Iptables vs Nftables.

José María Caballero Alba

February 8, 2015

Contents

1 Prologo

Es interesante comparar el uso de iptables de manera practica y compararlo con lo que seria la traducción a nftables, tenemos que tener en cuenta que algunas opciones de iptables no están disponibles en nftables por estar aún en desarrollo y viceversa.

2 Diferencias principales

- La primera y mas distinguible es la sintaxis, en iptables, las flags van precedidas de dos guiones o uno (-p tcp) en nftables usa una sintaxis mas limpia, inspirada en tcpdump.

- Las tablas y las cadenas son totalmente configurables en nftables, contrariamente a iptables, que solo ofrece un set definido de tablas y cadenas, nftables permite crear cadenas y tablas propias con sus correspondientes configuraciones en función de lo que necesites.

- No hay distinción entre objetivos y targets, en nftables tenemos expresiones que son instrucciones para construir nuestras reglas. Esto es muy diferente en iptables que requiere sintaxis de objetivos y targets para usar protocolos.

- Se pueden especificar varias acciones en una sola linea en nftables, con iptables solo podemos especificar una.

- Los contadores en iptables son establecidos de manera fija para cada tabla y reglas, en nftables se puede establecer de manera opciones estos contadores.

- Infraestructura de manera genérica en lo sets en nftables, esto permite configuraciones avanzadas como diccionarios, mapas.

- Actualizar el kernel es una tarea que consume mucho tiempo, especialmente si quieres mantener mas de un firewall en tu red. Los distribuidores normalmente usan versiones antiguas del kernel de linux por razones de estabilidad. Con la nueva maquina de estados de nftables no es necesario actualizar el kernel para un nuevo protocolo, simplemente es necesario actualizar la utilidad nft.

- No es necesario el uso de guiones (-) o dobles guiones (–) para el uso de las banderas.[1]

[1] Esta información ha sido sacada de la wiki de nftables `http://wiki.nftables.org/wiki-nftables/index.php/Main_differences_with_iptables`

3 Configurando tablas

3.1 Creando tablas Iptables

Iptables cuenta con 5 tablas (raw, filter, nat, mangle, security) que son zonas en las que una cadena de puede aplicar.

Estas tablas son por defecto, que ya vienen con una serie de cadenas por defecto (input, output, postrouting, prerouting etc). Estas vienen sin reglas.

Para configurar las tablas tenemos el siguiente formato:

- iptables -t [tabla] [-AIRELNPDZ] [CADENA] [PARAMETROS] [-j ACCION]

En iptables para poder configurar un tabla, tenemos que usar por obligación una de las cadenas, ya que no nos deja atacar una tabla solamente.

Para hacer un ejemplo sencillo:

- iptables -t filter -P INPUT -j drop

Esto estará aplicando la política por defecto de rechazar los paquetes de la cadena input.

Debemos de tener en cuenta, que para configurar una tabla en iptables no obliga configurar también al menos una de las cadenas de esa tabla.

3.2 Creando tablas Nftables

En nftables seguimos contando con las mismas tablas que en iptables, que siguen siendo las zonas en las que podemos crear cadenas y después crear reglas en ellas. La diferencia principal y mas notable respecto a iptables que podemos crear nuestras tablas y además no nos obliga a configurar una cadena a la hora de crearla. Para configurar una tabla en nftables tenemos el siguiente formato:

- nft create table [ip] nombre_tabla

Como vemos tenemos un formato mas sencillo que además podemos crear la tabla solamente sin configurar nada y decidir si pertenece (o no) a la familia ip.

Un ejemplo muy sencillo de crear una tabla:

- nft create table filter

Esto creará la tabla de tipo ip (por defecto, si no lo hemos especificado) filter, la cual no contiene cadenas ni reglas. Solo contiene la definición de la tabla en si misma.

3.3 Listando tablas en Iptables

En iptables podemos mostrar listar las tablas, para especificar una en concreto podemos usar el parametro -t tabla. Para mostrar la tabla (junto con sus cadenas y sus reglas, no solo los nombres de las tablas) basta con:

- iptables -L

Esto no mostrara el contenido de la tabla, con sus cadenas y sus reglas, no nos esta "listando" las tablas. Al no usar el parámetro -t, se muestra por defecto la tabla filter.

Para listar una tabla en concreto, usaremos:

- iptables -t nombre_tabla -L

3.4 Listando tablas en Nftables

Al contrario que en iptables, nftables si que nos deja listar las tablas y no nos obliga a ver su contenido. Para listar las tablas simplemente podemos hacer esto:

- nft list tables

Debemos fijarnos que tables, termina en s, es decir plural, por lo tanto nos esta listando los nombres de las tablas.

Por el contrario, si queremos listar solo una tabla y ya si que nos enseña el contendido de esta tabla con sus cadenas y reglas, bastaría con:

- nft list table nombre_tabla

Esto no mostraría exactamente el mismo contendido que en iptables.

