Parte IV

Ataques às Redes I

Ataques às Redes I

Capítulo 5: Anatomia de Ataques

Capítulo 6: Ataques de Força Bruta

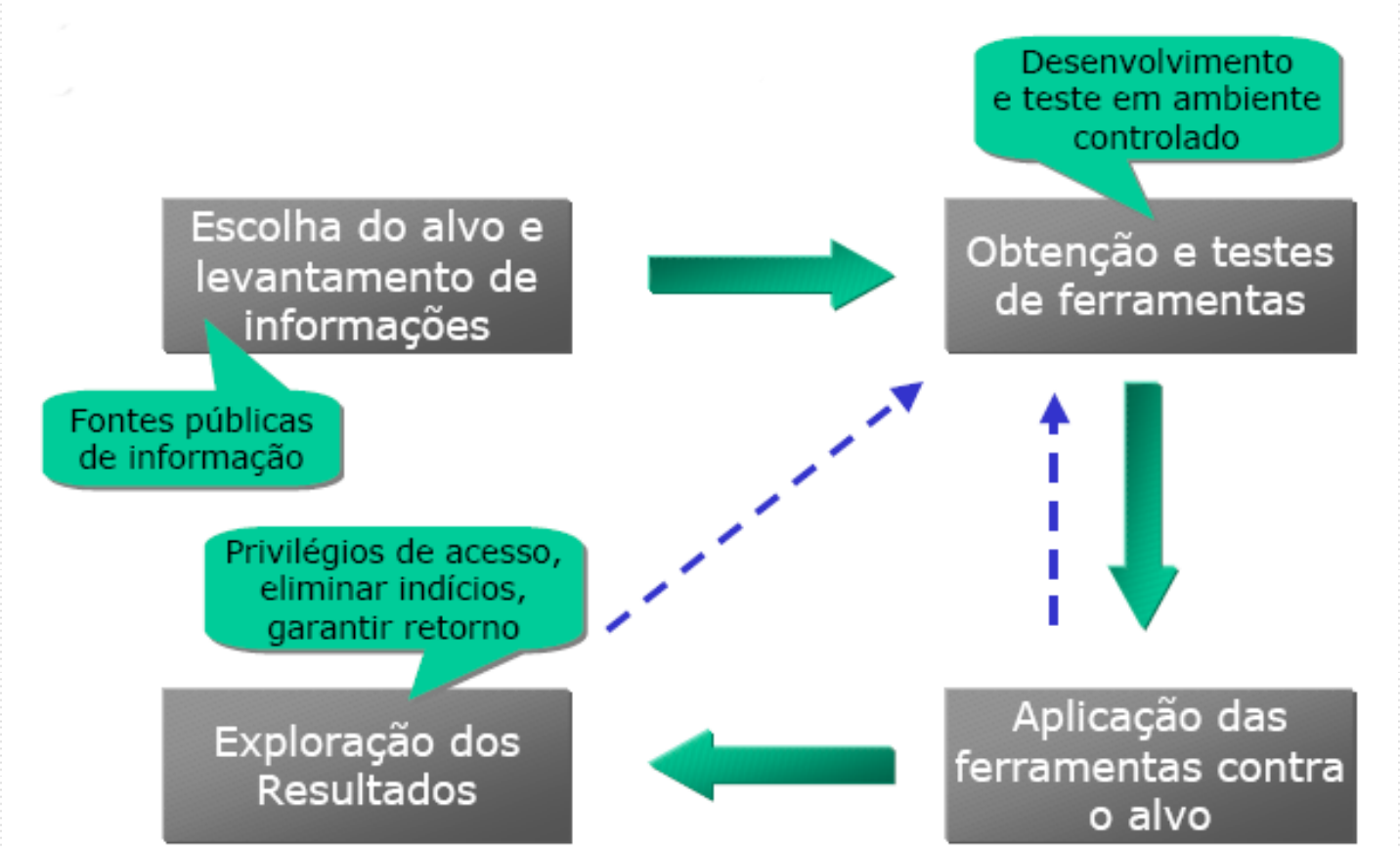
Capítulo 5

Anatomia de Ataques

A construção de um ataque

- O primeiro passo para entender um ataque é entender sua anatomia, ou seja a forma como se pode construir um ataque.

