Segurança Computacional

Conceitos básicos

Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Julho de 2019

Conteúdo

Introdução

Introdução

Segurança em Inglês:

- **Security**: ameaças intencionais (intrusões, ataques e roubo de informações)
- **Safety**: problemas causados pelo sistema aos usuários ou ao ambiente (erros de programação que provocam acidentes)
- **Reliability**: sistemas construídos para tolerar erros de software, de hardware ou dos usuários

Nesta disciplina serão considerados os aspectos de segurança relacionados ameaças **intencionais**.

Propriedades de segurança

Propriedades de segurança

Confidencialidade

Os recursos do sistema só podem ser lidos por usuários autorizados.

Integridade

Os recursos do sistema só podem ser modificados por usuários autorizados.

Disponibilidade

Os recursos devem estar sempre disponíveis aos usuários autorizados.

Propriedades complementares

Autenticidade

Todas as entidades do sistema são autênticas (usuário, conteúdo de arquivo, endereço de rede, etc)

Irretratabilidade (non-repudiation)

Todas as ações realizadas no sistema são conhecidas e não podem ser negadas por seus autores.

O sistema computacional (hardware, SO, bibliotecas, serviços, aplicações) deve garantir as propriedades de segurança!

Princípios de segurança

Princípios de segurança

Privilégio mínimo

Usuários e programas devem operar com o mínimo possível de permissões de acesso, para minimizar danos.

Contra-exemplo: serviços de rede executando como *root*.

Separação de privilégios

Sistemas de proteção baseados em mais de um controle; se o atacante burlar um dos controles, ainda não terá acesso.

Exemplo: autorização de dois gerentes em um sistema bancário.

Princípios de segurança

Mediação completa

Todos os acessos a recursos são avaliados pelos mecanismos de segurança, sendo impossível contorná-los.

Exemplo: verificações nas *syscalls* de acesso a arquivos.

Default seguro

Os acessos permitidos devem ser claramente indicados; se um acesso não for explicitamente permitido, ele deve ser negado.

Contra-exemplo: Sistemas com senha *default* em branco.

Princípios de segurança

Economia de mecanismo

O projeto de um sistema de proteção deve ser pequeno e simples, fácil de analisar, testar e validar.

Exemplo: kernel de sistema operacional L4.

Compartilhamento mínimo

Mecanismos compartilhados são fontes de problemas de segurança, devido à possibilidade de fluxos de informação imprevistos.

Exemplo: operação implementada como *syscall* ou como *libcall*.

Princípios de segurança

Projeto aberto

O projeto deve ser público e aberto, dependendo somente do segredo de poucos itens, como listas de senhas ou chaves, para ser avaliado por terceiros independentes.

Exemplo: algoritmos de criptografia RSA e AES.

Proteção adequada

Cada recurso deve ter um nível de proteção coerente com seu valor.

Exemplo: o nível de proteção em um servidor Web bancário é distinto daquele de um terminal público de acesso à Internet.

Princípios de segurança

Facilidade de uso

O uso dos mecanismos de segurança deve ser fácil e intuitivo, caso contrário eles serão evitados pelos usuários.

Exemplo: senhas muito complexas para sistemas simples.

Eficiência

Os mecanismos de segurança não devem afetar o desempenho do sistema ou das atividades de seus usuários.

Exemplo: controle de acesso aos arquivos em um SO.

Princípios de segurança

Elo mais fraco

A segurança do sistema é limitada pela segurança de seu elemento mais vulnerável: o SO, as bibliotecas, aplicações, a rede ou o próprio usuário.

Exemplo: ataques de engenharia social.

Podem ser encontrados muitos outros princípios de segurança.

Ameaças

Ameaças

Ação que põe em risco propriedades de segurança do sistema:

- **à Confidencialidade**: um processo vasculhar a memória de outros processos em busca de dados sensíveis
- **à Integridade**: um processo alterar as senhas de outros usuários ou instalar programas maliciosos
- **à Disponibilidade**: um usuário alocar para si todos os recursos do sistema (memória, CPU ou espaço em disco)

Modelo de ameaças: conjunto de ameaças possíveis ao sistema.

Para cada ameaça devem existir um controle que a impeça.

Vulnerabilidades

Vulnerabilidade

Problemas presentes no sistema, que podem permitir a violação das propriedades de segurança.

- Erro de **projeto**
- Erro de **implementação**
- Erro de **configuração**
- Erro de **operação**

Exemplos de vulnerabilidades

- No **projeto**: erros no protocolo de rede sem fio WEP;
- Na **implementação**: erro de programação no serviço de compartilhamento de arquivos do Windows (usuários externos podem acessar todos os arquivos do computador);
- Na **configuração**: uma conta de usuário sem senha, ou com uma senha pré-definida pelo fabricante, que permita a usuários não-autorizados acessar o sistema;
- Na **operação**: fazer um backup via rede sem usar protocolo cifrado.

