ENRUTAMIENTO CON NAT ESTÁTICO Y DINAMICO EN SISTEMAS *GNU/Linux. Nftables* 40

1.- HABILITAR EL NAT EN SISTEMAS *GNU/LINUX*.

Tal y como ya se mencionó, cuando se vio el enrutamiento convencional en sistemas *GNU/Linux*, la gestión de qué hacer con los paquetes que llegan a nuestra máquina (*input*, la IP de destino pertenece al rúter), que salen de ella (*output*, la IP de origen pertenece al rúter) o que la atraviesan (*forward*, ni la IP de destino ni la IP origen pertenecen al rúter) corresponde al gestor de paquetes de los sistemas *GNU/Linux*. Mediante su herramienta de administración es posible establecer reglas complejas de tratamiento de esos paquetes.

En la actualidad, la mayoría de las distribuciones *GNU/Linux*, entre las que se incluye *Debian*, incorporan en su núcleo (*kernel*) el subsistema de gestión de paquetes *Netfilter*. Que permite realizar el manejo de paquetes en diferentes estados del procesamiento.

El componente más "utilizado" construido sobre *Netfilter* es *nftables*[1], una herramienta de cortafuegos que permite no solamente filtrar paquetes, sino también realizar traducción de direcciones de red (NAT) o mantener registros de *log*, entre otras cosas.

Nftables, evolución de las *iptables*, es, en realidad, el nombre de la herramienta de configuración mediante la cual el administrador puede definir políticas de filtrado del tráfico que circula por la red, razón por la cual suele decirse que corresponde al espacio de usuario (*nft*) del *Netfilter*. Permitiendo establecer reglas acerca de qué hacer con los paquetes de la red. Las reglas se agrupan en cadenas: cada cadena es una lista ordenada de reglas que se comprueban secuencialmente (como una cadena) hasta encontrar una que sea de aplicación, tras la aplicación de esa regla, se abandona la búsqueda en esa cadena. Las cadenas se agrupan en tablas: cada tabla está asociada con un tipo diferente de procesamiento de paquetes.

Así, por ejemplo, existe la tabla *filter*, creada por defecto en el caso de los sistemas *Debian*, que contiene tres cadenas (*chains*), denominadas: *input*, *output* y *forward*. A cada una de estas cadenas se asociarán las reglas que se aplicarán, a cada uno de los paquetes IP, según el tipo de tráfico al que correspondan: de entrada (*input*), de salida (*output*) o de reenvío (*forward*). La estructura de la tabla *filter* puede verse en el fichero */etc/nftables.conf*, tal y como se recoge en la Figura 1.

```
#!/usr/sbin/nft -f

flush ruleset

table inet filter {
    chain input {
        type filter hook input priority 0;
    }
    chain forward {
        type filter hook forward priority 0;
    }
    chain output {
        type filter hook output priority 0;
    }
}
```

Figura 1

Además del control de los casos del tráfico normal, gestionado a través de la tabla *filter*, a nosotros nos interesará realizar traducción de direcciones (NAT), para lo cual debe utilizarse una nueva tabla, que denominaremos tabla *nat*, en la cual crearemos dos cadenas, la cadena *preenrutado*, en la que incorporaremos las reglas que deben verificarse antes de realizar el enrutamiento (por ejemplo en procesos de DNAT, ***D**estination* NAT), y la cadena *posenrutado*, correspondiente a las reglas a comprobar una vez hecho el enrutamiento, como por ejemplo, en nuestro caso, con el NAT. No se preocupe si de momento todo esto le resulta poco significativo, más adelante lo verá con claridad.

Aunque nosotros nos ceñiremos a las tablas y cadenas mencionadas anteriormente, la herramienta *nftables* permite crear cuantas tablas y cadenas se deseen con las distintas finalidades que se requieran.

