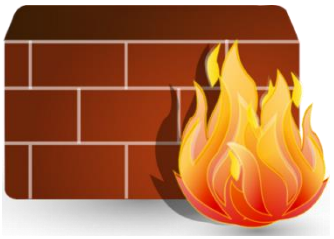
Universidade Federal do Ceará
Campus Quixadá



Iptables

Prof. Michel Sales Bonfim
Disciplina: Auditoria de Segurança de SI | **Curso**: Sistemas de Informação

Firewalls em Linux

Kernel e Firewall

- No Linux, as funções de Firewall são agregadas à própria arquitetura do kernel.
  - Tudo o que que é feito no sistema, deve ser de conhecimento do Kernel.
  - Tudo o que chega ou sai de um computador é processado pelo Kernel do sistema operacional desse computador.
- Vantagens:
  - No Linux, não é preciso comprar um Firewall corporativo caríssimo.
  - Firewall é *OpenSource* (gratuito).
- **Netfilter**:
  - Software acoplado ao sistema.
  - Auxilia o Kernel na agregação de funções de controle de fluxo internos, em termos de Firewall.

Netfilter

- Presente no Linux a partir do kernel série 2.4.x
  - Compatível com ipchains e ipfwadm
- Statefull Packet Inspection
- É um conjunto de situações de fluxo, agregadas inicialmente ao Kernel do Linux, e dividido em tabelas.
- Grande banco de dados que contém, em sua estrutura, 3 tabelas padrões, com funções específicas:
  - Filter
  - Nat
  - Mangle
- Cada uma dessas tabelas possuem situações de fluxo (chains), que lhes proporcionam a realização de seus objetivos.

Netfilter – Tabela Filter

- É a tabela padrão do Netfilter e trata das situações implementadas por um Firewall filtro de pacote.

- Situções de Fluxo (Chains)
  - **INPUT**: Tudo que entra no host
  - **FORWARD**: Tudo que chega ao host mas deve ser redirecionado a um host secundário ou outra interface de rede
  - **OUTPUT**: Tudo que sai do host

Netfilter – Tabela Nat

- Esta tabela implementa funções de **NAT (Network Address Translation)** ao host do Firewall

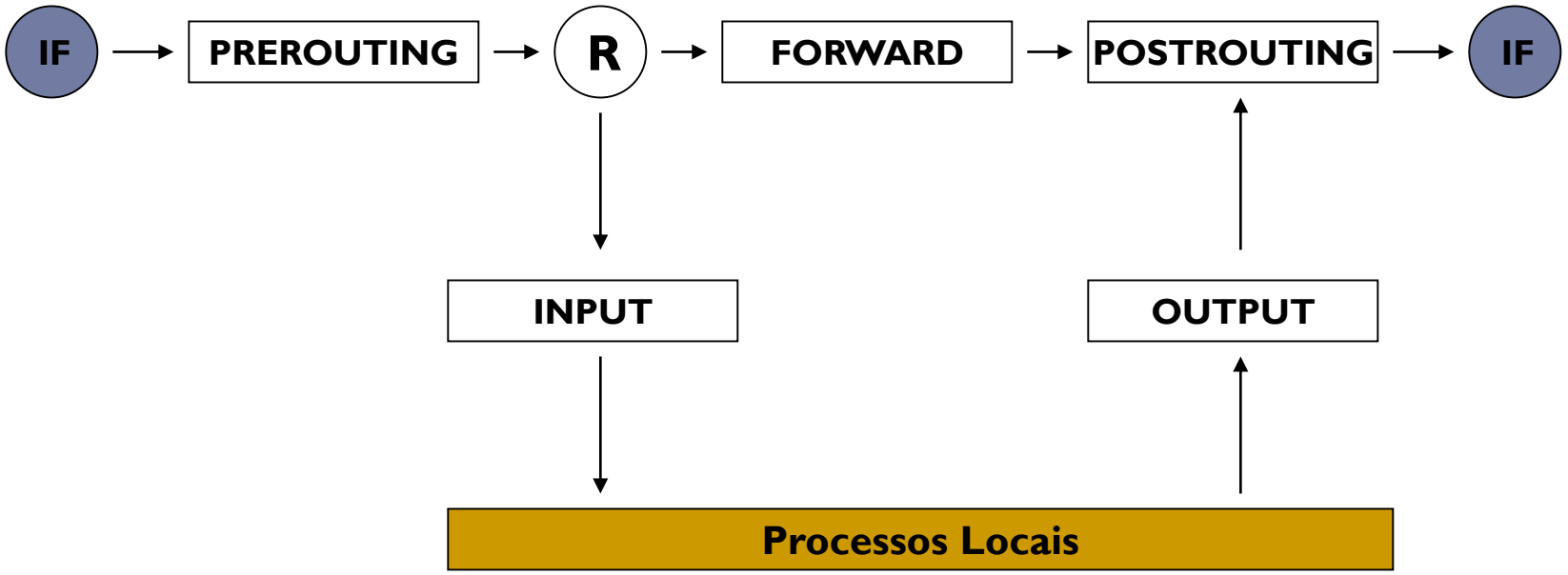
- Situações de Fluxo (Chains):
  - **PREROUTING**: Utilizado quando há necessidade de se fazer alterações em pacotes antes que os mesmos sejam roteados.
  - **OUTPUT**: Trata os pacotes emitidos pelo host Firewall
  - **POSTROUTING**: Utilizado quando há necessidade de se fazer alterações em pacotes após o tratamento de roteamento.

