Capítulo 6

Controle de pacotes

Neste capítulo trataremos de técnicas de *firewall* para filtro de pacotes, roteamentos e geração de logs via *iptables*. Inicialmente, serão abordados alguns aspectos importantes sobre roteamento de pacotes, que deverão ser muito bem compreendidos.

Roteamento de pacotes

Roteamento é o processo de encaminhamento de pacotes, efetuado por um roteador, de uma rede a outra. Existem equipamentos específicos para essa atividade e soluções via software: veremos dicas sobre roteamento (simples) feito por um *firewall/gateway Linux*.

O ***default gateway*** corresponde ao roteador que conhece as redes fora do alcance de nossa rede local. Ao realizarmos algum tipo de comunicação, é verificado inicialmente se o *host* que desejamos alcançar é conhecido em nossa rede local – através de um sinal de *broadcast* (*sinal enviado a todos os hosts da rede*). Se não for encontrado, a solicitação será repassada ao *default gateway*, que é o roteador responsável pelas rotas às demais redes. O *default gw* se tornará o intermediário de uma comunicação quando for possível atender a uma determinada solicitação – dessa forma não será feita uma verificação, por sinal de *broadcast*, na rede local pacote a pacote.

Existem basicamente duas formas de roteamento:

- ***Rotas estáticas***: em rotas estáticas serão predefinidos os caminhos necessários para atingir determinados *hosts* e/ou determinadas redes a tabela de roteamento é fixa. O caminho (rotas) para alcançar um determinado nó ou rede é adicionado manualmente pelo administrador, e o roteador se baseia nesta tabela que permanecerá estática até que o sistema seja reiniciado ou caso o administrador remova algumas rotas.
- ***Rotas dinâmicas***: é um tipo de *roteamento* "inteligente", efetuado normalmente por *daemons* e/ou *protocolos* (*como routed ou zebra*) especiais de *roteamento*. O roteador conhece uma ou mais rotas para um mesmo nó ou rede, e escolhe qual será utilizará.

> **Obs.:** O foco do livro não é roteamento, veremos apenas princípios básicos.

- **Definindo rotas estáticas no Linux**

- configuração de rede em **gateway xyz-net**

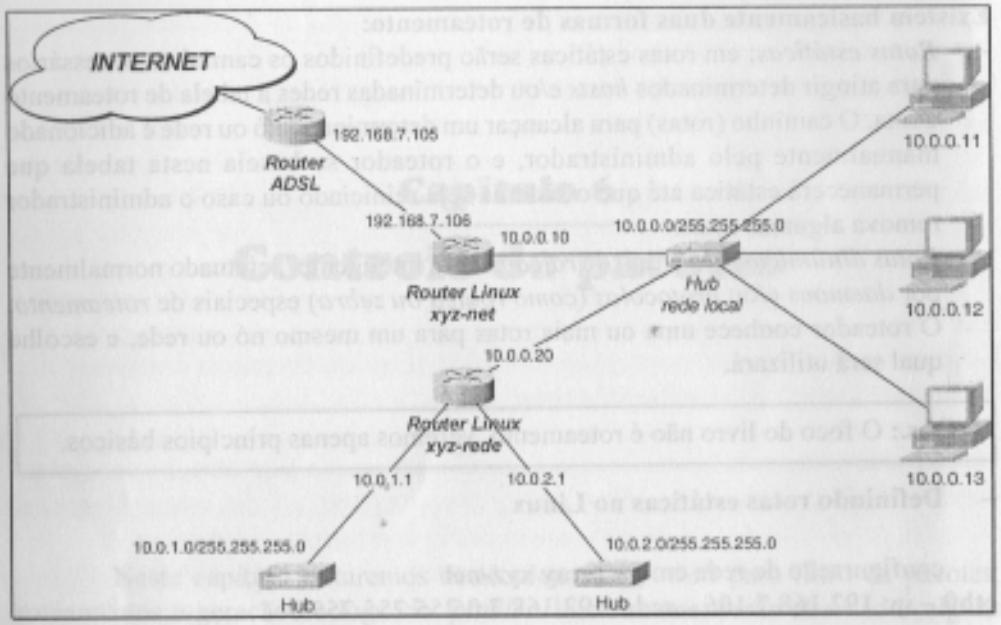
eth0 – ip: **192.168.7.106** – rede: 192.168.7.0/255.255.255.0
eth1 – ip: **10.0.0.10** – rede: 10.0.0.0/255.255.255.0

- configuração de rede em **gateway xyz-rede**

eth0 – ip: **10.0.0.20** – rede: 10.0.0.0/255.255.255.0
eth1 – ip: **10.0.1.1** – rede: 10.0.1.0/255.255.255.0
eth2 – ip: **10.0.2.1** – rede: 10.0.2.0/255.255.255.0

Supondo que o *gateway xyz-net* seja o responsável pelo acesso à *Internet* de todos *hosts* da rede local *10.0.0.0*. No entanto, os *hosts* da rede 10.0.0.0 também necessitam acessar servidores nas redes 10.0.1.0 e 10.0.2.0. Logo de imediato podemos perceber que existem dois *gateways* envolvidos (*gw xyz-net* para acesso à *Internet*, e *gw xyz-rede* para acessar as redes 10.0.1.0 e 10.0.2.0), mas "*só poderemos indicar um gw*" como *default gateway*. A primeira impressão é que as estações da rede 10.0.0.0 farão escolha entre acessar à *Internet* ou à rede de servidores – no entanto, existem algumas soluções de roteamento extremamente simples de serem implementadas.

Como desejamos que todos os clientes da rede 10.0.0.0 acessem a *Internet*, definiremos (nos clientes) como *default gateway* o *host* de *ip* 10.0.0.10, e incluiremos as rotas para as demais redes neste *gateway (xyz-net-10.0.0.10)*:



a) rotas adicionadas ao gateway xyz-net

route add -net 10.0.1.0 netmask 255.255.255.0 gw 10.0.0.20
route add -net 10.0.2.0 netmask 255.255.255.0 gw 10.0.0.20

Obs.: Quando uma solicitação for destinada às redes 10.0.1.0 e 10.0.2.0, o gateway (gw) de acesso à Internet repassará, via roteamento estático, a solicitação ao gateway 10.0.0.20 que conhece tais redes.
Outra solução, menos eficiente, seria a adição das rotas para as redes 10.0.1.0 e 10.0.2.0 em cada estação da rede 10.0.0.0, ao invés de adicioná-las em gw 10.0.0.10. O inconveniente é que as rotas deverão ser adicionadas a todas as estações (uma a uma). Em compensação, se o gw 10.0.0.10 "falhar" as estações continuarão acessando os servidores desejados porque a rota é conhecida localmente. O ideal é adicionar estas rotas em 10.0.0.10 e adicionar as rotas, localmente, para estações onde a comunicação com os servidores não fique na dependência do gw de Internet.

b) as estações na rede 10.0.0.0 deverão definir o host 10.0.0.10 como default gateway:

route add default gw 10.0.0.10 dev eth0 ou
route add default gw 10.0.0.10

– supondo que o roteador para a internet seja 192.168.7.105, em gateway-net o default gateway será definido da seguinte maneira:

route add default gw 192.168.7.105

Obs.: Em nosso router Linux (gw 10.0.0.10), recomendo que a rota para o default gw – que é o roteador para acesso à Internet do seu provedor de acesso – seja definida no arquivo de configuração de rede /etc/sysconfig/network. Edite este arquivo e defina GATEWAY=192.168.7.105 – se necessário indique, também, a interface de rede utilizando a variável GATEWAYDEV, por exemplo.

– quando for desejado adicionar rota a um host em específico da rede 10.0.2.0:

route add -host 10.0.2.10 gw 10.0.0.20

De forma resumida, a sintaxe do comando route é:

route add/del -host/-net <ip ou rede> gw/dev <ip ou dispositivo de rede>

Ao adicionar uma rota a determinada rede é preciso acrescentar o "netmask".