Las posibilidades de las *nftables* son vastísimas, existiendo infinidad de guías[2] y tutoriales, por ejemplo, en la URL:

https://www.redeszone.net/tutoriales/seguridad/nftables-firewall-linux-configuracion/

En cualquier caso, el estudio en detalle de *nftables* es algo que supera el objetivo de este documento, razón por la cual se verán algunos ejemplos básicos que permitan comprobar que el funcionamiento de un enrutado con NAT estático es igual al obtenido en *Windows Server 2022*.

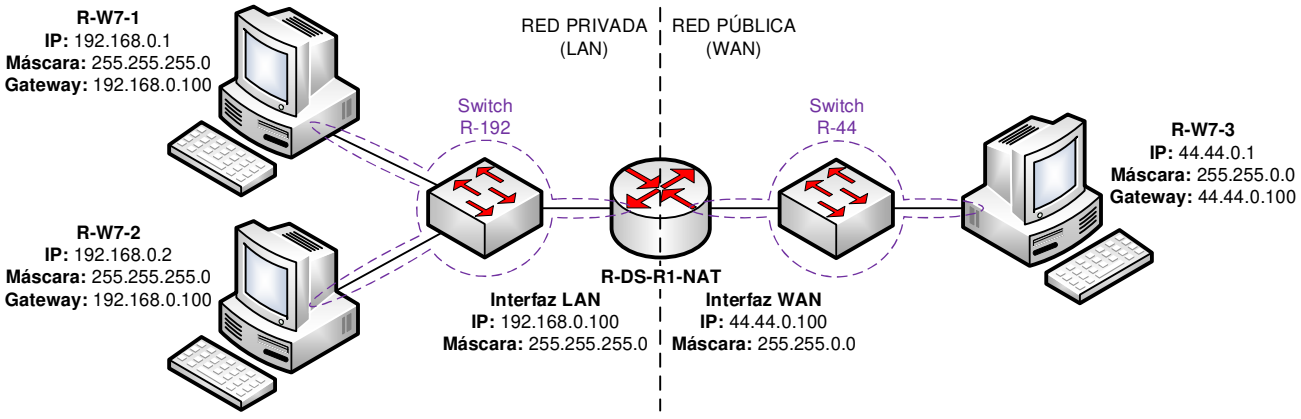

Figura 2

Partiremos del supuesto mostrado en la Figura 2, en el cual debe configurarse el R-DS-R1-NAT, que hace las funciones de rúter, para que haga NAT sobre todos los paquetes que salgan de la subred privada. De manera que traducirá cualquier IP origen, de la subred privada 192.168.0.0/24, a su IP pública, con lo cual todas las máquinas de la subred privada saldrán con la IP origen 44.44.0.100/16.

1.- Dar la configuración TCP/IP indicada a cada máquina.

En el resto del ejercicio asumiremos el siguiente contenido para el fichero */etc/network/interfaces* del R-DS-R1-NAT:

Opción 1

```
auto lo
   iface lo inet loopback

auto enp0s3
   iface enp0s3 inet static
   address 44.44.0.100
   netmask 255.255.0.0

auto enp0s8
   iface enp0s8 inet static
   address 192.168.0.100
   netmask 255.255.255.0
```

Opción 2

```
auto lo
   iface lo inet loopback

auto enp0s3
   iface enp0s3 inet static
   address 44.44.0.100/16

auto enp0s8
iface enp0s8 inet static
   address 192.168.0.100/24
```

2.- Activar el enrutamiento, en el R-DS-R1-NAT, y comprobar que todas las máquinas se ven entre sí. Recuerde que, para activar el enrutamiento, en sistemas *GNU/Linux*, es necesario que el fichero */proc/sys/net/ipv4/ip_forward* guarde el valor 1, por defecto guarda el valor 0, para lo cual puede obrarse de varias maneras, pero la más cómoda, en las versiones actuales de la mayoría de las distribuciones *GNU/Linux*, es acudir al fichero */etc/sysctl.conf* y "descomentar" la línea correspondiente, eliminando el carácter almohadilla, #, inicial de la misma. Para ello utilizaremos el editor de texto *nano*, en la forma siguiente:

sudo nano /etc/sysctl.conf

y acudiremos a la sección correspondiente al enrutamiento IPv4, quedando como:

```
#  Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```

Una vez realizado el cambio en el fichero, bastará con reiniciar el equipo (***sudo reboot***) o ejecuta un ***sudo sysctl -p*** para que sea efectiva la nueva configuración y dispongamos del servicio de enrutamiento habilitado.

3.- Para activar el NAT debemos realizar diversas operaciones previas.

3.1.- Creación de la tabla *nat* en las *nftables*.

Para crear la tabla *nat*, ejecutaremos la orden:

sudo nft add table ip nat

a través de la cual se le indica que debe crear una tabla denominada *nat* de la familia[3] *ip*, correspondiendo esta familia de tablas a aquellas cuyas reglas son de aplicación a los paquetes IPv4, la familia *ip6* a los de la versión 6 del protocolo IP y la *inet* a tablas cuyas reglas pueden aplicarse a ambas versiones del protocolo. La familia de tablas *ip* es la que se toma por defecto, de manera que podríamos utilizar la orden ***sudo nft add table nat*** con idéntica finalidad.

Una vez creada la tabla podemos comprobar su creación mediante la orden: ***sudo nft list tables***, cuya salida se muestra en la Figura 3. Adviértase que en la lista de tablas activas no se menciona a la tabla *filter*, cuya estructura se encuentra definida en el fichero */etc/nftables.conf*, tal y como se muestra en la Figura 1. Esto se debe a esa tabla todavía no se encuentra activa, ya que el servicio *nftables* aún no leyó ese fichero de configuración. Más adelante necesitaremos esa tabla y la activaremos como corresponde.

```
jefe@R-DS-R1-NAT:~$ sudo nft list tables
table ip nat
jefe@R-DS-R1-NAT:~$
```

Figura 3

Si en algún momento nos interesara borrar una tabla se utilizaría la orden:

nft delete table [<familia_tabla>] <nombre_tabla>

3.2.- Incorporar las cadenas *preenrutado* y *posenrutado* en la tabla *nat*.

Una vez creada la tabla *nat*, será necesario incorporarle las cadenas que le corresponden, la *preenrutado* y la *posenrutado*. Para crear estas cadenas se utilizará la orden cuya estructura se muestra a continuación:

nft add chain [<familia_tabla>] <nombre_tabla> <nombre_cadena> { type <tipo> hook <hook> priority <prioridad> \; [policy <policy> \;] }

En nuestro caso, ambas cadenas serán de tipo *nat* y el *hook* (el anclaje) en una será *prerouting* y en la otra *postrouting*. La prioridad se utiliza para colocar las cadenas en el orden que nos interese, a menor valor de prioridad antes se ejecutará esa cadena, en nuestro caso interesa que la cadena *preenrutado* se ejecute antes que la de *posenrutado*, según esto, le daremos una prioridad de -100 a la cadena *preenrutado* y de 100 a la de *postenrutado*, de acuerdo con lo recomendado[4]. La política de las cadenas, *policy*, la dejaremos por defecto, que es aceptar (*accept*). Más adelante, cuando trabajemos con la tabla *filter*, analizaremos las políticas por defecto.

```
jefe@R-DS-R1:~$ sudo nft list table nat
table ip nat {
        chain preenrutado {
                type nat hook prerouting priority dstnat; policy accept;
        }

