Guia de aprendizaje de IPTables/NetFilter

Version 1.0.3. Diciembre 2001.
© 2001,2002 Sancho Lerena <slerena@genterara.com>

Indice

Introducción a este documento

En primer caso, hay que aclarar que este documento no es una referencia técnica, para eso esta el man, asi que no esperes encontrar aquí la sintaxis punto por punto de iptables. Aquí encontraras casi todos los comandos explicados, de una manera diferente, basada en conceptos teóricos mas que en descripciones detalladas de las sintaxis de los comandos.

Antes de hablar de Firewalls, tenemos que tener muy muy claro como funciona una red, y no hablo de saber lo que es un navegador, es importante conocer los siguientes conceptos con mayor o menor soltura:

- Redes TCP/IP: Direccionamiento IP
- Redes TCP/IP: Enrutamiento
- Redes TCP/IP: Subnetting
- Redes TCP/IP: Funcionamiento de protocolos IP, TCP, UDP e ICMP.
- Redes TCP/IP: Ethernet
- Redes TCP/IP: Comandos básicos
- Redes TCP/IP: Servicios tipicos

Esta tampoco es una guía de "como instalar" Netfilter, digamos que es más bien una guia de "como usar NetFilter" y saber que cosas se pueden hacer y que cosas no (¿hay alguna cosa que no se pueda hacer? :) . Realmente, para los que conocemos sistemas de Filtrado y NAT, en firewall comerciales, debemos resaltar la potencia y versatilidad de NETFilter. Es de hecho, el sistema mas versátil, cómodo, eficiente y robusto de cuantos he conocido. Sin más preámbulos, empecemos a hablar de lo que és NetFilter.

Puedes encontrar esta guía actualizada (si tengo tiempo!) y mas información de Networking en general en la URL http://www.gnusec.com. Puedes contactar conmigo si tienes alguna duda, sugerencia o comentario al respecto de todo lo que aquí se dice (y lo que no se dice), mi direccion es slerena@genterara.com

Espero que te sirva este documento.

Agradecimientos

Pedro Turuelo, por sus comentarios, criticas constructivas y detectar algunos errores

Conceptos básicos sobre Firewalls

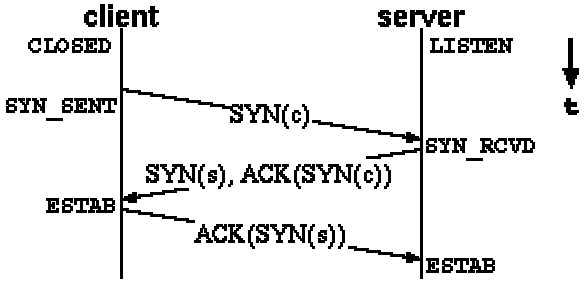
Sesion

Los diferentes tipos de protocolos, entre los cuales distintiguiremos únicamente entre TCP, UDP e ICMP; hay que tratarlos de diferente manera. En suma, todos son iguales desde el punto de vista de la capa de transporte, es decir, del protocolo IP; pero entre ellos hay diferencias sutiles que tenemos que interpretar desde el punto de vista de la *sesion*.

TCP establece una comunicación orientada a sesion, mientras que UDP e ICMP no lo son, cuando hablemos de que el firewall deja pasar las sesiones establecidas, estaremos hablando de TCP, aunque algunos tipos de firewalls pueden establecer internamente *sexiones virtuales*, como es el caso de CheckPoint FW-1 en las sesiones UDP y en el "ping" icmp.

Inicio de comunicación

Hay que tener en cuenta que aunque nosotros solo veamos una comunicación como IP y puerto hay mas detalles que intervienen en ella. En este caso hablaremos de la secuencia de establecimiento de conexión.



Inicialmente el origen lanza una señal SYN. El receptor la recibe y contesta con una señal SYN y otra ACK. Finalmente el origen confirma el establecimiento con una señal ACK. A partir de aquí toda la comunicación queda establecida.

¿Porqué es tan importante esto?, bien, no debemos olvidar que por una maquina que actúa de firewall pasan muchisimos paquetes y los tiene que inspeccionar todos, asi que el proceso que inspecciona los paquetes suele "fiarse" de la información contenida en ese establecimiento inicial. A partir de ahí cuando llega un paquete que no es de establecimiento, mira si pertenece a una conexión establecida y lo deja pasar. Este es el metodo de trabajo usual, por rendimiento.

Veremos mas adelante que podemos contemplar los paquetes de la red desde distintos puntos de vista. Tanto desde el establecimiento de la comunicación como desde el punto de vista de paquetes sueltos dentro de una sesión establecida y la importancia de ello.

Primera aproximacion a IPTables/Netfilter

Iptables es una herramienta que trabaja en el plano de usuario. Sirve para modificar los parámetros de trabajo del firewall del kernel de Linux. Esta herramienta por supuesto esta integrada en el nuevo sistema de filtrado y traduccion de direcciones de NETFilter, con lo que solo podemos disponer de ella a partir de los Kernel 2.4.x de Linux. La herramienta conserva un grado bastante alto de compatiblidad con ipchains, su predecesor de los kernel 2.2 de linux.

Con iptables meteremos las reglas en el sistema de filtrado y NAT de Linux. Mediante comandos, iremos insertando, borrando o modificando reglas, cada llamada a iptables ejecuta una accion, asi que lo normal es crear un script que vaya insertando todas las reglas para configurar la politica de seguridad de nuestro sistema. Actualmente ya existe con iptables una forma de "grabar" la información de la politica del sistema, aunque si no automatizamos esto, y apagamos la maquina, perdemos esa información, por eso realizar la política dentro de un script es mas que una buena idea, una necesidad. Aunque existe una herramienta para grabar y cargar la configuracion automáticamente, llamada **iptables-save** e **iptables-restore**, la practica de un script es siempre recomendable, ya que dentro de un

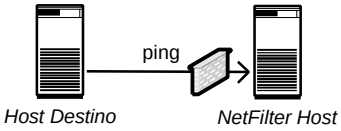
script se puede comentar cada regla de la politica de seguridad. Hablaremos de iptables-save e iptables-restore en el capítulo dedicado a hablar de herramientas externas de Iptables.

Lo primero que deberiamos hacer es un *man iptables* para hacernos una idea de la cantidad de parámetros que tiene y ver su sintaxis.

Antes de seguir, veamos un ejemplo práctico que nos servirá para ir haciéndonos una idea de cómo va esto:

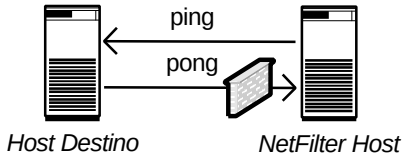
Imaginemos que lo unico que necesitamos es que nadie nos pueda hacer ping.

```
iptables –A INPUT –p icmp –j DROP
```



Con este comando, hemos añadido una regla a la cadena INPUT (ya hablaremos mas delante de lo que son las cadenas), que cuando analiza un paquete ICMP lo manda a cadena DROP. La cadena DROP es un "destino" similar a un /dev/null es decir, lo tira sin dejar rastro.

Quizas nos preguntemos que puede pasar si somos nosotros los que hacemos un ping a otro host, pues bien, la respuesta quizás no sea tan obvia, pero es lógica: no obtendremos respuesta.



NETFilter es un filtrado de paquetes con inspección de estados. Si recordamos lo que es un filtrado de paquetes básico recordaremos que simplemente analiza los paquetes hasta el nivel cuatro de la pila DARPA/OSI, es decir, comprueba la IP de origen, la IP de destino, el puerto de origen y el puerto de destino. No "piensa" a nivel de sesión, en este caso el paquete de vuelta de nuestro ping, lo estaria tirando por la regla que acabamos de meter en el firewall.

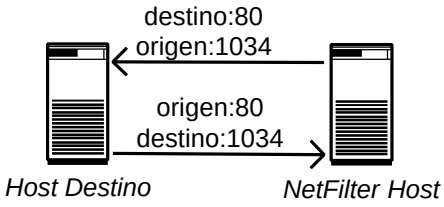
Decimos que tambien Netfilter hace inspección de estados, aquí esta realmente la potencia de Netfilter. Un filtrado común lo implemente hoy en dia cualquier sistema operativo, incluyendo los de Microsoft. Un firewall puede ser simplemente un filtrado de paquetes o puede ser un filtro de paquetes *inteligente*, que recuerda que hubo un paquete de ida y esperara el paquete de vuelta. A esto llamamos *inspecciones de estados.*

Inspeccion de estados en Netfilter

La inspección de estados no es propia de netfilter, por supuesto cualquier firewall que se precie la implementa. En Netfiler es muy flexible y se integra con la potencia del filtrado a bajo nivel de Netfilter.