Listando e removendo rotas

Ao digitar ***route***, obtemos uma saída como esta:

Tabela de Roteamento IP do Kernel

Destino	*Roteador*	*MáscaraGen.*	*Opções*	*Métrica*	*Ref*	*Uso*	*Iface*
192.168.7.0	*	*255.255.255.0*	*U*	*0*	*0*	*0*	*eth0*
127.0.0.0	*	*255.0.0.0*	*U*	*0*	*0*	*0*	*lo*
default	*192.168.7.105*	*0.0.0.0*	*UG*	*0*	*0*	*0*	*eth0*

Se quisermos remover a rota para o default gateway:

route del default gw 192.168.7.105 dev eth0 ou
route del default gw 192.168.7.105

A sintaxe pode ser exatamente a mesma utilizada no "*add*", trocando apenas por "*del*".

Para maiores detalhes, digite: man route .

Filtro de pacotes e roteamento (NAT)

Filtrando pacotes com iptables

A partir da família 2.4.x do *kernel* o controle de pacotes passou a ser implementado pelo módulo *netfilter* e, através do programa/módulo *iptables*, é que determinaremos as regras de acesso. O *ipchains* (utilizado na família 2.2.x) pode ser utilizado em modo de compatibilidade, mas não acredito que seja vantajoso em um *kernel 2.4*.

Recomenda-se não utilizar versões de *kernel* inferiores a 2.4.18 para configurar um *firewall* baseado em *iptables* – existem pequenas falhas que poderão tornar equivocada a nossa interpretação em relação a determinadas regras (acredito que se trate de pequenos *bugs*). É desejável utilizar uma versão de *kernel* igual ou superior a ***2.4.18***.

a) Principais vantagens

– **Filtro de pacotes statefull:**

A grande novidade no *iptables* é a capacidade de atuar sobre as camadas do protocolo *TCP*. Podemos definir regras baseadas em status de conexão: ***nova*** (***NEW***), ***estabelecida*** (***ESTABLISHED***) ***reincidente*** (***RELATED***) e ***inválida*** (***INVALID***).

– **Extremamente modular:**

Existem módulos que permitem tratar regras baseadas em status de conexão, endereço *MAC*, múltiplas portas, pacotes com má formação, etc. "*Novos módulos podem ser desenvolvidos e facilmente adicionados à ferramenta*".

b) Principais desvantagens

Como o *iptables* é um filtro de pacotes "novo", muitas pessoas ainda reagem quando se propõe um *firewall* desse porte. Alguns acham complexo. No entanto, a sintaxe é muito parecida com a do *ipchains* – e portar um script de *firewall* escrito para *ipchains* não é algo muito complexo.

Fluxo de pacotes

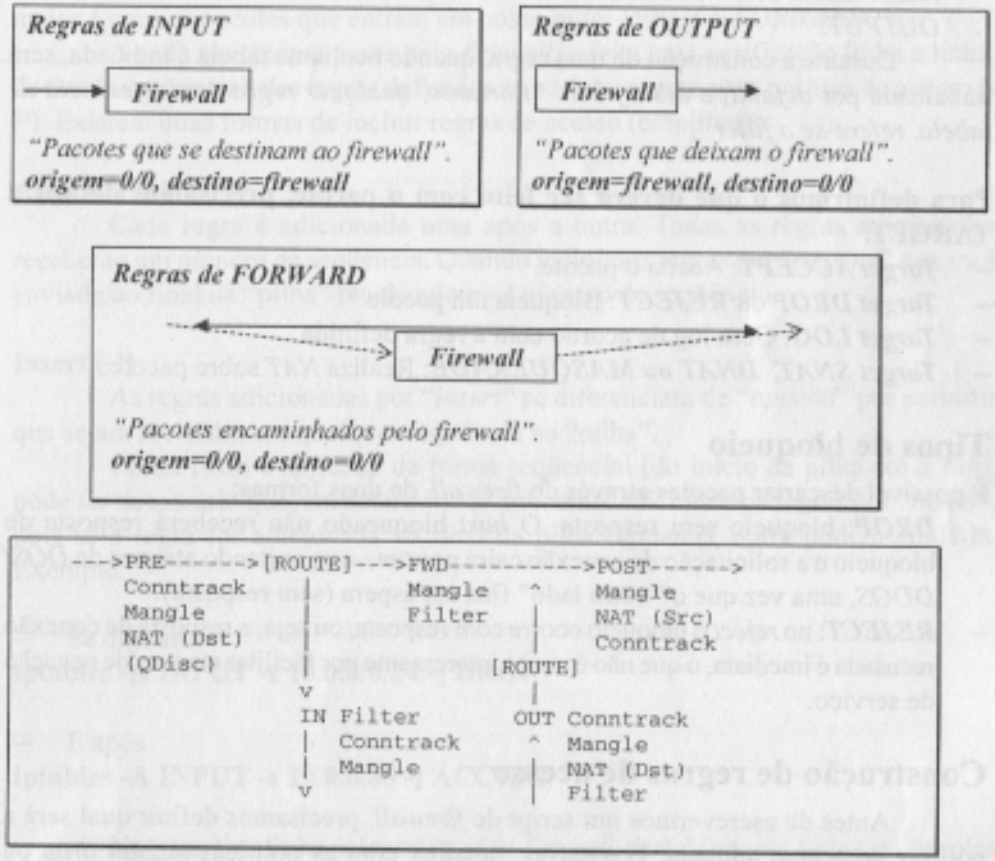
– ***Chain INPUT***: quando for desejado definir regras de acesso onde o ***destino*** da conexão for o próprio *firewall*, devemos criar regras de *INPUT*.

– ***Chain OUTPUT***: utilizado para tratar pacotes que deixam o *firewall*. Para ser uma *chain output*, a solicitação deve partir do *firewall* (origem da comunicação).

– ***Chain FORWARD***: regras de *FORWARD* se aplicam a pacotes que são encaminha-dos de uma rede a outra pelo *firewall*. Regras de *FORWARD* não implicam em *INPUT* e *OUTPUT* – são distintas (***lembre-se disso***).

– ***Chain PREROUTING***: tratar roteamento de pacotes que chegam ao *firewall*.

– ***Chain POSTROUTING***: tratar roteamento de pacotes que deixam (saem) o *firewall*.



A manipulação dessas chains dependerá da tabela (TABLE) escolhida:

– ***Table filter***: Esta tabela é *default* (não é necessário indicá-la na regra), sendo utilizada normalmente sob as *chains* de *INPUT*, *OUTPUT* e *FORWARD* para definir filtros de acesso (***aplica-se a qualquer chain***).

- ***Table mangle***: Nesta tabela podemos tratar pacotes antes que o roteamento ocorra. É possível aplicar *targets* de *ACCEPT ou DROP/REJECT* para filtros especiais, ou criar *marcas* (*MARK*) de *firewall* (utilizaremos este recurso para controlar banda). Aplica-se sob as *chains* de *PREROUTING* e *OUTPUT* (em versões de *kernel* mais atuais, como 2.4.18, é possível trabalhar com esta tabela para todas as *chains*).
- ***Table nat***: Necessária para realização do *NAT* (*Network Address Translation*). É com esta tabela que realizamos ou tratamos todos os roteamentos (*NAT*) desejados. Nesta tabela serão trabalhadas as *chains PREROUTING, POSTROUTING e OUTPUT*.

Durante a construção de uma regra, quando nenhuma tabela é indicada, será trabalhada por *default*, a *table filter*. *"Portanto, qualquer regra sem referência de tabela, refere-se a filter"*.

Para definirmos o que deverá ser feito com o pacote, precisamos definir a TARGET:

- ***Target ACCEPT***: Aceita o pacote.
- ***Target DROP*** ou ***REJECT***: Bloqueia um pacote.
- ***Target LOG***: Gera log de acordo com a regra definida.
- ***Target SNAT, DNAT ou MASQUERADE***: Realiza *NAT* sobre pacote.

Tipos de bloqueio

É possível descartar pacotes através do *firewall*, de duas formas:

- ***DROP***: bloqueio sem resposta. O *host* bloqueado não receberá resposta de bloqueio e a solicitação de conexão cairá por *time-out*, evitando ataques de *DOS/DDOS*, uma vez que o "outro lado" fica em espera (sem resposta).
- ***REJECT***: no *reject* o bloqueio ocorre com resposta, ou seja, a resposta de conexão recusada é imediata, o que não é muito interessante por facilitar ataques de negação de serviço.