Netfilter – Tabela Mangle

- Implementa alterações especiais em pacotes em um nível mais complexo.
  - Altera a prioridade de entrada e saída de um pacote baseado no tipo de serviço (TOS), a qual o pacote se destinava.

- Situações de Fluxo:
  - **PREROUTING**: Modifica pacotes dando-lhes um tratamento especial antes que sejam os mesmo roteados.
  - **OUTPUT**: Altera pacotes de forma "especial" gerados localmente antes que os mesmo sejam roteados.

Netfilter – Situações de Fluxo (Chains)



Iptables

- Para que possamos vir a moldar o Netfilter conforme nossas necessidade, levando em consideração que o mesmo deve estar copilado com o Kernel, precisamos de um ferramenta que nos sirva de Front-End nesta tarefa.
- Um Front-End possibilitará o controle das situações (chains) contidas nas tabelas agregando-lhes regras de tráfego.
- Histórico:
  - Kernel 2.0 - Ipfwadm
  - Kernel 2.2 - Ipchains
  - **Kernel 2.4 e superiores - Iptables**

Firewall Iptables

Introdução

- Ferramenta Front-End que permite manipular as tabelas do Netfilter.
  - Não confundir o Iptables como um firewall por si só.
  - A partir de agora Iptables/Netfilter.
- Concebido por Rusty Russel, em colaboração com Michel Neuling
- Incorporado a versão 2.4 do Kernel em julho de 1999.
  - Quarta geração de firewalls Linux.

Características

- Realiza suas tarefas de forma veloz, segura, eficaz e econômica (tanto no aspecto financeiro como de requerimento de hardware).
- Funções:
  - Filtro de pacotes statefull (Tabela Filter)
  - Suporte a SNAT e DNAT (Tabela Nat)
  - Desenvolvimento de QoS sobre o tráfego (Tabela Mangle)
  - Roteamento (redirecionamento de endereços e portas)
  - Monitoramento do Tráfego.
  - Bloqueios a ataques Spoofing, Syn-Flood, DDoS, scanners ocultos.
  - Utilização de módulos externos na composição das regras.

