Linux 2.4 Packet Filtering HOWTO

Rusty Russell, mailing list `netfilter@lists.samba.org`
Tradução para Português do Brasil: Guilherme Ude Braz (guilherme@linuxplace.com.br)

Revision: 1.19 Date: 2001/05/26 20:26:48

Conteúdo

1 Introdução

Bemvindo, gentil leitor.

Parte-se do pressuposto que você sabe o que são: endereço IP, endereço de rede, máscara de rede, roteamento e DNS. Caso contrário, é recomendável que você leia o Network Concepts HOWTO (em inglês, por enquanto).

Esse HOWTO passa de uma gentil introdução (a qual o deixará alegre e entusiasmado agora, porém desprotegido no mundo real) para uma discussão um pouco mais árida (que vai deixar todos, exceto as almas mais fortes, confusos e procurando por armamento pesado).

Sua rede não é **segura**. Permitir comunicação rápida e conveniente restringindo seu uso para o bem, e não para ações mal intencionadas involve questões muito mais amplas que o foco deste documento. Portanto, tais questões não serão tratadas neste HOWTO.

Então, apenas você pode decidir até onde vai o compromisso. Tentarei instruí-lo sobre o uso de algumas ferramentas disponíveis e algumas vulnerabilidades conhecidas, e espero que você utilize isso para o bem, e não para o mal.

2 Onde está o site oficial? Existe uma Mailing List?

Esses são os três sites oficiais:

- Obrigado *Filewatcher* `http://netfilter.filewatcher.org/` .
- Obrigado *The Samba Team and SGI* `http://netfilter.samba.org/` .

- Obrigado *Harald Welte* http://netfilter.gnumonks.org/ .

A mailing list oficial do netfilter está em

Netfilter List http://www.netfilter.org/contact.html#list .

3 Então, o que é um filtro de pacotes?

Um filtro de pacotes é um sofware que analiza o *cabeçalho (header)* dos pacotes enquanto eles passam, e decide o destino do pacote como um todo. Ele pode decidir entre **descartar (DROP)** o pacote (descartando-o como se nunca o tivesse recebido), **aceitar (ACCEPT)** o pacote (deixar o pacote seguir seu caminho), ou algo mais complicado que isso.

No Linux, filtragem de pacotes está implementada diretamente no kernel (como um módulo ou diretamente compilado), e há várias coisas interessantes que podem ser feitas com os pacotes, mas o princípio geral é analizar o cabeçalho dos pacotes e decidir o que ser feito com o pacote.

3.1 Porque filtrar os pacotes?

Controle. Segurança. Vigilância.

Controle:

quando uma máquina Linux é utilizada para conectar sua rede interna em outra rede (a Internet, por exemplo) há a oportunidade de permitir certo tipo de tráfego, e rejeitar outro. Por exemplo, o cabeçalho do pacote contém o endereço de destino do pacote, logo você pode evitar que os pacotes saiam em direção a um endereço. Por exemplo, eu utilizo o Konqueror para ler as tirinhas do Dilbert. Há publicidade do doubleclick.net nesta página, e o Konqueror gasta meu precioso tempo baixando-a. Pode-se dizer ao netfilter para não permitir pacotes vindos de doubleclick.net (há melhores maneiras de se fazer isso: veja o Junkbuster).

Segurança:

quando uma máquina Linux é a única coisa entre o caos da Internet e sua calma e organizada rede, é bom saber que é possível restringir o tráfego vindo do mundo externo. Por exemplo, você pode permitir que qualquer tipo de pacote saia da sua rede, mas você é preocupado com o famoso ataque 'Ping of Death (Ping da Morte)' vindo de mal-feitores do mundo externo. Como outro exemplo, você pode não gostar da idéia de permitir que qualquer um da rede externa conecte-se via telnet na sua máquina Linux, mesmo que todas as contas da máquina tenham senhas. Talvez você queira (como a maioria das pessoas) ser apenas um observador, e não um servidor. Simplesmente não deixe ninguém conectar, dizendo ao filtro de pacotes para rejeitar pacotes requisitando a criação de novas conexões.

Vigilância:

às vezes uma máquina mal configurada na rede local pode decidir enviar pacotes descontroladamente para o mundo externo. É interessante dizer ao filtro de pacotes para te informar se algo anormal ocorrer, talvez você possa fazer algo para resolver o problema, ou você pode ser apenas mais um curioso :)

3.2 Como eu filtro pacotes no Linux?

Os kernels Linux têm tido filtros de pacotes desde a série 1.1. A primeira geração, baseada no ipfw do BSD, foi portada por Alan Cox no final de 1994. Essa implementação foi melhorada por Jos Vos e outros para o Linux 2.0; a ferramenta userspace 'ipfwadm' controlava as regras de filtragem do kernel. Em meados de 1998, para o Linux 2.2, eu reescrevi muitas linhas de código do kernel, com a ajuda de Michael Neuling, e introduzi a ferramente userspace 'ipchains'. Finalmente, a ferramenta da quarta geração, o 'iptables', e mais um árduo trabalho de reedição do kernel ocorreu em meados de 1999 para o Linux 2.4. Este HOWTO concentra-se no iptables.

É necessário um kernel que possua a infraestrutura netfilter implementada: netfilter é um framework dentro do kernel Linux com o qual outras coisas (como o módulo do iptables) podem conectar-se. Isso significa que você precisa do kernel 2.3.15 ou posteriores, e respoder 'Y (SIM)' para CONFIG_NETFILTER na sua configuração do kernel.

A ferramenta `iptables` conversa com o kernel e fala quais são os pacotes a serem filtrados. Ao menos que você seja um programador, ou apenas um curioso, esta é a forma pela qual você controlará a filtragem dos pacotes.

3.2.1 iptables

A ferramenta `iptables` insere e apaga regras da tabela de filtragem de pacotes do kernel. Isso significa que qualquer coisa que você configure será perdida assim que o Linux for reiniciado.

`iptables` é uma substituição para `ipfwadm` e `ipchains`: veja 8 (Utilizando ipchains e ipfwadm) para saber como evitar a utilização de iptables se você está utilizando alguma dessas ferramentas.

3.2.2 Tornando as regras permanentes

Sua configuração atual de firewall está guardada no kernel, logo será perdida na próxima vez que a máquina for reiniciada. Escrever o iptables-save e o iptables-restore está na minha lista de afazeres. Quando eles existirem, prometo que funcionarão muito bem.