En el ejemplo anterior vimos que si en una regla deshabilitamos el protocolo ICMP (en este caso en la cadena de entrada (INPUT)), cuando nosotros queremos salir fuera (anticipo que lo hariamos por la cadena OUTPUT, aunque de esto todavia no hemos visto nada), podriamos salir ya que en principio no hay ninguna regla en la cadena de salida (OUTPUT), pero el problema estaria en la contestacion a nuestro *ping*. En este caso nunca tendriamos contestacion porque el paquete de vuelta ICMP se filtraría e iria a la papelera de nuestro firewall: la cadena DROP.

¿Cómo podemos decirle a nuestro firewall que deje pasar las conexiones que yo ya he establecido?. Bien, la solucion mas rápida puede ser añadir una regla en la cadena de entrada (INPUT) que deje pasar los paquetes, pero eso dejaria pasar TODOS los paquetes. Si recordamos el propósito inicial de nuestra política de seguridad, era simplemente evitar que nos hicieran ping.



¿ Como podemos hacerlo ?, parece evidente que si añadimos una regla que deje entrar ICMP para que nos respondan, tambien entraran para pedirnos contestacion (luego veremos mas adelante que podemos distinguir entre un echo-reply y un echo-request, pero esto ahora no nos interesa).

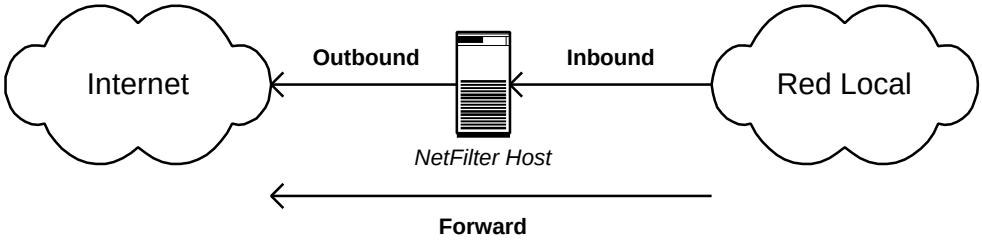
Lo que tenemos que hacer es decirle a nuestro firewall que deje pasar las conexiones establecidas. Esto se hace con un parámetro especial, que sirve exclusivamente para eso, de esta forma, dejariamos entrar las conexiones pre-establecidas, y el resto las cortaremos. Si hablamos de ICMP no estamos hablando de una sesión real, como podría ser TCP, pero si de una sesión "virtual" implementada por netfilter para reconocer la vuelta de los paquetes.

La implementación de esto desde un principio, seria con los comandos siguientes

```
iptables –A INPUT –t state –ESTABLISHED –J ACCEPT
iptables –A INPUT –p icmp –J DROP
```

Cadenas en Netfilter

Las cadenas son los lugares donde se interpretan las reglas de la política de seguridad. No es lo mismo un paquete que se genera en la propia maquina del firewall, que un paquete que atraviesa el firewall desde un interfaz de entrada, hasta un interfaz de salida, que un paquete que tiene como destino la máquina del firewall.



También podremos crear cadenas de usuario, aquí ya no hablamos del punto donde evaluamos el paquete, sino que hablamos de cadenas que derivaremos para organizar mejor nuestras reglas. Ahora veremos como utilizar la potencia que nos dan las cadenas.

Tenemos tres cadenas predefinidas que definen el punto de aplicación de la inspección del paquete.

Cadena de Entrada (INPUT) : La cadena de entrada tiene como destino la maquina del firewall, en cualquier de sus multiples interfaces. Los paquetes que se evalúen por esta regla siempre entrarán por el llamado intefaz de entrada. En esta regla no existe un interfaz de salida, ya que los paquetes solo entran.

Cadena de Salida (OUTPUT) : La cadena de salida tiene como origen de los paquetes la máquina del firewall en cualquiera de sus interfaces de red. Cuando originemos un paquete desde nuestro firewall, estableciendo una conexión con otro host, siempre se evaluará en esta cadena. En este tipo de cadena solo tendremos un interfaz de salida.

Cadena de tránsito (FORWARD): La cadena de transito evalúa los paquetes que no están incluidos en las otras dos cadenas anteriores. Es evidente que lo que evalua son conexiones que pasan a través de nuestro firewall, pero que ni su origen ni su destino final son la maquina del firewall. En toda paquete que cumple esta regla siempre tendremos un interfaz de entrada y un interfaz de salida.

Tenemos tambien varias cadenas predefinidas para tomar una acción con los paquetes, difieren en el concepto de las anteriores en que si aquellas definian el punto de interpretación del paquete, éstas definirán que haremos con el paquete que cumple la regla.

Cadena DROP. Tira el paquete. Cuando enviamos el paquete a esta cadena, le estamos diciendo al firewall que lo tire. Des esta forma el paquete finaliza ahí su transito, y no llega a niveles superiores de la pila DARPA/OSI o no es pasado a ninguna otro dispositivo de red.

Cadena REJECT. Tira el paquete e informa de ello al emisor. Es muy similar a la cadena DROP, sólo que informa al emisor (mediante un mensaje ICMP) que el paquete ha sido rechazado en el firewall. Podemos especificar el tipo de mensaje ICMP que queremos enviar (p.e enviar un icmp-net-unreachable o un icmp-net-prohibited) o tambien cerrar la conexión con un reset. Podemos consultar el man para mas información.

Cadena ACCEPT. Acepta el paquete y lo deja pasar. En este punto el paquete no se evalua mas desde el punto de vista del Firewall. Si el paquete tiene como destino final el firewall, pasara a capas superiores de la pila DARPA/OSI, si el paquete esta atravesando el sistema, se forwardeara a la tarjeta de salida para que desde ahí salga a su próximo destino.

Ademas de estas cadenas, que llamaremos cadenas finales, ya que tomen la accion que tomen el paquete ya no se evalúa mas, tenemos otro tipo de cadena de destino.

Cadena LOG. Registra la informacion de ese paquete y continua evaluando secuencialmente. La cadena LOG sirve para apuntar cierto tipo de transito en nuestros logs, veremos mas adelante como implementar esto de una forma efectiva y sin que nos llene de informacion inútil nuestro disco duro.

Como manipular cadenas en IPTables

Hemos hablado de cadenas, crearlas y de forma genérica cuales son y para que sirven, veamos ahora como usarlas.

-N, --new-chain <nombre_de_cadena>, crearemos una cadena.

-X <nombre_de_cadena>, borramos una cadena.

-I, --insert <nombre_de_cadena> { regla } , insertamos una regla al *principio* de una cadena.

-A, --append <nombre_de_cadena> { regla } ,insertamos una regla al *final* de una cadena.

-D, --delete <nombre_de_cadena> {# | regla } , borramos una regla de la cadena dada. Podemos especificar la posición (1, 2 ..) o la regla en formato convencional.

-R, --replace <nombre_de_cadena> { # regla } ,reemplazamos una regla de la cadena especificada. Hay que especificar el numero de la cadena que queremos reemplazar y luego la regla a introducir.

-L, --list <nombre_de_cadena>, lista todas las reglas pertenecientes a la cadena dada.

-F, --flush , elimina todas las reglas, es lo mismo que eliminar todas las reglas una por una.

Ahora veremos que tipos de regla tenemos para insertar, y que opciones son las comentadas en los párrafos anteriores.

Reglas de filtrado

Las reglas de filtrado son la base del Firewall. Hay dos tipos de reglas, reglas de filtrado de paquetes, que determinan básicamente que paquetes son aceptados y que paquetes rechazados, y reglas de NAT o de traducción/manipulacion de direcciones IP y otros datos del paquete. En NetFilter se le denomina genéricamente “packet mangling”, término que podríamos traducir por “manipulacion” del paquete en sentido genérico, ya que podemos cambiar datos como las direcciones IP, los puertos, datos de la cabecera del paquete y datos estadísticos del paquete IP.

Las reglas de filtrado son complejas, no exactamente por su sintaxis (que es bastante sencilla) sino porque generalmente en una política de seguridad seria, tienen que estar muy bien estructuradas para ser eficaces y/o eficientes. En primer lugar deberemos tener en cuenta que cuando añadimos reglas a una cadena, estas se evaluarán desde la inicial a la final, en un bucle, que terminará cuando se llegue a la última regla, o cuando una regla se ejecute. Diremos que una regla se ejecuta cuando llega a una condición donde la regla se cumple. Cuando una regla se ejecuta siempre se acomete una acción, de este modo se puede decir que la estructura de una regla es la siguiente:

SI el paquete X { **que entra por la cadena Y** } cumple { **condicion A** } [y **condicion B**] entonces { **acción** }

El paquete X es el que se está evaluando en este momento (se evalúa cada paquete por separado, obviamente), y éste entra por una cadena "Y". Puede ser INPUT, OUTPUT o FORWARD como vimos anteriormente. Esta es la primera condicion que nos vamos a encontrar. Las otras condiciones pueden ser únicamente una o varias, donde siempre se han de dar todas a la vez (condicion AND). La acción por lo general es ejecutar una cadena, lo mas común, a la cadena ACCEPT o a la cadena DROP.
Esto no es exactamene asi (sino que es mas complejo) en el caso de las reglas de NAT, donde mas adelante haremos hincapié en las diferencias y añadidos a estas explicaciones.

