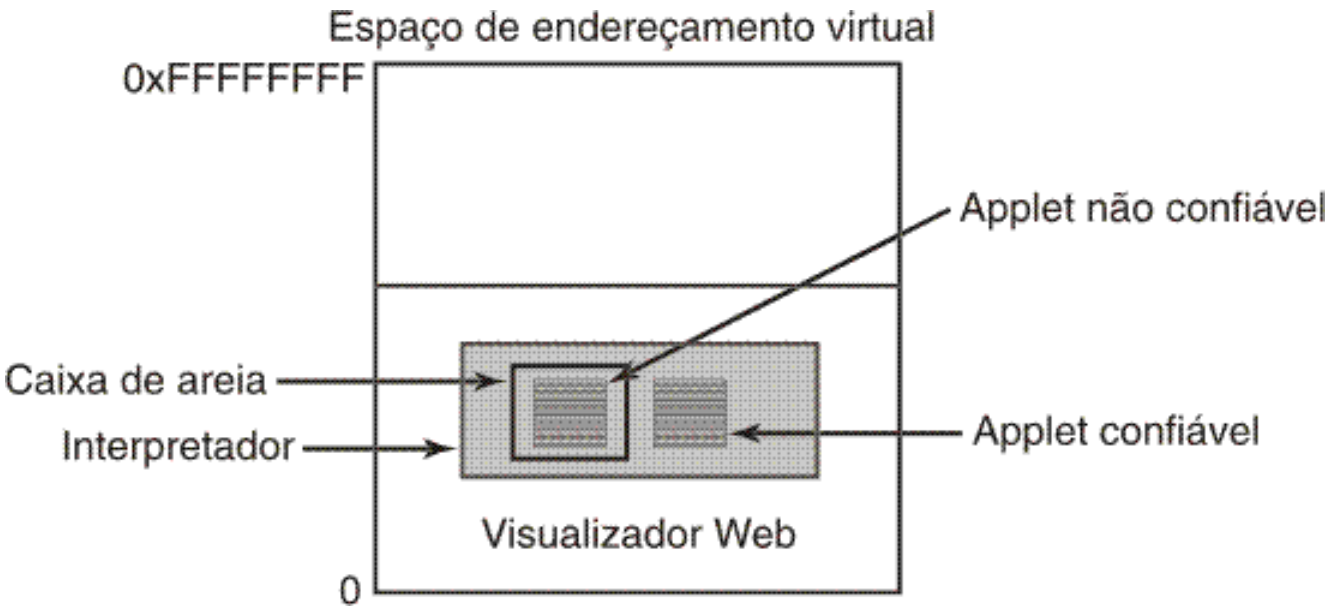

```
MOV R1, S1
SHR #24, S1
CMP S1, S2
TRAPNE
JMP (R1)
```

(b)

(a) Memória dividida em caixas de areia de 16MB

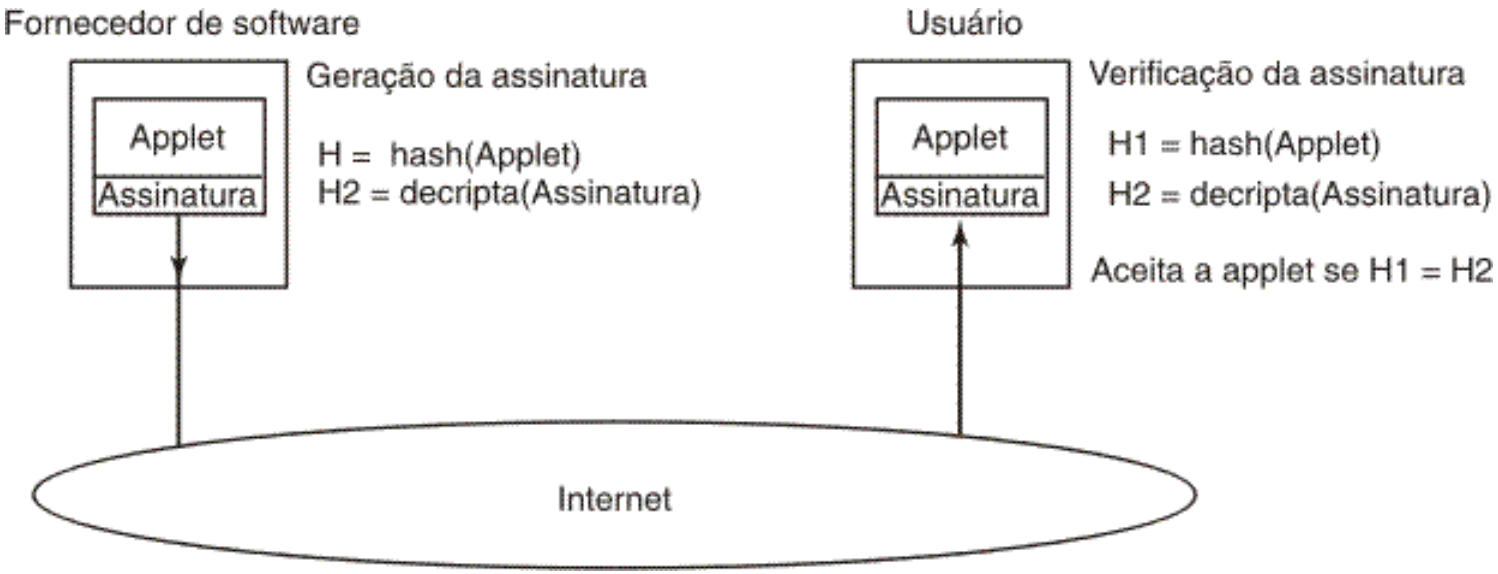
(b) Uma maneira de verificar a validade de uma instrução

