Departamento de Tecnología Electrónica
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Laboratorio 2
Filtrado de paquetes con Netfilter

Enunciados de Prácticas de Laboratorio
Tecnologías Avanzadas de la Información

1. Introducción y objetivos

La duración estimada de esta sesión de laboratorio es de **6 horas**. El propósito general de esta sesión de laboratorio es la configuración de un firewall para el sistema operativo Linux. Los principales objetivos se resumen como sigue:

- Describir el funcionamiento del filtrado de paquetes.
- Configurar un firewall.
- Configurar filtrado de paquetes en una máquina Linux mediante Netfilter / iptables.
- Elaborar reglas IPtables.

2. Descripción del firewall

Un cortafuegos o firewall es un sistema o grupo de sistemas utilizados para separar una máquina o una subred (zona protegida) del resto de la red (zona de riesgo), estableciendo una política de control de acceso entre ambos entornos. Es decir, el firewall actúa como punto de conexión segura entre dos o más sistemas informáticos. Un firewall puede ser un router, un PC o una red completa. Los principales elementos que pueden conformar un firewall, entendiendo el firewall como un sistema, son el filtro de paquetes y los proxys. En este laboratorio nos centraremos en el primer elemento, el filtrado de paquetes.

El filtrado de paquetes es un proceso que consiste en denegar o permitir el flujo de información entre la red interna que deseamos proteger y el resto o la red externa. Este filtrado se hace de acuerdo a unas reglas predefinidas, y según éstas, se examinan las cabeceras de los paquetes según van pasando a

 Paulino Ruiz de Clavijo Vázquez <paulino@dte.us.es>

través de él, decidiendo la acción a realizar con el paquete completo (aceptarlo, descartarlo, etc). El filtrado también se conoce como *screening*, y a los dispositivos que lo implementan se les denomina *chokes*.

Los firewalls de filtrado de paquetes actúan sobre la capa de red y la de transporte de la pila TCP/IP. Trabajan sobre la información de las cabeceras de los paquetes IP, sin llegar a analizar los datos. Por ejemplo, un firewall no puede evitar que un usuario de la red sobre la que actúa mande un email desde su equipo con otra cuenta de correo diferente de la de su trabajo. Lo que si podría es evitar que dicho equipo accediera al servidor de correo y así no se pudiese mandar ningún correo a nadie. Este filtrado se realiza a través de una lista de reglas. Las reglas pueden ser de diferentes tipos, aceptación, rechazo o denegación, entre otras.

Las implementaciones de los firewalls de filtrado de paquetes principalmente son de dos tipos: *stateless* y *stateful*. Un filtro estático o *stateless* (sin estado) analiza las cabeceras de cada paquete recibido (IP, TCP, UDP, ICMP, etc.) y toma una decisión de filtrado en función de los valores contenidos en los distintos campos de dichas cabeceras. No se establece ninguna relación entre los diferentes paquetes que atraviesan el filtro, aunque correspondan a una misma conexión. Este mecanismo de filtrado consume pocos recursos y es de fácil implementación.

Por el contrario un filtro dinámico o *stateful* (de estados) permite el control de un flujo de datos relacionados (por ejemplo, paquetes dentro de una misma conexión TCP o entre varias conexiones). Para llevarlo a cabo es necesario mantener en memoria los parámetros de cada conexión, tomando decisiones en función de la evolución de las mismas. Por ejemplo, sólo se permite el paso de datos en sentido entrante a través de un puerto TCP que haya sido previamente abierto en sentido saliente, o conexiones entrantes a un puerto dado desde una dirección origen cuando previamente se ha iniciado una conexión saliente hacia esa misma dirección desde otro puerto concreto.

Este modelo de filtrado es más sofisticado y permite un control más exhaustivo del tráfico resolviendo necesidades a nivel de paquete que antes tenían que resolverse a nivel de aplicación (mediante *proxies*).

2.1. Network Address Translation (NAT) / Enmascaramiento

NAT (Network Address Translation) es un mecanismo por el cual se alteran las cabeceras de los datagramas IP siendo su uso más común el cambio de las direcciones y puertos destino u origen en el router que lo implementa. En condiciones normales un datagrama IP viaja salto a salto a través de los routers manteniendo en su cabecera las direcciones IP origen y destino durante todo el trayecto. Cuando un router aplica NAT, altera el origen o destino del paquete en uno de los saltos, siendo necesario realizar igualmente la operación inversa en los paquetes de respuesta para que el origen pueda recibirlos. Principalmente la traslación NAT resuelve distintas necesidades:

- Cambio de la dirección y/o puerto destino de los datagramas recibidos en una red, redirigiéndolos a una máquina concreta en función del servicio requerido o distribuyéndolos entre distintas máquinas destino para el equilibrio de carga.
- Cambio de la dirección destino de un datagrama, para que el servicio requerido lo ofrezca un tercer equipo (proxy transparente).

- Cambio de la dirección de origen de un datagrama, asignándole otra dirección antes de reenviarlo hacia el destino. Este tipo de NAT se denomina enmascaramiento. Permite a uno o varios equipos de una red privada quedar visibles en otra red (por ejemplo Internet) a través de una dirección externa. Esta es la técnica utilizada en la actualidad para que múltiples equipos accedan a Internet a través de una única dirección pública.

Según lo visto podemos distinguir entre:

- Source NAT (SNAT): alteración del origen del datagrama, realizado después del encaminamiento del mismo y antes de su reenvío (el enmascaramiento es una forma de SNAT).
- Destination NAT (DNAT): alteración del destino, realizado antes del encaminamiento del datagrama (el port forwarding, el balanceo de carga y los proxy transparentes son ejemplos de DNAT).

2.2. Filtrado de paquetes en Linux

En Linux, el filtrado de paquetes está programado en el núcleo (como módulo o como componente estático). Los núcleos de Linux han ido evolucionando y con él su firewall, cambiando su implementación en las sucesivas versiones. Así, actualmente se utiliza el módulo *NetFilter* como firewall de filtrado de paquetes, el cual, junto con la herramienta *iptables* permite establecer las reglas de filtrado. El entorno Netfilter permite el filtrado de paquetes (ya sea con o sin estado), la traslación de direcciones y puertos (NAT /NAPT) y otras manipulaciones sobre el datagrama IP (*packet mangling*).

Netfilter ofrece una interfaz completa (o framework), dentro del núcleo de Linux, que permite interceptar y manipular paquetes de red. A Netfilter pueden conectarse otros módulos o componentes siendo el módulo más conocido *iptables*. Con este módulo y una herramienta con el mismo nombre, es posible manipular Netfilter desde el espacio de usuario. A veces se usa *iptables* para referirse a toda la infraestructura ofrecida por Netfilter. Con esto conseguimos que un módulo del kernel, más una utilidad de usuario controlen el flujo de paquetes que van desde y hacia las interfaces de red.

Otro módulo construido sobre Netfilter es *contrack* (Connection Tracking System) o sistema de seguimiento de conexiones. Con este módulo Netfilter puede funcionar como firewall de inspección de estados (*stateful inspection packet firewall*), también asociado a filtrado dinámico. A través de este mecanismo se puede asociar una cantidad de paquetes a una conexión en particular. Este tipo de inspección permite que el comportamiento del firewall cambie con respecto a la información contenida en el paquete, por ejemplo, de un paquete en particular que es parte de una conexión preexistente.

Con Netfilter, usando iptables podemos realizar:

- Filtrado de paquetes.
- Traducción de direcciones y puertos NAT.
- Manipulaciones sobre datagramas IP (packet mangling).
- Mantener registros de logs.
- Seguimiento de conexiones (*conntrack*)

Algunas de las características de Netfilter son:

- Permite el uso de distintas tablas de IP para realizar el filtrado: *nat*, *filter*, *mangle* y *raw*.
- Permite el uso de plugins para nuevas condiciones y acciones (Ej: *ipp2p* para filtrar conexiones a redes P2P). Así, no es necesario modificar los módulos para agregar una extensión adicional.
- De forma nativa maneja IPv4 e IPv6, usando la misma librería y el mismo código.

Netfilter realiza la gestión del filtrado mediante tablas organizadas en cadenas (chains) y éstas a su vez compuestas por reglas que se evalúan sobre los paquetes analizados en busca de condiciones según unos parámetros y, en caso de cumplirse alguna, ejecutando la acción asociada.