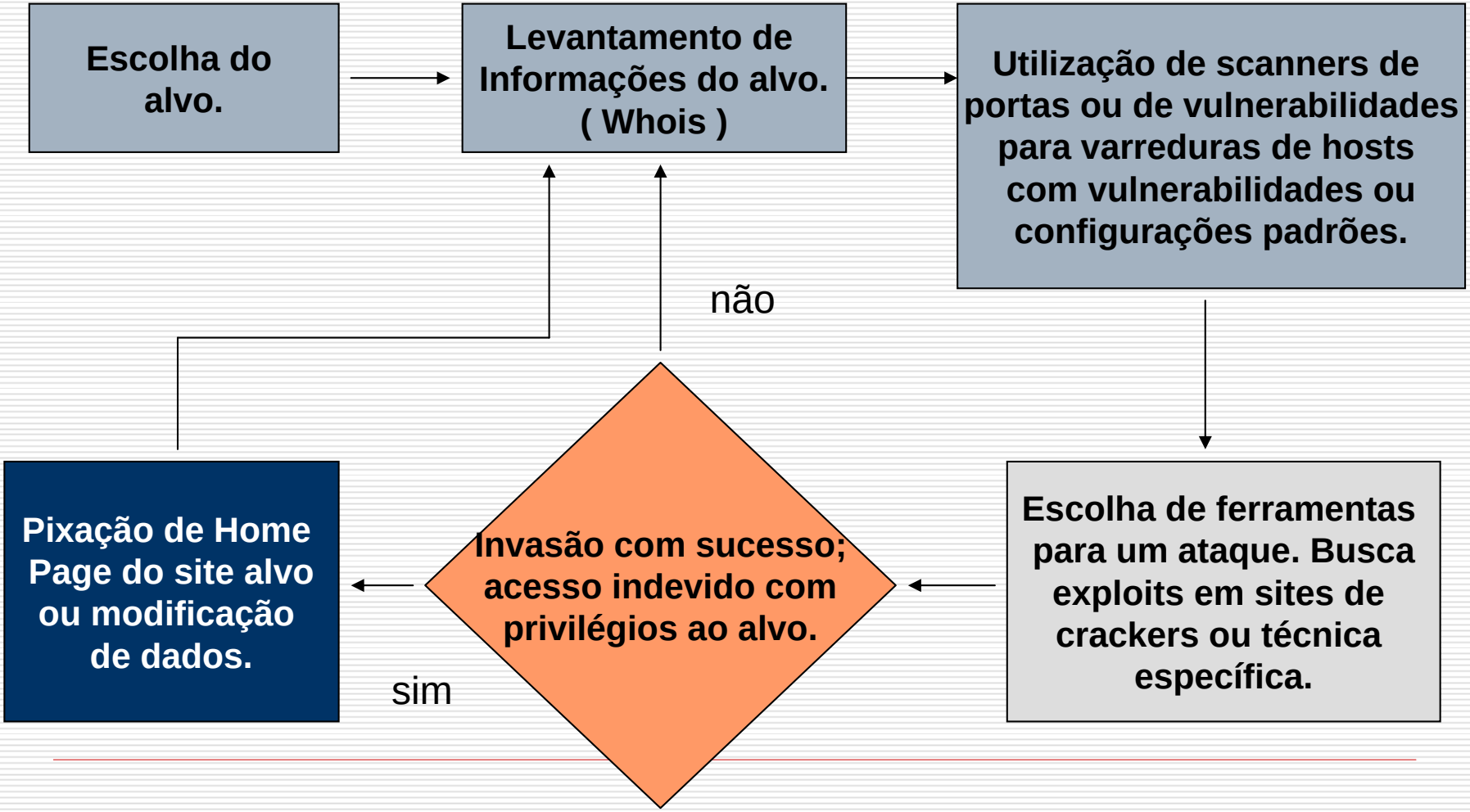
Anatomia simplificada de um Ataque



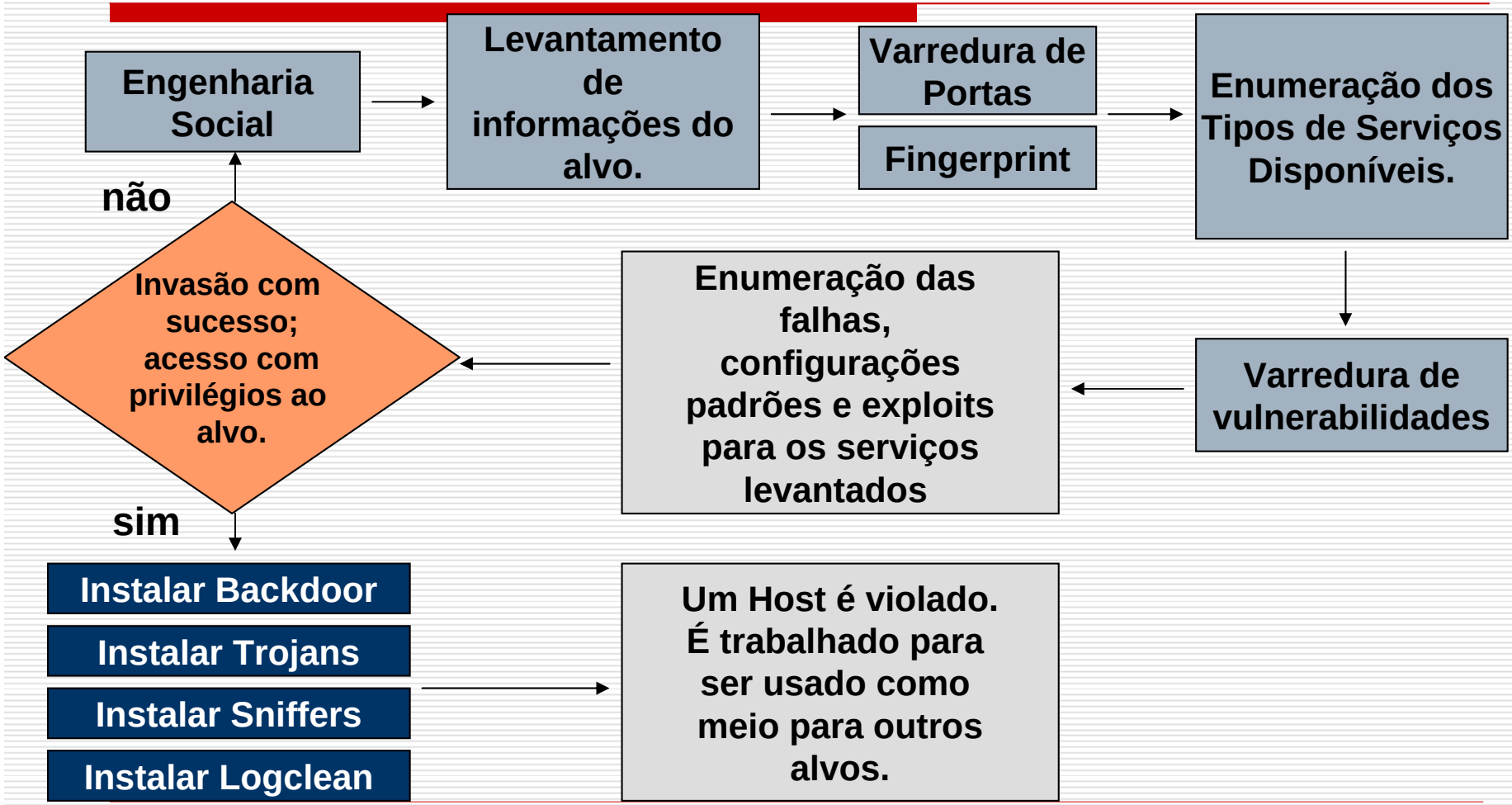
Anatomia de Ataques

- Um ataque é basicamente definido (com mais detalhes) em três etapas:
  - **Footprint**
  - **Fingerprint**
  - **Enumeração**

Anatomia de um Ataque Script Kiddie



Anatomia do Ataque Cracker



Rootkit

Ataque de Cracker

- “ ... a sua máquina nada tem de interessante ...”

- Máquina boa para ser “rooteda”
- Máquina boa para ser invadida e seus rastros serem apagados.
- O cracker consegue domínio total e usa a máquina para invadir outras.

Ataque de Cracker

- Caso o ataque seja detectado, tal máquina será identificada como a origem de ataques e seu administrador ,a priori, será responsável por possíveis danos.

- O cracker pode divulgar em canais IRC de crackers e script kiddies, as backdoors, a senha para se ter acesso à maquina comprometida.

Ataque de Cracker

- Com isso, o cracker além de ter apagado o seu rastro, chama a atenção do administrador para os “laranjas” que tenham acessado à máquina depois dele.

Etapas detalhadas de um Ataque

- Ver material impresso, distribuído em aula.

- Observações:

  - varreduras de portas
  - métodos de varreduras furtivas.
  - enumeração de informações em serviços.
  - ataques script kiddie e cracker

Técnicas de Varredura

- **Baseadas em portas**:

  Em que o scanner se comporta como um pseudo-cliente, identificando as **portas de serviços** ativos (**portas abertas**) em um determinado host.

Técnicas de Varredura

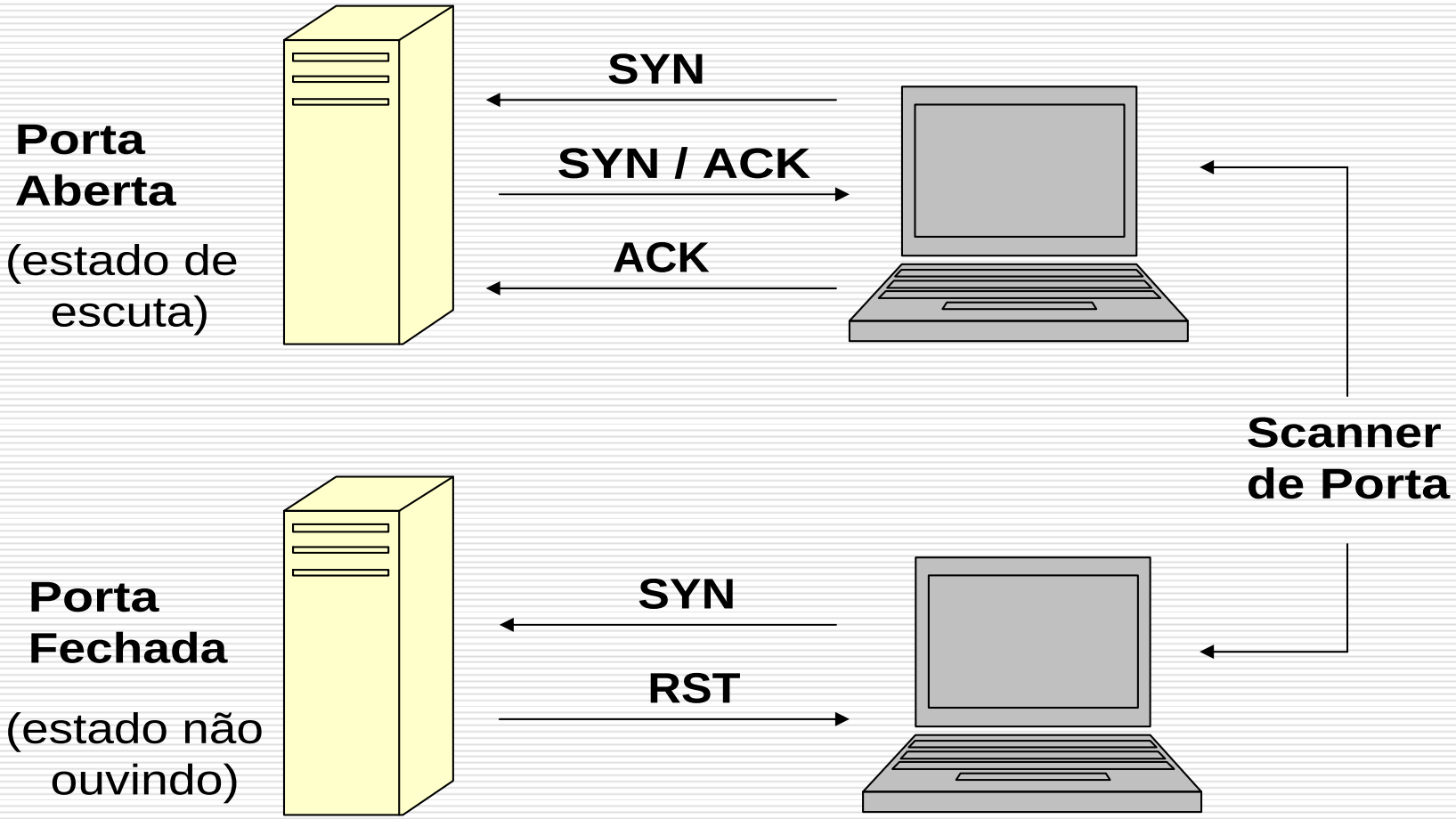
- **Baseadas nos serviços**

  Para levantamento de dados mais específicos.

Varreduras de portas clássicas

- TCP connect
- TCP syn (conexão semi-aberta)
- Baseadas na RFC 793 (não Microsoftware)
- TCP Xmas Tree (árvore de natal)
- TCP null (varreduras nulas)
- Microsoftware e RFC 793
- UDP
- ACK
- TCP window
- TCP fin/ack (fim de conexão)

TCP Connect


Porta Aberta
(estado de escuta)
SYN
SYN / ACK
ACK
Scanner de Porta
Porta Fechada
(estado não ouvindo)
SYN
RST

TCP Connect

- Quase todos scanners de portas usam esse recurso.
- Na prática, é um *handshake* para cada porta definida na varredura.
- Um *handshake* demanda duas mensagens TCP por porta.
- A varredura é facilmente detectável.
- Não é preciso nenhum privilégio especial.

TCP Connect

- Uma mensagem SYN é enviada.
- Se a **porta estiver (aberta) ouvindo com um serviço**, a conexão se sucederá.
- Um **SYN é retornado** estabelecendo **o número de sequência inicial**. Um **ACK** considera o **campo numérico de confirmação válido**.

