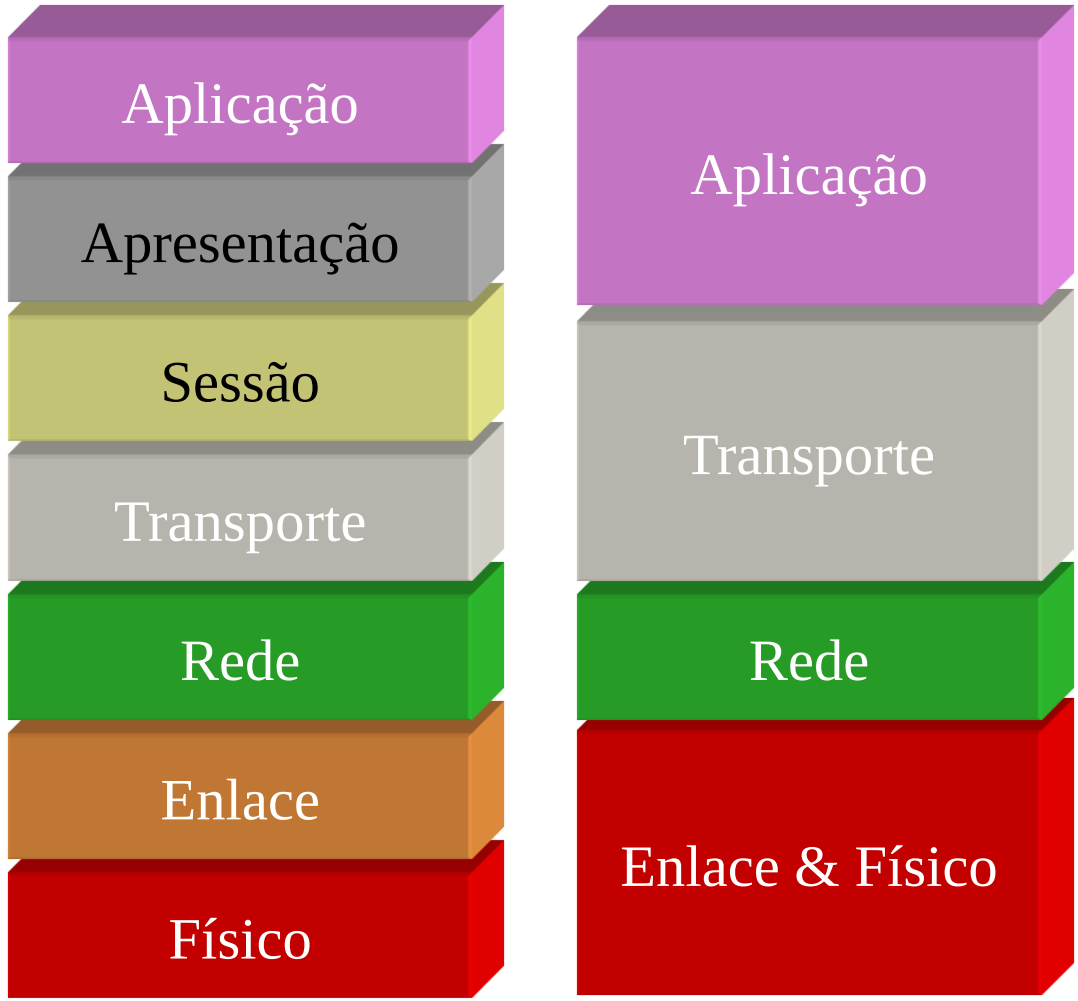
EN3611

Segurança de Redes

Vulnerabilidades e Ataques

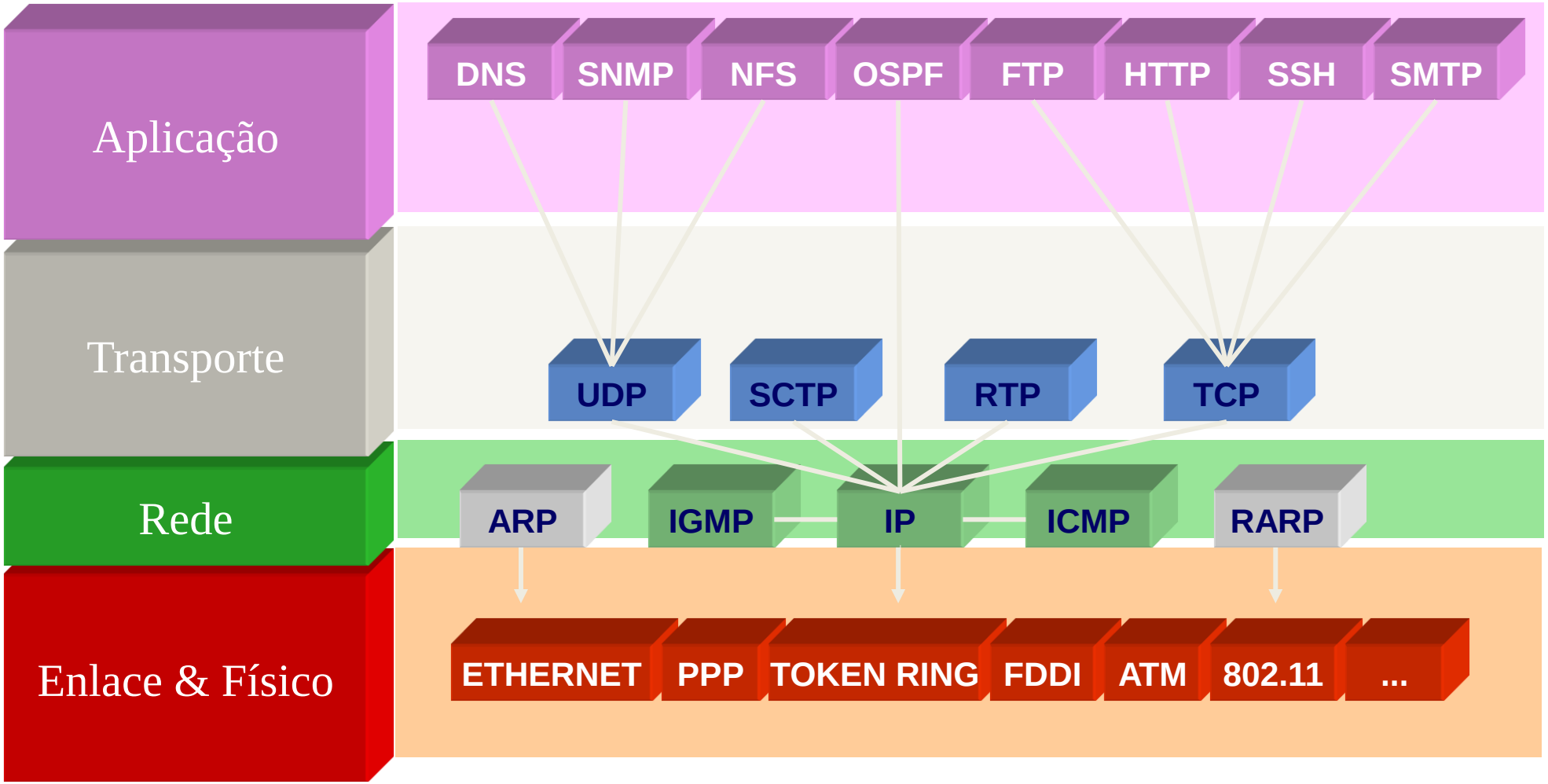
Prof. João Henrique Kleinschmidt

Arquiteturas OSI & TCP/IP

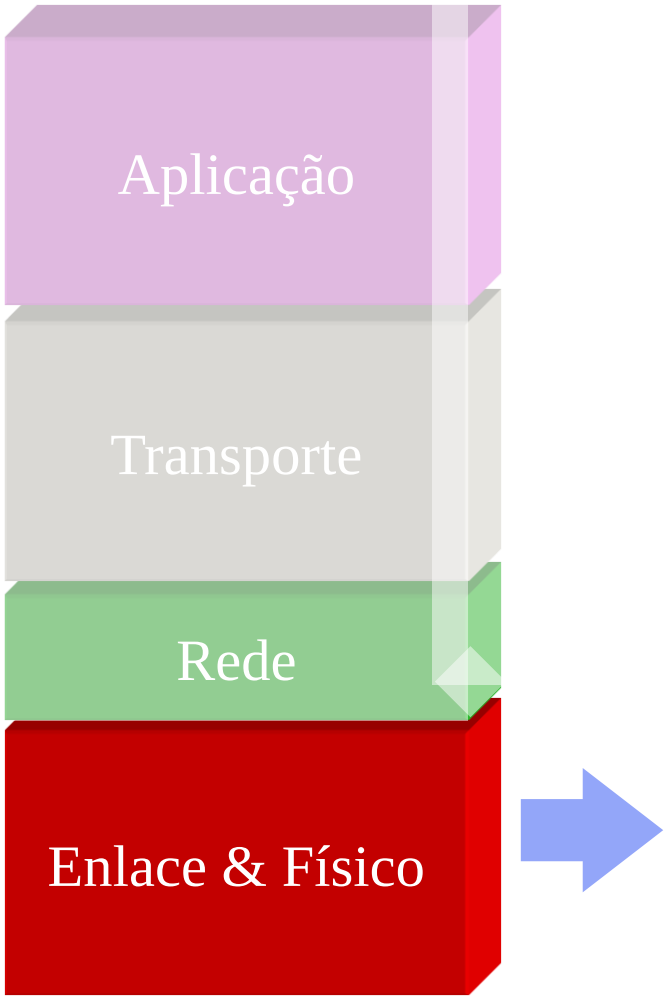


OSI **TCP/IP**

Família TCP/IP



Ameaças na pilha TCP/IP



Modelo TCP/IP

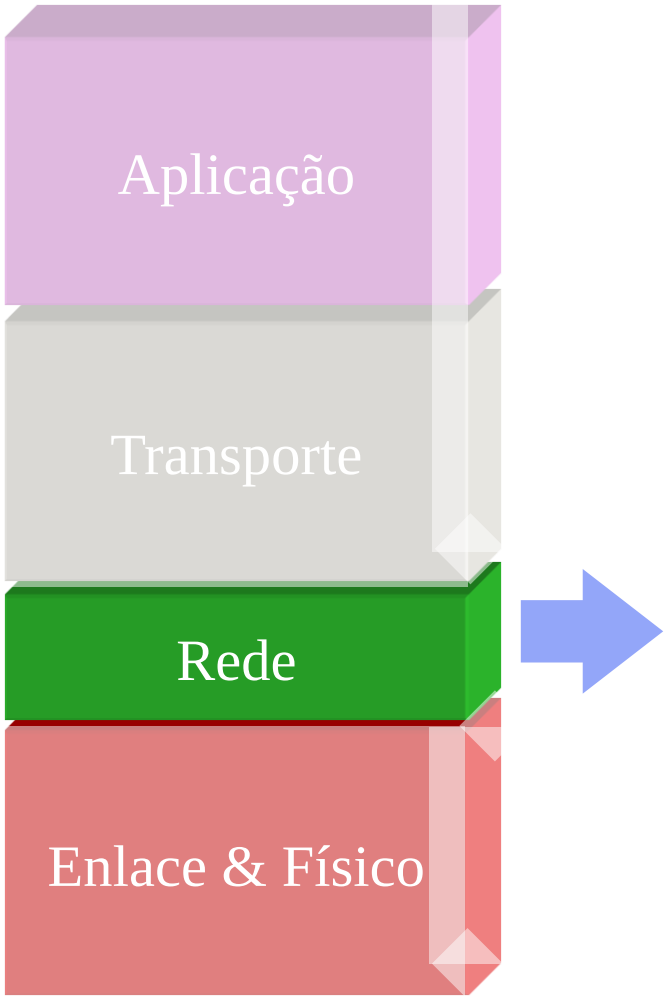
Funções do nível Enlace & Físico

Define as propriedades da rede, como níveis de voltagem, tipos e tamanhos de cabos, conectores, frequência; Transferência confiável no meio físico, acesso ao meio

Algumas Ameaças

- **Vandalismo**
  - **Acesso cabos lógicos e de força, disjuntores**
  - **Acesso a equipamentos, racks distribuídos no prédio**
- **Manutenção na rede elétrica interfere na rede**
- **Picos de energia afetam serviços**
- **Redes sem fio**
  - **Frequência do IEEE 802.11 é 2.4GHz (ISM)**
  - **Muitos dispositivos podem interferir (ex: fornos microondas, Bluetooth) e causar DoS (denial of service)**
  - **Captura do sinal, captura de pacotes**
  - **Criptografia de nível físico deve estar habilitada**
- **ARP cache poisoning (envenenamento tabela ARP)**

Ameaças na pilha TCP/IP



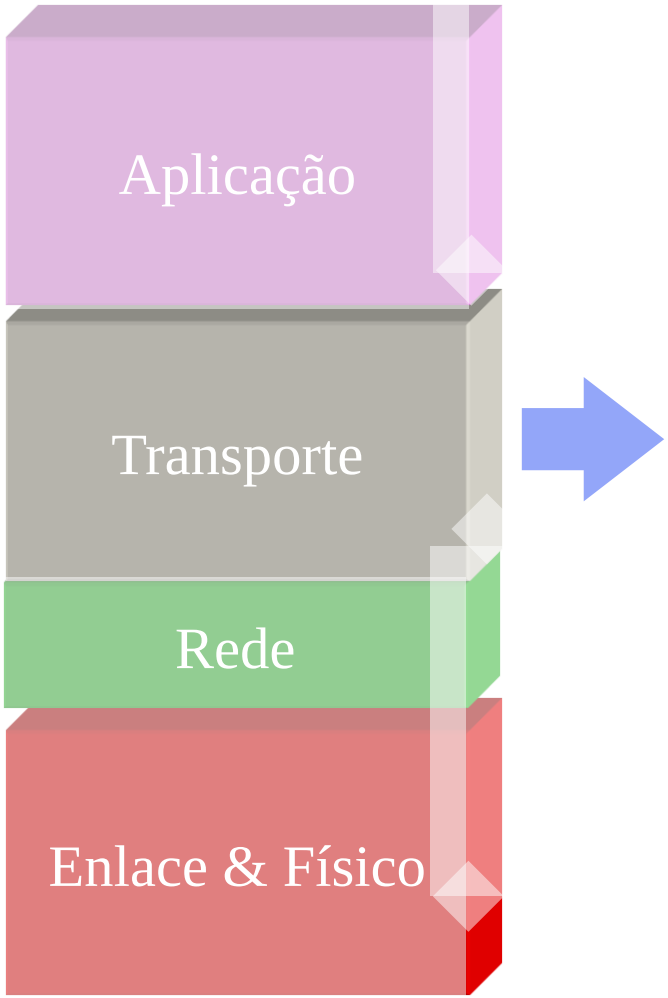
Funções do nível de Redes

Rotear dados através de várias redes até o destino final

Algumas Ameaças

- Vulnerabilidades em roteadores
  - Administradores usam senha de administração default do fabricante
  - Bugs no software permitem “buffer overflow”
- Firewalls mal configurados
- IP – Internet Protocol
  - IPv4 não oferece confidencialidade
  - Pacotes podem ser lidos na rede pública ou no ISP
  - Ataques spoofed (source IP falso)
  - DoS (ex: ping da morte)
- Vulnerabilidades no BGP, OSPF