Ej de capturas de pantalla:

iptables -t filter -L:

```
root@NEPTUNO:/home/caballeroalba# iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

nft list tables:

```
root@NEPTUNO:/home/caballeroalba# nft list tables
table filter
table filter2
```

Como vemos, en nftables tenemos la opción de listar solamente las tablas y de listar al modo de iptables para una tabla en concreto.

3.5 Borrando tablas en iptables

En iptables no podemos borrar una tabla, podemos vaciar sus cadenas, o eliminar las reglas, pero las tablas siguen existiendo.

3.6 Borrando tablas nftables

En nftables si que se nos permite borrar una tabla, con esto borraríamos la tabla, sus cadenas y sus reglas. El formato a usar seria el siguiente:

- nft delete table nombre_tabla

Y no es necesario nada mas, con esto podemos borrar una tabla en nftables, cosa que con iptables no podíamos.

3.7 Vaciado de tablas en iptables

En iptables para vaciar (flush) las tablas (no podemos borrar las cadenas, solo podemos vaciarlas) de manera general podríamos hacer:

- iptables -t filter –flush

Esto nos vaciaría todas las todas las cadenas de la tabla filter. Si omitimos el parámetro -t, nos vaciaría todas las cadenas de la tabla filter.

3.8 Vaciado de tablas en nftables.

En nftables si que podemos vaciar tablas de la misma manera, es decir, podemos eliminar todas la reglas de las cadenas que contienen la tabla:

- nft flush table nombre_tabla

Es nos eliminaría las reglas de la tabla especificada.

4 Configurando cadenas

4.1 Añadiendo cadenas en Iptables

En iptables ya tenemos por defecto cadenas asociadas a nuestras tablas, aunque se pueden añadir cadenas personalizadas, por ej, por defecto la tabla filter tiene las cadenas de serie input output y forward, así que si queremos añadir una cadena nueva a la tabla filter podemos hacerlo con la opcion -N de esta manera:

- iptables [-t tabla] -N nombre_cadena

Para la tabla filter:

- iptables -t filter -N nueva_cadena

O también:

- iptables -N nueva_cadena

Recordemos que en iptables la tabla por defecto es filter

```
root@NEPTUNO:/home/caballeroalba# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain nueva_cadena (0 references)
target     prot opt source               destination
```

4.2 Añadiendo tablas en nftables

En nftables no tenemos cadenas por defecto dentro de nuestras tablas, pero tenemos la posibilidad de añadir las cadena de tipo base (input, output, forward,

postrouting, prerouting) o crear cadenas personalizadas, con el siguiente formato seria para las tipo base:

- nft add chain ip nombre_tabla nombre_cadena {type filter hook [cadena_base] priority 0 \; }

Y un ejemplo:

- nft add chain ip filter postrouting { type filter hook postrouting priority 0 \; }

Esto no crearía la cadena de tipo base postrouting dentro de la tabla filter, si queremos tener cadenas personalizadas, simplemente hacemos:

- nft add chain nombre_tabla nombre_cadena

Ejemplo:

- nft add chain filter input

Esto no crea una cadena personalizada dentro de la tabla filter. Lo podemos apreciar en la siguiente captura:

```
table ip filter {
        chain postroutin {
                type filter hook postrouting priority 0;
        }

        chain input {
        }
```

4.3 Borrando cadenas en iptables

En iptables no podemos borrar las cadenas por defecto pero si las personalizadas, tenemos el siguiente formato:

- iptables [-t tabla] -X nombre_cadena

Si seguimos con el ej anterior:

- iptables -X nueva_cadena

Esto nos eliminara la cadena nueva_cadena de la tabla filter (por defecto) y las reglas contenidas en ella.

4.4 Borrando cadenas en nftables

En nftables por el contrario podemos borrar tanto cadenas base como no base, solo necesitamos el nombre de la cadena, no nos importa si es de un tipo o otro, tenemos el siguiente formato:

- nft delete chain nombre_tabla nombre_cadena

Y un ejemplo:

- nft delete chain filter output

El único requisito que nos impone nftables para poder borrar una cadena es que no este vacía, si no, nos dará el siguiente error:

```
root@NEPTUNO:/home/caballeroalba# nft delete chain filter input
<cmdline>:1:1-25: Error: Could not process rule: Device or resource busy
delete chain filter input
^^^^^^^^^^^^^^^^^^^^^^^^^
```

4.5 Vaciando cadenas en iptables

En iptables podemos vaciar cadenas por defecto o personalizadas con siguiente formato:

- iptables [-t nombre_tabla] –flush [nombre_cadena]

Y un ejemplo:

- iptables -t filter –flush INPUT

4.6 Vaciando cadenas en nftables

Para vaciar una cadena base o personalizada (nos da igual) basta con:

- nft flush chain nombre_tabla nombre_cadena

Y un ejemplo:

- nft flush chain filter input

Esto no vaciará las reglas que están en la cadena input.

5 Configurando reglas

5.1 Añadiendo reglas en iptables

En iptables podemos añadir reglas usando el siguiente formato:

- iptables [-t tabla] -A nombre_cadena [-i -o –sport –dport -p -j] [drop reject accept masquerade dnat snat]

Los parámetros están simplificados solo algunos de los mas usuales, se pueden ver todo usando man iptables

Y un ejemplo:

- iptables -A INPUT -p tcp –dport 22 -j DROP

Estamos haciendo en esta ultima regla, que en la tabla filter (por defecto) en la cadena INPUT, los paquetes que usen tcp por el puerto 22 (ssh) sean rechazados.

5.2 Añadiendo reglas en nftables

En nftables podemos ver a la hora de añadir reglas que es mucho mas comprensible para el usuario. El formato para añadir una regla es:

- nft add rule nombre_tabla nombre_cadena [tcp upd ip] [daddr saddr dport sport] [drop accept reject counter]

Los parámetros están simplificados, solo están algunos de los mas usuales, se ver todas las opciones posibles en man nft

Si quisiéramos usar el ejemplo anterior de iptables, bastaría con esto:

- nft add rule filter input tcp dport 22 drop

Podemos apreciar que nftables ofrece una interfaz mas clara en la creación de reglas respecto de iptables.

5.3 Eliminando reglas en iptables

Para eliminar reglas en iptables podemos hacerlo listando las reglas y viendo su numero, usando el parametro –line-number

- iptables [-t nombre_tabla] -L –line-number

Teniendo como resultado esto:

