

Práctica 7: Cortafuegos IPTABLES

Como ya se ha explicado en clase, la pila de protocolos TCP/IP forma parte de la mayoría de sistemas operativos actuales, como GNU/Linux, OS X o Windows. No siempre fue así, ya que por ejemplo no formaba parte de MS-DOS o CP/M.

Puesto que el código del sistema operativo maneja las peticiones de comunicación de las aplicaciones resulta bastante natural plantear la posibilidad de poder ajustar el comportamiento para especificar qué políticas son más adecuadas según el criterio del administrador. Diferentes sistemas operativos incluyen distintos modos de especificar esas preferencias. En el caso de GNU/Linux esta herramienta es iptables que permite al administrador consultar y modificar las políticas de gestión del tráfico de red en cada equipo. En esta práctica veremos algunos de los usos básicos de esta potente herramienta.

Un conocimiento detallado de iptables requiere mucho más tiempo que el de esta práctica. En esencia esta herramienta es una interfaz que permite modificar el comportamiento del sistema operativo frente a diferentes tipos de tráfico. Puesto que la gestión de las comunicaciones se realiza en el núcleo del sistema operativo sólo el administrador tiene acceso a esta herramienta que puede cambiar en un momento la conectividad del sistema, haciéndolo completamente invisible a otros sistemas en la red o bien todo lo contrario, si así se desea. Por este motivo y ya que no disponemos de acceso de administrador en los equipos de prácticas utilizaremos la orden `sudo` precediendo siempre a la orden iptables.

La primera operación es comprobar cuál es el estado de nuestro puesto de trabajo. Para ello teclea `sudo iptables -L` y observa el resultado.

El resultado debería de ser similar a éste:

```
$ 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
```

Esta situación inicial nos muestra que para cada una de las tres cadenas (**INPUT**, **FORWARD** y **OUTPUT**) la política 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 los equipos de laboratorio, si bien todos disponen de dos conexiones de red y podrían, llegado el caso, trabajar como routers.

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 localhost` y observa qué sucede.

Que no deja conectarse "Operation not permitted"

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 -F OUTPUT ACCEPT`
2. Comprueba que de nuevo los servicios de red vuelven a funcionar. Ejecuta la orden `ping localhost` y observa qué sucede ahora.

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

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.

Políticas por defecto

Se ha mencionado con anterioridad que mediante la orden `sudo iptables -F ...` se puede cambiar la política de aceptar (ACCEPT) o descartar (DROP) determinados paquetes. Pero 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.

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.

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 como en el ejercicio 1, 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 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.

A lo largo de esta práctica continuaremos basándonos en una política por defecto basada en aceptar todo el tráfico que no esté explícitamente rechazado por otras reglas.

Reglas encadenadas

Conforme se van ejecutando órdenes de iptables, se van añadiendo o eliminando reglas asociadas a cada uno de los flujos de entrada o salida. Podemos añadir una regla al final de una cadena con la orden -A (Append), al principio de la cadena con la orden -I (Insert) y eliminarla con la orden -D (DELETE). Por supuesto, al añadir la regla hay que indicar cuál es la cadena (INPUT, OUTPUT o FORWARD) para la que deseamos añadir o borrar la regla. También podemos borrar todas las reglas de una cadena con la opción -F (FLUSH).

Es importante entender que las reglas que se añaden 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. 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 localhost`. ¿Obtenemos respuesta? Prueba a hacer `ping www.upv.es` ¿funciona?
3. Para poder restablecer el tráfico local basta con eliminar la regla anterior, para ello tecleamos: `sudo iptables -D INPUT -i lo -j DROP` (recuerda que el parámetro -D significa eliminar la regla mientras que -A la añadía)

podemos enviar pero no recibir

En este ejercicio hemos visto como mediante el parámetro `-i` podemos especificar un dispositivo de red (puedes ejecutar `ifconfig` 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.

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).

Pero 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 profesor¹.
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? *→ Que no nos muestra lo que escribimos porque no recibimos la entrada del SSH*
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. *Ahora recibimos todo lo que habíamos escrito*

En este ejemplo hemos creado una regla que no especifica qué dispositivo de red sino qué protocolo (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. En el ejercicio anterior se bloqueaba el tráfico SSH entrante (INPUT). Si has tardado 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 es conveniente que lo vuelvas a repetir intentando ser algo más rápido (lo que permitirá que la conexión no se interrumpa y obtengas un resultado diferente).

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 `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? *No funciona porque hemos bloqueado la salida*
5. Prueba a visitar otra página, como por ejemplo `www.ua.es` ¿funciona? *Si, al servidor de la UPV*
6. 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.

¹ Al conectarte a `zoltar` emplea el usuario `redesXX` (donde XX es el número de tu ordenador). Pero al conectarte desde `zoltar` a tu ordenador, el usuario será `redlocal`.