Construção de regras de acesso

Antes de escrevermos um script de *firewall*, precisamos definir qual será a política de acesso adotada. É possível trabalhar com as políticas *default drop* ou *default accept*. Quando trabalhamos com a política *default drop*, o que não for explicitamente liberado no script é considerado bloqueado (por *drop*). Já no *default accept*, o que não for bloqueado no script é aceito (*accept*).

```
# Políticas de acesso
# Parâmetro -P (policy)
```



```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

Nas linhas anteriores definimos para *INPUT* e *FORWARD* a política *DROP*, e *ACCEPT*, para *OUTPUT*. A política de acesso se aplica a cada *chain*. Todas as *chains* são *accept* por *default*. Como os fluxos de *INPUT* (pacotes destinados ao *firewall*) e *FORWARD* (pacotes encaminhados – roteados) são os mais críticos, é comum aplicar *default drop* a eles, e *default accept* ao *OUTPUT*. A preocupação maior é com os pacotes que entram em nossa rede: *INPUT* e *FORWARD*.

A cada pacote que passa pelo *firewall*, é feita uma verificação linha a linha, de forma seqüencial, das regras definidas no script – exceto para política de acesso (*-P*). Existem duas formas de incluir regras de acesso (empilhar):

Append (-A)

Cada regra é adicionada uma após a outra. Todas as regras adicionadas receberão um número de seqüência. Quando incluímos regras por "*append*" a regra é enviada ao final da "pilha" recebendo *n+1* número de seqüência.

Insert (-I)

As regras adicionadas por "*insert*" se diferenciam de "*append*" por permitir que sejam incluídas em qualquer seqüência na "pilha".

Como filtro é aplicado de forma seqüencial (do início da pilha até o fim), pode ser necessário que, em determinado momento, adicionemos regras por "*insert*", pois se a regra for adicionada ao final da pilha (*append*), outra poderá anulá-la. Exemplo:

- Se digitarmos

```
iptables -A INPUT -s 10.0.0.0/24 -j DROP
```

- E após

```
iptables -A INPUT -s 10.0.0.88 -j ACCEPT
```

Na primeira linha informamos ao *firewall* para bloquear todos os pacotes vindos da rede *10.0.0.0/24* e, logo após, permitimos o acesso ao *host* 10.0.0.88. Como as regras foram adicionadas por "*append*" (**-A**), o *host* 10.0.0.88 será bloqueado, pois o pacote vindo do *host* 10.0.0.88 pertence à rede 10.0.0.0. Ou seja, o bloqueio será feito antes mesmo que a linha seguinte seja verificada. ***Portanto, muita atenção na seqüência como as regras forem definidas.***

Como adotamos uma política *default drop* para *INPUT*, a primeira coisa a fazer é abrir o acesso a pacotes em *loopback* (conexões locais). ***Este procedimento não é opcional!***

iptables -A INPUT -i lo -j ACCEPT

Existem diversos recursos do sistema que trabalham sob a interface *loopback*, portanto nunca a deixe bloqueada, pois certamente, surgirão problemas.

Passos para criar uma regra no iptables:

1. escolher qual será a tabela que trabalharemos (*filter, nat ou mangle*)
   iptables -t <filter/nat/mangle> ...
2. escolher uma *chain* (*INPUT, FORWARD e etc.*) ou criar uma nova *chain*
   iptables -t filter -A <INPUT/FORWARD...> ...
   iptables -t filter -N nova_chain ...
3. se necessário, indicar a interface de entrada (***input***) e/ou saída (***output***)
   iptables -A INPUT -i <lo/eth0/eth1...> ...
   iptables -A FORWARD -i eth0 -o eth1 ...
   "Como não foi indicada uma tabela, a regra adotará a filter (por default)".
4. se necessário, indicar de onde vem (***source***) e para onde vai (***destination***) o pacote
   iptables -A INPUT -i eth0 -s 10.0.0.10 ...
   iptables -A FORWARD -s 10.0.0.10 -d 200.199.xxx.yyy ...
5. se necessário, definir a qual porta de acesso (de um serviço) se aplica a regra
   iptables -A INPUT -s 10.0.0.0/24 -p tcp —dport 22 ...
6. escolher a *target* (*ACCEPT,DROP, REJECT* e etc.)
   iptables -A INPUT -s 10.0.0.88 -p tcp —dport 22 -j ACCEPT

Principais parâmetros:

–A: fazer append de uma chain
–I: fazer insert de uma chain
–N: criar uma nova chain
–F ou -X: elimina todas as regras
–s: define a origem do pacote

–d: define o destino do pacote
–i: interface de input (entrada)
–o: interface de output (saída)
–p: seleciona protocolo (tcp, udp, icmp...)
—dport: define a porta de destino da conexão
—sport: define a porta da origem da conexão
–j: o que será feito com o pacote (target)

Principais módulos do iptables:

O *netfilter* é, atualmente, o filtro de pacotes mais inteligente disponível para *Linux*. Por ser extremamente modular, podemos contar com módulos especiais para diferentes fins e novos módulos são facilmente escritos e disponibilizados – ou seja, uma nova funcionalidade pode ser adicionada escrevendo-se um novo módulo.

Em geral, a chamada a um módulo é feita da seguinte maneira:
–m <modulo> <—modulo> <valor>

a) Para tratar conexões cujo estado de conexão se apresenta como "estabelecido" ou "reincidente"
–m state —state ESTABLISHED,RELATED

Aqui é possível trabalhar com os estados:
NEW: conexões novas
ESTABLISHED: conexões estabelecidas
RELATED: conexões reincidentes (comum em serviços de ftp, por exemplo)
INVALID: conexões inválidas

A combinação dos estados, seguido de vírgulas, corresponde a um "ou" lógico. Ou seja:
iptables -A INPUT -m state —state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -o eth1 -m state —state NEW,INVALID -j DROP

Na primeira linha, qualquer conexão ***iniciada pelo firewall*** será ***permitida*** (*"só fale comigo quando eu falar contigo"*), seja uma conexão *estabelecida* ou *reincidente*. Na segunda linha, será feito bloqueio de pacotes novos ou inválidos que seriam encaminhados/roteados (*FORWARD*) à rede local (saindo pela *eth1*).

b) Aplicando regras baseadas em intervalos de tempo (muito utilizado em logs)
–m limit —limit 1/h —limit-burst 3

"No exemplo anterior o filtro se aplica a uma verificação por hora, para os três 'primeiros' pacotes".

É comum trabalhar com este módulo para limitar a freqüência com que *logs* serão gerados, principalmente para evitar ataques de *flood* (inundação) – assim evitamos que os *logs* sejam "engordados" com mensagens repetidas em curto intervalo de tempo.

c) Para trabalhar com endereço MAC
–m mac —mac 00:D0:09:01:02:03

d) Módulo de segurança (descarta pacotes com má formação/irregulares)
–m unclean

Este módulo fornece um nível a mais de segurança. Bloqueia pacotes com má formação, tamanho zero e etc., evitando ataques de *DoS*. No entanto, provoca um *overhead* a mais ao sistema e vários "alarmes falsos".

e) Para múltiplas portas de acesso
–m multiport -p <tcp/udp> —port <p1,p2,p3...pn>

Obs.: Não se preocupe em carregar os módulos do iptables (por modprobe) pois serão carregados quando referenciados.

```
###############  EXEMPLO  ##########################
#  Eliminando regras correntes - que estão em memória
#  -F para remover regras da tabela filter (chains comuns)
#  -X para remover regras de chains criadas pelo usuário

iptables -F
iptables -X

#####################################################
#  Defindo a política de acesso

iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```



```
#####################################################
#  Permitindo acessos em loopback

iptables -A INPUT -i lo -j ACCEPT

#####################################################
#  Módulos especiais
#  Permitindo conexões iniciadas pelo firewall (estabelecida /
reincidente)
#  Descartando qualquer pacote inválido (tamanho irregular e
etc)

iptables -A INPUT -m state –state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -m unclean -j DROP

#####################################################
#  Permitindo acesso SSH ao host 10.0.0.30

iptables -A INPUT -i eth0 -s 10.0.0.30 -p tcp –dport 22 -j
ACCEPT

#####################################################
#  Ou permitindo acesso SSH e FTP ao host 10.0.0.18

# iptables -A INPUT -s 10.0.0.18 -p tcp -m multiport –port
21,22 -j ACCEPT

#####################################################
#  Bloqueando sinais de ping

iptables -A INPUT -p icmp –icmp-type echo-request -j DROP
```

O script anterior é simples e seguro. É baseado em uma política *default drop*, e será suficiente para usuários domésticos. Trata-se de um script extremamente restrito, o único acesso permitido será por *ssh* e vindo do *host* 10.0.0.30 na interface *eth0*. Este exemplo poderia se enquadrar em uma rede *crossover* (apenas dois micros), com acesso discado à *Internet*.