```
Chain INPUT (policy ACCEPT)
num  target     prot opt source
1    DROP       tcp  --  anywhere
```

Una vez identificada la regla a eliminar, solamente necesitamos su numero para eliminarla usando el parametro -D y la cadena donde esta la regla, en el caso anterior:

- iptables -t filter -D INPUT 1

Para eliminar todas las reglas de una cadena (flush) podemos hacer uso del parámetro –flush indicando la cadena

- iptables -t filter –flush INPUT

5.4 Eliminando reglas en nftables

Para eliminar una regla en nftables lo podemos hacer de manera similar a como lo hacemos en iptables, listando las reglas de una tabla especifica (en nuestro caso filter) con el parámetro -a para que nos muestre el numero de las reglas dentro de la tabla.

- nft list [table | tables] [nombre_tabla -a]

Y un ejemplo:

- nft list table filter -a

Dando como resultado esto:

```
chain input {
        type filter hook input priority 0;
        drop # handle 2
}
```

De esta manera, identificamos la regla que queremos eliminar mediante su handle, para ejecutar a continuación la orden de eliminación con el siguiente formato:

- nft delete rule nombre_tabla nombre_cadena handle numero_de_regla

Para nuestro caso:

- nft delete rule filter input handle 2.

Para borrar todas las reglas de una cadena (flush)

- nft flush chain filter input

6 Reemplazo atómico de reglas

6.1 Reemplazo atómico de reglas en iptables

En iptables podemos hacer un cambio atómico de la reglas haciendo uso de un fichero de texto donde tengamos almacenadas las reglas y usando las utilidades iptables-save y iptables-restore.

Para guardar las reglas actuales:

- iptables-save [-c] [-t tabla] > archivo.txt

La opción c nos permite guardar también los contadores, y -t la tabla (por defecto filter)

Una vez tenemos nuestro fichero archivo.txt podemos restaurarlo en un futuro usando iptables-restore:

- iptables-restore [-c] [-n] < archivo.txt

La opción -c nos permite restablecer los contadores de bytes actuales a los del fichero y la opción -n nos permite no reescribir las reglas existentes en las tablas.

6.1.1 Reemplazo atómico de reglas en nftables

En nftables también es posible hacer un reemplazo atómico de reglas, para ello podemos guardar las reglas actuales al estilo de iptables guardándolas en un fichero de texto:

- nft list table nombre_tabla > reglas.txt

Esto nos guardará las reglas presentes en la tabla indicando al fichero reglas.txt. Cuando queramos restaurarlas simplemente hacemos:

- nft -f reglas.txt

6.1.2 Uso de scripts en el cambio de reglas

Dado que iptables y nftables usan un cambio atómico de reglas, no se recomienda el uso de scripts que tengan una serie de ordenes para hacer vaciado (flush) de reglas actuales, y acto seguido ingresen las nuevas reglas dentro de iptables/nftables. De esta manera no se está ejecutando un cambio atómico y por lo tanto cabe la posibilidad de que en este cambio no atómico se puedan filtrar o filtrar paquetes que no corresponden a nuestras reglas dentro del firewall.

7 Reporte de errores desde la linea de comandos

7.1 Reporte de errores desde la linea de comandos en iptables

Cuando tenemos algún tipo de error en iptables ya sea una tabla/cadena no encontrada o un error en el argumento introducido, iptables nos reporta el error correspondiente, por ej:

- iptables -A CADENA_ERRONEA DROP

Obtendremos la respuesta siguiente:

```
root@NEPTUNO:/home/caballeroalba# iptables -A CADENA_ERRONEA DROP
Bad argument `DROP'
Try `iptables -h' or 'iptables --help' for more information.
```

Como vemos, reporta el error al argumento DROP que no a la cadena, si quitamos el drop de la orden:

```
root@NEPTUNO:/home/caballeroalba# iptables -A CADENA_ERRONEA
iptables: No chain/target/match by that name.
```

Ahora si que reconoce la cadena errónea, por tanto puede dar lugar a errores de compresión por parte del usuario, si además incluimos varios fallos como:

- iptables -t tabla_falsa -A cadena_falsa drop

Obtendremos:

```
Bad argument `drop'
```

Como podemos observar, se han introducido 2 fallos pero muestra el error en el drop, cuando los errores se encuentran en los nombres de la tabla y cadena.

Esto puede ser confuso, veremos en el siguiente apartado que con nftables esta solventado este comportamiento.

7.2 Reporte de errores desde la linea de comandos en nftables

En nftables también tenemos el soporte para los errores introducidos desde la linea de comandos, si tomamos los mismo ejemplos del apartado anterior tendríamos:

- nft add rule filter cadena_falsa drop

Obtenemos lo siguiente:

```
add rule filter cadena_falsa drop
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Como vemos, nos da un error general, no un error en el drop como en iptables, si seguimos con el ejemplo anterior, si quitamos el argumento drop:

```
<cmdline>:1:29-29: Error: syntax error, unexpected end of file
add rule filter cadena_falsa
                            ^
```

Nos señala que tenemos un error en la sintaxis y nos señala donde esta el error, esto nos da una idea general del error mejor que en iptables.