Código Móvel (2)



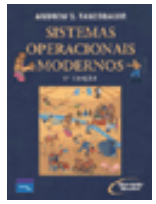
Applets podem ser interpretadas por um navegador Web

Código Móvel (3)



Como funciona a assinatura de código

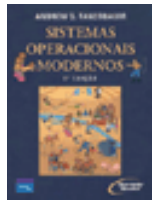
Segurança em Java (1)



- Uma linguagem tipificada e segura
  - compilador rejeita tentativas de mau uso de variável

- Verificação inclui …
  1. tentativa de forjar ponteiros
  2. violação de restrições de acesso sobre membros de classes privadas
  3. mau uso do tipo de uma variável
  4. geração de transbordo/falta na pilha
  5. conversão ilegal de variáveis para outro tipo

Segurança em Java (2)



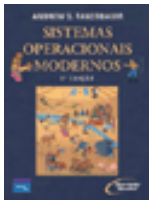
URL	Signer	Objeto	Ação
www.taxprep.com	TaxPrep	/usr/susan/1040.xls	Read
*		/usr/tmp/*	Read, Write
www.microsoft.com	Microsoft	/usr/susan/Office/–	Read, Write, Delete

Exemplos de proteção que pode ser especificada com o JDK 1.2

Mecanismos de Proteção
Domínios de Proteção (1)

Exemplos de três domínios de proteção

Domínios de Proteção (2)



	Objeto							
	Arquivo1	Arquivo2	Arquivo3	Arquivo4	Arquivo5	Arquivo6	Impressora1	Plotter2
Domínio 1	Leitura	Leitura Escrita						
2			Leitura	Leitura Escrita Execução	Leitura Escrita		Escrita	
3						Leitura Escrita Execução	Escrita	Escrita

