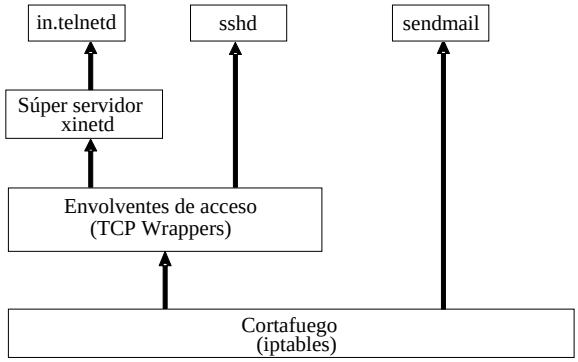
Práctica 8: Seguridad en Linux.

Autores: Enrique Bonet
Rogelio Montañana

1.- Introducción.

La arquitectura de seguridad de las actuales versiones de Linux está organizada en tres capas diferentes, como puede verse en la siguiente figura:



- La primera capa está formada por el cortafuegos, denominado **iptables**, el cual puede controlar el acceso a los niveles de subcapa de acceso al medio, capa de red, capa de transporte y en algún caso incluso la capa de aplicación.

- La segunda capa esta formada por los envolventes de acceso o **TCP Wrappers**, que controlan los permisos para acceder a un determinado servicio según el ordenador remoto que lo solicita.

- Por último, la tercera y última capa de seguridad se encuentra en el súper-servidor **xinetd**, cuyo funcionamiento es similar al de los envolventes de acceso pero con un nivel de configuración mucho más detallado.

Como puede observarse en la figura no todos los servicios son controlados por todas las capas de seguridad. Así en la configuración mostrada como ejemplo en la figura el programa sendmail (correo electrónico) se encuentra solo bajo el control del cortafuegos, el sshd (Secure shell Daemon) es controlado por el cortafuegos y por los TCP Wrappers, de forma que ambas capas de seguridad deben permitir el acceso a un ordenador remoto para que pueda hacer uso del servicio. Por último el telnet (TELetype NETwork) está controlado por las tres capas de seguridad, siendo necesario que las tres permitan el acceso para que un ordenador remoto pueda hacer uso del servicio. Esta organización de los servicios y su control es solo un ejemplo, en otros sistemas se pueden haber configurado de diferente manera los servicios, por ejemplo poniendo el sendmail bajo el control de los TCP Wrappers.

Las tres utilidades descritas se aplican de manera consecutiva y aditiva, de forma que cuando un paquete llega por la interfaz de red a un host en primer lugar lo analiza el **iptables** de acuerdo con las reglas que tiene definidas y decide si el paquete tiene o no permitida la entrada. En caso negativo el paquete es descartado sin más, independientemente de lo que

pudieran determinar las reglas de los TCP Wrappers o el xinetd. Si el paquete pasa el filtro del iptables los **TCP Wrappers** se encargarán de inspeccionar si está permitido el acceso, de acuerdo con las reglas configuradas; en caso negativo el acceso será denegado, independientemente de lo que dispongan las reglas de xinetd. Por último, si el paquete ha superado tanto los filtros del iptables como los del TCP Wrappers el **xinetd** analizará si está permitido el acceso solicitado. Solo cuando las tres utilidades hayan permitido el acceso podrá el paquete llegar finalmente al proceso servidor el que va dirigido y generar la acción correspondiente, que podría ser por ejemplo el establecimiento de una conexión TCP con dicho proceso. Por supuesto aquellos servicios que no son controlados por las tres capas (como en el caso de sendmail y sshd del ejemplo anterior) solo tendrán que superar el control establecido por las utilidades que les afectan.

En esta práctica realizaremos diversas pruebas que nos permitirán familiarizarnos con las posibilidades de control que nos ofrecen cada una de las tres herramientas mencionadas. Los apéndices A, B y C contienen una explicación detallada de los comandos y ficheros de configuración que se utilizarán a lo largo de la práctica.