TCP Connect

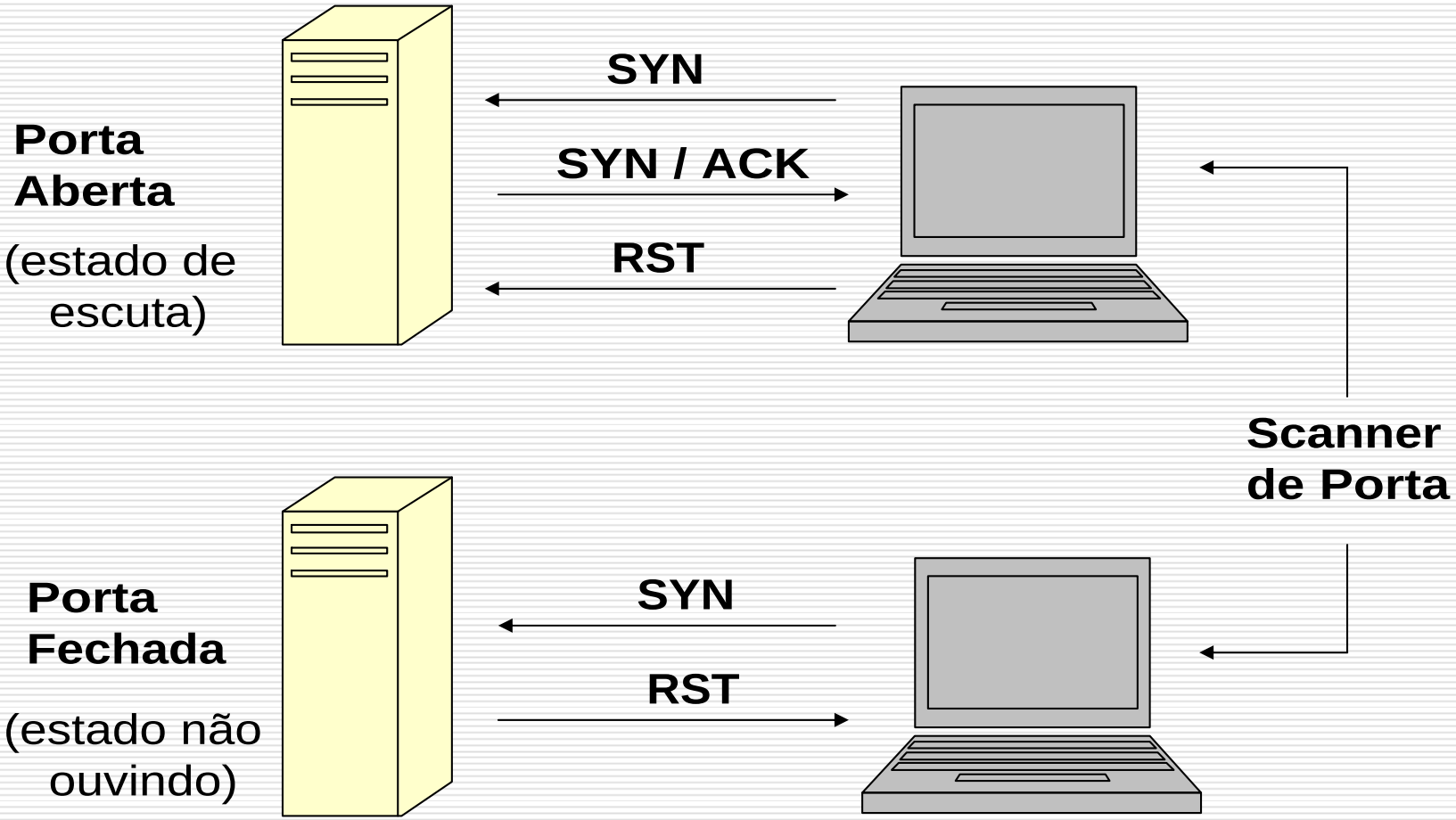
- Se a **porta estiver (fechada) sem serviço ouvindo**, uma mensagem **RST é retornada**, para **reiniciar o pedido de conexão**.

- A máquina-alvo mostrará uma conexão que falhou, porque a porta não está ouvindo, em estado de conexão.

TCP Connect

- Port Scanners
  - Nmap
  - Amap
  - Blaster

TCP SYN


Porta Aberta
(estado de escuta)
SYN
SYN / ACK
RST
Scanner de Porta
Porta Fechada
(estado não ouvindo)
SYN
RST

TCP SYN

- Técnica muito usada.
- O scanner envia uma mensagem SYN, como se estivesse pedindo uma conexão.
- Uma resposta da máquina-alvo com SYN/ACK indica que a porta se encontra ouvindo, através de um serviço.

TCP SYN

- Um RST indica que a porta não está ouvindo. O *handshake* é cancelado.
- A técnica ser conhecida como conexão semi-aberta, pois a exploração não demanda um *handshake* completo.

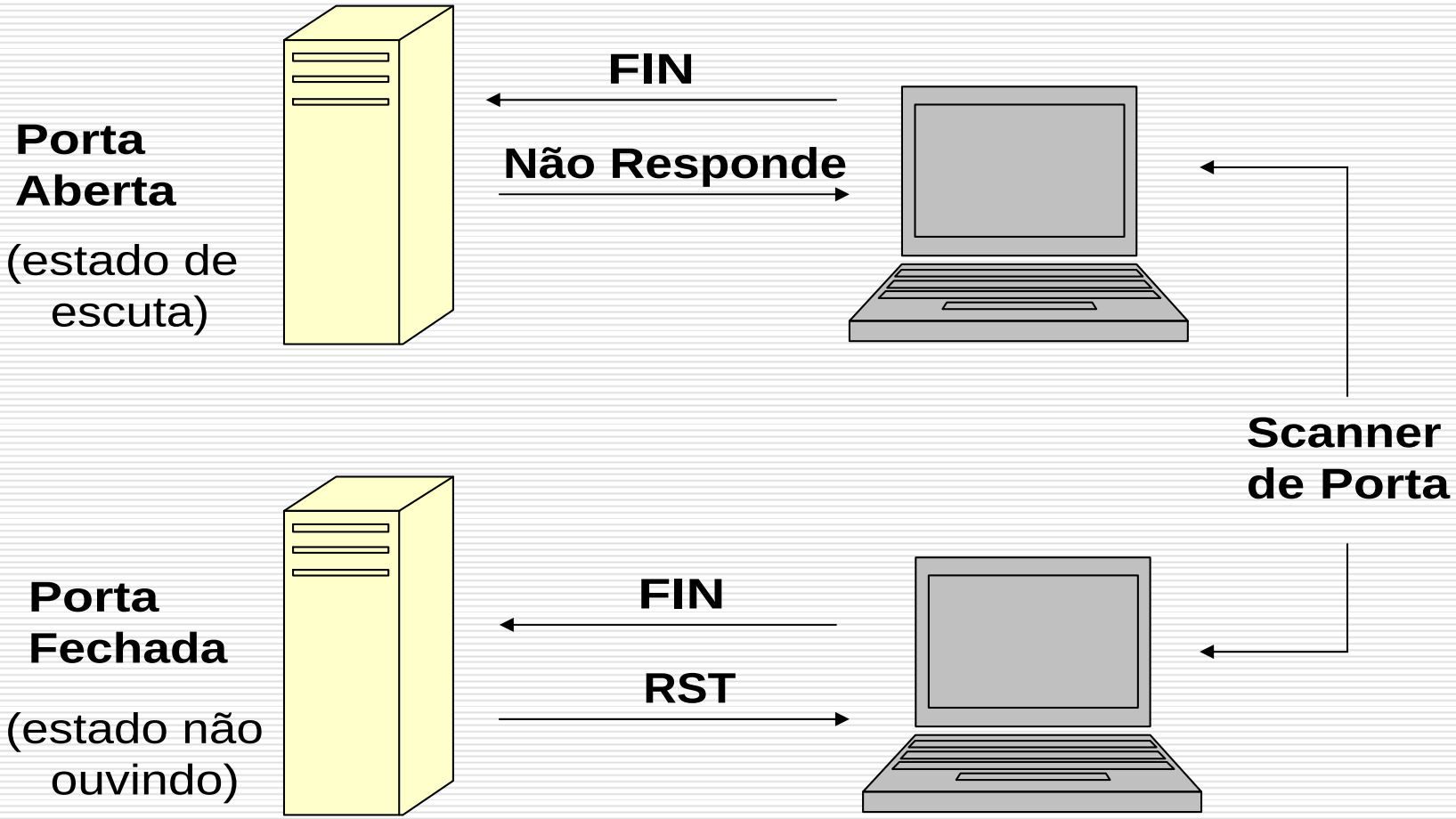
TCP SYN

- **Nmap** usa essa lógica.
- Mas, é comum encontrarmos scanners mal escritos que não enviam o RST após o SYN/ACK.
- É como se o scanner não tivesse recebido o SYN/ACK.
- Isto motiva o scanner a realizar uma segunda mensagem RST.
- Demanda privilégio de *root*.

TCP SYN

- Port Scanners
  - **Netstat** (Windows)
  - **Netcat**
  - **Amap** (ideal para leitura de *banners*)
  - **Blaster**
  - **Hping2**
  - **Nmap** (pode ser combinado com o **Amap**)

Varreduras baseadas na RFC 793



Varreduras baseadas na RFC 793

- Com **portas fechadas** (sem serviço), ao receberem **TCP FIN**, ou mensagem com prioridade **TCP FIN/ URG/ PSH**, ou mensagem **TCP NULL** (sem nenhum flag ativo), o host-alvo responde com um **TCP RST**.

Varreduras baseadas na RFC 793

- Quando a **porta estiver aberta (existe serviço)**, eles são ignorados. **O hos-alvo não responde**.

- O scanner não recebe nenhuma resposta, **pois não podem pertencer a nenhuma conexão estabelecida**.