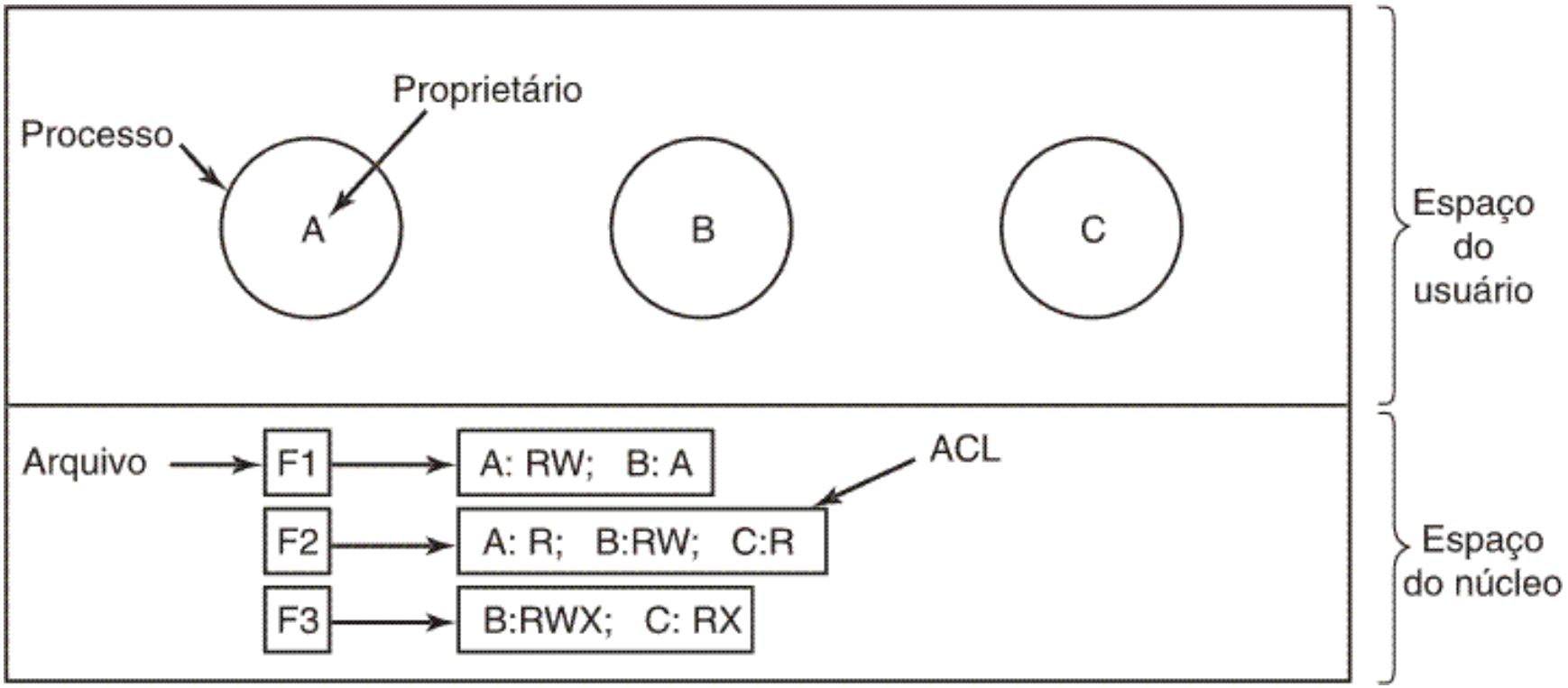
Uma matriz de proteção

Domínios de Proteção (3)

	Objeto										
	Arquivo1	Arquivo2	Arquivo3	Arquivo4	Arquivo5	Arquivo6	Impressora1	Plotter2	Domínio1	Domínio2	Domínio3
Domínio 1	Leitura	Leitura Escrita								Entra	
2			Leitura	Leitura Escrita Execução	Leitura Escrita		Escrita				
3						Leitura Escrita Execução	Escrita	Escrita			

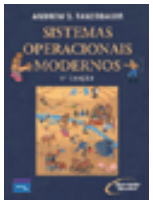
Uma matriz de proteção com domínios como objetos

Listas de Controle de Acesso (1)