Ameaças na pilha TCP/IP



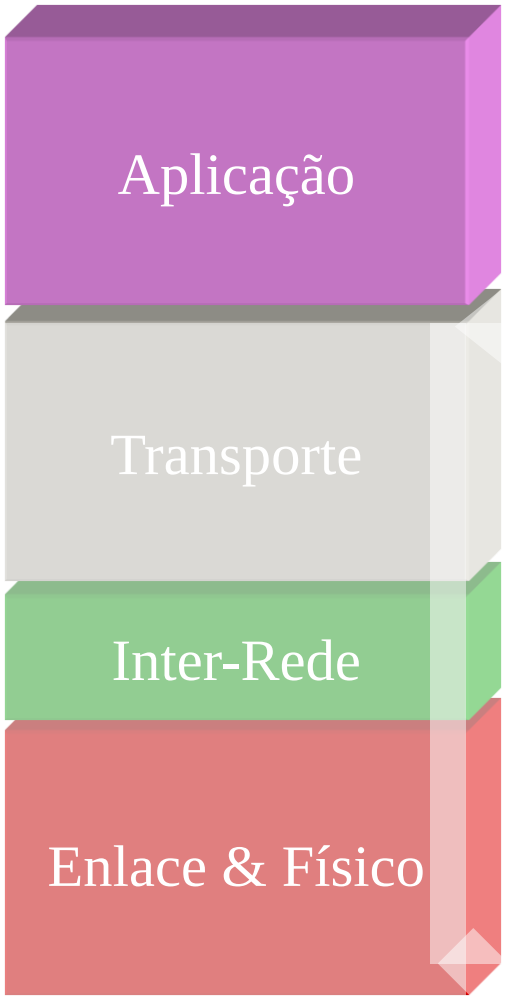
Funções do nível de Transporte

Prover controle de fluxo, sequenciamento de pacotes, controle de congestionamento e retransmissão de pacotes perdidos pela camada de rede (TCP)

Algumas Ameaças

- Aplicações TCP, UDP são alvo de "port scan"
  - Port scan permite "mapeamento da rede"
  - Porta aberta = serviço rodando
- Negação de Serviço (Dos)
  - SYN flood
  - TCP session hijacking
  - "man-in-the-middle attack" (MITM)
  - LAND/La tierra
  - Ping da morte
  - Nuke
  - …

Ameaças na pilha TCP/IP



Modelo TCP/IP

Funções do nível de Aplicação

Protocolos de alto nível e aplicações

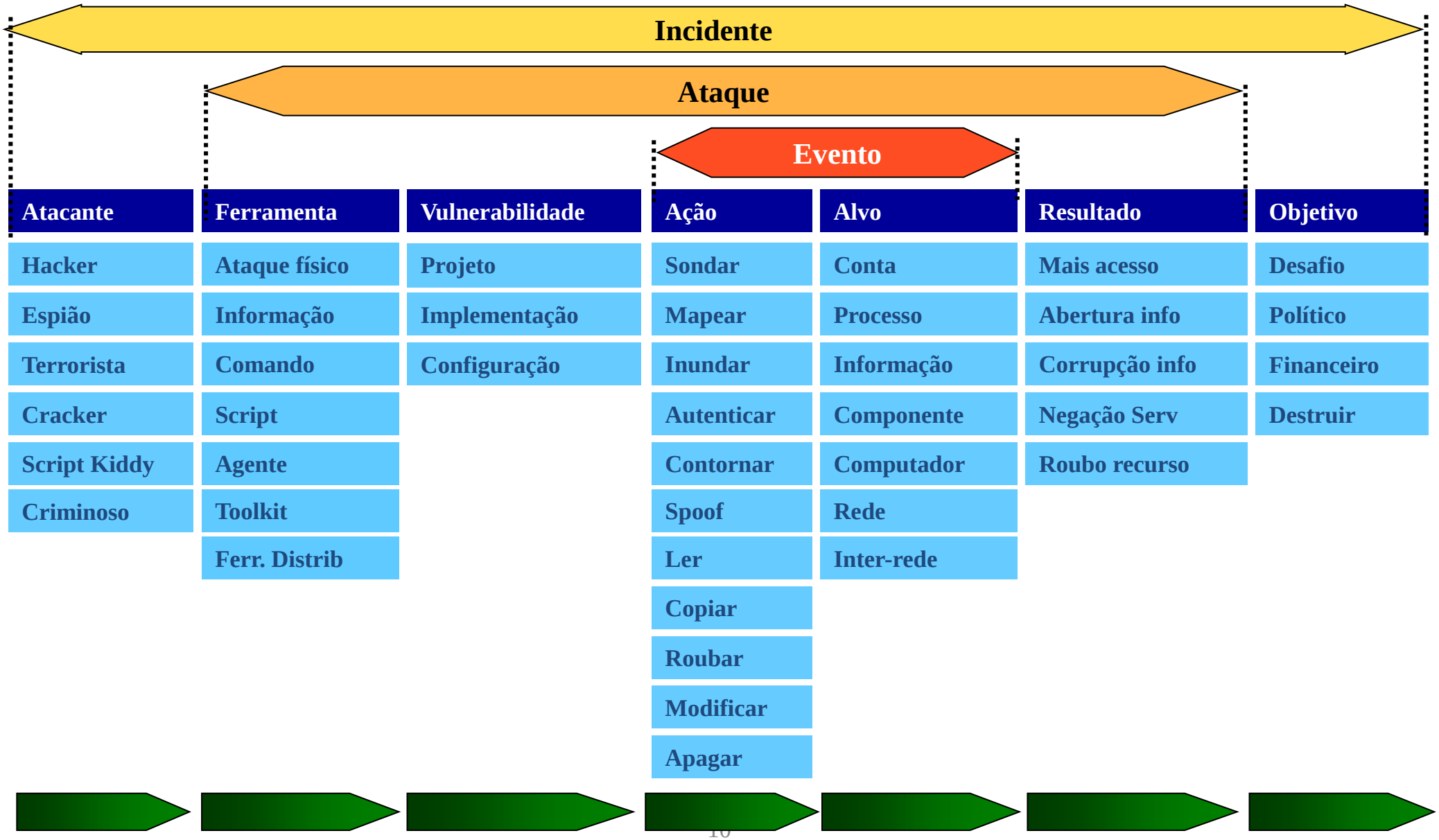
Algumas Ameaças

- Plugins para Browsers (ActiveX, Applets Java)
- Senhas enviadas sem criptografia
- Vírus, Worms, Trojans
  - Vírus são programas malignos que se replicam
  - Trojan horse → programa maligno disfarçado de benigno
  - Worms são vírus auto-replicáveis
- Bugs de software → vulnerabilidade
  - Muitos serviços rodam com privilégios
  - Vulnerabilidades em SNMP, SSH, FTP, …
  - Boa especificação, falhas na implementação
- Falha na configuração de serviços (FTP, HTTP, ...)
  - Boa especificação, boa implementação, falhas de configuração

Terminologia

- Ataque
  - acesso a dados ou uso de recursos sem autorização
  - execução de comandos como outro usuário
  - uso de falsa identidade
  - condução de negação de serviços (denial of service)
  - Violação de uma política de segurança
- Vulnerabilidade
  - É uma falha que pode permitir a condução de um ataque
- Incidente
  - A ocorrência de um ataque; exploração de vulnerabilidades
- Ameaça
  - Qualquer evento que pode causar dano a um sistema ou rede
  - Por definição, a existência de uma vulnerabilidade implica em uma ameaça
- Exploit code
  - Um código preparado para explorar uma vulnerabilidade conhecida
  - Uma ferramenta para ataques

Incidentes


Incidente
Ataque
Evento
Atacante
Ferramenta
Vulnerabilidade
Ação
Alvo
Resultado
Objetivo
Hacker
Espião
Terrorista
Cracker
Script Kiddy
Criminoso
Ataque físico
Informação
Comando
Script
Agente
Toolkit
Ferr. Distrib
Projeto
Implementação
Configuração
Sondar
Mapear
Inundar
Autenticar
Contornar
Spoof
Ler
Copiar
Roubar
Modificar
Apagar
Conta
Processo
Informação
Componente
Computador
Rede
Inter-rede
Mais acesso
Abertura info
Corrupção info
Negação Serv
Roubo recurso
Desafio
Político
Financeiro
Destruir

Por que estamos vulneráveis?

- Sistemas se tornaram muito complexos nas últimas décadas
- Controle de qualidade não tem tido força contra as pressões do mercado
- Deficiências no perfil dos programadores
- Milhares de computadores se somam à Internet todo mês
- Administradores de rede: pouco treinamento, muitas prioridades conflitantes
- Prioridade: manter o sistema funcionando
- Atualizar sistema: quando tiver tempo...
- Nem todas as portas estão fechadas, porque pessoas estão ocupadas fazendo coisas "úteis"
- Invasores dificilmente são pegos. Quando o são, não são punidos

Atacantes e seus objetivos

- **Hackers** – Atacam sistemas pelo desafio técnico, por status ou pela "adrenalina" da invasão. Constroem suas próprias ferramentas
- **Cracker, vândalo, defacer** – Hacker com fins destrutivos (blck hat)
- **Espião, Corporate raider** – Ataca sistemas dos concorrentes pelo roubo da informação para ganho financeiro ou competitivo
- **Terroristas** – Atacam para causar medo ou por motivação política
- **Criminoso, Carder** – Atacam sistema para obter ganho financeiro pessoal
- **Voyeur** – Atacam para ter acesso à informação
- **Script Kiddy** – Novato que se acha hacker, mas apenas segue "receitas de bolo" disponíveis na Internet (não sabem exatamente porque aquilo funciona)
- **Lamer, Luser** – que se diz hacker (hacker não se diz hacker) *"If you're a good hacker everybody knows you. If you're a great hacker, no one does."*
- **Phreaker** – Um hacker com maior atuação em telecomunicações
- **White Hat** – Um hacker, possivelmente aposentado, que trabalha como consultor de segurança. Odiado pelos "Black Hats“
- **Grupos hackers –** Ex: Cult of the Dead Cow

Tipos de Ataques

Ataques para obtenção de informações

- *Dumpster diving* ou *trashing* – verificar o lixo em busca de informações
- Engenharia social – técnica que explora as fraquezas humanas e sociais
- Ataque físico
- Informações livres
- Farejamento de pacotes (Packet sniffing)
- Scanner de portas (Port scanning)
- Scanning de vulnerabilidades
- Firewalking
- IP spoofing

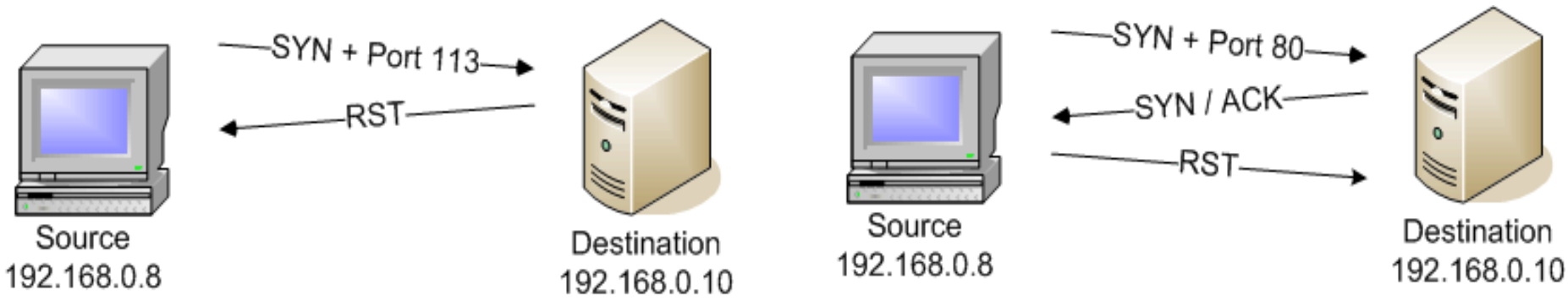
Packet sniffing

- Chamada também de *passive eavesdropping*
- “Farejam” o que passa pela rede
- Consiste na captura de informações pelo fluxo de pacotes de um mesmo segmento de rede
- Switches podem ser usados para dividir a rede em mais segmentos
- Softwares: *snoop* (Solaris), *tcpdump* (Linux), *wireshark*

Port scanning

- Ferramentas utilizadas para obtenção de informações referentes aos serviços que são acessíveis e definidas por meio do mapeamento das portas TCP e UDP
- Ex: *nmap*
- Detecta portas abertas do sistema, sistema operacional, número de sequência de pacotes TCP, usuário que está executando cada serviço. Alguns métodos:
  - TCP connect, TCP SYN, UDP, ICMP, FIN, RPC scan, FTP proxy.

TCP SYN port scanning



porta fechada

porta aberta

Scanning de vulnerabilidades

- Após o mapeamento dos sistemas e serviços que podem ser atacados, as vulnerabilidades específicas serão procuradas por meio do scanning de vulnerabilidades. O port scanning define os alvos, evitando, por exemplo, que vulnerabilidades específicas do Windows sejam testadas em Unix. Alguns riscos que podem ser mapeados:
  - Compartilhamento de arquivos
  - Configuração incorreta
  - Software desatualizado
  - Configurações de roteadores perigosas
  - Checagem de cavalos de tróia
  - Configuração de serviços
  - Possibilidade de negação de serviço (DoS)
  - Checagem de senhas fáceis de serem adivinhadas

Scanning de vulnerabilidades

- Esta técnica pode ser utilizada para demonstrar problemas de segurança que existem nas organizações. Ex: Nessus

- Uma vulnerabilidade reportada pode não corresponder à situação real do sistema ou uma vulnerabilidade importante pode deixar de ser reportada, pois a ferramenta funciona por meio de uma base de dados de ataques conhecidos.

- É usado tanto por administradores de segurança como por hackers.

Ataques para obtenção de informações

- Firewalking
  - Técnica implementada para obter informações sobre uma rede remota protegida por um firewall
- IP spoofing
  - Endereço real do atacante é mascarado, de forma a evitar que ele seja encontrado.

Ataques de Negação de Serviço (DoS)

- DoS é um ação que impede ou prejudica o uso autorizado de redes, sistemas ou aplicações, através do esgotamento de recursos como CPU, memória, largura de banda e espaço de disco
- Alguns exemplos
  - Usar toda a banda disponível da rede pela geração de um volume de tráfego descomunal
  - Enviar pacotes mal formados para um servidor para que o sistema operacional pare
  - Enviar pedidos ilegais para uma aplicação para fazê-la parar
  - Fazer vários pedidos que consomem CPU para que a máquina não tenha condições de atender outros pedidos
  - Criar várias conexões simultâneas para que o servidor não possa aceitar outras conexões
  - Criar vários arquivos imensos em disco para ocupar todo o espaço disponível

DoS
Ataques de refletor

- Em um ataque de refletor, uma máquina envia muitos pedidos com um endereço de origem falsificado (spoofing) para uma máquina intermediária
- Esse endereço falsificado é o endereço da máquina que se quer atacar
- As máquinas intermediárias enviam respostas dos pedidos à máquina atacada, sem intenção de fazer isso
- Exemplos
  - DNS, SNMP, ICMP echo, etc
- Uma rede pode evitar ser usada como origem do ataque, proibindo a saída de pacotes com endereços de origem que não pertençam a ela

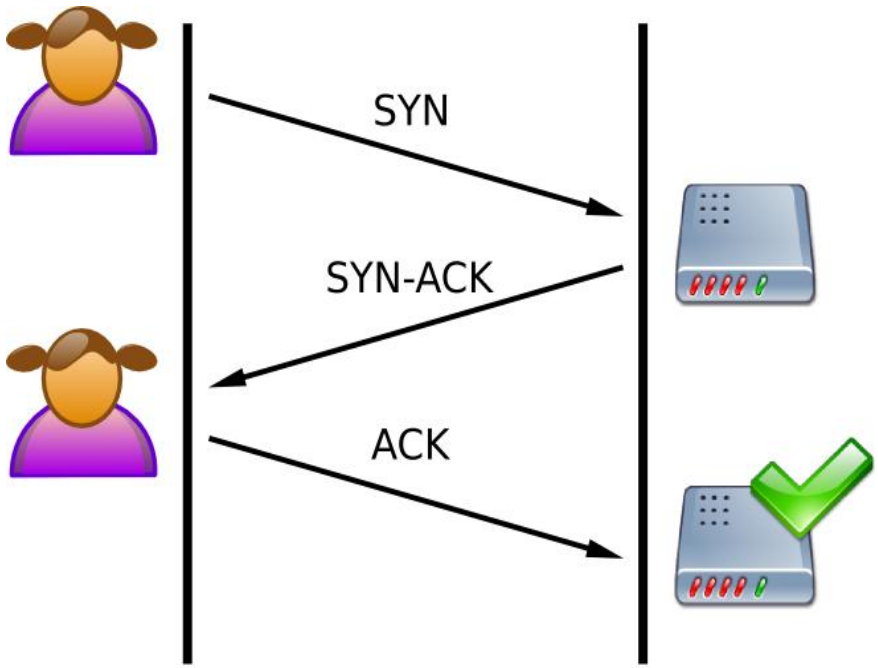
DoS
Ataques de amplificador

- É semelhante ao ataque de refletor, mas usa uma rede inteira como intermediária para gerar pedidos para a máquina atacada
- Isso é realizado enviando pacotes ICMP ou UDP para endereços de broadcast
- A esperança é que cada máquina que receba o pacote, responda para a máquina atacada
- Dessa forma, um único pacote pode ser multiplicado várias vezes, inundando a máquina atacada
- Uma rede consegue evitar ser usada como amplificador, proibindo pacotes de broadcast direcionados nos roteadores (depende da política de roteadores)