Preferencialmente, escreva o script em */etc/rc.d/rc.firewall* e atribua permissão de execução ao script (***chmod u+x /etc/rc.d/rc.firewall***).

O script será inicializado automaticamente no *boot*, mas, se necessário, podemos executá-lo a qualquer momento, digitando: */etc/rc.d/rc.firewall* .

Obs.: Regras de forward serão tratadas no item referente a NAT.

Para listar as regras de firewall

– Listando regras de todas as chains de uma determinada tabela
iptables -L
iptables -t nat -L
iptables -t mangle

– Listando apenas para uma determinada chain
iptables -L INPUT
iptables -t nat -L POSTROUTING

– Listagem detalhada (fornece quantidade de dados recebidos e enviados)
iptables -L -v

– Listagem com número de seqüência
iptables -L —line-numbers
iptables -t nat -L POSTROUTING —line-numbers

> ☞ **ATENÇÃO!**
>
> ***Não construa*** um *firewall/gateway* com versão de ***kernel inferior a 2.4.18***, pois algumas versões estão apresentando *bugs* gravíssimos (principalmente a *2.4.5*). Recomendo trabalhar com uma versão igual ou superior a 2.4.18.

Para remover regras de acesso

a) Baseando-se em uma regra de input ou append
Para a regra
iptables -A INPUT -i eth0 -s 10.0.0.30 -p tcp —dport 22 -j ACCEPT

A remoção será
iptables -D INPUT -i eth0 -s 10.0.0.30 -p tcp —dport 22 -j ACCEPT

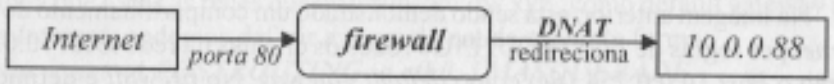
b) Baseado no número da regra
Localize a regra desejada
iptables -L INPUT —line-numbers

A remoção será
iptables -D INPUT <número>

NAT – Network Address Translator

– ***PREROUTING*** (para tratar pacotes que chegam ao *firewall*)
Regras de *PREROUTING* são definidas, normalmente, quando necessitamos realizar qualquer tipo de redirecionamento de conexão. É possível fazer de duas formas:

a) ***DNAT (DestinationNAT)***: Redireciona conexões a um *host* qualquer.
b) ***REDIRECT***: Redireciona conexões, apenas, a uma porta qualquer no *firewall*.



Redirecionando conexões, vindas da *Internet*, destinadas à porta 80 do *firewall* para o um servidor de *Web Apache* cujo endereço *ip* equivale a 10.0.0.88.

Se um servidor Proxy de *HTTP* (como *Squid*) estivesse configurado no *firewall*, para controlar os acessos de *Internet (navegação)*, poderíamos fazer redirecionamento de todas as solicitações na porta 80 (*HTTP*) para a porta do *Proxy* (*3128*, porta *default* no *Squid*). Esta técnica é conhecida como *Transparent-Proxy*, pois os clientes serão redirecionados ao *Proxy* automaticamente.

– ***POSTROUTING*** (para tratar pacotes que deixam o *firewall*)
Regras de *POSTROUTING* são adotadas quando necessitamos alcançar um *host* qualquer fora de nossa rede (ou seja, através de um *gateway*), mas o *gateway* definirá qual será a "origem" desta conexão.

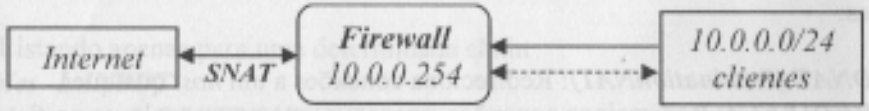
Exemplo:
Host "A": 10.0.0.10
Default Gw (firewall) "GW": 10.0.0.254

Se "A" fizer ping em um endereço IP qualquer, na Internet, o default "GW" repassará a solicitação de 10.0.0.10 a quem de destino, no entanto fará a troca do endereço de origem (10.0.0.10) para 10.0.0.254. Em outras palavras, "A" fará o ping através do endereço ip de "GW".

Regras de *POSTROUTING* podem ser escritas de duas formas:
a) ***MASQUERADE***: o *IP* de origem, da conexão, será sempre o *IP* da interface de rede do *gateway* (*firewall*) que permite alcançar o *host/rede* de destino. Esta *target* é muito utilizada para compartilhar acessos à *Internet;* assim, permitimos

a navegação de *Internet* à toda rede interna, utilizando apenas um único endereço *IP* (do *gateway*).

b) ***SNAT***: é uma *target* similar ao *masquerade*, porém o *IP* de origem precisa ser especificado. O *gateway* encaminhará pacotes ao *host* de destino utilizando o endereço *IP* especificado na regra.



Na imagem anterior, está sendo demonstrado um compartilhamento de acesso à *Internet* por regras de *POSTROUTING*. Todos os clientes na rede local 10.0.0.0/24 definirão o *host* 10.0.0.254 (*fw*) como *default gateway*. No *firewall* é definida uma regra de *masquerade* para a rede 10.0.0.0/24. Quando qualquer cliente da rede local solicitar acessos de *Internet*, seu endereço *ip* será traduzido para 10.0.0.254 antes que o pacote deixe o *firewall*, com destino à *Internet*. A resposta aos clientes é roteada (encami-nhada), automaticamente pelo *firewall* (sentido *firewall – clientes*).

Exemplos:

– configuração de rede no **gateway/firewall xyz:**

eth0 – ip: 10.0.0.139 – rede: 10.0.0.0/24 (10.0.0.138 router Internet)
eth1 – ip: 192.168.1.1 – rede: 192.168.1.0/24
eth2 – ip: 192.168.2.1 – rede: 192.168.2.0/24

– configuração de rede no **servidor de http**

eth0 – ip: 192.168.1.10

– configuração de rede no **cliente abc da rede local:**

eth0 – ip: 192.168.2.10

a) Se desejarmos que todos os hosts da rede 192.168.2.0/24 acessem a Internet, precisamos definir no gw/fw a seguinte regra:

iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -j MASQUERADE
ou
iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -j SNAT —to 10.0.0.139

Desta forma, o repasse de pacotes do *gw/fw* para rede 192.168.2.0/24 adotará como "*IP de origem*" o *IP* da interface que permite conexão com o *host* de destino. Da forma como a regra foi definida, os *hosts* da rede 192.168.2.0 poderão alcançar a *internet* e também a rede 192.168.1.0. No entanto, a permissão de acesso ainda não foi concebida – a permissão é feita por regras de *FORWARD*.

iptables -A FORWARD -o eth0 -s 192.168.2.0/24 -j ACCEPT
iptables -A FORWARD -o eth2 -d 192.168.2.0/24 -m state —state \
ESTABLISHED,RELATED -j ACCEPT

No exemplo acima permitimos o repasse de pacotes vindos da rede 192.168.2.0, destinados à Internet (eth0 – interface de rede com roteador de Internet). A construção da regra não permitirá a comunicação com a rede 192.168.1.0 porque na linha onde definimos o acesso da rede 192.168.2.0 restringimos o forward, apenas, à interface de rede eth0. **Não quebre a linha quando digitar a regra.**

b) Para que todas as redes que utilizam "gw/fw xyz" como default gateway acessem a Internet, podemos definir a regra de nat da seguinte forma:
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

Como *eth0* é a interface de rede com a *Internet*, definiremos que qualquer solicitação, partindo de qualquer *rede/host*, com destino à *Internet* será mascarada.

– **Exemplo de uma conexão mascarada:**

Se o host 192.168.2.10 fizer um ping a 200.199.111.111, a solicitação real será:

 # De 192.168.2.10 a 200.199.111.111 (echo-request)
1. source = 192.168.2.10, destination = 10.0.0.139
2. source = 10.0.0.139, destination = 200.199.111.111

 # Retorno do ping (echo-replay)
3. source = 200.199.111.111, destination = 10.0.0.139
4. source = 10.0.0.139, destination = 192.168.2.10

c) Para redirecionar conexões destinadas ao gw/fw na porta 80 para o servidor de web Apache em 192.168.2.10, definiríamos a regra da seguinte maneira:
iptables -t nat -A POSTROUTING -o eth2 -p tcp —dport 80 -j MASQUERADE
iptables -t nat -A PREROUTING -i eth0 -p tcp —dport 80 -j DNAT —to 192.168.2.10

O argumento "*-o eth2*" é definido para indicar que o *gw/fw* deve mascarar qualquer conexão onde o fluxo de pacotes se destine (*output*) a redes atrás da interface *eth2*.