Varreduras baseadas na RFC 793

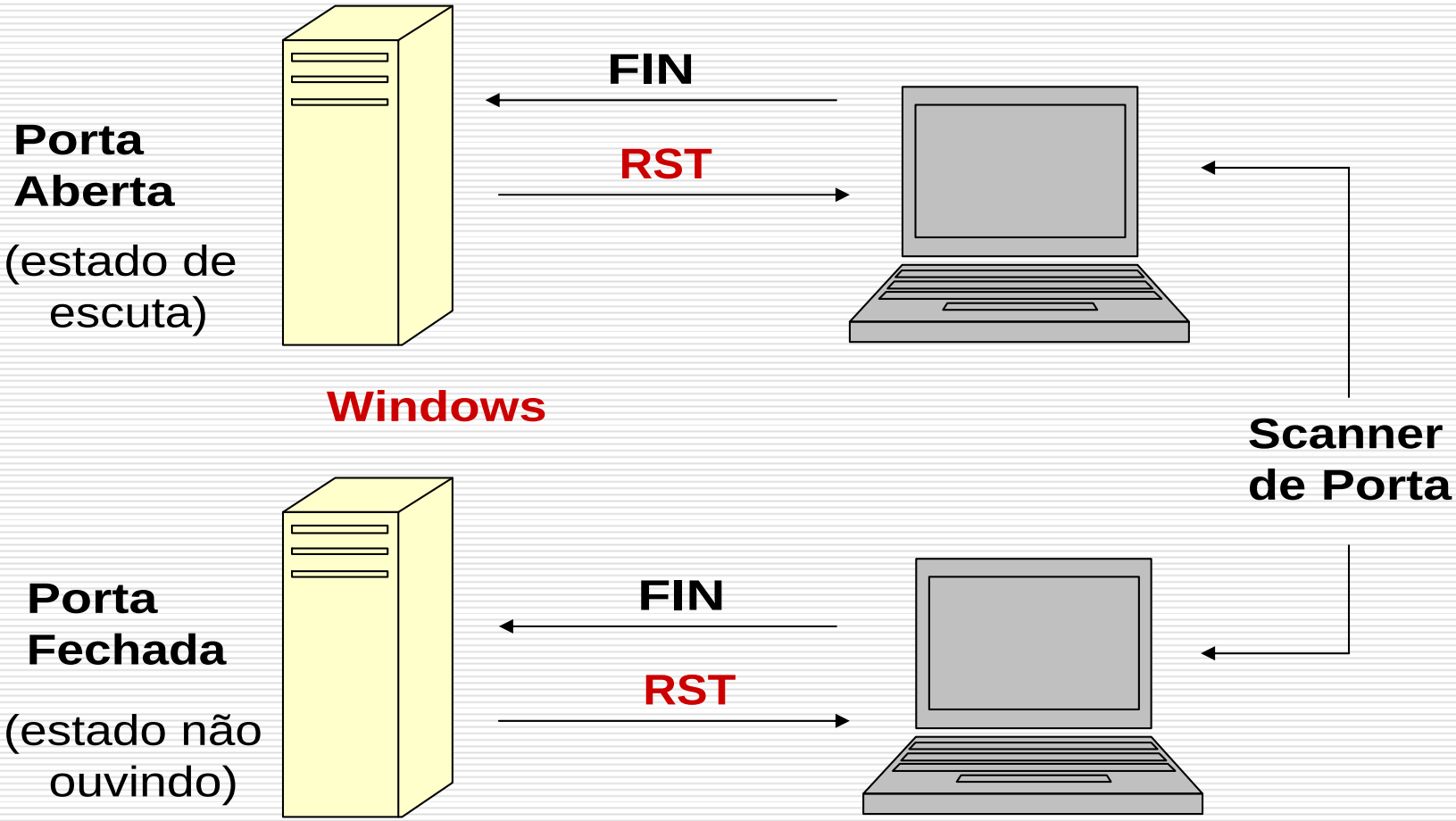
- Convém que um IDS na máquina-alvo, possa identificar varreduras baseadas na RFC 793.
- Se um IDS só identifica varreduras de início de conexão (TCP Connect e TCP SYN), a técnica RFC 793 passa despercebida.

Varreduras baseadas na RFC 793

- Não funcionam em pilhas TCP/IP Microsoftware, pois essas não seguem a RFC 793.

- Pilhas **TCP/ IP Microsoftware respondem com TCP RST** , tanto para **portas abertas**, como para **portas fechadas**, como segue:

Varreduras baseadas na RFC 793 em pilhas TCP/IP Microsoftware

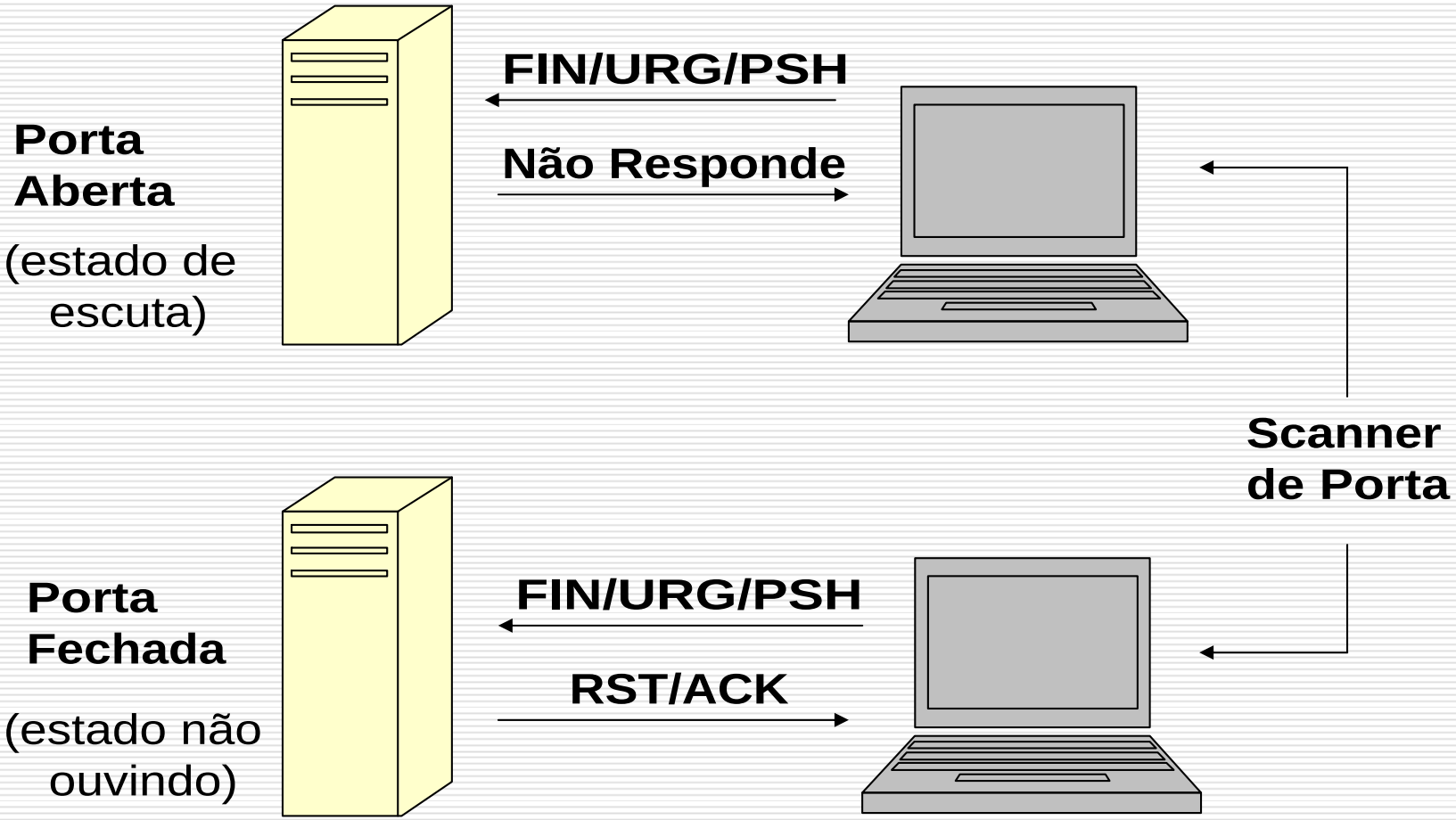


Varreduras baseadas na RFC 793

- Port Scanners

  - Hping2
  - Nmap

Varredura Xmas Tree



Varredura Xmas Tree

- Equivale a TCP FIN.
- Com **portas abertas** (com serviço), e mensagem com prioridade **TCP FIN/ URG/ PSH**, o host-alvo **não responde**.

Varredura Xmas Tree

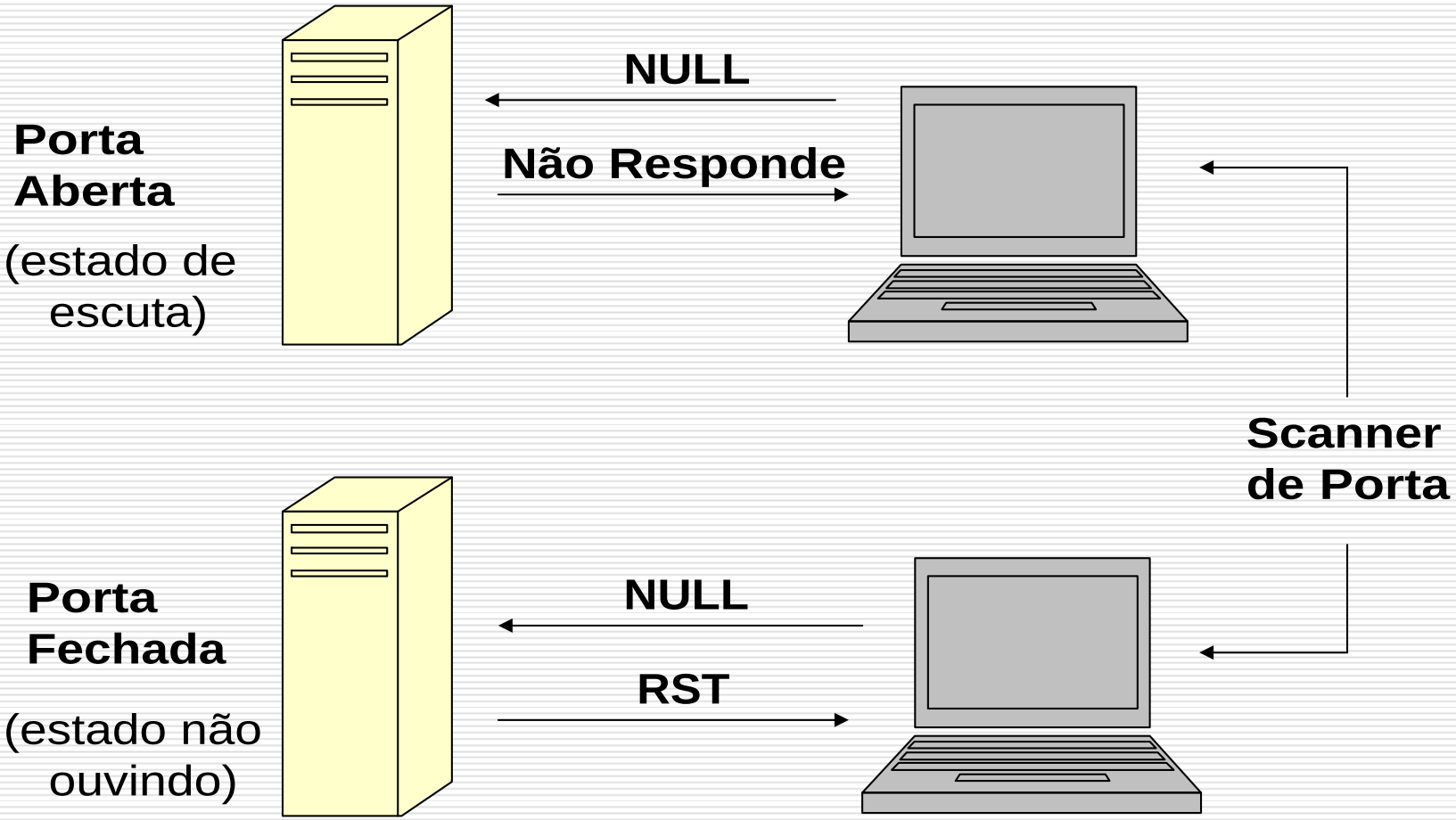
- Com **portas fechadas** (sem serviço), e mensagem com prioridade **TCP FIN/ URG/ PSH**, o host-alvo responde com um **TCP RST**.