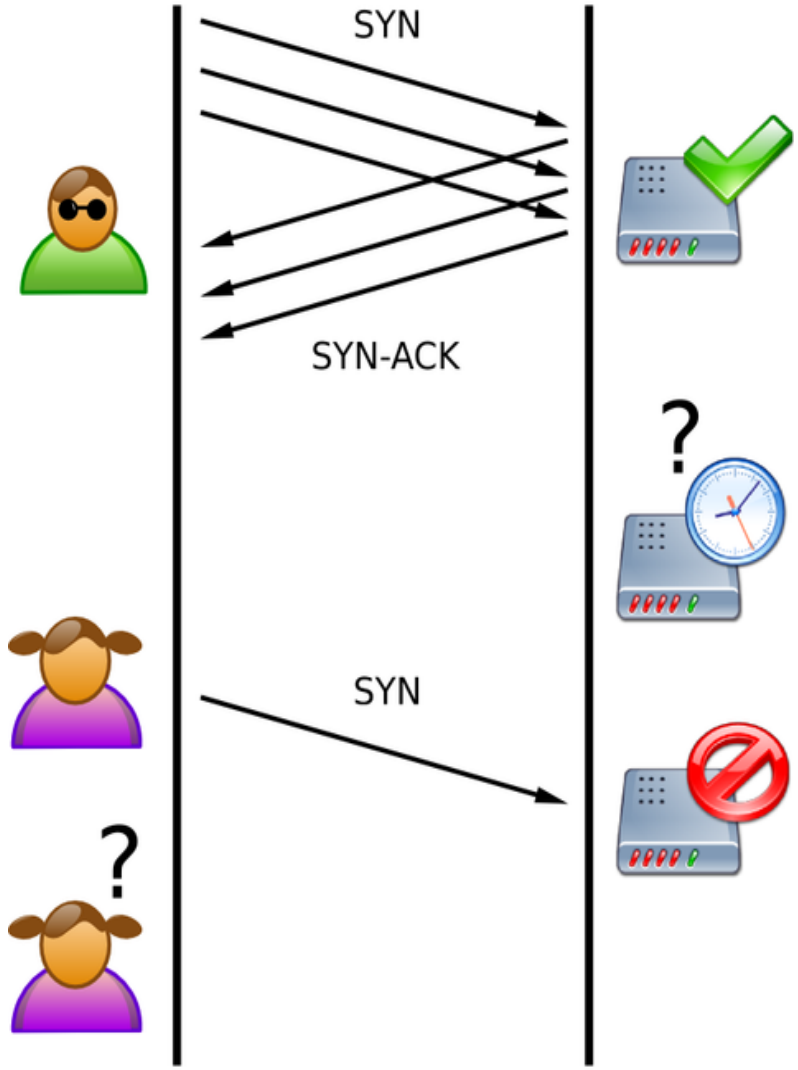
DoS – Ataques Syn Flood

- Em um ataque de inundação de SYN (synflood), um atacante inicia muitas conexões TCP em um curto período de tempo
- Ele usa um endereço falsificado e a conexão em três fases do TCP (three-way handshake) não é completada
  - Ou seja, ou a máquina não existe ou quando recebe um SYN ACK, não envia um pacote ACK
- O servidor atacado fica com muitas conexões incompletas "presas", que são liberadas por um temporizador (entre 2 a 4 minutos)
- Durante o ataque, o servidor fica praticamente impossibilitado de atender outras conexões, porque a sua tabela de conexões esgota a capacidade máxima

SYN Flood



Conexão normal entre usuário e servidor. O handshake em três vias é executado.



SYN Flood. Atacante envia pacotes SYN mas não retorna ACK. Usuário legítimo não consegue conexão.

Evitando SYN Flooding

- Comparar taxas de requisições de novas conexões e o número de conexões em aberto.
- Utilizar timeout e taxa máxima de conexões semi-abertas
- Aumento do tamanho da fila de pedidos de conexão
- Adicionar regra no firewall (Ex: *iptables*)

DoS -Fragmentação de pacotes de IP

- *Maximum Transfer Unit* (MTU) especifica a quantidade máxima de dados que podem passar em um pacote por um meio físico da rede. Ex: Ethernet – 1500 bytes
- Se pacote for maior que MTU, é quebrado em vários fragmentos. Quando chegam ao destino final são reagrupados, com base em *offsets*.
- Sistemas não tentam processar o pacote até que todos os fragmentos sejam recebidos e reagrupados. Isso cria a possibilidade de ocorrer um overflow na pilha TCP.
- Ex: Ping da Morte – envio de pacotes *ICMP Echo Request* (ping) com tamanho de 65535 bytes. Este valor é maior que o normal e diversos sistemas travavam devido à sobrecarga do buffer.
  - Solução: instalação de *patches*

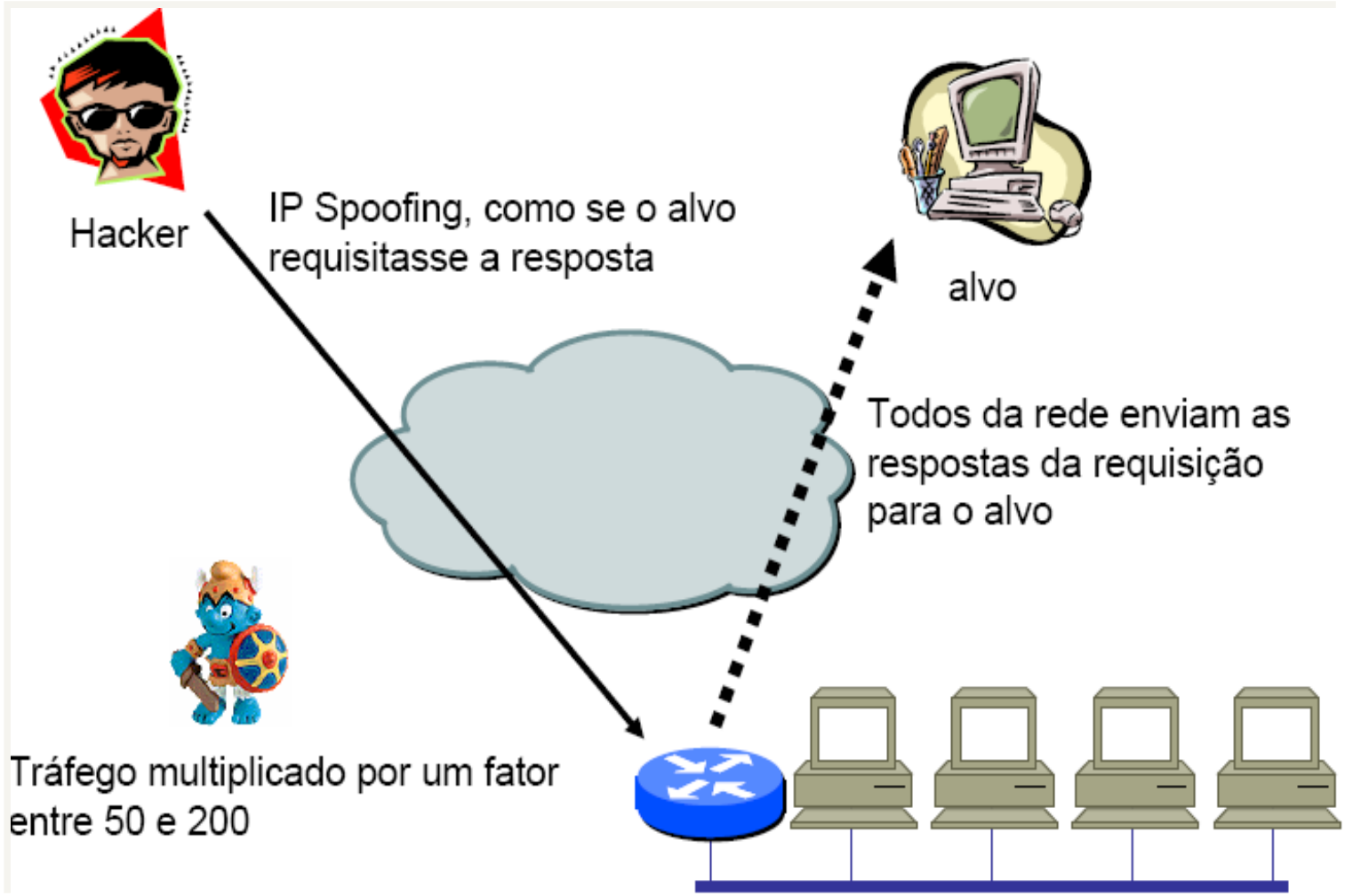
Fragmentação de pacotes de IP

- Ex: *teardrop* – pacotes TCP muito grandes. Vulnerabilidade explorada em versões anteriores do Windows e Linux.
- Firewalls não realizam desfragmentação. Um atacante pode criar um pacote como o primeiro fragmento e especificar uma porta que é permitida pelo firewall, como a porta 80.
- O firewall permite a passagem desse pacote e dos fragmentos seguintes para o host a ser atacado. Um desses fragmentos pode ter o valor de *offset* capaz de sobrescrever a parte inicial do pacote IP que especifica a porta TCP. O atacante modifica a porta inicial de 80 para 23, por exemplo, para conseguir acesso Telnet.
- Fragmentação também é usada como método de *scanning* no *nmap,* tornando sua detecção mais difícil.

DoS – Smurf e fraggle

- Ataque pelo qual um grande número de pacotes ping é enviado para o endereço IP de broadcast da rede, tendo como origem o endereço de IP da vítima (IP *spoofing*). Cada host da rede recebe a requisição e responde ao endereço de origem falsificado.
- Fraggle – utiliza pacotes *UDP echo* em vez de *ICMP echo*
- Ex: WinSmurf

Smurf e fraggle

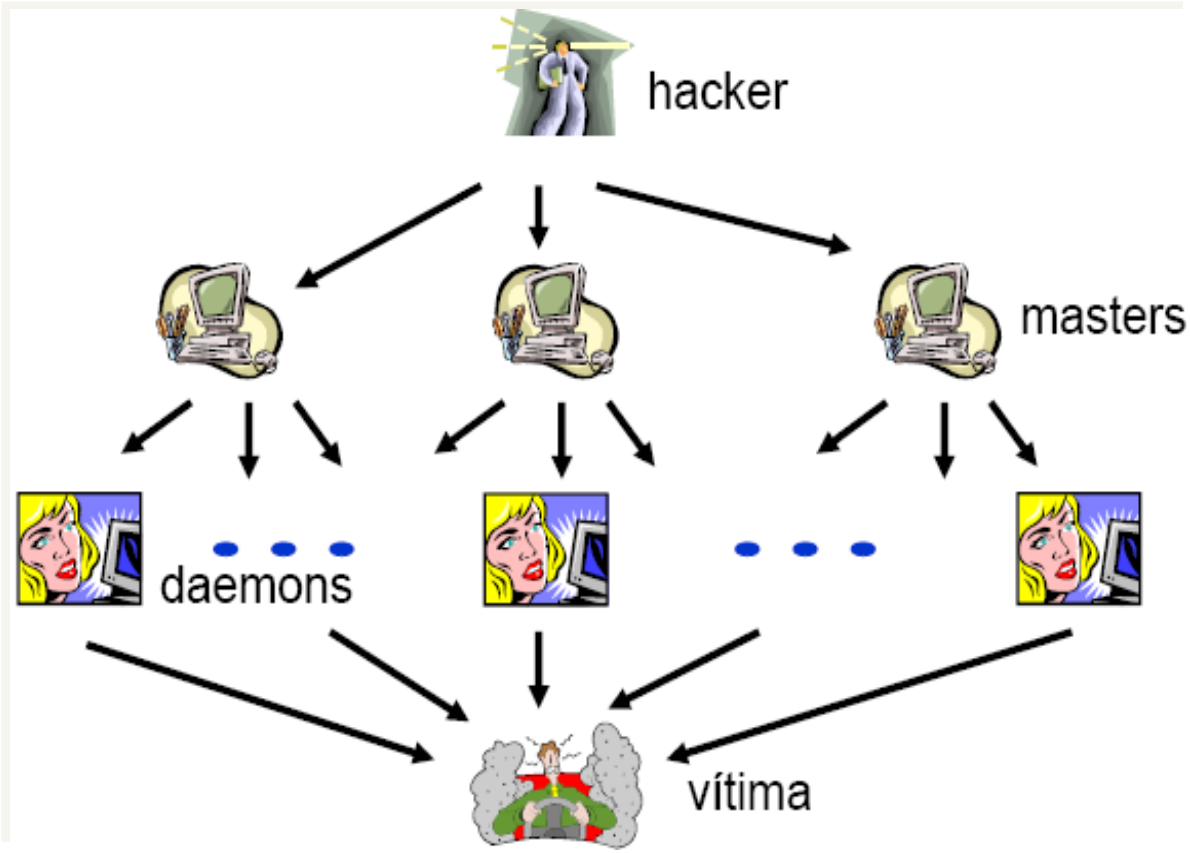


Ataques DoS distribuídos

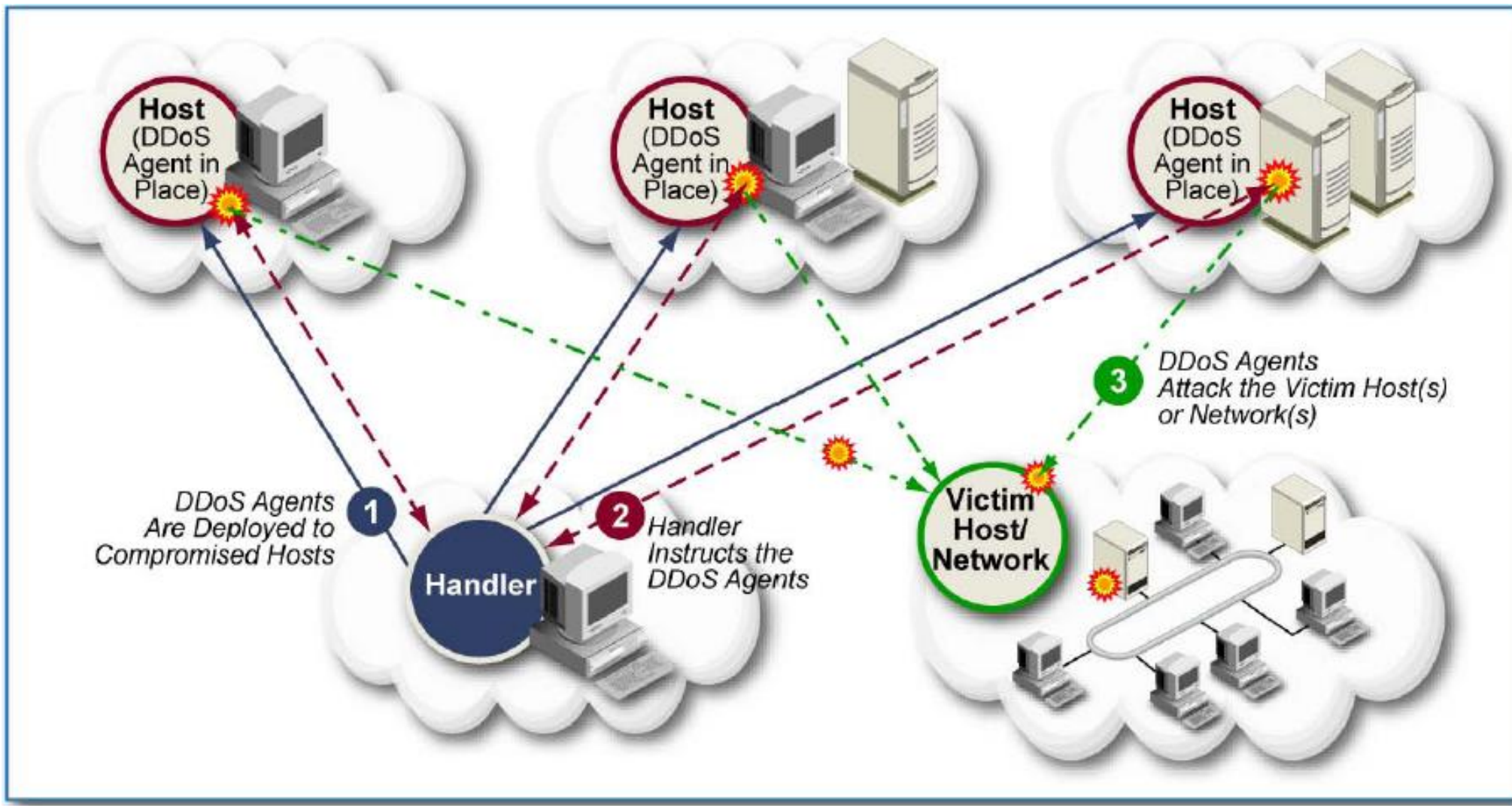
- Um ataque proveniente de uma única máquina geralmente não é capaz de causar dano a uma rede ou servidor
- Nos DoS distribuídos (DDoS – Distributed DoS), o atacando coordena um grande grupo de máquinas para que ataquem simultaneamente um único servidor
- Se um número grande máquinas são usadas, o tráfego gerado pode causar o esgotamento do enlace da rede ou número de conexões do SO
- DDoS são a forma mais perigosa atualmente de ataques DoS
- Ferramentas sofisticadas, utilizam criptografia para o tráfego de controle do hacker!
- Ataque coordenado pelo governo da Indonésia contra o domínio do Timor Leste em 1999.
- Ferrramentas: *trinoo*, TFN (*Tribe Flood Network*), *Stacheldraht*, TFN2K, worm *Code Red*

Ataques DoS distribuídos

- Hacker define alguns sistemas (*masters*) que se comunicam com os *zombies* ou *daemons*.
- Todos são vítimas do hacker.