Vulnerabilidades clássicas

- o serviço de impressão `lpr/lpd`, usado em alguns UNIX, pode ser instruído a imprimir um arquivo e a seguir apagá-lo (útil para imprimir arquivos temporários):

  ```
  lpr -r /etc/passwd
  ```

- Uma versão antiga do IIS não verificava adequadamente os pedidos dos clientes:

  ```
  http://www.servidor.com/../../../../windows/system.ini
  ```

Vulnerabilidade: *Buffer overflow*

```
1  int i, j, buffer[20], k;  // declara buffer[0] a buffer[19]
2
3  int main() {
4    i = j = k = 0 ;
5    printf ("Antes : i= %d, j= %d, k= %d\n", i, j, k) ;
6    for (i = 0; i<= 20; i++) // usa buffer[0] a buffer[20] <-- erro!
7       buffer[i] = random() ;
8    printf ("Depois: i= %d, j= %d, k= %d\n", i, j, k) ;
9  }
```

```
1  host:~> cc buffer-overflow.c -o buffer-overflow
2  host:~> buffer-overflow
3  Antes : i= 0,  j= 0,  k= 0
4  Depois: i= 21,  j= 35005211,  k= 0
```

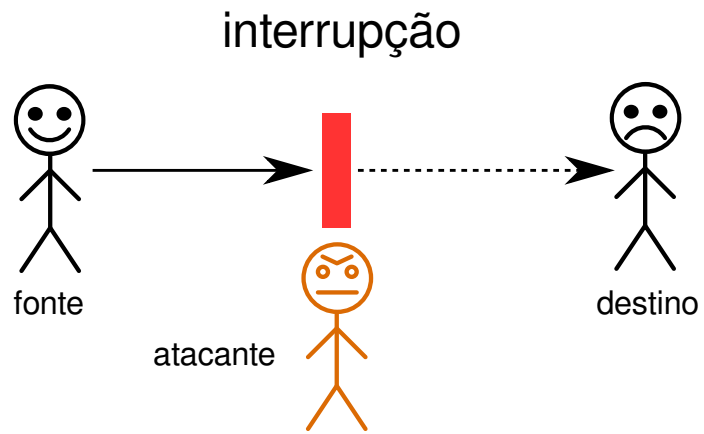
Ataques

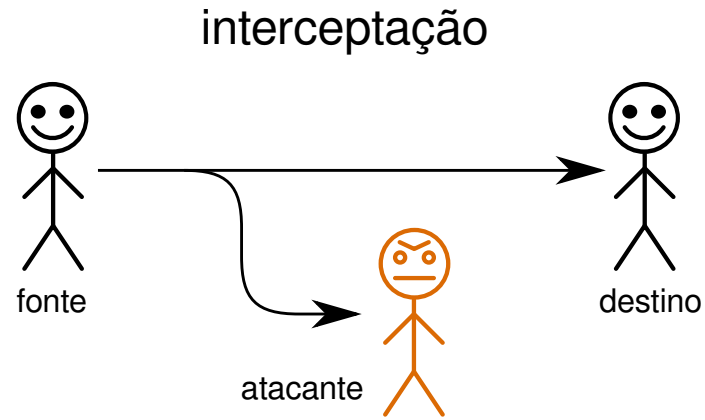
Ataques

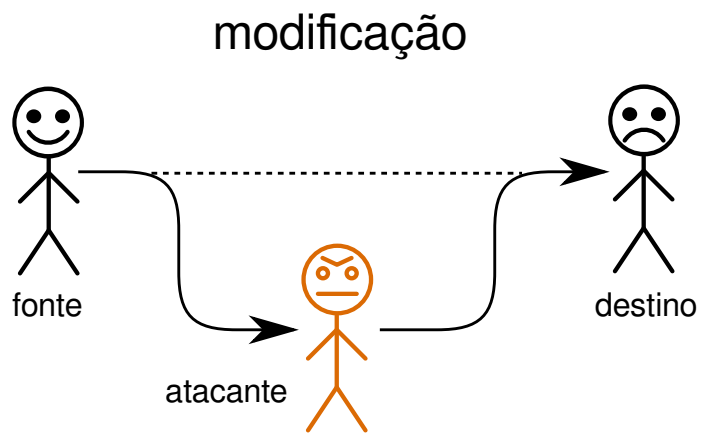
Ato de utilizar (ou explorar) uma vulnerabilidade para violar uma propriedade de segurança do sistema.

