1. Introducción: defensa en profundidad

El concepto de defensa en profundidad procede del ámbito militar y hace referencia a establecer múltiples línea de defensa, en vez de una única línea muy fuerte. La idea de este sistema es la de retrasar lo máximo posible el avance del enemigo.

Llevando este concepto al mundo de la informática, se crearán capas de defensa para evitar ataques directos a la información sensible de un entorno. Este sistema proporciona un mayor tiempo para defenderse a la vez de aumentar la probabilidad de detectar un ataque.

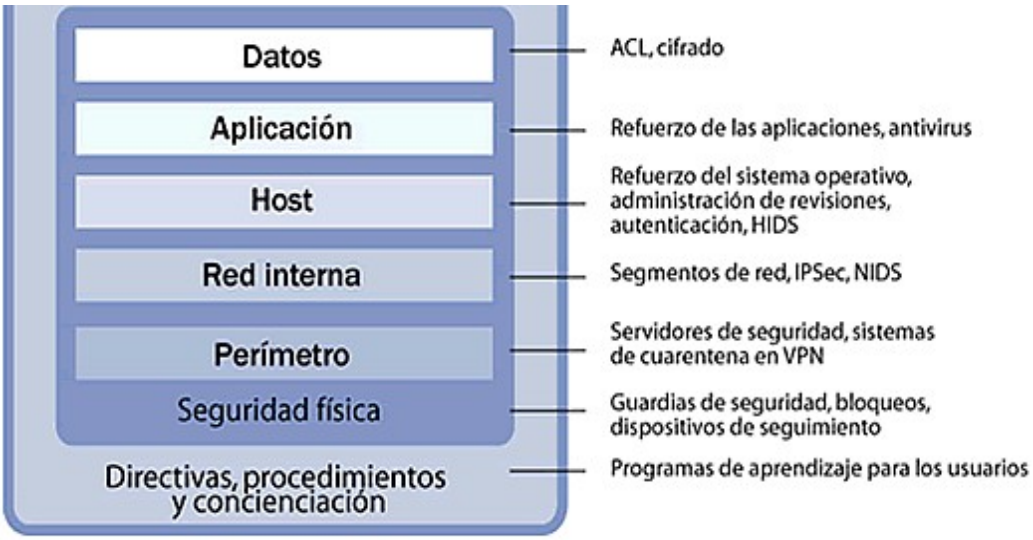


Fig. Seguridad en profundidad

Asegurar el perímetro es una de las estrategias defensivas más eficaces comúnmente utilizadas desde antaño en el campo de la seguridad. Baste para ello recordar las fortificaciones medievales en las que existía tan sólo un número restringido de puntos de entrada en los que se aplicaba un fuerte control de acceso. Reducir la superficie de exposición, crear puntos controlados de acceso y centrar las defensas en esos puntos permite optimizar la capacidad defensiva. Para reforzar la eficacia de la fortificación existían asimismo, elementos añadidos de seguridad como guardias, fosos, barreras naturales, puertas falsas o perímetros internos a la propia fortificación. También han sido muchas las puertas traseras o poternas construidas por comodidad, y que cuando no fueron debidamente mantenidas en secreto supusieron la caída de fortalezas.

El mundo de las tecnologías de la información y las comunicaciones (TIC) ha adoptado estas ideas a lo largo de los últimos años, creando arquitecturas y dispositivos que permitieran realizar las mismas tareas con idéntica eficacia. En el centro de estas tecnologías se encuentran los cortafuegos: puntos controlados de acceso.

2. Cortafuegos (Firewall)

Consiste en un dispositivo formado por uno o varios equipos que se sitúan entre la red de la empresa y la red exterior que analiza todos los paquetes que transitan entre ambas redes y filtran los que no deben ser reenviados, de acuerdo con algún criterio establecido de antemano.

Un Firewall trata de resolver los problemas de acceso seguro hacia/desde el exterior y para ello analiza los paquetes prestando atención a varios puntos:

- Procedencia: equipo o red de procedencia del paquete, interfaz por la que se recibe, etc.
- Destino: equipo o red de destino, interfaz por la que sale, puerto destino, etc.
- Protocolo empleado: IP, ICMP, TCP, UDP, etc.

- Contenido: tamaño del paquete, contenido de los datos de aplicación (url, palabras presentes en la página), etc.

En resumen, un firewall permite:

- Proteger y aislar las aplicaciones, servicios y máquinas de la red interna de tráfico entrante no deseado procedente de redes externas (Internet).
- Limitar o cortar el acceso de equipos de la red interna a servicios ofertados por redes externas a la organización (Internet).

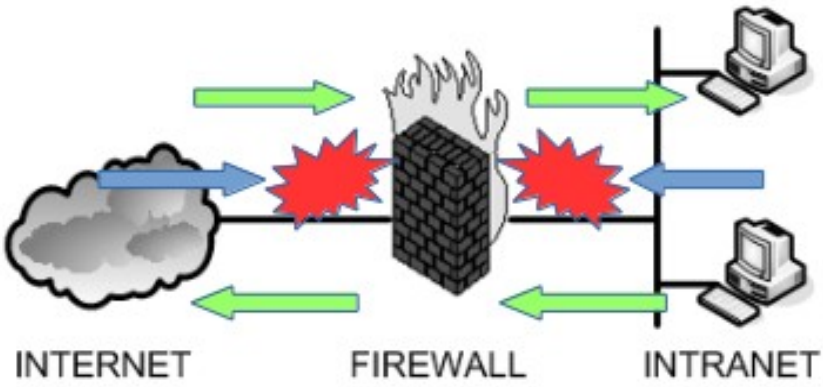


Fig. El firewall hace cumplir una política de control de acceso entre redes

Los cortafuegos no sólo se usan para separar Internet y la red interna de una organización, también se usan para separar entornos dentro de la organización que requieran niveles de seguridad diferentes. **La misión de un cortafuegos es la de hacer cumplir un política de control de acceso entre redes**; es decir, se permitirá o denegará el acceso a recursos de red a determinados clientes.

Estos privilegios de acceso estarán recogidos en una o varias reglas (***rules***), formando lo que se conoce como juego de reglas (***ruleset***).

	ID	Proto	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description
✖		*	*	*	192.168.0.255	*	*	none		no registrar broadcasts
▶		TCP	81.32.92.20	*	192.168.2.110	22 (SSH)	*	none		NAT ssh ubuntu server 10.04-2
▶		TCP	*	*	192.168.2.110	21 (FTP)	*	none		NAT ssh ubuntu server 10.04-2
▶		TCP	*	*	192.168.2.110	10000	*	none		NAT webmin ubuntu server 10.04-2

Fig. Conjunto de reglas (ruleset) de un firewall de red basado en pfSense

```
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     udp  --  8.8.8.8              0.0.0.0/0           udp spt:53
ACCEPT     udp  --  8.8.4.4              0.0.0.0/0           udp spt:53
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp spt:80
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp spt:443
```

Fig. Conjunto de reglas (ruleset) de un firewall de red basado en netfilter/iptables

3. Tipos de cortafuegos (Firewall)

Existen múltiples clasificaciones de cortafuegos en base a criterios como nivel de la pila TCP/IP en el que trabajan, ámbito de aplicación, funcionalidades, ...

En base al ámbito de aplicación:

- **Firewall de red:** controla el tráfico de la red permitiendo una gestión centralizada de la seguridad.
- **Firewall a nivel de sistema o de host:** instalado como aplicación local controlando el tráfico con origen o destino el equipo.

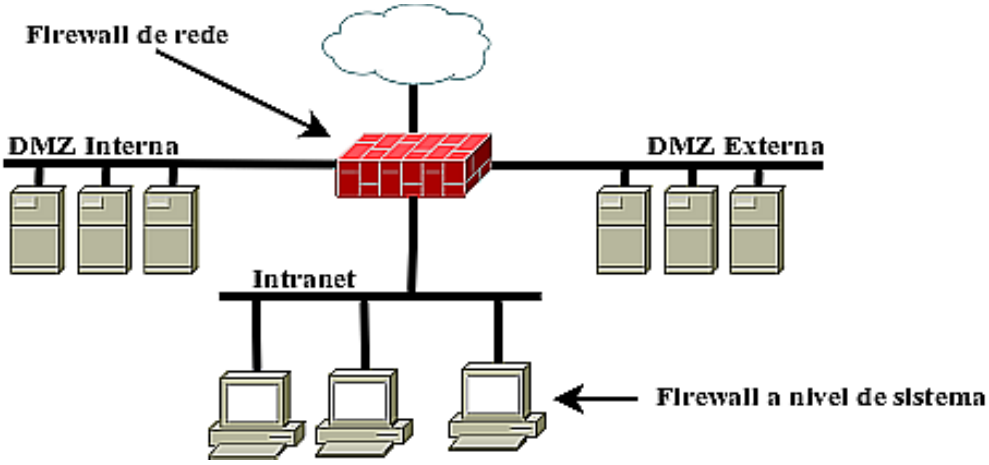


Fig. Ejemplo de ubicación de un firewall de red y de host (a nivel de sistema)

En base al nivel de la pila TCP/IP en la que trabajan:[1]

- **Network Level Firewalls:** similares a routers que trabajan fundamentalmente en los niveles de red y transporte. Se encargan de analizar los valores de las cabeceras de los datagramas IP, segmentos TCP y datagramas UDP y en base a una serie de reglas, establecidas por el administrador, deciden la acción a realizar con el paquete de datos: aceptarlo, descartarlo o reenviarlo. Además del filtrado, estos dispositivos suelen hacer NAT.
- **Application Level o Proxys:** trabajan en el nivel de aplicación actuando de intermediarios entre una red segura y una red no segura, aceptando peticiones de los clientes y actuando en su nombre ante los servidores. Entre sus labores típicas se encuentran:
  - El permitir (o no) las comunicaciones en función de:
    - Su origen y/o destino a nivel de aplicación.
    - El protocolo de nivel de aplicación usado.
    - La aplicación utilizada para comunicarse con el exterior (p.e. permitir firefox e I.E. y bloquear el resto de navegadores).
    - El contenido de los datos de aplicación (p.e. impedir el acceso a páginas deportivas, apuestas, pornográficas, redes sociales, ...).
  - La autenticación de los usuarios.

  Son considerados como los más seguros (al aislar completamente a los clientes) y los más inteligentes (al trabajar en la capa superior de la pila de protocolos). Sin embargo, estas características conllevan una mayor necesidad de recursos.

[1] Además de los indicados, también existen Firewalls de nivel 2 que filtran por MAC.

En la siguiente imagen se puede comparar el funcionamiento de un network level firewall y un firewall de nivel de aplicación. Con el network level firewall, el cliente contacta directamente con el servidor, saliendo a Internet a través del firewall (que será visto como un router por el cliente). Sin embargo, con el proxy el cliente no habla directamente con el servidor donde está la información que le interesa; el cliente contacta con su proxy e indica la información en la que está interesado, y es el proxy el que contacta con el servidor.

- Paquete 1: el proxy recibe una solicitud de un programa cliente (p. e., un cliente web como Firefox o IE que solicita http://www.edu.xunta.es).
- Paquete 2: el proxy reempaqueta la solicitud coma si fuese suya y la envía al servidor.
- Paquete 3: el servidor realiza el servicio solicitado y envía la información al proxy.
- Paquete 4: una vez obtenida la respuesta, el proxy la envía la cliente que la solicitó.

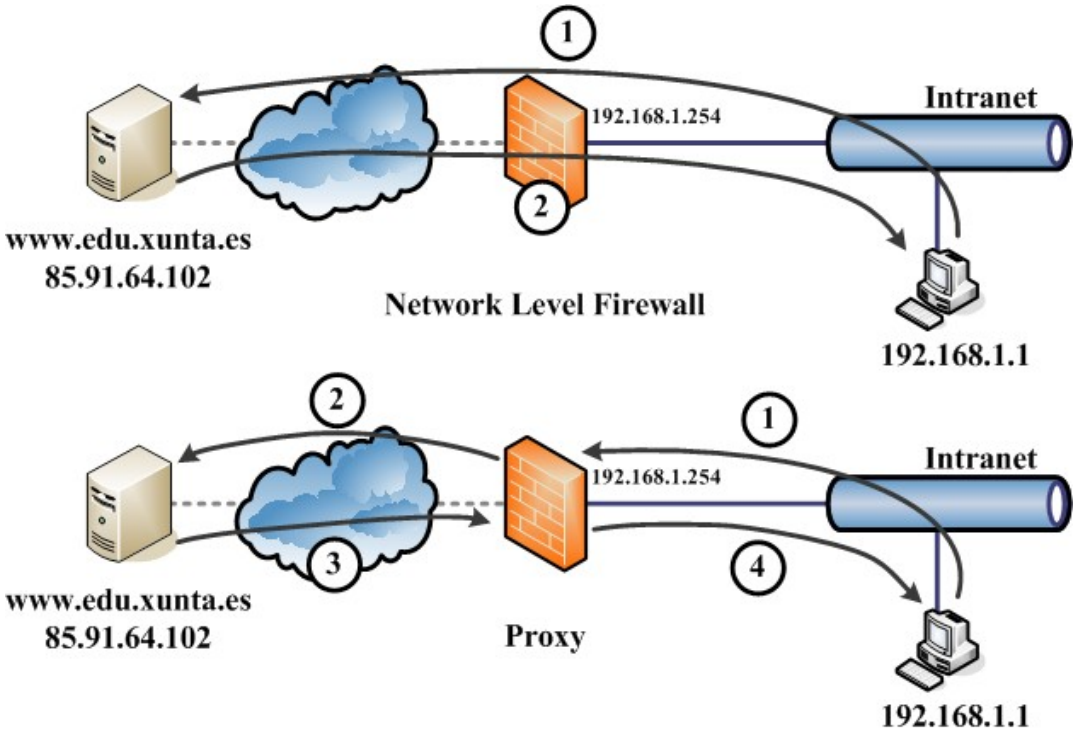


Fig. Comparación del funcionamiento entre network level firewall y firewall de nivel de aplicación

4. Netfilter e iptables: Firewall en LINUX.

Netfilter es un cortafuegos de estado presente en las versiones 2.4/2.6/3/4 del núcleo de Linux e Iptables es la aplicación que permite configurar Netfilter. Con ellas se puede hacer:

- Filtrado de paquetes en las capas 2 y 3 de la pila TCP/IP (packet filtering).
- Seguimiento de conexiones (connection tracking).
- Traducción de direcciones y puertos (NAT y NAPT).
- Manipulación de las cabeceras de los paquetes, etc.

Esto permite que con netfilter se puedan montar:

- Firewalls con filtrado de paquetes con o sin seguimiento de conexiones.
- Firewalls de host y de red.
- Routers-firewalls con NAT.
- Proxys transparentes usando NAT para redirigir las peticiones de los clientes.
- Sistemas de encaminamiento con calidad de servicio (QoS).
- Etc.

Netfilter está presente tanto en distribuciones Linux de carácter general, como Debian, Ubuntu, Red Hat como en distribuciones específicas de seguridad como Endian Firewall, BrazilFW y ZeroShell.

Para poder trabajar con Netfilter es necesario conocer su estructura interna; es decir, los elementos que la componen y su organización. Hay tres conceptos básicos: Reglas (rules), Cadenas (chains) y Tablas (tables).

REGLAS (rules):

Cada regla de iptables es una **línea que Netfilter comprueba para saber que hacer con un paquete**. Hay que ver las reglas en netfilter/iptables como líneas formadas por condiciones (*match*) y una acción. Si todas las condiciones que contiene la regla se cumplen, Netfilter ejecuta la acción especificada sobre el paquete en cuestión. La estructura básica de una regla iptables es:

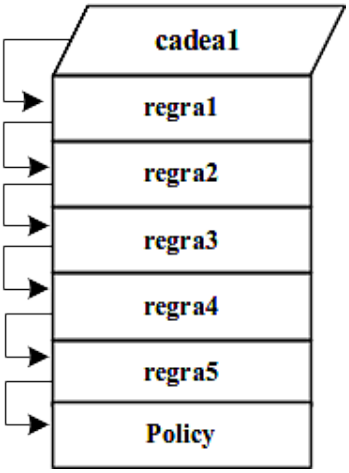
```
iptables [-t tábla] comando [condición] [-j acción]
```

- `-t` permite especificar la tabla en la que actuar (filter por defecto)
- **comando**: indica qué facer coa regra (engadir, insertar, borrar, …)
- **condición** *(matches):* requisitos que debe cumplir el paquete (protocolo, ip origen/destino, puerto origen/destino, interface, …) A modo de ejemplo:
  - `-p udp --dport 53` → indica que el paquete a nivel de transporte debe usar el protocolo UDP e ir dirigido al puerto destino el 53.
  - `-i eth0 -s 192.168.0.0/24` → indica que el paquete deber tener origen un equipo de la red 192.168.0.0/24 y ser recibido por la interfaz eth0.
- **acción**: indica a Netfilter qué hacer con el paquete si éste cumple todas las condiciones de la regla. A modo de ejemplo:
  - ACCEPT: el paquete se acepta.
  - DROP: el paquete se descarta de forma silenciosa.
  - REJECT: el paquete se descarta informando al emisor que su paquete se ha descartado.
  - LOG: se registra información sobre el paquete filtrado.
  - DNAT: hace NAT modificando la dirección (y/o puerto) destino del paquete.
  - SNAT: hace NAT modificando la dirección (y/o puerto) origen del paquete.
  - REDIRECT: redirección de puertos en la misma máquina.
  - Se puede no indicar ninguna acción en la cadena. En este caso se actualizan los contadores de paquetes y bytes de la regla e inmediatamente después se pasa a la siguiente regla de la cadena.

CADENAS (chains):

Una cadena es una **lista ordenada de reglas**. Para cada paquete se va comprobando si se le aplica cada regla de la cadena; es decir, si se cumple la condición:

- Si una regla no se aplica a un paquete, se pasa a la siguiente regla de la cadena.
- Si una regla sí se aplica a un paquete, se ejecuta la acción definida en dicha regla y (salvo excepciones) ya no se comprobarán más reglas de la cadena.




```
Chain INPUT (policy DROP)
num  target   prot opt source    destination

1    ACCEPT   udp  --  8.8.8.8     0.0.0.0/0    udp spt:53
2    ACCEPT   udp  --  8.8.4.4     0.0.0.0/0    udp spt:53
3    ACCEPT   tcp  --  0.0.0.0/0   0.0.0.0/0    tcp dpt:22
4    ACCEPT   tcp  --  0.0.0.0/0   0.0.0.0/0    tcp spt:80
5    ACCEPT   tcp  --  0.0.0.0/0   0.0.0.0/0    tcp spt:443
```

Una cadena puede tener definida una política (**policy**), que es la acción por defecto para la cadena. Cuando para un paquete no se aplica ninguna las reglas de la cadena, se ejecuta para él la política de la cadena.

Existen dos tipos de cadenas:

Cadenas predefinidas:

A un paquete al llegar a una maquina se le aplican las reglas de las cadenas predeterminadas en distintos momentos según el siguiente esquema:

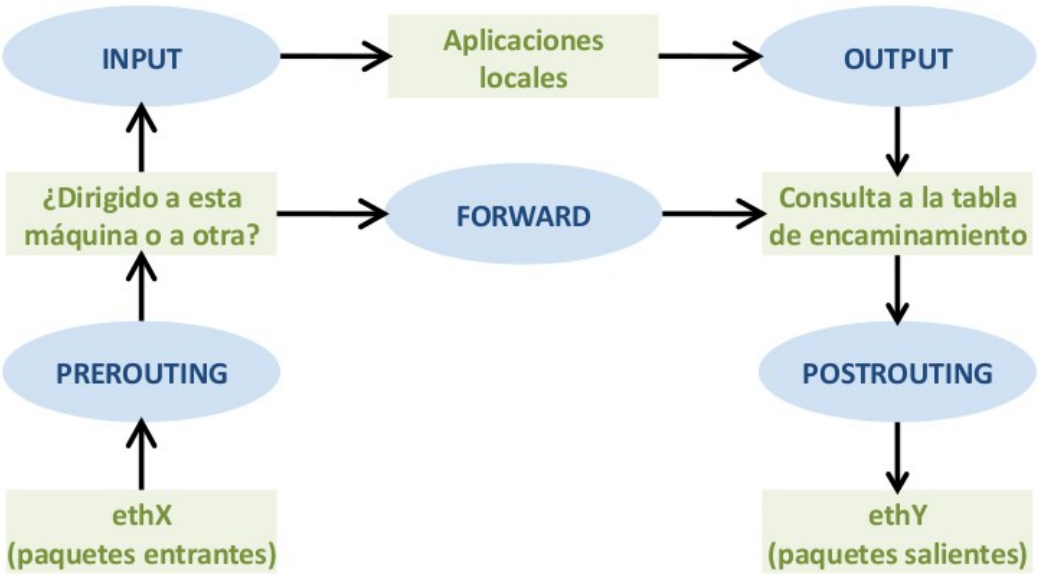


Fig. Cadenas predeterminadas en netfilter.

Las cadenas pueden interpretarse como puntos donde el sistema de filtrado puede manipular los paquetes. Existen las siguientes cadenas predefinidas:

- Cadena PREROUTING: reglas que se aplican a los paquetes que llegan a la máquina. Esta cadena se ejecuta antes de comprobar si el paquete es para la propia maquina o hay que reenviarlo.
- Cadena INPUT: reglas que se aplican a los paquetes destinados a la propia máquina. Esta cadena se ejecuta justo antes de entregarlos a la aplicación local.
- Cadena FORWARD: reglas que se aplican a los paquetes que han llegado a la máquina pero van destinados a otra y hay que reenviarlos. Esta cadena se ejecuta antes de consultar la tabla de encaminamiento.
- Cadena OUTPUT: reglas que se aplican a los paquetes creados por la propia máquina. Esta cadena se ejecuta justo después de que la aplicación le pase los datos a enviar al kernel del sistema operativo y antes de consultar la tabla de encaminamiento.

- Cadena POSTROUTING: reglas que se aplican a los paquetes que salen de la máquina, tanto los creados por ella como los que se reenvían. Esta cadena se ejecuta después de consultar la tabla de encaminamiento.

Únicamente las acciones ACCEPT y DROP pueden ser usadas cómo política en las cadenas predefinidas; siendo ACCEPT, la por defecto.

Cadenas definidas por el usuario:

Si un paquete está siendo procesado en una cadena, por ejemplo OUTPUT, es posible indicar una acción en una regla para hacer saltar al paquete la una cadena diferente. El paquete será procesado en la nueva cadena; regla por regla, hasta que cumpla una de ellas o llegue al final y vuelva la cadena original. Es muy recomendable dividir conjuntos de reglas complejas en cadenas separadas para facilitar la creación y gestión del firewall. Un ejemplo de uso de las cadenas de usuario sería una cadena que contenga las reglas para controlar el tráfico de administración remota del equipo.

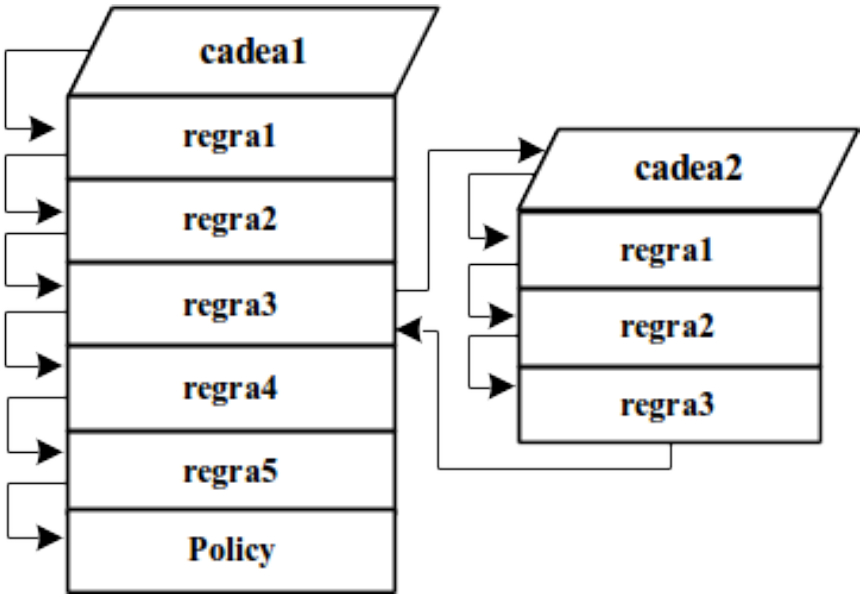


Fig. Flujo del procesamiento con salto a una cadena definida por el usuario

Todas las cadenas de usuario tienen como política la acción RETURN, que devuelve el paquete a la cadena origen. Esta política no puede ser cambiada; sin embargo, nada nos impide crear y añadir reglas a la cadena para establecer acciones para todos los paquetes que atraviesen la cadena; es decir, estaríamos creando nosotros una politica a mano. Por ejemplo, en la Cadena#2 de la figura anterior la regla #3 podría ser una regla que descarte todos los paquetes con DROP; es decir, todos los paquetes derivados a la Cadena#2 y que no sean explicítamente aceptados por las reglas #1 y #2, serán descartados sin volver a la Cadena#1. Creamos una política por defecto a mano para la Cadena#2.

TABLAS (tables):

Una tabla de iptables contiene un conjunto de cadenas, tanto predefinidas como de usuario. Una tabla concreta engloba las reglas (agrupadas en cadenas) relacionadas con un tipo de procesamiento de los paquetes. Netfilter define las siguientes tablas:

- **filter:** engloba las reglas de filtrado de paquetes; es decir, las que deciden que un paquete continúe su camino o sea descartado.
- **nat:** engloba las reglas de modificación de direcciones IP y puertos (NAT y NAPT).

- **mangle:** engloba las reglas de modificación de algunos campos de las cabeceras de los paquetes (TTL, ToS) y además permite marcarlos para que otros programas puedan reconocerlos (p.e. para proporcionar QoS).
- **raw:** engloba las reglas que permiten marcar excepciones al seguimiento que hace el kernel de las conexiones de la máquina de las cabeceras del paquete.

Tablas y cadenas

La **tabla filter** incluye las cadenas:
- FORWARD
- INPUT
- OUTPUT

La **tabla nat** incluye las cadenas:
- PREROUTING
- OUTPUT
- POSTROUTING

La **tabla mangle** incluye las cadenas:
- PREROUTING
- FORWARD
- INPUT
- OUTPUT
- POSTROUTING

La **tabla raw** incluye las cadenas:
- PREROUTING
- OUTPUT

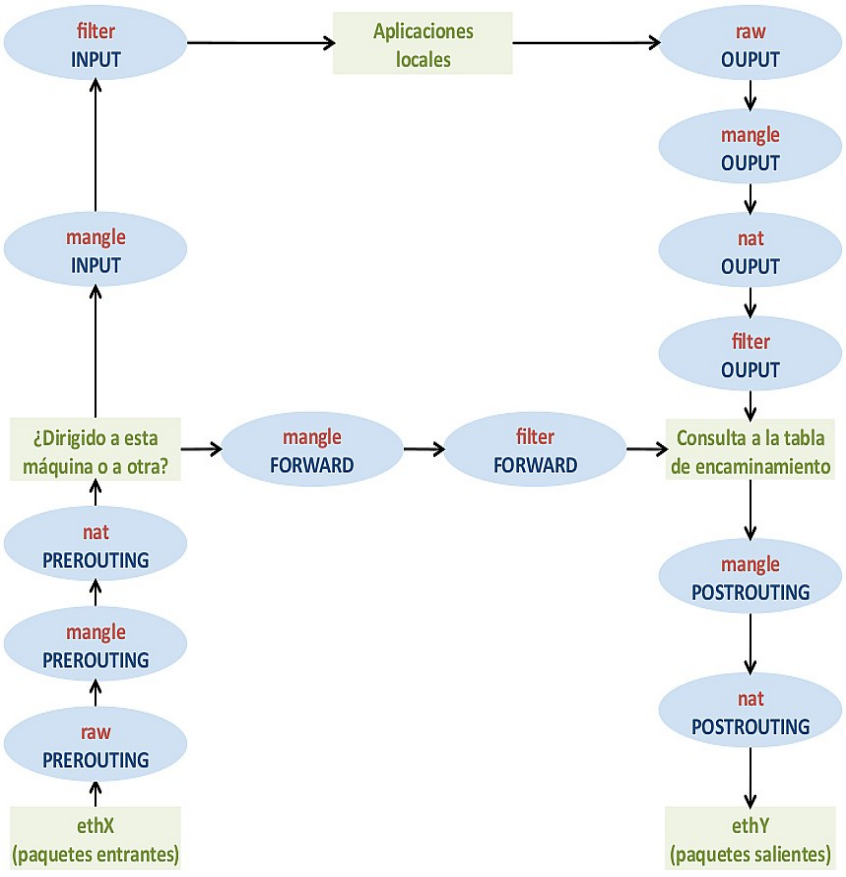


Fig. Tablas y cadenas predeterminadas en netfilter

En la anterior imagen puede verse la relación entre cadenas y tablas en netfilter. Además, la imagen permite tener una visión global de la ruta de los paquetes y de las posibilidades de procesamiento que proporciona netfilter en cada momento. Como ya se comentó, netfilter permite trabajar con los paquetes en diferentes puntos; por ejemplo, sobre un paquete recién llegado (PREROUTING) puede aplicarse NAT y cambiar la dirección IP destino para reenvialo la otro equipo.

Aunque la imagen anterior da una visión completa del esquema de funcionamiento de netfilter, se suele a trabajar con una versión más simplificada que deja de lado los procesamientos más avanzados (raw y mangle). Es muy importante tener presente estas imágenes al trabajar con netfilter/iptables para asociar la sintaxis de los comandos iptables con la secuencia de procesamiento de netfilter:

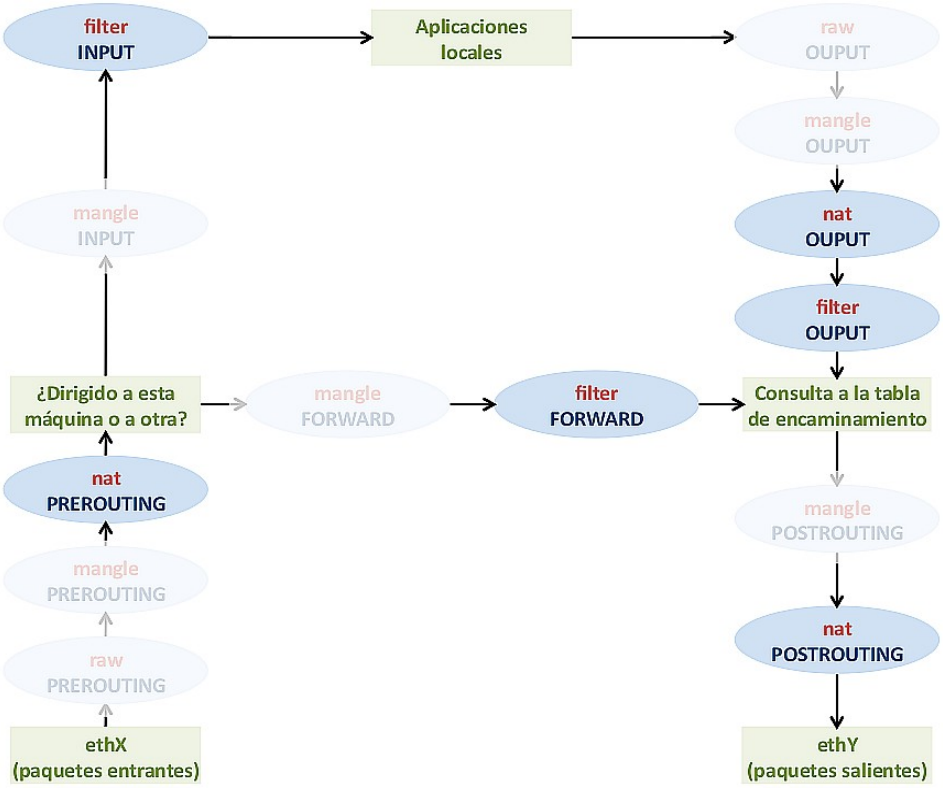
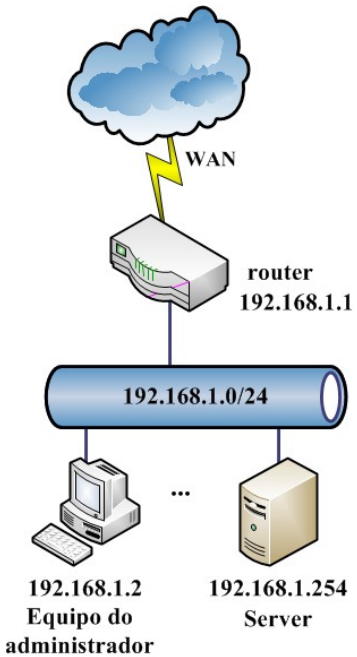


Fig. Tablas y cadenas predeterminadas más usadas en netfilter

Como ejemplo, se quiere implantar la siguiente política de tráfico para el server de la figura con netfilter/iptables:

- Podrá realizar consultas DNS a 8.8.8.8 y 8.8.4.4.
- Podrá visitar sitios web http/https.
- Podrá ser gestionado por ssh desde el equipo administrador.



En Ubuntu Server la configuración por defecto de netfilter es permitir todo el tráfico, tanto entrante como saliente. Esta configuración de permitir todo por defecto puede verificarse haciendo un listado de las reglas de la tabla filter:

```
$ sudo iptables -t filter -L
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
```

CleanUp Rules

La opción de 'permitir por defecto' es la más cómoda y viene activada por defecto en muchos firewalls y routers, pero no es la más segura. Obliga al administrador a crear regras para bloquear el tráfico no

deseado; y en caso de olvidarse de crear la regla correspondiente para bloquear un tipo de tráfico, este tráfico no deseado atravesará el firewall.

La política de 'denegar por defecto', donde se permite únicamente el tráfico requerido por las necesidades de la red y se rechaza el resto es más segura. Todo tráfico no autorizado en las reglas será eliminado; por lo que, para permitirlo habría que contactar con el administrador del firewall y que éste procediese a autorizarlo. Las reglas de bloquear todo por defecto se conocen como CleanUp Rules. En ipables podemos crearlas con:

```
$ sudo iptables -P INPUT DROP
$ sudo iptables -P OUTPUT DROP
$ sudo iptables -P FORWARD DROP
$ sudo iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
Chain FORWARD (policy DROP)
target     prot opt source               destination
Chain OUTPUT (policy DROP)
target     prot opt source               destination
```

Autorizar tráfico deseado

Para permitir la resolución DNS hay que:

- Permitir el envío de consultas dns: permitir que salgan de server los paquetes con destino el puerto 53/udp de los servidores autorizados.

```
$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.8.8 -j ACCEPT
$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.4.4 -j ACCEPT
```

  Estas regras hay que añadirlas en la cadena OUTPUT de la tabla filter por que en la cadena OUTPUT es donde están las reglas que se aplican a los paquetes creados por la propia máquina.

- Permitir entradas de las respuestas dns: aceptar los paquetes con destino server procedentes de los servidores autorizados y procedentes del puerrto 53/udp.

```
$ sudo iptables -A INPUT -p udp --sport 53 -s 8.8.8.8 -j ACCEPT
$ sudo iptables -A INPUT -p udp --sport 53 -s 8.8.4.4 -j ACCEPT
```

  Para permitir la entrada de las respuestas se añaden las reglas correspondientes en la cadena INPUT; ya que, en esta cadena están las reglas que se aplican a los paquetes destinados a la propia máquina.

Para permitir la navegación web hay que autorizar las conexiones con los puertos tcp 80 y 443:

```
$ sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --dport 443 -j ACCEPT
$ sudo iptables -A INPUT -p tcp --sport 80 -j ACCEPT
$ sudo iptables -A INPUT -p tcp --sport 443 -j ACCEPT
```

Otra posibilidad es usar la opcion `multiport` que permite definir varios puertos destino/origen en una misma regla:

```
$ sudo iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
$ sudo iptables -A INPUT -p tcp -m multiport --sports 80,443 -j ACCEPT
```

Acceso al servidor ssh:

```
$ sudo iptables -A INPUT -p tcp --dport 22 -s 192.168.1.2 -j ACCEPT
$ sudo iptables -A OUTPUT -p tcp --sport 22 -d 192.168.1.2 -j ACCEPT
```

DROP vs REJECT

En netfilter/iptables cuando decartamos paquetes podemos optar por DROP o REJECT:

- DROP proporciona un descarte silencioso descartanado el paquete sin informar al remitente. Esto hace que el emisor del paquete quede a la espera de recibir respuesta del destinatario hasta agotar los temporizadores y descartar la conexión por timeout.

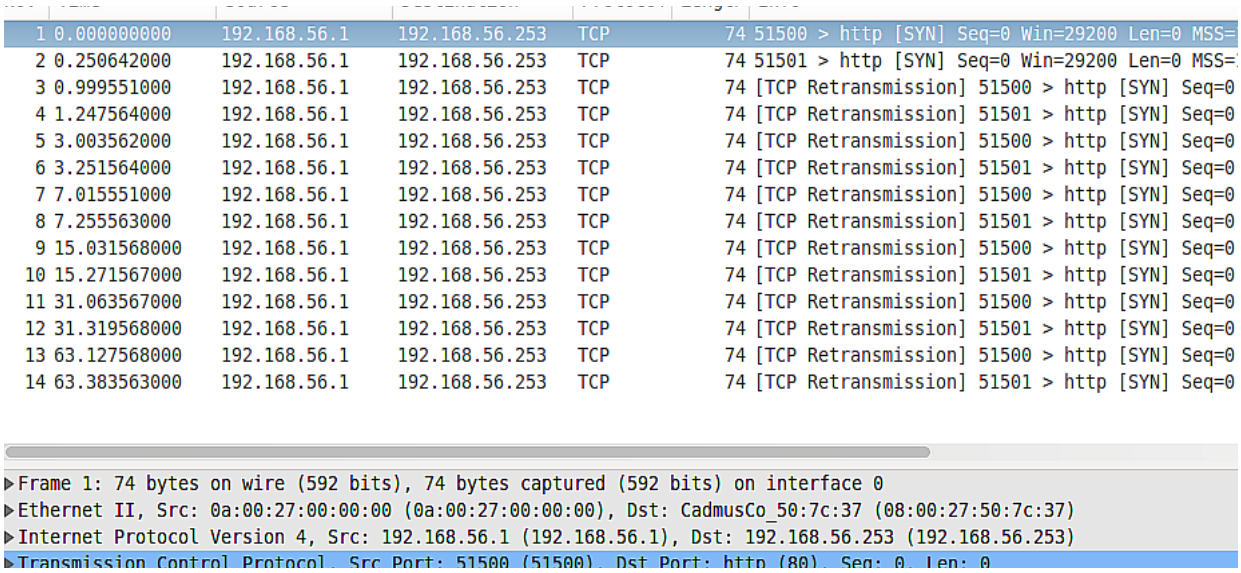


Fig. Descarte silencioso con DROP

- REJECT proporciona un descarte informado; es decir, descarte el paquete informando al remitente. La forma de informar consiste en el envío de un mensaje de error que permite al emisor darse por enterado de la situación y abortar la conexión sin esperar el timeout.

De usar REJECT, podemos especificar el tipo de mensaje de error usando `--reject-with type` donde `type` puede ser cualquiera de los siguientes (ver `man iptables-extensions`): `icmp-net-unreachable`, `icmp-host-unreachable`, `icmp-port-unreachable` (opción por defecto), `icmp-proto-unreachable`, `icmp-net-prohibited`, `icmp-host-prohibited`, `icmp-admin-prohibited`, `tcp-reset` (opción válida sólo con la opción `-p tcp`).

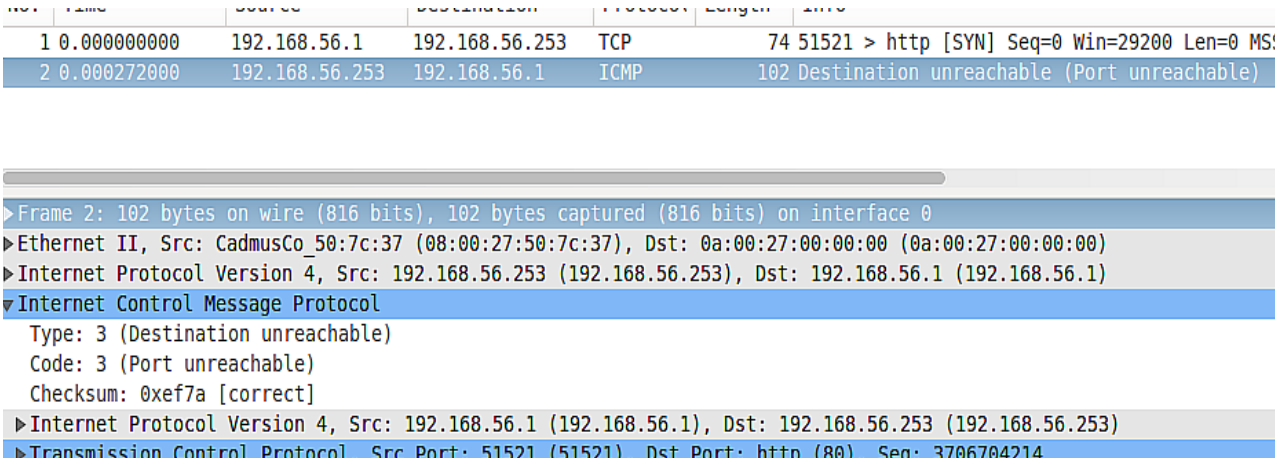


Fig. Descarte informado con REJECT

Dende el punto de vista de un cliente, DROP hace que el usuario tenga que esperar cierto tiempo hasta que la conexión se da por perdida (seguramente tras varios intentos de conexión) y con REJECT no. Usar REJECT durante la fase de pruebas del ruleset de un firewall ahorra tiempo al no esperar por los timeout; y después de verificado el funcionamiento, siempre se puede cambiar a DROP.

Un punto a tener en cuenta es que con DROP el firewall permanece oculto y con REJECT el firewall se descubre al enviar un mensaje de error con su IP. De todas formas, no por estar escondido un equipo va ser más seguro, pero tampoco hay ir anunciando su presencia.

5. Stateless y Stateful firewalls

Los network level firewalls se clasifican en dos grandes subtipos:

Static/Stateless Packet Filtering:

- Cada paquete se analiza de forma individual e independiente en base a una lista de control de acceso sobre la información de cabecera de las capas de red y transporte (dir. IP origen/destino, puertos TCP/UDP, flags TCP, …).
- No memoriza el estado de las conexiones a nivel de transporte permitidas con anterioridad. En el ejemplo anterior, la regla que permitía conexiones al puerto tcp/80, permitirá cualquier paquete tcp con puerto destino el 80, con independencia de que sea el primer paquete de una conexión tcp o el último, un paquete de datos o un paquete sin ningún sentido pero que cumple con el requisito de puerto destino tcp/80.
- Un ejemplo de este tipo de firewalls es un router con funciones de filtrado básicas.

Stateful/Dynamic Packet Filtering:

- Mejora al stateless por que manteniene información de estado de las conexiones establecidas. La información de estado de las comunicaciones se guarda en una **tabla de estado** que permite tratar a los paquetes no de forma individual, si no como paquetes participantes de una conexión existente o como paquetes iniciadores de una nueva comunicación.
- Gracias a la tabla de estado, el firewall puede conocer la historia de un paquete. Las entradas de la tabla de estado se borran al cerrarse la conexión o pasado un tiempo de inactividad.

Proto	Source -> Router -> Destination	State
icmp	192.168.0.253:12383 -> 192.168.0.100	0:0
tcp	192.168.1.253:443 <- 192.168.1.1:37048	ESTABLISHED:ESTABLISHED
udp	192.168.1.253:53 <- 192.168.1.1:54137	SINGLE:MULTIPLE
udp	192.168.0.253:38080 -> 80.58.32.97:53	MULTIPLE:SINGLE

Fig. Tabla de estado en un firewall pfSense

- Las reglas son 'dinámicas' y cambian en función de la situación. Por ejemplo, este tipo de firewall reconocerá todos los paquetes procedentes de equipos de la red de la empresa y únicamente permitirá tráfico del exterior que sea respuesta a peticiones internas.
- Ejemplos de stateful packet filtering son pfSense y netfilter/iptables. Como netfilter forma parte del kernel de Linux, todos los Linux pueden funcionar como un firewall de estado, tanto las distribuciones de carácter general (Ubuntu, Debian, Suse, …) como las específicas para ejercer de firewall (Endian, IPFire, IPCop, Zeroshell, …).

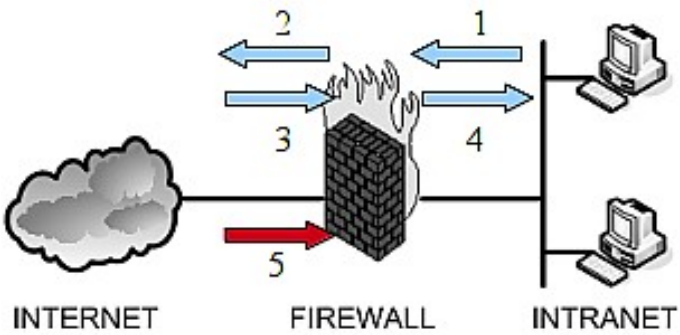


Fig. Ejemplo de funcionamiento de un firewall stateful

1. El equipo A lanza una petición para ver una página web en un servidor de Internet.
2. El firewall deja pasar el paquete en base a las reglas y además guarda en su tabla de estado toda la información relativa a esta conexión tcp.
3. El firewall recibe una respuesta del servidor a la solicitud enviada por el equipo A.
4. Gracias a la información guardada en la tabla de estado, el firewall reconoce este paquete como un paquete legítimo perteneciente a la conexión anteriormente autorizada y lo deja pasar hacia el interior de la organización. Así seguirá haciendo con los paquetes de esta conexión autorizada (tanto los salientes como los entrantes).
5. El firewall recibe un paquete desde Internet para abrir una conexión con el equipo A de la Intranet. Como no hay reglas que lo dejen entrar y no está asociado en la tabla de estado a alguna conexión conocida y autorizada previamente, el firewall no lo deja pasar.

6. Seguimiento de conexiones (Connection Tracking) en netfilter/iptables

El seguimiento de conexiones permite a Netfilter saber en que estado se encuentra una conexión determinada[2]. Dependiendo del protocolo al que pertenecen, los paquetes de comunicaciones pueden estar en diferentes estados dentro del núcleo, por lo que para simplificar el tratamiento de los estados de los paquetes en Netfilter se definen 5 estados genéricos que engloban los estados particulares de todos los protocolos y a los que podremos referirnos con iptables, usando la condición `--ctstate`, para filtrar paquetes basándonos en su estado.

- **NEW**: Primer paquete de una conexión.
- **ESTABLISHED**: Señala una conexión ya establecida. El único requisito para llegar a este estado es que tras el envío de un paquete, el sistema haya recibido una respuesta del destinatario.
- **RELATED**: Una conexión se considera RELATED cuando está relacionada con una conexión ya establecida (en estado ESTABLISHED). Esto es, primero es necesario tener una conexión ya establecida que lanza una nueva conexión, que es considerada como RELATED. Un ejemplo de conexiones RELATED son las conexiones de datos FTP que son lanzadas por las conexiones de control FTP para la transferencia de archivos, o los mensajes de error ICMP.
- **INVALID**: Cuando un paquete no puede ser identificado o no se encuentra en ningún estado se le considera en estado INVALID. Se recomienda descartar todos los paquetes que lleguen en este estado al cortafuegos.
- **UNTRACKED**: Un paquete se encuentra en este estado cuando se ha deshabilitado el seguimiento de conexiones sobre él.

Trabajar con estados es muy interesante; ya que, además de aumentar el control sobre el tráfico, permite hacer rulesets más sencillos. Por ejemplo, detrás del firewall hay un servidor ssh y se quiere que el firewall sólo acepte conexiones tcp al puerto 22. Llegaría con usar dos reglas: una primera regla que acepte todos los paquetes tcp en estado NEW al puerto 22, y una segunda que acepte los paquetes en ESTABLISHED.

A continuación se verán algunos ejemplos de comandos iptables donde se trabaja con el estado del paquete:

```
$ sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

[2] Aquí por conexión se entiende comunicaciones tcp, udp o icmp.

Este comando añade una regla para permitir la salida de paquetes desde el propio equipo pertenecientes a conexiones autorizadas previamente por otras reglas o paquetes asociados a esas conexiones autorizadas.

- `-A OUTPUT` añade la regra al final de la cadena OUTPUT (donde están las regras que se aplican a los paquetes creados por la propia máquina).
- `-m conntrack --ctstate ESTABLISHED,RELATED` para indicar que se aplicará a los paquetes en estado ESTABLISHED y RELATED. Es decir, paquetes asociados a conexiones establecidas y autorizadas previamente.
- `-j ACCEPT` para indicar que los paquetes que cumplan todas las condiciones serán aceptados y continuarán procesándose (se enviarán a la tabla de encaminamiento y después a POSTROUTING).

```
$ sudo iptables -A OUTPUT -p udp  --dport 53  -d 80.58.32.97 - m conntrack –
ctstate NEW  -j ACCEPT
```

Este comando añade una regla para permitir lanzar consultas dns al servidor DNS con IP 80.58.32.97.

- `-A OUTPUT` añade la regla al final de la cadena OUTPUT.
- `-p udp` para indicar que la regla se aplica sobre paquetes que a nivel de transporte viajan sobre el protocolo udp.
- `--dport 53` para indicar el puerto destino de los paquetes.
- `-d 80.58.32.97` para indicar que la IP destino de los paquetes es 80.58.32.97.
- `- m conntrack --ctstate NEW` para autorizar el primer paquete de una conexión.
- `-j ACCEPT` para indicar que los paquetes que cumplan todas las condiciones serán aceptados y continuarán procesándose.

Es posible crear reglas iptables con estados usando el match `-m state --state` *`lista_de_estados`*. Hacer reglas con `-m state --state` y con `-m conntrack --ctstate` viene siendo equivalente; ya que, el match `conntrack` es el reemplazo de `state`. Siendo un poco más estrictos, `state` es considerado un 'subconjunto' de `conntrack`, que permite más estados. Por tanto, las siguientes reglas son equivalentes:

```
$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$ sudo iptables -A OUTPUT -p udp --dport 53 -d 80.58.32.97 -m state --state NEW -
j ACCEPT
$ sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
$ sudo iptables -A OUTPUT -p udp --dport 53 -d 80.58.32.97 -m conntrack --ctstate
NEW -j ACCEPT
```

En algunas distribuciones de Linux, el fichero `/proc/net/ip_conntrack` mantiene una tabla con todas las conexiones abiertas; en otras, hay que usar la herramienta `conntrack` para acceder a esa información:

```
$ sudo apt-get update
$ sudo apt-get install conntrack
$ sudo conntrack -L
unknown  2 599 src=192.168.1.1 dst=224.0.0.1 [UNREPLIED] src=224.0.0.1
dst=192.168.1.1 mark=0 use=1
tcp       6  431996  ESTABLISHED  src=192.168.56.1  dst=192.168.56.254  sport=46621
dport=22  src=192.168.56.254  dst=192.168.56.1  sport=22  dport=46621  [ASSURED]
mark=0 use=1
tcp       6  73  TIME_WAIT  src=192.168.1.2  dst=192.168.1.254  sport=53705  dport=22
src=192.168.56.253 dst=192.168.1.2 sport=22 dport=53705 [ASSURED] mark=0 use=1
```

```
tcp      6 431999 ESTABLISHED src=192.168.56.1 dst=192.168.56.254 sport=49370
dport=22  src=192.168.56.254  dst=192.168.56.1  sport=22  dport=49370  [ASSURED]
mark=0 use=1
udp      17 176 src=192.168.56.253 dst=8.8.8.8 sport=47920 dport=53 src=8.8.8.8
dst=192.168.1.254 sport=53 dport=47920 [ASSURED] mark=0 use=1
tcp      6 116 TIME_WAIT src=192.168.56.253 dst=104.83.9.145 sport=40273 dport=80
src=104.83.9.145 dst=192.168.1.254 sport=80 dport=40273 [ASSURED] mark=0 use=1
conntrack v1.4.1 (conntrack-tools): 6 flow entries have been shown.
```

El firewall de host del ejemplo anterior con estados y aplicando algunas buenas prácticas sería:

Reglas ESTABLISHED, RELATED

Aceptar todos los paquetes con destino server que pertenecen a una conexión ya autorizada y establecida:

```
$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Regla para permitir la salida de server de todos los paquetes que pertenecen a una conexión ya autorizada y establecida:

```
$ sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Red de lazo cerrado

La regra de loopback está pensada para permitir el tráfico del host dirigido al propio host; y sin ella, puede haber problemas de comunicaciones entre aplicaciones corriendo en el propio equipo. Para crear estas reglas se acostumbra a hacer de cualquiera de las siguientes formas e inmediatamente después de crear las regras para las conexiones ESTABLISHED, RELATED:

```
$ sudo iptables -A INPUT -i lo -j ACCEPT
$ sudo iptables -A OUTPUT -o lo -j ACCEPT
```

O también:

```
$ sudo iptables -A INPUT -i lo -m state --state NEW -j ACCEPT
$ sudo iptables -A OUTPUT -o lo -m state --state NEW -j ACCEPT
```

Control de acceso al firewall

En primer lugar, hay que garantizar el acceso al firewall desde equipos autorizados craendo una regla anti-bloqueo (**Anti-Lockdown Rule**):

```
$ sudo iptables -A INPUT -p tcp --dport 22 -s 192.168.1.2 -m conntrack --ctstate
NEW -j ACCEPT
```

En segundo lugar, hay que registrar y prohibir todos los intentos de acceso desde equipos no autorizados con una regla de bloqueo (**Lockdown Rule** o **Stealth Rule**)

```
$ sudo iptables -A INPUT -p tcp --dport 22 -j LOG --log-prefix "iptables: SSH
Bloqueo"
$ sudo iptables -A INPUT -p tcp --dport 22 -j DROP
```

Los intentos de acceso no autorizados se bloquean y quedan registrados en el archivo `/var/log/kern.log`. Para facilitar la localización de los eventos las líneas tienen el texto "iptables: SSH Bloqueo".

```
$ less /var/log/kern.log
Jan 31 13:56:51 fw-linux kernel: [18004.839588] iptables: SSH Bloqueo IN=eth1
OUT=       MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00       SRC=192.168.56.253
DST=192.168.1.254  LEN=60  TOS=0x00  PREC=0x00  TTL=64  ID=28309  DF  PROTO=TCP
SPT=37556 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
Jan 31 14:10:55 fw-linux kernel: [18848.794862] iptables: SSH Bloqueo IN=eth1
OUT=       MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00       SRC=192.168.56.253
DST=192.168.1.254  LEN=200  TOS=0x00  PREC=0x00  TTL=64  ID=28324  DF  PROTO=TCP
SPT=37556 DPT=22 WINDOW=592 RES=0x00 ACK PSH URGP=0
```

```
Jan 31 14:10:57 fw-linux kernel: [18850.155472] iptables: SSH Bloqueo IN=eth1
OUT= MAC=08:00:27:d1:74:0e:08:00:27:50:7c:37:08:00 SRC=192.168.56.253
DST=192.168.56.254 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=48444 DF PROTO=TCP
SPT=37557 DPT=22 WINDOW=29200 RES=0x00 SYN URGP=0
```

Autorizar establecimiento de conexiones

Ahora lo que falta es autorizar el establecimiento de conexiones aceptando el primer paquete. La siguiente regla permite que server lance consultas dns al servidor 8.8.8.8.

```
$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.8.8 -m conntrack --ctstate NEW
-j ACCEPT
```

Una vez autorizado el primer paquete, cualquier paquete de la conexión será aceptado gracias a las reglas `ESTABLISHED,RELATED`:

- La regla "`OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED`" permitirá la salida de los paquetes necesarios para que el equipo hable con el servidor 8.8.8.8.
- La regra "`INPUT -m conntrack --ctstate ESTABLISHED,RELATED`" permitirá la entrada de los mensajes de respuesta del servidor 8.8.8.8 a la consulta efectuada por el equipo.

El resto de reglas sería:

```
$ sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.4.4 -m conntrack --ctstate NEW
-j ACCEPT
$ sudo iptables -A OUTPUT -p tcp -m multiport --dports 80,443 -m conntrack --
ctstate NEW -j ACCEPT
```

Registro y detección con -j LOG

Los paquetes que no han sido autorizados deben registrarse antes de ser descartados por las reglas CleanUP; sin embargo, puede que haya tráfico como el broadcast y multicast presente en la red y que no interese registrar. Para este tipo de tráfico se crearán las **No Logging Rules**.

```
$ sudo iptables -A INPUT -d 192.168.1.255 -j DROP
$ sudo iptables -A INPUT -d 255.255.255.255 -j DROP
$ sudo iptables -A INPUT -d 224.0.0.0/4 -j DROP
```

Para registrar el tráfico no autorizado se creará una **Log Denied Rule** antes de las Clean Up:

```
$ sudo iptables -A INPUT -j LOG --log-prefix "iptables: tráfico denegado"
$ sudo iptables -A OUTPUT -j LOG --log-prefix "iptables: tráfico denegado"
$ sudo iptables -A FORWARD -j LOG --log-prefix "iptables: tráfico denegado"
```

En `kern.log` se vuelcan además de registros de netfilter otro tipo de información, por lo que puede ser deseable desviar los logs de iptables a un archivo propio para su mejor revisión. En Ubuntu Linux el proceso a seguir es el siguiente:

```
$ sudo nano /etc/rsyslog.d/00-iptables.conf
:msg,contains,"iptables: " -/var/log/iptables.log
& ~
```

Una vez creado el archivo e indicado que las líneas que tegan la expresión "iptables: " sean guardadas en `/var/log/iptables.log` se procede a reiniciar el servicio `rsyslog`:

```
$ sudo service rsyslog restart
```

A partir de ahora, todos los registros de iptables se guardarán en `/var/log/iptables.log`.

Registro y detección con -j NFLOG

Otro método para tener los logs de iptables en un archivo diferente (o incluso en varios) es usar el servicio `ulogd` y `NFLOG` en las reglas iptables en vez de LOG. Para usar este sistema empezamos instalando `ulogd`:

```
$ sudo apt-get install ulogd2
```

A partir de ahora, cuando deseemos registrar incidencias en netfilter/iptables ya podemos usar NFLOG. En el ejemplo, reemplazamos la regla de registro de accesos ssh no autorizados:

```
$ sudo iptables -R SSH 4 -j NFLOG --nflog-prefix "iptables: SSH BLoqueo"
```

Con la configuración por defecto de ulogd se registrarán las incidencias en el archivo /var/log/ulog/syslogemu.log

```
$ ls -lahF /var/log/ulog/
total 2.5K
drwxr-xr-x 2 root root    3 Oct 15 22:07 ./
drwxrwxr-x 9 root syslog 25 Oct 15 22:07 ../
-rw-r--r-- 1 root root    0 Oct 15 22:07 syslogemu.log
$ cat /var/log/ulog/syslogemu.log
Oct 15 22:17:20 fw-linux iptables: SSH BLoqueo IN=eth0 OUT= MAC=00:16:3e:f1:b3:36:08:00:27:48:e3:f7:08:00 SRC=192.168.1.150 DST=192.168.1.254 LEN=60 TOS=00 PREC=0x00 TTL=64 ID=53870 DF PROTO=TCP SPT=35008 DPT=22 SEQ=432581504 ACK=0 WINDOW=29200 SYN URGP=0 MARK=0
Oct 15 22:17:21 fw-linux iptables: SSH BLoqueo IN=eth0 OUT= MAC=00:16:3e:f1:b3:36:08:00:27:48:e3:f7:08:00 SRC=192.168.1.150 DST=192.168.1.254 LEN=60 TOS=00 PREC=0x00 TTL=64 ID=53871 DF PROTO=TCP SPT=35008 DPT=22 SEQ=432581504 ACK=0 WINDOW=29200 SYN URGP=0 MARK=0
```

Clean Up Rule

Para descartar el resto del tráfico, se puede establecer la política de las cadenas a DROP como se explicó en el punto anterior o crear una última regla en las cadenas descartando todo con DROP o REJECT:

```
$ sudo iptables -A INPUT -j DROP
$ sudo iptables -A OUTPUT -j DROP
$ sudo iptables -A FORWARD -j DROP
```