        chain posenrutado {
                type nat hook postrouting priority srcnat; policy accept;
        }
}
jefe@R-DS-R1:~$
```

Figura 4

Llevando todo esto a la línea de comandos, las órdenes quedarán de la forma:

sudo nft add chain ip nat preenrutado {type nat hook prerouting priority -100 \;}

sudo nft add chain ip nat posenrutado {type nat hook postrouting priority 100 \;}

Creadas las dos cadenas de la tabla *nat*, podemos verificar su creación ejecutando la orden: ***sudo nft list table nat***, cuya salida se muestra en la Figura 4.

Adviértase que las prioridades de las cadenas constan como `dsnat` (nat destino), para la de *preenrutado*, y como `scrnat`, nat origen, para la de *posenutado*, Figura 4, lo que nos indica que las hemos establecido correctamente.

Si en algún momento quisiéramos borrar una cadena, utilizaríamos la orden:

nft delete chain [<familia_tabla>] <nombre_tabla> <nombre_cadena>

3.3.- Incorporación de la regla del NAT a la cadena *posenrutado*.

nft add rule <familia_tabla> <nombre_tabla> <nombre_cadena> <condiciones> <acción>

En nuestro caso, queremos ligar a la cadena *posenrutado* de la tabla *nat* una regla según la cual a todo aquel paquete IPv4, que salga (***o**utput*) por la interfaz (***ifname***) de IP pública (***oifname "enp0s3"***) se le sustituya la IP origen por la IP pública del rúter (***snat to 44.44.0.100***). Que llevado a la línea de comando quedaría como:

sudo nft add rule ip nat posenrutado oifname "enp0s3" snat to 44.44.0.100

Una vez incorporada la regla, comprobaremos si se añadió correctamente utilizando la orden:

sudo nft list table nat

Cuya salida se muestra en la Figura 5, en la que puede verse la regla creada en la cadena *posenrutado*, tal y como se esperaba.

```
jefe@R-DS-R1:~$ sudo nft list table nat
table ip nat {
        chain preenrutado {
                type nat hook prerouting priority dstnat; policy accept;
        }

        chain posenrutado {
                type nat hook postrouting priority srcnat; policy accept;
                oifname "enp0s3" snat to 44.44.0.100
        }
}
jefe@R-DS-R1:~$
```

Figura 5

Llegados a este punto, ya podremos comprobar el NAT levantando un *sniffer* en R-44, filtrando la captura por *icmp*, y lanzando un ***ping -n 2 44.44.0.1*** desde el R-W7-1. La captura correspondiente se muestra en la Figura 6 y en ella puede verse como las peticiones de eco, tramas 1 y 3, le llegan al R-W7-2 con la IP origen 44.44.0.100, IP a la que envía los ecos solicitados, tramas 2 y 4. Como vemos, todo funciona según lo esperado.

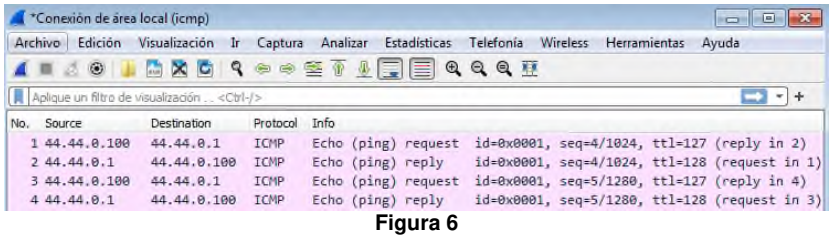

Figura 6

A la regla del NAT incorporada debe hacérsele un comentario importante. Solo es posible utilizar el ***snat***, en la regla, si el rúter tiene configuración IP estática. En el supuesto de que la configuración TCP/IP se le diera por DHCP, debe utilizarse el ***masquerade***, quedando la orden en la forma:

```
jefe@R-DS-R1:~$ sudo nft list table nat
table ip nat {
        chain preenrutado {
                type nat hook prerouting priority dstnat; policy accept;
        }