2.2.1. Cadenas (chains) en Netfilter

Las cadenas son agrupaciones de reglas que se deben aplicar a los paquetes en momentos concretos del flujo de los mismos a su paso por el sistema Netfilter. Indican cuando actuar sobre los paquetes, siendo las posibles cadenas que nos encontramos las siguientes:

- INPUT: Determina la acción a realizar cuando un paquete coincide con la regla a la entrada de la interfaz (filtrado de tráfico entrante). Se aplica a los paquetes destinados a la propia máquina.
- OUTPUT: Determina la acción a realizar cuando un paquete coincide con la regla, a la salida de la interfaz (filtrado de tráfico saliente). Se aplica a los paquetes originados en la propia máquina.
- FORWARD: Determina la acción a tomar cuando un paquete se envía desde una interfaz a otra. Se trata de una cadena transversal que se encuentra de forma intermedia entre las dos siguientes.
- PREROUTING: Es el primer estadio en el sistema Netfilter. Determina la primera acción a realizar antes de que el paquete entre en el sistema.
- POSTROUTING: Determina la acción a realizar, justo antes de enviar el paquete a la interfaz destino.

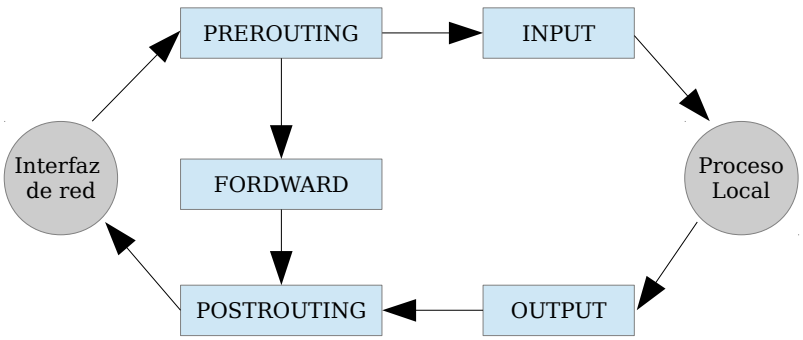

Figura 1. Esquema del procesamiento de NetFilter.

Cuando se recibe un paquete por cualquier interfaz se lleva a cabo, en primer lugar, una suma de comprobación (*checksum*). Si es correcta, los paquetes transitan hacia la evaluación de la cadena PREROUTING (en caso de existir), cuyas reglas se encargarán de determinar el tratamiento que deberá de darse al paquete en función de la dirección destino:

- Si el paquete va dirigido a la propia máquina, éste es enviado a la cadena INPUT, que en caso de superarse será procesado localmente.
- Si la dirección destino del paquete es distinta de la local, y está activada la función de reenvío, se evalúa la cadena FORWARD. Si se superan las reglas de esta cadena se reenvía el paquete. Si la

función de reenvío no estaba habilitada, el paquete se descarta (DROP). Se consideran paquetes no locales a aquellos que, por lo general, pertenecen a otra subred.

La cadena OUTPUT solamente es utilizada cuando los paquetes han sido originados localmente. Además, los paquetes que pasen por la cadena OUTPUT necesariamente pasan por POSTROUTING.

En todos los casos, el último estadio de Netfilter es POSTROUTING. Antes de que los paquetes abandonen el sistema Netfilter y sean enviados a la interfaz destino son recibidos por la cadena POSTROUTING.

2.2.2. Tablas en Netfilter

Las tablas son usadas para indicar al sistema de filtrado el tipo de procesamiento que se debe aplicar a los paquetes. Una tabla maneja una cierta cantidad de reglas internas que se organizan en cadenas y existen cuatro tablas por defecto: *filter, mangle, nat y raw*.

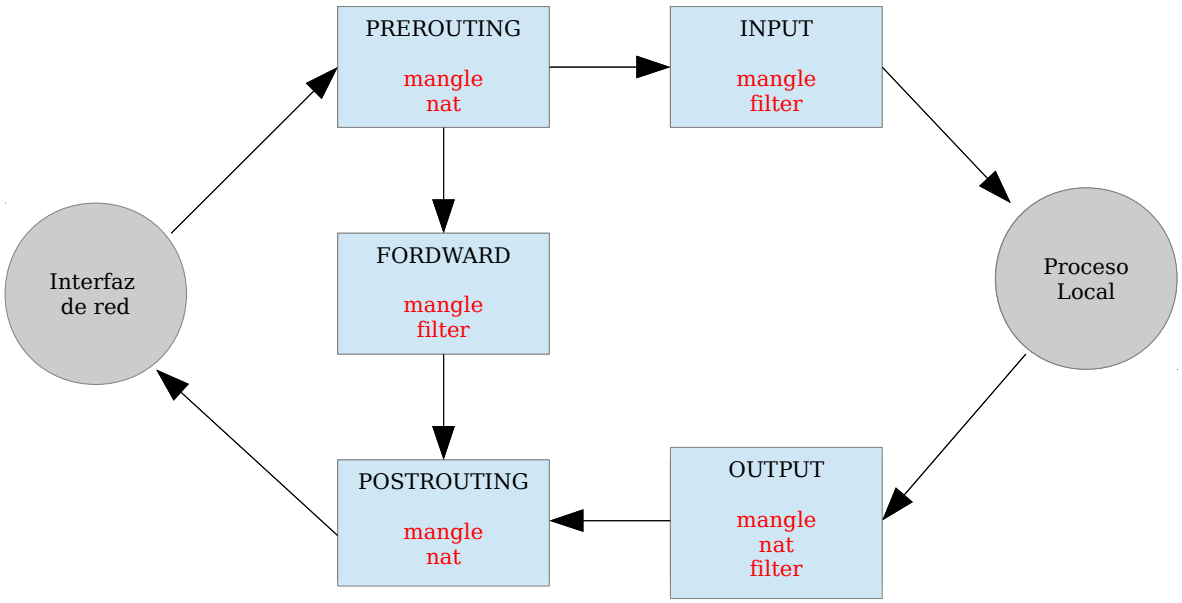

Figura 2. Tablas en el esquema del procesamiento de Netfilter.

- FILTER: Se usa para el filtrado general de paquetes y es la tabla predeterminada de Netfilter. Decide los paquetes que entran y los que no, y está compuesta por las siguientes cadenas: INPUT, OUTPUT y FORWARD.
- NAT: Controla la traducción de direcciones. Permite alterar las direcciones origen y destino del datagrama, analizando algunas propiedades del mismo. Está compuesta por las cadenas: PREROUTING, POSTROUTING y OUTPUT. Algunos usos típicos de NAT son:
  - SNAT (Source NAT): También conocido como *IP Masquerading*. Se cambia la dirección IP origen del datagrama al pasar por el router (después del encaminamiento y antes de su reenvío).
  - DNAT (Destination NAT): Se modifica la dirección IP destino antes del encaminamiento. Algunas aplicaciones de DNAT son:
    - Proxy transparente: Se redirecciona el datagrama a otro equipo que será quien proporcione los servicios requeridos.

- Balanceo de carga: Se cambia la dirección destino de los datos recibidos en un balanceador para redirigirlos a una máquina concreta en función del servicio requerido o entre máquinas distintas para balancear la carga.
- Port Forwarding: El router recibe las peticiones para la subred en la que trabaja y cambia la IP hacia la que tiene que dirigirse.

- MANGLE: Analiza ciertas características del paquete y lo marca en función de su naturaleza, para que reciba cierto tratamiento específico (ej: diferenciar tráfico en función de los servicios, etc.). Mangle permite la reescritura completa de paquetes (o tramas completas). En definitiva la tabla mangle controla los procesos de modificación del contenido y las opciones del paquete. Las cadenas que se agrupan en esta tabla son: INPUT, OUTPUT, FORWARD, PREROUTING y POSTROUTING.
- RAW: Se usa para configurar excepciones en el seguimiento de paquetes. La acción que siempre se usa para esta tabla es NOTRACK. Las cadenas que se organizan en está tabla son: PREROUTING y OUTPUT.

Es importante conocer que cada tabla tiene unas cadenas predeterminadas que no se pueden eliminar. A estas las cadenas de una tabla predeterminadas se pueden unir cadenas creadas por nosotros mismos para un mejor funcionamiento del filtrado o el enrutamiento.

Para comprender mejor el funcionamiento se muestran a continuación algunos ejemplos del recorrido que pueden realizar los paquetes en función de su destino.

