Tutorial de IPtables 1.1.19es

Oskar Andreasson

blueflux@koffein.net

Tutorial de IPtables 1.1.19es
by Oskar Andreasson

Dedicatorias

Por encima de todo quiero dedicar este documento a mi maravillosa novia Ninel. Nunca podré apoyarte tanto como tú me has apoyado a mí. Deseo hacerte al menos tan feliz como tú me has hecho a mí.

También quiero dedicar este trabajo a todos los programadores y encargados del mantenimiento de Linux por su trabajo tan increíblemente duro. Gracias a éllos se hace posible éste maravilloso sistema operativo.

Table of Contents

List of Tables

Notas de la traducción

Quiero expresar mi agradecimiento a Oskar Andreasson por este tutorial. Gracias a él todos aquellos que deseamos conocer los secretos de IPtables y por ende los cortafuegos, tenemos una buena guía para empezar y adquirir un nivel bastante aceptable. Asímismo quiero dar las gracias a mis compañeros de traducción, por la paciencia y tesón demostrados. Por último, me atrevo a pedir a todo aquél que lea esta traducción que sea condescendiente con nosotros: hemos traducido el tutorial de la mejor manera que hemos sabido, pero pueden haber fallos y agradeceríamos vuestras sugerencias y correcciones.

Xavier Bartol (mailto:webdev@arrakis.es)

Acerca del autor

Soy alguien con demasiados ordenadores viejos en sus manos. Tengo mi propia red de área local (*LAN*) y deseo que todas las máquinas puedan conectarse a Internet, al tiempo que la red interna permanece suficientemente segura. En este sentido, iptables es una buena mejora del antiguo ipchains. Con ipchains puedes construir una red medianamente segura desechando cualquier paquete entrante (de Internet) no destinado a unos puertos determinados. Sin embargo hay temas, como el *FTP* pasivo, o *DCC* saliente en *IRC*, que plantean problemas. Estos servicios asignan puertos en el servidor, informan al cliente sobre ellos y entonces esperan a que el cliente conecte. Al principio tuve algunos problemas con el código de iptables y de alguna manera pensé que no estaba listo para publicarlo y utilizarlo en entornos productivos. Sin embargo, ahora recomendaría actualizarse a cualquiera que esté utilizando ipchains (o incluso el antiguo ipfwadm), a menos que sea feliz con los resultados que está obteniendo y no necesite ninguna de las características adicionales que ofrece iptables.

Cómo leer este tutorial

Este documento se ha escrito únicamente para empezar a conocer el maravilloso mundo de iptables. Nunca se planteó para albergar información sobre fallos de seguridad específicos en iptables o Netfilter. Si encuentras fallos o comportamientos extraños en iptables o cualquiera de sus componentes, debes contactar con la lista de correo de Netfilter y comentarles el problema. Éllos podrán decirte si en realidad es un fallo o si ya está corregido. Aunque son muy raros los fallos de seguridad encontrados en iptables o en Netfilter, todavía se cuelan uno o dos de vez en cuando y se detallan convenientemente en la página principal de Netfilter (http://www.netfilter.org), que es donde debes ir para encontrar información sobre estos temas.

Lo anterior también implica que los conjuntos de reglas (rule-sets) disponibles en este tutorial no se han escrito para remediar fallos en Netfilter. Su finalidad principal es mostrar cómo crear reglas fácilmente que solucionen todos los problemas con que nos podamos encontrar. Por ejemplo, este tutorial no explica cómo cerrar el puerto HTTP simplemente porque el servidor web Apache sea vulnerable en su versión 1.2.12 (en realidad veremos cómo cerrarlo, pero no por este motivo).

Este documento se ha escrito simplemente para ofrecer un manual bueno y sencillo sobre cómo empezar con iptables, pero siendo al mismo tiempo lo más completo posible. No contiene ningún objetivo o regla de concordancia que estén en el "patch-o-matic" (la sección de la web de Netfilter que se encarga de ofrecer los últimos desarrollos aplicables al núcleo de iptables), simplemente porque requeriría demasiado esfuerzo mantener actualizada una lista así. Si necesitas información sobre estas actualizaciones, deberías leer (y entender al 100%) los comentarios que acompañan a cada parche y cualquier otro documento disponible en la web de Netfilter (http://www.netfilter.org).

Requisitos previos

Este documento requiere algunos conocimientos sobre Linux/Unix, programación en consola (shell scripting), algunos conocimientos básicos sobre las interioridades del núcleo, así como saber compilar tu propio núcleo.

He intentado por todos los medios eliminar todos los requisitos previos para aprovechar al máximo este documento, pero ciertamente es casi imposible no necesitar ciertos conocimientos previos.

Convenciones tipográficas

Para hacer referencia a comandos, archivos y otras informaciones específicas se emplean las siguientes convenciones tipográficas.

- Los ejemplos de código y los resultados de los comandos (las salidas en pantalla) se muestran con tipografía de

  ancho fijo

  (tipo Courier), mientras que los comandos escritos por el usuario se muestran en

  ancho fijo y negrita

  :

  ```
  [blueflux@work1 neigh]$ ls
  default  eth0  lo
  [blueflux@work1 neigh]$
  ```

- Todos los comandos y nombres de programa se muestran en **negrita**.
- Todo aquello referente al sistema como el hardware (la máquina en sí), las interioridades del núcleo (kernel) o entes abstractos como la interfaz de bucle local (loopback interface) se muestran en *cursiva*.
- La respuesta del ordenador ***se muestra así*** en el texto.
- Los archivos y las rutas de acceso se muestran así: /usr/local/bin/iptables.

Chapter 1. Introducción

Por qué se ha escrito este documento

Bueno, encontraba un gran vacío en los CÓMOs (HOWTO's) existentes, en los cuales faltaba informacion sobre las funciones "iptables" y "Netfilter" del nuevo kernel 2.4.x de Linux. Entre otras cosas, voy a intentar responder las preguntas que algunos podéis tener sobre las nuevas posibilidades, como la comparación por estado (state matching). Muchas de estas cuestiones se ilustrarán con un ejemplo rc.firewall.txt (http://iptables-tutorial.frozentux.net/scripts/rc.firewall.txt) que podéis usar en vuestros scripts /etc/rc.d/. [Sí, este archivo se basa en el COMO sobre enmascaramiento (masquerading), para aquellos que lo reconozcan].

Además hay un pequeño guión que he escrito por si te lías tanto como yo durante la configuración. Está disponible como rc.flush-iptables.txt (http://iptables-tutorial.frozentux.net/scripts/rc.flush-iptables.txt).

Cómo se ha escrito

He consultado a Marc Boucher y otras personas del equipo del núcleo de NetFilter. Les agradezco de todo corazón su trabajo y su ayuda en este tutorial, que escribí para boingworld.com y ahora lo mantengo en mi propio sitio web frozentux.net. Este documento os guiará paso a paso a través del proceso de configuración y espero que os ayude a comprender algo más acerca del paquete iptables. Basaré la mayor parte del estudio desarrollado aquí en el ejemplo del archivo rc.firewall, ya que encuentro en ese ejemplo una buena vía para aprender a usar iptables. He decidido estudiar sólamente las cadenas básicas y desde ahí adentrarme en todas y cada una de las cadenas atravesadas en su orden correspondiente. De esta forma el tutorial es un poquito mas complicado de seguir, aunque también es la forma más lógica. Siempre que encuentres algo difícil de comprender, simplemente vuelve a este tutorial.

Términos empleados en este documento

Este documento contiene algunos términos que pueden necesitar explicaciones más detalladas antes que los leas. Esta sección intentará cubrir los más obvios y cómo los he escogido para usarlos en el documento.

DNAT - Destination Network Address Translation (Traducción de la Dirección de Red de Destino). DNAT se refiere a la técnica de traducir la dirección IP de destino de los paquetes, o sea, de cambiarla. Se emplea conjuntamente con SNAT para permitir a varios hosts (máquinas) compartir una sola dirección IP de conexión con Internet entre todos ellos y al mismo tiempo que todavía puedan seguir ofreciendo funciones de servidor. Normalmente ésto se consigue asignando diferentes puertos a la dirección IP externa (la dirección enrutable) y a continuación informando al encaminador/enrutador (router) de Linux dónde debe enviar el tráfico.

SNAT - Source Network Address Translation (Traducción de la Dirección de Red de Origen). Se refiere a las técnicas empleadas para traducir la dirección de origen de un paquete en otra distinta. Como en DNAT, se emplea para que varios hosts puedan compartir una misma dirección IP de salida a Internet. Estas técnicas se emplean debido a que cada vez hay menos direcciones de Internet disponibles a causa de la propia estructura de IPv4 (la versión 6 del protocolo IP, IPv6, solucionará este problema).

Flujo (de datos) (Stream) - Este término se refiere a una conexion que envía y recibe paquetes relacionados entre sí de algún modo. Básicamente, he usado este término para cualquier conexión que envíe 2 ó más paquetes en ambas direcciones. En TCP ésto puede corresponder a una conexión que envía un paquete SYN y entonces responde con un paquete SYN/ACK; pero también puede ser equivalente a una conexión que envía un paquete SYN y responde con un mensaje ICMP de servidor (host) inalcanzable. En otras palabras, uso este término muy libremente.

Estado (State) - Este término se refiere a en qué estado se encuentra el paquete, de acuerdo con el estándar RFC 793 - Transmission Control Protocol (http://iptables-tutorial.frozentux.net/other/rfc793.txt) o bien con los estados de usuario empleados en Netfilter/iptables. Ten en cuenta que los estados usados interna y externamente no siguen totalmente la especificación RFC 793. La razón principal es que Netfilter tiene hacer varias suposiciones acerca de las conexiones y los paquetes.

Espacio de usuario (User space) - Con este término me refiero a cualquier cosa que tenga lugar fuera del núcleo. Por ejemplo, la ejecución de **iptables -h** tiene lugar fuera del núcleo, mientras que **iptables -A FORWARD -p tcp -j ACCEPT** tiene lugar (parcialmente) dentro del núcleo, puesto que se añade una nueva regla al conjunto de reglas.

Espacio del núcleo (Kernel space) - Esto es más o menos lo opuesto al espacio de usuario. Ésto implica las acciones que tienen lugar dentro del núcleo y no fuera de él.

Chapter 2. Preparativos

El objetivo de este capítulo es iniciarte y ayudarte a entender el papel que hoy día tienen Netfilter e **iptables** dentro de Linux. Este capítulo debería conseguir que estés listo para experimentar e instalar tu cortafuegos. Dándole el tiempo necesario y con la perseverancia adecuada, conseguirás que funcione exactamente como desees que lo haga.

Dónde conseguir iptables

El paquete de espacio de usuario de **iptables** se puede descargar desde la página de inicio de Netfilter (http://netfilter.samba.org/). El paquete **iptables** también utiliza las capacidades del espacio del núcleo, las cuales pueden configurarse durante la ejecución de **make configure**. Los pasos necesarios se discutirán a continuación.

Configuración del núcleo (kernel)

Para ejecutar lo más básico de **iptables** tienes que configurar las siguientes opciones en el núcleo mientras ejecutas **make config** o uno de sus comandos relacionados:

CONFIG_PACKET - Esta opción permite que las aplicaciones y las utilidades que lo necesiten puedan trabajar directamente con distintos periféricos de red. Ejemplos de estas utilidades son tcpdump o snort.

Note

En sentido estricto, CONFIG_PACKET no es necesario para que iptables funcione, pero puesto que tiene tantos usos diferentes, he decidido incluirlo. Si crees que no lo necesitas, no lo incluyas.

CONFIG_NETFILTER - Esta opción se requiere cuando vas a utilizar tu ordenador como cortafuegos o como puerta de enlace (gateway) con Internet. En otras palabras, es imprescindible para que funcione cualquier cosa de las que se explican en este tutorial. Entiendo que éso es lo que deseas, ya que estás leyendo el tutorial.

Y, por supuesto, necesitas añadir los controladores (drivers) necesarios para que tus interfases funcionen correctamente, es decir, el adaptador Ethernet y las interfases *PPP* y *SLIP*. Todo lo anterior sólo añade un poco de lo más básico de iptables. En realidad no serás capaz de hacer nada realmente productivo, ya que sólo añade la estructura básica al núcleo. Si quieres utilizar las opciones más avanzadas de iptables, tendrás que configurar las opciones necesarias en el núcleo. A continuación te mostraré las opciones disponibles en una versión 2.4.9 básica del núcleo y las explicaré brevemente:

CONFIG_IP_NF_CONNTRACK - Este módulo es necesario para efectuar el seguimiento de las conexiones. El seguimiento de las conexiones lo emplean, entre otros, la traducción de direcciones (*NAT*) y el enmascaramiento (*Masquerading*). Si necesitas proteger con un cortafuegos las máquinas de una red local, definitivamente debes marcar esta opción. Por ejemplo, este módulo lo necesita el script *rc.firewall.txt* para funcionar.

CONFIG_IP_NF_FTP - Este módulo es necesario si quieres hacer seguimiento de conexiones en las conexiones *FTP*. Puesto que estas conexiones son bastante difíciles de monitorizar en condiciones normales, el conntrack necesita lo que se denomina un asistente y esta opción lo compila en el núcleo. Si no añades este módulo no serás capaz de hacer transferencias FTP correctamente a través del cortafuegos o la puerta de enlace.

CONFIG_IP_NF_IPTABLES - Esta opción es necesaria si quieres realizar algún tipo de filtrado, enmascaramiento (*masquerading*) o traducción de direcciones (*NAT*). Añade toda la estructura de identificación de iptables al núcleo. Sin ésto, no serás capaz de hacer nada en absoluto con iptables.

CONFIG_IP_NF_MATCH_LIMIT - Este módulo no es imprescindible, pero se emplea en el ejemplo *rc.firewall.txt*. Esta opción añade la comparación LIMIT (límite), ofreciendo la posibilidad de controlar el número de paquetes por minuto que se deben comparar, gobernado por la regla adecuada. Por ejemplo, con **-m limit --limit 3/minute** compararíamos un máximo de 3 paquetes por minuto. Mediante este módulo también podemos evitar ciertos ataques de denegación de servicios (en inglés: Denial of Service attacks, DoS attacks).

CONFIG_IP_NF_MATCH_MAC - Este módulo nos permite comparar paquetes basándonos en las direcciones físicas *MAC*: cada adaptador de red Ethernet tiene su propia dirección *MAC*, distinta a la de cualquier otro adaptador, aunque sea de la misma marca y modelo. Así, por ejemplo podremos bloquear paquetes en función de la dirección *MAC* utilizada y bloquear ordenadores concretos puesto que la dirección *MAC* de esos ordenadores raramente cambia (ya que raramente se sustituye el adaptador Ethernet por uno nuevo). No se utiliza esta opción ni en el ejemplo *rc.firewall.txt* ni en ningún otro sitio.

CONFIG_IP_NF_MATCH_MARK - Nos permite utilizar la comparación **MARK**. Por ejemplo, podemos utilizar el objetivo **MARK** para marcar paquetes, de forma que más adelante se puedan comparar y filtrar paquetes dependiendo de si tienen la marca o no. De hecho esta opción es la comparación **MARK** y más adelante veremos el objetivo **MARK**.

CONFIG_IP_NF_MATCH_MULTIPORT - Este módulo permite que comparemos paquetes con un amplio rango de puertos de origen o de destino. Normalmente ésto no sería posible, pero con este módulo sí lo es.

CONFIG_IP_NF_MATCH_TOS - Con esta comparación podemos comparar paquetes en base a su campo *TOS*, es decir, su Tipo de Servicio (*Type Of Service*). El tipo de servicio se puede establecer mediante determinadas reglas en la tabla *mangle* y mediante los comandos ip/tc.

CONFIG_IP_NF_MATCH_TCPMSS - Esta opción nos ofrece la posibilidad de comparar los paquetes *TCP* en función de su campo *MSS*.

CONFIG_IP_NF_MATCH_STATE - Aquí tenemos una de las mayores novedades respecto a **ipchains**. Con este módulo podemos realizar comparaciones por flujos de paquetes (stateful matching). Por ejemplo, si en una conexión *TCP* ya hemos visto tráfico en dos direcciones, los paquetes que les sigan serán considerados como **ESTABLISHED** (establecido), aplicándoles por éllo las mismas acciones que a los paquetes que iniciaron el flujo. Este módulo se usa ampliamente en el ejemplo *rc.firewall.txt*.

CONFIG_IP_NF_MATCH_UNCLEAN - Este módulo nos brinda la posibilidad de comparar paquetes *IP*, *TCP*, *UDP* e *ICMP* que no cumplen con las normas o son inválidos. En condiciones normales se desecharán estos paquetes, pero nunca sabremos si son legítimos o no. Además, ten en cuenta que esta comparación todavía está en fase experimental y puede que no funcione correctamente en todos los casos.

CONFIG_IP_NF_MATCH_OWNER - con esta opción tendremos la oportunidad de comparar en base al propietario de la conexión. Por ejemplo, podremos permitir acceso a Internet únicamente al usuario "root". Este módulo se escribió para mostrar lo que se podía lograr con el nuevo **iptables**. Ten en cuenta que esta comparación es experimental y puede que no le funcione bien a todo el mundo.

CONFIG_IP_NF_FILTER - este módulo añade la tabla *filter* básica que permitirá efectuar el filtrado *IP*. En la tabla *filter* encontraremos las cadenas *INPUT*, *FORWARD* y *OUTPUT*. Este módulo es necesario si pretendemos hacer algún tipo de filtrado en los paquetes que recibamos y/o enviemos.

CONFIG_IP_NF_TARGET_REJECT - este objetivo nos permite especificar que se debe enviar un mensaje de error *ICMP* como respuesta a los mensajes entrantes, en lugar de simplemente desecharlos e ignorarlos. Ten en cuenta que las conexiones *TCP*, al contrario que las *ICMP* y las *UDP*, siempre se reinician o rechazan con un paquete *TCP RST*.

CONFIG_IP_NF_TARGET_MIRROR - sirve para permitir a los paquetes que sean devueltos ("rebotados") al remitente. Por ejemplo, si configuramos un objetivo *MIRROR* en el puerto de destino *HTTP*, en nuestra cadena *INPUT*, y alguien intenta acceder a este puerto, le devolveremos sus paquetes y como resultado probablemente acabará viendo su propia pagina web inicial (homepage).

CONFIG_IP_NF_NAT - este módulo permite que se efectúe la traducción de dirección de red (*network address translation*), o *NAT*, en sus diferentes variantes. La opción nos da acceso a la tabla nat en iptables y es necesaria si queremos hacer reenvío a puertos (port forwarding), enmascaramiento (masquerading), etc. Ten en cuenta que esta opción no es imprescindible para el cortafuegos y el enmascaramiento de una *LAN*, pero deberías tenerlo activo a no ser que seas capaz de asignar direcciones IP únicas para cada uno de los hosts. Así pues, esta opción es necesaria para que el script de ejemplo *rc.firewall.txt* funcione correctamente, y es ciertamente imprescindible si no puedes asignar direcciones IP únicas a cada host.

CONFIG_IP_NF_TARGET_MASQUERADE - este módulo añade el objetivo **MASQUERADE**. Por ejemplo, si no sabemos qué dirección IP tenemos para conectar a Internet, ésta será la forma ideal de conseguir la IP en vez de utilizar *DNAT* o *SNAT*. En otras palabras, si utilizamos *DHCP*, *PPP*, *SLIP* o cualquier otra conexión que nos asigne una IP, necesitamos utilizar este objetivo en lugar de *SNAT*. El enmascaramiento produce una carga en el sistema algo mayor que *NAT*, pero funcionará sin que necesitemos conocer previamente la dirección IP.

CONFIG_IP_NF_TARGET_REDIRECT - este objetivo es útil al emplearlo junto a *proxies de aplicación*, por ejemplo. En vez de dejar simplemente que el paquete pase, lo remapeamos para que se dirija a nuestra máquina local. En otras palabras, de esta forma tenemos la posibilidad de crear un *proxy transparente*.

CONFIG_IP_NF_TARGET_LOG - esta opción añade el objetivo **LOG** y su funcionalidad a **iptables**. Podemos utilizar este módulo para registrar determinados paquetes en el syslogd y así ver qué les está pasando. Ésto sólo ya es inestimable de cara a las auditorías de seguridad, forenses o de depuración de errores del script que estés escribiendo.

CONFIG_IP_NF_TARGET_TCPMSS - esta opción se puede emplear para evitar a los Proveedores de Servicios de Internet (ISPs) y a los servidores que bloquean los paquetes "*ICMP Fragmentation Needed*". Los efectos de esta acción pueden ser páginas web que no lleguen, pequeños correos que sí lleguen mientras que los grandes no lo consigan, las conexiones ssh funcionan pero las conexiones scp se pierden ("mueren") tras el saludo inicial, ... En estos casos podemos utilizar el objetivo *TCPMSS* para superar el problema ajustando nuestro *MSS* (tamaño máximo de segmento) al *PMTU* (Path Maximum Transmit Unit, unidad máxima de transmisión). De esta forma seremos capaces de trabajar con lo que los autores de Netfilter llaman (en la ayuda para la configuración del kernel) "criminally brain-dead ISPs or servers", algo así como "servidores o ISPs absolutamente descerebrados".

CONFIG_IP_NF_COMPAT_IPCHAINS - añade un modo de compatibilidad con el obsoleto **ipchains**. No confíes en este módulo como solución a largo plazo para resolver la migración de los núcleos 2.2 a los 2.4, ya que es probable que desaparezca con la llegada del núcleo 2.6.

CONFIG_IP_NF_COMPAT_IPFWADM - módulo de compatibilidad con el obsoleto **ipfwadm**. Ni se te ocurra recurrir a este módulo como solución a largo plazo.

Como puedes observar hay un buen puñado de opciones. Acabo de explicar brevemente los tipos de comportamiento extras que puedes esperar con cada módulo, aunque sólo se trata de las opciones disponibles en un núcleo 2.4.9 simple, sin ningún extra. Si quieres ver más opciones, te recomiendo que mires las funciones existentes en el "*patch-o-matic*" (POM) de la zona de usuario de *Netfilter*, pues encontrarás montones de opciones extras para el núcleo. Los parches del *POM* son opciones extra que supuestamente se añadirán al núcleo en el futuro, pero que todavía no se han desarrollado lo suficiente como para añadírselos. Las razones pueden ser variadas, como que el parche todavía no sea estable, o que Linus Torvalds no pueda mantenerlo, o incluso que no quiera introducir el parche en el desarrollo del núcleo por ser todavía experimental.

Necesitarás tener compiladas en el núcleo las opciones siguientes, o bien tenerlas como módulos, para que el script *rc.firewall.txt* pueda funcionar. Si necesitas ayuda acerca del resto de opciones necesarias para los otros scripts, léete el capítulo sobre los scripts de ejemplo.

- CONFIG_PACKET
- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_FTP
- CONFIG_IP_NF_IRC
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_TARGET_LOG
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_TARGET_MASQUERADE

Como mínimo necesitarás lo anterior para utilizar el script *rc.firewall.txt*. Para el resto de scripts explicaré lo que necesitan en sus respectivas secciones. Por el momento intentemos centrarnos en el ejemplo principal.

Configuración de la zona de usuario

Para empezar veremos cómo se compila el paquete **iptables**. Es importante comprender que la mayor parte de la configuración y compilación de iptables va de la mano de la configuración y compilación del núcleo (kernel). Determinadas distribuciones de Linux vienen con el paquete **iptables** preinstalado, como ocurre por ejemplo con Red Hat. Sin embargo en esta distribución por defecto está desactivado y ahora veremos cómo activarlo.

Compilando las aplicaciones de la zona de usuario

Empieza por descomprimir (desempaquetar) el paquete **iptables**. Para este ejemplo hemos utilizado el paquete *iptables 1.2.6a* y el núcleo 2.4 sin extras. Descomprime como es habitual, utilizando el comando **bzip2 -cd iptables-1.2.6a.tar.bz2 | tar -xvf -** (aunque también puedes teclear **tar -xjvf iptables-1.2.6a.tar.bz2**, que con las versiones más recientes de **tar** dará el mismo resultado). El paquete debería haberse descomprimido en un nuevo directorio llamado ***iptables-1.2.6a***. Para una explicación más completa de la compilación y la forma de hacer que funcione el programa, léete el fichero iptables-1.2.6a/INSTALL.

Tras ésto, tienes la opción de configurar e instalar módulos/opciones extra en el núcleo. El procedimiento que describiremos sólo comprobará e instalará parches estándar que están pendientes de incluirse en el núcleo; existen otros parches de tipo experimental que sólo estarán disponibles al seguir otro tipo de configuración.

> Algunos de estos parches son realmente experimentales y no sería una buena idea instalarlos. Sin embargo, hay montones de comparaciones y objetivos extremadamente interesantes en este punto de la instalación, así que no tengas miedo de, al menos, echarles un vistazo.
>
> Para empezar a instalar, ejecutaremos un comando similar al siguiente desde el directorio raíz del paquete iptables:

make pending-patches KERNEL_DIR=/usr/src/linux/

La variable KERNEL_DIR debería indicar la localización actual de tu núcleo. Normalmente este directorio será /usr/src/linux/, aunque puede ser diferente y lo más probable es que ya sepas dónde se encuentra el código fuente del núcleo.

Note

Con el comando anterior sólo añadiremos determinados parches que de todas formas están a punto de entrar en el núcleo. Puede que hayan más parches y añadidos que los desarrolladores de Netfilter estén a punto de añadir al núcleo, pero que todavía no se encuentran listos para dar este paso. Una forma de instalar estos extras es mediante el siguiente comando:

make most-of-pom KERNEL_DIR=/usr/src/linux/

De esta forma se preguntará si se instalan algunos añadidos de lo que en la jerga de Netfilter se llaman **patch-o-matic**, pero eludiendo los parches más extremos, ya que pueden causar la desintegración del núcleo. Date cuenta que se ha dicho "se preguntará" porque ésto es lo que hará el comando: preguntará antes de cambiar algo en el código fuente del núcleo. Para poder instalar *todo* el material presente en el *patch-o-matic* necesitarás ejecutar el siguiente comando:

make patch-o-matic KERNEL_DIR=/usr/src/linux/

No olvides leer completamente la ayuda de cada parche antes de hacer nada. Algunos parches destruyen a otros parches, mientras que otros pueden destruir el núcleo si se usan junto a otros determinados parches del *patch-o-matic*, etc.

Note

No es imprescindible seguir los pasos anteriores los pasos anteriores, por éllo puedes ignorarlos si no quieres "parchear" tu núcleo. Sin embargo, hay cosas muy interesantes en el *patch-o-matic* a las que sería conveniente echar un vistazo, por lo que no estaría de más ejecutar los comandos aunque sólo fuera para ver lo que contienen.

En este punto ya has terminado la parte de la instalación referida al *patch-o-matic* y es momento de compilar un nuevo núcleo utilizando los nuevos parches que hayas añadido al código fuente. No olvides configurar de nuevo el núcleo, ya que probablemente los nuevos parches no se habrán añadido aún a las opciones configuradas. Por otra parte, si quieres puedes compilar primero el programa **iptables** y dejar para el final la compilación del núcleo.

Compila ahora el programa de zona de usuario **iptables**. Para éllo ejecuta el siguiente comando:

make KERNEL_DIR=/usr/src/linux/

La aplicación de zona de usuario debería compilarse correctamente. Si no es así, tendrás que buscar soluciones por tí mismo, o puedes subscribirte a la lista de correo de Netfilter (en inglés), dónde tendrás la oportunidad de pedir que alguien te ayude a solucionar tus problemas. Puede que haya algo que no vaya bien durante la instalación, pero no te preocupes: intenta pensar de manera lógica sobre el problema y averigua qué es lo que va mal, o pídele a alguien que te ayude.

Si todo ha ido bien, estarás preparado para instalar los archivos binarios. Para éllo, escribe:

make install KERNEL_DIR=/usr/src/linux/

En principio todo debería funcionar en el programa. Para utilizar los cambios efectuados en **iptables** deberías recompilar y reinstalar tu núcleo y sus módulos, si es que todavía no lo has hecho. Para una explicación más detallada de la instalación a partir del código fuente de las aplicaciones de zona de usuario, lee el archivo INSTALL que viene con el código fuente, pues contiene excelente información sobre todo lo referente a la instalación.

Instalación en Red Hat 7.1

La distribución Red Hat en su versión 7.1 viene precompilada con un núcleo 2.4.x que contiene a *Netfilter* e **iptables**. Además, contiene los programas básicos de zona de usuario, así como los ficheros de configuración necesarios para ejecutarlos. Sin embargo, los de Red Hat han desactivado el paquete completo y en su lugar utilizan el módulo **ipchains** por compatibilidad con versiones anteriores. Molesto, por no decir otra cosa, y fuente de multitud de consultas en las listas de correo preguntando por qué no funciona **iptables**. Así pues, echemos un vistazo a cómo desactivar el módulo ipchains y cómo instalar **iptables** en su lugar.

> La instalación por defecto de Red Hat 7.1 incluye una decepcionante versión antigua de las aplicaciones de zona de usuario, por lo que es posible que desees compilar una nueva versión de estas aplicaciones, además de instalar un núcleo nuevo y personalizado antes de exigirle el máximo rendimiento a **iptables**.

Antes de empezar tendrás que desactivar los módulos de **ipchains** de manera que no vuelvan a arrancar de ahora en adelante. Para conseguirlo deberás cambiar algunos nombres de fichero en los subdirectorios existentes en /etc/rc.d/. Con el siguiente comando tendrás suficiente:

chkconfig --level 0123456 ipchains off

Una vez ejecutado modificarás todos los enlaces virtuales ("soft links") que apuntan al script /etc/rc.d/init.d/ipchains y los convertirás en K92ipchains. La primera letra, que por defecto es una S, le indica a los scripts de inicio que se ejecuten. Al cambiarla por una K se le indica a los scripts que terminen el servicio, o que no lo ejecuten si no estaba arrancado. De esta forma el servicio ya no arrancará nunca.

Sin embargo, para detener un servicio que se está ejecutando en este momento necesitamos introducir otro comando: **service**, que puede utilizarse para actuar sobre servicios actualmente en ejecución. Así pues, para parar el servicio **ipchains** escribiremos:

service ipchains stop

Por último, para arrancar el servicio **iptables** necesitamos saber en qué niveles de ejecución ("run-levels") queremos que se inicie. Normalmente serán los niveles 2, 3 y 5, que se usan para:

- 2. Multiusuario sin NFS o lo mismo que el nivel 3 si no existe ninguna red.

- 3. Modo multiusuario completo, es decir, el nivel de ejecución habitual en el que se trabaja.
- 5. X11. Este modo se utiliza cuando al arrancar el sistema entras directamente en las Xwindows (el modo gráfico). Actualmente cada vez son más las distribuciones que entran por defecto en este modo.

Para conseguir que **iptables** se ejecute en estos niveles de ejecución, escribiremos:

chkconfig --level 235 iptables on

Es decir, instruímos a **iptables** para que se ejecute en los niveles 2, 3 y 5. Si quieres que se ejecute en cualquier otro nivel, deberás añadirlo al comando anterior. Sin embargo, ninguno de los otros niveles debería usarse para ejecutar iptables, por lo que en realidad no necesitas activarlo en éllos: el nivel 1 es el modo monousuario (1 sólo usuario), empleado cuando necesitas arreglar una instalación. El nivel 4 no debería usarse y el nivel 6 es para apagar el ordenador.

Para activar el servicio **iptables**, ejecutaremos:

service iptables start

No hay ninguna regla en el script de **iptables**. Para añadirlas en Red Hat 7.1 normalmente se emplean dos métodos. El primero es editar el script /etc/rc.d/init.d/iptables, teniendo como efecto secundario no deseado el hecho de que borrarás todas las reglas si actualizaste el paquete iptables mediante un RPM. El otro método consistiría en cargar/crear el conjunto de reglas y luego guardarlo con el comando **iptables-save**, de forma que después se carguen automáticamente mediante los scripts rc.d.

Primero explicaremos cómo configurar **iptables** mediante el socorrido método de copiar y pegar al script "init.d" de **iptables**. Para añadir reglas que se tengan que ejecutar cuando el sistema arranca el servicio, añádelas en la sección "start)", o en la función "start()". Ten en cuenta que si añades las reglas en la sección "start)" no debes olvidar parar la función "start()" dentro de esa misma sección "start)". Además, tampoco olvides editar la sección "stop)" para indicarle al script qué debe hacer, por ejemplo cuando es sistema se esté cerrando (apagando), o cuando se entre en un nivel de ejecución que no necesite de **iptables**. Por último, no olvides repasar las secciones "restart" y "condrestart". También debes tener en cuenta que todo este trabajo no servirá de nada si, por ejemplo, actualizas automáticamente tus paquetes mediante el Red Hat Network, o si actualizas mediante el paquete RPM de **iptables**.

En cuanto al segundo método de realizar la configuración, se basa en los siguientes pasos: primero crea un conjunto de reglas en un fichero de script de línea de comandos (un script de shell), o bien directamente con **iptables**, que cumpla con tus necesidades y empieza a experimentar con él durante un tiempo. Cuando tengas una configuración que funcione correctamente (o al menos que tú creas que no tiene errores), utiliza el comando **iptables-save**. Puedes utilizar: **iptables-save > /etc/sysconfig/iptables**, con lo que grabarías el conjunto de reglas en el fichero /etc/sysconfig/iptables. Este fichero lo usará en el futuro el script rc.d de **iptables** para cargar de nuevo el conjunto de reglas. Otra forma de conseguir un resultado equivalente es mediante el comando **service iptables save**, que automáticamente guardará el script en el fichero /etc/sysconfig/iptables. Así, la próxima vez que reinicies el sistema, el script rc.d de **iptables** utilizará el comando **iptables-restore** para cargar el conjunto de reglas desde el fichero /etc/sysconfig/iptables. Atención: no mezcles ambos métodos, ya que se pueden perjudicar gravemente el uno al otro y dejar la configuración del cortafuegos completamente inútil.

Cuando hayas terminado todos estos pasos, puedes desinstalar los paquetes **ipchains** e **iptables** preinstalados. De esta forma no permitiremos al sistema que mezcle las nuevas aplicaciones de zona de usuario de **iptables** con las viejas versiones preinstaladas. Sin embargo, este paso sólo es necesario si vas a instalar **iptables** desde el código fuente. No es raro que el paquete nuevo se mezcle con el viejo, puesto que la instalación basada en rpm instala el paquete en directorios no estándar, de forma que la nueva instalación no sobreescribirá a la vieja. Para efectuar la desinstalación, escribe:

rpm -e iptables

Y como no tiene mucho sentido mantener instalado **ipchains** cuando no lo vas a utilizar nunca más, desinstálalo de la misma forma:

rpm -e ipchains

Tras todo ésto, habrás finalizado la actualización del paquete **iptables** mediante el código fuente, habiendo seguido las instrucciones del código. Además ningún viejo ejecutable, librería o fichero auxiliar permanecerá instalado.

Chapter 3. Atravesando tablas y cadenas

En este capítulo hablaremos de cómo atraviesan los paquetes las diferentes cadenas y en qué orden. También analizaremos el orden en que las tablas son atravesadas. Más adelante veremos cuán valioso resulta ésto para escribir nuestras propias reglas. También veremos los momentos en que ciertos componentes, también dependientes del núcleo, entran en escena (por poner un ejemplo, las diferentes decisiones de enrutado). Ésto es especialmente necesario si pretendemos escribir reglas en **iptables** para poder cambiar las reglas/patrones de enrutado de los paquetes, o sea, si pretendemos conocer el por qué y el cómo los paquetes son enrutados, al hacer por ejemplo **DNAT** y **SNAT**. Y por supuesto, no olvidaremos los bits *TOS* (Type Of Service, o tipo de servicio).

Generalidades

Cuando un paquete entra en el cortafuegos (firewall), alcanza el hardware y es procesado en el núcleo por su driver correspondiente. Después el paquete empieza a recorrer una serie de etapas en el núcleo antes de ser enviado a la aplicación adecuada (localmente), reenviada hacia otro host, o cualquier otra operación.

En primer lugar, echemos un vistazo a un paquete destinado a nuestro propio host local (nuestra máquina). Recorrerá los siguientes pasos antes de ser entregado a la aplicación que lo requiere:

Table 3-1. Host local de destino (nuestra propia máquina)

Etapa	Tabla	Cadena	Comentario
1			En los cables (por ejemplo Internet)

Etapa	Tabla	Cadena	Comentario
2			Llega a la interfaz de red (por ejemplo eth0)
3	mangle	PREROUTING	Esta cadena se usa normalmente para modificar/"deformar" (mangle) paquetes, es decir, cambiar el *TOS* y cosas así.
4	nat	PREROUTING	Esta cadena se usa principalmente para la traducción de direcciones de red de destino (*DNAT*, Destination Network Address Translation). Debes evitar filtrar en esta cadena ya que será puenteada (bypassed) o esquivada en ciertos casos.
5			Decisión de enrutamiento, o sea, ¿está el paquete destinado a nuestro host local o debe ser reenviado?, ¿hacia dónde?.
6	mangle	INPUT	En este punto se alcanza la cadena *INPUT* de la tabla mangle. Usaremos esta cadena para modificar/"retocar" paquetes después de que hayan sido enrutados, pero antes de que se envíen al proceso de destino.
7	filter	INPUT	Aquí es donde filtraremos todo el tráfico entrante destinado a nuestro host local. Ten en cuenta que todo el tráfico entrante pasa a través de esta cadena, sin importar la interfaz por la que entre o de dónde proceda.

Etapa	Tabla	Cadena	Comentario
8			Proceso/aplicación local (es decir, programa cliente/servidor)

Date cuenta que esta vez el paquete ha atravesado la cadena *INPUT* en lugar de la cadena *FORWARD* y ésto es bastante lógico. Lo más probable es que ahora al principio sea lo único que te parecerá realmente lógico acerca de cómo se atraviesan las tablas y cadenas, pero conforme sigas pensando en éllo con el paso del tiempo, lo irás viendo todo aún más claro.

Ahora nos centraremos en los paquetes que salen de nuestro host y las etapas por las que pasan.

Table 3-2. Host local de origen (nuestra própia máquina)

Etapa	Tabla	Cadena	Comentario
1			Proceso/aplicación local (es decir, programa cliente/servidor)
2			Decisión de enrutamiento. Qué dirección de origen usar, qué interfaz de salida usar, y otra información que necesita ser recopilada.
3	mangle	OUTPUT	Aquí es donde se modifican los paquetes; se sugiere que no filtres en esta cadena porque pueden producirse efectos secundarios.
4	nat	OUTPUT	Esta cadena puede ser usada para hacer NAT a los paquetes que salen desde el firewall.
5	filter	OUTPUT	Aquí es donde filtramos los paquetes salientes de nuestro host local.

Etapa	Tabla	Cadena	Comentario
6	mangle	POSTROUTING	La cadena *POSTROUTING* de la tabla mangle se usa principalmente cuando queremos modificar los paquetes antes de que dejen nuestro host, pero después de tomar las decisiones de enrutamiento. Esta cadena será alcanzada tanto por los paquetes que atraviesan el cortafuegos, como por los generados por él mismo.
7	nat	POSTROUTING	Aquí es donde efectuamos la traducción de las direcciones de red de origen (*SNAT*, Source Network Address Translation) como ya se ha descrito anteriormente. Es conveniente que no filtres en esta cadena ya que pueden producirse efectos secundarios y determinados paquetes podrían colarse incluso aunque se haya establecido la política **DROP** (desechar) como política por defecto.
8			Sale al exterior por alguna interfaz (por ejemplo, eth0)
9			En los cables (por ejemplo, Internet)

En este ejemplo estamos asumiendo que el paquete está destinado a otro host de otra red. El paquete sigue los siguientes pasos, de esta manera:

Table 3-3. Paquetes Reenviados (Forwarded)

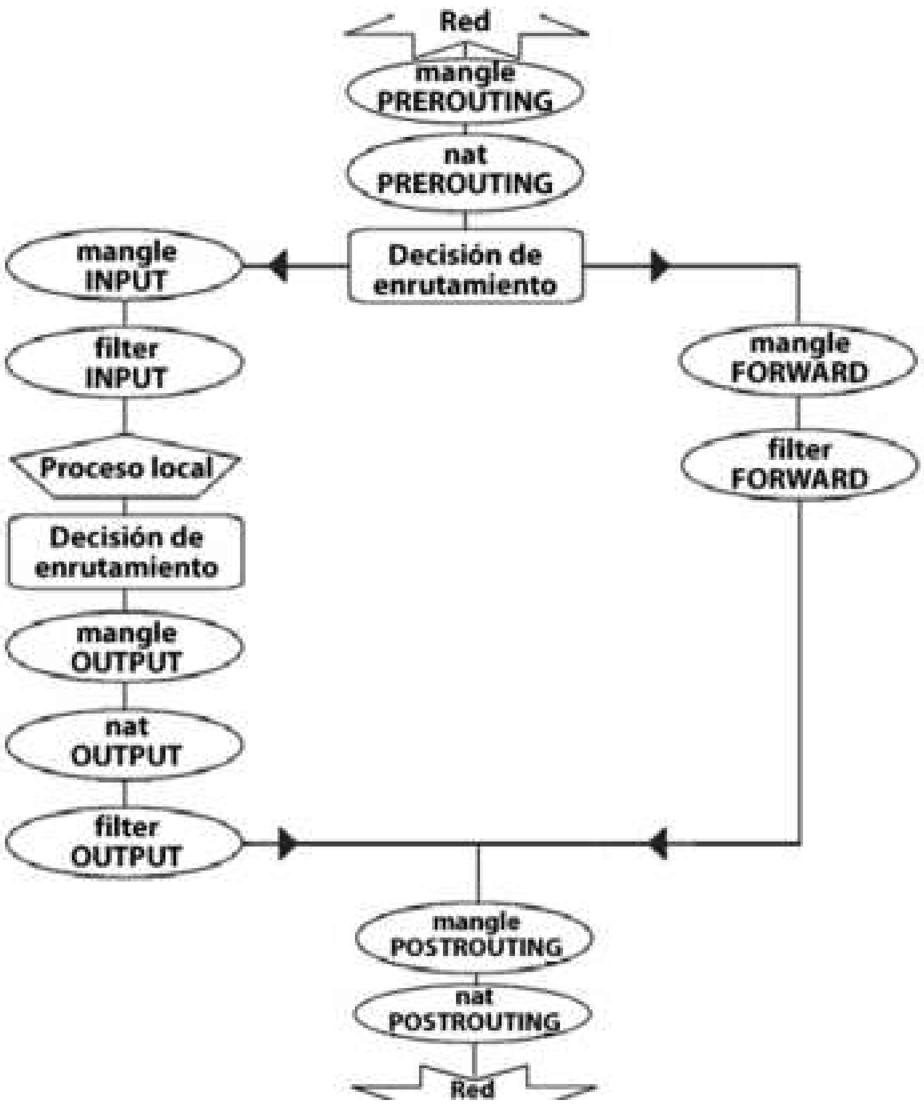
Etapa	Tabla	Cadena	Comentario
1			En los cables (es decir, Internet)
2			Llega hasta la interfaz de red (es decir, eth0)
3	mangle	PREROUTING	Esta cadena se usa normalmente para modificar paquetes, o sea, para cambiar el *TOS* y acciones similares.
4	nat	PREROUTING	Esta cadena se usa principalmente para hacer *DNAT* (traducción de dirección de destino). El *SNAT* (traducción de dirección de origen) se realiza más adelante. Debes evitar filtrar en esta cadena ya que en ciertos casos será puenteada o esquivada.
5			Decisión de enrutamiento, o sea, ¿el paquete está destinado a nuestro propio host local, o debe ser reenviado?, ¿hacia dónde?.
6	mangle	FORWARD	El paquete es enviado a la cadena *FORWARD* de la tabla mangle. Ésto puede aprovecharse para necesidades muy específicas dónde queremos modificar paquetes después de la decisión de enrutamiento inicial, pero antes de la última decision de enrutamiento, hecha justo antes de que el paquete sea enviado.

Etapa	**Tabla**	**Cadena**	**Comentario**
7	filter	FORWARD	El paquete es enrutado hacia la cadena *FORWARD*. Solamente los paquetes reenviados pasan por aquí y es donde hacemos todo el filtrado. Ten en cuenta que todos los paquetes reenviados (en cualquier dirección) pasan por aquí, así que necesitarás pensar en éllo cuando escribas tu conjunto de reglas.
8	mangle	POSTROUTING	Esta cadena se usa para efectuar los tipos específicos de modificación de paquetes (packet mangling) que queramos llevar a cabo después de que todos los tipos de decisiones de enrutamiento se hayan tomado, pero estando el paquete aún en esta máquina.
9	nat	POSTROUTING	Esta cadena debe ser usada principalmente y sobretodo para efectuar *SNAT*. Debes evitar filtrar aquí ya que ciertos paquetes podrían "pasar por delante" de la cadena sin ni siquiera rozarla. Aquí es también dónde se realiza el enmascaramiento (Masquerading).
10			Sale por la interfaz de salida (por ej. eth1).
11			En los cables de nuevo (es decir, la red local).

Como puedes ver hay bastantes pasos que dar. El paquete puede ser bloqueado en cualquiera de las cadenas de **iptables** o en algún otro sitio si está malformado; sin embargo, estamos principalmente interesados en la parte relacionada con **iptables**. Ten en cuenta que no hay cadenas o tablas específicas para interfases diferentes ni nada por el estilo. La cadena *FORWARD* es siempre atravesada por todos los paquetes que son reenviados a través de este cortafuegos/router.

¡No uses la cadena *INPUT* para filtrar en el escenario anterior! *INPUT* está pensada sólamente para nuestro host local, no para hacer enrutamientos a ningún otro destino.

Hemos visto hasta ahora cómo las cadenas son atravesadas en tres escenarios independientes. Si deseáramos hacer un buen esquema de todo ésto, podría asemejarse a algo como:



Para aclarar esta imagen considera lo siguiente: si llega un paquete que no está destinado a nuestra máquina local, en la primera decisión de enrutamiento será dirigido hacia la cadena *FORWARD*. Por otra parte, si el paquete está destinado a una *dirección IP* que está siendo escuchada por nuestra máquina local, el paquete sería enviado a la cadena *INPUT* y después a la máquina local.

También conviene advertir el hecho de que los paquetes pueden estar destinados a la máquina local y, sin embargo, la dirección de destino ser modificada por la cadena *PREROUTING* haciendo *NAT*. Dado que ésto tiene lugar antes de la primera decisión de enrutamiento, el paquete será revisado tras dicho cambio. Debido a ésto, el camino a seguir por el paquete (debido a una modificación de la IP de destino) podría cambiar antes de que la decisión de enrutamiento sea tomada. Date cuenta que *todos* los paquetes irán a través de un camino u otro de esta imagen. Si haces *DNAT* devolviendo el paquete a la misma red por la que vino, seguirá viajando por el resto de cadenas hasta que regrese a dicha red.

Si crees que necesitas más información puedes usar el script *rc.test-iptables.txt*. Este script "de laboratorio" debería proporcionarte las reglas necesarias para comprobar cómo son atravesadas las tablas y las cadenas.

Tabla mangle

Esta tabla debe ser principalmente usada para modificar paquetes, como ya hemos dicho antes. En otras palabras, aquí puedes usar libremente las comparaciones de modificación para cambiar el campo *TOS* (Type Of Service), entre otras cosas.

Es más que recomendable no usar esta tabla como filtro: ni **DNAT**, ni **SNAT**, ni **Masquerading** trabajan en esta tabla.

Objetivos para los que únicamente se destina la tabla mangle:

- TOS
- TTL
- MARK

El **TOS** es usado para definir y/o cambiar el campo *Type Of Service* del paquete. Puede ser usado para configurar políticas en la red considerando cómo deben ser enrutados los paquetes y tareas similares. Ten en cuenta que éste tema no ha sido perfeccionado y no está realmente implementado en Internet, por lo que la mayoría de los routers no hacen caso del valor de este campo y aún más, a veces actúan de manera imperfecta sobre los paquetes que reciben. En otras palabras, no lo uses para paquetes que vayan hacia Internet, a menos que quieras tomar decisiones de enrutamiento sobre ellos con iproute2.

El **TTL** es usado para cambiar el campo *TTL* (Time To Live) de un paquete y con éllo conseguir que los paquetes tengan un *TTL* específico. Una buena razón para hacerlo podría ser que no queramos ser descubiertos por ciertos proveedores de servicios de internet (ISP) demasiado fisgones: a algunos ISPs no les gustan los clientes que usan multiples ordenadores bajo una única conexión y saben que se trata de una única conexión porque de un mismo host llegan paquetes con diferentes valores de *TTL* (entre otros muchos signos delatores).

El **Mark** se usa para para marcar (*mark*) los paquetes con valores específicos. Estas marcas pueden ser reconocidas posteriormente por los programas iproute2 para realizar diferentes enrutamientos dependiendo de la marca que tengan o no tengan los paquetes. También podemos limitar en ancho de banda y realizar Class Based Queuing (colas basadas en clases, CBQ) según dichas marcas.

Tabla nat

Esta tabla debe ser usada sólo para hacer *NAT* (Network Address Translation) a los diferentes paquetes. En otras palabras, debe ser empleada solamente para traducir el campo origen del paquete o el campo destino. Ten en cuenta que tal como hemos dicho antes, sólo el primer paquete de un flujo alzanzará esta cadena. Después, al resto de paquetes del mismo flujo de datos se les aplicará la misma acción que al primero. Los objetivos que hacen este tipo de cosas son:

- DNAT
- SNAT
- MASQUERADE

El objetivo **DNAT** (Destination Network Address Translation) se emplea principalmente en los casos donde se tiene una IP pública y se quiere redirigir los accesos al firewall hacia algún otro host (en una "zona desmilitarizada", *DMZ*, por ejemplo). Dicho de otro modo, cambiamos la dirección de destino del paquete y lo re-enrutamos a otro host.

SNAT (Source Network Address Translation) es principalmente usada para cambiar la dirección de origen de los paquetes. La mayoría de las veces querrás esconder tus redes locales, *DMZ*, etc. Un ejemplo muy bueno podría ser cuando queremos sustituir las direcciones IP de la red local que está tras el cortafuegos, por la dirección IP del propio cortafuegos, que posee una IP pública hacia fuera. Con este objetivo el firewall automaticamente hará **SNAT** y **de-SNAT** sobre los paquetes, lo cual hace posible que las conexiones provenientes de la *LAN* salgan a Internet. Por ejemplo, si tu red usa 192.168.0.0/máscara_de_red, los paquetes nunca regresarán de Internet, porque IANA ha designado dicho rango de direcciones (entre otras) como privadas y sólo para ser usadas en redes locales aisladas.

El objetivo **MASQUERADE** se usa exactamente para lo mismo que **SNAT**, pero **MASQUERADE** requiere un poquito más de trabajo del procesador. La razón es que cada vez que llega un paquete al objetivo **MASQUERADE**, automáticamente chequea qué dirección IP debe asignarle, en lugar de hacer como **SNAT**, que simplemente utiliza la dirección IP configurada. **MASQUERADE** hace posible trabajar con las direcciones IP Dinámicas por *DHCP* que tu ISP pueda proporcionarte a través de conexiones a Internet vía *PPP*, *PPPoE* o *SLIP*.

tabla Filter

La tabla *filter* se usa principalmente para el filtrado de paquetes: podemos comparar y filtrar paquetes de la forma que queramos. Se trata del lugar en que miramos el contenido de los paquetes y tomamos la determinación de desecharlos (**(DROP)**) o aceptarlos (**(ACCEPT)**). Por supuesto, podemos hacer filtrado antes, pero esta tabla en particular fue diseñada para realizar todas las operaciones de filtrado. Casi todos los objetivos se pueden usar en esta tabla. Trataremos ésto más ampliamente, pero desde ahora ya sabes que esta tabla es el lugar correcto para llevar a cabo el principal filtrado.

Chapter 4. La máquina de estados

Este capítulo trata sobre la máquina de estados y la explica detalladamente, de forma que llegarás a comprender cómo trabaja. También hay una gran cantidad de ejemplos que clarifican cómo son tratados los estados por la máquina de estados: estos ejemplos nos ayudarán a entenderlo todo de la manera más práctica posible.

Introducción

La máquina de estados es una parte especial de iptables que no debería llamarse así ("máquina de estados"), puesto que en realidad es una máquina de seguimiento de conexiones. Sin embargo, mucha gente la conoce por el primer nombre. A lo largo del capítulo utilizaré ambos nombres como si fueran equivalentes, aunque ésto no debería ocasionar demasiados problemas. El seguimiento de conexiones se efectúa para que la estructura de Netfilter sepa cuál es el estado de cada conexión específica. Los cortafuegos que trabajan de esta manera normalmente se denominan "stateful firewalls" (cortafuegos que consideran las peticiones y respuestas de una máquina como una única conexión de un mismo flujo de datos; podríamos traducirlo como "cortafuegos de flujos"), mientras que los que no lo hacen se denominan "non-stateful firewalls" (cada petición es independiente, aunque provenga de la misma máquina; podríamos traducirlo como "cortafuegos de conexiones"). Los cortafuegos de tipo "stateful" son mucho más seguros, pues nos permiten elaborar conjuntos de reglas mucho más precisos.