Ejercicio 6

1. Asegúrate mediante la orden `sudo iptables -L` que has eliminado la regla del ejercicio 5. *Sudo iptables -A OUTPUT -p tcp ! -d www.upv.es --dport 80 -j DROP*
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`.
3. 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.
4. Añade otra regla para permitir el tráfico de salida del protocolo `ssh` destinado únicamente a `zoltar.redes.upv.es`. *Sudo iptables -A OUTPUT -p tcp -d zoltar.redes.upv.es --dport 22 -j ACCEPT*
5. 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 al ordenador de algún compañero (`rdcXX.redes.upv.es`). 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.
6. Elimina la regla referente al tráfico `ssh` con la orden `-D`.
7. Añade estas dos nuevas reglas:
 - a. `sudo iptables -A OUTPUT -p tcp ! -d zoltar.redes.upv.es -j DROP`
 - b. `sudo iptables -A OUTPUT -p tcp -d www.upv.es --dport 80 -j ACCEPT`
8. 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`? *la regla a bloquee upv.es por tanto tenemos que cambiar el orden*
9. Vamos a comprobarlo. Elimina la primera regla del punto 7: `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`
10. Teclea `sudo iptables -L` para visualizar el estado actual de la lista de reglas.
11. 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.
12. Ahora, elimina todas las reglas mediante la orden `sudo iptables -F`.

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 por defecto de descartar paquetes (DROP) las reglas deberían ser para aceptar determinado tipo de tráfico. Pero además de estas funciones las reglas pueden producir acciones de registro que se anotarán en el registro de sucesos del sistema (en el archivo `/var/log/messages` en los equipos del laboratorio).

Para crear reglas que generen una entrada en el registro cuando coincida con la regla de un paquete 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 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 21 de nuestro servidor es un nuevo cliente.

Ejercicio 7

1. Crea la regla que realizará el registro:
`sudo iptables -A INPUT -p tcp --dport 21 -j LOG`
2. Ahora vamos a acceder al servidor FTP local. Teclea `ftp localhost` o bien dirige el navegador al url `ftp://localhost` ¿qué sucede? *No tenemos servicio ftp*
3. Es posible que ya te hayas dado cuenta que no hay un servidor FTP en tu ordenador. No importa. Ahora teclea `dmesg` y analiza la última línea del listado donde aparezca el flag SYN activado. ¿Qué ves? ¿Entiendes lo que significa? *Que se han registrado todas las conexiones.*

Con la regla del apartado 7.1 se produce el registro de cada paquete que llega a tu equipo y va destinado al servidor FTP (incluso aunque no tengas servidor FTP). Sin embargo, existe un serio problema que no resulta aparente de momento, gracias a que no dispones de servidor FTP local. El problema reside en que esa regla de registro se cumple para cada segmento FTP recibido de cada conexión. Eso quiere decir 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 21 -j LOG`
2. Y ahora vamos a crear una regla de registro que realmente sí detecta las peticiones de conexión al puerto 21. Para ello tecleamos:
`sudo iptables -A INPUT -p tcp --dport 21 --tcp-flags ALL SYN -j LOG`
3. Si ahora intentamos conectarnos al servidor de FTP (que no tenemos) observaremos con la orden `dmesg` un resultado similar al del ejercicio anterior.

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 que tengan el bit SYN activado.

Para que podamos comprobar que esta idea realmente funciona vamos a conectarnos a un servidor que si dispone de un servidor FTP, como es el ordenador `zoltar.redes.upv.es` y a pedir que se registre cada conexión que realicemos al mismo en el registro de sucesos.

Ejercicio 9

1. Vamos a añadir una nueva regla que registre las conexiones al servidor FTP de zoltar.redes.upv.es, tecleamos: `sudo iptables -A OUTPUT -p tcp --dport 21 --tcp-flags ALL SYN -j LOG` (fíjate que hemos cambiado a OUTPUT pero seguimos analizando el puerto de destino). Como no se especifica el servidor de destino esta regla registrará todas las conexiones a cualquier servidor FTP efectuadas desde este ordenador.
2. Ya sea desde una ventana de terminal o desde el navegador accede por FTP al servidor zoltar.redes.upv.es como usuario *anonymous*.
3. Comprueba mediante la orden `dmesg` que el acceso ha quedado registrado como una única línea en el registro. 
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. También es posible modificar alguno de los datos contenidos en los paquetes o sus cabeceras.

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 www.redes.upv.es
2. Teclea la siguiente regla: `sudo iptables -t nat -A OUTPUT -p tcp -d 158.42.0.0/16 --dport 80 -j DNAT --to 155.54.212.103:80`
3. Vuelve a cargar la página web del apartado 1. ¿Qué sucede?
4. Prueba a acceder a la web www.disca.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 redigidos al servidor web de la dirección 158.42.250.41

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 realiza uno de los routers que proporcionan 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 que 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.redes.upv.es, sean redirigidas a la dirección 155.54.212.103 (servidor web www.um.es).
2. Comprueba que la redirección funciona correctamente conectándote:
 - a. Al servidor www.upv.es y www.cfp.upv.es. Si tu redirección es correcta te aparecerá la página www.um.es.

- b. Al servidor www.redes.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 la misma. 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 administradores no tienen 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 (podemos usar la orden ping para determinarlo).

Aunque puede que sea una leyenda urbana, se dice que algunos ISP podrían analizar si un cliente en particular está cursando tráfico con uno o varios valores del TTL para suponer que el cliente podría tener varios ordenadores (con distintos sistemas operativos) compartiendo la conexión de red. Actualmente en España la mayoría de ISP no sólo no parecen tener nada en contra de esto sino que incluso regalan los routers NAT para que los clientes puedan fácilmente compartir su conexión entre varios ordenadores.

De todos modos, 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 webs: www.redes.upv.es , www.etsit.upv.es y www.uji.es

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.