Como alguns serviços apresentam particularidades de rede (como o tipo de protocolo ou mecanismos de segurança, por exemplo), existirão casos em que deveremos mascarar as conexões até o *host* servidor, ao qual se destina nosso redirecionamento, como foi demonstrado nas duas regras anteriores.

d) Se desejássemos redirecionar as conexões de ssh (secure shell) para o servidor 192.168.2.10:

iptables -t nat -A PREROUTING -i eth0 -p tcp —dport 22 -j DNAT —to 192.168.2.10

Supondo que o servidor de ssh em 192.168.20.10 foi configurado para responder na porta 2200 (escuta em 2200), podemos redirecionar da seguinte maneira:

***iptables -t nat -A PREROUTING -i eth0 -p tcp —dport 22 -j DNAT ***
—to 192.168.2.10:2200

"Quando o gw/fw receber uma solicitação de conexão na porta 22, será feito um redirecionamento para 192.168.20.10 na porta 2200. Lembrando que a quebra de linha só deve ser feita se for digitado '\' ao final da linha".

e) Para redirecionar conexões entre determinadas portas, usaremos a target REDIRECT (muito comum para realizar proxy transparente):

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

Neste exemplo, assume-se que o servidor *proxy* está escutando na porta 3128 e rodando na mesma máquina em que se configurou a regra.

Script completo de firewall para compartilhar acesso à Internet

```
# Fazendo flush
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X

# Variáveis do firewall
CONFIAVEL1=10.0.0.10
CONFIAVEL2=10.0.0.88
REDEMASQ=10.0.0.0/24

# Política de acesso
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# Habilitando a proteção contra "TCP SYN Cookie"
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```



```
# Ocultando a rota de origem dos pacotes (evita spoof)
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done

# Evita ataques de spoof (na rede interna)
# Em alguns casos de roteamento esta linha deverá ser desativada
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done

# Permissões de acesso ao firewall
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state —state ESTABLISHED,RELATED -j ACCEPT

# Trocando o intervalo de portas locais
PORT_INI=61000
PORT_FIM=65095
echo $PORT_INI $PORT_FIM > /proc/sys/net/ipv4/ip_local_port_range

# Criando perfil de acesso administrativo (apenas por SSH)
iptables -N ADMIN
iptables -A ADMIN -i eth1 -p tcp —dport 22 —syn -j LOG —log-
level info \
        —log-prefix "[Acesso Admin]: "
iptables -A ADMIN -i eth1 -p tcp —dport 22 —syn -j ACCEPT

# Aplicando a permissão de acesso ssh a CONFIAVEL1 e 2
iptables -A INPUT -i eth1 -s $CONFIAVEL1 -j ADMIN
iptables -A INPUT -i eth1 -s $CONFIAVEL2 -j ADMIN

# Para tratar as quedas dos roteadores ADSL
GW='route | grep default | awk '{print $2}''
iptables -A INPUT -i eth0 -s $GW -p tcp ! —syn -j ACCEPT

# Compartilhando o acesso à Internet
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -s $REDEMASQ -o eth0 -j MASQUERADE

# Permitindo requisições de DNS
# iptables -A FORWARD -o eth0 -s $REDEMASQ -p tcp —dport 53 -j ACCEPT
iptables -A FORWARD -o eth0 -s $REDEMASQ -p udp —dport 53 -j ACCEPT
```

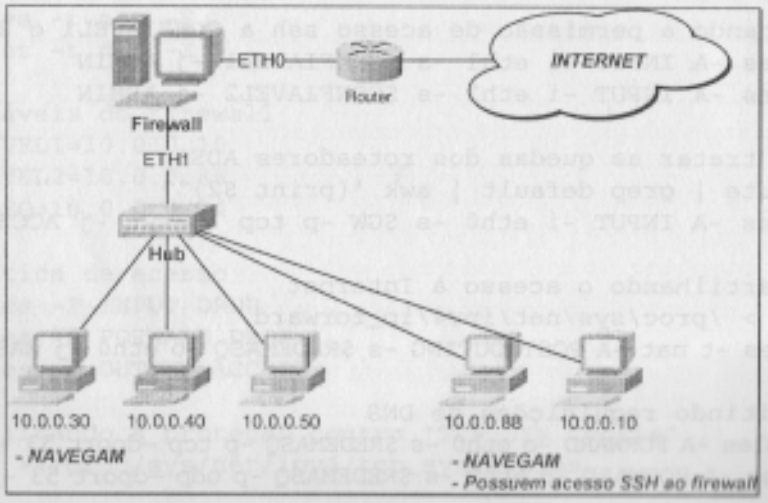
```
# Bloquear pacotes com estado "novo/inválido" que saiam pela
eth1
# Permitir pacotes com estado "estabelecido/reinicidente"
iptables -A FORWARD -o eth1 -m state –state NEW,INVALID -j DROP
iptables -A FORWARD -o eth1 -m state –state ESTABLISHED,RELATED
\
                -j ACCEPT

# Permitindo qualquer solicitação de Internet que parta da rede
10.0.0.0/24
iptables -A FORWARD -i eth1 -s $REDEMASQ -j ACCEPT

# Bloqueia qualquer pacote (forward) que não liberado por
regras anteriores
iptables -A FORWARD -j DROP

# Bloqueando e logando sinais de ping ao firewall
iptables -N LPI
iptables -A LPI -m limit –limit 5/hour –limit-burst 3 -p icmp
–icmp-type echo-request \
                -j LOG –log-level info –log-prefix "Log ping: "
iptables -A LPI -j REJECT –reject-with icmp-port-unreachable
iptables -A INPUT -p icmp –icmp-type echo-request -j LPI

# Bloqueia qualquer pacote (input) que não liberado por regras
anteriores
iptables -A INPUT -j DROP
```



O script anterior permite a dois *hosts* da rede interna acesso ao *firewall* via *ssh*.

Compartilha o acesso à *Internet* (interface *eth0*) para rede local (interface *eth1*). Sinais de *ping* serão bloqueados e logados em uma seqüência de 3 pacotes (com sinais de "*echo-request*" – *ping*), logados 5 vezes por hora.

Foram ativadas proteções *anti-Spoof* e *SynCookie*. Nenhum *host* possui permissão para iniciar conexões com o *firewall*, exceto os definidos nas variáveis *CONFIAVEL1* e *2* para conexões remotas por *SSH* (*Secure Shell*). Conexões, novas ou inválidas, destinadas à rede interna serão bloqueadas.

O forward na porta 53 (*DNS*) está permitido apenas para o protocolo *UDP*, que é utilizado nas consultas de *DNS* (*resolvedor de nomes*). Assim, os clientes poderão trabalhar com os servidores de *DNS* que desejarem. Na linha anterior é possível verificar que a permissão para *TCP* está comentada (ou seja, sem permissão), pois a porta *53/tcp* é utilizada na comunicação entre os servidores de *DNS*.

A finalização *DROP* para *INPUT* e *FORWARD*, ocorre como método preventivo. Assim, novas regras não poderão ser adicionadas por *Append*.

> **Obs.:** Para trabalhar com PREROUTING/POSTROUTING deveremos nos certificar de que o roteamento(ip_forward) de pacotes está ativo.

Podendo ser feito de diferentes formas:

1. Via script de firewall:
   echo 1 > /proc/sys/net/ipv4/ip_forward
2. Em /etc/sysconfig/network
   FORWARD_IPv4=true
3. Em /etc/sysctl.conf
   net.ipv4.ip_forward=1
   Por linha de comando < ***sysctl -w net.ipv4.ip_forward=1*** >

– Para ativar proxy transparente, adicione logo abaixo da linha "iptables -t nat -A POSTROUTING -s $REDEMASQ -o eth0 -j MASQUERADE"

por:

***iptables -t nat -A POSTROUTING -i eth1 -p tcp —dport 80 -j REDIRECT ***
—to-port 3128

Principais dicas

Novas chains x Logs