Ataques DoS distribuídos



Ataque ativo contra o TCP

- Sequestro de conexões
- Redirecionamento de conexões TCP para uma determinada máquina (man-in-the-middle). Todo byte enviado por um host é identificado por um número de sequência que é reconhecido (*acknowledgement*) pelo receptor.
- O ataque tem como base a dessincronização nois dois lados TCP. O atacante cria os números de sequência válidos, colocando-se entre os dois hosts e enviando pacotes válidos para ambos. Ataque mais sofisticado, deve-se interromper a conexão em um estágio inicial e criar nova conexão.

- Prognóstico de número de sequência do TCP
- Possibilita a construção de pacotes TCP de uma conexão, podendo injetar tráfego e fazendo se passar por um outro equipamento.
- Atualmente, alguns sistemas implementam padrões de incremento do número de sequência mais eficiente, que dificulta seu prognóstico e, consequentemente, os ataques.

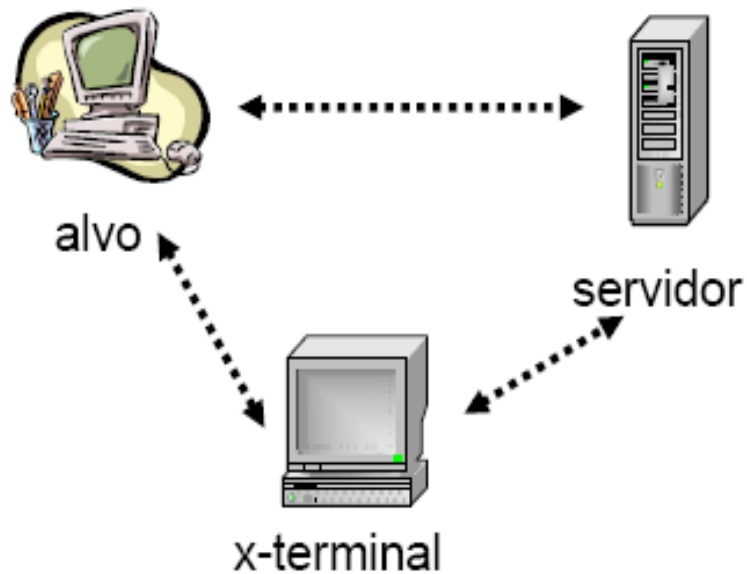
Ataque de Mitnick

- Ataque clássico contra o especialista em segurança Tsutomu Shimomura no natal de 2004.

- Técnicas utilizadas: IP spoofing, sequestro de conexão TCP, negação de serviço, prognóstico de número de sequência.

Ataque de Mitnick

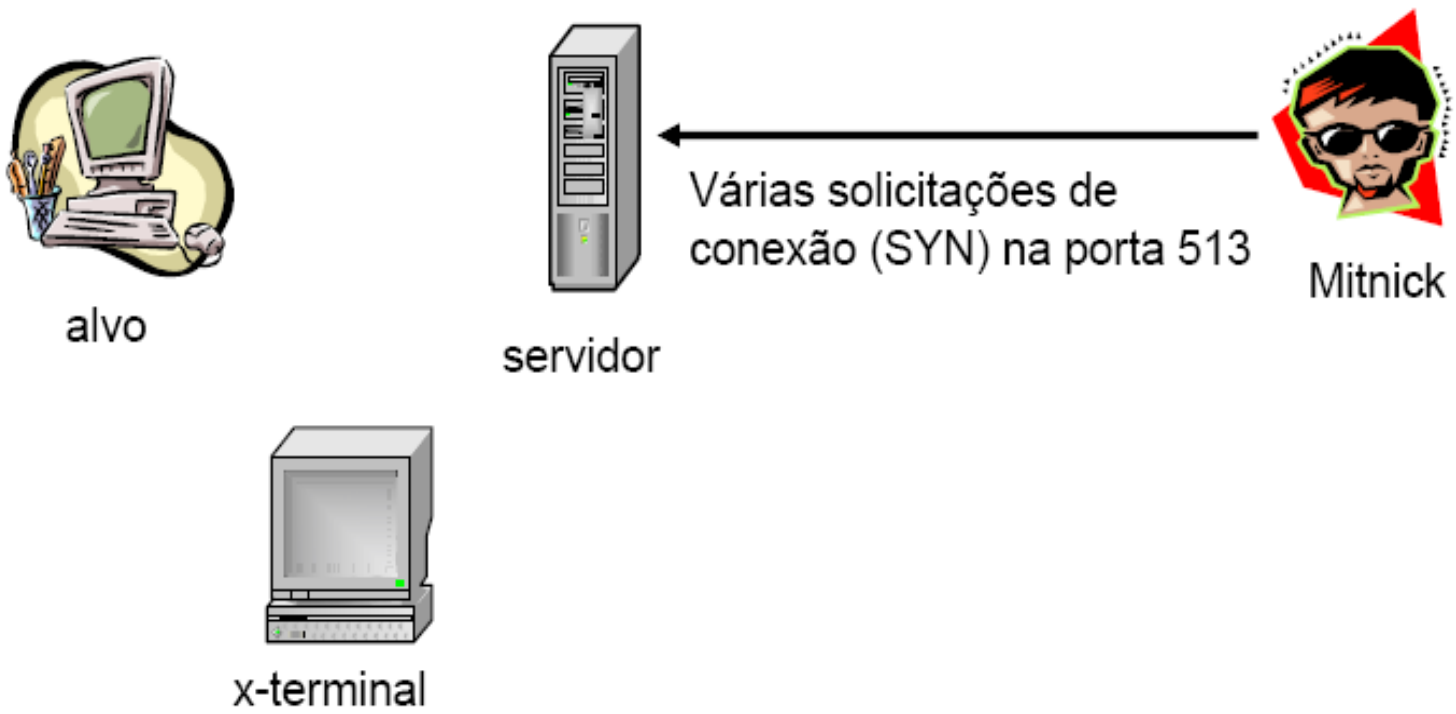
- Passo 1 – verificar a relação de confiança entre os 3 equipamentos:



 # finger -l @target
 # finger -l @server
 # finger -l root@server
 # finger -l @x-terminal
 # showmount -e x-terminal
 # rpcinfo -p x-terminal
 # finger -l root@x-terminal

Ataque de Mitnick

Passo 2 – SYN Flooding no servidor, usando IP Spoofing:



Ataque de Mitnick

- Passo 3 – Estudar o comportamento do número de seqüência do x-terminal:

apollo.it.luc.edu > x-terminal: S 1382726990

x-terminal > apollo.it.luc.edu: S 2021824000 ack 1382726991

apollo.it.luc.edu > x-terminal: R 1382726991

apollo.it.luc.edu > x-terminal: S 1382726991

x-terminal > apollo.it.luc.edu: S 2021952000 ack 1382726992

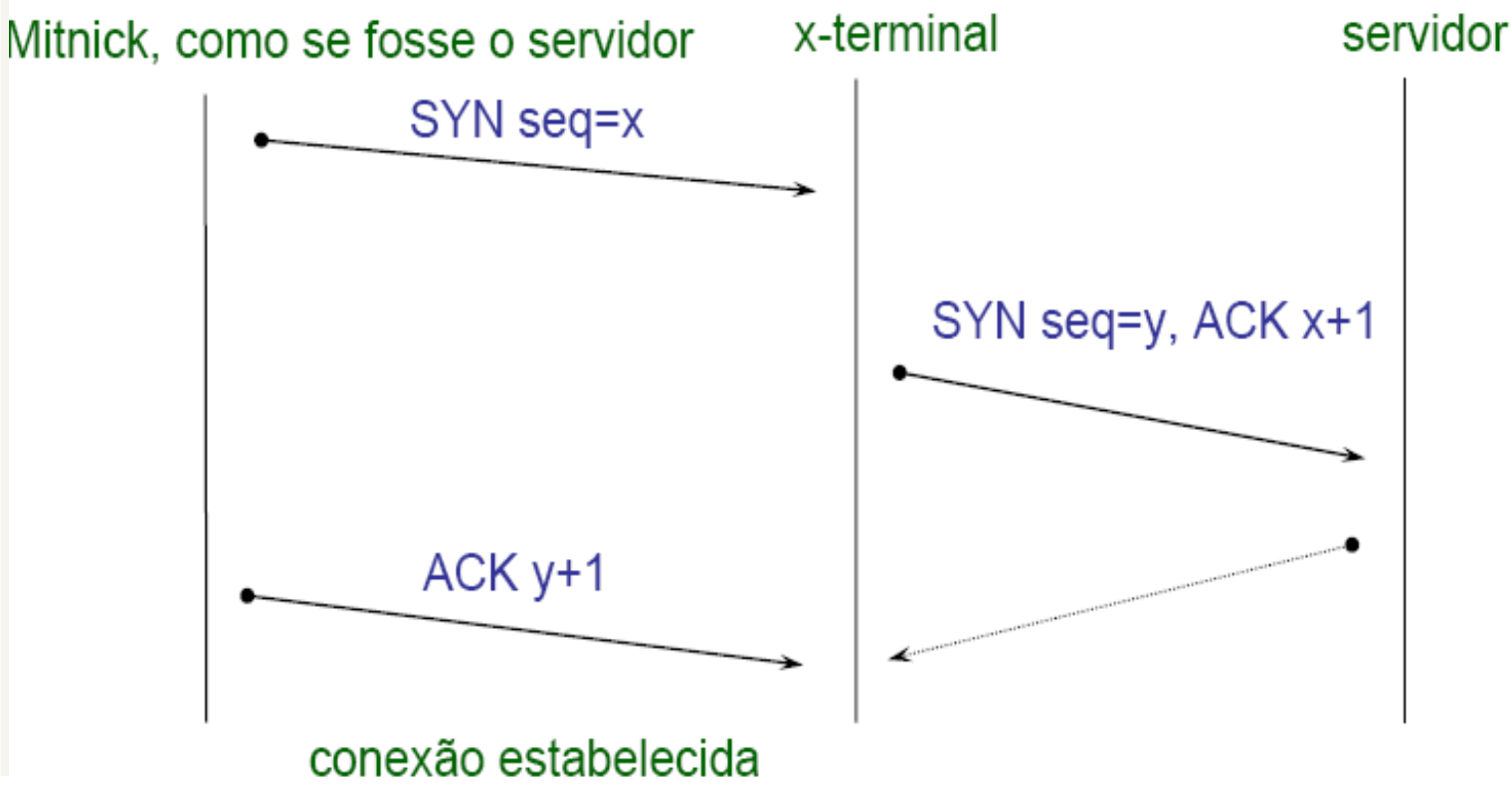
apollo.it.luc.edu > x-terminal: S 1382726992

x-terminal > apollo.it.luc.edu: S 2022080000 ack 1382726993

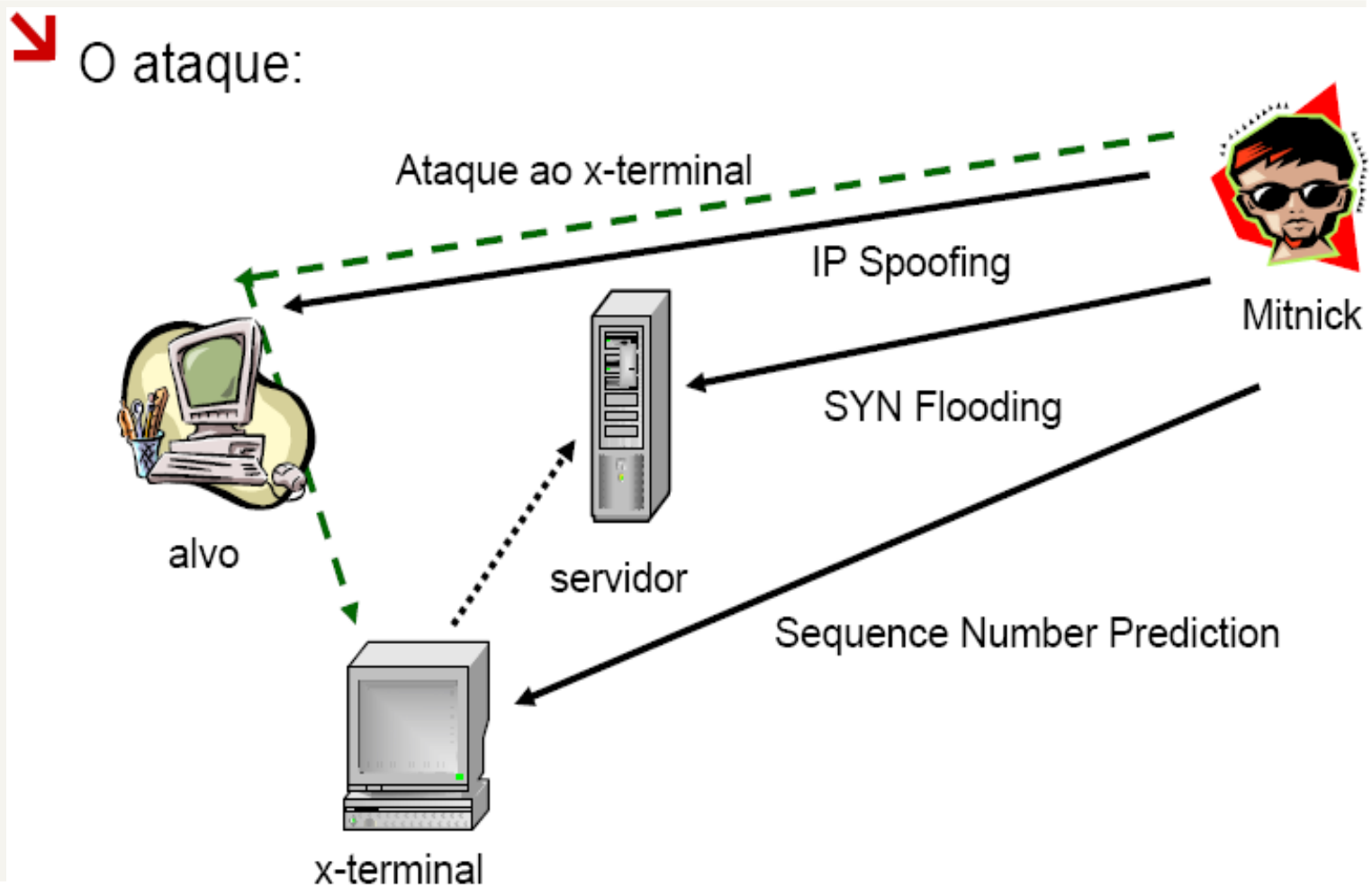
Comportamento do x-terminal: acréscimo de 128.000

