[anterior] [Conteúdo] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [próximo]

Guia Foca GNU/Linux
Capítulo 10 - Firewall iptables

Este capítulo documenta o funcionamento do firewall `iptables` que acompanha a série do kernel 2.4, opções usadas, e aponta alguns pontos fundamentais para iniciar a configuração e construção de bons sistemas de firewall.

10.1 Introdução

O *Firewall* é um programa que como objetivo proteger a máquina contra acessos indesejados, tráfego indesejado, proteger serviços que estejam rodando na máquina e bloquear a passagem de coisas que você não deseja receber (como conexões vindas da Internet para sua segura rede local, evitando acesso aos dados corporativos de uma empresa ou a seus dados pessoais). No kernel do Linux 2.4, foi introduzido o firewall `iptables` (também chamado de netfilter) que substitui o `ipchains` dos kernels da série 2.2. Este novo firewall tem como vantagem ser muito estável (assim como o `ipchains` e `ipfwadm`), confiável, permitir muita flexibilidade na programação de regras pelo administrador do sistema, mais opções disponíveis ao administrador para controle de tráfego, controle independente do tráfego da rede local/entre redes/interfaces devido a nova organização das etapas de roteamento de pacotes.

O `iptables` é um firewall em nível de pacotes e funciona baseado no endereço/porta de origem/destino do pacote, prioridade, etc. Ele funciona através da comparação de regras para saber se um pacote tem ou não permissão para passar. Em firewalls mais restritivos, o pacote é bloqueado e registrado para que o administrador do sistema tenha conhecimento sobre o que está acontecendo em seu sistema.

Ele também pode ser usado para modificar e monitorar o tráfego da rede, fazer NAT (masquerading, source nat, destination nat), redirecionamento de pacotes, marcação de pacotes, modificar a prioridade de pacotes que chegam/saem do seu sistema, contagem de bytes, dividir tráfego entre máquinas, criar proteções anti-spoofing, contra syn flood, DoS, etc. O tráfego vindo de máquinas desconhecidas da rede pode também ser bloqueado/registrado através do uso de simples regras. As possibilidades oferecidas pelos recursos de filtragem `iptables` como todas as ferramentas UNIX maduras dependem de sua imaginação, pois ele garante uma grande flexibilidade na manipulação das regras de acesso ao sistema, precisando apenas conhecer quais interfaces o sistema possui, o que deseja bloquear, o que tem acesso garantido, quais serviços devem estar acessíveis para cada rede, e iniciar a construção de seu firewall.

O `iptables` ainda tem a vantagem de ser modularizável, funções podem ser adicionadas ao firewall ampliando as possibilidades oferecidas. Usei por 2 anos o `ipchains` e afirmo que este é um firewall que tem possibilidades de gerenciar tanto a segurança em máquinas isoladas como roteamento em grandes organizações, onde a passagem de tráfego entre redes deve ser minuciosamente controlada.

Um firewall não funciona de forma automática (instalando e esperar que ele faça as coisas por você), é necessário pelo menos conhecimentos básicos de rede tcp/ip, roteamento e portas para criar as regras que farão a segurança de seu sistema. A segurança do sistema depende do controle das regras que serão criadas por você, as falhas humanas são garantia de mais de 95% de sucesso nas invasões.

Enfim o `iptables` é um firewall que agradará tanto a pessoas que desejam uma segurança básica em seu sistema, quando administradores de grandes redes que querem ter um controle minucioso sobre o tráfego que passam entre suas interfaces de rede (controlando tudo o que pode passar de uma rede a outra), controlar o uso de tráfego, monitoração, etc.

10.1.1 Versão

É assumido que esteja usando a versão 1.2.3 do `iptables` e baseadas nas opções do kernel 2.4.16 (sem o uso de módulos experimentais). As explicações contidas aqui podem funcionar para versões posteriores, mas é recomendável que leia a documentação sobre modificações no programa (changelog) em busca de mudanças que alterem o sentido das explicações fornecidas aqui.

10.1.2 Um resumo da história do iptables

O `iptables` é um código de firewall das versões 2.4 do kernel, que substituiu o `ipchains` (presente nas séries 2.2 do kernel). Ele foi incluído no kernel da série 2.4 em meados de Junho/Julho de 1999.

A história do desenvolvimento (desde o porte do `ipfw` do `BSD` para o `Linux` até o `iptables` (que é a quarta geração de firewalls do kernel) está disponível no documento, `Netfilter-howto`.

10.1.3 Características do firewall iptables

- Especificação de portas/endereço de origem/destino
- Suporte a protocolos TCP/UDP/ICMP (incluindo tipos de mensagens icmp)
- Suporte a interfaces de origem/destino de pacotes
- Manipula serviços de proxy na rede
- Tratamento de tráfego dividido em chains (para melhor controle do tráfego que entra/sai da máquina e tráfego redirecionado.
- Permite um número ilimitado de regras por chain
- Muito rápido, estável e seguro
- Possui mecanismos internos para rejeitar automaticamente pacotes duvidosos ou mal formados.
- Suporte a módulos externos para expansão das funcionalidades padrões oferecidas pelo código de firewall
- Suporte completo a roteamento de pacotes, tratadas em uma área diferente de tráfegos padrões.
- Suporte a especificação de tipo de serviço para priorizar o tráfego de determinados tipos de pacotes.
- Permite especificar exceções para as regras ou parte das regras
- Suporte a detecção de fragmentos
- Permite enviar alertas personalizados ao `syslog` sobre o tráfego aceito/bloqueado.
- Redirecionamento de portas
- Masquerading
- Suporte a SNAT (modificação do endereço de origem das máquinas para um único IP ou faixa de IP's).
- Suporte a DNAT (modificação do endereço de destino das máquinas para um único IP ou fixa de IP's)
- Contagem de pacotes que atravessaram uma interface/regra
- Limitação de passagem de pacotes/conferência de regra (muito útil para criar proteções contra, syn flood, ping flood, DoS, etc).

10.1.4 Ficha técnica

Pacote: `iptables`

- `iptables` - Sistema de controle principal para protocolos ipv4
- `ip6tables` - Sistema de controle principal para protocolos ipv6
- `iptables-save` - Salva as regras atuais em um arquivo especificado como argumento. Este utilitário pode ser dispensado por um shell script contendo as regras executado na inicialização da máquina.
- `iptables-restore` - Restaura regras salvas pelo utilitário `iptables-save`.

10.1.5 Requerimentos

É necessário que o seu kernel tenha sido compilado com suporte ao `iptables` (veja Habilitando o suporte ao iptables no kernel, Seção 10.1.15. O requerimento mínimo de memória necessária para a execução do `iptables` é o mesmo do kernel 2.4 (4MB). Dependendo do tráfego que será manipulado pela(s) interface(s) do firewall ele poderá ser executado com folga em uma máquina 386 SX com 4MB de RAM.

Como as configurações residem no kernel não é necessário espaço extra em disco rígido para a execução deste utilitário.

10.1.6 Arquivos de logs criados pelo iptables

Todo tráfego que for registrado pelo `iptables` é registrado por padrão no arquivo `/var/log/kern.log`.

10.1.7 Instalação

```
apt-get install iptables
```

O pacote `iptables` contém o utilitário `iptables` (e `ip6tables` para redes ipv6) necessários para inserir suas regras no kernel. Se você não sabe o que é ipv6, não precisará se preocupar com o utilitário `ip6tables` por enquanto.

10.1.8 Enviando Correções/Contribuindo com o projeto

A página principal do projeto é `http://netfilter.filewatcher.org`. Sugestões podem ser enviadas para a lista de desenvolvimento oficial do `iptables`: `http://lists.samba.org`.

10.1.9 O que aconteceu com o `ipchains` e `ipfwadm`?

O `iptables` faz parte da nova geração de firewalls que acompanha o kernel 2.4, mas o suporte ao `ipchains` e `ipfwadm` ainda será mantido através de módulos de compatibilidade do kernel até 2004. Seria uma grande falta de consideração retirar o suporte a estes firewalls do kernel como forma de obrigar a "aprenderem" o `iptables` (mesmo o suporte sendo removido após este período, acredito que criarão patches "externos" para futuros kernels que não trarão mais este suporte). Se precisa do suporte a estes firewalls antes de passar em definitivo para o `iptables` leia Habilitando o suporte ao iptables no kernel, Seção 10.1.15.

Se você é um administrador que gosta de explorar todos os recursos de um firewall, usa todos os recursos que ele oferece ou mantém uma complexa rede corporativa, tenho certeza que gostará do `iptables`.

10.1.10 Tipos de firewalls

Existem basicamente dois tipos de firewalls:

- `nível de aplicação` - Este tipo de firewall analisam o conteúdo do pacote para tomar suas decisões de filtragem. Firewalls deste tipo são mais intrusivos (pois analisam o conteúdo de tudo que passa por ele) e permitem um controle relacionado com o conteúdo do tráfego. Alguns firewalls em nível de aplicação combinam recursos básicos existentes em firewalls em nível de pacotes combinando as funcionalidade de controle de tráfego/controle de acesso em uma só ferramenta. Servidores proxy, como o `squid`, são um exemplo deste tipo de firewall.
- `nível de pacotes` - Este tipo de firewall toma as decisões baseadas nos parâmetros do pacote, como porta/endereço de origem/destino, estado da conexão, e outros parâmetros do pacote. O firewall então pode negar o pacote (DROP) ou deixar o pacote passar (ACCEPT). O `iptables` é um excelente firewall que se encaixa nesta categoria.

  Firewall em nível de pacotes é o assunto explicado nesta seção do guia mas será apresentada uma explicação breve sobre o funcionamento de análise de strings do `iptables`.

Os dois tipos de firewalls podem ser usados em conjunto para fornecer uma camada dupla de segurança no acesso as suas máquinas/máquinas clientes.

10.1.11 O que proteger?

Antes de iniciar a construção do firewall é bom pensar nos seguintes pontos:

- Quais serviços precisa proteger. Serviços que devem ter acesso garantido a usuários externos e quais serão bloqueados a todas/determinadas máquinas. É recomendável bloquear o acesso a todas portas menores que 1024 por executarem serviços que rodam com privilégio de usuário `root`, e autorizar somente o acesso as portas que realmente deseja (configuração restritiva nesta faixa de portas).
- Que tipo de conexões eu posso deixar passar e quais bloquear. Serviços com autenticação em texto plano e potencialmente inseguros como rlogin, telnet, ftp, NFS, DNS, LDAP, SMTP RCP, X-Window são serviços que devem ser ter acesso garantido somente para máquinas/redes que você confia. Estes serviços podem não ser só usados para tentativa de acesso ao seu sistema, mas também como forma de atacar outras pessoas aproveitando-se de problemas de configuração.

  A configuração do firewall ajuda a prevenir isso, mesmo se um serviço estiver mal configurado e tentando enviar seus pacotes para fora, será impedido. Da mesma forma se uma máquina Windows de sua rede for infectada por um trojan não haverá pânico: o firewall poderá estar configurado para bloquear qualquer tentativa de conexão vinda da internet (cracker) para as máquinas de sua rede.

  Para cópia de arquivos via rede insegura (como através da Internet), é recomendado o uso de serviços que utilizam criptografia para login e transferência de arquivos (veja Servidor ssh, Capítulo 15) ou a configuração de uma VPN.
- Que máquinas terão acesso livre e quais serão restritas.
- Que serviços deverão ter prioridade no processamento.
- Que máquinas/redes NUNCA deverão ter acesso a certas/todas máquinas.
- O volume de tráfego que o servidor manipulará. Através disso você pode ter que balancear o tráfego entre outras máquinas, configurar proteções contra DoS, syn flood, etc.
- O que tem permissão de passar de uma rede para outra (em máquinas que atuam como roteadores/gateways de uma rede interna).
- Etc.

A análise destes pontos pode determinar a complexidade do firewall, custos de implementação, prazo de desenvolvimento e tempo de maturidade do código para implementação. Existem muitos outros pontos que podem entrar na questão de desenvolvimento de um sistema de firewall, eles dependem do tipo de firewall que está desenvolvendo e das políticas de segurança de sua rede.

10.1.12 O que são regras?

As regras são como comandos passados ao `iptables` para que ele realize uma determinada ação (como bloquear ou deixar passar um pacote) de acordo com o endereço/porta de origem/destino, interface de origem/destino, etc. As regras são armazenadas dentro dos chains e processadas na ordem que são inseridas.

As regras são armazenadas no kernel, o que significa que quando o computador for reiniciado tudo o que fez será perdido. Por este motivo elas deverão ser gravadas em um arquivo para serem carregadas a cada inicialização.

Um exemplo de regra: `iptables -A INPUT -s 123.123.123.1 -j DROP`.

10.1.13 O que são chains?

Os *Chains* são locais onde as regras do firewall definidas pelo usuário são armazenadas para operação do firewall. Existem dois tipos de chains: os embutidos (como os chains *INPUT*, *OUTPUT* e *FORWARD*) e os criados pelo usuário. Os nomes dos chains embutidos devem ser especificados sempre em maiúsculas (note que os nomes dos chains são case-sensitive, ou seja, o chain `input` é completamente diferente de `INPUT`).

10.1.14 O que são tabelas?

Tabelas são os locais usados para armazenar os chains e conjunto de regras com uma determinada característica em comum. As tabelas podem ser referenciadas com a opção *-t tabela* e existem 3 tabelas disponíveis no `iptables`:

- `filter` - Esta é a tabela padrão, contém 3 chains padrões:
  - `INPUT` - Consultado para dados que chegam a máquina
  - `OUTPUT` - Consultado para dados que saem da máquina
  - `FORWARD` - Consultado para dados que são redirecionados para outra interface de rede ou outra máquina.

  Os chains *INPUT* e *OUTPUT* somente são atravessados por conexões indo/se originando de localhost.

  OBS: Para conexões locais, somente os chains *INPUT* e *OUTPUT* são consultados na tabela filter.
- `nat` - Usada para dados que gera outra conexão (masquerading, source nat, destination nat, port forwarding, proxy transparente são alguns exemplos). Possui 3 chains padrões:
  - `PREROUTING` - Consultado quando os pacotes precisam ser modificados logo que chegam. É o chain ideal para realização de DNAT e redirecionamento de portas (Fazendo DNAT, Seção 10.4.4).
  - `OUTPUT` - Consultado quando os pacotes gerados localmente precisam ser modificados antes de serem roteados. Este chain somente é consultado para conexões que se originam de IPs de interfaces locais.
  - `POSTROUTING` - Consultado quando os pacotes precisam ser modificados após o tratamento de roteamento. É o chain ideal para realização de SNAT e IP Masquerading (Fazendo SNAT, Seção 10.4.3).
- `mangle` - Utilizada para alterações especiais de pacotes (como modificar o tipo de serviço (TOS) ou outros detalhes que serão explicados no decorrer do capítulo. Possui 2 chains padrões:
  - `INPUT` - Consultado quando os pacotes precisam ser modificados antes de serem enviados para o chain *INPUT* da tabela *filter*.
  - `FORWARD` - Consultado quando os pacotes precisam ser modificados antes de serem enviados para o chain *FORWARD* da tabela *filter*.