Aplicativos

- **iptables**: aplicativo principal do pacote iptables para protocolos ipv4.
- **ip6tables**: aplicativo principal do pacote ptables para protocolos ipv6.
- **iptables-save**: aplicativo que salva todas as regras inseridas na sessão ativa e ainda em memória em um determinado arquivo pelo administrador do Firewall.
- **iptables-restore**: aplicativo que restaura todas as regras salvas pelo software iptables-save.

Sintaxe

```
# iptables [tabela] [comando] [ação] [alvo]
```

Sintaxe - Tabelas

- Utilizar a opção **-t**

```
# iptables -t filter

# iptables -t mangle

# iptables -t nat
```

- A tabela **filter** é a padrão do Iptables.

Sintaxe - Comando

- **[comando] : manipula a tabela através das regras e chains correspondentes.**
  - **`-A`** anexa a regra ao fim da lista já existente.
  - **`-D`** apaga a regra especificada.
  - **`-L`** lista as regras existentes na lista.
  - **`-P`** altera a política padrão das chains.
  - **`-F`** remove todas as regras, ou remove todas as regras referentes a um determinado chain.
  - **`-I`** insere uma nova regra, mas no início da lista de regras.
  - **`-R`** substitui uma regra já adicionada por outra.
  - **`-N`** permite inserir uma nova chain na tabela especificada.
  - **`-E`** Renomeia uma nova chain criada.
  - **`-X`** apaga uma chain criada pelo administrador do Firewall.

Sintaxe - Ação

- **[ação] :**
  - `-p` especifica o protocolo,
  - `-i` especifica a interface de entrada a ser utilizada;
  - `-o` especifica a interface de saída a ser utilizada;
  - `-s` endereço de origem do pacote (IP) ou sub-rede;
  - `-d` endereço de destino do pacote (IP) ou sub-rede;
  - `!` exceção a uma determinada regra;
  - `-j` define o alvo para um pacote;
  - `--sport` aplicar filtros com base na porta de origem;
  - `--dport` aplicar filtros com base na porta de destino.

Sintaxe - Alvo

- **[alvo] : quando um pacote se adequa a uma regra, ele deve ser direcionado a um alvo e quem especifica é a própria regra.**
  - **ACCEPT** Corresponde a aceitar, ou seja, permite a entrada do pacote.
  - **DROP** Corresponde a descartar o pacote, sem infomar ao dispositivo emissor o que houve.
  - **REJECT** Corresponde a rejeitar o pacote, retornando uma mensagem de erros ao host emissor, informando o que houve.
  - **LOG** Cria uma entrada de log no arquivos /var/log/messages, sobre a utilização dos demais alvos.
  - **RETURN** Retorna o processamento do chain anterior sem processar o resto do chain atual.
  - **QUEUE** Encarrega um programa, em nível de usuário, de administrar o processamento do fluxo.
  - **SNAT** Altera o endereço de origem das máquinas cliente antes dos pacotes serem roteados
  - **DNAT** Altera o endereço de destino das máquinas clientes.
  - **REDIRECT** Realiza o redirecionamento de portas (utilizado em conjunto com a opção –to-port)
  - **TOS** Prioriza a entrada e saída de pacotes passeado no TOS

Observação

- Antes de continuarmos com os exemplos, devemos habilitar o Firewall para encaminhar pacotes.
- Sem isso, a chain Forward nunca será utilizada e será impossível trabalhar com NAT.

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

Exemplo 1:

- Listagem de registro de regras:

```
[root@johann /]# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source                destination
Chain FORWARD (policy ACCEPT)
target prot opt source                destination
Chain OUTPUT (policy ACCEPT)
target prot opt source                destination
```

Exemplo 1:

- Listagem de registro de regras:

```
[root@johann /]# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target prot opt source                destination
Chain POSTROUTING (policy ACCEPT)
target prot opt source                destination
Chain OUTPUT (policy ACCEPT)
target prot opt source                destination
```

Exemplo 2:

- Aplicando regra-padrão na tabela filter (DROP):

```
[root@johann /]# iptables -P INPUT DROP
[root@johann /]# iptables -P FORWARD DROP
[root@johann /]# iptables -P OUTPUT DROP
```

Exemplo 3:

- Liberar totalmente o tráfego de entrada de nossa interface Loopback (lo). Esta regra é obrigatória, por permitir a comunicação entre processos.

```
[root@johann /]# iptables -A INPUT -i lo -j ACCEPT
```

Exemplo 4:

- Rejeitar pacotes entrantes pela interface de rede eth1

```
[root@johann /]# iptables -A FORWARD -i eth1 -j REJECT
```

Exemplo 5:

- Deletar a segunda regra inserida sobre a chain FORWARD:

```
[root@johann /]# iptables -D FORWARD 2
```

Exemplo 6:

- Pacotes que entram por qualquer interface de rede, com exceção da eth0, sejam descartados.

```
[root@johann /]# iptables -A FORWARD -i ! eth0 -j DROP
```

Exemplo 7:

- Descartar qualquer pacote oriundo do IP 10.0.80.32, destinado ao IP 10.0.30.4

```
[root@johann /]# iptables -A FORWARD -s 10.0.80.32 -d
10.0.30.4 -j DROP
```

Exemplo 8:

- Proibir que qualquer pacote oriundo de a LAN (10.0.30.0/8) possa direcionar-se ao site www.sexo.com.br

```
[root@johann /]# iptables -A FORWARD -s
10.0.30.0/8 -d www.sexo.com.br -j DROP
```

Exemplo 9:

- Pacotes TCP destinados à porta 80 de nosso host firewall deverão ser descartados:

```
[root@johann /]# iptables -A INPUT -p tcp -dport 80 -j DROP
```

Exemplo 10:

- Fazer com que os pacotes destinado a porta 25 de nosso host Firewall sejam arquivados em um log (/var/log/messages)

```
[root@johann /]# iptables -A INPUT -p tcp -dport 25 -j LOG
[root@johann /]# iptables -A INPUT -p tcp -dport 25 -j LOG
[root@johann /]# iptables -A INPUT -p tcp -dport 25 -j
DROP
```

Habilitando o Controle de Estados das Conexões

Introdução

- Todo o fluxo de pacotes é registrado em uma tabela de estados
  - Essa característica garante ao firewall ser do tipo "statefull", ou seja acompanha os estados das conexões, que significa dizer que é capaz de tomar decisões baseadas nas entradas contidas nessa tabela.
- A tabela de estados fica em /proc/net/ip_conntrack
- A tabela de estados é manipulada na chain **PREROUTING** e **OUTPUT**
  - Quando o pacote entra para ser roteado e quando sai para ser roteado.
- Em regra geral, todo pacote é inserido na tabela de estados, a menos que se utilize a tabela raw com o jump NOTRACK.

Controle de Estados

- Utiliza 4 diferentes regras:
  - **NEW**: Essa regra causa match, ou seja, coincide com qualquer pacote que esteja entrando no host pela primeira vez, podendo ser ou não uma conexão TCP. Ao contrário do que muitos pensam, o NEW não se refere apenas a pacotes TCP-SYN mas a qualquer pacote que esteja iniciando uma comunicação com o host, podendo ser TCP, UDP ou ICMP.
  - **ESTABLISHED**: Essa regra é a mais simples de todas pois coincide com pacotes posteriores aos que iniciaram a comunicação. Enquanto o NEW permite a abertura de novas conexões, o ESTABLISHED permite a passagem dos pacotes posteriores.
  - **RELATED**: Essa regra casa com pacotes que iniciem uma nova conexão em alguma porta, porém que esteja relacionada com alguma outra conexão. Esse cenário é um tanto atípico já que poucos protocolos tem esse comportamento. Um bom exemplo seria o FTP que abre duas conexões em cada host e muitas vezes a porta da segunda conexão é aleatória, o que garante uma boa dor de cabeça para o administrador do firewall.
  - **INVALID**: O estado invalid casa com qualquer pacote que não consta na tabela de estados ou sem identificação e há muitas razões para isso sendo a melhor opção bloquear esses tipos de pacote:

Controle de Estados

- O que não se deve fazer

```
1  # Fechando as polices
2  iptables -P INPUT DROP
3  iptables -P OUTPUT DROP
4  # Abrindo a porta 80 para entrada
5  iptables -A INPUT -p tcp --dport 80 -j ACCEPT
6  # Abrindo a porta 80 para saída
7  iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT
```

Controle de Estados

- Forma correta:

```
# Fechando as polices
iptables -P INPUT DROP
iptables -P OUTPUT DROP
# Liberando que novas conexões se iniciem na porta 80
iptables -A INPUT -p tcp --syn --dport 80 -j ACCEPT
# Liberando conexões já estabelecidas entrem no firewall
iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
# Liberando conexões já estabelecidas saiam do firewall
iptables -A OUTPUT -m state --state ESTABLISHED -j ACCEPT
```

Aplicando NAT

NAT

- É **uma forma de mascaramento**.
- Muito utilizado em **roteadores**.
- Só que **desempenha função de encaminhamento** de pacotes (forwarding).
- Técnica útil quando se deseja colocar um servidor Web ou servidor de email, atrás de um Firewall, **usando-se IP's falsos**, com intuito de escondê-los contra invasões.

IPTables - Tabela NAT

- **Funções de um Firewall NAT**
  - **SNAT** (Source Nat)
    (tradução de endereço IP de origem)
  - **DNAT** (Destination NAT)
    (tradução de endereço IP de destino)
  - **Proxy Transparente**

SNAT

- O Firewall altera o **endereço IP** ou **porta de origem**, antes dos pacotes serem enviados.

- O Firewall pode enviar um pacote do host "A" ao host "B" e informar ao host "B" que tal pacote foi enviado pelo host "C".

SNAT

- Qualquer regra aplicada a **SNAT** utiliza-se somente da *chain* **POSTROUTING**.

- Antes de iniciarmos a manipulação de qualquer regra da Tabela NAT, tem-se que habilitar a função de re-direcionamento (forward) no kernel Linux:

```
>echo "1" > /proc/sys/net/ipv4/ip_forward
```

Exemplo 1: SNAT

```
# iptables -t nat -A POSTROUTING -s 10.0.3.1
-o eth1 -j SNAT -to 192.111.22.33
```

- Com IPTables informamos ao Netfilter que atribua à tabela NAT (`-t nat`) sob o chain (`POSTROUTING`) (os pacotes devem ser modificados após o tratamento de roteamento).
- Uma nova regra (`-A`) ao fim da lista.
- Qualquer pacote que tenha como origem o host 10.0.3.1 (`-s 10.0.3.1`) e que deve sair pela interface eth1 (`-o eth1`) deve ter seu **endereço de origem alterado** (`-j SNAT`) para 192.111.22.33 (`-to 192.111.22.33`).

Exemplo 2: SNAT

```
# iptables -t nat -A POSTROUTING -s 10.0.3.0/8 -o
eth0 -j SNAT -to 192.111.22.33
```

- Com IPTables informamos ao Netfilter que atribua à tabela NAT (`-t nat`) sob o chain (`POSTROUTING`) (os pacotes devem ser modificados após o tratamento de roteamento).
- Uma nova regra (`-A`) ao fim da lista.
- Qualquer pacote que tenha como origem o host 10.0.3.0/8 (`-s 10.0.3.1/8`) e que deve sair pela interface eth0 (`-o eth0`) deve ter seu **endereço de origem alterado** (`-j SNAT`) para 192.111.22.33 (`-to 192.111.22.33`).

