Práctica 7: Cortafuegos IPTABLES

Usaremos la máquina virtual con Linux y una conexión VPN establecida a la UPV en la máquina aniftriona

La pila de protocolos TCP/IP forma parte de prácticamente la totalidad de los sistemas operativos actuales, como GNU/Linux, Windows, macOS, Android o iOS. Puesto que es el sistema operativo el encargado de manejar las peticiones de comunicación de las aplicaciones, resulta bastante natural plantear la posibilidad de poder ajustar el comportamiento de dichas comunicaciones para especificar qué políticas son más adecuadas según el criterio del administrador. Cada sistema operativo incluye su propio modo de especificar esas preferencias. En el caso de GNU/Linux esta herramienta es *iptables*.

Seguro que conoces el término "cortafuegos" (firewall) en un contexto de redes de computadores. Se trata de dispositivos (tradicionalmente hardware) que permiten filtrar determinado tipo de tráfico con el fin de proteger algunos equipos y redes de amenazas exteriores.

Muchas empresas colocan 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.

Iptables es una herramienta muy potente que permite al administrador consultar y modificar las políticas de gestión del tráfico de red en cada equipo. Entre otras cosas, permite las funciones propias de un *firewall* (como bloqueo de tráfico no deseado), la gestión del protocolo NAT en los *routers*, el registro de eventos o la modificación del contenido de la cabecera de los datagramas IP. En esta práctica veremos algunos de los usos básicos de esta potente herramienta.

Una de las principales desventajas de esta herramienta es que la configuración de las *iptables* no queda almacenada en el sistema, por lo que al apagar o reiniciar el equipo la configuración se pierde. El administrador debe crear manualmente un *script* para que se ejecute durante el arranque del sistema operativo para cargar la configuración deseada en las *iptables*.

Debido a lo complicado que puede llegar a ser el manejo de las *iptables* y al hecho de que la configuración no se mantenga al apagar la máquina, en los sistemas Linux actuales podemos tener disponible la herramienta *ufw* (*Uncomplicated FireWall*) que permite administrar las opciones más habituales de una forma mucho más amigable. *Ufw* hace uso de las *iptables*, por lo que podemos considerar que es solo una herramienta para facilitar la configuración de las *iptables*. Además, *ufw* configura el arranque del sistema operativo para que en los siguientes reinicios del sistema se cargue la última configuración.

Aún así, es posible que en el sistema Linux que tengamos que configurar no dispongamos de ninguna herramienta adicional que nos permita configurar las *iptables* y tengamos que utilizar esta herramienta, por lo que un conocimiento de su funcionamiento básico es necesario para cualquier administrador.

Conceptos básicos

En *iptables* los datagramas que llegan o salen del sistema a través de cualquier interfaz de red, incluyendo las virtuales como *loopback*, son procesados siguiendo el siguiente diagrama de flujo.