Por enquanto, coloque os comandos necessários para configurar suas regras em um script de inicialização. Esteja certo de que o script faça algo inteligente caso algum dos comandos falhe (usualmente 'exec /sbin/sulogin').

4 Quem é você, e porque está alterando o meu kernel?

Eu sou Rusty; o mantenedor do Linux IP Firewall e mais um programador que apareceu na hora certa e no lugar certo. Eu escrevi ipchains (veja 3.2 (Como eu filtro pacotes no Linux?) para saber quem realmente trabalhou no projeto), e aprendi o suficiente para que o filtro de pacotes tenho sido feito da forma correta. Eu espero.

WatchGuard `http://www.watchguard.com` , uma excelente empresa de firewall que vende uma interessante plug-in Firebox, se ofereceu para pagar-me para fazer nada, assim eu poderia gastar todo o meu tempo escrevendo tudo isso, e mantendo coisas anteriores. Eu previ 6 meses, e levei 12, mas senti, ao terminar, que havia feito a coisa da forma que ela deveria ser. Depois de muitas reedições, um crash no meu disco rígido, o roubo de meu laptop, alguns sistemas de arquivos corrompidos e uma tela quebrada, aqui está.

Eu gostaria de esclarecer alguns desentendimentos das pessoas: eu nao sou nenhum guru do kernel. Sei disso porque meu trabalho me colocou em contato com alguns dos verdadeiros gurus: David S. Miller, Alexey Kuznetsov, Andi Kleen, Alan Cox. De qualquer forma, eles estão ocupados fazendo a verdadeira mágica, deixando-me em paz, fazendo meu serviço, que é mais seguro.

5 Guia ultra rápido do Rusty para filtragem de pacotes

A maioria das pessoas possui uma simples conexão PPP connection com a Internet, e não querem ninguém entrando em sua rede:

```
## Carregando módulos de acompanhamento de conexões (desnecessário se compilados diretamente no kernel).
# insmod ip_conntrack
# insmod ip_conntrack_ftp

## Cria chain que rejeita novas conexões, exceto as vindas da rede interna.
# iptables -N block
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A block -j DROP

## Saltar das chains INPUT e FORWARD para a chain block.
# iptables -A INPUT -j block
# iptables -A FORWARD -j block
```

6 Como os pacotes passam pelos filtros

O kernel começa com três listas de regras na tabela 'filter'; tais listas são denominadas **firewall chains** ou apenas **chains**. As três chains chamam-se **INPUT**, **OUTPUT** e **FORWARD**.

Para os fãs de ASCII-art, as chains estão organizadas da seguinte forma: **(Nota: é uma organização bem distinta dos kernels 2.0 e 2.2!)**

```
                          _____
Entrada                  /     \         Saída
      -->[Decisão de]-->|FORWARD|------->
         [Roteamento]    \_____/        ^
              |                         |
              v                        ____
             ___                      /    \
            /   \                    |OUTPUT|
           |INPUT|                    \____/
            \___/                       ^
              |                         |
              --> Processamento Local --
```

Os três "círculos"representam as chains mencionadas acima. Quando o pacote atinge um círculo no diagrama, a chain é examinada a fim de decidir o destino do pacote. Se a chain diz para rejeitar (DROP) o pacote, ele é descartado, mas se a chain diz para aceitar o pacote (ACCEPT), ele continua a viajar no diagrama.

Uma chain é uma lista de **regras**. Cada regra diz 'se o cabeçalho do pacote se parece com isso, então aqui está o que deve ser feito com o pacote'. Se a regra não associa-se com o pacote, então a próxima regra na chain é consultada. Se não há mais regras a consultar, o kernel analiza a **política** da chain para decidir o que fazer. Em um sistema preocupado com segurança, a política diz ao kernel para rejeitar (DROP) o pacote.

1. Quando o pacote chega (pela placa Ethernet, por exemplo) o kernel analiza o destino do pacote: isso é chamado roteamento (routing).
2. Se ele é destinado a própria máquina, o pacote desce no diagrama, indo para a chain INPUT. Se ele passar pela chain INPUT, então a máquina recebe o pacote.
3. Depois, se o kernel não tem suporte a forwarding, ou não sabe como repassar (forward) o pacote, este é descartado. Se há suporte a forwarding e o pacote é destinado a outra interface de rede (se você possui outra), o pacote vai para a chain FORWARD. Se ele for aceito (ACCEPT), ele será enviado.
4. Finalmente, um programa rodando na máquina firewall pode enviar pacotes. Esses pacotes passam pela chain OUTPUT imediatamente: se ela aceitar o pacote, ele continua seu caminho, caso contrários ele é descartado.

7 Utilizando o iptables

O iptables tem uma página de manual muito detalhada (`man iptables`), se você necessitar de informações mais específicas. Aqueles mais familiarizados com o ipchains provavelmente gostarão de ler 10 (Diferenças entre iptables e ipchains); eles são bem similares.

Há muitas coisas que podem ser feitas com `iptables`. Você começa com três chains built-in `INPUT`, `OUTPUT` e `FORWARD` as quais não podem ser apagadas. Estas são as operações para gerenciar chains:

1. Criar nova chain (-N).
2. Apagar uma chain vazia (-X).
3. Mudar a política de uma chain built-in. (-P).
4. Listar as regras de uma chain (-L).
5. Apagar todas as regras de uma chain (-F).
6. Zerar os contadores de pacotes e bytes de todas as regras de uma chain (-Z).

Há várias formas de manipular regras dentro de uma chain:

1. Adicionar uma nova regra na chain (-A).
2. Inserir uma nova regra em alguma posição da chain (-I).
3. Substituir uma regra em alguma posição da chain (-R).

4. Apagar uma regra em alguma posição da chain (-D).
5. Apagar a primeira regra que associa (com alguma condição) numa chain (-D).

7.1 O que será visto quando seu computador iniciar

O iptables pode ser um módulo, chamado ('iptable_filter.o'), que deve ser automaticamente carregado quando você rodar `iptables` pela primeira vez. Ele também pode ser compilado diretamente no kernel.

Antes dos comandos do iptables rodarem (cuidado: algumas distribuições rodarão iptables em seus scripts de inicialização), não haverão regras em quaisquer chains ('INPUT', 'FORWARD' and 'OUTPUT') e todas as chains terão a política ACCEPT. A política padrão da chain FORWARD pode ser alterada fornecendo o parâmetro 'forward=0' para o módulo iptable_filter.