- PREROUTING - Consultado quando os pacotes precisam ser modificados antes de ser enviados para o chain *PREROUTING* da tabela *nat*.
- POSTROUTING - Consultado quando os pacotes precisam ser modificados antes de serem enviados para o chain *POSTROUTING* da tabela *nat*.
- OUTPUT - Consultado quando os pacotes precisam ser modificados antes de serem enviados para o chain *OUTPUT* da tabela *nat*.

Veja A tabela mangle, Seção 10.5 para mais detalhes sobre a tabela mangle.

10.1.15 Habilitando o suporte ao iptables no kernel

Para usar toda a funcionalidade do firewall iptables, permitindo fazer o controle do que tem ou não permissão de acessar sua máquina, fazer Masquerading/NAT em sua rede, etc., você precisará dos seguintes componentes compilados em seu kernel (os módulos experimentais fora ignorados intencionalmente):

```
*
* Network Options:
*

Network packet filtering (replaces ipchains) [Y/m/n/?]
 Network packet filtering debugging [Y/m/n/?]

e na Subseção:

*
*   IP: Netfilter Configuration
*
Connection tracking (required for masq/NAT) (CONFIG_IP_NF_CONNTRACK) [M/n/y/?]
  FTP protocol support (CONFIG_IP_NF_FTP) [M/n/?]
  IRC protocol support (CONFIG_IP_NF_IRC) [M/n/?]
IP tables support (required for filtering/masq/NAT) (CONFIG_IP_NF_IPTABLES) [Y/m/n/?]
  limit match support (CONFIG_IP_NF_MATCH_LIMIT) [Y/m/n/?]
  MAC address match support (CONFIG_IP_NF_MATCH_MAC) [M/n/y/?]
  netfilter MARK match support (CONFIG_IP_NF_MATCH_MARK) [M/n/y/?]
  Multiple port match support (CONFIG_IP_NF_MATCH_MULTIPORT) [M/n/y/?]
  TOS match support (CONFIG_IP_NF_MATCH_TOS) [M/n/y/?]
  LENGTH match support (CONFIG_IP_NF_MATCH_LENGTH) [M/n/y/?]
  TTL match support (CONFIG_IP_NF_TTL) [M/n/y/?]
  tcpmss match support (CONFIG_IP_NF_MATCH_TCPMSS) [M/n/y/?]
  Connection state match support (CONFIG_IP_NF_MATCH_STATE) [M/n/?]
  Packet filtering (CONFIG_IP_NF_FILTER) [M/n/y/?]
    REJECT target support (CONFIG_IP_NF_TARGET_REJECT) [M/n/?]
  Full NAT (CONFIG_IP_NF_NAT) [M/n/?]
    MASQUERADE target support (CONFIG_IP_NF_TARGET_MASQUERADE) [M/n/?]
    REDIRECT target support (CONFIG_IP_NF_TARGET_REDIRECT) [M/n/?]
  Packet mangling (CONFIG_IP_NF_MANGLE) [M/n/y/?]
    TOS target support (CONFIG_IP_NF_TARGET_TOS) [M/n/?]
    MARK target support (CONFIG_IP_NF_TARGET_MARK) [M/n/?]
  LOG target support (CONFIG_IP_NF_TARGET_LOG) [M/n/y/?]
  TCPMSS target support (CONFIG_IP_NF_TARGET_TCPMSS) [M/n/y/?]
```

Esta configuração permite que você não tenha problemas para iniciar o uso e configuração do seu firewall iptables, ela ativa os módulos necessários para utilização de todos os recursos do firewall iptables. Quando conhecer a função de cada um dos parâmetros acima (durante o decorrer do texto), você poderá eliminar muitas das opções desnecessárias para seu estilo de firewall ou continuar fazendo uso de todas ;-)

OBS1: A configuração acima leva em consideração que você NÃO executará os códigos antigos de firewall ipfwadm e ipchains. Caso deseje utilizar o ipchains ou o ipfwadm, será preciso responder com "M" a questão "IP tables support (required for filtering/masq/NAT) (CONFIG_IP_NF_IPTABLES)". Será necessário carregar manualmente o módulo correspondente ao firewall que deseja utilizar (modprobe iptables_filter.o no caso do iptables).

Não execute mais de um tipo de firewall ao mesmo tempo!!!

OBS2: É recomendável ativar o daemon kmod para carga automática de módulos, caso contrário será necessário compilar todas as partes necessárias embutidas no kernel, carregar os módulos necessários manualmente ou pelo iptables (através da opção *--modprobe=módulo*).

10.1.16 Ligando sua rede interna a Internet

Se a sua intenção (como da maioria dos usuários) é conectar sua rede interna a Internet de forma rápida e simples, leia Fazendo IP masquerading (para os apressados), Seção 10.4.2 ou Fazendo SNAT, Seção 10.4.3. Um exemplo prático de configuração de Masquerading deste tipo é encontrado em Conectando sua rede interna a Internet, Seção 10.8.3.

Após configurar o masquerading, você só precisará especificar o endereço IP da máquina masquerading (servidor) como *Gateway* da rede. No Windows 9x/NT/2000 isto é feito no Painel de Controle/Rede /Propriedades de Tcp/IP. No Linux pode ser feito com route add default gw IP_do_Servidor.

10.2 Manipulando chains

O iptables trabalha com uma tabela de regras que é analisada uma a uma até que a última seja processada. Por padrão, se uma regra tiver qualquer erro, uma mensagem será mostrada e ela descartada. O pacote não conferirá e a ação final (se ele vai ser aceito ou rejeitado) dependerá das regras seguintes.

As opções passadas ao iptables usadas para manipular os chains são **SEMPRE** em maiúsculas. As seguintes operações podem ser realizadas:

10.2.1 Adicionando regras - A

Como exemplo vamos criar uma regra que bloqueia o acesso a nosso própria máquina (127.0.0.1 - loopback). Primeiro daremos um ping para verificar seu funcionamento:

```
#ping 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=255 time=0.6 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.5 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.5/0.5/0.6 ms
```

Ok, a máquina responde, agora vamos incluir uma regra no chain INPUT (*-A INPUT*) que bloqueie (*-j DROP*) qualquer acesso indo ao endereço 127.0.0.1 (*-d 127.0.0.1*):

```
iptables -t filter -A INPUT -d 127.0.0.1 -j DROP
```

Agora verificamos um novo ping:

```
#ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes

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

Desta vez a máquina 127.0.0.1 não respondeu, pois todos os pacotes com o destino 127.0.0.1 (-d 127.0.0.1) são rejeitados (-j DROP). A opção *-A* é usada para adicionar novas regras no final do chain. Além de *-j DROP* que serve para rejeitar os pacotes, podemos também usar *-j ACCEPT* para aceitar pacotes. A opção -j é chamada de *alvo da regra* ou somente *alvo* pois define o destino do pacote que atravessa a regra (veja Especificando um alvo, Seção 10.3.6). Bem vindo a base de um sistema de firewall :-)

OBS1: - O acesso a interface loopback não deve ser de forma alguma bloqueado, pois muitos aplicativos utilizam soquetes tcp para realizarem conexões, mesmo que você não possua uma rede interna.

OBS2: - A tabela *filter* será usada como padrão caso nenhuma tabela seja especificada através da opção -t.

10.2.2 Listando regras - L

A seguinte sintaxe é usada para listar as regras criadas:

```
iptables [-t tabela] -L [chain] [opções]
```

Onde:

tabela

É uma das tabelas usadas pelo iptables. Se a tabela não for especificada, a tabela *filter* será usada como padrão. Veja O que são tabelas?, Seção 10.1.14 para detalhes.

chain

Um dos chains disponíveis na tabela acima (veja O que são tabelas?, Seção 10.1.14) ou criado pelo usuário (Criando um novo chain - N, Seção 10.2.6). Caso o chain não seja especificado, todos os chains da tabela serão mostrados.

opções

As seguintes opções podem ser usadas para listar o conteúdo de chains:

- `-v` - Exibe mais detalhes sobre as regras criadas nos chains.
- `-n` - Exibe endereços de máquinas/portas como números ao invés de tentar a resolução DNS e consulta ao `/etc/services`. A resolução de nomes pode tomar muito tempo dependendo da quantidade de regras que suas tabelas possuem e velocidade de sua conexão.
- `-x` - Exibe números exatos ao invés de números redondos. Também mostra a faixa de portas de uma regra de firewall.
- `--line-numbers` - Exibe o número da posição da regra na primeira coluna da listagem.

Para listar a regra criada anteriormente usamos o comando:

```
#iptables -t filter -L INPUT

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             localhost
```

O comando `iptables -L INPUT -n` tem o mesmo efeito, a diferença é que são mostrados números ao invés de nomes:

```
#iptables -L INPUT -n

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  0.0.0.0/0            127.0.0.1


#iptables -L INPUT -n --line-numbers

Chain INPUT (policy ACCEPT)
num target     prot opt source               destination
1   DROP       all  --  0.0.0.0/0            127.0.0.1

#iptables -L INPUT -n -v
Chain INPUT (policy ACCEPT 78 packets, 5820 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   194 DROP       icmp --  *      *       0.0.0.0/0            127.0.0.1
```

Os campos assim possuem o seguinte significado:

Chain INPUT

Nome do chain listado

(policy ACCEPT 78 packets, 5820 bytes)

política padrão do chain (veja Especificando a política padrão de um chain - P, Seção 10.2.12).

pkts

Quantidade de pacotes que atravessaram a regra (veja Zerando contador de bytes dos chains - Z, Seção 10.2.11).

bytes

Quantidade de bytes que atravessaram a regra. Pode ser referenciado com K (Kilobytes), M (Megabytes), G (Gigabytes).

target

O alvo da regra, o destino do pacote. Pode ser ACCEPT, DROP ou outro chain. Veja Especificando um alvo, Seção 10.3.6 para detalhes sobre a especificação de um alvo.

prot

Protocolo especificado pela regra. Pode ser udp, tcp, icmp ou all. Veja Especificando um protocolo, Seção 10.3.3 para detalhes.

opt

Opções extras passadas a regra. Normalmente "!" (veja Especificando uma exceção, Seção 10.3.5) ou "f" (veja Especificando fragmentos, Seção 10.3.4).

in

Interface de entrada (de onde os dados chegam). Veja Especificando a interface de origem/destino, Seção 10.3.2.

out

Interface de saída (para onde os dados vão). Veja Especificando a interface de origem/destino, Seção 10.3.2.

source

Endereço de origem. Veja Especificando um endereço de origem/destino, Seção 10.3.1.

destination

Endereço de destino. Veja Especificando um endereço de origem/destino, Seção 10.3.1.

outras opções

Estas opções normalmente aparecem quando são usadas a opção -x:

- `dpt ou dpts` - Especifica a porta ou faixa de portas de destino.
- `reject-with icmp-port-unreachable` - Significa que foi usado o alvo REJECT naquela regra (veja Alvo REJECT, Seção 10.3.6.1).

10.2.3 Apagando uma regra - D

Para apagar um chain, existem duas alternativas:

Quando sabemos qual é o número da regra no chain (listado com a opção `-L`) podemos referenciar o número diretamente. Por exemplo, para apagar a regra criada acima:

```
iptables -t filter -D INPUT 1
```

Esta opção não é boa quando temos um firewall complexo com um grande número de regras por chains, neste caso a segunda opção é a mais apropriada.

Usamos a mesma sintaxe para criar a regra no chain, mas trocamos `-A` por `-D`:

```
iptables -t filter -D INPUT -d 127.0.0.1 -j DROP
```

Então a regra correspondentes no chain INPUT será automaticamente apagada (confira listando o chain com a opção "-L"). Caso o chain possua várias regras semelhantes, somente a primeira será apagada.

OBS: Não é possível apagar os chains defaults do `iptables` (*INPUT*, *OUTPUT*...).

10.2.4 Inserindo uma regra - I

Precisamos que o tráfego vindo de `192.168.1.15` não seja rejeitado pelo nosso firewall. Não podemos adicionar uma nova regra (-A) pois esta seria incluída no final do chain e o tráfego seria rejeitado pela primeira regra (nunca atingindo a segunda). A solução é inserir a nova regra antes da regra que bloqueia todo o tráfego ao endereço `127.0.0.1` na posição 1:

```
iptables -t filter -I INPUT 1 -s 192.168.1.15 -d 127.0.0.1 -j ACCEPT
```

Após este comando, temos a regra inserida na primeira posição do chain (repare no número 1 após INPUT) e a antiga regra número 1 passa a ser a número 2. Desta forma a regra acima será consultada, se a máquina de origem for 192.168.1.15 então o tráfego estará garantido, caso contrário o tráfego com o destino 127.0.0.1 será bloqueado na regra seguinte.

10.2.5 Substituindo uma regra - R

Após criar nossa regra, percebemos que a nossa intenção era somente bloquear os pings com o destino `127.0.0.1` (pacotes ICMP) e não havia necessidade de bloquear todo o tráfego da máquina. Existem duas alternativas: apagar a regra e inserir uma nova no lugar ou modificar diretamente a regra já criada sem afetar outras regras existentes e mantendo a sua ordem no chain (isso é muito importante). Use o seguinte comando:

```
iptables -R INPUT 2 -d 127.0.0.1 -p icmp -j DROP
```

O número `2` é o número da regra que será substituída no chain INPUT, e deve ser especificado. O comando acima substituirá a regra 2 do chain INPUT (-R INPUT 2) bloqueando (-j DROP) qualquer pacote icmp (-p icmp) com o destino `127.0.0.1` (-d 127.0.0.1).

10.2.6 Criando um novo chain - N

Em firewalls organizados com um grande número de regras, é interessante criar chains individuais para organizar regras de um mesmo tipo ou que tenha por objetivo analisar um tráfego de uma mesma categoria (interface, endereço de origem, destino, protocolo, etc) pois podem consumir muitas linhas e tornar o gerenciamento do firewall confuso (e conseqüentemente causar sérios riscos de segurança). O tamanho máximo de um nome de chain é de 31 caracteres e podem conter tanto letras maiúsculas quanto minúsculas.

```
iptables [-t tabela] [-N novochain]
```

Para criar o chain *internet* (que pode ser usado para agrupar as regras de internet) usamos o seguinte comando:

```
iptables -t filter -N internet
```

Para inserir regras no chain *internet* basta especifica-lo após a opção -A:

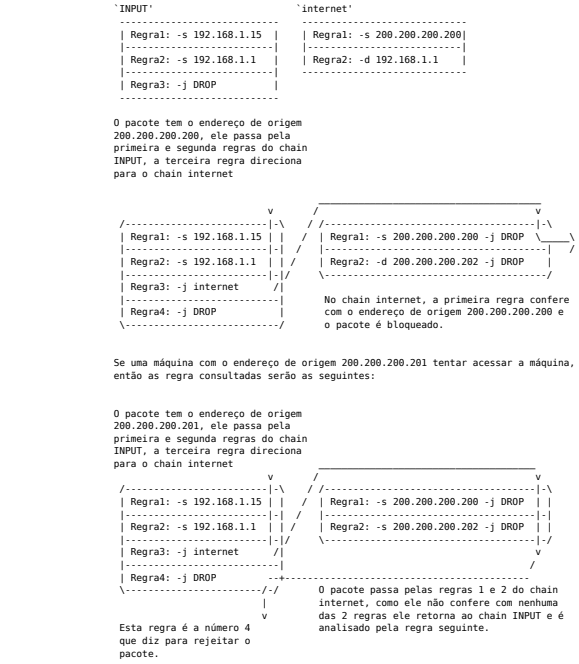
```
iptables -t filter -A internet -s 200.200.200.200 -j DROP
```

E então criamos um pulo (-j) do chain *INPUT* para o chain *internet*:

```
iptables -t filter -A INPUT -j internet
```

OBS: O chain criando pelo usuário pode ter seu nome tanto em maiúsculas como minúsculas.

Se uma máquina do endereço 200.200.200.200 tentar acessar sua máquina, o `iptables` consultará as seguintes regras:



10.2.7 Renomeando um chain criado pelo usuário - E

Se por algum motivo precisar renomear um chain criado por você na tabela *filter*, *nat* ou *mangle*, isto poderá ser feito usando a opção *-E* do `iptables`:

```
iptables -t filter -E chain-antigo novo-chain
```

Note que não é possível renomear os chains defaults do `iptables`.

10.2.8 Listando os nomes de todas as tabelas atuais

Use o comando `cat /proc/net/ip_tables_names` para fazer isto. É interessante dar uma olhada nos arquivos dentro do diretório `/proc/net`, pois os arquivos existentes podem lhe interessar para outras finalidades.

10.2.9 Limpando as regras de um chain - F

Para limpar todas as regras de um chain, use a seguinte sintaxe:

```
iptables [-t tabela] [-F chain]
```

Onde:

tabela

Tabela que contém o chain que desejamos zerar.

chain

Chain que desejamos limpar. Caso um chain não seja especificado, todos os chains da tabela serão limpos.

```
iptables -t filter -F INPUT
iptables -t filter -F
```

10.2.10 Apagando um chain criado pelo usuário - X

Para apagarmos um chain criado pelo usuário, usamos a seguinte sintaxe:

```
iptables [-t tabela] [-X chain]
```

Onde:

tabela

Nome da tabela que contém o chain que desejamos excluir.

chain

Nome do chain que desejamos apagar. Caso não seja especificado, todos os chains definidos pelo usuário na tabela especificada serão excluídos.

OBS: - Chains embutidos nas tabelas não podem ser apagados pelo usuário. Veja os nomes destes chains em O que são tabelas?, Seção 10.1.14.

```
iptables -t filter -X internet
iptables -X
```

10.2.11 Zerando contador de bytes dos chains - Z

Este comando zera o campo *pkts* e *bytes* de uma regra do `iptables`. Estes campos podem ser visualizados com o comando `iptables -L -v`. A seguinte sintaxe é usada:

```
iptables [-t tabela] [-Z chain] [-L]
```

Onde:

tabela

Nome da tabela que contém o chain que queremos zerar os contadores de bytes e pacotes.

chain

Chain que deve ter os contadores zerados. Caso não seja especificado, todos os chains da tabela terão os contadores zerados. Note que as opções *-Z* e *-L* podem ser usadas juntas, assim o chain será listado e imediatamente zerado. Isto evita a passagem de pacotes durante a listagem de um chain.

```
iptables -t filter -Z INPUT
```

10.2.12 Especificando a política padrão de um chain - P

A política padrão determina o que acontecerá com um pacote quando ele chegar ao final das regras contidas em um chain. A política padrão do `iptables` é "ACCEPT" mas isto pode ser alterado com o comando:

```
iptables [-t tabela] [-P chain] [ACCEPT/DROP]
```

Onde:

tabela

Tabela que contém o chain que desejamos modificar a política padrão.

chain

Define o chain que terá a política modificada. O chain deve ser especificado.

ACCEPT/DROP

ACCEPT aceita os pacotes caso nenhuma regra do chain conferir (usado em regras permissivas). DROP rejeita os pacotes caso nenhuma regra do chain conferir (usado em regras restritivas).

A política padrão de um chain é mostrada com o comando `iptables -L`:

```
# iptables -L INPUT

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             localhost
```

No exemplo acima, a política padrão de INPUT é ACCEPT (policy ACCEPT), o que significa que qualquer pacote que não seja rejeitado pela regra do chain, será aceito. Para alterar a política padrão deste chain usamos o comando:

```
iptables -t filter -P INPUT DROP
```

NOTA: As políticas de acesso PERMISSIVASS (ACCEPT) normalmente são usadas em conjunto com regras restritivas no chain correspondentes (tudo é bloqueado e o que sobrar é liberado) e políticas RESTRITIVAS (DROP) são usadas em conjunto com regras permissivas no chain correspondente (tudo é liberado e o que sobrar é bloqueado pela política padrão).

10.3 Outras opções do iptables

10.3.1 Especificando um endereço de origem/destino

As opções `-s` (ou --src/--source)e `-d` (ou --dst/--destination) servem para especificar endereços de *origem* e *destino* respectivamente. É permitido usar um endereço IP completo (como 192.168.1.1), um hostname (debian), um endereço fqdn (www.debian.org) ou um par *rede/máscara* (como 200.200.200.0/255.255.255.0 ou 200.200.200.0/24).

Caso um endereço/máscara não sejam especificados, é assumido *0/0* como padrão (todos as máquinas de todas as redes). A interpretação dos endereços de origem/destino dependem do chain que está sendo especificado (como INPUT e OUTPUT por exemplo).

OBS: Caso seja especificado um endereço fqdn e este resolver mais de um endereço IP, serão criadas várias regras, cada uma se aplicando a este endereço IP específico. É recomendável sempre que possível a especificação de endereços IP's nas regras, pois além de serem muito rápidos (pois não precisar de resolução DNS) são mais seguros para evitar que nosso firewall seja enganado por um ataque de IP spoofing.

```
# Bloqueia o tráfego vindo da rede 200.200.200.*:
 iptables -A INPUT -s 200.200.200.0/24 -j DROP

# Bloqueia conexões com o destino 10.1.2.3:
iptables -A OUTPUT -d 10.1.2.3 -j DROP

# Bloqueia o tráfego da máquina www.dominio.teste.org a rede 210.21.1.3
# nossa máquina possui o endereço 210.21.1.3
iptables -A INPUT -s www.dominio.teste.org -d 210.21.1.3 -j DROP
```

10.3.2 Especificando a interface de origem/destino

As opções *-i* (ou --in-interface) e *-o* (ou --out-interface) especificam as interfaces de origem/destino de pacotes. Nem todos as chains aceitam as interfaces de origem/destino simultaneamente, a interface de entrada (-i) nunca poderá ser especificada em um chain OUTPUT e a interface de saída (-o) nunca poderá ser especificada em um chain INPUT. Abaixo uma rápida referência:

TABELA	CHAIN	INTERFACE	
		ENTRADA (-i)	SAÍDA (-o)
filter	INPUT	SIM	NÃO
	OUTPUT	NÃO	SIM
	FORWARD	SIM	SIM
nat	PREROUTING	SIM	NÃO
	OUTPUT	NÃO	SIM
	POSTROUTING	NÃO	SIM
mangle	PREROUTING	SIM	NÃO
	OUTPUT	NÃO	SIM

O caminho do pacote na interface será determinado pelo tipo da interface e pela posição dos chains nas etapas de seu roteamento. O chain OUTPUT da tabela filter somente poderá conter a interface de saída (veja a tabela acima). O chain FORWARD da tabela filter é o único que aceita a especificação de ambas as interfaces, este é um ótimo chain para controlar o tráfego que passa entre interfaces do firewall.

Por exemplo para bloquear o acesso do tráfego de qualquer máquina com o endereço 200.123.123.10 vinda da interface ppp0 (uma placa de fax-modem):

```
iptables -A INPUT -s 200.123.123.10 -i ppp0 -j DROP
```

A mesma regra pode ser especificada como

```
iptables -A INPUT -s 200.123.123.10 -i ppp+ -j DROP
```

O sinal de "+" funciona como um coringa, assim a regra terá efeito em qualquer interface de ppp0 a ppp9. As interfaces ativas no momento podem ser listadas com o comando `ifconfig`, mas é permitido especificar uma regra que faz referência a uma interface que ainda não existe, isto é interessante para conexões intermitentes como o PPP. Para bloquear qualquer tráfego local para a Internet:

```
iptables -A OUTPUT -o ppp+ -j DROP
```

Para bloquear a passagem de tráfego da interface ppp0 para a interface eth1 (de uma de nossas redes internas):

```
iptables -A FORWARD -i ppp0 -o eth1 -j DROP
```

10.3.3 Especificando um protocolo

A opção `-p` (ou --protocol) é usada para especificar protocolos no `iptables`. Podem ser especificados os protocolos *tcp*, *udp* e *icmp*. Por exemplo, para rejeitar todos os pacotes UDP vindos de

200.200.200.200:

```
iptables -A INPUT -s 200.200.200.200 -p udp -j DROP
```

OBS1: Tanto faz especificar os nomes de protocolos em maiúsculas ou minúsculas.

10.3.3.1 Especificando portas de origem/destino

As portas de origem/destino devem ser especificadas após o protocolo e podem ser precedidas por uma das seguintes opções:

- `--source-port ou --sport` - Especifica uma porta ou faixa de portas de origem.
- `--destination-port ou --dport` - Especifica uma porta ou faixa de portas de destino.

Uma faixa de portas pode ser especificada através de `PortaOrigem:PortaDestino`:

```
# Bloqueia qualquer pacote indo para 200.200.200.200 na faixa de
# portas 0 a 1023
iptables -A OUTPUT -d 200.200.200.200 -p tcp --dport :1023 -j DROP
```

Caso a *PortaOrigem* de uma faixa de portas não seja especificada, 0 é assumida como padrão, caso a *Porta Destino* não seja especificada, 65535 é assumida como padrão. Caso precise especificar diversas regras que envolvam o tratamento de portas diferentes, recomendo da uma olhada em Especificando múltiplas portas de origem/destino, Seção 10.6.6, antes de criar um grande número de regras.

10.3.3.2 Especificando mensagens do protocolo ICMP

O protocolo ICMP não possui portas, mas é possível fazer um controle maior sobre o tráfego ICMP que entra/sai da rede através da especificação dos tipos de mensagens ICMP. Os tipos de mensagens devem ser especificados com a opção "--icmp-type *CódigoICMP*" logo após a especificação do protocolo icmp:

```
iptables -A INPUT -s 200.123.123.10 -p icmp --icmp-type time-exceeded -i ppp+ -j DROP
```

A regra acima rejeitará mensagens ICMP do tipo "time-exceeded" (tempo de requisição excedido) que venham do endereço 200.123.123.10 através da interface *ppp+*.

Alguns tipos de mensagens ICMP são classificados por categoria (como o próprio "time-exceeded"), caso a categoria "time-exceeded" seja especificada, todas as mensagens daquela categoria (como "ttl-zero-during-transit", "ttl-zero-during-reassembly") conferirão na regra especificada.Os tipos de mensagens ICMP podem ser obtidos com o comando `iptables -p icmp -h`:

```
echo-reply (pong)
destination-unreachable
   network-unreachable
   host-unreachable
   protocol-unreachable
   port-unreachable
   fragmentation-needed
   source-route-failed
   network-unknown
   host-unknown
   network-prohibited
   host-prohibited
   TOS-network-unreachable
   TOS-host-unreachable
   communication-prohibited
   host-precedence-violation
   precedence-cutoff
source-quench
redirect
   network-redirect
   host-redirect
   TOS-network-redirect
   TOS-host-redirect
echo-request (ping)
router-advertisement
router-solicitation
time-exceeded (ttl-exceeded)
   ttl-zero-during-transit
   ttl-zero-during-reassembly
parameter-problem
   ip-header-bad
   required-option-missing
timestamp-request
timestamp-reply
address-mask-request
address-mask-reply
```

OBS1: Não bloqueie mensagens do tipo "host-unreachable" e "source-quench", pois terá sérios problemas no controle de suas conexões. A primeira diz que o destino está inalcançavel e a segunda que o host está sobrecarregado, assim os pacotes devem ser enviados mais lentamente.

10.3.3.3 Especificando pacotes syn

Pacotes syn são usados para iniciarem uma conexão, o uso da opção --syn serve para especificar estes tipos de pacotes. Desta maneira é possível bloquear somente os pacotes que iniciam uma conexão, sem afetar os pacotes restantes. Para que uma conexão ocorra é necessário que a máquina obtenha a resposta a pacotes syn enviados, caso ele seja bloqueado a resposta nunca será retornada e a conexão não será estabelecida.

```
iptables -A INPUT -p tcp --syn --dport 23 -i ppp+ -j DROP
```

A regra acima bloqueia (-j DROP) qualquer tentativa de conexão (--syn) vindas da interface ppp+ ao telnet (--dport 23) da máquina local, conexões já efetuadas ão são afetadas por esta regra. A opção --syn somente pode ser especificada para o protocolo tcp.

ATENÇÃO: - A situação de passagem de pacotes durante deve ser levada em conta durante a inicialização do firewall, bloqueando a passagem de pacotes durante o processo de configuração, criando regras que bloqueiam a passagem de pacotes (exceto para a interface loopback) até que a configuração do firewall esteja completa, pode ser uma solução eficiente.

Outra alternativa segura é configurar as regras de firewall antes das interfaces de rede se tornarem ativas (usando a opção "pre-up comando_firewall" no arquivo de configuração `/etc/network/interfaces` em sistemas `Debian`.

10.3.4 Especificando fragmentos

A opção "-f" (ou --fragment) permite especificar regras que confiram com fragmentos. Fragmentos são simplesmente um pacote maior dividido em pedaços para poder ser transmitido via rede TCP/IP para remontagem do pacote pela máquina de destino.

Somente o primeiro fragmento possui detalhes de cabeçalho para ser processado, os segundos e seguintes somente possuem alguns cabeçalhos necessários para dar continuidade ao processo de remontagem do pacote no destino.

Uma regra como

```
iptables -A INPUT -s 200.200.200.1 -f -j DROP
```

derrubará os fragmentos de 200.200.200.1 derrubará o segundo pacote e pacotes seguintes enviados por 200.200.200.1 até nós.

OBS1: Note que se o cabeçalho do pacote não tiver detalhes suficientes para checagem de regras no `iptables`, a regra simplesmente não ira conferir.

OBS2: Não é preciso especificar a opção "-f" para conexões NAT, pois os pacotes são remontados antes de entrarem no código de filtragem.

OBS3: A opção "-f" também pode ser usada para evitar o flood por fragmentos (bomba de fragmentos) que, dependendo da intensidade, podem até travar a máquina.

10.3.5 Especificando uma exceção

Muitos parâmetros (como o endereço de origem/destino, protocolo, porta, mensagens ICMP, fragmentos, etc) podem ser precedidos pelo sinal "!" que significa exceção. Por exemplo:

```
iptables -t filter -A INPUT ! -s 200.200.200.10 -j DROP
```

Diz para rejeitar todos os pacotes EXCETO os que vem do endereço `200.200.200.10`.

```
iptables -A INPUT -p tcp ! --syn -s 200.200.200.10 ! -i eth0 -j DROP
```

Diz para bloquear todos os pacotes EXCETO os que iniciam conexões (! --syn), EXCETO para pacotes vindos pela interface eth0 (! -i eth0).