Exemplo 3: SNAT

```
# iptables -t nat -A POSTROUTING -s 10.0.3.1 -o eth0
-j SNAT -to 192.111.22.33-192.111.22.66
```

- Com IPTables informamos ao Netfilter que atribua à tabela NAT (**`-t nat`**) sob o chain (**`POSTROUTING`**) (os pacotes devem ser modificados após o tratamento de roteamento).
- Uma nova regra (**`-A`**) ao fim da lista.
- Qualquer pacote que tenha como origem o host 10.0.3.1 (**`-s 10.0.3.1`**) e que deve sair pela interface eth0 (**`-o eth0`**) deve ter seu **endereço de origem alterado** (**`-j SNAT`**) para qualquer IP na faixa 192.111.22.33 à 192.111.22.66 (**`-to 192.111.22.33-192.111.22.66`**).

DNAT

- **Altera o endereço IP ou porta de destino**, dos pacotes que atravessam o Firewall, antes do pacote ser enviado ao seu destino final.

- Receber um pacote destinado à porta 80 do host "A" e encaminhá-lo à porta 3128 do host "B".

- Possibilita o desenvolvimento de:
  - **Proxies transparentes**,
  - **Balanceamento de carga**.

DNAT

- Usar somente o chain PREROUTING.

- Antes de iniciarmos a manipulação de qualquer regra da Tabela NAT, tem-se que habilitar a função de re-direcionamento (forward) no kernel Linux:

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

Exemplo 1: DNAT

```
# iptables -t nat -A PREROUTING -s 10.0.3.1
-i eth1 -j DNAT -to 192.111.22.33
```

- Com IPTables informamos ao Netfilter que atribua à tabela NAT (**`-t nat`**) sob o chain (**`PREROUTING`**) (os pacotes devem ser redirecionados logo que chegam).
- Uma nova regra (**`-A`**) ao fim da lista.
- Qualquer pacote que tenha como origem o host 10.0.3.1 (**`-s 10.0.3.1`**) e que entre pela interface eth1 (**`-i eth1`**) deve ter seu endereço de destino alterado (**`-j DNAT`**) para 192.111.22.33 (**`-to 192.111.22.33`**)

Exemplo 2: DNAT

```
# iptables -t nat -A PREROUTING -i eth0 -j
        DNAT -to 192.11.22.10-192.11.22.13
```

- Com IPTables informamos ao Netfilter que atribua à tabela NAT (-t nat) sob o chain PREROUTING (os pacotes devem ser redirecionados logo que chegam).
- Uma nova regra (-A) ao fim da lista.
- E que qualquer pacote que entre na interface eth0 (-i eth0), independente de quem o enviou deve ser automaticamente redirecionado aos hosts 192.11.22.10, 192.11.22.11, 192.11.22.12, 192.11.22.13 (`-to 192.11.22.10-192.11.22.13`) .

Exemplo 3: DNAT

```
# iptables -t nat -A PREROUTING -i eth2 -j
          DNAT -to 192.11.22.58:22
```

- Com IPTables informamos ao Netfilter que atribua à tabela NAT (-t nat) sob o chain PREROUTING (os pacotes devem ser redirecionados logo que chegam).
- Uma nova regra (-A) ao fim da lista.
- E qualquer pacote que entre na interface eth2 (**`-i eth2`**), independente de quem o enviou, deve ser automaticamente redirecionado ao host 192.11.22.58 (**`-to 192.11.22.58:22`**), e, independente da porta solicitada, deverá ser enviado à porta 22 (serviço SSH).