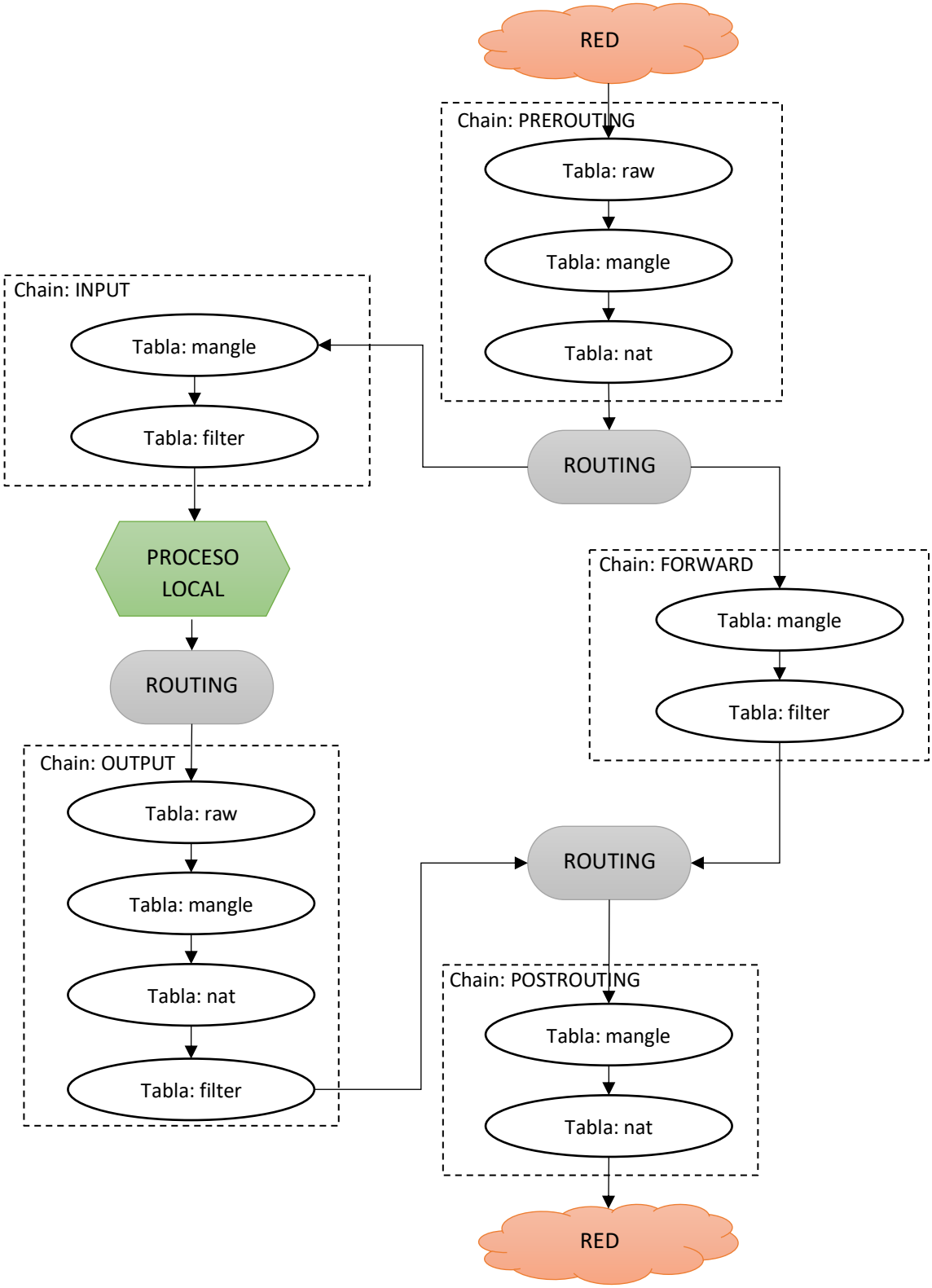

Ilustración 1: Diagrama de flujo de las iptables

Para entender dicho diagrama de flujo, hay que tener en cuenta que las *iptables* están organizadas en una serie de tablas. Cada tabla está formada por un conjunto de cadenas predefinidas (*chains*) y las cadenas contienen reglas que se recorren en orden consecutivo.

En dicho diagrama podemos identificar tres rutas posibles que pueden seguir los datagramas IP en las *iptables*. En un *host*, el tráfico de entrada recorrería la ruta que va desde la **red** (la representada arriba) hasta el **proceso local**. El tráfico de salida recorrería las tablas desde el **proceso local** hasta la red (representada, en este caso en la parte inferior). En los *routers* el tráfico de paso recorrería la ruta desde la **red** situada arriba hasta la **red** situada abajo, atravesando la parte de la cadena FORWARD (situada a la derecha).

Por ejemplo, un datagrama de entrada a un ordenador personal pasará por las siguientes fases en *iptables*.

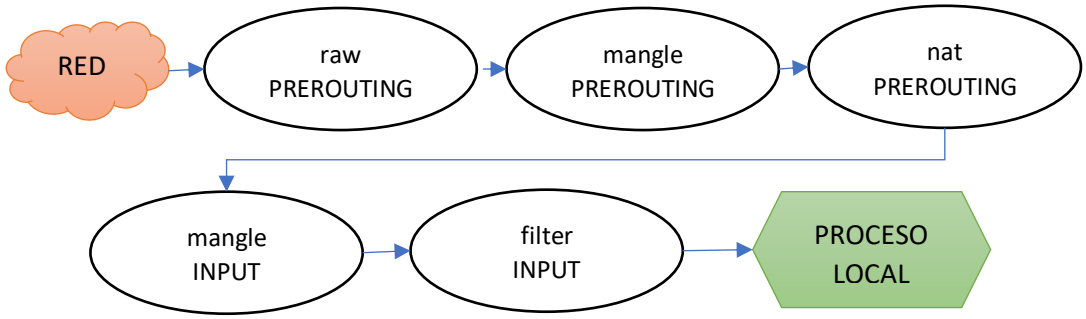

Ilustración 2: Diagrama de flujo de las iptables *para el tráfico de entrada al dispositivo*

> *NOTA: El nombre en minúscula es el nombre de la tabla, y el nombre en mayúscula es el nombre de la cadena de reglas.*

Como podemos ver, en cada una de las fases no se procesan todas las reglas de una tabla, sino que se procesan las reglas pertenecientes a una cadena de dicha tabla. Puede comprobarse en el diagrama anterior que la tabla *mangle* procesa dos cadenas distintas en momentos distintos, la cadena PREROUTING y la cadena INPUT.