Por último quedan las condiciones, que realmente son el grueso de las reglas de filtrado, podemos filtar por muchas características, pero las más comunes son:

- Filtrar por interfaz (según porque interfaz llegue el paquete, o por cual salga, o ambas).
- Filtrar por direccion IP, de destino, de origen o ambas.
- Filtrar por puerto TCP, UDP o tipo de mensaje ICMP.
- Filtrar en funcion de bits de la cabecera IP (TTL, TOS) y TCP (SYN,ACK,FIN,etc…)
- Filtrar en funcion de otros campos, en los que no entraremos en detalle dada su complejidad.

Filtrando por interfaz

Lo más común es que vuestro firewall tenga unas tres interfaces o más, generalmente, estos interfaces tendrán nombres como hme0 (solaris), eth0 (intel), etc. Estos son los nombres físicos que utilizaremos para evaluar la condición. Podemos evaluar si estamos diciendole que el paquete *entra* por la interfaz o si *sale* por la interfaz.

Parámetros:

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

Le decimos que el paquete se evalua si entra por la interfaz dada. Esto funciona en paquetes que entran por las cadenas INPUT, FORWARD y PREROUTING. Si se usa el argumento !, significa que se niega la condicion, esto es, la regla se cumpliría si entrara por cualquier interfaz menos la especificada.
Si el nombre del interfaz termina con un "+" cualquier interfaz cuyo nombre comience por la cadena especificada, cumplirá la condición.

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

Le decimos que el paquete se evalua si sale por la interfaz dada. . Esto funciona en paquetes que entran por las cadenas OUTPUT, FORWARD y POSTROUTING.

Veamos un ejemplo práctico. Queremos filtrar todos los paquetes de entrada la máquina física del Firewall y dejar pasar los que van en tránsito. Suponiendo que el interfaz externo es el eth0, las reglas podrían ser asi:

```
Iptables –A INPUT –i eth0 –J DROP
Iptables –A FORWARD –i eth0 –J ACCEPT
```

Es un ejemplo un poco pobre, porque habria que añadir mas reglas, tales como una regla para que los paquetes sepan volver, es decir, lo que comentamos al principio sobre las sesiones establecidas, una regla tal como:

```
Iptables –A FORWARD –m state -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Generalmente las reglas de filtrado por interfaz son demasiado genéricas como para usarlas solas, se combinan con otras reglas, como las de direcciones IP o servicio. Ahora veremos como aplicarlas juntamente con las de direcciones IP para evitar el Spoofing.

Filtrando por Direccion

Podemos filtrar por la direccion de Destino o la direccion de Origen, y las direcciones que filtremos pueden ser rangos, o direcciones IP sueltas o direciones físicas Ethernet (MAC).

Para esto disponemos de las siguientes opciones:

--mac-source [!] dirección

Compara que el origen sea de la direccion fisica ethernet (MAC) en la forma XX:XX:XX:XX:XX:XX. Obviamente, solo podemos comprarar MAC de origen y esto se hace en las cadenas INPUT, FORWARD y PREROUTING.
Con el parámetro ! delante, haremos que se niegue el argumento, es decir, tener el sentido inverso, en este caso, seria decir, "cualquier direccion MAC diferente de esta, cumple la condición".

-s, --source [!] <direccion_ip>
-d, --destination [!] <direccion_ip>

Donde aquí especificamos el origen o el destino de los paquetes para cumplir esta condición. Podemos especificar un nombre de host o de red (que sea posteriormente resuelto por DNS, cosa que enlentecerá el proceso), una direccion IP simple o una direccion IP con máscara numérica decimal de la forma /xx donde xx será el numero de bits de la máscara. P.e, una mascara de tipo C, 255.255.255.0 sería /24.
El parámetro ! si se pone delante, actúa de negador igual que en los casos anteriores, es decir especifica cualquier direccion excepto la especificada.

Para evitar el spoofing podemos combinar las reglas de interfaz con las reglas de direccion IP. El caso es el siguiente. Nuestra red interna (192.168.1.0/24) donde tenemos los servidores esta accesible a través de la interfaz eth1 y la interfaz externa es eth0. Ambas estan conectadas de forma separada fisicamente, por medio de diferentes VLANes en un Switch o en diferentes Hubs. Queremos evitar que entre ningun paquete con dirección IP local (192.168.1.0/24) por la interfaz eth0, lo que podría ser un intento de spoofing. Las reglas serían las siguientes.

```
Iptables –A FORWARD –i eth0 –s ! 192.168.1.0/24 –j ACCEPT
Iptables –A INPUT –i eth0 –s ! 192.168.1.0/24 –j ACCEPT
```

Filtrando por Puertos y/o mensajes ICMP

Habitualmente el "grano fino" de una politica de seguridad se centra en que servicios pueden ser accedidos con según que IP's a través de según que interfaces. Para esto tenemos el filtrado de "servicios", por llamar de forma genérica a toda la gama de protocolos TCP, UDP y de mensajes ICMP que podemos filtrar con Linux/Netfilter.

Entremos en detalle. Primero distingamos los paquetes filtrados por TCP/UDP que se filtran por número de puerto o rango de puertos y los mensajes ICMP, que se filtran por tipo de mensaje. Primero hablemos de los filtrados de puertos TCP/UDP.

Cualquier conexión TCP o UDP lleva un puerto asignado. Cuando se establece una comunicación entre dos puntos de la red, existen dos sockets, que son una IP y un puerto. Tendremos un socket origen y un socket destino, conocidos por sus direcciones IP y sus puertos. A menudo se habla de los puertos como si sólo existiera el puerto destino, ya que es lo que mas acostumbrado nos tienen las documentaciones de diversos productos. Tenemos que tener en cuenta que en una sesion Http, nuestro navegador contacta con un servidor remoto, con una ip destino tal como "yahoo.com" (214.132.42.8) y un puerto destino (80). Nuestro navegador tendra tambien una ip origen (110.10.4.89) y un puerto origen (1240). En según que tipo de conexiones puede ser importante conocer y tener en cuenta este factor, tal como es el caso de querer filtrar las conexiones activas de una sesion FTP, donde la comunicación TCP de datos se establece desde el servidor FTP con un puerto de origen 20 a un puerto de destino negociado por el protocolo. Ahí deberiamos tener en cuenta el puerto de origen y no el de destino, como es habitual.

```
--protocol { tcp | udp } [–-sport | --dport [!]n:m]
```

Podemos establecer una condicion para filtar genericamente protocolos tcp o udp, pero no tendria mucho sentido, ais que necesitamos especificar el puerto de origen y/o el de destino. Los puertos pueden ser espeficados de forma individual o por rangos, especificandolo con rango desde n a m.

Un ejemplo de esto, para permitir conexiones HTTP que pasen a traves del firewall hacia la maquina 192.168.1.9

```
iptables –A FORWARD –d 192.168.1.9 --protocol tcp --dport 80 -j ACCEPT
```

Otro ejemplo de esto, para permitir conexiones SMB que pasen a traves del firewall:

```
iptables –A FORWARD --protocol udp --dport 137:139 -j ACCEPT
iptables –A FORWARD --protocol tcp --dport 137:139 -j ACCEPT
```

(Se podria afinar mas, ya que no todas las comunicaciones son UDP y TCP por esos tres puertos, pero es una buena aproximación).

Hablemos del filtrado de mensajes ICMP, su sintaxis es la siguiente:

```
--protocol icmp –icmp-type [!]{tipomensaje}
```

El filtrado por mensajes ICMP es bastante parecido, solo que en vez de especificar puertos, especificaremos tipos de mensaje. Los tipos de mensaje ICMP son los siguientes:

echo-reply, destination-unreachable, network-unreachable, host-unreachable, protocol-unreachable, port-unreachable, fragmentation-needed, source-route-failed, network-unknown, host-unknown, network-prohibited,host-prohibited, TOS-network-unreachable, TOS-host-unreachable, communication-prohibited, host-precedence-violation, precedence-cutoff, source-quench, redirect, network-redirect, host-redirect, TOS-network-redirect, TOS-host-redirect, echo-request, router-advertisement, router-solicitation, time-exceeded (ttl-exceeded) , ttl-zero-during-transit, ttl-zero-during-reassembly, parameter-problem, ip-header-bad, required-option-missing, timestamp-request, timestamp-reply, address-mask-request, address-mask-reply.

Son bastantes, y la mayoria de ellos no deberiamos dejarlos pasar, ya que en pocos y contados casos se utilizan algunos que no sean los relativos al ping (echo-reply, echo-request) o network-unreachable y time-exceeded. Algunos como “redirect” pueden ser en si mismos parte de un ataque. Asi que lo mas sabio es permitir los mensajes justos a las maquinas que lo necesiten.

Si queremos permitir un ping simplemente, lo más lógico es dejar pasar esos dos mensajes. Podemos pensar en el sentido del ping, ¿hacia donde se hace el ping?. De esta pregunta derivará una regla que deje pasar un mensaje en un sentido y el otro mensaje en el sentido contrario, quizas sea rizar mucho el rizo, pero si nos hacen un Flood de ICMP (muchos pings, grandes y seguidos), quizas os acordeis de este consejo.

Una regla que dejara entrar un ping a una maquina 192.168.1.8 y que esta pudiera contestar seria:

```
Iptables –A FORWARD –d 192.168.1.8  –p icmp –icmp-type echo-request –j ACCEPT
Iptables –A FORWARD –s 192.168.1.8 –p icmp –icmp-type echo-reply –j ACCEPT
```

Igual sólo quereis evitar redirects sobre la maquina del firewall, con lo que una regla como la siguiente bastaría.

```
Iptables –A INPUT –p icmp –icmp-type redirect –j DROP
```

Filtrando por Bits de la cabecera IP y TCP

No soy ningún gurú de las comunicaciones, pero quien no sabe lo que es un ataque basado en comportamientos no planifados de los protocos (SYN Flood, p.e), de esto se trata este apartado, como evitar ciertos comportamientos anormales a la hora de recibir paquetes con cabeceras malformadas.

Empecemos por el filtrado de las cabeceras TCP, que es lo mas común y útil. Aquí hablamos de “flags”, refiriendonos a los bits de la cabecera TCP codificados en forma de cadena para entendernos.

--tcp-flags [!] lista-a lista-b

Comprara la lista de Flags especificada en "lista-a", cada flag separado por comas, y si se da el caso que los flags de la cabecera TCP del paquete estan contenidos en la lista-A, entonces se cumple la regla y se reescriben los flags con los contenidos en la lista-B. Los flags han de separarse con comas, y los tipos posibles son los siguientes: SYN ACK FIN RST URG PSH ALL NONE.

Lo mas obvio que nos puede ocurrir es tirar a la basura directamente todos los paquetes que llegen con los todos los bits habilitados o los que llegan sin un solo bit.

```
Iptables –A input –p tcp –-tcp-flags ALL –j DROP
Iptables –A input –p tcp –-tcp-flags NONE –j DROP
```

Y bueno, como experimento, podemos intentar regular el trafico de paquetes de intentos de conexión (SYN) y limitarlo a un numero máximo por unidad de tiempo. Para eso existe un modulo de limite de tiempo que no he explicado, pero que introduciré aquí y que terminaré de explicar cuando hable del loggin.

```
-m limit --limit 10/m
```

Limita el trafico a 10 paquetes por minuto, ¿un poco escaso no?. Habrá que valorar el tipo de tráfico que queremos dejar pasar y a partir de ahí restringir los SYN que podran pasar. Si tenemos unas 15 conexiones por segundo, podemos poner el maximo en 50, de forma que una regla como la siguente podria ser una regla anti-SYNFlood:

```
iptables –A input –d servidorweb –p tcp –-dport 80 –-tcp-flags SYN –m limit –
limit 50/s –j ACCEPT
```

En este caso dejamos pasar solo 30 paquetes SYN por segundo. Si alguien nos hace un SYNFlood con una media de unos 500-700 paquetes por segundo, el servidor WEB ni se enterará (quizas perdamos conexiones lícitas nuevas durante el tiempo que dure el ataque), pero no se vera afectado por un posible ataque de denagación de servicio. Quizas nos llenen el ancho de banda de entrada pero nada más.

Tambien, como curiosidad (nunca lo he usado), podemos filtrar en funcion del tipo de servicio que tenga asignado el paquete (TOS), esto se hace simplemente:

-m tos --tos <TipodeServicio>

Network Address Translation (NAT)

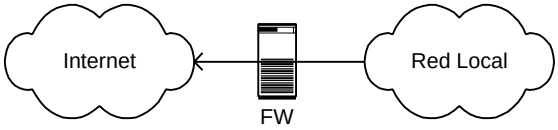
El NAT consiste simplemente en modificar las direcciones de destino o de origen de los paquetes IP. El NAT tambien puede modificar los puertos de origen o de destino de los paquetes TPC/UDP, esto se suele llamar PAT (Port Address Translation).

El NAT es una de las cosas que más suelen complicar las redes cuando éstas alcanzan proporciones considerables. Se puede decir que el NAT nace por la necesidad urgente de ahorrar direcciones IP públicas. Si alguien todavía no conoce el problema existente debido a la escasez de direcciones IP públicas, deberia replantearse el seguir leyendo este documento.

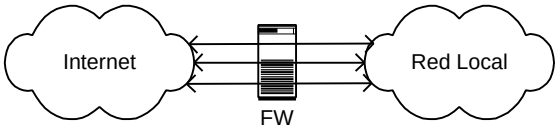
A pesar de que la idea original fue el ahorro, el NAT ofrece muchas ventajas, sobre todo a la hora de securizar redes, y es que si nuestra red interna está realizada con IP's no direccionables desde el exterior, tendremos un punto de ventaja sobre nuestros posibles agresores. El primer trabajo de un hacker es obtener toda la información posible sobre su victima, esto incluye un mapa de red. Si nosotros nos ocultamos en internet tras una sola IP pública, el hacker tendra que pasar por esa IP. Si nosotros tenemos todas nuestras maquinas, conectadas a la red directamente, cada una con una IP publica, podra rastrearlas todas directamente. Es una primera aproximación a las ventajas desde el punto de vista de la seguridad.

Existen varios tipos de NAT según varias varíen sus características, además, por si fuera poco, cada fabricante, programador o autor los llama de una manera diferente, asi podremos encontrarlos que unos lo llaman *masquerading,* otros *dynamic-nat*, o simplemente *smart-nat*¸ *hide-nat, static-nat* y un largo etcétera más. Seguramente te suene haber oído uno o mas de esos nombres, pero al final todos hacen resumen de dos tipos de nat:

? **NAT de 1 a N**, donde podemos utilizar una sola direccion para N maquinas. Esto es util si tenemos una sola IP publica y queremos utilizarla para salir por ella con todas las maquinas de nuestra red local. Aunque en algunos fabricantes solo puede ser un nat de salida, tambien veremos como hacerlo de entrada en función de una multiplexación por puerto



? **NAT de 1 a 1**, esto puede ser util para asignarle una IP publica a uno de nuestros servidores de la DMZ, tanto para conexiones entrantes como para conexiones salientes.



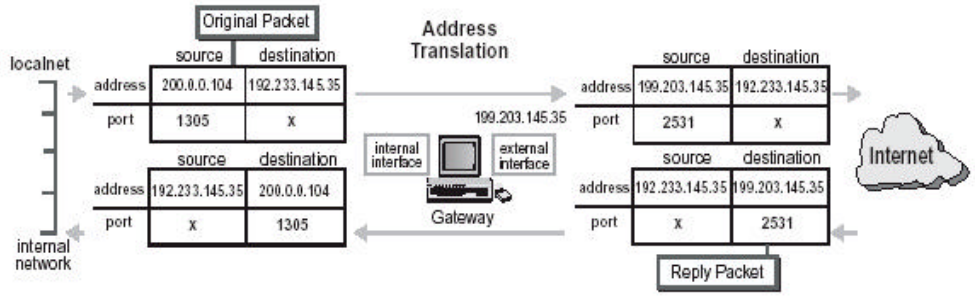
Además de por su cardinalidad, podemos identificar dos tipos de NAT, que suelen estar mas o menos relacionados con la clasificacion anterior. Hablamos de si un NAT puede ser en **origen** o en **destino.**
Cuando decimos que un NAT es en origen, lo llamamos SourceNAT o simplemente **SNAT.** Este NAT lo que hace es modificar la direccion de origen del paquete. El origen está situado siempre en nuestra red local, porque si perdemos la perspectiva de donde esta el origen y donde esta el destino, no se diferenciaria del NAT de destino o **DNAT.** El DNAT por supuesto, es modificar la direccion IP del destino. Veamos algunos ejemplos reales de para que necesitamos hacer esto

Source NAT

¿Para que querríamos hacer un SNAT 1:N?.

Si tienes 10 maquinas en tu red local, con un direccionamiento tal que 192.168.1.0/24 y una máquina con un modem que se conecta a un proveedor que le da una IP dinámica cada vez que se conecta, igual te interesa salir a internet con todas las maquinas de tu red local. Esto, claro está lo puedes hacer instalando un proxy, pero no estamos hablando de ese caso, que por otro lado siempre estara mas limitado que salir utilizando NAT (la discusión de el porqué es asi no entra dentro de los objetivos de este documento, pero si no te lo crees, siempre puedes ponerme un email).

Si quieres salir limpiamente, por mucho que enrutes una de tus maquinas (por ejemplo 200.0.0.104) a traves de la maquina que tiene el modem (por ejemplo 200.0.0.1), cuando el paquete salga por internet en cuanto llegue al router de tu proveedor o cualquier otro, este lo tirará y ahí se quedará tu intento de conectar, en el mejor de los casos el paquete podria llegar al destino, pero jamás sabría volver. ¿Por qué?, porque ese paquete tiene una direccion de origen inválida en la red publica de internet, tienes que hacer que todos los paquetes que salen a internet tengan una direccion de origen válida, en este caso solo disponemos de una, la IP que te da tu proveedor, asi que lo que haremos es cambiar la direccion de los paquetes de tu maquina 200.0.0.104 y hacer que la IP de origen sea la publica. Las conexiones vuelta de nuevo pondrán la IP local que tenia la conexión originalmente. Aquí tienes un grafico que ilustra bastante bien este tema.



¿Cómo sabe el Firewall que direcciones originales tenian las conexiones?. Digamos simplemente que hay una mecánica detrás que hace que esto sea posible y no haya error. Existen unas tablas de direcciones, y el mecanismo básicamente consiste en que la sesión *NATeada* tiene un nuevo puerto de origen, y ese es el identificador que utilizar el gateway que hace NAT para identificar la máquina original de la conexión y su IP local.

El SNAT de 1:N es lo que en Linux se suele conocer por masquerading, aunque el termino masquerading es mas amplio y abarca a todos los tipos de NAT. En otros sistemas se conoce por HideNAT (Checkpoint) o DynamicNAT (Cabletron, y otros) o SmartNat (radware). El SNAT de 1:N es por otra parte el NAT mas sencillo de montar. Para ello solo necesitamos una máquina con una IP pública y ponerla de gateway por defecto en el resto de máquinas que quieran salir por ella, en maquinas linux:

```
route add default  gw x.x.x.x
```

Donde X.X.X.X es la direccion interna (de la LAN) de la maquina que hará NAT, que será nuestro firewall con NetFilter.

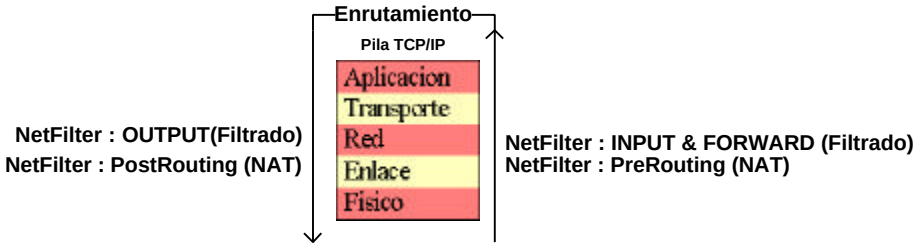
Nunca debemos olvidar que una cosa es el NAT y otra cosa es el filtrado, es decir, el NAT no tiene que ver en principio con la seguridad, es simplemente un elemento más del networking de nuestro sistema. Las cosas parecen sencillas pero nunca lo son, y veremos como podemos securizar nuestra red de varias maneras desde el punto de vista del networking puro y duro, una es mediante el uso de NAT restrictivo y otra es mediante el routing.

Asi pues, si queremos salir fuera, deberemos tener en cuenta dos puntos.

- Si podemos salir (seguridad). Habrá que ver si tenemos reglas de filtrado que nos permitan salir.
- Si podemos salir (networking). Habrá que ver si hacemos NAT para esas maquinas.

Este es el orden en el que NetFilter trabaja, recomiendo que se estudie con cuidado, porque no todos los sistemas de NAT funcionan igual, unos hacen el NAT al final, como Checkpoint FW-1 mientras que otros según el tipo de NAT lo hacen antes o después. En NETFilter siempre es lo primero, de forma que las reglas de Filtrado siempre se harán teniendo en cuenta que las IP's son las que quedan después de hacer el NAT.

Otra cosa muy importante que debemos tener en cuenta es que una conexión que se transforma en un NAT sigue la el concepto de sesión que discutimos anteriormente, es decir, aunque la comunicación sean dos flujos de paquetes IP, uno con direccion de origen en la IP de nuestro host, y otro con direccion de destino con la IP de nuestro host, debemos tener en cuenta solamente el sentido original de la comunicación, es decir, que la vuelta de nuestros paquetes no deberá ser transformada mediante NAT sino que se hara automáticamente.



¿Cómo implementamos esto con NetFilter?.

Netfilter tiene una tabla se que usa para decirle al interfaz que las reglas que se introducen no son de filtrado sino de NAT, esta tabla tiene por defecto una serie de cadenas.

Cadena PREROUTING, esta es la primera cadena que se procesa. Aquí se hace el NAT de destino (DNAT), se modifica la IP del destino y se pone la transformada, que será una IP de alguna de nuestras LAN.

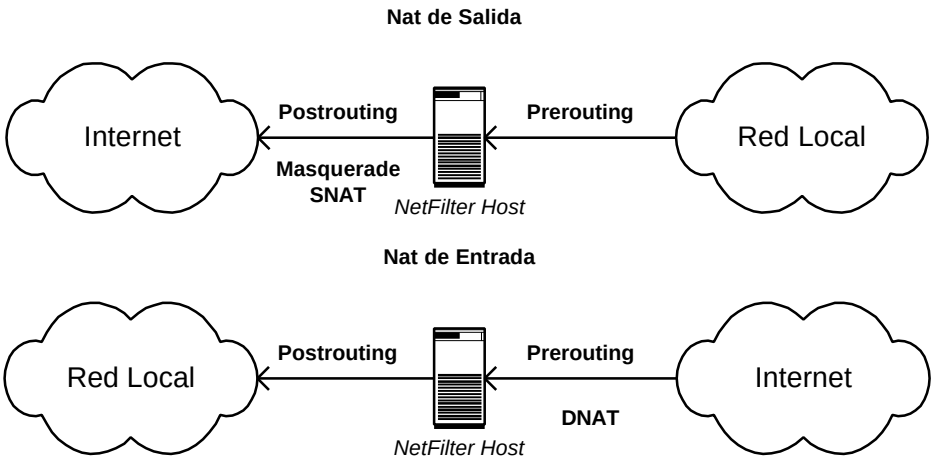
Cadena POSTROUTING, esta es la última cadena que procesa el sistema. En ella podemos tener dos tipos de cadena de salida, o MASQUERADE o SNAT. Aquí es donde se cambia la IP de salida del paquete que sale (OUTPUT chain) o que atraviesa nuestro host (FORWARD).

Cadena MASQUERADE. Es una cadena de destino, y es igual que SNAT excepto porque si utilizamos esta cadena, y el interfaz está caido no la guardará en la tabla de conexiones pendientes. Si tienes una IP dinámica es recomendable que uses esta cadena en vez de la cadena SNAT.

Cadena SNAT. Es una cadena de destino y es usada para modificar la direccion de origen del paquete, igual que la cadena MASQUERADE. A diferencia de esta ultima, si el interfaz esta caido, simplemente irá guardando los paquetes para que cuando puedan salir, siga procesandolos.

Cadena DNAT. Es una cadena de destino, y es usada para modificar la direccion de destino del paquete. Se evalúa en la cadena PreRouting y es la primera que examina NETFilter.

Esta es la representacion gráfica de donde actual de las cadenas de la tabla nat según el tipo de tipo de NAT:



La sintaxis de los comandos de NAT de NetFilter es practicamente igual a la de filtrado:

Sintaxis:

```
iptables -t nat -A PREROUTING  -j DNAT  --to {host_destino}
iptables -t nat -A POSTROUTING -j SNAT  --from {host_origen}
iptables -t nat -A POSTROUTING -j MASQUERADING --from {host_origen}
```

Este sería el esquema básico de las reglas de nat, aunque ahora habrá que filtrar según nos convenga esas reglas genéricas. Generalmente nos interesará filtrar por interfaz, puertos de destino, direcciones de origen y/o direcciones de destino.

Por ejemplo, si nos interesara sacar a nuestros usuarios de la red interna, que estan con direccionamiento 192.168.1.0/24, cuya red esta conectada en nuestro host con NetFilter a través de la eth1, podríamos usar esta regla:

```
iptables -t nat -A POSTROUTING -j SNAT –i eth1  --from <dir_pub>
```

o bien

```
iptables -t nat -A POSTROUTING -j SNAT –s 192.168.1.0/24 --from <dir_pub>
```

Donde <dir_pub> seria la IP pública por la que saldría hacia internet. En un mundo real, seguramente queramos regular la salida de nuestros usuarios, asi que utilizaremos reglas de filtrado para segurarnos de que los usuarios solo hacen lo que nosotros queramos, imaginemos que solo tienen que usar http y ftp:

```
iptables -A FORWARD -j ACCEPT –s 192.168.1.0/24 –p tcp –-dport 21
iptables -A FORWARD -j ACCEPT –s 192.168.1.0/24 –p tcp –-dport 80
```

Destination NAT

El DNAT difiere muy poco respecto al SNAT, además de lo obvio (que se cambia la dirección IP de destino en vez de la de origen), existe un cambio sustancial respecto al SNAT: podemos utilizar direcciones IP diferentes a las que tiene el firewall, es decir, podemos transformar entre direcciones y que ninguna de esas direcciones pertenezca al firewall. Esto se hace con un concepto de networking llamado proxy-arp.

Proxy ARP

¿Para que queremos liarnos mas?, bien, imaginemos que tenemos 6 direcciones IP públicas de las cuales una la utilizamos en nuestro firewall. Y las otras cinco queremos reservarlas para nuestros diferentes servidores de la DMZ, si nuestro rango va del 0 a 7, siendo la 0 la de red y la 7 la de broadcast, la 1 seria la del firewall y del 2 al 6 serian para nuestras máquinas. Nuestras máquinas utilizan direccionamiento privado y necesitamos que el firewall haga NAT, tanto DNAT como SNAT (el DNAT para que la gente desde fuera entre en nuestros servidores) y el SNAT para que los servidores que tengan iniciar conexiones hacia fuera (p.e: DNS) puedan hacerlo. Bien, el problema está con el mecanismo ethernet para entregar tramas.

Si nuestro firewall tiene las direcciones publica.1 y privada.1, si llega un paquete para publica.4 que es nuestro servidor WEB, el firewall como dispositivo ethernet que es, seguirá el siguiente algoritmo:

Algoritmo ethernet de entrega de trama

1) El servidor que tiene el paquete IP comprueba que la direccion de destino esta dentro de su misma red. Si no lo está, sale de este bucle y lo enruta hacia la máquina correspondiente (vuelta al **1).**
2) Si la IP destino está en la misma red, mira en su tabla de ARP a ver si tiene una direccion fisica (a nivel MAC) para la IP de destino. Si la tiene en la caché pasa al paso **5**
3) Si no tiene direccion MAC para ese host, intenta conseguirla mediante difusión ARP. Esto consiste en que difunde por su segmento ethernet un paquete ARP que contiene un campo a 1's en la direccion MAC y la direccion IP de destino en el campo IP (simplificando). Si existe un host que tenga esa IP, contestará a ese host con un paquete que contenga su direccion MAC, y su direccion IP.
4) Si el paso **3** ha encontrado un host con esa MAC, le entregará el paquete IP. Sino, descartará el paquete por no poder entregarlo al host final.

Si no existiera el Proxy-ARP el paquete para publica.4 no seria entregado nunca, el firewall nunca lo reclamaria como suyo, no lo podria transformar a prividada.4 y el circuito nunca seria completado: nunca funcionaria.

Proxy-ARP consiste básicamente en que la maquina intermedia (gateway) responde a una peticion de una IP con su direccion MAC. En este caso el firewall asociará la MAC de la interfaz que esta en el mismo segmento que el host que le entregará el paquete desde internet (el router), y la asociará a todas aquellas IP's publicas de las cuales esté haciendo DNAT, en esta caso publica.2-6.

Veamos un ejemplo;

```
eth0      Link encap:Ethernet  HWaddr 52:54:4C:03:E4:CD
          inet addr:212.122.142.182  Bcast:217.126.145.192  Mask:255.255.255.192
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14589007 errors:2 dropped:0 overruns:0 frame:192
          TX packets:13180750 errors:0 dropped:0 overruns:0 carrier:0
          collisions:1427 txqueuelen:100
          Interrupt:10 Base address:0x300
```

```
eth1      Link encap:Ethernet  HWaddr 00:A0:C9:4C:F9:09
          inet addr:192.168.1.254  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10646482 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9864649 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:9 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:77 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

¿Cuál es la interfaz con IP pública?, la eth0 claramente. Su direccion MAC es `52:54:4C:03:E4:CD` y si desearamos hacer proxy-arp las direcciones `212.122.142.183, 212.122.142.184 y 212.122.142.185,` tendriamos que asociar la MAC a esas IP's, esto podemos hacerlo de varias maneras:

- Ruta estática ARP
- Con IP's virtuales en el host.
- Con una ruta estática IP en el host que reenvía el paquete. De esta forma evitamos el mecanismo de ProxyARP y le decimos al router que envia el paquete que para llegar a publica.4 hay que pasar antes por publica.1, es un metodo "limpio", pero generalmente no tenemos esa posibilidad.

Lo segundo esta totalmente desaconsejado porque puede provocar problemas de routing y no funciona en todos los dispositivos que hacen NAT (aunque si en NetFilter). Para poner una ruta estática ARP, bastará con utilizar el comando arp, su sintaxis:

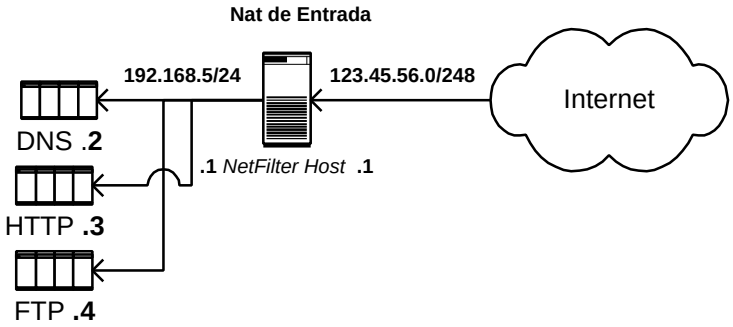
```
fw:/etc/firewall# arp --help
Usage:
  arp [-vn]  [<HW>] [-i <if>] [-a] [<hostname>]             <-Display ARP cache
  arp [-v]          [-i <if>] -d  <hostname> [pub][nopub]    <-Delete ARP entry
  arp [-vnD] [<HW>] [-i <if>] -f  [<filename>]              <-Add entry from file
  arp [-v]   [<HW>] [-i <if>] -s  <hostname> <hwaddr> [temp][nopub] <-Add entry
  arp [-v]   [<HW>] [-i <if>] -s  <hostname> <hwaddr> [netmask <nm>] pub  <-''-
  arp [-v]   [<HW>] [-i <if>] -Ds <hostname> <if> [netmask <nm>] pub      <-''-

        -a                      display (all) hosts in alternative (BSD) style
        -s, --set               set a new ARP entry
        -d, --delete            delete a specified entry
        -v, --verbose           be verbose
        -n, --numeric           dont resolve names
        -i, --device            specify network interface (e.g. eth0)
        -D, --use-device        read <hwaddr> from given device
        -A, -p, --protocol      specify protocol family
        -f, --file              read new entries from file or from /etc/ethers
```

En el ejemplo anterior, las tres rutas ARP que necesitariamos serían literalmente asi:

```
arp –s 212.122.142.183 52:54:4C:03:E4:CD pub
arp –s 212.122.142.184 52:54:4C:03:E4:CD pub
arp –s 212.122.142.185 52:54:4C:03:E4:CD pub
```

Con esto asociamos nuestra MAC externa a esas IP's, ahora cuando venga un paquete para esas IP's el host que tenga estas entradas en su tabla ARP, las reclamará como suyas y una vez que las tenga en su pila TCP/IP las procesará Netfilter convenientemente.