```
iptables -N LPI
iptables -A LPI -m limit —limit 5/hour —limit-burst 3 -p icmp —icmp-type echo-request
         -j LOG —log-level info —log-prefix "Log ping: "
iptables -A LPI -j REJECT —reject-with icmp-port-unreachable
iptables -A INPUT -p icmp —icmp-type echo-request -j LPI
```

Na regra citada, criamos uma nova *chain* chamada *LPI* (***-N LPI***). O nome pode ser definido a critério de cada um – foi definido *LPI* para passar uma idéia de "*LogPIng*". Para evitar um *flood* (enxurrada de mensagens nos *logs* do sistema), foi utilizado o módulo "*limit*" a fim de limitar a freqüência com que serão feitos *logs* de sinais *icmp* de "*echo-request*". Ou seja, serão logadas três seqüências (***—limit-burst***) de *ping*, com um máximo de cinco registros de *log* por hora.

Sinais de *ping* e *traceroute* utilizam o protocolo *icmp* (***-p icmp***). Um sinal de "*echo-request*" (***—icmp-type***) é enviado quando utilizamos o *ping*, e "*echo-replay*" é o sinal de reposta.

Os *logs* no *iptables* serão feitos via *target LOG* (*-j LOG*). Poderemos definir o nível de detalhamento do *log* (*de forma análoga ao syslogd*) utilizando o parâmetro "*—log-level*", e "*—log-prefix*" para definir qual será o texto que antecederá o *log* propriamente dito.

Portas de comunicação e Origem/Destino

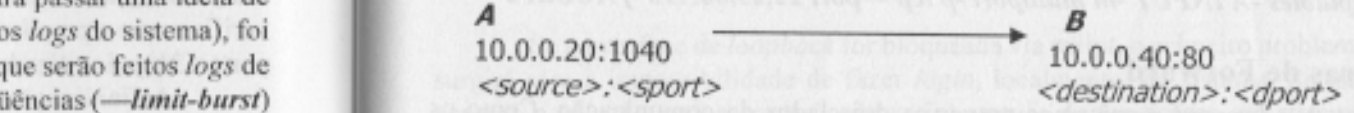
Principais portas de acesso:

21	(ftp) Utilizado para gerenciamento de arquivos (comunicação em texto puro)
22	(ssh) Secure Shell – para acessos remotos (fornece comunicação encriptada)
23	(telnet) Similar ao ssh, porém sem encriptação (comunicação em texto puro)
25/110	(smtp/pop3) Serviços de e-mail – envio/recebimento
110	(pop3) Recebimento de e-mail
53	(dns) Serviço de DNS (serviço de nome de domínio)
80	(http) Páginas de Web (HyperText – Hipertexto)
443	(https) Páginas de Web Seguras (com *SSL – Secure Socket Layer*)
901	(swat) Configurador do Samba (interface *Web*)

Para conferir as diferentes portas de acesso previstas pelo sistema:
cat /etc/services | more

Em *services* é que definimos um nome pelo qual determinada porta responderá (criação de "apelidos"). É possível acrescentar novas definições de porta a */etc/services*, no entanto, não altere nenhuma já existente (principalmente em se tratando de padrões como *http*, *ftp* e etc.), pois muitos programas se baseiam nesta definição, e alterar o padrão pode acarretar em sérios problemas de acesso ou segurança.

Toda comunicação é composta por um lado de "*origem*" e outro de "*destino*". Cada lado é composto por endereço *IP* e porta de acesso. Quando um "*host A*", solicita um serviço de *Web*, ao "*host B*", dizemos que a origem "*A*" solicita um pedido de conexão à porta 80 do "*host servidor B*" (destinatário). Neste tipo de comunicação uma porta local (alta) em "*A*" será utilizada para estabelecer conexão com a porta 80 em "*B*".



Quando quisermos aplicar regras à origem de uma conexão (*source*), trabalharemos com o argumento "*-s*" e se quisermos aplicar a regra à respectiva porta, incluiremos na regra o argumento "*—sport*" ou "*--source-port*".

Para tratar conexões ***destinadas*** a um *host* qualquer, criaremos regras com o argumento "*-d*", e "*—dport*" ou "*--destination-port*" para tratar a respectiva porta.

Exemplo:

Supondo que nosso *firewall* (10.0.0.254) disponibiliza o serviço *ssh*, mas desejamos permitir acesso "*apenas ao ip 10.0.0.100*":
<source=10.0.0.100>:<sport=1024 ou superior>
<destination=10.0.0.254>:<dport=22>

"O firewall possui duas interfaces de rede: eth0 interface com Internet e eth1 interface com a rede local".
iptables -A INPUT -i eth1 -s 10.0.0.100 -p tcp —dport 22 -j ACCEPT

> **Obs.:** Sempre que for necessário identificar a porta de origem ou destino em uma regra de firewall, será obrigatória a identificação do protocolo (tcp/udp/icmp/igmp). Regras com "—port" ou "—sport" sem atribuição de protocolo (-p) acarretará em erro de sintaxe.

Tratando intervalos de portas e/ou múltiplas portas

a) bloquear todas as portas acima da 80:
iptables -A INPUT -p tcp —dport 80: -j DROP

b) bloquear intervalo de portas entre 20 e 80:
iptables -A INPUT -p tcp —dport 20:80 -j DROP

c) permitir apenas duas portas (22-ssh, e 25-smtp):
iptables -A INPUT -m multiport -p tcp —port 22,25 -j ACCEPT

d) mesclando multiplas portas e intervalos (permissão):
iptables -A INPUT -m multiport -p tcp —port 22,25,80:110 -j ACCEPT

Regras de Forward

No forward precisamos prever os dois lados da comunicação. Como os exemplos demonstrados no livro aplicam política *default drop* para *FORWARD*, é necessário criar regras de *accept* prevendo a solicitação e a resposta a uma dada comunicação:
iptables -A FORWARD -s <rede_interna> ... -j ACCEPT
iptables -A FORWARD -d <rede_interna> ... -j ACCEPT

Note, no exemplo anterior que, em se tratando de um *gateway* de *Internet* (compartilhar acesso) a segurança seria comprometida facilmente, pois permitimos totalmente o repasse de pacotes para a rede interna.

Para tratar "este problema" podemos modificar a regra para:
iptables -A FORWARD -d <rede_interna> -m state —state NEW,INVALID -j DROP
***iptables -A FORWARD -d <rede_interna> -m state ***
—state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -s <rede_interna> -j ACCEPT

Na regra anterior, será feito um bloqueio a qualquer conexão nova ou inválida que se destine a nossa rede local. Só permitiremos comunicação com a rede local quando a conexão estiver estabelecida ou for reincidente (*ftp*, por exemplo), desta forma apenas a rede local iniciará conexões.

Interface de comunicação

Faremos referência a interfaces de comunicação de duas formas. Quando tratarmos pacotes que entram (*input*) por uma interface, faremos uso do parâmetro "*-i*", e "*-o*" para tratar pacotes que saem (*output*) por uma determinada interface.

Obviamente que a utilização destes parâmetros varia de acordo com a *chain* em questão:

Chain	*Interfaces possíveis*
INPUT	-i
OUTPUT	-o
FORWARD	-i e/ou –o
POSTROUTING	-o
PREROUTING	-i