Un datagrama de salida en un ordenador personal con Linux seguirá el siguiente diagrama:

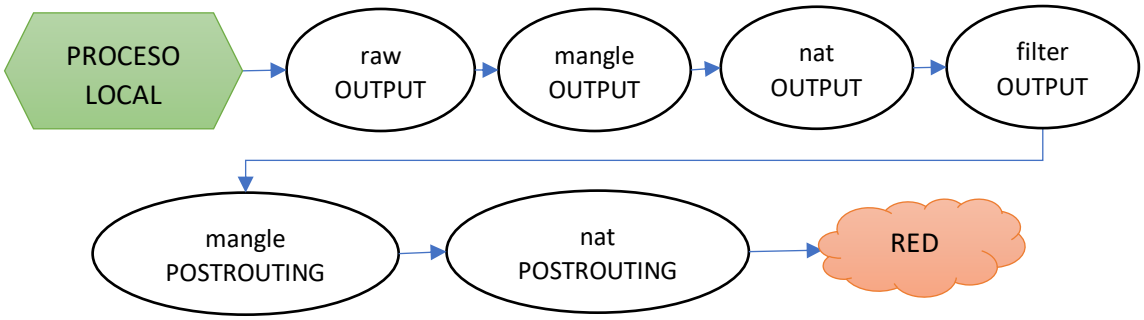

Ilustración 3: Diagrama de flujo de las iptables *para el tráfico de salida del dispositivo*

Podemos comprobar que los datagramas de salida de una máquina y los de entrada procesan cadenas distintas en cada tabla, por lo que, a la hora de añadir reglas que afecten al tráfico hemos de tener muy presente sobre qué tráfico queremos actuar, sobre el de salida o sobre el de entrada.

Las tablas de *iptables*

iptables cuenta con cinco tablas:

- **raw**: filtra los paquetes antes que cualquier otra tabla. En los dos diagramas anteriores podemos ver que la primera cadena por la que pasa un datagrama en ambos pertenece a la tabla *raw*.
- **mangle**: Contiene reglas para modificar algunos campos de las cabeceras de los protocolos TCP/IP como, por ejemplo, ToS o TTL en IP, o los *flags* SYN, RST, etc. en TCP.
- **nat:** Se utiliza para realizar el servicio de traducción de direcciones (NAT).
- **security:** Se utiliza para las reglas de conexión de red *Mandatory Access Control*.
- **filter:** Es la tabla por defecto. Contiene las reglas de filtrado. Mediante esta tabla se puede, por ejemplo, descartar o aceptar datagramas IP según diferentes condiciones.

La tabla *filter* es la tabla más utilizada, puesto que sobre esta tabla se realizan la mayor parte de las operaciones propias de un *firewall*. Dentro de la tabla *filter*, en concreto actuaremos sobre las cadenas INPUT y OUTPUT.

Las cadenas de reglas en *iptables*

Como se ha comentado anteriormente, cada tabla está formada por varias cadenas de reglas. Por lo general, una regla está formada por una condición y una acción. Para evaluar una regla, *iptables* comprueba la condición en el datagrama que se está procesando y si se cumple, entonces realiza la acción.

Las principales acciones que vamos a utilizar por ahora en esta práctica son ACCEPT y DROP. Una regla cuya acción sea ACCEPT indica que el datagrama debe ser aceptado. DROP, por el contrario, descartará el datagrama.

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 DROP impide el acceso a los puertos de una forma más cortés pero también permite a un posible atacante comprobar más rápidamente qué puertos se encuentran abiertos en nuestro sistema.

Es importante entender que las reglas de una cadena se procesarán de forma secuencial hasta que se encuentre alguna que se ajuste a las propiedades del paquete analizado. En ese caso se realizará sobre el paquete la acción especificada por la regla, por ejemplo, DROP o ACCEPT, y ya no se siguen procesando las restantes reglas de la cadena. Por el contrario, si se procesan todas las reglas de la cadena sin que ninguna se ajuste a las características del paquete, se le aplicará la política por defecto para esa cadena.