Bien, solucionado el problema de proxy-arp, imaginemos que queremos montar lo siguiente:



Supongamos la MAC del adaptador externo de nuestro FW es: `52:54:4C:03:E4:CD`

Primero pondremos las reglas estaticas de ARP:

```
arp –s 123.45.56.2 52:54:4C:03:E4:CD pub
arp –s 123.45.56.3 52:54:4C:03:E4:CD pub
arp –s 123.45.56.4 52:54:4C:03:E4:CD pub
```

Luego pondremos las reglas de NAT:

```
iptables –t nat –A PREROUTING –j DNAT -d 123.45.56.2 –to 192.168.5.2
iptables –t nat –A PREROUTING –j DNAT -d 123.45.56.3 –to 192.168.5.3
iptables –t nat –A PREROUTING –j DNAT -d 123.45.56.4 –to 192.168.5.4
```

Y finalmente las reglas de filtrado:

```
iptables –A FORWARD –j ACCEPT -d 192.168.5.2 –p udp –dport 53
iptables –A FORWARD –j ACCEPT -d 192.168.5.3 –p tcp –dport 80
iptables –A FORWARD –j ACCEPT -d 192.168.5.4 –p tcp –dport 21
```

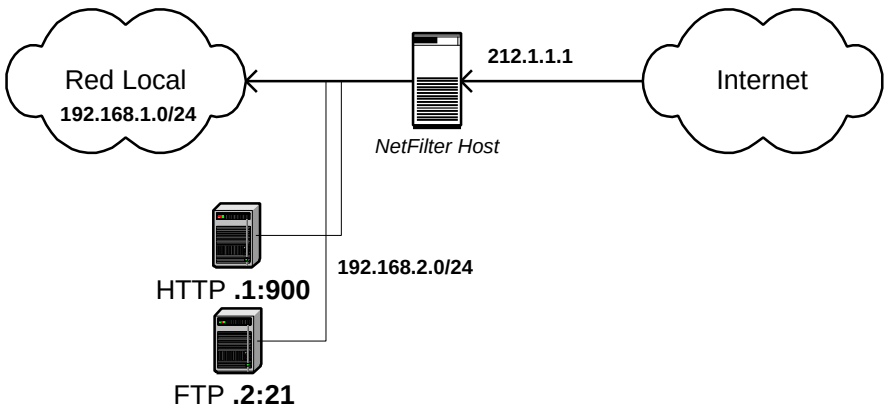
No olvidemos las reglas de aceptar sesiones establecidas para que puedan enviar los paquetes de vuelta :

```
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Port Multiplexing y PAT

Esto es una extensión del NAT, y sirve para por ejemplo, teniendo una sola IP pública, utilizarla para un numero N de servidores en conexiones entrantes, incluyendo el firewall y toda una red interna de M hosts que la usarán para salir fuera.

El PAT es hacer cambio de puertos. Podemos por supuesto, hacer PAT y NAT a la vez, imaginemos que necesitamos hacer un DNAT con la direccion pública del firewall y modificar el puerto de destino original por un puerto diferente:



La idea es que el servidor WEB externamente "parece" que esta escuchando por el puerto 80 como es standard. La url en un navegador seria http://212.1.1.1:80
Tambien tenemos el caso contrario, que es que tenemos un servidor FTP escuchando por el puerto standard (21) y queremos que sea visible como que escucha por el puerto 21000.

Las reglas de nat que necesitariamos serian las siguientes:

```
iptables -t nat -A PREROUTING -d 212.1.1.1 -p tcp --dport 21000 -j DNAT --to 192.168.2.2:21  # FTP
iptables -t nat -A PREROUTING -d 212.1.1.1 -p tcp --dport 80 -j DNAT --to 192.168.2.1:900    # HTTP
```

Si nos fijamos con anteción ¿Qué hemos hecho?, añadir :puerto en la direccion IP, es cuanto necesitamos para decirle que use ese puerto como puerto destino.

Script de ejemplo

Esta no seria una buena guía sin un script de ejemplo completo, de hecho es tan completo que es el que yo uso en mi sistema. Los datos reales estan cambiados, pero si teneis un poco de mala leche e imaginación seguro que sois capaces de deducir donde está funcionando este script. Sed buenos...

```
fw:/etc/firewall# cat fw-start
#!/bin/sh
# 15/Apr/2001 -> 20/Sep/2001
AUT="Sancho Lerena, <slerena@iname.com>"
VER="v6.4 Nov/01"

echo "Scripts de arranque del Firewall (Iptables/Netfilter) v" $VER
echo "Copyright (c) 2000-2002 " $AUT
echo "================================================================="

# ==================
#  Inicializacion
# ==================

# Resumen de Adaptadores en Gatekeeper
#
# eth0 - Adaptador externo
# eth2 - Adaptador a la DMZ
# eth1 - Adaptador a la Red Interna

MEFISTO="192.168.15.3"
CALIGULA="192.168.15.2"
VULCANO="192.168.15.4"
ZOID="192.168.10.1"
PRIVATE="192.168.10.0/24"
DMZ="192.168.15.0/24"
VIRTUAL="197.16.15.125"
DNS="192.168.15.4"
FIREWALL="192.168.15.1"
GATEKEEPER="192.168.10.254"
ETHER="197.16.15.180"
LOCALNET="192.168.0.0/16"
CASTOR="192.168.15.5"
POLLUX="192.168.15.6"
GEMINI="192.168.15.10"

date >> /var/log/fw-event.log
echo "Iniciando Script de arranque" >> /var/log/fw-event.log
logger "Iniciando Script de arranque del FW"

echo "Iniciando modulos"
insmod ip_nat_ftp > /dev/null


echo "Desactivando IP forwarding..."
echo 0 > /proc/sys/net/ipv4/ip_forward

echo "Borrando reglas actuales..."
iptables -F
iptables -F -t nat
iptables -X STEALTH > /dev/null
iptables -X NOFORWARD > /dev/null
iptables -X SPOOFING > /dev/null
iptables -X INPUT2 > /dev/null

echo "Creando cadenas auxiliCALIGULA"
iptables -N STEALTH
iptables -N SPOOFING
iptables -N NOFORWARD
iptables -N INPUT2
```

```
# ==================
# AntiSpoofing
# ==================

iptables -A SPOOFING -j LOG --log-level warning --log-prefix "Spoofing Detectado."
iptables -A SPOOFING -j DROP

echo "Activando proteccion AntiSpoofing en el FW"
iptables -A FORWARD -i eth0 -s $LOCALNET -j SPOOFING
iptables -A FORWARD -i eth1 -s ! $PRIVATE -j SPOOFING
iptables -A FORWARD -i eth2 -s ! $DMZ -j SPOOFING

# ============================
# Tiramos paquetes malformados
# ============================

echo "Activando proteccion de paquetes malformados"
echo "Block XMAS packets"
iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
iptables -A FORWARD -p tcp --tcp-flags ALL ALL -j DROP
echo "Block NULL packets"
iptables -A INPUT -p tcp --tcp-flags ALL NONE -j DROP
iptables -A FORWARD -p tcp --tcp-flags ALL NONE -j DROP

# ==================
#  NAT
# ==================

# Reglas de Source NAT para maquinas de la DMZ
iptables -t nat -A POSTROUTING -o eth0 -s $CALIGULA     -j SNAT --to $VIRTUAL
iptables -t nat -A POSTROUTING -o eth0 -s $MEFISTO      -j SNAT --to $VIRTUAL
iptables -t nat -A POSTROUTING -o eth0 -s $VULCANO      -j SNAT --to $VIRTUAL
iptables -t nat -A POSTROUTING -o eth0 -s $PRIVATE      -j SNAT --to $VIRTUAL
iptables -t nat -A POSTROUTING -o eth0 -s $CASTOR       -j SNAT --to $VIRTUAL
iptables -t nat -A POSTROUTING -o eth0 -s $POLLUX       -j SNAT --to $VIRTUAL


# Destination NAT para maquinas de la DMZ

iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 21 --to $VULCANO    # FTP
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 22 --to $VULCANO    # SSH
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 23 --to $VULCANO    # Telnet
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p udp --dport 53 --to $VULCANO    # DNS
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p udp --sport 53 --to $VULCANO    # DNS
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p icmp           --to $VULCANO    # ICMP
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 80 --to $VULCANO    # HTTP
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 25 --to $VULCANO    # SMTP
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 110 --to $VULCANO   # POP3
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 6346 --to $VULCANO  # GNUTElla

iptables -t nat -A PREROUTING -d $VIRTUAL -p tcp --dport 5900 -j DNAT  --to $MEFISTO     # VNC :0
iptables -t nat -A PREROUTING -d $VIRTUAL -p tcp --dport 21000 -j DNAT --to $MEFISTO:21  # FTP
iptables -t nat -A PREROUTING -d $VIRTUAL -p tcp --dport 5631 -j DNAT  --to $MEFISTO     # PcAnywhere
iptables -t nat -A PREROUTING -d $VIRTUAL -p tcp --dport 5632 -j DNAT  --to $MEFISTO     # PcAnyWherE

iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 33 --to $CALIGULA:23   # Telnet
iptables -t nat -A PREROUTING -d $VIRTUAL -j DNAT -p tcp --dport 5901 --to $CALIGULA    # VNC :1

#echo "HTTP Proxy transparente a traves de SQUID en VULCANO"
iptables -t nat -A PREROUTING -d ! $LOCALNET -s $PRIVATE -p tcp --dport 80 -j DNAT --to $VULCANO:8080
iptables -t nat -A PREROUTING -d ! $LOCALNET -s $ETHER   -p tcp --dport 80 -j DNAT --to $VULCANO:8080