Para realizar esta práctica es aconsejable formar parejas de ordenadores, de forma que esos dos ordenadores corresponderán a lo que llamaremos a lo largo de la práctica la "red privada". Cada alumno deberá conocer en todo momento su dirección IP y la del ordenador de su compañero en la red privada. Debemos además tener en cuenta que cada ordenador tiene dos interfaces, la Ethernet (eth0) con una dirección 147.156.xxx.yyy, y la interfaz virtual loopback (l0), con la dirección 127.0.0.1. Si establecemos una determinada regla que impida por ejemplo conexiones Telnet a la dirección 147.156.xxx.yyy no podremos hacer *'telnet 147.156.xxx.yyy'*, pero sí podremos hacer *'telnet 127.0.0.1'* o bien *'telnet localhost'*, que es equivalente.

2.- Restauración de la configuración inicial del host

Para probar las herramientas debemos partir de una configuración inicial 'neutra' evitando los problemas que nos podría producir los residuos de configuraciones dejadas en el host pro compañeros de prácticas anteriores; para ello vamos en primer lugar a borrar todos los ficheros de configuración ejecutando los siguientes comandos:

```
rm -f /etc/sysconfig/iptables
rm -f /etc/hosts.allow
rm -f /etc/hosts.deny
rm -f /etc/xinetd.conf
rm -f -r /etc/xinetd.d
```

A continuación obtendremos el fichero *Redes.tar* del URL siguiente:

http://informatica.uv.es/guia/asignatu/R/apuntes/laboratorio/programas/Redes.tar

y ejecutaremos el comando:

```
tar xvpPf Redes.tar
```

Una vez ejecutado comprobaremos que los ficheros que antes hemos borrado existen de nuevo.

3.- Configuración del servidor xinetd.

xinetd (eXtended InterNET Daemon) es un proceso servidor también llamado 'super-servidor' porque se encarga de poner en marcha diversos servicios en un host cuando son requeridos por los clientes, reduciendo de esta forma la cantidad de procesos activos cuando lo que se quiere es ofrecer servicios cuya utilización va a ser esporádica. Así por ejemplo si queremos ofrecer un servicio HTTP no necesitamos al arrancar el host poner en marcha un proceso específico que esté escuchando en el puerto 80 por si en algún momento se conecta algún cliente. En su lugar el proceso *xinetd* escuchará el puerto 80 y en el momento que reciba alguna petición de conexión al puerto 80 arrancará el proceso HTTP servidor; cuando la conexión TCP al servidor se termine el servidor HTTP desaparecerá, repitiéndose el proceso cuando vuelva a aparecer una petición de conexión al puerto 80 por parte de algún cliente. Algunos servicios sencillos, como el daytime, pueden ser incluso atendidos directamente por el propio proceso *xinetd*, sin necesidad de lanzar ningún proceso independiente. Estos se denominan servicios 'internos'.

Además de reducir la cantidad de recursos utilizados para ofrecer servicios en un host el super-servidor xinetd permite establecer un control en el acceso mediante unos ficheros de configuración específicos de cada servicio. Estos ficheros se encuentran normalmente en el directorio */etc/xinetd.d* y son los que utilizaremos en la primera parte de esta práctica.

En determinadas versiones de Linux y UNIX en lugar del *xinetd* se utiliza el *inetd*. Aunque también actúa como super-servidor, el *inetd* no incorpora las funciones de control de acceso que tiene *xinetd*, con lo que en ese caso este nivel de seguridad no está disponible

Como ya hemos comentado no es obligatorio que todos los servicios de un host estén controlados por *xinetd*. Por ejemplo en nuestro caso el *sshd* no está controlado por el *xinetd*.

Antes de configurar el super-servidor xinetd desactiva el cortafuegos mediante el comando:

```
service iptables stop
```

De esta forma evitarás posibles interferencias de acceso debidas a las reglas contenidas en este componente.

Como hemos dicho los servicios controlados por el servidor *xinetd* se configuran mediante un conjunto de ficheros situados en el directorio */etc/xinetd.d*. Dicho directorio es leído al arrancar el servidor *xinetd*, por lo que cada vez que realices un cambio en un fichero del directorio debes reiniciar el servidor *xinetd* ejecutando el comando:

```
service xinetd restart
```

Trasládate ahora al directorio */etc/xinetd.d* ejecutando el comando:

```
cd /etc/xinetd.d
```

verás que hay allí un conjunto de ficheros cuyos nombres corresponden a los servicios que son arrancados y controlados por xinetd.