Varredura Xmas Tree

- Port Scanners

  - Hping2
  - Netstat
  - Nmap

TCP Null (sem flags ativos)



TCP Null (sem flags ativos)

- Equivalente a TCP FIN.
- Tem-se resposta **TCP RST** para **portas fechadas**.
- **Não se tem resposta** para **portas abertas**.

TCP Null (sem flags ativos)

- Port Scanners

  - Hping2
  - Netstat
  - Nmap

Microsoftware e a RFC 793

Varredura ACK

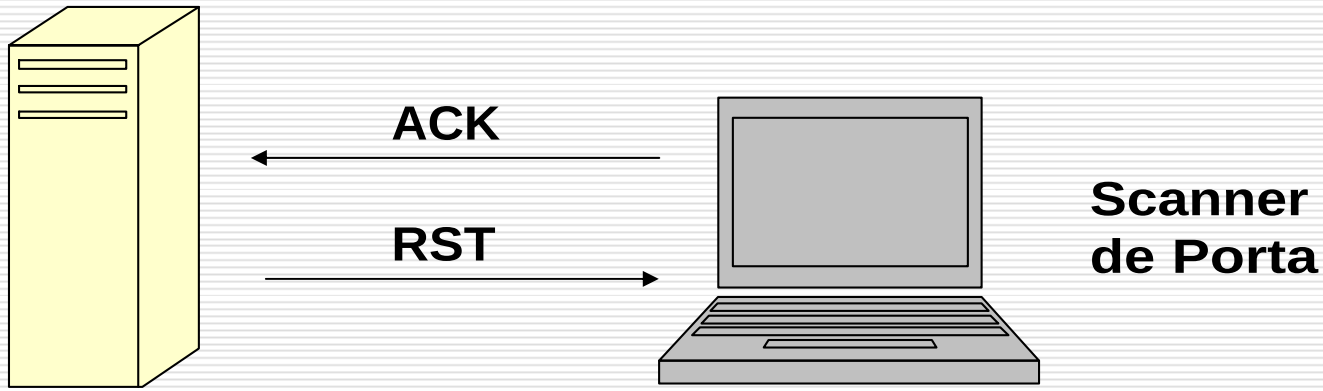
Porta Aberta

(estado de escuta)

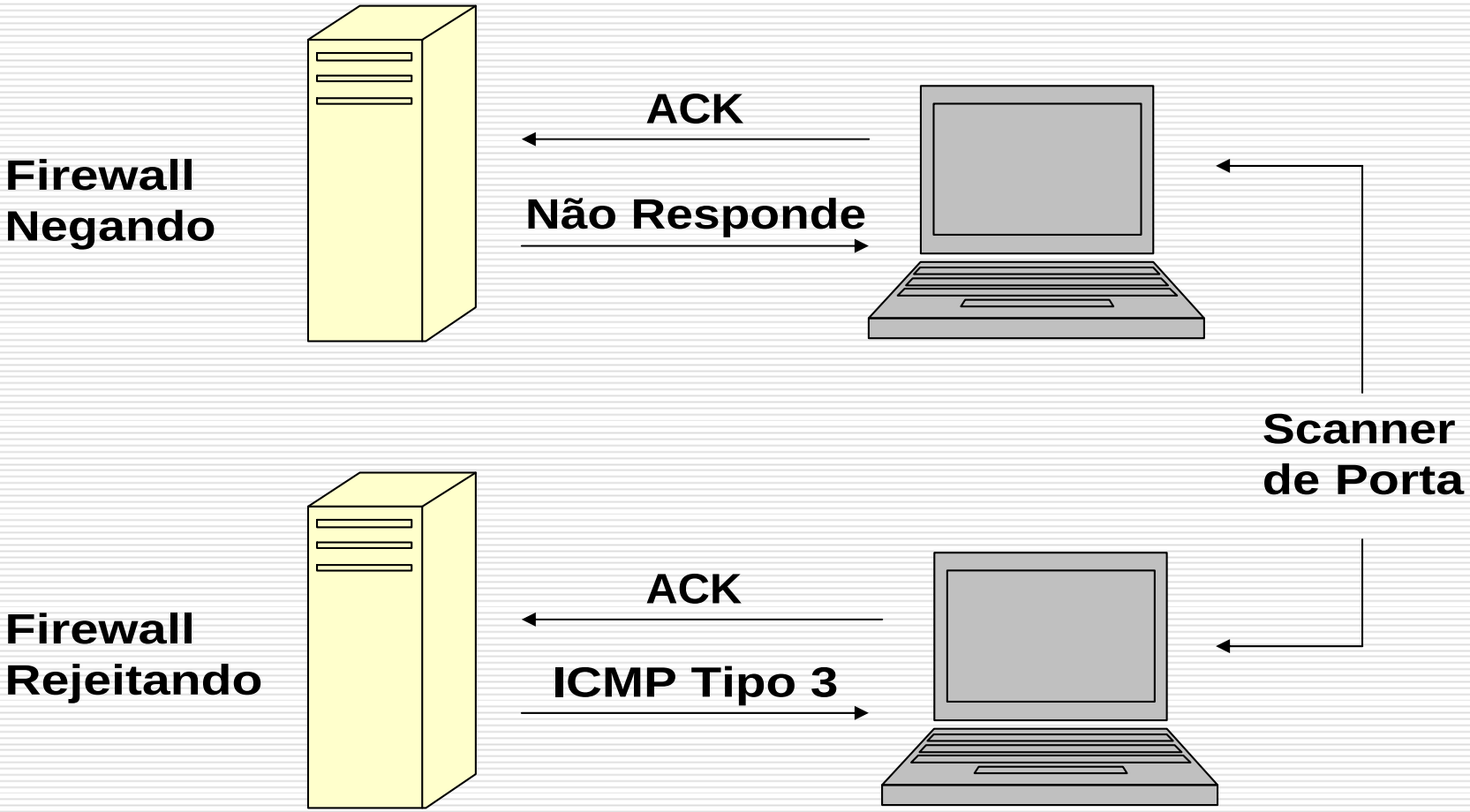
ou

Porta Fechada

(estado não ouvindo)



Varreduras ACK (continuação)



Varreduras ACK

- Técnica usada para identificar Firewalls.
- Um TCP ACK, que não pertença a nehuma conexão estabelecida, é gerado pelo scanner.
- Se um RST é devolvido pela máquina-alvo, tanto em uma porta aberta com em uma fechada, as portas são classificadas como não tendo Firewalls.

Varreduras ACK

- Sendo um ICMP 3 ou nenhuma resposta é devolvida, é assumido que as portas são filtradas, ou seja existe Firewall.

- Port Scanner

  Hping2

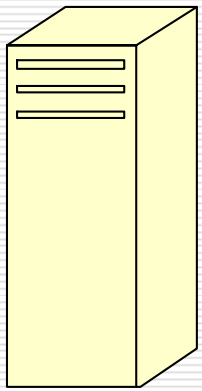
- Exemplo:

  > hping2 ip.ip.ip.ip **–ack** –p <porta aberta ou fechada> -c 3

Varredura TCP Window

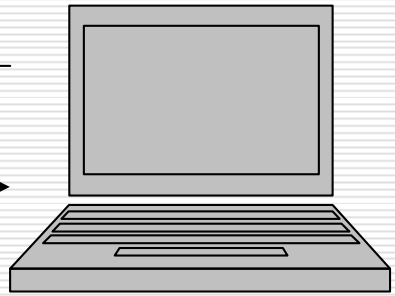
Porta Aberta

(estado de escuta)