Ataque de Mitnick

Passo 4 – Abusar da relação de confiança entre o x-terminal e o servidor:



Ataque de Mitnick



Ataques no nível da aplicação

- Falhas de programação
- Ataques na Web
- Problemas com o SNMP
- Vírus, worms e cavalos de tróia
  - Cada vez mais sofisticados. Ex: Stuxnet (2010)

Erros comuns de programação

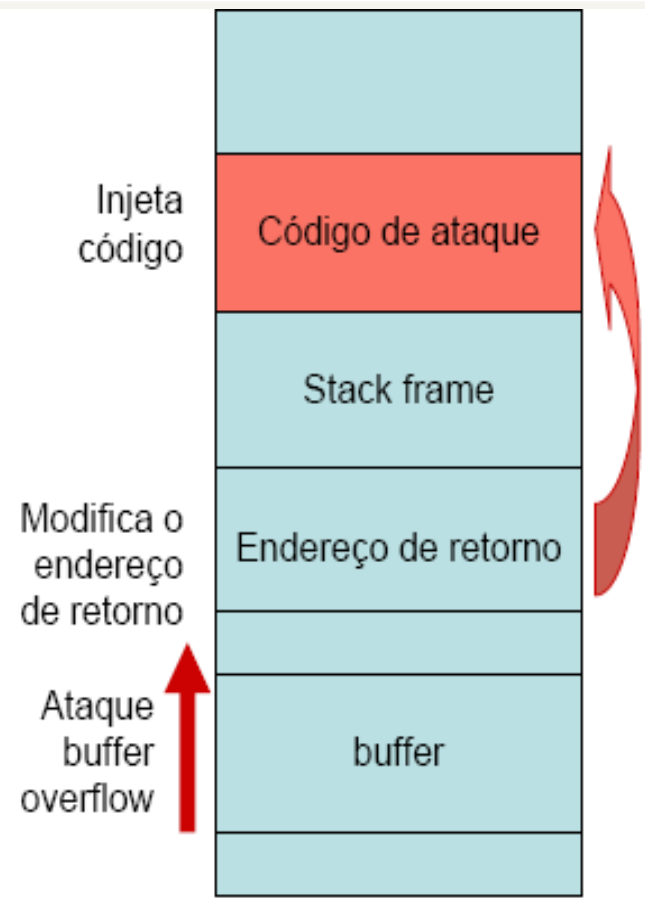
- Estouro de buffer (buffer overflow)
- Validação de entradas
- Condições de corrida

Estouro de Buffer (Buffer overflow)

- Atacante explora bugs de implementação, nos quais o controle do buffer não é feito adequadamente. Envia mais dados do que o buffer pode manipular.
- Possibilidade de execução de comandos arbitrários, perda ou modificação dos dados, perda do controle do fluxo de execução (gpf no Windows)
- Um dos ataques mais comuns e difíceis de ser detectados
  - Site eBay (1999)
  - DLLs do Windows, servidor Sendmail, bibliotecas do SunRPC (2003)

Buffer overflow

- Insere um string grande em uma rotina que não checa os limites
- O string reescreve o endereço de retorno
- O string injeta o código
- A função pula para o código injetado

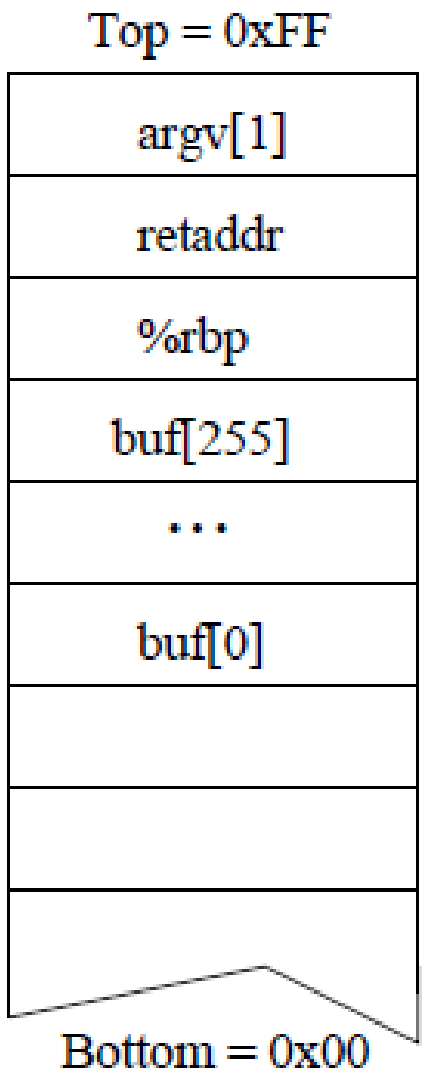


Buffer Overflow

```
void foobar(char *s) {
        char buf[256];
        strcpy(buf, s);
}
int main(int argc, char **argv) {
        foobar(argv[1]);
        return 0;
}
```

O espaço reservado para variável é de 256 bytes. O que acontece quando passamos um argumento muito maior? A função *strcpy(dst, src)* lê o conteúdo do src até encontrar um zero (final de string) copiando o conteúdo para dst. No entanto não é garantido que o destino possa comportar todo os bytes de *src*.

Buffer Overflow



Estado da stack no momento antes da chamada de *strcpy*(). A cópia dos dados é feita escrevendo o primeiro byte de src (s) na primeira posição de dst (buf), sucessivamente até o que *strcpy* encontre um caractere null em src.

Quando a função *foobar*() for chamada, é inserido na pilha o endereço de retorno (*retaddr*). Ao término da função é restaurado o valor do *retaddr* para o registrador.

Se podemos sobrescrever o valor do *retaddr* é possível conseguir o controle do fluxo das instruções. Basta então apontá-lo para um buffer que contenha as instruções que desejamos executar .

```
 1  #include <stdio.h>
 2  #include <string.h>
 3  #include <stdlib.h>
 4
 5  void foobar(char *s) {
 6          char buf[256];
 7          strcpy(buf, s);
 8  }
 9  int main(int argc, char **argv) {
10          foobar(argv[1]);
11          return 0;
12  }
13
14  void lostfunc() {
15          printf("Exploited.\n");
16          exit(0);
17  }
```

Veja que a função *lostfunc* não é referenciada em nenhum momento e não deve ser executada exceto se o fluxo do programa for desviado!

Versão segura do programa

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  void foobar(char *s) {
6          char buf[256];
7          strncpy(buf, s, sizeof(buf)-1);
8          buf[sizeof(buf)-1] = 0;
9  }
10 int main(int argc, char **argv) {
11         foobar(argv[1]);
12         return 0;
13 }
```

A função *strncpy*(dst; src; n) é como a *strcpy* porém faz o *bound checking* necessário para não permitir um overflow. A diferença é que no máximo n bytes de src são copiados para dst, como especificado no terceiro argumento.

A linha 7 copia 255 bytes de s para buf guardando a última posição para o delimitador de final de string (o caractere null ou zero) atribuído em 8.

Exemplo C/C++

Funções Inseguras	Substitutos Seguros
gets(s)	fgets(s, n, stdin)
sprintf(s, fmt, ...)	snprintf(s, n, fmt, ...)
strcat(dst, src)	strncat(dst, src, n)
strcpy(dst, src)	strncpy(dst, src, n)
vsprintf(str, fmt, ap)	vsnprintf(str, n, fmt, ap)

Validação de Entradas

A aplicação que é alimentada pelo usuário deve lidar com entradas inusitadas e maliciosas. Deve-se acreditar que toda entrada de dados é maliciosa! Não se deve confiar em absolutamente nenhuma informação fornecida pelo usuário.

Como não há uma fórmula para solucionar este problema, devemos adotar uma metodologia. Uma abordagem muito aceita entre os programadores envolvidos com segurança é codificar o software de forma que ele aceite um conjunto restrito e precisamente mapeado de possibilidades; rejeitando todas as demais.

Analisar entradas inválidas não faz sentido e é um esforço inútil. Exceto quando se trata de linguagens com gramáticas bem definidas, não se pode prever nem tratar todas as possibilidades.

Injeção SQL

• *SQL Injection* é um dos tipos de vulnerabilidades decorrentes do mal processamento de entradas. Imagine um sistema onde o login e a senha de um usuário são passados através de um formulário web. As informações fornecidas são validadas em um banco de dados, e o acesso é garantido se a consulta acusar o sucesso da autenticação.

Ao submeter a informação o usuário faz com que o servidor execute uma rotina de autenticação.

Quando as informações fornecidas combinam com alguma das linhas do banco de dados, o sistema imprime a mensagem Acesso permitido! :-), caso contrário, imprime Acesso negado. :-(.

Injeção SQL

```php
$login  = $_POST['login'];
$passwd = $_POST['passwd'];

/* autenticacao */
if( $login != null ) {
        mysql_connect(localhost, "root", "31337");
        @mysql_select_db("foobar") or die("erro do banco");
        $qry = "select * from users
                where login='$login' and password='$passwd'";
        $res = mysql_query($qry);
        if( $res != null && mysql_numrows($res) >= 1 )
                echo "Acesso permitido! :-)";
        else
                echo "Acesso negado. :-(";
} else {
/* formulario html */
```

Para Maria se autenticar no sistema: abre seu navegador, carrega o formulário, e entra com o nome e senha corretos. As variáveis $*login e $passwd* serão preenchidas com os valores passados pelo comando *POST*. A autenticação é feita através do comando *SQL*, que busca uma entrada da tabela *users* que contenha ao mesmo tempo o login e a senha fornecidos pelo usuário. No caso da Maria, o comando *SQL processado será select * from users* where login='maria' and password='maria123'.

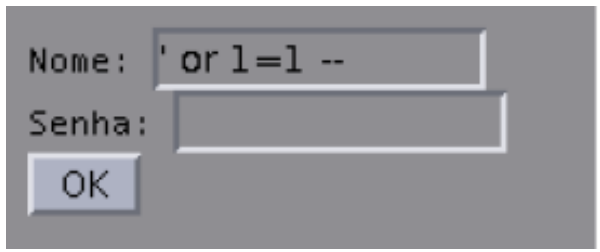
Injeção SQL

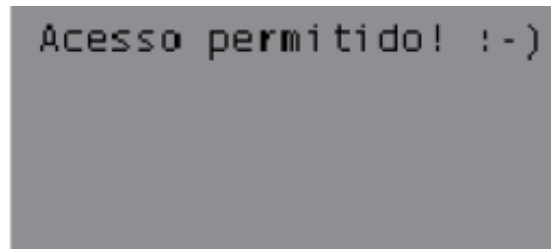
• Uma falha é susceptível a SQL injection quando é possível escolher um input tal que a query resultante contenha um comando arbitrário.
Vejamos novamente a codificação da query *SQL:*
$qry= "select * from users where login='$login' and password='$passwd'"
O que acontece quando a variável $*login* contém um dos caracteres de controle da linguagem *SQL?* Por exemplo, se o token de comentário (- -) fosse atribuído à variável $*login*, o que aconteceria?
Ocorre que a variável $*login* é concatenada com as outras partes do comando sem haver nenhum tratamento prévio. As informações fornecidas pelo usuário são substituídas na cadeia de caracteres $*qry* sem que seja assegurada a consistência do comando resultante.
Portanto, os campos Nome e Senha do formulário são extensões do comando *SQL!* Os valores não são devidamente resguardados como um dos parâmetros da cláusula *WHERE. Se o campo Nome tem valor "'--" (*como no exemplo acima) e Senha é deixado em branco, temos o seguinte resultado:

```
$qry= "select * from users where login=''-- ' and password = ''"
```

Injeção SQL

• Note o token de comentário “- - “ *(terminado por um caractere branco).* Toda a linha à direita dele é ignorado, resultando em:
*$qry= "select * from users where login= ‘ ‘ "*
Esta modificação ainda não é o suficiente para burlar a autenticação do programa, já que não há entradas com o campo *login* nulo na tabela *users.*
Partindo do princípio que não conhecemos nenhum usuário que nos permitiria entrar algo como: “*maria’- - “ podemos concatenar uma condição OU com uma sentença verdadeira, para obter como resposta todas as linhas da* tabela *users.* A query resultante da entrada da cadeia *“’or 1 = 1 - - “* seria*:*
*$qry= "select * from users where login=‘ ‘ or 1=1"*
Este comando fará com que todas as linhas da tabela sejam retornadas, já que 1 = 1 é sempre verdadeiro.
Assim, o mecanismo de autenticação foi burlado. O atacante foi capaz de violar o controle de acesso sem nem mesmo precisar de um usuário forjado.





Injeção SQL

• A maneira correta de implementar a construção da query é validando a entrada antes de ser concatenada ao comando, assegurando que não há comandos injetados dentro do valor recebido.

Uma das maneiras de se proteger do ataque é processar a entrada do usuário com a função *mysql_real_escape_string(),* que adiciona escapes à todos os caracteres reconhecidos como ameaças para um comando na linguagem *SQL* que são, na verdade, os próprios caracteres especiais ou tokens da linguagem. Poderíamos usar a função para adicionar escapes no conteúdo da variável $*login*, desta forma:

```
$login = mysql_real_escape_string($login);
$qry = "select * from users
        where login='$login' and password='$passwd'";
```

A query resultante seria esta:

```
select * from users where login='\' or 1=1 -- ' and password=''
```

Note que o segundo acento é precedido de um escape, e isto faz com que todo o argumento, incluindo a seqüência de comentário "- - ", seja considerado parte do campo *login na busca.* O resultado é um sistema mais seguro, já que este novo mecanismo é capaz de evitar a injeção do comando.

Cross-Site Scripting (XSS)

- Aproveita o mal processamento dos dados de entrada na construção de páginas *HTML*. O *XSS* permite burlar mecanismos de controle de acesso para obter informações confidenciais do usuário, ou executar scripts em sua máquina.

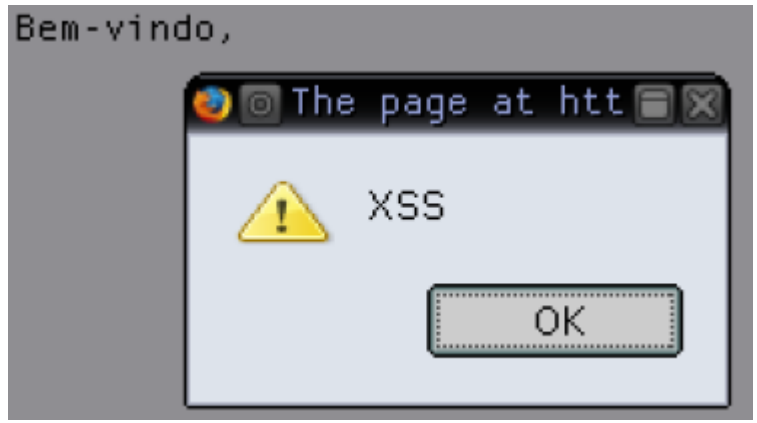
```
<html>
        <head><title>Exemplo de XSS</title></head>
        <body>
<?php
        $name = $_POST['name'];