Uso de listas de controle de acesso para gerenciar o acesso a arquivos

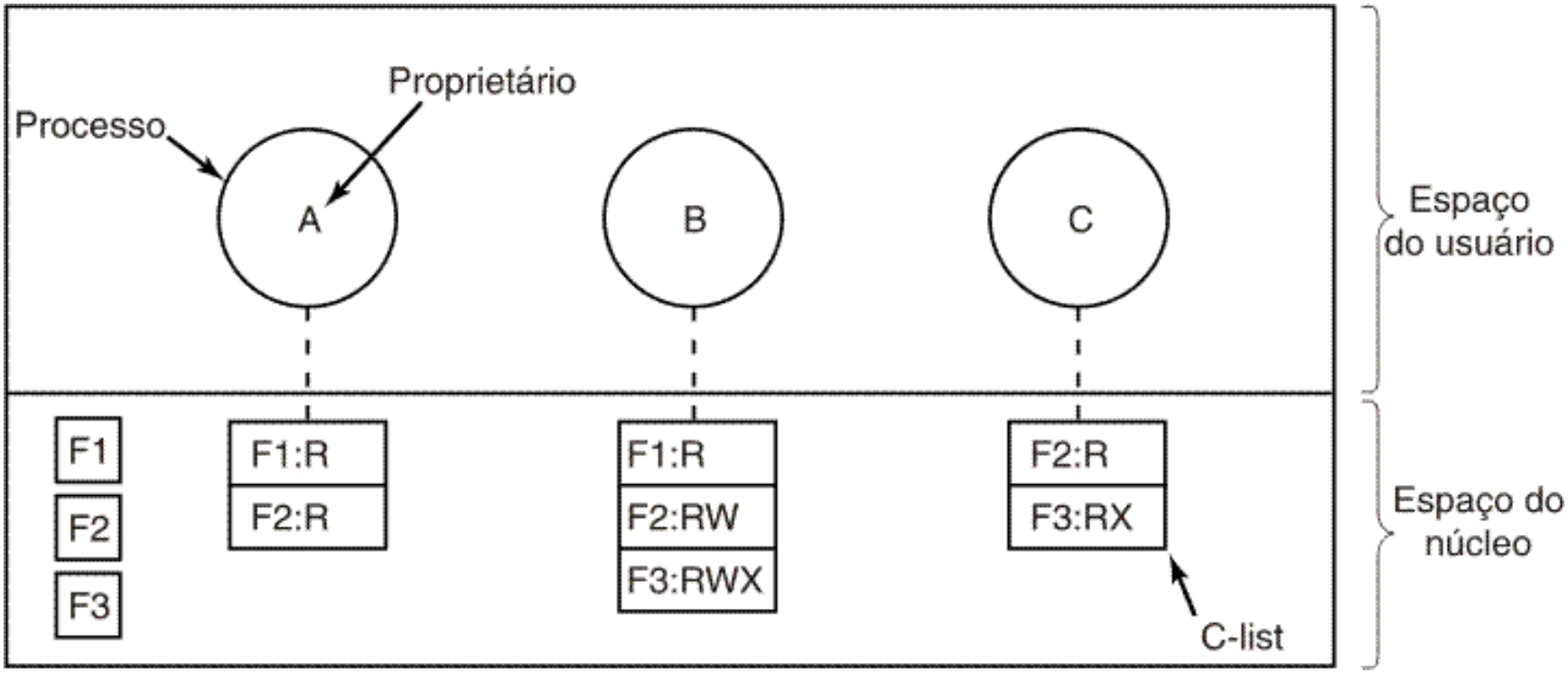
Listas de Controle de Acesso (2)



Arquivo	Lista de controle de acesso
Senha	ana, sysadm: RW
Dados_pombos	bill, crdpmb: RW; ana, crdpmb: RW; ...

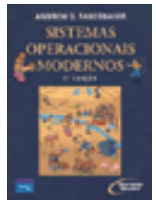
Duas listas de controle de acesso

Capacidades (1)



Cada processo tem uma lista de capacidades

Capacidades (2)