7.2 Operações em uma única regra

Este é o arroz com feijão da filtragem de pacotes: manipular regras. Usualmente, você provavelmente utilizará os comandos para adicionar (-A) e apagar (-D). Os outros (-I para inserir e -R para substituir) são apenas extensões destes conceitos.

Cada regra especifica uma série de condições que o pacote deve atender, e o que fazer com o mesmo se ele atendê-las (um alvo 'target'). Por exemplo, você pode desejar descartar todos os pacotes ICMP vindos do endereço IP 127.0.0.1. Então neste caso nossas condições são que o protocolo precisa ser ICMP e o endereço de origem deve ser 127.0.0.1. Nosso alvo (target) é 'DROP'.

127.0.0.1 é a interface 'loopback', que existirá mesmo que você não tenha nenhuma conexão de rede real. Pode-se utilizar o programa 'ping' para gerar esse tipo de pacote (ele simplesmente manda um ICMP tipo 8 (requisição de eco) que é respondido pelos hosts com um ICMP tipo 0 (resposta de eco)).

```
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP
# ping -c 1 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
#
```

Pode-se ver que o primeiro ping é bem sucedido (a opção '-c 1' diz ao ping para mandar um único pacote).

Então foi adicionada (-A) à chain 'INPUT', uma regra especificando que pacotes vindos de 127.0.0.1 ('-s 127.0.0.1') com o protocolo ICMP ('-p icmp') devem ser mandados para DROP ('-j DROP').

Logo depois, testamos nossa regra, usando um segundo ping. Haverá uma pausa antes que o programa desista de esperar pelo pacote que nunca viria.

Pode-se apagar regras de duas maneiras. Primeiramente, desde que sabemos que existe apenas uma regra na chain INPUT, podemos utilizar um número para designar a regra, como abaixo:

```
# iptables -D INPUT 1
#
```

Para apagar a regra número 1 na chain INPUT.

A segunda forma, é fazer o mesmo que faríamos para adicionar a regra, trocando -A por -D. Isso é útil quando você tem uma chain muito complexa e não quer contar para descobrir que a regra 37 deve ser apagada. Neste caso usaria-se:

```
# iptables -D INPUT -s 127.0.0.1 -p icmp -j DROP
#
```

A sintaxe do -D deve ter exatamente as mesmas opções que seriam passadas para a opção -A (ou -I ou -R). Se existem várias regras idênticas na mesma chain, apenas a primeira será apagada.

7.3 Especificações para filtragem

Foi visto o uso do '-p' para especificar o protocolo, e '-s' para especificar o endereço de origem, mas existem outras opções que podem ser utilizadas para especificar outras características dos pacotes. O que segue é uma explicação exaustiva.

7.3.1 Especificando endreços IP de origem e destino

Endereços IP de Origem ('-s', '–source' or '–src') e destino ('-d', '–destination' or '–dst') podem ser especificados em quatro formas diferentes. A mais comum é utilizar o nome completo, como 'localhost' ou 'www.linuxhq.com'. A segunda é dizer o IP, como '127.0.0.1'.

A terceira e a quarta formas permitem a especificação de um grupo de endereços IP, como '199.95.207.0/24' ou '199.95.207.0/255.255.255.0'. Ambas especificam qualquer endereço IP de 199.95.207.0 até 199.95.207.255; os dígitos depois da '/' dizem quais partes do endereço IP são significantes. '/32' ou '/255.255.255.255' é o padrão (associando o endereço IP inteiro). Para especificar qualquer endereço IP pode-se utilizar '/0', conforme descrito abaixo:

```
[ NOTA: '-s 0/0' é redundante. ]
# iptables -A INPUT -s 0/0 -j DROP
#
```

Isso raramente é utilizado, uma vez que o efeito da regra acima é o mesmo que se não fosse especificada nenhuma origem.

7.3.2 Especificando Inversão

Muitas flags, incluindo '-s' (or '–source') e '-d' ('–destination') podem ter seus argumentos precedidos de '!' (pronunciado 'não') para associar-se com endereços DIFERENTES aos passados à opção. Por exemplo, '-s ! localhost' associa-se com qualquer pacote que **não** venha de localhost.

7.3.3 Especificando protocolo

O protocolo pode ser especificado com '-p' (ou '–protocol'). O protocolo pode ser um número (se você sabe o valor numérico dos protocolos) ou um nome para casos especiais como 'TCP', 'UDP' or 'ICMP'. Digitar em maiúsculas ou minúsculas não faz diferença, 'tcp' funciona tão bem como 'TCP'.

Se o nome do protocolo é precedido de '!', a fim de invertê-lo, como '-p! TCP', a regra especificará todos os pactoes que **não** são TCP.

7.3.4 Especificando uma interface

As opções '-i' (or '–in-interface') e '-o' (or '–out-interface') especificam o nome de uma **interface** a especificar. Uma interface é um dispositivo físico pelo qual o pacote veio ('-i') ou está saindo ('-o'). Pode-se usar o comando `ifconfig` para listar as interfaces ativas.

Pacotes atravessando a chain `INPUT` não possuem interface de saída, logo qualquer regra utilizando '-o' nessa chain nunca aplicar-se-á. Da mesma forma, pacotes atravessando a chain `OUTPUT` não têm interface de entrada, logo qualquer regra utilizando '-i' nessa chain nunca aplicar-se-á.

Apenas pacotes passando pela chain `FORWARD` têm interfaces de entrada e saída.

Não há nenhum problema em especificar uma interface que ainda não existe; a regra não associar-se-á com quaisquer pacotes até que a interface torne-se ativa. Isso é extremamente útil para links discados PPP (usualmente com a interface `ppp0`) e similares.

Como um caso especial, um nome de interface terminando com um '+' vai associar-se com todas as interfaces (existam elas ou não) que comecem com aquela string. Por exemplo, para especificar uma regra que se associa com todas as interfaces PPP, a opção `-i ppp+` deve ser criada.

O nome da interface pode ser precedido de '!' para se associar com um pacote que **não** é representado pelas interface(s) especificada(s).

7.3.5 Especificando fragmentos

Às vezes um pacote é muito grande para ser enviado todo de uma vez. Quando isso ocorre, o pacote é dividido em **fragmentos**, e é enviado como múltiplos pacotes. Quem o recebe, remonta os fragmentos reconstruindo o pacote.

O problema com os fragmentos é que o fragmento incial tem os campos de cabeçalho completos (IP + TCP, UDP e ICMP) para examinar, mas os pacotes subsequentes só têm um pequeno grupo de cabeçalhos (somente IP, sem os campos dos outros protocolos). Logo examinar o interior dos fragmentos subsequentes em busca de cabeçalhos de outros protocolos (como extensões de TCP, UDP e ICMP) é impossível.