Algunas órdenes útiles

Existen algunas órdenes en *iptables* que son bastante útiles y que nos pueden sacar de algún apuro cuando cometemos errores al crear reglas.

Para **ver el estado de una tabla** en *iptables* podemos utilizar la siguiente orden (los corchetes significan que son parámetros opcionales):

```
sudo iptables [-t tabla] -L [cadena]
```

La tabla por defecto es la tabla *filter* por lo que si no indicamos el parámetro **-t** veremos el estado de dicha tabla. Además, si no indicamos ninguna cadena nos mostrará el estado de todas las cadenas de la tabla.

Ejemplos:

```
sudo iptables -L
```

La orden anterior nos muestra el estado de todas las cadenas de la tabla *filter*.

```
sudo iptables -L INPUT
```

La orden anterior nos muestra el estado de la cadena INPUT de la tabla *filter*.

En ocasiones, cuando hemos añadido varias reglas a una tabla de forma errónea y no tenemos claro cómo corregir los errores, la solución más fácil es borrar todas las reglas de la tabla e insertarlas de nuevo.

Para **borrar todas las reglas de una tabla o de una cadena** se usa el parámetro **-F** (de **f**lush).

Ejemplos:

```
sudo iptables -F
```

La orden anterior borra todas las reglas de todas las cadenas de la tabla *filter*.

```
sudo iptables -t nat -F
```

La orden anterior borra todas las reglas de todas las cadenas de la tabla *nat*.

```
sudo iptables -F OUTPUT
```

La orden anterior borra todas las reglas la cadena OUTPUT de la tabla *filter*.

Políticas por defecto

La política por defecto de una cadena indica el comportamiento por defecto de los datagramas en el caso en el que la cadena no contenga ninguna regla, o en el caso en el que un datagrama no cumpla las condiciones de ninguna de las reglas de dicha cadena.

Este comportamiento genérico que afecta a todo tipo de tráfico puede, después, ser refinado añadiendo nuevas reglas más específicas que modifiquen, para paquetes específicos, el comportamiento por defecto. 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.

Con *iptables* el administrador define una política por defecto 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 que resulta "molesto" sin alterar el resto de los 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.

Cómo cambiar la política por defecto con la orden *iptables*

Un conocimiento detallado de *iptables* requiere mucho más tiempo que el de esta práctica. Puesto que la gestión de las comunicaciones se realiza en el núcleo del sistema operativo sólo el administrador tiene acceso a esta herramienta. Por este motivo, en nuestra máquina virtual de Linux, utilizaremos la orden sudo precediendo siempre a la orden *iptables*.

Prueba a ejecutar la siguiente línea para obtener el estado actual de todas las cadenas de la tabla *filter*.

```
$ sudo 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
```

> *NOTA: Aunque es poco probable que suceda en nuestra máquina virtual, si como resultado obtuviéramos muchas más cadenas con el prefijo "ufw-" es porque se ha configurado alguna regla usando la herramienta ufw. Para desactivar ufw ejecuta la orden:* ***`sudo ufw reset`*** *y reinicia el sistema.*

Esta situación inicial nos muestra que para cada una de las tres cadenas (INPUT, FORWARD y OUTPUT) la política por defecto a seguir es aceptar cualquier tipo de tráfico (**policy ACCEPT**) no importando su origen o destino (source, destination) ni el protocolo utilizado. Esta configuración, que podríamos denominar completamente abierta, no pone restricciones sobre

el tráfico de entrada (INPUT) o de salida (OUTPUT) ni tampoco sobre el posible tráfico que "atravesara" nuestro ordenador si éste actuara como un *router* (FORWARD). Esta última posibilidad no está habilitada en nuestra máquina virtual que actúa como un simple host y no como router.

Ejercicio 1

1. Vamos a comenzar modificando la configuración inicial para comprobar sus efectos. Para ello emplearemos la opción **-P** (*policy*) tecleando la siguiente orden:

   ```
   sudo iptables -P OUTPUT DROP
   ```

2. A continuación, ejecuta la orden **ping 127.0.0.1** y observa qué sucede[1].

Lo que ocurre es que esta orden 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).

Ejercicio 2

1. Como esta primera experiencia no parece demasiado útil porque produce una desconexión total, vamos a devolver al equipo a su estado inicial. Teclea ahora:

   ```
   sudo iptables -P OUTPUT ACCEPT
   ```

2. Comprueba que de nuevo los servicios de red vuelven a funcionar. Ejecuta la orden **ping 127.0.0.1** y observa qué sucede ahora.