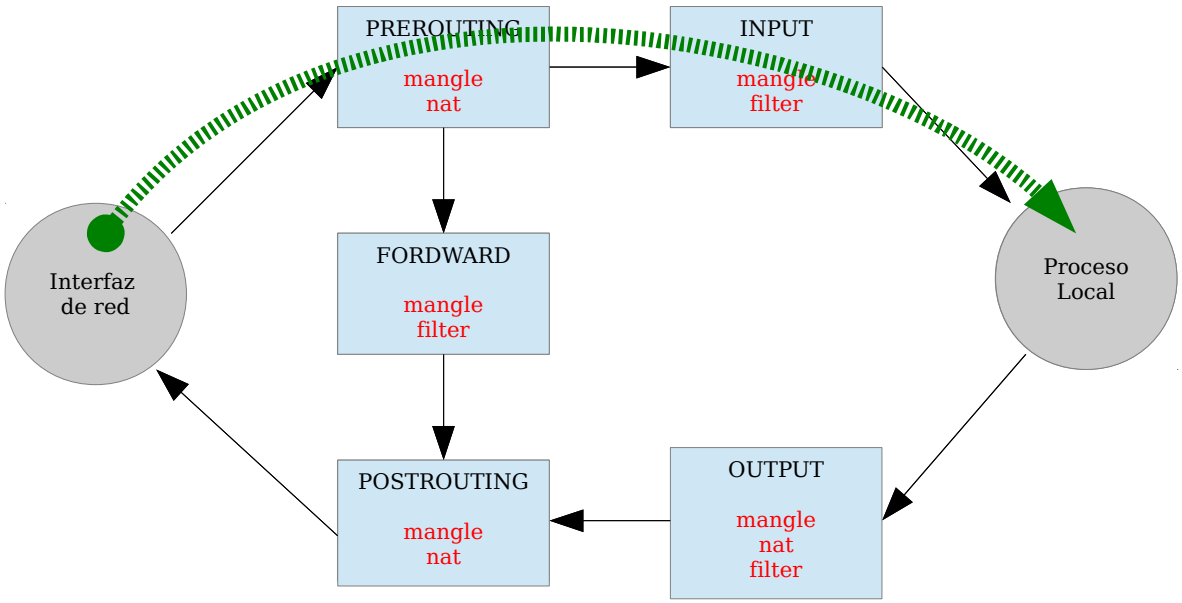

Figura 3. Flujo de paquetes entrantes con destino local.

Tabla	Cadena	Notas
		Datagrama recibido por una interfaz de red
Mangle	PREROUTING	Permite alterar algún parámetro de la cabecera (ej: campo TOS)
Nat	PREROUTING	Permite realizar DNAT
		Decisión de encaminamiento: ▪ Si dir.destino ≠ dir.local ⇒ Salta a Forward (reenvío) ▪ Si no, continúa
Mangle	INPUT	Alteraciones antes de procesamiento
Filter	**INPUT**	**Filtrado del tráfico entrante**
		Entrega al proceso local

Tabla 1. Recorrido de los paquetes entrantes con destino local.

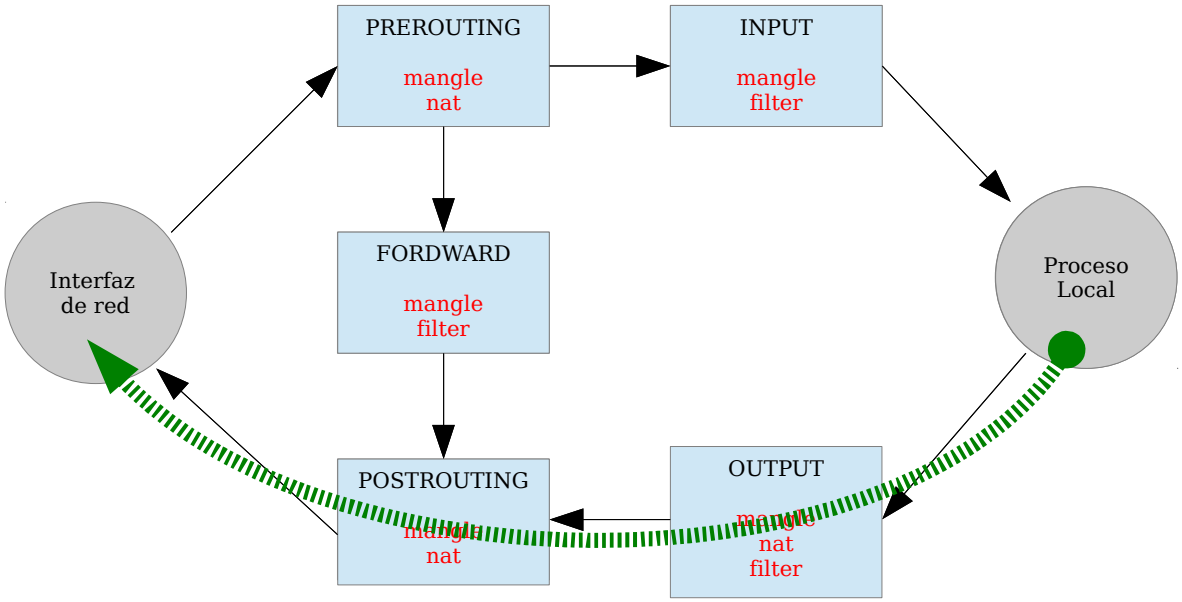

Figura 4. Flujo de paquetes salientes originados en local.

Tabla	Cadena	Notas
		Entrega por el proceso local
Mangle	OUTPUT	Permite alterar algún parámetro de la cabecera.
Nat	OUTPUT	Traslación de direcciones
Filter	**OUTPUT**	**Filtrado de tráfico saliente**
Mangle	POSTROUTING	Alteraciones antes de procesamiento
Nat	POSTROUTING	Permite realizar SNAT
		Entrega al interfaz

Tabla 2. Recorrido del paquete saliente con origen local.

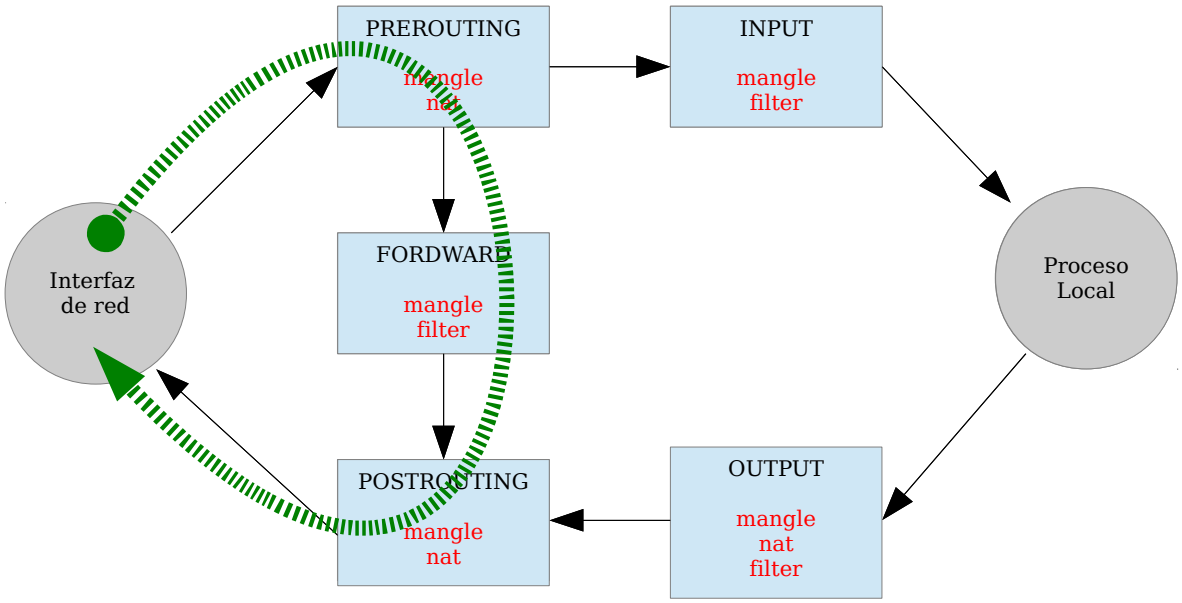

Figura 5. Flujo de paquetes reenviados.

Tabla	Cadena	Notas
		Datagrama recibido por una interfaz de red
Mangle	PREROUTING	Permite alterar algún parámetro de la cabecera
Nat	PREROUTING	Permite realizar DNAT
		Decisión de encaminamiento: ▪ Si dir.destino ≠ dir.local ⇒ Continúa ▪ Si no, salta a INPUT
Mangle	FORWARD	Alteraciones antes del reenvío
Filter	FORWARD	Filtrado del tráfico reenviado
Mangle	POSTROUTING	Alteraciones después del reenvío
Nat	POSTROUTING	Permite realizar SNAT
		Entrega al interfaz

Tabla 3. Recorrido de paquetes reenviados.