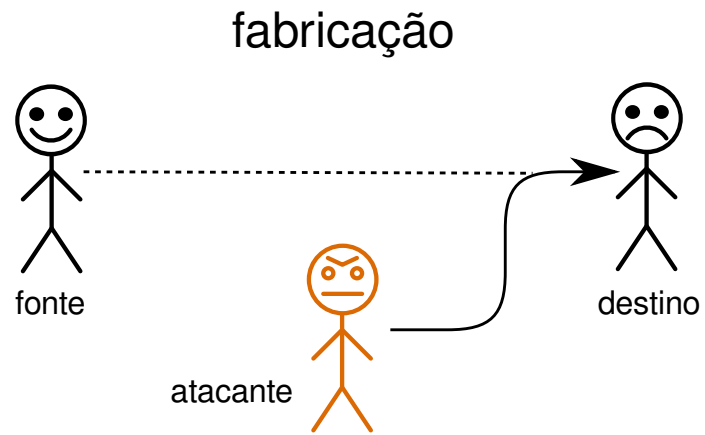
- **Interrupção**: impedir o fluxo normal das informações ou acessos; é um ataque à disponibilidade do sistema;
- **Interceptação**: obter acesso indevido a um fluxo de informações, sem necessariamente modificá-las; é um ataque à confidencialidade;
- **Modificação**: modificar de forma indevida informações ou partes do sistema, violando sua integridade;
- **Fabricação**: produzir informações falsas ou introduzir módulos ou componentes maliciosos no sistema; é um ataque à autenticidade.

Ataques


interrupção
fonte
atacante
destino


interceptação
fonte
atacante
destino


modificação
fonte
atacante
destino


fabricação
fonte
atacante
destino

Ataques

Quanto à atividade do atacante:

- **Ataque passivo**: o atacante apenas observa e captura informações.
- **Ataque ativo**: o atacante atua para alterar o sistema, os dados ou dificultar o uso dos usuários válidos.

Quanto à localização do atacante:

- **Ataque local**: executado por um usuário válido do sistema.
- **Ataque remoto**: realizado através da rede, sem uma conta local.

Ataques

Elevação de privilégios (*Privilege escalation*)

- Mecanismo para aumentar os privilégios de um usuário
- Necessário para atividades administrativas
- Bugs podem permitir elevação não-controlada

Tipos:

- **Vertical**: usuário ganha acesso mais elevado
- **Horizontal**: usuário ganha acesso de outro usuário

Ataques

Exemplos de elevação de privilégios:

Controlados:

- Comandos `su` e `sudo` (Linux)
- "Executar como administrador" (Windows)
- Flags `setuid`/`setgid` (Unix)

Não-controlados:

- "Rootear" telefone celular (Android)
- "Jailbreak" telefone celular (iPhone)

Ataques de negação de serviço

- DoS – *Denial of Service*
- Podem ser locais, remotos ou distribuídos
- Visam prejudicar a disponibilidade do sistema
- Comuns na Internet (impedir acesso a servidores)
- Baseados no consumo dos recursos locais (CPU, RAM)

Exemplo de DoS local: *Fork Bomb*:

```
1  int main()
2  {
3    while (1)    // laço infinito
4       fork() ;  // reproduz o processo
5  }
```

Malware

Malware

Malware = Malicious Software

- **Vírus**: trecho de código que se infiltra em programas executáveis, usando-os como suporte de execução e replicação.
- **Worm**: programa que se propaga de forma autônoma, explorando vulnerabilidades nos serviços de rede para invadir e instalar-se em sistemas remotos.
- **Trojan**: programa aparentemente normal, mas com funcionalidades desconhecidas do usuário e executadas sem que ele o saiba.

Malware

- **Exploit**: programa escrito para explorar vulnerabilidades conhecidas, como prova de conceito ou parte de um ataque.
- **Sniffer**: captura pacotes de rede do próprio computador ou da rede local, analisando-os em busca de informações sensíveis.
- **Keylogger**: captura e analisa as informações digitadas pelo usuário na máquina local, sem seu conhecimento.

Malware

- **Rootkit**: conjunto de programas para ocultar a presença de um intruso no SO, modificando os mecanismos que mostram os processos em execução, arquivos nos discos, portas e conexões de rede, etc.
- **Backdoor**: facilita a entrada posterior do atacante em um sistema invadido. Usa um processo servidor de conexões remotas (com SSH, telnet ou um protocolo ad-hoc).
- **Ransomware**: sequestra os arquivos do usuário, cifrando-os com uma chave secreta, que só é fornecida mediante pagamento.

Muitos *malwares* se encaixam em mais de uma categoria!

Infraestrutura de segurança

Base de Computação Confiável

TCB - *Trusted Computing Base*

Conjunto de todos os elementos de hardware e software críticos para a segurança de um sistema.

- Mecanismos de hardware:
  - tabelas de páginas/segmentos
  - modo usuário/núcleo do processador
  - instruções privilegiadas
- Subsistemas do SO:
  - controle de acesso aos arquivos
  - portas de rede privilegiada
  - autenticação de usuários

Infraestrutura de segurança

Áreas de atuação do sistema computacional (AAA):

- **Autenticação**: identificar usuários e recursos
  - pares *login/senha*
  - biometria
  - certificados criptográficos
- **Autorização**: definir ações permitidas ou negadas
  - regras descrevendo as ações permitidas aos usuários
  - política de controle de acesso
- **Auditoria**: registrar atividades executadas
  - contabilização de uso dos recursos
  - a análise posterior de situações de uso indevido
  - identificação de comportamentos suspeitos.

Infraestrutura de segurança