En la situación actual, que es la misma que al principio, todo el tráfico puede fluir sin restricción.

Cómo añadir y borrar reglas en las cadenas con la orden *iptables*

En esta práctica podemos hacer que nuestro ordenador rechace, de manera selectiva, determinado tipo de tráfico. Para ello vamos a necesitar reglas un poco más finas que la empleada en el primer ejercicio.

[1] La dirección "127.0.0.1" corresponde al interfaz localhost de IPv4 de nuestra máquina. Si empleamos en su lugar directamente "localhost", el paquete se enviaría por la interfaz loopback de IPv6, cuyas iptables son diferentes y se gestionan mediante otro comando (ip6tables).

Utilizando la orden *iptables*, se puede añadir o eliminar reglas a cada una de las cadenas de las distintas tablas. Podemos añadir una regla al final de una cadena con la opción **-A** (*Append*), al principio de la cadena con **-I** (*Insert*) y eliminarla con **-D** (*Delete*). Por supuesto, al añadir la regla hay que indicar cuál es la tabla (*filter*, *mangle*, *nat*,…) y la cadena (INPUT, OUTPUT,…) a la que deseamos añadir o borrar la regla. Si no indicamos la tabla, la tabla a la que se añadirá la regla será a la tabla *filter*.

También podemos borrar todas las reglas de una cadena con la opción **-F** (*Flush*).

Recuerda que las reglas se procesan de forma secuencial y una vez que la condición de una de las reglas se cumpla, se ejecuta la acción, y ya no se procesan más reglas de dicha cadena. 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).

Ejercicio 3

1. Vamos a bloquear el tráfico local, es decir, el que se produce sobre el dispositivo *lo* y para ello tecleamos la siguiente orden:

   ```
   sudo iptables -A INPUT -i lo -j DROP
   ```

2. Seguidamente verificamos si tiene el efecto deseado tecleando **ping 127.0.0.1** ¿obtenemos respuesta? Prueba a hacer **ping 158.42.4.23** ¿funciona?

3. Para poder restablecer el tráfico local basta con eliminar la regla anterior, pare ello tecleamos:

   ```
   sudo iptables -D INPUT -i lo -j DROP
   ```

   (recuerda que el parámetro -D significa eliminar la regla mientras que -A es para añadirla)

En este ejercicio hemos visto como mediante el parámetro **-i** podemos especificar una interfaz de red (puedes ejecutar "*ip addr"* para ver la lista de dispositivos de red y su configuración actual), y con **-j** podemos especificar qué hacer con el tráfico que coincida con esa regla.

Como sabes, el dispositivo "lo" no es en realidad una tarjeta de red sino la representación de las comunicaciones internas mediante la dirección de *loopback* 127.0.0.1

En el ejemplo hemos determinado que todo el tráfico que se reciba (INPUT) por el dispositivo "lo" tiene que descartarse (DROP).

Para que muchas reglas sean útiles no basta poder especificar el dispositivo, sino que es necesario poder afinar indicando qué protocolo y/o puertos definen una regla en particular.

Ejercicio 4

1. Comprueba que puedes acceder mediante SSH a zoltar.redes.upv.es y establece una conexión utilizando el usuario y contraseña que te indique el profesor:

   ssh *usuario*@zoltar.redes.upv.es

2. Abre otra terminal en tu ordenador, y en ella teclea:

   sudo iptables -A INPUT -p tcp --sport 22 -j DROP

3. Ahora vuelve a la ventana de tu conexión SSH con zoltar y teclea 123456 ¿qué sucede?, ¿por qué?
4. Vuelve a la segunda terminal, y ahora teclea:

   sudo iptables -D INPUT -p tcp --sport 22 -j DROP

   ¿Qué sucede ahora? ¿Aún funciona tu sesión ssh con zoltar?
5. Finaliza la sesión ssh y cierra esa terminal.

En el ejercicio anterior se bloqueaba el tráfico SSH entrante (cadena INPUT). Si has tardado más de un minuto entre los pasos 2, 3 y 4 del ejercicio es posible que la conexión SSH se haya interrumpido. En ese caso en conveniente que lo repitas intentando ser algo más rápido (lo que permitirá que la conexión no se interrumpa y obtengas un resultado diferente).