2.3. Reglas iptables

Como se indicó anteriormente, para el uso de Netfilter desde el espacio de usuario es necesario utilizar, además de los módulos del kernel, la herramienta *iptables*. A través del comando *iptables* es posible insertar, eliminar y modificar reglas dentro de Netfilter. Un comando básico de *iptables* está compuesto por 7 partes de la siguiente forma:

```
iptables [-t table] COMANDO CADENA condición acción [opciones]
├──────┤ ├────────┤ ├──────┤ ├─────┤ ├──────┤ ├─────┤ ├──────┤
   1        2        3       4       5       6       7
```

```
iptables -t filter -A INPUT -p tcp --dport 23 -j DROP
├──────┤ ├────────┤ ├┤ ├───┤ ├──────────────┤ ├─────┤
   1        2      3  4          5            6
```

Cuando no se indica la tabla a usar, por defecto se usa la tabla *filter* y las partes numeradas tienen el siguiente significado:

1. El comando *iptables* como punto de partida.
2. La tabla a usar (*filter*, *nat*, *mangle*). Si no se pone nada se usa por defecto *filter*.

3. El comando que deseamos aplicar a una cadena (insertar regla, modificar una existente, eliminar, etc.). Para definir la operación se usan una serie de comandos.
4. La cadena a usar, que puede ser una de las cadenas por defecto (INPUT, OUTPUT, FORWARD, PREROUTING o POSTROUTING) o bien aquellas creadas por el usuario.
5. La condición que especifica los criterios que debe de cumplir un paquete (los campos que lo componen) para que sea aplicable la acción.
6. La acción a realizar para aquellos paquetes que cumplan la condición.
7. Una serie de opciones que podemos aplicar para ajustar la acción.

Mediantes los comandos le indicamos a *iptables* qué deseamos hacer con la regla que vamos a definir. Esto es, agregar una regla a una cadena, modificar una regla existente en una cadena, eliminar el nombre de una cadena, etc. Describimos algunos de los comandos más comunes (todos deben escribirse en mayúsculas), entre los que distinguimos por un lado los destinados al manejo de cadenas, y por otro, al manejo de reglas dentro de una cadena.

Comando	**Descripción**
-h	Lista los comandos de iptables.
MANEJO DE REGLAS	
-A	Agrega una regla al final de la cadena especificada.
-D i	Elimina la regla i-ésima de una cadena.
-I i	Inserta una regla dentro de una cadena en la posición i-ésima.
-R i	Reemplaza la regla i-ésima por otra nueva en la cadena especificada.
-F	Elimina todas las reglas de una cadena. Es equivalente a borrar todas las reglas una por una.
-L	Lista las reglas de una cadena especificada.
-C	Verifica una regla antes de añadirla a la cadena especificada por el usuario. Se suele implementar en arquitecturas con reglas complejas.
MANEJO DE CADENAS	
-E	Renombra una cadena.
-N name	Crea una nueva cadena y se le pone nombre.
-X name	Borra una cadena especificada. Ha de estar vacía previamente.
-P	Cambia la política por defecto sobre una cadena, de forma que cuando los paquetes atraviesan la cadena sin cumplir ninguna regla, se envían a un objetivo como puede ser ACCEPT o DROP.
-Z	Pone a cero los contadores de todas las reglas de una cadena.

Tabla 4. Comandos iptables.

Las reglas se construyen por concatenación de condiciones y acciones asociadas. Cada condición evalúa una propiedad del paquete. Para indicar a Netfilter que hacer con los paquetes de una transacción, se deben crear reglas lo más precisas posible. La idea es que la condición sea inequívoca, tanto como para quien creó la regla (usuario) como para el kernel.

Una regla está formada por una condición y una acción, `REGLA = <condición,acción>`, y el procesamiento se resume como sigue:

1. Si se cumple la condición, se ejecuta la acción asociada y se detiene el procesamiento de la cadena.
2. Si no se cumple la condición se pasa a la siguiente regla.
3. Si no ha cumplido la condición de ninguna regla en la cadena se ejecuta la política por defecto

sobre el paquete (p.e. ACCEPT ó DROP).

A medida que se van ejecutando órdenes de *iptables*, se van añadiendo o eliminando reglas asociadas a cada uno de los flujos de entrada o salida. Es importante entender que las reglas que se añaden se procesarán de forma secuencial (DROP y ACCEPT finalizan el procesamiento). Esto supone que para deshacer un efecto la solución es eliminar la regla que lo causa en vez de intentar añadir otra que contradiga a la primera (por ejemplo no vale añadir un ACCEPT después de un DROP porque el segundo no anulará el primero). En *iptables* las acciones se ejecutan con el parámetro `-j [acción]`, como por ejemplo el siguiente comando: `iptables -A INPUT -j DROP`

Una condición (coincidencia / match) ocurre cuando un paquete cumple con los criterios indicados dentro de alguna de las cadenas. Alguno de estos criterios pueden basarse en función de algún parámetro como, por ejemplo, el tipo de protocolo (TCP, IP, ICMP, etc.), algún puerto en particular, un usuario propietario del paquete (OWNER), el estado de la transacción (INVALID), o la combinación de todos ellos. Las condiciones se construyen usando una serie de operadores que determinan la lógica que el paquete debe cumplir. Algunos de los más importantes son los siguientes:

Operador	**Descripción**
-p [protocolo]	Indica sobre qué protocolo ha de realizarse la comprobación. Algunos valores pueden ser *tcp*, *udp*, *icmp* o podemos referirnos a todos si se omite. Los nombres de protocolos reconocidos por el istemas están indicados en `/etc/protocols`. Ejemplo: `-p tcp,udp`
-s [ip/mascara destino]	Indica la dirección IP del origen del paquete. Puede indicarse también de la forma IP/máscara para decirle a Netfilter que se trata de un grupo de hosts. Si no se especifica ésta condición se tomará como origen todas las direcciones de difusión. Ejemplo: `-s 192.168.1.0/24`
-d [ip/mascara destino]	Indica la dirección IP destino de la transacción. Ejemplo: `-d 192.168.1.0/24`
-i [interfaz]	Indica la interfaz de entrada, desde donde se realiza la transacción o se reciben los paquetes, para una regla en particular. (Nota: sólo usado por las cadenas INPUT, FORWARD y PREROUTING). Con la tabla *filter* sólo se podrán utilizar las cadenas INPUT y FORWARD y PREROUTING cuando se utilice *nat* o *mangle*. Ejemplos: `-i eth0` , `-i eth+`
-o [interfaz]	Indica la interfaz de salida de la transacción para una regla en particular (Nota: sólo usada por OUTPUT, FORWARD y POSTROUTING en las tablas *nat* y *mangle*). Ejemplo: `-o eth0` , `-o eth+`
-f	Aplicación de la regla sólo a los paquetes fragmentados.
-m	Uso de módulos para extender funcionalidades.

Tabla 5. Operadores iptables.

Además de estos operadores existen otros que junto a éstos extienden sus funcionalidades concretando aún más la condición que se desea determinar. A estos operadores se denominan extensiones y se muestran en la siguiente tabla.

Operador	Extensión	Descripción
-p [protocolo]	--sport	Puerto origen. Sólo para tcp y udp. Ejemplos: `-p tcp --sport 0:53` `-p tcp,udp --sport 1023`
	--dport	Puerto destino. Sólo para tcp y udp. Ejemplos: `-p tcp --dport 23` `-p tcp,udp --dport 0:1023`
	--tcp-flags arg1 arg2	Para los paquetes TCP que se analicen se comprueban una serie de flags. En esta opción deben establecerse dos argumentos: ▪ arg1: Los indicadores a comprobar. ▪ arg2: Indicadores habilitados. Los valores que se pueden usar son: ACK, RST, FIN, SYN, URG, PSH. Ejemplo: `-p tcp --tcp-flags SYN,ACK,RST SYN` En el ejemplo se comprueban todos los indicadores pero solo SYN debe estar habilitado.
	--syn	Indica que el flag SYN debe estar activado y que el indicador ACK debe ponerse a cero cuando, en un mensaje TCP, se realiza una petición de establecimiento de conexión. Ejemplo: `-p tcp --syn`
	--icmp-type	Selecciona los paquetes ICMP y comprueba de qué tipo de mensaje (de control) se trata. Ejemplos: `-p icmp --icmp-type echo-reply` `-p icmp --icmp-type time-exceded`
-m [módulo]	mac	Módulo MAC: Comprueba la dirección MAC de los paquetes. *-m mac --mac-source <dir.MAC>*, se comprueba la dirección MAC origen Ejemplo: `-m mac --mac-source 00:02:3F:34:9B:E1 -j DROP`
	state	Módulo STATE: Comprueba el estado de la conexión del paquete. Puede ser: ▪ NEW: Creación de nueva conexión. ▪ ESTABLISHED: El paquete forma parte de una conexión establecida. ▪ RELATED: Conexión relacionada a una ya establecida. ▪ INVALID: Paquetes no identificados en ninguna conexión. Ejemplo: `-m state --state NEW -j DROP`
	limit	Módulo LIMIT: Establece un número límite de veces que una regla puede ser aceptada en un determinado periodo de tiempo. Sintaxis: *-m limit --limit <número>[/tiempo]* Ejemplo: `-m limit --limit 5/s` El tiempo puede ser: *second, min, hour, day.* También se puede especificar un número determinado de paquetes a recibir: *-m limit --limit-burst <nPaquetes>*
	multiport	Módulo MULTIPORT: Comprobación de varios puertos simultáneamente. Hay que especificar el tipo de protocolo. Sintaxis: *-m multiport --[d/s]ports <puertos>* Ejemplo: `-p tcp -m multiport --dports 53,80,442`