# Filtrado: FORWARDING
# ====================

echo "Activamos filtrado de forward (FORWARD)..."
echo "Dejamos pasar las conexiones ESTABLECIDAS o RELATIVAS a las establecidas..."
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

echo "Las conexiones bidireccionales..."
iptables -A FORWARD -p icmp --icmp-type echo-reply -j ACCEPT                    # ping
iptables -A FORWARD -p icmp --icmp-type echo-request -j ACCEPT                  # ping
iptables -A FORWARD -p udp --dport 53 -j ACCEPT                                 # DNS
iptables -A FORWARD -p udp --sport 53 -j ACCEPT                                 # DNS
iptables -A FORWARD -s $LOCALNET -p udp --dport 161:162 -j ACCEPT               # SNMP

echo "Las conexiones entrantes hacia " $VULCANO
iptables -A FORWARD -d $VULCANO -p tcp --dport 21 -j ACCEPT      # FTP en CALIGULA
iptables -A FORWARD -d $VULCANO -p tcp --dport 22 -j ACCEPT      # SSH
iptables -A FORWARD -d $VULCANO -p tcp --dport 23 -j ACCEPT      # Telnet
iptables -A FORWARD -d $VULCANO -p tcp --dport 80 -j ACCEPT      # HTTP Apache puerto 80
iptables -A FORWARD -d $VULCANO -p tcp --dport 8080 -j ACCEPT    # HTTP Proxy SQUID
iptables -A FORWARD -d $VULCANO -p tcp --dport 25 -j ACCEPT      # SMTP
iptables -A FORWARD -d $VULCANO -p tcp --dport 110 -j ACCEPT     # POP
iptables -A FORWARD -d $VULCANO -p tcp --dport 6346 -j ACCEPT    # Gnutella
```

```
echo "Las conexiones entrantes hacia " $MEFISTO
iptables -A FORWARD -d $MEFISTO -p tcp --dport 5900    -j ACCEPT       # VNC
iptables -A FORWARD -d $MEFISTO -p tcp --dport 21      -j ACCEPT       # FTP en MEFISTO
iptables -A FORWARD -d $MEFISTO -p tcp --dport 5631:5632    -j ACCEPT # PCAnyWhere

echo "Las conexiones entrantes hacia " $CALIGULA
iptables -A FORWARD -d $CALIGULA -p tcp --dport 5901 -j ACCEPT # VNC
iptables -A FORWARD -d $CALIGULA -p tcp --dport 23   -j ACCEPT # SSH

echo "Las conexiones entrantes hacia " $POLLUX
iptables -A FORWARD -d $POLLUX -p tcp --dport 5901 -j ACCEPT # VNC
iptables -A FORWARD -d $POLLUX -p tcp --dport 23   -j ACCEPT # SSH

echo "Las conexiones entrantes hacia " $CASTOR
iptables -A FORWARD -d $CASTOR -p tcp --dport 5901 -j ACCEPT # VNC
iptables -A FORWARD -d $CASTOR -p tcp --dport 23   -j ACCEPT # SSH

echo "Trafico SMB sobre " $MEFISTO
iptables -A FORWARD -s $LOCALNET -d $MEFISTO -p tcp --dport 135:139 -j ACCEPT             # SMB
iptables -A FORWARD -s $LOCALNET -d $MEFISTO -p udp --dport 135:139 -j ACCEPT             # SMB

echo "Las conexiones de " $DMZ " a Internet "
iptables -A FORWARD -s $DMZ -d ! $LOCALNET -j ACCEPT

echo "Conexiones con origen en " $PRIVATE
iptables -A FORWARD -s $PRIVATE -j ACCEPT     # La red Interna tiene salida total. Ojito.

# ================================
# Filtado: NOFORWARD Security Rule
# ================================

echo "Activando registro del trafico cerrado"
iptables -A FORWARD -s ! $LOCALNET -j NOFORWARD

echo "Rellenando resto de la chain NOFORWARD: Registramos la entrada y la tiramos"
iptables -A NOFORWARD -m limit --limit 10/m -j LOG --log-level warning --log-prefix " NOFORWARD Rule
Activated : "
iptables -A NOFORWARD -j DROP
iptables -P FORWARD DROP

# ================================
# Filtrado : OUTPUT ChaiN
# ================================

echo "Activamos filtrado de salida (OUTPUT)..."
iptables -A OUTPUT -d $VULCANO -p udp --dport 53 -j ACCEPT               # DNS para el fw
iptables -A OUTPUT -d $LOCALNET -p icmp -j ACCEPT                      # Ping a la red local
iptables -A OUTPUT -d $ZOID -p tcp --sport 22 -j ACCEPT                 # SSH ZOID
iptables -A OUTPUT -d $VULCANO -p tcp --dport 25 -j ACCEPT               # SMTP hacia VULCANO
iptables -A OUTPUT -p tcp --sport 32001 -j ACCEPT                      # NTOP Output
iptables -A OUTPUT -d $VULCANO -p tcp --sport 22 -j ACCEPT               # SSH VULCANO
iptables -A OUTPUT -d $LOCALNET -p udp --sport 161:162 -j ACCEPT       # SNMP
iptables -A OUTPUT -d $LOCALNET -p tcp --sport 161:162 -j ACCEPT       # SNMP Traps
iptables -A OUTPUT -o eth1 -p udp --dport 67 -j ACCEPT                 # DHCP/BootP Outbound
iptables -A OUTPUT -d $VULCANO -p tcp --dport 21 -j ACCEPT               # FTP hacia VULCANO
iptables -A OUTPUT -d $VULCANO -p tcp --dport 20 -j ACCEPT               # FTP Data
iptables -P OUTPUT DROP                                                # del FW no debe salir nada

# ===============
# Stealth Rule :)
# ===============

echo "Rellenando resto de la chain STEALTh: Registramos la entrada y la tiramos"
iptables -A STEALTH -m limit --limit 10/m -j LOG --log-level warning --log-prefix " STEALTH Rule
Activated : "
iptables -A STEALTH -j DROP


# ==================
# SynAttack defense
# ==================

echo "Activando proteccion ante SYNFlood"
iptables -A INPUT -p tcp --tcp-flags SYN SYN -m limit --limit 100/s -j INPUT2
iptables -A INPUT -j INPUT2

# ================================
# Filtrado : INPUT Chain
# ================================

echo "Actvivamos filtrado de entrada (INPUT)..."

iptables -A INPUT2 -i eth1 -p icmp --icmp-type echo-request -j ACCEPT # Ping
iptables -A INPUT2 -i eth2 -p icmp --icmp-type echo-request -j ACCEPT # Ping
iptables -A INPUT2 -i eth1 -p icmp --icmp-type echo-reply -j ACCEPT   # Ping
iptables -A INPUT2 -i eth2 -p icmp --icmp-type echo-reply -j ACCEPT   # Ping
iptables -A INPUT2 -s $LOCALNET -p tcp --dport 32001 -j ACCEPT        # NTOP Input
iptables -A INPUT2 -i eth2 -s $VULCANO -p tcp --dport 22 -j ACCEPT    # Aceptamos SSH desde CALIGULA
```

```
iptables -A INPUT2 -i eth2 -s $VULCANO -p udp --sport 53 -j ACCEPT    # Aceptamos DNS de respuesta
iptables -A INPUT2 -i eth1 -s $ZOID -p tcp --dport 22 -j ACCEPT       # Aceptamos SSH desde ZOID
iptables -A INPUT2 -i eth2 -s $VULCANO -p tcp --sport 25 -j ACCEPT    # Respuesta SMTP desde VULCANO
iptables -A INPUT2 -i eth1 -p udp --dport 67 -j ACCEPT                # DHCP/bootP Inbound en Private
iptables -A INPUT2 -s $LOCALNET -p udp --dport 161:162 -j ACCEPT      # Accept SNMP
iptables -A INPUT2 -s $LOCALNET -p tcp --dport 161:162 -j ACCEPT      # Accept SNMP Traps
iptables -A INPUT2 -s $VULCANO -p tcp --sport 21 -j ACCEPT            # Respuesta de VULCANO de FTP
iptables -A INPUT2 -s $VULCANO -p tcp --sport 20 -j ACCEPT            # FTP Data
iptables -A INPUT2 -s ! $LOCALNET -j STEALTH                          # Regla de DROP
iptables -A INPUT2 -j DROP                                            # El resto a la mIeRdA :)
iptables -P INPUT DROP                                                # Por si las moscas

# ======
# END
# ======

echo "Activando IP forwarding..."
echo 1 > /proc/sys/net/ipv4/ip_forward
echo "Fin configuracion Firewall"
```