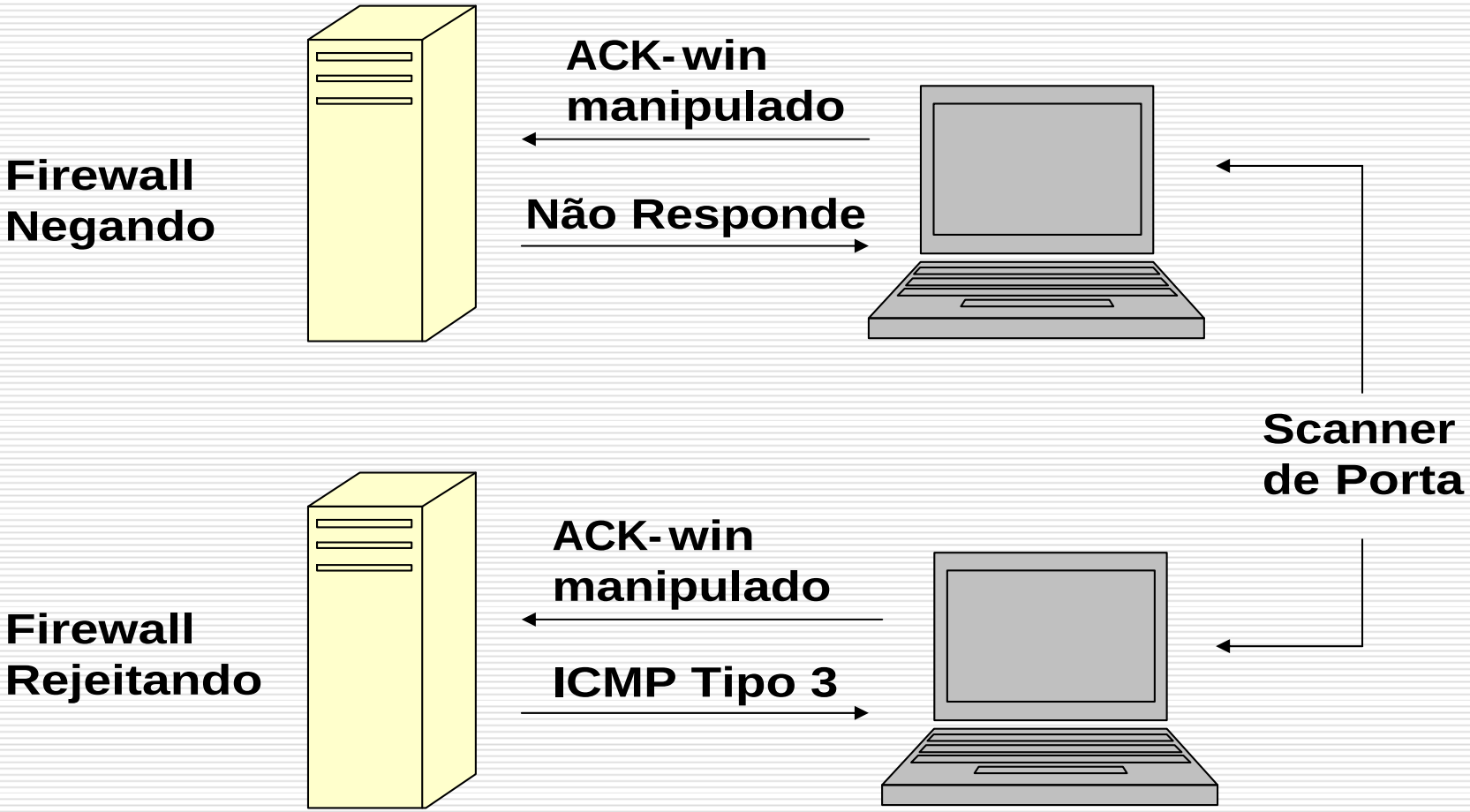
ACK-win

RST



Scanner de Porta

Varredura TCP Window (continuação)



Varredura TCP Window

- Técnica avançada.
- Tem como objetivo identificar portas protegidas por Firewall, e não portas abertas com outros serviços.
- Nmap envia ACK-win. Voltando **RST**, **a porta não está filtrada**.

Varredura TCP Window

- **Não tendo resposta** ou **voltando ICMP 3**, a porta está filtrada e assim existe Firewall.

- Por Scanner
  - Nmap

- Exemplo: > nmap **–sW** ip.ip.ip.ip

Varredura FIN/ACK

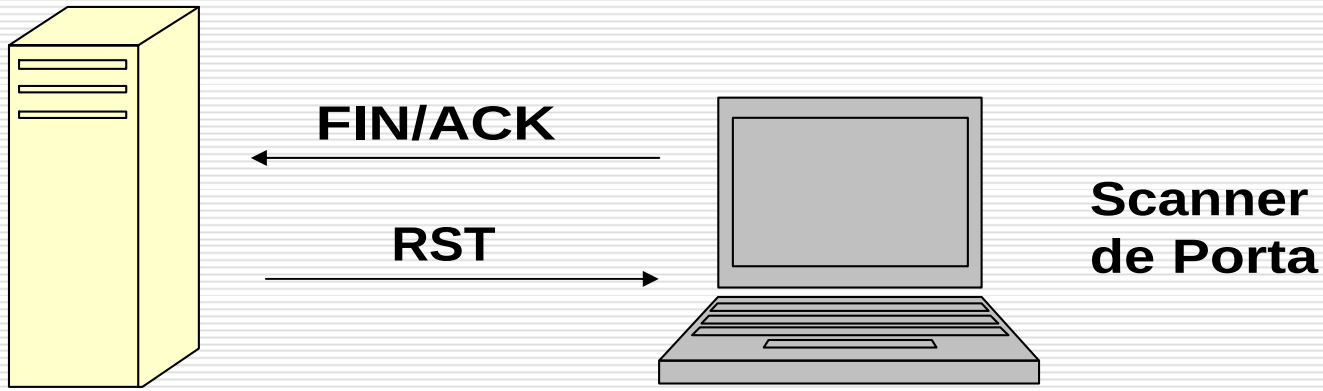
Porta Aberta

(estado de escuta)

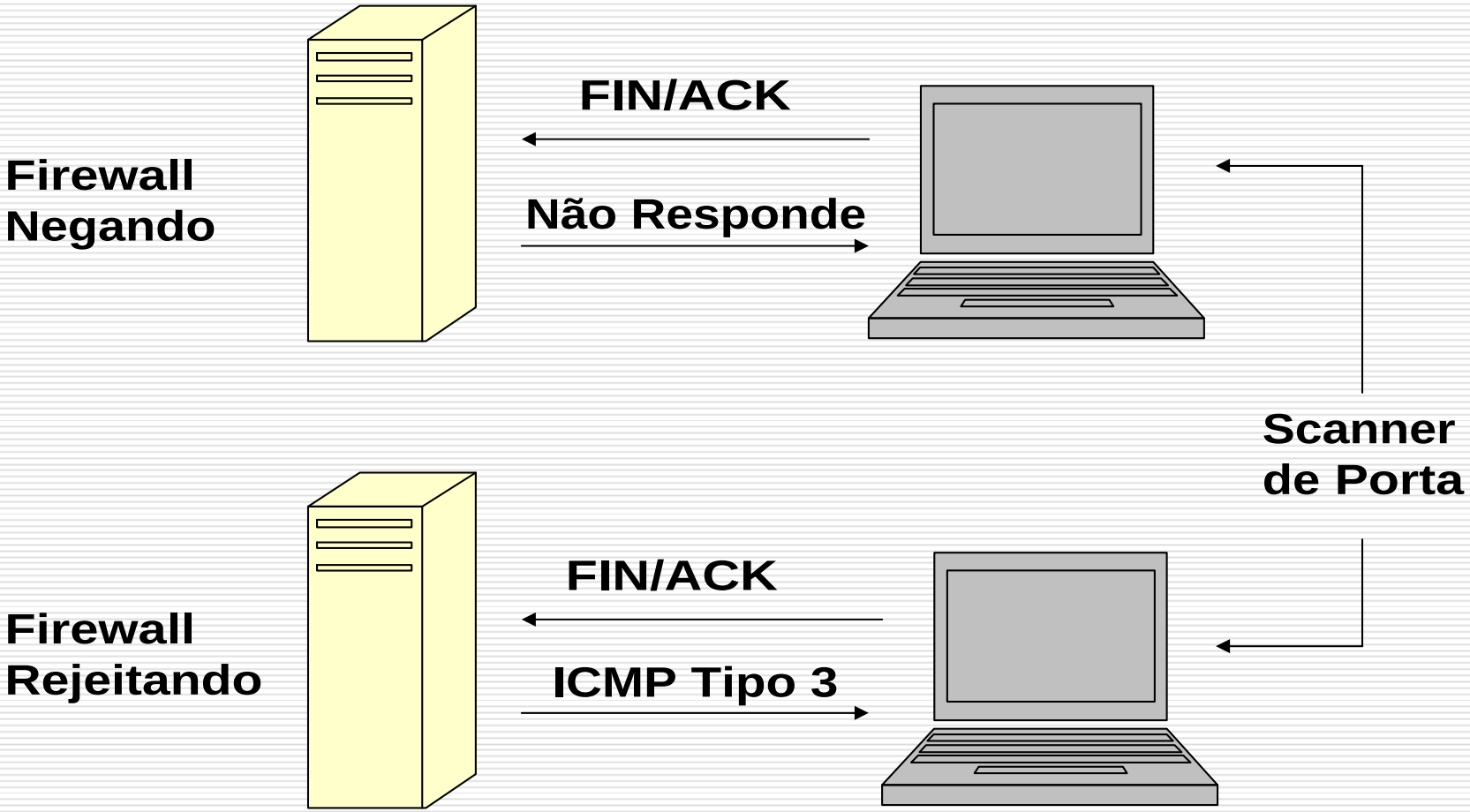
ou

Porta Fechada

(estado não ouvindo)



Varreduras FIN/ACK (continuação)



Varreduras FIN/ACK

- Forma de identificar um Firewall.
- Bit FIN ativo.
- Comportamento similar à varredura ACK.
- Port Scanners
  - Hping2
  - Nmap
- Exemplos:

  hping2 ip.ip.ip.ip –fin –ack –p <porta aberta ou fechada> -c 3

  nmap –sM ip.ip.ip.ip

Escondendo um Firewall

- Enganar um scanner como o Nmap.
- Se **Nmap receber um TCP RST** como resposta, ele envia dois pacotes.
- Para esses dois pacotes enviados, **Nmap assume que a porta não está filtrada**.

Escondendo um Firewall

- Se **Nmap recebe ICMP 3** como resposta, ele **assume que a porta é filtrada por um Firewall** que rejeita pacotes.

Escondendo um Firewall

- Se **Nmap não recebe nenhuma resposta**, ele envia mais quatro pacotes e, não obtendo nenhuma resposta, **ele assume a porta como filtrada**.