        chain posenrutado {
                type nat hook postrouting priority srcnat; policy accept;
                oifname "enp0s3" masquerade
        }
}
jefe@R-DS-R1:~$
```

Figura 7

sudo nft add rule ip nat posenrutado oifname "enp0s3" masquerade

En la Figura 7 se muestra la regla con el ***masquerade*** incorporada en la cadena *posenrutado*. El ***masquerade*** puede utilizarse tanto cuando el equipo tiene configuración TCP/IP estático como cuando la tiene por DHCP, ya que lo que hará será buscar la IP configurada en la interfaz indicada en la regla, para cada uno de los paquetes IPv4 que enrute. Obviamente esta búsqueda resta rendimiento al enrutado, razón por la cual cuando se trabaja con configuración TCP/IP estática en el equipo debe utilizarse el ***snat***, ya que la IP a utilizar en el NAT figura de forma explícita en la regla, Figura 5, lo que evita búsquedas innecesarias, aumentado el rendimiento del proceso de enrutado.

Hasta el momento se habló de las cadenas *preenrutado* y *posenrutado*, de la tabla *nat*, pero quizá no se haya explicado adecuadamente el por qué el NAT debe incorporarse en la cadena *posenrutado*.

Para entender el uso de la cadena *posenrutado* analizaremos el caso mostrado en la Figura 8. En ella se representa una LAN, constituida por dos subredes privadas distintas, que comparten una misma salida al exterior. Al mismo tiempo, el rúter se utiliza para el enrutamiento interior entre ambas subredes privadas.

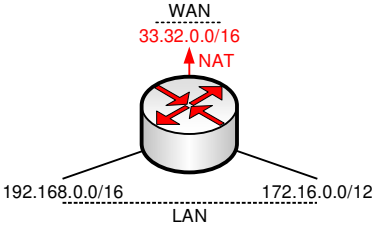

Figura 8

Al rúter de la Figura 8 se le pueden presentar dos casos; que le llegue un paquete de una de las subredes privadas con destino en la otra subred privada,

en cuyo caso no debe realizar ningún NAT, ya que se trata de tráfico interior. El otro caso sería aquel en el cual recibe un paquete de cualquiera de las subredes privadas con destino en el exterior, en algún lugar de la WAN, en este caso sí debe hacer el NAT, sea cual sea la subred origen del paquete. Según esto, para tomar la decisión sobre si se debe, o no, hacer el NAT sobre un paquete, el rúter debe saber, antes, a través de qué interfaz lo va a enviar, ya que solo debe cambiarle la dirección origen si lo enruta a través de la interfaz pública.

Por lo tanto, es claro que la decisión de hacer NAT debe tomarse a posteriori del enrutado, con lo cual la regla corresponderá, sin duda alguna, a la cadena *posenrutado*.

3.4.- El fichero */etc/nftables.conf*.

Visto el correcto funcionamiento del NAT, reiniciaremos el R-DS-R1 y podremos comprobar que el NAT dejó de funcionar. Es más, si ejecutamos un ***sudo nft list ruleset***, veremos que el origen de ese comportamiento se encuentra en que desapareció la tabla *nat* creada anteriormente.

Este comportamiento se debe a que todo lo hecho desde la línea de comandos se guarda, exclusivamente, en la memoria RAM, de manera que al reiniciar el equipo se pierde. Para evitar el que esto vuelva a ocurrir, crearemos de nuevo la tabla, las cadenas y la regla tal y como hicimos previamente, ejecutando las siguientes órdenes (recuerde que las tiene disponible en el *buffer* del teclado):

sudo nft add table ip nat

sudo nft add chain ip nat preenrutado {type nat hook prerouting priority -100 \;}

sudo nft add chain ip nat posenrutado {type nat hook postrouting priority 100 \;}

sudo nft add rule ip nat posenrutado oifname "enp0s3" masquerade

Creada, de nuevo, toda la estructura, verificaremos que todo funciona tal y como debe, lazando el ***ping -n 2 44.44.0.1*** desde el R-W7-1 y comprobando que en la captura realizada sobre el enlace R-44 circulan los paquetes con la IP pública del rúter, 44.44.0.100/16, tal y como se muestra en la Figura 6.

Como ya se conoce, el servicio *nftables* utiliza como fichero de configuración de tablas, cadenas y reglas el */etc/nftables.conf*, de manera que cuando se levanta el servicio lee ese fichero y crea, en memoria, las estructuras que corresponda. Dado que ya se dispone de ese fichero, Figura 1, lo que haremos será incorporarle al final del mismo la tabla *nat*, respetando la ya existente tabla *filter*, que utilizaremos en breve, para evitarnos el tener que crearla.

Para concatenar al final del fichero la tabla *nat* utilizaremos la orden ya conocida (debe ejecutarse con el rol de *root* ya adquirido, ***sudo su***, no basta con el ***sudo***):

nft list ruleset >> /etc/nftables.conf

con la diferencia que de que en este caso la salida de la misma la dirigimos para concatenarla (**>>**) al final del fichero que nos interesa, en lugar de utilizar la salida estándar, el monitor. Recuerde que si hace una redirección simple (**>**), en lugar de concatenar la tabla *nat*, sustituirá el contenido actual del fichero por la tabla *nat*, perdiendo la tabla *filter* ya contenida y que nos interesa conservar.

Si todo funcionó como debiera, el contenido del fichero */etc/nftables.conf* debe de ser el mostrado en la Figura 9, en la que pueden verse las estructuras de ambas tablas, tal y como se deseaba.