En este ejemplo hemos creado una regla que no especifica qué dispositivo de red sino qué protocolo (**-p tcp**) y también el puerto origen de los segmentos (**--sport 22**). En una conexión SSH a zoltar los paquetes que vienen de zoltar tienen como puerto origen el 22 que es el puerto en el que se ofrece el 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.

También es posible crear reglas que atiendan a las direcciones de los paquetes.

Ejercicio 5

1. Abre un navegador y carga la página www.upv.es
2. Ahora en una ventana de terminal teclea[2]:

```
sudo iptables -A OUTPUT -p tcp -d www.upv.es --dport 80 -j DROP
```

3. Visualiza el estado de la cadena con la orden **`sudo iptables -L`**
4. Intenta recargar la página en el navegador. ¿Qué sucede?
5. Prueba a visitar otra página, como por ejemplo www.ua.es ¿funciona?
6. Elimina la regla que has añadido en el punto 2 de este ejercicio.
7. Teclea **`sudo iptables -L`** para visualizar el estado actual de la lista de reglas.

Como puedes observar, en este ejercicio hemos creado una regla que descarta el tráfico TCP saliente destinado al servidor www.upv.es y destinado al puerto 80 (HTTP). Esta regla no afecta al tráfico similar enviado a cualquier otro servidor. Con esta regla sólo se impide el acceso al servidor web principal de la UPV. 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 durante toda la sesión de prácticas.

En ocasiones, también puede resultar conveniente emplear el operador de negación ! para modificar la interpretación de los valores especificados por algunos parámetros como: protocolo (-p), origen (-s) o destino (-d) del paquete, o el puerto al que va destinado (--dport) o del que proviene (--sport). Así, por ejemplo, **sudo iptables -A OUTPUT ! -d localhost** ..., creará una regla que se aplicará a cualquier paquete que se vaya a enviar y que no esté destinado a localhost. Versiones anteriores empleaban la convención "-x ! valor" en vez de la actual "! -x valor" por lo que no debe sorprenderte si encuentras esa otra notación. De momento ambas producen el mismo resultado.

[2] Actualmente muchos servidores webs utilizan el protocolo HTTPS (puerto 443) además del puerto 80. Es posible que en algunos ejercicios debas bloquear ambos puertos.

Ejercicio 6

1. Asegúrate mediante la orden **`sudo iptables -L`** que has eliminado la regla del ejercicio 5.
2. Añade una única regla que permita sólo el tráfico de salida destinado al puerto 80 con destino a www.upv.es y descarte el resto de tráfico destinado al puerto 80. Para poder especificar el número de puerto en una regla es necesario especificar también el protocolo de transporte mediante la opción -p. En nuestro caso, **-p tcp**. Abre un navegador e intenta recargar las páginas www.upv.es y www.redes.upv.es . Comprueba que la regla que has añadido funciona correctamente. No deberías de tener acceso al segundo servidor.
3. Añade otra regla para permitir el tráfico de salida del protocolo ssh destinado únicamente a zoltar.redes.upv.es
4. Ahora en una ventana de terminal intenta acceder mediante ssh a zoltar.redes.upv.es. Es suficiente con que obtengas respuesta del destino, no es necesario que completes el acceso. Después, desde otra ventana de terminal intenta acceder por ssh a cualquier otro host (por ejemplo 158.42.4.23). Comprueba que sigues teniendo acceso a la web de la upv. Con estos tres pasos podrás comprobar si la regla que has añadido funciona correctamente.
5. Elimina la regla referente al tráfico ssh con la orden -D.
6. Añade estas dos nuevas reglas:

```
sudo iptables -A OUTPUT -p tcp ! -d zoltar.redes.upv.es -j DROP
sudo iptables -A OUTPUT -p tcp -d www.upv.es --dport 80 -j ACCEPT
```

7. Intenta recargar la página www.upv.es en el navegador. ¿Funciona? Comprueba con la orden **`sudo iptables -L`** cuál es el contenido de la lista de reglas, e intenta explicar el significado de las tres reglas que aparecen. Piensa en cómo afecta el orden en el que están. ¿Cómo habría que modificarlas para poder acceder a www.upv.es?
8. Vamos a comprobarlo. Elimina la siguiente regla:

   ```
   sudo iptables -D OUTPUT -p tcp ! -d zoltar.redes.upv.es -j DROP.
   ```

   Ahora vamos a añadirla al final de la cadena:

   ```
   sudo iptables -A OUTPUT -p tcp ! -d zoltar.redes.upv.es -j DROP
   ```

9. Teclea **`sudo iptables -L`** para visualizar el estado actual de la lista de reglas.
10. Comprueba ahora que has recuperado el acceso al servidor www.upv.es. Como ves el orden en el que se procesan las reglas es realmente importante.
11. Ahora, elimina todas las reglas mediante la orden **`sudo iptables -F`**

Registro de sucesos

La funcionalidad de *iptables* no sólo permite especificar reglas para aceptar o descartar paquetes (como hemos visto en varios de los ejemplos). Además de estas funciones, las reglas pueden producir acciones de registro que se anotarán en el registro de sucesos del sistema (puedes ver este registro ejecutando la instrucción **`dmesg`**).