	recent	Módulo RECENT: Permite monitorizar conexiones recientes y limitarlas. Se añaden IPs a una lista con la que se comparan posteriores intentos de conexión. Evita ataques de fuerza bruta y DoS. Algunos operadores que extienden a recent son: ▪ --set: Crea una nueva lista de IPs ▪ --name: Renombra una lista (por defecto es la lista DEFAULT). ▪ --update: Comprueba si las IPs ya existen en una lista. ▪ --hitcount: Número máximo de ocurrencias. ▪ --seconds: Intervalo de tiempo en segundos.

Tabla 6. Extensiones.

Además, las condiciones pueden precederse de alguno de los siguientes operadores:

- "!" excluye los adaptadores especificados
- "+" todos los adaptadores deben concordar en una regla particular

Por último, en el parámetro de acción del comando *iptables*, las acciones al final de la regla indican el destino final en el proceso de filtrado de un paquete o de una transacción. Este destino para el paquete es efectivo una vez que se ha cumplido la condición. Algunas acciones son comunes de todas las cadenas aunque otras, son específicas. Las acciones básicas son las siguientes:

- 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.
- MASQUERADE [dirección_ip]: Enmascaramiento de la dirección IP origen de forma dinámica. Esta acción es sólo válida en la tabla NAT en la cadena POSTROUTING
- DNAT --to [dirección_ip][:puerto]: Enmascaramiento de la dirección destino. Muy usada para re-enrutado de paquetes.
- SNAT --to [dirección_ip][:puerto]: Enmascaramiento de la IP origen. De forma similar a MASQUERADE, pero con IP fija.
- LOG: Crea una entrada en el fichero de log.

Otras acciones adicionales son: DENY, REDIRECT, RETURN y MIRROR. En principio, sólo se corresponde una acción por cada condición cumplida.

2.4. Política por defecto

Se ha indicado con anterioridad como mediante el comando `iptables -P` se puede cambiar la política predeterminada a aceptar todo (ACCEPT) o descartar todo (DROP). Este comportamiento genérico, que afecta a todo tipo de tráfico, puede después ser refinado añadiendo nuevas reglas que modifiquen el comportamiento por predeterminado. Por ejemplo, se puede decidir aceptar todo el tráfico inicialmente pero después añadir una nueva regla que impida determinado tipo de tráfico, como una conexión FTP.

En vez de descartar un paquete mediante DROP es posible realizar un REJECT, que envía un datagrama

ICMP de *puerto inalcanzable* (ésta es la acción por defecto). Emplear REJECT en lugar de DENY impide el acceso a los puertos de una forma más cortés pero también permite a un posible atacante comprobar rápidamente qué puertos se encuentran abiertos en nuestro sistema.

Con iptables el administrador define una política predeterminada para el tráfico entrante o saliente y después, con un conjunto de reglas adicionales, habilita o bloquea determinado tráfico de red. En este proceso resulta fundamental definir bien cuál es la política por defecto más conveniente. Si lo que se desea es un sistema lo más restringido posible, entonces lo más conveniente es descartar cualquier tipo de tráfico excepto el que se autorice explícitamente. En este caso podemos comenzar impidiendo cualquier tráfico saliente para después añadir tan sólo aquellas comunicaciones que deseamos autorizar, como por ejemplo los accesos al servidor DNS y las conexiones SSH a un determinado servidor. Cualquier otro tráfico distinto del autorizado será rechazado por esa restrictiva política por defecto.

Sin embargo, también es posible que lo que deseemos sea tan sólo impedir cierto tipo de tráfico sin alterar el resto de servicios. Quizá queremos evitar que una determinada aplicación pueda funcionar, por ejemplo que los usuarios no puedan imprimir en una cierta impresora remota desde ese ordenador. En este caso se impone partir de una política por defecto que acepte todo tipo de tráfico para después introducir una regla que bloquee específicamente el tráfico que se dirija a esa impresora de red.

3. Realización del laboratorio

A continuación se realizará un ejercicio guiado mostrando los comandos para realizar cada una de las tareas planteadas.

Tarea 1.- Compruebe la lista de reglas activas mediante el comando `iptables -L`.

Como situación inicial se nos mostrará la tabla *filter* puesto que no se ha especificado la tabla en la ejecución del comando. Para esta tabla aparecen sus tres cadenas por defecto (INPUT, FORWARD y OUTPUT) y para cada cadena, se muestra la política predeterminada a seguir, es este caso, es aceptar todo tipo de tráfico (*policy ACCEPT*) sin importar ni el origen, ni el destino ni el protocolo usado. Esta configuración se puede denominar permisiva y no pone restricciones sobre el tráfico de entrada (INPUT) o de salida (OUTPUT), ni tampoco sobre el posible tráfico que atravesará este equipo si éste actuara como un router (FORWARD).

Tarea 2.- Pruebe el comando `iptables -P OUTPUT DROP`.

T2.1.- Ahora realice `ping localhost` ¿que ocurre?.

T2.2.- Devuelva el equipo a su estado inicial mediante `iptables -P OUTPUT ACCEPT`. Pruebe de nuevo el comando `ping` para asegurarse que funciona correctamente.

La orden de la Tarea 2.- modifica la política por defecto para todo el tráfico saliente, incluido aquel que no abandona el sistema (*localhost*) de modo que el sistema se ha convertido en una especie de agujero negro en la red del que ningún paquete puede salir. Un efecto parecido (aunque no tan drástico) se puede producir si desconectamos el cable de red (en este caso el ping anterior seguiría funcionando correctamente).

Muchas infraestructuras sitúan un cortafuegos entre su red local y la conexión a Internet. Este

dispositivo incluye algunas reglas que filtran determinados paquetes con el fin de mejorar la seguridad de la red interna. En este laboratorio se puede hacer que nuestro equipo rechace, de manera selectiva, determinado tipo de tráfico. Para ello vamos a necesitar reglas un poco más elaboradas que la empleada en el ejercicio anterior.

Tarea 3.- Se va bloquear el tráfico local, es decir, el que se produce en el dispositivo *lo* mediante el comando `iptables -A INPUT -i lo -j DROP`.

T3.1.- Verifique si tiene el efecto deseado tecleando `ping localhost`. ¿Obtenemos respuesta?. Ahora pruebe a hacer `ping www.dte.us.es` ¿Funciona?