Se você está fazendo acompanhamento de conexões ou NAT, todos os fragmentos serão remontados antes de atingirem o código do filtro de pacotes, então você não precisa se preocupar sobre os fragmentos.

Caso contrário, é importante entender como os fragmentos são tratados pelas regras de filtragem. Qualquer regra de filtragem que requisitar informações que não existem *não* será válida. Isso significa que o primeiro fragmento é tratado como um pacote qualquer. O segundo e os seguintes não serão. Então uma regra `-p TCP -sport www` (especificando 'www' como porta de origem) nunca se associar-se-á com um fragmento (exceto o primeiro). O mesmo se aplica à regra oposta `-p TCP -sport ! www`.

De qualquer forma, você pode especificar uma regra especial para o segundo e os fragmentos que o sucedem, utilizando a flag '-f' (ou '–fragment'). Também é interessante especificar uma regra que *não* se aplica ao segundo e os outros fragmentos, precedendo a opção '-f' com '!'.

Geralmente é reconhecido como seguro deixar o segundo fragmento e os seguintes passarem, uma vez que o primeiro fragmento será filtrado, logo isso vai evitar que o pacote todo seja remontado na máquina destinatária. De qualquer forma, há bugs que levam à derrubada de uma máquina através do envio de fragmentos. A decisão é sua.

Nota para os administradores de rede: pacotes mal formados (pacotes TCP, UDP e ICMP muito pequenos para que o código do firewall possa ler as portas ou o código e tipo dos pacotes ICMP) são descartados quando ocorrem tais análises. Então os fragmentos TCP começam na posição 8.

Como um exemplo, a regra a seguir vai descartar quaisquer fragmentos destinados ao endereço 192.168.1.1:

```
# iptables -A OUTPUT -f -d 192.168.1.1 -j DROP
#
```

7.3.6 Extensões ao iptables: Novas Implementações

O `iptables` é **extensível**, assim, tanto o kernel quanto o iptables podem ser extendidos para fornecer novas funcionalidades.

Algumas dessas extensões são padronizadas, e outras são mais exóticas. Extensões podem ser feitas por qualquer um e distribuídas separadamente para usuários de nichos mais específicos.

As extensões do kernel geralmente estão no subdiretório de módulos do kernel como, por exemplo, /lib/modules/2.3.15/net. Elas são carregadas por demanda se seu kernel foi compilado com a opção CONFIG_KMOD marcada, não havendo a necessidade de inserí-las manualmente.

As extensões do iptables são bibliotecas compartilhadas as quais geralmente estão em /usr/local/lib/iptables/, mas uma distribuição os colocaria em /lib/iptables ou /usr/lib/iptables.

Extensões vêm em dois tipos diferentes: novos alvos (targets), e novas associações (depois falaremos mais sobre novos alvos 'targets'). Alguns protocolos automaticamente oferecem novos testes: atualmente são eles TCP, UDP e ICMP como mostrado abaixo.

Para esses protocolos você poderá especificar novos testes na linha de comando depois da opção '-p', que carregará a extensão. Para novos testes explícitos, utilize a opção '-m' para carregar a extensão, depois de -m, todas as opções da extensão estarão disponíveis.

Para conseguir ajuda com uma extensão, utilize a opção que a carrega ('-p', '-j' ou '-m') sucedida por '-h' ou '–help', conforme o exemplo:

```
# iptables -p tcp --help
#
```

Extensões TCP As extensões TCP são automaticamente carregadas se é especificada a opção '-p tcp'. Elas provêm as seguintes opções (nenhuma associa-se com fragmentos).

–tcp-flags

seguida por uma opcional '!', e por duas strings indicando flags, permite que sejam filtradas flags TCP específicas. A primeira string de flags é a máscara: uma lista de flags que serão examinadas. A segunda string diz quais flags devem estar ativas para que a regra se aplique. Por exemplo:

```
# iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK -j DROP
```

Essa regra indica que todas as flags devem ser examinadas ('ALL' é sinônimo de 'SYN,ACK,FIN,RST,URG,PSH'), mas apenas SYN e ACK devem estar ativadas. Também há um argumento 'NONE' que significa nenhuma flag.

–syn

opcionalmente precedido por '!', é um atalho para '–tcp-flags SYN,RST,ACK SYN'.

–source-port

seguido por '!' opcional, e uma única porta TCP, ou um conjunto (range) de portas. As portas podem ser descritas pelo nome, conforme listado em /etc/services, ou pelo número. Conjuntos (ranges) são dois nomes de portas separados por ':', ou (para especificar uma porta maior ou igual à especificada) uma porta sucedida por ':', ou (para especificar uma porta menor ou igual à especificada), uma porta precedida por ':'.

–sport

é sinônimo de '–source-port'.

–destination-port

e

–dport

funcionam de forma idêntica a

–source-port

, a única diferença é que elas indicam a porta de destino, e não a porta de origem.

–tcp-option

seguida por '!' opcional e um número, associa-se com um pacote com a opção TCP igual ao do número passado. Um pacote que não tem um cabeçalho TCP completo é automaticamente descartado se há uma tentativa de examinar suas opções TCP.

Uma explicação sobre as flags TCP Às vezes é util permitir conexões TCP em uma única direção, mas não nas duas. Por exemplo, você permitirá conexões em um servidor WWW externo, mas não conexões vindas deste servidor.

Uma tentativa ingênua seria bloquear pacotes TCP vindos do servidor. Infelizmente, conexões TCP necessitam de pacotes bidirecionais para um funcionamento perfeito.

A solução é bloquear apenas os pacotes utilizados para requisitar uma conexão. Tais pacotes são chamados pacotes **SYN** (ok, tecnicamente eles são pacotes com a flag SYN marcada, e as flags RST e ACK desmarcadas, mas dizemos pacotes SYN como atalho). Ao não permitir tais pacotes, nós impedimos conexões vindas do servidor.

A opção '–syn' é utilizada para isso: só é válida para regras que especificam TCP como protocolo. Por exemplo, para especificar conexões TCP vindas do endereço 192.168.1.1:

```
-p TCP -s 192.168.1.1 --syn
```

Essa opção pode ser invertida se precedida por '!', o que significa qualquer pacote exceto os que iniciam conexões.

Extensões UDP Essas extensões são automaticamente carregadas se a opção '-p udp' é especificada. Ela provê as opções '–source-port', '–sport', '–destination-port' and '–dport' conforme detalhado para o TCP acima.

Extensões ICMP Essas extensões são automaticamente carregadas se a opção '-p icmp' é especificada. Ela só possui uma opção diferente das demais:

–icmp-type

seguida por '!' opcional, e um nome de tipo icmp (por exemplo, 'host-unreachable'), ou um tipo numérico (exemplo '3'), ou um tipo numérico e código separados por '/' (exemplo '3/3'). Uma lista de tipos icmp é passada utilizando-se '-p icmp –help'.

Outras extensões Outras extensões no pacote netfilter são extensões de demonstração, que (caso instaladas) podem ser chamadas com a opção '-m'.

mac

Este módulo deve ser explicitamente especificado com '-m mac' ou '–match mac'. Ele é usado para associar-se com o endereço Ethernet (MAC) de origem do pacote, e logo só é útil nas chains PREROUTING e INPUT. Só provê uma única opção:

–mac-source

seguida por '!' opcional e um endereço ethernet passado em notação hexa, exemplo '–mac-source 00:60:08:91:CC:B7'.

limit

Este módulo deve ser explicitamente especificado com '-m limit' ou '–match limit'. É usado para restringir a taxa de pacotes, e para suprimir mensagens de log. Vai fazer com que a regra seja válida apenas um número de vezes por segundo (por padrão 3 vezes por hora, com um limite máximo def 5). Possui dois argumentos opcionais:

–limit

seguido de um número; especifica a média máxima de pacotes (ou LOGs, etc) permitida por segundo. Pode-se especificar a unidade de tempo, usando '/second', '/minute', '/hour' ou '/day', ou abreviações dos mesmos (assim, '5/second' é o mesmo que '5/s').

–limit-burst

seguido de um número, indicando o máximo de entradas antes do limite tornar-se válido.

Essa extensão pode ser usada com o alvo (target) LOG para criar registros (logs) limitados por taxa de incidência. Para entender o funcionamento disso, olhe a regra abaixo, que loga pacotes com os parâmetros padrão de limite:

```
# iptables -A FORWARD -m limit -j LOG
```

Na primeira vez que essa regra é analizada, o pacote será logado; na realidade, uma vez que o padrão máximo é 5, os cinco primeiros pacotes serão logados. Depois disso, passar-se-ão vinte minutos antes de que essa regra faça um novo logo, independente do número de pacotes que entrarem. Além disso, a cada vinte minutos que se passam sem que um pacote associe-se com a regra, o contador é diminuido em uma unidade. Logo, se nenhum pacote associar-se com a regra em 100 minutos, o limite estará zerado, voltando ao estágio inicial.

Nota: não é possível criar uma regra com tempo de recarga superior a 59 horas, então se você configura uma taxa média de um por dia, seu limite máximo deve ser menor que 3.

Esse módulo também pode ser usado para evitar uma série de ataques do tipo negativa de serviço (denial of service 'DoS') com uma taxa mais rápida, a fim de aumentar a sensibilidade do sistema.

Proteção contra Syn-flood:

```
# iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT
```

Port scanner suspeito:

```
# iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
```

Ping da morte:

```
# iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
```

Esse módulo funciona como uma "porta com resposta retardada", conforme o gráfico abaixo.

```
       taxa (pkt/s)
             ^        .---.
             |       / DoS \
             |      /       \
Auge do DoS -|.....:.........\.......................
 = (média *  |    /:          \
média-máxim) |   / :           \         .-.
             |  /  :            \       /   \
             | /   :             \     /     \
Fim do DoS  -|/....:..............:.../.......\..../.
 = média     |     :              :`-'         `--'
-------------+-----+--------------+------------------> time (s)
  LOGICA =>  Passa |  Não passa   |    Passa
```

Digamos que aprovamos um pacote por segundo com um limite máximo de cinco pacotes, mas os pacotes começam a vir quatro por segundo, durante três segundos, e recomeçam depois de mais 3 segundos.

```
        <--Flood 1-->           <---Flood 2--->

Total  ^                   Line  __--      YNNN
de     |               Rate  __--      YNNN
Pacotes|            mum  __--      YNNN
    10 |        Maxi __--         Y
```

```
     |         __--           Y
     |     __--             Y
     | __--    YNNN
     |-    YNNN
   5 |    Y
     |   Y                          Legenda:  Y -> Regra válida
     |  Y                                     N -> Regra deixou de valer
     | Y
     |Y
   0 +--------------------------------------------------->  Time (seconds)
      0   1   2   3   4   5   6   7   8   9  10  11  12
```

```
Nota de Tradução: Desculpem-me, mas eu não consegui traduzir esse gráfico sem
deformá-lo completamente, então algumas coisas ficaram em inglês :)
```

Percebe-se que os cinco primeiros pacotes podem exceder um pacote por segudo, depois o limite máximo passa a ser válido. Se há uma nova tentativa de flood, os pacotes são aceitos, mas apenas no limite estabelecido pela regra (1 pacote por segundo pois a cota do limite máximo já foi gasta).

owner

Esse módulo tenta associar-se com várias características do criador do pacote, para pacotes gerados localmente. Só é válido na chain OUTPUT, e mesmo assim alguns pacotes (como respostas de ping ICMP) podem nao ter dono, e nunca serão analizados pela regra.

–uid-owner userid

Associa-se com o pacote se este foi criado por um processo com o userid igual ao passado na opção.

–gid-owner groupid

Associa-se com o pacote se este foi criado por um processo com o groupid igual ao passado na opção.

–pid-owner processid

Associa-se com o pacote se este foi criado por um processo com o proccessid igual ao passado na opção.

–sid-owner sessionid

Associa-se com o pacote se este foi criado por um processo com o sessionid igual ao passado na opção.

unclean

Esse módulo experimental deve ser explicitamente especificado com '-m unclean' ou '–match unclean'. Ele faz diversas checagens de sanidade nos pacotes. Esse módulo não foi auditado, e não deve ser utilizado como dispositivo de segurança (ele provavelmente torna as coisas piores, uma vez que provavelemente está cheio de bugs). Não possui opções

Checagens de estado dos pacotes (state match) O critério para filtragem de pacotes mais útil é provido pela extensão 'state' que interpreta a análise do controle da conexão feita pelo módulo 'ip_conntrack'. Essa extensão é altamente recomendada.