```
#!/usr/sbin/nft -f

flush ruleset

table inet filter {
    chain input {
        type filter hook input priority 0;
    }
    chain forward {
        type filter hook forward priority 0;
    }
    chain output {
        type filter hook output priority 0;
    }
}

table ip nat {
    chain preenrutado {
        type nat hook prerouting priority dstnat; policy accept;
    }
    chain posenrutado {
        type nat hook postrouting priority srcnat; policy accept;
        oifname "enp0s3" snat to 44.44.0.100
    }
}
```

Figura 9

Para entender el contenido de este fichero, es necesario indicar que se trata de un *script*, razón por la cual en su primera línea (`#!/usr/sbin/nft -f`) se indica la ruta absoluta del intérprete de comandos que se desea utilizar para su ejecución, precedida de los caracteres `#!`, cuya secuencia suele denominarse *shebang*[5] (que podría traducirse como asunto) o "número mágico"[6]. En este caso se hace referencia al intérprete de comandos *nft*, que es del *nftables*. El argumento `-f` le indica, al intérprete, que ese archivo debe ejecutarlo[7].

En la segunda línea se encuentra la orden `flush ruleset`, que se encarga de eliminar de la memoria todas las tablas, cadenas y reglas de las *nftables* que pudieran existir, antes de crear las nuevas estructuras contenidas en el fichero a ejecutar. De esta manera se evitan problemas por duplicidades o incompatibilidades entre lo existente y lo que se va a crear.

A continuación, aparece la estructura de la tabla *filter*, que como puede verse pertenece a la familia *inet*, lo que significa que las reglas de sus cadenas son de aplicación tanto para los paquetes IPv4, que se procesen, como los de IPv6. Tal y como se aprecia, la tabla *filter* posee tres cadenas: *input*, *output* y *forward*; que como ya se comentó serán las que se utilicen para incorporar las reglas correspondientes a los tráficos de entrada (*input*), de salida (*output*) y de reenvío (*forward*).

Termina el fichero con la estructura de la tabla *nat* que acabamos de concatenarle.

Incorporada la tabla *nat* en el fichero, reiniciaremos de nuevo el R-DS-R1. Y podremos comprobar que, a pesar de encontrarse la estructura de ambas tablas en el fichero, el NAT no funciona. Además, ejecutando un ***sudo nft list ruleset*** veremos que ninguna de las tablas se encuentra activa. Para entender lo que está ocurriendo empezaremos por comprobar el estado del servicio *nftables*, para lo cual ejecutaremos el comando:

```
jefe@R-DS-R1-NAT:~$ systemctl status nftables.service
● nftables.service - nftables
     Loaded: loaded (/lib/systemd/system/nftables.service; disabled; vendor preset: enabled)
     Active: inactive (dead)
       Docs: man:nft(8)
             http://wiki.nftables.org
jefe@R-DS-R1-NAT:~$
```

Figura 10

systemctl status nftables.service

que nos indicará que el servicio se encuentra inactivo, tal y como se muestra en la Figura 10.

A la vista de este resultado, intentaremos levantar manualmente el servicio, utilizando la orden:

sudo systemctl start nftables.service

tras lo cual volveremos a comprobar su estado:

systemctl status nftables.service

que en esta ocasión nos indicará que se encuentra activo. Ejecutando un:

sudo nft list ruleset

podremos comprobar que el servicio *nftables* leyó correctamente el fichero */etc/nftables.conf* y cargó en memoria las dos tablas, con sus correspondientes cadenas, tal y como se muestra en la Figura 11. Obviamente la regla del NAT también se encuentra activa, Figura 11; y como consecuencia, podremos verificar que la traducción de dirección origen funciona correctamente, lanzando un ***ping -n 2 44.44.0.1*** desde el R-W7-1 y capturando el tráfico en R-44, tal y como se hizo en la Figura 6.