        if( $name != null ) {
                echo "Bem-vindo, ", $name;
        } else {
?>
        <div>
        <form action="form.php" method="post" name="f">
        Nome:
        <input name="name" type="text" id="name" size="32">
        <br/>
        <input name="submit" type="submit" value="OK">
        </form>
        </div>
<?php
        }
?>
        </body>
</html>
```

Cross-Site Scripting (XSS)

• O script constrói e imprime uma mensagem simples para o usuário: Bem-vindo, $nome. Note que, novamente, o nome do usuário é concatenado à uma página HTML sem nenhuma tratamento especial para a linguagem.

O que acontece quando inserimos código no formulário?

Cross-Site Scripting (XSS)



• O script foi executado, fazendo o navegador mostrar uma janela com a mensagem XSS. O conteúdo do campo no formulário foi injetado na página montada para exibição.

Os navegadores mais populares usam um mecanismo de controle de acesso que isola a execução do script por contextos definidos pela origem (servidor) e protocolo utilizado na comunicação.

Também é possível fazer *phishing* em sites com vulnerabilidades *XSS* em alguns casos específicos, já que, como há a injeção de código *HTML*, pode-se criar uma página forjando formulários de senha, etc.

Estas vulnerabilidades são encontradas de muitas formas nas aplicações reais, e existem diferentes técnicas para explorar cada uma delas. No entanto, a essência do ataque é a mesma.

Condições de Corrida

• As condições de corrida ocorrem em ambientes que suportam multiprogramação.

Este problema acontece quando dois ou mais processos acessam os mesmos recursos "simultaneamente".

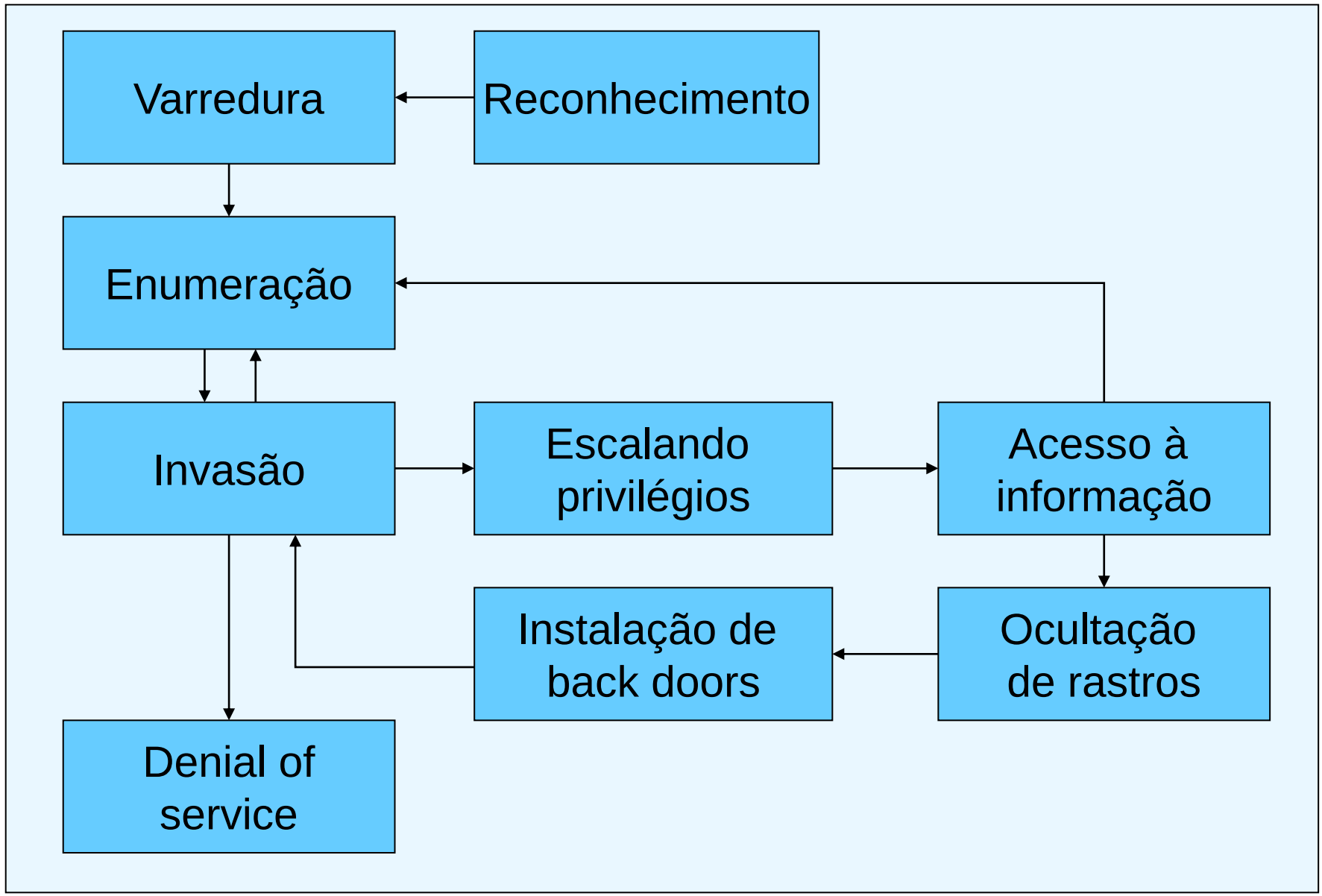
Um recurso pode ser modificado, intencionalmente ou não, por um processo, e será requisitado por um segundo, fazendo que este se comporte de maneira não esperada.

Condições de Corrida

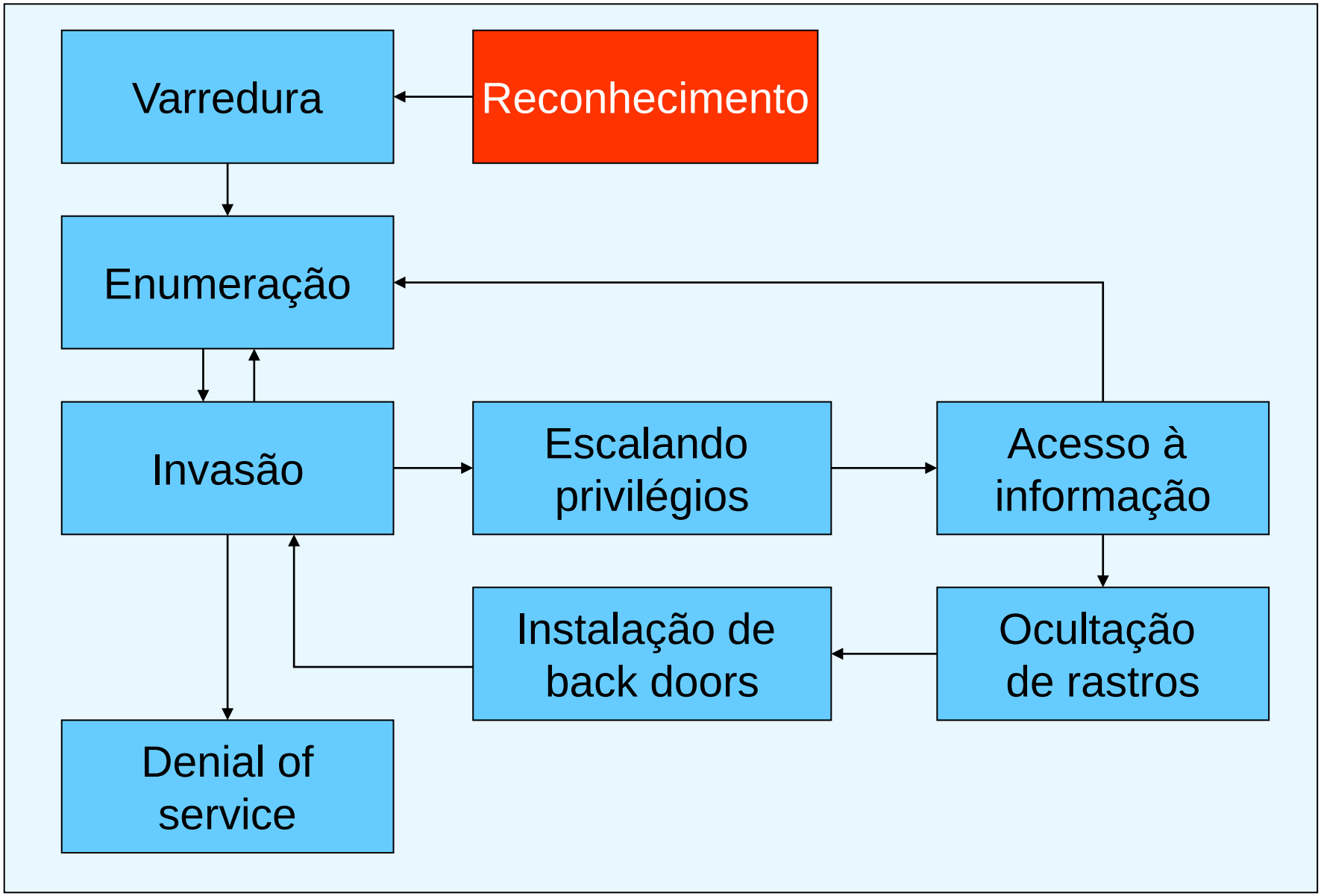
• Semáforos e locks são mecanismos que previnem acesso concorrente em um objeto por diferentes processos.

Uma operação que não pode ser interrompida em relação a um objeto é chamada atômica.

Anatomia de um ataque



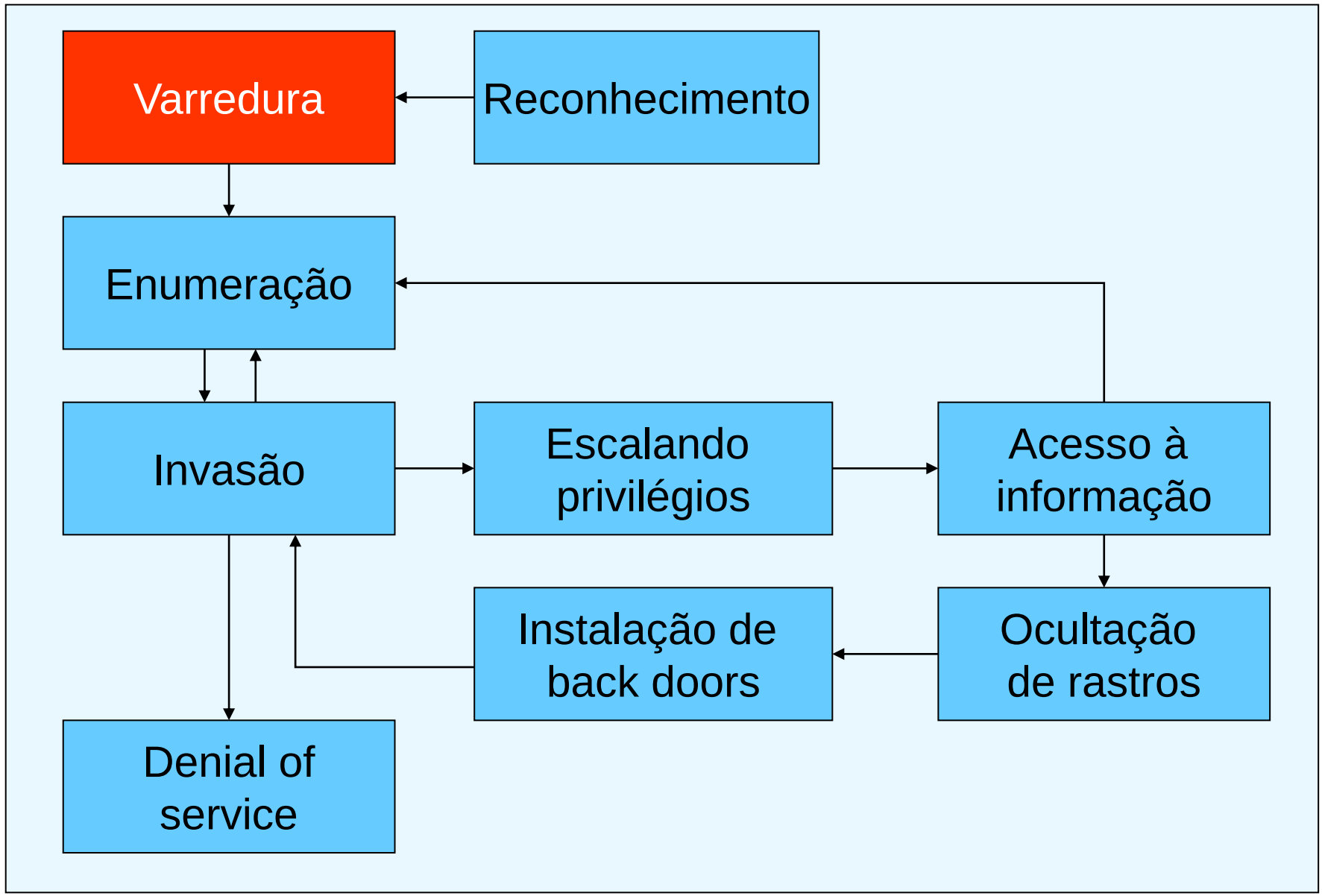
Anatomia de um ataque



1. Footprinting (reconhecimento)

- Informações básicas podem indicar a postura e a política de segurança da empresa
- Coleta de informações essenciais para o ataque
  - Nomes de máquinas, nomes de login, faixas de IP, nomes de domínios, protocolos, sistemas de detecção de intrusão
- São usadas ferramentas comuns da rede
- Engenharia Social
  - Qual o e-mail de fulano?
  - Aqui é Cicrano. Poderia mudar minha senha?
  - Qual o número IP do servidor SSH? e o DNS?

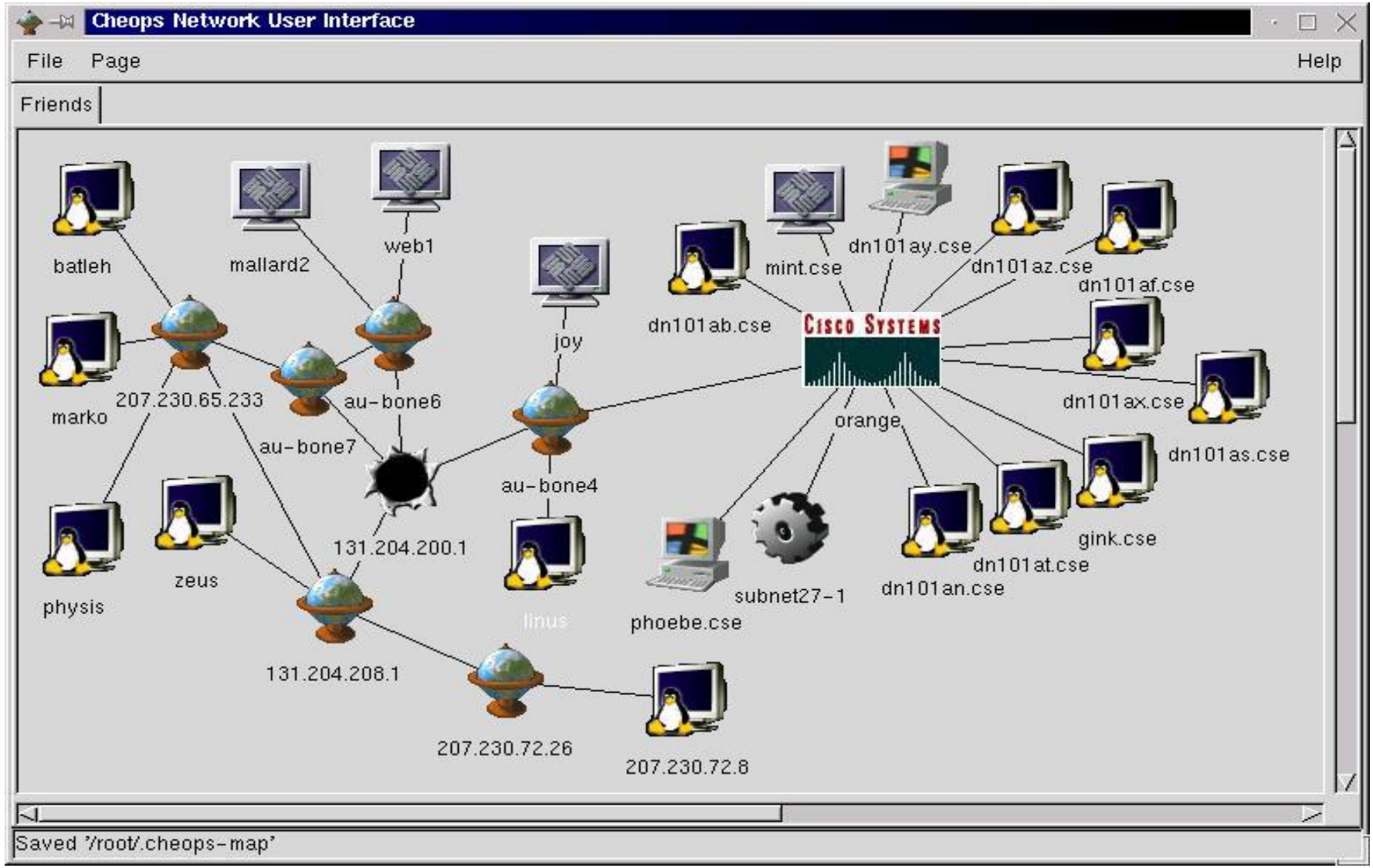
Anatomia de um ataque



2. Scanning (varredura ou mapeamento)

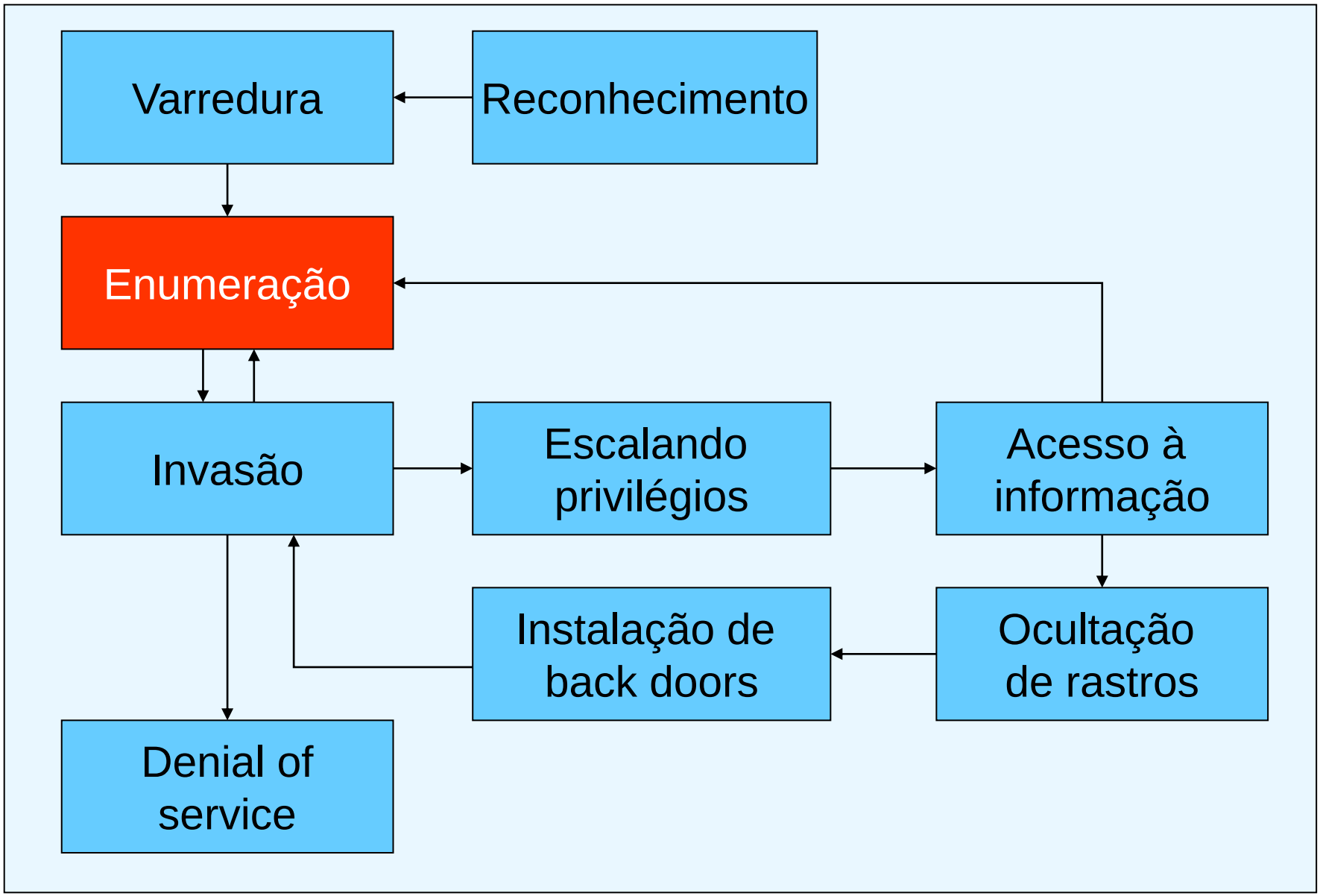