En iptables los paquetes se pueden relacionar con las conexiones mediante cuatro "estados" diferentes: **NEW** (nuevo), **ESTABLISHED** (establecido), **RELATED** (relacionado) e **INVALID** (inválido o no válido). Los trataremos en profundidad más adelante. Mediante la comparación **--state** podemos controlar fácilmente qué o quién tiene permiso para iniciar nuevas sesiones.

Todo el seguimiento de las conexiones la realiza una estructura especial del núcleo llamada "conntrack". El "conntrack" puede cargarse como módulo o como una parte más del núcleo, pero sea como sea la mayoría de las veces necesitaremos y pediremos un seguimiento de conexiones más específico que el que proporciona por defecto el motor del conntrack. Por éllo, hay unas partes específicas que se encargan de los protocolos *TCP*, *UDP* o *ICMP*, entre otros. Estos módulos captan información específica, única, de los paquetes, de forma que pueden mantener un seguimiento de cada flujo de datos. La información que recopilan los módulos se utiliza para indicarle al conntrack en qué estado se encuentra el flujo en cada momento. Por ejemplo, los flujos *UDP* normalmente se identifican y distinguen por su *dirección IP de destino*, su *dirección IP de origen*, su *puerto de destino* y su *puerto de origen*.

En anteriores núcleos teníamos la posibilidad de ejecutar o parar la desfragmentación de paquetes, sin embargo, desde que se introdujeron iptables y Netfilter y en particular desde que se introdujo el seguimiento de conexiones, esta opción fue eliminada. La razón es que el seguimiento de conexiones no funciona correctamente sin defragmentar los paquetes, por lo que se ha introducido en el conntrack y el proceso se realiza automáticamente. No se puede parar si no paramos también el seguimiento de conexiones. En otras palabras, desde el momento en que se emplea el seguimiento de conexiones, se emplea la desfragmentación de paquetes.

Todo el seguimiento de conexiones se efectúa en la cadena *PREROUTING*, excepto los paquetes generados localmente, que son controlados en la cadena *OUTPUT*. Ésto significa que iptables realizará todo el recálculo de estados en estas dos cadenas de la tabla Nat: si somos nosotros los que enviamos el paquete inicial del flujo, su estado se establece como **NEW** (nuevo) en la cadena *OUTPUT*, mientras que al recibir el paquete de retorno (el que "contesta" al paquete inicial) el estado del flujo se cambia a **ESTABLISHED** (establecido) en la cadena *PREROUTING*. Por el contrario, si el primer paquete no lo hemos originado nosotros, el estado NEW se establece en la cadena *PREROUTING*.

Las "entradas" del conntrack

Entendamos como "entrada" el registro que mantiene el conntrack de cada conexión y toda la información necesaria correspondiente a esa conexión. Así pues, veamos una entrada del conntrack tal como la podríamos encontrar en /proc/net/ip_conntrack y aprendamos a interpretarla (en este fichero encontraremos un listado de todas las entradas presentes en la base de datos del conntrack). Si tienes cargado el módulo *ip_conntrack*, un "**cat**" de /proc/net/ip_conntrack podría parecerse a:

```
tcp      6 117 SYN_SENT src=192.168.1.6 dst=192.168.1.9 sport=32775 \
     dport=22 [UNREPLIED] src=192.168.1.9 dst=192.168.1.6 sport=22 \
     dport=32775 use=2
```

Este ejemplo contiene toda la información que el módulo conntrack mantiene para saber en qué estado se encuentra una conexión determinada. Para empezar tenemos el protocolo, que en este caso es el tcp. A continuación tenemos la misma información en su notación decimal equivalente ("6"). Después viene el tiempo de vida restante para esta entrada en particular: en el ejemplo a esta conexión le quedan 117 segundos, que irán disminuyendo hasta que se vea más tráfico perteneciente a la conexión. Cuando llega ese nuevo tráfico el contador se establece de nuevo a su valor por defecto (valor que depende del estado en que se encuentra la conexión en el momento de recibir nuevos paquetes). Sigue la información del estado actual en que se encuentra la entrada. En el ejemplo estamos viendo un paquete que se encuentra en el estado ***SYN_SENT***. El valor interno de una conexión es ligeramente distinto a los empleados externamente por **iptables**. El valor ***SYN_SENT*** nos dice que tenemos delante una conexión que sólo a visto un paquete *TCP SYN* en una dirección. Después vemos la *dirección IP de origen*, la *dirección IP de destino*, el *puerto de origen* y el *puerto de destino*. En este punto vemos una clave específica que nos indica que no hemos visto ningún tráfico de retorno para esta conexión. Para terminar, vemos los datos que esperamos en los paquetes de retorno: las direcciones de origen y destino (que están invertidas respecto a las anteriores, ya que el paquete será devuelto hacia nosotros) y los puertos de origen y destino (también invertidos respecto a los anteriores). Así pues, éstos son los valores que podrían ser de nuestro interés.

Las entradas del seguimiento de conexiones pueden tomar diferentes valores, todos éllos especificados en las cabeceras ("headers") disponibles en los archivos linux/include/netfilter-ipv4/ip_conntrack*.h. Estos valores dependen del subprotocolo *IP* que estemos usando. Los protocolos *TCP*, *UDP* o *ICMP* toman los valores por defecto especificados en linux/include/netfilter-ipv4/ip_conntrack.h. Lo estudiaremos en detalle al revisar cada protocolo, sin embargo, no ahondaremos demasiado en este capítulo puesto que sólo se utilizan dentro del conntrack. Por otra parte, según cambia el estado también cambia el valor por defecto del tiempo que debe transcurrir hasta eliminar la conexión (el "contador de vida restante de la conexión" explicado anteriormente).

> Note
>
> Desde hace poco hay un nuevo parche en el "patch-o-matic" de iptables llamado "tcp-window-tracking". Este parche añade, entre otras cosas, todos los tiempos límite mencionados a variables especiales del "sysctl", con lo que se pueden cambiar (los tiempos) al instante, mientras el sistema está en marcha. Gracias a ésto ya no es necesario recompilar el núcleo cada vez que se deseen cambiar los límites de los contadores de tiempo.
>
> Estos límites se pueden modificar utilizando llamadas al sistema específicas que se encuentran disponibles en el directorio /proc/sys/net/ipv4/netfilter. Concretamente debes buscar las variables /proc/sys/net/ipv4/netfilter/ip_ct_*.

Cuando una conexión ha visto tráfico en ambas direcciones, se modificará su entrada del conntrack eliminando la clave ***[UNREPLIED]*** (que nos indica que la conexión no ha visto tráfico en ambas direcciones) y añadiendo (casi al final de la entrada) la clave ***[ASSURED]***. Esta clave nos dice que la conexión está asegurada y por éllo no será eliminada si alcanzamos el límite máximo de conexiones simultáneas soportadas por el conntrack, todo lo contrario a aquellas que no están marcadas como ***[ASSURED]***, que sí serán borradas al alcanzar el límite de conexiones. El número de conexiones mantenidas en la tabla del seguimiento de conexiones depende de una variable, que puede establecerse a través de las funciones de ip-sysctl en los núcleos Linux más recientes. El valor por defecto de esta variable depende en gran medida de la cantidad de memoria que tengas instalada: para 128 MB de RAM tendrás un máximo de 8192 entradas, mientras que con 256 MB de RAM llegarás hasta las 16376 entradas. Puedes ver y modificar estos valores a través de la propiedad /proc/sys/net/ipv4/ip_conntrack_max.

Estados del espacio de usuario

Como has podido ver, los paquetes pueden tomar distintos estados dentro del núcleo, dependiendo del protocolo considerado. Sin embargo, fuera del núcleo sólo tenemos los cuatro estados descritos anteriormente. Principalmente estos estados se pueden emplear junto a la comparación de estados (state match), con lo cual esta comparación será capaz de diferenciar paquetes en función del estado en que se encuentren dentro del seguimiento de conexiones. Los estados válidos son: **NEW**, **ESTABLISHED**, **RELATED** e **INVALID**. La siguiente tabla explica brevemente cada posible estado:

Table 4-1. Estados de espacio de usuario

Estado	Explicación

Estado	**Explicación**
NEW	El estado **NEW** (nuevo) nos indica que el paquete es el primero que vemos. Esto significa que el primer paquete que el módulo conntrack vea en una conexión será etiquetado de esta manera. Por ejemplo, si vemos un paquete *SYN* que además es el primero de una conexión, coincidirá con el criterio del conntrack y será etiquetado como "nuevo". Sin embargo, el primer paquete puede que no sea un paquete *SYN* y aún así ser considerado como **NEW**. Este comportamiento puede llevar a determinados problemas en determinados casos, pero también puede ser extremadamente útil si necesitamos captar conexiones perdidas de otros cortafuegos, o si una conexión a excedido su tiempo de espera, pero en realidad no ha sido cerrada.
ESTABLISHED	El estado "**ESTABLISHED**" (establecido) ha visto tráfico en ambas direcciones y por tanto admitirá continuamente los paquetes de ese flujo. Las conexiones "establecidas" son bastante fáciles de comprender: el único requisito para alcanzar el estado "**ESTABLISHED**" es que un host envíe un paquete y obtenga una respuesta del otro host. El estado "**NEW**" (nuevo) cambiará al estado "establecido" en cuanto llegue un paquete de respuesta al cortafuegos (o cuando este paquete pase por el cortafuegos). Los mensajes de error *ICMP*, las redirecciones, ..., también se pueden considerar como "**ESTABLISHED**", si hemos enviado un paquete que a su vez genera el mensaje de error *ICMP*.

Estado	Explicación
RELATED	El estado "**RELATED**" (relacionado) es uno de los más complejos. Una conexión se considera "relacionada" cuando está ligada a otra conexión ya "establecida". Por este motivo, para que una conexión se considere en estado "**RELATED**" primero debemos tener otra conexión en estado "**ESTABLISHED**": la conexión "establecida" generará una conexión externa a la conexión principal, y esta nueva conexión será considerada como "relacionada" siempre que el módulo conntrack pueda entender que está relacionada con la principal. Un buen ejemplo: las conexiones *FTP-data* son consideradas como relacionadas con el puerto de control FTP (*FTP control*); otro ejemplo son las conexiones *DCC* generadas con el *IRC*. Puede utilizarse para permitir las respuestas *ICMP*, las transferencias *FTP* y los *DCC*s (protocolos de conexión Directa de Cliente a Cliente) a través del cortafuegos. Ten en cuenta que muchos protocolos *TCP* (además de algunos *UDP*) que dependen de este mecanismo son bastante complejos y envían información de la conexión conjuntamente con la carga de datos de los segmentos *TCP* o *UDP*, por lo que requieren de módulos de ayuda especiales para ser correctamente interpretados.
INVALID	El estado "**INVALID**" (inválido) implica que el paquete no puede ser identificado o que no tiene ningún estado. Ésto puede ser debido a varias razones, como que el sistema se ha quedado sin memoria disponible, o a mensajes *ICMP* de error que no responden a ninguna conexión conocida. Normalmente es una buena idea eliminar (**DROP**) todo aquello que se encuentre en este estado.

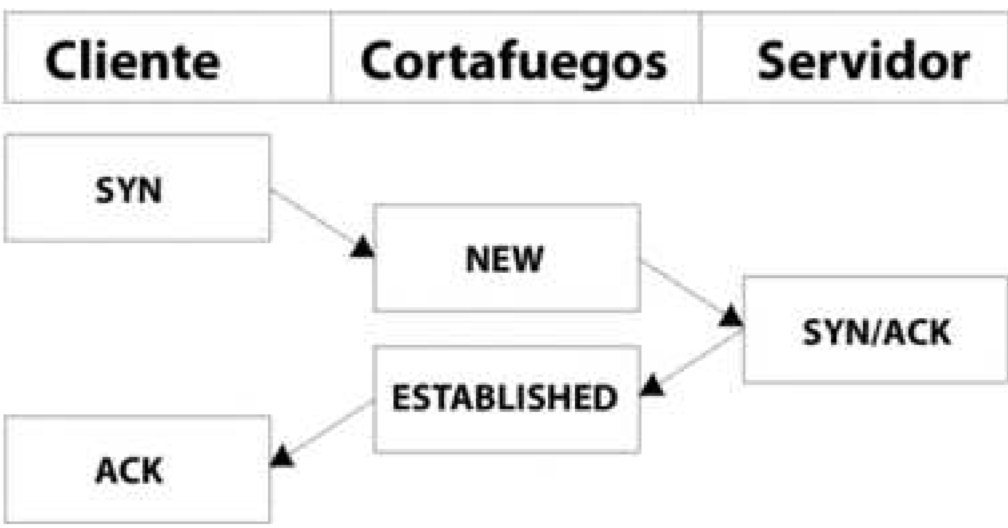
Los anteriores estados pueden usarse conjuntamente con la comparación **--state** para diferenciar paquetes en función de su estado en el seguimiento de conexiones. Ésto es lo que hace que la máquina de estados sea tan increíblemente fuerte y eficiente para nuestro cortafuegos. En el pasado, frecuentemente nos veíamos obligados a abrir todos los puertos por encima del 1024 para permitir el tráfico de retorno a nuestra red local. Con la máquina de estados funcionando ya no es necesario, pues podemos abrir el cortafuegos sólo para el tráfico de retorno, no para todo tipo de tráfico.

Conexiones TCP

En esta sección y las siguientes vamos a repasar en profundidad los estados y cómo son gestionados para cada uno de los tres protocolos básicos: *TCP*, *UDP* e *ICMP*. Asimismo, veremos cómo se gestionan por defecto las conexiones si no se pueden incluir en ninguno de los anteriores protocolos. Empezaremos con el protocolo *TCP*, ya que es un protocolo de "flujos" en sí mismo (stateful protocol: como ya se ha explicado en la introducción, "stateful" implicaría que todos los paquetes de un mismo flujo son considerados como la misma cosa, como un todo que llega "en porciones"); además contiene muchos detalles interesantes respecto a la máquina de estados en iptables.

Una conexión *TCP* siempre se inicia con el "apretón de manos en tres pasos" (3-way handshake), que establece y negocia la conexión sobre la que se enviarán los datos. La sesión entera se inicia con un paquete *SYN*, seguido por un paquete *SYN/ACK* y finalmente por un paquete *ACK*, para confirmar el establecimiento de la sesión (1-"hola, ¿quieres hablar conmigo?", 2-"de acuerdo", 3-"bien, pues empecemos"). En este momento la conexión se establece y está preparada para empezar a enviar datos. El gran problema es: ¿cómo maneja el seguimiento de conexiones todo este tráfico? En realidad de una forma bastante simple.

Al menos en lo que concierne al usuario, el seguimiento de conexiones trabaja más o menos de la misma manera para todo tipo de conexiones. Échale un vistazo al gráfico de abajo para ver exactamente en qué estado entra el flujo durante las diferentes fases de la conexión. Como puedes observar, desde el punto de vista del usuario el código del seguimiento de conexiones realmente no acompaña al flujo de la conexión *TCP*. Una vez ha visto un paquete (el *SYN*), considera la conexión como nueva (**NEW**). En cuanto ve el paquete de retorno (*SYN/ACK*), considera la conexión como establecida (**ESTABLISHED**). Si piensas en éllo un momento, entenderás por qué: con esta particular forma de trabajar puedes permitir a los paquetes **NEW** y **ESTABLISHED** que abandonen tu red local, pero permitiendo únicamente a las conexiones **ESTABLISHED** que vuelvan a la red local y funcionen correctamente. Por el contrario, si la máquina de seguimiento de conexiones tuviera que considerar todo establecimiento de conexión como **NEW**, no seríamos capaces de detener los intentos de conexión desde el exterior hacia nuestra red local, puesto que tendríamos que permitir el retorno de todos los paquetes **NEW**. Para complicarlo todo un poco más, existen varios estados internos más dentro del núcleo que se usan para las conexiones *TCP*, pero que no están disponibles en el espacio de usuario. Básicamente siguen los estándares de estados especificados en *RFC 793 - Protocolo de Control de Transmisiones*, páginas 21-23 (en inglés). Los tendremos en cuenta un poco más adelante en esta misma sección.



Como puedes ver, desde el punto de vista del usuario es bastante simple. Sin embargo, desde el punto de vista del núcleo el esquema es un poco más complejo. Veamos un ejemplo que ilustra a la perfección cómo cambian los estados de la conexión en la tabla /proc/net/ip_conntrack. El primer estado es registrado a la recepción del primer paquete *SYN* de una conexión.

```
tcp      6 117 SYN_SENT src=192.168.1.5 dst=192.168.1.35 sport=1031 \
     dport=23 [UNREPLIED] src=192.168.1.35 dst=192.168.1.5 sport=23 \
     dport=1031 use=1
```

Como puedes ver en la entrada anterior, tenemos un estado concreto en el cual un paquete SYN ha sido enviado (se establece la bandera ***SYN_SENT***) y que todavía no ha recibido ninguna respuesta (de ahí la bandera ***[UNREPLIED]***). El siguiente estado interno se alcanzará cuando se vea un paquete en la otra dirección.

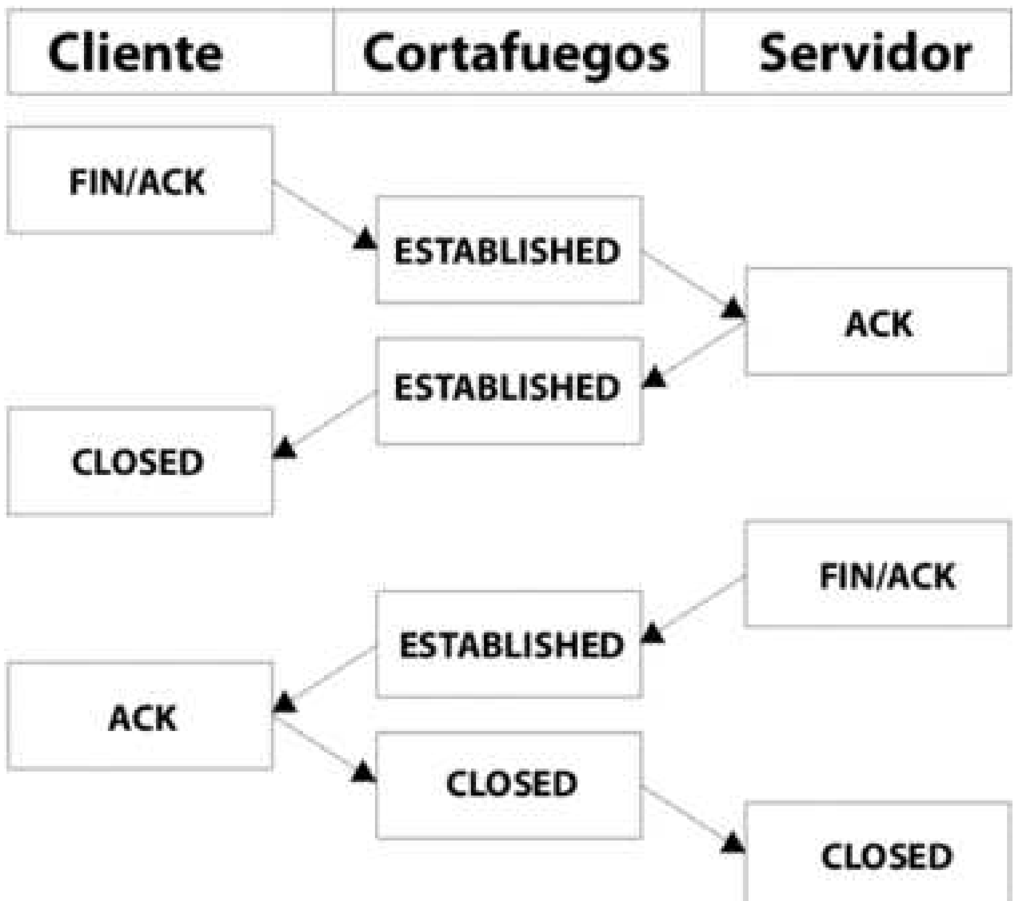
```
tcp      6 57 SYN_RECV src=192.168.1.5 dst=192.168.1.35 sport=1031 \
     dport=23 src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 \
     use=1
```

En este caso hemos recibido el correspondiente paquete *SYN/ACK* como respuesta. En cuanto llega, el estado cambia una vez más, esta vez a ***SYN_RECV***: indica que el paquete *SYN* original llegó correctamente y que el paquete *SYN/ACK* de retorno también ha atravesado correctamente el cortafuegos. Además, esta entrada del seguimiento de conexiones ya ha visto tráfico en ambas direcciones y por éllo se considera que ha obtenido respuesta. Este hecho no es explícito, pero se considera asumido de la misma forma que ocurría con la bandera ***[UNREPLIED]*** anterior. El último paso se dará cuando veamos el paquete *ACK* final del "apretón de manos" (3-way handshake).

```
tcp      6 431999 ESTABLISHED src=192.168.1.5 dst=192.168.1.35 \
     sport=1031 dport=23 src=192.168.1.35 dst=192.168.1.5 \
     sport=23 dport=1031 use=1
```

En este último ejemplo ya hemos visto el paquete *ACK* final y la conexión a entrado en el estado **ESTABLISHED** (establecida), al menos hasta donde llega el control de los mecanismos internos de iptables. Después de unos pocos paquetes más, la conexión se convertirá en ***[ASSURED]*** ("asegurada"), tal como se ha mostrado en la introducción del capítulo.

Cuando una conexión *TCP* se cierra, lo hace de la siguiente manera y tomando los siguientes estados:



Como puedes ver, en realidad la conexión nunca se cierra hasta que se envía el último paquete *ACK*. Ten en cuenta que este gráfico sólo muestra cómo se cierra una conexión en circunstancias normales. También puede cerrarse enviando un paquete *RST* ("reset", reiniciar) si la conexión se tiene que rechazar. En este caso la conexión será cerrada tras un periodo de tiempo predeterminado.

En condiciones normales, cuando una conexión *TCP* se cierra, entra en el estado ***TIME_WAIT*** ("tiempo de espera"), que por defecto es de 2 minutos. Este lapso de tiempo se emplea para que todos los paquetes que se han quedado "atascados" de alguna manera puedan atravesar igualmente el conjunto de reglas, incluso después de que la conexión se haya cerrado; de esta forma se dispone de una especie de "buffer"/colchón de tiempo para que los paquetes que se han quedado parados en algún enrutador congestionado, puedan llegar al cortafuegos o al otro extremo de la conexión sin problemas.

Si la conexión se reinicia por un paquete *RST*, el estado cambia a ***CLOSE*** ("cerrar"). Esta orden implica que la conexión por defecto dispone de 10 segundos antes de que se cierre definitivamente. Los paquetes *RST* no piden consentimiento de ninguna clase y cortan la conexión directamente.

También hay otros estados además de los que ya se han comentado. A continuación tienes la lista completa de los posibles estados que puede tomar un flujo *TCP* y sus tiempos límites.

Table 4-2. Estados internos

Estado	Tiempo límite
NONE	30 minutos
ESTABLISHED	5 días
SYN_SENT	2 minutos
SYN_RECV	60 segundos

Estado	**Tiempo límite**
FIN_WAIT	2 minutos
TIME_WAIT	2 minutos
CLOSE	10 segundos
CLOSE_WAIT	12 horas
LAST_ACK	30 segundos
LISTEN>	2 minutos

Estos tiempos no son de ninguna manera definitivos, ya que pueden cambiar con las revisiones del núcleo, además de poderse cambiar a través del sistema de ficheros proc mediante las variables /proc/sys/net/ipv4/netfilter/ip_ct_tcp_*. Sin embargo, en la práctica los valores por defecto deberían estar bastante bien definidos. Los tiempos se indican en "jiffies" (centésimas de segundo), es decir, 3000 significa 30 segundos.

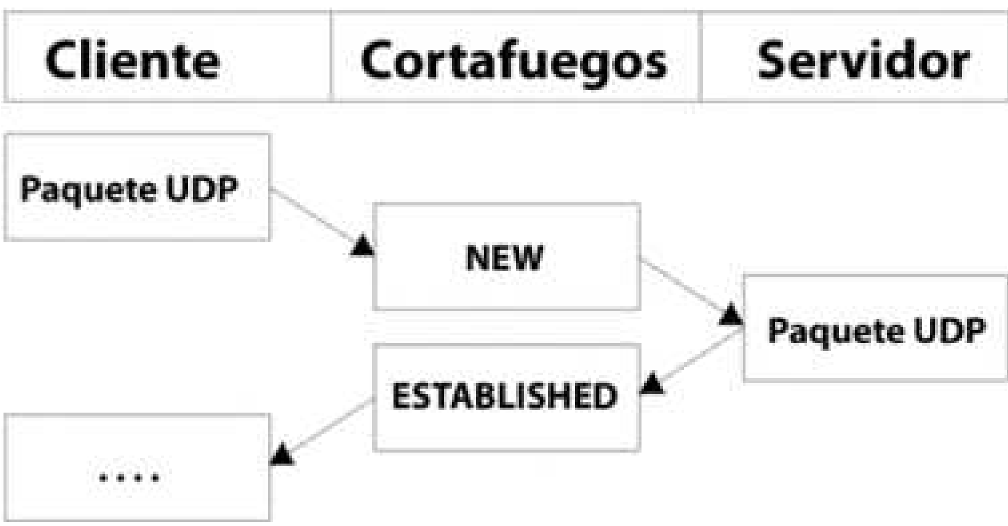
Note

Ten en cuenta que la parte del espacio de usuario de la máquina de estados no se fija en las banderas *TCP* establecidas en los paquetes *TCP*. Hacer éso normalmente es una mala idea, ya que puedes querer permitir que los paquetes en el estado **NEW** puedan atravesar el cortafuegos, pero cuando especificas la bandera **NEW** lo que en la mayoría de las ocasiones quieres permitir son los paquetes *SYN*.

En la implementación actual de los estados no ocurre ésto, ya que incluso un paquete sin ningún valor establecido o sin una bandera *ACK*, se considerará como **NEW**. Este tipo de comparación puede ser útil en sistemas con cortafuegos redundantes, pero en general es una malísima idea en tu red personal, dónde sólo tienes un cortafuegos. Para evitar este comportamiento puedes usar el comando explicado en la sección *Paquetes cuyo estado es NEW pero cuyo bit SYN no se ha establecido* del apéndice *Problemas y preguntas frecuentes*. Otra forma de conseguirlo es instalando la extensión **tcp-window-tracking** que encontrarás en el **patch-o-matic**, que permitirá que el cortafuegos sea capaz de hacer el seguimiento de los estados en función de las configuraciones de las ventanas *TCP*.

Conexiones UDP

Las conexiones *UDP* son en sí mismas "conexiones sin flujo". Existen varias razones para éllo, principalmente porque no implican ningún establecimiento o cierre de conexión; más que nada les falta algún tipo de secuenciamiento: recibir dos datagramas *UDP* en un orden específico no dice nada acerca del orden en que fueron enviados. Sin embargo es posible establecer estados en las conexiones dentro del núcleo. Veamos cómo se puede seguir una conexión y cómo podría verse en el conntrack.



Como puedes observar, desde el punto de vista del espacio de usuario la conexión se inicia casi exactamente de la misma forma que una conexión *TCP*. Internamente, sin embargo, la información del conntrack es bastante diferente, pero intrínsecamente los detalles son los mismos. Para empezar, veamos la entrada tras haber enviado el paquete *UDP* inicial.

```
udp      17 20 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025 \
     [UNREPLIED] src=192.168.1.5 dst=192.168.1.2 sport=1025 \
     dport=137 use=1
```

Como puedes ver en los dos primeros valores, se trata de un paquete *UDP*. El primer valor es el nombre del protocolo, mientras que el segundo es el número del protocolo. Es lo mismo que se encuentra en las conexiones *TCP*. El tercer valor indica el "tiempo de vida" en segundos que le queda a esta entrada de estado. Tras ésto vienen los valores de origen y destino del paquete que hemos visto, la bandera ***[UNREPLIED]***, que nos indica que hasta el momento no ha habido respuesta al paquete, y por fin un listado de los valores que se esperan en los paquetes de respuesta. Estos últimos valores son los mismos que antes pero en sentido inverso. El tiempo de vida ("timeout") por defecto es de 30 segundos.

```
udp      17 170 src=192.168.1.2 dst=192.168.1.5 sport=137 \
     dport=1025 src=192.168.1.5 dst=192.168.1.2 sport=1025 \
     dport=137 use=1
```

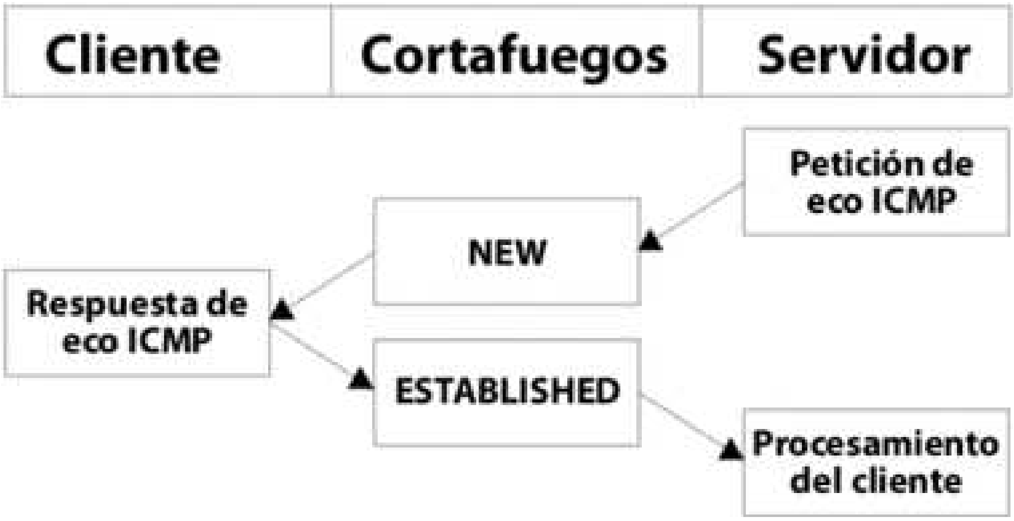
En este ejemplo el servidor ya ha visto una respuesta al paquete inicial y la conexión se considera **ESTABLISHED** (establecida). Este detalle no se muestra en el seguimiento de conexiones, como puedes ver. La diferencia principal es la ausencia de la bandera ***[UNREPLIED]***. Además el tiempo de vida por defecto a cambiado a 180 segundos: en el ejemplo ya han pasado 10 segundos y por eso el valor es 170; en diez segundos más el valor disminuirá a 160 y así hasta que se agote el tiempo. Sin embargo, hay un detalle que se ha visto antes y que ahora falta: la bandera ***[ASSURED]***. Para que se establezca esta bandera en el seguimiento de conexiones, debe haber habido un mínimo de tráfico en la conexión.

```
udp      17 175 src=192.168.1.5 dst=195.22.79.2 sport=1025 \
     dport=53 src=195.22.79.2 dst=192.168.1.5 sport=53 \
     dport=1025 [ASSURED] use=1
```

En este momento la conexión ya está asegurada. Este ejemplo parece exactamente igual que el anterior, excepto por la bandera ***[ASSURED]***. Si la conexión no es utilizada en 180 segundos, entonces caduca. Este tiempo (180 segundos) es un valor relativamente bajo, pero debería ser suficiente para la mayoría de los casos. Cada vez que un paquete coincide con los valores indicados en el paquete inicial (los valores que se esperan de los paquetes de respuesta ya comentados antes) y atraviesa el cortafuegos, el contador vuelve al valor por defecto, igual que ocurre con el resto de estados internos.

Conexiones ICMP

Los paquetes *ICMP* no pueden estar más lejos de ser conexiones de flujo (o "stateful connections"), ya que sólo se usan para el control de las conexiones y nunca deberían establecer ninguna conexión por sí mismos. Sin embargo, hay cuatro tipos *ICMP* que generarán paquetes de retorno y que presentan 2 estados diferentes: estos mensajes *ICMP* pueden tomar los estados **NEW** (nuevo) y **ESTABLISHED** (establecido). Los tipos *ICMP* de los que hablamos son: *Echo request* (petición de eco) y su *reply* (respuesta o *Echo reply*), *Timestamp request* (petición de "marca de tiempo", el valor del momento exacto en que se envía el paquete) y su *reply*, *Information request* (petición de información) y su *reply*, y por último *Address mask request* (petición de máscara de subred) y su *reply*. De éstos, los tipos *timestamp request* e *information request* están obsoletos y lo más probable es que simplemente se puedan desechar. Sin embargo, los mensajes de *Echo* se emplean en varias configuraciones, como hacer "pings" a los hosts. Las peticiones de máscara de subred (*Address mask requests*) no se suelen utilizar, pero pueden ser útiles en ocasiones y vale la pena permitirles que atraviesen el cortafuegos. Para tener una idea de cómo funciona una conexión de este tipo, veamos la siguiente imagen:



Como puedes observar, el host envía una petición de eco (*echo request*) al destinatario y el cortafuegos considera el paquete como nuevo (**NEW**). El destinatario responde con una respuesta de eco (*echo reply*) que el cortafuegos considera mediante el estado "establecido" (**ESTABLISHED**). Cuando ya se ha visto la primera petición de eco, se crea la siguiente entrada en el ip_conntrack.

```
icmp     1 25 src=192.168.1.6 dst=192.168.1.10 type=8 code=0 \
     id=33029 [UNREPLIED] src=192.168.1.10 dst=192.168.1.6 \
     type=0 code=0 id=33029 use=1
```

Como ya te habrás dado cuenta, esta entrada es un poco diferente a las entradas estándar para los paquetes *TCP* y *UDP*. Vemos el protocolo, su número, el tiempo de vida que le queda a la entrada, así como las direcciones de origen y destino. La diferencia viene después: tenemos 3 campos nuevos llamados ***type*** (tipo), ***code*** (código) e ***id*** (identidad/identificación). En realidad no son nada especiales: el valor ***type*** indica el tipo *ICMP* y el valor ***code*** indica el código *ICMP*. Todos éllos están disponibles en el apéndice *Tipos ICMP*. El valor de ***id*** indica el *ICMP ID*: cada paquete *ICMP* toma un valor ID (de identificación) cuando se envía y al responder el receptor también establece este mismo ID; de esta forma al llegar la respuesta al host que envió la petición se puede unir esa respuesta a la petición *ICMP* correcta.

En el siguiente campo volvemos a encontrar la bandera ***[UNREPLIED]*** que hemos visto anteriormente. Como entonces, la entrada del seguimiento de conexiones que estamos viendo implica que sólo se ha visto tráfico en una dirección. Por último vemos los datos que se esperan en el paquete de respuesta *ICMP*, que obviamente son los valores iniciales pero invertidos. Por lo que respecta al tipo y al código, se cambian por los valores adecuados al paquete de retorno, de forma que una petición de eco se transforma en una respuesta de eco, etc. El *ICMP ID*, lógicamente, se mantiene idéntico al paquete de petición.

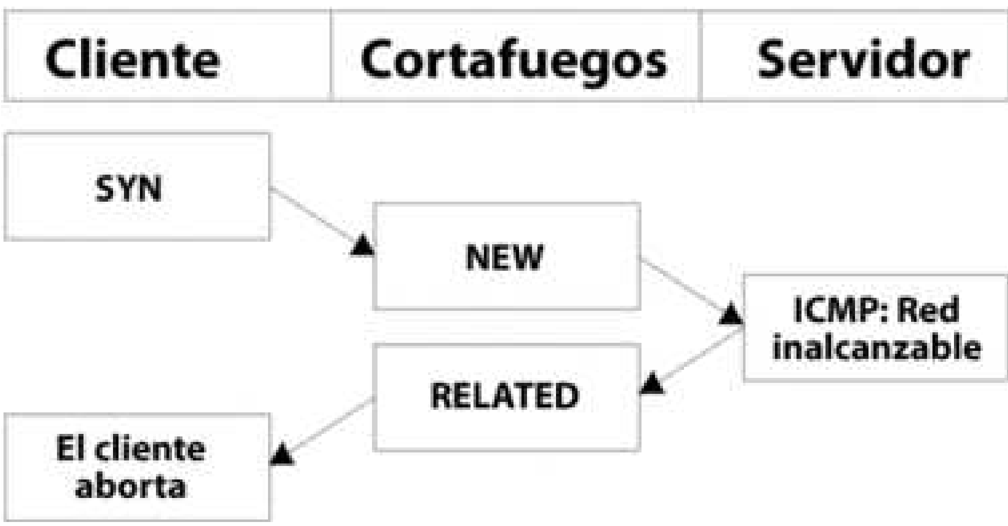
El paquete de respuesta se considera como **ESTABLISHED**, tal como ya se ha explicado. Sin embargo, sabemos con seguridad que tras la respuesta *ICMP* no habrá absolutamente ningún tráfico legal en la misma conexión. Por esta razón la entrada del seguimiento de conexiones es destruida en cuanto la respuesta ha atravesado la estructura de Netfilter.

En cada uno de los casos anteriores la petición se considera como nueva (**NEW**), mientras que la respuesta se considera como establecida (**ESTABLISHED**). Más exactamente, cuando el cortafuegos ve un paquete de petición lo considera como nuevo, mientras que cuando el host envía un paquete de respuesta a la petición, se considera como establecido.

> **Note** Ésto implica que el paquete de respuesta debe seguir el criterio fijado en la entrada del seguimiento de conexiones, para poder ser considerado como establecido, de la misma manera que ocurre en el resto de tipos de tráfico.

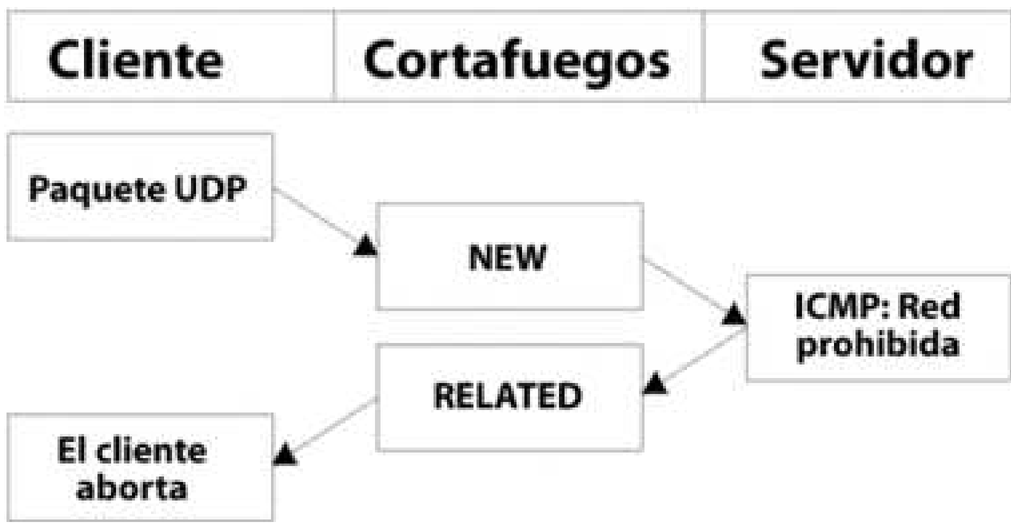
Las peticiones ICMP tienen un tiempo de vida por defecto de 30 segundos, que puedes cambiar en la entrada /proc/sys/net/ipv4/netfilter/ip_ct_icmp_timeout. En general éste es un buen valor, ya que permitirá captar la mayoría de paquetes en tránsito.

Otra parte importantísima de *ICMP* es el hecho de que se emplea para indicarle a los hosts qué ha pasado con las conexiones *UDP* y *TCP* o con los intentos de conexión. Por este motivo las respuestas ICMP muy a menudo serán reconocidas como relacionadas (**RELATED**) con conexiones o intentos de conexión. Un ejemplo sencillo sería el de los mensajes *ICMP Host unreachable* (mensaje de host inalcanzable, no se puede llegar al host) e *ICMP Network unreachable* (mensaje de red inalcanzable, no se puede llegar hasta la red). Este tipo de mensajes se envían automáticamente de vuelta a nuestro host cuando el paquete no consigue efectuar la conexión con un host porque la red o el host de destino están fuera de servicio, de manera que el último enrutador (router) que intente conectar con el destino nos contestará con un mensaje *ICMP* diciéndonos lo que ocurre. En este caso la respuesta *ICMP* se considera como un paquete relacionado (**RELATED**). El siguiente gráfico debería aclarar lo que ocurre:



En el ejemplo enviamos un paquete *SYN* a una dirección concreta: ésto es considerado como una nueva conexión (**NEW**) por el cortafuegos. Sin embargo no se puede llegar a la red a la que se está intentando enviar el paquete, por lo que un router (el último) nos devuelve un error *ICMP* de red inalcanzable. El código de seguimiento de conexiones puede reconocer este paquete como relacionado (**RELATED**) gracias a la entrada ya existente, por lo que la respuesta *ICMP* es correctamente enviada al cliente que generó el paquete inicial y, si todo va bien, éste abortará la conexión. Mientras tanto el cortafuegos ya habrá eliminado la entrada del seguimiento de conexiones, puesto que sabe que se trata de un mensaje de error.

El mismo comportamiento se experimenta con las conexiones *UDP* si tienen problemas como en el caso anterior. Todo mensaje *ICMP* devuelto como respuesta a conexiones *UDP* es considerado como relacionado (**RELATED**). Observa el siguiente gráfico:



En esta ocasión se envía un paquete *UDP* al host. Esta conexión *UDP* se considera como nueva **NEW**. Sin embargo la red está prohibida administrativamente por algún cortafuegos o router antes de llegar a destino y por éllo nuestro cortafuegos recibe un mensaje *ICMP Network Prohibited* (red prohibida). El cortafuegos sabe que este mensaje *ICMP* de error está relacionado con la conexión *UDP* abierta y lo envía como un paquete **RELATED** (relacionado) al cliente. Acto seguido, elimina la entrada del seguimiento de conexiones mientras el cliente recibe el mensaje y aborta la conexión.

Conexiones por defecto

En algunos casos el conntrack no sabe cómo gestionar un protocolo específico, bien sea porque no conoce el protocolo o porque no sabe cómo funciona. En estos casos sigue un comportamiento por defecto, como ocurre por ejemplo con *NETBLT*, *MUX* y *EGP*. El procedimiento seguido es básicamente el mismo que en el seguimiento de las conexiones *UDP*: el primer paquete se considera como nuevo (**NEW**) y el tráfico de respuesta y subsiguiente es considerado como establecido (**ESTABLISHED**).

Cuando se emplea el comportamiento por defecto todos los paquetes afectados tendrán el mismo "tiempo de vida" (timeout) por defecto, que se puede establecer a través de la variable /proc/sys/net/ipv4/netfilter/ip_ct_generic_timeout. El valor por defecto es de 600 segundos (10 minutos). En función del tipo de tráfico que estés intentando enviar a través de un enlace que utilice este comportamiento por defecto, es posible que necesites cambiarlo, especialmente si estás conectado a través de satélite o algo similar, pues la comunicación puede llegar a tardar bastante tiempo.

Los protocolos complejos y el seguimiento de conexiones

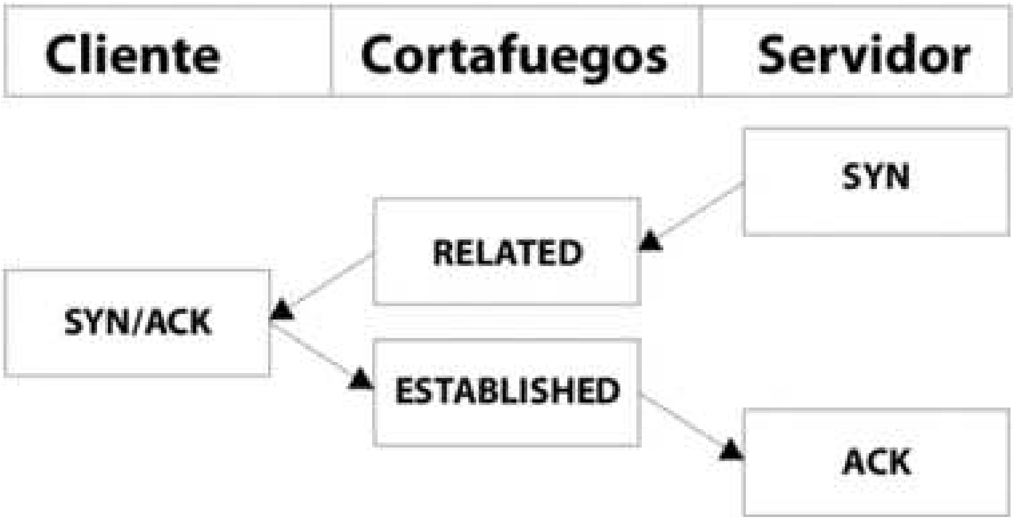
Determinados protocolos son más complejos que otros, lo cual implica que a la hora de realizar el seguimiento de conexiones el trabajo será más difícil. Buenos ejemplos de éllo son los protocolos *ICQ*, *IRC* y *FTP*. Cada uno de éllos incluye información dentro del bloque de datos de los paquetes, por lo que para funcionar correctamente requieren de asistentes especiales para el seguimiento de conexiones.

Empecemos por ejemplo con el procolo *FTP*. Este protocolo comienza abriendo una única conexión llamada sesión "*FTP control*" (sesión de control ftp). En cuanto enviamos comandos a través de esta sesión, se abren otros puertos para transportar el resto de datos relacionados con esos comandos. Estas conexiones se pueden crear de dos maneras:

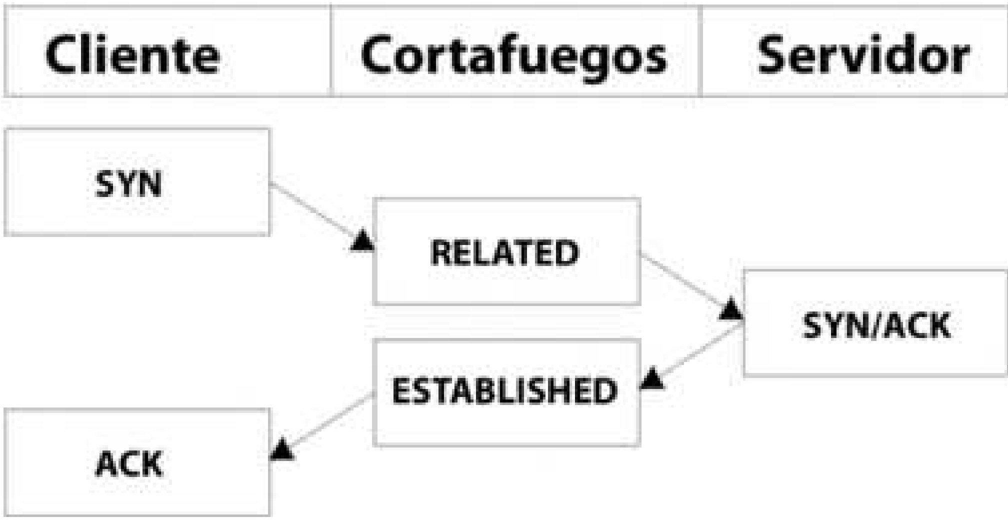
activamente o pasivamente. Cuando una conexión se genera activamente, el cliente *FTP* envía al servidor un puerto y una dirección IP a los que conectarse. Tras ésto el cliente *FTP* abre el puerto indicado y el servidor conecta con él desde su propio puerto 20 (conocido como "FTP-Data"), enviando datos a través de él.

El problema radica en que el cortafuegos no tiene conocimiento de esas conexiones extras, ya que son negociadas a través del bloque de datos que transporta el paquete (lo que se conoce como "payload" o carga de información; recordemos que a grosso modo un paquete IP tiene una cabecera con información sobre la conexión y un bloque de datos que es lo que desea el host que lo recibe). Debido a ésto el cortafuegos no será capaz de saber que debería dejar conectarse al servidor con el cliente a través de estos puertos concretos.

La solución al problema pasa por añadir un asistente especial al módulo del seguimiento de conexiones, que escaneará los datos de la conexión de control para detectar sintaxis e información específicas: cuando encuentre la información adecuada, la añadirá en una nueva entrada etiquetada como relacionada (**RELATED**) y gracias a esta nueva entrada el servidor ya será capaz de efectuar un seguimiento de la conexión. Observa la siguiente imagen para entender los estados cuando el servidor *FTP* ya ha creado la conexión con el cliente.



El ftp pasivo ("*Passive FTP*") funciona completamente a la inversa: el cliente *FTP* le indica al servidor que quiere determinados datos, a lo que el servidor responde con una dirección IP y un puerto a los que conectarse. Tras recibir estos datos, el cliente se conectará a ese puerto desde su propio puerto 20 (el puerto FTP-data) y cogerá los datos en cuestión. Si tienes un servidor *FTP* tras tu cortafuegos, necesitarás de este módulo además de los módulos estándar de iptables para permitir que los clientes desde Internet puedan conectar correctamente con tu servidor *FTP*. De igual forma necesitarás el módulo si eres extremadamente restrictivo con tus usuarios y sólo quieres dejar que utilicen servidores *HTTP* y *FTP* de Internet, bloqueando cualquier otro puerto. Observa la siguiente imagen y su relación con el ftp pasivo ("Passive *FTP*").



Algunos asistentes del conntrack ya están disponibles en el núcleo, más concretamente, en el momento de escribir estas líneas los protocolos *FTP* e *IRC* ya disponen de estos asistentes. Si no puedes encontrar en el núcleo los asistentes que necesitas, debes echar un vistazo al *patch-o-matic*, en la sección de zona de usuario de iptables: podrás encontrar más asistentes del conntrack, como es el caso de los protocolos *ntalk* o *H.323*. Si no están disponibles en el patch-o-matic, todavía dispones de algunas opciones más, como el CVS ("Concurrent Versioning System") de iptables, si recientemente a entrado en él. O también puedes ponerte en contacto con la lista de correo Netfilter-devel y preguntar si está disponible. Si no lo está y no está planeado añadirlo, te quedas solo y lo más probable es que quieras leer el "CÓMO" de Rusty Russell: Rusty Russell's Unreliable Netfilter Hacking HOW-TO (en inglés), del cual tienes un enlace en el apéndice *Otras fuentes y enlaces*.

Los asistentes del conntrack pueden ser compilados estáticamente o como módulos en el núcleo. Si se compilan como módulos puedes cargarlos con el siguiente comando:

modprobe ip_conntrack_*

Ten en cuenta que el seguimiento de conexiones no tiene nada que ver con la traducción de direcciones (*NAT*), por lo que es posible que necesites más módulos si también estás "NATeando" (traduciendo) las conexiones. Por ejemplo, si quisieras traducir las direcciones y realizar un seguimiento de las conexiones *FTP*, necesitarías también el módulo *NAT*. Todos los asistentes *NAT* tienen el mismo prefijo en su nombre: "ip_nat_", seguido por el nombre del asistente en cuestión. Por ejemplo, el asistente para *FTP* con *NAT* (traducción de direcciones) se llamaría *ip_nat_ftp* y el módulo *IRC* se llamaría *ip_nat_irc*. Los asistentes del conntrack siguen la misma regla de nomenclatura, por lo que el asistente del conntrack para *IRC* se llamaría *ip_conntrack_irc*, mientras que el asistente para *FTP* sería *ip_conntrack_ftp*.

Chapter 5. Salvando y restaurando grandes conjuntos de reglas

El paquete **iptables** tiene dos herramientas muy útiles, especialmente cuando te enfrentas a conjuntos de reglas muy grandes. Estas herramientas son **iptables-save** e **iptables-restore**, y se usan respectivamente para salvar y restaurar conjuntos de reglas en un formato de fichero específico, que parece bastante diferente del código del intérprete de comandos (shell) estándar que puedes ver en el resto del tutorial.

Considerando la velocidad

Una de las razones más convincentes para utilizar los comandos **iptables-save** e **iptables-restore** es que aceleran considerablemente la carga y la copia de los conjuntos de reglas más grandes. El principal problema al ejecutar un script en la línea de comandos que contenga reglas de **iptables**, es que cada vez que se invoca a **iptables** en el script se procede en primer lugar a extraer el conjunto de reglas completo del espacio del núcleo de Netfilter, para a continuación insertar, añadir o efectuar el cambio que sea necesario por el comando específico. Por último, devolverá el conjunto de reglas desde su propia memoria al espacio del núcleo. Utilizando un script, estas operaciones se ejecutan en cada una de las reglas que queramos insertar y consecuentemente cada vez cuesta más extraer e insertar el conjunto de reglas, pues cada vez hay más reglas en el conjunto.

Para resolver este problema se han creado los comandos **iptables-save** e **iptables-restore**. El primero (**iptables-save**) se usa para guardar el conjunto de reglas en un fichero de texto con un formato especial, mientras que **iptables-restore** carga ese fichero de texto de nuevo en el núcleo. Lo mejor de todo ésto es que cargan y guardan el conjunto de reglas mediante una sóla petición: **iptables-save** captará el conjunto de reglas completo del núcleo y lo guardará en un fichero en un sólo paso; **iptables-restore** cargará en el núcleo un conjunto de reglas específico en un paso por cada tabla. O sea, en vez de copiar el conjunto de reglas desde el núcleo, que en un ejemplo grande podría significar repetir el proceso 30.000 veces (si hubieran 30.000 reglas) y después volverlo a cargar en el núcleo otras tantas veces, podremos copiar desde el núcleo de una vez y volver a cargar en el núcleo en 3 pasos (suponiendo que hayan sólo 3 tablas).

Como puedes comprender, estas herramientas son exactamente lo que necesitas si estás trabajando con una gran cantidad de reglas y debes insertarlas en el conjunto de reglas. Sin embargo tienen sus inconvenientes, como discutiremos a continuación.

Inconvenientes con la restauración

Es posible que te estés preguntando si **iptables-restore** soporta de alguna manera los scripts. Y la respuesta es que de momento no, y lo más probable es que nunca lo haga. Este es el principal problema derivado del uso de **iptables-restore**, puesto que no podrás hacer una gran cantidad de tareas con estos ficheros. Por ejemplo: ¿qué ocurre cuando tienes una conexión a la que se le asigna dinamicamente su dirección IP y quieres captar esta dirección cada vez que el ordenador arranca para utilizarla en tus scripts? Con **iptables-restore**, ésto es prácticamente imposible.

Una posibilidad de conseguirlo es crear un pequeño script que primero capte los valores que quieres usar en las reglas, después busque ciertas claves en el fichero creado por **iptables-save** y las sustituya por los valores captados anteriormente. Una vez llegados a este punto, puedes salvar el fichero modificado en un fichero temporal y utilizar el comando **iptables-restore** para cargar los nuevos valores. Sin embargo ésta manera de proceder causa un montón de problemas y no podrás utilizar correctamente **iptables-save**, pues probablemente eliminará los valores añadidos manualmente (por el script). En definitiva es una solución, cuando menos, poco acertada.

Otra posible solución sería cargar los ficheros con **iptables-restore** y a continuación ejecutar un script que inserte reglas más dinámicas en los lugares adecuados. Por descontado, esta manera de proceder es tan poco acertada como la anterior. La cuestión de fondo es que **iptables-restore** no está adaptado para aquellas configuraciones dónde las direcciones IP se asignan al cortafuegos dinámicamente, o dónde deseas diferentes comportamientos dependiendo de los valores de determinadas opciones de configuración.

Otro inconveniente de **iptables-restore** e **iptables-save** es que de momento no son plenamente funcionales. El problema radica en los pocos usuarios que utilizan estos comandos y por tanto son pocos los que buscan fallos, de forma que es posible que algunas comparaciones y objetivos se inserten mal, y ésto lleve a comportamientos extraños o no esperados. A pesar de todo, aunque existen estos problemas, recomendaría encarecidamente utilizar estas herramientas, ya que deberían funcionar perfectamente bien para la mayoría de conjuntos de reglas, siempre que no contengan algunos de los nuevos objetivos o comparaciones que no sabe cómo manejar convenientemente. En un par de frases: si tus conjuntos de reglas no están absolutamente "a la última", estas herramientas funcionan como caídas del cielo.

iptables-save

El comando **iptables-save** es una herramienta para guardar el conjunto de reglas existente en iptables a un fichero que puede utilizar **iptables-restore**. Este comando es bastante fácil de usar y sólo tiene dos argumentos. Échale un vistazo al siguiente ejemplo para entender la sintaxis apropiada del comando.

iptables-save [-c] [-t *tabla*]

La opción **-c** le indica a **iptables-save** que guarde también los valores existentes en los contadores de bytes y de paquetes. Ésto puede ser útil si queremos reiniciar el cortafuegos sin perder los valores de estos contadores, que servirán, por ejemplo, para continuar con nuestras rutinas estadísticas sin problemas. Por supuesto el valor por defecto es no conservar los datos de los contadores.

La opción **-t** indica a **iptables-save** qué tablas guardar. Sin este argumento el comando guardará en el fichero todas las tablas disponibles. A continuación tienes un ejemplo de la salida que puedes esperar del comando **iptables-save** si no tienes ningún conjunto de reglas cargado (lógicamente la salida es en inglés).

```
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:17 2002
*filter
:INPUT ACCEPT [404:19766]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [530:43376]
COMMIT
# Completed on Wed Apr 24 10:19:17 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:17 2002
*mangle
:PREROUTING ACCEPT [451:22060]
:INPUT ACCEPT [451:22060]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [594:47151]
:POSTROUTING ACCEPT [594:47151]
COMMIT
# Completed on Wed Apr 24 10:19:17 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:17 2002
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [3:450]
:OUTPUT ACCEPT [3:450]
COMMIT
# Completed on Wed Apr 24 10:19:17 2002
```

Como puedes ver los comentarios comienzan con el signo #. Cada tabla se marca así: *<nombre-tabla>, como por ejemplo ***mangle**. A continuación, dentro de cada tabla se encuentran las especificaciones de las cadenas y las reglas. Una especificación de cadena es similar a: :<nombre-cadena> <política-cadena> [<contador-paquetes>:<contador-bytes>]. El ***nombre de cadena*** puede ser, por ej., *PREROUTING*, la política ya se ha descrito antes y puede ser *ACCEPT*. Por otra parte, los contadores de paquetes y de bytes son los mismos que obtenemos con **iptables -L -v**. Por último, cada declaración de tabla finaliza con la clave ***COMMIT***. Esta clave nos indica que en ese punto debemos enviar al núcleo todas las reglas que se han ido leyendo.

El ejemplo anterior es bastante básico y por éllo considero que es apropiado mostrar un breve ejemplo que contiene un pequeño *Iptables-save*. Si quisiéramos ejecutar **iptables-save** con éste conjunto de reglas cargado en el núcleo, la salida sería algo similar a (otra vez, en inglés):

```
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*filter
:INPUT DROP [1:229]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth1 -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*mangle
:PREROUTING ACCEPT [658:32445]
:INPUT ACCEPT [658:32445]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [891:68234]
:POSTROUTING ACCEPT [891:68234]
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
```

```
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*nat
:PREROUTING ACCEPT [1:229]
:POSTROUTING ACCEPT [3:450]
:OUTPUT ACCEPT [3:450]
-A POSTROUTING -o eth0 -j SNAT --to-source 195.233.192.1
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
```

Como puedes ver, a cada comando se le ha añadido como prefijo los contadores de bytes y paquetes, puesto que se ha empleado la opción **-c** en el comando. Excepto éste detalle, la línea de comandos es casi igual al script. El único problema ahora es cómo guardar la salida del comando en un fichero, aunque la solución es bastante simple y ya deberías conocerla si has trabajado con Linux antes: sólo se trata de dirigir la salida del comando hacia un fichero con el nombre que desees. Ésto puede parecerse a:

iptables-save -c > /etc/iptables-save

El comando anterior guardará el conjunto de reglas con los valores de sus contadores en un fichero llamado /etc/iptables-save.

iptables-restore

El comando **iptables-restore** se emplea para volver a cargar en el núcleo el conjunto de reglas guardado con **iptables-save**. Sin embargo, por ahora carga toda la información desde la entrada estándar y no desde un fichero. La sintaxis es:

iptables-restore [-c] [-n]

El argumento **-c** reestablece los contadores de bytes y paquetes y es la opción que debes usar cuando quieras volver a cargar los valores guardados con **iptables-save** de estos contadores. La opción también puede escribirse en su forma extendida: **--counters**.

El argumento **-n** le indica a **iptables-restore** que no sobreescriba las reglas existentes en la tabla o tablas en que esté escribiendo. El comportamiento por defecto de **iptables-restore** es eliminar cualquier regla preexistente. La opción en su "versión larga" sería: **--noflush**.

Para cargar conjuntos de reglas con el comando **iptables-restore** tenemos varias alternativas, aunque veremos la más simple y más utilizada.

cat /etc/iptables-save | iptables-restore -c

Simplificando: con éllo imprimiremos en la salida estándar (la pantalla) el contenido del conjunto de reglas existente en el fichero/etc/iptables-save y a continuación esa salida se dirigirá al comando **iptables-restore**, que captará el conjunto de reglas y lo cargará en el núcleo, incluyendo los valores de los contadores de bytes y paquetes. Ésta es la forma más sencilla de comenzar a usar el comando, aunque el ejemplo anterior se puede modificar hasta la saciedad, con diferentes posibilidades de redireccionamiento de la salida. Sin embargo, ésto se sale del objetivo de éste capítulo y prefiero dejar al lector que experimente por su cuenta.

El conjunto de reglas debería ahora estar cargado correctamente en el núcleo y todo debería funcionar. Si no es así, probablemente has cometido algún fallo al escribir los comandos, o habrás encontrado un "bug".

Chapter 6. Cómo se escribe una regla

En este capítulo se desarrollará en profundidad cómo escribir tus propias reglas. Una regla puede definirse como las instrucciones que seguirá el cortafuegos al bloquear o permitir diferentes conexiones y paquetes en una cadena específica. Cada línea que escribas y añadas a una cadena se debe considerar una regla. También repasaremos las comparaciones básicas que tienes disponibles y cómo utilizarlas, así como los diferentes objetivos disponibles y cómo puedes crear nuevos objetivos (es decir, nuevas sub-cadenas).

Conceptos Básicos

Como ya se ha dicho, cada regla es una línea que lee el núcleo para saber qué hacer con un paquete. Si todos los criterios (o comparaciones) se cumplen, entonces se ejecuta la instrucción objetivo (o salto). Normalmente deberías escribir tus reglas con una estructura similar a ésta:

iptables [-t *tabla*] comando [comparación (match)] [objetivo/salto (target/jump)]

En ningún sitio se especifica que la instrucción objetivo (el salto) deba ser la última función de la línea. Sin embargo, deberías seguir esta estructura para conseguir que sea lo más legible posible. De todas formas, la mayoría de las reglas que veas se han escrito de esta forma. Así pues, si lees un script de otra persona, lo más seguro es que reconozcas la estructura y entiendas fácilmente la regla.

Si quieres utilizar cualquier otra tabla que no sea la estándar, puedes especificarla en la parte que dice [tabla]. Sin embargo, no es necesario especificar qué tabla usar, puesto que por defecto **iptables** utiliza la tabla *filter* (filtro) para incluir todos los comandos. Tampoco es preciso que especifiques la tabla justo en ese punto de la regla. La verdad es que puede estar casi en cualquier parte de la línea. De todas formas, está más o menos aceptado que se especifique la tabla al principio.

Eso sí, debes tener en cuenta que el comando debería ser siempre lo primero, o a lo sumo justo detrás de la especificación de la tabla. Se emplea el "comando" para decirle al programa qué hacer, como por ejemplo insertar o añadir una regla al final de una cadena, o borrar una regla. Veremos ésto en detalle más adelante.

La comparación es la parte de la regla enviada al núcleo que especifica el carácter concreto del paquete, lo cual lo diferencia de todos los demás paquetes. Aquí es dónde se puede determinar de qué dirección IP viene el paquete, de qué interfaz de red, la IP destino, el puerto, el protocolo o cualquier cosa. Hay un montón de comparaciones diferentes que pueden emplearse y que se verán en detalle en este capítulo.

Por último tenemos el objetivo del paquete. Si todas las comparaciones se cumplen para un paquete, le decimos al núcleo qué hacer con él. Por ejemplo, podemos hacer que envíe el paquete a otra cadena que hayamos creado y que es parte de esta tabla. Podemos hacer que deseche el paquete y no haga nada más con él, o podemos enviar una respuesta específica al remitente. También estudiaremos los detalles en este capítulo.

Tablas

La opción **-t** especifica qué tabla utilizar. Por defecto se emplea la tabla *filter*, aunque se puede usar cualquiera de las siguientes tablas (ten en cuenta que este es un corto resumen del capítulo *Atravesando tablas y cadenas*):

Table 6-1. Tablas

Tabla	Descripción

Tabla	Descripción
nat	La tabla *nat* se emplea principalmente para la traducción de direcciones de red (*Network Address Translation*). Los paquetes que son filtrados por esta tabla acaban con sus IPs modificadas, de acuerdo con nuestras reglas (las de la tabla). De todos los paquetes de un mismo flujo, sólo el primero pasa por esta tabla: asumiendo que se permite el paso del primer paquete de un flujo, al resto de paquetes del ese flujo se les aplican automáticamente las mismas acciones realizadas con el primer paquete (sus IPs son "*NAT*-eadas" o filtradas por la tabla *NAT*; o se enmascaran; ...). En otras palabras, sólo el primer paquete pasa por la tabla NAT y al resto del flujo se le trata exactamente igual sin necesidad de pasar por la tabla. Esta es la razón principal por la que no deberías hacer ningún filtrado en esta tabla, y lo comentaremos en detalle más adelante. La cadena *PREROUTING* se emplea para modificar los paquetes en cuanto llegan al cortafuegos. La cadena *OUTPUT* se utiliza para modificar los paquetes generados localmente (es decir, en el cortafuegos) antes de tomar la decisión de enrutado (direccionamiento). Por último la cadena *POSTROUTING* se usa para modificar los paquetes que están a punto de abandonar el cortafuegos.

Tabla	Descripción
mangle	Esta tabla se emplea principalmente para "retocar" paquetes. Entre otras cosas, se puede cambiar el contenido de diferentes paquetes y el de sus cabeceras. Por ejemplo, se pueden cambiar los campos **TTL**, **TOS** o **MARK**. Se debe tener en cuenta, sin embargo, que modificar el campo **MARK** no es realmente un cambio del paquete, pero se establece su valor para el paquete dentro del espacio del núcleo. Otras reglas o programas pueden usar ese valor más adelante para filtrar o efectuar un enrutado avanzado; tc es un ejemplo. La tabla consta de cinco cadenas: *PREROUTING*, *POSTROUTING*, *OUTPUT*, *INPUT* y *FORWARD*. *PREROUTING* se emplea para modificar los paquetes en cuanto llegan al cortafuegos y antes de que alcancen la decisión de enrutado. *POSTROUTING* se emplea para modificar los paquetes en cuanto las decisiones de enrutado se han tomado. *OUTPUT* se emplea para modificar los paquetes generados localmente, antes de que alcancen la decisión de enrutado. *INPUT* se emplea para modificar paquetes que se han redirigido al ordenador local, pero antes de que la aplicación del espacio de usuario llegue a ver la información. *FORWARD* se emplea para modificar paquetes que acaban de pasar por la primera decisión de enrutado, pero antes de que lleguen a la última decisión de enrutado. Ten en cuenta que esta tabla no puede emplearse para cualquier tipo de traducción de direcciones de red (*Network Address Translation*) o enmascaramiento (*Masquerading*): la tabla nat es la que tiene esa función.

Tabla	Descripción
filter	La tabla *filter* se debería emplear exclusivamente para filtrar paquetes. Por ejemplo, se pueden **DROP** (desechar), **LOG** (añadir a un registro de sucesos), **ACCEPT** (aceptar) o **REJECT** (rechazar) paquetes sin problemas, igual que en el resto de tablas. Tenemos tres cadenas en esta tabla: la primera (*FORWARD*) se usa para todos los paquetes que no se generan localmente y que no están destinados a nuestra máquina (el cortafuegos, en otras palabras). *INPUT* se emplea en todos los paquetes que se destinan a nuestra máquina (el cortafuegos), mientras que *OUTPUT* se emplea para todos los paquetes generados localmente.

Todo lo anterior debería haber aclarado los conceptos básicos sobre las tres tablas diferentes que hay disponibles. Deben usarse para propósitos completamente diferentes y además deberías saber cómo utilizar cada una de las cadenas existentes. Si no comprendes cómo se usan, es fácil que introduzcas un agujero en el cortafuegos que tarde o temprano descubrirá y explotará alguien. Ya se han discutido en profundidad las tablas y cadenas en *Atravesando tablas y cadenas*. Si no lo entiendes a la perfección, te recomiendo que vuelvas a leerlo hasta que lo comprendas totalmente.

Comandos

En esta sección trataremos los distintos comandos y qué se puede hacer con ellos. Un comando le indica a **iptables** qué hacer con el resto de la regla que enviamos al analizador. Normalmente desearemos añadir o eliminar algo en una tabla determinada. En iptables están disponibles los siguientes comandos:

Table 6-2. Comandos

Comando	**-A**, **--append**
Ejemplo	**iptables -A INPUT ...**
Descripción	Este comando añade la regla al final de la cadena. La regla siempre se pondrá la última en el conjunto de reglas y lógicamente se comprobará la última, a no ser que posteriormente añadas más reglas con este mismo comando.
Comando	**-D**, **--delete**
Ejemplo	**iptables -D INPUT --dport 80 -j DROP**, **iptables -D INPUT 1**
Descripción	Este comando borra una regla de la cadena. Esto puede hacerse de dos maneras: bien introduciendo la regla completa a comparar (como en el ejemplo anterior), bien especificando el número de regla que deseas eliminar. Si empleas el primer método, deberás escribir exactamente lo mismo que haya en la cadena a borrar. Si empleas el segundo método, deberás señalar el número exacto que tiene la regla en la cadena: las reglas están numeradas progresivamente desde la primera, empezando con el número 1.
Comando	**-R**, **--replace**
Ejemplo	**iptables -R INPUT 1 -s 192.168.0.1 -j DROP**
Descripción	Este comando sustituye la entrada existente en la línea especificada. Funciona de la misma forma que el comando **--delete**, pero en lugar de eliminar completamente la entrada, la sustituye por una nueva. El uso más habitual de este comando puede ser la experimentación con iptables.
Comando	**-I**, **--insert**

Ejemplo	**iptables -I INPUT 1 --dport 80 -j ACCEPT**
Descripción	Se inserta una regla en la posición de la cadena que especifiquemos. En el ejemplo anterior se insertará en la posición nº 1 en la cadena *INPUT*, por lo que a partir de entonces será la primera regla en esa cadena.
Comando	**-L**, **--list**
Ejemplo	**iptables -L INPUT**
Descripción	Este comando ofrece una lista de todas las entradas de la cadena especificada. En el ejemplo la lista mostrará todas las entradas de la cadena *INPUT*. Sin embargo está permitido no especificar ninguna cadena en particular, con lo cual el comando listará todas las cadenas de la tabla especificada (para especificar tablas, lee la sección *Tablas*). El resultado final depende de otras opciones enviadas al analizador, como pueden ser **-n**, **-v**, etc.
Comando	**-F**, **--flush**
Ejemplo	**iptables -F INPUT**
Descripción	Este comando elimina todas las reglas de una cadena, comenzando desde la que se ha especificado. Es equivalente a borrar cada regla una a una, pero bastante más rápido. Se puede emplear sin opciones, con lo que borrará todas las reglas de todas las cadenas en la tabla especificada.
Comando	**-Z**, **--zero**
Ejemplo	**iptables -Z INPUT**

Descripción	Este comando obliga a comenzar desde cero a todos los contadores de una cadena especificada, o de todas las cadenas de una tabla. Si has utilizado la opción **-v** del comando **-L**, probablemente habrás visto el contador de paquetes al principio de cada campo. Para "poner a cero" este contador, utiliza la opción **-Z**. Esta función hace lo mismo que **-L**, salvo que **-Z** no hace ningún listado de las reglas. Si se emplean juntas **-L** y **-Z** (lo cual es correcto), las cadenas serán listadas primero y luego los contadores se reiniciarán (se pondrán a cero).
Comando	**-N**, **--new-chain**
Ejemplo	**iptables -N allowed**
Descripción	Este comando hace que el núcleo cree una nueva cadena con el nombre especificado en la cadena especificada. En el ejemplo anterior se crea una cadena llamada **allowed**. Ten en cuenta que no puede haber ninguna cadena ni ningún objetivo con el mismo nombre.
Comando	**-X**, **--delete-chain**
Ejemplo	**iptables -X allowed**

Descripción	Este comando borra de la tabla la cadena especificada. Para que funcione, no debe haber ninguna regla que esté relacionada con la cadena que se va a borrar. En otras palabras, deberás borrar o cambiar todas las reglas que tengan algún vínculo con esa cadena antes de borrarla. Si se usa el comando sin opciones, todas las cadenas creadas por el usuario serán eliminadas y sólo permanecerán aquellas que pertenezcan a la tabla especificada, es decir, aquellas que se instalan con iptables. A modo de ejemplo, si escribimos **iptables --delete-chain -t filter** lo que conseguiremos será eliminar todas las cadenas "de usuario" de la tabla filter, permaneciendo las cadenas por defecto de esta tabla; si, en cambio, escribimos **iptables --delete-chain**, iptables considera que especificamos la tabla por defecto (la tabla filter), con lo cual el resultado será exactamente el mismo que en el ejemplo anterior.
Comando	**-P**, **--policy**
Ejemplo	**iptables -P INPUT DROP**
Descripción	Este comando hace que el núcleo establezca la política u objetivo por defecto en una cadena. Todos los paquetes que no coincidan con ninguna regla emplearán esa política de la cadena. Los objetivos permitidos son: **DROP** y **ACCEPT** (pueden haber más; envíame un correo si es así).
Comando	**-E**, **--rename-chain**
Ejemplo	**iptables -E allowed disallowed**
Descripción	El comando **-E** hace que **iptables** cambie el nombre de una cadena del primer al segundo nombre. En el ejemplo anterior cambiaríamos el nombre de la cadena de allowed (permitido) a disallowed (no permitido). Ten en cuenta que ésto no afecta a la forma de actuar de la tabla, sino que es simplemente un cambio cosmético.

Siempre deberías escribir una línea completa de comando, a no ser que sólo quieras ver la ayuda de **iptables** o conocer la versión del comando. Para mostrar la versión, emplea la opción **-v** y para la ayuda, usa la opción **-h**. Vamos, como siempre.

Ahora vamos a ver unas cuantas opciones que pueden usarse con varios comandos. Observarás que se indica con qué comandos se pueden emplear y qué resultados ofrecen. Ten en cuenta que no se incluye ninguna opción que afecte a reglas o comparaciones. Estas otras opciones se verán en profundidad más adelante.

Table 6-3. Opciones

Opción	**-v**, **--verbose**
Comandos con los que se emplea	**--list**, **--append**, **--insert**, **--delete**, **--replace**
Descripción	Este comando ofrece una salida detallada (completa, descriptiva) y se emplea principalmente con el comando **--list**. En este caso se mostrará la dirección de la interfaz, las opciones de la regla y las máscaras TOS. Asímismo, también se incluyen los contadores de bytes y paquetes para cada regla. Estos contadores emplean las abreviaturas K (x1,000), M (x1,000,000) y G (x1,000,000,000). Para obtener el tamaño exacto, puedes emplear la opción **-x**, descrita más adelante. Si se usa esta opción con los comandos **--append**, **--insert**, **--delete** o **--replace**, el programa mostrará información detallada sobre cómo se ha interpretado la regla, si se ha insertado correctamente, etc.
Opción	**-x**, **--exact**
Comandos con los que se emplea	**--list**

Descripción	Esta opción expande las numeraciones, o sea, no se emplean los múltiplos K, M o G, si no que se ven los paquetes y bytes exactos de los contadores de la regla en cuestión. Esta opción sólo es útil con el comando **--list** y no tiene ningún interés en los demás comandos.
Opción	**-n**, **--numeric**
Comandos con los que se emplea	**--list**
Descripción	Esta opción muestra valores numéricos: las direcciones IP y los puertos se listarán con sus números y no con sus nombres de servidor, red o aplicación. Sólo se utiliza con **--list** e ignora la opción por defecto de traducir todos los valores numéricos a servidores y nombres (donde sea posible).
Opción	**--line-numbers**
Comandos con los que se emplea	**--list**
Descripción	Empleando esta opción con **--list**, cada regla se listará con su número de línea. Puede ser conveniente saber qué número tiene cada regla al insertar reglas. Esta opción sólo tiene efecto con el comando **--list**.
Opción	**-c**, **--set-counters**
Comandos con los que se emplea	**--insert**, **--append**, **--replace**

Descripción	Esta opción se usa cuando se crea o modifica una regla de forma que se reinicializan sus contadores de paquetes y bytes. La sintaxis sería algo así: **--set-counters 20 4000**, lo cual le diría al núcleo que fijara el contador de paquetes a 20 y el contador de bytes a 4000.
Opción	**--modprobe**
Comandos con los que se emplea	Todos
Descripción	Esta opción se emplea para indicarle a **iptables** qué módulo utilizar al probar módulos o al añadirlos al núcleo. Se puede usar por ejemplo cuando el comando **modprobe** no está en ningún directorio de la ruta de búsqueda. De esta forma el programa sabrá qué hacer si se necesita un módulo que no se ha cargado previamente. Esta opción se puede utilizar con todos los comandos

Comparaciones ("matches")

En esta sección hablaremos sobre las comparaciones. He decidido agrupar las comparaciones en cinco categorías diferentes: para empezar tenemos las *comparaciones genéricas*, que pueden usarse en todas las reglas; a continuación vienen las *comparaciones TCP* que sólo pueden aplicarse a los paquetes *TCP*; después vienen las *comparaciones UDP* y las *comparaciones ICMP*, que sólo se aplican a paquetes *UDP* e *ICMP* respectivamente; por último queda un grupo de comparaciones especiales, como las de estado, las de propietario, comparaciones límite, etc. Este último grupo se ha clasificado en subcategorías, aunque en realidad no tienen por qué ser comparaciones diferentes. Espero que esta clasificación sea razonable y todo el mundo pueda entenderla.

Comparaciones genéricas

Una comparación genérica es un tipo de comparación que está siempre disponible, sea cual sea el protocolo con el que trabajemos y sin importar qué extensiones de comparación se hayan cargado: no se necesitan parámetros especiales para emplearlas. He incluído aquí la comparación **--protocol**, aunque podría parecer más adecuado que estuviera en las comparaciones de protocolo. Por ejemplo, si queremos emplear una *comparación TCP*, necesitaremos usar la comparación **--protocol** con la opción *TCP*. Sin embargo **--protocol** también es una comparación por sí misma, ya que puede usarse para comparar protocolos específicos. Las siguientes comparaciones siempre están disponibles.

Table 6-4. Comparaciones genéricas

Comparación	**-p, --protocol**
Ejemplo	**iptables -A INPUT -p tcp**
Descripción	Esta comparación se emplea para comprobar la presencia de los siguientes protocolos: *TCP*, *UDP*, *ICMP* o sus respectivos valores numéricos, tal como se especifican en el archivo endterm="protocolstxt.title>. Si no puede identificar uno de éllos devolverá un error. El protocolo puede corresponder a uno de los tres mencionados, aunque también puede corresponder a TODOS (ALL). "ALL" significa que se corresponde con todos los protocolos (TCP, UDP e ICMP). El comando también acepta listas de protocolos delimitados por comas, por ej. **udp,tcp** que buscaría paquetes *UDP* y *TCP*. Si a esta comparación se le especifica el valor cero (0) será como si se especificaran TODOS (ALL) los protocolos (éste es el valor por defecto si no se usa la comparación **--protocol**). También se puede invertir la comparación con el símbolo **!**; o sea que **--protocol ! tcp** quiere decir que se tienen que comparar todos los protocolos excepto el TCP (o sea, *UDP* e *ICMP*).

Comparación	**-s**, **--src**, **--source**
Ejemplo	**iptables -A INPUT -s 192.168.1.1**
Descripción	Esta es la comparación del origen, que se basa en la dirección IP de origen de los paquetes. El uso genérico compara direcciones IP únicas, como *192.168.1.1*. Sin embargo también acepta máscaras de red en un formato de "bit" CIDR, especificando el número de unos (1) en la parte izquierda de la máscara de red: cada número es el equivalente decimal de un número binario de 8 bits, en el que la combinación de "1"s y "0"s en cadenas de ocho números, equivale a un valor decimal; por ej., 8 unos seguidos en binario equivalen a 255 y si tenemos "8 unos, punto, 8 unos, punto, 8 unos, punto, 8 ceros", en notación decimal, la que todos gastamos, tenemos "255.255.255.0". Ésto significa que podemos indicar */24* (tres veces ocho, sumados) para emplear una máscara de red *255.255.255.0*. Así podemos comparar rangos completos de direcciones IP, como nuestras redes locales o segmentos de red tras un cortafuegos, especificándolo más o menos así: *192.168.0.0/24* (con ello se buscarían paquetes que cayeran dentro del rango *192.168.0.1- 192.168.0.254*). Otra forma de hacerlo es con una máscara de red tradicional: *255.255.255.255* (es decir, *192.168.0.0/255.255.255.0*). También se puede invertir la comparación escribiendo el símbolo **!**: si empleamos una comparación como **--source ! 192.168.0.0/24**, lo que conseguiremos es comparar todos los paquetes que NO provengan del rango *192.168.0.x*. Por defecto se comparan todas las direcciones.
Comparación	**-d**, **--dst**, **--destination**
Ejemplo	**iptables -A INPUT -d 192.168.1.1**

Descripción	La comparación **--destination** (destino) se usa en función de la dirección/direcciones de destino de los paquetes. Se emplea casi igual que la comparación **--source** y tiene la misma sintaxis, excepto que se basa en saber a dónde se dirijen los paquetes. Para comparar un rango de direcciones IP, podemos especificar una máscara de forma "ortodoxa" o indicando el número de unos (1) contados desde la parte izquierda de la máscara de red (como en **--source**, la notación CIDR se basa en números binarios, formados con ristras de unos y ceros que equivalen a números decimales). Por ej.: *192.168.0.0/255.255.255.0* y *192.168.0.0/24* son equivalentes. También se puede invertir la comparación con el símbolo **!**, o sea que **--destination ! 192.168.0.1** buscará todos los paquetes que no vayan destinados a la dirección IP *192.168.0.1*.
Comparación	**-i, --in-interface**
Ejemplo	**iptables -A INPUT -i eth0**
Descripción	Esta comparación la emplearemos para reconocer a través de qué interfaz proviene un paquete entrante. Esta opción sólo es válida en las cadenas *INPUT*, *FORWARD* y *PREROUTING*, devolviendo un error si se usa en cualquier otro sitio. Si no especificamos una interfaz concreta, por defecto se asume el valor **+**, que se emplea para comparar una cadena de letras y números; o sea que por defecto el núcleo comparará todos los paquetes sin importarle a través de qué interfaz vienen. El símbolo **+** también puede añadirse al tipo de interfaz, de forma que **eth+** revisará los paquetes que entren por cualquier tarjeta de red Ethernet. También es posible invertir el significado de esta opción con ayuda del símbolo **!**. La línea tendrá una sintaxis parecida a: **-i ! eth0**, que aceptará los paquetes de todas las interfaces Ethernet por donde llegan datos, excepto la interfaz *eth0*.

Comparación	**-o, --out-interface**
Ejemplo	**iptables -A FORWARD -o eth0**
Explicación	Esta comparación se emplea con los paquetes que están a punto de abandonar la interfaz de salida. Ten en cuenta que esta comparación sólo está disponible en las cadenas *OUTPUT*, *FORWARD* y *POSTROUTING*, de hecho son justo las opuestas a las de la comparación **--in-interface**. Aparte de ésto, ambas funcionan prácticamente igual. La extensión **+** implica comparar todas las tarjetas del mismo tipo, es decir **eth+** controlará todas las tarjetas *eth* y de forma similar los demás casos. Para invertir el significado de la comparación puedes usar el símbolo **!** de la misma forma que en **--in-interface**. Si no se especifica ninguna comparación **--out-interface**, por defecto se comparan todas las tarjetas, independientemente de dónde vaya el paquete.
Comparación	**-f, --fragment**
Ejemplo	**iptables -A INPUT -f**

Descripción	Esta comparación se emplea para chequear la segunda y la tercera partes de un paquete fragmentado. La razón de ello es que cuando existen paquetes fragmentados no hay manera de saber ni las direcciones de origen/destino de los paquetes, ni los tipos *ICMP*, ni otros tantos detalles necesarios. Además los paquetes fragmentados pueden emplearse en ciertos casos como armas de ataque contra otros ordenadores. Fragmentos de paquetes como éstos no serán chequeados por otras reglas y por éllo se ha creado esta comparación. Como viene siendo habitual, esta opción puede combinarse con el símbolo **!**; sin embargo en este caso el símbolo debe preceder a la comparación, es decir **! -f**. Cuando esta comparación se invierte, se comparan sólo las cabeceras de los paquetes y/o los paquetes no fragmentados. O lo que es lo mismo, en cualquier paquete fragmentado se comparará el primer fragmento, pero no el segundo, ni el tercero, ... Asímismo se comparará cualquier paquete no fragmentado durante la transmisión. Recuerda que siempre puedes usar las muy buenas opciones de defragmentación que hay en el núcleo. Por otra parte, si usas el seguimiento de conexiones te darás cuenta que no aparece ningún paquete fragmentado, puesto que se procesan los fragmentos antes de que lleguen a cualquier tabla o cadena de **iptables**.

Comparaciones implícitas

Esta sección describe las comparaciones que se cargan implícitamente, o sea, de forma totalmente automática, como por ejemplo cuando se compara **--protocol tcp** sin ningún criterio añadido. Actualmente hay tres grupos de comparaciones implícitas, uno para cada uno de los tres protocolos principales. Son las *comparaciones TCP*, *comparaciones UDP* y *comparaciones ICMP*. El grupo de comparaciones basadas en *TCP* contiene un conjunto de criterios singulares que sólo están disponibles para los paquetes *TCP*. De la misma manera actúan los criterios de los grupos para *UDP* e *ICMP*. Por otra parte, pueden haber comparaciones explícitas que se carguen cuando se les indica (explícitamente) y lógicamente estas comparaciones no son automáticas, si no que debes especificarlas correctamente. Para ésto deberás emplear la opción **-m** o **--match**, que explicaré más adelante.

Comparaciones TCP

Estas comparaciones son específicas del protocolo y sólo están disponibles al trabajar con paquetes y flujos *TCP*. Para emplearlas debes indicar **--protocol tcp** en la línea de comandos antes de empezar a utilizarlas. Por ello **--protocol tcp** debe estar a la izquierda de las comparaciones específicas del protocolo. Como ya se ha dicho, estas comparaciones se cargan automáticamente.

Table 6-5. Comparaciones TCP

Comparación	**--sport**, **--source-port**
Ejemplo	**iptables -A INPUT -p tcp --sport 22**

Descripción	Esta comparación se emplea para comparar paquetes basándose en su dirección de origen. Si no se indica nada se comparan todos los puertos origen. La comparación puede tener un nombre de servicio o bien el número de un puerto. Si especificas un nombre de servicio, éste debe existir en el fichero */etc/services*, puesto que **iptables** emplea este archivo para interpretar los nombres de los servicios. Si especificas el puerto por su número, la regla se cargará ligeramente más rápido, ya que **iptables** no tiene que buscar el nombre de servicio. Sin embargo, la comparación puede llegar a ser un poco más difícil de leer que si emplearas el nombre del servicio. Si estás escribiendo un conjunto de reglas de 200 ó más reglas, ciertamente deberías usar números de puerto, puesto que la diferencia es realmente notable (en una máquina lenta, esto puede significar hasta 10 segundos de diferencia si has configurado un conjunto de unas 1000 reglas). También puedes usar la comparación **--source-port** para comparar cualquier rango de puertos, como por ej. **--source-port 22:80**, en que se compararán los paquetes que provengan de los puertos 22 a 80. Si se omite el primer puerto del rango, se asume que quieres empezar por el 0 (es una opción implícita), o sea que **--source-port :80** buscará paquetes que provengan de los puertos 0 a 80. Y de la misma forma, si el último puerto es el que se omite, implícitamente se está designando el puerto 65535; así, **--source-port 22:** implicará buscar paquetes que provengan de cualquier puerto entre el 22 y el 65535. Si inviertes el orden de los puertos, si no los especificas de menor a mayor, iptables interpreta automáticamente el rango "correcto": si escribes **--source-port 80:22**, iptables interpretará **--source-port 22:80**. Por ello, si por algún motivo lo que realmente deseas es el rango desde el puerto 80 hasta el 65535 y desde el 0 al 22 (lo que puede entenderse por 80:22), debes hacer uso del símbolo **!**, con lo que escribirás **--source-port ! 23:79**. Date cuenta que a primera vista la única diferencia es el símbolo de exclamación, pero el significado varía mucho (en este caso se ignoran todos los puertos entre el 23 y el 79, como deseábamos). Asímismo con **--source-port ! 22** se compararán todos los paquetes excepto aquellos que provengan del puerto 22. Debes saber también que esta comparación no acepta puertos múltiples o rangos de puertos múltiples separados por comas. Para mayor información sobre este tema, léete la extensión de comparación multipuerto.

Comparación	**--dport**, **--destination-port**
Ejemplo	**iptables -A INPUT -p tcp --dport 22**
Descripción	Esta comparación busca paquetes *TCP* basándose en el puerto de destino. Utiliza la misma sintaxis que la comparación **--source-port**, por lo que acepta puertos, rangos de puertos e inversiones. También "corrije" el orden de los rangos de puertos (80:22 se leerá 22:80) y se asumen los valores 0 y 65535 en caso de no indicar el primer o el último puerto, respectivamente (:80 y 22: , respectivamente). Esta comparación tampoco acepta puertos o rangos múltiples separados por comas (para eso está la extensión de comparación multipuerto).
Comparación	**--tcp-flags**
Ejemplo	**iptables -p tcp --tcp-flags SYN,FIN,ACK SYN**

Descripción	Esta comparación busca en las banderas (flags) de los paquetes *TCP*. Para empezar la comparación lee una lista de banderas para comparar (una máscara) y después lee una lista de banderas que deben estar establecidas con el valor 1, o sea estar activadas. Ambas listas deben estar delimitadas por comas. La comparación reconoce las banderas *SYN*, *ACK*, *FIN*, *RST*, *URG* y *PSH*, además de interpretar las palabras ALL (todas) y NONE (ninguna). ALL y NONE casi no necesitan explicación: ALL significa que se deben emplear todas las banderas, mientras que NONE significa no usar ninguna bandera para la comparación. De aquí que **--tcp-flags ALL NONE** signifique que se deben chequear todas las banderas *TCP* y comprobar si ninguna de ellas está activada. Esta opción también puede ser invertida por el símbolo **!**, es decir, si se especifica **! SYN,FIN,ACK SYN**, lo que se obtiene es una comparación que busque todos los paquetes que tengan activados los bits *ACK* y *FIN*, pero no el bit *SYN*: la máscara de búsqueda incluye las banderas SYN, FIN y ACK, por lo que se chequearán sólo estas banderas; entonces, si fuera una búsqueda directa, se determinaría si la bandera SYN está activada, al contrario que las FIN y ACK, que al no incluirse en la segunda lista deberían estar desactivadas; sin embargo, al ser una comparación invertida se buscará que FIN y ACK sí estén activadas y que SYN no esté activada. Date cuenta también que las comas no deben incluir espacios (puedes ver la sintaxis correcta en el ejemplo de arriba).
Comparación	**--syn**
Ejemplo	**iptables -p tcp --syn**

Descripción	Se puede decir que la comparación **--syn** es una reliquia de los días de ipchains y que permanece para garantizar la compatibilidad con viejas reglas y para facilitar la transición de ipchains a iptables. Se usa para comparar paquetes que tengan activado el bit *SYN* y que no tengan activados los bits *ACK* y *RST*. Es decir, hace lo mismo que la comparación **--tcp-flags SYN,RST,ACK SYN**. Estos paquetes se emplean principalmente para solicitar nuevas conexiones a un servidor. Si bloqueas estos paquetes lo que consigues en la práctica es bloquear cualquier intento de conexión desde el exterior. Sin embargo, no habrás bloqueado las conexiones hacia el exterior y este fallo es el que explotan muchos ataques de hoy en día (por ej., "hackear"/romper la integridad y seguridad de un servicio legítimo y a partir de ahí instalar un programa o ejecutable similar que permita iniciar una conexión existente hacia tu host, en lugar de abrir un nuevo puerto en el host). Esta comparación también puede ser invertida con el sígno de exclamación (**!**) de esta forma: **! --syn**. Ésto hace que se reconozcan los paquetes con los bits *RST* o *ACK* activados, o sea, los paquetes de una conexión ya establecida.
Comparación	**--tcp-option**
Ejemplo	**iptables -p tcp --tcp-option 16**

Descripción	Esta comparación se emplea para filtrar paquetes dependiendo de sus opciones *TCP*. Una *Opción TCP* (*TCP Option*) es una parte específica de la cabecera de cada paquete que consta de 3 campos diferentes. El primero tiene un tamaño de 8 bits y nos dice qué Opciones se emplean en el flujo. El segundo también es de 8 bits y nos dice qué tamaño tiene el campo de opciones. La razón para que exista este campo indicando el tamaño es que las *Opciones TCP* son, valga la redundancia, opcionales. Para seguir los estándares no necesitamos implementar todas las opciones, sinó que simplemente podemos mirar qué tipo de opción es y si la soportamos, puesto que de no soportarla miraremos cuál es el tamaño del campo y entonces ignoraremos esta parte de los datos del paquete. Esta comparación se emplea para buscar diferentes opciones *TCP* dependiendo de sus valores decimales. También puede ser invertido con **!**, de forma que la comparación busque todas las opciones pero ignore la opción indicada. Para ver una lista completa con todas las opciones échale un vistazo a *Internet Engineering Task Force*, que mantiene una lista de todos los números estándar empleados en Internet.

Comparaciones UDP

Esta sección describe las comparaciones que sólo trabajarán con paquetes *UDP*. En cuanto especificas la comparación **--protocol UDP**, las opciones se cargan implícitamente (sin que haga falta indicarlas) y automáticamente ya están disponibles. Recuerda que los paquetes *UDP* no están orientados a la conexión y por éllo no existen banderas para establecer su valor en el paquete de forma que indiquen cuál es la utilidad del datagrama (como abrir o cerrar una conexión) o si únicamente sirven para enviar datos. Asímismo, tampoco requieren ningún tipo de reconocimiento: si se pierden, simplemente se han perdido (si no tenemos en cuenta los mensajes de error *ICMP*). Ésto significa que hay muchas menos comparaciones con las que trabajar en paquetes *UDP*, comparados con los paquetes *TCP*. Cabe destacar que la máquina de estados trabaja con todo tipo de paquetes, a pesar de que los paquetes *UDP* e *ICMP* se entienden como protocolos sin conexión. Más aún, la máquina de estados trabaja prácticamente igual con los paquetes *UDP* que con los *TCP*.

Table 6-6. Comparaciones UDP

Comparación	**--sport**, **--source-port**
Ejemplo	**iptables -A INPUT -p udp --sport 53**
Descripción	Esta comparación trabaja exactamente igual que su homónima en *TCP* y se emplea para hacer comparaciones basadas en los puertos *UDP* de origen. Acepta rangos de puertos, puertos únicos e inversiones, con la misma sintaxis que lo hace la homónima en *TCP*: para especificar un rango indicarás el puerto inicial y el final (por ej. 22:80), con lo cual se compararán los puertos *UDP* incluídos entre ambos (desde el 22 hasta el 80, en el ejemplo); si el valor inicial se omite se asumirá el valor 0; si se omite el puerto final, se asume el puerto 65535; si el puerto con valor más alto se indica antes que el puerto con valor más bajo, iptables entenderá el rango inverso (si se escribe 80:22, se entiende 22:80). Las comparaciones con un único puerto *UDP* serán similares al ejemplo inicial. Para invertir la selección de puertos, añade el símbolo de exclamación **!**, como en **--source-port ! 53** (con ello se compararán todos los puertos excepto el 53). La comparación puede interpretar nombres de servicios, siempre que estén disponibles en el archivo */etc/services*. Sin embargo, no se aceptan puertos múltiples ni rangos múltiples separados por comas. Para más información sobre este tema léete la extensión de comparaciones multipuerto.

Comparación	**--dport**, **--destination-port**
Ejemplo	**iptables -A INPUT -p udp --dport 53**
Descripción	Podemos decir lo mismo en ésta y en la comparación anterior. Es exactamente lo mismo que su homónima en *TCP*, a excepción que aquí se comparan puertos *UDP*: compara paquetes basados en su puerto *UDP* de destino. Acepta rangos de puertos, puertos únicos e inversiones. Para reconocer un sólo puerto, usarás por ej. **--destination-port 53**; en cambio, para invertir la selección usarás **--destination-port ! 53**. En el primer caso la comparación buscará todos los paquetes dirigidos al puerto 53, mientras que en el segundo caso (el invertido) se buscarán todos los paquetes excepto los dirigidos al puerto 53. En el caso de un rango de puertos, puedes escribir **--destination-port 8:19**, con lo cual se captarán todos los paquetes dirigidos a los puertos comprendidos entre el 8 y el 19. Si se omite el primer puerto, se asume el valor 0. Si por el contrario se omite el segundo, se asumirá el valor 65535. Si se invierte el orden de los puertos (19:8), el programa los reordena automáticamente (se convierte en 8:19). Esta comparación no admite puertos y rangos múltiples (para ello existe la extensión de comparación multipuerto).

Comparaciones ICMP

Los paquetes *ICMP* son incluso más efímeros, tienen una vida más corta, que los paquetes *UDP*, dado que no están orientados a la conexión. El protocolo *ICMP* se usa principalmente para comunicación de errores, control de conexiones y tareas similares. *ICMP* no es un protocolo subordinado al protocolo IP, sinó más bien un protocolo que mejora al protocolo IP y le ayuda a gestionar los errores. Las cabeceras de los paquetes *ICMP* son muy similares a las cabeceras IP, pero presentan varias diferencias. La principal característica de este protocolo es la cabecera de "tipo", la cual nos indica para qué es el paquete. Por ejemplo, si intentamos acceder a una dirección IP inaccesible, normalmente recibiremos un ***ICMP host unreachable*** (mensaje ICMP de host inalcanzable) como respuesta. Para ver una lista completa de los "tipos" *ICMP*, lee el apéndice *Tipos ICMP*. Sólo hay una comparación específica para paquetes *ICMP* y es de esperar que sea suficiente con élla. Esta comparación se carga implícitamente al emplear la comparación **--protocol ICMP** y tenemos acceso a ella automáticamente. Recuerda que todas las comparaciones genéricas se pueden usar con ella, por lo que nos podemos centrar en las direcciones de origen y destino.

Table 6-7. Comparaciones ICMP

Comparación	**--icmp-type**
Ejemplo	**iptables -A INPUT -p icmp --icmp-type 8**
Descripción	Esta comparación se usa para especificar el *tipo ICMP* a comparar. Los *tipos ICMP* se pueden especificar por su valor numérico o por su nombre. Los valores numéricos están definidos en el RFC 792. Para obtener un listado completo de los nombres *ICMP*, ejecuta **iptables --protocol icmp --help** en la línea de comandos o mira en el apéndice *Tipos ICMP*. Esta comparación también puede ser invertida con el símbolo **!**, como en **--icmp-type ! 8**. Ten en cuenta que algunos *tipos ICMP* están obsoletos, mientras que otros pueden ser "peligrosos" para un host desprotegido, puesto que pueden (entre otras cosas) redirigir paquetes a lugares incorrectos.

Comparaciones explícitas

Las comparaciones explícitas son aquellas que se deben cargar específicamente con la opción **-m** o **--match**. Por ej., las comparaciones de estado exigen añadir la directiva **-m state** antes de introducir la comparación que deseas usar. Algunas de estas comparaciones pueden ser específicas de un protocolo. Otras pueden ser independientes de los protocolos, como las de los estados de conexión: **NEW** (NUEVO; el primer paquete de una conexión que todavía no está establecida), **ESTABLISHED** (ESTABLECIDO; una conexión que ya está registrada en el núcleo), **RELATED** (RELACIONADO; una nueva conexión que ha sido creada por otra conexión ya establecida y más antigua), etc. Existen unas cuantas, por otra parte, que pueden haber evolucionado con propósitos experimentales o de prueba, o simplemente para mostrar de qué es capaz iptables. Ésto significa que no todas estas comparaciones tienen una aplicación directa o "útil" a primera vista. Sin embargo es perfectamente posible que en tu caso concreto encuentres utilidad a alguna comparación específica. Además, cada vez hay más con cada nueva versión de **iptables** y que les encuentres utilidad dependerá en gran medida de tu imaginación y tus necesidades. La diferencia entre comparaciones cargadas implícita o explícitamente es que las implícitas se cargarán automáticamente cuando, por ej., compares las propiedades de los paquetes *TCP*, mientras que las explícitas nunca se cargarán automáticamente: es tarea enteramente tuya que descubras y actives las comparaciones explícitas.

Comparación Límite

La comparación **limit** (límite o limitación) debe ser cargada de forma explícita mediante la opción **-m limit**. Como ejemplo típico, esta comparación puede usarse para limitar el registro de actividad de reglas específicas. Ésto es, puedes comprobar que los paquetes no hayan excedido un determinado valor, a partir del cual se limita el registro de ese evento en cuestión. Imagina un límite de tiempo: puedes limitar las veces que una regla puede ser alcanzada en un periodo de tiempo dado, como cuando quieres minimizar los efectos de un "ataque de denegación de servicio por desbordamiento" (*DoS*). Éste es su principal uso, pero hay más, por supuesto. La comparación también puede ser invertida escribiendo el símbolo **!** antes de la comparación límite: **-m limit ! --limit 5/s**, que quiere decir que todos los paquetes se compararán después de sobrepasar el límite de 5 veces por segundo.

Para ampliar la explicación, se puede decir que es básicamente como un "monedero virtual". Consideremos un monedero del que sacamos X monedas para gastar y al que introducimos monedas cada cierto tiempo. X se define dependiendo de cuántos paquetes recibimos que concuerdan con la comparación, por lo que suponiendo que son 3 paquetes, sacaremos del monedero 3 monedas en esa unidad de tiempo. La opción **--limit** nos dice con cuántos paquetes (monedas) rellenaremos el monedero por unidad de tiempo, mientras que **--limit-burst** indica la cantidad de monedas iniciales del monedero, así como cuántas de esas monedas admite el monedero. Así pues, escribiendo **--limit 3/minute --limit-burst 5**, crearemos un "monedero" con 5 "monedas"; si recibimos a continuación 5 paquetes seguidos que concuerden con la comparación, como resultado el "monedero" se vaciará. Tras 20 segundos, el "monedero" se rellena con otra "moneda" (recordemos que se rellena a razón de 3 por minuto) y a ese ritmo seguirá "rellenándose" hasta que se alcance el tope impuesto por el **--limit-burst** o hasta que sean usadas y el "monedero" vuelva a estar vacío.

Vamos a ver otro ejemplo algo más realista para aclarar mejor el funcionamiento:

1. Creamos una regla que contenga la siguiente comparación: -m limit --limit 5/second --limit-burst 10/second. El contador se establece con un valor inicial de 10. Cada paquete que coincida con el filtro de la regla resta una unidad al contador.
2. Recibimos una serie de paquetes (1-2-3-4-5-6-7-8-9-10) que coinciden con la regla, todos éllos en 1/1000 de segundo.
3. Ahora el contador está a cero ("vacío") y los paquetes que coincidan con la regla no serán filtrados por ésta y saltarán a la siguiente regla si existe, o en caso de no existir seguirán la política por defecto de la cadena.
4. Por cada 1/5 de segundo sin paquetes que coincidan, el contador aumenta una unidad hasta el valor máximo de 10. 1 segundo

después de haber recibido los 10 paquetes anteriores, volveremos a aceptar hasta 5 paquetes seguidos.

5. Y por supuesto, el contador irá descontando 1 unidad por cada paquete correcto que se reciba.

Table 6-8. Opciones de comparación límite

Comparación	**--limit**
Ejemplo	**iptables -A INPUT -m limit --limit 3/hour**
Descripción	Mediante esta opción se establece el valor más alto del ratio medio de comparaciones positivas para la comparación **limit**. Se especifica mediante un valor numérico y un valor opcional de medida del tiempo. Las unidades de tiempo aceptadas actualmente son: /**second** (segundo), /**minute** (minuto), /**hour** (hora) y /**day** (día). El valor por defecto es de 3 cada hora, o lo que es lo mismo **3**/**hour**. Así pues, con esta opción se consigue indicar a **limit** cuántas veces se le permite actuar a la comparación por unidad de tiempo (por cada minuto, por ej.).
Comparación	**--limit-burst**
Ejemplo	**iptables -A INPUT -m limit --limit-burst 5**

Descripción	Mediante esta opción le dices a **iptables** el número máximo de paquetes que concuerden con la comparación en un tiempo determinado. Este número (la cantidad de paquetes que han llegado a la comparación) disminuye en una unidad por cada unidad de tiempo (definida por la opción **--limit**) en que el evento no se presenta (no llega ningún paquete), hasta llegar al valor mínimo (1). Si se repite el evento (empiezan a llegar paquetes), el contador se va incrementando hasta que llega al tope establecido por **--limit-burst**, para volver a empezar (si no llegan paquetes el contador disminuye, mientras que cuando van llegando aumenta hasta el umbral definido por --limit-burst). El valor por defecto es 5. La manera más simple de comprobar cómo funciona esta opción es utilizando el ejemplo *Limit-match.txt*. Con él podrás comprobar por tí mismo cómo funciona simplemente enviando pings a diferentes intervalos de tiempo y con diferentes valores tope. Todas las respuestas de eco (echo replies) serán bloqueadas hasta que se vuelva a alcanzar el umbral límite.

Comparación MAC

La comparación *MAC* (*Ethernet Media Access Control*, o control de acceso a tarjetas Ethernet) se usa para comparar paquetes en función de su *dirección MAC de origen* [MAC source address; ésta es la dirección física de la tarjeta de red Ethernet: se trata de un código registrado en el hardware de la tarjeta que la hace única frente a todas las demás tarjetas fabricadas y por fabricar, es decir, no debería haber dos tarjetas Ethernet con la misma dirección MAC]. Por ahora esta comparación está un poco limitada, aunque en el futuro es posible que evolucione y sea más útil. Insistamos un poco: esta comparación SÓLO compara paquetes en función de su dirección MAC de ORIGEN.

Ten en cuenta que para usar este módulo lo cargamos explícitamente con la opción **-m mac**. Y digo ésto porque mucha gente se pregunta si la opción correcta no debería ser **-m mac-source**, pero no es el caso.

Table 6-9. Opciones de la comparación MAC

Comparación	**--mac-source**
Ejemplo	**iptables -A INPUT -m mac --mac-source 00:00:00:00:00:01**
Descripción	Como ya se ha dicho, esta comparación se emplea para buscar paquetes basándose en su *dirección MAC de origen*. La *dirección MAC* indicada debe tener el siguiente aspecto: *XX:XX:XX:XX:XX:XX*, pues de lo contrario no sería correcta (el programa la consideraría una dirección "ilegal"). La comparación también puede ser invertida con **!**, y se parecerá a **--mac-source ! 00:00:00:00:00:01**. En este ejemplo se aceptarán todos los paquetes excepto aquellos que provengan de la tarjeta Ethernet especificada (la que tenga la dirección MAC 00:00:00:00:00:01). Ten en cuenta que las direcciones MAC sólo se emplean en las redes de tipo Ethernet, por lo que sólo será viable emplear la comparación con tarjetas Ethernet. Además, sólo será válida en las cadenas *PREROUTING*, *FORWARD* e *INPUT* y en ninguna más.

Comparación Mark

Esta extensión se emplea para comparar paquetes basándose en las marcas que tengan establecidas. Una **marca** (**mark**) es un campo especial mantenido por el núcleo, que se asocia a los paquetes mientras viajan por el ordenador. Las marcas pueden usarse por diferentes rutinas del núcleo para tareas como la configuración y el filtrado del tráfico. Hoy por hoy sólo existe una manera de establecer una marca en Linux y ésta es mediante el objetivo **MARK** en **iptables**. Anteriormente se establecía con el objetivo **FWMARK** de **ipchains** y por éllo hay gente que todavía se refiere a **FWMARK** en tareas avanzadas de enrutado. El campo "marca" se establece mediante un número entero positivo de entre los 4.294.967.296 valores posibles en un entorno de 32 bits. O sea, que es más bien difícil que puedas verte limitado por este valor durante bastante tiempo.

Table 6-10. Opciones de la comparación de marca

Comparación	**--mark**
Ejemplo	**iptables -t mangle -A INPUT -m mark --mark 1**
Descripción	Empleamos la comparación para buscar paquetes que hayan sido marcados con anterioridad. Las marcas pueden establecerse gracias al objetivo **MARK**, que será tratado en la próxima sección. Todos los paquetes que atraviesan *Netfilter* reciben un campo de marca (*mark field*) especial y se asocian a él. Cabe destacar que este "*mark field*" no se propaga ni dentro ni fuera del paquete, sinó que permanece en el ordenador que lo creó. Si el *campo marca* coincide con la marca, entonces hay concordancia. El *campo marca* es un número entero positivo, por lo que pueden haber un máximo de 4.294.967.296 marcas diferentes (en entornos de 32 bits). También puedes utilizar una máscara en la marca, con lo cual su aspecto se podrá parecer a **--mark 1/1**. Si se especifica la máscara se efectuará una suma lógica (AND) con la marca especificada antes de ejecutar la comparación.

Comparación Multipuerto

La extensión **multiport** se emplea para especificar puertos múltiples y rangos de puertos múltiples. Si no tuviéramos la funcionalidad de esta comparación, necesitaríamos escribir múltiples reglas del mismo tipo simplemente para comparar diferentes puertos no contiguos (que no se pueden especificar mediante un rango).

Note: No puedes emplear la comparación estándar de puertos junto a la comparación múltiple de puertos en la misma regla; por ej., no puedes escribir: **--sport 1024:63353 -m multiport --dport 21,23,80**, puesto que simplemente no funcionarán las dos a la vez, sinó que iptables ejecutará el primer elemento de la regla e ignorará el otro (en este caso ignorará la comparación multipuerto).

Table 6-11. Opciones de comparación multipuerto

Comparación	**--source-port**
Ejemplo	**iptables -A INPUT -p tcp -m multiport --source-port 22,53,80,110**
Descripción	Mediante esta comparación se buscarán los puertos de origen indicados, que serán como máximo 15. Los puertos se delimitarán mediante comas, como en el ejemplo. La comparación sólo se puede emplear conjuntamente con las comparaciones **-p tcp** o **-p udp**. Básicamente es una versión ampliada de la comparación estándar **--source-port**.
Comparación	**--destination-port**
Ejemplo	**iptables -A INPUT -p tcp -m multiport --destination-port 22,53,80,110**
Descripción	De la misma forma que la anterior, se comparan puertos múltiples, aunque esta vez son los puertos de destino de los paquetes. También presenta el límite de una lista con un máximo de 15 puertos y sólo puede utilizarse junto a **-p tcp** o **-p udp**.

Comparación	**--port**
Ejemplo	**iptables -A INPUT -p tcp -m multiport --port 22,53,80,110**
Descripción	Esta extensión puede emplearse para comparar paquetes basándose en su puerto de origen/destino. Funciona igual que las dos anteriores, con las mismas limitaciones, pero con la diferencia de que sólo se aceptarán paquetes que provengan y se destinen al mismo puerto; en el ejemplo estaríamos hablando de paquetes que provengan del puerto 22 y vayan al puerto 22, provengan del 53 y vayan al 53, etc.

Comparación Propietario (Owner)

La extensión **owner** se emplea para comparar paquetes en función de la identidad del proceso que los creó. Se puede especificar como la identificación (ID) del proceso del usuario que ejecutó la orden en cuestión; este usuario puede ser el grupo, el proceso, la sesión o el comando en sí mismo. Esta extensión se creó originalmente para demostrar de qué es capaz **iptables**. La comparación **owner** sólo funciona en la cadena *OUTPUT* por razones obvias: es prácticamente imposible encontrar alguna información sobre la identidad del proceso que envió el paquete desde el otro extremo y menos aún si existe algún punto intermedio entre el origen y el destino. Incluso en la cadena *OUTPUT* no tiene un comportamiento muy fiable, pues ciertos paquetes pueden no tener un propietario (como ocurre por ejemplo con las diferentes *respuestas ICMP*, ya que éstas nunca serán aceptadas por la comparación).

Table 6-12. Opciones de comparación de propietario

Comparación	**--uid-owner**
Ejemplo	**iptables -A OUTPUT -m owner --uid-owner 500**

Descripción	En esta comparación se aceptarán todos los paquetes que hayan sido generados por el usuario especificado (en inglés se habla de ***User ID*** o ***UID***, siendo 500 la ID del ejemplo). Así pues, con esta regla podemos filtrar paquetes dependiendo del usuario que los haya creado. Una utilidad podría ser bloquear cualquier intento de abrir una conexión fuera del cortafuegos, siempre que el usuario no sea el administrador ("superuser", "su" o "root"). Otro posible uso sería bloquear a cualquier usuario que intente enviar paquetes por el puerto *HTTP* excepto al usuario ***http*** (con lo que minimizamos los ataques de troyanos que trabajen a través de ese puerto).
Comparación	**--gid-owner**
Ejemplo	**iptables -A OUTPUT -m owner --gid-owner 0**
Descripción	En este caso se compara el grupo al que pertenece el usuario que crea el paquete (en inglés, ***Group ID*** o ***GID***). Con esta opción podemos bloquear a todos los usuarios que no pertenezcan al grupo ***network***, de forma que sólo éstos últimos puedan acceder a Internet; o como ya se ha dicho antes, permitir sólo a los miembros del grupo ***http*** crear paquetes que salgan por el puerto *HTTP*.
Comparación	**--pid-owner**
Ejemplo	**iptables -A OUTPUT -m owner --pid-owner 78**

Descripción	En este caso se comparan los paquetes en función del número de proceso que lo generó (en inglés, *Process ID* o *PID*). Esta comparación es un poco más complicada de usar, pero un ejemplo podría ser al querer limitar el envío de paquetes por el puerto *HTTP*: suponiendo que el proceso número 78 (PID 78) sea el único que debe tener permiso, escribiendo la regla del ejemplo lo lograríamos, eso sí, siempre y cuando este proceso no esté dividido en varios procesos paralelos (threaded). Para facilitar las cosas, podemos escribir un pequeño script que capte el *PID* del demonio deseado desde la salida generada por el comando **ps** y cree una regla para ese PID. A modo de ejemplo tienes una regla en *Pid-owner.txt*.
Comparación	**--sid-owner**
Ejemplo	**iptables -A OUTPUT -m owner --sid-owner 100**
Descripción	Como en el resto del grupo, esta comparación busca paquetes basándose en la identificación de sesión empleada por el programa que genera los paquetes (en inglés, *Session ID* o *SID*). El valor del SID de un proceso es el del proceso mismo y todos los subprocesos creados a partir de él. Estos subprocesos pueden ser procesos paralelos (threads) o "hijos" del proceso original. A modo de ejemplo, en un equipo con el *HTTPD* dividido en procesos paralelos, todos estos procesos *HTTPD* "hijos" deberían tener el mismo *SID* que su proceso "padre" (el proceso *HTTPD* original); la mayoría de los demonios *HTTPD* funcionan de esta manera, como Apache y Roxen, por citar alguno. Para mostrártelo tienes a tu disposición un pequeño script llamado *Sid-owner.txt*. Con unos cuantos retoques este script podría ejecutarse cada hora de forma que compruebe si el *HTTPD* realmente está en marcha, arrancándolo si no es así y en ese caso modificando adecuadamente la cadena *OUTPUT*.

Comparación de Estado

La extensión **state** (estado) se emplea conjuntamente con el código de seguimiento de conexiones (connection tracking) del núcleo. La comparación de estado accede a la máquina de seguimiento de conexiones y averigua en qué estado se encuentra el paquete. Éste procedimiento funciona con prácticamente todos los protocolos, incluyendo aquellos que no poseen estado, como el *ICMP* y el *UDP*. En cualquier caso por defecto habrá un intervalo de espera para la conexión, a partir del cual será eliminada de la base de datos de seguimiento de conexiones. La comparación se debe cargar explícitamente añadiendo **-m state** a la regla, teniendo acceso a partir de entonces a una nueva comparación llamada "estado". El concepto de comparación de estado se trata en profundidad en el capítulo *La máquina de estados*, ya que es un tema bastante extenso.

Table 6-13. Comparaciones de Estado

Comparación	**--state**
Ejemplo	**iptables -A INPUT -m state --state RELATED,ESTABLISHED**

Descripción	Esta opción le indica a la comparación **state** (estado) qué paquetes deben ser comparados, en función de sus estados. Por ahora hay 4 estados que puedan ser utilizados: **INVALID**, **ESTABLISHED**, **NEW** y **RELATED** (inválido, establecido, nuevo y relacionado, respectivamente). **INVALID** implica que el paquete no está asociado a ningún flujo o conexión conocida y que puede contener datos o cabeceras erróneas. **ESTABLISHED** indica que el paquete pertenece a una conexión ya establecida, totalmente válida y que "ha visto" un flujo de paquetes en ambas direcciones. **NEW** significa que el paquete ha creado o está creando una nueva conexión, o que está asociado a una conexión que todavía no "ha visto" paquetes en ambas direcciones. Por último, **RELATED** es para paquetes que empiezan una nueva conexión pero están asociados a otra conexión ya establecida (como puede ocurrir en una *transferencia de datos FTP*, o en un *error ICMP* asociado a una conexión *TCP* o *UDP*). Ten en cuenta que el estado **NEW** ignora los bits *SYN* en los paquetes *TCP* que están intentando empezar una nueva conexión, por lo que no debe ser usado sin modificar en los casos en que exista sólo un cortafuegos y no haya balance de carga entre diferentes cortafuegos. Sin embargo, pueden haber circunstancias donde ésto sea útil. Para mayor información sobre cómo utilizarlo, lee el capítulo *La máquina de estados*.

Comparación TOS

La comparación **TOS** puede emplearse para comparar paquetes basándose en su *campo TOS* (*TOS field*). *TOS* es la abreviatura de *Type Of Service* (Tipo de Servicio), tiene un tamaño de 8 bits y se encuentra en las cabeceras IP. Esta comparación se carga explícitamente cuando añadimos **-m tos** a la regla. El Tipo de Servicio se suele usar para informar a los hosts intermedios por los que van pasando los paquetes de la llegada de un flujo y de su contenido (bueno, en realidad no hace eso, pero informa sobre cualquier requerimiento específico del flujo, como por ej. si tiene que reenviarse lo más rápidamente posible, o si necesita que se le permita contener tanta información como sea posible). De qué manera manejen los routers y administradores esa información depende exclusivamente de éllos. La mayoría la ignora, mientras que algunos intentan manejar los flujos y datos de la manera más acorde a la información del *TOS*.

Table 6-14. Comparaciones TOS

Comparación	**--tos**
Ejemplo	**iptables -A INPUT -p tcp -m tos --tos 0x16**

Descripción	Esta comparación se usa tal como se ha descrito y puede filtrar paquetes en función de su *campo TOS* y la información que contiene. Ésto es de utilidad, entre otras cosas, al trabajar conjuntamente con **iproute2** y con funciones avanzadas de Linux, marcando paquetes para un uso posterior. La comparación reconoce como opción un valor hexadecimal o también numérico, aunque posiblemente también acepte uno de los nombres resultantes de ejecutar "**iptables -m tos -h**". En el momento de escribir el tutorial, se reconocen los siguientes nombres/valores/valores hexadecimales: ***Minimize-Delay 16 (0x10)***, ***Maximize-Throughput 8 (0x08)***, ***Maximize-Reliability 4 (0x04)***, ***Minimize-Cost 2 (0x02)*** y ***Normal-Service 0 (0x00)***. ***Minimize-Delay*** se refiere a minimizar el retardo en dar curso a los paquetes (ejemplos de servicios estándar que lo requieren son *telnet*, *SSH* y *FTP-control*). ***Maximize-Throughput*** implica un rendimiento tan grande como sea posible (un protocolo estándar podría ser *FTP-data*). ***Maximize-Reliability*** pide que se maximice la fiabilidad de una conexión y que se empleen líneas tan fiables como sea posible (por ej. para *BOOTP* y *TFTP*). ***Minimize-Cost*** pide que se minimice el coste de paquetes que atraviesan cada vínculo intermedio hasta el cliente o el servidor (por ej. encontrando la ruta que cueste menos de atravesar; algunos protocolos que podrían requerir ésto son *RTSP* (Real Time Stream Control Protocol) y otros protocolos de transporte de flujos de video/radio). ***Normal-Service***, por último, podría servir para cualquier protocolo que no tenga necesidades especiales.

Comparación TTL

La comparación **TTL** se emplea para filtrar paquetes en función de su campo *TTL* (Time To Live, "tiempo de vida"), el cual se encuentra en las cabeceras IP. Este campo contiene 8 bits de datos y su valor va disminuyendo en una unidad cada vez que el paquete es procesado por un host intermedio entre el host origen y el destinatario. Si el valor *TTL* llega a ser 0, se transmite al remitente un mensaje informándole del problema. Este mensaje es un *ICMP* tipo 11 código 0 (el TTL ha tomado valor nulo durante el tránsito) o código 1 (el TTL ha tomado valor nulo durante el reensamblaje). Esta comparación únicamente efectúa un filtrado dependiendo del valor del *TTL* de los paquetes, pero sin cambiar nada. Para cargar esta comparación, necesitas añadir **-m ttl** a la regla.

Table 6-15. Comparaciones TTL

Comparación	**--ttl**
Ejemplo	**iptables -A OUTPUT -m ttl --ttl 60**
Descripción	Con esta opción se especifica el valor *TTL* a comparar. Este valor será de tipo numérico y se comparará con el valor presente en cada uno de los paquetes. No hay inversión posible y no hay otras opciones. A modo de ejemplo, se puede emplear para efectuar chequeos en busca de fallos (debug) en tu red local, como podrían ser hosts de la red local que parecen tener problemas para conectar con otros hosts situados en Internet; o también se puede emplear para encontrar posibles troyanos que se hayan introducido en tu red. Sin embargo su uso es relativamente limitado y su utilidad depende principalmente de tu imaginación. Por ejemplo, se pueden encontrar hosts con valores anormales de *TTL* (debidos posiblemente a una pila *TCP/IP* mal implementada, o a una simple mala configuración).

Comparación "Unclean" ("sucio")

La comparación **unclean** no tiene opciones y sólo necesita que sea cargada explícitamente cuando quieras usarla. Sin embargo, debes tener en cuenta que se considera experimental y puede que no funcione bien en todo momento, así como tampoco se preocupará de todos los paquetes "sucios" o todos los problemas. La comparación "unclean" intenta filtrar paquetes que parecen malformados o inusuales, como aquellos paquetes con cabeceras incorrectas, checksums (sumas de verificación) incorrectos, ... Este filtro puede utilizarse, por ejemplo, para desechar (**DROP**) conexiones y para buscar flujos incorrectos; pero debes recordar que al no funcionar siempre correctamente, puede cortar conexiones correctas o legales.

Objetivos/Saltos (Targets/Jumps)

Cuando la comparación de una regla encuentra un paquete que coincide con las condiciones que impone (tal como hemos visto hasta ahora), se recurre al objetivo/salto dónde se le indica a la regla qué debe hacer con ese paquete. Hay dos objetivos básicos que trataremos primero: **ACCEPT** y **DROP**, pero antes de empezar veamos brevemente cómo se efectúa un salto.

La orden de salto se ejecuta de la misma manera que la orden de objetivo, excepto en que el salto necesita que exista una cadena dentro de la misma tabla a la que pueda dirigirse. Para crear una cadena, tal como ya se ha explicado, se utiliza el comando **-N**. Por ejemplo, supongamos que queremos crear una cadena en la tabla "filter" llamada **tcp_packets**, por lo que escribiremos:

iptables -N tcp_packets

A partir de entonces podremos referirnos a esa cadena para efectuar un salto siempre que lo necesitemos:

```
iptables -A INPUT -p tcp -j tcp_packets
```

De esta forma todos los paquetes TCP saltarán desde la cadena **INPUT** hasta la cadena **tcp_packets** y serán filtrados por élla. Si algún paquete llega al final de la cadena y no ha habido una concordancia (el paquete no presenta coincidencias con las reglas de la cadena), volverá a la cadena original (en nuestro caso la cadena INPUT) justo después de la regla que originó el salto y seguirá pasando por el resto de reglas de esta cadena. Por el contrario, si alguna de las reglas de la cadena "tcp_packets" acepta (**ACCEPT**) los paquetes, éstos efectuarán el salto/objetivo de la regla y seguirán su curso por el resto de cadenas, sin pasar nunca por el resto de reglas de la cadena de origen (INPUT). Para mayor información de cómo los paquetes atraviesan las tablas y cadenas, léete el capítulo *Atravesando tablas y cadenas*.

Por otra parte, los objetivos especifican la acción a ejecutar con el paquete en cuestión. Así, se podrá aceptar (**ACCEPT**) o desechar (**DROP**) el paquete dependiendo de lo que queramos hacer. Existen además otra serie de acciones que podemos querer hacer con el paquete, y las describiremos más adelante. Debes tener en cuenta que saltar a un objetivo puede dar diferentes resultados, dependiendo del que se trate. Algunos objetivos harán que el paquete deje de atravesar una cadena específica y todas las superiores a élla, como se ha descrito antes (buenos ejemplos de éllo son **DROP** y **ACCEPT**): las reglas que son paradas no pasarán ninguna regla más, ni de esa misma cadena, ni de ninguna cadena "superior" a élla (superiores serán aquellas que han producido saltos hacia la cadena actual: la cadena A produce un salto a la cadena B y ésta última a la cadena C, donde el paquete se para en alguna regla y por tanto deja de atravesar la cadena C y sus superiores, la A y la B). Otros objetivos pueden efectuar alguna acción sobre el paquete, después del cual éste continuará pasando por el resto de reglas. Por ejemplo tenemos los objetivos **LOG**, **ULOG** y **TOS**: con éllos podemos registrar los paquetes, modificarlos y después pasárselos a otras reglas en el mismo conjunto de cadenas. Podemos querer hacer ésto para, por ejemplo, poder modificar tanto el valor del *TTL* como el del *TOS* de un paquete/flujo específico. Algunos objetivos aceptan opciones adicionales (qué valor *TOS* utilizar, ...), mientras otros no necesitan opciones obligatoriamente (aunque se puedan incluir si se desea, como ocurre con los prefijos de los registros, los puertos que se enmascararán, ...). Intentaremos incluir todos estos puntos tal como describamos los objetivos, así que empecemos a ver qué tipo de objetivos hay.

Objetivo ACCEPT

Este objetivo no necesita ninguna opción adicional. En cuanto la especificación de la comparación es satisfecha por un paquete y se indica ACCEPT (aceptar) como objetivo, la regla se acepta y el paquete no continúa atravesando ni la cadena actual, ni cualquier otra de la misma tabla. Advierte sin embargo que un paquete que haya sido aceptado por una cadena todavía podría atravesar las cadenas de otras tablas y podría ser desechado en éllas. Para emplear este objetivo no hay más que indicar **-j ACCEPT**.

Objetivo DNAT

El objetivo **DNAT** se emplea para efectuar traducciones de direcciones de red de destino (*Destination Network Address Translation*), que significa que se emplea para reescribir la dirección IP de destino de un paquete. Con este objetivo en la regla, en cuanto un paquete coincide con la comparación, él y todos los paquetes pertenecientes a ese mismo flujo de datos verán modificada su dirección de destino y serán redirigidos a la red/host/dispositivo adecuado. Este objetivo puede ser extremadamente útil cuando, por ej., tienes un host ejecutando el servidor web dentro de una red local (*LAN*), pero sin una IP real que ofrecerle y que sea válida en Internet. Entonces puedes indicarle al cortafuegos que cuando lleguen paquetes dirigidos a su propio puerto *HTTP* los reenvíe hacia el servidor web real dentro de la red local. También se pueden especificar todo un rango de direcciones IP de destino y el mecanismo **DNAT** escogerá aleatoriamente la dirección IP de destino para cada flujo de datos. Mediante este sistema seremos capaces de abordar una especie de balance de carga.

Observa que el objetivo **DNAT** sólo está disponible en las cadenas *PREROUTING* y *OUTPUT* de la tabla *nat*, así como aquellas cadenas a las que las dos anteriores hayan dirigido un salto (es decir, cualquier cadena a la que se haya saltado desde una de las dos mencionadas). Insistimos en que las cadenas que contengan objetivos **DNAT** no pueden usarse desde otras cadenas, como podría ser el caso de la cadena *POSTROUTING*.

Table 6-16. Objetivo DNAT

Opción	**--to-destination**
Ejemplo	**iptables -t nat -A PREROUTING -p tcp -d 15.45.23.67 --dport 80 -j DNAT --to-destination 192.168.1.1-192.168.1.10**

Descripción	La opción **--to-destination** le indica al mecanismo *DNAT* qué *IP de Destino* establecer en la cabecera IP y dónde enviar los paquetes que concuerden con el filtro. El ejemplo anterior enviará todos los paquetes destinados a la dirección IP 15.45.23.67 a un rango de direcciones *LAN*, en este caso el rango 192.168.1.1 a 192.168.1.10. Ten en cuenta que, tal como se ha dicho, un flujo concreto siempre usará el mismo host, mientras que a cada flujo se le asignará aleatoriamente una dirección IP a la que siempre serán destinados todos los paquetes de ese flujo. También se podría haber especificado una sola dirección IP, en cuyo caso siempre estaríamos conectados al mismo host. También podemos añadir un puerto o rango de puertos a los que será redirigido el tráfico. Para ello sólo tenemos que añadir el puerto o rango de puertos después de ":", como en **--to-destination 192.168.1.1:80** para un puerto único, o como en **--to-destination 192.168.1.1:80-100** para un rango de puertos. Como puedes ver, la sintaxis es la misma para los objetivos **DNAT** y **SNAT**, aunque es evidente que los resultados de uno y otro son totalmente diferentes. Por otra parte, las especificaciones de puertos sólo son válidas en aquellas reglas dónde se especifiquen los protocolos TCP o UDP en la opción **--protocol**.

Puesto que el objetivo **DNAT** exige bastante trabajo para funcionar correctamente, he decidido ampliar la explicación sobre cómo usarlo. Para empezar estudiaremos un pequeño ejemplo acerca de cómo deberían hacerse las cosas normalmente: queremos publicar nuestro sitio web aprovechando nuestra conexión a Internet, aunque sólo tenemos una dirección IP y el *servidor HTTP* se encuentra en nuestra red interna. El cortafuegos tiene la dirección IP externa **$INET_IP**, nuestro *servidor HTTP* tiene la dirección IP interna **$HTTP_IP** y por último el cortafuegos tiene la dirección IP interna **$LAN_IP**. Lo primero que hay que hacer es añadir la siguiente regla a la cadena *PREROUTING* de la tabla nat:

iptables -t nat -A PREROUTING --dst $INET_IP -p tcp --dport 80 -j DNAT --to-destination $HTTP_IP

Desde ahora todos los paquetes que provengan de Internet y se dirijan al puerto 80 de nuestro cortafuegos, serán redirigidos a nuestro servidor *HTTP* interno. Si lo compruebas desde Internet, todo debería funcionar perfectamente. Pero, ¿qué ocurre cuando intentas conectar desde un host de la misma red local a la que pertenece el *servidor HTTP*?. Sencillo, simplemente no podrás. En realidad éste es un problema de enrutado: comenzaremos estudiando lo que ocurre en el caso normal. Para mantener la estructura del ejemplo, consideremos que la máquina externa tiene una dirección IP **$EXT_BOX**.

1. El paquete deja el host (con IP **$EXT_BOX**) que quiere conectar con la página web para dirigirse a **$INET_IP**.
2. El paquete llega al cortafuegos.
3. El cortafuegos efectúa la traducción **DNAT** en el paquete y lo envía a través del resto de cadenas.
4. El paquete deja el cortafuegos y viaja hacia $HTTP_IP.
5. El paquete llega al servidor *HTTP* y la máquina *HTTP* responde a través del cortafuegos (siempre que la base de datos de enrutado tenga definido el cortafuegos como el puente entre la red local y **$EXT_BOX**). En general éste será el puente por defecto del servidor *HTTP*.
6. El cortafuegos deshace la traducción que había hecho con **DNAT** sobre el paquete, de forma que parece que es el cortafuegos el que responde.
7. El paquete de respuesta viaja de vuelta al cliente **$EXT_BOX**.

Ahora consideremos qué ocurre si el paquete se origina en un cliente de la misma red a la que pertenece el servidor *HTTP*. El cliente tiene la dirección IP **$LAN_BOX**, mientras el resto de máquinas mantienen los mismos valores (nombres).

1. El paquete deja **$LAN_BOX** para dirigirse a **$INET_IP**.

2. El paquete llega al cortafuegos.
3. El paquete sufre la traducción **DNAT** y todas las acciones necesarias se toman en consecuencia, si bien al paquete no se le efectúa ninguna traducción **SNAT** y mantiene la misma dirección IP de origen.
4. El paquete sale del cortafuegos y alcanza el servidor *HTTP*.
5. El servidor *HTTP* intenta responder al paquete y observa en las bases de datos de enrutado que el paquete viene de una máquina local de la misma red, por lo que intenta enviar el paquete directamente a la *dirección IP de origen* (que a partir de ese momento se convierte en la *dirección IP de destino*).
6. El paquete llega al cliente, que no sabe qué hacer puesto que el paquete devuelto no proviene del host al que envió la petición original. Por éllo el cliente desecha el paquete y continúa esperando la respuesta "auténtica".

La solución más simple para este problema es efectuar la traducción **SNAT** a todos los paquetes que entren al cortafuegos y a los que sabemos que también se les va a aplicar la traducción **DNAT**. Por ej., consideremos la siguiente regla: vamos a efectuar una traducción *SNAT* a los paquetes que entren al cortafuegos y estén destinados a **$HTTP_IP** en el puerto 80, de forma que parecerá que provengan de **$LAN_IP**. Ésto forzará al servidor *HTTP* a devolver los paquetes a través del cortafuegos, que invertirá la traducción **DNAT** y los reenviará al cliente. La regla será algo así:

iptables -t nat -A POSTROUTING -p tcp --dst $HTTP_IP --dport 80 -j SNAT \ --to-source $LAN_IP

Recuerda que la cadena *POSTROUTING* se procesa en último lugar, después del resto de cadenas, por lo que el paquete será "retraducido" cuando llegue a esa cadena específica. Por éllo comparamos los paquetes en función de su dirección interna.

Esta última regla afectará seriamente el registro de actividades, por lo que es altamente recomendable no emplear este método, si bien el ejemplo todavía es válido para aquellos que no pueden permitirse crear una "zona desmilitarizada" (DMZ) específica o algo parecido. Lo que ocurrirá es que el paquete llegará desde Internet, sus direcciones IP serán traducidas por SNAT y por DNAT, llegando por fin al servidor HTTP (por ejemplo). El servidor sólo verá una petición que viene del cortafuegos y, por tanto, registrará *todas* las peticiones de Internet como si vinieran del cortafuegos.

También pueden haber consecuencias más serias. Piensa en un servidor SMTP de la red local: éste admite peticiones desde la red interna y tienes configurado el cortafuegos de forma que redirija el tráfico SMTP hacia el servidor. Lo que has creado en la práctica es un servidor SMTP "universal" [en inglés se llama "open relay SMTP server", es decir, un servidor de correo que acepta servir mensajes que no ha escrito ningún usuario local, ni van destinados a ningún usuario local], ¡con un registro de eventos horroroso!

Así pues, será mejor que resuelvas estos problemas configurando un servidor DNS distinto y separado para tu red local, o creando una "zona desmilitarizada" (DeMilitarized Zone, DMZ) separada de la red interna. Esta última opción es la preferida, si es que tienes el dinero necesario para éllo.

¿Crees que ya está todo solucionado? Bueno, por ahora sí, salvo que consideres un último aspecto dentro de todo este cuadro. ¿Qué pasaría si el cortafuegos intenta acceder al servidor *HTTP*? ¿Dónde irá a parar? Como puede suponerse, intentará acceder a su propio servidor *HTTP* y no al servidor que hay en **$HTTP_IP**. Para corregir este problema necesitamos añadir una regla **DNAT** más a la cadena *OUTPUT*. Siguiendo con el ejemplo anterior, la regla podría quedar así:

```
iptables -t nat -A OUTPUT --dst $INET_IP -p tcp --dport 80 -j DNAT \ --to-destination $HTTP_IP
```

Con esta última regla todo debería funcionar perfectamente: todas aquellas redes que no pertenezcan a la misma red del servidor *HTTP* funcionarán como la seda; todos los hosts de la misma red del servidor *HTTP* podrán conectar sin tropiezos; por último, el cortafuegos será capaz de efectuar conexiones por sí mismo. Todo funciona bien y no tiene por qué haber ningún tipo de problema.

Note: Cabe destacar que estas reglas sólo controlan cómo se efectúan las traducciones *DNAT* y *SNAT*. Por éllo, quizá necesites otras reglas en la tabla "filter" (en la cadena *FORWARD*) para que los paquetes puedan atravesar el resto de cadenas. No olvides que todos los paquetes han pasado primero por la cadena *PREROUTING* y pueden haber cambiado sus direcciones de destino debido al *DNAT* de esta última cadena.

Objetivo DROP

El objetivo **DROP** (desechar) hace precisamente éso: desechar o "tirar" paquetes y no gastar ni ún segundo más de trabajo del procesador en éllos. Un paquete que llegue a una regla, coincida exactamente con el patrón de búsqueda de la comparación y sea desechado, será inmediatamente bloqueado. Ten en cuenta que esta acción puede llegar a tener en determinadas ocasiones un efecto no deseado, ya que puede dejar sockets (conexiones host-hardware) "muertos" en cualquier host. Una solución más acertada, cuando se puedan dar estas circunstancias, es usar el objetivo **REJECT** (rechazar), especialmente si quieres impedir que los escáners de puertos recojan demasiada información privada, como qué puertos tienes filtrados y otros detalles. Volviendo al objetivo, si se efectúa la acción **DROP** a un paquete dentro de una subcadena, este paquete ya no será procesado en ninguna de las cadenas principales de ninguna tabla: el paquete estará "muerto" de inmediato y, como ya se ha dicho antes, el objetivo no enviará ninguna información en ninguna dirección, ni siquiera a intermediarios como los routers (se puede decir que a partir de la acción de desechar el paquete es como si éste nunca hubiera existido).

Objetivo LOG

El objetivo **LOG** se ha diseñado especialmente para registrar información detallada sobre los paquetes. Ésto podría considerarse ilegal, o simplemente como una forma de buscar defectos y fallos. Este objetivo ofrece información específica sobre los paquetes, como la mayoría de las cabeceras IP y otros datos considerados interesantes. Ésto se consigue a través del servicio de registro del núcleo, normalmente **syslogd**, y puede ser leída directamente con los registros de **dmesg**, o bien mediante otros programas. Es un objetivo excelente para depurar y afinar tus conjuntos de reglas, puesto que puedes ver qué paquetes van dónde y qué reglas se aplican en qué paquetes. También puede ser una gran idea usar el objetivo **LOG** en lugar del objetivo **DROP** mientras estás testeando una regla de la que no estás seguro al 100% en un cortafuegos en servicio que esté usando más gente, pues un error en el conjunto de reglas puede causar problemas severos de conectividad a todos los usuarios. Y si estás generando un registro realmente extenso, puede que encuentres interesante el objetivo **ULOG**, ya que puede escribir los registros directamente en bases de datos como las de *MySQL*.

Note

Si obtienes registros de eventos directamente en pantalla y no es éso lo que quieres, no busques el problema en **iptables** o en *Netfilter*, sinó más bien en la configuración de tu **syslogd** (lo más probable es que el problema esté en el fichero /etc/syslog.conf). Para información sobre este tipo de problema, repásate la ayuda escribiendo en la línea de comando: **man syslog.conf**.

El objetivo **LOG** hoy por hoy tiene 5 opciones que pueden ser interesantes si necesitas información específica, o si quieres establecer diferentes opciones con valores específicos. Las explicamos a continuación.

Table 6-17. Opciones del objetivo LOG

Opción	**--log-level**
Ejemplo	**iptables -A FORWARD -p tcp -j LOG --log-level debug**
Descripción	Esta es la opción con la que le indicas a **iptables** y a **syslog** qué nivel de registro utilizar. Para ver una lista completa de niveles de registro, léete el manual de syslog.conf. Normalmente se presentan los siguientes niveles de registro (o prioridades, como normalmente se les suele llamar): *debug*, *info*, *notice*, *warning*, *warn*, *err*, *error*, *crit*, *alert*, *emerg* y *panic*. El nivel *error* es el mismo que *err*, *warn* es lo mismo que *warning* y *panic* es lo mismo que *emerg*. Sin embargo la primera opción de cada pareja se considera obsoleta y está condenada a desaparecer, por lo que no uses *error*, *warn* ni *panic*. La prioridad define la severidad del mensaje registrado. Todos los mensajes se registran a través del servicio ofrecido por el núcleo. O sea que escribiendo **kern.=info /var/log/iptables** en tu fichero syslog.conf y dejando a continuación que todos tus mensajes de **LOG** de **iptables** usen el nivel de información de registro (info), causará que todos los mensajes aparezcan en el fichero /var/log/iptables. Recuerda que también pueden haber otros mensajes de otras partes del núcleo que utilicen la prioridad de información. Si quieres conocer más detalles acerca del registro, te recomiendo que leas los manuales de **syslog** y syslog.conf, así como otros COMOs (HOWTOs) y ayudas publicadas en Internet.
Opción	**--log-prefix**
Ejemplo	**iptables -A INPUT -p tcp -j LOG --log-prefix "INPUT packets"**

Descripción	Esta opción le dice a **iptables** que añada un prefijo específico a todos los mensajes del registro, lo cual es útil al emplear aplicaciones como **grep** para efectuar un seguimiento de problemas específicos y de registros generados por diferentes reglas. El prefijo puede tener hasta 29 caracteres, incluyendo espacios en blanco y otros caracteres especiales.
Opción	**--log-tcp-sequence**
Ejemplo	**iptables -A INPUT -p tcp -j LOG --log-tcp-sequence**
Descripción	Esta opción añadirá los números de la *Secuencia TCP* (*TCP Sequence*) a cada mensaje registrado. El número de la *Secuencia TCP* es un número especial que identifica a cada paquete e indica dónde encaja dentro de una *secuencia TCP*, además de cómo debe ser reensamblado un flujo. Ten en cuenta que esta opción constituye un riesgo de seguridad si los registros pueden ser leídos por usuarios no autorizados o hasta por el resto del mundo. De la misma forma es un riesgo el acceso no autorizado a cualquier registro que contenga datos generados por **iptables**.
Opción	**--log-tcp-options**
Ejemplo	**iptables -A FORWARD -p tcp -j LOG --log-tcp-options**
Descripción	Esta opción registra las diferentes opciones presentes en las cabeceras de los paquetes *TCP* y puede ser útil cuando se intentan depurar los fallos o lo que pudiera llegar a ir mal. La opción no tiene ninguna variable ni nada parecido, como le ocurre a la mayoría de opciones de **LOG**.
Opción	**--log-ip-options**
Ejemplo	**iptables -A FORWARD -p tcp -j LOG --log-ip-options**

Descripción	Con la opción **--log-ip-options** registraremos la mayoría de opciones presentes en las cabeceras de los paquetes IP. Trabaja exactamente igual que la opción **--log-tcp-options**, salvo porque trabaja sobre las opciones IP. Ésto puede ser útil al tratar de depurar o perseguir determinados delincuentes, además de para corregir errores (de la misma forma que la anterior opción).

Objetivo MARK

El objetivo **MARK** se usa para establecer marcas (marks) de **Netfilter** asociadas a paquetes específicos. Este objetivo sólo es válido en la tabla *mangle* y no funcionará fuera de élla. Los valores de las marcas pueden usarse conjuntamente con las capacidades de enrutado avanzado de Linux, de forma que se envíen diferentes paquetes por diferentes rutas y que se les pueda indicar que usen diferentes disciplinas para hacer cola (qdisc), ... Para mayor información acerca del enrutado avanzado, pásate por *Linux Advanced Routing and Traffic Control HOW-TO*. Ten en cuenta que el valor de la marca no se establece en el paquete mismo, sinó que es un valor asociado al paquete dentro del núcleo. Dicho de otro modo: no puedes esperar que estableciendo una "**MARK**a" en un paquete, ese valor permanezca en otro host. Si es éso lo que buscas, mejor mira en el objetivo **TOS**, que modificará el *valor TOS* de la cabecera IP.

Table 6-18. Opciones del objetivo MARK

Opción	**--set-mark**
Ejemplo	**iptables -t mangle -A PREROUTING -p tcp --dport 22 -j MARK --set-mark 2**

Descripción	La opción **--set-mark** se necesita para establecer una marca, que tomará un valor entero. Por ejemplo, podemos establecer una marca con un valor "2" en un flujo específico de paquetes, o en todos los paquetes de un host específico, y efectuar un enrutado avanzado sobre ese flujo/host para incrementar o disminuir el ancho de banda de la red, ...

Objetivo MASQUERADE

El objetivo **MASQUERADE** se usa básicamente de la misma forma que el objetivo **SNAT**, pero sin requerir ninguna opción **--to-source**. La razón es que el objetivo **MASQUERADE** se creó para trabajar, por ej., con conexiones telefónicas estándar (dial-up), o con conexiones *DHCP*, que obtienen direcciones IP dinámicas al conectar a la red en cuestión. Ésto quiere decir que sólo debes usar el objetivo **MASQUERADE** con conexiones IP asignadas dinámicamente, en las que antes de conectar con el ISP no sabemos la dirección que tendremos. Si tienes una conexión IP estática, debes emplear el objetivo **SNAT** en su lugar.

Enmascarar una conexión significa que estableces la dirección IP utilizada por una tarjeta de red específica, en lugar de utilizar la opción **--to-source**, por lo que la dirección IP se obtiene automáticamente de la información de la tarjeta específica. El objetivo **MASQUERADE** también tiene la particularidad de que las conexiones se olvidan, se pierden, cuando una interfaz se viene abajo (o como se suele decir, "se cuelga"), lo cual es extremadamente bueno si "matamos" una interfaz específica. En este caso, si hubiéramos utilizado el objetivo **SNAT**, podríamos haber llegado a tener un montón de datos viejos de seguimiento de conexión, que podrían haber estado "flotando" por la memoria durante días, devorando casi toda la memoria de seguimiento de conexiones. Así pues el enmascaramiento es, en general, la opción adecuada al trabajar con líneas telefónicas estándar, dónde probablemente se nos asignará una IP diferente cada vez que conectemos. En estos casos al desconectar la conexión se pierde totalmente, puesto que se nos asigna una IP diferente al reconectar, por lo que resulta bastante tonto mantener las entradas existentes del seguimiento de conexiones.

A pesar de todo, es posible usar el objetivo **MASQUERADE** en lugar del **SNAT** aunque tengas una IP estática, si bien no es conveniente puesto que genera un gasto extra de recursos y pueden haber futuras inconsistencias que desbaratarán tus scripts y los harán "inútiles".

Recuerda que el objetivo **MASQUERADE** sólo es válido en la cadena *POSTROUTING* de la tabla nat, igual que el objetivo **SNAT**. Además, el objetivo **MASQUERADE** acepta una opción, aunque ésta sea opcional.

Table 6-19. Objetivo MASQUERADE

Opción	**--to-ports**
Ejemplo	**iptables -t nat -A POSTROUTING -p TCP -j MASQUERADE --to-ports 1024-31000**
Descripción	La opción **--to-ports** se utiliza para establecer el puerto o puertos origen de los paquetes salientes. Puedes especificar un solo puerto, como en **--to-ports 1025**, o puedes especificar un rango de puertos, como en **--to-ports 1024-3000** (con los puertos inicial y final del rango separados por un guión). Esta forma de indicarlo difiere de la selección por defecto de SNAT, tal como se describe en *Objetivo SNAT*. La opción **--to-ports** sólo es válida si en la regla se especifican los protocolos TCP o UDP mediante la comparación **--protocol**.

Objetivo MIRROR

El objetivo **MIRROR** es únicamente experimental y de demostración, por lo que estás avisado para no utilizarlo: puede dar como resultado bucles infinitos (entre otras lindezas) que desemboquen en serias Denegaciones de Servicio (Denial of Service, DoS). Este objetivo se emplea para invertir los campos de origen y destino en la cabecera *IP* y después retransmitir el paquete. Esto puede crear efectos realmente divertidos y apuesto a que gracias a este objetivo más de un endemoniado cracker ha penetrado en su propio ordenador. El resultado de utilizar este objetivo es simple, por decirlo de alguna manera: digamos que establecemos un objetivo **MIRROR** para el puerto 80 del ordenador A y el host B (que proviene de yahoo.com) quiere acceder al servidor *HTTP* del host A; en estas condiciones el objetivo **MIRROR** devolverá al host de yahoo la página web de su propio servidor *HTTP* en Yahoo.com.

Debes tener en cuenta que el objetivo **MIRROR** sólo es válido cuando se utiliza en las cadenas *INPUT*, *FORWARD* y *PREROUTING*, así como en aquellas cadenas definidas por uno mismo y que sean llamadas desde alguna de las anteriores cadenas. Además, los paquetes salientes derivados de este objetivo no pasan por ninguna de las cadenas normales de las tablas filter, nat o mangle, lo que puede dar lugar a bucles infinitos y un rosario de problemas que pueden derivar en dolores de cabeza imprevistos. Por ejemplo, un host puede enviar un paquete camuflado (spoofed) con un **TTL** de 255 a otro host que utilice el objetivo **MIRROR**, de forma que parezca que el paquete viene de un tercer host que también usa el objetivo **MIRROR**; como consecuencia el paquete irá y volverá de un extremo al otro incesantemente, hasta que se completen el número de saltos (hops) definidos; si sólo hay un salto entre hosts, el paquete irá y vendrá 240-255 veces (no está mal para un cracker, pues enviando 1,5 kbytes de datos será capaz de consumir 380 kbytes de la conexión; siempre teniendo en cuenta que éste sería el mejor resultado para el cracker, script kiddie, novato, o como quiera llamársele).

Objetivo QUEUE

El objetivo **QUEUE** se emplea para poner en cola los paquetes dirigidos a programas y aplicaciones del Espacio de Usuario. Se usa con programas o utilidades que son externos a iptables y puede utilizarse, por ej., en contabilidad de red, o con aplicaciones específicas y avanzadas que filtren o den servicios de caché (proxy) a los paquetes. No desarrollaremos este objetivo en profundidad, puesto que la programación de esas aplicaciones excede los objetivos de este tutorial: para empezar costaría mucho tiempo y además esa documentación no tendría nada que ver con la programación de Netfilter e iptables; todo ésto debería estar perfectamente descrito en "COMO Programar Netfilter" (Netfilter Hacking HOW-TO).

Objetivo REDIRECT

El objetivo **REDIRECT** sirve para redirigir paquetes y flujos hacia la máquina. Ésto quiere decir que podemos redirigir todos los paquetes destinados a los puertos *HTTP* hacia un *caché HTTP* como squid, o hacia nuestro host. Los paquetes generados localmente son "mapeados" a la dirección 127.0.0.1, o lo que es lo mismo, se reescribe la dirección de destino con la de nuestro host en los paquetes que son reenviados o acciones similares. El objetivo **REDIRECT** es extremadamente recomendable si queremos un servicio de caché transparente, donde los hosts de la red local ni siquiera advierten que existe un caché (proxy).

Como viene siendo habitual, este objetivo sólo funciona en determinadas cadenas, que en este caso son *PREROUTING* y *OUTPUT*, ambas de la tabla nat. Asímismo, también es válido en cualquier cadena creada por el usuario, siempre que sea llamada por una de las anteriores cadenas. Y en cuanto a las posibles opciones, se reducen a una.

Table 6-20. Objetivo REDIRECT

Opción	**--to-ports**
Ejemplo	**iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080**

Descripción	La opción **--to-ports** especifica el puerto o rango de puertos de destino que se debe usar. Sin esta opción, el puerto de destino nunca se modifica. Si deseamos especificar un solo puerto, escribiremos **--to-ports 8080**, mientras que si deseamos un rango de puertos escribiremos **--to-ports 8080-8090**; en este último caso le estamos indicando al objetivo **REDIRECT** que redirija los paquetes a los puertos comprendidos entre el 8080 y el 8090. Como supondrás, esta opción sólo está disponible en aquellas reglas que especifiquen el protocolo *TCP* o bien el *UDP* con la comparación **--protocol**, puesto que no tendría sentido en ningún otro lugar.

Objetivo REJECT

El objetivo **REJECT** (rechazar) funciona básicamente igual que el objetivo **DROP**, aunque en este caso sí que se devuelve un mensaje de error al host que envió el paquete bloqueado. El objetivo **REJECT**, hoy por hoy, sólo es válido en las cadenas *INPUT*, *FORWARD* y *OUTPUT* o sus "sub-cadenas"; después de todo, son las únicas cadenas donde tiene sentido utilizar este objetivo. De momento sólo hay una opción que modifique su forma de trabajar, pero en cambio tiene un montón de variables dónde escoger (la mayoría son fáciles de entender, siempre que tengas un conocimiento básico sobre *TCP/IP*).

Table 6-21. Objetivo REJECT

Opción	**--reject-with**
Ejemplo	**iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset**

Descripción	Esta opción le indica a **REJECT** qué respuesta devolver al host que envía el paquete que estamos rechazando. Una vez tenemos un paquete que coincide con la comparación de una regla en la que hemos especificado este objetivo, lo primero que hara nuestro host es enviar la respuesta asociada y entonces se desechará el paquete, tal como lo desecharía el objetivo **DROP**. Hoy por hoy tenemos disponibles los siguientes tipos de rechazo: ***icmp-net-unreachable***, ***icmp-host-unreachable***, ***icmp-port-unreachable***, ***icmp-proto-unreachable***, ***icmp-net-prohibited*** y ***icmp-host-prohibited***. El mensaje de error por defecto que se devuelve al host es **port-unreachable**. Todos los mensajes anteriores son *mensajes de error ICMP* y pueden definirse a tu gusto. Puedes encontrar información acerca de sus variados propósitos en el apéndice *Tipos ICMP*. También tenemos la opción **echo-reply**, aunque sólo se puede usar con reglas que capten paquetes *ICMP ping* [cuando el host A hace "ping" sobre el host B, lo que intenta saber es si B está disponible, si (en principio) podría contactar con él]. Por último hay una opción más llamada **tcp-reset**, que sólo puede usarse junto al protocolo *TCP*. Esta opción le dice a **REJECT** que envíe un paquete *TCP RST* como respuesta. Los paquetes *TCP RST* se emplean para cerrar de una forma elegante conexiones *TCP* abiertas. Si quieres saber más sobre *TCP RST* léete este documento: *RFC 793 - Protocolo de Control de Transmisiones*. Tal como se describe en el manual de **iptables** (**man iptables**), el principal uso de esta opción es bloquear sondeos (pruebas) de identificación, cosa que ocurre frecuentemente al enviar correo a servidores de correo mal configurados, que de otra forma no aceptarán tu correo.

Objetivo RETURN

El objetivo **RETURN** hará que el paquete que está atravesando una cadena pare allí donde se encuentre con el objetivo; si está en una "sub-cadena", el paquete continuará atravesando la cadena superior desde donde la dejó; si, por el contrario, es una cadena principal (tal como la cadena *INPUT*), se ejecutará la política por defecto sobre el paquete. La política por defecto normalmente está definida como **ACCEPT**, **DROP** o algo similar.

Por ejemplo, un paquete entra en la cadena INPUT y llega a una regla en la que coincide con su comparación y que tiene definida la acción **--jump CADENA_DE_EJEMPLO** como objetivo; como consecuencia el paquete empezará a atravesar la **CADENA_DE_EJEMPLO** hasta que llegue a una regla donde coincida con la comparación y cuyo objetivo sea **--jump RETURN**; entonces volverá a la cadena *INPUT* y seguirá desde la regla siguiente a la que produjo el salto. Otro ejemplo sería que el paquete llegara a una regla de la cadena *INPUT* y se le aplicara la acción **--jump RETURN**, a consecuencia de lo cual se le aplicaría inmediatamente la política por defecto y abandonaría esta cadena, para seguir por el resto de cadenas de la tabla, si las hay.

Objetivo SNAT

El objetivo **SNAT** se emplea para efectuar las traducciones de dirección de red de origen (*Source Network Address Translation*), lo cual implica que este objetivo reescribirá la dirección IP de origen en la cabecera IP del paquete (ésto es deseable cuando varios hosts tienen que compartir una conexión a Internet). Así pues, podemos activar el reenvío de paquetes IP a nivel de núcleo y escribir una regla **SNAT** que cambiará la dirección **IP de origen** de todos los paquetes que salgan de la red local, por la dirección **IP de origen** de nuestra conexión a Internet. Si no se efectuara esta traducción, el mundo exterior no sabría dónde enviar los paquetes de respuesta, puesto que la mayoría de las redes internas utilizan las direcciones IP especificadas por el IANA para las redes locales. Si los paquetes llegaran tal cual, nadie sabría quién los ha enviado. El objetivo **SNAT** efectúa todas las traducciones necesarias para llevar a cabo esa tarea, haciendo que todos los paquetes que salgan de la red local parezca que salen de un host concreto (por ej. el cortafuegos).

El objetivo **SNAT** sólo es válido en la tabla nat y dentro de la cadena *POSTROUTING*. Sólo el primer paquete de una conexión es modificado por **SNAT**, después del cual todos los paquetes pertenecientes a la misma conexión serán modificados de la misma manera que el primero: el primer paquete llega al cortafuegos y entra en la tabla nat, dónde pasa en primer lugar por la cadena PREROUTING y se le modifica la dirección de destino con DNAT; a continución llega a la cadena POSTROUTING y se le modifica la dirección de origen con SNAT; después sigue su curso a través del resto de tablas, cadenas y reglas; en cambio el segundo y siguientes paquetes del mismo flujo ni siquiera tocan la tabla nat, sinó que se les aplican directamente las mismas traducciones que al primer paquete. Entre otras razones, es por esta forma de trabajar de la tabla nat por lo que no se recomienda filtrar en élla.

Table 6-22. Objetivo SNAT

Opción	**--to-source**
Ejemplo	**iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to-source 194.236.50.155-194.236.50.160:1024-32000**

Descripción	La opción **--to-source** se emplea para especificar qué dirección origen debe utilizar el paquete. La forma más simple de usar esta opción es indicar la dirección IP que deseamos utilizar como dirección **IP de origen** en las **cabeceras IP**. Si desearamos balancear la carga entre varias direcciones IP, podemos utilizar un rango separado por un guión. En el caso del ejemplo anterior el rango se presenta como: 194.236.50.155-194.236.50.160. Considerando este rango, la dirección IP de origen de cada flujo que creemos será asignada aleatoriamente a cualquier dirección de las que comprenda el rango y todos los paquetes de cada flujo particular usarán siempre la misma IP; es decir, en el momento de abrir una conexión se crea un flujo de datos al que se le asignará una dirección IP de origen del rango y todos los paquetes de ese flujo tendrán la misma dirección IP de origen; cuando se cree otro flujo se le asignará otra IP que compartirán todos los paquetes del flujo. También podemos especificar a **SNAT** un rango de puertos, con lo cual todos los puertos de origen serán confinados a los puertos indicados. En el ejemplo anterior esta característica se indica mediante un rango después de "dos puntos", es decir, ":1024-32000". Pero todo ésto sólo es válido si en alguna parte de la comparación se ha especificado -p tcp o -p udp. Además, iptables siempre intentará evitar la alteración de un puerto, pero si dos hosts intentan usar los mismos puertos, iptables le asignará otro puerto a uno de ellos. Si no se especifica ningún rango de puertos, entonces todos los puertos por debajo del 512 se asignarán a otros puertos por debajo del 512 (siempre que sea necesario, claro). De la misma manera, todos los que estén entre los puertos 512 y 1023 serán asignados a puertos por debajo del 1024. Al resto de puertos se les asignará un valor de 1024 o superior. Tal como ya se ha dicho, iptables intentará mantener el puerto origen usado por la estación de trabajo que crea la conexión. Además, fíjate que todo ésto no hace referencia en ningún momento a los puertos de destino, por lo que si un cliente intenta establecer contacto con un servidor **HTTP** que está fuera del cortafuegos, no se le asignará al puerto **FTP control**.

Objetivo TOS

El objetivo **TOS** se emplea para establecer el campo *Tipo de Servicio* (*Type of Service*) de la cabecera IP. Este campo tiene un tamaño de 8 bits y se usa para ayudar a direccionar (enrutar) los paquetes. Éste es uno de los campos que pueden usarse directamente en **iproute2** y su subsistema de políticas de enrutado. Vale la pena destacar que si controlas varios cortafuegos y enrutadores separados, ésta es la única manera de difundir información de encaminamiento (enrutado) junto a los paquetes que circulan entre esos enrutadores y cortafuegos. Como ya se ha comentado, el objetivo **MARK** (que establece una "**MARKa**" asociada a un paquete específico) sólo está disponible dentro del núcleo y no se puede enviar junto con el paquete. Si necesitas enviar información de enrutado con un paquete o flujo específicos, debes recurrir al *campo TOS*, pues se diseñó para éso.

En la actualidad hay muchos enrutadores en Internet que trabajan muy mal con este tema, por lo que puede ser bastante inútil intentar modificar el campo *TOS* antes de enviar los paquetes a Internet. Como mucho los enrutadores no prestarán atención al *campo TOS*. En el peor de los casos, lo leerán y harán alguna cosa incorrecta. Sin embargo, tal como se ha dicho antes, el *campo TOS* puede ser muy útil si tienes una *WAN* (Red de Área Extensa) o una *LAN* (Red de Área Local) de dimensiones respetables y con múltiples enrutadores. De hecho tienes la posibilidad de darle a los paquetes diferentes encaminamientos y preferencias, basándote en el valor de su *TOS* (aunque ésto sólo puedas tenerlo bajo control en tu propia red).

Caution

El objetivo **TOS** sólo es capaz de establecer valores o nombres específicos a los paquetes. Estos valores predefinidos se pueden encontrar en los archivos del "kernel include" (ficheros de código fuente del núcleo), o más exactamente en el archivo `Linux/ip.h`. Las razones son muchas y de hecho no deberías necesitar ningún otro valor; sin embargo, hay formas de superar esta limitación: para evitar la limitación de sólo poder establecer valores nominales (con nombres) a los paquetes, puedes usar el parche FTOS disponible en *Paksecured Linux Kernel patches*, web mantenida por Matthew G. Marsh; sin embargo, ¡sé extremadamente cauteloso con este parche!, pues nunca deberías tener la necesidad de utilizar ningún valor aparte de los valores por defecto, a excepción de casos extremos.

Ten en cuenta que este objetivo sólo es válido dentro de la tabla *mangle* y no puede usarse fuera de élla.

También es importante que entiendas que algunas viejas versiones de iptables (1.2.2 o anteriores) proporcionan una implementación incompleta de este objetivo y no corrigen la suma de comprobación (checksum) del paquete a la hora de modificarlo, por lo que generan mal los paquetes y como resultado necesitan una retransmisión. Ésto lo más probable es que obligue a una nueva modificación y con éllo que el ciclo vuelva a empezar, dando como resultado que la conexión nunca funcione.

El objetivo **TOS** sólo tiene la opción que explico a continuación.

Table 6-23. Objetivo TOS

Opción	**--set-tos**
Ejemplo	**iptables -t mangle -A PREROUTING -p TCP --dport 22 -j TOS --set-tos 0x10**

Descripción	Esta opción le dice al procedimiento de modificación **TOS** qué valor establecer en los paquetes que coincidan con el filtro. Este valor es numérico (hexadecimal o decimal). Como el valor *TOS* tiene 8 bits, el número estará comprendido entre 0 y 255 (en hexadecimal entre 0x00 y 0xFF). Recuerda que en el objetivo TOS estándar estás limitado a los valores nominales (con nombre) disponibles, que deberían estar más o menos estandarizados. Estos valores estándar son: ***Minimize-Delay*** [minimizar el retardo] (valor decimal/dec 16, valor hexadecimal/hex 0x10), ***Maximize-Throughput*** [maximizar el rendimiento] (valor dec 8, valor hex 0x08), ***Maximize-Reliability*** [maximizar la fiabilidad] (valor dec 4, valor hex 0x04), ***Minimize-Cost*** [minimizar el coste] (valor dec 2, valor hex 0x02) ó ***Normal-Service*** [servicio normal] (valor dec 0, valor hex 0x00). El valor por defecto en la mayoría de paquetes es ***Normal-Service***, ó 0. Además, obviamente puedes utilizar los nombres en lugar de los valores hexadecimales (y viceversa) para establecer el valor *TOS*; sin embargo, normalmente se recomienda utilizar el nombre, pues los valores asociados con los nombres pueden cambiar en el futuro. Para ver una lista completa de los "valores descriptivos", ejecuta en la línea de comandos: **iptables -j TOS -h**. Esta lista se considera completa en la versión 1.2.5 de iptables y debería seguir así por un tiempo.

Objetivo TTL

Este objetivo necesita del parche **TTL** del patch-o-matic, en el directorio raíz, en *http://www.netfilter.org/documentation/index.html#FAQ - Las Frequently Asked Questions (Preguntas Frecuentes) oficiales de Netfilter*.

El objetivo **TTL** se usa para modificar el campo *Time To Live* de la cabecera IP. Una aplicación útil para ésto puede ser cambiar todos los valores *Time To Live* al mismo valor en todos los paquetes dirigidos al exterior de la red. Una razón para necesitar ésto es que tengas un *ISP* abusivo que no te permita tener más de una máquina conectada a la misma conexión de Internet, y que lo persiga activamente. Estableciendo todos los valores **TTL** al mismo valor le harás un poco más difícil (al ISP) darse cuenta de lo que estás haciendo. Así pues, podemos establecer el valor **TTL** de todos los paquetes de salida a un valor estandarizado, como 64, que es el valor estándar del núcleo de Linux.

Para más información sobre cómo establecer el valor por defecto usado por Linux, léete el *ip-sysctl.txt*, que puedes encontrar en el apéndice *Otras fuentes y enlaces*.

El objetivo **TTL** sólo es válido en la tabla *mangle* y en ninguna tabla más. De momento tiene 3 opciones que describo a continuación.

Table 6-24. Objetivo TTL

Opción	**--ttl-set**
Ejemplo	**iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-set 64**
Descripción	La opción **--ttl-set** le dice al objetivo **TTL** qué valor *TTL* establecer en el paquete. Un valor de referencia bastante correcto podría estar alrededor de 64, un valor ni muy alto ni muy bajo. Lo que no debes hacer es establecer un valor demasiado alto, pues puede afectar a tu red local y además resulta un poco inmoral, puesto que el paquete puede empezar a rebotar entre dos enrutadores mal configurados y cuanto más alto sea el *TTL*, más ancho de banda consumirá innecesariamente en un caso así. Este objetivo puede emplearse para limitar lo alejados que pueden estar nuestros clientes, como por ej. en el caso de los servidores DNS, dónde no queremos que los clientes estén muy lejos.
Opción	**--ttl-dec**

Ejemplo	**iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-dec 1**
Descripción	La opción **--ttl-dec** hace que se decremente el tiempo de vida del paquete una cantidad equivalente al valor indicado tras la opción (en el ejemplo, el decremento indicado es de uno). Así pues, si el TTL de un paquete entrante es de 53 y hemos establecido un decremento **--ttl-dec 3**, el paquete verá reducido su TTL hasta el valor 49. La razón de esta diferencia estriba en que el código de gestión de redes reducirá automáticamente el TTL en un punto, por lo que el paquete sufrirá un decremento de 3+1, desde 53 hasta 49. Mediante este sistema podemos limitar lo alejados que pueden estar los usuarios de nuestros servicios. Por ejemplo, los usuarios siempre deben usar un DNS cercano, y en consecuencia podemos comparar todos los paquetes que dejen nuestro servidor DNS y reducir el TTL en varias unidades. Por supuesto, la opción **--set-ttl** es más adecuada para este uso.
Opción	**--ttl-inc**
Ejemplo	**iptables -t mangle -A PREROUTING -i eth0 -j TTL --ttl-inc 1**

Descripción	La opción **--ttl-inc** es la opuesta a la anterior, es decir, se incrementa el TTL en una cantidad expresada a continuación de la opción. Ten en cuenta que en este caso el código de gestión de redes también actúa automáticamente reduciendo el TTL en 1, por lo que si escribimos **--ttl-inc 4**, un paquete entrante con un *TTL* de 53 dejará el host con un valor *TTL* de 56, ésto es, 53+4-1. Con esta opción podemos hacer nuestro cortafuegos un poco más invisible a los trazadores de rutas (traceroutes), entre otras cosas: al establecer el valor *TTL* de los paquetes entrantes a un valor más alto, conseguimos que el cortafuegos se oculte a los trazadores. Los trazadores de rutas son utilidades amadas y odiadas al mismo tiempo, puesto que facilitan información excelente sobre problemas con las conexiones y dónde ocurren, si bien al mismo tiempo le ofrecen al hacker/cracker que te haya localizado muy buena información sobre tus conexiones con el exterior. Para ver un buen ejemplo de cómo puedes usarlo, mira el script *Ttl-inc.txt*.

Objetivo ULOG

Este objetivo se emplea para proporcionar capacidades de registro en el espacio de usuario a los paquetes que concuerden. Si un paquete coincide con la comparación del filtro (de la regla) y se ha establecido el objetivo **ULOG**, la información del paquete es difundida mediante una "multidifusión" (multicast), junto con el paquete entero, a través de una conexión virtual de red (netlink socket). A partir de entonces, uno o varios procesos de espacio de usuario pueden subscribirse a varios grupos de multidifusión y recibir el paquete. Ésta es una capacidad más completa y sofisticada de registro que hasta ahora sólo utilizan iptables y Netfilter y que tiene mucha más capacidad para el registro de paquetes. Este objetivo nos permite registrar información en bases de datos *MySQL*, entre otras, haciendo más fácil la búsqueda de paquetes específicos y el agrupamiento de entradas de registro. Puedes encontrar las aplicaciones de espacio de usuario de ULOGD en *Página del proyecto ULOGD*.

Table 6-25. Objetivo ULOG

Opción	**--ulog-nlgroup**
Ejemplo	**iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-nlgroup 2**
Descripción	La opción **--ulog-nlgroup** le dice al objetivo **ULOG** a qué grupo de conexión de red enviar el paquete. Existen 32 grupos de conexión de red, que se especifican con un núemero del 1 al 32, o sea que si queremos contactar con el grupo 5, simplemente escribiremos **--ulog-nlgroup 5**. El grupo por defecto es el 1.
Opción	**--ulog-prefix**
Ejemplo	**iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-prefix "SSH connection attempt: "**
Descripción	La opción **--ulog-prefix** trabaja igual que el prefijo del objetivo **LOG** estándar: esta opción le añade un prefijo definido por el usuario a todas las entradas de registro. Este prefijo puede tener hasta 32 caracteres y es realmente útil para diferenciar distintos mensajes de registro y de dónde vienen.
Opción	**--ulog-cprange**
Ejemplo	**iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-cprange 100**

Descripción	La opción **--ulog-cprange** le indica a **ULOG** cuántos bytes del paquete enviar al demonio de espacio de usuario de **ULOG**. Si especificamos un valor de 100, copiaremos 100 bytes del total del paquete al espacio de usuario, lo cual en principio incluirá la cabecera completa, más algunos datos básicos del paquete. Si por el contrario especificamos 0, se copiará el paquete entero al espacio de usuario sin importar su tamaño. El valor por defecto es 0, y por lo tanto se copiará el paquete completo al espacio de usuario.
Opción	**--ulog-qthreshold**
Ejemplo	**iptables -A INPUT -p TCP --dport 22 -j ULOG --ulog-qthreshold 10**
Descripción	Esta opción indica el número de paquetes a poner en cola dentro del núcleo antes de enviar realmente los datos al espacio de usuario. Por ejemplo, si establecemos un umbral de 10 como en el ejemplo, el núcleo acumulará 10 paquetes antes de transmitirlos al espacio de usuario mediante un sólo mensaje multi-parte. El valor por defecto es 1 a causa de la compatibilidad con versiones más antiguas (el demonio del espacio de usuario no sabía cómo manejar mensajes multi-parte).

Chapter 7. El archivo rc.firewall

Este capítulo tratará sobre una configuración ejemplo del cortafuegos y cómo podría quedar el script. Hemos empleado una configuración básica, para después profundizar en su forma de trabajar y en qué modificaciones le hemos hecho. Con éllo obtendremos una idea general de cómo resolver diferentes problemas y de lo que necesitas pensar antes de poner a trabajar tus scripts. Se puede utilizar tal cual con algunos cambios en las variables, aunque no es recomendable puesto que es posible que no funcione correctamente junto a la configuración de tu red. Sin embargo, si tienes una configuración muy básica, es más que probable que funcione bastante bien con sólo hacerle algunos cambios.

> Ten en cuenta que pueden haber formas más eficientes de crear un conjunto de reglas, pero el script se ha escrito con la legibilidad en mente, de forma que cualquiera pueda entenderlo sin necesidad de tener grandes conocimientos de programación BASH (BASH scripting) antes de leerlo.

Ejemplo de rc.firewall

De acuerdo, ya tienes todo listo y estás preparado para revisar un ejemplo de script de configuración. O al menos deberías estarlo si has llegado hasta aquí. El ejemplo *rc.firewall.txt* (también incluído en el apéndice *Código fuente de los scripts de ejemplo*) es bastante largo pero sin demasiados comentarios. En vez de buscar los comentarios, te recomiendo que leas el script entero para tener una idea general y luego vuelvas a estas líneas para obtener todos los detalles prácticos.

Explicación del rc.firewall

Opciones de configuración

La primera sección que observarás en el ejemplo rc.firewall.txt (http://iptables-tutorial.frozentux.net/scripts/rc.firewall.txt) es la de la configuración. Siempre debería ser modificada, pues contiene la información que resulta imprescindible para tu configuración actual. Por ejemplo, tu dirección IP normalmente será distinta, por lo cual se especifica aquí. La variable **$INET_IP** siempre debería ser una dirección IP válida, si es que tienes una (si no es así, probablemente deberías ver *rc.DHCP.firewall.txt*, pero de todas formas sigue leyendo este script, ya que ofrece una introducción de bastantes cosas interesantes). Del mismo modo, la variable **$INET_IFACE** debe señalar al adaptador utilizado para tu conexión a Internet, como puede ser *eth0*, *eth1*, *ppp0*, *tr0*, etc, por nombrar unos pocos de los adaptadores posibles.

Este script no contiene ninguna configuración especial para DHCP o PPPoE, por lo que estas secciones permanecen vacías. Lo mismo ocurre con el resto de secciones vacías, que sin embargo se mantienen para que puedas ver las diferencias entre los diferentes scripts de una forma más efectiva. Si por algún motivo necesitaras estas configuraciones, siempre puedes crear una mezcla a partir de los diferentes scripts o también crear el tuyo propio desde cero.

La sección *Configuración de la red local (LAN)* incluye la mayoría de las opciones necesarias de configuración de tu *LAN*. Por ejemplo, necesitas especificar el nombre y la *dirección IP* de la interfaz física conectada a tu *LAN*, así como el *rango IP* que utiliza la red local.

Asímismo, podrás observar que hay una sección de configuración del host local. Sin embargo es casi seguro que no tendrás que cambiar ningún valor en esta sección, puesto que normalmente utilizarás "127.0.0.1" como *dirección IP* y la interfaz casi con toda seguridad se llamará *lo*.

Por fin, justo bajo la configuración del host local, verás una breve sección perteneciente a iptables. En principio sólo contiene la variable **$IPTABLES**, que le indica al script la situación correcta de la aplicación **iptables** en tu instalación. El lugar concreto puede variar algo, aunque al compilar a mano el paquete iptables la ruta por defecto es /usr/local/sbin/iptables. Sin embargo, muchas distribuciones sitúan la aplicación en otro directorio, como puede ser /usr/sbin/iptables, por ejemplo.

Carga inicial de módulos extra

Para empezar, vemos que se comprueba si los archivos de dependencias de módulos están actualizados ejecutando el comando **/sbin/depmod -a**. A continuación se cargan los módulos que se requerirán en el script. Evita siempre cargar módulos que no vayas a necesitar y si es posible trata de evitar tener módulos sólo "por si acaso" a no ser que los vayas a usar. La razón de ésto es por seguridad, ya que de otra manera supondrá un esfuerzo extra escribir reglas adicionales. Así, si por ejemplo quieres tener soporte para los objetivos ***LOG***, **REJECT** y **MASQUERADE** y no los tienes compilados estáticamente en el núcleo, deberás cargar los módulos como sigue:

/sbin/insmod ipt_LOG
/sbin/insmod ipt_REJECT
/sbin/insmod ipt_MASQUERADE

Caution

En éste y los demás scripts de ejemplo forzamos la carga de módulos, lo cual puede originar fallos durante el proceso de carga. Estos fallos pueden deberse a muchos factores y generarán mensajes de error. Si alguno de los módulos más básicos no se cargan, el error más probable es que el módulo o funcionalidad esté compilado estáticamente en el núcleo. Para más información al respecto, léete la sección

Problemas en la carga de módulos del apéndice *Problemas y preguntas frecuentes*.

Seguimos con los módulos no requeridos y encontramos la opción de cargar el módulo ***ipt_owner***, que entre otras cosas se puede utilizar para permitir únicamente a determinados usuarios la posibilidad de establecer determinadas conexiones. Este módulo no se utiliza en el presente ejemplo, pero básicamente podrías permitir sólo al usuario *root* establecer conexiones *FTP* y *HTTP* con "redhat.com" y **DROP** (denegárselo/desechar) a todos los demás. También puedes denegar el acceso desde tu máquina hacia Internet a todos los usuarios excepto a tu propio usuario (tú mismo) y a root, lo cual puede no gustar a los demás, pero estarás un poco más seguro ante los ataques tipo "bouncing" (de forma que parezca que la conexión la realiza tu host) y ante los ataques en que el hacker sólo utilizará tu host como máquina intermedia. Para mayor información sobre la comparación ***ipt_owner*** léete la sección *Comparación Propietario (Owner)* del capítulo *Cómo se escribe una regla*.

En este momento también podemos cargar módulos extra para la comparación de estados. Todos los módulos que mejoran el comportamiento del código de comparación de estados y de seguimiento de las conexiones se llaman *ip_conntrack_** e *ip_nat_**. Los asistentes para el seguimiento de las conexiones son módulos especiales que le indican al núcleo cómo hacer un seguimiento adecuado de conexiones de un tipo específico. Sin estos asistentes, el núcleo no sabrá qué buscar al intentar hacer un seguimiento de esas conexiones específicas. Por otra parte, los asistentes *NAT* son a su vez extensiones de los anteriores asistentes y le indican al núcleo qué buscar en paquetes específicos, además de cómo interpretarlos para que las conexiones funcionen. Por ejemplo, *FTP* es un protocolo complejo por definición, que envía la información de conexión dentro del contenido del paquete. Así pues, si una de tus máquinas "*NAT*eadas" conecta con un servidor *FTP* de Internet, enviará su propia *dirección IP* de la red local dentro del paquete y le indicará al servidor *FTP* que realice la conexión con esa dirección. Puesto que las direcciones locales no son válidas fuera de tu propia red local, el servidor *FTP* no sabrá qué hacer con éllas y la conexión se perderá.

Los asistentes *FTP NAT* efectúan las conversiones necesarias en estas conexiones, de manera que el servidor *FTP* sabrá con quién debe establecer la conexión. El mismo caso ocurre con las transferencias (envíos) *DCC* de archivos y los chats. La creación de este tipo de conexiones precisa la *dirección IP* y los puertos con los que conectar a través del *protocolo IRC*, lo cual exige ciertas "traducciones". Sin estos asistentes determinadas tareas con *FTP* e *IRC* se podrán llevar a cabo, qué duda cabe, pero habrá otras que no. Por ejemplo, serás capaz de recibir ficheros mediante *DCC*, pero no podrás enviarlos. Éllo es debido a la manera de establecer conexiones de *DCC*. Al iniciar una conexión, le indicas al receptor que quieres enviarle un fichero y dónde debe conectarse. Sin los asistentes para la conexión *DCC*, este aviso inicial parecerá indicarle al receptor que debe conectarse a algún host de su propia red local (la del receptor). Como resultado la conexión se perderá. Sin embargo, cuando seas tú el que descargue ficheros no habrá problemas puesto que lo más probable es que el que te diga dónde debes conectar, te envíe la dirección correcta.

Note

Si estás experimentando problemas con los mIRC DCCs y tu cortafuegos, mientras que todo funciona perfectamente con otros clientes IRC, léete la sección *Problemas con mIRC DCC* del apéndice *Problemas y preguntas frecuentes*.

En el momento de escribir estas líneas, sólo existe la opción de cargar módulos que añadan soporte para los protocolos *FTP* e *IRC*. Si necesitas una explicación más profunda sobre estos módulos (conntrack y nat), lee el apéndice *Problemas y preguntas frecuentes*. También se pueden encontrar asistentes para seguimiento de conexiones (conntrack) de tipo *H.323* (entre otros) en el *patch-o-matic*, así como otros asistentes para *NAT*. Para poder utilizarlos tienes que utilizar el *patch-o-matic* y compilar tu propio núcleo. Para ver una explicación detallada de cómo hacer ésto, léete el capítulo *Preparativos*.

Ten en cuenta que necesitas cargar los módulos *ip_nat_irc* e *ip_nat_ftp* si quieres que funcione correctamente la traducción de direcciones (*Network Addres Translation*) tanto en el protocolo *FTP* como en el *IRC*. Además, tendrás que cargar los módulos *ip_conntrack_irc* e *ip_conntrack_ftp* antes de cargar los módulos *NAT* anteriores. Se emplean de la misma forma que los módulos conntrack y permitirán al ordenador efectuar traducciones *NAT* en estos dos protocolos.

Configuración de /proc

Es ahora cuando arrancamos el reenvío IP (*IP forwarding*) al enviar el valor "1" a /proc/sys/net/ipv4/ip_forward con el comando siguiente:

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

Puede valer la pena pensar dónde y cuándo arrancamos el reenvío IP. En este script, como en el resto de este tutorial, se arranca antes de crear cualquier tipo de *filtros IP* (o sea, los conjuntos de reglas de **iptables**). Esto implica que durante un breve lapso de tiempo el cortafuegos aceptará el reenvío de cualquier tipo de tráfico desde cualquier lugar, estando este lapso comprendido entre un milisegundo y algunos minutos, dependiendo del script que estemos ejecutando y de la velocidad de la máquina. Así pues, la "gente interesada" (los "chicos malos") tendrá su oportunidad de atravesar el cortafuegos. Como podrás comprender en realidad esta opción debería ejecutarse *después* de crear todas las reglas del cortafuegos, si bien he decidido hacerlo antes para mantener una estructura consistente en todos los scripts.

Si necesitaras soporte para *IPs* dinámicas, como cuando usas *SLIP*, *PPP* o *DHCP*, puedes activar la siguiente opción (ip_dynaddr) ejecutando lo siguiente:

echo "1" > /proc/sys/net/ipv4/ip_dynaddr

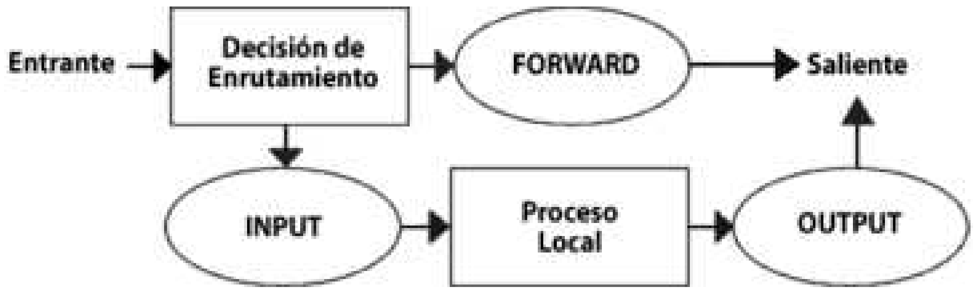
Si existe cualquier otra opción que necesites arrancar, debes hacerlo de la forma indicada en los dos ejemplos anteriores. Existen otros documentos que explican cómo hacerlo de otra manera, aunque ésto queda fuera de los objetivos del tutorial. Puedes leer un buen (y corto) documento sobre el sistema proc disponible con el núcleo y que también tienes en el apéndice *Otras fuentes y enlaces*. Este apéndice es un buen punto de partida cuando busques información de áreas específicas que no hayas encontrado aquí.

> Note
>
> El script rc.firewall.txt, así como el resto de scripts de este tutorial, contiene una pequeña sección de configuraciones no requeridas de proc, las cuales pueden ser un buen punto de partida cuando algo no funciona tal como deseas. Sin embargo, no cambies nada antes de saber qué significa.

Desplazamiento de las reglas entre cadenas

En esta sección se describirán brevemente las alternativas que he escogido en el tutorial en lo que concierne a las cadenas específicas de usuario y algunas opciones más específicas del script rc.firewall.txt. Puedo estar equivocado en la forma de enfocar algunos temas, pero espero explicarlos con sus posibles problemas allí donde ocurran y en el momento en que ocurran. Además también se recuerda brevemente el capítulo *Atravesando tablas y cadenas*. Esperemos que la ayuda de un ejemplo real sea suficiente para que recuerdes cómo se atraviesan las tablas y cadenas.

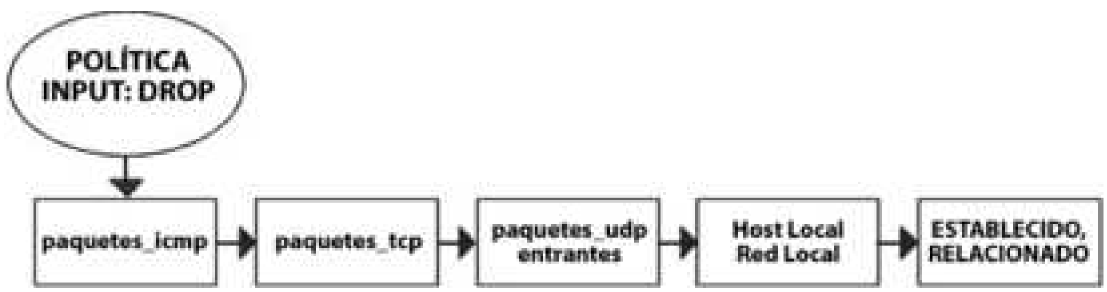
He dispuesto las diferentes cadenas de usuario de forma que ahorren el máximo de CPU, manteniendo al mismo tiempo la máxima seguridad y legibilidad posibles. En lugar de dejar a los paquetes *TCP* atravesar las reglas *ICMP*, *UDP* y *TCP*, simplemente escojo los paquetes *TCP* y les hago atravesar una cadena de usuario. De esta forma no sobrecargamos en exceso el sistema. La siguiente imagen intentará explicar los fundamentos sobre el camino que sigue un paquete entrante al atravesar Netfilter. Con todo éllo espero aclarar los objetivos de este script. Todavía no vamos a discutir detalles específicos, pues ésto se hará más adelante, en este mismo capítulo. En realidad es una simplificación de la imagen que puedes encontrar en el capítulo *Atravesando tablas y cadenas*, dónde se discutió el proceso completo en profundidad.



Basándonos en este gráfico, vamos a dejar claro qué nos hemos planteado. La totalidad de este ejemplo asume que el escenario al que nos referimos tiene una red local conectada a un cortafuegos y el cortafuegos conectado a Internet. También se asume que tenemos una dirección *IP* estática para conectar con Internet (todo lo contrario a cuando utilizamos *DHCP*, *PPP*, *SLIP* u otros). En este caso, queremos permitir al cortafuegos que actúe como servidor de ciertos servicios de Internet y además confiamos plenamente en nuestra red local, por lo que no bloquearemos ningún tráfico desde ésta. Por fin, este script tiene como principal prioridad permitir únicamente el tráfico que explícitamente deseamos permitir. Para conseguirlo, estableceremos las políticas por defecto de las cadenas como *DROP* (desechar). En la práctica ésto eliminará todas las conexiones y paquetes que no hayamos permitido explícitamente dentro de nuestra red o nuestro cortafuegos.

En otras palabras, deseamos que la red local pueda conectar con Internet. Puesto que la red local es de plena confianza, deseamos permitir todo tipo de tráfico desde la red hacia Internet. Sin embargo, Internet es una red a la que no concedemos confianza alguna, por lo que deseamos bloquear cualquier acceso desde élla hacia nuestra red local. Así pues, en base a estas premisas generales, veamos qué debemos hacer y qué es lo que no queremos ni debemos hacer.

Para empezar, ya hemos dicho que deseamos que la red local sea capaz de conectarse a Internet. Para conseguirlo necesitaremos traducir mediante *NAT* todos los paquetes, ya que ningún ordenador de la red local tiene una dirección IP real [refiriéndonos a la conexión directa a Internet]. Ésto lo haremos dentro de la cadena *PREROUTING*, que se crea al final de este script. Esto significa que también tendremos que efectuar algún filtrado en la cadena *FORWARD*, ya que de lo contrario cualquiera desde el exterior tendrá acceso total a nuestra red. Confiamos al máximo en la red local y por éllo permitimos específicamente todo tráfico desde ésta hacia Internet. Ya que no queremos permitir a nadie del exterior acceder a la red local, bloquearemos todo tráfico desde el exterior excepto las conexiones establecidas y las relacionadas, lo cual significa que permitiremos todo el tráfico de retorno desde Internet hacia la red local.



En cuanto al cortafuegos, quizá estemos algo cortos de capital, o quizá deseemos ofrecer algunos servicios a los navegantes de Internet. Así pues, hemos decidido permitir el acceso a *HTTP*, *FTP*, *SSH* e *IDENTD* a través del cortafuegos. Todos estos protocolos están disponibles en el cortafuegos, por lo que deben ser admitidos en la cadena *INPUT*, así como también debe estar permitido todo el tráfico de retorno (hacia Internet) a través de la cadena *OUTPUT*. Sin embargo, seguimos confiando completamente en la red local y tanto el periférico de bucle local como su *dirección IP* también son de confianza. Debido a ésto queremos añadir reglas especiales para permitir todo el tráfico desde la red local así como la interfaz de bucle local. Además, no queremos permitir el tráfico de paquetes específicos o cabeceras de paquetes específicas en coyunturas específicas, así como tampoco deseamos que determinados rangos de direcciones IP puedan alcanzar el cortafuegos desde Internet (por ejemplo, el rango de direcciones *10.0.0.0/8* está reservado para las redes locales, de aquí que normalmente no queramos dejar pasar paquetes provenientes de este rango, ya que en un 90% de los casos serán intentos de acceso falseados mediante spoofing; sin embargo, antes de implementar ésto, debes tener en cuenta que determinados Proveedores de Servicios de Internet utilizan este rango en sus propias redes; si te interesa una explicación más detallada sobre este problema, lee el capítulo *Problemas y preguntas frecuentes*).

Puesto que tenemos un servidor *FTP* ejecutándose en el servidor y deseamos que los paquetes atraviesen el menor número de reglas posibles, añadimos una regla al principio de la cadena *INPUT* que permita el paso a todo el tráfico establecido y relacionado. Por la misma razón deseamos dividir las reglas en subcadenas, de manera que los paquetes sólo tengan que atravesar el menor número de reglas posible. Con éllo conseguiremos que el conjunto de reglas consuman el menor tiempo posible con cada paquete, eliminando redundancias en la red.

En este script decidimos dividir los diferentes paquetes según las familias de protocolos, por ejemplo *TCP*, *UDP* o *ICMP*. Todos los paquetes *TCP* atraviesan una cadena específica llamada "*tcp_packets*", que contendrá reglas para todos los puertos y protocolos *TCP* que queramos permitir. Además, deseamos efectuar algunos chequeos extra a los paquetes *TCP*, por lo que nos gustaría crear otra subcadena para todos aquellos paquetes que sean aceptados por utilizar números de puerto válidos en el cortafuegos. Decidimos llamar a esta subcadena "*allowed*", y debe contener unos pocos chequeos para acabar de aceptar los paquetes. En cuanto a los paquetes *ICMP*, atravesarán la cadena "*icmp_packets*". Cuando se decidió el contenido de esta cadena, no se vió necesario realizar ningún chequeo extra antes de aceptar los paquetes, siempre que coincidieran con los tipos y códigos permitidos; así pues se aceptan directamente. Por último tenemos que manejar los paquetes *UDP* y los enviaremos a la cadena "*udp_packets*". Todos los paquetes que coincidan con los tipos permitidos serán aceptados de inmediato sin más chequeos.

Dado que estamos trabajando con una red relativamente pequeña, esta máquina también se utiliza como estación de trabajo secundaria y para imponer una carga extra debido a éllo, queremos permitir que determinados protocolos específicos puedan contactar con el cortafuegos, como **speak freely** e **ICQ**.

Por último, tenemos la cadena *OUTPUT*. Ya que confiamos bastante en el cortafuegos, permitimos casi todo el tráfico de salida. No efectuamos ningún bloqueo específico de usuario, así como tampoco efectuamos ningún bloqueo de protocolos específicos. Sin embargo, no queremos que la gente utilice esta máquina para camuflar (spoof) paquetes que dejen el cortafuegos, por lo que sólo permitiremos el tráfico desde las direcciones asignadas al cortafuegos. Lo más probable es que implementemos ésto añadiendo reglas que acepten (*ACCEPT*) todos lo paquetes que dejen el cortafuegos siempre que procedan de una de las *direcciones IP* que le han sido asignadas, siendo desechados como política por defecto todos los que no cumplan esta premisa.

Estableciendo las políticas por defecto

Casi al comenzar el proceso de creación de nuestro conjunto de reglas, establecemos las políticas por defecto en las diferentes cadenas con un comando bastante simple, como puedes ver a continuación:

iptables [-P {cadena} {política}]

La política por defecto se utiliza siempre que los paquetes no coincidan con ninguna regla de ninguna cadena. Por ejemplo, supongamos que llega un paquete que no encaja en ninguna regla de todas las que tenemos en nuestro conjunto de reglas. Si ésto ocurre debemos decidir qué debe pasarle al paquete en cuestión y es aquí donde entra en escena la política por defecto.

Se cuidadoso con la política por defecto que estableces en las cadenas de otras tablas, pues no están pensadas para filtrar como la tabla FILTER y pueden desembocar en comportamientos realmente extraños.

Definiendo cadenas de usario en la tabla Filter

Ahora ya tienes una buena idea de qué queremos conseguir con el cortafuegos, así que empecemos con la creación del conjunto de reglas. Es el momento de cuidarnos de especificar todas las reglas y cadenas que deseamos crear y utilizar, así como todos los conjuntos de reglas en las cadenas.

Después de esta introducción, empezaremos por crear las diferentes cadenas especiales que necesitamos, mediante el comando **-N**. Las nuevas cadenas se crean vacías, sin ninguna regla en éllas, y tal como ya se ha dicho son: *icmp_packets*, *tcp_packets*, *udp_packets* y la cadena *allowed*, que es utilizada por la cadena *tcp_packets*. Los paquetes entrantes por **$INET_IFACE**, de tipo *ICMP*, serán redirigidos a la cadena *icmp_packets*. Los paquetes de tipo *TCP*, serán redirigidos a la cadena *tcp_packets* y los paquetes de tipo *UDP* a la cadena *udp_packets*. Todo ésto se explica en profundidad más adelante, en la sección *La cadena INPUT*. Crear una cadena es bastante sencillo, pues sólo consiste en una corta declaración similar a:

iptables [-N *cadena*]

En las secciones siguientes desarrollaremos ampliamente cada una de las cadenas que acabamos de crear. Veremos cómo son y qué reglas contendrán, así como qué conseguiremos con éllas.

La cadena "bad_tcp_packets"

La cadena *bad_tcp_packets* se ha concebido para contener reglas que inspeccionen los paquetes entrantes e identifiquen cabeceras malformadas, entre otros problemas. Así pues, decidimos incluir únicamente dos filtros: el primero bloquea todos los paquetes *TCP* entrantes que sean considerados como nuevos (**NEW**), pero no tengan establecido el bit *SYN*; el segundo bloquea los paquetes *SYN*/*ACK* que se consideren nuevos (*NEW*). Esta cadena se puede utilizar para comprobar posibles inconsistencias como las anteriores, o también como los chequeos de puertos *XMAS*, etc. También podemos añadir reglas que busquen el estado **INVALID**.

Si deseas entender a fondo el "NEW not SYN", deberías leerte la sección *Paquetes cuyo estado es NEW pero cuyo bit SYN no se ha establecido* del apéndice *Problemas y preguntas frecuentes*, acerca del estado "NEW and non-SYN" de los paquetes que atraviesan otras reglas. Bajo determinadas circunstancias a estos paquetes se les puede permitir atravesar el cortafuegos, aunque en el 99% de los casos no querremos que pasen, por lo que registramos su llegada en los registros (logs) y los desechamos (DROP).

El motivo para rechazar(*REJECT*) los paquetes SYN/ACK que se consideran nuevos es muy simple. Se describe en profundidad en la sección *Paquetes SYN/ACK y NEW* del apéndice *Problemas y preguntas frecuentes*. Básicamente, hacemos ésto como cortesía hacia otros hosts, ya que les prevendrá de ataques de predicción de secuencia numérica (sequence number prediction attack).

La cadena "allowed"

Si un paquete que llega a la interfaz **$INET_IFACE** es del tipo *TCP*, viajará por la cadena *tcp_packets*, y si la conexión es a través de un puerto en el que permitimos el tráfico, desearemos hacer algunos chequeos finales para ver si realmente le permitiremos el paso al paquete o no. Todo estos chequeos finales se realizan en la cadena *allowed* (permitido).

Empezamos comprobando si estamos ante un paquete *SYN*. Si es así, lo más probable es que sea el primer paquete de una nueva conexión, por lo que, obviamente, le permitimos el paso. Entonces comprobamos si el paquete procede de una conexión establecida (**ESTABLISHED**) o relacionada (**RELATED**). Si lo es, de nuevo permitimos el paso. Una conexión **ESTABLISHED** es aquélla que ha "visto" tráfico en ambas direcciones y puesto que nos hemos encontrado con un paquete *SYN*, la conexión debe estar en el estado **ESTABLISHED**, de acuerdo con la máquina de estados. La última regla de esta cadena desechará (**DROP**) todo lo demás. Ésto viene a englobar todo aquéllo que no ha "visto" tráfico en ambas direcciones, es decir, o no hemos respondido al paquete *SYN*, o están intentando empezar una conexión mediante un paquete "no-*SYN*". *No* existe ninguna utilidad práctica en comenzar una conexión sin un paquete *SYN*, a excepción de la gente que está haciendo escaneo de puertos. Hasta dónde yo sé, no hay actualmente ninguna implementación de *TCP*/*IP* que soporte la apertura de una conexión *TCP* con algo diferente a un paquete *SYN*, por lo que casi con total certeza se tratará de un escaneo de puertos y es recomendable desechar (**DROP**) estos intentos de conexión.

La cadena "tcp_packets"

La cadena *tcp_packets* especifica qué puertos se podrán alcanzar desde Internet a través del cortafuegos. Sin embargo, hay más chequeos para estos paquetes, ya que los enviamos a la cadena "allowed", descrita anteriormente.

La opción **-A tcp_packets** le indica a **iptables** en qué cadena debe añadir la nueva regla, la cuál se incluirá al final de la lista de reglas de esa cadena. La opción **-p TCP** especifica el primer requisito a cumplir por el paquete, es decir, debe ser un paquete *TCP*; de la misma forma, **-s 0/0** busca (compara) las direcciones de origen, empezando por la 0.0.0.0 con máscara de red 0.0.0.0, es decir *todas* las direcciones de origen. Aunque éste es el comportamiento por defecto, lo indico para que todo quede lo más claro posible (en este caso todos los puertos de origen cumplirán el requisito exigido por la regla). La opción **--dport 21** indica el puerto de destino 21, o sea que si el paquete está destinado al puerto 21, cumplirá el último criterio de selección de la regla. Si todos los criterios coinciden, el paquete será enviado a la cadena *allowed*. Si no coincide con alguna de las opciones, será devuelto a la cadena original (la que generó el salto a la cadena *tcp_packets*).

Tal como queda el ejemplo, permito el puerto *TCP* 21, o sea, el puerto de control *FTP*, empleado para controlar las conexiones *FTP*. Ésto unido a que más alante se permiten todas las conexiones relacionadas (**RELATED**), permitirá las conexiones ftp PASIVAS y ACTIVAS, pues en principio el módulo *ip_conntrack_ftp* ya se habrá cargado. Si no queremos permitir ningún tipo de conexión *FTP*, podemos descargar (o no cargar) el módulo *ip_conntrack_ftp* y borrar (o no añadir) la línea **$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed** del fichero rc.firewall.txt.

El puerto 22 es el *SSH* y emplear este puerto cuando dejas a cualquiera acceder a tu máquina mediante shell, es definitivamente mucho mejor que permitir el acceso mediante telnet por el puerto 23. Ten en cuenta que estás trabajando con un cortafuegos y siempre es una mala idea permitir a alguien más que tú mismo cualquier tipo de acceso a la máquina del cortafuegos. Los usuarios con acceso al cortafuegos siempre deben limitarse al mínimo imprescindible y ninguno más.

El puerto 80 es el de *HTTP*, o sea, tu servidor web. Borra esta línea si no quieres ejecutar un servidor web directamente en tu cortafuegos.

Y, por último, se permite el puerto 113, que es el *IDENTD* y puede ser necesario para algunos protocolos como IRC, para trabajar correctamente. Ten en cuenta que puede ser interesante utilizar las librerías **oidentd** si quieres efectuar traducciones *NAT* de varios hosts en tu red local. **oidentd** admite peticiones de retransmisión de *IDENTD* a las máquinas correctas de tu red local.

Si necesitas añadir más puertos abiertos en el script, parece bastante obvio cómo deberías hacerlo: simplemente copia y pega una de las líneas de la cadena *tcp_packets* y cambia el puerto que necesitas abrir.

La cadena "udp_packets"

Si se nos presenta un paquete *UDP* en la cadena *INPUT*, lo enviamos a la cadena *udp_packets*, dónde comenzaremos las comprobaciones empezando por el protocolo con **-p UDP** y después comprobaremos que proviene de cualquier dirección comenzando por la 0.0.0.0 y con máscara de red 0.0.0.0, o sea, que de nuevo cualquier dirección de origen es válida. Sin embargo sólo aceptamos puertos *UDP* específicos que queramos mantener abiertos para hosts de Internet. Puedes observar que no necesitamos abrir agujeros dependiendo del puerto de origen del host que envía el paquete, ya que de éllo se debería ocupar la máquina de estados. Sólo necesitamos abrir puertos en nuestro host si vamos a ejecutar un servidor en algún puerto *UDP*, como el servidor *DNS*, etc. Los paquetes que lleguen al cortafuegos y sean parte de una conexión ya establecida (por nuestra red local), serán aceptados automáticamente por las reglas *--state ESTABLISHED,RELATED* del principio de la cadena *INPUT*.

Tal como está el ejemplo, NO aceptamos (**ACCEPT**) paquetes *UDP* entrantes por el puerto 53, pues es el puerto usual cuando se efectúan búsquedas *DNS*. Observa que la regla está ahí, pero por defecto está comentada (#) y por éllo no se aceptan paquetes destinados al puerto

53. Si deseas que tu cortafuegos actúe como servidor *DNS*, elimina el símbolo de comentario (para permitir el tráfico por ése puerto).

Personalmente también permito el acceso por el puerto 123, que es el empleado por *NTP* (o *Network Time Protocol*). Este protocolo se emplea para ajustar la hora del reloj de tu ordenador de acuerdo a la hora de ciertos "servidores de horas" (time servers) que poseen relojes *muy* precisos. La mayoría de la gente probablemente no utiliza este protocolo y por éllo por defecto no está permitido el acceso a este puerto. Al igual que antes, sólo tienes que eliminar el símbolo de comentario para activar la regla.

En el ejemplo autorizamos el uso del puerto 2074, empleado por ciertas aplicaciones *multimedia* en tiempo real, como **speak freely**, con la que puedes hablar con otra gente en tiempo real (lógicamente a través de Internet), mediante unos altavoces y un micrófono o bien unos auriculares con micro. Si no deseas utilizar este servicio, simplemente comenta la línea (#).

El puerto 4000 es el del protocolo *ICQ*. Debería ser un protocolo conocidísimo, usado por la aplicación de Mirabilis denominada **ICQ**. Hay al menos 2 ó 3 clones diferentes de **ICQ** para Linux y es uno de los programas de chat más ampliamente difundido. Dudo que se necesiten más explicaciones.

Una vez en este punto, tienes disponibles dos reglas más para cuando experimentes un número excesivo de entradas en el registro debidas a distintas circunstancias. La primera regla bloqueará los paquetes de difusión (broadcast packets) hacia el rango de puertos entre el 135 y el 139. Este tipo de paquetes los emplean *NetBIOS* o *SMB* para gran parte de los usuarios de las plataformas de Microsoft. Con éllo bloquearemos todas las entradas de registro que pudiéramos tener a causa de Redes Microsoft desde el exterior de nuestra red local. La segunda regla también se ha creado para evitar los problemas de un excesivo número de entradas en el registro, pero en este caso se encarga de las peticiones *DHCP* que provienen del exterior. Ésto es especialmente cierto si tu red exterior se basa en una red de tipo "Ethernet no conmutado", donde los clientes reciben sus *direcciones IP* a través de *DHCP*. En estas circunstancias y debido únicamente a éllo, puedes verte barrido por una gran cantidad de registros.

Ten en cuenta que las dos últimas reglas son especialmente opcionales, puesto que habrá quien tenga interés en conservar este tipo de registros. Si estás teniendo problemas por un excesivo número de entradas legítimas en el registro, prueba a eliminar las originadas por estas clases de paquetes. También hay algunas reglas más del mismo tipo justo antes de las reglas de registro en la cadena *INPUT*.

La cadena "icmp_packets"

Aquí es donde decidimos qué tipos *ICMP* permitiremos. Si un paquete *ICMP* llega a través de la interfaz eth0 hasta la cadena *INPUT*, lo enviamos a la cadena ***icmp_packets*** tal como ya se ha explicado en los casos anteriores. Es entonces cuando comprobamos a qué tipo *ICMP* pertenece y si le permitimos el paso o no. Por el momento, sólo permito el paso a las *peticiones de eco ICMP* (*Echo requests*), al "*tiempo de vida del paquete nulo durante el tránsito*" (*TTL equals 0 during transit*) y al "*tiempo de vida nulo durante el reensamblado*" (*TTL equals 0 during reassembly*). La razón para no permitir el paso a ningún otro tipo ICMP como opción por defecto es que prácticamente todos éllos deben ser gestionados por las reglas para el estado "paquete relacinado" (RELATED).

Si un paquete ICMP es enviado como respuesta a un paquete existente o a un flujo de paquetes, se considera como relacionado (RELATED) con el flujo original. Para más información sobre los estados, léete el capítulo *La máquina de estados*.

A continuación explicaré las razones por las que permito estos tipo de paquetes *ICMP*: las peticiones de eco se utilizan para pedir una respuesta de eco de la máquina destino, lo cual sirve principalmente para hacer "pings" a otros hosts y ver si están disponibles en alguna red. Sin esta regla los demás hosts no serán capaces de hacer ping sobre nuestra máquina y no podrán saber si estamos disponibles en alguna conexión de red. Hay que destacar sin embargo que hay quien elimina esta regla, simplemente porque no desean ser "vistos" (ser detectables) en Internet. Borrando esta regla consigues que cualquier ping que llegue a tu cortafuegos sea inútil, ya que el cortafuegos no le contestará.

En cuanto al "tiempo excedido" (*TTL equals 0 during transit* o bien *TTL equals 0 during reassembly*), nos convendrá que los paquetes ICMP de este tipo tengan permiso para atravesar el cortafuegos siempre que deseemos conocer la ruta hasta un host (o sea, efectuar un trace-route), o también en los casos en los que un paquete que hayamos enviado a alcanzado el estado "Time To Live=0" y deseemos recibir notificación del evento.

Por ejemplo, cuando efectúas un trace-route a alguien, empiezas enviando un paquete con *TTL* = 1, lo cual implica que en el primer salto entre hosts (en el primer "hop") el TTL se reduce en una unidad y se iguala a 0, de forma que se envía de vuelta un paquete ICMP de tipo Time Exceeded desde el primer punto de la ruta de acceso que lleva al host que tratamos de localizar. Entonces se envía otro paquete, pero esta vez con *TTL* = 2 y en el segundo "hop" la pasarela (gateway) devolverá un ICMP Time Exceeded, con lo que ya se conocen los dos primeros puntos de la ruta. Continuando de manera análoga se llegará hasta el host que deseamos alcanzar, con lo que conoceremos la ruta completa y podremos saber qué host intermedio tiene problemas.

Para ver un listado completo de los tipos *ICMP*, léete el apéndice *Tipos ICMP*. Para más información acerca de los tipos *ICMP* y su utilización, te recomiendo leer los siguietes documentos e informes:

- *Protocolo de Mensajes de Control de Internet* por Ralph Walden.
- *RFC 792 - Protocolo de Mensajes de Control de Internet* por J. Postel.

Note: Debo advertir que desde tu punto de vista puedo estar equivocado al bloquear algunos de los tipos *ICMP*, pero en mi caso todo funciona perfectamente después de bloquear todos los tipos *ICMP* a los que no permito el paso.

La cadena INPUT

Tal como la he definido en el script, la cadena *INPUT* utiliza principalmente otras cadenas para hacer su trabajo. De esta forma no sobrecargamos demasiado a iptables y funcionará mucho mejor en máquinas lentas, que de otra manera desecharían paquetes en situaciones de mucha carga para el sistema. Ésto lo conseguimos al buscar detalles concretos que deberían coincidir en los paquetes del mismo tipo, enviándolos entonces a cadenas especificadas por el usuario. Actuando así podremos subdividir nuestro conjunto de reglas de forma que cada paquete deba atravesar el menor número de reglas y con éllo el cortafuegos sufrirá una menor carga debida al filtrado de paquetes.

Para empezar realizamos ciertas comprobaciones para encontrar paquetes malformados o incorrectos: lo conseguimos enviando todos los *paquetes TCP* a la cadena *bad_tcp_packets*, la cual incluye unas pocas reglas para comprobar malformaciones o anomalías que no deseamos aceptar. Para una explicación completa, lee la sección *La cadena "bad_tcp_packets"* de este capítulo.

Es ahora cuando empezamos a buscar tráfico de las redes que normalmente son de confianza, como el procedente del adaptador de red local (lan_iface), o el tráfico entrante y saliente de nuestra interfaz de *bucle local* (loopback interface, lo_iface), incluyendo también todas nuestras *direcciones IP* actualmente asignadas (y ésto significa todas, incluyendo nuestra *dirección IP de Internet*). Basándonos en lo anterior hemos decidido añadir la regla que permite la actividad de la red local (*LAN*) al principio, ya que en nuestro caso la red local genera más tráfico que la conexión de Internet. Éllo permite una menor sobrecarga ya que se debe comparar cada paquete en cada regla, por lo que es una buena idea averiguar qué tipo de tráfico atraviesa con más frecuencia el cortafuegos. Cuando sepamos la carga relativa de cada tipo, podremos reordenar las reglas para que sean más eficientes, produciendo así una menor sobrecarga al cortafuegos y una menor congestión de tu red.

Antes de empezar con las "reglas auténticas" que deciden qué permitimos o no desde la interfaz de Internet, tenemos una regla relacionada configurada para reducir la carga del sistema. Es una regla de estado que permite el paso a todos los paquetes que sean parte de un flujo *ESTABLISHED* (establecido) o bien *RELATED* (relacionado) con la *dirección IP* de Internet. Existe una regla equivalente en la cadena "allowed", lo cual es redundante ya que antes de llegar a esa cadena, en este punto del script los paquetes ya son filtrados. Sin embargo la regla **--state ESTABLISHED,RELATED** de la cadena "allowed" se ha mantenido por varias razones, entre las que se encuentra el deseo de algunos de poder cortar y pegar esta función.

Tras éllo y todavía en la cadena *INPUT*, filtramos todos los paquetes *TCP* que lleguen a la interfaz **$INET_IFACE**, enviándolos a la cadena ***tcp_packets***. Después se efectúa la misma operación con los paquetes *UDP*, enviándolos a la cadena *udp_packets*. Por último los paquetes *ICMP* son enviados a la cadena *icmp_packets*.

Normalmente un cortafuegos recibirá un mayor número de paquetes *TCP*, en menor cantidad los paquetes *UDP* y por último los *ICMP*. Sin embargo ten en cuenta que ésto es lo normal pero no tiene por qué ser tu caso. Igual que hicimos con las reglas específicas del tipo de tráfico de la red, es conveniente averiguar qué paquetes causan un mayor tráfico para reordenar convenientemente estas tres reglas. Es preciso tener en cuenta que una máquina equivalente a un *Pentium III*, con una tarjeta *Ethernet* de 100Mbit trabajando a plena potencia, puede claudicar ante un simple conjunto de 100 reglas mal escrito. Es importante tenerlo en cuenta a la hora de escribir el conjunto de reglas para tu red local.

Ahora nos encontramos con una regla extra, que por defecto es opcional (#) y puede utilizarse para evitar un excesivo número de registros en el caso de que tengas una red de Microsoft en el exterior del cortafuegos Linux: los clientes con sistema operativo Microsoft tienen la mala costumbre de enviar toneladas de paquetes de multidifusión (multicast) al rango 224.0.0.0/8, por lo que tenemos la oportunidad de bloquear esos paquetes de forma que no inunden los registros. También hay dos reglas más que hacen algo parecido en la cadena *udp_packets*, descrita en *La cadena "udp_packets"*.

Antes de llegar a la política por defecto de la cadena *INPUT*, registramos el tráfico que no se ha filtrado aún, de manera que seamos capaces de encontrar posibles problemas y/o fallos: puede que sean paquetes que simplemente no queremos dejar pasar, puede ser alguien intentando hacernos algo malo, o puede ser un problema de nuestro cortafuegos que no deja pasar algún tráfico que debería estar permitido. En cualquier caso queremos saberlo, de forma que podamos solucionar el problema. De todas formas no registramos más de 3 paquetes por minuto, ya que no queremos inundar los registros con basura y con éllo llenar por completo la partición del registro; además, incluímos un prefijo a todas las entradas para saber de dónde provenían.

Todo lo que no haya sido filtrado será desechado (**DROP**) por la política por defecto de la cadena *INPUT*, que ha sido establecida casi al principio de este script, en la sección *Estableciendo las políticas por defecto*.

La cadena FORWARD

Esta cadena tiene muy pocas reglas en el escenario que establecimos al principio. Tenemos una regla que envía todos los paquetes a la cadena *bad_tcp_packets*, que ya ha sido descrita anteriormente y se ha definido de manera que pueda ser empleada por varias cadenas, independientemente del tipo de paquete que la atraviese.

Después de este primer chequeo para buscar paquetes *TCP* incorrectos, tenemos las reglas principales de la cadena *FORWARD*. La primera regla permite el tráfico sin restricciones desde nuestra interfaz de red local (**$LAN_IFACE**) hacia cualquier otra interfaz. En otras palabras, permite todo el tráfico desde nuestra *LAN* hacia Internet. La segunda regla permite el tráfico establecido y relacionado (**ESTABLISHED**, **RELATED**) de vuelta a través del cortafuegos. Es decir, permite que los paquetes pertenecientes a conexiones iniciadas por nuestra red interna fluyan libremente hacia nuestra red. Estas reglas son necesarias para que nuestra red local sea capaz de acceder a Internet, ya que la política por defecto de la cadena *FORWARD* se ha establecido previamente como **DROP** (desechar). Ésta es una forma de actuar bastante inteligente, ya que permitirá la conexión a Internet a los hosts de nuestra red local, pero al mismo tiempo bloqueará a los hosts de Internet que estén intentando conectar con nuestros hosts internos.

Por último, igual que hicimos en la cadena *INPUT*, tenemos una regla que registrará los paquetes que de alguna manera no tengan permiso para pasar a través de la cadena *FORWARD*. Con éllo probablemente veremos algun caso de paquete incorrectamente formado, entre otros problemas. Una causa posible pueden ser los intentos de ataque de hackers, aunque otras veces simplemente serán paquetes malformados. Esta regla es prácticamente la misma que la de la cadena INPUT, salvo por el prefijo de registro, que en este caso es **"IPT FORWARD packet died: "**. Los prefijos se emplean usualmente para separar entradas de registro y pueden emplearse para distinguir registros y saber qué cadenas los produjeron, así como algunas opciones de las cabeceras.

La cadena OUTPUT

Puesto que esta máquina sólo la utilizo yo y es al mismo tiempo cortafuegos y estación de trabajo, permito prácticamente todo lo que proviene de las direcciones de origen siguientes: **$LOCALHOST_IP**, **$LAN_IP** o **$STATIC_IP**. Cualquier otra dirección debería ser un intento de spoofing (camuflaje de direcciones), aunque dudo que nadie que conozca pudiera hacer ésto en mi máquina. Por último registramos todo aquello que resulte desechado: si se desecha, definitivamente querremos saber por qué, de manera que podamos atajar el problema.

Puede que sea un asqueroso error, o que sea un paquete extraño que ha sido "camuflado" (spoofed). Tras registrarlos, desechamos (**DROP**) los paquetes como política por defecto.

La cadena PREROUTING de la tabla nat

La cadena *PREROUTING* es básicamente lo que indica su nombre: realiza la traducción de direcciones de red en los paquetes, antes de que lleguen a la decisión de asignación de ruta que les enviará a las cadenas *INPUT* o *FORWARD* de la tabla filter. La única razón por la que nos referimos a esta cadena en el script es que nos sentimos obligados a repetir que NO DEBES efectuar ningún tipo de filtrado en élla. La cadena *PREROUTING* sólo es atravesada por el primer paquete de un flujo, lo cual significa que todos los demás paquetes del flujo serán totalmente ignorados por esta cadena. Tal como está definido este script, no utilizamos en absoluto la cadena *PREROUTING*, si bien es el lugar donde deberíamos trabajar si quisiéramos hacer alguna traducción *DNAT* en paquetes específicos: por ejemplo si quisieras albergar tu *servidor web* dentro de tu red local. Para más información acerca de la cadena *PREROUTING*, léete el capítulo *Atravesando tablas y cadenas*.

Caution

La cadena *PREROUTING* no debe usarse para ningún tipo de filtrado puesto que, entre otras causas, esta cadena sólo es atravesada por el primer paquete de un flujo. La cadena *PREROUTING* debe usarse únicamente para efectuar traducciones de direcciones de red, a no ser que sepas realmente lo que estás haciendo.

Activando SNAT y la cadena POSTROUTING

Así pues, nuestro último cometido es tener en funcionamiento la traducción de direcciones (*Network Address Translation*), ¿no? Por lo menos lo es para mí. Para empezar añadimos una regla a la tabla *nat*, en la cadena *POSTROUTING*, que "traducirá" todos los paquetes que salgan de nuestra interfaz conectada a Internet. En mi caso se trata de la interfaz *eth0*. Sin embargo, existen variables específicas añadidas a todos los scripts de ejemplo que pueden usarse para configurar automáticamente estos valores. La opción **-t** le indica a **iptables** en qué tabla insertar la regla, que en nuestro caso es la tabla *nat*. El comando -A indica que queremos agregar (Append) una nueva regla a una cadena existente llamada *POSTROUTING*, mientras que **-o $INET_IFACE** indica que buscará todos los paquetes salientes a través de la interfaz **INET_IFACE** (o *eth0*, como valor por defecto en este script). Por último definimos el objetivo para que efectúe un **SNAT** a los paquetes. Es decir, todos los paquetes que concuerden con esta regla verán traducida su dirección de origen, para que parezca que provienen de tu interfaz de Internet. Ten en cuenta que debes definir la dirección IP que se les asignará a los paquetes salientes en el objetivo SNAT mediante la opción **--to-source**.

En este script hemos decidido utilizar el objetivo **SNAT** en lugar de **MASQUERADE** (enmascarar) debido a dos razones: la primera es que este script se supone que trabaja en un cortafuegos con *dirección IP* estática; la segunda razón es consecuencia de la primera, ya que resulta más rápido y eficiente utilizar el objetivo *SNAT*, siempre que sea posible. Por supuesto, también se ha usado para mostrar cómo podría funcionar y cómo debería utilizarse en un caso real. Si no tienes *direccion IP* estática, definitivamente deberías pensar en cambiar al objetivo **MASQUERADE**, que ofrece una forma simple y sencilla de traducir las direcciones y que además captará automáticamente la *dirección IP* a utilizar. Ésto consume un poco más de potencia del sistema, pero valdrá la pena si utilizas *DHCP*, por ejemplo. Si quieres saber más acerca de cómo funciona el objetivo **MASQUERADE**, léete el script *rc.DHCP.firewall.txt*.

Chapter 8. Scripts de ejemplo

El objetivo de este capítulo es dar una explicación sucinta de cada script disponible con este tutorial, así como un vistazo general de los scripts y los servicios que ofrecen. Los scripts no son en modo alguno perfectos y es posible que no se ajusten exactamente a tus necesidades. En otras palabras: eres tú el que debes ajustar los scripts a tus necesidades. El resto del tutorial debería ayudarte a efectuar los cambios necesarios. La primera sección del tutorial versa acerca de la estructura establecida en cada script, con lo cual debería ser más fácil entender cada apartado de los scripts.

Estructura del script rc.firewall.txt

Todos los scripts de este tutorial se han escrito basados en una estructura específica. La razón es que deben estar bastante parejos y ser fácilmente diferenciables entre ellos. Dicha estructura debería estar suficientemente documentada en este breve capítulo, dónde se intenta proporcionar una pequeña explicación de por qué se han escrito todos los scripts de la manera en que se ha hecho y por qué se ha escogido esa estructura.

> Note
>
> A pesar de que ésta es la estructura elegida, ten presente que quizá no es la estructura que más te conviene para tus scripts. Es simplemente la estructura que yo he escogido, ya que se ajusta a la necesidad de que sea fácil de leer y de seguir, de acuerdo con mi lógica.

La estructura

Esta es la estructura que todos los scripts de este tutorial deberían seguir. Si difieren en algo será probablemente un error por mi parte, a no ser que se diga específicamente el por qué se ha abandonado la estructura.

1. *Configuración* - En primer lugar nos encontramos con las opciones de configuración que todos los scripts deben usar. Las opciones de configuración deben ser casi siempre lo primero en cualquier script en línea de comandos (shell-script).
   - 1.1. *Internet* - Este es el apartado de configuración referente a la conexión a Internet. Puedes saltarte este apartado si no tienes ninguna conexión a Internet. Ten en cuenta que podría haber más subsecciones que las listadas aquí, pero sólo aquéllas referentes a nuestra conexión a Internet.
     - 1.1.1. *DHCP* - Si se requieren opciones específicas para DHCP en el script, las configuraremos aquí.
     - 1.1.2. *PPPoE* - Si existe la posibilidad de que el usuario quiera usar este script en concreto y por cualquier motivo resulta que está usando una conexión PPPoE, añadiremos las opciones específicas aquí.
   - 1.2. *LAN* - Si hay una LAN habilitada detrás del cortafuegos (firewall), añadiremos las opciones pertinentes en esta sección. Ésto será lo más probable, de aquí que esta sección esté casi siempre habilitada.
   - 1.3. *DMZ* - Si hay alguna razón para éllo, añadiremos las opciones de configuración de una DMZ ("Zona Desmilitarizada") en este punto. La mayor parte de los scripts carecen de esta sección, principalmente porque ninguna red casera normal o pequeña red de empresa tendrá DMZs.
   - 1.4. *Localhost* - Estas opciones pertenecen a nuestro host local. Es muy improbable que estos valores cambien, pero los hemos puesto como variables de todas formas. En principio no debería haber motivos para hacer cambios en estas variables.

1.5. *iptables* - Esta sección contiene la configuración específica de iptables. En la mayoría de scripts y situaciones, sólo se requiere una variable que indique dónde está ubicado el archivo binario (ejecutable) "iptables".

1.6. *Otras* - Si existe cualquier otra opción o variable concreta debe ser incluida, antes que en cualquier otro sitio, en su subsección (si pertenecen a la conexión a Internet, deben colocarse en dicha subsección, etc). Si no van en ninguna, entonces deben colocarse directamente en algún lugar entre las opciones de configuración.

2. *Carga de módulos* - Esta sección del script debe albergar una lista de módulos. La primera parte debe contener los módulos requeridos (necesarios), mientras que la segunda contendrá los no requeridos.

> Ten en cuenta que algunos modulos que permiten aumentar la seguridad o añadir determinados servicios o posibilidades, se pueden haber añadido aunque no estén estrictamente requeridos. Cuando así suceda, normalmente se mencionará expresamente en los scripts de ejemplo.

2.1. *Modulos requeridos* - Esta sección debería contener los módulos requeridos y, posiblemente, los módulos que mejoren la seguridad o añadan servicios especiales al administrador o a los clientes.

2.2. *Modulos No requeridos* - Esta sección contendrá los módulos que no son requeridos para operaciones normales. Todos estos modulos aparecen por defecto como si fueran comentarios, por lo que si quieres añadir el servicio que proporcionan, deberás eliminar el símbolo de comentario (#).

3. *Configuración proc* - Esta sección tratará cualquier configuración especial que necesite el "sistema de ficheros proc" (éste es un sistema virtual creado bajo demanda que permite interactuar con el núcleo y los procesos que se ejecutan en él). Si algunas opciones son requeridas, serán listadas como tales, pero si no es así se escribirán por defecto como comentarios y se listarán bajo las configuraciones no-requeridas de proc. La mayoría de las configuraciones proc útiles serán listadas aquí, pero ni mucho menos se incluirán todas.

   3.1. *Configuración proc requerida* - Esta sección debe contener todas las configuraciones proc requeridas por el script en cuestión. También podría contener posiblemente configuraciones que aumenten la seguridad o que pueden añadir servicios o posibilidades especiales para el administrador o los clientes.

   3.2. *Configuración proc no-requerida* - Esta sección debe contener las configuraciones proc no-requeridas que podrían resultar utiles. Todas ellas se escriben como si fueran comentarios porque no son necesarias para que el script trabaje. Esta lista está lejos de contener todas las configuraciones de proc.

4. *Escritura de las reglas* - Después de todo lo anterior, el script ya debe estar listo para que se inserte el conjunto de reglas. He decidido clasificar todas las reglas tras la tabla y los nombres de las cadenas. Todas las cadenas específicas de usuario se crean antes de que hagamos nada en las cadenas por defecto del sistema. También he decidido organizar las cadenas y sus reglas específicas en el mismo orden en que son mostradas con el comando **iptables -L**.

   4.1. *Tabla Filter* - Antes que nada vamos directos a la tabla filter y a su contenido. En primer lugar debemos configurar las políticas de la tabla.

      4.1.1. *Configuración de las políticas* - empezamos por configurar todas las políticas por defecto para las cadenas del sistema. Normalmente pondré la política DROP por defecto en las cadenas de la tabla filter y específicamente usaré la política ACCEPT para los servicios y flujos que yo quiera permitir. De esta manera podremos evitar que la gente use puertos que no queremos que estén disponibles.

4.1.2. *Creación de las cadenas específicas del usuario* - en este punto creamos todas las cadenas propias que queremos usar más tarde en esta tabla. No podremos referirnos a estas cadenas en las del sistema si no están ya creadas, por lo que debemos llegar a éllas lo antes posible.

4.1.3. *Crear los contenidos de las cadenas creadas por el usuario* - después de crear las cadenas específicas de usuario, podemos escribir todas las reglas que llevarán esas cadenas. La única razón que tengo para escribir estos datos en este punto es por tener las cadenas propias y sus reglas lo más juntas posible. Sin embargo puedes ponerlas más adelante en el script, es decisión tuya.

4.1.4. *Cadena INPUT* - llegados aquí ya no nos quedan muchas más cosas que hacer en la tabla filter, así que nos adentramos en la cadena INPUT. Ahora es cuando debemos añadir todas las reglas a la cadena INPUT.

> Note
>
> Es en este momento cuando empezamos a seguir la salida del comando **iptables -L**, como podrás combrobar. No hay razón para que continúes con esta estructura, pero no obstante, trata de evitar mezclar datos de diferentes tablas y cadenas, ya que puede hacerse muy duro leer y comprender el conjunto de reglas, así como encontrar la solución a posibles problemas.

4.1.5. *Cadena FORWARD* - vamos a añadir las reglas a la cadena FORWARD. Ningún comentario acerca de esta decisión.

4.1.6. *Cadena OUTPUT* - lo último que queda por hacer en la tabla filter es añadir las reglas que van en la cadena OUTPUT. En principio no debería haber mucho por hacer en esta cadena.

4.2. *Tabla nat* - después de la tabla anterior, centramos nuestra atención en la tabla nat. Abordamos esta tabla después de la tabla filter debido a varias razones referentes a estos scripts: para empezar no queremos activar todo el mecanismo de reenvío (forwarding) y la función NAT en una etapa demasiado temprana, pues posiblemente podría dejar pasar los paquetes a través del cortafuegos en un momento erróneo (es decir, cuando el NAT se activa, pero ninguna regla de filtrado ha arrancado todavía). Además, veo la tabla nat como una especie de capa que está por encima de la tabla filter y que la rodea de alguna manera. Así pues la tabla filter debería ser el núcleo, mientras que la tabla nat actuaría como una capa que la recubre y por último la tabla mangle rodearía a la tabla nat como una segunda capa. Ésto puede ser erróneo desde algún punto de vista, pero no está demasiado lejos de la realidad.

4.2.1. *Configurar las políticas* - antes que nada establecemos las políticas por defecto en la tabla nat. Normalmente estoy satisfecho con la política por defecto que hemos establecido al principio, es decir, ACCEPT. Esta tabla no debe ser usada para filtrar y no debemos bloquear los paquetes aquí, ya que pueden ocurrir algunas cosas realmente feas en dichos casos debido a nuestras presunciones. Dejo estas cadenas configuradas como ACCEPT puesto que no hay razón para no hacerlo.

4.2.2. *Creación de las cadenas específicas del usuario* - ahora crearemos cualquier cadena específica del usuario que queramos añadir a la tabla nat. Normalmente no tengo ninguna, pero he añadido esta sección de todas formas, sólo por si acaso. Ten en cuenta que las cadenas específicas del usuario deben ser creadas antes de que puedan ser usadas en las cadenas del sistema.

4.2.3. *Creación del contenido de las cadenas específicas del usuario* - ahora es el momento de añadir las reglas a las cadenas que irán en la tabla nat. Podemos decir lo mismo que con las cadenas específicas del usuario en la tabla filter. Añadimos estas reglas aquí, ya que no veo razón alguna para no hacerlo.

4.2.4. *Cadena PREROUTING* - La cadena PREROUTING es la usada para hacer DNAT a los paquetes en caso de que tengamos la necesidad de hacerlo. En la mayoría de los scripts esta característica no se usa, o como mínimo está en forma de comentario; ésto es así para no abrir enormes agujeros hacia nuestra red local sin tener conocimiento de éllos. En algunos scripts tenemos esta cadena habilitada por defecto, ya que el único propósito de dichos scripts es proveer de tales servicios.

4.2.5. *Cadena POSTROUTING* - la cadena POSTROUTING se usa bastante a menudo en los scripts que he escrito, puesto que la mayor parte de éllos se basan en el hecho de que tienes una o más redes locales que quieres defender de Internet mediante un cortafuegos. Principalmente intentaremos usar el objetivo SNAT pero en ciertos casos estaremos forzados a usar el objetivo MASQUERADE en su lugar.

4.2.6. *Cadena OUTPUT* - la cadena OUTPUT es escasamente usada en algún script. Aunque lo parezca, no es que no funcione, pero no he sido capaz de encontrar un buena razón para usar esta cadena. Si alguien encuentra una razón para usarla, que me envíe unas líneas y añadiré esta cadena en el tutorial.

4.3. *Tabla mangle* - la última tabla en la que hacer algo es la tabla mangle. Normalmente no usaré esta tabla, y no debería usarla nadie, a no ser que se tengan necesidades específicas como por ejemplo enmascarar todos los paquetes para que usen el mismo TTL o para cambiar el campo TOS, etc. En otras palabras, me decanto por dejar estas partes de los scripts más o menos en blanco, con unas pocas excepciones dónde añado unos cuantos ejemplos acerca de para qué se puede usar.

4.3.1. *Configurar las políticas* - configurar las políticas por defecto de la cadena. Ocurre prácticamente lo mismo que en la tabla nat: esta tabla no está hecha para filtrar y de ahí que debas evitar hacerlo. Yo no he configurado ninguna política en la tabla mangle en ninguno de los scripts y te recomiendo que tú tampoco lo hagas.

4.3.2. *Crear cadenas específicas del usuario* - crear todas las cadenas específicas del usuario. Ya que apenas uso la tabla mangle en los scripts, no he creado ninguna cadena aquí, puesto que es claramente inservible si no hay datos en los que emplearlas. Sin embargo, esta sección fue añadida por si hay alguien que la necesite en un futuro.

4.3.3. *Crear el contenido de las cadenas específicas del usuario* - si tienes alguna cadena dentro de esta tabla debes añadirle las reglas en este momento.

4.3.4. *PREROUTING* - hasta este momento apenas hay información en ningún script de este tutorial que contenga reglas aquí.

4.3.5. *Cadena INPUT* - hasta este momento apenas hay información en ningún script de este tutorial que contenga reglas aquí.

4.3.6. *FORWARD chain* - hasta este momento apenas hay información en ningún script de este tutorial que contenga reglas aquí.

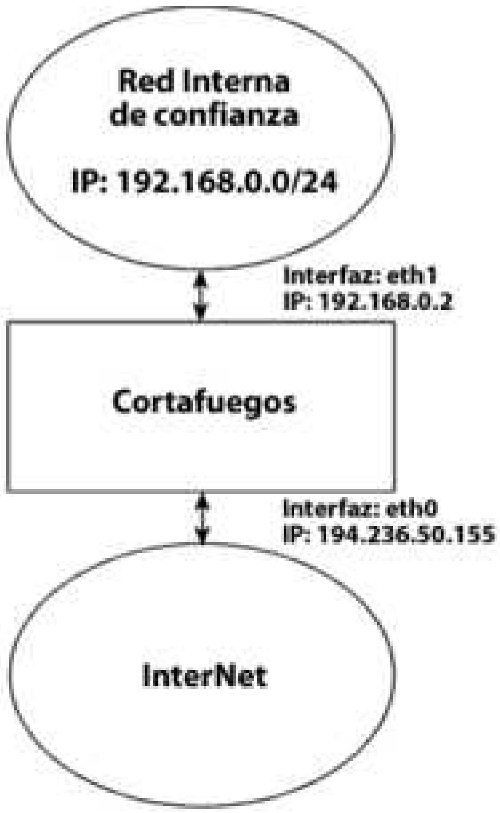
4.3.7. *OUTPUT chain* - hasta este momento apenas hay información en ningún script de este tutorial que contenga reglas aquí.

4.3.8. *POSTROUTING chain* - hasta este momento apenas hay información en ningún script de este tutorial que contenga reglas aquí.

Esperemos que todo ésto ayude a entender mejor cómo está estructurado cada script y por qué está estructurado de dicha manera.

Ten en cuenta que estas descripciones son extremadamente breves, y principalmente deben ser vistas exclusivamente como una breve explicación de cómo y por qué los scripts han sido divididos como se ha hecho. Nadie dice que ésta sea la única y mejor manera de hacerlo.

rc.firewall.txt

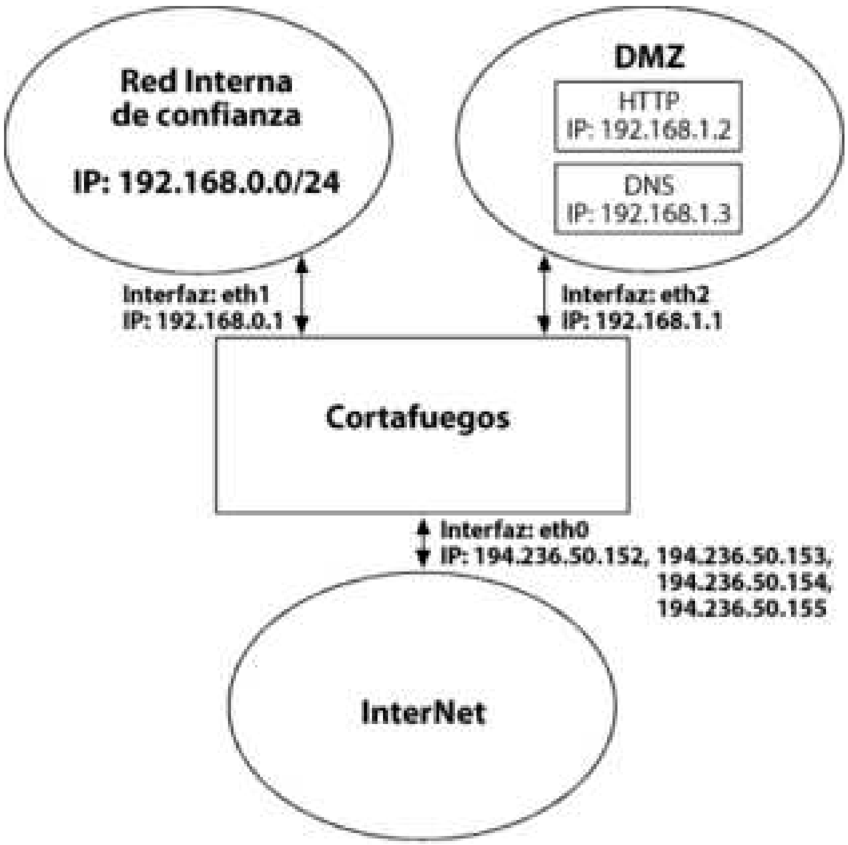


El script rc.firewall.txt (http://iptables-tutorial.frozentux.net/scripts/rc.firewall.txt) es el "centro neurálgico" en el que se basan el resto de scripts. El capítulo *El archivo rc.firewall* debería explicar detenidamente cada detalle del script. Principalmente se ha escrito para una red local casera dual, como por ejemplo cuando tienes una red local y una conexión a Internet. Este script también asume que tienes una dirección IP estática para la conexión a Internet, por lo que no usa ni *DHCP*, ni *PPP*, ni *SLIP*, ni ningún otro protocolo que te asigne automáticamente una dirección IP. Si lo que buscas es un script que funcione con esas configuraciones, mejor échale un vistazo al script *rc.DHCP.firewall.txt*.

El script rc.firewall.txt necesita que las siguientes opciones sea compiladas estáticamente en el núcleo, o bien compiladas como módulos. Sin alguna de éllas quedará penalizado en mayor o menor medida, puesto que habrán funcionalidades requeridas, necesarias en el script, que quedarán inutilizables. Asímismo, conforme vayas cambiando el script para adaptártelo, posiblemente necesitarás más opciones compiladas en el núcleo (dependiendo siempre de lo que quieras utilizar).

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_LOG

rc.DMZ.firewall.txt



El script rc.DMZ.firewall.txt (http://iptables-tutorial.frozentux.net/scripts/rc.DMZ.firewall.txt) va destinado a aquellos que tienen una *Red Interna de Confianza* (*Trusted Internal Network*), una *Zona "Desmilitarizada"* (*De-Militarized Zone*) y una *Conexión a Internet* (*Internet Connection*). La *Zona Desmilitarizada* en este caso es "*NAT*-eada" 1-a-1 y requiere que hagas algo de solapamiento (aliasing) en tu cortafuegos, ésto es, debes conseguir que la máquina reconozca paquetes de más de una IP. Hay varias formas de conseguirlo: una es establecer la traducción

NAT de 1 a 1; si dispones de una subred completa, otra forma sería crear una subred, dándole al cortafuegos una IP a la vez interna y externa. A partir de aquí, puedes establecer las IPs en las máquinas de la zona *DMZ* tal como desees. Sin embargo, ten encuenta que este sistema te "robará" dos IPs: una para la dirección de difusión (broadcast) y otra más para la dirección de red. Al final es decisión tuya qué implementar, pero este tutorial sólo te dará las herramientas para conseguir configurar la parte del cortafuegos y del *NAT*, siendo el resto tarea tuya, puesto que no te indicará qué debes hacer a partir de ahí (ésto se sale fuera de la intención del tutorial).

El script rc.DMZ.firewall.txt requiere las siguientes opciones compiladas en el núcleo, bien de forma estática, bien en forma de módulos. Sin estas opciones al menos disponibles en el núcleo, no serás capaz de utilizar la funcionalidad del script. O sea, que obtendrás un montón de errores respecto a módulos, objetivos/saltos o comparaciones no encontrados. Si tienes en mente efectuar un control del tráfico o algo similar, debes asegurarte que también tienes todas las opciones necesarias compiladas en el núcleo.

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_LOG

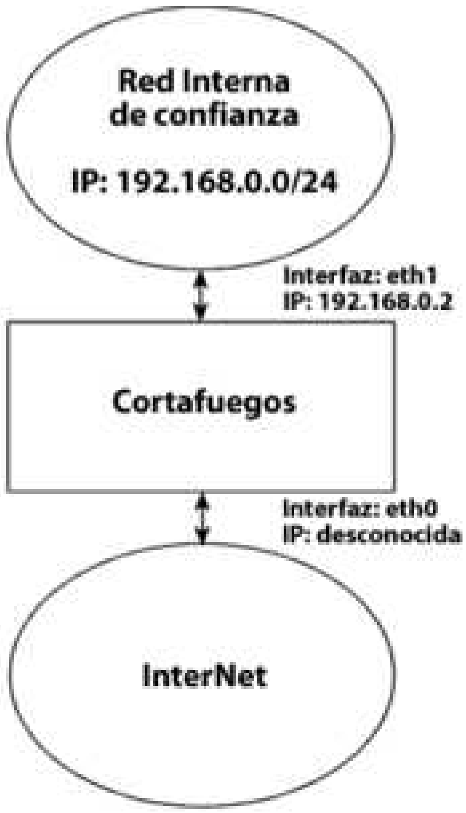
Como puedes ver en la imagen, necesitas tener dos redes internas con este script. Una utiliza el rango IP 192.168.0.0/24 y se trata de una *Red Interna de Confianza*. La otra utiliza el rango IP 192.168.1.0/24, tratándose de la *Zona Desmilitarizada* a la que le efectuaremos la traducción *NAT* 1-a-1. Por ejemplo, si alguien desde Internet envía un paquete a nuestra DNS_IP, utilizaremos *DNAT* para enviar el paquete a nuestro *DNS* en la red *DMZ*. Cuando el *DNS* comprueba el paquete, lo envía a la dirección IP del *DNS* de la red interna y no a la IP del *DNS* externo. Si el paquete no hubiera sido traducido, el *DNS* nunca hubiera respondido al paquete. Veamos a continuación un pequeño ejemplo del aspecto del código *DNAT*:

**$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $DNS_IP **
--dport 53 -j DNAT --to-destination $DMZ_DNS_IP

Ésto viene a significar los siguiente: para empezar, la traducción *DNAT* sólo se puede efectuar en la cadena *PREROUTING* de la tabla *nat*. Después, buscamos el protocolo *TCP* en nuestra interfaz $INET_IFACE con la IP de destino que coincida con nuestra $DNS_IP y que esté dirigido al puerto 53, que es el puerto *TCP* para las transferencias de zona entre servidores de nombres. Si tenemos un paquete con estas características, hacemos que efectúe un salto *DNAT*. En resumen, efectuamos un *DNAT*. Después de ésto especificamos dónde queremos que vaya el paquete mediante la opción **--to-destination** y le damos el valor $DMZ_DNS_IP (la IP del *DNS* de nuestra red *DMZ*). Así es como trabaja básicamente el *DNAT*. Cuando la respuesta al paquete "*DNAT*eado" se envía a través del cortafuegos, automáticamente se "des-DNATea".

Llegados a este punto, deberías comprender suficientemente bien cómo funciona todo, de manera que no te suponga ningún problema importante comprender bastante bien este script. Si hay algo que no entiendes y no ha sido tratado en el resto del tutorial, envíame un correo, pues probablemente sea un fallo por mi parte.

rc.DHCP.firewall.txt



El script rc.DHCP.firewall.txt (http://iptables-tutorial.frozentux.net/scripts/rc.DHCP.firewall.txt) es casi idéntico al original (*rc.firewall.txt*). Sin embargo, en este caso no se usa la variable **STATIC_IP**, lo cual es la mayor diferencia con el script original (rc.firewall.txt). La razón es que no funcionaría en una conexión con IP dinámica. Los cambios necesarios en el script original son mínimos, pero ha habido gente que me ha preguntado cómo solventar el problema de las IPs dinámicas, así que este script será una buena solución para todos los que tengan las mismas dudas. Este script es válido para todos aquellos que utilicen conexiones *DHCP*, *PPP* y *SLIP* para acceder a Internet.

El script rc.DHCP.firewall.txt requiere las siguientes opciones compiladas estáticamente en el núcleo, o al menos como módulos, para funcionar correctamente.

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_MASQUERADE
- CONFIG_IP_NF_TARGET_LOG

Los cambios principales efectuados al script consisten en la supresión de la variable STATIC_IP, como ya he dicho, y de todas las referencias a élla. En lugar de utilizarla, el script ejecuta la mayor parte del filtrado a través de la variable INET_IFACE. Es decir, se ha cambiado **-d $STATIC_IP** por **-i $INET_IFACE**. Básicamente éstos son los únicos cambios necesarios.

De todas formas, hay algunas cosas más en las que debemos pensar. Ya no podemos filtrar en la cadena *INPUT* en función de, por ejemplo, **--in-interface $LAN_IFACE --dst $INET_IP**. Ésto nos obliga a filtrar únicamente basándonos en interfases en aquellos casos dónde las máquinas internas deben acceder a la IP direccionable de Internet. Un buen ejemplo es cuando ejecutamos un servidor *HTTP* en nuestro cortafuegos: si nos dirigimos a la página principal, la cual contiene enlaces estáticos a su mismo host (que puede tener alguna solución de dns dinámico, o dyndns), nos encontraremos con un problema realmente peliagudo. La máquina "*NAT*eada" le preguntará al *DNS* por la IP del servidor *HTTP*, para luego intentar acceder a esa IP. Si filtrásemos basándonos en la interfaz y en la IP, la máquina "*NAT*eada" sería incapaz de llegar al servidor *HTTP*, puesto que la cadena *INPUT* simplemente desecharía los paquetes. Ésto también puede aplicarse de alguna forma en el caso de que tengamos una IP estática, pero aquí tenemos la oportunidad de evitar el problema añadiendo reglas que chequeen si hay paquetes destinados a la INET_IP que provengan de la interfaz *LAN*, en cuyo caso los aceptará.

Como puedes observar, puede ser una buena idea conseguir un script (o escribirlo) que gestione más razonablemente las IPs dinámicas. Así, por ejemplo podemos escribir un script que capte la IP a través de **ifconfig** y la asigne a una variable, tras el arranque inicial de la conexión a Internet. Una buena forma de conseguirlo sería emplear los scripts ip-up que ofrece **pppd**, entre otros programas. Un buen sitio para buscar scripts es la página web sobre iptables "linuxguruz.org", dónde podrás encontrar y descargar montones de éllos. Encontrarás el enlace en el apéndice *Otras fuentes y enlaces*.

> **Note:** El script rc.DHCP.firewall.txt puede ser un poco más inseguro que el rc.firewall.txt, por lo que te sugiero que utilices éste último al no ser tan susceptible a ataques desde el exterior.

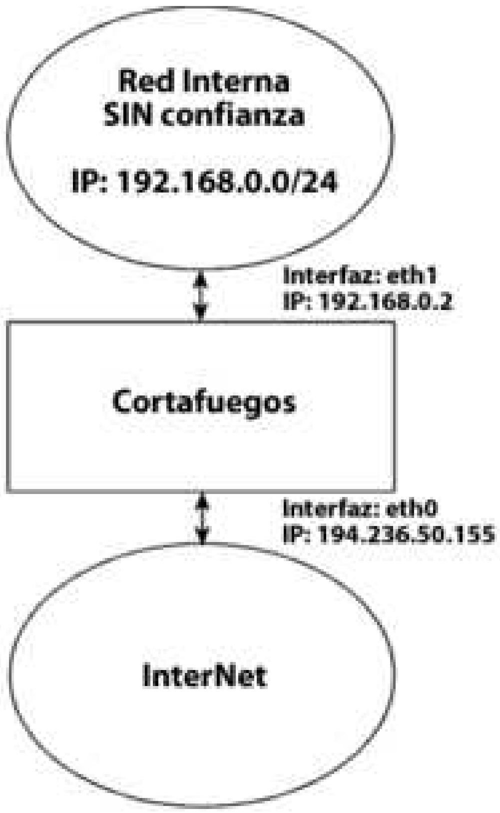
Además, existe la posibilidad de añadir algo parecido a lo siguiente en tus scripts:

```
INET_IP=‘ifconfig $INET_IFACE | grep inet | cut -d : -f 2 | \
cut -d ' ' -f 1‘
```

Con éllo se captaría automáticamente la *dirección IP* de la variable **$INET_IFACE**, buscando la línea que contuviera la *dirección IP* y eliminando todo lo innecesario hasta obtener una dirección manejable. Para hacer todo ésto de una manera más elaborada, puedes aplicar las porciones de código disponibles en el script retreiveip.txt (scripts/retrieveip.txt), que captarán automáticamente tu dirección IP de Internet al ejecutarlo. Ten en cuenta que, en cambio, con éllo se pueden obtener comportamientos algo extraños, como conexiones ralentizadas desde y hacia el cortafuegos en la parte interna. Los comportamientos extraños más comunes se describen en la siguiente lista:

1. Si el script se ejecuta desde otro script, que a su vez es ejecutado por, por ejemplo, el *demonio PPP*, todas las conexiones activas en ese momento se "colgarán", debido a las reglas "NEW not SYN" (léete el capítulo *Paquetes cuyo estado es NEW pero cuyo bit SYN no se ha establecido*). Es posible evitarlo si, por ejemplo, eliminas las reglas *NEW not SYN*, aunque esta solución es cuestionable.
2. Si tienes reglas que son estáticas y siempre tienen que estar activas, es bastante difícil añadir y borrar reglas contínuamente sin dañar las ya existentes. Por ejemplo, si quieres bloquear todos los intentos de conexión al cortafuegos provenientes de los hosts de tu *LAN* y, al mismo tiempo, ejecutar un script desde el *demonio PPP*, ¿cómo lo consigues sin borrar las reglas activas que están bloqueando tu *LAN*?
3. Puede complicarse innecesariamente, tal como se ha visto, lo cual puede llevar a comprometer la seguridad. Si el script se mantiene simple, es más fácil descubrir problemas y mantener un orden.

rc.UTIN.firewall.txt



El script rc.UTIN.firewall.txt (http://iptables-tutorial.frozentux.net/scripts/rc.UTIN.firewall.txt), al contrario que el resto de scripts, bloqueará la red local (*LAN*) que haya por detrás nuestro. Es decir, que no nos fiaremos de nadie en ninguna red a la que estemos conectados. Además tampoco permitiremos a aquellos que estén en nuestra *LAN* que puedan hacer nada excepto tareas específicas en Internet. Lo único que permitiremos será el acceso a los servicios *POP3*, *HTTP* y *FTP* de Internet. Tampoco permitimos el acceso al cortafuegos a los usuarios internos en mayor medida que los usuarios desde Internet.

El script rc.UTIN.firewall.txt requiere que las opciones listadas más abajo se compilen estáticamente, o como módulos, en el núcleo. Sin una o más de éllas, tendrá un comportamiento defectuoso puesto que habrán funcionalidades necesarias que no podrán emplearse. Además, conforme vayas modificando el script según tus necesidades, posiblemente necesites más opciones compiladas en el núcleo.

- CONFIG_NETFILTER
- CONFIG_IP_NF_CONNTRACK
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_MATCH_LIMIT
- CONFIG_IP_NF_MATCH_STATE
- CONFIG_IP_NF_FILTER
- CONFIG_IP_NF_NAT
- CONFIG_IP_NF_TARGET_LOG

El script sigue la regla de oro de no fiarse de nadie, ni siquiera de tus propios empleados [en muchas ocasiones, aún menos de éllos]. Es una triste realidad, pero una gran parte de los "hacks" y "cracks" (accesos ilícitos) que sufre una compañía son debidos a personal de la propia plantilla. Quizá este script te sugiera algunas pistas sobre qué puedes hacer con tu cortafuegos para hacerlo más invulnerable. No es demasiado diferente al script rc.firewall.txt, pero ofrece algunas ideas sobre lo que normalmente dejaríamos pasar, ...

rc.test-iptables.txt

El script rc.test-iptables.txt (http://iptables-tutorial.frozentux.net/scripts/rc.test-iptables.txt) se puede utilizar para chequear las diferentes cadenas, aunque puede necesitar algún ajuste que otro en función de tu configuración, como activar el **ip_forwarding**, configurar el *enmascaramiento* (masquerading), etc. Prácticamente le irá bien a todo aquél que tenga la configuración básica y las tablas básicas cargadas en el núcleo. Lo que hace es establecer algunos objetivos **LOG**, que registrarán las peticiones y respuestas a los ping. De esta forma obtendrás información sobre qué cadenas han sido atravesadas y en qué orden. Por ejemplo, ejecuta el script y después lanza el siguiente comando en la shell:

ping -c 1 un.host.de.internet

Y **tail -n 0 -f /var/log/messages** mientras se está ejecutando el primer comando. De esta forma deberías ver las diferentes cadenas empleadas y en qué orden, a no ser que las entradas del registro se trastoquen por algún motivo.

Note

Este script se escribió únicamente para tareas de chequeo. O sea, no es una buena idea tener reglas de este estilo que lo registren todo, puesto que las particiones que tengas asignadas para los ficheros de registro pueden llenarse rápidamente y éllo podría convertirse en un ataque de Denegación de Servicios (DoS) muy efectivo contra tí, pudiendo conducir a ataques reales que no serían registrados tras el ataque DoS inicial.

rc.flush-iptables.txt

El script rc.flush-iptables.txt (http://iptables-tutorial.frozentux.net/scripts/rc.flush-iptables.txt) en realidad no se debería llamar un script en sí mismo. Lo que hará es reiniciar y "lavarle la cara" a todas tus tablas y cadenas. El script comienza por establecer la política por defecto a **ACCEPT** en las cadenas *INPUT*, *OUTPUT* y *FORWARD* de la tabla *filter*. Tras ésto, reiniciará las políticas por defecto de las cadenas *PREROUTING*, *POSTROUTING* y *OUTPUT* de la tabla *nat*. Se hace todo ésto en primer lugar para no tener que preocuparnos de conexiones cerradas y paquetes que no pueden entrar. El script está pensado para configurar y depurar errores en tu cortafuegos, por lo que sólo tenemos en cuenta el trabajo de abrir el cortafuegos y reiniciar todo a sus valores por defecto.

Tras éllo se limpian todas las cadenas, en primer lugar en la tabla *filter* y a continuación en la tabla *NAT*. De esta forma nos aseguramos que no hay reglas redundantes escondidas por cualquier sitio. Cuando ya se ha conseguido ésto, se salta a la siguiente sección, dónde borraremos todas las cadenas especificadas por el usuario en las tablas *NAT* y *filter*. Tras este paso se considera ejecutado el script. Sin embargo, puedes considerar necesario añadir reglas para limpiar la tabla *mangle*, si es que la utilizas.

> Un apunte final: determinadas personas me han escrito pidiéndome que añada este script en el script original rc.firewall utilizando la sintaxis de Red Hat Linux, de forma que escribas algo parecido a "rc.firewall start" y el script se ejecute. Sin embargo no lo voy a hacer ya que éste es un tutorial y debería ser entendido principalmente como un lugar dónde encontrar ideas, por lo que no debería llenarse de scripts de shell [línea de comandos] y sintaxis extrañas. Añadir tales funcionalidades haría los scripts difíciles de leer desde el punto de vista que me he propuesto, ya que el tutorial se ha escrito con la legibilidad como pilar fundamental y así seguirá siendo.

Limit-match.txt

El script limit-match.txt (http://iptables-tutorial.frozentux.net/scripts/limit-match.txt) es un test de chequeo de menor importancia que te permitirá comprobar la comparación límite (limit match) y ver cómo funciona. Carga el script y empieza a enviar paquetes ping a diferentes intervalos para ver cuáles atraviesan el cortafuegos y con qué frecuencia lo consiguen. Todas las *echo replies* (respuestas de eco) serán bloqueadas hasta que se alcance de nuevo el umbral del "burst limit".

Pid-owner.txt

El script pid-owner.txt (http://iptables-tutorial.frozentux.net/scripts/pid-owner.txt) es un pequeño ejemplo que muestra cómo podemos usar la comparación "PID owner" [propietario del número de identificación de proceso]. No hace nada útil, pero deberías ser capaz de ejecutarlo y ver en la salida de **iptables -L -v** que la regla está funcionando.

Sid-owner.txt

El script sid-owner.txt (http://iptables-tutorial.frozentux.net/scripts/sid-owner.txt) es otro pequeño ejemplo, que en este caso muestra cómo podemos usar la comparación "SID owner". No hace nada útil, pero deberías ser capaz de ejecutarlo y ver en la salida de **iptables -L -v** que la regla está funcionando.

Ttl-inc.txt

Un pequeño script de ejemplo: ttl-inc.txt (http://iptables-tutorial.frozentux.nert/scripts/ttl-inc.txt). Este script muestra cómo hacer invisible al cortafuegos/enrutador frente a trazadores de ruta (traceroutes), que de otra forma ofrecerían bastante información a posibles atacantes.

Iptables-save

Un pequeño script de ejemplo utilizado en el capítulo *Salvando y restaurando grandes conjuntos de reglas* y que ilustra cómo se puede usar el comando iptables-save. Este script no es funcional, por lo que no debería usarse para otra cosa distinta a tenerlo como referencia.

Appendix A. Explicaciones detalladas sobre comandos especiales

Haciendo un listado del conjunto de reglas activo

Para conseguir un listado del conjunto de reglas que estás empleando, debes ejecutar una opción especial del comando **iptables**, que ya hemos desarrollado brevemente en el capítulo
Cómo se escribe una regla. Deberás escribir lo siguiente:

iptables -L

Con éllo obtendrás un listado del conjunto de reglas con algunas transformaciones para que sea más legible. Por ej., los puertos se traducirán según las equivalencias presentes en el archivo /etc/services y las direcciones IP serán traducidas según los nombres equivalentes que reconozca el servidor *DNS*. En concreto ésto último puede dar lugar a algún problema: intentará resolver (traducir) direcciones IP de la red local (como ***192.168.1.1*** ó ***192.168.0.0/16***) a nombres más legibles (como puede ser www.google.com); sin embargo estas direcciones son privadas y no podrá resolver nada, con lo cual el comando parecerá colgarse mientras intenta traducir la IP. Para evitar este problema tendremos que hacer algo similar a:

iptables -L -n

Otra cosa que puede ser interesante es poder ver algunas estadísticas sobre cada política, regla y cadena. Podemos conseguirlo añadiendo la opción (flag) "verbose" (detallado):

iptables -L -n -v

No olvides que también es posible enumerar las tablas nat y mangle mediante la opción -t; así:

iptables -L -t nat

Además hay unos cuantos ficheros a los que puede ser interesante echar un vistazo en el sistema de ficheros /proc. Por ejemplo, puede interesarnos saber qué conexiones hay actualmente en la tabla de seguimiento de conexiones: esta tabla contiene las diferentes conexiones que se están monitorizando en este momento y sirve como tabla básica dónde conocer el estado en que actualmente se encuentra una conexión. Esta tabla no se puede editar, y aunque se pudiera, no sería una buena idea. Para verla puedes ejecutar el siguiente comando:

cat /proc/net/ip_conntrack | less

Con el anterior comando veremos todas las conexiones monitorizadas, si bien puede resultar algo difícil entenderlo todo.

Actualizando y "aseando" tus tablas

Si en algún momento "estropeas" tus tablas de **iptables**, hay comandos para "arreglarlas", de forma que no tendrás que reiniciar el sistema. Ya me han preguntado sobre ésto en un par de ocasiones, así que he pensado contestar aquí: si has añadido una regla por error, lo más sencillo es cambiar el parámetro **-A** por el **-D** en la línea que has añadido por error. **iptables** encontrará esa línea errónea y la borrará; si tuvieras varias líneas exactamente iguales en la cadena, borrará la primera cadena que encuentre y sea coincidente con la regla que deseas borrar. Si ésto no es lo que deseas, puedes intentar usar la opción **-D** como sigue: **iptables -D INPUT 10**, ejemplo que borraría la décima regla de la cadena *INPUT*.

Tambien hay situaciones en las que querrás borrar una cadena entera. Para ésto, puedes usar la opción **-F**: por ej., con **iptables -F INPUT** borrarás la cadena *INPUT* entera, aunque no modificarás su política por defecto, que si está establecida como *DROP* (desechar), con el ejemplo anterior bloquearás completamente la cadena *INPUT* (todo paquete que entre en esta cadena será desechado). Para modificar la política por defecto de la cadena, haz lo mismo que hiciste para establecerla, por ejemplo **iptables -P INPUT ACCEPT** (en este ejemplo cambiarás de desechar a aceptar los paquetes).

He creado un pequeño script (http://iptables-tutorial.frozentux.net/scripts/rc.flush-iptables.txt) (disponible también como un apéndice) que limpiará y reinicializará tus tablas de **iptables**, lo cual puede que creas conveniente cuando quieras poner a punto el archivo rc.firewall.txt. Sólo un apunte, si has estado trasteando en la tabla *mangle*, este script no borrará nada en élla, aunque es bastante simple añadir las pocas líneas de código necesarias para hacerlo. La razón es simple: no utilizo la tabla *mangle* para nada en el script rc.firewall.txt.

Appendix B. Problemas y preguntas frecuentes

Problemas en la carga de módulos

Puedes tener algún problema a la hora de cargar módulos. Por ejemplo, pueden haber errores que indiquen que no existe ningún módulo con ese nombre. A modo de ejemplo, pueden tener el siguiente aspecto (como podrás ver a lo largo del tutorial, los ejemplos no se han traducido y continúan en inglés, ya que todo el sistema Netfilter/iptables se ha programado en inglés y de esta manera estudiarás los mensajes con el aspecto más similar a como puedas verlos en tu propio ordenador):

```
insmod: iptable_filter: no module by that name found
```

Traducción: no se ha encontrado ningún módulo con ese nombre. Sin embargo no hay por qué preocuparse. Ese o esos módulos posiblemente hayan sido compilados estáticamente en el núcleo (kernel). Esto será lo primero que tendrás que comprobar cuando intentes solucionar el problema. La forma más sencilla de saber si los módulos ya se han cargado o si se han compilado estáticamente en el núcleo, es ejecutar un comando que utilice esa funcionalidad específica y ver lo que ocurre. En el caso anterior no se ha podido cargar la tabla de filtrado (tabla *filter*). Si esta funcionalidad no está cargada, seremos incapaces de utilizar la tabla de filtrado, por lo que podremos probar con:

iptables -t filter -L

Con éllo se deberían listar todas las cadenas de la tabla *filter*, o bien podría fallar. Si todo es correcto, el resultado puede ser semejante a lo siguiente (dependiendo si tienes alguna regla en la tabla o no):

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Si no tienes la tabla *filter* cargada obtendrás un error parecido a:

```
iptables v1.2.5: can't initialize iptables table 'filter': Table \
    does not exist (do you need to insmod?)
Perhaps iptables or your kernel needs to be upgraded.
```

Este error ya es algo más serio, puesto que para empezar nos dice que no tenemos esa funcionalidad compilada en el núcleo. Además, no es posible encontrar el módulo en nuestras rutas de acceso a los módulos. Esto puede significar que has olvidado instalar los módulos, que has olvidado ejecutar **depmod -a** para actualizar las bases de datos de los módulos, o que no has compilado la funcionalidad como módulo o estáticamente en el núcleo. Por supuesto pueden haber otras razones para que el módulo no esté cargado, pero estas son las más comunes. La mayoría de ellas se resuelven fácilmente. El primer problema se puede eliminar ejecutando **make modules_install** en el directorio de instalación del núcleo (siempre que el código fuente ya se haya compilado y los módulos ya se hayan creado). El segundo problema se resuelve ejecutando **depmod -a** una vez y comprobar si todo funciona a partir de entonces. El tercer problema se sale de los objetivos de estas explicaciones y debes solucionarlo por tu cuenta. Probablemente encontrarás información útil en Linux Documentation Project homepage (http://www.tldp.org) (en inglés) o también en TLDP-ES/LuCAS (http://es.tldp.org) (más o menos la misma página que la anterior, pero en castellano).

Otro error que puede ocurrir cuando ejecutes iptables es:

```
iptables: No chain/target/match by that name
```

Este error indica que no existe esa cadena, ese objetivo o esa comparación. Depende de un montón de factores, siendo el más común que te has equivocado al escribir la cadena, objetivo o comparación que deseabas. También puede ser que no hayas cargado el módulo adecuado, que no fue compilado en el núcleo o que iptables no ha sido capaz de cargar automáticamente el módulo. En general debes revisar todo lo anterior, pero también deberías revisar si la regla está correctamente escrita (en el conjunto de reglas de iptables).

Paquetes cuyo estado es NEW pero cuyo bit SYN no se ha establecido

Existe cierta *característica* de **iptables** que no está muy bien documentada y puede ser obviada por mucha gente (sí, incluso yo). Si empleas el estado **NEW** de los paquetes, aquellos con el bit *SYN* sin establecer pasarán por el cortafuegos. Se ha programado así porque en determinadas ocasiones se necesita considerar que un paquete puede ser parte de una conexión ya establecida (**ESTABLISHED**) en, por ejemplo, otro cortafuegos. Con éllo es posible tener dos o más cortafuegos y poder "caerse" ("colgarse") uno de ellos sin pérdida de datos: el trabajo de filtrado de la subred lo puede asumir automáticamente el cortafuegos secundario. Sin embargo ésto conduce al hecho de que el estado **NEW** permitirá casi cualquier tipo de conexión TCP, independientemente de si se trata de un "saludo" inicial (un establecimiento de conexión) o no. Para evitar este problema se añaden las siguientes reglas a las cadenas *INPUT*, *OUTPUT* y *FORWARD* de los cortafuegos:

```
$IPTABLES -A INPUT -p tcp ! --syn -m state --state NEW -j LOG \ --log-prefix "New not syn:"
$IPTABLES -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
```

Las reglas anteriores evitarán este problema. Nos encontramos ante un comportamiento mal documentado del proyecto **Netfilter**/**iptables** que debería ser claramente expuesto para el conocimiento general. Para ser más claro, que quede como un enorme aviso acerca de este tipo de comportamiento en tu cortafuegos.

Ten en cuenta que existen algunas incompatibilidades entre las reglas anteriores y las malas implementaciones (adaptaciones) del TCP/IP realizadas por Microsoft. Las reglas anteriores conducirán a ciertas condiciones en que los paquetes generados por los productos de Microsoft serán etiquetados con el estado **NEW** y por tanto registrados y desechados. De todas formas, por lo que sé no derivarán en pérdidas de conexiones. El problema ocurre cuando una conexión se cierra, se envía el paquete *FIN*/*ACK* final, la máquina de estados (state machine) de **Netfilter** cierra la conexión y a partir de entonces ya no existe en la tabla del seguimiento de conexiones (conntrack table). Llegados a este punto, la defectuosa implementación de Microsoft envía otro paquete que es considerado **NEW**, pero carece del bit *SYN* y por ello es interceptado por las reglas anteriores. O sea, no te preocupes demasiado por estas reglas, aunque si lo haces incluye la opción **--log-headers** a la regla y registra también las cabeceras, de forma que podrás analizar mejor cómo es el paquete.

Aún existe otro problema conocido con estas reglas: cuando alguien está conectado al cortafuegos (por ejemplo desde la red local) y has configurado el script (guión, conjunto de comandos de shell) para que se active al producirse una conexión *PPP*. En este caso, cuando estableces una conexión *PPP*, aquel que estuviera previamente conectado a través de la red local será poco menos que eliminado. Aunque es cierto que ésto sólo ocurre cuando estás trabajando con el seguimiento de conexiones y los códigos base de NAT cargados ambos como módulos, y además los módulos son cargados y descargados cada vez que ejecutas el script. Otra forma de llegar a este problema es ejecutar el script rc.firewall.txt mediante una conexión telnet, desde un servidor situado fuera de la red del cortafuegos. Para aclararlo: conectas mediante **telnet** u otro tipo de conexión, arrancas los módulos de seguimiento de conexiones, después cargas las reglas "**NEW** not *SYN*" y por último, el **cliente/demonio telnet** intenta enviar algo. Es entonces cuando el código de seguimiento de conexiones no reconoce esta conexión como legal, puesto que no tiene constancia de ningún paquete en ninguna dirección para esta conexión; además no habrá ningún bit *SYN* establecido, puesto que no es el primer paquete de la conexión. Como consecuencia, el paquete será interceptado por estas reglas y después de añadirlo al registro, será eliminado.

Paquetes SYN/ACK y NEW

Ciertos ataques de "camuflaje" (spoofing) TCP emplean una técnica llamada Predicción de Secuencia Numérica (Sequence Number Prediction). En éllos el atacante simula la dirección IP de otra máquina, mientras ataca a alguien e intenta predecir la secuencia numérica empleada por el ordenador al que está suplantando.

Veamos un caso típico. "Jugadores": atacante [A], intentando enviar paquetes a la víctima [V], e intentando hacerse pasar por otra máquina [O] (otro host).

1. [A] le envía un SYN a [V] con la dirección IP de origen de [O].
2. [V] responde a [O] con un SYN/ACK.
3. Entonces [O] debería responder a un SYN/ACK desconocido con un RST y entonces el ataque no tendría éxito, pero asumamos que [O] no está operativo (está desbordado, apagado o tras un cortafuegos que ha rechazado los paquetes).
4. Si [O] no se interpone entre [A] y [V], entonces [A] podrá "hablar" con [V] haciéndose pasar por [O], siempre que pueda predecir la secuencia de números correcta.

Mientras no enviemos el paquete RST en respuesta al SYN/ACK desconocido en el paso 3, permitiremos que [V] sea atacado y nos echen la culpa a nosotros. Así pues, será pura cortesía enviar adecuadamente el RST a [V]. Si utilizas las reglas "NEW not SYN" (explicadas antes) en tu conjunto de reglas, los paquetes SYN/ACK serán desechados. Por éllo se añaden las siguientes reglas en la cadena *bad_tcp_packets*, justo antes de las reglas "NEW not SYN":

```
iptables -A bad_tcp_packets -p tcp --tcp-flags SYN,ACK SYN,ACK \
-m state --state NEW -j REJECT --reject-with tcp-reset
```

Gracias a éllo, las posibilidades de ser [O] son relativamente pequeñas, al tiempo que son unas reglas seguras en casi todos los casos, excepto cuando ejecutas varios cortafuegos redundantes, en los que los unos se hacen cargo a menudo de los flujos de datos de los otros. En este caso algunas conexiones pueden ser bloqueadas, incluso si son legítimas. Esta regla también permite que algunos exploradores de puertos (portscanners) sean capaces de ver nuestro cortafuegos, pero no serán capaces de averiguar mucho más que ésto (que el cortafuegos está ahí).

Proveedores de Acceso a Internet que emplean direcciones IP asignadas

He añadido este problema porque un amigo me dijo algo que había olvidado completamente. Ciertos Proveedores de Acceso a Internet (ISPs) realmente estúpidos asignan las direcciones IP de las redes locales (asignadas por el *IANA*) para la configuración de conexión de todos sus clientes. Por ejemplo, el ISP y monopolio sueco Telia emplea este método por ejemplo en sus servidores *DNS* (Servidor de Nombres de Dominio), que cubren el rango de direcciones 10.x.x.x. El problema es que en este script no se permiten conexiones desde ninguna dirección IP del rango 10.x.x.x, debido a las posibilidades de simulación de la dirección IP de origen (el "spoofing", ya discutido en el punto anterior, es una técnica mediante la cual el ordenador atacante modifica la cabecera de los paquetes IP de forma que el servidor/cortafuegos crean que provienen de una dirección IP en la que se confía o que al menos no es sospechosa, de forma que atraviesan el cortafuegos sin problemas). Bueno, tristemente es una situación en la que debes olvidarte de estas reglas para esas direcciones, insertando un **ACCEPT** antes de la sección Spoof, para permitir el tráfico desde esos servidores *DNS*, o puedes simplemente convertir en comentario esa parte del script. Más o menos se debe parecer a ésto:

```
/usr/local/sbin/iptables -t nat -I PREROUTING -i eth1 -s \
    10.0.0.1/32 -j ACCEPT
```

Me gustaría tomarme un momento para maldecir a esos ISPs. Estos rangos de direcciones IP no se han asignado para tonterías como ésa, por lo menos no que yo sepa. Para la red de grandes empresas, para la red local de tu casa, éstos sí son buenos ejemplos y no el hecho de que estés obligado a abrir tu red por la "feliz" idea de algún listillo.

Permitir peticiones DHCP a través de iptables

En realidad esta es una tarea bastante sencilla, una vez sabes cómo trabaja *DHCP*. Sin embargo debes ser algo cuidadoso acerca de lo que dejas pasar por tu cortafuegos y lo que no. Para empezar, debes saber que *DHCP* trabaja sobre el protocolo *UDP*. Por éllo es lo primero que debes mirar. Además, debes chequear a qué interfaz le enviamos peticiones y cuál las responde. Por ejemplo, si es la interfaz *eth0* la que está configurada para trabajar a través de *DHCP*, no debes permitir peticiones *DHCP* en *eth1*. Para que la regla sea más específica, sólo permitiremos los puertos *UDP* realmente empleados por *DHCP*, que deberían ser los puertos 67 y 68. Estos serán los criterios que emplearemos para comparar paquetes y permitir (o no) su paso. Así pues, la regla quedará poco más o menos así:

```
$IPTABLES  -I INPUT -i $LAN_IFACE -p udp --dport 67:68 --sport \
    67:68 -j ACCEPT
```

Ten en cuenta que de esta forma se permite todo el tráfico hacia y desde los puertos *UDP* 67 y 68, aunque ésto no tiene por qué ser un gran problema, puesto que sólo permite peticiones desde servidores efectuando la conexión desde los mismos puertos (67 y 68). Si te preocupa esta confianza casi ciega, la regla se puede hacer más restrictiva, por supuesto, pero debería bastar para aceptar todas las peticiones y actualizaciones *DHCP* sin abrir un agujero demasiado grande en el cortafuegos.

Problemas con mIRC DCC

mIRC emplea una configuración especial que le permite conectar a través de un cortafuegos y conseguir que las conexiones DCC funcionen correctamente sin que el cortafuegos se entere de nada. Si esta opción se emplea conjuntamente con iptables y, en concreto, con los módulos ip_conntrack_irc e ip_nat_irc, simplemente no funcionará. El problema es que mIRC efectuará automáticamente un cambio de la dirección de red (NAT) al paquete y cuando éste llegue al cortafuegos, iptables no sabrá qué hacer con él. La cuestión es que mIRC no espera que exista ningún cortafuegos lo suficientemente eficiente como para hacer bien el cambio de dirección: no se cree que simplemente preguntando al servidor IRC su dirección IP, el cortafuegos pueda enviar peticiones DCC con esa dirección.

Vamos, que activar la opción "estoy detrás de un cortafuegos" ("I am behind a firewall") de la configuración de mIRC y emplear los módulos ip_conntrack_irc e ip_nat_irc, únicamente servirá para crear entradas en el registro diciendo "Paquete de envío DCC falsificado" ("Forged DCC send packet").

La solución más simple es no emplear dicha opción de la configuración de mIRC y dejar que iptables haga su trabajo. O sea, debes decirle a mIRC que *no* está detrás de un cortafuegos.

Appendix C. Tipos ICMP

He aquí un listado completo de los tipos ICMP. Las columnas "Pregunta" y "Error" indican si los diferentes tipos son un mensaje de error (al que nunca se contestará) o una pregunta (a la que se podrá contestar):

Table C-1. Tipos ICMP

TIPO	CÓDIGO	Descripción	Pregunta	Error
0	0	Echo Reply (respuesta de eco a un comando ping)	x	
3	0	Network Unreachable (red inalcanzable)		x
3	1	Host Unreachable (host inalcanzable)		x
3	2	Protocol Unreachable (protocolo inalcanzable)		x
3	3	Port Unreachable (puerto inalcanzable)		x
3	4	Fragmentation needed but no-frag. bit set (es necesaria la fragmentación, pero se ha establecido el bit de no-fragmentar)		x
3	5	Source routing failed (ha fallado el encaminamiento exigido en origen: se ha especificado el camino/enrutado que deben seguir los paquetes y uno de los puntos de la ruta no está disponible)		x
3	6	Destination network unknown (dirección de destino desconocida)		x
3	7	Destination host unknown (host de destino desconocido)		x

TIPO	CÓDIGO	Descripción	Pregunta	Error
3	8	Source host isolated (host de origen aislado; este tipo está obsoleto)		x
3	9	Destination network administratively prohibited (red de destino prohibida administrativamente)		x
3	10	Destination host administratively prohibited (host de destino prohibido administrativamente)		x
3	11	Network unreachable for TOS (red inalcanzable para el TOS, el tipo de servicio)		x
3	12	Host unreachable for TOS (host inalcanzable para el TOS)		x
3	13	Communication administratively prohibited by filtering (comunicación prohibida administrativamente mediante filtrado)		x
3	14	Host precedence violation (violación del precedente del host)		x
3	15	Precedence cutoff in effect (está actuando el límite de precedente)		x
4	0	Source quench (le indica al origen "que se calme un poco" porque está saturando la capacidad de proceso del receptor; actualmente está en desuso para no saturar aún más la comunicación)		
5	0	Redirect for network (indica que debes redireccionar tus comunicaciones a otra red)		

TIPO	CÓDIGO	Descripción	Pre-gunta	Error
5	1	Redirect for host (indica que debes redireccionar tus comunicaciones a otro host)		
5	2	Redirect for TOS and network (indica que debes redireccionar tus comunicaciones con otro TOS y a otra red)		
5	3	Redirect for TOS and host (indica que debes redireccionar tus comunicaciones con otro TOS y a otro host)		
8	0	Echo request (petición de eco/ping)	x	
9	0	Router advertisement (aviso de existencia del enrutador: "¡Hola, estoy aquí!")		
10	0	Router solicitation (solicitud de existencia de enrutador: "¿Hay alguien ahí?")		
11	0	TTL equals 0 during transit (TTL igual a 0 durante el tránsito: al paquete se le ha acabado su tiempo de vida antes de alcanzar su destino)		x
11	1	TTL equals 0 during reassembly (TTL igual a 0 durante el reensamblado: si un paquete ha sido fragmentado y durante su reensamblaje el TTL llega a 0, se genera este error)		x

TIPO	CÓDIGO	Descripción	Pregunta	Error
12	0	IP header bad (catchall error) [cabecera IP errónea (error de "cajón de sastre", o error para todo lo que no esté especificado en otro sitio; o si lo prefieres, error por defecto)]		x
12	1	Required options missing (faltan opciones requeridas)		x
13	0	Timestamp request (petición de la hora en origen: "¿Qué hora es ahí?"; está obsoleto)	x	
14		Timestamp reply (respuesta a la petición de hora: "Son las ..."; está obsoleto)	x	
15	0	Information request (petición de información; está obsoleto)	x	
16	0	Information reply (respuesta a la petición de información; está obsoleto)	x	
17	0	Address mask request (petición de máscara de red: cuando un host sin disco duro se inicializa, efectúa una petición para saber qué máscara de red debe utilizar)	x	
18	0	Address mask reply (respuesta a la petición de máscara de red)	x	

Appendix D. Otras fuentes y enlaces

He aquí una lista de enlaces y fuentes donde he conseguido información (mientras no se indique lo contrario, los documentos están en inglés):

- ip-sysctl.txt (http://iptables-tutorial.frozentux.net/other/ip-sysctl.txt) - del núcleo 2.4.14. Algo corto, pero una buena referencia sobre los controles de gestión IP y qué le hacen al núcleo. Documento traducido al castellano.
- Protocolo de Mensajes de Control de Internet (http://www.ee.siue.edu/~rwalden/networking/icmp.htm) - Un documento bueno y breve que describe el protocolo ICMP detalladamente. Escrito por Ralph Walden.
- RFC 792 - Protocolo de Mensajes de Control de Internet (http://ipsysctl-tutorial.frozentux.net/other/rfc792.txt) - La fuente de información definitiva acerca de los paquetes ICMP. Sea cual sea la información técnica que necesites saber acerca del protocolo ICMP, aquí es donde primero debes buscar. Escrito por J. Postel.
- RFC 793 - Protocolo de Control de Transmisiones (http://iptables-tutorial.frozentux.net/other/rfc793.txt) - Esta es la fuente de información original sobre cómo debe comportarse TCP en todos los hosts. Este documento es el estándar sobre el funcionamiento de TCP desde 1981 en adelante. Extremadamente técnico, pero de obligada lectura para aquél que quiera aprender TCP en profundidad. Originalmente fue un estándar del Departamento de Defensa (Americano) escrito por J. Postel.
- ip_dynaddr.txt (http://iptables-tutorial.frozentux.net/other/ip_dynaddr.txt) - del núcleo 2.4.14. Una referencia muy corta sobre las opciones de configuración de ip_dynaddr, disponibles a través de sysctl y el sistema de ficheros proc. Documento traducido al castellano.

- iptables.8 (http://iptables-tutorial.frozentux.net/other/iptables.html) - El manual de iptables 1.2.4. Ésta es una versión HTML del manual, que es una referencia excelente al leer/escribir conjuntos de reglas de iptables. Ténlo siempre a mano.
- Tabla de reglas del cortafuegos (http://iptables-tutorial.frozentux.net/other/firewall_rules_table_final.pdf) - Un pequeño documento PDF donado amablemente a este proyecto por Stuart Clark, que ofrece un formulario de referencia donde escribir toda la información necesaria para el cortafuegos, de una forma sencilla.
- http://www.netfilter.org/ - El sitio web oficial de **Netfilter** e **iptables**. De obligada visita para todo aquél que quiera configurar **iptables** y **Netfilter** en Linux.
- http://www.netfilter.org/documentation/index.html#FAQ (http://www.netfilter.org/documentation/index.html#FAQ) - Las *Frequently Asked Questions* (Preguntas Frecuentes) oficiales de **Netfilter**. Éste es también un buen lugar de comienzo cuando empiezas a preguntarte qué hacen **iptables** y **Netfilter**.
- http://www.netfilter.org/unreliable-guides/packet-filtering-HOWTO/index.html (http://www.netfilter.org/unreliable-guides/packet-filtering-HOWTO/index.html) - La Guía "Poco fiable" (Unreliable Guide) sobre filtrado de paquetes de Rusty Russells. Excelente documentación sobre filtrado básico de paquetes con **iptables**, escrito por uno de los programadores del núcleo de **iptables** y de **Netfilter**.
- http://www.netfilter.org/unreliable-guides/NAT-HOWTO/index.html (http://www.netfilter.org/unreliable-guides/NAT-HOWTO/index.html) - La Guía "Poco fiable" (Unreliable Guide) sobre la traducción de direcciones de red de Rusty Russells. Excelente documentación sobre *Network Address Translation* (NAT) en **iptables**, escrito por uno de los programadores del núcleo de **iptables** y de **Netfilter**.
- http://www.netfilter.org/unreliable-guides/netfilter-hacking-HOWTO/index.html (http://www.netfilter.org/unreliable-guides/netfilter-hacking-

HOWTO/index.html) - La Guía "Poco fiable" (Unreliable Guide) sobre optimización (hacking) de Netfilter, por Rusty Russells. Uno de los pocos documentos acerca de cómo escribir código en el espacio de usuario en **Netfilter** e **iptables**.

- http://www.linuxguruz.org/iptables/ - Excelente página de enlaces (links) a la mayoría de las páginas de Internet sobre **iptables** y **Netfilter**. También mantiene una lista de *scripts para iptables* destinados a distintos propósitos.
- http://www.islandsoft.net/veerapen.html - Excelente discusión acerca de cómo hacer más seguro **iptables** automáticamente y cómo hacer pequeños cambios que permitirán a tu ordenador añadir automáticamente todo sitio hostil a una lista especial de sitios prohibidos en **iptables**.
- /etc/protocols (http://iptables-tutorial.frozentux.net/other/protocols.txt) - Un ejemplo de fichero de protocolos sacado de la distribución Slackware Se puede usar para saber qué número tienen los diferentes protocolos, como pueden ser el protocolo IP, el ICMP o el TCP.
- /etc/services (http://iptables-tutorial.frozentux.net/other/services.txt) - Un ejemplo de fichero de servicios, también de la distribución Slackware. Es muy recomendable que lo leas de vez en cuando, especialmente si necesitas una visión general de qué puertos utilizan los distintos protocolos.
- Internet Engineering Task Force (http://www.ietf.org) (sin traducción; vendría a ser "Equipo especial de ingenieros de Internet") - Éste es uno de los grupos más grandes en lo que respecta a establecer y mantener estándares de Internet. Son los que mantienen el "almacén" (repository) de RFC y lo integran un gran grupo de compañías y particulares que trabajan conjuntamente para asegurar la interoperabilidad de Internet.
- Linux Advanced Routing and Traffic Control HOW-TO (http://www.lartc.org) (CÓMO sobre Control de Tráfico y Encaminamiento Avanzado de Linux) - Este documento es uno de los mejores y más grandes acerca del encaminamiento avanzado de

Linux. Mantenido por Bert Hubert.

- Paksecured Linux Kernel patches (http://www.paksecured.com/patches/) (parches Paksecured del núcleo de Linux) - una web que contiene todos los parches (patches) del núcleo, escrito por Matthew G. Marsh. Además de otros, el parche FTOS está disponible en ésta web.
- Página del proyecto ULOGD (http://www.gnumonks.org/gnumonks/projects/project_details?p_id=1) - La página web oficial del proyecto ULOGD.
- El Linux Documentation Project (http://www.tldp.org) (Proyecto de Documentación sobre Linux) es un buen lugar dónde buscar documentación. La mayoría de los grandes documentos para Linux están disponibles aquí, de lo contrario, tendrás que rebuscar en la Red fijándote mucho y pasando por muchas páginas hasta que encuentres lo que buscas. Si hay algo sobre lo que quieras saber más, pásate por esta web. Sin embargo recuerda que todo está en inglés. Para ver una parte de esos documentos traducidos al español tendrás que ir a la página hermana: El Proyecto LuCAS (http://es.tldp.org), dónde encontrarás bastantes documentos traducidos, aunque no todos, ni tampoco tendrás siempre las últimas actualizaciones.
- http://kalamazoolinux.org/presentations/20010417/conntrack.html (http://kalamazoolinux.org/presentations/20010417/conntrack.html) - Esta presentación contiene una explicación excelente de los módulos de seguimiento de conexiones (conntrack) y cómo trabajan en Netfilter. Si estás interesado en este tema, debes leertelo.
- http://www.docum.org - Información excelente sobre los comandos *CBQ*, **tc** e **ip** de Linux. Uno de los pocos sitios que tiene información sobre estos programas. Mantenido por Stef Coene.
- http://lists.samba.org/mailman/listinfo/netfilter (http://lists.samba.org/mailman/listinfo/netfilter) - La lista de correo oficial de Netfilter. Extremadamente útil si tienes dudas sobre algo no tratado en este tutorial o que no se encuentre en algún enlace de los aquí listados.

Y, por supuesto, el código fuente de **iptables**, la documentación y los particulares que me ayudaron.

Appendix E. Agradecimientos

Quiero agradecer a las siguientes personas su ayuda durante la creación de este documento:

- *Fabrice Marie* (mailto:fabriceATcelestixDOTcom), por las grandes correcciones a mis horribles gramática y ortografía. Asímismo un enorme agradecimiento por actualizar el tutorial al formato DocBook con sus "make files" y demás.
- *Marc Boucher* (mailto:marc+nfATmbsiDOTca), por aclararme algunos aspectos del código de comparación de estados.
- *Frode E. Nyboe* (mailto:fenATimprobusDOTcom), por mejorar notablemente las reglas de rc.firewall e inspirarme mientras reescribía el conjunto de reglas, además de ser quien introdujo el paso por múltiples tablas dentro de un mismo archivo.
- *Chapman Brad* (mailto:kakadu_crocATyahooDOTcom), *Alexander W. Janssen* (mailto:yallaATynfonaticDOTde), ambos me hicieron ver que estaba equivocado acerca de cómo los paquetes atraviesan el NAT básico y las tablas de filtrado, así como el orden en que aparecen.
- *Michiel Brandenburg* (mailto:michielbATstackDOTnl), *Myles Uyema* (mailto:mylesATpuckDOTnetherDOTnet), por ayudarme con una parte del código de comparación de estados y hacerlo funcionar.
- *Kent 'Artech' Stahre* (mailto:artechATboingworldDOTcom), por ayudarme con los gráficos (eres mejor que muchos de los grafistas que conozco ;)). También te agradezco por buscar los posibles errores en el tutorial.
- *Anders 'DeZENT' Johansson*, por aconsejarme sobre extraños ISPs y algunos otros que utilizan redes reservadas en Internet, o al menos en la Internet que te permiten utilizar.
- *Jeremy 'Spliffy' Smith* (mailto:di99smjeATchlDOTchalmersDOTse), por darme consejos sobre detalles que podrían confundir a la gente y por probar y buscar errores en el texto.

Y por supuesto a todo aquel con quien he hablado y pedido opinión acerca del tutorial. Perdón por no haber mencionado a todo el mundo.

Appendix F. Historial

Versión 1.1.19 (21 Mayo 2003) http://iptables-tutorial.frozentux.net Por: Oskar Andreasson Colaboradores: Peter van Kampen, Xavier Bartol, Jon Anderson, Thorsten Bremer y el Equipo de Traducción al Español Version 1.1.18 (24 Abril 2003) http://iptables-tutorial.frozentux.net Por: Oskar Andreasson Colaboradores: Stuart Clark, Robert P. J. Day, Mark Orenstein and Edmond Shwayri. Versión 1.1.17 (6 Abril 2003) http://iptables-tutorial.frozentux.net Por: Oskar Andreasson Colaboradores: Geraldo Amaral Filho, Ondrej Suchy, Dino Conti, Robert P. J. Day, Velev Dimo, Spencer Rouser, Daveonos, Amanda Hickman, Olle Jonsson y Bengt Aspvall. Versión 1.1.16 (16 Diciembre 2002) http://iptables-tutorial.frozentux.net Por: Oskar Andreasson Colaboradores: Clemens Schwaighower, Uwe Dippel y Dave Wreski. Versión 1.1.15 (13 Noviembre 2002) http://iptables-tutorial.frozentux.net Por: Oskar Andreasson Colaboradores: Mark Sonarte, A. Lester Buck, Robert P. J. Day, Togan Muftuoglu, Antony Stone, Matthew F. Barnes y Otto Matejka. Versión 1.1.14 (14 Octubre 2002) http://iptables-tutorial.frozentux.net Por: Oskar Andreasson Colaboradores: Carol Anne, Manuel Minzoni, Yves Soun, Miernik, Uwe Dippel, Dave Klipec y Eddy L O Jansson. Versión 1.1.13 (22 Agosto 2002) http://iptables-tutorial.haringstad.com Por: Oskar Andreasson Colaboradores: Un montón de gente informándome de una mala versión HTML. Versión 1.1.12 (19 Agosto 2002) http://www.netfilter.org/tutorial/ Por: Oskar Andreasson Colaboradores: Peter Schubnell, Stephen J. Lawrence, Uwe Dippel, Bradley Dilger, Vegard Engen, Clifford Kite, Alessandro Oliveira, Tony Earnshaw, Harald Welte, Nick Andrew y Stepan Kasal. Versión 1.1.11 (27 Mayo 2002) http://www.netfilter.org/tutorial/ Por: Oskar Andreasson Colaboradores: Steve Hnizdur, Lonni Friedman, Jelle Kalf, Harald Welte, Valentina Barrios y Tony Earnshaw. Versión 1.1.10 (12 Abril 2002) http://www.boingworld.com/workshops/linux/iptables-tutorial/ Por: Oskar Andreasson Colaboradores: Jelle Kalf, Theodore Alexandrov, Paul Corbett, Rodrigo Rubira Branco, Alistair Tonner, Matthew G. Marsh, Uwe Dippel, Evan Nemerson y Marcel J.E. Mol. Versión 1.1.9 (21 Marzo 2002) http://www.boingworld.com/workshops/linux/iptables-tutorial/ Por: Oskar Andreasson Colaboradores: Vince Herried, Togan Muftuoglu, Galen Johnson, Kelly Ashe, Janne Johansson, Thomas Smets, Peter Horst, Mitch Landers, Neil Jolly, Jelle Kalf, Jason Lam y Evan Nemerson. Versión 1.1.8 (5 Marzo 2002) http://www.boingworld.com/workshops/linux/iptables-tutorial/ Por: Oskar Andreasson Versión 1.1.7 (4 Febrero 2002) http://www.boingworld.com/workshops/linux/iptables-tutorial/ Por: Oskar Andreasson Colaboradores: Parimi Ravi, Phil Schultz, Steven McClintoc, Bill Dossett, Dave Wreski, Erik Sjölund, Adam Mansbridge, Vasoo Veerapen, Aladdin y Rusty Russell. Versión 1.1.6 (7 Diciembre 2001) http://people.unix-fu.org/andreasson/ Por: Oskar Andreasson Colaboradores: Jim Ramsey, Phil Schultz, Göran Båge, Doug Monroe, Jasper Aikema, Kurt Lieber, Chris Tallon, Chris Martin, Jonas Pasche, Jan Labanowski, Rodrigo R. Branco, Jacco van Koll y Dave Wreski. Versión 1.1.5 (14 Noviembre 2001) http://people.unix-fu.org/andreasson/ Por: Oskar Andreasson Colaboradores: Fabrice Marie, Merijn Schering y Kurt Lieber. Versión 1.1.4 (6 Noviembre 2001) http://people.unix-fu.org/andreasson Por: Oskar Andreasson Colaboradores: Stig W. Jensen, Steve Hnizdur, Chris Pluta y Kurt Lieber. Versión 1.1.3 (9 Octubre 2001) http://people.unix-fu.org/andreasson Por: Oskar Andreasson Colaboradores: Joni Chu, N.Emile Akabi-Davis y Jelle Kalf. Versión 1.1.2 (29 Septiembre 2001) http://people.unix-fu.org/andreasson Por: Oskar Andreasson Versión 1.1.1 (26 Septiembre 2001) http://people.unix-fu.org/andreasson Por: Oskar Andreasson Colaboradores: Dave Richardson. Versión 1.1.0 (15 Septiembre 2001) http://people.unix-fu.org/andreasson Por: Oskar Andreasson Versión 1.0.9 (9 Septiembre 2001) http://people.unix-fu.org/andreasson Por: Oskar Andreasson Versión 1.0.8 (7 Septiembre 2001) http://people.unix-fu.org/andreasson Por: Oskar Andreasson Versión 1.0.7 (23 Agosto 2001) http://people.unix-fu.org/andreasson Por: Oskar Andreasson Colaboradores: Fabrice Marie. Versión 1.0.6 http://people.unix-fu.org/andreasson Por: Oskar Andreasson Versión 1.0.5 http://people.unix-fu.org/andreasson Por: Oskar Andreasson Colaboradores: Fabrice Marie.

Appendix G. Licencia de Documentación Libre GNU

Versión 1.1, Marzo de 2000

Note

Traducción a Español de la GNU Free Document License 1.1 (GFDL)

Los autores de esta traducción son:

- Igor Támara: ikks@bigfoot.com (mailto:ikksATbigfootDOTcom)
- Pablo Reyes: reyes_pablo@hotmail.com (mailto:reyes_pabloAThotmailDOTcom)

Esta es una traducción no oficial de la GNU Free Document License (GFDL), versión 1.1 a Español. No ha sido publicada por la Free Software Foundation, y no establece legalmente los términos de distribución para trabajos que usen la GFDL -- sólo el texto de la versión original en Inglés de la GFDL lo hace. Sin embargo, esperamos que esta traducción ayudaré a hablantes de español a entender mejor la GFDL.

This is an unofficial translation of the GNU Free Document License (GFDL) into Spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms for works that uses the GFDL --only the original English text of the GFDL does that. However, we hope that this translation will help Spanish speakers understand the GFDL better.

La versión original de la GFDL esta disponible en: http://www.gnu.org/copyleft/fdl.html

0. PREÁMBULO

El propósito de esta licencia es permitir que un manual, libro de texto, u otro documento escrito sea "libre" en el sentido de libertad: asegurar a todo el mundo la libertad efectiva de copiarlo y redistribuirlo, con o sin modificaciones, de manera comercial o no. En segundo término, esta licencia preserva para el autor o para quien publica una manera de obtener reconocimiento por su trabajo, al tiempo que no se considera responsable de las modificaciones realizadas por terceros, lo cual significa que los trabajos derivados del documento deben a su vez ser libres en el mismo sentido. Complementa la Licencia Pública General GNU, que es una licencia de copyleft diseñada para el software libre.

Hemos diseñado esta Licencia para usarla en manuales de software libre, ya que el software libre necesita documentación libre: Un programa libre debe venir con los manuales que ofrezcan la mismas libertades que da el software. Pero esta licencia no se limita a manuales de software; puede ser usada para cualquier trabajo textual, sin tener en cuenta su temática o si se publica como libro impreso. Recomendamos esta licencia principalmente para trabajos cuyo fin sea instructivo o de referencia.

1. APLICABILIDAD Y DEFINICIONES

Esta Licencia se aplica a cualquier manual u otro documento que contenga una nota del propietario de los derechos que indique que puede ser distribuido bajo los términos de la Licencia. El "Documento", en adelante, se refiere a culquiera de dichos manuales o trabajos. Cualquier miembro del público es un como "Usted".

Una "Versión Modificada" del Documento significa cualquier trabajo que contenga el Documento o una porción del mismo, ya sea una copia literal o con modificaciones y/o traducciones a otro idioma.

Una "Sección Secundaria" es un apéndice titulado o una sección preliminar al prólogo del Documento que tiene que ver exclusivamente con la relación de quien publica, o con los autores del Documento, o con el tema general del Documento (o asuntos relacionados) y cuyo contenido no entre directamente en tal tema general. (Por ejemplo, si el Documento es en parte un texto de matemáticas, una Sección Secundaria puede no explicar matemáticas). La relación puede ser un asunto de conexión histórica, o de posición egal, comercial, filosófica, ética o política con el tema o la materia del texto.

Las "Secciones Invariantes" son ciertas Secciones Secundarias cuyos títulos son denominados como Secciones Invariantes en la nota que indica que el documento es liberado bajo esta licencia.

Los "Textos de Cubierta" son ciertos pasajes cortos de texto que se listan, como Textos de Título o Textos al respaldo de la página de tí, en la nota que indica que el documento es liberado bajo esta Licencia.

Una copia "Transparente" del Documento, significa una copia para lectura en máquina, representada en un formato cuya especificación está disponible al público general, cuyos contenidos pueden ser vistos y editados directamente con editores de texto genéricos o (para imágenes compuestas por pixeles) con programas genéricos de dibujo o (para dibujos) con algún editor gráfico ampliamente disponible, y que sea adecuado para exportar a formateadores de texto o para traducción automática a una variedad de formatos adecuados para ingresar a formateadores de texto. Una copia hecha en un formato de un archivo que no sea Transparente, cuyo formato ha sido diseñado para impedir o dificultar subsecuentes modificaciones posteriores por parte de los lectores no es Transparente. Una copia que no es "Transparente" es llamada "Opaca".

Como ejemplos de formatos adecuados para copias Transparentes están el ASCII plano sin formato, formato de Texinfo, formato de LaTeX, SGML o XML usando un DTD disponible publicamente, y HTML simple que siga los estándares, diseñado para modificaciones humanas. Los formatos Opacos incluyen PostScript, PDF, formatos propietarios que pueden ser leídos y editados unicamente en procesadores de palabras propietarios, SGML o XML para los cuáles los DTD y/o herramientas de procesamiento no están disponibles generalmente, y el HTML generado por máquinas, producto de algún procesador de palabras solo para propósitos de salida.

La "Pagina de Título" en un libro impreso significa, la pagina de titulo misma, más las páginas siguientes que sean necesaria para mantener, legiblemente, el material que esta Licencia requiere en la página de titulo. Para trabajos en formatos que no tienen página de título como tal, "Página de Título" significa el texto cercano a la aparición más prominente del título del trabajo, precediendo el comienzo del cuerpo del trabajo.

2. COPIA LITERAL

Puede copiar y distribuir el Documento en cualquier medio, sea en forma comercial o no, siempre y cuando esta Licencia, las notas de derecho de autor, y la nota de licencia que indica que esta Licencia se aplica al Documento se reproduzca en todas las copias, y que usted no adicione ninguna otra condición a las expuestas en en esta Licencia. Usted no puede usar medidas técnicas para obstruir o controlar la lectura o copia posterior de las copias que usted haga o distribuya. Sin embargo, usted puede aceptar compensación a cambio de las copias. Si distribuye un número suficientemente grande de copias también deberá seguir las condiciones de la sección 3.

También puede prestar copias, bajo las mismas condiciones establecidas anteriormente, y puede exhibir copias publicamente.

3. COPIADO EN CANTIDAD

Si publica copias impresas del Documento que sobrepasen las 100, y la nota de Licencia del Documento exige Textos de Cubierta, debe incluir las copias con cubiertas que lleven en forma clara y legible, todos esos Textos de Cubierta: Textos de título en la portada, y Textos al respaldo de la página de tílo. Ambas cubiertas deben identificarlo a Usted clara y legiblemente como quien publica tales copias. La portada debe mostrar el título completo con todas las palabras igualmente prominentes y visibles. Además puede adicionar otro material en la cubierta. Las copias con cambios limitados en las cubiertas, siempre que preserven el título del Documento y satisfagan estas condiciones, pueden considerarse como copias literales.

Si los textos requeridos para la cubierta son muy voluminosos para que ajusten legiblemente, debe colocar los primeros (tantos como sea razonable colocar) en la cubierta real y continuar el resto en páginas adyacentes.

Si publica o distribuye copias Opacas del Documento cuya cantidad exceda las 100, debe incluir una copia Transparente, que pueda ser leída por una máquina, con cada copia Opaca o entregar en o con cada copia Opaca una dirección en una red de computadores accesible publicamente que contenga una copia completa Transparente del Documento, sin material adicional, a la cual el público en general de la red pueda acceder a bajar anónimamente sin cargo usando protocolos públicos y estandarizados. Si usted hace uso de la última opción, deberá tomar medidas necesarias, cuando comience la distribución de las copias Opacas en cantidad, para asegurar que esta copia Transparente permanecerá accesible en el sitio por lo menos un año después de su última distribución de copias Opacas (directamente o a través de sus agentes o distribuidores) de esa edición al público.

Se solicita, aunque no es requisito, que contacte a los autores del Documento antes de redistribuir cualquier gran número de copias, para darle la oportunidad de que le provean una versión actualizada del Documento.

4. MODIFICACIONES

Puede copiar y distribuir una Versión Modificada del Documento bajo las condiciones de las seccions 2 y 3 anteriores, siempre que usted libere la Versión Modificada bajo esta misma Licencia, con la Versión Modificada haciendo el rol del Documento, por lo tanto licenciando la distribución y modificación de la Versión Modificada a quienquiera posea una copia de esta. Además, debe hacer lo siguiente en la Versión Modificada:

A. Usar en la Página de Título (y en las cubiertas, si hay alguna) un título distinto al del Documento, y de versiones anteriores (que deberían, si hay alguna, estar listados en la sección de Historia del Documento). Puede usar el mismo título de versiones anteriores al original siempre y cuando quien publicó originalmente otorge permiso.
B. Listar en la Página de Título, como autores, una o más personas o entidades responsables por la autoría o las modificaciones en la Versión Modificada, junto con por lo menos cinco de los autores principales del Documento (Todos sus autores principales, si hay menos de cinco).
C. Incluir en la Página de Título el nombre de quién publica la Versión Modificada, como quien publica.
D. Preservar todas las notas de derechos de autor del Documento.
E. Adicionar una nota de derecho de autor apropiada a sus modificaciones, adyacente a las otras notas de derecho de autor.
F. Incluir, immediatamente después de la nota de derecho de autor, una nota de licencia dando el permiso público para usar la Versión Modificada bajo los términos de esta Licencia, de la forma mostrada en el anexo al final de este documento.
G. Preservar en esa nota de licencia el listado completo de Secciones Invariantes y de los Textos de Cubiertas que sean requeridos como se especifique en la nota de Licencia del Documento.
H. Incluir una copia sin modificación de esta Licencia.
I. Preservar la sección llamada "Historia", y su título, y adicionar a esta una sección estableciendo al menos el título, el año,los nuevos autores, y quién publique la Versión Modificada como reza en la Página de Título. Si no hay una sección titulada "Historia" en el Documento, crear una estableciendo el título, el año, los autores y quieng publicó el Documento como reza en la Página de Título, añadiendo además una sección describiendo la Versión Modificada como se estableció en la oración anterior.

J. Preservar la localización en red, si hay , dada en la Documentación para acceso público a una copia Transparente del Documento, tanto como las otras direcciones de red dadas en el Documento para versiones anteriores en las cuáles estuviese basado. Estas pueden ubicarse en la sección "Historia". Se puede omitir la ubicación en red para un trabajo que sea publicado por lo menos cuatro años antes que el Documento mismo, o si quien publica originalmente la versión da permiso explícitamente.

K. En cualquier sección titulada "Agradecimientos" o "Dedicatorias", preservar el título de la sección, y preservar en la sección toda la sustancia y el tono de los agradeimientos y/o dedicatorias de cada contribuyente que estén incluídas.

L. Preservar todas las Secciones Invariantes del Documento, sin alterar su texto ni sus títulos. Números de sección o el equivalente no son considerados parte de los títulos de la sección.

M. Borrar cualquier sección titulada "Aprobaciones". Tales secciones no pueden estar incluidas en las Versiones Modificadas.

N. No retitular ninguna sección existente como "Aprobaciones" o conflictuar con el título de alguna Sección Invariante.

Si la Versión Modificada incluye secciones o apendices nuevos o preliminares al prólogo que califiquen como Secciones Secundarias y contienen material no copiado del Documento, puede opcionalmente designar algunas o todas esas secciones como invariantes. Para hacerlo, adicione sus títulos a la lista de Secciones Invariantes en la nota de licencia de la Versión Modificada. Tales títulos deben ser distintos de cualquier otro título de sección.

Puede adicionar una sección titulada "Aprobaciones", siempre que contenga unicamente aprobaciones de su Versión Modificada por varias fuentes--por ejemplo, observaciones de peritos o que el texto ha sido aprobado por una organización como la definición oficial de un estándar.

Puede adicionar un pasaje de hasta cinco palabras como un Texto de Portada, y un pasaje de hasta 25 palabras al respaldo de la página de título al final de la lista de Textos de Cubierta en la Versión Modificada.

Solamente un pasaje de Texto de Portada y uno al respaldo de la página de título puede ser adicionado por (o a manera de arreglos hechos por) una entidad. Si el Documento ya incluye un texto de cubierta para la misma portada o al respaldo de la portada, previamente adicionado por usted o por arreglo hecho por la misma entidad, a nombre de la cual usted está actuando, usted no puede adicionar otro; pero puede reemplazar el anterior, con permiso explícito de quien publicó previamente y agregó el texto anterior

El(los) autor(es) y quien(es) publica(n) el Documento no dan con esta Licencia permiso para usar sus nombres para publicidad o para asegurar o implicar aprobación de cualquier Versión Modificada.

5. COMBINANDO DOCUMENTOS

Puede combinar el Documento con otros documentos liberados bajo esta Licencia, bajo los términos definidos en la sección 4 anterior para versiones modificadas, siempre que incluya en la combinación todas las Secciones Invariantes de todos los documentos originales, sin modificar, y listadas todas como Secciones Invariantes del trabajo combinado en su nota de licencia.

El trabajo combinado necesita contener solamente una copia de esta Licencia, y múltiples Seccions Invariantes idénticas pueden ser reemplazadas por una sola copia. Si hay múltiples Secciones Invariantes con el mismo nombre pero con contenidos diferentes, haga el título de cada una de estas secciones único adicionándole al final del mismo, entre paréntesis, el nombre del autor o de quien publicó originalmente esa sección, si es conocido, o si no, un número único. Haga el mismo ajuste a los títulos de sección en la lista de Secciones Invariantes en la nota de licencia del trabajo combinado.

En la combinación, debe combinar cualquier sección titulada "Historia" de los varios documentos originales, formando una sección titulada "Historia"; de la misma forma combine cualquier seción titulada "Agradecimientos", y cualquier sección titulada "Dedicatorias". Debe borrar todas las secciones tituladas "Aprobaciones."

6. COLECCIONES DE DOCUMENTOS

Puede hacer una colección consistente del Documento y otros documentos liberados bajo esta Licencia, y reemplazar las copias individuales de esta Licencia en los varios documentos con una sola copia que esté incluida en la colección, siempre que siga las reglas de esta Licencia para cada copia literal de cada uno de los documentos en cualquiera de los demás aspectos.

Puede extraer un solo documento de una de tales colecciones, y distribuirlo individualmente bajo esta Licencia, siempre que inserte una copia de esta Licencia en el documento extraido, y siga esta Licencia en todos los otros aspectos concernientes a la copia literal de tal documento.

7. AGREGACIÓN CON TRABAJOS INDEPENDENTES

Una recopilación del Documento o de sus derivados con otros documentos o trabajos separados o independientes, en cualquier tipo de distribución o medio de almacenamiento, no como un todo, cuenta como una Versión Modificada del Documento, siempre que no se aleguen derechos de autor por la recopilación. Tal recopilación es llamada un "agregado", y esta Licencia no aplica a los otros trabajos auto-contenidos así recopilados con el Documento, o a cuenta de haber sido recopilados, si no son ellos mismos trabajos derivados del Documento.

Si el requerimiento de la sección 3 sobre el Texto de Cubierta es aplicable a estas copias del Documento, entonces si el Documento es menor que un cuarto del agregado entero, los Textos de Cubierta del Documento pueden ser colocados en cubiertas que enmarquen solamente el Documento entre el agregado. De otra forma deben aparecer en cubiertas enmarcando todo el agregado.

8. TRADUCCIÓN

La Traducción es considerada como una clase de modificación, Así que puede distribuir traducciones del Documento bajo los términos de la sección 4. Reemplazar las Secciones Invariantes con traducciones requiere permiso especial de los dueños de derecho de autor, pero usted puede incluir traducciones de algunas o todas las Secciones Invariantes adicionalmente a las versiones originales de tales Secciones Invariantes. Puede incluir una traducción de esta Licencia siempre que incluya también la versión en Inglés de esta Licencia. En caso de un desacuerdo entre la traducción y la versión original en Inglés de esta Licencia, la versión original en Inglés prevalecerá.

9. TERMINACIÓN

No se puede copiar, modificar, sublicenciar, o distribuir el Documento excepto por lo permitido expresamente bajo esta Licencia. Cualquier otro intento de copia, modificación, sublicenciamiento o distribución del Documento es nulo, y serán automáticamente terminados sus derechos bajo esa licencia. Sin embargo, los terceros que hayan recibido copias, o derechos, de su parte bajo esta Licencia no tendrán por terminadas sus licencias siempre que tales personas o entidades se encuentren en total conformidad con la licencia original.

10. REVISIONES FUTURAS DE ESTA LICENCIA

La Free Software Foundation puede publicar versiones nuevas y revisadas de la Licencia de Documentación Libre GNU de tiempo en tiempo. Tales nuevas versiones serán similares en espíritu a la presente versión, pero pueden diferir en detalles para solucionar nuevos problemas o intereses. Vea http://www.gnu.org/copyleft/.

Cada versión de la Licencia tiene un número de versión que la distingue. Si el Documento especifica que se aplica una versión numerada en particular de esta licencia o "cualquier versión posterior", usted tiene la opción de seguir los términos y condiciones de la versión especificada o cualquiera posterior que haya sido publicada (no como un borrador) por la Free Software Foundation. Si el Documento no especifica un número de versión de esta Licencia, puede escoger cualquier versión que haya sido publicada(no como un borrador) por la Free Software Foundation.

Cómo usar esta Licencia para sus documentos

Para usar esta licencia en un documento que usted haya escrito, incluya una copia de la Licencia en el documento y ponga el siguiente derecho de autor y nota de licencia justo después de la página de título:

```
Derecho de Autor (c) AÑO SU NOMBRE.
Se otorga permiso para copiar, distribuir y/o modificar este documento
bajo los términos de la Licencia de Documentación Libre GNU, Versión 1.1
o cualquier otra versión posterior publicada por la Free Software
Foundation; con las Secciones Invariantes siendo LISTE SUS TÍTULOS, con
los Textos de Portada siendo LISTELO, y con los Textos al respaldo de la
página de título siendo LISTELOS. Una copia de la licencia
es incluida en la sección titulada "Licencia de Documentación Libre GNU".
```

Si no tiene Secciones Invariantes, escriba "Sin Secciones Invariantes" en vez de decir cuáles son invariantes. Si no tiene Texto de Portada, escriba "Sin Texto de Portada" en vez de "con los Textos de Portada siendo LISTELO"; e igualmente para los Textos al respaldo de la página de título.

Si su documento contiene ejemplos de código de programa no triviales, recomendamos liberar estos ejemplos en paralelo bajo su elección de licencia de software libre, tal como la Licencia Pública General GNU (**GNU General Public License**), para permitir su uso en software libre.

Notas de la traducción

1. *copyleft*: es un nuevo término acuñado por GNU. Nace de un juego de palabras en Inglés, en vez de "copyright" usan "copyleft", indicando que no se restringe la copia, sino por el contrario se permite mientras que el derecho a seguir copiando no se restrinja. http://www.gnu.org/copyleft/copyleft.es.html

2. *licenciatario*: a quien se le otorga la licencia.

Traducción:

- Igor Támara: ikks@bigfoot.com (mailto:ikksATbigfootDOTcom)
- Pablo Reyes: reyes_pablo@hotmail.com (mailto:reyes_pabloAThotmailDOTcom)

Revisión 1 (25.May.2000): Vladimir Támara Patiño vtamara@gnu.org (mailto:vtamaraATgnuDOTorg)

Appendix H. GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not

"Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover

Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix I. Licencia Pública General GNU

Versión 2, Junio de 1991

La autora de esta traducción es:

- Palmira Granados (mailto:palmisATlinuxDOTajuscoDOTupnDOTmx): Licenciada en Derecho por la Escuela libre de derecho (Mexico).

NOTA IMPORTANTE:

Esta es una traducción no oficial al español de la GNU General Public License. No ha sido publicada por la Free Software Foundation, y no establece legalmente las condiciones de distribución para el software que usa la GNU GPL. Estas condiciones se establecen solamente por el texto original, en inglés, de la GNU GPL. Sin embargo, esperamos que esta traducción ayude a los hispanoparlantes a entender mejor la GNU GPL.

IMPORTANT NOTICE:

This is an unofficial translation of the GNU General Public License into Spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL -- only the original English text of the GNU GPL does that. However, we hope that this translation will help Spanish speakers understand the GNU GPL better.

0. Preámbulo

Las licencias de la mayoría de los programas de cómputo están diseñadas para coartar la libertad de compartirlos y cambiarlos. Por el contrario, la Licencia Pública General GNU pretende garantizar esa libertad de compartir y cambiar Software Libre a fin de asegurar que el software sea libre para todos sus usuarios. Esta Licencia Pública General se aplica a la mayor parte del software de la Free Software Foundation y a cualquier otro programa cuyos autores se comprometan a usarla. (Algunos otros paquetes de software de la Free Software Foundation están protegidos bajo la Licencia Pública General de Librería GNU.) Esta última licencia también puede aplicarse a nuevos paquetes de software.

Cuando se hable de Software Libre, se hace referencia a libertad, no a precio. Las Licencias Públicas Generales GNU están diseñadas para asegurar que el usuario tenga libertad de distribuir copias de Software Libre (y de recibir una remuneración por este servicio, si así se desea), que ese mismo usuario reciba el código fuente o que tenga la posibilidad de recibirlo, si así lo desea, que pueda cambiar o modificar el software o utilice sólo partes del mismo en nuevos paquetes de Software Libre; y que dicho usuario tenga pleno conocimiento de estas facultades.

Con la finalidad de proteger los derechos antes mencionados, es necesario establecer restricciones que prohíban a cualquiera negar esos derechos o pedir la renuncia a los mismos. Estas restricciones se traducen en ciertas responsabilidades para el usuario que distribuye o modifica copias de software protegido bajo estas licencias.

Por ejemplo, si una persona distribuye copias de un paquete de Software Libre protegido bajo esta licencia, ya sea de manera gratuita o a cambio de una contraprestación, esa persona debe dar a los receptores de esa distribución todos y cada uno de los derechos que él o ella misma tenga. Asimismo, esa persona debe asegurarse que dichos receptores reciban o tengan la posibilidad de recibir el código fuente. De igual manera, debe mostrarles esta licencia a fin de que tengan conocimiento de los derechos de los que son titulares.

La protección que otorga la presente licencia se hace de dos maneras simultáneas: (1) se otorga protección al software bajo la ley de copyright, y (2) se ofrece la protección bajo esta licencia, la cual otorga permiso legal para copiar, distribuir y/o modificar el software.

Asimismo, a fin de proteger a cada uno de los autores y a los creadores mismos de esta licencia, es importante hacer notar y que todos entiendan que no existe ninguna garantía de cualquier paquete de Software Libre por la cual se deba responder. Esto es, si el software es modificado por alguna persona distinta del autor y distribuido con esas modificaciones, los receptores de esa distribución deben saber que lo que han recibido no es la obra original, y que por lo tanto, cualquier problema surgido de las modificaciones no se reflejará en la reputación del autor original.

Finalmente, cualquier programa de Software Libre es amenazado por patentes de Software. Esta licencia tiene la finalidad de evitar el peligro que representa que los redistribuidores de programas de Software Libre obtengan individualmente licencias de patentes, haciendo de esta forma, programas de Software Propietario. Para lograr esto, queda totalmente claro que cualquier patente debe otorgar licencias que permitan el uso libre del programa para todos o no otorgar licencia alguna.

Los términos y condiciones específicos para copiar, distribuir o modificar son los siguientes:

1. TÉRMINOS Y CONDICIONES PARA LA COPIA, DISTRIBUCIÓN Y MODIFICACIÓN

1. Esta licencia se aplica a cualquier programa u otra obra que contenga un aviso puesto por el titular de los derechos de autor en el que se establezca que el mismo puede ser distribuido bajo los

términos de esta Licencia Pública General. El Programa se refiere a cualquier programa u obra, y Obra basada en el Programa se refiere por su parte, a, ya sea al Programa mismo a cualquier obra derivada del mismo según la ley de Derechos de Autor; esto es, una obra que contenga el Programa o una porción del mismo, ya sea que esta porción sea exactamente igual o modificada y/o traducida a otro idioma. (En adelante, una traducción se considerará de manera enunciativa, mas no limitativa, como una modificación.)

Actividades distintas de copiar o distribuir no son abarcadas por esta licencia; están fuera de su alcance. El acto de correr el Programa no está restringido, y el producto que resulte del Programa está protegido sólo si su contenido constituye una obra basada en el Programa (independientemente de haber sido creado por el Programa que corre.) El que esto ocurra de esa manera depende de lo que el Programa haga.

2. Está permitido copiar y distribuir por cualquier medio copias fieles del código fuente del Programa tal y como fue recibido, siempre y cuando se publique en cada copia, de manera conspicua y apropiada, el aviso apropiado de derechos de autor y la renuncia a responder por la garantía correspondiente al Programa, se mantengan intactos los avisos referentes a esta licencia y a la respectiva ausencia de cualquier garantía; y se entregue a los receptores del Programa una copia de esta licencia.

   Exigir una remuneración por el acto físico de transferir una copia está permitido; asimismo, también está permitido ofrecer una garantía a cambio de una contraprestación.

3. Está permitido modificar la copia o copias del Programa o cualquier parte del mismo, creando de esta forma, una Obra basada en el Programa. Asimismo, está permitido copiar y distribuir las modificaciones antes mencionadas o la obra misma bajo los términos de la Sección 1 mencionada anteriormente, y siempre y cuando se cumplan de igual manera las condiciones siguientes:

   1. Colocación de avisos, en la obra misma y por parte de quien realiza las modificaciones, en los que se informe que los

archivos fueron modificados y la fecha de esas modificaciones.

2. Otorgamiento de una licencia bajo los términos establecidos en esta Licencia Pública General que abarque la obra en su totalidad y sin cargo a terceras personas para el caso en el que se distribuya o publique una obra que contenga todo o parte del Programa o que constituya una obra derivada del mismo.
3. Si el programa modificado normalmente lee comandos de manera interactiva cuando corre, cuando empiece a correr con dicho propósito interactivo, es necesario que aparezca un aviso que incluya la leyenda de derechos de autor correspondiente, así como la ausencia de responsabilidad por la garantía. Asimismo, dicho aviso deberá establecer que los usuarios de dicho programa tienen autorización para redistribuirlo bajo las mismas condiciones en las que les fue distribuido y les deberá informar cómo podrán tener acceso a una copia de esta licencia. (La excepción a esta condición tiene lugar cuando se trata de una Obra basada en un Programa que es en sí mismo interactivo, pero no envía normalmente un aviso.)

Las condiciones antes mencionadas se aplican a las obras modificadas como un todo. En el caso en el que las secciones de dicha obra que no se deriven del Programa sean identificables y razonablemente independientes y puedan separarse entre ellas, esta licencia y sus términos no se aplicarán a dichas secciones cuando éstas sean distribuidas como obras separadas. Sin embargo, cuando esas mismas secciones se distribuyan como parte de la Obra basada en el Programa, dicha distribución deberá hacerse de acuerdo a los términos de esta licencia, cuyas autorizaciones para otros licenciatarios tendrán los mismos alcances, sin importar qué parte creó quién.

Por medio de esta sección no se pretende exigir derechos o impugnar los derechos originados de una obra creada en su totalidad por otra persona, sino más bien se tiene como finalidad ejercer el derecho de controlar la distribución de obras derivadas o colectivas basadas en el Programa.

Asimismo, la sola inclusión de otra obra que no se base en el

Programa aunada al Programa (o a una Obra basada en el Programa) dentro de un medio de almacenamiento o distribución no provoca que dicha obra deba regirse por esta licencia.

4. Copiar y distribuir el Programa (o una Obra basada en el Programa de acuerdo a la sección 2), bajo los términos de las secciones 1 y 2 mencionadas anteriormente, ya sea en código objeto o en su forma ejecutable está permitido, siempre y cuando dicho Programa se acompañe también por cualquiera de los siguientes:

   A. El código fuente respectivo completo y leíble por una máquina, el cual debe ser distribuido bajo los términos establecidos en las secciones 1 y 2 mencionadas anteriormente y a través de un medio normalmente usado para el intercambio de software;

   B. Una oferta por escrito y con una validez mínima de tres años, de proporcionar a cualquier tercera persona, por una cuota que no exceda el costo del acto físico de distribuir, bajo los términos de las secciones 1 y 2 antes mencionadas; y a través de un medio normalmente usado para el intercambio de software; una copia del respectivo código fuente completo y leíble por una máquina; o,

   C. Toda la información recibida respecto a la oferta de distribución del código fuente correspondiente. (Esta alternativa está permitida únicamente para distribuciones no comerciales y siempre y cuando el Programa se haya recibido en código objeto o en forma ejecutable junto con esta oferta de acuerdo a la subsección b antes mencionada.)

   El código fuente de una obra se refiere a la forma preferida para hacerle modificaciones. En una obra ejecutable, el código fuente completo se refiere a todo el código fuente de todos los módulos que contiene, además de cualquier archivo de definición de interfaz asociado y de los scripts utilizados para controlar la compilación e instalación del ejecutable. Sin embargo, como una excepción especial, el código fuente distribuido no debe incluir cualquier cosa que sea normalmente distribuida (ya sea en forma de binarios o de código fuente) con los principales componentes del sistema operativo (como compilador, kernel, etc.) sobre el cual el ejecutable corre, a menos que el mismo componente acompañe al ejecutable.

Si la distribución del ejecutable o del código objeto se lleva a cabo mediante el ofrecimiento de acceso a una copia en un lugar designado, el ofrecimiento de acceso al código fuente en el mismo lugar equivale a la distribución de dicho código fuente, aun cuando terceras personas no estén obligadas a copiar el código fuente junto con el código objeto.

5. El Programa no puede copiarse, modificarse, sublicenciarse ni distribuirse a menos que se haga bajo los términos y condiciones de esta licencia. Cualquier intento por hacer lo anterior de otra forma, será nulo y extinguirá automáticamente los derechos surgidos de esta licencia. Sin embargo, las licencias de las personas que hayan recibido copias o derechos bajo esta licencia, seguirán vigentes mientras dichas personas cumplan con sus obligaciones.

6. Mientras no se firme la presente licencia no existe obligación de aceptarla. Sin embargo, no existe autorización, y por lo tanto está legalmente prohibido, modificar o distribuir el Programa o una Obra basada en el Programa a menos que se acepten los términos y condiciones de la presente licencia. Por lo anterior, del acto de modificar o distribuir el Programa o una Obra basada en el Programa se presume la aceptación de los términos y condiciones de la presente licencia para copiar, distribuir o modificar dicho Programa u Obra basada en el Programa.

7. Cada vez que se distribuya el Programa (o cualquier Obra basada en el Programa), quien recibe la copia del mismo recibe también, de manera automática una licencia de parte del licenciante original para copiar, distribuir o modificar el Programa bajo los términos y condiciones de esta licencia. No podrán imponerse más restricciones al ejercicio de los derechos del licenciatario que los establecidos en esta licencia. Quien distribuye el Programa no es responsable por el cumplimiento de la presente licencia por parte de terceras personas.

8. En el caso en el que como consecuencia de orden judicial o de las pretensiones demandadas por violación a una patente o por

cualquier otra razón (de manera enunciativa, mas no limitativa) se imponen condiciones (ya sea por orden judicial, contrato o por otro medio) que se contradicen con las condiciones de esta licencia, estas últimas no se eximen de su cumplimiento. Como consecuencia de la imposibilidad de cumplir con ambas obligaciones mencionadas, el Programa no podrá distribuirse. Por ejemplo, si una licencia de una patente prohíbe la redistribución gratuita del Programa por parte de quienes reciben copias del mismo de manera directa o indirecta, entonces la única forma de cumplir con ambas licencias, ésta y la de la patente, será abstenerse de distribuir el Programa.

En el caso en el que cualquier parte de esta sección sea declarada inválida o inexigible bajo cualquier circunstancia particular, el resto de la misma continuará surtiendo sus efectos para esa circunstancia, al igual que la sección en su totalidad para las demás circunstancias.

El propósito de esta sección no es inducir a la violación de patentes o del ejercicio de otros derechos intelectuales, como tampoco impugnar la validez de tales demandas por incumplimiento, sino mas bien, pretende proteger la integridad del sistema de distribución del Software Libre, el cual consiste en la práctica y uso de licencias públicas. Mucha gente ha hecho generosas contribuciones a paquetes de software distribuidos bajo este sistema confiando en la aplicación de dicho sistema; y es decisión del autor/donante distribuir el software a través de cualquier otro sistema sin que un licenciatario pueda interferir en esa decisión.

Esta sección pretende aclarar todo aquello que se considera consecuencia del resto de esta licencia.

En el caso en el que la distribución y/o uso del Programa esté restringida en ciertos países, ya sea por patentes o interfases protegidas por el sistema de propiedad intelectual, el titular original de los derechos de autor del Programa que lo sujeta a esta licencia tiene la facultad de agregar una limitación de tipo geográfico a la distribución, por virtud de la cual se excluya a dichos países; de manera que la distribución del mismo se permita únicamente en los países no excluidos. En este caso, dicha limitación se tiene como parte integrante de esta licencia.

9. Es facultad de la Free Software Foundation publicar, en cualquier momento, tanto versiones revisadas como versiones de reciente creación, de la Licencia Pública General. Las versiones nuevas pueden diferir en detalles a fin de afrontar y resolver nuevos problemas o preocupaciones, pero conservando siempre el espíritu de la presente versión.

   Cada versión tendrá asignado un número. En el caso en el que el Programa especifique un número de versión de esta licencia para su aplicación y además, incluya la frase y cualquier versión posterior, el licenciatario podrá sujetarse, a su elección, a los términos y condiciones de la versión expresamente mencionada o de cualquiera de las versiones posteriores de la misma publicadas por la Free Software Foundation. Por otro lado, en el caso en el que el programa no especifique un número de versión de licencia, el licenciatario podrá elegir cualquier versión que haya sido publicada por la Free Software Foundation.

10. En el caso en el que se deseen incorporar partes del Programa a otros paquetes de Software Libre cuyas condiciones de distribución difieran a estas, es necesario solicitar permiso por escrito al autor. Cuando se trate de software cuyo titular de los de los derechos de autor correspondientes sea la Free Software Foundation, la solicitud de permiso deberá dirigirse a ésta última, quien en algunas ocasiones hace excepciones como esta. La decisión emitida por la Free Software Foundation se basará tomando en cuenta la finalidad de preservar el estatus libre de todos los derivados del Software Libre y de promocionar que se comparta y se reutilice el software en general.

11. EXCLUSIÓN DE GARANTÍA

    COMO CONSECUENCIA DE QUE EL PROGRAMA SE LICENCIE COMO GRATUITO, EN LA MEDIDA EN QUE LA LEY APLICABLE LO PERMITA, NO EXISTIRÁ GARANTÍA ALGUNA POR LA QUE SE DEBA RESPONDER. SALVO DISPOSICIÓN ESCRITA EN CONTRARIO, LOS TITULARES DE LOS DERECHOS DE AUTOR

RESPECTIVOS Y/U OTRAS PARTES PONEN A DISPOSICIÓN EL PROGRAMA SIN GARANTÍA DE NINGÚN TIPO, EXPRESA O IMPLÍCITA, INCLUYENDO DE MANERA ENUNCIATIVA MAS NO LIMITATIVA, LAS GARANTÍAS IMPLÍCITAS DE TIPO COMERCIAL U OTRAS INHERENTES A ALGÚN PROPÓSITO ESPECÍFICO. EL RIESGO DE QUE EL PROGRAMA ESTÉ EN PERFECTAS CONDICIONES Y FUNCIONE TAL Y COMO DEBE FUNCIONAR CORRE POR CUENTA DE QUIEN LO RECIBE, AL IGUAL QUE LOS GASTOS NECESARIOS PARA SU SERVICIO, REPARACIÓN O CORRECCIÓN EN EL DADO CASO EN EL QUE DICHO PROGRAMA CONTENGA DEFECTOS.

12. A MENOS QUE ASÍ LO DISPONGA LA LEY APLICABLE O EXISTA ACUERDO ESCRITO EN CONTRARIO, NINGÚN TITULAR DE LOS DERECHOS DE AUTOR O PERSONA FACULTADA, SEGÚN LAS SECCIONES ANTERIORES DE LA PRESENTE, PARA MODIFICAR Y/O DISTRIBUIR EL PROGRAMA SERÁ RESPONSABLE POR LOS DAÑOS YA SEAN GENERALES, ESPECIALES, INCIDENTALES O CONSECUENCIALES RESULTADO DEL USO O INCAPACIDAD DE USO DEL PROGRAMA (INCLUYENDO DE MANERA ENUNCIATIVA MAS NO LIMITATIVA LA PÉRDIDA DE INFORMACIÓN, INEXACTITUD EN LA INFORMACIÓN, PÉRDIDAS SUFRIDAS POR EL USUARIO DEL PROGRAMA O POR TERCERAS PERSONAS O LA INCAPACIDAD DEL PROGRAMA PARA OPERAR CON OTROS PROGRAMAS), AUN CUANDO DICHO TITULAR O CUALQUIER OTRA PERSONA HAYA ADVERTIDO DICHA POSIBILIDAD DE DAÑO.

FIN DE LOS TÉRMINOS Y CONDICIONES

2. Cómo aplicar estos términos a los nuevos programas

En el caso en el que se esté desarrollando un Programa nuevo y se tenga la intención de hacerlo de uso público, la mejor forma de lograrlo es haciéndolo Libre, y de esta forma, permitir a cualquiera redistribuirlo y cambiarlo bajo los términos y condiciones de esta Licencia.

A fin de lograr lo anterior, se deben incluir los siguientes avisos al Programa. Es más seguro incluir dichos avisos al principio de cada archivo de código fuente para aclarar de manera más eficiente la exclusión de garantía mencionada. Asimismo, cada archivo debe tener por lo menos la frase Derechos Reservados (Copyright para los países con ese sistema) relativa a los derechos de autor, así como la referencia al lugar donde se encuentre la leyenda y especificaciones completas de los mismos.

> <La referencia al nombre del Programa y a una idea de lo que el mismo hace.> Derechos Reservados (C) <año> <nombre del autor>
>
> Este es un Software Libre; como tal redistribuirlo y/o modificarlo está permitido, siempre y cuando se haga bajo los términos y condiciones de la Licencia Pública General GNU publicada por la Free Software Foundation, ya sea en su versión 2 ó cualquier otra de las posteriores a la misma.
>
> Este Programa se distribuye con la intención de que sea útil, sin embargo carece de garantía, ni siquiera tiene la garantía implícita de tipo comercial o inherente al propósito del mismo Programa. Ver la Licencia Pública General GNU para más detalles.
>
> Se debe haber recibido una copia de la Licencia Pública General GNU con este Programa, si este no fue el caso, favor de escribir a la Free Software Foundation, Inc., 59 Temple Place Suite 330, Boston, MA 02111-1307, USA.

Asimismo, se deben incluir las direcciones de correo electrónico y convencional del autor del Programa a fin de contactarlo.

En el caso en el que el Programa sea interactivo se debe incluir un aviso cuando inicie el modo interactivo como el siguiente:

> Gnomovision versión 69, Derechos Reservados © primer año de publicación y nombre del autor Gnomovision carece totalmente de garantía; para más detalles teclee show w. Este es Software Libre y está permitido redistribuirlo bajo ciertas condiciones; teclee show c para más detalles.

Los comandos hipotéticos show w y show c deberán desplegar las partes correspondientes de la Licencia Pública General. Obviamente, los comandos que se utilicen pueden ser distintos a los mencionados, inclusive pueden ser clicks del ratón o elementos del menú, según sea más conveniente.

Asimismo, el autor del Programa debe obtener de su empleador (en caso en que dicho autor trabaje como programador) o de su escuela, si ese es el caso, una renuncia firmada a los derechos de autor por el Programa, si es que fuera necesario. El siguiente es un ejemplo:

> Yoyodyne, Inc., por medio de la presente se renuncia a cualquier derecho de autor que corresponda al programa Gnomovision cuyo autor es James Hacker. <Firma Ty Coon>, 1 de Abril de 1989 Ty Coon, Presidente de Vice

Esta Licencia Pública General prohíbe incorporar el Programa a programas propietarios. En el caso en el que se trate de un programa que a su vez sea una librería de subrutinas, es más conveniente permitir ligas de aplicaciones propietarias con la librería. Si es esto lo que se desea, entonces se debe usar la Licencia Pública General Menor de GNU, en lugar de esta licencia.

Appendix J. GNU General Public License

Version 2, June 1991

0. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

1. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is

included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under

these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   A. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   B. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

C. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so,

and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

11. NO WARRANTY

    BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS"

WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

2. How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program’s name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker. <signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix K. Código fuente de los scripts de ejemplo

Script de ejemplo rc.firewall

```
#!/bin/sh
#
# rc.firewall - Script de cortafuegos con IP simple para Linux 2.4.x e iptables
#
# Copyright (C) 2001  Oskar Andreasson <bluefluxATkoffeinDOTnet>
#
# Este programa es software libre; puedes redistribuirlo y/o modificarlo
# bajo los términos expresados en la "GNU General Public License", tal como
# lo publica la Free Software Foundation; versión 2 de la Licencia.
#
# Este programa se distribuye con el deseo de que sea útil, pero
# SIN NINGUNA GARANTÍA; incluso sin garantía implícita de COMPRA-VENTA
# o ADECUACIÓN A PROPÓSITO PARTICULAR. Para más detalles, referirse a la
# GNU General Public License.
#
# Deberías haber recibido una copia de la GNU General Public License
# junto a este programa o desde el sitio web de dónde lo bajaste;
# si no es así, escribe a la Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA  02111-1307   USA
#
#
###########################################################################
#
# 1. Opciones de configuración.
#

#
# 1.1 Configuración de Internet.
#

INET_IP="194.236.50.155"
```

```
INET_IFACE="eth0"
INET_BROADCAST="194.236.50.255"

#
# 1.1.1 DHCP
#

#
# 1.1.2 PPPoE
#

#
# 1.2 Configuración de la red local.
#
# El rango IP de la LAN (red de área local) y la IP del host local. El valor
# "/24" significa que sólo se utilizarán los primeros 24 bits de los 32 bits
# que tiene una dirección IP. Es lo mismo que la máscara de red "255.255.255.0".
#

LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
LAN_BROADCAST_ADDRESS="192.168.255.255"
LAN_IFACE="eth1"

#
# 1.3 Configuración de la DMZ.
#

#
# 1.4 Configuración del host local.
#

LO_IFACE="lo"
LO_IP="127.0.0.1"

#
# 1.5 Configuración de IPTables.
#

IPTABLES="/usr/sbin/iptables"
```

```
#
# 1.6 Otras configuraciones.
#

###########################################################################
#
# 2. Carga de módulos.
#

#
# Necesario para la carga inicial de los módulos.
#

/sbin/depmod -a

#
# 2.1 Módulos requeridos.
#

/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state

#
# 2.2 Módulos no-requeridos.
#

#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ipt_MASQUERADE
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc

###########################################################################
```

```
#
# 3. Configuración de /proc.
#

#
# 3.1 Configuración requerida de proc.
#

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

#
# 3.2 Configuración no-requerida de proc.
#

#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr

###########################################################################
#
# 4. Configuración de las reglas.
#

######
# 4.1 Tabla Filter.
#

#
# 4.1.1 Establecimiento de políticas.
#

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

#
# 4.1.2 Creación de "cadenas de usuario."
#

#
# Crea una cadena para paquetes tcp incorrectos.
```

```
#

$IPTABLES -N bad_tcp_packets

#
# Crea cadenas separadas para que los paquetes ICMP, TCP y UDP las atraviesen.
#

$IPTABLES -N allowed
$IPTABLES -N tcp_packets
$IPTABLES -N udp_packets
$IPTABLES -N icmp_packets

#
# 4.1.3 Creación de contenido en las cadenas de usuario
#

#
# Cadena bad_tcp_packets
#

$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

#
# Cadena de "permitidos".
#

$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP

#
# Reglas TCP.
#

$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 22 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 80 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 113 -j allowed
```

```
#
# Puertos UDP.
#

#$IPTABLES -A udp_packets -p UDP -s 0/0 --destination-port 53 -j ACCEPT
#$IPTABLES -A udp_packets -p UDP -s 0/0 --destination-port 123 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --destination-port 2074 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --destination-port 4000 -j ACCEPT

#
# En Redes Microsoft te verás inundado por difusiones de paquetes
# (broadcasts). Con las siguientes líneas evitarás que aparezcan en
# los registros.
#

#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d $INET_BROADCAST \
#--destination-port 135:139 -j DROP

#
# Si recibimos peticiones DHCP desde el exterior de nuestra red, nuestros
# registros también resultarán inundados. Con esta regla evitaremos
# que queden registrados.
#

#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d 255.255.255.255 \
#--destination-port 67:68 -j DROP

#
# Reglas ICMP.
#

$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

#
# 4.1.4 Cadena INPUT.
#

#
# Paquetes TCP incorrectos que no queremos.
```

```
#

$IPTABLES -A INPUT -p tcp -j bad_tcp_packets

#
# Reglas para redes especiales que no son parte de Internet.
#

$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_BROADCAST_ADDRESS -
j ACCEPT

#
# Regla especial para peticiones DHCP desde la red local, que de otra manera
# no son correctamente gestionadas.
#

$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT

#
# Reglas para paquetes entrantes desde Internet.
#

$IPTABLES -A INPUT -p ALL -d $INET_IP -m state --state ESTABLISHED,RELATED \
-j ACCEPT
$IPTABLES -A INPUT -p TCP -i $INET_IFACE -j tcp_packets
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -j udp_packets
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_packets

#
# Si dispones de una Red Microsoft fuera de tu cortafuegos, también puedes
# verte inundado por multidifusiones (multicasts). Desechamos estos paquetes
# para no desbordar los registros.
#

#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP

#
```

```
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT packet died: "

#
# 4.1.5 Cadena FORWARD.
#

#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

#
# Acepta los paquetes que sí queremos reenviar.
#

$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet died: "

#
# 4.1.6 Cadena OUTPUT.
#

#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets

#
```

```
# Reglas especiales de la tabla OUTPUT para decidir qué IPs están permitidas.
#

$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

######
# 4.2 Tabla Nat.
#

#
# 4.2.1 Establece las políticas.
#

#
# 4.2.2 Crea cadenas definidas por el usuario.
#

#
# 4.2.3 Crea contenido en las cadenas de usuario.
#

#
# 4.2.4 Cadena PREROUTING.
#

#
# 4.2.5 Cadena POSTROUTING.
#

#
# Activación del Reenvío IP simple y la Traducción de Dirección de Red (NAT).
#
```

```
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP

#
# 4.2.6 Cadena OUTPUT.
#

######
# 4.3 Tabla Mangle.
#

#
# 4.3.1 Establece las políticas.
#

#
# 4.3.2 Crea cadenas definidas por el usuario.
#

#
# 4.3.3 Crea contenido en las cadenas de usuario.
#

#
# 4.3.4 Cadena PREROUTING.
#

#
# 4.3.5 Cadena INPUT.
#

#
# 4.3.6 Cadena FORWARD.
#

#
# 4.3.7 Cadena OUTPUT.
#

#
# 4.3.8 Cadena POSTROUTING.
```

```
#
```

Script de ejemplo rc.DMZ.firewall

```
#!/bin/sh
#
# rc.DMZ.firewall - Script de cortafuegos para la IP de la DMZ en
#                   Linux 2.4.x con iptables
#
# Copyright (C) 2001  Oskar Andreasson <bluefluxATkoffeinDOTnet>
#
# Este programa es software libre; puedes redistribuirlo y/o modificarlo
# bajo los términos expresados en la "GNU General Public License", tal como
# lo publica la Free Software Foundation; versión 2 de la Licencia.
#
# Este programa se distribuye con el deseo de que sea útil, pero
# SIN NINGUNA GARANTÍA; incluso sin garantía implícita de COMPRA-VENTA
# o ADECUACIÓN A PROPÓSITO PARTICULAR. Para más detalles, referirse a la
# GNU General Public License.
#
# Deberías haber recibido una copia de la GNU General Public License
# junto a este programa o desde el sitio web de dónde lo bajaste;
# si no es así, escribe a la Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA  02111-1307   USA
#
#
###############################################################################
#
# 1. Opciones de Configuración.
#

#
# 1.1 Configuración de Internet.
#
```

```
INET_IP="194.236.50.152"
HTTP_IP="194.236.50.153"
DNS_IP="194.236.50.154"
INET_IFACE="eth0"

#
# 1.1.1 DHCP
#

#
# 1.1.2 PPPoE
#

#
# 1.2 Configuración de la Red Local (Local Area Network).
#
# el rango de IPs de tu LAN y la IP de tu host. El valor \24 significa
# que sólo se usarán los primeros 24 de los 32 bits de una dirección IP.
# Es lo mismo que la máscara de red 255.255.255.0
#

LAN_IP="192.168.0.2"
LAN_IFACE="eth1"

#
# 1.3 Configuración DMZ.
#

DMZ_HTTP_IP="192.168.1.2"
DMZ_DNS_IP="192.168.1.3"
DMZ_IP="192.168.1.1"
DMZ_IFACE="eth2"

#
# 1.4 Configuración del host local.
#

LO_IFACE="lo"
LO_IP="127.0.0.1"

#
```

```
# 1.5 Configuración de IPTables.
#

IPTABLES="/usr/sbin/iptables"

#
# 1.6 Otra configuración.
#

###########################################################################
#
# 2. Carga de módulos.
#

#
# Necesario para la carga inicial de módulos.
#
/sbin/depmod -a

#
# 2.1 Módulos requeridos.
#

/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state

#
# 2.2 Módulos no-requeridos.
#

#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ipt_MASQUERADE
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
```

```
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc

############################################################################
#
# 3. Configuración de /proc.
#

#
# 3.1 Configuración requerida de proc.
#

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

#
# 3.2 Configuración no-requerida de proc.
#

#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr

############################################################################
#
# 4. Configuración de las reglas.
#

######
# 4.1 Tabla Filter.
#

#
# 4.1.1 Establecimiento de políticas.
#

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

#
# 4.1.2 Creación de cadenas de usuario.
```

```
#

#
# Creación de una cadena para paquetes TCP incorrectos.
#

$IPTABLES -N bad_tcp_packets

#
# Creación de cadenas separadas para los paquetes ICMP, TCP y UDP.
#

$IPTABLES -N allowed
$IPTABLES -N icmp_packets

#
# 4.1.3 Creación de contenido en las cadenas de usuario.
#

#
# Cadena bad_tcp_packets.
#

$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

#
# Cadena de "permitidos".
#

$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP

#
# Reglas ICMP.
#

# Changed rules totally
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
```

```
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

#
# 4.1.4 Cadena INPUT.
#

#
# Paquetes TCP incorrectos que no deseamos.
#

$IPTABLES -A INPUT -p tcp -j bad_tcp_packets

#
# Paquetes desde Internet hacia este equipo.
#

$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_packets

#
# Paquetes desde la red local (LAN), DMZ o el host local.
#

#
# Desde la interfaz DMZ hacia la IP del cortafuegos de la zona DMZ.
#

$IPTABLES -A INPUT -p ALL -i $DMZ_IFACE -d $DMZ_IP -j ACCEPT

#
# Desde la interfaz de la red local hacia la IP del cortafuegos de la red local.
#

$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_BROADCAST_ADDRESS -
j ACCEPT

#
# Desde la interfaz del host local hacia la IP del host local.
#

$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
```

```
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT

#
# Regla especial para peticiones DHCP desde la red local, que de otra forma
# no serían gestionadas correctamente.
#

$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT

#
# Todos los paquetes que han establecido una conexión y todos los que
# dependen de éllos, provenientes de Internet hacia el cortafuegos.
#

$IPTABLES -A INPUT -p ALL -d $INET_IP -m state --state ESTABLISHED,RELATED \
-j ACCEPT

#
# En Redes Microsoft te verás inundado por difusiones de paquetes
# (broadcasts). Con las siguientes líneas evitarás que aparezcan en
# los registros.
#

#$IPTABLES -A INPUT -p UDP -i $INET_IFACE -d $INET_BROADCAST \
#--destination-port 135:139 -j DROP

#
# Si recibimos peticiones DHCP desde el exterior de nuestra red, nuestros
# registros también resultarán inundados. Con esta regla evitaremos
# que queden registrados.
#

#$IPTABLES -A INPUT -p UDP -i $INET_IFACE -d 255.255.255.255 \
#--destination-port 67:68 -j DROP

#
# Si dispones de una Red Microsoft fuera de tu cortafuegos, también puedes
# verte inundado por multidifusiones (multicasts). Desechamos estos paquetes
# para no desbordar los registros.
#
```

```
#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT packet died: "

#
# 4.1.5 Cadena FORWARD.
#

#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

#
# Sección DMZ.
#
# Reglas generales.
#

$IPTABLES -A FORWARD -i $DMZ_IFACE -o $INET_IFACE -j ACCEPT
$IPTABLES -A FORWARD -i $INET_IFACE -o $DMZ_IFACE -m state \
--state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A FORWARD -i $LAN_IFACE -o $DMZ_IFACE -j ACCEPT
$IPTABLES -A FORWARD -i $DMZ_IFACE -o $LAN_IFACE -m state \
--state ESTABLISHED,RELATED -j ACCEPT

#
# Servidor HTTP.
#

$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_HTTP_IP \
--dport 80 -j allowed
$IPTABLES -A FORWARD -p ICMP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_HTTP_IP \
-j icmp_packets
```

```
#
# Servidor DNS.
#

$IPTABLES -A FORWARD -p TCP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP \
--dport 53 -j allowed
$IPTABLES -A FORWARD -p UDP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP \
--dport 53 -j ACCEPT
$IPTABLES -A FORWARD -p ICMP -i $INET_IFACE -o $DMZ_IFACE -d $DMZ_DNS_IP \
-j icmp_packets

#
# Sección LAN (red local).
#

$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet died: "

#
# 4.1.6 Cadena OUTPUT.
#

#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets

#
# Reglas especiales de la cadena OUTPUT para decidir qué direcciones IP
# están permitidas.
#
```

```
$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

######
# 4.2 Tabla Nat.
#

#
# 4.2.1 Establecimiento de políticas.
#

#
# 4.2.2 Creación de cadenas de usuario.
#

#
# 4.2.3 Creación de contenido en las cadenas de usuario.
#

#
# 4.2.4 Cadena PREROUTING.
#

$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $HTTP_IP --dport 80 \
-j DNAT --to-destination $DMZ_HTTP_IP
$IPTABLES -t nat -A PREROUTING -p TCP -i $INET_IFACE -d $DNS_IP --dport 53 \
-j DNAT --to-destination $DMZ_DNS_IP
$IPTABLES -t nat -A PREROUTING -p UDP -i $INET_IFACE -d $DNS_IP --dport 53 \
-j DNAT --to-destination $DMZ_DNS_IP

#
# 4.2.5 Cadena POSTROUTING.
#
```

```
#
# Activación del Reenvío IP simple y la Traducción de Dirección de Red (NAT).
#

$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP

#
# 4.2.6 Cadena OUTPUT.
#

######
# 4.3 Tabla Mangle.
#

#
# 4.3.1 Establecimiento de políticas.
#

#
# 4.3.2 Creación de cadenas de usuario.
#

#
# 4.3.3 Creación de contenido en las cadenas de usuario.
#

#
# 4.3.4 Cadena PREROUTING.
#

#
# 4.3.5 Cadena INPUT.
#

#
# 4.3.6 Cadena FORWARD.
#

#
# 4.3.7 Cadena OUTPUT.
```

```
#

#
# 4.3.8 Cadena POSTROUTING.
#
```

Script de ejemplo rc.UTIN.firewall

```
#!/bin/sh
#
# rc.firewall - Script para cortafuegos UTIN con Linux 2.4.x e iptables
#
# Copyright (C) 2001  Oskar Andreasson <bluefluxATkoffeinDOTnet>
#
# Este programa es software libre; puedes redistribuirlo y/o modificarlo
# bajo los términos expresados en la "GNU General Public License", tal como
# lo publica la Free Software Foundation; versión 2 de la Licencia.
#
# Este programa se distribuye con el deseo de que sea útil, pero
# SIN NINGUNA GARANTÍA; incluso sin garantía implícita de COMPRA-VENTA
# o ADECUACIÓN A PROPÓSITO PARTICULAR. Para más detalles, referirse a la
# GNU General Public License.
#
# Deberías haber recibido una copia de la GNU General Public License
# junto a este programa o desde el sitio web de dónde lo bajaste;
# si no es así, escribe a la Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA  02111-1307   USA
#
#
###########################################################################
#
# 1. Opciones de configuración.
#

#
```

```
# 1.1 Configuración de Internet.
#

INET_IP="194.236.50.155"
INET_IFACE="eth0"
INET_BROADCAST="194.236.50.255"

#
# 1.1.1 DHCP
#

#
# 1.1.2 PPPoE
#

#
# 1.2 Configuracion de la Red de Área Local.
#
# El rango IP de la LAN (red de área local) y la IP del host local. El valor
# "/24" significa que sólo se utilizarán los primeros 24 bits de los 32 bits
# que tiene una dirección IP. Es lo mismo que la máscara de red "255.255.255.0".
#

LAN_IP="192.168.0.2"
LAN_IP_RANGE="192.168.0.0/16"
LAN_IFACE="eth1"

#
# 1.3 Configuración de la DMZ ("Zona Desmilitarizada").
#

#
# 1.4 Configuración del host local.
#

LO_IFACE="lo"
LO_IP="127.0.0.1"

#
# 1.5 Configuración de IPTables.
#
```

```
IPTABLES="/usr/sbin/iptables"

#
# 1.6 Otras configuraciones.
#

###########################################################################
#
# 2. Carga de módulos.
#

#
# Necesario para la carga inicial de los módulos.
#

/sbin/depmod -a

#
# 2.1 Módulos requeridos.
#

/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_state

#
# 2.2 Módulos no-requeridos.
#

#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ipt_MASQUERADE
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
```

```
#/sbin/modprobe ip_nat_irc

###########################################################################
#
# 3. Configuración de /proc.
#

#
# 3.1 Configuración requerida de proc.
#

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

#
# 3.2 Configuración no-requerida de proc.
#

#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr

###########################################################################
#
# 4. Configuración de las reglas.
#

######
# 4.1 Tabla Filter.
#

#
# 4.1.1 Establecimiento de políticas.
#

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

#
# 4.1.2 Creación de "cadenas de usuario."
#
```

```
#
# Crea una cadena para paquetes tcp incorrectos.
#

$IPTABLES -N bad_tcp_packets

#
# Crea cadenas separadas para que los paquetes ICMP, TCP y UDP las atraviesen.
#

$IPTABLES -N allowed
$IPTABLES -N tcp_packets
$IPTABLES -N udp_packets
$IPTABLES -N icmp_packets

#
# 4.1.3 Creación de contenido en las cadenas de usuario
#

#
# Cadena bad_tcp_packets
#

$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \
--log-prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

#
# Cadena de "permitidos".
#

$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP

#
# Reglas TCP.
#

$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed
```

```
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 22 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 80 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 113 -j allowed

#
# Puertos UDP.
#

#$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 53 -j ACCEPT
#$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 123 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 2074 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 4000 -j ACCEPT

#
# En Redes Microsoft te verás inundado por difusiones de paquetes
# (broadcasts). Con las siguientes líneas evitarás que aparezcan en
# los registros.
#

#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d $INET_BROADCAST \
#--destination-port 135:139 -j DROP

#
# Si recibimos peticiones DHCP desde el exterior de nuestra red, nuestros
# registros también resultarán inundados. Con esta regla evitaremos
# que queden registrados.
#

#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d 255.255.255.255 \
#--destination-port 67:68 -j DROP

#
# Reglas ICMP.
#

$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

#
# 4.1.4 Cadena INPUT.
#
```

```
#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A INPUT -p tcp -j bad_tcp_packets

#
# Reglas para redes especiales que no son parte de Internet.
#

$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LO_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $LAN_IP -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -s $INET_IP -j ACCEPT

#
# Reglas para paquetes entrantes desde cualquier lugar.
#

$IPTABLES -A INPUT -p ALL -d $INET_IP -m state --state ESTABLISHED,RELATED \
-j ACCEPT
$IPTABLES -A INPUT -p TCP -j tcp_packets
$IPTABLES -A INPUT -p UDP -j udp_packets
$IPTABLES -A INPUT -p ICMP -j icmp_packets

#
# Si dispones de una Red Microsoft fuera de tu cortafuegos, también puedes
# verte inundado por multidifusiones (multicasts). Desechamos estos paquetes
# para no desbordar los registros.
#

#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT packet died: "

#
```

```
# 4.1.5 Cadena FORWARD.
#

#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

#
# Acepta los paquetes que sí queremos reenviar.
#

$IPTABLES -A FORWARD -p tcp --dport 21 -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 80 -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -p tcp --dport 110 -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet died: "

#
# 4.1.6 Cadena OUTPUT.
#

#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets

#
# Reglas especiales de la tabla OUTPUT para decidir qué IPs están permitidas.
#

$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
```

```
$IPTABLES -A OUTPUT -p ALL -s $INET_IP -j ACCEPT

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

######
# 4.2 Tabla Nat.
#

#
# 4.2.1 Establece las políticas.
#

#
# 4.2.2 Crea cadenas definidas por el usuario.
#

#
# 4.2.3 Crea contenido en las cadenas de usuario.
#

#
# 4.2.4 Cadena PREROUTING.
#

#
# 4.2.5 Cadena POSTROUTING.
#

#
# Activación del Reenvío IP simple y la Traducción de Dirección de Red (NAT).
#

$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to-source $INET_IP

#
# 4.2.6 Cadena OUTPUT.
```

```
#

######
# 4.3 Tabla Mangle.
#

#
# 4.3.1 Establece las políticas.
#

#
# 4.3.2 Crea cadenas definidas por el usuario.
#

#
# 4.3.3 Crea contenido en las cadenas de usuario.
#

#
# 4.3.4 Cadena PREROUTING.
#

#
# 4.3.5 Cadena INPUT.
#

#
# 4.3.6 Cadena FORWARD.
#

#
# 4.3.7 Cadena OUTPUT.
#

#
# 4.3.8 Cadena POSTROUTING.
#
```

Script de ejemplo rc.DHCP.firewall

```
#!/bin/sh
#
# rc.DHCP.firewall - Script de cortafuegos con IP basada en DHCP bajo
#                    Linux 2.4.x e iptables
#
# Copyright (C) 2001  Oskar Andreasson <bluefluxATkoffeinDOTnet>
#
# Este programa es software libre; puedes redistribuirlo y/o modificarlo
# bajo los términos expresados en la "GNU General Public License", tal como
# lo publica la Free Software Foundation; versión 2 de la Licencia.
#
# Este programa se distribuye con el deseo de que sea útil, pero
# SIN NINGUNA GARANTÍA; incluso sin garantía implícita de COMPRA-VENTA
# o ADECUACIÓN A PROPÓSITO PARTICULAR. Para más detalles, referirse a la
# GNU General Public License.
#
# Deberías haber recibido una copia de la GNU General Public License
# junto a este programa o desde el sitio web de dónde lo bajaste;
# si no es así, escribe a la Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA  02111-1307   USA
#
#
###########################################################################
#
# 1. Opciones de configuración.
#

#
# 1.1 Configuración de Internet.
#

INET_IFACE="eth0"

#
# 1.1.1 DHCP
#

#
```

```
# Información respecto a la conexión a Internet mediante DHCP, si es necesaria.
#
# Establece la variable DHCP a "no" si no obtienes una IP mediante DHCP. Si la
# obtienes mediante DHCP, establece la variable como "yes" e indica la dirección
# IP adecuada del servidor DHCP en la variable DHCP_SERVER.
#

DHCP="no"
DHCP_SERVER="195.22.90.65"

#
# 1.1.2 PPPoE
#

#
# Opciones de configuración respectivas a PPPoE (PPP over Ethernet).
#
# Si experimentas problemas con tu conexión PPPoE, como la imposibilidad de
# descargar correos electrónicos grandes, mientras que los pequeños bajan sin
# problemas, ..., puedes establecer esta opción como "yes" y es posible que
# se solucione el problema. Esta opción activa una regla en la cadena
# PREROUTING de la tabla 'mangle' que ajustará el tamaño de todos los paquetes
# enrutados al PMTU (Path Maximum Transmit Unit) [el tamaño máximo de los
# paquetes al pasar a través de interfases Ethernet].
#
# Ten en cuenta que es mejor establecer este valor en el propio paquete PPPoE,
# puesto que esta opción de configuración en PPPoE producirá una carga menor
# al sistema.
#

PPPOE_PMTU="no"

#
# 1.2 Configuracion de la Red de Área Local.
#
# El rango IP de la LAN (red de área local) y la IP del host local. El valor
# "/24" significa que sólo se utilizarán los primeros 24 bits de los 32 bits
# que tiene una dirección IP. Es lo mismo que la máscara de red "255.255.255.0".
#

LAN_IP="192.168.0.2"
```

```
LAN_IP_RANGE="192.168.0.0/16"
LAN_BROADCAST_ADDRESS="192.168.255.255"
LAN_IFACE="eth1"

#
# 1.3 Configuración de la DMZ ("Zona Desmilitarizada").
#

#
# 1.4 Configuración del host local.
#

LO_IFACE="lo"
LO_IP="127.0.0.1"

#
# 1.5 Configuración de IPTables.
#

IPTABLES="/usr/sbin/iptables"

#
# 1.6 Otras configuraciones.
#

###########################################################################
#
# 2. Carga de módulos.
#

#
# Necesario para la carga inicial de los módulos.
#

/sbin/depmod -a

#
# 2.1 Módulos requeridos.
#

/sbin/modprobe ip_conntrack
```

```
/sbin/modprobe ip_tables
/sbin/modprobe iptable_filter
/sbin/modprobe iptable_mangle
/sbin/modprobe iptable_nat
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_limit
/sbin/modprobe ipt_MASQUERADE

#
# 2.2 Módulos no-requeridos.
#

#/sbin/modprobe ipt_owner
#/sbin/modprobe ipt_REJECT
#/sbin/modprobe ip_conntrack_ftp
#/sbin/modprobe ip_conntrack_irc
#/sbin/modprobe ip_nat_ftp
#/sbin/modprobe ip_nat_irc

###########################################################################
#
# 3. Configuración de /proc.
#

#
# 3.1 Configuración requerida de proc.
#

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

#
# 3.2 Configuración no-requerida de proc.
#

#echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter
#echo "1" > /proc/sys/net/ipv4/conf/all/proxy_arp
#echo "1" > /proc/sys/net/ipv4/ip_dynaddr

###########################################################################
#
# 4. Configuración de las reglas.
```

```
#

######
# 4.1 Tabla Filter.
#

#
# 4.1.1 Establecimiento de políticas.
#

$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP

#
# 4.1.2 Creación de "cadenas de usuario."
#

#
# Crea una cadena para paquetes tcp incorrectos.
#

$IPTABLES -N bad_tcp_packets

#
# Crea cadenas separadas para que los paquetes ICMP, TCP y UDP las atraviesen.
#

$IPTABLES -N allowed
$IPTABLES -N tcp_packets
$IPTABLES -N udp_packets
$IPTABLES -N icmp_packets

#
# 4.1.3 Creación de contenido en las cadenas de usuario
#

#
# Cadena bad_tcp_packets
#
```

```
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG \ --log-
prefix "New not syn:"
$IPTABLES -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP

#
# Cadena de "permitidos".
#

$IPTABLES -A allowed -p TCP --syn -j ACCEPT
$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A allowed -p TCP -j DROP

#
# Reglas TCP.
#

$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 21 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 22 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 80 -j allowed
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 113 -j allowed

#
# Puertos UDP.
#

$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 53 -j ACCEPT
if [ $DHCP == "yes" ] ; then
 $IPTABLES -A udp_packets -p UDP -s $DHCP_SERVER --sport 67 \
 --dport 68 -j ACCEPT
fi

#$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 53 -j ACCEPT
#$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 123 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 2074 -j ACCEPT
$IPTABLES -A udp_packets -p UDP -s 0/0 --source-port 4000 -j ACCEPT

#
# En Redes Microsoft te verás inundado por difusiones de paquetes
# (broadcasts). Con las siguientes líneas evitarás que aparezcan en
# los registros.
#
```

```
#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE \
#--destination-port 135:139 -j DROP

#
# Si recibimos peticiones DHCP desde el exterior de nuestra red, nuestros
# registros también resultarán inundados. Con esta regla evitaremos
# que queden registrados.
#

#$IPTABLES -A udp_packets -p UDP -i $INET_IFACE -d 255.255.255.255 \
#--destination-port 67:68 -j DROP

#
# Reglas ICMP.
#

$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 8 -j ACCEPT
$IPTABLES -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

#
# 4.1.4 Cadena INPUT.
#

#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A INPUT -p tcp -j bad_tcp_packets

#
# Reglas para redes especiales que no son parte de Internet.
#

$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -s $LAN_IP_RANGE -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LO_IFACE -j ACCEPT
$IPTABLES -A INPUT -p ALL -i $LAN_IFACE -d $LAN_BROADCAST_ADDRESS -
j ACCEPT

#
# Regla especial para peticiones DHCP desde la red local, que de otra manera
```

```
# no son correctamente gestionadas.
#

$IPTABLES -A INPUT -p UDP -i $LAN_IFACE --dport 67 --sport 68 -j ACCEPT

#
# Reglas para paquetes entrantes desde Internet.
#

$IPTABLES -A INPUT -p ALL -i $INET_IFACE -m state --state ESTABLISHED,RELATED \ -
j ACCEPT
$IPTABLES -A INPUT -p TCP -i $INET_IFACE -j tcp_packets
$IPTABLES -A INPUT -p UDP -i $INET_IFACE -j udp_packets
$IPTABLES -A INPUT -p ICMP -i $INET_IFACE -j icmp_packets

#
# Si dispones de una Red Microsoft fuera de tu cortafuegos, también puedes
# verte inundado por multidifusiones (multicasts). Desechamos estos paquetes
# para no desbordar los registros.
#

#$IPTABLES -A INPUT -i $INET_IFACE -d 224.0.0.0/8 -j DROP

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT INPUT packet died: "

#
# 4.1.5 Cadena FORWARD.
#

#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A FORWARD -p tcp -j bad_tcp_packets

#
```

```
# Acepta los paquetes que sí queremos reenviar.
#

$IPTABLES -A FORWARD -i $LAN_IFACE -j ACCEPT
$IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A FORWARD -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT FORWARD packet died: "

#
# 4.1.6 Cadena OUTPUT.
#

#
# Paquetes TCP incorrectos que no queremos.
#

$IPTABLES -A OUTPUT -p tcp -j bad_tcp_packets

#
# Reglas especiales de la tabla OUTPUT para decidir qué IPs están permitidas.
#

$IPTABLES -A OUTPUT -p ALL -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -s $LAN_IP -j ACCEPT
$IPTABLES -A OUTPUT -p ALL -o $INET_IFACE -j ACCEPT

#
# Registra paquetes extraños que no concuerdan con lo anterior.
#

$IPTABLES -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
--log-level DEBUG --log-prefix "IPT OUTPUT packet died: "

######
# 4.2 Tabla Nat.
#
```

```
#
# 4.2.1 Establece las políticas.
#

#
# 4.2.2 Crea cadenas definidas por el usuario.
#

#
# 4.2.3 Crea contenido en las cadenas de usuario.
#

#
# 4.2.4 Cadena PREROUTING.
#

#
# 4.2.5 Cadena POSTROUTING.
#

if [ $PPPOE_PMTU == "yes" ] ; then
 $IPTABLES -t nat -A POSTROUTING -p tcp --tcp-flags SYN,RST SYN \
 -j TCPMSS --clamp-mss-to-pmtu
fi
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j MASQUERADE

#
# 4.2.6 Cadena OUTPUT.
#

######
# 4.3 Tabla Mangle.
#

#
# 4.3.1 Establece las políticas.
#

#
# 4.3.2 Crea cadenas definidas por el usuario.
```

```
#

#
# 4.3.3 Crea contenido en las cadenas de usuario.
#

#
# 4.3.4 Cadena PREROUTING.
#

#
# 4.3.5 Cadena INPUT.
#

#
# 4.3.6 Cadena FORWARD.
#

#
# 4.3.7 Cadena OUTPUT.
#

#
# 4.3.8 Cadena POSTROUTING.
#
```

Script de ejemplo rc.flush-iptables

```
#!/bin/sh
#
# rc.flush-iptables - Reinicia iptables a sus valores por defecto.
#
# Copyright (C) 2001  Oskar Andreasson <bluefluxATkoffeinDOTnet>
#
# Este programa es software libre; puedes redistribuirlo y/o modificarlo
```

```
# bajo los términos expresados en la "GNU General Public License", tal como
# lo publica la Free Software Foundation; versión 2 de la Licencia.
#
# Este programa se distribuye con el deseo de que sea útil, pero
# SIN NINGUNA GARANTÍA; incluso sin garantía implícita de COMPRA-VENTA
# o ADECUACIÓN A PROPÓSITO PARTICULAR. Para más detalles, referirse a la
# GNU General Public License.
#
# Deberías haber recibido una copia de la GNU General Public License
# junto a este programa o desde el sitio web de dónde lo bajaste;
# si no es así, escribe a la Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA  02111-1307   USA

#
# Configuraciones.
#
IPTABLES="/usr/sbin/iptables"

#
# Reinicia las políticas por defecto en la tabla Filter.
#
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT
$IPTABLES -P OUTPUT ACCEPT

#
# Reinicia las políticas por defecto en la tabla Nat.
#
$IPTABLES -t nat -P PREROUTING ACCEPT
$IPTABLES -t nat -P POSTROUTING ACCEPT
$IPTABLES -t nat -P OUTPUT ACCEPT

#
# Reinicia las políticas por defecto en la tabla Mangle.
#
$IPTABLES -t mangle -P PREROUTING ACCEPT
$IPTABLES -t mangle -P OUTPUT ACCEPT

#
# Elimina todas las reglas de las tablas Filter y Nat.
#
```

```
$IPTABLES -F
$IPTABLES -t nat -F
$IPTABLES -t mangle -F
#
# Borra todas las cadenas que no vienen por defecto con las
# tablas Filter y Nat.
#
$IPTABLES -X
$IPTABLES -t nat -X
$IPTABLES -t mangle -X
```

Script de ejemplo rc.test-iptables

```
#!/bin/bash
#
# rc.test-iptables - Script de prueba para las tablas y cadenas de iptables.
#
# Copyright (C) 2001  Oskar Andreasson <bluefluxATkoffeinDOTnet>
#
# Este programa es software libre; puedes redistribuirlo y/o modificarlo
# bajo los términos expresados en la "GNU General Public License", tal como
# lo publica la Free Software Foundation; versión 2 de la Licencia.
#
# Este programa se distribuye con el deseo de que sea útil, pero
# SIN NINGUNA GARANTÍA; incluso sin garantía implícita de COMPRA-VENTA
# o ADECUACIÓN A PROPÓSITO PARTICULAR. Para más detalles, referirse a la
# GNU General Public License.
#
# Deberías haber recibido una copia de la GNU General Public License
# junto a este programa o desde el sitio web de dónde lo bajaste;
# si no es así, escribe a la Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA  02111-1307   USA
#

#
```

```
# Tabla Filter, todas las cadenas.
#
iptables -t filter -A INPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter INPUT:"
iptables -t filter -A INPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter INPUT:"
iptables -t filter -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter OUTPUT:"
iptables -t filter -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter OUTPUT:"
iptables -t filter -A FORWARD -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter FORWARD:"
iptables -t filter -A FORWARD -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter FORWARD:"

#
# Tabla NAT, todas las cadenas excepto la OUTPUT, que no funcionaría.
#
iptables -t nat -A PREROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat PREROUTING:"
iptables -t nat -A PREROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat PREROUTING:"
iptables -t nat -A POSTROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat POSTROUTING:"
iptables -t nat -A POSTROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat POSTROUTING:"
iptables -t nat -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat OUTPUT:"
iptables -t nat -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat OUTPUT:"

#
# Tabla Mangle, todas las cadenas.
#
iptables -t mangle -A PREROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle PREROUTING:"
iptables -t mangle -A PREROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle PREROUTING:"
iptables -t mangle -I FORWARD 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle FORWARD:"
iptables -t mangle -I FORWARD 1 -p icmp --icmp-type echo-reply \
```

```
-j LOG --log-prefix="mangle FORWARD:"
iptables -t mangle -I INPUT 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle INPUT:"
iptables -t mangle -I INPUT 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle INPUT:"
iptables -t mangle -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle OUTPUT:"
iptables -t mangle -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle OUTPUT:"
iptables -t mangle -I POSTROUTING 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle POSTROUTING:"
iptables -t mangle -I POSTROUTING 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle POSTROUTING:"
```