IPTABLES

¿ Qué es un firewall ?

Un FW o cortafuegos, es un dispositivo que, como su palabra lo indica, hace las veces de “barrera”, su misión es, de alguna forma bloquear el paso a cierto tipo de información, por lo tanto si seguimos esta lógica debería :

- Poder “escuchar” la totalidad del tráfico que deseemos analizar.
- Estar en capacidad de “desarmar” los encabezados de cada protocolo.
- Tener patrones estandarizados para comprender cada protocolo.

Todo lo anterior mencionado los vemos con “***wireshark***” cuando nos despliega los campos de cada protocolo, o cuando deseamos implantar cualquier tipo de filtro.

Si lo que queremos filtrar tiene su destino hacia un “host” específico o hacia una red, por lo tanto aquí tenemos una primera clasificación.

- FW's de hosts (a veces asociados a Fws personales y/o a servidores). Por
- Fws de red..

Este diferenciación es importante pues el primero de ellos filtra hacia el nivel de “Aplicación” de ese mismo equipo, sin embargo el segundo de ellos debe decidir si volverá a “enrutar o no” esa trama hacia la red destino.

En definitiva un FW de red, será transmitir o no cada trama por una de esas interfaces, y por lo tanto deberá “reconstruirla” al completo en los niveles de red y enlace, pues saldrá hacia una nueva red con una diferente MAC origen.

¿ Cómo funciona un firewall ?

Funciona en base a reglas que va recorriendo secuencialmente trama a trama y una a una, verificando si cumple o no con ellas para luego adoptar una resolución.

Cuando se instala un FW no trae ninguna regla configurada, lo primero que debemos hacer es comenzar a definir sus reglas o el conjunto de ellas que suele denominarse “Política”de ese FW, y para ello existen dos grandes filosofías.

- Política permisiva.
- Política restrictiva.

La **política permisiva** consiste en aceptar todo el tráfico inicialmente, y poco a poco comenzar a "ajustar" las reglas hasta llegar a la situación deseada.

Una **política restrictiva** parte del supuesto inverso "***Negar todo***", y paulatinamente ir abriendo caminos por medio de las reglas que se determine como necesarias, siempre a través de un detallado análisis.

Es evidente que la **política restrictiva** es la más robusta, pues partimos de la base que no entrará ni saldrá tráfico que no haya sido evaluado previamente, en cambio una **política permisiva** nos puede sorprender el paso de algún tipo de información que expresamente no hayamos tenido en cuenta, pero reiteramos una vez más: esta decisión en la mayoría de los casos dependerá de la situaciones reinante, y nos veremos obligados a adaptar una u otra.

Las reglas de un FW

Las reglas, independientemente de la tecnología y/o producto que empleemos, responden a un esquema básico que generalmente es del siguiente tipo:

Interfaz IP_Origen Puerto_Origen "SENTIDO" IP_Destino Puerto_Destino Acción

- La **interfaz**, puede ser exclusivamente una de las que cuente ese dispositivo, más de una o todas.

- Las **Ips** y los **Puertos**, son suficientemente claros. Sólo cabe la salvedad que suele representarse con "**Any**" cuando es cualquiera (o todos) ellos, y también que generalmente permiten el empleo de "**mascaras**" de la forma "/" y también la concatenación del tipo 135-137 (puertos: 135, 136 y 137), la separación por medio de comas o punto y comas: 22, 23, 53, 110 (puertos: 22, 23, 53 y 110 exclusivamente), etc.

- El "**Sentido**" se suele representar como: "**in**" (entrante), "**out**" (saliente) o "**both**" (ambos).

- La "**Acción**" se la relaciona habitualmente a: "**Accept**" (Aceptar), "**Deny**" o "**Drop**" (Negar), "**PASS**" (Pasar) y "**Log**" (guardar en logs).