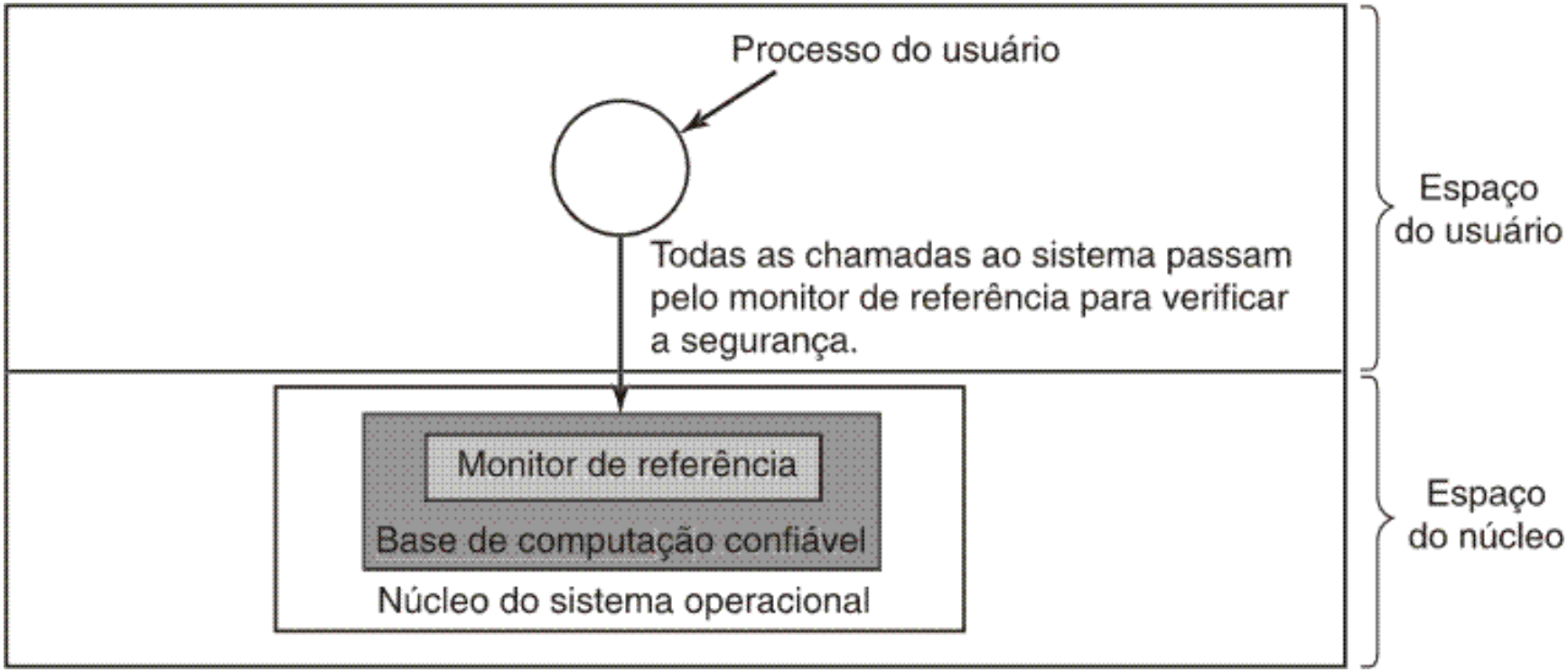


- Capacidade criptograficamente protegida

servidor	objeto	direitos	f (objetos, direitos, verificação)

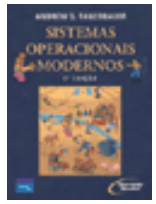
- Direitos genéricos
  1. copia capacidade
  2. copia objeto
  3. remove capacidade
  4. destrói objeto

Sistemas Confiáveis
Base de Computação Confiável



Um monitor de referência

Modelos Formais de Sistemas Seguros



Objetos

	Compilador	Caixa postal7	Secreto
Érico	Lê Executa		
Henrique	Lê Executa	Executa Escreve	
Roberto	Lê Executa		Executa Escreve

(a)

Objetos

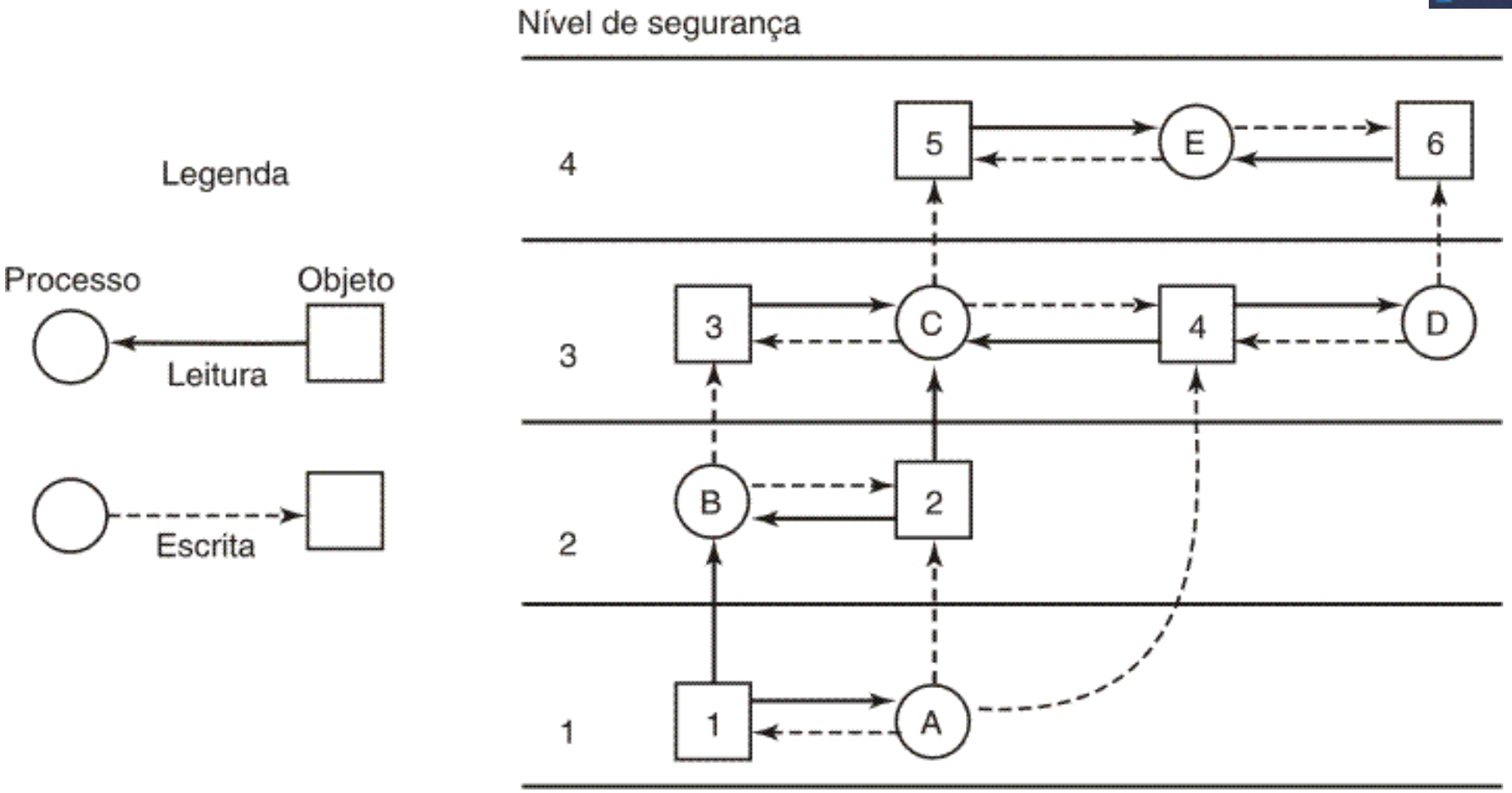
	Compilador	Caixa postal7	Secreto
Érico	Lê Executa		
Henrique	Lê Executa	Executa Escreve	
Roberto	Lê Executa	Executa	Executa Escreve