Proxy Transparente

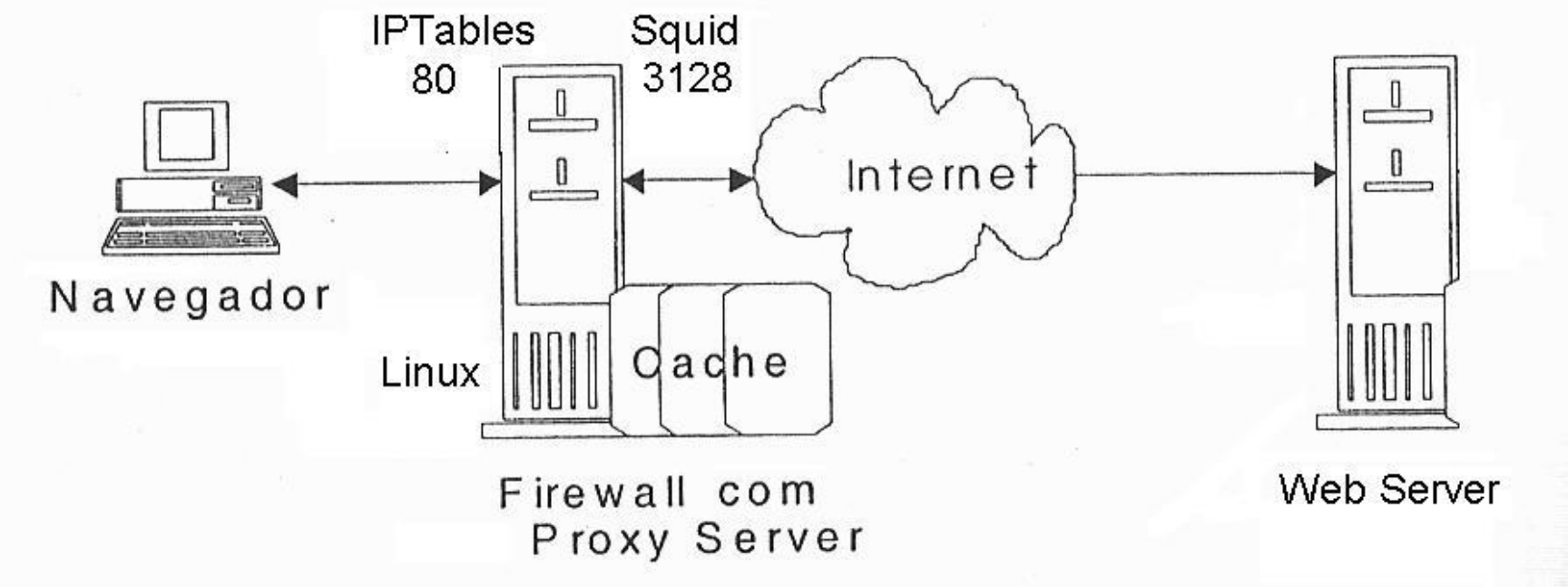
- Transparente: **"parece não existir, mas existe"**.
- **Redireciona portas em um mesmo host de destino**.
- **Não confundir com DNAT**, que altera o endereço de destino de pacotes de uma máquina A para uma máquina B, através do Firewall. Redireciona IP's.

Exemplo: Proxy-Cache Squid

- Squid tem por padrão disponibilizar consultas Web através da porta 3128, enquanto que a maioria dos clientes Web costumam realizar solicitações à porta 80 (padrão HTTP).

- Com Firewall IPTables + Squid numa mesma máquina Linux, o Proxy Transparente pode ser configurado.

Firewall + Proxy



Firewall como Proxy Transparente

```
# iptables -t nat -A PREROUTING -i eth0
-p tcp -dport 80 -j REDIRECT -to-port 3128
```

- Com IPTables informamos ao Netfilter que atribua à tabela NAT (-t nat) sob o chain PREROUTING (os pacotes devem ser redirecionados logo que chegam).
- Uma nova regra (-A) ao fim da lista.
- E qualquer pacote que entre na interface eth0 (`-i eth0`) e encaminhado à porta 80 (`-dport 80`) deve ser imediatamente redirecionado (`-j REDIRECT`) à porta 3128 deste mesmo host (`-to-port 3128`).