Las reglas se irán siguiendo secuencialmente hasta llegar a la última en cada una de las tramas que sean capturadas, si al llegar a la última, ninguna de ellas "ha aplicado", entonces no se adoptará ninguna acción. Por esta razón, es que se suele encontrar casi siempre como última regla algo similar a:

Interfaz_(Any) Any Any ↔ Any Any Deny

Con esto estamos negando cualquier trama que haya llegado hasta aquí,y con ello aseguramos que no pase nada más que lo que las reglas permiten, esto se da en la **política permisiva**.

Firewall en Linux

Como todo FW, su funcionamiento está basado en reglas, pero estas se agrupan en cadenas que a su vez forman parte de tablas.

Netfilter, de forma nativa ofrece tres tablas (por defecto es **filter**):

- **filter**: Filtrado.
- **NAT**: Conversión (traslado) de direcciones.
- **Mangle**: manipulación de paquetes.

De forma nativa ofrece tres cadenas :

- **INPUT**: es para filtrar paquetes que vienen hacia este host.
- **OUTPUT**: es para filtrar paquetes que salen de este host.
- **FORWARD**: reencaminar paquetes.

La sintaxis básica de **iptables** es de la forma :

$ iptables -t filter -A INPUT <opciones>
$ iptables -A INPUT <opciones>

La estructura completa de una regla básicamente seria:

iptables → -t → tabla → tipo_operación → cadena → regla_con_parámetros → Acción

Las operaciones básicas sobre las cadenas (existen más) son :

- -***t*** (tabla): indica qué tabla se empleará.
- -***i*** (interfaz de entrada): mismo formato que ***ifconfig***.
- -***o*** (interfaz de salida): mismo formato que ***ifconfig***.
- -***p*** (protocolo): ***tcp***, ***udp***, etc.
- -***s***, -***d***: dirección IP fuente o destino (Se debe aclarar máscara, ej:/24).
- -***sport***, -***dport***: puertos fuente o destino.
- -***A*** (add): agrega una regla al final de la cadena.
- -***I*** (insert): agrega una regla al principio de la cadena.
- -***R*** (replace): reemplaza una regla por otra.
- -***D*** (delete): elimina una regla.
- -***F*** (flush): elimina todas las reglas de la cadena. Es equivalente a borrar las reglas una por una.
- -***L*** (list): muestra las reglas dentro de la cadena.
- -***Z***: pone a cero todos los contadores.

Las acciones básicas:

- ***ACCEPT***: aceptar el paquete/transacción.
- ***DROP:*** rechaza el paquete/transacción.
- ***REJECT:*** rechaza el paquete/transacción. A diferencia de ***DROP***, notifica al emisor que el paquete/transacción fue descartado.

Para listar la reglas que tenemos :

$ iptables -L -n

Chain INPUT (policy ACCEPT)

target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

- Las 3 cadenas (INPUT/FORWARD/OUTPUT) estan vacías
- Las 3 cadenas tienen como politica default "ACCEPT".

Para listar la tabla de nat :

$ iptables -L -n -t nat

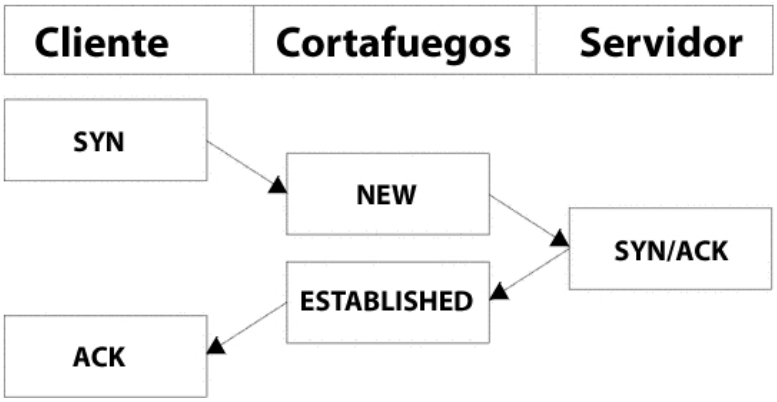
Chain PREROUTING (policy ACCEPT)
target prot opt source destination

Chain POSTROUTING (policy ACCEPT)
target prot opt source destination

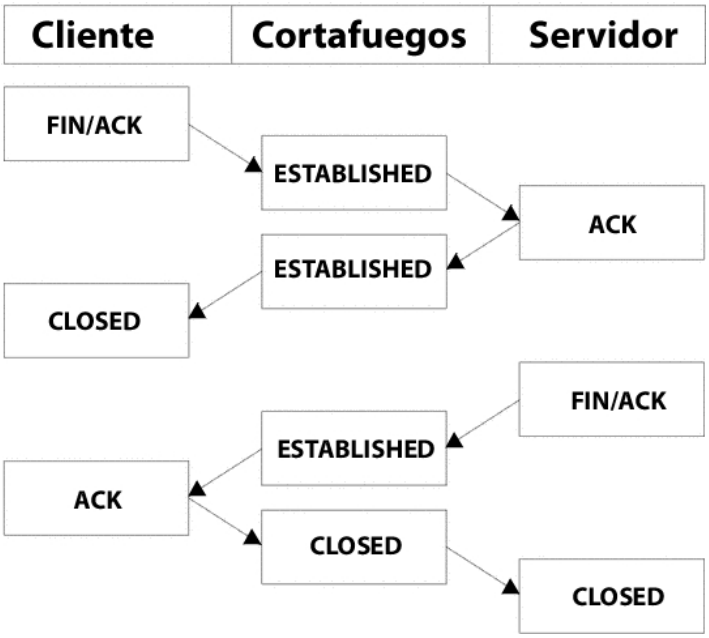
Chain OUTPUT (policy ACCEPT)
target prot opt source destination

- Las 3 cadenas (PREROUTING/POSTROUTING/OUTPUT) estan vacías
- Las 3 cadenas tienen como politica default "ACCEPT".

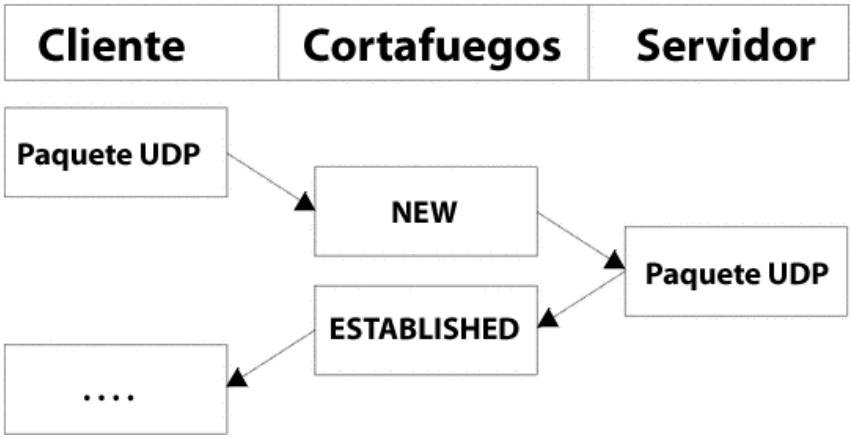
Cuando se realiza una conexión TCP, lo hace de la siguiente manera y tomando los siguientes estados:



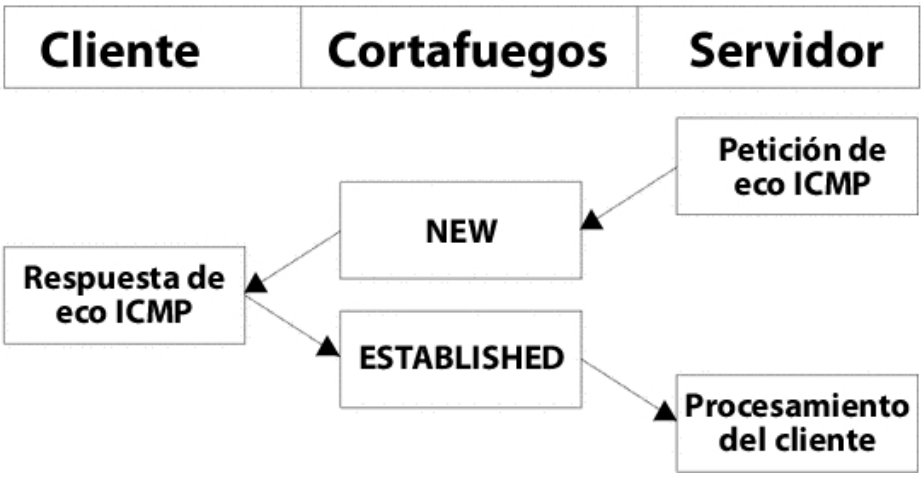
Cuando una conexión TCP se cierra, lo hace de la siguiente manera y tomando los siguientes estados:

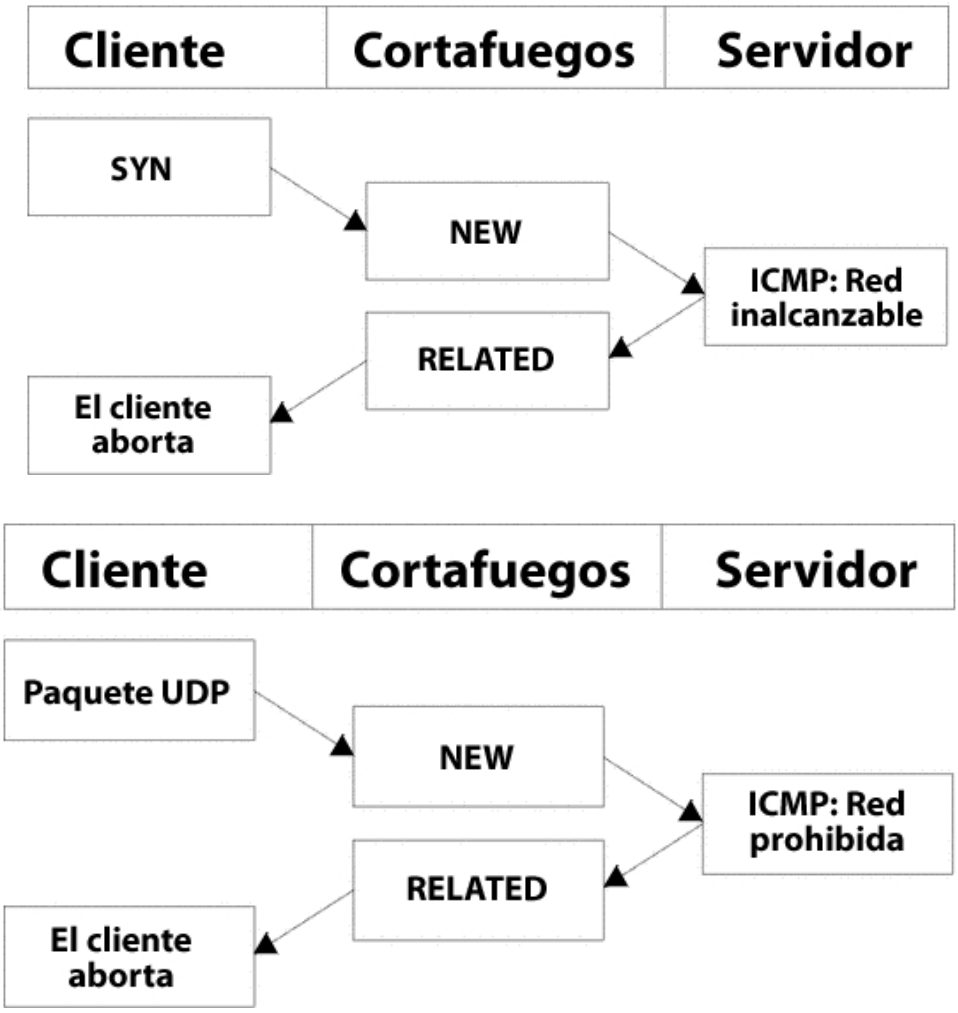


Conexiones UDP :



Conexiones ICMP :





Ejemplos :

Si queremos evitar que nos realicen un ping desde una ip remota realizamos lo siguiente :

$ iptables -A INPUT -p icmp -s 192.168.0.101 -j DROP

Qué es esto? Simple: Agregar (append) la siguiente regla a la cadena **INPUT**: al recibir un paquete del tipo **icmp** (***-p o --protocol***) con origen (***-s o --source***) "192.168.0.101" y con cualquier destino (ya que no lo especificamos), enviamos ese paquete (***-j o --jump***) a DROP, que por cierto es lo mismo que dejarlo tirado =).

$ iptables -L -n

Chain INPUT (policy ACCEPT)
target prot opt source destination
DROP icmp -- 192.168.0.101 0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Para borrar esta regla realizamos lo siguiente :

$ iptables -D INPUT -p icmp -s 192.168.0.101 -j DROP

o bien de forma mas fácil :

$ iptables -L -n --line-numbers

Chain INPUT (policy ACCEPT)
num target prot opt source destination
1 *DROP icmp -- 192.168.0.101 0.0.0.0/0*

Chain FORWARD (policy ACCEPT)
num target prot opt source destination

Chain OUTPUT (policy ACCEPT)
num target prot opt source destination

Como vemos en INPUT nos muestra el numero de linea, entonces sabiendo el número de linea podemos borrarla :

$ iptables -D INPUT 1

Si queremos denegar el servicio de httpd.

$ iptables -A INPUT -p tcp -s 192.168.0.101 --dport 80 -j DROP

Lo mejor es tener una ***política restrictiva*** para poder denegar por defecto, es decir si no coincide con la primer regla ni la próxima por defecto deniega, en este caso nuestras reglas tendrán que permitir es decir :

$ iptables -F
$ iptables -P INPUT -j DROP

El problema aquí es que aunque nosotros podamos realizar conexiones salientes, las respuestas de los servidores serán descartadas. Para solucionar esto, agregamos la siguiente regla:

$ iptables -A INPUT –m state ---state ESTABLISHED,RELATED -j ACCEPT

Por la cual dejamos pasar cualquier paquete cuya conexión ya se ha establecido (**ESTABLISHED**), o cuya conexión es nueva, pero está relacionada a una conexión ya establecida (**RELATED**).

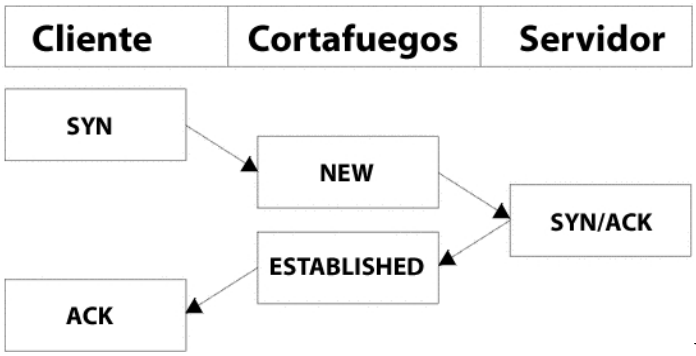
$ iptables -L INPUT -n

Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED

Con lo cual, nadie podrá iniciar una conexión a nuestra maquina, salvo que especifiquemos. Ahora vamos a empezar a permitir conexiones :

$ iptables -A INPUT -i lo -s 127.0.0.1 -j ACCEPT

Cuando se realiza una conexión TCP, lo hace de la siguiente manera y tomando los siguientes estados:



Para que desde la red ***192.168.0.0/24*** puedan entrar por ***ssh*** realizo lo siguiente :

$ iptables -A INPUT -i eth0 -p tcp -s 192.168.0.0/24 --dport 22 --syn -m state --state NEW -j ACCEPT

Con lo cual, nadie podrá iniciar una conexión a nuestra maquina, salvo que especifiquemos. Ahora vamos a empezar a permitir conexiones :

$ iptables -A INPUT -i lo -s 127.0.0.1 -j ACCEPT

Si queremos permitir la entrada de la maquina 192.168.0.101 al puerto 80.

$ iptables -A INPUT -i eth0 -p tcp -s 192.168.0.101 ---dport 80 -j ACCEPT

Utilizando LOG

$ iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$ iptables -A INPUT -i lo -j ACCEPT
$ iptables -A INPUT INPUT -p tcp --dport 22 -j LOG --log-prefix "Ingresando al ssh" --log-level 4
$ iptables -A INPUT -p tcp --dport 22 --syn -m state --state NEW -j ACCEPT
$ iptables -P INPUT DROP

Como vemos primero tenemos que declarar el ***LOG*** y luego va la regla, ***--log-prefix*** nos indica el mensaje que saldrá en los log (***syslog***,***message***,etc), ***--log-level 4*** que activará ***syslog*** solo cuando el mensaje sea de tipo ***warning(4)*** o superior.

Algo que seria muy interesante es modificar el archivo ***/etc/rsyslog.conf***

$ vi /etc/rsyslog.conf
kernel.warning /var/log/iptables.log

$ /etc/init.d/rsyslog restart

Cada regla tiene dos campos, tal como el que añadimos al final. El primer campo (***kern.warning***) es el "***selector***" que se compone de dos partes separadas por el punto: ***facility.priority***, '***facility***' que en este caso es ***kern*** (***kernel***) representa el subsistema que produce el mensaje y '***priority***' define la

severidad del mensaje. Estas pueden ser en orden ascendente: ***debug***, ***info***, ***notice***, ***warning***, ***error***, ***crit***, ***alert***, ***emerg***.

Entonces '***warning***' es el nivel ***4*** de prioridad e indica entonces que todos los mensajes provenientes del ***kernel*** que tengan un nivel de prioridad ***4*** o mayor se bitacorizarán en el archivo y se ignoran los de ***debug***, ***info*** y ***notice*** que son del ***3*** hacía abajo y que generalmente son irrelevantes.

Es importante aclarar que los mensajes del sistema también seguirán guardándose en y otros que se tengan definidos en ya que lo que hicimos fue 'añadir' una línea más y no modificamos nada de lo que ya estaba ahí.

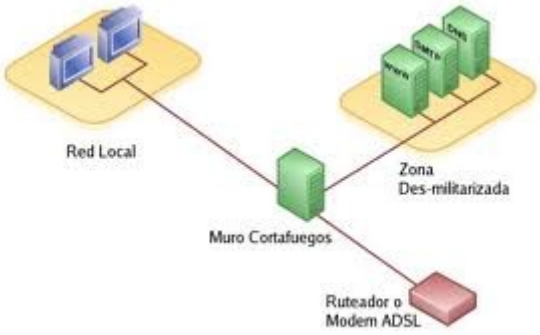
Utilizando PREROUTING

Nos permite redireccionar el puerto 22 a 2020.

$ iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 -j REDIRECT --to-port 2020

¿ Qué es una zona desmilitarizada ?

Una zona desmilitarizada (DMZ), es parte de una red que no está dentro de la red interna (LAN) pero tampoco está directamente conectada hacia internet. Podría resumirse como una red que se localiza entre dos redes. En términos más técnicos se refiere a un área dentro del cortafuegos donde los sistemas que la componen tienen acceso hacia las redes interna y externa, sin embargo no tienen acceso completo hacia la red interna y tampoco acceso completamente abierto hacia la red externa. Los cortafuegos y dispositivos de encaminamiento (***routers***) protegen esta zona con funcionalidades de filtrado de tráfico de red.



¿ Qué es red privada ?

Una **Red Privada** es aquella que utiliza direcciones IP establecidas en el RFC 1918. Es decir, direcciones IP reservadas para **Redes Privadas** dentro de los rangos 10.0.0.0/8 (desde 10.0.0.0 hasta 10.255.255.255), 172.16.0.0/12 (desde 172.16.0.0 hasta 172.31.255.255) y 192.168.0.0/16 (desde 192.168.0.0 hasta 192.168.255.255).

¿ Qué es NAT ?

NAT (acrónimo de ***Network Address Translation*** o ***Traducción de dirección de red***), también conocido como ***enmascaramiento de IP***, es una técnica mediante la cual las direcciones de origen y/o destino de paquetes IP son reescritas mientras pasan a través de un dispositivo de encaminamiento (***router***) o muro cortafuegos. Se utiliza para permitir a múltiples anfitriones en una **Red Privada** con direcciones IP para **Red Privada** para acceder hacia una Internet utilizando una sola dirección IP pública.

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

¿ Qué es IP masquerading ?

Este tipo de ***NAT*** normalmente es sinónimo de ***SNAT***, pero ***iptables*** distingue dos casos:

- ***SNAT***: Cuando la dirección IP pública que sustituye a la IP origen es estática (***SNAT*** también significa ***Static NAT***).

- ***MASQUERADE***: Cuando la dirección IP pública que sustituye a la IP origen es dinámica, caso bastante habitual en conexiones a Internet domésticas.

¿ Qué es SNAT ?

Este tipo de ***NAT*** es en el que ***se cambia la dirección IP de origen***, es la situación más utilizada cuando estamos utilizando una dirección IP privada (RFC 1918) en una red local y establecemos una conexión con un equipo de Internet. Un equipo de la red (normalmente la puerta de enlace) se encarga de cambiar la dirección IP privada origen por la dirección IP pública, para que el equipo de Internet pueda contestar. Los pasos que se seguirían serían algo como:

- Un equipo de una red local con una dirección IP privada (supongamos 192.168.3.14) quiere solicitar una página web (puerto 80/tcp) del equipo de Internet ***www.wordpress.com***

- Realiza una consulta DNS y obtiene que el equipo que aloja dicha página tiene la dirección IP 76.74.254.126

- Consulta su tabla de encaminamiento y como no está en la misma red que el servidor web de wordpress, envía el paquete con la solicitud de la página al equipo que es su destino por defecto (puerta de enlace o ***gateway***), que supongamos tiene la dirección ***192.168.3.254***.

- El ***gateway***, que en este caso debe actuar como dispositivo de ***NAT***, recibe el paquete y comprueba la dirección IP destino, como no es la suya, lo envía a su propio destino por defecto (***gateway***) que ya será una dirección IP pública.

- Antes de que el paquete salga por la interfaz de red externa, se le cambia la dirección IP origen (***192.168.3.14***) por la dirección IP pública (supongamos que fuese ***80.58.1.14***) y se guarda la petición en lo que se denomina tablas de ***NAT*** (anotando también el puerto origen, supongamos que fuese el ***5015/tcp***).

- El paquete viaja por Internet saltando de router a router hasta que llega a su destino.

- El equipo ***76.74.254.126*** recibe una petición desde la dirección ***80.58.1.14*** y la contesta, por lo que el paquete de vuelta llevará ahora dirección IP origen ***76.74.254.126***, dirección IP destino ***80.58.1.14***, puerto origen ***80/tcp*** y puerto destino ***5015/tcp***.

- La contestación del servidor web de ***wordpress.com*** llega a la interfaz externa del dispositivo de ***NAT***, que consulta las tablas de ***NAT*** y comprueba (gracias al puerto origen) que corresponde con una petición realizada desde el equipo ***192.168.3.14***, por lo que modifica la dirección IP destino por ésta y se lo envía directamente.

 # iptables -t nat -A PREROUTING -i eth0 -d 80.58.1.14 -j SNAT –to-

source=192.168.3.254

 # iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.254 -j SNAT –to-source=80.58.1.14

¿ Qué es DNAT ?

DNAT, (acrónimo de ***Destination Network Address Translation*** o t***raducción de dirección de red de destino***) es una técnica mediante la cual se hace público un servicio desde una **Red Privada**. Es decir permite redirigir puertos hacia direcciones IP de **Red Privada**. El uso de esta técnica puede permitir a un usuario en Internet alcanzar un puerto en una **Red Privada** (dentro de una **LAN**) desde el exterior a través de un encaminados (***router***) o muro cortafuegos donde ha sido habilitado un **NAT**.

 # iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 2050 -j DNAT --to 192.168.1.10:22

Masquerade

En este ejemplo permitimos que toda nuestras consulta hacia el exterior pase de red (eth0) salga directamente por el ppp0 (módem).

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