T3.2.- Ejecute el comando `iptables -L -v` y vea el contador de paquetes que han satisfecho la regla anterior. Tal y como se muestra a continuación, identifique la regla añadida en el formato mostrado.

```
Chain INPUT (policy ACCEPT 6 packets, 494 bytes)
 pkts bytes target     prot opt in     out     source               destination
    2   168 DROP       all  --  lo     any     anywhere             anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 8 packets, 618 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

T3.3.- Para poder restablecer el tráfico local solo hay que eliminar la regla anterior mediante el comando `iptables -F INPUT`. Este último comando borra todas las reglas de INPUT.

En este ejercicio se ha visto como mediante el parámetro *-i*, se especifica un dispositivo de red y con *-j* se indica qué hacer con el tráfico que coincida con esa regla.

Hay que considerar que el dispositivo *lo* no es en realidad un dispositivo físico de red, sino la representación de las comunicaciones internas mediante la dirección de bucle local 127.0.0.1. En el ejemplo se ha establecido que todo el tráfico que se reciba (INPUT) por el dispositivo *lo* tiene que descartarse (DROP). Para conseguir reglas más útiles no basta poder especificar el dispositivo, sino que es necesario poder afinar indicando qué protocolo y/o puertos disparan una regla en particular.

Tarea 4.- Compruebe que se puede acceder mediante SSH a un equipo que tenga este servicio activo y establezca una conexión utilizando un usuario y contraseña mediante `ssh usuario@host_remoto`. Puede intentar conectarse al equipo de algún compañero usando la dirección IP: `ssh tai@192.168.20.X`.

T4.1.- Abra otro terminal en su ordenador y en ella ejecute el comando `iptables -A INPUT -p tcp --sport 22 -j DROP`.

T4.2.- Ahora vuelva a la ventana de la conexión SSH y teclee algún comando. ¿Qué sucede? ¿Por qué?. Compruébelo con el comando `iptables -L -v`.

T4.3.- Vuelva al segundo terminal y ejecute el comando `iptables -F INPUT`.

T4.4.- ¿Qué sucede? ¿Aún sigue conectado por SSH al equipo remoto?

T4.5.- Piense en una regla que tenga el mismo efecto que el de la tarea T4.1.- pero usando la cadena OUTPUT. Pruébela para asegurarse que funciona.

En este ejemplo se ha creado una regla que no especifica el dispositivo de red sino el protocolo (TCP) y también el puerto origen de los segmentos (*--sport 22*). En una conexión SSH los paquetes que vienen

del servidor SSH tienen como puerto origen el 22 que es el puerto predeterminado del servicio SSH.

De modo análogo es posible aplicar esta misma estrategia para bloquear el acceso a cualquier otro servicio. No obstante, las reglas se pueden aplicar tanto al tráfico entrante como al saliente, o bien a ambos. En el ejercicio anterior se bloqueaba el tráfico SSH entrante (INPUT). Si se tarda más de un minuto entre el paso 2 y el paso 4 del ejercicio, es posible que la conexión SSH se haya interrumpido. En ese caso lo más conveniente es repetirlo intentando ser algo más rápido (lo que permitirá que la conexión no se interrumpa y se obtenga un resultado diferente).

También es posible crear reglas que atiendan a las direcciones de los paquetes, así se muestra en la siguiente tarea.

Tarea 5.- Abra un navegador y cargue la página *https://www.dte.us.es.* Si no tiene instalado ninguno utilice *aptitude* para instalar *firefox*.

T5.1.- Ahora en una ventana de terminal ejecute el comando `iptables -A OUTPUT -p tcp -d www.dte.us.es --dport 443 -j DROP`.

T5.2.- Intente recargar la página en el navegador. ¿Qué ocurre?.

T5.3.- Pruebe a visitar otra página. ¿Funciona?.

Se puede observar en este ejercicio como se ha creado una regla que descarta el tráfico TCP saliente hacia al servidor *www.dte.us.es* y destinado al puerto 443 (HTTPS). Esta regla no afecta al tráfico similar enviado a cualquier otro servidor. Con esta regla sólo se impide el acceso al servidor WEB principal del DTE. Es posible crear un conjunto de reglas similares para poder impedir el acceso a una lista de distintos servidores WEB. Recuerda que a menos que borres esta regla, su efecto perdurará hasta que el equipo sea reiniciado.

3.1. Registro de sucesos

La funcionalidad de iptables no sólo permite especificar reglas para descartar paquetes (como hemos visto en varios de los ejemplos). Con una política predeterminada para descartar paquetes (DROP) es necesario más reglas, que deberían ser especificadas para aceptar determinados tipos de tráfico. Pero además de estas funciones las reglas pueden producir acciones de registro que se anotarán en el registro de sucesos del sistema (archivo */var/log/kern.log*). Para crear reglas que al dispararse generen una entrada en el registro se usará la opción *–j LOG*. Estas reglas no determinan si el paquete se acepta o se rechaza, por que se aplicará la acción que corresponda como si esta regla de registro no existiera.

El interés de poder registrar determinados sucesos asociados con el tráfico de red depende de las situaciones que se estén considerando. Puede tener un efecto informativo para el administrador sobre multitud de datos que pudieran estar registrados en otros lugares o no. Por ejemplo, si deseamos conocer cuántas personas se conectan cada día a nuestro servidor FTP es muy posible que el programa servidor disponga de un detallado archivo de registro con esa información, pero si se trata de un servidor muy elemental podría no generar tipo alguno de información de registro.

Vamos a suponer que nos encontramos en este segundo caso y que nos piden determinar cuántos clientes se conectan cada día a un servidor mediante SSH. Lo primero que necesitamos es determinar

que condición consideramos como válida para establecer que ha llegado un nuevo cliente. La más sencilla (aunque no necesariamente la más correcta) es considerar que cada nueva conexión al puerto 22 de nuestro servidor es un nuevo cliente.

Tarea 6.- Cree la regla que realice el registro de eventos mediante `iptables -A INPUT -p tcp --dport 22 -j LOG`.

T6.1.- Ahora acceda al servidor SSH desde otra máquina de la misma red, por ejemplo desde *Servidor1* con el comando `ssh tai@192.168.7.1`.

T6.2.- Ahora en la máquina *gateway* desde un terminal ejecute el comando `dmesg -H` y analice las últimas líneas que aparecen, puede avanzar y retroceder con las teclas *AvPag/RePag*. ¿Se pueden interpretar?.

Con la regla de la Tarea 6.- se genera una entrada en el registro cada vez que un paquete llega al equipo y va destinado al servidor SSH. Sin embargo existe un serio problema que no resulta aparente. El problema reside en que esa regla de registro se cumple para cada segmento SSH recibido de cada conexión. Eso quiere decir que un mismo cliente puede generar miles de entradas en una misma conexión (fíjese en el número de entradas generadas en *dmesg*). Llenar un archivo de registro con muchos datos poco significativos no es práctico. Para realizar un registro más selectivo se necesita que sólo se registre una vez a cada cliente. Una forma de hacer esto es considerar sólo el segmento del comienzo de la conexión, que por tanto tiene el indicador SYN activo.

Tarea 7.- Borre todas las reglas mediante mediante: `iptables -F`

T7.1.- Ahora cree una regla de LOG que genere entradas en el registro sólo si detecta las peticiones de conexión al puerto 22. Para ello use el comando:
`iptables -A INPUT -p tcp --dport 22 --tcp-flags ALL SYN -j LOG`.

T7.2.- Los mensajes mostrados con `dmesg` quedan registrados en el fichero de bitácora `/var/log/kern.log`. Para ver este fichero en tiempo real utilice el siguiente comando `tail -f /var/log/kern.log`, considere que el terminal se quedará bloqueado hasta que pulse la combinación CTRL+C.

T7.3.- Si ahora intenta conectarse al servidor SSH observará en el terminal donde se ejecuta el comando *tail* un resultado similar al del ejercicio anterior, pero en tiempo real.