Seguimos con los ejemplos anteriores, si usamos una tabla falsa:

```
add rule tabla_falsa cadena drop
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Nos da un error general. COMPLETAR CON MAS INFORMACIÓN PREGUNTAR

8 Posibles acciones sobre los paquetes

8.1 Posibles acciones sobres los paquetes en iptables

8.1.1 Aceptando y rechazando paquetes

En iptables, para poder aceptar paquetes, debemos de tener en cuenta el siguiente formato:

- iptables -A [nombre_cadena] [-t tabla] [-i interfaz] [-p] [tcp|udp] [–dport | –sport] [-puerto] -j ACCEPT

Y un ejemplo, de que queremos aceptar paquetes de por ej el puerto ftp:

- iptables -A INPUT -t filter -i wlan0 -p tcp –dport 21 -j ACCEPT

Teniendo como resultado:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ftp
```

Para rechazar paquetes, tenemos el mismo formato, solo cambiando el ultimo parámetro:

- iptables -A [nombre_cadena] [-t tabla] [-i interfaz] [-p] [tcp|udp] [–dport | –sport] [-puerto] -j DROP

8.1.2 Saltando a cadenas

En iptables podemos hacer el salto de paquetes a cadenas propias , por ejemplo, podemos crear previamente la cadena propia:

- iptables -N cadena_propia

Una vez creada, podemos añadir reglas a esta cadena para después saltar ella desde las cadenas base de la siguiente manera:

- iptables -A INPUT -p tcp -j cadena_propia

Esto hará que los paquetes que vengan a nuestro sistema (cadena INPUT) por el protocolo tcp serán direccionados hacia nuestra cadena propia.

8.1.3 Log de paquetes

En iptables podemos hacer un log de los paquetes que pasan por nuestro firewall, basta con usar el parametro LOG, por ejemplo, si queremos que nuestro sistema registre las conexiones ssh a nuestro sistema, debemos usar el siguiente formato:

- iptables -A [cadena] [parametros de filtrado] -j LOG [–log-prefix ' '] [–log-level NUM]

Por ejemplo:

- iptables -A INPUT -p tcp -m tcp –dport 22 -j LOG –log-prefix 'acceso a ssh desde fuera ' –log-level 4

Donde:

- -p indica el protocolo
- -m indica que se marquen los paquetes con la marca tcp
- -dport es el puerto destino
- -j la acción a realizar, en este caso el log del paquete
- –log-prefix indica que se usará el prefijo entrecomillado
- –log-level 4 activará syslog solo cuando el mensaje sea de tipo waring (4)

De esta manera, si hacemos un intento de conexión a nuestro host por ssh y vemos el archivo log (/var/log/syslog) veremos lo siguiente:

```
Jan  7 10:45:00 NEPTUNO kernel: [ 2895.041813] acceso a ssh desde fue
T= MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1 DST=12
=60 TOS=0x00 PREC=0x00 TTL=64 ID=55176 DF PROTO=TCP SPT=59289 DPT=22
0 RES=0x00 SYN URGP=0
```

De esta manera podemos hacer un log preciso y monitorizar los accesos a un servidor por ej, para servicios críticos hubiera en este, como podrían ser ssh, mysql, rsync, ftp, etc.

8.1.4 NAT

Para usar nat en iptables, necesitamos usar la tabla nat (por defecto usamos filter) y antes que nada, ya que vamos a usar nat y por tanto un comportamiento de router, debemos activar el bit de ip_forward:

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

1. SOURCE NAT

Al usar nat en POSTROUTING estamos cambiando las direcciones origen de los paquetes y estos, en el caso de que haya una comunicación con algún host externo a nuestra red , los paquetes de respuesta deben de llegar de nuevo al host original, así que debemos de volver a traducir la dirección de destino hacia ese host, esto es automático en iptables.

Para hacer SOURCE NAT (dinamico) con la red que normalmente usamos, por ejemplo 192.168.1.0/24

- iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o wlan0 -j MASQUERADE

Donde:

- -A POSTROUTING añade la regla a la cadena POSTROUTING
- -s indica la red en notación barra de la red que queremos hacer nat
- -o eth0 la interfaz de salida de los paquetes
- -j MASQUERADE la acción en este caso, indica que los paquetes de salida, tendrán como ip de origen la de nuestro host (el que hace nat) si ha sido

asignada de manera dinámica (DHCP) y nos evita tener que poner nuestra ip dinamica, igualmente se puede hacer un snat estático con la ip que queramos (que debería ser la nuestra) usando -j SNAT –to :

- iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o wlan0 -j SNAT –to 8.8.8.8

1. DESTINATION NAT

DESTINATION NAT se usa para modificar la dirección de destino de un paquete que llega a nuestro sistema, por ejemplo, si estamos usando nat, los dispositivos de nuestra red no son visibles, solo el host que usa nat, es el visible desde internet, por tanto, si hay un servidor web en nuestra, debemos de redirigir los paquetes que vengan por el puerto 80 a este host, esto sería de la siguiente manera:

- iptables -t nat -A PREROUTING -p tcp –dport 80 -i eth0 -j DNAT –to 192.168.1.5

Donde:

- -A PREROUTING indica que se añadirá a la cadena PREROUTING
- -p tcp el protocolo usado
- –dport 80 el puerto de destino (destinantion port)
- -i eth0 la interfaz de entrada el paquete
- -j DNAT –to la acción sobre el paquete, en este caso, cambiar el destino a 192.168.1.5

De esta manera estamos alterando el paquete modificando su destino, por estamos implementando dnat sobre nuestro firewall a los paquetes que vengan por el puerto 80 desde fuera de nuestra red.