Lembre-se de permitir o acesso para a interface *loopback* caso sua política de acesso para *INPUT* esteja definida como *DROP*.
iptables -A INPUT -i lo -j ACCEPT

Se a interface de *loopback* for bloqueada via script, o primeiro problema que surgirá será a impossibilidade de fazer *login*, localmente, no sistema. Se isto lhe ocorrer, a solução será a correção do script através de um acesso, ao sistema, em modo *single*.

Zerando regras de firewall

Para zerar regras de *firewall* (*flush*), faremos de duas formas. Para fazer flush nas regras próprias do *iptables* utilizaremos "*-F*". E "*-X*", para remover regras definidas em "*user-space*" – *chains* criadas por usuários via parâmetro "*-N*".

Ao digitar < *iptables -F* >, estaremos fazendo flush apenas na tabela *filter*. Quando for necessário fazer *flush* nas tabelas "*nat*" ou "*mangle*" deveremos indicar a tabela. Se desejarmos fazer um *flush* completo, a sintaxe seria esta:

iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X

Verifique as políticas de acesso de cada *chain* quando houver necessidade de um flush (para isto, digite < *iptables -L* >). Se a política de acesso for *default drop* para *INPUT* e fizermos *flush* na tabela *filter*, ficaremos sem acesso a nada (praticamente "travados") até que novas regras sejam incluídas ou a política modificada para *accept*. É possível fazer flush por *chain*.
iptables -P FORWARD ACCEPT
iptables -F FORWARD

Regras de inversão e pacotes que iniciam conexões

Pacotes que iniciam conexões apresentam o *flag-tcp syn* ativo. É comum criar regras com este tipo de tratamento, em conexões com *router/modem ADSL*. Se configurarmos, um *firewall/gateway Linux* para compartilhar o acesso de Internet e o acesso for "provido" por um roteador *ADSL*, precisamos ter em mente:

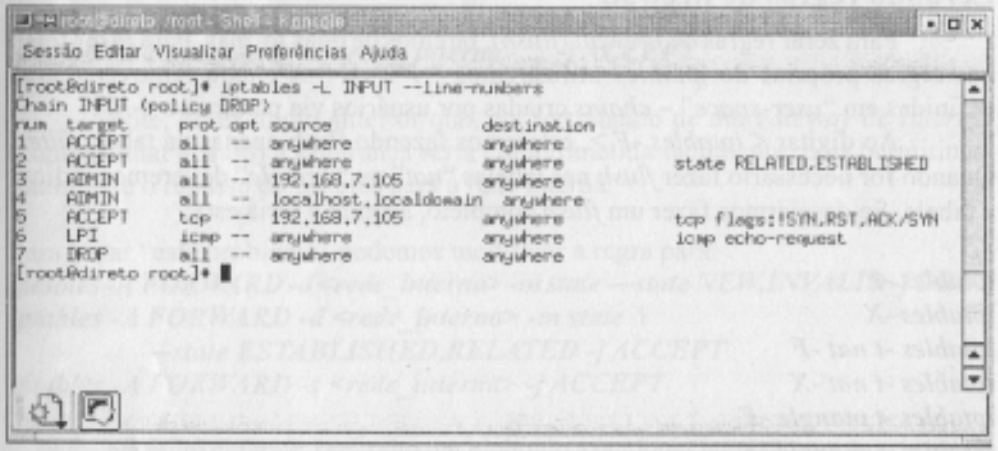
- roteadores *ADSL* apresentam quedas repentinas (em curto espaço de tempo) e no retorno tentam comunicação com o *firewall*.
- se o *firewall* estiver com *input* configurado para *default drop* teremos problemas no retorno do *roteador*.
- como solução acrescentaremos a seguinte regra:

iptables -A INPUT -s <ip_router> -p tcp ! –syn -j ACCEPT

Desta forma estamos permitindo a comunicação do *router* ("*-s ip_router*") com o *firewall*, desde que o roteador não inicie ("*! —syn* " – equivale a *not —syn*) conexão com o *firewall*.

Verificando regras de firewall (Input x Append x Delete)

a) Para listar apenas as regras de *INPUT*

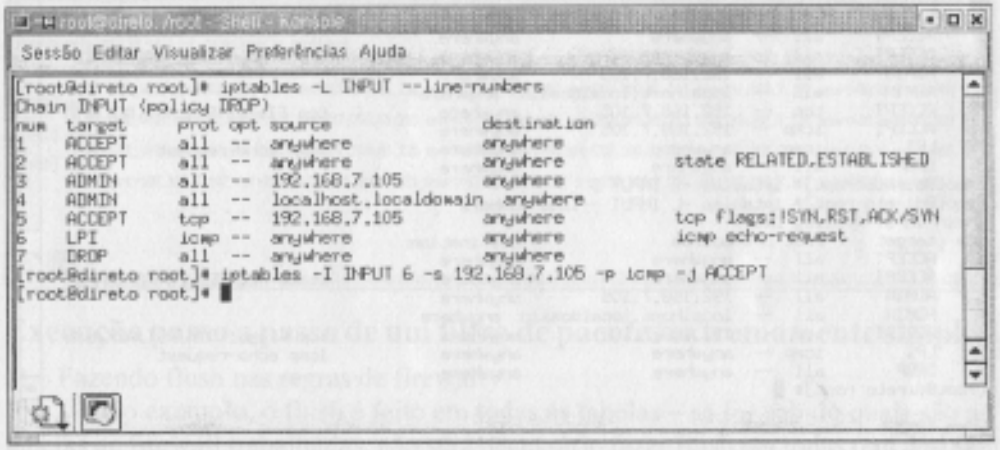


De acordo com a regra exibida anteriormente percebemos que:

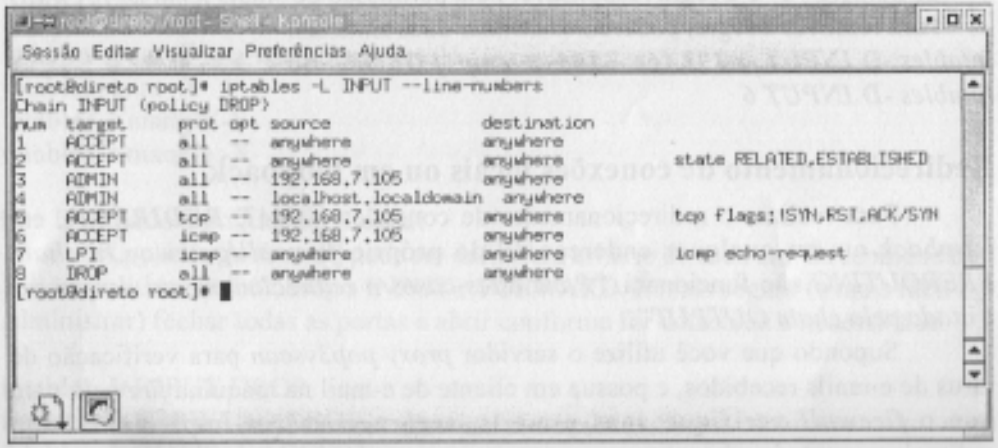
Linha 1 – acesso livre para interface *loopback* (como a exibição não foi em modo "detalhado (*-v*)" o nome da interface não foi exibido);
Linha 2 – permite o *INPUT* de conexões reincidentes ou estabelecidas (iniciadas pelo *firewall – RELATED,ESTABLISHED*);
Linhas 3 e 4 – permissão *ADMIN* (*chain* de usuário – para permitir *ssh*, por exemplo) aos *hosts* de endereço *ip* 192.168.7.105 e 127.0.0.1 (local) .

Linha 5 – permite pacotes vindos de 192.168.7.105 que não iniciem conexões.
Linha 6 – aplica *chain LPI* a pacotes com *flag-icmp echo-request* ativo – para bloqueio e *log* de sinais de *ping*, por exemplo;
Linha 7 – bloqueia qualquer pacote, destinado ao *firewall*, não tratado anteriormente.

b) Se analisarmos as regras anteriores, verificaremos que o *host* 192.168.7.105 tem permissão de acesso administrativo (*ADMIN* – acesso *ssh*), mas não poderá pingar o *firewall*. Se desejarmos permitir o *ping*, apenas, ao *host* 192.168.7.105, precisaremos incluir a permissão antes da regra de número 6 – que é a regra responsável pelo bloqueio de qualquer sinal de *echo-request*.

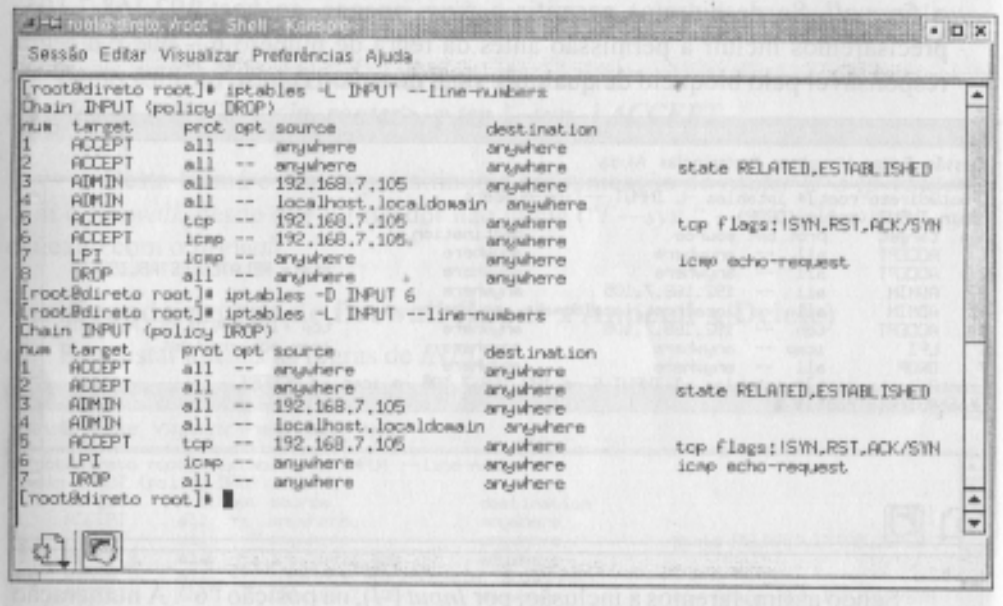


Sendo assim, faremos a inclusão, por *Input* (*-I*), na posição "6". A numeração de todas as regras a partir desta numeração será incrementada em "1".



c) Listando as regras de *firewall* após a inclusão de uma nova regra na posição "6". Repare que a regra de número "6", corresponde à regra digitada no exemplo anterior:
iptables -I INPUT 6 -s 192.168.7.105 -p icmp -j ACCEPT

d) Se não quisermos, mais, que o *host* 192.168.7.105 possa pingar nosso *firewall*, poderemos remover a regra a partir do número de identificação.



– Com o comando "***iptables -L INPUT —line-numbers***" percebemos que a regra de número "6" é a regra responsável pela permissão de sinais *icmp* ao *firewall*.
– Para remover a regra, poderemos utilizar duas sintaxes:
iptables -D INPUT -s 192.168.7.105 -p icmp -j DROP **ou**
iptables -D INPUT 6

Redirecionamento de conexões locais ou em loopback

Para realizar o redirecionamento de conexões (***DNAT, REDIRECT***), em *loopback* ou em qualquer endereço *IP* do próprio *firewall/gateway*, a *chain PREROUTING* não funcionará. *"Para estes casos o redirecionamento deverá ser tratado pela chain OUTPUT".*

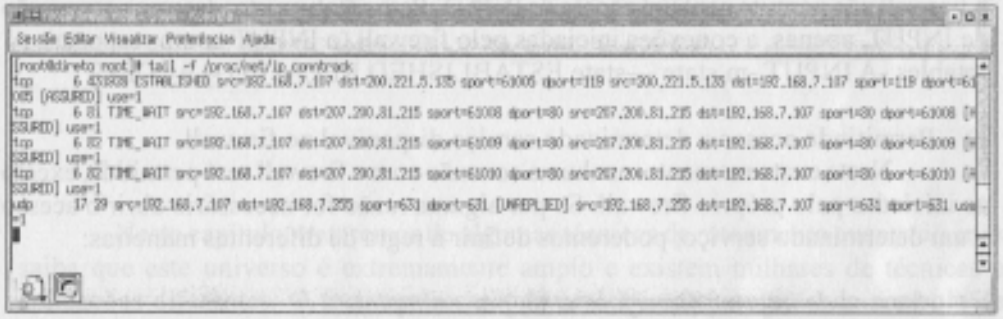
Supondo que você utilize o servidor *proxy pop3vscan* para verificação de vírus de e-mails recebidos, e possua em cliente de e-mail na máquina *firewall*, para que o *firewall* verifique seus e-mails, será necessário implementar um redirecionamento localmente:

iptables -t nat -A OUTPUT -o lo -p tcp —dport 110 -j REDIRECT —to-port 8110
iptables -t nat -A OUTPUT -o eth0 -p tcp —dport 110 -j REDIRECT —to-port 8110

"Equivale a um proxy transparente para pop3".

Acompanhando as conexões estabelecidas

Com o *netfilter* (*ip_conntrack*) podemos acompanhar o fluxo, detalhado, de pacotes que passam pelo *firewall* (*hosts* de origem e destino, portas, *status* de conexão e etc). Para tal, basta listar o conteúdo do arquivo ***/proc/net/ip_conntrack***.



Execução passo a passo de um filtro de pacotes extremamente simples

a) Fazendo flush nas regras de firewall

No exemplo, o flush é feito em todas as tabelas – se for sabido quais são as tabelas de firewall trabalhadas, não será necessário fazer flush em todas (em caso de dúvida, faça o flush em todas).