Observando ahora las líneas creadas en el fichero *kern.log* se ve como sólo se ha generado una única entrada. La regla de registro ahora presta atención al campo de *flags* de la cabecera TCP, se analizan todos los bits de la cabecera (por eso el valor ALL) y se registran todos los segmentos recibidos que tengan el bit SYN activado.

3.2. Modificando direcciones y/o puertos de destino (NAT)

Ahora se propone modificar direcciones y/o puertos de destino NAT. En el siguiente ejercicio vamos a crear una regla en la que aplicaremos las funciones de NAT disponibles en Netfilter.

Tarea 8.- Abra el navegador y visite la página *http://www.dte.us.es.*

T8.1.- Mediante el uso del comando `ping` o el comando `host` averigüe la dirección IP del sitio web.

T8.2.- Cree la siguiente regla `iptables -t nat -A OUTPUT -p tcp -d 150.214.141.196 --dport 80 -j DNAT --to 150.214.188.143:80`.

T8.3.- Repita la tarea anterior pero para el protocolo HTTPS y navegue a https://www.dte.us.es. ¿Por qué con el protocolo HTTP Firefox no detecta una posible alteración del sitio Web y con HTTPS sí?

T8.4.- En el mismo terminal se mantendrá visible el listado de reglas usando el comando *watch* del siguiente modo: `watch iptables -t nat -L -v`. El terminal se quedará bloqueado, cuando desee terminar la visualización continua pulse la combinación CTRL+C.

T8.5.- Tras volver a recargar la página en el navegador ¿que ocurre?,¿y en la ventana con el terminal que ejecuta el comando *watch*?

T8.6.- Elimine todas las reglas de la tabla NAT para evitar futuros problemas, para ello utilice le forma correcta la opción `--flush` / `-F` del comando `iptables`.

Si repasamos la orden creada se observa como ahora la acción de la regla *–j* no es ni LOG ni DROP como en casos anteriores, sino DNAT. Esta regla permite reescribir las direcciones y/o los puertos de destino de un segmento. En este caso todos los segmentos TCP dirigidos a la dirección IP de la WEB del DTE (150.214.141.196) serán redirigidos al servidor web de la dirección 158.42.250.41 (portal del LSI).

Se puede observar en esta regla que se especifica una nueva tabla *-t nat*, en los casos mostrados hasta ahora la tabla usada era la tabla por defecto *-t filter*, la cual no era necesaria especificarla en la línea de comandos. Esta nueva tabla permite realizar operaciones de traducción de puertos y direcciones como las que realiza un router convencional.

3.3. Cambios en otros campos (MANGLE)

Otra de las posibilidades existentes con Netfilter es la modificación de los valores de otros campos en el tráfico analizado. Uno de los posibles es el campo de tipo de servicio (TOS) de la cabecera IP, así, personalizándolo para una determinada aplicación es posible adecuar el valor de este campo a las necesidades de la misma. Incluso aunque su valor no sea respetado más allá de nuestro sistema, puede ser interesante para que distintos flujos de tráfico simultáneos se puedan tratar adecuadamente según nuestra elección.

Para el último ejemplo vamos a escoger algo más sencillo, el campo de tiempo de vida (TTL) de la cabecera IP. Con él se especifica el número máximo de saltos entre routers permitidos para alcanzar un destino. El valor utilizado por nuestro equipo para este campo viene prefijado en la configuración del sistema operativo y aunque se puede modificar, la mayoría de administradores no suelen tener razones para hacerlo. Sin embargo, no todos los sistemas operativos utilizan el mismo valor, y si nos fijamos, es posible que veamos computadores con distintos valores iniciales para el TTL, se puede usar el comando *ping* para determinarlo.

Supongamos que deseamos modificar el valor del campo TTL para cierto tipo de tráfico (no para todos nuestros paquetes). Es posible crear una regla con netfilter que realice ese cambio selectivo.

Tarea 9.- Use el comando `traceroute` hacia 150.214.141.196 (*www.dte.us.es*) para averiguar el valor

TTL para llegar a un cierto destino.

T9.1.- Instale la herramienta *mtr* con el comando `apt install mtr` y úsela para los casos anteriores.

T9.2.- Realice la misma prueba con *www.google.es* cuya IP es 216.58.201.131.

T9.3.- Ahora, añada la siguiente regla `iptables -t mangle -A POSTROUTING -j TTL --ttl-set 5`.

T9.4.- Compruebe si puede acceder a 150.214.141.196 (*www.dte.us.es)* y 150.214.141.131 (*www.informatica.us.es*).

T9.5.- Compruebe si ahora puede acceder a 216.58.201.131 (*www.google.es*).

T9.6.- ¿Por qué en esta tarea se realiza el acceso por IP y no se usa el nombre?

T9.7.- Finalmente elimine todas las reglas de la tabla *mangle* para evitar futuros problemas.

Con ese valor tan sólo se pueden realizar cuatro saltos antes de que el datagrama sea descartado. Esto limita enormemente el número de redes que se pueden visitar. Un valor TTL=1 impediría atravesar un único router y solo permitiría la comunicación directa con otros ordenadores en la misma subred.

3.4. Ejercicios no guidados

El objetivo de las tareas de esta sección es crear reglas y comprobar mediante un test que cumplen el objetivo deseado. Para realizar las comprobaciones se utilizará la utilidad *netcat*, por ello, la siguiente tarea muestra un uso básico de esta herramienta para comprobar si determinados puertos TCP admiten conexiones. El comando se instala con `apt install netcat-traditional` pero por simplicidad ya se encuentra preinstalado las máquinas virtuales.

Tarea 10.- Use un terminal para poner *netcat* a escuchar el puerto TCP 8000. El comando es `nc -l -p 8000`, tras su ejecución el terminal parecerá que está bloqueado, pero no es así, está esperando datos entrantes en el puerto 8000 para mostrarlos por la salida.

T10.1.- Para listar los programas que están escuchando en diferentes puertos utilice otro terminal y ejecute el comando `ss -lptn`. Localice en la salida del comando anterior el programa *nc* escuchando en el puerto 8000.

T10.2.- Ahora desde ese otro terminal vuelva a usar *netcat* pero ahora para conectar a un puerto remoto, en este caso al 8000 mediante el comando `nc localhost 8000`. Tras conectar escriba un texto en un terminal y compruebe que se transfiere al otro terminal.

T10.3.- Para finalizar el ejercicio sobre *netcat* repita la prueba conectando con el ordenador de algún compañero. Debe poner *netcat* a escuchar en un puerto e indicarle a un compañero su dirección IP para que utilice *netcat* y se conecte con su terminal. El comando será parecido a `nc 192.168.20.X 8000` y observe como los terminales funcionan como un chat.

Tras la prueba con *netcat*, en la siguiente parte del laboratorio debe guardar las reglas creadas en cada tarea en un fichero llamado *lab2.sh* situado en el directorio */root/bin*. Este fichero será un fichero de shell ejecutable en el que se escribirá la secuencia de sentencias a ejecutar. Para evitar que algunos comandos se ejecuten utilice el carácter "**#**", deberá usarlo tanto para comentar una la línea, como para escribir sus propios comentarios dentro del mismo fichero. Además de escribir en el fichero el comando iptables correspondiente, para cada una de las reglas y según se indique, debe realizar un test con la herramienta *netcat*. En primer lugar se creará el fichero de shell ejecutable siguiendo los siguientes

pasos:

Tarea 11.- Desde la línea de comandos entre en el directorio */root* y cree un directorio llamado *bin*. Para asegurarse que está en la ubicación correcta, entre en el nuevo directorio */root/bin* con el comando `cd`. Ahora, ejecute el comando `pwd` (este comando muestra la ruta completa actual) para asegurarse que está en */root/bin* y no en el directorio del sistema */bin*.

T11.1.- Cree un fichero llamado `lab2.sh`, para ello puede usar el editor *nano*: `nano lab2.sh`.

T11.2.- Copie el contenido de mostrado en el código 1 en el fichero y guarde los cambios.