```
iptables -A INPUT -s 200.200.200.10 ! -p tcp -j DROP
```

Bloqueia todos os pacotes vindos de 200.200.200.10, EXCETO os do protocolo tcp.

10.3.6 Especificando um alvo

O alvo (-j) é o destino que um pacote terá quando conferir com as condições de uma regra, um alvo pode dizer para bloquear a passagem do pacote (-j DROP), aceitar a passagem do pacote (-j ACCEPT), registrar o pacote no sistema de log (-j LOG), rejeitar o pacote (-j REJECT), redirecionar um pacote -j REDIRECT, retornar ao chain anterior sem completar o processamento no chain atual (-j RETURN), passar para processamento de programas externos (-j QUEUE), fazer source nat (-j SNAT), destination nat (-j DNAT), etc. Podem existir mais alvos, pois o `iptables` é modularizável, e módulos que acrescentam mais funções podem ser carregados em adição aos já existentes no kernel.

Nos exemplos anteriores vimos o uso de diversos alvos como o DROP e o ACCEPT. Apenas farei uma breve referência sobre os alvos mais usados em operações comuns dos chains. Os alvos REDIRECT, SNAT e DNAT serão explicados em uma seção seguinte:

ACCEPT

O pacote é ACEITO e o processamento das regras daquele chains é concluído. Pode ser usado como alvo em todos os chains de todas as tabelas do `iptables` e também pode ser especificado na política padrão das regras do firewall (veja Especificando a política padrão de um chain - P, Seção 10.2.12).

DROP

Rejeita o pacote e o processamento das regras daquele chain é concluído. Pode ser usado como alvo em todos os chains de todas as tabelas do `iptables` e também pode ser especificado na política padrão das regras do firewall (veja Especificando a política padrão de um chain - P, Seção 10.2.12).

REJECT

Este é um módulo opcional que faz a mesma função do alvo *DROP* com a diferença de que uma mensagem ICMP do tipo "icmp-port-unreachable" (TCP/UDP) ou "host-unreachable" (ICMP) é retornada para a máquina de origem. Pode ser usado como alvo somente nos chains da tabela (não como política padrão).

LOG

Este módulo envia uma mensagem ao `syslog` caso a regra confira, o processamento continua normalmente para a próxima regra (o pacote não é nem considerado ACEITO ou REJEITADO).

RETURN

Retorna o processamento do chain anterior sem processar o resto do chain atual.

QUEUE

Passa o processamento para um programa a nível de usuário.

10.3.6.1 Alvo REJECT

Para ser usado, o módulo `ipt_REJECT` deve ser compilado no kernel ou como módulo. Este alvo rejeita o pacote (como o *DROP*) e envia uma mensagem ICMP do tipo "icmp-port-unreachable" como padrão para a máquina de origem.

É um alvo interessante para bloqueio de portas TCP, pois em alguns casos da a impressão que a máquina não dispõe de um sistema de firewall (o alvo DROP causa uma parada de muito tempo em alguns portscanners e tentativas de conexão de serviços, revelando imediatamente o uso de um sistema de firewall pela máquina). O alvo REJECT vem dos tempos do ipchains e somente pode ser usado na tabela *filter*. Quando um pacote confere, ele é rejeitado com a mensagem ICMP do tipo "port unreachable", é possível especificar outro tipo de mensagem ICMP com a opção *--reject-with tipo_icmp*.

OBS: REJECT pode ser usado somente como alvo na tabela filter e não é possível especifica-lo como política padrão do chain filter (como acontecia no `ipchains`. Uma forma alternativa é inserir como última regra uma que pegue todos os pacotes restantes daquele chain e tenha como alvo REJECT (como `iptables -A INPUT -j REJECT`), desta forma ele nunca atingirá a política padrão do chain.

```
# Rejeita pacotes vindos de 200.200.200.1 pela interface ppp0:
iptables -A INPUT -s 200.200.200.1 -i ppp+ -j REJECT
```

10.3.6.2 Especificando LOG como alvo

Este alvo é usado para registrar a passagem de pacotes no `syslog` do sistema. É um alvo muito interessante para ser usado para regras que bloqueiam determinados tráfegos no sistema (para que o administrador tome conhecimento sobre tais tentativas), para regras de fim de chain (quando você tem um grande conjunto de regras em um firewall restritivo e não sabe onde suas regras estão sendo bloqueadas), para satisfazer sua curiosidade, etc.

```
# Para registrar o bloqueio de pacotes vindos de 200.200.200.1 pela interface ppp0
iptables -A INPUT -s 200.200.200.1 -i ppp+ -j LOG
# Para efetuar o bloqueio
iptables -A INPUT -s 200.200.200.1 -i ppp+ -j REJECT
```

Note que no exemplo anterior a regra que registra o pacote (-j LOG) deve aparecer antes da regra que REJEITA (-j REJECT), caso contrário a regra de LOG nunca funcionará. A regra que REJEITA poderia também ser trocada por uma regra que ACEITA, caso queira registrar um pacote que deve ser aceito (se a política padrão do seu firewall for restritiva (-P DROP). A única coisa que muda nas regras de log é o alvo da regra, isto facilita a implementação de grandes conjuntos de regras de firewall.

A regra acima mostrará a seguinte saída no `syslog` do sistema:

```
Aug 25 10:08:01 debian kernel: IN=ppp0 OUT= MAC=10:20:30:40:50:60:70:80:90:00:00:00:08:00 SRC=200.200.200.1 DST=200.210.10.10 LEN=61 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=1031 DPT=53 LEN=41
```

Os campos possuem o seguinte significado:

Aug 25 10:08:01

Mês, dia e hora do registro do pacote.

debian

Nome do computador que registrou o pacote.

kernel:

Daemon que registrou a mensagem, no caso o `iptables` faz parte do próprio kernel.

IN=ppp0

Especifica a interface de entrada (de onde o pacote veio).

OUT=

Especifica a interface de saída (para onde o pacote foi).

MAC=10:20:30:40:50:60:70:80:90:00:00:00:08:00

Endereço mac da interface de rede (pode ser obtido com `arp interface`).

SRC=200.200.200.1

Endereço de origem do pacote.

DST=200.210.10.10

Endereço de destino do pacote.

SEQ=234234343

Número de seqüência da recepção. É ativado com a opção *--log-tcp-sequence*.

LEN=61

Tamanho em bytes do pacote IP.

TOS=0x00

Prioridade do cabeçalho TOS (Tipo). Veja a seção Especificando o tipo de serviço, Seção 10.5.1 para mais detalhes.

PREC=0x00

Prioridade do cabeçalho TOS (Precedência). Veja a seção Especificando o tipo de serviço, Seção 10.5.1 para mais detalhes.

TTL=64

Tempo de vida do pacote. No exemplo, 64 roteadores (hops).

ID=0

Identificação única destes datagrama. Esta identificação também é usada pelos fragmentos seguintes deste pacote.

DF

Opção "Don't fragment" (não fragmentar) do pacote. Usada quando o pacote é pequeno o bastante para não precisar ser fragmentado.

MF

Opção "More Fragments" (mais fragmentos) estão para ser recebidos.

FRAG=100

Tamanho do fragmento especificado em pacotes de 8 bits. No exemplo acima, o pacote tem o tamanho de 800 bytes (100*8).

PROTO=UDP

Nome do protocolo. Pode ser TCP, UDP ou ICMP

SPT=1031

Porta de origem da requisição.

DPT=53

Porta de destino da requisição.

LEN=41

Tamanho do pacote.

O log acima mostra uma consulta DNS (porta destino 53) para nossa máquina (INPUT) de 200.200.200.1 para 200.210.10.10.

O problema é que em um grande número de regras será difícil saber qual regra conferiu (pois teríamos que analisar o endereço/porta origem/destino) e o destino do pacote (se ele foi ACEITO ou BLOQUEADO) pois você pode ter regras para ambas as situações. Por este motivo existem algumas opções úteis que podemos usar com o alvo LOG:

--log-prefix "descrição"

Permite especificar uma descrição para a regra do firewall de até 29 caracteres. Caso tiver espaços, devem ser usadas "aspas".

--log-level nível

Especifica o nível da mensagem no `syslog`.

--log-tcp-options

Registra campos do cabeçalho TCP nos logs do sistema.

--log-ip-options

Registra campos do cabeçalho IP nos logs do sistema

--log-tcp-sequence

Registra os números de seqüencia TCP. Evite ao máximo o uso desta opção, pois a seqüencia de números TCP pode ser a chave para um seqüestro de seção ou IP spoofing em seu sistema caso algum usuário tenha acesso a estes logs. Caso utilize tcp/ip em servidores públicos, o uso desta opção ajudará a entender bem os ataques DoS causados por syn-flood e porque ativar os SynCookies (veja Proteção contra syn flood, Seção 10.6.4).

OBS1: Lembre-se que estas opções são referentes ao alvo LOG e devem ser usadas após este, caso contrário você terá um pouco de trabalho para analisar e consertar erros em suas regras do firewall.

OBS2: Caso esteja usando o firewall em um servidor público, recomendo associar um limite a regra de log, pois um ataque poderia causar um DoS enchendo sua partição. Leia mais sobre isso em Limitando o número de vezes que a regra confere, Seção 10.6.2.

```
# Complementando o exemplo anterior:
# Para registrar o bloqueio de pacotes vindos de 200.200.200.1 pela interface ppp0
iptables -A INPUT -s 200.200.200.1 -i ppp+ -j LOG --log-prefix "FIREWALL: Derrubado "
# Para efetuar o bloqueio
iptables -A INPUT -s 200.200.200.1 -i ppp+ -j REJECT
```

Retornará a seguinte mensagem no syslog:

```
Aug 25 10:08:01 debian kernel: FIREWALL: Derrubado IN=ppp0 OUT= MAC=10:20:30:40:50:60:70:80:90:00:00:00:08:00 SRC=200.200.200.1 DST=200.210.10.10 LEN=61 TOS=0x00 PREC=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=1031 DPT=53 LEN=41
```

Agora você sabe o que aconteceu com o pacote (Rejeitado). A padronização de mensagens de firewall é também importante para a criação de scripts de análise que poderão fazer a análise dos logs do seu firewall (para criação de estatísticas que podem servir como base para a criação de novas regras de firewall ou eliminação de outras).

OBS: Se você sente falta da função "-l" do `ipchains` que combina o alvo e log na mesma regra você pode criar um alvo como o seguinte:

```
iptables -N log-drop
iptables -A log-drop -j LOG
iptables -A log-drop -j DROP
```

E usar "log-drop" como alvo em suas regras. Mesmo assim esta solução é "limitada" em relação a "-l" do `ipchains` porque o `iptables` não inclui detalhes de qual chain bloqueou o pacote/qual pacote foi bloqueado, assim é necessário a especificação da opção *--log-prefix* para as mensagens se tornarem mais compreensíveis. Esta limitação pode ser contornada utilizando um firewall feito em linguagem shell script, desta forma você terá um controle maior sobre o seu programa usando funções e integração com outros utilitários.

10.3.6.3 Especificando RETURN como alvo

O alvo RETURN diz ao `iptables` interromper o processamento no chain atual e retornar o processamento ao chain anterior. Ele é útil quando criamos um chain que faz um determinado tratamento de pacotes, por exemplo bloquear conexões vindas da internet para portas baixas, exceto para um endereço IP específico. Como segue:

```
1-) iptables -t filter -A INPUT -i ppp0 -j internet
2-) iptables -t filter -j ACCEPT
3-) iptables -t filter -N internet
4-) iptables -t filter -A internet -s www.debian.org -p tcp --dport 80 -j RETURN
5-) iptables -t filter -A internet -p tcp --dport 21 -j DROP
6-) iptables -t filter -A internet -p tcp --dport 23 -j DROP
7-) iptables -t filter -A internet -p tcp --dport 25 -j DROP
8-) iptables -t filter -A internet -p tcp --dport 80 -j DROP
```

Quando um pacote com o endereço `www.debian.org` tentando acessar a porta www (80) de nossa máquina através da internet (via interface ppp0), o chain número 1 confere, então o processamento continua no chain número 4, o chain número 4 confere então o processamento volta para a regra número 2, que diz para aceitar o pacote.

Agora se um pacote vem com o endereço `www.dominio.com.br` tentando acessar a porta www *80) de nossa máquina através da internet (via interface ppp0), o chain número 1 confere, então o processamento continua no chain número 4, que não confere. O mesmo acontece com os chains 5, 6 e 7. O chain número 8 confere, então o acesso é bloqueado.

Como pode notou, o alvo RETURN facilita bastante a construção das regras do seu firewall, caso existam máquinas/redes que sejam exceções as suas regras. Se ela não existisse, seria necessário especificar diversas opções -s, -d, etc para poder garantir o acesso livre a determinadas máquinas.

10.3.7 Salvando e Restaurando regras

As regras que você está trabalhosamente criando e testando manualmente enquanto manipula o `iptables` podem ser salvas de 2 formas; uma delas é escrevendo um shell script que tenha todos os comandos, um por linha. Isto é recomendado quando tem um firewall grande e que exige uma boa padronização de regras, bem como sua leitura, comentários. O script shell também permite o uso de funções presente no interpretador de comando, portanto se você é uma pessoa que gosta de interagir com as funções do shell e deixar as coisas mais flexíveis, prefira esta opção.

A outra forma é usando as ferramentas `iptables-save` e `iptables-restore` baseada na idéia do `ipchains-save` e `ipchains-restore`. O `iptables-save` deve ser usado sempre que modificar regras no firewall iptables da seguinte forma:

```
iptables-save >/dir/iptables-regras
```

Uma das vantagens do uso do `iptables-save` é ele também salvar os contadores de chains, ou seja, a quantidade de pacotes que conferiram com a regra. Isto também pode ser feito com algumas regras adicionais em seu shell script, caso tenha interesse nesses contadores para estatísticas ou outros tipos de relatórios.

Para restaurar as regras salvas, utilize o comando:

```
iptables-restore </dir/iptables-regras
```

10.4 A tabela nat (Network Address Translation) - fazendo nat

A tabela *nat* serve para controlar a tradução dos endereços que atravessam o código de roteamento da máquina `Linux`. Existem 3 chains na tabela *nat: PREROUTING*, *OUTPUT* e *POSTROUTING* (veja O que são tabelas?, Seção 10.1.14 para maiores detalhes).

A tradução de endereços tem inúmeras utilidades, uma delas é o Masquerading, onde máquinas de uma rede interna podem acessar a Internet através de uma máquina Linux, redirecionamento de porta, proxy transparente, etc. Esta seção abordará os tipos de NAT, exemplos de como criar rapidamente uma conexão IP masquerading e entender como a tradução de endereços funciona no iptables.

Se sua intenção é ligar sua rede a Internet existem duas opções:

- Você possui uma conexão que lhe oferece um endereço IP dinâmico (a cada conexão é dado um endereço IP - como uma conexão PPP) então o IP masquerading é o que precisa (veja Fazendo IP masquerading (para os apressados), Seção 10.4.2 ou Fazendo IP Masquerading, Seção 10.4.3.1).
- Você tem uma conexão que lhe oferece um endereço IP permanente (ADSL, por exemplo) então o SNAT é o que precisa (veja Fazendo SNAT, Seção 10.4.3).

10.4.1 Criando um novo chain na tabela NAT

O procedimento para criação de um novo chain nesta tabela é o mesmo descrito em Criando um novo chain - N, Seção 10.2.6 será necessário somente especificar a tabela nat (-t nat) para que o novo chain não seja criado na tabela padrão (-t filter).