```
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X
```

b) Definindo a política de firewall

A política de firewall pode ser definida a critério de cada um. É recomendado aplicar políticas default drop a INPUT e FORWARD. É mais seguro (e mais fácil de administrar) fechar todas as portas e abrir conforme for detectada a necessidade.

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

c) Permitindo acessos em loopback

Ao adotarmos uma política default drop para INPUT, precisaremos abrir o acesso à interface loopback, pois o sistema também faz acesso a esta interface.

```
iptables -A INPUT -i lo -j ACCEPT
```

> **Obs.:** Se o seu script adotar política default drop para OUTPUT, uma regra de acesso à interface loopback será necessária (idem a INPUT).

d) Acessos iniciados pelo firewall

Com as regras digitadas, até então, nenhuma conexão será permitida porque o firewall não permite nenhum pacote de INPUT. Para resolver isto, daremos permissão de INPUT, apenas, a conexões iniciadas pelo firewall (o INPUT como resposta).

```
iptables -A INPUT -m state —state ESTABLISHED,RELATED -j ACCEPT
```

e) Permitindo acesso a determinado serviço disponível no firewall

Neste exato momento, nenhuma conexão com o firewall será permitida, exceto as iniciadas pelo próprio firewall. Se por alguma razão for necessário abrir o acesso a um determinado serviço, poderemos definir a regra de diferentes maneiras:

– *Acessos de internet (interface eth0, por exemplo)*

```
iptables -A INPUT -i eth0 -p tcp --dport <porta> -j ACCEPT
```

– *Apenas acessos da rede local (interface eth1, por exemplo)*

```
iptables -A INPUT -i eth1 -p tcp —dport <porta> -j ACCEPT
```

– *Apenas ao IP 10.0.0.88 da rede local*

```
iptables -A INPUT -i eth1 -s 10.0.0.88 -j ACCEPT
```

– *Permitir e logar apenas acessos da rede local*

```
iptables -N LOGLOCAL
iptables -A LOGLOCAL -p tcp -dport <porta> -j LOG \
            —log-level info —log-prefix "Acesso xyz: "
iptables -A LOGLOCAL -p tcp —dport <porta> -j ACCPET

iptables -A INPUT -i eth1 -j LOGLOCAL
```

> **Obs.:** O recomendado é não diponibilizar serviços no host firewall – no máximo acesso remoto por SSH.

Para obter mais informações sobre iptables, aconselho a leitura:
www.netfilter.org
www.iptablesbr.cjb.net

Capítulo 7

Roteamento avançado por iproute2

Neste capítulo, trataremos de algumas técnicas de roteamento avançado, mas saiba que este universo é extremamente amplo e existem milhares de técnicas e aplicações diferentes. O foco neste capítulo será a implementação de controle de banda por *CBQ* e seleção de *default gateway* baseado em porta de acesso – ambas as soluções baseadas na combinação *iptables* e *iproute2*.

Rede x Roteamento

Todos os controles de rede implementados por *ifconfig* e *route* podem ser feitos por uma ferramenta mais robusta, o *iproute2* – como configuração de rede, roteamento e etc. A utilização é semelhante, no entanto, o nível de recursos e informações da rede são superiores no *iproute2*.

O pacote *iproute2* deverá estar instalado em seu sistema. O binário responsável por todo o processo visto neste capítulo se chama "*ip*". Serão abordados apenas aspectos relevantes à compreensão das técnicas de controle de banda e seleção de *default gateway*, que serão tratadas adiante.

Com o comando *ip* atuaremos sobre:

– *rede*: ***ip link***
– *host*: ***ip addr***
– *rotas*: ***ip route***

É possível indicar, apenas, a primeira letra após o comando *ip*, mas não é recomendado utilizar esta sintaxe na construção de scripts, pois pode tornar confusa a interpretação. Exemplo: