



Universidad Simón Bolívar  
Laboratorio F  
Laboratorio Docente de Computación

## **IPTABLES y Squid**

Daniela A. Torres Faría  
daniela@ldc.usb.ve

Valle de Sartenejas, 9 de junio de 2010

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Firewall . . . . .	1
1.2. Proxy . . . . .	2
<b>2. Historia</b>	<b>3</b>
<b>3. IPTABLES</b>	<b>5</b>
3.1. Tablas . . . . .	6
3.2. Destinos . . . . .	6
3.3. Seguimiento de Conexiones . . . . .	9
3.4. El comando iptables . . . . .	11
3.4.1. Especificación de Reglas . . . . .	11
<b>4. Proxy, Squid</b>	<b>14</b>
4.1. Proxy . . . . .	14
4.1.1. Ventajas y Desventajas del uso de Proxy . . . . .	14
4.1.1.1. Ventajas . . . . .	14
4.1.1.2. Desventajas . . . . .	15
4.1.2. Funcionamiento . . . . .	15
4.1.3. Tipos de Proxy . . . . .	16
4.1.3.1. Proxy Transparente . . . . .	16
4.1.3.2. Proxy Reverso . . . . .	17
4.1.3.3. Proxy Abierto . . . . .	17
4.2. Squid . . . . .	18
4.2.1. Características . . . . .	18
4.2.2. Compatibilidad . . . . .	20
<b>5. Implementación</b>	<b>21</b>
5.1. Configuración de NAT y Firewall para la red privada . . . . .	22

5.2. Instalación y configuración de servidor DHCP para la red privada . . . . .	25
5.3. Instalación y configuración de Servidor Squid . . . . .	25

# Capítulo 1

## Introducción

Con la creciente evolución de los sistemas y redes de computadoras, la seguridad se ha convertido en uno de los principales puntos de enfoque al momento de administrar y configurar servicios de red.

Existen dos soluciones que, en estos momentos, se pueden considerar indispensables en materia de seguridad de redes:

### 1.1. Firewall

Es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. Se trata de un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar, descifrar, el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios. Los cortafuegos pueden ser implementados en hardware o software, o una combinación de ambos. Los cortafuegos se utilizan con frecuencia para evitar que los usuarios de Internet no autorizados tengan acceso a redes privadas conectadas a Internet, especialmente intranets. Todos los mensajes que entren o salgan de la intranet pasan a través del cortafuegos, que examina cada mensaje y bloquea aquellos que no cumplen los criterios de seguridad especificados. También es frecuente conectar al cortafuegos a una tercera red, llamada Zona desmilitarizada o DMZ, en la que se ubican los servidores de la organización que deben permanecer accesibles desde la red exterior. Un cortafuegos correctamente configurado añade una protección necesaria a la red, pero que en ningún caso debe considerarse suficiente.

El firewall niega el acceso a aquellos paquetes que deseen entrar a la red cuyo puerto destino

---

este bloqueado dentro de la configuración establecida por el administrador de la red.

## 1.2. Proxy

Es un programa o dispositivo que realiza una acción en representación de otro. Su finalidad más habitual es la de servidor proxy, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

Un proxy permite a otros equipos conectarse a una red de forma indirecta a través de él. Cuando un equipo de la red desea acceder a una información o recurso, es realmente el proxy quien realiza la comunicación y a continuación traslada el resultado al equipo inicial. En unos casos esto se hace así porque no es posible la comunicación directa y en otros casos porque el proxy añade una funcionalidad adicional, como puede ser la de mantener los resultados obtenidos, en una caché que permita acelerar sucesivas consultas coincidentes. Con esta denominación general de proxy se agrupan diversas técnicas.

Además de esto un proxy es una herramienta que puede ser usado en materia de seguridad, ya que este puede analizar las peticiones realizadas por las maquinas de la red y bloquear algunas de ellas según sea la configuración establecida, por ejemplo una de ellas sería la de establecer reglas que prohíban el acceso a páginas web de contenido inapropiado o el acceso a ciertos protocolos de mensajería instantánea.

## Capítulo 2

# Historia

El proyecto Netfilter/iptables comenzó en 1998 con Rusty Russell, también autor del proyecto que lo precedió, ipchains. A medida que el proyecto fue creciendo, fundó el Netfilter Core Team (o simplemente el coreteam) en 1999, que se trata del grupo de personas que se encarga directamente del desarrollo y mantenimiento del proyecto. El software que ellos produjeron (al que llamaremos netfilter de aquí en adelante) está licenciado bajo la licencia GPL (GNU General Public License), y fue incorporado al núcleo Linux 2.3 en marzo de 2000. En agosto de 2003, Harald Welte fue designado líder del coreteam. En febrero de 2007 entra a formar parte del coreteam el andaluz Pablo Neira.

Antes de iptables, los programas más usados para crear firewalls en Linux eran ipchains en el núcleo Linux 2.2 e ipfwadm en el núcleo Linux 2.0, que a su vez se basaba en ipfw de BSD. Tanto ipchains como ipfwadm alteran el código de red para poder manipular los paquetes, ya que no existía un framework general para el manejo de paquetes hasta la aparición de netfilter. iptables mantiene la idea básica introducida en Linux con ipfwadm: listas de reglas en las que se especifica qué matchear dentro de un paquete y qué hacer con ese paquete. ipchains agrega el concepto de cadenas de reglas (chains) e iptables extendió esto a la idea de tablas: se consultaba una tabla para decidir si había que NAT-ear un paquete, y se consultaba otra para decidir como filtrar un paquete. Adicionalmente, se modificaron los tres puntos en los que se realiza el filtrado en el viaje de un paquete, de modo que un paquete pase solo por un punto de filtrado.

Mientras que ipchains e ipfwadm combinan filtrado de paquetes y NAT (específicamente tres tipos de NAT, llamados masquerading o enmascaramiento de IP, port forwarding o redireccionamiento de puertos, y redirection o redirección), netfilter hace posible por su parte separar las operaciones sobre los paquetes en tres partes: packet filtering (filtrado de paquetes), connection tracking (seguimiento de conexiones) y Network Address Translation (NAT o traducción de direcciones de red). Cada parte se

conecta a las herramientas de netfilter en diferentes puntos para acceder a los paquetes. Los subsistemas de seguimiento de conexiones y NAT son más generales y poderosos que los que realizaban ipchains e ipfwadm.

Esta división permite a iptables, a su vez, usar la información que la capa de seguimiento de conexiones ha determinado acerca del paquete: esta información estaba antes asociada a NAT. Esto hace a iptables superior a ipchains, ya que tiene la habilidad de monitorizar el estado de una conexión y redirigir, modificar o detener los paquetes de datos basados en el estado de la conexión y no solamente por el origen, destino o contenido del paquete. Un cortafuegos que utilice iptables de este modo se llama cortafuegos stateful, contrario a ipchains que solo permite crear un cortafuegos stateless (excepto en casos muy limitados). Podemos decir entonces que ipchains no está al tanto del contexto completo en el cual un paquete surge, mientras que iptables sí y por lo tanto iptables puede hacer mejores decisiones sobre el futuro de los paquetes y las conexiones.

iptables, el subsistema NAT y el subsistema de seguimiento de conexiones son también extensibles, y muchas extensiones ya están incluidas en el paquete básico de iptables, tal como la extensión ya mencionada que permite la consulta del estado de la conexión. Extensiones adicionales se distribuyen junto a la utilidad iptables, como parches al código fuente del núcleo junto con una herramienta llamada patch-o-matic que permite aplicar los parches.

Una versión de iptables para IPv6 ya fue escrita, llamada ip6tables al igual que la herramienta de administración.

Squid es un Servidor Intermediario (Proxy) de alto desempeño que se ha venido desarrollando desde hace varios años y es hoy en día un muy popular y ampliamente utilizado entre los sistemas operativos como GNU/Linux y derivados de Unix. Es muy confiable, robusto y versátil y se distribuye bajo los términos de la Licencia Pública General GNU (GNU/GPL). Siendo sustento lógico libre, está disponible el código fuente para quien así lo requiera.

## Capítulo 3

# IPTABLES

Iptables permite al administrador del sistema definir reglas acerca de qué hacer con los paquetes de red. Las reglas se agrupan en cadenas: cada cadena es una lista ordenada de reglas. Las cadenas se agrupan en tablas: cada tabla está asociada con un tipo diferente de procesamiento de paquetes.

Cada regla especifica qué paquetes la cumplen (match) y un destino que indica qué hacer con el paquete si éste cumple la regla. Cada paquete de red que llega a una computadora o que se envía desde una computadora recorre por lo menos una cadena y cada regla de esa cadena se comprueba con el paquete. Si la regla cumple con el datagrama, el recorrido se detiene y el destino de la regla dicta lo que se debe hacer con el paquete. Si el paquete alcanza el fin de una cadena predefinida sin haberse correspondido con ninguna regla de la cadena, la política de destino de la cadena dicta qué hacer con el paquete. Si el paquete alcanza el fin de una cadena definida por el usuario sin haber cumplido ninguna regla de la cadena o si la cadena definida por el usuario está vacía, el recorrido continúa en la cadena que hizo la llamada (lo que se denomina `implicit target RETURN` o `RETORNO` de destino implícito). Solo las cadenas predefinidas tienen políticas.

En iptables, las reglas se agrupan en cadenas. Una cadena es un conjunto de reglas para paquetes IP, que determinan lo que se debe hacer con ellos. Cada regla puede desechar el paquete de la cadena (cortocircuito), con lo cual otras cadenas no serán consideradas. Una cadena puede contener un enlace a otra cadena: si el paquete pasa a través de esa cadena entera o si cumple una regla de destino de retorno, va a continuar en la primera cadena. No hay un límite respecto de cuán anidadas pueden estar las cadenas. Hay tres cadenas básicas (`INPUT`, `OUTPUT` y `FORWARD`: `ENTRADA`, `SALIDA` y `REENVÍO`) y el usuario puede crear tantas como desee. Una regla puede ser simplemente un puntero a una cadena.

## 3.1. Tablas

Hay tres tablas ya incorporadas, cada una de las cuales contiene ciertas cadenas predefinidas. Es posible crear nuevas tablas mediante módulos de extensión. El administrador puede crear y eliminar cadenas definidas por usuarios dentro de cualquier tabla. Inicialmente, todas las cadenas están vacías y tienen una política de destino que permite que todos los paquetes pasen sin ser bloqueados o alterados.

Además de las cadenas ya incorporadas, el usuario puede crear todas las cadenas definidas por el usuario que quiera dentro de cada tabla, las cuales permiten agrupar las reglas en forma lógica.

Cada cadena contiene una lista de reglas. Cuando un paquete se envía a una cadena, se lo compara, en orden, contra cada regla en la cadena. La regla especifica qué propiedades debe tener el paquete para que la regla coincida, como número de puerto o dirección IP. Si la regla no coincide, el procesamiento continúa con la regla siguiente. Si la regla, por el contrario, coincide con el paquete, las instrucciones de destino de las reglas se siguen (y cualquier otro procesamiento de la cadena normalmente se aborta). Algunas propiedades de los paquetes solo pueden examinarse en ciertas cadenas (por ejemplo, la interfaz de red de salida no es válida en la cadena de ENTRADA). Algunos destinos solo pueden usarse en ciertas cadenas y/o en ciertas tablas (por ejemplo, el destino SNAT solo puede usarse en la cadena de POSRUTEO de la tabla de traducción de direcciones de red).

## 3.2. Destinos

El destino de una regla puede ser el nombre de una cadena definida por el usuario o uno de los destinos ya incorporados ACCEPT, DROP, QUEUE, o RETURN (aceptar, descartar, encolar o retornar, respectivamente). Cuando un destino es el nombre de una cadena definida por el usuario, al paquete se lo dirige a esa cadena para que sea procesado (tal como ocurre con una llamada a una subrutina en un lenguaje de programación). Si el paquete consigue atravesar la cadena definida por el usuario sin que ninguna de las reglas de esa cadena actúe sobre él, el procesamiento del paquete continúa donde había quedado en la cadena actual. Estas llamadas entre cadenas se pueden anidar hasta cualquier nivel deseado.

Existen los siguientes destinos ya incorporados:

### **ACCEPT (aceptar)**

Este destino hace que netfilter acepte el paquete. El significado de esto depende de cuál sea la cadena realizando esta aceptación. Un paquete que se acepta en la cadena de ENTRADA se le permite ser recibido por el sistema (host), un paquete que se acepta en la cadena de SALIDA se

le permite abandonar el sistema y un paquete que se acepta en la cadena de REDIRECCIÓN se le permite ser encaminado (routing) a través del sistema.

### **DROP (descartar)**

Este destino hace que netfilter descarte el paquete sin ningún otro tipo de procesamiento. El paquete simplemente desaparece sin ningún tipo de indicación al sistema o aplicación de origen, de que fue descartado en el sistema de destino. Esto se refleja en el sistema que envía el paquete a menudo, como un communication timeout (alcance del máximo tiempo de espera en la comunicación), lo que puede causar confusión, aunque el descarte de paquetes entrantes no deseados se considera a veces una buena política de seguridad, pues no da ni siquiera el indicio a un posible atacante de que el sistema destino existe.

### **QUEUE (encolar)**

Este destino hace que el paquete sea enviado a una cola en el espacio de usuario. Una aplicación puede usar la biblioteca libipq, también parte del proyecto netfilter/iptables, para alterar el paquete. Si no hay ninguna aplicación que lea la cola, este destino es equivalente a DROP.

### **RETURN (retorno)**

Hace que el paquete en cuestión deje de circular por la cadena en cuya regla se ejecutó el destino RETURN. Si dicha cadena es una subcadena de otra, el paquete continuará por la cadena superior como si nada hubiera pasado. Si por el contrario la cadena es una cadena principal (por ejemplo la cadena INPUT), al paquete se le aplicará la política por defecto de la cadena en cuestión (ACCEPT, DROP o similar).

Existen destinos de extensión disponibles y algunos de los más comunes son:

### **REJECT (rechazo)**

Este destino tiene el mismo efecto que 'DROP', salvo que envía un paquete de error a quien envió originalmente. Se usa principalmente en las cadenas de ENTRADA y de REDIRECCIÓN de la tabla de filtrado. El tipo de paquete se puede controlar a través del parámetro '-reject-with'. Un paquete de rechazo puede indicar explícitamente que la conexión ha sido filtrada (un paquete ICMP filtrado administrativamente por conexión), aunque la mayoría de los usuarios prefieren que el paquete indique simplemente que la computadora no acepta ese tipo de conexión (tal paquete será un paquete tcp-reset para conexiones TCP denegadas, un icmp-port-unreachable para sesiones UDP denegadas o un icmp-protocol-unreachable para paquetes no TCP y no UDP). Si el parámetro '-reject-with' no se especifica, el paquete de rechazo por defecto es siempre icmp-port-unreachable.

**LOG (bitácora)**

Este destino lleva un log o bitácora del paquete. Puede usarse en cualquier cadena en cualquier tabla, y muchas veces se usa para debuggear (análisis de fallos, como ser la verificación de qué paquetes están siendo descartados).

**ULOG**

Este destino lleva un log o bitácora del paquete, pero no de la misma manera que el destino LOG. El destino LOG le envía información al log del núcleo, pero ULOG hace multidifusión de los paquetes que coincidan con esta regla a través de un socket netlink, de manera que programas del espacio de usuario puedan recibir este paquete conectándose al socket.

**DNAT**

Este destino hace que la dirección (y opcionalmente el puerto) de destino del paquete sean reescritos para traducción de dirección de red. Mediante la opción '-to-destination' debe indicarse el destino a usar. Esto es válido solamente en las cadenas de SALIDA y PRERUTEO dentro de la tabla de nat. Esta decisión se recuerda para todos los paquetes futuros que pertenecen a la misma conexión y las respuestas tendrán su dirección y puerto de origen cambiados al original (es decir, la inversa de este paquete).

**SNAT**

Este destino hace que la dirección (y opcionalmente el puerto) de origen del paquete sean reescritos para traducción de dirección de red. Mediante la opción '-to-source' debe indicarse el origen a usar. Esto es válido solamente en la cadena de POSRUTEO dentro de la tabla de nat y, como DNAT, se recuerda para todos los paquetes que pertenecen a la misma conexión.

**MASQUERADE**

Esta es una forma especial, restringida de SNAT para direcciones IP dinámicas, como las que proveen la mayoría de los proveedores de servicios de Internet (ISPs) para módems o línea de abonado digital (DSL). En vez de cambiar la regla de SNAT cada vez que la dirección IP cambia, se calcula la dirección IP de origen a la cual hacer NAT fijándose en la dirección IP de la interfaz de salida cuando un paquete coincide con esta esta regla. Adicionalmente, recuerda cuales conexiones usan MASQUERADE y si la dirección de la interfaz cambia (por ejemplo, por reconectarse al ISP), todas las conexiones que hacen NAT a la dirección vieja se olvidan.

### 3.3. Seguimiento de Conexiones

Una de las características de importancia que está construída por encima del framework de netfilter es el seguimiento de conexiones (connection tracking). El seguimiento de conexiones le permite al núcleo llevar cuenta de todas las conexiones o sesiones lógicas de red y de este modo relacionar todos los paquetes que pueden llegar a formar parte de esa conexión. La traducción de dirección de red (NAT) depende de esta información para traducir todos los paquetes relacionados de la misma manera e iptables puede usar esta información para actuar como un cortafuegos stateful.

El seguimiento de conexiones clasifica cada paquete en uno de cuatro estados:

**NEW (nuevo)**

Intentando crear una conexión nueva.

**ESTABLISHED (establecido)**

Parte de una conexión ya existente.

**RELATED (relacionado)**

Relacionada, aunque no realmente parte de una conexión existente.

**INVALID (inválido)**

No es parte de una conexión existente e incapaz de crear una conexión nueva.

Un caso común sería que el primer paquete que el cortafuegos ve se clasificará como NEW, la respuesta se clasificará como ESTABLISHED y un error ICMP sería RELATED. Un paquete de error ICMP que no coincide con una conexión conocida será INVALID.

El estado de la conexión es completamente independiente de cualquier estado de TCP. Si la terminal (host) responde con un paquete SYN ACK para señalar una conexión TCP entrante nueva, la conexión TCP misma no se establece, pero la conexión a la que se le hace el seguimiento sí se establece. Este paquete 'coincidirá el estado ESTABLISHED.

Sin embargo, una conexión a la que se le hace el seguimiento de un protocolo sin estado como UDP tiene un estado de conexión.

Es más, mediante el uso de módulo que pueden agregarse, al seguimiento de conexión se le puede dar conocimiento de los protocolos de la capa de aplicación y así que entienda que dos o más conexiones distintas están relacionadas". Por ejemplo, considérese el protocolo FTP. Se establece una conexión de control, pero cada vez que se transfiere información, se establece otra conexión para que la transfiera. Si se carga el módulo *nf-conntrack-ftp*, el primer paquete de una conexión de datos FTP se clasificará como RELATED en lugar de NEW, ya que lógicamente es parte de una conexión existente.

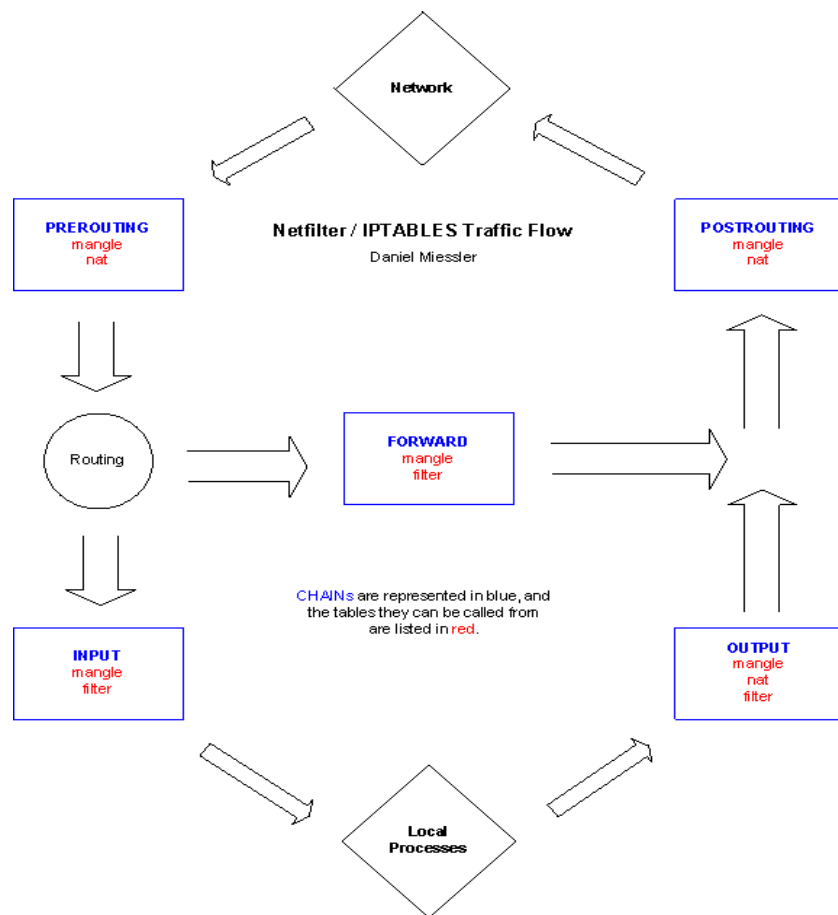


Figura 3.1: Ejemplo de tráfico de paquetes con filtrado iptables. Las cadenas están representadas en azul y las tablas en color rojo.

iptables puede usar la información del seguimiento de conexiones para conseguir hacer reglas de filtrado de paquetes más potentes y más fáciles de manejar. La extensión de `.estado` le permite a las reglas de iptables que examinen la clasificación de seguimiento de conexión para un paquete. Por ejemplo, una regla puede permitir el paso solo a paquetes NEW desde dentro del cortafuegos hacia afuera, pero permitir paquetes RELATED y ESTABLISHED en ambas direcciones. Esto permite el paso de paquetes de respuesta normales desde el exterior (ESTABLISHED), pero no permite que lleguen conexiones nuevas desde el exterior al interior. Sin embargo, si una conexión de datos FTP necesita llegar desde el exterior del cortafuegos hacia el interior, será permitido, porque el paquete se clasificará correctamente como RELATED para la conexión de control FTP, en vez de NEW.

## 3.4. El comando iptables

En la mayoría de los sistemas Linux, iptables está instalado como `/sbin/iptables`. La sintaxis detallada del comando iptables está documentada en su página de man, la cual puede verse tipeando el comando `"man iptables"` desde la línea de comandos.

### Flags mas comunes

`-t tabla`

Hace que el comando se aplique a la tabla especificada. Si esta opción se omite, el comando se aplica a la tabla `filter` por defecto.

`-v`

Produce una salida con detalles (del inglés, *verbose*).

`-n`

Produce una salida numérica (es decir, números de puerto en lugar de nombres de servicio y direcciones IP en lugar de nombres de dominio).

`--line-numbers`

Cuando se listan reglas, agrega números de línea al comienzo de cada regla, correspondientes a la posición de esa regla en su cadena.

### 3.4.1. Especificación de Reglas

La mayoría de las formas de comandos de iptables requieren que se les indiquen una especificación de reglas, que es usada para *matchear* un subconjunto particular del tráfico de paquetes de red procesados por una cadena. La especificación de regla incluye también un destino que especifica qué hacer con paquetes que son *matcheados* por la regla. Las siguientes opciones se usan (frecuentemente combinadas unas con otras) para crear especificaciones de reglas.

`-j destino, --jump destino`

Especifica el destino de una regla. El destino es el nombre de una cadena definida por el usuario (creada usando la opción `-N`, uno de los destinos ya incorporados, `ACCEPT`, `DROP`, `QUEUE`, o `RETURN`, o un destino de extensión, como `REJECT`, `LOG`, `DNAT`, o `SNAT`. Si esta opción es omitida en una regla, entonces el *matcheo* de la regla no tendrá efecto en el destino de un paquete, pero los contadores en la regla se incrementarán.

-i [!] in-interface, -in-interface [!] in-interface

Nombre de una interfaz a través de la cual un paquete va a ser recibido (solo para paquetes entrando en las cadenas de INPUT, FORWARD y PREROUTING). Cuando se usa el argumento '!' antes del nombre de la interfaz, el significado se invierte. Si el nombre de la interfaz termina con '+', entonces cualquier interfaz que comience con este nombre será matcheada. Si esta opción se omite, se matcheará todo nombre de interfaz.

-o [!] out-interface, -out-interface [!] out-interface

Nombre de una interfaz a través de la cual un paquete va a ser enviado (para paquetes entrando en las cadenas de FORWARD, OUTPUT y POSTROUTING). Cuando se usa el argumento '!' antes del nombre de la interfaz, el significado se invierte. Si el nombre de la interfaz termina con '+', entonces cualquier interfaz que comience con este nombre será matcheada. Si esta opción se omite, se matcheará todo nombre de interfaz.

-p [!] protocol, -protocol [!] protocol

Matchea paquetes del nombre de protocolo especificado. Si '!' precede el nombre de protocolo, se matchean todos los paquetes que no son el protocolo especificado. Nombres de protocolo válidos son icmp, udp, tcp... Una lista de todos los protocolos válidos puede encontrarse en el archivo /etc/protocols.

-s [!] origen[/prefijo], -source [!] origen[/prefijo]

Matchea paquetes IP viniendo de la dirección de origen especificada. La dirección de origen puede ser una dirección IP, una dirección IP con un prefijo de red asociado, o un nombre de terminal (hostname). Si '!' precede al origen, se matchean todos los paquetes que no vienen del origen especificado.

-d [!] destino[/prefijo], -destination [!] destino[/prefijo]

Matchea paquetes IP yendo a la dirección de destino especificada. La dirección de destino puede ser una dirección IP, una dirección IP con un prefijo de red asociado, o un nombre de terminal (hostname). Si '!' precede al origen, se matchean todos los paquetes que no van al destino especificado.

-destination-port [!] [puerto[:puerto]], -dport [!] [puerto[:puerto]]

Matchea paquetes TCP o UDP (dependiendo del argumento a la opción -p) destinados a los puertos o rango de puertos (cuando se usa la forma puerto:puerto) especificados. Si '!' precede la especificación de puertos, se matchean todos los paquetes TCP o UDP que no están destinados a los puertos o rango de puertos especificados.

`-source-port [!] [puerto[:puerto]], -sport [!] [puerto[:puerto]]` Matchea paquetes TCP o UDP (dependiendo del argumento a la opción `-p`) que vienen de los puertos o rango de puertos (cuando se usa la forma `puerto:puerto`) especificados. Si `'!'` precede la especificación de puertos, se matchean todos los paquetes TCP o UDP que no vienen de los puertos o rango de puertos especificados.

`-tcp-flags [!] mask comp`

Matchea paquetes TCP que tienen marcadas o desmarcadas ciertas banderas del protocolo TCP. El primer argumento especifica las banderas a examinar en cada paquete TCP, escritas en una lista separada por comas (no se permiten espacios). El segundo argumento es otra lista separada por comas de banderas que deben estar marcadas dentro de las que se debe examinar. Estas banderas son: SYN, ACK, FIN, RST, URG, PSH, ALL, y NONE. Por lo tanto, la opción `-tcp-flags SYN, ACK, FIN, RST SYN` solo va a matchear paquetes con la bandera SYN marcada y las banderas ACK, FIN y RST desmarcadas.

`! -syn`

Matchea paquetes TCP que tienen la bandera SYN marcada y las banderas ACK, FIN y RST desmarcadas. Estos paquetes son los que se usan para iniciar conexiones TCP. Al bloquear tales paquetes en la cadena de INPUT, se previenen conexiones TCP entrantes, pero conexiones TCP salientes no serán afectadas. Esta opción puede combinarse con otras, como `-source`, para bloquear o dejar pasar conexiones TCP entrantes solo de ciertas terminales o redes. Esta opción es equivalente a `-tcp-flags SYN, RST, ACK SYN`. Si `'!'` precede a `-syn`, el significado de la opción se invierte.

Para finalizar este capítulo vale la pena resaltar que se presentará un ejemplo de uso de la herramienta iptables en conjunto con el proxy squid.

# Capítulo 4

## Proxy, Squid

### 4.1. Proxy

El término proxy hace referencia a un programa o dispositivo que realiza una acción en representación de otro. Su finalidad más habitual es la de servidor proxy, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

El uso más común es el de servidor proxy, que es un ordenador que intercepta las conexiones de red que un cliente hace a un servidor de destino, de ellos, el más famoso es el servidor proxy web (comúnmente conocido solamente como «proxy»). Intercepta la navegación de los clientes por páginas web, por varios motivos posibles: seguridad, rendimiento, anonimato, etc. También existen proxies para otros protocolos, como el proxy de FTP. El proxy ARP puede hacer de enrutador en una red, ya que hace de intermediario entre ordenadores.

#### 4.1.1. Ventajas y Desventajas del uso de Proxy

##### 4.1.1.1. Ventajas

- Control: sólo el intermediario hace el trabajo real, por tanto se pueden limitar y restringir los derechos de los usuarios, y dar permisos sólo al proxy.
- Ahorro. Por tanto, sólo uno de los usuarios (el proxy) ha de estar equipado para hacer el trabajo real.

- **Velocidad.** Si varios clientes van a pedir el mismo recurso, el proxy puede hacer caché: guardar la respuesta de una petición para darla directamente cuando otro usuario la pida. Así no tiene que volver a contactar con el destino, y acaba más rápido.
- **Filtrado.** El proxy puede negarse a responder algunas peticiones si detecta que están prohibidas.
- **Modificación.** Como intermediario que es, un proxy puede falsificar información, o modificarla siguiendo un algoritmo.
- **Anonimato.** Si todos los usuarios se identifican como uno sólo, es difícil que el recurso accedido pueda diferenciarlos. Pero esto puede ser malo, por ejemplo cuando hay que hacer necesariamente la identificación.

#### 4.1.1.2. Desventajas

- **Abuso.** Al estar dispuesto a recibir peticiones de muchos usuarios y responderlas, es posible que haga algún trabajo que no toque. Por tanto, ha de controlar quién tiene acceso y quién no a sus servicios, cosa que normalmente es muy difícil.
- **Carga.** Un proxy ha de hacer el trabajo de muchos usuarios.
- **Intromisión.** Es un paso más entre origen y destino, y algunos usuarios pueden no querer pasar por el proxy. Y menos si hace de caché y guarda copias de los datos.
- **Incoherencia.** Si hace de caché, es posible que se equivoque y dé una respuesta antigua cuando hay una más reciente en el recurso de destino. En realidad este problema no existe con los servidores proxy actuales, ya que se conectan con el servidor remoto para comprobar que la versión que tiene en cache sigue siendo la misma que la existente en el servidor remoto.
- **Irregularidad.** El hecho de que el proxy represente a más de un usuario da problemas en muchos escenarios, en concreto los que presuponen una comunicación directa entre 1 emisor y 1 receptor (como TCP/IP).

#### 4.1.2. Funcionamiento

Cuando un equipo de la red desea acceder a una información o recurso, es realmente el proxy quien realiza la comunicación y a continuación traslada el resultado al equipo inicial. En unos casos esto se hace así porque no es posible la comunicación directa y en otros casos porque el proxy añade una funcionalidad adicional, como puede ser la de mantener los resultados obtenidos (p.ej.: una página web) en una caché que permita acelerar sucesivas consultas coincidentes.

El proxy web funciona de la siguiente manera:

1. El cliente realiza una petición (p. ej. mediante un navegador web) de un recurso de Internet (una página web o cualquier otro archivo) especificado por una URL.
2. Cuando el proxy caché recibe la petición, busca la URL resultante en su caché local. Si la encuentra, contrasta la fecha y hora de la versión de la página demanda con el servidor remoto. Si la página no ha cambiado desde que se cargo en caché la devuelve inmediatamente, ahorrándose de esta manera mucho tráfico pues sólo intercambia un paquete para comprobar la versión. Si la versión es antigua o simplemente no se encuentra en la caché, lo captura del servidor remoto, lo devuelve al que lo pidió y guarda o actualiza una copia en su caché para futuras peticiones.

El caché utiliza normalmente un algoritmo para determinar cuándo un documento está obsoleto y debe ser eliminado de la caché, dependiendo de su antigüedad, tamaño e histórico de acceso. Dos de esos algoritmos básicos son el LRU (el usado menos recientemente, en inglés "Least Recently Used") y el LFU (el usado menos frecuentemente, "Least Frequently Used").

Los proxies web también pueden filtrar el contenido de las páginas Web servidas. Algunas aplicaciones que intentan bloquear contenido Web ofensivo están implementadas como proxies Web. Otros tipos de proxy cambian el formato de las páginas web para un propósito o una audiencia específicos, para, por ejemplo, mostrar una página en un teléfono móvil o una PDA. Algunos operadores de red también tienen proxies para interceptar virus y otros contenidos hostiles servidos por páginas Web remotas.

Un cliente de un ISP manda una petición a Google la cual llega en un inicio al servidor Proxy que tiene este ISP, no va directamente a la dirección IP del dominio de Google. Esta página concreta suele ser muy solicitada por un alto porcentaje de usuarios, por lo tanto el ISP la retiene en su Proxy por un cierto tiempo y crea una respuesta en mucho menor tiempo. Cuando el usuario crea una búsqueda en Google el servidor Proxy ya no es utilizado; el ISP envía su petición y el cliente recibe su respuesta ahora sí desde Google.

### 4.1.3. Tipos de Proxy

#### 4.1.3.1. Proxy Transparente

Muchas organizaciones (incluyendo empresas, colegios y familias) usan los proxies para reforzar las políticas de uso de la red o para proporcionar seguridad y servicios de caché. Normalmente, un proxy Web o NAT no es transparente a la aplicación cliente: debe ser configurada para usar el proxy,

manualmente. Por lo tanto, el usuario puede evadir el proxy cambiando simplemente la configuración. Una ventaja de tal es que se puede usar para redes de empresa.

Un proxy transparente combina un servidor proxy con NAT(Network Address Translation) de manera que las conexiones son enrutadas dentro del proxy sin configuración por parte del cliente, y habitualmente sin que el propio cliente conozca de su existencia. Este es el tipo de proxy que utilizan los proveedores de servicios de internet (ISP). En España, la compañía más expandida en cuanto a ADSL se refiere, ISP Telefónica, dejó de utilizar proxy transparente con sus clientes a partir de Febrero de 2006.

#### 4.1.3.2. Proxy Reverso

Un reverse proxy es un servidor proxy instalado en el domicilio de uno o más servidores web. Todo el tráfico entrante de Internet y con el destino de uno de esos servidores web pasa a través del servidor proxy. Hay varias razones para instalar un reverse proxy”:

1. Seguridad: el servidor proxy es una capa adicional de defensa y por lo tanto protege los servidores web.
2. Cifrado / Aceleración SSL: cuando se crea un sitio web seguro, habitualmente el cifrado SSL no lo hace el mismo servidor web, sino que es realizado por el reverse proxy”, el cual está equipado con un hardware de aceleración SSL (Security Sockets Layer).
3. Distribución de Carga: el reverse proxy” puede distribuir la carga entre varios servidores web. En ese caso, el reverse proxy” puede necesitar reescribir las URL de cada página web (traducción de la URL externa a la URL interna correspondiente, según en qué servidor se encuentre la información solicitada).
4. Caché de contenido estático: Un reverse proxy” puede descargar los servidores web almacenando contenido estático como imágenes u otro contenido gráfico.

#### 4.1.3.3. Proxy Abierto

Este tipo de proxy que acepta peticiones desde cualquier ordenador, esté o no conectado a su red.

En esta configuración el proxy ejecutará cualquier petición de cualquier ordenador que pueda conectarse a él, realizándola como si fuera una petición del proxy. Por lo que permite que este tipo de proxy se use como pasarela para el envío masivo de correos de spam. Un proxy se usa, normalmente,

para almacenar y redirigir servicios como el DNS o la navegación Web, mediante el cacheo de peticiones en el servidor proxy, lo que mejora la velocidad general de los usuarios. Este uso es muy beneficioso, pero al aplicarle una configuración `.^abierta.^` todo internet, se convierte en una herramienta para su uso indebido.

Debido a lo anterior, muchos servidores, como los de IRC, o correo electrónicos, deniegan el acceso a estos proxys a sus servicios, usando normalmente listas negras ("BlackList").

Vale recalcar que también se puede implementar un proxy implementando un NAT que tiene por detras una red privada.

## 4.2. Squid

Squid es un popular programa de software libre que implementa un servidor proxy y un dominio para caché de páginas web, publicado bajo licencia GPL. Tiene una amplia variedad de utilidades, desde acelerar un servidor web, guardando en caché peticiones repetidas a DNS y otras búsquedas para un grupo de gente que comparte recursos de la red, hasta caché de web, además de añadir seguridad filtrando el tráfico. Está especialmente diseñado para ejecutarse bajo entornos tipo Unix.

Squid ha sido desarrollado durante muchos años y se le considera muy completo y robusto. Aunque orientado a principalmente a HTTP y FTP es compatible con otros protocolos como Internet Gopher. Implementa varias modalidades de cifrado como TLS, SSL, y HTTPS.

### 4.2.1. Características

Squid se configura desde un unico fichero, el `/etc/squid.conf`, es muy sencillo de configurar ya que todos los comandos internos estan comentados para facilitar la configuracion. Los comandos basicos necesarios los puedes buscar y poner los valores por defecto.

Además de esto squid posee las siguientes características:

**Proxy y Caché de HTTP, FTP, y otras URL** Squid proporciona un servicio de Proxy que soporta peticiones HTTP, HTTPS y FTP a equipos que necesitan acceder a Internet y a su vez provee la funcionalidad de caché especializado en el cual almacena de forma local las páginas consultadas recientemente por los usuarios. De esta forma, incrementa la rapidez de acceso a los servidores de información Web y FTP que se encuentra fuera de la red interna.

**Proxy para SSL** Squid también es compatible con SSL (Secure Socket Layer) con lo que también acelera las transacciones cifradas, y es capaz de ser configurado con amplios controles de acceso sobre las peticiones de usuarios.

**Jerarquías de caché** Squid puede formar parte de una jerarquía de caches. Diversos proxys trabajan conjuntamente sirviendo las peticiones de las páginas. Un navegador solicita siempre las páginas a un sólo proxy, si este no tiene la página en la caché hace peticiones a sus hermanos, que si tampoco las tienen las hacen a su/s padre/s... Estas peticiones se pueden hacer mediante dos protocolos: HTTP e ICMP.

**ICP, HTCP, CARP, caché digests** Squid sigue los protocolos ICP, HTCP, CARP y caché digests que tienen como objetivo permitir a un proxy "preguntarle.<sup>a</sup> otros proxys caché si poseen almacenado un recurso determinado.

**Caché transparente** Squid puede ser configurado para ser usado como proxy transparente de manera que las conexiones son enrutadas dentro del proxy sin configuración por parte del cliente, y habitualmente sin que el propio cliente conozca de su existencia. De modo predefinido Squid utiliza el puerto 3128 para atender peticiones, sin embargo se puede especificar que lo haga en cualquier otro puerto disponible o bien que lo haga en varios puertos disponibles a la vez.

**WCCP** A partir de la versión 2.3 Squid implementa WCCP (Web Cache Control Protocol). Permite interceptar y redirigir el tráfico que recibe un router hacia uno o más proxys caché, haciendo control de la conectividad de los mismos. Además permite que uno de los proxys caché designado pueda determinar como distribuir el tráfico redirigido a lo largo de todo el array de proxys caché.

**Control de acceso** Ofrece la posibilidad de establecer reglas de control de acceso. Esto permite establecer políticas de acceso en forma centralizada, simplificando la administración de una red.

**Aceleración de servidores HTTP** Cuando un usuario hace petición hacia un objeto en Internet, este es almacenado en el caché, si otro usuario hace petición hacia el mismo objeto, y este no ha sufrido modificación alguna desde que lo accedió el usuario anterior, Squid mostrará el que ya se encuentra en el caché en lugar de volver a descargarlo desde Internet. Esta función permite navegar rápidamente cuando los objetos ya están en el caché y además optimiza enormemente la utilización del ancho de banda.

**SNMP** Squid permite activar el protocolo SNMP, este proporciona un método simple de administración de red, que permite supervisar, analizar y comunicar información de estado entre una gran variedad de máquinas, pudiendo detectar problemas y proporcionar mensajes de estados.

**Caché de resolución DNS** Squid está compuesto también por el programa dnsserver, que se encarga de la búsqueda de nombres de dominio. Cuando Squid se ejecuta, produce un número configurable de procesos dnsserver, y cada uno de ellos realiza su propia búsqueda en DNS. De este modo, se reduce la cantidad de tiempo que la caché debe esperar a estas búsquedas DNS.

### 4.2.2. Compatibilidad

Squid puede ejecutarse en los siguientes Sistemas Operativos:

- AIX
- BSDI
- Digital Unix
- FreeBSD
- HP-UX
- IRIX
- GNU/Linux
- Mac OS X
- NetBSD
- NeXTStep
- OpenBSD
- SCO Unix
- SunOS/Solaris
- Windows NT

Para Windows se utilizan los paquetes Cygwin/GnuWin32. Obs.: Existen, actualmente, paquetes que corren squid en W32 sin necesidad de Cygwin.

## Capítulo 5

# Implementación

Para poder dar un ejemplo claro de uso de ambas herramientas en cuestión es pertinente explicar el proceso de como llevar a cabo la implantación de un firewall y el uso de squid como proxy.

Es necesario tener disponible 2 computadoras de las cuales una debe tener 2 interfaces de red físicas para poder llevar a cabo la instalación a describir más adelante.

Los puntos a tratar son:

1. La máquina con 2 interfaces de red será el gateway de una red privada 10.161.0.\*
2. Un hub que permitirá la conexión con la red privada.
3. Instalación de un servidor DHCP que atienda las peticiones de las máquinas dentro de la red privada.
4. Sólo se permitirán conexiones entrantes a los clientes de los puertos asociados a los servicios web, ssh y ftp.
5. Sólo se permitirán conexiones salientes de los clientes a los puertos asociados a los servicios web, ssh y ftp.
6. Instalación de un servidor squid en la máquina tomada como gateway el cual bloquea páginas no deseadas y de contenido delicado.
7. Generar estadísticas de páginas visitadas por los usuarios por medio de Squid.

## 5.1. Configuración de NAT y Firewall para la red privada

NAT (Network Address Translation - Traducción de Dirección de Red) es un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que se asignan mutuamente direcciones incompatibles. Consiste en convertir en tiempo real las direcciones utilizadas en los paquetes transportados. También es necesario editar los paquetes para permitir la operación de protocolos que incluyen información de direcciones dentro de la conversación del protocolo.

Su uso más común es permitir utilizar direcciones privadas (definidas en el RFC 1918) para acceder a Internet. Existen rangos de direcciones privadas que pueden usarse libremente y en la cantidad que se quiera dentro de una red privada. Si el número de direcciones privadas es muy grande puede usarse solo una parte de direcciones públicas para salir a Internet desde la red privada. De esta manera simultáneamente sólo pueden salir a Internet con una dirección IP tantos equipos como direcciones públicas se hayan contratado. Esto es necesario debido al progresivo agotamiento de las direcciones IPv4.

Para configurar un NAT para nuestra red privada lo podemos hacer mediante el uso de reglas con el comando iptables además de esto para que la configuración permanezca es necesario hacer un script que se ejecute cada vez que la maquina inicie; este script puede además tener reglas para el filtrado de paquetes y de esta manera configurar el firewall.

El script puede ser algo como lo siguiente:

```
#!/bin/sh

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos politica por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

```
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Empezamos a filtrar
## Nota: eth0 es el interfaz conectado al router y eth1 a la LAN
# El localhost se deja (por ejemplo conexiones locales a mysql)
/sbin/iptables -A INPUT -i lo -j ACCEPT

# Al firewall tenemos acceso desde la red local
iptables -A INPUT -s 10.161.0.0/24 -i eth1 -j ACCEPT

## Ahora con regla FORWARD filtramos el acceso de la red local
## al exterior. Como se explica antes, a los paquetes que no van dirigidos al
## propio firewall se les aplican reglas de FORWARD

##### ICMP rules (Internet Control Message Protocol)

iptables -A FORWARD -p ICMP -s 0/0 --icmp-type 0 -j ACCEPT
iptables -A FORWARD -p ICMP -s 0/0 --icmp-type 3 -j ACCEPT
iptables -A FORWARD -p ICMP -s 0/0 --icmp-type 5 -j ACCEPT
iptables -A FORWARD -p ICMP -s 0/0 --icmp-type 11 -j ACCEPT

# Aceptamos que vayan a puertos 80
iptables -A FORWARD -s 10.161.0.0/24 -i eth1 -p tcp --dport 80 -j ACCEPT
# Aceptamos que vayan a puertos https
iptables -A FORWARD -s 10.161.0.0/24 -i eth1 -p tcp --dport 443 -j ACCEPT

#Aceptamos el puerto ssh
iptables -A FORWARD -s 10.161.0.0/24 -i eth1 -p tcp --dport 22 -j ACCEPT

# Aceptamos que consulten los DNS
iptables -A FORWARD -s 10.161.0.0/24 -i eth1 -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -s 10.161.0.0/24 -i eth1 -p udp --dport 53 -j ACCEPT

# Y denegamos el resto. Si se necesita alguno, ya avisaran
```

```
iptables -A FORWARD -s 10.161.0.0/24 -i eth1 -j DROP

# Ahora hacemos enmascaramiento de la red local
# y activamos el BIT DE FORWARDING (imprescindible!!!!)
iptables -t nat -A POSTROUTING -s 10.161.0.0/24 -o eth0 -j MASQUERADE

# Con esto permitimos hacer forward de paquetes en el firewall, o sea
# que otras máquinas puedan salir a través del firewall.
echo 1 > /proc/sys/net/ipv4/ip_forward

## Y ahora cerramos los accesos indeseados del exterior:
# Nota: 0.0.0.0/0 significa: cualquier red

# Cerramos el rango de puerto bien conocido
iptables -A INPUT -s 0.0.0.0/0 -p tcp --dport 1:1024 -j DROP
iptables -A INPUT -s 0.0.0.0/0 -p udp --dport 1:1024 -j DROP

# Cerramos un puerto de gestión: webmin
iptables -A INPUT -s 0.0.0.0/0 -p tcp --dport 10000 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

Para poder hacer que se ejecute cada vez que se inicie la computadora es necesario crear un link simbólico desde `/etc/init.d/rc.iptables` a `/etc/rc3.d/S15iptables` de la siguiente manera:

```
ln -s /etc/init.d/rc.iptables /etc/rc3.d/S15iptables
```

## 5.2. **Instalación y configuración de servidor DHCP para la red privada**

El servidor DHCP será el encargado de proporcionar direcciones ip a máquinas que se conecten al hub de nuestra red privada, para esto es necesario la instalación del paquete `dhcp3-server` y luego

realizar una posterior configuración sobre el archivo `/etc/dhcp3/dhcpd.conf`.

Para nuestro caso es necesario que nuestro servidor dhcp proporcione direcciones ip a cualquier máquina que se conecte a nuestra red y haga una petición. Para esto realizamos la configuración de nuestro servidor de la siguiente manera:

```
ddns-update-style none;

option domain-name-servers 159.90.10.77;

default-lease-time 86400;
max-lease-time 604800;

authoritative;

subnet 10.161.0.0 netmask 255.255.255.0 {
    range 10.161.0.100 10.161.0.250;
    option subnet-mask 255.255.255.0;
    option broadcast-address 10.161.0.255;
    option routers 10.161.0.1;
    allow unknown-clients;
}
```

## 5.3. Instalación y configuración de Servidor Squid

Para nuestra instalación y configuración básica de nuestro servidor squid vamos a seguir los siguientes pasos:

### 1. Instalar el proxy

Para instalar Squid escribe en un terminal:

```
sudo aptitude install squid
```

### 2. Configurar el proxy

Para configurar el squid hay que editar el archivo `/etc/squid/squid.conf`

### 2.1 Nombrar el proxy

Squid necesita conocer el nombre de la máquina. Para ello, ubica la línea visible hostname.

```
visible_hostname dione
```

### 2.2 Elegir el puerto

Por defecto, el puerto de escucha del servidor proxy será 3128. Para elegir otro puerto, ubica la línea:

```
http_port 3128
```

### 2.3 Elegir la interfaz

Por defecto el servidor proxy escucha por todas las interfaces. Por razones de seguridad, sólo debes hacer que escuche en tu red local. Por ejemplo si la tarjeta de red ligada a tu LAN tiene el IP 10.0.0.1, modifica la línea a:

```
http_port 10.0.0.1:3177
```

### 2.4 Definir los derechos de acceso

Por defecto, nadie está autorizado a conectarse al servidor proxy, excepto dione. Entonces hay que crear una lista de autorización. Por ejemplo vamos a definir un grupo que abarca toda la red local.

Ubica la línea del archivo que comienza por acl localhost... Al final de la sección, agrega:

```
acl lanhome src 10.0.0.0/255.255.255.0
```

### Autorizar los puertos no estándar

Por defecto, Squid sólo autoriza el tráfico HTTP en algunos puertos (80, etc.) Esto puede ocasionar problemas a algunas páginas web que utilizan otros puertos. Ejemplo: `http://toto.com/:81/images/titi.png` sería bloqueado por Squid.

Para evitar que lo bloquee, encuentra la siguiente línea y comentamos:

```
http_access deny !Safe_ports
```

A partir de este momento todo estará configurado y solo es necesario que se reinicie el servicio. Para poder monitorear el comportamiento de squid podemos revisar los logs en `/var/log/squid/access.log`

### Modificar el tamaño del caché

Por defecto, el caché de Squid está activado, lo que permite que las páginas se carguen más rápido. El tamaño por defecto es de 100 MB (ubicado en `/var/spool/squid`).

Para cambiar su tamaño, modifica el archivo `/etc/squid/squid.conf` Encuentra la línea: `cache_dir ufs /var/spool`

Modifícala, puedes cambiar el valor de 100 por el valor que desees (por ejemplo 200 para 200 Mo): `cache_dir ufs /var/spool/squid 200 16 256`

Finalmente para poder hacer que las máquinas de la red privada se conecten al servidor squid es necesario agregar una regla más a nuestro script de iptables:

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

Ya hemos configurado nuestro servidor squid para que funcione como intermediario, ahora solo queda colocar reglas sobre páginas web que no queremos que nuestras máquinas de la red privada accedan para esto es necesario colocar un acl y negarle el acceso a ese acl y lo hacemos de la siguiente manera:

```
vi /etc/squid/prohibidas
```

En el fichero que se abre escribimos las palabras (una por línea). Por ejemplo:

- sex
- porn
- xxx
- megaupload
- rapidshare
- isohunt

Y cerramos el fichero guardando los cambios. De esta manera hemos creado un fichero en el directorio `/etc/squid` llamado `prohibidas`. Cuando terminemos de configurar Squid no podremos acceder a páginas con direcciones del tipo:

- `www.loqueseaxxx.com`
- `www.pornloquesea.com`
- `www.loqueseasex.com`

A continuación editamos el fichero de configuración de Squid escribiendo en la consola:

```
vim /etc/squid/squid.conf
```

 Y agregamos las líneas:

```
acl no_permitidos url_regex \/etc/squid/prohibidas"  
http_access deny no_permitidos
```

Luego reiniciamos nuestro servidor squid y todo quedará configurado. Para configuraciones más avanzadas de bloqueo de páginas y contenido se puede instalar y configurar un módulo de squid llamado Squidguard.

# Bibliografía

- [1] Wikipedia, la enciclopedia libre, <http://es.wikipedia.org/wiki/Netfilter/iptables>
- [2] Wikipedia, la enciclopedia libre, <http://es.wikipedia.org/wiki/Proxy>
- [3] <http://www.alejandrox.com/2008/05/filtra-contenidos-de-paginas-web-con-squid-y-squidguard/>