Netfilter/iptables

De Wikipedia, la enciclopedia libre

Netfilter es un framework disponible en el núcleo Linux que permite interceptar y manipular paquetes de red. Dicho framework permite realizar el manejo de paquetes en diferentes estados del procesamiento. Netfilter es también el nombre que recibe el proyecto que se encarga de ofrecer herramientas libres para cortafuegos basados en Linux.

El componente más popular construido sobre *Netfilter* es **iptables**, una herramienta de cortafuegos que permite no solamente filtrar paquetes, sino también realizar traducción de direcciones de red (NAT) para IPv4 o mantener registros de log. El proyecto Netfilter no sólo ofrece componentes disponibles como módulos del núcleo sino que también ofrece herramientas de espacio de usuario y librerías.

iptables es el nombre de la herramienta de espacio de usuario mediante la cual el administrador puede definir políticas de filtrado del tráfico que circula por la red. El nombre *iptables* se utiliza frecuentemente de forma errónea para referirse a toda la infraestructura ofrecida por el proyecto Netfilter. Sin embargo, el proyecto ofrece otros subsistemas independientes de iptables tales como el *connection tracking system* o sistema de seguimiento de conexiones, que permite encolar paquetes para que sean tratados desde espacio de usuario. *iptables* es un software disponible en prácticamente todas las distribuciones de Linux actuales.

Contenido

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 cortafuegos 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.

Resumen de operación

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 *objetivo* 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.

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.

- filter table (*Tabla de filtros*) — Esta tabla es la responsable del filtrado (es decir, de bloquear o permitir que un paquete continúe su camino). Todos los paquetes pasan a través de la tabla de filtros. Contiene las siguientes cadenas predefinidas y cualquier paquete pasará por una de ellas:
  - *INPUT chain* (Cadena de ENTRADA) — Todos los paquetes destinados a este sistema atraviesan esta cadena (y por esto se la llama algunas veces *LOCAL_INPUT* o *ENTRADA_LOCAL*)
  - *OUTPUT chain* (Cadena de SALIDA) — Todos los paquetes creados por este sistema atraviesan esta cadena (a la que también se la conoce como *LOCAL_OUTPUT* o *SALIDA_LOCAL*)
  - *FORWARD chain* (Cadena de REDIRECCIÓN) — Todos los paquetes que meramente pasan por este sistema para ser encaminados a su destino recorren esta cadena

- nat table (*Tabla de traducción de direcciones de red*) — Esta tabla es la responsable de configurar las reglas de reescritura de direcciones o de puertos de los paquetes. El primer paquete en cualquier conexión pasa a través de esta tabla; los veredictos determinan como van a reescribirse todos los paquetes de esa conexión. Contiene las siguientes cadenas redefinidas:
  - *PREROUTING chain* (Cadena de PRERUTEO) — Los paquetes entrantes pasan a través de esta cadena antes de que se consulte la tabla de ruteo local, principalmente para DNAT (*destination*-NAT o traducción de direcciones de red de destino)
  - *POSTROUTING chain* (Cadena de POSRUTEO) — Los paquetes salientes pasan por esta cadena después de haberse tomado la decisión del ruteo, principalmente para SNAT (*source*-NAT o traducción de direcciones de red de origen)
  - *OUTPUT chain* (Cadena de SALIDA) — Permite hacer un DNAT limitado en paquetes generados localmente

- mangle table (*Tabla de destrozo*) — Esta tabla es la responsable de ajustar las opciones de los paquetes, como por ejemplo la calidad de servicio. Todos los paquetes pasan por esta tabla. Debido a que está diseñada para efectos avanzados, contiene todas las cadenas predefinidas posibles:
  - *PREROUTING chain* (Cadena de PRERUTEO) — Todos los paquetes que logran entrar a este sistema, antes de que el ruteo decida si el paquete debe ser reenviado (cadena de REENVÍO) o si tiene destino local (cadena de ENTRADA)
  - *INPUT chain* (Cadena de ENTRADA) — Todos los paquetes destinados para este sistema pasan a través de esta cadena
  - *FORWARD chain* (Cadena de REDIRECCIÓN) — Todos los paquetes que exactamente pasan por este sistema pasan a través de esta cadena
  - *OUTPUT chain* (Cadena de SALIDA) — Todos los paquetes creados en este

sistema pasan a través de esta cadena
- *POSTROUTING chain* (Cadena de POSRUTEO) — Todos los paquetes que abandonan este sistema pasan a través de esta cadena

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).

Destinos de reglas

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).

Hay muchos destinos de extensión disponibles. 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

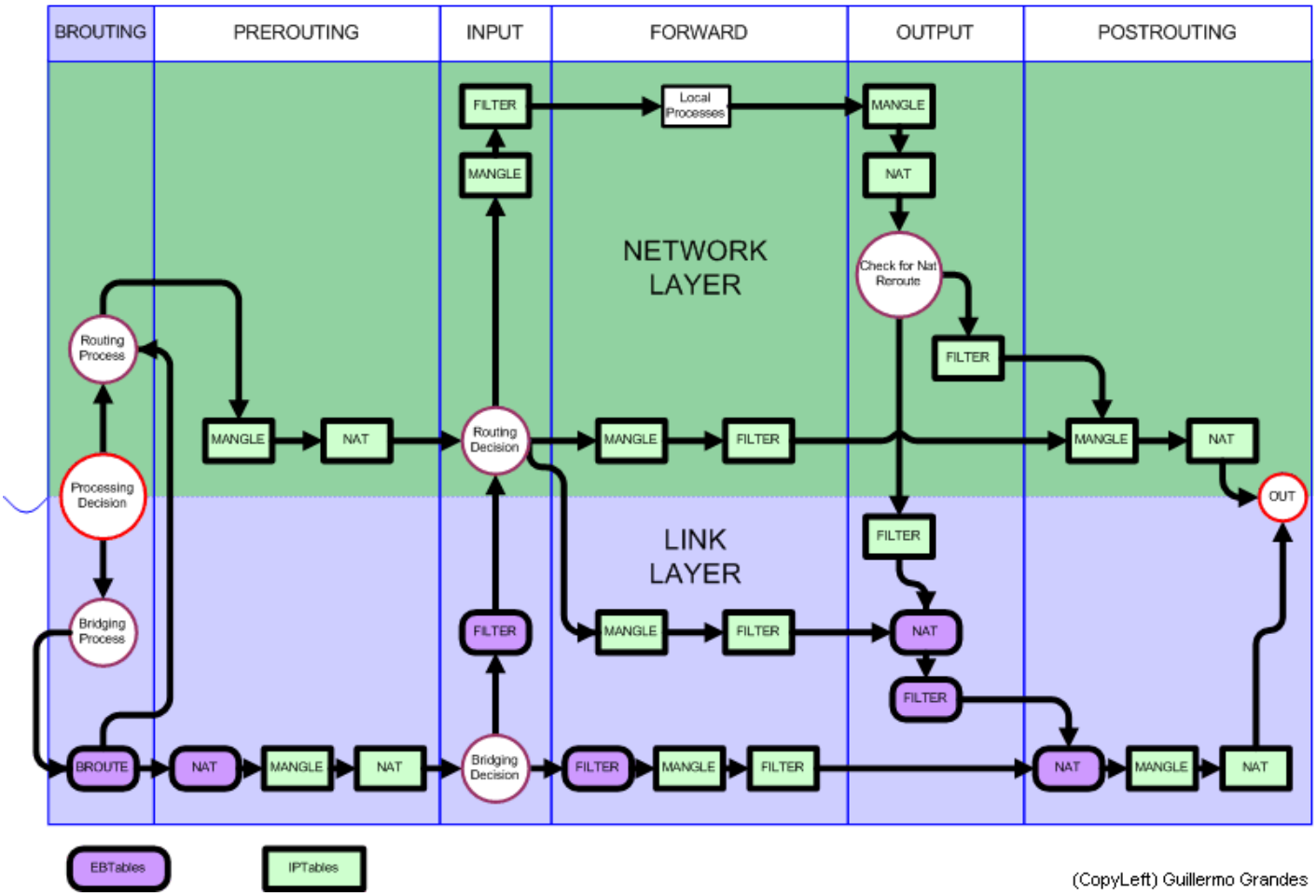
Este destino hace que la dirección (y opcionalmente el puerto) de origen del paquete sean reescritos para traducción de dirección de red. Mediante la opción '--to-source' debe indicarse el origen a usar. Esto es válido solamente en la cadena de POSRUTEO dentro de la tabla de nat y, como DNAT, se recuerda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

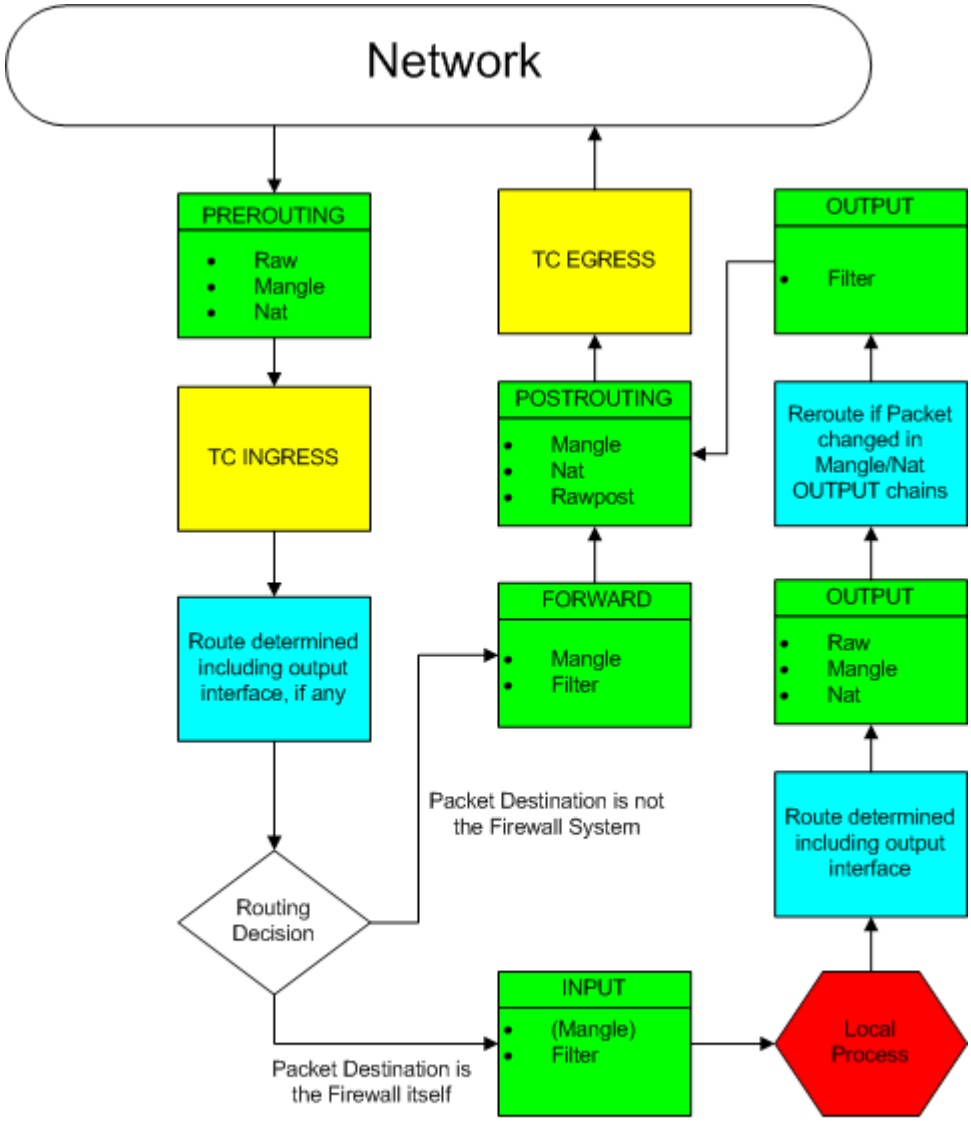
MASQUERADE

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.

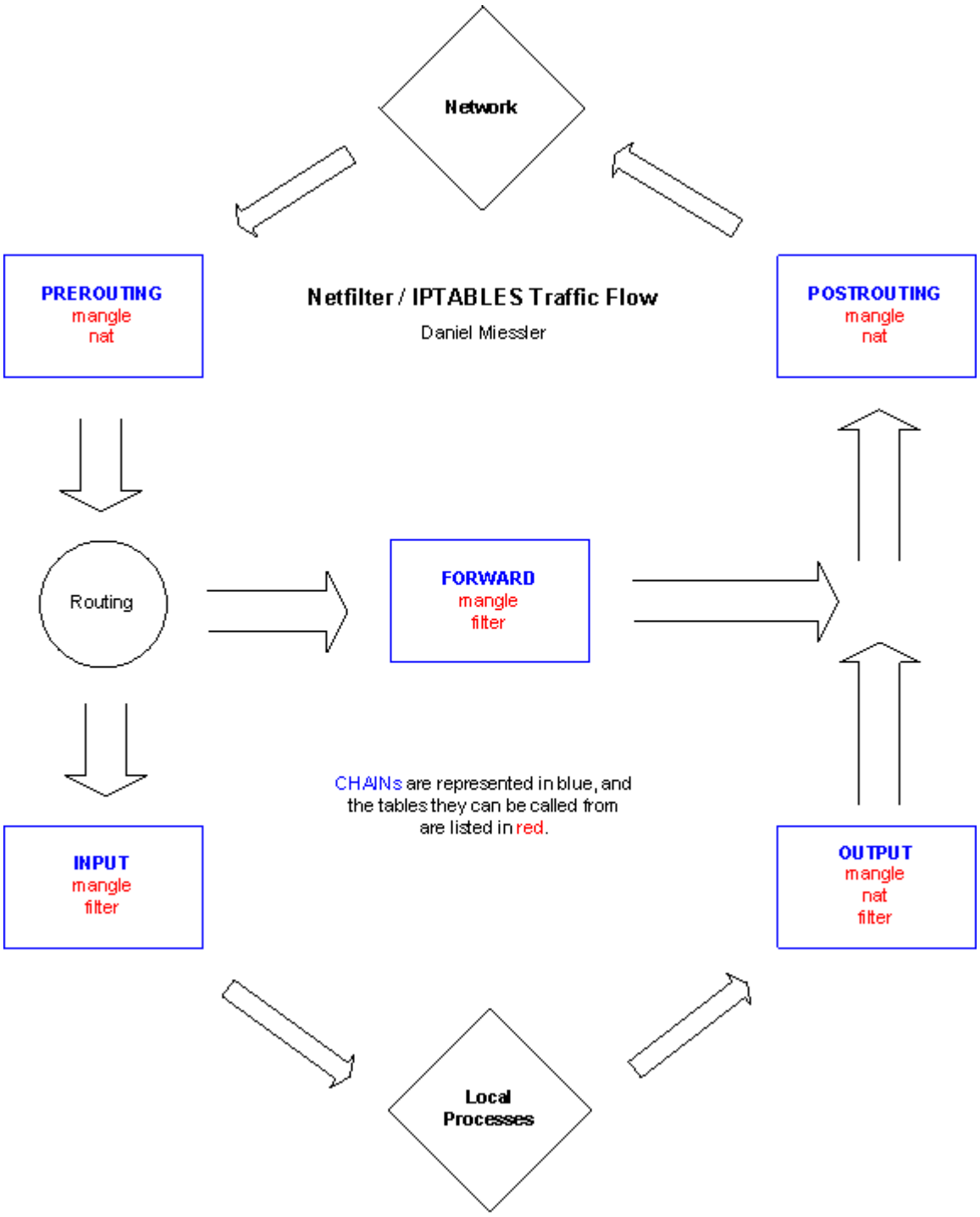
Diagramas

Los siguientes diagramas pueden resultar de utilidad para entender mejor como es que un paquete recorre las tablas y cadenas del núcleo de netfilter:





Netfilter Packet Flow


Network
PREROUTING
mangle
nat
Netfilter / IPTABLES Traffic Flow
Daniel Miessler
POSTROUTING
mangle
nat
Routing
FORWARD
mangle
filter
CHAINs are represented in blue, and the tables they can be called from are listed in red.
INPUT
mangle
filter
OUTPUT
mangle
nat
filter
Local Processes

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 seguimento 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.

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.

iptables

iptables es una aplicación en espacio de usuario que le permite a un administrador de sistema configurar las tablas, cadenas y reglas de netfilter (descritas más arriba). Debido a que iptables requiere privilegios elevados para operar, el único que puede ejecutarlo es el superusuario. 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 tecleando el comando "man iptables" desde la línea de comandos.

Opciones comunes

En cada una de las formas de invocación de iptables que se muestra a continuación, las siguientes opciones comunes están disponibles:

-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.

Especificaciones de las reglas

La mayoría de las formas de comandos de iptables requieren que se les indiquen una especificación de reglas, que es usada para comparar 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 comparados 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 comparado 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 comparada. Si esta opción se omite, se comparará 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 comparada. Si esta opción se omite, se comparará todo nombre de interfaz.

-p [!] *protocol*
--protocol [!] *protocol*
Compara paquetes del nombre de protocolo especificado. Si '!' precede el nombre de protocolo, se comparan 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*]
Compara 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 comparan todos los paquetes que no vienen del origen especificado.

-d [!] *destino*[/*prefijo*]
--destination [!] *destino*[/*prefijo*]
Compara 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.

Invocación

El comando iptables tiene las siguientes formas de invocación. Ítems entre llaves, {...|...|...}, son requeridos, pero solo se puede indicar uno de los ítems separados por '|'. Ítems entre corchetes, [...], son opcionales.

iptables { -A | --append | -D | --delete } *cadena especificación-de-regla* [*opciones*]

Esta forma de invocación del comando agrega (-A o --append) o elimina (-D o --delete) una regla de la cadena especificada. Por ejemplo, para agregar una regla a la cadena de INPUT en la tabla *filter* (la tabla por defecto cuando la opción -t no se especifica) que descarte todos los paquetes UDP, usamos este comando:

iptables -A INPUT -p udp -j DROP

Para borrar la regla agregada por el comando anterior, usamos este comando:

iptables -D INPUT -p udp -j DROP

El comando anterior borra en verdad la primera regla de la cadena de INPUT que coincida con la especificación de regla "-p udp -j DROP". Si hay varias reglas idénticas en la cadena, solo se borra la primera regla que concuerde.

iptables { -R | --replace | -I | --insert } *cadena numregla especificación-de-regla* [*opciones*]

Esta forma de invocación del comando reemplaza (-R o --replace) una regla existente o inserta (-I o --insert) una regla nueva en la cadena especificada. De hecho, para reemplazar la cuarta regla en la cadena de INPUT por una regla que descarte todos los paquetes ICMP, usamos este comando:

```
iptables -R INPUT 4 -p icmp -j DROP
```

Para insertar una regla nueva en el segundo lugar de la cadena de OUTPUT que descarte todo el tráfico TCP dirigido al puerto 80 en cualquier terminal, usamos este comando:

```
iptables -I OUTPUT 2 -p tcp --dport 80 -j DROP
```

iptables { -D | --delete } *cadena numregla* [*opciones*]

Esta forma de invocación del comando elimina una regla del índice numérico especificado en la cadena especificada. Las reglas se numeran comenzando desde 1. Por ejemplo, para eliminar la tercer regla de la cadena FORWARD, usamos este comando:

```
iptables -D FORWARD 3
```

iptables { -L | --list | -F | --flush | -Z | --zero } [*cadena*] [*opciones*]

Esta forma de invocación del comando se usa para listar las reglas en una cadena (-L o --list), tirar (es decir, eliminar) todas las reglas de una cadena (-F o --flush), o para poner en cero el byte y los contadores de paquetes de una cadena (-Z o --zero). Si no se especifica ninguna cadena, la operación se realiza para todas las cadenas. Por ejemplo, para listar las reglas en la cadena de OUTPUT, usamos este comando:

```
iptables -L OUTPUT
```

Para tirar todas las cadenas, usamos este comando:

```
iptables -F
```

Para poner en cero el byte y los contadores de paquetes de la cadena de PREROUTING en la tabla *nat*, usamos este comando:

```
iptables -t nat -Z PREROUTING
```

iptables { -N | --new-chain } *cadena*
iptables { -X | --delete-chain } [*cadena*]

Esta forma de invocación del comando se usa para crear (-N o --new-chain) una cadena definida por el usuario nueva o para eliminar (-X o --delete-chain) una cadena definida por el usuario existente. Si no se especifica ninguna cadena con las opciones -X o --delete-chain, se eliminan todas las cadenas definidas por el usuario. No es posible eliminar cadenas ya incorporadas, como ser las cadenas de INPUT o OUTPUT en la tabla *filter*.

iptables { -P | --policy } *cadena destino*

Esta forma de invocación del comando se usa para especificar la política de destino para una cadena. De hecho, para especificar la política de destino para la cadena de INPUT en DROP, usamos este comando:

```
iptables -P INPUT DROP
```

iptables { -E | --rename-chain } *nombre-de-cadena-viejo nombre-de-cadena-nuevo*

Esta forma de invocación del comando se usa para renombrar una cadena definida por el usuario.

Firewall con iptables

Para generar este *script* con *iptables*, se deben seguir los siguientes pasos:

1. Se crea un archivo de texto con un editor de texto plano cualquiera, como ser vi, emacs o nano, entre muchísimos otros. A este archivo se lo llama, por ejemplo, iptables_ipt
2. Se le da permiso de ejecución mediante el comando chmod u+x o también chmod 700 iptables_ipt.
3. Se ejecuta desde la línea de comandos ./iptables_ipt

```
#!/bin/bash

iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -m conntrack --ctstate NEW ! -i eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
iptables -A FORWARD -i eth1 -o eth1 -j REJECT
iptables -P INPUT DROP
iptables -P FORWARD DROP
echo "Firewall Activado"
```

Para este ejemplo, eth0 es la puerta local, y eth1 es la puerta de enlace a Internet. Puede además agregarse bloqueo de puertos específicos, como por ejemplo:

```
iptables -A INPUT -p tcp --dport 25 -j DROP # smtp filtrado

# Corroboramos con nmap los puertos y nos dirá algo parecido

$sudo nmap -sT "nuestra IP"

 * 22/tcp   filtered  ssh *
 * 25/tcp   filtered  smtp *
```

Filtrado de puertos

Para filtrar un puerto específico, como el 22 o 443, el código recomendable sería el siguiente:

```
sudo iptables -t filter -I INPUT -p tcp --dport 22 -j DROP
sudo iptables -t filter -I INPUT -p tcp --dport 443 -j DROP
```

Enlaces externos

- Wikimedia Commons alberga contenido multimedia sobre **Netfilter/iptables**. Commons

- (en inglés) Página oficial del proyecto Netfilter/iptables
- Links al tema en inglés
- Frontends (opcionales):
  - http://shorewall.net/ - Shorewall
  - http://firehol.sourceforge.net/ - FireHOL
  - http://www.fwbuilder.org/ - Firewall Builder (GUI)
  - http://www.fs-security.com/ - Firestarter (GUI)
  - http://dag.wieers.com/home-made/dwall/ - Dwall
  - http://www.solsoft.com/netfilterone - NetfilterOne (GUI)
  - http://kmyfirewall.sourceforge.net/ - KMyFirewall (GUI)
  - http://www.simonzone.com/software/guarddog/ - Guarddog (GUI)
  - http://weblog.tinixtech.com.ar - Programming Iptables.
  - http://www.hacktimes.com/detecci_n_control_y_contenci_n_de_tr_fico_innecesa/ - Detección, control y contención de tráfico innecesario con Iptables
  - http://www.tutorial-es.com/seguridad/sistema/39-seguridad-con-reglas-iptables-firewall - Seguridad con reglas Iptables
  - http://fedoraproject.org/wiki/SystemConfig/firewall - Systems config firewall incluido con Fedora y RedHat Enterprise Linux

Categorías:

- Software para Linux
- Software libre
- Seguridad informática

- Português
- Русский

-