```
#!/bin/bash

set -x   # Esta línea activa la depuración de los scripts
set -e   # Esta línea para la ejecución de los comandos cuando hay un error

# Esta línea comienza por '#' así que es un comentario
echo “Esta línea es un comando, no comienza por #”

# Comandos para listar reglas activas de cada tabla


# Comando para vaciar una cadena
```

Código 1. Contenido de ejemplo para el fichero /root/bin/lab2.sh.

T11.3.- Establezca los permisos de ejecución para el script *lab2.sh* mediante el comando `chmod +x lab2.sh`. Para comprobar si se ha cambiado ejecute el comando `ls -l` y observe los indicadores de permisos de los ficheros.

T11.4.- Ejecute el script mediante `./lab2.sh` y compruebe con la salida del comando si tiene el permiso de ejecución recién establecido.

T11.5.- Edite el script y comente la línea que ejecuta el comando *echo* y tras guardar los cambios vuelva a ejecutarlo.

Tarea 12.- Realice los siguientes ejercicios escribiendo el comando *iptables* correspondiente en el fichero *lab2.sh* y ejecutando dicho script. Utilice varias ventanas de terminal y no olvide escribir un comentario sobre lo que está haciendo dentro del mismo fichero para su posterior comprensión. Debe buscar un modo de comprobar si cada una de las reglas añadidas cumple su cometido.

T12.1.- Cree una regla que descarte los mensajes ICMP entrantes de origen local, el cual corresponde al interfaz *lo*. Compruebe esta regla mediante el comando *ping*.

T12.2.- Cree una regla que descarte todo el tráfico ICMP entrante. Compruebe de nuevo la regla como crea oportuno.

T12.3.- Permita que se puedan enviar paquetes ICMP a través de la interfaz externa (red 192.168.20.0/24) y realice el test correspondiente.

T12.4.- Deniegue las sesiones *telnet* entrantes desde la interfaz externa. Para testear esta regla no instale un servidor *telnet*, use el comando `nc -l -p telnet` en un terminal. Debe comprobar que se puede usar el puerto *telnet* desde localhost pero no desde el ordenador de un compañero.

T12.5.- Deniegue las conexiones al puerto 22 por la interfaz interna. La interfaz interna está conectadas las máquinas *Servidor1* y *Servidor2*, por lo que debe realizar la prueba desde una de sus maquinas virtuales internas usando SSH.

T12.6.- Rechace la conexión al puerto 65000, protocolo UDP, desde los computadores de la red 192.168.7.0/24. Para comprobar esa regla debe usar el comando *netcat* en modo UDP, para ello ejecute `nc -h` para mostrar la ayuda donde se indica como hacerlo.

T12.7.- Rechace el tráfico saliente en el rango de puertos 6881-6889, usados habitualmente por el protocolo *BitTorrent*, busque en Internet que tipo de protocolo es este BitTorrent y como se suele filtrar.

T12.8.- Deniegue el tráfico desde la interfaz interna a la externa para que no se pueda compartir Internet, debe usar FORWARD en la regla.

T12.9.- Deniegue el tráfico saliente hacia el puerto 25 (correo electrónico SMTP) desde la interfaz externa a los paquetes marcados con las banderas SYN y ACK a la vez.

3.5. Caso práctico

Finalmente en base a la red virtual desplegada para los laboratorios de la asignatura (figura 6), se debe configurar la red estableciendo una concreta configuración para la máquina Gateway. Esta configuración será persistente, es decir, aunque la máquina se reinicie las reglas establecidas se cargarán automáticamente.

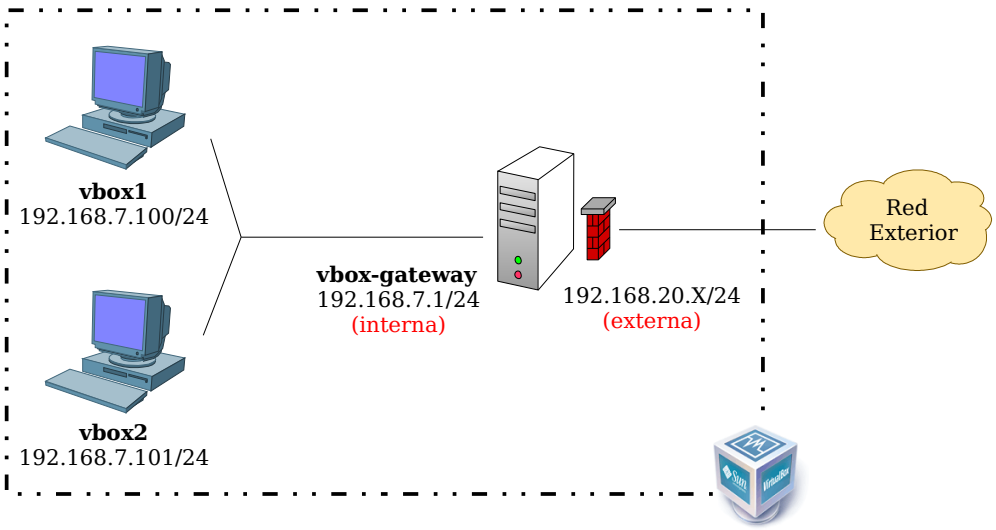

Figura 6. Esquema de configuración de la red virtual.

Como ejercicio final, se propone crear un script ejecutable que contenga las reglas para que la máquina Gateway comparta la conexión a Internet mediante NAT con una serie de restricciones.

Tarea 13.- Cree un nuevo fichero en el directorio */root/bin* llamado *firewall.sh*. Compruebe con `pwd` que no se ha equivocado de directorio y establezca el permiso de ejecución al fichero. Edítelo y en la primera línea escriba `#!/bin/bash`.

T13.1.- En las primeras líneas del fichero escriba los comandos *iptables* necesarios para vaciar todas las reglas de todas las tablas. Al menos son 3 comandos los necesarios.

T13.2.- Escriba una regla que permita el reenvío de paquetes a través del router desde la red 192.168.7.0/24 para compartir la conexión a Internet (SNAT). Consulte por Internet las diferentes soluciones existentes, puede usar una regla tipo SNAT o MASQUERADE ambas funcionarán.

T13.3.- Para que opere la regla anterior, debe activar el reenvío de paquetes. Debe editar el fichero */etc/sysctl.conf* y activar la línea `net.ipv4.ip_forward=1`. Ejecute el comando `sysctl -p` o reinicie el equipo para que surja efecto.

T13.4.- Ejecute como *root* el script mediante el comando `./firewall.sh`. Compruebe si las máquinas *Servidor1* y *Servidor2* tienen acceso a Internet usando el comando *ping*.

Tras el reinicio de la máquina Gateway las reglas iptables no son persistentes, para cargarlas ha sido necesario la ejecución como usuario root del script *firewall.sh*. Las distribuciones Linux tienen varios programas para hacer persistentes las reglas de netfilter. Para servidores basados en Debian una solución flexible y simple es el paquete *iptables-persistent*.

Tarea 14.- Instale *iptables-persistent* con *apt*, durante la instalación le preguntará si guarda la reglas actuales, conteste `Sí`.

T14.1.- Acceda la directorio */etc/iptables* y liste los ficheros en este directorio. Edite con *nano* el fichero `rules.v4`. ¿Se parece el contenido de este fichero al fichero *firewall.sh* que está construyendo?

T14.2.- Reinicie la máquina ejecutando el comando `reboot`, compruebe si tras el reinicio las máquinas internas siguen teniendo conexión a Internet y se han cargado las reglas de netfilter. Para ello, lístelas con el comando `iptables -t nat -L`.

A partir de aquí se va a establecer una configuración completa del firewall añadiendo las reglas necesarias al fichero *firewall.sh*. Concretamente se trata de determinar qué paquetes estarán permitidos para salir de la red a través del firewall. Este primer ejercicio es de filtrado, por ello debe usar la tabla *filter* (la cual es usada por omisión). Los paquetes a analizar son los dirigidos a Internet desde las máquinas internas de la red, por ello debe usar la cadena FORWARD añadiendo las reglas mediante `-A FORWARD`. Siguiendo esta indicación general realice la siguiente tarea.

Tarea 15.- Añada las reglas indicadas a continuación, probando en cada caso si funcionan correctamente, para ello, ejecute el script *firewall.sh* con cada cambio que realice, al estar activas las opciones `set -x` y `set -e`, en caso de error, el script se dentendrá en la línea donde está el error mostrando previamente todas las órdenes ejecutadas.

T15.1.- La primera regla debe establecer en el firewall la política predeterminada, a descartar todo el tráfico procedente de la red interna que no haya sido explícitamente aceptado por una regla definida, la política se establece con la opción `-P` de iptables. Compruebe si opera correctamente mediante el comando *ping* en las máquinas internas, asegúrese que la máquina Gateway no ha perdido la conexión a Internet.

T15.2.- La tarea anterior suele llevar a confusión a veces, ya que se debe cambiar la *política* predeterminada y por ello no hay que añadir reglas. Con esto, compruebe que ha realizado la tarea anterior correctamente listando las reglas de la cadena FILTER, no debería tener ninguna regla en la lista y debe identificar que efectivamente se ha cambiado la política a DROP: `policy DROP`.

T15.3.- Cree las reglas necesarias para que los equipos de la red interna puedan usar correcta mente el comando `ping` hacia máquinas externas.

T15.4.- Cree dos reglas que permitan el tráfico DNS saliente y entrante. Recuerde que el tráfico DNS utiliza el protocolo UDP en un determinado puerto. Para probar si opera correctamente desde

las máquinas internas utilice el comando `ping` con un nombre de dominio existente, por ejemplo `ping google.es` y verifique que el sistema resuelve la IP correspondiente.

T15.5.- Intente actualizar la lista de paquetes con `aptitude` o `apt update` en las máquinas internas. ¿Qué ocurre?.

T15.6.- Cree varias reglas para que los equipos de la red interna puedan salir a Internet y tener tráfico WEB (puertos 80/TCP y 443/TCP). Compruebe si funciona el comando `apt update` cuando tenga la regla operativa.

T15.7.- Para evitar instalar un escritorio gráfico en las máquinas internas, instale el paquete *elinks*, el cual, es un navegador WEB en modo texto. Úselo para comprobar si las reglas de la tarea anterior operan correctamente.

Una vez finalizada la configuración del firewall es necesario hacerlo persistente a los reinicios del sistema. Para ello se instaló anteriormente el paquete *iptables-persistent*, pero éste no opera de forma automática, por ello realice la siguiente tarea para aprender como mantener el firewall actualizado.

Tarea 16.- Cargue el firewall ejecutando el script `/root/bin/firewall.sh` y liste las reglas con el comando `iptables -L -v` para verificar que están cargadas correctamente.

T16.1.- Reinicie la máquina gateway, tras el reinicio vuelva a listar las reglas de netfilter y verá como las reglas no son persistentes a pesar de tener instalado de tareas anteriores el paquete *iptables-persistent*.

T16.2.- El paquete *iptables-persistent* carga las reglas durante el inicio del sistema desde el fichero `/etc/iptables/rules.v4`. Cuando se modifiquen las reglas se debe actualizar este fichero con el comando `iptables-save`, por tanto, ejecute los comandos siguientes para cargar el firewall y hacerlo persistente:

```
/root/bin/firewall.sh
iptables-save > /etc/iptables/rules.v4
```

Código 2. Comandos necesarios para conseguir que las reglas de netfilter sean persistentes.

T16.3.- Reinicie la máquina y tras iniciar, liste las reglas con iptables para verificar que se han cargado correctamente.