- De posse das informações coletadas, determinar
  - Quais sistemas estão ativos e alcançáveis
  - Portas de entrada ativas em cada sistema
- Ferramentas
  - Unix: nmap
  - Windows: netscantools pro ($), superscan (free)
  - system banners, informações via SNMP
- Descoberta da Topologia
  - Automated discovery tools: cheops, ntop, …
  - Comandos usuais: ping, traceroute, nslookup
- Detecção de Sistema Operacional
  - Técnicas de fingerprint (nmap)
- Busca de senhas contidas em pacotes (sniffing)
- Muitas das ferramentas são as mesmas usadas para gerenciamento e administração da rede

Mapeamento de rede



Tela do Cheops (http://cheops-ng.sourceforge.net)

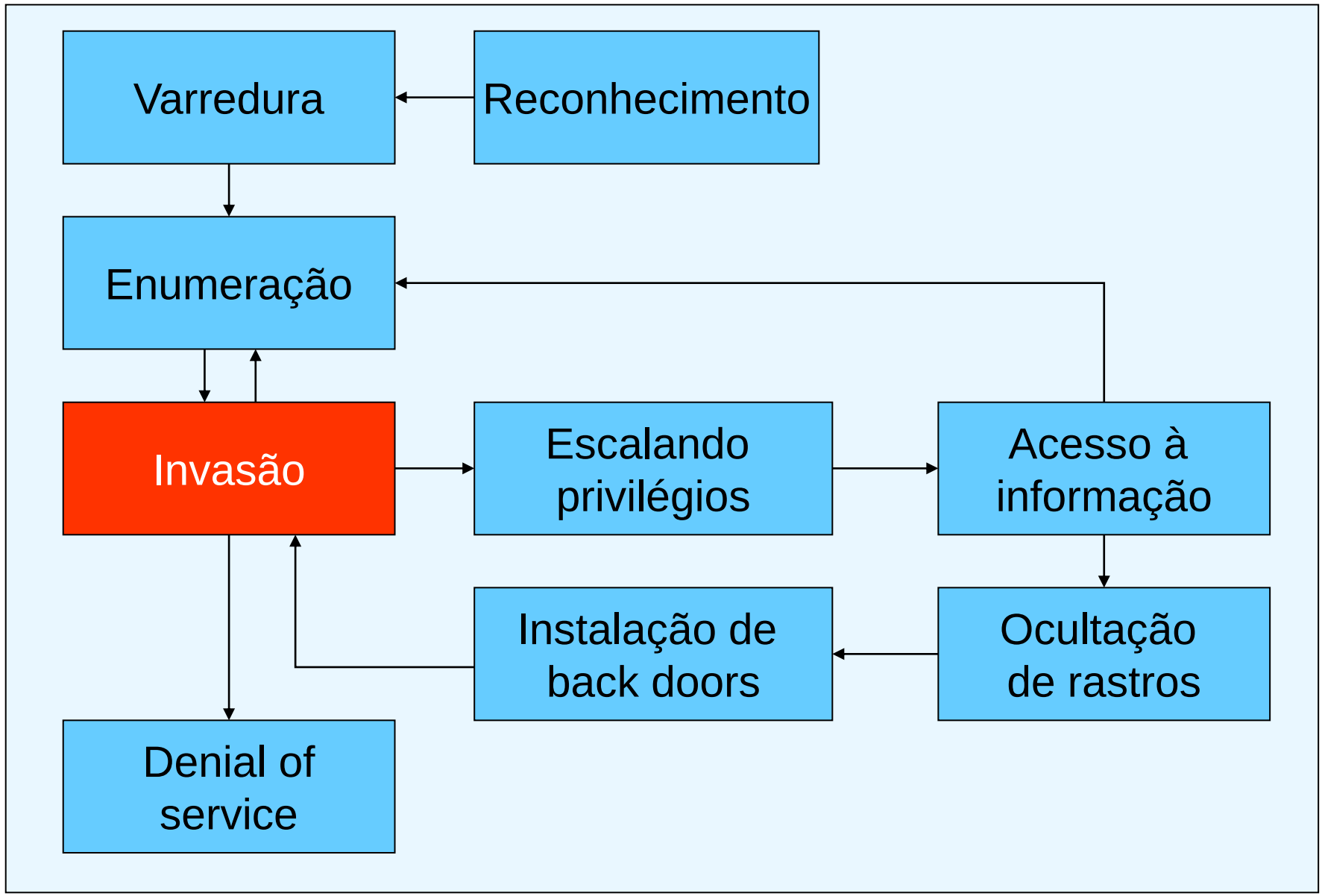
Anatomia de um ataque



3. Enumeration (enumeração)

- Coleta de dados intrusiva
  - Consultas diretas ao sistema
  - Está conectado ao sistema e pode ser notado
- Identificação de logins válidos
- Banners identificam versões de HTTP, FTP servers
  - Exemplo: Comando: **telnet www.host.com 80**
    Resposta: **HTTP/1.0 Bad Request**
    server Netscape Commerce/1.12
- Identificação de recursos da rede
  - Compartilhamentos (windows) - Comandos net view, nbstat
  - Exported filesystems (unix) - Comando showmount
- Identificação de Vulnerabilidades comuns
  - Nessus, SAINT, SATAN, SARA, TARA, ...
- Identificação de permissões
  - find /home/joao -perm 0004
    (arquivos de joao que podem ser lidos por qualquer um...)

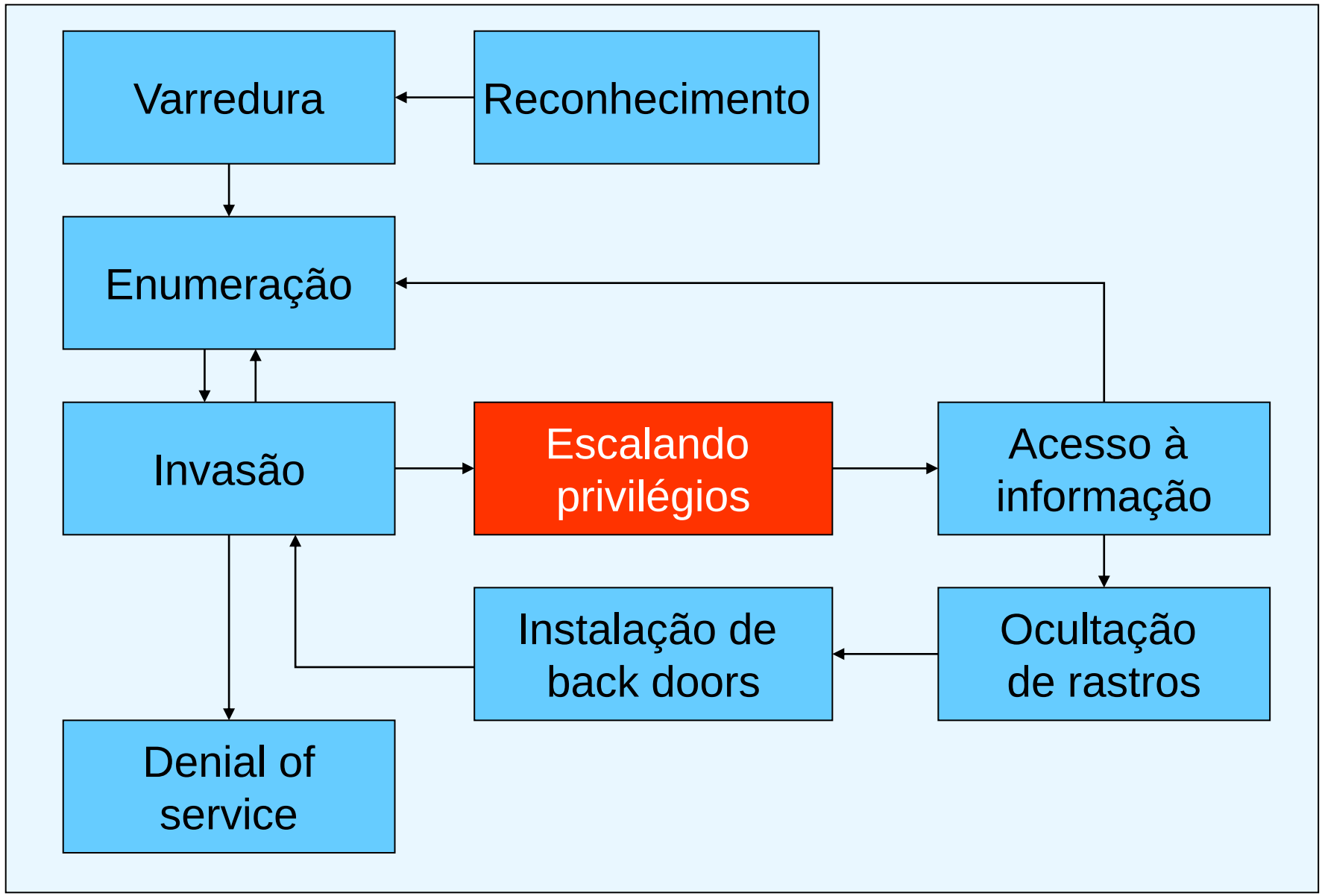
Anatomia de um ataque



4. Ganhando acesso (invasão)

- Informações coletadas norteiam uma estratégia de ataque
- Invasores mantêm um “Banco de dados” de vulnerabilidades
  - Bugs de cada SO, kernel, serviço, aplicativo – por versão
  - Tentam encontrar sistemas com falhas conhecidas
- Busca privilégio de usuário comum (pelo menos)
- Técnicas
  - Password sniffing, password crackers, password guessing
  - Session hijacking (sequestro de sessão)
  - Ferramentas para bugs conhecidos (buffer overflow)
- Hackers também constróem as suas próprias ferramentas, personalizadas

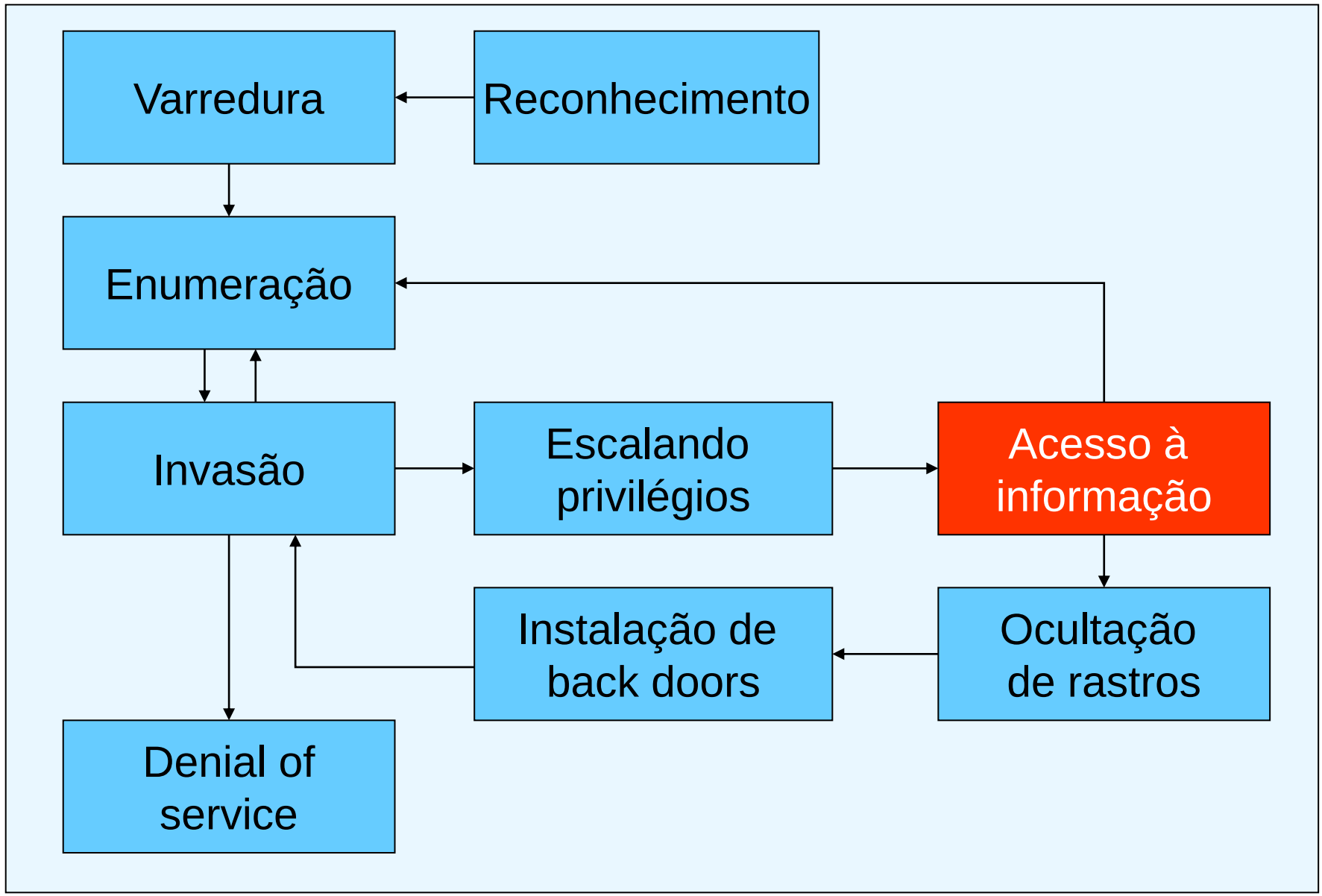
Anatomia de um ataque



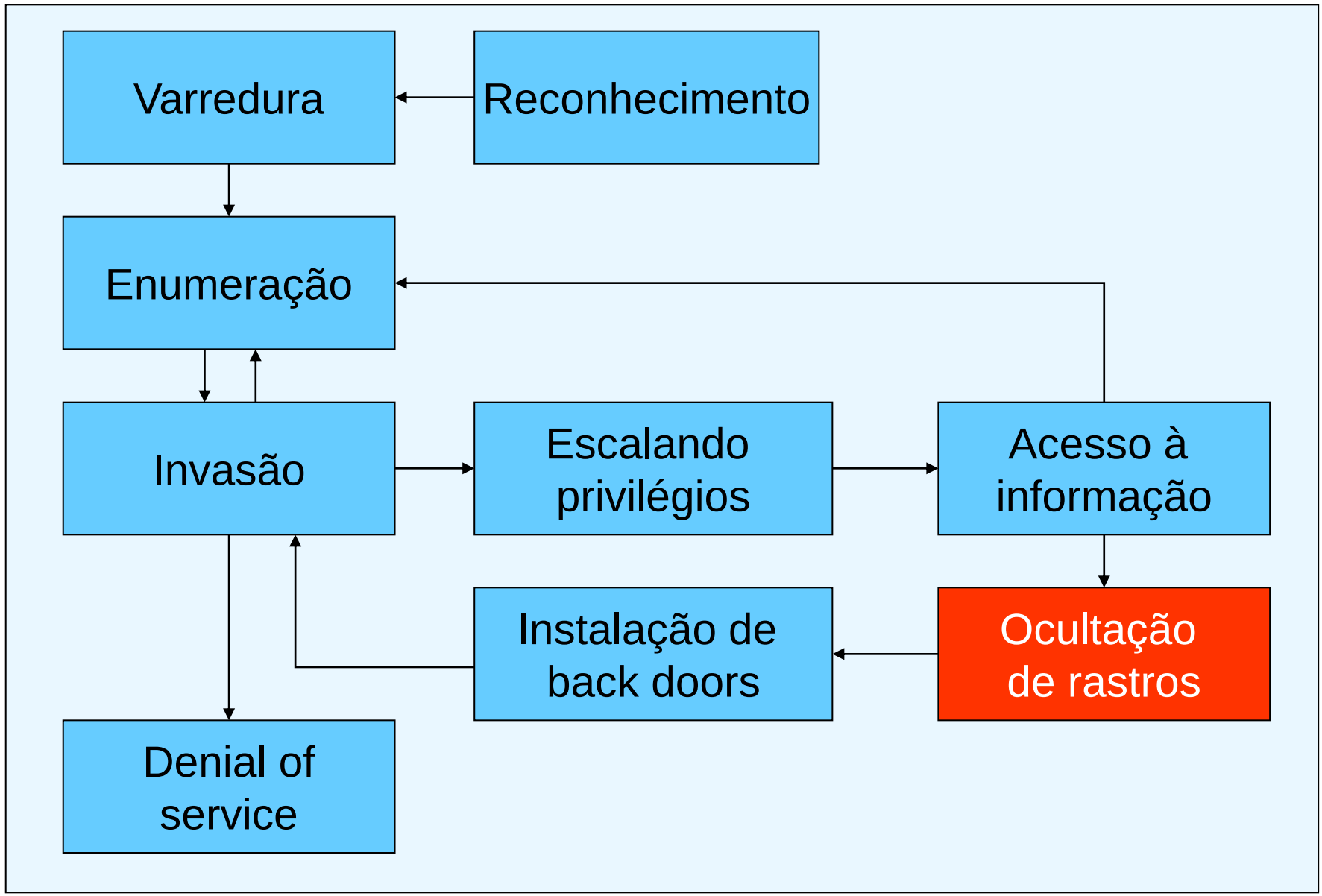
5. Escalada de privilégios

- Uma vez com acesso comum, busca acesso completo ao sistema (administrator, root)
- Ferramentas específicas para bugs conhecidos
  - "Exploits"
- Técnicas
  - Password sniffing, password crackers, password guessing
  - Session hijacking (sequestro de sessão)
  - Replay attacks
  - Buffer overflow
  - Trojans

Anatomia de um ataque



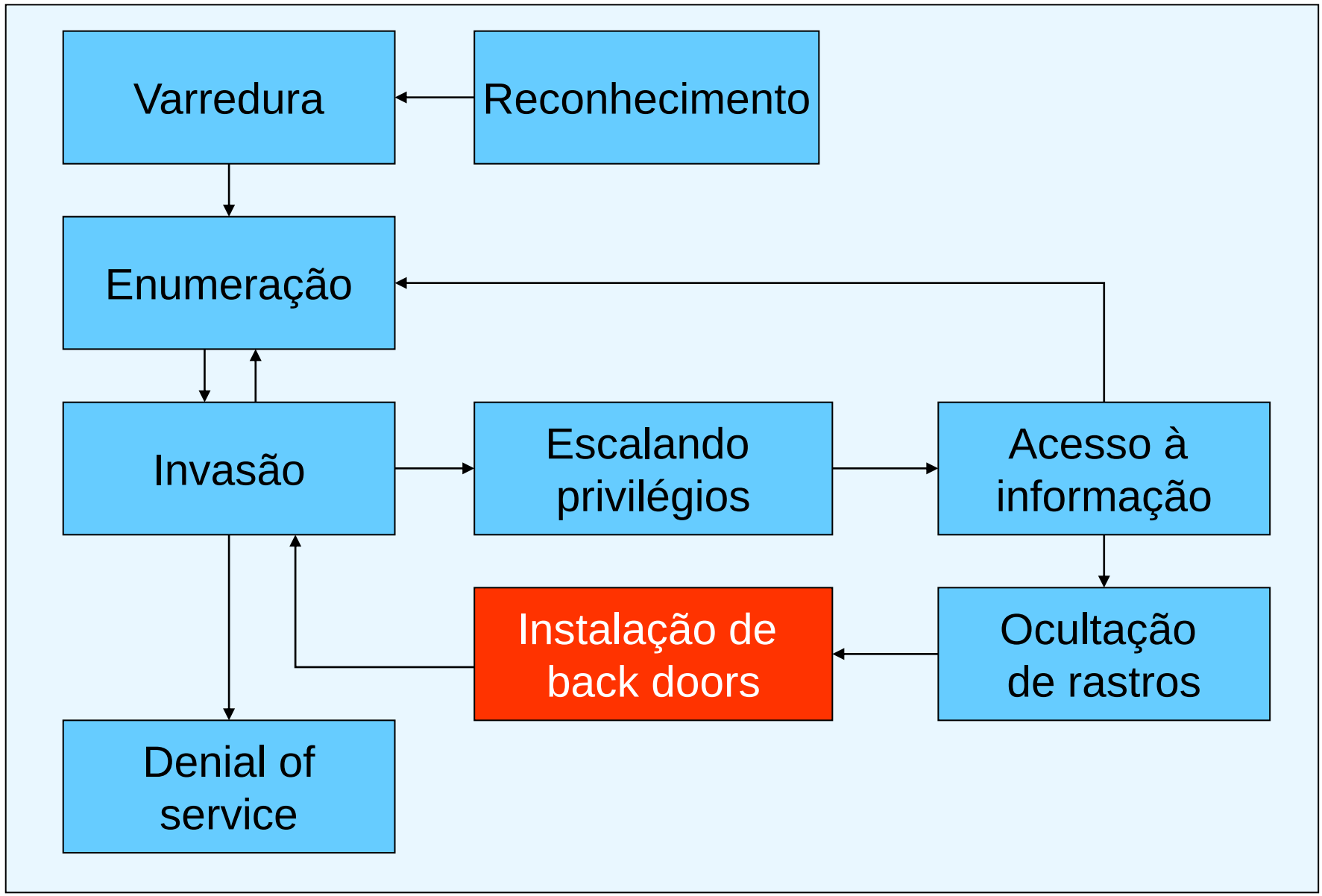
Anatomia de um ataque



7. Ocultação de rastros

- Invasor usa tenta evitar detecção da presença
- Usa ferramentas do sistema para desabilitar auditoria
  - Windows: c:\ auditpol /disable
    (atividade do invasor)