```
iptables -t nat -N intra-inter
```

Que criará o chain chamado *intra-inter* na tabela *nat*. Para inserir regras neste chain será necessário especificar a opção "-t nat".

10.4.2 Fazendo IP masquerading (para os apressados)

Você precisará de um kernel com suporte ao iptables (veja Habilitando o suporte ao iptables no kernel, Seção 10.1.15 e ip_forward e então digitar os dois comandos abaixo para habilitar o masquerading para todas as máquinas da rede 192.168.1.*:

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j MASQUERADE
echo "1" >/proc/sys/net/ipv4/ip_forward
```

A configuração do servidor Linux está completa, agora os clientes da rede precisarão ser configurados para usar o endereço IP do servidor Linux como gateway. É recomendável instalar um servidor proxy e DNS na máquina Linux para acelerar o desempenho das requisições/resolução de nomes das máquinas em rede. A utilização de bits TOS também pode trazer um grande aumento de velocidade para os diferentes serviços da rede (veja Especificando o tipo de serviço, Seção 10.5.1).

10.4.3 Fazendo SNAT

SNAT (source nat - nat no endereço de origem) consiste em modificar o endereço de origem das máquinas clientes antes dos pacotes serem enviados. A máquina roteadora é inteligente o bastante para lembrar dos pacotes modificados e reescrever os endereços assim que obter a resposta da máquina de destino, direcionando os pacotes ao destino correto. Toda operação de SNAT é feita no chain *POSTROUTING*.

É permitido especificar endereços de origem/destino, protocolos, portas de origem/destino, interface de entrada/saída (dependendo do chain), alvos, etc. É desnecessário especificar fragmentos na tabela nat, pois eles serão remontados antes de entrar no código de roteamento.

O SNAT é a solução quando você tem acesso a internet através de um único IP e deseja fazer que sua rede tenha acesso a Internet através da máquina Linux. Nenhuma máquina da Internet poderá ter acesso direto as máquinas de sua rede interna via SNAT.

OBS: A observação acima não leva em conta o controle de acesso externo configurado na máquina que estiver configurando o iptables, uma configuração mau realizada pode expor sua máquina a acessos externos indesejados e comprometer sua rede interna caso alguém consiga acesso direto ao servidor.

É necessário especificar SNAT como alvo (-j SNAT) quando desejar que as máquinas de sua rede interna tenha acesso a Internet através do IP fixo da máquina Linux (para conexões intermitentes como PPP, veja Fazendo IP Masquerading, Seção 10.4.3.1). O parâmetro --to IP:portas deve ser usado após o alvo SNAT. Ele serve para especificar um endereço IP, faixa de endereços e opcionalmente uma porta ou faixa de portas que será substituída. Toda a operação de SNAT é realizada através do chain POSTROUTING:

```
# Modifica o endereço IP dos pacotes vindos da máquina 192.168.1.2 da rede interna
# que tem como destino a interface eth1 para 200.200.217.40 (que é o nosso endereço
# IP da interface ligada a Internet).
iptables -t nat -A POSTROUTING -s 192.168.1.2 -o eth1 -j SNAT --to 200.200.217.40
```

Os pacotes indo para a Internet (nossa conexão é feita via eth1, nossa interface externa) vindo do endereço 192.168.1.2, são substituídos por 200.241.200.40 e enviados para fora. Quando a resposta a requisição é retornada, a máquina com iptables recebe os pacotes e faz a operação inversa, modificando o endereço 200.241.200.40 novamente para 192.168.1.2 e enviando a resposta a máquina de nossa rede interna. Após definir suas regras de NAT, execute o comando echo "1" >/proc/sys/net/ipv4/ip_forward para habilitar o suporte a redirecionamento de pacotes no kernel.

Também é possível especificar faixas de endereços e portas que serão substituídas:

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j SNAT --to 200.200.217.40-200.200.217.50
```

Modifica o endereço IP de origem de todas as máquinas da rede 192.168.1.0/24 que tem o destino a interface eth0 para 200.241.200.40 a 200.241.200.50. O endereço IP selecionado é escolhido de acordo com o último IP alocado.

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j SNAT --to 200.200.217.40-200.200.217.50:1-1023
```

Idêntico ao anterior, mas faz somente substituições na faixa de portas de origem de 1 a 1023.

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j SNAT --to 200.200.217.40-200.200.217.50 --to 200.200.217.70-200.200.217.73
```

Faz o mapeamento para a faixa de portas 200.200.217.40 a 200.200.217.50 e de 200.200.217.70 a 200.200.217.73.

OBS1: Se por algum motivo não for possível mapear uma conexão NAT, ela será derrubada.

OBS2: Tenha certeza que as respostas podem chegar até a máquina que fez o NAT. Se estiver fazendo SNAT em um endereço livre em sua rede (como 200.200.217.73).

OBS3: Como notou acima, o SNAT é usado quando temos uma conexão externa com um ou mais IP's fixos. O Masquerading é uma forma especial de SNAT usada para funcionar em conexões que recebem endereços IP aleatórios (PPP).

OBS4: Não se esqueça de habilitar o redirecionamento de pacotes após fazer suas regra de NAT com o comando: echo "1" >/proc/sys/net/ipv4/ip_forward, caso contrário o redirecionamento de pacotes não funcionará.

10.4.3.1 Fazendo IP Masquerading

O IP Masquerading é um tipo especial de SNAT usado para conectar a sua rede interna a internet quando você recebe um IP dinâmico de seu provedor (como em conexões ppp). Todas as operações de IP Masquerading são realizadas no chain *POSTROUTING*. Se você tem um IP fixo, deve ler Fazendo SNAT, Seção 10.4.3.

Para fazer IP Masquerading de uma máquina com o IP 192.168.1.2 para ter acesso a Internet, use o comando:

```
iptables -t nat -A POSTROUTING -s 192.168.1.2/32 -o ppp0 -j MASQUERADE
```

A diferença é que o alvo é *-j MASQUERADE*. O comando acima faz IP Masquerading de todo o tráfego de 192.168.1.2 indo para a interface ppp0: O endereço IP dos pacotes vindos de 192.168.1.2 são substituídos pelo IP oferecido pelo seu provedor de acesso no momento da conexão, quando a resposta é retornada a operação inversa é realizada para garantir que a resposta chegue ao destino. Nenhuma máquina da internet poderá ter acesso direto a sua máquina conectava via Masquerading.

Para fazer o IP Masquerading de todas as máquinas da rede 192.168.1.*:

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp0 -j MASQUERADE
```

Após definir a regra para fazer Masquerading (SNAT), execute o comando echo "1" >/proc/sys/net/ipv4/ip_forward para habilitar o suporte a redirecionamento de pacotes no kernel.

10.4.4 Fazendo DNAT

DNAT (Destination nat - nat no endereço de destino) consiste em modificar o endereço de destino das máquinas clientes. O destination nat é muito usado para fazer redirecionamento de pacotes, proxyes transparentes e balanceamento de carga.

Toda operação de DNAT é feita no chain *PREROUTING*. As demais opções e observações do SNAT são também válidas para DNAT (com exceção que somente é permitido especificar a interface de origem no chain PREROUTING).

```
# Modifica o endereço IP destino dos pacotes de 200.200.217.40 vindo da interface eth0
# para 192.168.1.2.
iptables -t nat -A PREROUTING -s 200.200.217.40 -i eth0 -j DNAT --to 192.168.1.2
```

Também é possível especificar faixas de endereços e portas que serão substituídas no DNAT:

```
iptables -t nat -A PREROUTING -i eth0 -s 192.168.1.0/24 -j DNAT --to 200.200.217.40-200.200.217.50
```

Modifica o endereço IP de destino do tráfego vindos da interface 192.168.1.0/24 para um IP de 200.241.200.40 a 200.241.200.50. Este é um excelente método para fazer o balanceamento de carga entre servidores. O endereço IP selecionado é escolhido de acordo com o último IP alocado.

```
iptables -t nat -A PREROUTING -i eth0 -s 192.168.1.0/24 -j DNAT --to 200.200.217.40-200.200.217.50:1024:5000
```

Idêntico ao anterior, mas faz somente substituições na faixa de portas de destino de 1024 a 5000. A operação acima é a mesma realizada pelo `ipmasqadm` dos kernels da série 2.2.

OBS1: Se por algum motivo não for possível mapear uma conexão NAT, ela será derrubada.

OBS2: Não se esqueça de conferir se o `ip_forward` está ajustado para `1`: `echo "1" >/proc/sys/net/ipv4/ip_forward`.

10.4.4.1 Redirecionamento de portas

O redirecionamento de portas permite a você repassar conexões com destino a uma porta para outra porta na mesma máquina. O alvo *REDIRECT* é usado para fazer esta operação, junto com o argumento *--to-port* especificando a porta que será redirecionada. Este é o método DNAT específico para se para fazer proxy transparente (para redirecionamento de endereços/portas, veja Fazendo DNAT, Seção 10.4.4). Todas as operações de redirecionamento de portas é realizada no chain *PREROUTING* e *OUTPUT* da tabela *nat*.

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 81
```

Redireciona as conexões indo para a porta 80 para a porta 81 (rodando `squid`) no firewall.

ATENÇÃO: O `squid` possui suporte a proxy transparente, e poderá atender as requisições acima da regra acima.

10.4.5 Monitorando conexões feitas na tabela nat

Use o comando `cat /proc/net/ip_conntrack` para listar todas as conexões atuais tratadas pelo módulo nat.

10.5 A tabela mangle

A tabela *mangle* serve para especificar ações especiais para o tratamento do tráfego que atravessa os chains. Nesta tabela existem cincos chains: *PREROUTING*, *POSTROUTING*, *INPUT*, *OUTPUT* e *FORWARD* (veja O que são tabelas?, Seção 10.1.14 para maiores detalhes).

Em geral, cada um deste chain é processado antes do chain correspondente na tabela *filter* e *nat* para definir opções especiais para o tráfego (por exemplo, o chain *PREROUTING* da tabela *mangle* é processado antes do *PREROUTING* da tabela *nat*). O chain *OUTPUT* da tablea *mangle* corresponde ao *OUTPUT* da tabela *nat*. Opções como o *Tipo de Serviço (TOS)* é especificado nesta tabela para classificar e aumentar consideravelmente a velocidade de tráfego considerados em tempo real. Mesmo após o tráfego ser estabelecido, os chains da tabela mangle continuam ativos para garantir que as opções especiais relacionadas com a conexão continuem fazendo efeito (veja os exemplos de Caminho percorrido pelos pacotes nas tabelas e chains, Seção 10.7).

10.5.1 Especificando o tipo de serviço

O tipo de serviço é um campo existente no cabeçalho de pacotes do protocolo ipv4 que tem a função especificar qual é a prioridade daquele pacote. A prioridade é definida usando o algoritmo FIFO do próprio kernel, sendo uma das alternativas de controle/priorização de tráfego das mais simples e rápidas.

Uma das vantagens da utilização do tipo de serviço é dar prioridade ao tráfego de pacotes interativos (como os do ICQ, IRC, servidores de chat), etc. Com o TOS especificado, mesmo que esteja fazendo um download consumindo toda a banda de sua interface de rede, o tráfego com prioridade interativa será enviado antes, aumentando a eficiência do uso de serviços em sua máquina.

Em testes realizados em minha conexão de 56K, o uso de regras TOS aumentou bastante o desempenho em tráfego interativo (em torno de 300%), durante o uso total da banda da interface ppp em grande consumo de banda.

Usamos o alvo TOS (-j TOS) para especificar a modificação do tipo de serviço nos pacotes que atravessam as regras do firewall, acompanhada do argumento *--set-tos TOS* que define a nova prioridade dos pacotes. Os valores aceitos são os seguintes:

Espera Mínima

> É especificado através de *Minimize-Delay*, *16* ou *0x10*

Máximo Processamento

> É especificado através de *Maximize-Throughput*, *8*, ou *0x08*.

Máxima Confiança

> É especificado através de *Maximize-Reliability*, *4* ou *0x04*.

Custo mínimo

> Especificado através de *Minimize-Cost*, *2* ou *0x02*.

Prioridade Normal

> Especificado através de *Normal-Service*, *0* ou *0x00*.

Os pacotes vem por padrão com o valor TOS ajustado como *prioridade normal* (bits tos ajustados para 0x00). O tipo *Mínima Espera* é o de maior prioridade, recomendado para tráfego interativo.

10.5.1.1 Especificando o TOS para tráfego de saída

Este é o mais usado, pois prioriza o tráfego que sai da máquina (com destino a Internet, por exemplo). Sua operação é realizada através do chain *OUTPUT* ou *POSTROUTING*.

Para priorizar todo o tráfego de IRC de nossa rede interna indo para a interface ppp0:

```
iptables -t mangle -A OUTPUT -o ppp0 -p tcp --dport 6666-6668 -j TOS --set-tos 16
```

O bit TOS é ajustado para *Espera mínima* e será enviado antes dos pacotes com prioridade normal para fora. Para priorizar a transmissão de dados ftp saindo da rede:

```
iptables -t mangle -A OUTPUT -o ppp0 -p tcp --dport 20 -j TOS --set-tos 8
```

Para priorizar o tráfego de ICQ da rede:

```
iptables -t mangle -A OUTPUT -o ppp0 -p tcp --dport 5190 -j TOS --set-tos 16
```

Existem muitas outras otimizações que podem ser feitas, só depende dos requerimentos e análise de cada serviço da rede pelo administrador.

OBS: - Os pacotes que atravessam o alvo TOS somente tem os bits tipo do serviço modificados, eles não serão de qualquer forma rejeitados.

10.5.1.2 Especificando o TOS para o tráfego de entrada

Este prioriza o tráfego que entra da máquina. Sua operação é realizada no chain *INPUT* ou *PREROUTING*. Não faz muito sentido o uso deste chain dentro de uma rede pequena/média, pois o tráfego que recebermos será priorizado pelo chain de saída de outras máquinas da internet/outras redes antes de chegar a nossa (desde que elas também estejam usando TOS).

Para priorizar o processamento do tráfego interativo vindo de servidores IRC para nossa rede:

```
iptables -t mangle -A PREROUTING -i eth0 -p tcp --sport 6666-6668 -j TOS --set-tos 0x10
```

Modifica o tipo de serviço para *mínima espera* de todo o tráfego enviado por servidores de IRC vindo da interface eth0.

OBS: - Os pacotes que atravessam o alvo TOS somente tem os bits tipo do serviço modificados, eles não serão de qualquer forma rejeitados. \

10.6 Outros módulos do iptables

Os módulos do iptables são especificados com a opção *-m módulo* ou *--match módulo* e permitem expandir a funcionalidade do firewall através de novas conferências e recursos de filtragem adicionais, como limitar a conferência de regras do firewall (um método útil de limitar ping floods, syn floods, etc).

10.6.1 Conferindo de acordo com o estado da conexão

Este módulo permite especificar regras de acordo com o estado da conexão do pacote, isto é feito através da interpretação da saída do módulo `ip_conntrack`. O parâmetro *--state OPÇÕES* deve acompanhar este módulo. As opções permitidas são as seguintes:

- `NEW` - Confere com pacotes que criam novas conexões
- `ESTABLISHED` - Confere com conexões já estabelecidas
- `RELATED` - Confere com pacotes relacionados indiretamente a uma conexão, como mensagens de erro icmp, etc.
- `INVALID` - Confere com pacotes que não puderam ser identificados por algum motivo. Como respostas de conexões desconhecidas.

Caso seja necessário especificar mais de uma opções estas devem ser separadas por vírgulas.