Especificando '-m state' a opção '–state' torna-se disponível, a qual é uma lista dos estados possíveis dos pacotes separada por vírgula (a opção '!' indica para que a regra **não** se associe com os estados passados). Os estados são:

NEW
: Um pacote que cria uma nova conexão.

ESTABLISHED
: Um pacote que pertence a uma conexão existente (um pacote de resposta, um pacote saindo por uma conexão na qual já houveram respostas).

RELATED
: Um pacote relacionado, mas que não faz parte, de uma conexão existente, como um erro ICMP, ou (com o módulo FTP carregado), um pacote estabelecendo uma conexão de dados FTP.

INVALID
: Um pacote que não pôde ser identificado por alguma razão: isso inclui falta de memória e erros ICMP que não correspondem a qualquer conexão conhecida. Geralmente tais pacotes devem ser descartados.

7.4 Especificações de alvo (Target)

Agora que sabemos quais as análises podem ser feitas em um pacotes, precisamos de uma forma de dizer o que fazer com os pacotes que passam nas condições que estabelecemos. Isso é chamado de **alvo (target)** da regra.

Há dois alvos bem simples: DROP (descartar) e ACCEPT (aceitar). Nós já os vimos. Se a regra se associa com o pacote e seu alvo é um desses dois, nenhuma outra regra é consultada: o destino do pacote já foi decidido.

Há dois tipos de alvos diferentes dos descritos acima: as extensões e as chains definidas por usuários.

7.4.1 Chains definidas por usuários

Uma funcionalidade que o `iptables` herdou do `ipchains` é a possibilidade do usuário criar novas chains, além das três padrão (INPUT, FORWARD e OUTPUT). Por convenção, chains definidas pelo usuário são escritas em minúsculas, diferenciando-as (como criar chains definidas pelo usuário será descrito abaixo 7.5 (Operações em uma chain)).

Quando um pacote associa-se com uma regra cujo alvo (target) é uma chain definida pelo usuário, o pacote passa a ser analizado pelas regras dessa chain definida pelo usuário. Se a chain não decide o que deve ser feito com o pacote, o mesmo volta a ser analizado pela chain padrão que continha a regra que o levava para a chain definida pelo usuário.

Mais arte ASCII. Considere duas chains: `INPUT` (a chain padrão) e `test` (uma chain definida pelo usuário).

```
 'INPUT'                         'test'
----------------------------    ----------------------------
| Regra1: -p ICMP -j DROP  |    | Regra1: -s 192.168.1.1   |
|--------------------------|    |--------------------------|
```

```
| Regra2: -p TCP -j test   |    | Regra2: -d 192.168.1.1   |
|--------------------------|    ----------------------------
| Regra3: -p UDP -j DROP   |
----------------------------
```

Considere um pacote TCP vindo de 192.168.1.1, indo para 1.2.3.4. Ele entra na chain `INPUT` e é testado pela Regra1 - não se associa. Já a Regra2 se associa, e seu alvo é `test`, logo a próxima regra examinada está no início de `test`. A Regra1 na chain `test` se associa, mas não especifica um alvo, então a próxima regra é examinada, Regra2. Ela não se associa, e nós chegamos no final da chain. Então, a chain `INPUT` volta a ser examinada, e como a úlitma regra desta chain que foi examinada foi a Regra2, a regra a ser examinada agora é a Regra3, que também não se associa com o pacote.

Logo, o caminho que o pacote faz é o seguinte:

```
                        v    __________________________
 `INPUT'                |   /    `test'                v
------------------------|--/    -----------------------|----
| Regra1                | /|    | Regra1               |   |
|-----------------------|/-|    |----------------------|---|
| Regra2                /  |    | Regra2               |   |
|--------------------------|    -----------------------v----
| Regra3                /--+___________________________/
------------------------|---
                        v
```

Chains definidas por usuário podem ter como alvo outras chains definidas por usuário (mas não faça loops: seus pacotes serão descartados se entrarem em loop).

7.4.2 Extensões ao iptables: Novos alvos (targets).

O outro tipo de alvo é uma extensão. Uma extensão-alvo consiste em um módulo do kernel, e uma extensão opcional ao `iptables` para prover opções de linha de comando. Há diversas extensões na distribuição padrão do netfilter:

LOG

Esse módulo provê logging dos pacotes submetidos. Possui as seguintes opçoes adicionais: packets. It provides these additional options:

–log-level

Seguido de um número de nível ou nome. Os nome válidos (sensíveis a maiúsculas/minúsculas) são 'debug', 'info', 'notice', 'warning', 'err', 'crit', 'alert' and 'emerg', correspondendo a números de 7 até 0. Veja a página de manual do syslog.conf para uma explicaçao mais detalhada desses níveis.

–log-prefix

Seguido de uma string de até 29 characters, esta será adicionada no início da mensagem de log, permitindo melhor identificação da mesma.

Esse módulo é últil após de uma comparação por limite (limit match), assim, você não enche seus logs.

REJECT

Esse módulo tem o mesmo efeito de 'DROP', a não ser que o remetente tenha enviado uma mensagem de erro ICMP 'port unreachable' (porta inalcançável) A mensagem de erro ICMP não será enviada se (veja RFC 1122):

- O pacote que está sendo filtrado era uma mensagem de erro ICMP no início, ou um tipo desconhecido de ICMP
- O pacote sendo filtrado era um fragmente sem cabeçalho
- Já enviamos muitas mensagens de erro ICMP para o mesmo destinatário recentemente.

REJECT também tem o argumento opcional '-reject-with' que altera o pacote de resposta utilizado: veja a página de manual.

7.4.3 Alvos especiais padrão

Há dois tipos especiais de alvos padrão: `RETURN` e `QUEUE`.

`RETURN` tem o mesmo efeito de chegar ao final da chain: para uma regra numa chain padrão, a política da chain é executada. Para uma chain definida por usuário, a travessia continua na chain anterior, exatamente após a regra que saltou para a chain definida pelo usuário.

`QUEUE` é um alvo especial, que manda o pacote para uma fila para processamento em nível de usuário. Para isso ser útil, dois componentes são necessários:

- um "gerenciador de filas", que trata com a mecânica de passar os pacotes do kernel para o espaço do usuário;
- uma aplicação no espaço do usuário para receber, possivelmente manipular e decidir o que fazer com os pacotes.

O gerenciador de filas padrão para o iptables IPv4 iptables é o módulo ip_queue, que é distribuído com o kernel e é atualmente experimental.