8.1.5 Marcas sobre los paquetes

Una funcionalidad de iptables es la posibilidad de marcar los paquetes con un numero, se implementa usando el parámetro –set-mark, por ej podríamos marcar los paquetes que salgan por el puerto 80 para ver que paginas visitan los usuarios de nuestra red:

- iptables -A PREROUTING -i eth0 -t mangle -p tcp –dport 80 -j MARK –set-mark 1

Donde:

- -t mangle se refiere a la tabla mangle
- -j MARK –ser-mark es la acción en este caso, marcar el paquete con un 1

Información obtenida de http://tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.netfilter.html

8.1.6 Queueing de paquetes

Queue es una acción especial de iptables que pasa al paquete en una cola destinada al procesamiento en espacio de usuario. Para que esto sea útil, es necesario otros 2 componentes adicionales:

- Un controlador de cola, que lleva la mecánica real de paso de paquetes entre el núcleo y el espacio de usuario.
- Un proceso en espacio de usuario que recibe, posiblemente manipula, y dicta veredicto sobre los paquetes.

Un ejemplo de como usar iptables para poner en cola los paquetes para su proceso en espacio de usuario:

- modprobe iptable_filter

- modprobe ip_queue (anticuado, ahora se debe usar nfnetlink_queue)
- iptables -A OUTPUT -p icmp -j QUEUE

Con esta regla, los paquetes ICMP generados de forma local (como los que crea, digamos, ping), se pasan al módulo ip_queue, que entonces intenta pasarlos a una aplicación en espacio de usuario. Si no hay una aplicación esperando por ellos, entonces se descartan.

Se puede comprobar el estado de ip_queue mediante:

- /proc/net/ip_queue (anticuado, sustituir ip_queue por nfnetlink_queue)

  Nota: la información sobre el queue en iptables, se ha sacado en `http://www.netfilter.org/documentation/HOWTO/es/packet-filtering-HOWTO-7.html`

8.2 Posibles acciones sobre los paquetes en nftables

8.2.1 Aceptando y rechazando paquetes

Suponiendo que ya tengamos (como lo visto en los ejemplos anteriores) una cadena y una tabla, para aceptar los paquetes de una cadena especifica podemos hacer:

- nft add rule nombre_tabla nombre_cadena [operadores] accept

Y un ejemplo especifico, por ejemplo aceptar el uso de ftp (hacía nuestro host):

- nft add rule filter input tcp dport 21 accept

Y lo tendríamos de esta manera:

```
chain input {
        type filter hook input priority 0;
        tcp dport ftp accept
}
```

Para rechazar paquetes haríamos exactamente lo mismo pero cambiando accet por drop:

- nft add rule nomber_tabla nombre_cadena [operadores] drop

8.2.2 Saltando a cadenas

De manera similar a iptables, también podemos tener cadenas personalizadas, una vez tengamos creado alguna, podemos redirigir los paquetes que venga por una cadena base (por ej input) a esta cadena para tratarlos ahí. El formato a seguir es el siguiente:

- nft add rule nombre_tabla nombre_cadena [operadores] jump cadena_personalizada

y un ejemplo:

- nft add rule filter input tcp dport 9999 jump prueba_cadena_personal

Nos quedaría lo siguiente:

```
chain input {
        type filter hook input priority 0;
        tcp dport ftp accept
        tcp dport 9999 jump prueba
}

chain prueba {
        counter packets 0 bytes 0
}
```

8.2.3 Log de paquetes

Para poder hacer un log de paquetes completo en nftables, necesitamos un mayor o igual al 3.17, si se usa un kernel menor, se puede usar el siguiente comando para permitir el log:

- modprobe ipt_LOG

Para usar el log de paquetes hacemos:

- nft add rule nombre_tabla nombre_cadena [operadores] log

Y el mismo ejemplo que usamos en iptables para el loggin de conexiones ssh

- nft add rule filter input tcp dport 22 ct state new log prefix \"Nueva conexion ssh \" accept

Donde:

- ct state para indicar el estado en el que debe de estar la conexion (new, established, related, etc)
- new el estado propiamente
- log prefix Le estamos indicando que en el log se pondra el prefijo que Nueva conexion ssh

Si miramos el log en /var/log/syslog al hacer una conexión ssh a nuestro propio sistema tendremos el siguiente resultado:

```
nueva conexion ssh IN=lo OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:0
```

Esta captura está recortada.

8.2.4 NAT

Como en iptables, nftables también puede hacer nat y de manera mas simple por su sintaxis, de nuevo, distinguimos entre SNAT (source nat) y DNAT (destination nat)

1. SNAT

Para hacer SNAT y querer usar nuestro host como si un router se tratara, debemos de activar el bit de ip_forward como hicimos en el ejemplo de iptables:

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

Una vez hecho esto podemos empezar a establecer SNAT con los siguientes pasos:

- nft add table nat
- nft add chain nat prerouting { type nat hook prerouting priority 0 \; }
- nft add chain nat postrouting { type nat hook postrouting priority 0 \; }

Con esto hemos creado la tabla nat con las cadenas base prerouting y postrouting (creamos las 2 para el ejemplo de snat y dnat, aunque no es necesario si solo se implementa uno de los casos).

Siguiendo el ejemplo que hicimos en iptables, para hacer nat (snat o dnat) de una red seguimos este formato:

- nft add rule nat [postrouing| prerouting] ip [saddr|daddr] [red notación barra|ip] [oif|iif] [interfaz de red] [snat|ndat] [ip]

Y el ejemplo practico de hacer snat

- nft add rule nat postrouting ip saddr 192.168.1.0/24 oif wlan0 snat 8.8.8.8

Donde:

- saddr indica la ip o la red de la que proceden los paquetes
- oif define la interfaz por la que se recogen los paquetes, en este caso wlan0
- snat indica la ip que tendrán los paquetes como origen al salir de la red

De esta manera, si nuestro host es un router que esta conectado a la red indicada los host que tengan configurada como gateway nuestro host, cambiará el la ip de origen a 8.8.8.8 de los paquetes que salgan de nuestra red.

1. DNAT

Para hacer dnat, si tenemos algún servidor web en nuestra red y queremos que sea accesible desde el exterior si se esta usando nat, debemos de seguir el mismo formato que en el anterior caso, y bastaría con lo siguiente:

- nft add rule nat prerouting iif wlan0 tcp dport 80 dnat 192.168.1.3

Donde:

- iif indica la interfaz por donde vendrán los paquetes, en este caso wlan0
- tcp dport 80 el puerto y protocolo por donde vendrán los paquetes.
- dnat hace el traslado de ip destino, cambiaría la ip de nuestro host (el router) por 192.168.1.3 (la del servidor web)

De esta manera, se podrá acceder al servidor web desde el exterior de nuestra red. Cabe decir que nftables nos permite hacer también un set de puertos para un mismo destino, por ej:

- nft add rule nat prerouting iif wlan0 tcp dport {80, 443} dnat 192.168.1.3

De esta manera estamos redirigiendo tantos los paquetes que vengan por el puerto 80 y el 443.

1. NAT FLAGS

Como extra a iptables, nftables implementa a partir del kernel 3.18 el mapeo de puertos aleatorios, además de incluir también que se mantengan estos números de puertos para cada conexión. Los parámetros son:

- random: Puertos origen aleatorios.
- random-fully: Puertos (origen y destino) aleatorios.
- persistent: Mantiene los puertos para el origen y destino para cada conexión.

Para el ejemplo anterior, lo usaríamos de la siguiente manera:

- nft add rule nat postrouting masquerade random,persistent
- nft add rule nat postrouting ip saddr 192.168.1.0/24 oif wlan0 snat 8.8.8.8 random-fully

  Nota: Los ejemplos sobre nat han sido inspirados según el manual de nftables disponible en: `http://wiki.nftables.org/wiki-nftables/index.php/Performing_Network_Address_Translation_%28NAT%29`

8.3 Marcas sobre los paquetes

Como en iptables, nftables permite marcar los paquetes con números usando la opción set mark, para repetir el ejemplo usado en el apartado de iptables en la cual marcamos los paquetes que sean por el puerto 80 para saber que paginas web se visitaban en nuestra red , debemos de seguir el siguiente formato:

- nft add rule filter output tcp dport 80 mark set 9999

8.4 Queueing de paquetes

Como en iptables, podemos hacer queueing de paquetes en nftables para encolarlos hacia el espacio de usuario, esta característica esta disponible desde el kernel 3.14 y es necesario tener la librería libnetfilter_queue (disponible en algunos repositorios oficiales o en el git `https://git.netfilter.org/libnetfilter_queue/`

Para hacerlo seguimos el siguiente formato:

- nft add nombre_tabla nombre_cadena [operadores] queue [num valor o valor_minimo-valor_maximo]

Si no añadimos un numero, el valor por defecto sera 0.

Ejemplo practico:

- nft add filter input counter queue num 45

Con esto ya estamos marcando los paquetes que vayan a nuestro host, si generamos un poco de trafico, podemos verlo en acción:

Debemos de tener en cuenta de que si no hay ningún programa escuchando a queue, los paquetes serán rechazados. COMPLETAR/PREGUNTAR util queue

También podemos habilitar una opción para el que en el caso de que no haya ninguna aplicación escuchando en el espacio de usuario para evitar el encolamiento del paquete y que se pierda. Por tanto, el paquete sera aceptado:

- nft add filter input counter queue num 45 bypass

9 Uso de matchs

Tanto en iptables como en nftables podemos hacer uso de mach para señalar los protocolos, puertos, ips, etc, pero hay algunos que son interesantes a resaltar ya que su uso puede ser de mucha utilidad como el limite de trafico o la metainformación del paquete.

9.1 Metainformacion de paquetes

La metainformación que nos proporcionan los paquetes pueden ser muy útil, ya que nos puede decir quien es el creador del paquete. En iptables, tenemos la siguientes opciones respecto al creador de los paquetes:

- –uid-owner puede capturar los paquetes según la user id.
- –gid-owner puede capturar los paquetes según la id de un grupo.
- –pid-owner puede capturar los paquetes según la id de un proceso.
- –sid-owner puede capturar los paquetes según la id de una session.

Y y un ejemplo:

- iptables -A OUTPUT -o wlan0 -m owner –uid-owner caballeroalba -j ACCEPT

Aquí le decimos a iptables que acepte lo paquetes con la id el usuario caballeroalba.

En nftables, también esta implementada estas características:

- nft add rule filter output oif wlan0 meta skuid caballeroalba accept

Como vemos se utiliza una sintaxis mas sencilla.

9.2 Limite de trafico

Tanto en iptables como en nftables se puede hacer uso de parámetros para el limite de trafico que nos permite por ejemplo, paliar un ataque DDOS.

En iptables un ejemplo de un ping abusivo:

- iptables -A FORWARD -p icmp –icmp-type echo-request -m limit –limit 1/s -j ACCEPT

  Nota: este ejemplo ha sido sacado de `http://www.netfilter.org/documentation/HOWTO/es/packet-filtering-HOWTO-7.html`

Y en nftables:

- nft add rule filter forward icmp type echo-request limit rate 10/second accept

10 Estructuras de datos avanzadas en nftables

Nftables permite el uso de estructuras de datos avanzadas para la mejora de rendimiento en la clasificación de paquetes, esta es una característica no disponible en iptables, esto incluye:

- Sets
- Diccionarios
- Intervalos
- Mapas

10.1 Sets

En nftables viene incluida una infraestructura genérica de conjuntos que permite usar todos selectores soportados para construir sets. Con esta infraestructura es posible representar diccionarios y mapas.

10.1.1 Sets anonimos

Los sets anonimos son aquellos que son:

- Los atados a una reglas, si una regla es eliminada, el set también lo es.
- Los que no tienen un nombre propio, el kernel les asigna uno automáticamente
- No actualizables, no se pueden añadir o quitar elementos cuando son atados a una regla.

La siguiente regla representa un set anonimo:

- nft add rule filter output tcp dport { 80, 443 } counter

Donde {80,443 } representa al set anonimo.

10.1.2 Named sets

Se pueden crean set con un nombre asociado y añadirlos a una tabla ya creada por ej:

- nft add set nombre_tabla nombre_set {type [ipv4_addr | ipv6_addr | ether_addr | inet_proto | inet_service | mark]}

Y un ejemplo:

- nft add set filter ip_list { type ipv4_addr \; }

De esta manera hemos añadido un set de direcciones ipv4 a la tabla filter:

```
table ip filter {
        set ip_list {
                type ipv4_addr
        }
```

Ahora podemos añadir elementos de manera única:

- nft add element filter ip_list {192.168.1.5}

O usando un set anonimo:

- nft add element filter ip_list { 192.168.1.6, 192.168.1.7}

Si comprobamos:

```
table ip filter {
        set ip_list {
                type ipv4_addr
                elements = { 192.168.1.7, 192.168.1.6, 192.168.1.5}
        }
```

Ahora podemos enlazar este set en una regla, por ejemplo añadir un contador a los paquetes que son tienen como destino las ips del set:

- nft add rule filter input ip daddr @ip_list counter

Mirando la tabla, vemos enlazado el set en la cadena input:

```
chain input {
        type filter hook input priority 0;
        ip daddr @ip_list counter packets 0 bytes 0
}
```

Además a los named set, se pueden añadir o quitar elementos, cosa que no podemos hacer en los set anónimos.

Para comprobar la diferencia real con iptables, podemos ver este ej hecho por Eric leblond en este articulo why you love nftables

Tenemos la siguientes acciones en nftables:

- ip6tables -A INPUT -p tcp -m multiport –dports 22,80,443 -j ACCEPT
- ip6tables -A INPUT -p icmpv6 –icmpv6-type neighbor-solicitation -j ACCEPT
- ip6tables -A INPUT -p icmpv6 –icmpv6-type echo-request -j ACCEPT
- ip6tables -A INPUT -p icmpv6 –icmpv6-type router-advertisement -j ACCEPT
- ip6tables -A INPUT -p icmpv6 –icmpv6-type neighbor-advertisement -j ACCEPT

Y se pueden expresar en nftables en solo 2 operaciones:

- nft add rule ip6 filter input tcp dport {telnet, http, https} accept
- nft add rule ip6 filter input icmpv6 type { nd-neighbor-solicit, echo-request, nd-router-advert, nd-neighbor-advert } accept

£Se aprecia la diferencia?

10.1.3 Listando sets

Usando el siguiente formato se pueden listar sets:

- nft list set nombre_tabla nombre_set

10.2 Diccionarios

Los diccionarios o verdict maps es una funcionalidad destacable en nftables. Permiten relacionar una acción a un elemento. Internamente usan un set generico.

10.2.1 Diccionarios anónimos

Suponiendo que tenemos varias cadena personalizadas podemos ver como podemos podemos crear en una linea varias acciones, el formato para hacer un diccionario anónimo es:

- nft add rule nombre_tabla nombre_cadena ip protocol vmap { [protocolo1 jump cadena1],[proto2 jump cadena2] ...}

De esta manera estamos creando en una linea n saltos que serán en las cadenas personalizadas según su protocolo

Un ejemplo sacado la wiki de nftables:

- nft add rule ip filter input ip protocol vmap { tcp : jump tcp-chain, udp : jump udp-chain , icmp : jump icmp-chain }

En este ejemplo se crea el mapa y se asocian los protocolos tcp, udp y icmp a distintas cadenas.

10.2.2 Diccionarios con nombre

Podemos crear diccionarios con nombre siguiendo el formato siguiente:

- nft add map nombre_tabla nombre_diccionario { type [tipo] : verdict\; }

Un ejemplo:

- nft add map filter diccionario { type ipv4_addr : verdict\; }

Y podemos añadir elementos del tipo del que sea el diccionario:

- nft add element filter diccionario { 192.168.1.3 : accept, 192.168.1.5 drop }

Una vez hecho esto, podemos usar el diccionario las acciones asociadas a sus elementos:

- nft add rule filter ouput daddr vmap @diccionario

Realmente en esta regla se estaría provocando lo siguiente:

- nft add rule filter output daddr 192.168.1.3 accept
- nft add rule filter output daddr 192.168.1.5 drop

10.2.3 Intervalos

Nftables permite el uso de expresiones con intervalos, los rangos se leen como valor_minimo-valor_maximo, los intervalos sirven tanto para ips como para puertos, algunos ejemplo:

- nft add rule filter output ip daddr 8.8.8.4-8.8.8.8 drop
- nft add rule filter output tcp ports 50-60 drop

También podemos hacer uso de los sets para establecer los intervalos desde el propio set:

- nft add rule ip filter input ip saddr { 192.168.1.1-192.168.1.200, 192.168.2.1-192.168.2.200 } drop

Y de la misma manera los diccionarios:

- nft add rule ip filter forward ip daddr vmap { 192.168.1.1-192.168.1.200 : jump chain-dmz, 192.168.2.1-192.168.20.250 : jump chain-desktop }

10.2.4 Mapas

Los mapas asocian una key con un valor, y se pueden definir en la misma linea de la regla a crear, un ejemplo de su uso:

- nft add rule ip nat prerouting dnat tcp dport map { 80 : 192.168.1.100, 8888 : 192.168.1.101 }

Podemos ver que en función del puerto hacemos dnat a una ip o otra haciendo uso de un mapa, esto nos evita tener que crear dos reglas una para ip en función del puerto.

Nota: los ejemplos de mapas y intervalos han sido sacados de la wiki de nftables `http://wiki.nftables.org/wiki-nftables/index.php/Intervals`

11 Expresiones en nftables

Nftables proporciona un conjunto de expresiones para hacer de operadores y poder tener reglas mas precisas. Estos son:

- ne no igual, se puede usar !=
- lt menor que, se puede usar <
- gt mayor que, se puede usar >
- le menor o igual, se puede usar <=
- ge mayor o igual, se puede usar >=

Si se usan los simbolos < o > es necesario usar la \< o \> ya que el terminal lo entendería como la salida o entrada estándar.

Un ejemplo simple de poder usar estos operadores:

- nft add rule filter output tcp dport !=80 drop

12 Ejemplo de uso practico con nftables

Para terminar podemos exponer un uso practico de un practico de nftables en una red, el problema es:

1. Tenemos una empresa, en la cual tenemos 3 subredes: 192.168.1.0/24, 192.168.2.0/24 y 192.168.3.0/24
2. Para la red 192.168.1.0/24 tenemos los requisitos siguientes:
   (a) Esta compuesta por 2 servidores, uno web (192.168.1.2)y otro ftp (192.168.12), en los cuales se debe implementar nat
   (b) Los servidores SOLO deben de escuchar/responder peticiones, jamas hacerlas hacia el exterior
3. Para red 192.168.2.0/24 tenemos los siguientes requisitos:
   (a) Esta formada por trabajadores de la empresa, por tanto pueden acceder hacia el exterior
   (b) No se permite el uso de ciertas paginas web (facebook y twitter)
   (c) No pueden recibir peticiones del exterior.
4. Para la red 192.168.3.0/24 tenemos los siguientes requisitos:
   (a) Esta formada por equipos en pruebas, por tanto solo pueden tener acceso entre ellos y nunca pueden recibir peticiones desde el exterior ni tampoco acceder a este.

Suponiendo que tenemos un equipo que hace router, y este esta conectado a las 3 redes y a internet, podemos configurar nftables de la siguiente manera para solventar los requisitos siguientes:

Para la red 192.168.1.0/24:

- nft add rule nat prerouting ip daddr 192.168.1.2 tcp dport 80 dnat 192.168.1.2
- nft add rule nat prerouting ip daddr 192.168.1.3 tcp dport 21 dnat 192.168.1.3

Las dos anteriores se pueden resumir haciendo uso de un mapa en :

- nft add rule ip nat prerouting dnat tcp dport map { 80 : 192.168.1.2, 21 : 192.168.1.3 } (suponiendo estas ips las de los servidores)
- nft add rule filter input ip saddr 192.168.1.0/24 ct state new drop
- nft add rule filter input ip saddr 192.168.1.0/24 ct state established accept

Para la red 192.168.2.0/24:

- nft add rule filter input ip saddr 192.168.2.0/24 ct state new, established accept
- nft add rule filter input ip daddr 192.168.2.0/24 ct state new drop
- nft add rule filter input ip saddr 192.168.2.0/24 ip daddr {8.8.8.8, 8.8.8.4} drop (suponiendo facebook.es -> 8.8.8.8 y twitter.es -> 8.8.8.4)

Para la red 192.168.3.0/24

- nft add rule filter input ip saddr 192.168.3.0/24 drop
- nft add rule filter input ip daddr 192.168.3.0/24 drop