(b)

(a) Um estado autorizado

(b) Um estado não autorizado

Segurança Multiníveis (1)



O modelo de segurança multiníveis Bell-La Padula

Segurança Multiníveis (2)

O Modelo Biba

- Princípios para garantir a integridade dos dados

4. Propriedade de integridade simples
   - um processo só pode escrever objetos em seu nível de segurança ou inferior

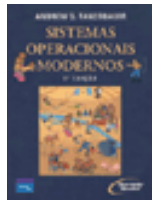
5. Propriedade de integridade*
   - um processo só pode ler objetos em seu nível de segurança ou superior

O Livro Laranja sobre Segurança (1)

Critério	D	C1	C2	B1	B2	B3	A1
Política se segurança							
Controle de acesso discricionário		X	X	→	→	X	→
Reutilização de objeto			X	→	→	→	→
Rótulos				X	X	→	→
Integridade dos rótulos				X	→	→	→
Exportação de informação rotulada				X	→	→	→
Rotulação de saída legível por humanos				X	→	→	→
Controle de acesso obrigatório				X	X	→	→
Rótulo de sensibilidade do sujeito		X			X	→	→
Rótulos de dispositivo		X			X	→	→
Contabilidade							
Identificação e autenticação		X	X	X	→	→	→
Auditoria			X	X	X	X	→
Caminho confiável					X	X	→