Nota: Para limpiar el registro de sucesos puedes ejecutar el comando:

```
sudo dmesg -C
```

Para crear reglas que generen una entrada en el registro cuando coincida con la regla de un paquete se usará la opción **-j LOG**. Las reglas cuya acción sea LOG no toman ninguna decisión sobre el datagrama que estén procesando, ni lo aceptan ni lo rechazan, por lo que cuando una regla de este tipo se active **no finaliza el procesamiento de las reglas de la cadena actual**.

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 SSH 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 al servidor.

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.

Ejercicio 7

1. Crea la regla que realizará el registro:

   ```
   sudo iptables -A INPUT -p tcp --dport 22 -j LOG
   ```

2. Ahora vamos a instalar un servidor SSH en nuestra máquina, para ello teclea:

   ```
   sudo apt install ssh
   ```

   y acepta la instalación. Para comprobar que el servidor está en marcha puedes ejecutar: **`systemctl status ssh.service`** y ver que escucha en el puerto 22.
3. Ahora vamos a acceder al servidor SSH local. Ejecuta **ssh 127.0.0.1**. No hace falta que inicies sesión en *ssh*, simplemente interrúmpelo con **Ctrl+C**
4. Ahora teclea **dmesg** y analiza las líneas del listado. Busca una donde aparezca el **flag SYN** activado. ¿Qué ves? ¿Entiendes lo que significa?

Con la regla del apartado 7.1 se produce el registro de cada paquete que llega a tu equipo y va destinado al servidor SSH. Sin embargo, existe un serio problema y es que esa regla de registro

se cumple para cada segmento de SSH recibido, por lo que un mismo cliente puede generar miles de entradas en una misma conexión. Llenar un archivo de registro con muchos datos poco significativos es una muy mala idea.

Para realizar un registro en condiciones 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, como ya sabemos, tiene activo el **flag SYN.**

Ejercicio 8

1. Comienza anulando la regla anterior:

   ```
   sudo iptables -D INPUT -p tcp --dport 22 -j LOG
   ```

2. Y ahora vamos a crear una regla de registro que realmente sí detecta las peticiones de conexión al puerto 22. Para ello tecleamos:

   ```
   sudo iptables -A INPUT -p tcp --dport 22 --tcp-flags ALL SYN -j LOG
   ```

3. Si ahora intentamos conectarnos al servidor de SSH observaremos con la orden **dmesg** que solo se registra un mensaje por intento de conexión

Puede parecer que hemos repetido lo mismo que el ejercicio 7 pero no es así. Ahora la regla de registro 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 por el puerto 22 que tengan el bit SYN activado.

Vamos a añadir ahora una regla en las *iptables* que nos permita registrar a qué servidores web se intentan conectar los usuarios de nuestra máquina. Para ello deberemos registrar los segmentos TCP de salida que vayan dirigidos a los puertos 80 y 443 y que tengan el *flag* SYN activado.

Ejercicio 9

1. Vamos a añadir una nueva regla que registre las conexiones de salida a cualquier servidor web, tecleamos (las dos líneas son una única orden):

   ```
   sudo iptables -A OUTPUT -p tcp --match multiport
     --dports 80,443 --tcp-flags ALL SYN -j LOG
   ```

   (fíjate que hemos cambiado a OUTPUT. Además, hemos utilizado la opción *multiport* para filtrar usando varios puertos destino). Esta regla registrará todas las conexiones a cualquier servidor HTTP o HTTPS efectuadas desde este ordenador.
2. Accede a varios servidores servidor web,
3. Comprueba mediante la orden **dmesg** que se ha registrado los accesos a las páginas web. Busca aquellas líneas con el valor DPT=80 o DPT=443. ¿Has obtenido más de un mensaje por cada servidor web? Si es así, ¿por qué crees que ocurre esto?
4. Comprueba mediante la orden **sudo iptables -L -v** la lista de reglas que está activa en este momento en tu sistema. Observa que en esta presentación la columna pkts te indica cuántas veces se ha cumplido cada regla. También puedes ver el número de paquetes y bytes recibidos (INPUT) y enviados (OUTPUT) y, si fuera el caso, los reenviados (FORWARD).
5. Elimina la regla que has añadido en este ejercicio.