```
jefe@R-DS-R1-NAT:~$ sudo nft list ruleset
table inet filter {
        chain input {
                type filter hook input priority filter; policy accept;
        }

        chain forward {
                type filter hook forward priority filter; policy accept;
        }

        chain output {
                type filter hook output priority filter; policy accept;
        }
}
table ip nat {
        chain preenrutado {
                type nat hook prerouting priority dstnat; policy accept;
        }

        chain posenrutado {
                type nat hook postrouting priority srcnat; policy accept;
                oifname "enp0s3" snat to 44.44.0.100
        }
}
jefe@R-DS-R1-NAT:~$
```

Figura 11

Es interesante comprobar, en la Figura 11, como las prioridades de las tres cadenas de la tabla *filter* aparecen como *filter*, cuando en el fichero de configuración constan como 0. Dando idea de que se establecieron correctamente, en forma paralela a lo ocurrido con las prioridades de las ordenes de creación de las cadenas *preenrutado* y *posenrutado* que se sustituyeron por *dstnat* y *srcnat*, respectivamente.

Concluidas las pruebas, todo parece funcionar correctamente en el servicio *nftables*, y el único problema que tenemos es que el servicio no se levanta automáticamente al arrancar el equipo, tal y como nos gustaría. Para solventar esa dificultad, tan solo debemos incorporarlo en la secuencia de arranque del *sysctl*, para lo cual ejecutaremos la orden:

sudo systemctl enable nftables.service

que nos creará las entradas correspondientes, en los archivos de secuencia de arranque, para que se levante el servicio automáticamente en cada inicio del equipo. Para probarlo bastará con reiniciar el equipo y verificar que las tablas del *nfttables* se encuentran activas y el NAT funcionando.

Para terminar el apartado sobre habilitar el NAT, debe indicarse que a diferencia de lo que ocurre al activar el servicio NAT en los sistemas *Windows Server 2022*, en sistemas *GNU/Linux* no se establece ningún tipo de restricción automática en el acceso desde el exterior a la red privada, dejándose en manos del administrador la inclusión de las reglas correspondientes en el cortafuegos (*firewall*) del sistema. Para comprobarlo bastará con lanzar un ***ping***, desde R-W7-3, hacia cualquiera de los equipos de la LAN, y verificar que se reciben los ecos sin problema alguno.

2.- CERRANDO EL *FIREWALL* A LOS ACCESOS EXTERIORES Y OTROS AJUSTES.

Tal y como ya se comentó, el *firewall* de los sistemas *GNU/Linux* no establece, de forma automática, ninguna regla específica de control de accesos, desde la red pública a la red privada, al habilitar el NAT, a diferencia de la forma de actuar del *Windows Server 2022*.

Para comprobar la afirmación anterior, bastará con lanzar un ***ping 192.168.0.1*** desde el R-W7-3 y verificar que se obtienen los ecos correspondientes, desde el R-W7-1. Sin embargo, en este comportamiento se esconde algo que quizá nos sorprenda. En las tramas 1 y 2 de la Figura 12 se recoge la captura, sobre R-44 y filtrada por el protocolo ICMP, correspondiente a un ***ping -n 1 192.168.0.1*** lanzado desde el R-W7-3. En la trama 1 se muestra la petición de eco lanzada por el R-W7-3 (IP 44.44.0.1), con destino en el R-W7-1, con IP 192.168.0.1. De manera que, hasta aquí, todo parece normal en el direccionamiento IP. A la hora