

REDES: Capítulo 9

SQUID e IPTABLES



[UNIVERSIDAD TECNOLÓGICA NACIONAL]



[FACULTAD REGIONAL DELTA]

Capítulo 09 - SQUID E IPTABLES



ÍNDICE

¿Qué es un proxy?	4
TIPOS	4
SQUID	6
Configuración básica	7
PARÁMETRO HTTP_PORT	8
PARÁMETRO CACHE_MEM	9
PARÁMETRO CACHE_DIR	10
PARÁMETRO FTP_USER	11
PROXY ACELERADO	12
ESTABLECIENDO EL IDIOMA POR DEFECTO	13
ARCHIVO DE CONFIGURACIÓN BÁSICO	13
LISTAS Y REGLAS DE CONTROL DE ACCESO	15
LOGS DE ACCESO	18
Iptables	18
TABLAS DE IPTABLES	18
Descripción de tablas y funciones	19
¿QUÉ SIGNIFICA LA ENTRADA DE PAQUETES?	19
¿QUÉ SIGNIFICA LA SALIDA DE PAQUETES?	20
¿QUÉ SIGNIFICA REENVÍO DE PAQUETES?	20
Políticas	20
Reglas de firewall	21
¿Cómo configurar un firewall avanzado?	23
Iptables	25

¿Por qué implementar NAT?	25
¿CÓMO USAR IPTABLES PARA HACER NAT?	27
¿CÓMO ENMASCARAR?	29
¿CÓMO SE HACE EL REENVÍO DE PUERTOS?	29
¿CÓMO CREAR UN PROXY TRANSPARENTE?	30
¿CÓMO CONFIGURAR UN FIREWALL DE FILTRADO CON IPTABLES?	30
Stateful Firewall	34
OBTENIENDO LAS HERRAMIENTAS NECESARIAS	34
¿CUÁL SERÍA EL DISEÑO BÁSICO?	35
¿CÓMO DEFINIR REGLAS?	36
¿QUÉ ES Y CÓMO SE CONFIGURA EL SEGUIMIENTO DE CONEXIÓN?	38
¿QUÉ SON LOS ESTADOS DE CONEXIÓN?	39
¿CÓMO AÑADIR UNA STATEFULL RULE?	41

Capítulo 09 - SQUID E IPTABLES

¿Qué es un proxy?



Básicamente un servidor de proxy es un equipo que hace de intermediario entre nosotros y alguno otro equipo destino.

Existen distintas formas de realizar esto, cada una con su particularidad que analizaremos en breve. Por lo que no hay que confundir con que el proxy, aparte de hacer las veces de intermediario, almacena en cache. El cache es algo opcional que puede tener el software que hace de proxy.

Vamos a abordar dos tipos de proxy; pero existen más. Los dos tipos son los proxy que no requieren configuración en el cliente por parte del usuario, y los que no requieren configuración del cliente por parte del usuario.

TIPOS

El primer tipo de proxy que abordaremos será el proxy transparente.

Este tipo de proxy no requiere la intervención del usuario en la aplicación para que esta use el proxy. Significa que el cliente (navegador, ftp) no tiene conocimiento que esta siendo tratado a través de un proxy.



Por lo general se intercepta el que tenga un puerto destino específico y se lo redirecciona al puerto en el cual se encuentra atendiendo el servidor proxy.

Por ejemplo si quiero “proxear” todo el tráfico con destino al puerto 80 o sea, proxear todo el tráfico http colocaremos una regla en cualquiera de los equipos por los que atraviesa al tráfico (router, firewall, etc.) que diga si el puerto destino es el puerto 80, redireccionar el tráfico a la IP del proxy y puerto del proxy server.

Esto lo que hará es que todo el tráfico con destino puerto 80, generalmente (casi exclusivamente) http.

Dado que requiere una redirección, lo encontraremos generalmente en equipos que hacen de router, firewall, proxy ya que es menos engorroso realizar una redirección de un mismo equipo al mismo equipo, y aparte porque en los routers o firewalls suele concentrarse una importante cantidad de tráfico.

El segundo tipo de proxy que nos podremos encontrar el proxy explícito, ya que explícitamente deberemos indicarle a la aplicación que deberá de utilizar un proxy.



Al ser explícitamente configurado, la aplicación conoce que su tráfico está siendo enviado a través de un servidor proxy.

Squid



Squid es el software para servidor Proxy más popular y extendido entre los sistemas operativos basados sobre UNIX®.



Es muy **confiable**, **robusto** y **versátil**; y al ser **software libre**, además de estar disponible el código fuente, está libre del pago de licencias por uso o con restricción a un uso con determinado número de usuarios.



Squid **funciona** como Proxy y cache con los protocolos HTTP, FTP, GOPHER y WAIS, Proxy de SSL, cache transparente, WWCP, aceleración HTTP, cache de consultas DNS y otras muchas más como filtración de contenido y control de acceso por IP y por usuario.



Squid **no funciona** como proxy para servicios como SMTP, POP3, TELNET, SSH, etc. Para lograrlo tendremos que implementar enmascaramiento de IP a través de un NAT (Network Address Translation) o bien hacer uso de un servidor SOCKS como Dante.

Antes de continuar, es importante que tengamos en cuenta que los espacios vacíos hacen que Squid no funcione de manera correcta así que antes de seguir, veremos cómo debemos descomentar una línea. A simple vista pareciera que nuestros dos ejemplos son iguales pero no es así...

Primero vamos a instalar squid:

```
apt-get install squid3
```

Fijémonos bien en los detalles:

Opción incorrectamente des-comentada:

```
>> http_port 3128
```

Opción correctamente des-comentada:

```
>>http_port 3128
```



En el primero de los ejemplos hay un espacio antes de `http_port`, tengamos mucho cuidado con este tipo de errores ya que impedirán que funcione nuestro servidor.

Configuración básica

Squid utiliza el archivo de configuración localizado en `/etc/squid/squid.conf`. Existe un gran número de parámetros, a continuación veremos los más importantes:

- `http_port`
- `cache_dir`
- Al menos una Lista de Control de Acceso
- Al menos una Regla de Control de Acceso
- `httpd_accel_host`
- `httpd_accel_port`
- `httpd_accel_with_proxy`

PARÁMETRO HTTP_PORT



Squid por defecto utilizará el puerto 3128 para atender peticiones, sin embargo podemos especificar que lo haga en cualquier otro puerto o bien que lo haga en varios puertos a la vez.

En el caso de que estemos configurando un Proxy Transparente, regularmente se utilizará el puerto 80 y se valdrá del re-direccionamiento de peticiones a fin de no tener que modificar la configuración de los navegadores Web para utilizar el servidor Proxy. Bastará con que utilicemos una puerta de enlace al servidor.



Es importante recordar que los servidores Web, como Apache, también utilizan dicho puerto (puerto 80), por lo que será necesario reconfigurar el servidor HTTP para utiliza otro puerto disponible, o bien desinstalar o deshabilitar el servidor HTTP.

Hoy en día ya no es del todo práctico utilizar un Proxy Transparente, a menos que se trate de un servicio de café Internet u oficina pequeña, siendo que uno de los principales problemas con los que lidian los administradores es el mal uso y/o abuso del acceso a Internet por parte del personal.

Puede resultar más conveniente configurar un servidor Proxy con restricciones por contraseña. Algunos programas utilizados comúnmente por los usuarios suelen traer por defecto el puerto 8080 (servicio de cacheo WWW).



Si queremos aprovechar esto a nuestro favor y ahorrarnos tener que dar explicaciones innecesarias al usuario, podemos especificar que Squid escuche peticiones en dicho puerto también.

A continuación especificaremos que el servidor proxy escuche en dos puertos a la vez.

Para lograrlo vamos a buscar la variable `http_port` y modificarla de la siguiente manera:

```
# You may specify multiple socket addresses on multiple lines.  
#  
# Default: http_port 3128  
http_port 3128  
http_port 8080
```

Podemos incrementar la seguridad, vinculando el servicio a una IP que solo se pueda acceder desde la red local. Considerando que el servidor utilizado posee una IP 10.10.3.1, podemos definir las líneas de la siguiente manera:

```
#  
# You may specify multiple socket addresses on multiple lines.  
#  
# Default: http_port 3128  
http_port 10.10.3.1:3128  
http_port 10.10.3.1:8080
```

PARÁMETRO `CACHE_MEM`

El parámetro `cache_mem` establece la cantidad ideal de memoria para lo siguiente:

- **Objetos en tránsito:** son los objetos que están siendo transferidos. Si pensamos en un solo cliente conectándose a un blog, la cantidad de este tipo de objetos será mínima. Pero pensemos en miles de clientes conectándose a varios sitios. En este caso será más que importante la cantidad de objetos que están transitando por nuestro proxy.
- **Objetos Hot:** son los objetos que squid guardó en memoria y sirve a los diferentes clientes.
- **Objetos negativamente almacenados en el caché.**

Los datos de estos objetos se almacenan en bloques de 4 Kb. El parámetro `cache_mem` especifica un límite máximo en el tamaño total de bloques acomodados, donde los objetos en tránsito tienen mayor prioridad.

Sin embargo los objetos Hot y aquellos negativamente almacenados en el caché podrán utilizar la memoria no utilizada hasta que esta sea requerida. De ser necesario, si un objeto en tránsito es mayor a la cantidad de memoria especificada, Squid excederá lo que sea necesario para satisfacer la petición.



Por defecto se establecen 8 MB. Podemos especificar una cantidad mayor si lo consideramos necesario, dependiendo esto de los hábitos de los usuarios o necesidades establecidas por el administrador.

Si poseemos un servidor con al menos 128 MB de RAM, estableceremos 16 MB como valor para este parámetro:

```
cache_mem 32 MB
```

PARÁMETRO CACHE_DIR



Este parámetro vamos a utilizarlo para establecer qué tamaño queremos que tenga el caché en el disco duro disponible para Squid.

Para entender esto un poco mejor, responderemos a esta pregunta, ¿cuánto deseamos almacenar de Internet en el disco rígido?

Por defecto Squid utilizará un cache de 100 MB, de modo tal que encontraremos la siguiente línea:

```
cache_dir ufs /var/spool/squid 100 16 256
```



Podemos incrementar el tamaño del cache hasta donde lo deseemos. Mientras más grande el cache, más objetos se almacenarán en él y por lo tanto se utilizará menos el ancho de banda.

La siguiente línea establece un cache de 700 MB:

```
cache_dir ufs /var/spool/squid 700 16 256
```



Los números 16 y 256 significan que el directorio del cache contendrá 16 subdirectorios con 256 niveles cada uno. No modifiquemos estos números ya que no hay necesidad de hacerlo.



Importante:

Si especificamos un determinado tamaño de cache y éste excede al espacio real disponible en el disco rígido, Squid se bloqueará inevitablemente.

PARÁMETRO FTP_USER



Si accedemos a un servidor FTP de manera anónima, por defecto Squid enviará como contraseña Squid@.

Si queremos que el acceso anónimo a los servidores FTP sea más informativo, o bien si deseamos acceder a servidores FTP que validan la autenticidad de la dirección de correo especificada como contraseña, podemos especificar la dirección de correo electrónico que uno consideremos pertinente por ejemplo:

```
ftp_user proxy@linuxcollege.com.ar
```

PROXY ACELERADO



Esta función permite navegar rápidamente cuando los objetos ya están en el cache de Squid optimizando enormemente la utilización del ancho de banda.

En la sección HTTPD-ACCELERATOR OPTIONS deben habilitarse los siguientes parámetros:

```
httpd_accel_host virtual
httpd_accel_port 0
httpd_accel_with_proxy on
```

Si se trata de un Proxy transparente deben utilizarse con las siguientes opciones, considerando que se hará uso del cache de un servidor HTTP (Apache) como auxiliar.

Debe especificarse la IP de cualquier servidor HTTP en la red local o bien el valor virtual:

```
httpd_accel_host 10.10.3.1
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

ESTABLECIENDO EL IDIOMA POR DEFECTO



Squid incluye la traducción a distintos idiomas de las distintas páginas de error e informativas que son desplegadas en un momento dado.

Dichas traducciones las podemos encontrar en `/usr/lib/squid/errors/`. Para poder hacer uso de las páginas de error traducidas al español, es necesario cambiar un enlace simbólico localizado en `/etc/squid/errors` para que apunte hacia `/usr/lib/squid/errors/Spanish` en lugar de hacerlo hacia `/usr/lib/squid/errors/English`.

Vamos a eliminar primero el enlace simbólico actual:

```
# rm -f /etc/squid/errors
```

Luego crearemos un enlace al directorio de los mensajes en español.

```
# ln -s /usr/share/squid/errors/Spanish errors
```

ARCHIVO DE CONFIGURACIÓN BÁSICO



El archivo de configuración de Squid es un archivo que por defecto es demasiado extenso y cualquier error que pasemos por alto requerirá buscar en más de dos mil líneas de configuración donde se encuentra el mismo.

01

Es por este motivo que lo que haremos será crear un archivo de configuración mínimo que dé permiso de navegación total a todos los clientes.

Muchos de los parámetros ya han sido explicados anteriormente:

```
cd /etc/squid3
```

```
mv squid.conf squid.conf_bkp
```

02

Y luego cargamos el archivo de configuración con los siguientes datos. Cuidado, algunos seguramente van a averiar en base a su topología o configuración de red:

```
# Append Domain
append_domain    .carreralinux.com.ar

#supongamos que nuestro proxy atiende en la ip 172.16.2.254
http_port 172.16.2.254:3128
http_port 172.16.2.254:8080

cache_mem 128 MB

cache_dir ufs /var/spool/squid3 700 16 256

access_log /var/log/squid3/access.log squid

error_directory /usr/share/squid3/errors/Spanish

ftp_user proxy@carreralinux.com.ar

http_access  allow all

#Refresh pattern:
refresh_pattern ^ftp:          1440      20%      10080
refresh_pattern ^gopher:       1440      0%       1440
refresh_pattern -i (/cgi-bin/|\?) 0        0%        0
refresh_pattern (Release|Package(.gz)*)$ 0        20%
2880
refresh_pattern .               0        20%      4320

# HTTP OPTIONS
acl shoutcast rep_header X-HTTP09-First-Line ^ICY\s[0-9]
```

```
# Apache to signal ETag correctly on such responses
acl apache rep_header Server ^Apache
```

```
# Domain
hosts_file /etc/hosts
```

03

```
# Leave coredumps in the first cache dir
coredump_dir /var/spool/squid3
```

Y reiniciamos Squid. Esto puede tardar:

```
/etc/init.d/squid3 restart
```

Ahora si podremos configurar un browser para hacer uso del proxy.

LISTAS Y REGLAS DE CONTROL DE ACCESO

Más temprano que tarde nos encontraremos con la tarea de tener que filtrar tráfico a través del proxy. Básicamente se basa en 2 pasos:

- 1.- Si el tráfico matchea una regla (ACL)
- 2.- Que se hace con ese tráfico (acción)

Normalmente una ACL lleva la sintaxis:

```
acl [nombre] [criterio] [composición]
```

Normalmente una acción lleva la sintaxis:

```
http_access [allow|deny] [nombre_acl]
```

01

Dentro del archivo de configuración podemos ver una línea correspondiente a la “política por defecto”. Esto es, que el tráfico que no matchea ninguna ACL, realiza la siguiente acción:

```
cat /etc/squid3/squid.conf | grep allow
http_access allow all
```

- 02 Esto es un error cuando queremos filtrar. Cualquier filtrado, sin importar el OS, sin importar la aplicación, por defecto filtraremos (contrario a lo que tenemos ahora) y luego dejaremos pasar cierto tráfico. Por lo que cambiaremos esta línea:

```
http_access allow all
```

Por esta línea:

```
http_access deny all
```

- 03 Particularmente, vamos a colocar todas las ACL y todas las acciones dentro de un archivo, que luego lo incluiremos desde squid.conf, por lo que antes de la política por defecto pondremos la siguiente línea:

```
include /etc/squid3/rules.conf
```

- 04 Supongamos que queremos que nuestra IP de cliente tenga acceso full a internet y que la misma es 10.10.3.100. Haremos una regla que haga match con esta IP y luego definiremos la acción:

```
vi /etc/squid3/rules.conf
```

- 05 Ingresamos:

```
acl CLA_alumno1 src 10.10.3.100/32
```

- 06 Y al final:

```
#Acciones
```

```
http_access allow CLA_alumno1
```

- 07 Guardamos y hacemos un reload de squid. No hagamos un restart ya que interrumpiríamos todas las conexiones que están establecidas y aparte tarda mucho ya que elimina y crea de nuevo el cache:

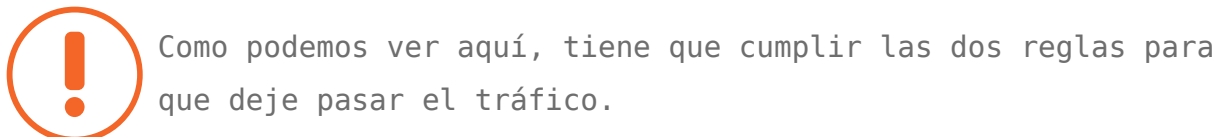
```
/etc/init.d/squid/reload
```

- 08 Luego configuremos un browser desde ese cliente específico e intentemos acceder. Y probemos lo mismo en otro cliente con otra IP.

- 09 A partir de aquí hay diferentes técnicas y formas de filtrar. Vamos a ver algunos ejemplos más. Supongamos que deseamos habilitar tráfico a ciertos dominios específicos. Para esto deberemos de utilizar el `dstdomain`. Entonces, agreguemos las siguientes ACLs:

```
acl CLA_net          src 10.10.3.0/24
acl CLA_OK_DOMAINS  dstdomain .carreralinux.com
acl CLA_OK_DOMAINS  dstdomain .carreralinux.com.ar
acl CLA_OK_DOMAINS  dstdomain .linuxfoundation.org
```

- 10 Y al final agregamos la siguiente acción:
- ```
http_acces allow cla_net CLA_OK_DOMAINS
```



Existen otros criterios para matchear una ACL:

- Mac address
- IP o red de origen/destino
- Expresiones regulares del dominio solicitado
- Palabras en la URL
- Fecha/hora
- Puerto destino
- Protocolo
- y más

## LOGS DE ACCESO

Los logs de acceso se dividen en dos:

1.- Por un lado tenemos los que manejan el comportamiento del servicio:

```
/var/log/squid3/cache.log
```

2.- Y por otro lado tenemos los que manejan el acceso a las páginas:

```
/var/log/squid3/access.log
```

Por un lado tenemos la hora en formato Unix, el tamaño. También nos proporciona el verbo http junto con la dirección IP que realiza ese request. También tendremos la URL solicitada junto con la respuesta (200, 404, 500, 304, etc...)

## Iptables



Un firewall es un equipo que permite o deniega la entrada de paquetes a nuestro equipo o a toda la red.

El firewall, a su vez, es capaz de cambiar el destino de los paquetes que recibe o envía ya sea para direccionarlos a un puerto determinado o a un equipo determinado.

## TABLAS DE IPTABLES

Iptables trabaja con tres tablas:

- tabla de filter
- tabla de nat
- tabla de mangle

Las tablas, a su vez, contienen cadenas.

Cadenas de la tabla filter:

- INPUT
- OUTPUT
- FORWARD

Cadena de la tabla nat:

- PREROUTING
- POSTROUTING
- FORWARD

Cadena de la tabla de mangle:

- INPUT
- OUTPUT
- FORWARD
- PREROUTING
- POSTROUTING
- FORWARD

## DESCRIPCIÓN DE TABLAS Y FUNCIONES



La tabla de filter se usa cuando necesitamos escribir reglas que permitan la entrada de paquetes (INPUT), la salida de paquetes (OUTPUT), y/o el reenvío de paquetes (FORWARD).

### ¿QUÉ SIGNIFICA LA ENTRADA DE PAQUETES?

Significa que alguna conexión desde algún destino conocido o no, está intentando llegar a nuestro equipo usando un puerto de servicio, que se encuentra abierto.

### ¿QUÉ SIGNIFICA LA SALIDA DE PAQUETES?

Un requerimiento hecho por nuestra red que será capaz de pasar por el firewall tratando de alcanzar un objetivo remoto.

### ¿QUÉ SIGNIFICA REENVÍO DE PAQUETES?

Un requerimiento que tiene que pasar por el firewall, y este último sea el encargado final de hacerlo llegar al destino.

Tabla de nat se usa cuando necesitamos cambiar el destino del paquete. Ya sea que el paquete tenga que:

- salir de nuestro equipo
- necesite entrar a él y tenga que cambiar de puerto
- que necesitemos llevarlo a otro equipo.

## POLÍTICAS



Las cadenas a su vez tienen políticas (policies). Estas políticas aplicadas a una tabla en particular le dicen al kernel lo que tiene que hacer.



Las políticas son inflexibles y no se cambian.

Una política de aceptar en la tabla filter hace que todos los paquetes para todas las interfaces (todas las placas de red), sean aceptados. La política contraria (DROP) hace que todos los paquetes que llegan a todas las interfaces sean descartados. Si vemos las cosas de este modo podríamos solamente aceptar todo o descartar todo.

Sin embargo el firewall nos permitirá crear saltos (jumps) a las políticas principales. Así podríamos decirle a los paquetes que los puertos del 1 al 69 estén cerrados, que el puerto 80 está abierto, y que los puertos del 81 al 1024 están cerrados. De esta forma habríamos creado un salto a una política de descartar paquetes.



Entonces lo primero que tenemos que hacer a la hora de configurar nuestro firewall es **saber qué política vamos a implementar**.

Si la política es descartar todos los paquetes, vamos a tener que hacer saltos para aceptar algunos, si la política es aceptar entonces debemos hacer saltos para descartar los que lleguen a los otros puertos.

## REGLAS DE FIREWALL

Cuando queremos ver el valor de las tablas, las cadenas y sus políticas por defecto debemos ejecutar el comando:

```
iptables -t filter -L
```

Este comando le dice al kernel que queremos listar los valores de las cadenas de la tabla de filter. Cuando queremos cambiar la política de una cadena por defecto debemos usar:

```
iptables -t filter -P INPUT DROP
```

En este caso le estamos diciendo al kernel que el valor para la cadena INPUT de la tabla de filter es descartar. Si queremos hacer un salto a las cadenas podemos hacer lo siguiente:

```
iptables -t filter -A INPUT -t tcp -s 0/0 -i eth0 --dport 80 -j ACCEPT
```



Con este comando estamos diciéndole al kernel que queremos agregar (-A INPUT) un nuevo valor a la tabla de filter usando el protocolo tcp (-p tcp), desde cualquier origen (-s 0/0), con destino nuestra interface eth0 (-eth0), que llegue al puerto 80 (--dport 80), y tiene que tomar un salto ACCEPT (-j ACCEPT).

Veamos ahora:

¿Cómo sería nuestro pequeño firewall si queremos dejar abierto solamente el puerto 80?

```
iptables -t filter -P INPUT DROP
iptables -t filter -A INPUT -p tcp -s 0/0 -i eth0 --dport 80 -j ACCEPT
```

Para filtrar el ping podríamos hacer una regla como esta:

```
#iptables -t filter -A INPUT -p icmp --icmp-type echo-request -j DROP
```

Cuando lo que queremos es compartir la conexión a internet tenemos que usar la tabla de nat y la cadena de POSTROUTING. Veamos a continuación cómo hacerlo:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -s 10.0.0.0/8 -j MASQUERADE
```




La primera línea le dice al kernel que tiene que habilitar el forwardo de ip. La segunda habilita POSTROUTING para hacer un intercambio de ip entre el numero ip privado y el numero ip público.

Ahora en los clientes tenemos que definir al equipo en el cual tenemos estas reglas como default gw:

```
route add default gw 10.10.3.30
```

Suponiendo que 10.10.3.30 sea el gateway de esta red.



**Atención:** No debemos olvidarnos de colocar en el archivo `/etc/resolv.conf` los números ip del servidor de nombres de nuestro proveedor de Internet.

## ¿CÓMO CONFIGURAR UN FIREWALL AVANZADO?

Sólo para que refresquemos la memoria hay distintos tipos de Firewall y cada uno de ellos cumple una función distinta. A continuación veremos una tabla con las ventajas y desventajas de cada uno:

| Puerto | Descripción                                                                                                                                                                                                        | Ventajas                                                                                                                                                                              | Desventajas                                                                                                         |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| nat    | Traducción de direcciones de red. Traduce direcciones privadas de red en una o varias direcciones IP públicas. Así se enmascaran todas las peticiones y se vuelven a direccionar las respuestas a un único origen. | Se protegen muchas máquinas posicionándolas detrás de una dirección IP pública.<br>Las restricciones de acceso desde y hacia la LAN se pueden configurar abriendo o cerrando puertos. | No se pueden prevenir o detectar acciones peligrosas una vez que se ha establecido una conexión fuera del firewall. |

|                             |                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                |                                                                                                                                                     |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Filtrado de Paquetes</p> | <p>Este tipo de Firewall lee cada paquete que entra en la red. Puede hacerlo a través de la información contenida en el encabezado y filtra los paquetes basándose en reglas implementadas por el administrador. Los últimos Kernels han implementado soluciones propias para el filtrado de paquetes.</p> | <p>Es totalmente configurable a través de iptables. No necesita una configuración especial del lado del cliente ya que todo se resuelve al nivel de la capa del Router y no de la aplicación. La transmisión de paquetes es rápida porque se resuelve a través del Kernel.</p> | <p>No puede filtrar los paquetes al nivel de contenido. Redes muy complejas pueden tornar muy difícil la tarea de definir e implementar reglas.</p> |
| <p>PROXY</p>                | <p>Filtra todas las peticiones de determinado protocolo de un cliente y luego las transformara y emitirá a internet. Actúa como un intermediario entre usuarios remotos y la red local.</p>                                                                                                                | <p>Le da al administrador control acerca de cuáles aplicaciones funcionan en las afueras de la LAN. Algunos cachean información que es frecuentemente solicitada para que pueda ser accedida más velozmente. Todas las peticiones pueden ser logueadas y monitoreadas.</p>     | <p>Puede transformarse en un cuello de botella, ya que todas las peticiones deben pasar por él.</p>                                                 |



## Iptables



Iptables se usa para configurar, mantener e inspeccionar las reglas de firewall IP del núcleo Linux.

Es un descendiente directo de ipchains (que vino de ipfwadm, que vino del ipfw IIRC de BSD).

Los módulos del kernel pueden registrar una tabla nueva, e indicarle a un paquete que atravesase una tabla dada. Este método de selección de paquetes se utiliza para el filtrado de paquetes, para la Traducción de Direcciones de Red (NAT) y para la manipulación general de paquetes antes del enrutamiento.



Una de las ventajas de iptables sobre ipchains es que es pequeño y rápido.

## ¿Por qué implementar NAT?

El servicio de Traducción de Direcciones de Redes IP es algo así como el hermano mayor estandarizado del servicio de Enmascarado IP de Linux.



NAT proporciona características que no posee el Enmascarado IP que lo hacen eminentemente más apropiado para su uso en los diseños firewalls corporativos y en instalaciones a mayor escala.

En nuestra tabla vimos algunas de las ventajas del uso de NAT, trabajemos sobre ellas un poco más.

La mayoría de los ISP (Proveedor de Servicios de Internet) nos dan una sola dirección IP cuando nos conectamos con ellos. Podemos enviar paquetes con cualquier dirección que nos plazca, pero sólo obtendremos respuestas a los paquetes con esa IP de origen.



Si deseamos utilizar varias máquinas diferentes (como una red casera) para conectar a Internet a través de un enlace, necesitamos NAT.

Este es, de lejos, el uso más común de NAT hoy en día, conocido normalmente como “enmascaramiento” (masquerading) en el mundo de Linux. Algunos prefieren llamarle SNAT, porque se cambia la dirección de origen (source) del primer paquete.

A veces queremos cambiar el destino de los paquetes que entran en la red. Con frecuencia esto se debe (como antes), a que sólo tenemos una dirección IP, pero deseamos que la gente sea capaz de llegar a las máquinas detrás de la que tiene la IP “real”. Si volvemos a escribir el destino de los paquetes entrantes, podremos conseguirlo.



Una variante común de esto es el **balanceo de carga**, en la que se toma un cierto número de máquinas, repartiendo los paquetes entre ellas. Este tipo de NAT se llamó **reenvío de puerto** (**port-forwarding**) en anteriores versiones de Linux.

Si queremos simular que cada paquete que pase por nuestra máquina Linux esté destinado a un programa en la propia máquina, utilizaremos proxies transparentes.

Un proxy es un programa que se pone entre nuestra red y el mundo real, filtrando las comunicaciones entre ambos como pudimos apreciar en el módulo anterior. La parte transparente se debe a que la red nunca tendrá por qué enterarse que se está comunicando con un proxy, a menos, claro, que el proxy no funcione.

Se puede configurar Squid para que trabaje de esta manera, y a esto se le llamó re-dirección o proxy transparente en anteriores versiones de Linux.

## ¿CÓMO USAR IPTABLES PARA HACER NAT?

Necesitamos crear reglas NAT que le digan al núcleo qué conexiones cambiar y cómo hacerlo. Para ello, usaremos la muy versátil herramienta iptables, y le diremos que altere la tabla de NAT usando la opción `-t nat`.



La tabla de reglas NAT contiene tres listas llamadas “cadenas”. Cada regla se examina por orden hasta que una coincide.



Las tres cadenas se llaman PREROUTING (para Destination NAT, según los paquetes entran), POSTROUTING (para SOURCE NAT, según los paquetes salen), y OUTPUT (para Destination NAT con los paquetes generados en la propia máquina).

Iptables toma cierto número de decisiones estándar que se listarán ahora. Todas las opciones con doble guion pueden ser abreviadas, siempre que iptables pueda distinguirlas de otras opciones posibles.

Si el núcleo tiene la implementación de iptables como módulo, necesitará cargar el módulo `ip_tables.o` antes:

```
modprobe ip_tables
```



La opción más importante aquí es la opción de selección de **tabla**: `-t`.

Para todas las operaciones de NAT, querremos usar `-t nat` para la tabla NAT.



La segunda más importante es `-A` para añadir una nueva regla al final de una cadena (`-A POSTROUTING`), o `-I` para insertarla al principio (`-I PREROUTING`).



Podemos especificar el **origen** (**-s** o **--source**) y el **destino** (**-d** o **--destination**) de los paquetes sobre los que queremos hacer NAT.

Estas opciones pueden ir seguidas por una IP sencilla (192.168.1.1), un nombre (www.linux.org), o una dirección de red (192.168.1.0/24 o 192.168.1.0/255.255.255.0).



Podemos especificar qué interfaz de **entrada** (**-i** o **--in-interface**) o de **salida** (**-o** o **--out-interface**) mirar, pero lo que se puede especificar depende de en qué cadena esté poniendo la regla: **en PREROUTING sólo podemos elegir la interfaz de entrada, y en POSTROUTING (y OUTPUT) sólo la de salida.**

Si usamos la equivocada, iptables nos avisará con un mensaje de error. Dijimos antes que se puede especificar una dirección de origen y destino. Si omitimos la opción de origen, entonces será cualquier dirección de origen. Si omitimos la de destino, será cualquier dirección de destino.

También podemos indicar un protocolo específico (**-p** o **--protocol**), como TCP o UDP; sólo los paquetes de este protocolo coincidirán con la regla.

La razón principal para hacer esto es que al especificar uno de los protocolos TCP o UDP se permiten más opciones: específicamente las opciones **--source-port** y **--destination-port** (abreviadas **--sport** y **--dport**).



Estas opciones nos permiten indicar que sólo los paquetes con un determinado origen y destino coincidirán con la regla.

Esto es útil para redireccionar peticiones web (puertos TCP 80 u 8080) y dejar los demás paquetes tranquilos. Estas opciones deben seguir a la **-p** (que tiene el efecto secundario de cargar la biblioteca compartida de extensión para ese protocolo).

Podemos usar números de puerto, o un nombre de archivo `/etc/services`.

## ¿CÓMO ENMASCARAR?

Si tenemos una conexión PPP con IP dinámica simplemente queremos decirle a nuestra máquina que todos los paquetes que salgan de la red interna deberían aparentar salir de la máquina que tiene el enlace PPP.

#(-A) agrega una regla a la tabla NAT (-t nat), después del encaminamiento (POSTROUTING) para todos los paquetes que salgan por ppp0 (-o ppp0) enmascarando la conexión (-j MASQUERADE ). Tipiamos:

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

para poner en marcha el reenvío de IP (IP forwarding):

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

Si la salida a internet la tenemos por la interfaz de red eth1, entonces la instrucción quedará así:

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Así habremos enmascarado todo lo que salga por eth1.

## ¿CÓMO SE HACE EL REENVÍO DE PUERTOS?



Podemos usar **iptables -t nat** para hacer reenvío de puertos en caso de querer colocar, por ejemplo, un servidor web en la red interna (con un ip no certificado) pero que se vea desde Internet.

Agregaremos una regla previa al encaminamiento (-A PREROUTING) a la tabla NAT (-t nat) de manera que los paquetes TCP (-p tcp) que vayan a 1.2.3.4 (-d 1.2.3.4), puerto 80 (--dport 80) tengan una correspondencia de destino (-j DNAT) con 192.168.1.1, puerto 80 (--to 192.168.1.1:80):

```
iptables -A PREROUTING -t nat -p tcp -d 1.2.3.4 --dport 80 -j
DNAT --to 192.168.1.1:80
```

Si deseamos que esta regla altere también las conexiones locales (aquellas que se originen en la propia máquina que hace NAT), podemos insertar la misma regla en la cadena OUTPUT (que es para los paquetes locales de salida):

```
iptables -A OUTPUT -t nat -p tcp --dport 80 -j DNAT --to
192.168.1.1:80
```

## ¿CÓMO CREAR UN PROXY TRANSPARENTE?

Como dijimos anteriormente, un proxy es un programa que se pone entre nuestra red y el mundo real, filtrando las comunicaciones entre ambos. Envía el tráfico que entra dirigido al puerto 80 (web) a nuestro proxy Squid (transparente):

```
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j RE-
DIRECT --to- port 3128
```

## ¿CÓMO CONFIGURAR UN FIREWALL DE FILTRADO CON IPTABLES?

Un Firewall, como ya sabemos es un dispositivo lógico que protege una red privada del resto de la red (pública). Una manera sencilla y muy básica de implementar uno sería:

```
iptables -A INPUT -j DROP
iptables -A FORWARD -j DROP
```

Estas dos instrucciones impiden el acceso a la máquina y cualquier acceso que se haga a través de ella. Estas instrucciones sellan “herméticamente” la máquina.



Pero hay que tener cuidado, pues si ésta es la única regla, entonces ni siquiera se podrá hacer login.

Para evitar este problema le indicaremos al firewall que todas las conexiones que se hagan localmente (interfaz loopback) sean aceptadas.

Esto lo haremos así:

```
iptables -I INPUT -d 127.0.0.1 -i lo -j ACCEPT
```

Estas tres reglas implementan un firewall muy simple y básico. Desde luego, con estas reglas, la red interna que estamos protegiendo no tendrá salida a internet por lo que debemos habilitar otras cosas, además de que la administración remota del firewall será imposible.



También hay que tomar en cuenta la protección contra ciertos ataques (syn flooding, DoS, etc).

Comencemos. Esta instrucción significa que acepte cualquier paquete que venga de la red 192.168.0.0/24 por la interfaz de red eth0:

```
iptables -A INPUT -s 192.168.0.0/24 -i eth0 -j ACCEPT
```

Niega cualquier paquete del protocolo icmp (ping) que venga de la red 172.16.0.0/16:

```
iptables -A INPUT -s 172.16.0.0/16 -p icmp -j DROP
```

Acepta cualquier paquete que venga y vaya a la misma red (192.168.1.0/24) por la interfaz eth1:

```
iptables -A FORWARD -s 192.168.1.0/24 -d 192.168.1.0/24 -i eth1
-j ACCEPT
```

## EJERCICIO PRÁCTICO

Vamos a crear un archivo llamado `rc.firewall` que solucione la siguiente situación.

Se dispone de 32 estaciones de trabajo y tres servidores. Se toma la red 192.168.0.0/24 para identificar a todas las máquinas y se distribuyen de la siguiente manera:

- Desde 192.168.0.11 - 192.168.0.42 para las Estaciones.
- 192.168.0.5 para el Servidor web.
- 192.168.0.4 para el Servidor de correo.
- 192.168.0.3 para el Servidor ftp.
- 192.168.0.2 queda apartado para una futura impresora en red.
- 192.168.0.1 para la interfaz interna del Firewall (eth0).
- 200.200.200.200 es el único IP certificado que nos asignó nuestro ISP y que se asignará a la interfaz externa del firewall (eth1).

Las reglas a implantar son:

- Los tres servidores deben verse desde internet
- Se debe habilitar el acceso a internet a todas las máquinas de la LAN
- Cualquier otro servicio debe ser denegado por el firewall
- La administración remota del firewall sólo la puede realizar la máquina con el IP 50.150.150.150

### Resolución del ejercicio:

Nuestro archivo `rc.firewall` será parecido a lo siguiente:

01 Primero pongamos a funcionar nuestro firewall:

```
iptables -A INPUT -j DROP
iptables -A FORWARD -j DROP
iptables -I INPUT -d 127.0.0.1 -i lo -j ACCEPT
iptables -I INPUT -s 150.150.150.150 -j ACCEPT
```



02

Ahora enmascaremos las máquinas y démosle salida a internet:

```
iptables -I FORWARD -s 192.168.0.0/24 -i eth0 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

03

Luego filtremos (reenviemos los puertos a los servidores #respectivos; puerto 80 a 192.168.0.5, 25 y 110 para 192.168.0.4 y #21 para 192.168.0.3):

```
iptables -A PREROUTING -t nat -p tcp -d 200.200.200.200 --dport
80 -j DNAT --to 192.168.0.5:80
iptables -A PREROUTING -t nat -p tcp -d 200.200.200.200 --dport
25 -j DNAT --to 192.168.0.4:25
iptables -A PREROUTING -t nat -p tcp -d 200.200.200.200 --dport
110 -j DNAT --to 192.168.0.4:110
#iptables -A PREROUTING -t nat -p tcp -d 200.200.200.200 --dport
21 -j DNAT --to 192.168.0.3:21
iptables -A OUTPUT -t nat -p tcp --dport 80 -j DNAT --to
192.168.0.5:80
iptables -A OUTPUT -t nat -p tcp --dport 25 -j DNAT --to
192.168.0.4:25
iptables -A OUTPUT -t nat -p tcp --dport 110 -j DNAT --to
192.168.0.4:110
iptables -A OUTPUT -t nat -p tcp --dport 21 -j DNAT --to
192.168.0.3:21
```

04

# Para que el ftp funcione correctamente debemos activar ciertos módulos:

```
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp
```

05

# Por último ponemos en marcha el reenvío de IP (IP forwarding):

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



La **usurpación de la dirección IP** (Spoofing en inglés) consiste en hacerle creer a uno que un paquete que viene del mundo externo viene de la interface por la cual llega.

Esta técnica es muy usada por los crackers, pero podemos hacer que el kernel prevenga este tipo de intrusión. Sólo con ingresar la siguiente línea se vuelve imposible este tipo de ataque:

```
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter
```

### Stateful Firewall

Vamos a ver primero por qué se le llama Stateful a este tipo de firewall y veremos luego por qué es la mejor forma de protegerse.



Este firewall podrá funcionar tanto en una estación de trabajo, un servidor o un router. Descartará o rechazará el tráfico que no nos interese.

### OBTENIENDO LAS HERRAMIENTAS NECESARIAS

Además de tener la herramienta iptables en nuestro sistema, debemos tener ciertas funcionalidades en el kernel.

Habitualmente el kernel que trae nuestra distribución ya trae habilitado o están compilados como módulos, todo el soporte para netfilter. Pero cómo es posible que en algún momento tengamos que compilar el kernel por separado detallaremos a continuación algunos puntos que se deberán tener en cuenta:

```
<*>Packet socket
[*] Network packet filtering (replaces ipchains)
<*> Unix domain sockets
```

Y en el submenú IP: Netfilter Configuration ---> habilitar todas las opciones, para tener todas las funcionalidades de netfilter.

No se usarán todas pero es bueno habilitarlas para luego experimentar con ellas. Lo mejor será habilitarlas como módulo para que se carguen en el momento que sean usadas.



Algo que no deberíamos habilitar es la opción **[ ] IP: TCP Explicit Congestion Notification support**. Esta hace que los paquetes que se envían tengan el bit ECN activado, lo cual puede traer problemas con algunos routers.

## ¿CUÁL SERÍA EL DISEÑO BÁSICO?

Para empezar a crear nuestro Stateful Firewall usaremos la herramienta iptables. Esta es usada para interactuar con las reglas de filtrado de paquetes del kernel.

```
iptables -P INPUT DROP
```



Con este comando estaremos protegidos de ataques, porque el comando le dice al kernel que descarte todos los paquetes de red que entran a la PC.

Antes de continuar veamos bien que hace este comando.

El comando iptables -P es usado para definir la política predeterminada de las cadenas internas, en este caso definimos la política predeterminada de la cadena INPUT, una cadena ya definida en el kernel por la que pasan todos los paquetes entrantes.

Al setearla en DROP le estamos diciendo al kernel que cualquier paquete que alcance el final de la cadena INPUT tendrá que ser descartado, y hasta que no hayamos añadido reglas a la cadena todos los paquetes alcanzaran el final y por lo tanto todos serán descartados.

Este comando por sí solo no sirve de nada; sin embargo, esta es una buena estrategia para diseñar el firewall, nosotros empezamos descartando todos los paquetes de forma predeterminada y entonces empezamos a abrir gradualmente el firewall según nuestras necesidades.

### ¿CÓMO DEFINIR REGLAS?

En este ejemplo asumiremos que estamos diseñando un firewall para una máquina con dos placas de red: eth0 y eth1.



eth0 está conectada a nuestra LAN, mientras que eth1 está conectada a un ADSL router, el cual usa la interface ppp0 para conectarse a internet (pppoe).

Ahora “mejoraremos” nuestro firewall añadiendo una línea más:

```
iptables -P INPUT DROP
iptables -A INPUT -i ! ppp0 -j ACCEPT
```

La línea de comando iptables -A añade una regla al final de la cadena INPUT que indicará junto con la política que cuando un paquete ingrese desde cualquier interface (lo, eth0, o ppp0), se dirigirá a la cadena INPUT, si concuerda con la primera regla el paquete es aceptado y ninguna otra tarea es realizada, si no, la política predeterminada de INPUT es aplicada y el paquete es descartado.



Para que nuestro firewall sea útil necesitamos elegir y permitir que algunos paquetes que provengan de Internet lleguen a nuestra máquina.

Hay dos métodos para lograrlo:

- usa reglas estáticas (stateless rules)
- usa reglas dinámicas (stateful rules).

Veamos por ejemplo qué pasa con las páginas web. Si queremos que nuestra máquina sea capaz de bajar (navegar), por páginas web, podemos añadir una regla estática que podría ser siempre verdadera para todo paquete http, sin dar importancia al origen:

```
iptables -A INPUT --sport 80 -j ACCEPT
```

Como todo el tráfico web estándar es originado desde el puerto 80, esta regla efectivamente permite a nuestra máquina navegar sin dificultades. Sin embargo, este método tradicional tiene algunos problemas, porque no todo el tráfico se origina en el puerto 80.

Por ejemplo podemos llegar a encontrar una dirección que se vea así:

```
http://www.falsa.com:81
```



Esta URL apunta a un sitio web en el puerto 81, por lo tanto no será visto por nuestro Firewall.

En realidad este no es el único problema que nos trae utilizar reglas estáticas, el mayor inconveniente es que nosotros no tenemos control sobre el puerto origen de un paquete que llega; ya que éste podría ser fácilmente alterado por un intruso.

#### Por ejemplo:

Si un intruso conoce como está diseñado nuestro firewall, éste podría saltar el firewall simplemente haciendo que todos sus intentos de conexión sean originados en el puerto 80 en su máquina! o tal vez podría realizar ataques de denegación de servicios (DoS) controlando que todo el ataque se origine desde el puerto 80.



Es por eso que Iptables y los kernels 2.4.x proveen todo lo que necesitamos para habilitar un filtrado dinámico (stateful filtering) de los paquetes.

## ¿QUÉ ES Y CÓMO SE CONFIGURA EL SEGUIMIENTO DE CONEXIÓN?

En vez de abrir agujeros en nuestro firewall basándonos en características estáticas de protocolos, nosotros podemos usar la funcionalidad de Linux connection tracking para tomar decisiones basadas en el estado de conexión dinámico de los paquetes.



Conntrack trabaja asociando a cada paquete con un canal de comunicación bidireccional o conexión.

Por ejemplo consideremos qué sucede cuando usamos telnet para conectar a una máquina remota.

Si vemos el tráfico de red a nivel de paquetes, todo lo que observaremos será una serie de paquetes yendo de una máquina a la otra. Sin embargo, a un nivel mayor, éste intercambio de paquetes es actualmente un canal de comunicación bidireccional entre nuestra máquina y la máquina remota.

Generalmente los firewalls solo ven paquetes individuales, no reconocen si ellos pertenecen o no a una conexión existente. Aquí es donde la tecnología connection tracking entra en juego.



La funcionalidad de seguimiento de Linux puede “ver” el alto nivel de las conexiones, reconociendo nuestra sesión de telnet como una entidad lógica simple.

Conntrack también puede reconocer intercambio de paquetes UDP y ICMP como “conexiones” lógicas, a pesar que UDP Y ICMP son protocolos sin conexión.



Esto es muy útil, ya que nos permitirá usar conntrack para manejar el intercambio de paquetes UDP Y ICMP.

Podremos ver una lista de conexiones activas de red en las que nuestra máquina está participando usando el comando `cat /proc/net/ip_conntrack`, aún sin haber configurado un Firewall.



Esa funcionalidad de Linux está trabajando de fondo, manteniendo un seguimiento de todas las conexiones.

## ¿QUÉ SON LOS ESTADOS DE CONEXIÓN?

Conntrack no solo reconoce conexiones, también clasifica a cada paquete que se revisa dentro de cuatro estados.

El **primer estado** del que hablaremos se llama NEW. Cuando tipiamos `telnet remote.host.com`, el paquete inicial o serie de paquetes que se originan en nuestra máquina y son destinados a `remote.host.com` para realizar la conexión, están en el estado NEW.

Sin embargo tan pronto recibimos una simple respuesta de `remote.host.com`, cualquier paquete siguiente que enviemos a `remote.host.com` como parte de esta conexión ya no será considerado como perteneciente a este estado.



Un paquete es considerado NEW cuando éste está estableciendo una nueva conexión y ningún tráfico ha sido recibido aún desde el host remoto (como parte de esta conexión por supuesto).

También podemos tener paquetes NEW de entrada, éstos son generados generalmente por hosts remotos que intentan conectarse a nuestra máquina.

### Por ejemplo:

El primer paquete que nuestro servidor Web recibe como parte de una petición HTTP es considerado como un paquete NEW de entrada. Sin embargo una vez que la máquina contestó, cualquier paquete que recibamos referidos a esa conexión particular ya no estará en este estado.

Una vez que una conexión tiene tráfico en ambas direcciones, los paquetes adicionales relacionados con esa conexión pasan al **segundo estado** llamado ESTABLISHED.

El **tercer estado** es llamado RELATED.



Paquetes RELATED son aquellos que están iniciando una nueva conexión, pero están relacionados con otra conexión actualmente existente.

El estado RELATED puede ser usado para regular conexiones que son parte de un protocolo de multi-conexión como FTP, así como paquetes de error relacionados con conexiones existentes (o sea un paquete ICMP de error acerca de una conexión existente).

Finalmente, tenemos al **cuarto estado**, INVALID.



Los paquetes clasificados como INVALID son aquellos que no entran en ninguna de las otras tres categorías.

Son paquetes que no intentan establecer conexiones ni pertenecen a ninguna conexión existente.



Estos paquetes no son automáticamente descartados, es necesario añadir reglas o configurar bien la política predeterminada para manejarlos.



## ¿CÓMO AÑADIR UNA STATEFULL RULE?

```
iptables -P INPUT DROP
iptables -A INPUT -i ! ppp0 -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Esta simple regla, al ser añadida al final de la cadena INPUT nos permite conectarnos con cualquier host remoto, ya sea FTP, TELNET, WEB, hacer ping etc., pero no permite que ningún host remoto pueda conectarse con nosotros, ni siquiera a través del ping o del envío de paquetes UDP.

Veamos cómo trabaja esta regla. Supongamos que queremos conectarnos por telnet al host remote.host.com. Bien, nuestra máquina envía un paquete para iniciar la conexión, este paquete es clasificado como NEW y nuestro firewall nos permite salir ya que lo hemos configurado para bloquear los paquetes de entrada y no los de salida.

Cuando obtenemos un paquete de respuesta desde remote.host.com, este paquete entra en la cadena INPUT. No concuerda la primera regla y pasa a la segunda, si concuerda con ésta será aceptado (ACCEPT) sino habrá llegado al final y se le aplicará la política predeterminada que indica descartar el paquete (DROP).

En nuestro caso el paquete será aceptado, cuando el kernel inspecciona un paquete de entrada, primero reconoce si el paquete es parte de una conexión existente. De esta forma podrá decidir si este paquete es NEW o ESTABLISHED. Al ser un paquete de entrada, chequeará si ésta conexión ha tenido paquetes de salida antes, en este caso encontrará que nuestro paquete inicial NEW fue enviado al host remoto, de forma tal que lo clasificará como ESTABLISHED así como cualquier otro paquete que recibamos o enviemos como parte de esa conexión.



Veamos qué pasaría si alguien de afuera deseará conectarse por telnet a nuestra máquina.

El paquete inicial que recibimos es clasificado como NEW, este no concuerda con la primera regla y avanza a la segunda, al no ser un paquete ESTABLISHED o RELATED tampoco concuerda con esta regla y llega al final de la cadena, donde se le aplica la política predeterminada; es decir, es descartado.