Escondendo um Firewall

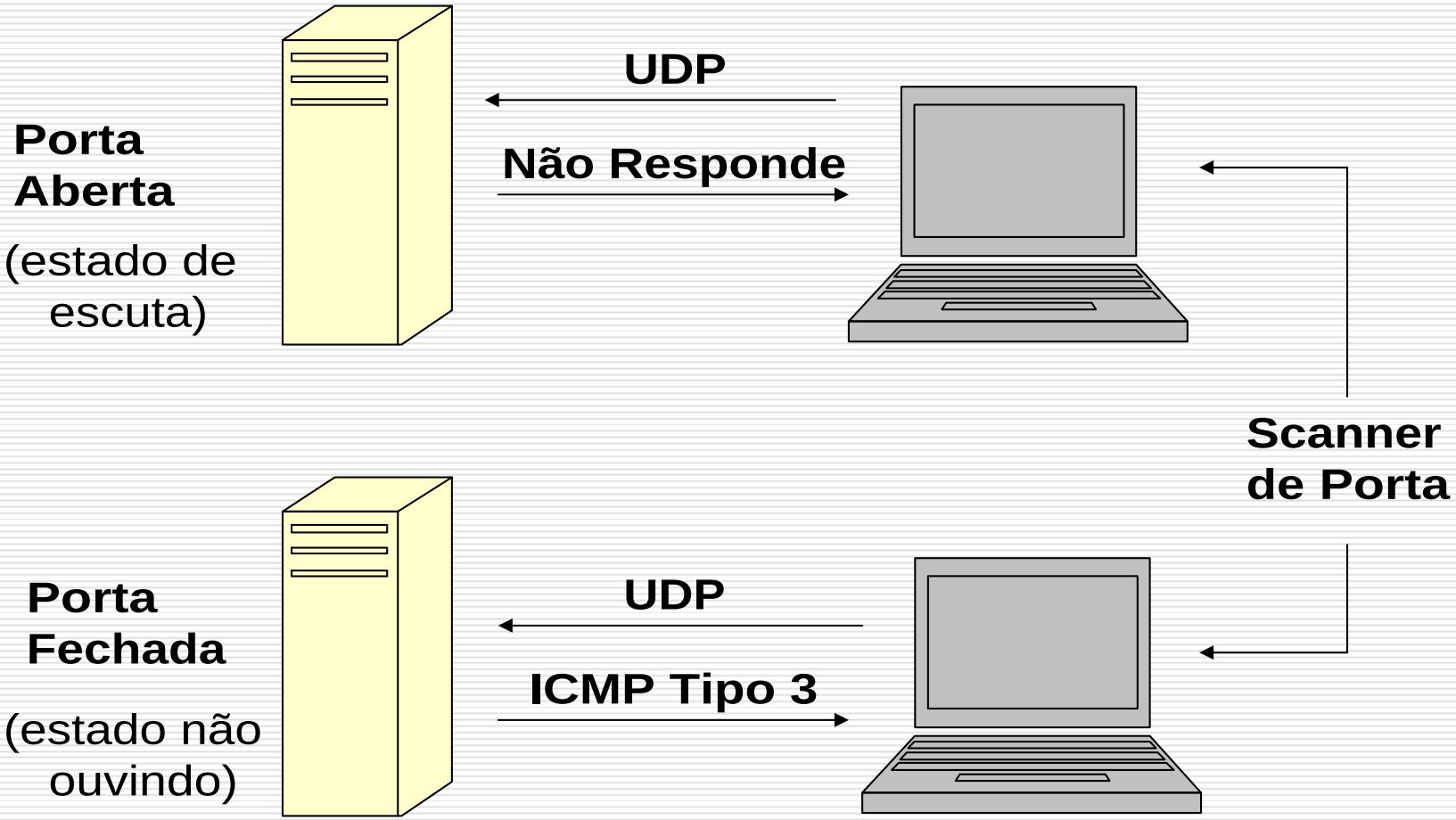
- Definindo-se uma **política para o firewall**, em que a porta 22 somente aceite conexões IP já pré-definidas, qualquer outro pacote IP será rejeitado.

- Na política, definimos **o firewall não rejeitar pacotes com ICMP** (não responder com ICMP), **mas com TCP RST**.

Escondendo um Firewall

- Isto, faz com que seja gerado um **falso negativo**, ou seja, uma informação tal que, a ocorrência existe, mas não é identificada (**o Firewall existe, mas o Nmap não o vê**).

Varreduras UDP



Varreduras UDP

- Técnica que descobre os serviços UDP ativos, ou seja, as portas UDP abertas em um host.
- Datagramas de 0 bytes são emitidos a cada porta da máquina-alvo.
- Se um datagrama chega em uma porta fechada, a máquina-alvo devolve uma mensagem de erro ICMP 3. Caso não retorne nada, supõe-se que a porta é aberta.

Varreduras UDP

- Port Scanners
  - Netstat
  - Hping2
  - Nmap

Varreduras RPC

- Varredura baseada em serviço.
- Lavantando dados de RPC com Nmap:

```
nmap –sR  ip.ip.ip.ip
```

Varredura Bounce

- Técnica que consiste em utilizar um serviço de um determinado host para levantar informações sobre outros serviços.

Varreduras baseadas no Cabeçalho do Protocolo IP

Varreduras ICMP e Discovery

Formas de Furtivas de Varreduras

Varreduras Furtiva Temporizadas

- Também conhecida como “Slow Scan”.
- **Temporiza** o envio de pacotes.
- Obtidas através do **Nmap** com a opção -T.
- Tipos de Varredura:

Métodos de Varreduras Furtivas

- **FTP Bounce / Proxy Bounce**
- **IP Decoy**
- Port Decoy
- Randon Port
- Slow Scan
- Coordinated
- Fragmenting
- **Spoofing / Sniffer**
- Multiprocessing
- Stateless

- **Servem para Intrusões.**

Varreduras Furtiva Temporizadas

- Tipos de Varredura:

Enumeração de informações em serviços

- SMTP scan
- SNMP scan
- SMB scan
- Reverse Ident
- RPC scan

Ferramentas de Ataque

- Constrói-se ou escolhe-se as ferramentas para a invasão.

- Rootkits:
  - Sniffer
  - Trojan
  - Backdoor
  - LogClean

Para concretizar um Ataque

- Instalação de **Sniffers**.
- Técnicas de **Backdoor**.
- **Apagamento de rastros ou forjar logs**, eliminando o rastro do invasor ou dificultando a auditoria (CleanLogs).
- **Ataques DoS**,
- **Ataques DDoS**, **DRDoS**

Ataques sem intrusão

- Existem formas de ataque que não têm objetivos de intrusão.

- Exemplos:
  - Spam em servidores que permitem ***relay*** (retrasmissão).

  - DoS, DDoS, DRDoS

Ataques sem intrusão

- Algumas supostas invasões ocorrem sem nenhuma intrusão no sistema.
- Como nos casos de ataques de **entrada inesperada**.

Para Auto-Monitoramento

- Verificadores de Senha (**John the Ripper**),
- Auditoria de Segurança de Sistemas (**Nmap**),
- Scanner de Segurança para identificar vulnerabilidades (**Nessus**).
- Firewalls, Web Proxy
- IDS de Host (**Tripware**),
- IDS de rede (**Snort**)

Melhor Proteção

- Estabelecimento de Políticas de Segurança.
- Informações Criptografadas em protocolos (S/MIME, SSH, SSL, TSL, IPSec...).
- Redes Privadas Virtuais (VPN com SSL, IPSec)

Capítulo 6

Força Bruta

Força Bruta

Auditoria de Senhas

Crackeando Senhas

O conceito de Intrusão

- **Ameaça** ou Tentativa (quando o invasor pula o muro).
- **Ataque** (concretiza o arrombamento).
- **Invasão** (quando obtém sucesso).

Para concretizar um Ataque

- **Obter meio de acesso não autorizado** a um sistema remoto com configurações padrões.

Força Bruta

- ***Auditando*** ou ***Crackeando*** Senhas.

- Força Bruta para *crackear* senhas em **Serviços**:
  - POP, FTP, Telnet, Proxy-Web,
  - Web Servers, roteadores ou SO

Força Bruta para Auditar ou Crackear Senhas

Auditando ou *Crackeando* Senhas

- Muitas vezes, as senhas são consideradas o lado mais fraco em uma política de segurança.
- É da natureza humana procurar a solução mais fácil para um problema.
- Usuários tendem a não criar senhas longas e complexas. Pois é difícil de lembrar.

Auditando ou *Crackeando* Senhas

- Muitas vezes tendem a criar senhas com algo no seu ambiente.
- Isso torna fácil para um invasor deduzir uma senha, ou fácil para um decifrador de senhas determinara essas senhas fáceis de lembrar.

Auditando ou *Crackeando* Senhas

- A maioria das empresas ainda conta com senhas, como único mecanismo de autenticação para acesso aos recursos de seus sistemas.

- Responsabilidade da equipe de segurança: garantir que as senhas correspondam a um requisito mínimo de segurança.

Auditando ou *Crackeando* Senhas

- **Contramedida**: o uso de verificadores de senha ou ferramentas de auditoria de senhas para reforçar políticas de senha.

  - ajuda a reduzir o risco imposto por senhas mal escolhidas.

  - Exemplos:
    - Jack Cracker (mais clássica)
    - Nutcracker (Escrito em Perl)
    - **John the Ripper**

Usando John the Ripper

- Alternativa ao Jack Cracker.
- Bem mais rápido e sofisticado que o Jack Cracker.
- Favorito de muitos *script kiddies* e *crackers*.
- É o preferido para auditoria de senha.
- Escrito em linguagem C.

John the Ripper

- A maioria dos sistemas usa MD5, ao invés de DES.
- Pode ser configurado para usar o tempo inativo do processador, para decifrar sessões.
- Permite a restauração de sessões de decifração.

John the Ripper

- Possui uma variedade de conjuntos de regras configuráveis.
- Qualquer software de vulnerabilidade de segurança deve ser instalado numa máquina que não seja de produção, pois este software possibilita a qualquer usuário, a decifragem de senhas do sistema.

John the Ripper

- Caso precise, usar permissões bem restritas, para os **arquivos de saída** e para o **arquivo usado para fazer auditoria**, como permissões **700**, com permissão de propriedade de *root*.

- Download em: http://www.openwall.com/john

John the Ripper – version 1.x

- Copie para o seu diretório de trabalho e descompacte:
  `>./tar xzf **john-1.x**.tar`

- Será criado um diretório: `john-1.x`

- Dentro do qual deverá ser acessado o subdiretório: `/src`

John the Ripper – version 1.x

- `>cd john-1.6/src/`
- Será exibida a lista de possíveis compilações, mostrando ser uma ferramenta multiplataforma:
  `>./make`
- `/usr` diretório Linux dos arquivos de perfis e configurações pessoais dos usuários.

John the Ripper – version 1.x

- >./tar xzvf john-1.x.tar.gz
  –c /usr/local/src

- Entre no diretório onde foi descompactado e leia o arquivo readme:
  >cd /usr/local/src/readme
  >less readme

- Dentro do readme tem as instruções de como proceder a instalação.

John the Ripper – version 1.x

- Compilando o John ...
  `>cd ./scr`

- `>make ...` (para a compilação, deve ser indicada a plataforma na qual será instalada John).