O seguinte é um rápido exemplo de como utilizar o iptables para enfileirar pacotes para processamento em nível de usuário:

```
# modprobe iptable_filter
# modprobe ip_queue
# iptables -A OUTPUT -p icmp -j QUEUE
```

Com essa regra, pacotes ICMP de saída gerados localmente (como, por exemplo, criados pelo ping) são passados ao módulo ip_queue, que então tenta mandar os pacotes para uma aplicação em nível de usuário. Se não há nenhuma aplicação em nível de usuário está esperando, os pacotes são descartados.

Para escrever uma aplicação em nível de usuário, utilize a API libipq, que é distribuída com o iptables. Códigos de exemplo podem ser encontradas com as ferramentas testsuite (por exemplo redirect.c) no CVS.

O estado de ip_queue pode ser consultado através de:

```
/proc/net/ip_queue
```

O comprimento máximo da fila (exemplo, o número de pacotes entregues ao espaço do usuário sem que nenhuma resposta fosse passada) pode ser controlado por:

```
/proc/sys/net/ipv4/ip_queue_maxlen
```

O valor padrão para o comprimento máximo da fila é 1024. Uma vez que esse limite foi atingido, novos pacotes serão descartados até que o tamanho da fila diminua abaixo do limite. Bons protocolos como o TCP interpretam pacotes descartados como congestionamento, que talvez voltem quando a fila diminua. De qualquer forma, experimentos para determinar o tamanho ideal da fila pois às vezes o tamanho padrão é muito pequeno.

7.5 Operações em uma chain

Uma funcionalidade muito importante do `iptables` é a habilidade de ajuntar regras em chains (cadeias). Você pode dar o nome que quiser para as chains, mas é recomendada a utilização de minúsculas para evitar confusão com as chains padrão ou com os alvos (targets). Nomes de chains podem ter até 31 letras.

7.5.1 Criando uma nova chain

Criemos uma nova chain. Devido à minha imensa criatividade, vou chamá-la de `test`. Utiliza-se as opções '-N' ou '–new-chain':

```
# iptables -N test
#
```

É tão simples assim. Agora é possível acrescentar regras conforme descrito em todo o documento.

7.5.2 Apagando uma Chain

Apagar uma chain é tão simples quanto criá-la, utilizando as opções '-X' ou '–delete-chain'. Porque '-X'? Bem, todas as boas letras já haviam sido utilizadas.

```
# iptables -X test
#
```

Há uma série de restrições ao se apagar uma chain: ela deve estar vazia (veja 7.5.3 (Esvaziando uma chain), logo abaixo) e ela não deve ser alvo de NENHUMA regra. É impossível apagar nenhuma das três chains padrão.

Se uma chain não for especificada, *todas* as chains definidas por usuário serão apagadas, desde que seja possível.

7.5.3 Esvaziando uma chain

Há uma forma muito simples de retirar todas as regras de uma chain, utilizando as opções '-F' (ou '–flush').

```
# iptables -F FORWARD
#
```

Se uma chain nao for especificada, *todas* as chains serão esvaziadas.

7.5.4 Listando uma chain

Pode-se listar todas as regras de uma chain utilizando as opções '-L' (ou '–list')

O valor 'refcnt' listado para cada chain definida por usuário é o número de regras que têm tal chain como alvo (target). Este valor deve ser zero (e a chain deve estar vazia) antes que essa chain possa ser apagada.

Se o nome da chain é omitido, todas as chains são listadas, até as vazias.

Há três opçoes que podem acompanhar '-L'. A opção '-n' (numérico) é muito útil uma vez que previne que o `iptables` tente resolver os endereços IP, o que (se você utiliza DNS assim como a maioria das pessoas) causaria grandes atrasos se o DNS não está configurado corretamente, ou você está filtrando requisições DNS. Essa opção também faz as portas TCP e UDP serem impressas como números em vez de nomes.

A opção '-v' mostra todos os detalhes das regras, como os contadores de pacotes e bytes, as comparações TOS, e as interfaces. Caso tal opção não seja passada, esses valores sao omitidos.

Note que os contadores de pacotes e bytes são impressos com os sufixos 'K', 'M' ou 'G' para 1.000, 1.000.000 e 1.000.000.000 respectivamente. Utilizando a flag '-x' (expandir números) os números serão impressos inteiros, independente de seu tamanho.

7.5.5 Zerando contadores

É útil zerar os contadores. Isso pode ser feito com a opção '-Z' (ou '–zero').

Considere o seguinte:

```
# iptables -L FORWARD
# iptables -Z FORWARD
#
```

No exemplo acima, alguns pacotes passariam pelos comandos '-L' e '-Z'. Por isso, você pode utilizar '-L' e '-Z' *em conjunto*, para zerar os contadores enquantos estes são lidos.

7.5.6 Escolhendo política

Já discutimos sobre o que ocorre quando um pacote chega ao fim de uma chain padrão quando o caminho dos pacotes através das chains. Nesse caso, a **política** da chain determina o destino do pacote. Apenas as chains padrão (`INPUT`, `OUTPUT` e `FORWARD`) têm políticas, porque se um pacote chega ao fim de uma chain definida por usuário, o caminho do pacote continua pela chain anterior.

A política pode ser tanto `ACCEPT (aceitar)` quanto `DROP (rejeitar)`, por exemplo:

```
# iptables -P FORWARD DROP
#
```

8 Utilizando ipchains e ipfwadm

Há módulos na distribuição do netfilter chamados ipchains.o e ipfwadm.o. Carregue-os em seu kernel (NOTA: eles são incompatíveis com ip_tables.o!). Entao você poderá usar ipchains ou ipfwadm como nos bons e velhos tempos.

Isso ainda terá suporte por mais algum tempo. Penso que uma fórmula razoável seria 2 * [anúncio de substituição - versão estável inicial], depois dessa data que uma versão estável da substituição está disponível.

Isso significa que o fim do suporte para o ipfwadm é:

```
2 * [Outubro de 1997 (versão 2.1.102) - Março 1995 (ipfwadm 1.0)]
        + Janeiro 1999 (versão 2.2.0)
    = Março 2004.
```

Isso significa que o fim do suporte para o ipfwadm é:

```
2 * [Agosto 1999 (versão 2.3.15) - Outubro 1997 (versão 2.2.0)]
        + Julho 2000 (versão 2.4.0?)
    = Março 2004.
```

Logo não se preocupe até 2004.

9 Misturando NAT e Filtragem de Pacotes

É muito comum a utilização de Network Address Translation (veja o NAT HOWTO) em conjunto com a filtragem de pacotes. A boa notícia é que eles se misturam muito bem.