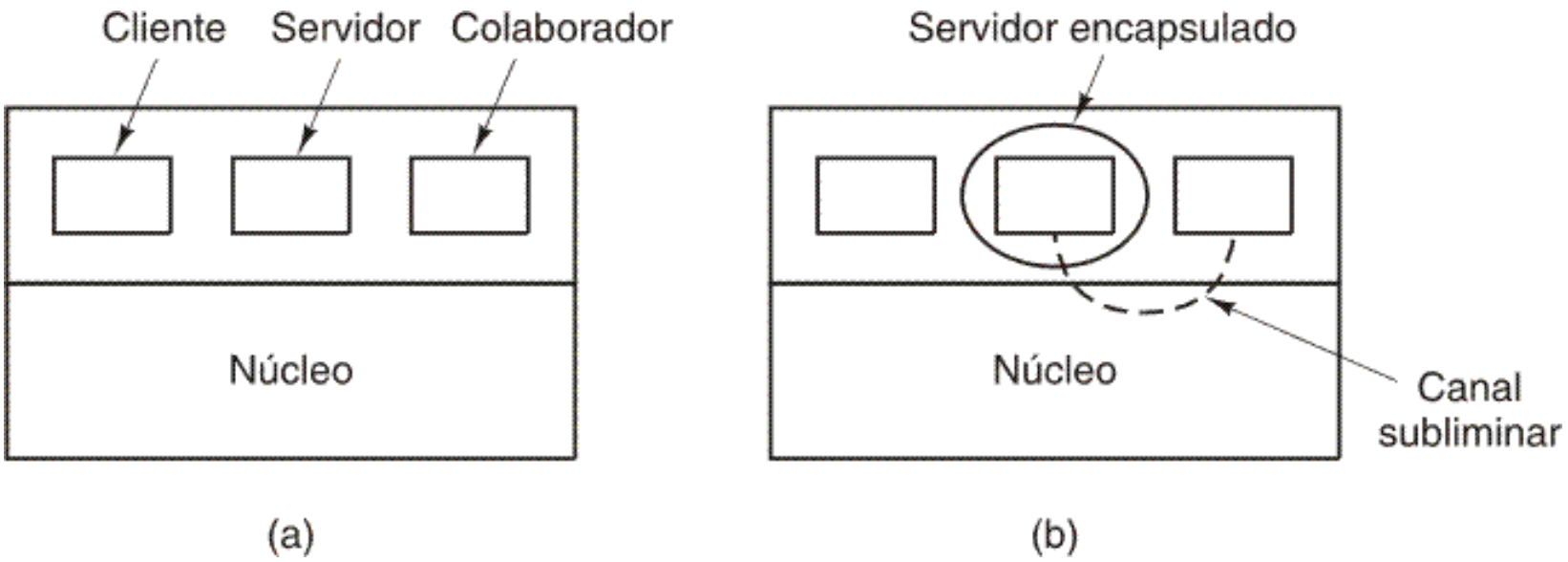
- Símbolo X significa novos requisitos
- Símbolo -> indica que requisitos da próxima categoria inferior também se aplicam

O Livro Laranja sobre Segurança (2)



Critério	D	C1	C2	B1	B2	B3	A1
Política se segurança							
Controle de acesso discricionário		X	X	→	→	X	→
Reutilização de objeto			X	→	→	→	→
Rótulos				X	X	→	→
Integridade dos rótulos				X	→	→	→
Exportação de informação rotulada				X	→	→	→
Rotulação de saída legível por humanos				X	→	→	→
Controle de acesso obrigatório				X	X	→	→
Rótulo de sensibilidade do sujeito		X			X	→	→
Rótulos de dispositivo		X			X	→	→
Contabilidade							
Identificação e autenticação		X	X	X	→	→	→
Auditoria			X	X	X	X	→
Caminho confiável					X	X	→

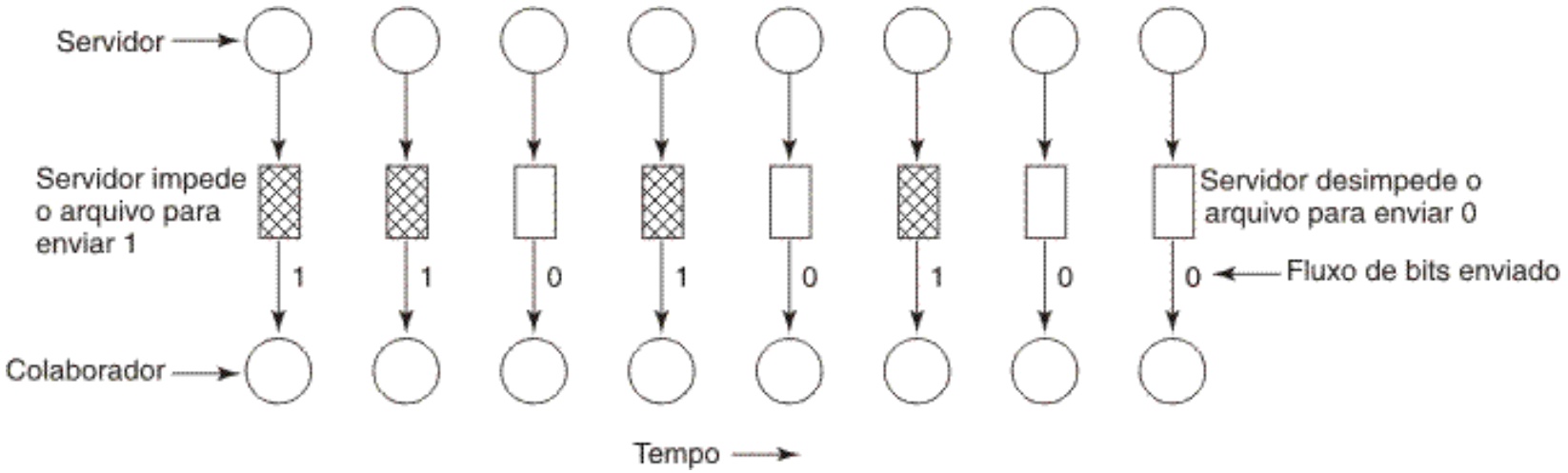
Canais Subliminares (1)