- Em Linux:
  `>make linux-x86-any-elf`

- `>make install`

John the Ripper – version 1.x

- Depois da instalação, os arquivos executáveis estarão no diretório:
  `/usr/local/john-1.x/run/`

- Depois de instalado, pode-se testar as senhas:
  `>cd ../run`

- `/etc` diretório que contém os arquivos de configuração de utilitários ou programas do sistema

John the Ripper

- shadow – arquivo de senhas do sistema Linux.
- Testar as senhas na forma básica de uso do John: >./john /etc/shadow

John the Ripper

- Exemplificando o modo single :

  >./john –single /etc/shadow

- Quebrando a senha de um usuário específico: livianvital

  >./john –show –users:livianvital
  /etc/shadow

John the Ripper

- Extraindo a senha do shadow:

```
>./unshadow /etc/passwd
  /etc/shadow > <arquivo-de-senha>
```

Argumentos do John

- >./john **-single** /etc/shadow
  utiliza as informações de login como base para a wordlist. Forma de simples de cracking.

- >./john **-wordfile**:/temp/dictionary.txt /etc/shadow
  modo de wordlist, serão usadas apenas as palavras contidas no arquivo dictionary.txt

Argumentos ...

- >./john **–rules** /etc/shadow
  Habilita regras para o modo de wordlist.

- >./john **–incremental** ..... Modo poderoso de cracker baseado em combinações.

- >./john **–external** ..... Modo de combinação que possibilita a utilização de definições externas.

John the Ripper

- Em **uma situação ideal**, não convém decifrar o arquivo shadow (arquivo que contém as senhas criptografadas) de uma máquina, na mesma máquina em que se encontra o arquivo shadow.

John the Ripper

- Se precisar executar o John the Ripper na mesma máquina, cuidar com o arquivo john.pot no diretório install directory/john-1.x/run/**john.pot**
- É em john.pot que estão todas as senhas decifradas.
- Usar este arquivo com permissões restritivas ...

John the Ripper

- Dicionário de palavras (supostas senhas) com 2.290 palavras `..../john-1.x/run/`**`password.lst`**

- Para ampliar o dicionário, fazer *download* de outros dicionários, e concatenar ao dicionário *default*. Usar esse último como padrão.

John the Ripper

- Se quiser usar uma lista de palavras diferente da padrão:

  >./john **-wordfile:**[diretorio/arquivo]

- Interrompendo o processamento do arquivo de senha: CNTL-C

- Para reiniciar a sessão:
  >./john **–restore** [arquivo a restaurar]

John the Ripper

- Para mostrar todas as senhas decifradas e usuários associados:

```
>./john -show /etc/shadow
```

- Regra estabelecida para verificar senhas de uma maneira concentrada. Configurando o arquivo john.ini localizado em `install directory/run/` , pode-se configurar conjuntos de regras únicos, dependendo das necessidades. Documentação sobre regras está em `install directory/docs/RULES` .

John the Ripper

- Os administradores utilizam Verificadores de Senha (Jack Cracker, Nutcracker, John the Ripper) **em seu ambiente**, para **auditar as senhas** de seu sistema, **descobrindo senhas fracas** e motivando uma política de senhas fortes.

Auditando Senhas

- **Contramedida**:

  Configurar o **SO** para verificar o tamanho e a complexidade de senhas através de módulos de autenticação conectáveis (**PAM – Pluggable Authentication Modules**) fornecidos com a distribuição.

- **PAM** é a biblioteca que permite autenticar usuários.

PAM

- Login local.

- Login Remoto:

  - **servidor de autenticação** (a base de usuários não está na mesma máquina do usuário, mas em uma máquina da rede.

PAM

- Modificar o programa *login* para que ele suporte autenticação remota.

- Se surgir um novo algoritmo de criptografia, mais rápido, que gostaríamos de usar, termos que modificar novamente o programa *login*.

PAM

- Num SO, muitos programas (aplicações ou serviços) utilizam algum tipo de autenticação de usuários.

- Imagine se esses programas tenham que ser reescritos, cada vez que algum dos critérios de autenticação seja alterado. ... ...

PAM

- SUN criou o PAM e liberou através de RFC.
- O **Linux** derivou sua implementação do PAM, a partir desse documento.
- Configurando o PAM no Linux, o programa (aplicação ou serviço) precisa ser reescrito apenas uma vez, justamente para suportar o próprio PAM.

PAM

- A partir daí o programa (aplicação ou serviço) delega a responsabilidade de autenticação para o PAM.

PAM

- No caso de se querer mudar o algoritmo de criptografia para senhas, basta que o PAM seja modificado para que todos os programas, passem automaticamente e de modo transparente, a usufruir dessa nova forma de autenticação.

PAM

- É possível configurar a autenticação de forma individual para cada programa (aplicação ou serviço).

- Com isso, pode-se ter um usuário usando certos recursos de HW, desde que os mesmos sejam acessados pelo console da máquina. Se o *login* não tiver sido feito pelo console, o acesso ao recurso de HW é negado.

PAM

- Nenhum programa (aplicação ou serviço) sabe alguma coisa sobre recursos de HW. Eles não precisam saber!
- O PAM se encarrega disso.
- O PAM vai além da autenticação.

PAM

- Os módulos do PAM podem ser de quatro **tipos**:
  - auth
  - account
  - passwd
  - session

PAM – Tipo auth

- Tipo de módulo que **verifica se o usuário é mesmo quem ele diz ser**.
- Pode pedir apenas o ***username*** e uma ***password***.
- Ou usar **biometria**: autenticar atarvés da impressão digital, imagem da retina ou impressão de voz.

PAM – Tipo account

- Autorização e Acesso
- Verifica se o usuário está autorizado a uitilizar o serviço ao qual está se autenticando.

PAM – Tipo passwd

- Usado quando se deseja mudar a senha.
- Podem ser adicionados módulos que verifiquem se uma senha é forte ou fraca.

PAM – Tipo session

- Encarregada de executar o que for necessário para **criar o ambiente do usuário**.

- Fornecer **acesso a alguns dispositivos locais**:
  - áudio,
  - CD-ROM,
  - fazer registro de eventos nos arquivos de *log* do sistema SO,
  - ou montar sistemas de arquivos.

Exemplo de Módulos PAM

- **pam-pwdb**
  Pode ser usado com todos os quatro tipos.

- **pam-console**
  Normalmente usado como *session*.

Força Bruta em Serviços

Força Bruta em Serviços

- Técnicas clássicas e “barulhentas”.
- A maioria dos sistemas gera *logs* de tentativas de conexão.
- Ferramentas:
  - Sdi.brutus.pl (Melo, S. 2004 p.130)
  - Blaster (Melo, S. 2004 p.130)
  - **Hydra** (Melo, S. 2004 p.131)

Hydra

- Escrita na linguagem C.
- Desenvolvida para ambientes POSIX.
- Objetivo: descobrir ***username*** e/ou ***password*** a partir de um serviço.
- Arquivos: **`userlist.txt`**
  `passwd.txt`

Hydra

- Front End **Xhydra** em GTK.
- Linux, UNIX BSD, Solaris, Mac OS/X e outros UNIX-like.
- Windows com Cygwin, com suporte a IPV4 e IPV6.
- Sistemas móveis baseados em processadores ARM (Zaurus, Ipaq).

Hydra

- **Prova o conceito de recursos de segurança**, com a possibilidade de mostrar a facilidade de se **obter acesso não-autorizado** a um sistema remoto, ...
- ... dentro de um cenário em que o administrador mantém **configurações padrões de contas** e **senhas fracas** nos serviços disponíveis.

Hydra

- É possível testar os seguintes serviços:

  Telnet, FTP, HTTP, HTTPS, HTTP-Proxy, LDAP, SMB, SMBNT, MS-SQL, MySQL, POP3, IMAP, NNTP, ICQ, PCNFS, VNC, SOCKS5, REXEC, SAP/R3, Cisco Auth, Cisco Enable, Cisco AAA.

Hydra

- SSH e Oracle.
- Pode usar a técnica de *Bounce* para força-bruta em aplicação Web utilizando um Proxy-Web mal configurado.

Hydra

- Compilando Hydra:
  ```
  >./configure
  >make
  >make install
  ```

- Compilando o Front End GTK
  ```
  >cd hydra-gtk
  >./configure && make && make
   install
  ```

Hydra

- Compilando em Palm Pilot e Mobiles baseados em processadores ARM:
  >./configure-palm
  >./configure-arm

- Por padrão, o Hydra será instalado em /usr/local/bin/...
  Seu binário é “hydra” e o binário do Front End é “xhydra”.

Hydra

- **`hydra <ip-alvo> <def-serviço> <opções>`**

- Opções Especiais:

  opção “-m”

  Alguns serviços requerem técnicas de força-bruta com a opção “-m”.

  WWW, SSL, HTTP, HTTPS,

Hydra

- Restaurando uma sessão abortada ou travada.
  - CNTL C
  - hydra.restore (arquivo)
  - > ???

- Performance no uso do Hydra
  - opção “-t”
  - desempenho depende do protocolo.
  - o mais rápido, é geralmente, o POP3.

Hydra

- Outras opções em serviços como:

  SMBNT, LDAP, serviços Cisco, SAP/R3

- O aplicativo **PW-INSPECTOR**:

  Utilitário para manipular wordlist, extraindo de uma wordlist uma segunda wordlist seguindo padrão pré-definido pelos seus parâmetros.

PW-INSPECTOR

- Serve para criar outras *wordlists*, quando o atacante sabe o perfil de senha que o alvo utiliza, resultando assim numa redução da lista de senhas (*wordlist*).

  Por exemplo: senhas com o mínimo de 6 caracteres.

PW-INSPECTOR

- Seja uma wordlist com vários tipos de senhas: `words.txt`

- Ordenando words.txt
  ```
  >cat words.txt | sort | uniq >
   dictionary.txt
  ```

- Extraindo de `dictionary.txt` apenas as senhas que atendam ao padrão:
  ```
  >cat dictionary.txt | pw-inspector –m
   –c 2 –n > passlist.txt
  ```