Primeiramente, você projeta e constrói seu filtro de pacotes ignorando totalmente qualquer NAT a ser feito. As origens e destinos que o filtro de pacotes verá serão as origens e destinos reais. Por exemplo, se você fará DNAT para mandar conexões do endereço 1.2.3.4 porta 80 para 10.1.1.1 porta 8080, o filtro de pacotes verá pacotes indo para 10.1.1.1 porta 8080 (o destino real), e não 1.2.3.4 porta 80. Similarmente, você pode ignorar o masquerading: os pacotes vão parecer que têm como origem o real enderço IP interno (por exemplo 10.1.1.1), e as respostas vão parecer que têm como destino esse mesmo endereço.

Pode-se utilizar a extensão de checagem do estado dos pacotes (state match) sem fazer com que o filtro de pacotes tenha qualquer trabalho extra, uma vez que NAT já requer acompanhamento de conexão. Para melhorar o simples exemplo de masquerading descrito no NAT HOWTO a fim de rejeitar quaisquer novas conexões vindo na interface ppp0, isso seria feito:

```
# Fazer masquerade pela interface ppp0
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

# Rejeitar conexões novas (NEW) e inválidas (INVALID) de pacotes com destino à
maquina local ou que devem ser repassados vindos de ppp0.
iptables -A INPUT -i ppp0 -m state --state NEW,INVALID -j DROP
iptables -A FORWARD -i ppp0 -m state --state NEW,INVALID -j DROP

# Habilitar IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

10 Diferenças entre iptables e ipchains

- Primeiramente, os nomes das chains padrão passaram a ser escritos com MAIÚSCULAS em vez de minúsculas, porque as chains INPUT e OUTPUT apenas recebem pacotes destinados localmente e gerados localmente. Antes, essas chains viam todos os pacotes entrando e saindo, respectivamente.
- A flag '-i' agora significa interface de entrada, e apenas funciona nas chains INPUT e FORWARD. Regras nas chains FORWARD e OUTPUT que utilizavam '-i' devem ser alteradas para '-o'.
- Portas TCP e UDP agora precisam ser descritas com as opções –source-port ou –sport (ou –destination-port/–dport), que devem se colocadas depois das opções '-p tcp' ou '-p udp', já que essas carregam as extensões TCP ou UDP respectivamente.
- A flag TCP -y agora é –syn, e deve ser posicionada depois de '-p tcp'.
- Finalmente o alvo DENY agora é DROP.
- Zerar chains enquanto listando-as funciona.
- Zerar chains padrão também apaga os contadores de suas políticas.
- Listar as chains também mostra os contadores.
- REJECT e LOG são agora alvos-extensões, ou seja, eles são módulos do kernel separados
- Nomes das chains podem ter até 31 caracteres.
- MASQ agora é MASQUERADE e utiliza uma sintaxe diferente. REDIRECT, embora tenha mantido o nome, também sofreu uma mudança de sintaxe. Veja o NAT HOWTO para mais informaçoes sobre como configurar ambos.
- A opção -o não é mais utilizada para mandar os pacotes para o dispositivo userspace (veja -i acima). Pacotes agora são mandados para o espaço do usuário com o alvo QUEUE.
- Provavelmente milhares de outras coisas que esqueci.

11 Conselhos sobre o projeto de filtros de pacotes

Um desejo comum na área de segurança de computadores é bloquear tudo, e então abrir os buracos necessários. Isso pode ser chamado de "o que não está explicitamente permitido, é proíbido". Eu recomendo essa visão para uma maior segurança.

Não rode nenhum serviço desnecessário, mesmo que você ache que bloqueou o acesso aos mesmos

Se você está criando um firewall dedicado, comece sem rodar qualquer serviço, e bloqueando todos os pacotes, e então adicione serviços e vá deixando os pacotes passarem conforme necessário.

Recomendo segurança profunda: combine tcp-wrappers (para conexões na própria máquina do filtro de pacotes), proxies (para conexões que apenas passam pelo filtro de pacotes), verificação de rotas e filtragem de pacotes. Verificação de rotas é quando um pacote vem de uma interface inexperada ele é descartado: por exemplo, se a sua LAN possui endereços 10.1.1.0/24, e um pacote com essa origem vem pela sua interface externa, ele será descartado. Isso pode ser habilitado para uma interface (ppp0) conforme descrito abaixo:

```
# echo 1 > /proc/sys/net/ipv4/conf/ppp0/rp_filter
#
```

Ou para todas as interfaces existentos e futuras conforme abaixo:

```
# for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
#     echo 1 > $f
# done
#
```

Debian faz isso por padrão sempre que possível. Se você tem roteamento assimétrico (exemplo, você espera pacotes vindos de locais estranhos) você vai querer desabilitar esse tipo de filtragem nessas interfaces.

Registrar (logging) é muito útil em um firewall para saber se algo não está funcionando corretamente, mas em um firewall de produção é sempre interessante utilizar o controle de limite (limit match), a fim de prevenir que alguém faça flood em seus logs.

É altamente recomendado o acompanhamento de conexões para sistemas seguros: isso provoca um pouco de overhead, uma vez que todas as conexões são acompanhadas, mas é muito útil para controlar os acessos à sua rede. Você terá de carregar o módulo 'ip_conntrack.o' se seu kernel não carrega módulos automaticamente, ou ele não está compilado diretamente no kernel. Se você quer acompanhar conexões de protocolos complexos, você terá de carregar um módulo auxiliar apropriado (por exemplo, 'ip_conntrack_ftp.o').

```
# iptables -N no-conns-from-ppp0
# iptables -A no-conns-from-ppp0 -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A no-conns-from-ppp0 -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A no-conns-from-ppp0 -i ppp0 -m limit -j LOG --log-prefix "Bad packet from ppp0:"
# iptables -A no-conns-from-ppp0 -i ! ppp0 -m limit -j LOG --log-prefix "Bad packet not from ppp0:"
# iptables -A no-conns-from-ppp0 -j DROP

# iptables -A INPUT -j no-conns-from-ppp0
# iptables -A FORWARD -j no-conns-from-ppp0
```

Construir um bom firewall está além do escopo deste HOWTO, mas meu conselho é "sempre seja detalhista". Veja o HOWTO Segurança (Security HOWTO) para mais informações sobre como testar sua máquina.

Tradução para o Português do Brasil: Guilherme Ude Braz (guilherme@linuxplace.com.br).