Processos cliente, servidor e colaborador

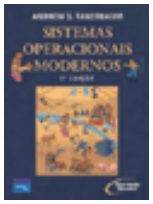
Servidor encapsulado ainda pode passar informações ao colaborador por canais subliminares

Canais Subliminares (2)

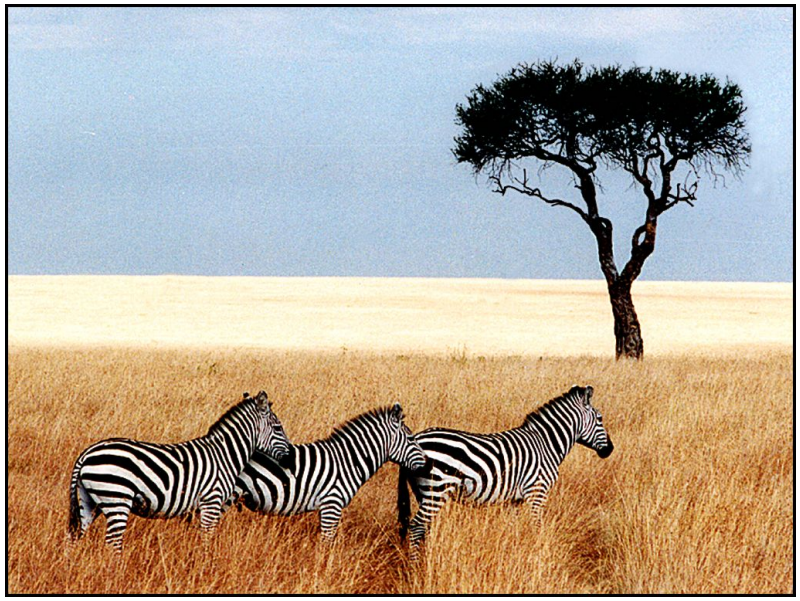


Um canal subliminar usando impedimento de arquivo

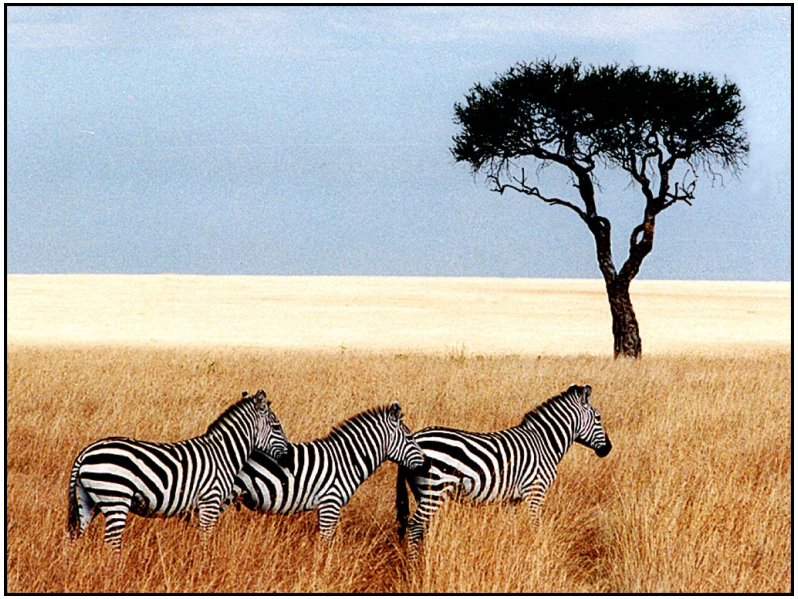
Canais Subliminares (3)



- Imagens parecem as mesmas
- Imagem à direita contém os textos de 5 peças de Shakespeare
  - criptografados, inseridos nos bits menos significativos de cada valor de cor



Zebras



Hamlet, Macbeth, Julius Caesar
Mercador de Veneza, Rei Lear