# Linux 2.4 Filtrado de Paquetes (Packet Filtering) COMO

Rusty Russell, *netfilter@lists.samba.org*
Traducido por Ricardo J. Cárdenes Medina *a1402@dis.ulpgc.es*
v1.0.1 Mon May 1 18:09:31 CST 2000

Este documento describe el uso de iptables para filtrar los paquetes malos en los núcleos 2.4 de Linux.

## Índice

# 1. Introducción

Bienvenido, gentil lector.

Se asume que conoce los conceptos de dirección IP, dirección de red, máscara de red, encaminamiento (*routing*) y DNS. En caso contrario, le recomiendo que lea el Conceptos de Redes COMO `http://www.mhpsc.com/~ricardo/COMOs/Conceptos-de-Redes-COMO`

Este COMO está entre una suave introducción (que le dejará sintiéndose entre algodones, pero desprotegido ante el mundo real), y una cruda exposición completa (que dejaría confusas, paranoides y buscando artillería pesada a todas las almas excepto aquellas más duras).

Su red no es **segura**. El problema de permitir comunicaciones rápidas y convenientes al tiempo que se restringe su uso con buenas intenciones, y no con malas es algo congruente a otros problemas intratables como permitir que se hable libremente en un teatro a la vez que se prohibe un grito de «¡Fuego!». Esto no lo vamos a resolver en el espacio de este COMO.

De manera que puede decidir hasta dónde llega el compromiso. Intentaré instruirle en el uso de algunas de las herramientas disponibles y algunas vulnerabilidades de las que tener cuidado, con la esperanza de que las use para el bien, y no para propósitos malvados. Otro problema equivalente.

# 2. ¿Dónde está el sitio web oficial? ¿Hay una lista de correo?

Hay tres sitios oficiales:

- Gracias a Penguin Computing: `http://netfilter.filewatcher.org/` .

- Gracias a el equipo Samba y a SGI *http://netfilter.samba.org/* .
- Gracias a Harald Welte *http://netfilter.gnumonks.org/* .

La lista oficial de correo de netfilter está en el Listserver de Samba: *http://www.netfilter.org/contact.html#list*

# 3. ¿Y qué es un Filtro de Paquetes?

Un filtro de paquetes es un software que examina la *cabecera* de los paquetes según van pasando, y decide la suerte del paquete completo. Podría decidir descartarlo (**DROP**) (esto es, como si nunca lo hubiera recibido), aceptarlo (**ACCEPT**) (dejar que pase), o cosas más complicadas.

En Linux, el filtrado de paquetes está programado en el núcleo (como módulo o como componente estático), y hay algunas cosas curiosas que podemos hacer con los paquetes, pero aún sigue ahí el principio general de mirar las cabeceras para decidir la suerte del paquete.

## 3.1. ¿Para qué querría un Filtro de Paquetes?

Control. Seguridad. Vigilancia.

**Control:**

cuando está usando una máquina Linux para conectar su red interna a otra (por ejemplo, Internet), tiene la oportunidad de permitir ciertos tipos de tráfico, y restringir otros. Por ejemplo, la cabecera de un paquete contiene la dirección de destino del paquete, de manera que puede evitar que salgan los que van a cierto sitio fuera de la red. Otro ejemplo: yo uso le Netscape para acceder a los archivos de Dilbert. Hay anuncios de doubleclick.net en la página, y Netscape malgasta mi tiempo bajándoselos alegremente. Diciéndole a mi filtro de paquetes que no permita que ningún paquete vaya de o hacia las direcciones de doubleclick.net resuelve el problema (hay mejores maneras de hacer esto, sin embargo: vea el Junkbuster).

**Seguridad:**

cuando su máquina Linux es lo único entre el caos de Internet y su bonita y ordenada red, es bueno saber que puede restringir lo que llega aporreando su puerta. Por ejemplo, podríamos permitir todo lo que salga de la red, pero puede que esté preocupado por el bien conocido «Ping de la Muerte» que puede llegar de gente maliciosa del exterior. En otro ejemplo, podría ser que no quisiera que la gente de fuera pueda hacer telnet a su máquina Linux, incluso aunque todas sus cuentas tengan clave. Quizá quiera (como la mayoría) ser un observador en Internet, y no un servidor (). Simplemente, no quiere dejar que nadie se conecte, haciendo que el filtro de paquetes rechace los paquetes entrantes que se usan para establecer conexiones.

**Vigilancia:**

algunas veces, una máquina mal configurada de la red local puede decidir vomitar paquetes al mundo exterior. Es bueno decirle al filtro de paquetes que le avise si ocurre algo anormal; quizá usted pueda hacer algo al respecto, o puede que sencillamente sea curioso por naturaleza.

## 3.2. ¿Cómo hago el Filtrado de Paquetes con Linux?

Los núcleos de Linux han tenido capacidades de filtrado de paquetes de desde la serie 1.1. La primera generación, basada en el ipfw de BSD, fue adaptada por Alan Cox a finales de 1994. Fue mejorada por Jos Vos y otros para Linux 2.0; la herramienta «ipfwadm», ejecutada en espacio de usuario, controlaba las reglas de filtrado del núcleo. A mediados de 1998, rehíce en gran medida esa parte del núcleo, con la ayuda de Michael Neuling, e introduje la herramienta «ipchains». Finalmente, a mediados de 1999 hubo otra reescritura del núcleo, y la herramienta de cuarta generación, «iptables», para Linux 2.4. Es en esta iptables en la que se centra este COMO.

Neceista un núcelo que contenga la infraestructura netfilter. Netfilter es un área de trabajo general dentro del núcleo, a la que pueden conectarse otras cosas (como el módulo de iptables). Esto significa que necesitará un núcleo 2.3.15 o más nuevo, y responder «Y» a CONFIG_NETFILTER en la configuración del núcleo.

La herramienta `iptables` se comunica con el núcleo y le dice qué paquetes filtrar. A menos que sea programador, o muy curioso, es ésta la manera en que controlará el filtrado de paquetes.

### 3.2.1. iptables

La heramienta `iptables` inserta y elimina reglas de la tabla de filtrado de paquetes del núcleo. Esto significa que cualquier cosa que establezca, se perderá cuando reinicie; vea 3.2.2 (Hacer permanentes las reglas) para estar seguro de que serán restauradas la siguiente vez que Linux arranque.

`iptables` es un sustituto para `ipfwadm` e `ipchains`: vea 8 (Uso de ipchains e ipfwadm) para ver cómo evitar iptables y evitarse muchos dolores de cabeza si ya estaba usando una de estas otras herramientas.

### 3.2.2. Hacer permanentes las reglas

La configuración que su cortafuegos tiene en este momento está almacenada en el núcleo, de manera que se perderá cuando reinicie. En mi lista TODO (Por Hacer) está escribir iptables-save e iptables-restore. Cuando existan, serán increíbles, lo prometo.

Mientras tanto, escriba las órdenes necesarias para configurar sus reglas en un guión (script) de inicio. Asegúrese de que hace algo inteligente en el caso de que alguna de las reglas falle (normalmente «exec /sbin/sulogin»)

# 4. ¿Quién demonios es usted, y qué hace jugando con mi núcleo?

oy Rusty; el que mantiene el Linux IP Firewall y sólo otro de los programadores que estaba en el sitio adecuado en el momento preciso. Escribí ipchains (vea 3.2 (¿Cómo hago el Filtrado de Paquetes con Linux?) para leer los créditos a la gente que hizo el verdadero trabajo), y aprendí lo suficiente como para hacer funcionar el filtrado de paquetes de forma correcta. Espero.

WatchGuard *`http://www.watchguard.com`* , una excelente compañía que hace cortafuegos, y que vende el maravilloso plug-in Firebox, ofreció pagarme por no hacer nada, de maner que pudiera usar mi tiempo escribiendo este material, y mantener el que había escrito antes. Predije 6 meses, y me llevó 12, pero al final siento que lo he hecho Bien. Muchas reescrituras, un disco duro estropeado, un portátil robado, un par de sistemas de ficheros corruptos y una pantalla rota más tarde, aquí está.

Aprovechando estas líneas, quiero aclarar algunos conceptos confusos a la gente: no soy un gurú del núcleo. Lo sé, porque mi trabajo en el núcleo me ha puesto en contacto con algunos de ellos: David S. Miller, Alexey Kuznetsov, Andi Kleen, Alan Cox. sin embargo, ellos están muy ocupados haciendo la magia en las profundidades, dejándome vadear en la orilla donde las cosas son más seguras.

# 5. La Guía Verdaderamente Rápida de Rusty para el Filtrado de Paquetes

La mayoría de la gente tiene sólo una conexión PPP a Internet, y no quiere que nadie entre a su red, o al cortafuegos:

```
## Insertar los módulos de seguimiento de la conexión (no es necesario
## si están en el núcleo).
# insmod ip_conntrack
# insmod ip_conntrack_ftp

## Crear una cadena que bloquee las conexiones nuevas, excepto si vienen
## de dentro.
# iptables -N block
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A block -j DROP

## Saltar a esa cadena desde las cadenas INPUT y FORWARD.
# iptables -A INPUT -j block
# iptables -A FORWARD -j block
```

# 6. Cómo pasan los paquetes por los filtros

El núcleo empieza con tres listas de reglas en la tabla de «filtros»; estas listas se llaman **cadenas cortafuegos** o sencillamente **cadenas**. Se llaman **INPUT**, **OUTPUT** y **FORWARD**

**¡Esto es muy diferente a la manera de trabaja de los núcleos 2.0 y 2.2!**

Para los fans del «ASCII-art», las cadenas están dispuestas de esta manera:

```
                              -----
Entrada                      /     \         Salida
       -->[Decisión de   ]--->|FORWARD|------->
          [Encaminamiento]     \_____/        ^
               |                              |
               v                             ----
              ---                           /    \
             /   \                         |OUTPUT|
            |INPUT|                         \____/
             \___/                            ^
               |                              |
                ------> Proceso Local -------
```

Estos tres círculos representan las tres cadenas que mencioné antes. Cuando un paquete alcanza un círculo en el diagrama, se examina esa cadena para decidir la suerte del paquete. Si la cadena dice que hay que descartar (DROP) el paquete, se le mata ahí mismo, pero si la cadena dice que hay aceptarlo (ACCEPT), continúa su camino por el diagrama.

Una cadena es una lista de **reglas**. Cada regla dice «si el paquete se parece a esto, entonces esto otro es lo que hay que hacer con él». Si la regla no se ajusta al paquete, entonces se consulta la siguiente regla en la lista. Al final, si no hay más reglas por consultar, el núcleo mira la **política** de la cadena para decidir qué hacer. En un sistema consciente de la seguridad, esta política suele decirle al núcleo que descarte (DROP) el paquete.

1. Cuando llega un paquete (digamos, por la tarjeta Ethernet) el núcleo mira primero su destino: a esto se le llama «encaminamiento» (routing).

2. Si está destinado a esa misma máquina, el paquete entra en el diagrama hacia la cadena INPUT. Si pasa de aquí, cualquier proceso que esté esperando por el paquete, lo recibirá.

3. En caso contrario, si el núcleo no tiene las capacidades de reenvío activadas, o no sabe hacia dónde reenviar el paquete, se descarta el paquete. Si está activado el reenvío, y el paquete está destinado a otra interfaz de red (si tenemos otra), entonces el paquete pasa directamente a la cadena FORWARD de nuestro diagrama. Si es ACCEPTado, entonces saldrá de la máquina.

4. Finalmente, si un programa que se ejecuta en la máquina puede enviar paquetes de red. Estos paquetes pasan por la cadena OUTPUT de forma inmediata: si los acepta (ACCEPT), entonces el paquete continúa hacia afuera, dirigido a la interfaz a la que estuviera destinada.

# 7. Uso de iptables

iptables dispone de una página de manual bastante detallada (`man iptables`), por si necesita más detalles particulares. Aquellos de ustedes que estén familiarizados con ipchains, posiblemente quieran echar un vistazo a 10 (Diferencias entre iptables e ipchains); son bastante similares.

Con `iptables` se pueden hacer muchas cosas diferentes. Empezamos con tres cadenas de uso interno: `INPUT`, `OUTPUT` y `FORWARD`, que no se pueden borrar. Veamos las operacioes que se pueden hacer en todas las cadenas:

1. Crear una nueva cadena (-N).

2. Borrar una cadena vacía (-X).

3. Cambiar la política de una cadena de uso interno (-P).

4. Listar las reglas de una cadena (-L).

5. Vaciar de reglas una cadena (-F).

6. Poner a cero los contadores de paquetes y bytes de todas las reglas de una cadena (-Z).

Las reglas de una cadena se pueden manipular de varias maneras:

1. Añadir una nueva regla a una cadena (-A).

2. Insertar una nueva regla en alguna posición de la cadena (-I).
3. Mover una regla a otra posición dentro de una cadena (-R).
4. Borrar una regla de un sitio en particular de una cadena (-D).
5. Borrar la primera regla que coincida con los parámetros dados en una cadena (-D).

## 7.1. Qué verá cuando el ordenador arranque

iptables puede ser un módulo, llamado («iptable_filter.o»), que debería cagarse de forma automática la primera vez que ejecute `iptables`. También lo puede hacer parte permanente del núcleo.

Antes de ejecutar alguna orden iptables (tenga cuidado: algunas distribuciones puede que ejecuten iptables en sus guiones de iniciación), no habrá reglas en ninguna de las cadenas de uso interno («INPUT», «FORWARD» y «OUTPUT»), todas las cadenas tendrán la política de ACCEPTar. Se puede alterar la política por defecto de FORWARD proporcionando la opción «forward=0» al módulo iptable_filter.

## 7.2. Operaciones sobre una sola regla

Este es el pan de cada día del filtrado de paquetes: la manipulación de reglas. Lo más común es que utilice las órdenes de agregar (-A) y eliminar (-D). Las otras (-I para insertar y -R para cambiar de sitio) son sólo extensiones de estos conceptos.

Cada regla especifica un conjunto de condiciones que debe cumplir el paquete, y qué hacer si se ajusta a ellas (un «objetivo»). Por ejemplo, podríamos querer descartar todos los paquetes ICMP que viniesen de la dirección IP 127.0.0.1. De manera que en este caso nuestras condiciones son que el protocolo sea ICMP y que la dirección de origen del paquete sea 127.0.0.1. Nuestro objetivo será «DROP».

127.0.0.1 es la interfaz «loopback», de la que dispondrá incluso si no tiene una conexión de red real. Puede usar el programa «ping» para generar tales paquetes (sencillamente envía paquetes ICMP de tipo 8 (echo request) que todos las máquinas cooperantes deberían responder cortésmente con un paquete ICMP de tipo 0 (echo reply)). Esto lo hace útil para las pruebas.

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

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

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

Aquí puede ver que el primer «ping» tuvo éxito (el «-c 1» le indica a ping que sólo envíe un paquete).

Entonces añadimos (-A) a la cadena «INPUT», una regla que especifica que los paquetes que vengan de 127.0.0.1 («-s 127.0.0.1») con protocolo ICMP («-p icmp») deberíamos saltar a DROP («-j DROP»).

Luego probamos nuestra regla, usando el segundo ping. Habrá una pausa antes de que el programa se canse de esperar por una respuesta que nunca llegará.

Podemos borrar la regla de dos maneras. Primero, como sabemos que es la única regla en la cadena, podemos usar un borrado por número:

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

Para borrar la regla número uno de la cadena INPUT.

La segunda manera es repetir la orden -A, pero cambiando -A por -D. Es útil cuando se tiene una compleja cadena de reglas y no queremo estar contándolas para averiguar que es la regla 37 la que queremos eliminar. En este caso, usaríamos:

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

La sintaxis de -D debe tener exactamente las mismas opciones que la orden -A (o -I, o -R). Si hay varias reglas idénticas en la misma cadena, sólo se borrará la primera.

## 7.3. Especificaciones del filtrado

Hemos usado «-p» para especificar el protocolo, y «-s» para la dirección de origen, pero podemos usar otras opciones para especificar las características de los paquetes. Lo que sigue es un compendio exhaustivo:

### 7.3.1. Especificar las direcciones IP de Origen y Destino

Las direcciones IP de origen («-s», «–source», o «–src») y destino («-d», «–destination», o «–dst») se pueden especificar de cuatro maneras. La más común es usar el nombre completo, tal como «localhost» o «www.linuxhq.com». La segunda manera es especificar la dirección IP, como «127.0.0.1».

Las tercera y cuarta maneras permiten especificar un grupo de direcciones IP, como «199.95.207.0/24» o «199.95.207.0/255.255.255.0». Ambas especifican cualquier dirección entre 199.95.207.0 y 199.95.207.255, ambas inclusive; los dígitos tras la «/» dicen qué partes de la dirección IP son significativas. «/32» o «/255.255.255.255» es la opción por defecto (coincide con toda la dirección IP). Para especificar cualquier dirección IP, se debe usar «/0», de esta manera:

```
[ NOTA: «-s 0/0» es redundante en este caso. ]
# iptables -A INPUT -s 0/0 -j DROP
#
```

Se usa rara vez, ya que el efecto que se consigue en esta regla es lo mismo que si no se especificase la opción «-s».

### 7.3.2. Especificar una inversión

Hay muchos indicadores, como «-s» (o «–source») y «-d» («–destination») cuyos respectivos argumentos pueden ir precedidos por «!» (se pronuncia «not» o «no»), para que coincidan con direcciones que NO sean iguales a las proporcionadas. Por ejemplo, «-s ! localhost» coincide con cualquier paquete que **no** venga de localhost.

### 7.3.3. Especificar el protocolo

Se puede especificar el protocolo con el indicador «-p» (o «–protocol»). El protocolo puede ser un número (si sabe los valores numéricos) o un nombre en el caso especial de «TCP», «UDP» o «ICMP». No importa si lo pone en mayúscula o minúscula; «tcp» valdrá lo mismo que «TCP».

El nombre de protocolo puede ir prefijado de una «!», para invertirlo, de manera que «-p ! TCP» especifica paquetes que **no** sean TCP.

### 7.3.4. Especificar la Interfaz

Las opciones «-i» (o «–in-interface») y «-o» (o «–out-interface») especifican el nombre de una **interfaz** con la que coincidir. Una interfaz es el dispositivo físico por el que entra («-i») o sale («-o») un paquete. Puede usar la orden `ifconfig` para obtener una lista de las interfaces que están «up» (esto es, funcionando en ese momento).

Los paquetes que pasan por la regla `INPUT` no tienen un interfaz de salida, con lo que nunca se activará una regla de esta cadena que use «-o». De forma similiar, los paquetes que atraviesan `OUTPUT` no tienen interfaz de salida, de manera que ninguna regla que use «-i» en esta cadena funcionará.

Sólo los paquetes que pasan por la cadena `FORWARD` tienen a la vez interfaz de entrada y de salida.

Es perfectamente correcto especificar una interfaz que no existe en este momento; la regla no será activada por nada hasta que la interfaz empiece a funcionar. Esto es extremadamente útil con enlaces PPP de llamada (normalmente la interfaz `ppp0`) y similares.

Como caso especial, un nombre de interfaz que acabe en «+» coincidirá con todas las interfaces (que existan en ese momento o no) cuyo nombre empiece de esa manera. Por ejemplo, para especificar una regla que funcione para todas las interfaces PPP, se podría usar la opción `-i ppp+`.

El nombre de la interfaz puede ir precedido por «!» para coincidir con un paquete que **no** vaya por la(s) interfaz/ces especificada(s). does **not** match the specified interface(s).

### 7.3.5. Especificar fragmentos

Algunas veces ocurre que un paquete es demasiado grande para pasar por el cable de un solo golpe. Cuando esto ocurre, el paquete se divide en **fragmentos**, que se envían como varios paquetes. El otro extremo reensambla los fragmentos para reconstruir el paquete entero.

El problema de los fragmentos es que el primero de ellos tiene todos los campos de cabecera (IP + TCP, UDP e ICMP) que hay que examinar, pero los siguiente sólo tienen un subconjunto de la cabecara (IP sin los campos adicionales de protocolo). Por lo tanto, no es posible buscar cabeceras de protocolos (como se hace con las extensiones TCP, UDP e ICMP) en los siguientes fragmentos.

Si está haciendo un seguimiento de conexión o NAT, entonces se reunirán todos los fragmentos antes de que alcancen el código de filtrado de paquetes, de manera que no se deba preocupar por ellos.

Por otro lado, es importante comprender cómo son tratados los fragmentos por las reglas de filtrado. Cualquier regla de filtrado que pida información que no tenemos *no* será activada. Esto significa que el primer fragmento se trata como cualquier otro paquete. El segundo y los siguientes no. Por tanto una regla que diga `-p TCP -sport www` (que especifica como puerto origen el «www») nunca será activada por un fragmento (excepto el primero). Tampoco la regla opuesta `-p TCP -sport ! www`.

Sin embargo, podemos especificar una regla específicamente para el segundo fragmento y los siguientes, usando la opción «-f» (o «–fragment»). También es válido especificar que una regla *no* se va a aplicar al segundo fragmento ni a los siguientes, precediendo «-f» con «!».

Normalmente se considera seguro dejar pasar el segundo fragmento y siguientes, ya que el filtrado afectará al primero, y por lo tanto se evita que el destino de los paquetes pueda reensamblarlos por completo; sin embargo, se conocen fallos de programación que permiten poner fuera de combate máquinas simplemente enviándoles fragmentos. Usted elige.

Nota sobre las cabeceras: los paquetes malformados (los TCP, UDP e ICMP demasiado cortos como para que el código del cortafuegos pueda leer los puertos o el tipo y código de ICMP) son descartados cuando se intenta examinarlos. Lo mismo pasa con los fragmentos TCP que empiezan en la posición 8.

Por ejemplo, la siguiente regla descartará cualquier fragmento dirigido a 192.168.1.1:

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

### 7.3.6. Extensiones a iptables: Nuevas coincidencias

`iptables` es **extensible**, lo que significa que se pueden extender tanto el núcleo como la herramienta iptables para proporcionar nuevas características.

Algunas de estas extensiones son estándar, y otras más exóticas. Las extensiones las puede hacer otra gente y pueden ser distribuidas por separado para diferentes nichos de usuarios.

Las extensiones al núcleo normalmente residen un directorio de módulos, como `/lib/modules/2.3.15/net`. Normalmente el núcleo los carga cuando son necesario, si fue compilado con la opción CONFIG_KMOD, de manear que no sería necesario cargarlos de forma manual.

Las extensiones al programa iptables son bibliotecas compartidas que residen normalmente en /usr/local/lib/iptables/, aunque una distribución podría ponerlas en /lib/iptables o /usr/lib/iptables.

Las extensiones son de dos tipos: nuevos objetivos (targets) y nuevas coincidencias (matches); hablaremos ahora sobre los nuevos objetivos. Algunos protocolos ofrecen de manera automática nuevos tipos de comprobaciones: en la actualidad son TCP, UDP e ICMP, como se muestra más adelante.

Para estos protocolos podrá especificar nuevas comprobaciones en la línea de órdenes tras la opción «-p», lo que cargará las extensiones. Para nuevas comprobaciones explícitas, use la opción «-m» para cargar las extensiones, tras lo cual estarán disponibles.

Para obtener ayuda sobre una extensión, use la opción que se usa para cargarla («-p», «-j» o «-m») seguida por «-h» o «–help», por ejemplo:

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

```
#
```

**Extensiones TCP** Las extensiones TCP se cargan de forma automática si se especifica «-p tcp». Esto proporcionar las siguientes opciones (ninguna de las cuales se activa con fragmentos).

**–tcp-flags**

Seguido por una «!» opcional, y después dos cadenas de indicadores (flags), le permite filtrar dependiendo de ciertos indicadores de TCP. La primera cadena es la máscara: una lista de los indicadores que desea examinar. La segunda cadena indica cuales deben estar activos. Por ejemplo,

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

Esto indica que deben ser examinados todos los indicadores («ALL» es sinónimo de «SYN,ACK,FIN,RST,URG,PSH»), pero sólo deben estar activos SYN y ACK. Hay otro argumento llamado «NONE», que significa «ningún indicador».

**–syn**

precedido de forma opcional por un «!», es equivalente a «–tcp-flags SYN,RST,ACK SYN».

**–source-port**

seguido de forma opcional por un «!», y después un puerto o rango de puertos TCP. Estos pueden estar representados por su nombre, tal como viene en /etc/services, o por número. Los rangos pueden ser dos nombres de puerto separados por un «-», un puerto seguido de un «-» (para especificar un puerto mayor o igual al indicado), o un puerto precedido de «-» (para especificar un puerto menor o igual al indicado).

**–sport**

sinónimo a «–source-port».

**–destination-port**

y

**–dport**

son lo mismo que lo anterior, sólo que especifican el puerto de destino, en lugar del de origen.

**–tcp-option**

seguido por un «!» (opcional) y un número, se ajusta a paquetes con una opción TCP igual a ese número. Un paquete que no tenga una cabecera TCP completa, será descartado automáticamente si se intenta examinar sus opciones TCP.

**Explicación de los indicadores (flags) TCP** A veces es útil permitir conexiones TCP en una dirección, pero no en la otra. Por ejemplo, puede que queira permitir conexiones a un servidor WWW externo, pero no desde ese mismo servidor.

La solución del inexperto sería bloquear los paquetes TCP que vengan del servidor. Desafortunadamente, las conexiones TCP precisan que los paquetes fluyan en ambas direcciones para poder funcionar.

La solución es bloquear sólo los paquetes que se usan para solicitar una conexión. A éstos se les llama paquetes **SYN** (ok, técnicamente son paquetes con el indicador SYN activo, y los FIN y ACK inactivos, pero los llamamos paquetes SYN para abreviar). Rechazando estos paquetes, podemos detener intentos de conexión en su inicio.

El indicador (flag) «–syn» se usa para este propósito: sólo es válido para las reglas que especifican TCP como protocolo. Por ejemplo, para especificar intentos de conexión TCP desde 192.168.1.1:

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

Este indicador puede ser invertido precediéndolo con un «!», lo que significa cualquier paquete excepto el de inicio de conexión.

**Extensiones UDP** Estas extensiones se cargan de forma automática si se especifica «-p udp». Proporciona las opciones «–source-port», «–sport», «–destination-port» y «–dport» con los mismos detalles que los indicados para TCP.

**Extensiones ICMP** Esta extensión se carga de forma automática si se especifica «-p icmp». Proporciona sólo una opción nueva:

**–icmp-type**

seguida de un «!» opcional, y luego del nombre de un tipo icmp (p.ej. «host-unreachable»), un tipo numérico («3»), o un tipo numérico y un código separados por «/» («3/3»). Puede ver una lista de los nombres de los tipos icmp disponibles usando «-p icmp –help».

**Otras extensiones de coincidencia (match)** Las otras extensiones del paquete netfilter son de demostración, que (si son instaladas) pueden ser invocadas con la opción «-m».

**mac**

Este módulo debe ser especificado de forma explícita con «-m mac» o «–match mac». Se usa para coincidencias en las direcciones Ethernet (MAC) de los paquetes entrantes, y por tanto sólo son útiles para los paquetes que pasan por las cadenas PREROUTING e INPUT. Proporciona sólo una opción:

**–mac-source**

seguida de un «!» opciona, y luego una dirección ethernet en notación hexadecimal separada por «:», por ejemplo «–mac-source 00:60:08:91:CC:B7».

**limit**

Este módulo debe ser especificado de forma explícita con «-m limit» o «–match limit». Se usa para restringir la tasa de coincidencias, como por ejemplo para suprimir mensajes de registro. Sólo se activará un número dado de veces por segundo (por defecto, 3 coincidencias por hora, a ráfagas de 5). Tiene dos argumentos opcionales:

**–limit**

seguido por un número; especifica el número máximo de coincidencias de media por segundo a permitir. El número puede especificar unidades de forma explícita, usando «/second», «/minute», «/hour», o «/day», o abreviadas (de manera que «5/second» es lo mismo que «5/s»).

**–limit-burst**

seguido de un número, indica la ráfaga más larga que se puede producir antes de comprobar el límite.

Este tipo de coincidencias se suele usar con el objetivo LOG para producir registros limitados por una tasa. Para comprender cómo funciona esto, veamos la siguiente regla, que registra los paquetes con los parámetros de limitación por defecto:

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

La primera vez que se alcanza esta regla, se registra el paquete; en realidad, como la ráfaga por defecto es de 5, se registrarán los primeros cinco paquetes. Después, pasarán veinte minutos antes de que vuelva a registrarse un paquete con esta regla. Además, cada veinte minutos que pasen sin que un paquete alcance la regla, la ráfaga «recuperará» un paquete; si no sucede nada durante 100 minutos, la ráfaga quedará completamente «recargada»; de vuelta entonces a la situación inicial.

Ahora mismo no se pueden crear reglas con un tiempo de recarga mayor a 59 horas, de manera que si establece una tasa de recarga de un paquete por día, entonces el número de paquetes por ráfaga deberá ser menor que 3.

También puede usar este módulo para evitar varios ataques por denegación de servicio (DoS) con una tasa más rápida para incrementar la respuesta.

Protección contra Syn-flood (inundación mediante Syn):

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

Furtivo buscando puertos (port scanner):

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

Ping de la muerte:

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

Este módulo trabaja como una «puerta de histéresis», tal como se muestra en el gráfico siguiente.

```
           tasa (pkt/s)
                 ^        .---.
                 |       / DoS \
                 |      /       \
Inicio del DoS-  |.....:.........\.......................
 = (limit *      |    /:          \
limit-burst)     |   / :           \         .-.
                 |  /  :            \       /   \
                 | /   :             \     /     \
Fin del DoS     -|/....:..............:.../.......\..../.
 = limit         |     :              :`-'         `--'
-----------------+-----+--------------+------------------> tiempo (s)
   LOGICA =>     Activa|   No activa  |    Activa
```

Digamos que las coincidencias son de un paquete por segundo, con una ráfaga de cinco paquetes, pero los paquetes comienzan a llegar a un ritmo de cuatro por segundo, durante tres segundos, y luego empezamos de nuevo con otros tres segundos.

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

Paquetes^                 Máxima  __--      YNNN
Totales |               Tasa  __--      YNNN
        |             de  __--      YNNN
     10 |       Línea __--         Y
        |         __--            Y
        |     __--               Y
        | __--    YNNN
        |-    YNNN
      5 |    Y
        |   Y                           Leyenda:  Y -> Coincidencia
        |  Y                                      N -> No coincide
        | Y
        |Y
      0 +--------------------------------------------------> Tiempo (seg.)
         0   1   2   3   4   5   6   7   8   9  10  11  12
```

Podemos ver que los primeros cinco paquetes tienen permiso para exceder la tasa de uno por segundo, y entonces entra en juego el límite. Si hay una pausa, entonces se permite otra ráfaga, pero no más allá de la tasa máxima establecida por la regla (un paquete por segundo después de «gastar» la ráfaga).

**owner**

Este módulo intenta ajustarse a varias características del creador del paquete, para aquellos generados de forma local. Sólo es válida para la cadena OUTPUT, e incluso algunos paquetes (como las respuestas a los paquetes ping) pueden no tener dueño, y por lo tanto nunca activar la regla.

**–uid-owner idusuario**

Coincide si el paquete fue creado por un proceso con el id efectivo de usuario dado (numérico).

**–uid-owner idgrupo**

Coincide si el paquete fue creado por un proceso con el id efectivo de grupo dado (numérico).

**–pid-owner idproceso**

Coincide si el paquete fue creado por un proceso con el id efectivo de proceso dado.

**–sid-owner idproceso**

Coincide si el paquete fue creado por un proceso perteneciente a la sesión de grupo dada.

**unclean**

Este módulo experimental debe ser especificado de forma explícita con «-m unclean» o «–match unclean». Hace varios controles de corrección sobre los paquetes. Este módulo no ha pasado una auditoría, y no debería ser usado como dispositivo de seguridad (posiblemente empeore las cosas, ya que podría tener fallos). No proporciona opciones.

**La coincidencia (match) State** El criterio de coincidencia más útil vienen proporcionado por la extensión «state», que interpreta el análisis de seguimiento de conexión del módulo «ip_conntrack». Lo recomiendo encarecidamente.

Especificar «-m state» permite una opción «–state» adicional, que es una lista separada de estados a buscar (el indicador «!» especifica que **no** se desea buscar esos estados). Los estados son:

**NEW**
: Paquete que crea una nueva conexión.

**ESTABLISHED**
: Paquete que pertenece a una conexión existente (esto es, que tuvo paquetes de respuesta).

**RELATED**
: Paquete que está relacionado a una conexión existente, pero que no es parte de ella, como un error ICMP o (con el módulo de FTP insertado), un paquete que establece una conexión de datos ftp.

**INVALID**
: Paquete que no pudo ser identificado por alguna razón: incluye quedarse sin memoria y errores ICMP que no corresponden a ninguna conexión conocida. Normalmente estos paquetes deberían ser descartados.

## 7.4. Especificación de objetivos

Ahora que sabemos qué exámenes podemos hacer a un paquete, necesitamos una manera de decir qué hacer con los paquetes que se ajustan a nuestras pruebas. A esto se le denomina **objetivo** (target) de una regla.

Hay dos objetivos implementados en el sistema bastante sencillos: DROP y ACCEPT. Ya los hemos visto ambos. Si una regla coincide con un paquete, y su objetivo es alguno de estos dos, no se consultarán más reglas: la suerte del paquete ha sido decidida.

Hay dos tipos de objetivos aparte de los que ya vienen: extensiones y cadenas definidas por el usuario.

### 7.4.1. Cadenas definidas por el usuario

Una característica potente que `iptables` hereda de `ipchains` es la capacidad del usuario de crear nuevas cadenas, aparte de las tres que ya vienen implementadas (INPUT, FORWARD y OUTPUT). Por concención, las cadenas definidas por el usuario van en minúsculas para distinguirlas (describiremos cómo crear nuevas cadenas luego en 7.5 (Operaciones con una cadena entera)).

Cuando un paquete coincide con una regla cuyo objetivo es una cadena definida por el usuario, el paquete empieza a atravesar esa otra cadena. Si en ella no se decide la suerte del paquete, una vez llegado el final de la cadena, se vuelve al mismo punto desde el que se saltó, a la siguiente regla de la antigua cadena.

Es el momento de un poco más de «ASCII art». Consideremos dos cadenas (bastante tontas): `INPUT` (la cadena interna) y `prueba` (una cadena definida por el usuario).

```
 `INPUT'                          `prueba'
 ----------------------------    ----------------------------
 | Regla1: -p ICMP -j DROP  |    | Regla1: -s 192.168.1.1   |
 |--------------------------|    |--------------------------|
 | Regla2: -p TCP -j prueba |    | Regla2: -d 192.168.1.1   |
 |--------------------------|    ----------------------------
```

```
| Regla3: -p UDP -j DROP    |
-----------------------------
```

Consideremos un paquete TCP que venga de 192.168.1.1, y vaya a 1.2.3.4. Entra en la cadena `INPUT`, y pasa se compara con Regla1 (no coincide). Coincide con Regla2, y su objetivo es `prueba`, de manera que la siguiente regla que se examina es el comienzo de `prueba`. La Regla1 de `prueba` coincide, pero no especifica un objetivo, de manera que se examina la siguiente regla, Regla2. No coincide, de manera que hemos llegado al final de la cadena. Volvemos a `INPUT`, donde acabábamos de examinar Regla2, de manera que comprobamos Regla3, que tampoco coincide.

De manera que el camino que hace el paquete es:

```
                        v    __________________________
 `INPUT'                |   /    `prueba'             v
------------------------|--/    -----------------------|----
| Regla1                | /|    | Regla1               |   |
|-----------------------|/-|    |----------------------|---|
| Regla2                /  |    | Regla2               |   |
|--------------------------|    -----------------------v----
| Regla3                /--+___________________________/
------------------------|---
                        v
```

Las cadenas definidas por el usuario pueden saltar a otras (pero no haga bucles, los paquetes serán descartados si se detecta que están en bucle).

### 7.4.2. Extensiones a iptables: Nuevos objetivos

El otro tipo de objetivo es una extensión. Una extensión de objetivo consiste en un módulo del núcleo, una extensión opcional a `iptables` para proporcionar nuevas opciones en línea de órdenes: Hay varias extensiones en la distribución de netfilter básica:

**LOG**

Este módulo proporciona un registro de paquetes coincidentes mediante el núcleo. Dispone de estas opciones adicionales:

**–log-level**

Seguido de un número (o nombre) de nivel. Los nombres válidos son (distingue mayúsculas/minúsculas) «debug», «info», «notice», «warning», «err», «crit», «alert», «emerg», que corresponden a los números 7 a 0. Vea la página de manual de syslog.conf si necesita una explicación de estos niveles.

**–log-prefix**

Seguido de una cadena de hasta 30 caracteres, que corresponden a un texto que se escribe al comienzo de cada «log» (registro), para permitir que sean identificados unívocamente.

Este módulo es más útil junto a un objetivo «limit», de manera que no se inunden los archivos de registro.

**REJECT**

Este módulo tiene el mismo efecto que «DROP», excepto que al remitente se le envía un mensaje de error ICMP «port unreachable». Fíjese que el mensaje de error ICMP no se envía si (vea el RFC 1122):

- El paquete a filtrar ya era un mensaje de error ICMP, o algún tipo desconocido de ICMP.
- El paquete a filtrar no era un fragmento con la cabecera del paquete original.
- Hemos enviado demasiados mensajes de error ICMP a ese destino recientemente.

REJECT también admite un argumento opcional «–reject-with» que altera el paquete re respuesta a usar: vea la página de manual.

### 7.4.3. Objetivos especiales de serie

Hay dos objetivos especiales que vienen de serie: `RETURN` y `QUEUE`.

`RETURN` tiene el mismo efecto que si hubiésemos llegado al final de la cadena: si la regla estaba en una cadena de las que hay por defecto, entonces se ejecuta la política de la cadena. Para una regla que esté en una cadena definida por el usuario, se sale de ella, y se continúa atravesando la cadena anterior al salto, justo después de la regla con la que se saltó.

`QUEUE` es un objetivo especial, que pasa al paquete en una cola destinada al procesamiento en espacio de usuario. Para que esto sea útil, hacen falta otros dos componentes:

- un «controlador de cola» (*queue handler*), que lleva la mecánica real de paso de paquetes entre el núcleo y el espacio de usuario; y
- un proceso en espacio de usuario que recibe, posiblemente manipula, y dicta veredicto sobre los paquetes.

El controlador de cola estándar de iptables para IPv4 es el módulo ip_queue, que se distribuye con el núcleo y está marcado como experimental.

El que sigue es un ejemplo rápido de cómo usar iptables para poner en cola los paquetes para su proceso en espacio de usuario:

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

Con esta regla, los paquetes ICMP generados de forma local (como los que crea, digamos, ping), se pasan al módulo ip_queue, que entonces intenta pasarlos a una aplicación en espacio de usuario. Si no hay una aplicación esperando por ellos, entonces se descartan.

Para escribir una aplicación en espacio de usuario, utilice el API libipq. Se distribuye con iptables. Encontrará código de ejemplo en el conjunto de herramientas de pruebas (como redirect.c) en el CVS.

Se puede comprobar el estado de ip_queue mediante:

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

Se puede controlar la longitud máxima de la cola (número de paquetes entregados al espacio de usuario sin dictar veredicto) mediante:

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

El valor por defecto para la longitud máxima de la cola es 1024. Una vez se alcance este límite, los paquetes nuevos serán descartados hasta que la longitud de la cola caiga por debajo del límite de nuevo. Los protocolos buenos como TCP interpretan a partir de los paquetes descartados que hay congestión en la red, y deberían reintentar cuando la cola se llene. Sin embargo, habrá que experimentar un poco para determinar una longitud máxima ideal para la cola para una situación dada si el valor por defecto es demasiado pequeño.

## 7.5. Operaciones con una cadena entera

Una característica bastante útil de `iptables` es la capacidad de agrupar reglas relacionadas en una cadena. Podemos poner el nombre que queramos a las cadenas, pero yo recomiendo usar letras minúsculas para evitar confundirlas con las cadenas y objetivos proporcionados por defecto. Los nombres de las cadenas pueden tener hasta 31 letras.

### 7.5.1. Crear una nueva cadena

Creemos una nueva cadena. Como soy una persona tan imaginativa, la llamaré `prueba`. Usaremos las opciones «-N» o «–new-chain»:

```
# iptables -N prueba
#
```

Es así de simple. Ahora podemos poner reglas en ella tal como se detalló anteriormente.

### 7.5.2. Borrar una cadena

Borrar una cadena es igualmente simple, y se utilizan las opciones «-X» o «–delete-chain». ¿Por qué «-X»? Bueno, todas las letras buenas estaban cogidas ya.

```
# iptables -X prueba
#
```

Hay un par de restricciones al borrar cadenas: deben estar vacías (vea 7.5.3 (Vaciar una cadena) más adelante) y no pueden ser el objetivo de ninguna regla. Tampoco puede borrar ninguna de las tres cadenas del sistema.

Si no especifica una cadena, entonces serán borradas *todas* las cadenas definidas por el usuario, si es posible.

### 7.5.3. Vaciar una cadena

Hay una manera sencilla de eliminar todas las reglas de una cadena, usando la orden «-F» (o «–flush»).

```
# iptables -F forward
#
```

Si no especifica una cadena, *todas* serán vaciadas.

#### 7.5.4. Listar el contenido de una cadena

Puede listar todas las reglas de una cadena usando la orden «-L» (o «–list»).

El «refcnt» mostrado para cada cadena definida por el usuario es el número de reglas que tienen esa cadena como objetivo. Deber ser cero (y la cadena estar vacía) antes de que la cadena pueda ser borrada.

Si se omite el nombre de la cadena, se muestran todas, incluso las vacías.

Hay tres opciones que pueden acompañar a «-L». La opción «-n» (numérico) es útil ya que evita que `iptables` intente averiguar el nombre de las direcciones IP, lo que (si está usando DNS como la mayoría de la gente) puede causar grandes retrasos si el DNS no está configurado adecuadamente, o está filtrando las consultas DNS. También hace que los puertos TCP y UDP aparezcan listados por número, en lugar de por nombre.

Las opciones «-v» muestran todos los detalles de las reglas, tales como los contadores de paquetes y bytes, las comparaciones TOS, y las interfaces. En cualquier otro caso, se omitirán estos valores.

Fíjese que los contadores de paquetes y bytes se muestran usando los sufijos «K», «M» o «G», indicando 1000, 1.000.000 y 1.000.000.000 respectivamente. Use el indicador «-x» (expandir números) para mostrar los números completos, sin importar lo grande que sean.

#### 7.5.5. Restablecer (poner a cero) los contadores

Es útil poder poner a cero los contadores. Esto se puede hacer con la opción «-Z» (o «–zero»).

El problema de esta solución es que a veces necesitamos saber el valor del contador inmediatamente antes de ser restablecidos. En el ejemplo anterior, podría ser que algunos paquetes pasasen entre las órdenes «-L» y «-Z». Por esta razón, puede usar «-L» y «-Z» *juntas*, para poner a cero los contadores, al mismo tiempo que lee su contenido.

#### 7.5.6. Establecer la política

Comentamos lo que sucede cuando un paquete llega al final de una cadena del sistema cuando discutimos antes cómo atraviesa el paquete las cadenas. En este caso, la **política** de la cadena determina la suerte del paquete. Sólo las cadenas del sistema (`INPUT`, `OUTPUT` y `FORWARD`) tienen políticas, porque si un paquete llega al final de una cadena definida por el usuario, se vuelve a la cadena anterior.

La política puede ser `ACCEPT` o `DROP`.

## 8. Uso de ipchains e ipfwdam

Hay módulos en la distribución de netfilter llamados `ipchains.o` e `ipfwadm.o`. Inserte uno de ellos en el núcleo (NOTA: ¡son incompatibles con `iptables.o`, `ip_conntrack.o` e `ip_nat.o`!). Entonces podrá usar ipchains o ipfwadm tal como en los buenos viejos tiempos.

Esta implementación durará un tiempo. Creo que 2 * [aviso de reemplazo - distribución estable inicial] es una fórmula razonable, antes de eliminarla tras la fecha en que esté disponible una versión estable de su reemplazo.

Esto significa que, para ipfwadm, el fin del soporte será (FIXME: poner la fecha real):

```
2 * [Octubre 1997 (versión 2.1.102) - Marzo 1995 (ipfwadm 1.0)]
        + Enero 1999 (versión 2.2.0)
    = Noviembre 2003.
```

Esto significa que con ipchains, el fin del soporte será (FIXME: poner las fechas reales):

```
2 * [Agosto 1999 (versión 2.3.15) - Octubre 1997 (versión 2.2.0)]
        + Enero 2000 (¿versión 2.3.0?)
    = Septiembre 2003.
```

Por lo tanto no tendrá que preocuparse hasta el 2004.

# 9. Mezclar NAT y el filtrado de paquetes

Es normal que desee hacer Network Address Translation (vea el NAT COMO) y filtrado de paquetes. Las buenas noticias es que se pueden mezclar extremadamente bien.

Puede diseñar su filtrado de paquetes ignorando completamente lo que haga con NAT. Los orígenes y destinos vistos por el filtrado de paquetes serán los «reales». Por ejemplo, si está haciendo DNAT para enviar cualquier conexión al puerto 1.2.3.4 puerto 80 hacia 10.1.1.1 puerto 8080, el filtro verá que el paquete va a 10.1.1.1 puerto 8080 (el destino real), no 1.2.3.4 puerto 80. De forma similar, puede ignorar el enmascaramiento: los paquetes parecerán venir de la dirección IP interna (digamos 10.1.1.1), y las respuestas aparecerán como que van hacia allí.

Puede usar la extensión de coincidencias «state» sin que el filtro de paquetes tenga que hacer trabajo extra, ya que NAT precisa de seguimiento de conexiones de todas formas. Para mejorar el ejemplo de enmascaramiento sencillo del NAT COMO de manera que evite nuevas conexiones que vengan de la interfaz ppp0, podría hacer esto:

```
# Enmascarar ppp0
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

# Prohibir paquetes NEW e INVALID que vengan de o sean redirigidos
# por ppp0
iptables -A INPUT -i ppp0 -m state --state NEW,INVALID -j DROP
iptables -A FORWARD -i ppp0 0 -m state --state NEW,INVALID -j DROP

# Activar el reenvío IP (IP Forwarding)
echo 1 > /proc/sys/net/ipv4/ip_forward
```

# 10. Diferencias entre iptables e ipchains

- Primero, los nombres de las cadenas del sistema han cambiado de minúsculas a MAYUSCULAS, porque las cadenas INPUT y OUTPUT ahora sólo aceptan paquetes destinados a la máquina y generados por la máquina. Solían atrapar todos los paquetes entrantes y salientes, respectivamente.
- El indicador «-i» ahora especifica la interfaz de entrada, y sólo funciona en las cadenas INPUT y FORWARD. Las reglas de las cadenas FORWARD y OUTPUT que usaban «-i» ahora deberán ser cambiadas por «-o».

- Los puertos TCP y UDP ahora deben ser indicados con las opciones –source-port o –sport (o–destination-port/–dport) , y deben ir después de las opciones «-p tcp» o «-b udp», que cargan respectivamente las extensiones TCP o UDP.
- El indicador TCP -y ahora es –syn, y debe ir tras «-p tcp».
- El objetivo DENY es ahora, por fin, DROP.
- Se puede poner a cero una cadena mientras se lista.
- Poner a cero una cadena interna también borra los contadores de política.
- Listar cadenas le da los contadores en forma de captura atómica.
- REJECT y LOG son ahora objetivos extendidos, lo que significa que van en módulos aparte.
- Los nombres de cadenas tienen ahora hasta 31 caracteres.
- MASQ es ahora MASQUERADE y usa una sintaxis diferente. REDIRECT, aunque conserva el nombre, también ha cambiado de sintaxis. Vea el NAT COMO si desea más información sobre cómo configurarlos.
- La opción -o ya no se utiliza para dirigir los paquetes al dispositivos de espacio de usuario (vea -i). Los paquetes se envían ahora al espacio de usuario mediante el objetivo QUEUE.
- Probablemente hay montón de cosas más que me estoy olvidando.

## 11. Consejos sobre el diseño del filtrado de paquetes

El sentido común en el campo de la seguridad informática es bloquearlo todo, y después abrir agujeros donde sea necesario. Normalmente se dice que «aquello que no está explícitamente permitido, está prohibido». Recomiendo este enfoque si la seguridad es de su máximo interés.

No ejecute servicios que no necesite, incluso si cree que ha bloqueado el acceso a ellos.

Si está creando un cortafuegos dedicado, empiece por no ejecutar nada, y bloquear todos los paquetes, para después añadir servicios y dejar pasar los paquetes según sea necesario.

Recomiendo asegurar en profundidad: combinar tcp-wrappers (para conexiónes con el propio filtro de paquetes), proxies (para conexiones que pasen a través del filtro de paquetes), verificación de rutas y filtro de paquetes. Es durante la verificación de rutas cuando es descartado un paquete que viene de una interfaz inesperada: por ejemplo, si su red interna tiene direcciones 10.1.1.0/24, y un paquete con esa dirección de origen llega por la interfaz externa, será descartado. Esto se puede activar para una interfaz (ppp0) de esta manera:

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

O para todas las interfaces existentes y futuras con:

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

Debian hace esto por defecto donde sea posible. Si tiene un encaminamiento asimétrico (espera que los paquetes lleguen de direcciones extrañas), puede que desee desactivar este filtro en esas interfaces.

Hacer un registro es útil cuando se está configurando un cortafuegos si algo no está funcionando, pero en un cortafuegos en producción, combínelo siempre con la coincidencia «limit», para evitar que puedan sobrecargar sus registros.

Recomiendo sobre todo el seguimiento de conexiones para los sistemas seguros: crea algo de sobrecarga, ya que se siguen todas las conexiones, pero es muy útil para controlar el acceso a sus redes. Puede que necesite cargar el módulo «ip_conntrack.o» si su núcleo no carga los módulos de forma automática, y no está ya incluido en el núcleo. Si quiere hacer un seguimiento preciso a protocolos complejos, necesitará cargar el módulo de ayuda apropiado (p.ej: «ip_conntrack_ftp.o»).

```
# iptables -N no-conns-from-ppp0
# iptables -A no-conns-from-ppp0 -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A no-conns-from-ppp0 -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A no-conns-from-ppp0 -i ppp0 -m limit -j LOG --log-prefix "Paquete malo de ppp0:"
# iptables -A no-conns-from-ppp0 -i ! ppp0 -m limit -j LOG --log-prefix "Paquete malo que no viene de ppp0:"
# iptables -A no-conns-from-ppp0 -j DROP

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

La construcción de un buen cortafuegos está más allá del propósito de este COMO, pero mi consejo es «sea siempre minimalista». Vea el Seguridad COMO si desea más información sobre cómo probar su máquina.