    c:\ auditpol /enable
- Toma cuidados para não deixar “buracos” nos logs
  - excessivo tempo de inatividade vai denuciar um ataque
- Existem ferramentas para remoção seletiva do Event Log
- Esconde arquivos “plantados” (back doors)
  - Exemplo 1: attrib +h file
  - Exemplo 2: cp backdoor.exe arquivo.ext:stream (esconde)
    cp arquivo.ext:stream backdoor.exe (recupera)
  - (conceito de file stream foi introduzido pelo Windows2000)

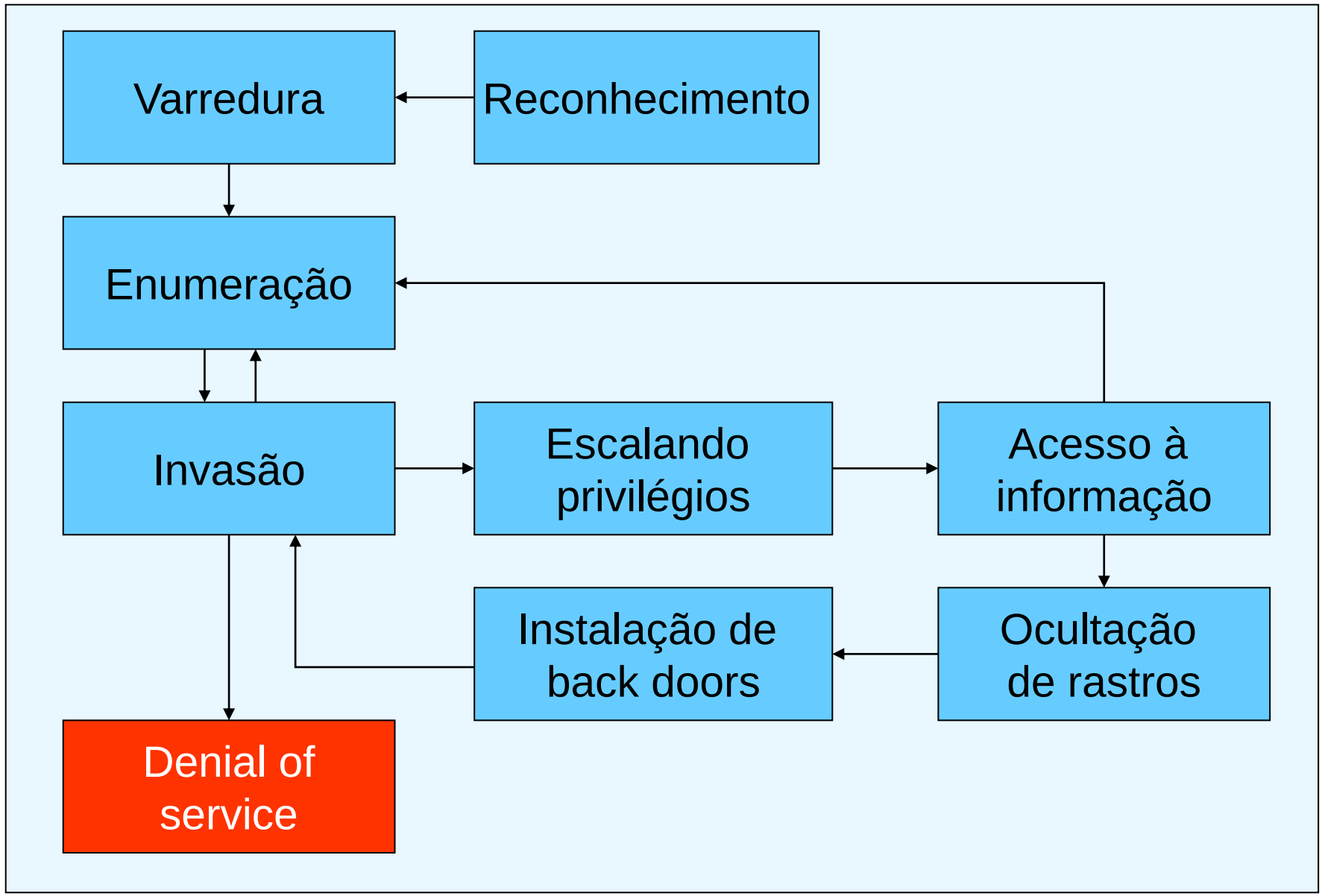
Anatomia de um ataque



8. Instalação de Back doors (porta dos fundos)

- Objetivo é a manutenção do acesso
  - Rootkits – ferramentas ativas, mas escondidas
  - Trojan horses – programas falsificados
  - Back doors – acesso/controle remoto sem autenticação
- Próxima invasão será mais fácil
- Trojans podem mandar informação para invasor
  - Captura teclado (ex: no login)
  - Manda um e-mail com a senha
- Rootkits se confundem com o sistema
  - Comandos modificados para não revelar o invasor
  - Exemplo: (unix) ps, who, mount e até partes do kernel!
- Back doors
  - Sistemas cliente/servidor
  - Cliente na máquina invasora controlando Servidor na máquina remota
  - Não aparecem na "Task List" do Windows NT/2k

Anatomia de um ataque



9. Denial of Service (negação de serviço)

- Ataques com objetivo de bloquear serviços, através de:
  - Consumo de banda de rede
  - Esgotamento de recursos
  - Exploração de falhas de programação (ex: ping da morte)
  - Sabotagem de Roteamento
    - RIP, BGP têm fraca autenticação
    - Pacotes não conseguem sair da rede
  - Sabotagem no DNS
    - DNS responde errado causando desvio de acesso
    - banco.com → hacker.com
- Se a amazon parar 30 minutos, qual o prejuízo?
- E se não parar mas ficar 20% mais lento ?
- DDoS →Distributed Denial of Service
  - Ataques coordenados de múltiplas fontes

Prevenindo Ataques

Fase	Ação preventiva
Reconhecimento	➔Política de senhas, Educação do usuário ➔Nunca anotar passwords, Segurança Física, Biometria
Mapeamento	➔Desabilitar serviços desnecessários ➔Usar senhas criptografadas e abolir Telnet, FTP, POP, rsh ➔Adotar SSL: SSH, sFTP, S/MIME, Webmail, scp, https (mesmo na LAN) ➔Kerberos ➔VPN, IPSec
Enumeração	➔Usar NAT para esconder números IP internos ➔TCP Wrappers, Redirecionamento ➔Honeypots ➔Restringir relações de confiança entre hosts
Invasão	➔Atualização do sistema: patches, service packs, hot fixes ➔Firewalls + Sistema de Detecção de Intrusão ➔Testes de penetração (Nessus, SATAN) ➔Backup de arquivos de configuração (fast recovery)
Escalada de privilégios	➔Permissões bem configuradas, educação do usuário
Ocultação de rastros	➔Cópia dos Logs em outra máquina em tempo real (redundância) ➔Ferramentas de integridade (checksums)
Instalação de Backdoors	➔Scripts de verificação, Ferramentas de integridade (checksums)