```
iptables -A INPUT -m state --state NEW -i ppp0 -j DROP
```

Bloqueia qualquer tentativa de nova conexão vindo da interface ppp0.

```
iptables -A INPUT -m state --state NEW,INVALID -i ppp0 -j LOG
```

Permite registrar novas conexões e pacotes inválidos vindos da interface ppp0.

10.6.2 Limitando o número de vezes que a regra confere

A opção *-m limit* permite especificar o número de vezes que uma regra conferirá quando todas as outras condições forem satisfeitas. O número padrão de conferência é de 3 por hora, a não ser que seja modificado através dos argumentos aceitos pelo *limit*:

- `--limit num/tempo` - Permite especificar a taxa de conferências do limit. O parâmetro *num* especifica um número e *tempo* pode ser
  - `s` - Segundo
  - `m` - Minuto
  - `h` - Hora
  - `d` - Dia

  Assim uma regra como `iptables -A INPUT -m limit --limit 5/m -j ACCEPT` permitirá que a regra acima confira apenas 5 vezes por minuto (*--limit 2/s*). Este limite pode ser facilmente adaptado para uma regra de log que confere constantemente não causar uma avalanche em seus logs. O valor padrão é 3/h.
- `--limit-burst num` - Especifica o número inicial máximo de pacotes que irão conferir, este número é aumentado por 1 a cada vez que o parâmetro *--limit* acima não for atingido. O valor padrão é 5.

10.6.3 Proteção contra ping da morte

A regra abaixo pode tomada como base para proteção contra ping flood:

```
iptables -t filter -A ping-chain -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT
iptables -t filter -A ping-chain -j DROP
```

A regra acima limita em 1 vez por segundo (--limit 1/s) a passagem de pings (echo requests) para a máquina `Linux`.

```
iptables -t filter -A ping-chain -i ppp0 -p icmp --icmp-type echo-reply -m limit --limit 1/s -j RETURN
iptables -t filter -A ping-chain -j DROP
```

Limita respostas a pings (echo reply) vindos da interface ppp0 (-i ppp0) a 1 por segundo.

ATENÇÃO: O exemplo acima é somente para a criação de suas próprias regras com limitações, caso um pacote não confira ele será bloqueado pela próxima regra. Se uma regra como esta for colocada no chain INPUT sem modificações, ela não terá o efeito desejado, podendo colocar em risco a sua instalação pela falsa impressão de segurança. Portanto, é recomendável sempre testar as modificações para ter certeza que elas tem efeito.

10.6.4 Proteção contra syn flood

A regra abaixo é uma boa proteção para os ataques syn floods:

```
iptables -t filter -A syn-chain -p tcp --syn -m limit --limit 2/s -j ACCEPT
iptables -t filter -A syn-chain -j DROP
```

Esta regra limita o atendimento de requisições de conexões a 2 por segundo. Outra forma de aumentar a segurança contra syn-floods é através do próprio kernel ativando a opção "TCP Synflood" na compilação e depois executando: `echo "1" >/proc/sys/net/ipv4/tcp_synflood`. No entanto, utilize estas opções com cautela em servidores que possuem um grande número de acessos para não ter problemas que afetem seu clientes.

ATENÇÃO: Os exemplos acima devem são somente exemplos para criação de suas próprias regras com limitações, caso um pacote não confira com a regra ele será bloqueado pela próxima regra. Se uma regra como esta for colocada no chain INPUT sem modificações, ela não terá o efeito desejado, podendo colocar em risco a sua instalação pela falsa impressão de segurança. Portanto, é recomendável sempre testar as modificações para ter certeza que elas tem efeito.

10.6.5 Proteção contra IP spoofing

A especificação de endereços de origem/destino junto com a interface de rede pode ser usado como um detector de ataques spoofing. A lógica é que todos os endereços que NUNCA devem vir da interface X devem ser negados imediatamente. As regras abaixo são colocadas no inicio do chain INPUT para detectar tais ataques:

```
iptables -A INPUT -s 192.168.1.0/24 -i ! eth0 -j DROP
iptables -A INPUT ! -s 192.168.1.0/24 -i eth0 -j DROP
```

A primeira regra diz para bloquear todos os endereços da faixa de rede 192.168.1.* que NÃO vem da interface eth0, a segunda regra diz para bloquear todos os endereços que não sejam 192.168.1.* vindos da interface eth0. O símbolo "!" serve para especificar exceções (veja Especificando uma exceção, Seção 10.3.5. O kernel do `Linux` automaticamente bloqueia a passagem de pacotes que dizem ser de 127.0.0.1 e não está vindo da interface loopback.

O método preferido para controlar o ip spoofing é através do código de roteamento do kernel (a não ser que esteja usando algum tipo de roteamento de origem assimétrico necessário por alguns programas especiais):

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

Desta forma qualquer endereço dizendo ser 192.168.1.5 vindo de `ppp0` será imediatamente rejeitado. Uma checagem adicional contra IP spoofing pode ser feita no arquivo `/etc/host.conf` (veja /etc/host.conf, Seção 4.6.2.2).

10.6.6 Especificando múltiplas portas de origem/destino

O módulo `multiport` permite que seja especificado múltiplas portas para um alvo. Podem ser especificadas até 15 portas em um único parâmetro e basta que uma porta confira para que a regra entre em ação, pois a comparação é feita usando condições "or". O parâmetro `multiport` deve ser acompanhado de um dos argumentos abaixo:

- `--source-port [porta1, porta2...]` - Faz a regra conferir se a porta de origem estiver presente entre as portas especificadas.
- `--destination-port [porta1, porta2...]` - Faz a regra conferir se a porta de destino estiver presente entre as portas especificadas.
- `--port [porta1, porta2...]` - Faz a regra conferir caso a porta de origem ou destino esteja presente no parâmetro.

Este módulo pode eliminar muitas regras de firewall que fazem o mesmo tratamento de pacotes para diversas portas diferentes.

```
iptables -A INPUT -p tcp -i ppp0 -m multiport --destination-port 21,23,25,80,110,113,6667 -j DROP
```

Bloqueia todos os pacotes vindo de ppp0 para as portas 21 (ftp), 23 (telnet), 25 (smtp), 80 (www), 110 (pop3), 113 (ident), 6667 (irc).

10.6.7 Especificando o endereço MAC da interface

O módulo `mac` serve para conferir com o endereço Ethernet dos pacotes de origem. Somente faz sentido se usado nos chains de PREROUTING (da tabela nat) ou INPUT (da tabela filter). Aceita como argumento a opção *--mac-source endereço*. O símbolo "!" pode ser usado para especificar uma exceção.

```
iptables -t filter -A INPUT -m mac --mac-source 00:80:AD:B2:60:0B -j DROP
```

Confere com a máquina com endereço ethernet igual a `00:80:AD:B2:60:0B`.

10.6.8 Conferindo com quem criou o pacote

Este módulo confere com o usuário que iniciou a conexão. É somente válido no chain *OUTPUT* da tabela filter. Os seguintes argumentos são válidas para este módulo:

- `--uid-owner UID` - Confere se o pacote foi criado por um processo com o UID especificado. Até o momento somente UID numéricos são aceitos.
- `--gid-owner GID` - Confere se o pacote foi criado por um usuário pertencente ao grupo GID. Até o momento somente GID numéricos são aceitos.
- `--pid-owner PID` - Confere se o pacote foi criado por um processo com o PID especificado.
- `--sid-owner ID` - Confere se o pacote foi criado por um processo no grupo de seção especificado.

OBS: - Lembre-se que pacotes que não possuem detalhes suficientes de cabeçalho nunca conferirão!

```
iptables -A OUTPUT -m owner --gid-owner 100 -p udp -j DROP
```

Rejeita um conexões indo para portas UDP de pacotes criados pelo usuários pertencentes ao grupo 100.

10.6.9 Conferindo com o conteúdo do pacote

O módulo `string` do `iptables` permite a inspeção de conteúdo de um pacote e tomar uma ação se determinado tipo de tráfego for encontrado em um pacote. Esta técnica pode ser usada tanto para segurança como para economia de banda dentro da rede. Esta opção *NÃO* torna o iptables como um firewall proxy, pois o proxy tem a habilidade de inspecionar o conteúdo, protocolo, comandos do pacote e decidir se o seu conteúdo é nocivo ou não. O firewall em nível de pacotes fazendo inspeção de conteúdo, chega a ser 3 a 10 vezes mais rápido do que um proxy, assim seu uso deve ser analisado dependendo do tráfego que circula pelo link e da segurança dos dados que trafegam através dele.

Uma boa prática é aliar esta opção a um IDS externo usando o alvo *QUEUE* e deixando o trabalho de espeção de conteúdo para ele. Um exemplo de restrição direta é o bloqueio do envio de qualquer informação confidencial sigilosa para fora da rede interna (número de contas, tudo que conferir com CPF, CGC, endereços de e-mail, memorandos, etc). De qualquer forma, analise o tráfego de sua rede antes de querer implementar qualquer solução baseada neste método sob o risco de afetar tráfego legítimo.

Outra utilidade eficiente é a diminuição de tráfego, pois podemos barrar programas que sobrecarregam o link em uma rede com muitos usuários como, por exemplo, usando o `Kazaa` ou qualquer outro programa para cópia de arquivos via Internet. Veja alguns exemplos:

```
# Bloqueia qualquer tentativa de acesso ao programa Kazaa
iptables -A INPUT -m string --string "X-Kazaa" -j DROP

# Não permite que dados confidenciais sejam enviados para fora da empresa
# e registra o ocorrido.
iptables -A OUTPUT -m string --string "conta" -j LOG --log-prefix "ALERTA: dados confidencial "
iptables -A OUTPUT -m string --string "conta" -j DROP

# Somente permite a passagem de pacotes que não contém ".exe" em seu conteúdo
iptables -A INPUT -m string --string ! ".exe" -j ACCEPT
```

10.6.10 Conferindo com o tempo de vida do pacote

O módulo `ttl` pode ser usado junto com as seguintes opções para conferir com o tempo de vida (TTL) de um pacote:

- `--ttl-eq [num]`
- `--ttl-lt [num]`
- `--ttl-gq [num]`

Veja alguns exemplos:

```
# Confere com todos os pacotes que tem o TTL maior que 100
iptables -A INPUT -m ttl --ttl-gt 100 -j LOG --log-prefix "TTL alto"

# Confere com todos os pacotes que tem o TTL igual a 1
iptables -A INPUT -m ttl --ttl-eq 1 -j DROP
```

OBS: Tenha um especial cuidado durante a programação de regras que usem TTL, como elas estão especialmente associadas com o estado da comunicação estabelecida entre as duas pontas e o tipo de protocolo, cuidados especiais devem ser tomados para que seu firewall não manipule de forma incorreta tráfego válido.

10.6.11 Conferindo com números RPC

O módulo `rpc` permite um controle especial sobre o tráfego RPC que chega até a sua máquina. Um uso útil é restringir a chamada a determinados números RPC e permitir outros (por exemplo, permitindo somente o serviço *keyserv* e bloqueando outros como o *ypserv* ou *portmapper*). As seguintes opções podem ser usadas com o módulo `nfs`:

- `--rpcs [procedimentos]` - Confere com a lista de chamadas RPC especificadas. Mais de um procedimento RPC pode ser especificado como `nome` ou `número` separando-os com vírgulas. Um arquivo útil que contém esta lista é o `/etc/rpc`.
- `--strict` - Ignora serviços RPC que não contenham a chamada *get* do portmapper. Em situações normais, o inicio de qualquer solicitação RPC.

Veja alguns exemplos:

```
# Para conferir com todas as chamadas RPC referentes a conexões iniciadas
# para o portmapper
iptables -A INPUT -m rpc --rpcs portmapper --strict -j DROP

# Para permitir que somente as chamadas para status e statmon sejam
# aceitas
iptables -A INPUT -m rpc --rpcs 100023,100024 -j ACCEPT
```

10.6.12 Conferindo com tipo de pacote

O módulo `pkttype` permite identificar um pacote do tipo *unicast* (direcionado a você), *broadcast* (direcionado a uma determinada rede, definida pela netmask) ou *multicast* (destinado a grupos de redes) e desta forma realizar ações em cima destes. O tipo de pacote é identificado logo após a opção *--pkt-type*. Veja alguns exemplos:

```
# Bloqueia a passagem de pacotes multicast de uma rede para outra
iptables -A FORWARD -i eth0 -o eth0 -m pkttype --pkt-type multicast -j DROP

# Como deve ter notado, é possível fazer a associação com diversas especificações
# de módulos, bastando apenas especificar uma opção "-m" para cada módulo
# adicional:
# Permite a passagem de pacotes broadcast de uma rede para outra com
# limitação de 5/s.
iptables -A FORWARD -i eth0 -o eth0 -m pkttype --pkt-type broadcast -m limit --limit 5/s -j ACCEPT
```

10.6.13 Conferindo com o tamanho do pacote

O tamanho do pacote pode ser usado como condição de filtragem através do módulo `length`. O tamanho do pacote é especificado através da opção *--length* e o argumento segue a mesma sintaxe da especificação de portas no `iptables` sendo separados por `:`. Veja alguns exemplos:

```
# Bloqueia qualquer pacote ICMP maior que 30Kb
iptables -A INPUT -i eth0 -m length --length 30000: -j DROP

# Bloqueia qualquer pacote com o tamanho entre 20 e 2000 bytes
iptables -A INPUT -i eth0 -m length --length 20:2000 -j DROP
```

10.7 Caminho percorrido pelos pacotes nas tabelas e chains

É MUITO importante entender a função de cada filtro e a ordem de acesso dos chains de acordo com o tipo de conexão e interface de origem/destino. Esta seção explica a ordem que as regra são atravessadas, isso lhe permitirá planejar a distribuição das regras nos chains, e evitar erros de localização de regras que poderia deixar seu firewall com sérios problemas de segurança, ou um sistema de firewall totalmente confuso e sem lógica.

Nos exemplos abaixo assumirei a seguinte configuração:

- A máquina do firewall com `iptables` possui o endereço IP `192.168.1.1` e conecta a rede interna ligada via interface `eth0` a internet via a interface `ppp0`.
- Rede interna com a faixa de endereços `192.168.1.0` conectada ao firewall via interface `eth0`
- Interface `ppp0` fazendo conexão com a Internet com o endereço IP `200.217.29.67`.
- A conexão das máquinas da rede interna (`eth0`) com a rede externa (`ppp0`) é feita via *Masquerading*.

Também utilizarei a sintaxe *CHAIN-tabela* para fazer referência aos chains e tabelas dos blocos ASCII: *INPUT-filter* - chain *INPUT* da tabela *filter*.

ATENÇÃO: A ordem de processamento das regras do `iptables`, é diferente do `ipchains` devido a inclusão do novo sistema de nat e da tabela mangle.

10.7.1 Ping de 192.168.1.1 para 192.168.1.1

- Endereço de Origem: `192.168.1.1`
- Endereço de Destino: `192.168.1.1`
- Interface de Entrada: `lo`

- Interface de Saída: `lo`
- Protocolo: `ICMP`
- Descrição: `Ping para o próprio firewall`

```
SAÍDA DE PACOTES (envio do ping para 192.168.1.1):
+-------------+    +----------+    +-------------+   +------------------+  +----------------+
|OUTPUT-mangle| => |OUTPUT-nat| => |OUTPUT-filter| =>|POSTROUTING-mangle|=>|POSTROUTING-nat |
+-------------+    +----------+    +-------------+   +------------------+  +----------------+

ENTRADA DOS PACOTES (Retorno da resposta ping acima):
+-----------------+   +------------+  +------------+
|PREROUTING-mangle| =>|INPUT-mangle|=>|INPUT-filter|
+-----------------+   +------------+  +------------+
```

Quando damos o ping (*echo request*) os pacotes seguem o caminho em *SAÍDA DE PACOTES* percorrendo os chains na ordem especificada e retornam via *ENTRADA DOS PACOTES* (*echo reply*). No envio da resposta da requisição de ping, o caminho de saída do pacote ignora os chains OUTPUT-nat e POSTROUTING-nat (já que não é necessário nat) mas sempre processa os chains correspondentes da tabela mangle na ordem indicada acima.

OBS1: Para conexões com destinos na própria máquina usando um endereço IP das interfaces locais, a interface será ajustada sempre para `lo` (loopback).

OBS2: Em qualquer operação de entrada/saída de pacotes, os dois chains da tabela *mangle* são sempre os primeiros a serem acessados. Isto é necessário para definir a prioridade e controlar outros aspectos especiais dos pacotes que atravessam os filtros.

OBS3: O chain *OUTPUT* da tabela *filter* é consultado sempre quando existem conexões se originando em endereços de interfaces locais.

10.7.2 Conexão FTP de 192.168.1.1 para 192.168.1.1

- Endereço de Origem: `192.168.1.1`
- Endereço de Destino: `192.168.1.1`
- Interface de Origem: `lo`
- Interface de Destino: `lo`
- Porta Origem: `1404`
- Porta Destino: `21`
- Protocolo: `TCP`
- Descrição: `Conexão ftp (até o prompt de login, sem transferência de arquivos).`

```
SAÍDA DOS PACOTES (envio da requisição para 192.168.1.1):
+-------------+    +----------+    +-------------+    +------------------+    +---------------+
|OUTPUT-mangle| => |OUTPUT-nat| => |OUTPUT-filter| => +POSTROUTING-mangle| => |POSTROUTING-nat|
+-------------+    +----------+    +-------------+    +------------------+    +---------------+

ENTRADA DE PACOTES (respostas da requisição vindas de 192.168.1.1):
+-----------------+    +------------+    +------------+
|PREROUTING-mangle| => |INPUT-mangle| => |INPUT-filter|
+-----------------+    +------------+    +------------+
```

A requisição ftp passa através dos chains especificados em *SAÍDA DOS PACOTES* e retorna por *ENTRADA DE PACOTES*. Após a conexão ser estabelecida, o caminho de *SAÍDA DE PACOTES* será:

```
+-------------+    +-------------+    +------------------+
|OUTPUT-mangle| => |OUTPUT-filter| => |POSTROUTING-mangle|
+-------------+    +-------------+    +------------------+
```

pois os dados de entrada que vem da interface externa, são passados diretamente a máquina do firewall, não necessitando de tratamento SNAT (os chains *OUTPUT-nat* e *POSTROUTING-nat* são processado somente uma vez a procura de regras que conferem, principalmente para fazer SNAT). Note novamente que mesmo não sendo necessário NAT, o chain POSTROUTING-mangle é checado.

OBS1: Para conexões com destinos na própria máquina usando um endereço IP das interfaces locais, a interface será ajustada sempre para `lo` (loopback).

OBS2: Em qualquer operação de entrada/saída de pacotes, os dois chains da tabela mangle são sempre os primeiros a serem acessados. Isto é necessário para definir a prioridade e controlar outros aspectos especiais dos pacotes que atravessam os filtros.

10.7.3 Conexão FTP de 192.168.1.1 para 192.168.1.4

- Endereço de Origem: `192.168.1.1`
- Endereço de Destino: `192.168.1.4`
- Interface de Origem: `eth0`
- Interface de Destino: `eth0`
- Porta Origem: `1405`
- Porta Destino: `21`
- Protocolo: `TCP`
- Descrição: `Conexão ftp (até o prompt de login, sem transferência de arquivos).`

```
SAÍDA DOS PACOTES (envio da requisição para 192.168.1.4):
+-------------+    +----------+    +-------------+    +------------------+    +---------------+
|OUTPUT-mangle| => |OUTPUT-nat| => |OUTPUT-filter| => +POSTROUTING-mangle| => |POSTROUTING-nat|
+-------------+    +----------+    +-------------+    +------------------+    +---------------+

ENTRADA DE PACOTES (respostas da requisição de 192.168.1.4):
+-----------------+    +------------+    +------------+
|PREROUTING-mangle| => |INPUT-mangle| => |INPUT-filter|
+-----------------+    +------------+    +------------+
```

A requisição ftp passa através dos chains especificados em *SAÍDA DOS PACOTES* com o destino `192.168.1.4` porta `21` e retorna por *ENTRADA DE PACOTES* para `192.168.1.1` porta `1405`. Após a conexão ser estabelecida, o caminho de *SAÍDA DE PACOTES* será:

```
+-------------+    +-------------+    +------------------+
|OUTPUT-mangle| => |OUTPUT-filter| => |POSTROUTING-mangle|
+-------------+    +-------------+    +------------------+
```

pois os dados não precisam de tratamento SNAT (os chains *OUTPUT-nat* e *POSTROUTING-nat* são processado somente uma vez a procura de regras que conferem, principalmente para fazer SNAT).

OBS: Em qualquer operação de entrada/saída de pacotes, os dois chains da tabela mangle são sempre os primeiros a serem acessados. Isto é necessário para definir a prioridade e controlar outros aspectos especiais dos pacotes que atravessam os filtros.

10.7.4 Conexão FTP de 200.217.29.67 para a máquina ftp.debian.org.br

- Endereço de Origem: `200.217.29.67`
- Endereço de Destino: `200.198.129.162`
- Interface de Origem: `ppp0`
- Interface de Destino: `ppp0`
- Porta Origem: `1407`
- Porta Destino: `21`
- Protocolo: `TCP`
- Descrição: `Conexão ftp (até o prompt de login, sem transferência de arquivos).`

```
SAÍDA DOS PACOTES (envio da requisição para 200.198.129.162):
+-------------+    +----------+    +-------------+    +------------------+    +---------------+
|OUTPUT-mangle| => |OUTPUT-nat| => |OUTPUT-filter| => +POSTROUTING-mangle| => |POSTROUTING-nat|
+-------------+    +----------+    +-------------+    +------------------+    +---------------+
```

```
ENTRADA DE PACOTES (respostas da requisição vindas de 200.198.129.162):
+-----------------+    +------------+    +------------+
|PREROUTING-mangle| => |INPUT-mangle| => |INPUT-filter|
+-----------------+    +------------+    +------------+
```

A requisição ftp passa através dos chains especificados em *SAÍDA DOS PACOTES* com o destino `200.198.129.162` porta `21` (após a resolução DNS de `www.debian.org.br`) e retorna por *ENTRADA DE PACOTES* para `200.217.29.67` porta `1407`. Após a conexão ser estabelecida, o caminho de saída de pacotes é:

```
+-------------+    +-------------+    +------------------+
|OUTPUT-mangle| => |OUTPUT-filter| => |POSTROUTING-mangle|
+-------------+    +-------------+    +------------------+
```

pois os dados não precisam de tratamento SNAT (os chains *OUTPUT-nat* e *POSTROUTING-nat* são processado somente uma vez a procura de regras que conferem, principalmente para fazer SNAT).

E após a conexão estabelecida, o caminho de entrada de pacotes passa a ser:

```
+-----------------+    +------------+    +------------+
|PREROUTING-mangle| => |INPUT-mangle| => |INPUT-filter|
+-----------------+    +------------+    +------------+
```

pois os dados não precisam de tratamento DNAT (o chain *PREROUTING-nat* é processado somente uma vez a procura de regras que conferem, principalmente para fazer DNAT).

OBS: Para qualquer operação de entrada/saída de pacotes, os dois chains da tabela mangle são sempre os primeiros a serem acessados. Isto é necessário para definir a prioridade e controlar outros aspectos especiais dos pacotes que atravessam os filtros.

10.7.5 Ping de 192.168.1.4 para 192.168.1.1

- Endereço de Origem: `192.168.1.4`
- Endereço de Destino: `192.168.1.1`
- Interface de Entrada: `eth0`
- Interface de Saída: `eth0`
- Protocolo: `ICMP`
- Descrição: `Ping de 192.168.1.4 para a máquina do firewall.`

```
ENTRADA DE PACOTES (recebimento da requisição, vinda de 192.168.1.4):
+-----------------+    +--------------+    +------------+    +------------+
|PREROUTING-mangle| => |PREROUTING-nat| => |INPUT-mangle| => |INPUT-filter|
+-----------------+    +--------------+    +------------+    +------------+

SAÍDA DE PACOTES (envio da resposta a 192.168.1.4)
+-------------+    +--------------+    +------------------+
|OUTPUT-mangle| => |OUTPUT-filter| => |POSTROUTING-mangle|
+-------------+    +-------------+    +------------------+
```

Quando damos o ping (*echo request*) os pacotes seguem o caminho em *ENTRADA DE PACOTES* percorrendo os chains na ordem especificada e retornam via *SAÍDA DOS PACOTES* (*echo reply*).

OBS1: Para qualquer operação de entrada/saída de pacotes, os dois chains da tabela mangle são sempre os primeiros a serem acessados. Isto é necessário para definir a prioridade e controlar outros aspectos especiais dos pacotes que atravessam os filtros.

10.7.6 Conexão FTP de 192.168.1.4 para 192.168.1.1

- Endereço de Origem: `192.168.1.4`
- Endereço de Destino: `192.168.1.1`
- Interface de Origem: `eth0`
- Interface de Destino: `eth0`
- Porta Origem: `1030`
- Porta Destino: `21`
- Protocolo: `TCP`
- Descrição: `Conexão ftp (até o prompt de login, sem transferência de dados).`

```
ENTRADA DOS PACOTES (envio da requisição vindas de 192.168.1.4):
+-----------------+    +--------------+    +------------+    +------------+
|PREROUTING-mangle| => |PREROUTING-nat| => |INPUT-mangle| => |INPUT-filter|
+-----------------+    +--------------+    +------------+    +------------+

SAÍDA DE PACOTES (respostas da requisição acima para 192.168.1.4):
+-------------+    +--------------+    +------------------+
|OUTPUT-mangle| => |OUTPUT-filter| => |POSTROUTING-mangle|
+-------------+    +-------------+    +------------------+
```

A requisição ftp passa através dos chains especificados em *ENTRADA DOS PACOTES* com o destino `192.168.1.1` porta `21` e retorna por *SAÍDA DE PACOTES* para `192.168.1.4` porta `1030`. Após a conexão ser estabelecida, o caminho de entrada de pacotes é:

```
+-----------------+    +------------+    +------------+
|PREROUTING-mangle| => |INPUT-mangle| => |INPUT-filter|
+-----------------+    +------------+    +------------+
```

pois os dados não precisam de tratamento DNAT (o chain *PREROUTING-nat* é processado somente uma vez a procura de regras que conferem, principalmente para fazer DNAT).

OBS: O roteamento é sempre realizado após o processamento do chain *PREROUTING* da tabela *nat*.

10.7.7 Conexão FTP de 192.168.1.4 para ftp.debian.org.br

- Endereço de Origem: `192.168.1.4`
- Endereço de Destino: `200.198.129.162`
- Interface de Origem: `eth0`
- Interface de Destino: `ppp0`
- Porta Origem: `1032`
- Porta Destino: `21`
- Protocolo: `TCP`
- Descrição: `Conexão ftp (até o prompt de login, sem transferência de dados).`

```
SAÍDA DOS PACOTES (requisição vindas de 192.168.1.4):
+-----------------+    +--------------+    +--------------+
|PREROUTING-mangle| => |PREROUTING-nat| => |FORWARD-mangle| => (continua abaixo)
+-----------------+    +--------------+    +--------------+
+--------------+    +-----------------+    +---------------+
|FORWARD-filter| => |POSTROUTING-mangle| => |POSTROUTING-nat|
+--------------+    +-----------------+    +---------------+

ENTRADA DE PACOTES (respostas da requisição acima, enviadas para 192.168.1.4):
+-----------------+    +--------------+    +--------------+    +-----------------+
|PREROUTING-mangle| => |FORWARD-mangle| => |FORWARD-filter| => |POSTROUTING-mangle|
+-----------------+    +--------------+    +--------------+    +-----------------+
```

A requisição ftp passa através dos chains especificados em *SAÍDA DOS PACOTES* com o destino `200.198.129.162` porta `21` (após a resolução DNS de `ftp.debian.org.br`) e retorna por *ENTRADA DE PACOTES* para `192.168.1.4` porta `1032`.

Note que o Masquerading regrava os pacotes; para a máquina `200.198.129.162` a conexão está sendo feita para `200.217.29.67`. As respostas de conexões vindas de `200.198.129.162` e indo para `200.217.29.67` são regravadas no firewall com o destino `192.168.1.4` e enviadas para a máquina correspondente. Após a conexão ser estabelecida, o caminho de saída de pacotes para 200.198.129.163 é:

```
+-----------------+    +--------------+    +--------------+    +-----------------+
|PREROUTING-mangle| => |FORWARD-mangle| => |FORWARD-filter| => |POSTROUTING-mangle|
+-----------------+    +--------------+    +--------------+    +-----------------+
```

Após a conexão estabelecida, o caminho da entrada de pacotes vindos de 200.198.129.163 é:

```
+-----------------+    +-------------+    +-------------+    +-----------------+
|PREROUTING-mangle| => |FORWARD-mangle| => |FORWARD-filter| => |POSTROUTING-mangle|
+-----------------+    +-------------+    +-------------+    +-----------------+
```

Isto acontece porque após feita a conexão Masquerading (via PREROUTING-nat), o firewall já sabe como reescrever os pacotes para realizar a operação de Masquerading, reescrevendo todos os pacotes que chegam de `www.debian.org.br` para `192.168.1.4`.

OBS: As conexões Masquerading feitas através da rede interna, são enviadas para `200.198.129.162` tem o endereço de origem ajustado para `200.217.29.67` que é o IP de nossa interface `ppp0`. Quando as respostas atravessam o firewall, os pacotes são checados pra saber se são uma resposta a uma conexão masquerading e fará a regravação dos pacotes substituindo o endereço de destino para `192.168.1.4`. Caso uma operação de Masquerading falhe, os pacotes serão Bloqueados.

10.7.8 Conexão FTP de 200.198.129.162 para 200.217.29.167

- Endereço de Origem: `200.198.129.162`
- Endereço de Destino: `200.217.29.67`
- Interface de Origem: `ppp0`
- Interface de Destino: `ppp0`
- Porta Origem: `3716`
- Porta Destino: `21`
- Protocolo: `TCP`
- Descrição: `Conexão ao serviço ftp do firewall`

```
ENTRADA DOS PACOTES (requisição vinda de 200.198.129.162):
+-----------------+    +-------------+    +-------------+    +------------+
|PREROUTING-mangle| => |PREROUTING-nat| => |INPUT-mangle | => |INPUT-filter|
+-----------------+    +-------------+    +-------------+    +------------+

SAÍDA DE PACOTES (respostas da requisição de 200.198.129.162):
+-------------+    +-------------+    +------------------+
|OUTPUT-mangle| => |OUTPUT-filter| => |POSTROUTING-mangle|
+-------------+    +-------------+    +------------------+
```

A requisição ftp passa através dos chains especificados em *ENTRADA DOS PACOTES* com o destino `200.217.29.67` (nossa interface `ppp0` local) porta `21` e retorna por *SAÍDA DE PACOTES* para `200.198.129.162` porta `3716` (também via `ppp0`). Após a conexão ser estabelecida, o caminho de entrada de pacotes é:

```
+-----------------+    +------------+    +------------+
|PREROUTING-mangle| => |INPUT-mangle| => |INPUT-filter|
+-----------------+    +------------+    +------------+
```

Isto acontece porque após feita a análise do chain *PREROUTING* (para necessidade de DNAT), a máquina já saberá tomar a decisão apropriada para gerenciar aquela conexão.

10.7.9 Gráfico geral da passagem dos pacotes

Este gráfico foi retirado do documento `netfilter-hacking-HOWTO.txt` e mostra a estrutura geral de passagem dos pacotes nas tabelas/chains. Os exemplos de passagem de pacotes acima poderão ser facilmente comparados com as etapas abaixo para compreender a estrutura do `iptables`.

```
E ---> PREROUTING ------> (ROTEAM.) ---> FORWARD ----------> POSTROUTING --> S
       Mangle e              |           Mangle       ^      Mangle
       NAT (DNAT))           |           Filter       |      NAT (SRC)
                             |                     (ROTEAM.)
                             v                       |
                             IN Mangle,              OUT - Mangle,
                             |  Filter                 ^    NAT (DNAT)
                             |                         |    Filter
                             v                         |
                    +--------------------------------------+
                    |            Processo Local            |
                    +--------------------------------------+
```

10.8 Exemplos de configurações do iptables

Exemplo de como bloquear todas as conexões para a máquina do firewall permitindo somente conexões da máquina Linux para fora.

10.8.1 Bloqueando conexões de fora para sua máquina

As regras a seguir servem para bloquear tentativas de conexões da interface de Internet (ppp0) a sua rede sem bloquear o tráfego de conexões já iniciadas. O tráfego de outras interfaces não é afetado com as regras a seguir:

```
iptables -A INPUT -i ppp0 -m state --state ! ESTABLISHED,RELATED -j DROP
```

Todas as conexões vindas de ppp0 de estado diferente de ESTABLISHED e RELATED (NEW e INVALID) serão derrubadas. Veja Conferindo de acordo com o estado da conexão, Seção 10.6.1 para detalhes.

```
iptables -A INPUT -i ppp0 --syn -j DROP
```

Este acima é mais simples e possui o mesmo efeito: Pacotes SYN são usados para iniciar conexões, derrubando pacotes deste tipo significa bloquear novas conexões. Pacotes de conexões já estabelecidas ainda são permitidos.

Estas regras acima servem para quem não deseja NENHUM acesso indevido a sua máquina. Existem outras formas de bloquear conexões de modo mais seletivo usando chains específicos, endereços de origem/destino, portas, etc., este tipo de configuração é muito usada caso precise fornecer algum tipo de serviço que seja acessível externamente e protegendo outros.

10.8.2 Monitorando tentativa de conexão de trojans em sua máquina

As regras abaixo alertam sobre a tentativa de conexão dos trojans "For Win" mais conhecidos. Coloquei isto aqui por curiosidade de algumas pessoas, pois máquinas `Linux` são imunes a este tipo de coisa:

```
#!/bin/sh

TROJAN_PORTS="12345 31336 31337 31338 3024 4092 5714 5742 2583 8787 5556 5557"

iptables -t filter -N trojans-in

for PORTA in ${TROJAN_PORTS};do
 iptables -A trojans-in -p tcp --sport=1024: --dport=${PORTA} -j LOG \
          --log-prefix "FIREWALL: Trojan ${PORTA} "
 iptables -A trojans-in -p tcp --sport=1024: --dport=${PORTA} -j DROP
done

iptables -t filter -A INPUT -i ppp0 -j trojans-in
```

A primeira linha do `iptables` cria o chain *trojans-in* dentro da tabela *filter* que usaremos para armazenar nossas regras de firewall. A segunda (dentro do laço for) faz uma regra de LOG para registrar as tentativas de acesso de trojans em nosso sistema, a terceira rejeita o acesso. A quarta regra do `iptables` cria de todo o tráfego vindo da interface ppp0 pra o chain trojans-in (queremos que só o tráfego vindo da internet seja analisado pelo chain *trojans-in*).

Muitas das portas especificadas na variável *TROJAN_PORTS* são antigas conhecidas de quem já brincou ou sofreram com o `Back Orifice`, `Win Crack`, `NetBus` (quem nunca passou pela fase de ter uma lista com mais de 100 netmasks e conseguir encontrar centenas de máquinas por dia infectadas pelo BO? :-).

No código acima a única coisa que precisa fazer para adicionar mais portas é inseri-las na variável *TROJAN_PORTS* e executar o programa. O laço do `for` executará as 2 regras para cada porta processada (economizando linhas e linhas de regras, me livrando de uma LER e poupando muitos bytes neste guia ;-).

Dependendo do número de portas alvo, este código pode ser muito simplificado usando o recurso multiport do `iptables` (veja Especificando múltiplas portas de origem/destino, Seção 10.6.6 para detalhes).

10.8.3 Conectando sua rede interna a Internet

O seguinte exemplo permite ligar sua rede interna com a faixa de IP's `192.168.1.*` a internet (usando uma conexão discada do tipo ppp):

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp+ -j MASQUERADE
echo "1" >/proc/sys/net/ipv4/ip_forward
```

10.8.4 Um exemplo de firewall simples

Esta seção possui um exemplo mais elaborado de firewall que servirá para máquinas conectadas via ppp com uma rede interna conectada via Masquerading. Este exemplo não é tão complexo e cobre as expectativas mais comuns de pessoas que gostam de explorar os potenciais de rede no Linux ou satisfazer sua curiosidade. Ele poderá ser facilmente adaptado para atender outro tipo de necessidade. A configuração assumida é a seguinte:

Máquina do firewall com 2 interfaces de rede, uma é eth0 com o IP 192.168.1.1 que serve de ligação a sua rede Interna, a outra é ppp0 que é a interface Internet.

Qualquer acesso externo a máquinas da rede interna é bloqueado.

Os usuários da rede local tem acesso livre ao servidor Linux.

Qualquer acesso externo a máquina do firewall é bloqueado, exceto conexões para o serviço Apache (httpd). Outras tentativas de conexões devem ser explicitamente registradas nos logs do sistema para conhecimento do administrador.

Todos os usuários possuem acesso livre a Internet via Masquerading, exceto que o acesso para o serviço www deve ser obrigatoriamente feito via squid, e o servidor smtp a ser usado deverá ser o do firewall Linux.

Prioridades serão estabelecidas para os serviços de telnet, IRC,talk e DNS.

```
#!/bin/sh
# Modelo de configuração de firewall
# Autor: Gleydson M. Silva
# Data: 05/09/2001
# Descrição: Produzido para ser distribuído livremente, acompanha o guia
#             Foca GNU/Linux. http://www.guiafoca.org
#

# É assumido um sistema usando kmod para carga automática dos módulos usados por
# esta configuração do firewall:
# ipt_filter
# ipt_nat
# ipt_conntrack
# ipt_mangle
# ipt_TOS
# ipt_MASQUERADE
# ipt_LOG

# Se você tem um kernel modularizado que não utiliza o kmod, será necessário
# carregar estes módulos via modprobe, insmod ou iptables --modprobe=modulo

##### Definição de política padrão do firewall #####
# Tabela filter
iptables -t filter -P INPUT DROP
iptables -t filter -P OUTPUT ACCEPT
iptables -t filter -P FORWARD DROP
# Tabela nat
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P OUTPUT ACCEPT
iptables -t nat -P POSTROUTING DROP
# Tabela mangle
iptables -t mangle -P PREROUTING ACCEPT
iptables -t mangle -P OUTPUT ACCEPT


##### Proteção contra IP Spoofing #####
for i in /proc/sys/net/ipv4/conf/*/rp_filter; do
 echo 1 >$i
done

##### Ativamos o redirecionamento de pacotes (requerido para NAT) #####
echo "1" >/proc/sys/net/ipv4/ip_forward

# O iptables define automaticamente o número máximo de conexões simultâneas
# com base na memória do sistema. Para 32MB = 2048, 64MB = 4096, 128MB = 8192,
# sendo que são usados 350 bytes de memória residente para controlar
# cada conexão.
# Quando este limite é excedido a seguinte mensagem é mostrada:
#  "ip_conntrack: maximum limit of XXX entries exceed"
#
# Como temos uma rede simples, vamos abaixar este limite. Por outro lado isto
# criará uma certa limitação de tráfego para evitar a sobrecarga do servidor.
echo "2048" > /proc/sys/net/ipv4/ip_conntrack_max


#################################################################
#                       Tabela filter                           #
#################################################################

##### Chain INPUT #####
# Criamos um chain que será usado para tratar o tráfego vindo da Internet e
iptables -N ppp-input

# Aceita todo o tráfego vindo do loopback e indo pro loopback
iptables -A INPUT -i lo -j ACCEPT
# Todo tráfego vindo da rede interna também é aceito
iptables -A INPUT -s 192.168.1.0/24 -i eth0 -j ACCEPT

# Conexões vindas da interface ppp0 são tratadas pelo chain ppp-input
iptables -A INPUT -i ppp+ -j ppp-input

# Qualquer outra conexão desconhecida é imediatamente registrada e derrubada
iptables -A INPUT -j LOG --log-prefix "FIREWALL: INPUT "
iptables -A INPUT -j DROP


##### Chain FORWARD ####
# Permite redirecionamento de conexões entre as interfaces locais
# especificadas abaixo. Qualquer tráfego vindo/indo para outras
# interfaces será bloqueado neste passo
iptables -A FORWARD -d 192.168.1.0/24 -i ppp+ -o eth0 -j ACCEPT
iptables -A FORWARD -s 192.168.1.0/24 -i eth0 -o ppp+ -j ACCEPT
iptables -A FORWARD -j LOG --log-prefix "FIREWALL: FORWARD "
iptables -A FORWARD -j DROP


##### Chain ppp-input ####
# Aceitamos todas as mensagens icmp vindas de ppp0 com certa limitação
# O tráfego de pacotes icmp que superar este limite será bloqueado
# pela regra "...! ESTABLISHED,RELATED -j DROP" no final do
# chain ppp-input
#
iptables -A ppp-input -p icmp -m limit --limit 2/s -j ACCEPT

# Primeiro aceitamos o tráfego vindo da Internet para o serviço www (porta 80)
iptables -A ppp-input -p tcp --dport 80 -j ACCEPT

# A tentativa de acesso externo a estes serviços serão registrados no syslog
# do sistema e serão bloqueados pela última regra abaixo.
iptables -A ppp-input -p tcp --dport 21 -j LOG --log-prefix "FIREWALL: ftp "
iptables -A ppp-input -p tcp --dport 25 -j LOG --log-prefix "FIREWALL: smtp "
iptables -A ppp-input -p udp --dport 53 -j LOG --log-prefix "FIREWALL: dns "
iptables -A ppp-input -p tcp --dport 110 -j LOG --log-prefix "FIREWALL: pop3 "
iptables -A ppp-input -p tcp --dport 113 -j LOG --log-prefix "FIREWALL: identd "
iptables -A ppp-input -p udp --dport 111 -j LOG --log-prefix "FIREWALL: rpc"
iptables -A ppp-input -p tcp --dport 111 -j LOG --log-prefix "FIREWALL: rpc"
iptables -A ppp-input -p tcp --dport 137:139 -j LOG --log-prefix "FIREWALL: samba "
iptables -A ppp-input -p udp --dport 137:139 -j LOG --log-prefix "FIREWALL: samba "
# Bloqueia qualquer tentativa de nova conexão de fora para esta máquina
iptables -A ppp-input -m state --state ! ESTABLISHED,RELATED -j LOG --log-prefix "FIREWALL: ppp-in "
iptables -A ppp-input -m state --state ! ESTABLISHED,RELATED -j DROP
# Qualquer outro tipo de tráfego é aceito
iptables -A ppp-input -j ACCEPT


#########################################################
#                   Tabela nat                          #
#########################################################

##### Chain POSTROUTING #####
# Permite qualquer conexão vinda com destino a lo e rede local para eth0
iptables -t nat -A POSTROUTING -o lo -j ACCEPT
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j ACCEPT

# Não queremos que usuários tenham acesso direto a www e smtp da rede externa, o
```

```
# squid e smtpd do firewall devem ser obrigatoriamente usados. Também registramos
# as tentativas para monitorarmos qual máquina está tentando conectar-se diretamente.
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp+ -p tcp --dport 80 -j LOG --log-prefix "FIREWALL: SNAT-www "
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp+ -p tcp --dport 25 -j LOG --log-prefix "FIREWALL: SNAT-smtp "
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp+ -p tcp --dport 25 -j DROP
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp+ -p tcp --dport 80 -j DROP
# É feito masquerading dos outros serviços da rede interna indo para a interface
# ppp0
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ppp+ -j MASQUERADE

# Qualquer outra origem de tráfego desconhecida indo para eth0 (conexões vindas
# de ppp+) são bloqueadas aqui
iptables -t nat -A POSTROUTING -o eth0 -d 192.168.1.0/24 -j LOG --log-prefix "FIREWALL: SNAT unknown"
iptables -t nat -A POSTROUTING -o eth0 -d 192.168.1.0/24 -j DROP
# Quando iniciamos uma conexão ppp, obtermos um endereço classe A (10.x.x.x) e após
# estabelecida a conexão real, este endereço é modificado. O tráfego indo para
# a interface ppp não deverá ser bloqueado. Os bloqueios serão feitos no
# chain INPUT da tabela filter
iptables -t nat -A POSTROUTING -o ppp+ -j ACCEPT

# Registra e bloqueia qualquer outro tipo de tráfego desconhecido
iptables -t nat -A POSTROUTING -j LOG --log-prefix "FIREWALL: SNAT "
iptables -t nat -A POSTROUTING -j DROP


################################################
#               Tabela mangle                  #
################################################

##### Chain OUTPUT #####
# Define mínimo de espera para os serviços ftp, telnet, irc e DNS, isto
# dará uma melhor sensação de conexão em tempo real e diminuirá o tempo
# de espera para conexões que requerem resolução de nomes.
iptables -t mangle -A OUTPUT -o ppp+ -p tcp --dport 21 -j TOS --set-tos 0x10
iptables -t mangle -A OUTPUT -o ppp+ -p tcp --dport 23 -j TOS --set-tos 0x10
iptables -t mangle -A OUTPUT -o ppp+ -p tcp --dport 6665:6668 -j TOS --set-tos 0x10
iptables -t mangle -A OUTPUT -o ppp+ -p udp --dport 53 -j TOS --set-tos 0x10
```

[anterior] [Conteúdo] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [próximo]

Guia Foca GNU/Linux

Versão 6.43 - domingo, 05 de setembro de 2010

Gleydson Mazioli da Silva gleydson@guiafoca.org