Pero la funcionalidad de *iptables* no termina aquí. En los siguientes ejercicios opcionales os dejamos otras posibles funcionalidades de esta potente herramienta.

Ejercicios opcionales

Modificando direcciones y/o puertos de destino: la tabla nat

Aunque por motivos de espacio estamos evitando dar detalles de todas las posibilidades de iptables, sí queremos al menos ilustrar con ejemplos algunas de sus características. En el siguiente ejercicio vamos a crear una regla para que todos los segmentos TCP afectados cambien su dirección de destino.

Ejercicio 10

1. Abre el navegador y visita la página http://www.disca.upv.es
2. Teclea la siguiente regla (las dos líneas son una única orden):

   ```
   sudo iptables -t nat -A OUTPUT -p tcp -d 158.42.0.0/16 --dport 80 -j DNAT --to
   158.42.4.23:80
   ```

3. Vuelve a cargar la página web del apartado 1. El prefijo http:// es importante para usar el puerto 80. ¿Qué sucede?
4. Prueba a acceder a la web www.upv.es

Si repasamos la orden creada podemos ver que ahora 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 servidores web dentro del campus (subnet 158.42.0.0/16) serán redirigidos al servidor web de la dirección 158.42.4.23

Se puede observar en esta regla que se especifica una nueva tabla (-t nat) mientras que hasta ahora la tabla usada era la tabla por defecto (-t filter) que, por lo tanto, no era necesario detallar en la línea de órdenes. Esta nueva tabla permite realizar operaciones de traducción de puertos y direcciones (NATP) como las que realizan los routers proporcionados con las conexiones de cable o ADSL. Sin embargo, no detallamos en la práctica cómo construir el conjunto de reglas para obtener esa funcionalidad.

Ejercicio 11

1. Visualiza el conjunto de reglas actual de iptables mediante la orden -L ¿Puedes ver la regla que has añadido en el ejercicio 10? ¿A qué crees que se debe?
2. Prueba de nuevo con la orden **`sudo iptables -t nat -L`**
3. Elimina la/s regla/s de la tabla nat.

Por lo tanto, la orden -L que permite visualizar el contenido de una tabla, nos muestra por defecto, el contenido de la tabla filter. Si queremos ver el contenido de otra tabla hay que emplear la opción -t, como hemos hecho en el ejercicio anterior (-t nat).

Ejercicio 12

Ahora vamos a realizar un ejercicio un poco más complejo.

1. Piensa qué reglas necesitarías añadir si deseas que las conexiones destinadas a cualquier servidor web de la UPV, excepto las destinadas a www.upv.es, sean redirigidas a la dirección 193.145.235.30 (servidor web www.ua.es).
2. Comprueba que la redirección funciona correctamente conectándote:
   - Al servidor www.disca.upv.es y www.cfp.upv.es. Si tu redirección es correcta te aparecerá la página www.ua.es.
   - Al servidor www.upv.es. Aquí debería aparecerte la página habitual del servidor de redes.
3. Elimina la/s regla/s que has añadido mediante **`sudo iptables -t nat -F`**

Cambios en otros campos

Con diferentes propósitos es posible modificar los valores de otros campos en nuestro tráfico. Uno de los candidatos es el campo de tipo de servicio (TOS) de la cabecera IP. Personalizado para una determinada aplicación es posible adecuar el valor del campo a las necesidades de esta. Incluso aunque su valor no sea respetado más allá de nuestro sistema, puede ser interesante para que distintos flujos de tráfico simultáneo 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. Es sabido que el valor utilizado por nuestro ordenador viene prefijado en la

configuración del sistema operativo y, aunque se puede modificar, la mayoría de los administradores no tienen razones para hacerlo.

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 iptables que realice ese cambio selectivo.

Ejercicio 13

1. ¿Cómo sabes el valor del TTL con que envías tu tráfico?
2. Ahora añade la siguiente regla:

   ```
   sudo iptables -t mangle -A POSTROUTING -j TTL --ttl-set 5
   ```

3. Comprueba si puedes acceder a las redes: www.upv.es, www.etsit.upv.es y www.ucla.edu

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.