En primer lugar vamos a ver si está habilitado el servicio *telnet*. La configuración del *telnet* se encuentra en el fichero *telnet* en el directorio */etc/xinetd.d*. Puedes ver su contenido mediante el comando:

```
more telnet
```

fíjate en el valor del parámetro '*disable*'. Si pone '*disable = no*' el telnet está habilitado y no has de hacer ningún cambio, si pone '*disable = yes*' deberás modificarlo y poner '*disable = no*' editando el fichero.

En caso de que hayas modificado el valor de 'disable' deberás reiniciar antes el *xinetd* para que los cambios surtan efecto ejecutando el comando:

```
service xinetd restart
```

A partir de aquí, aunque el guión no lo indique, deberás reiniciar el *xinetd* cada vez que cambies algún fichero del directorio *xinetd.d* para que las modificaciones tengan efecto.

Nota importante: para saber en un momento determinado qué puertos tienes abiertos te recuerdo que puedes usar: nmap localhost "que te mostrará los puertos TCP" nmap –sU localhost "que te mostrará los puertos UDP"

Prueba ahora a utilizar el comando '*telnet <dirección IP>*' donde <*dirección IP*> será la dirección IP de la interfaz eth0 de tu propio ordenador y comprueba si puedes acceder al servicio.

Pasemos ahora a ver las posibilidades de control de acceso que nos ofrece el *xinetd*. En primer lugar añade en el fichero *telnet* la siguiente línea:

```
only_from = <dirección IP>
```

Donde <*dirección IP*> es la dirección IP de la interfaz eth0 de tu ordenador. Ejecuta otra vez la conexión al servicio *telnet* de tu ordenador mediante el comando '*telnet <dirección IP>.*'

¿Qué respuesta obtienes?. ¿Y si intentas conectarte por *telnet* a tu dirección loopback, 127.0.0.1?. Explica lo que ocurre.

Añade ahora, a continuación de la línea anterior, la dirección IP de tu interfaz loopback mediante la siguiente línea:

```
only_from += 127.0.0.1
```

¿Qué sucede ahora si te conectas por *telnet* a tu dirección loopback?

El parámetro *only_from* permite utilizar no solo direcciones aisladas sino también rangos con la notación dirección/máscara (ej.: *147.156.10.0/24*), nombres de hosts (ej.: *marcello.uv.es*) o dominios (ej.: *.uv.es*, que abarcaría todos los ordenadores de la Universidad de Valencia)

Ahora edita el fichero *telnet*, quita la línea del *only_from* con la dirección loopback **y a**ñade la línea siguiente:

```
banner_fail = /tmp/mensaje.txt
```

donde */tmp/mensaje.txt* será un fichero de texto que debes crear y que contendrá un mensaje similar al siguiente:

```
Tu ordenador no esta autorizado a usar este servicio.
```

Intenta ahora acceder por *telnet* a tu dirección Ethernet y a tu dirección loopback. ¿Qué sucede en cada caso?. Explícalo.

Abre el wireshark, y captura y analiza los paquetes de entrada y salida de tu ordenador cuando un ordenador externo al tuyo intenta hacer un telnet al tuyo.

Comprueba ahora la hora de tu ordenador, y suponiendo que sean, por ejemplo, las 9:20, añade en el fichero *telnet* la siguiente línea:

```
access_times = 09:20-09:21
```

Conéctate al servicio *telnet* y comprueba que funciona. Deja establecida en una ventana una conexión, espera a que la hora del ordenador sea mayor que la hora indicada como fin del rango (a partir de las 9:22 en este ejemplo) e intenta una nueva conexión. ¿Se establece la nueva conexión?. ¿Qué ocurre con la conexión que se había establecido previamente, una vez superada la ventana marcada en *'access_times'*? ¿Se corta o se mantiene establecida?

Añade ahora las siguientes líneas:

```
per_source = 2
instances = 3
```

Ejecuta ahora desde varias ventanas de tu ordenador, y de forma simultánea, el comando:

```
telnet <dirección_IP>
```

donde <*dirección_IP*> es la dirección IP de la interfaz Ethernet de tu ordenador.

¿Cuántas conexiones simultáneas puedes abrir como máximo?.

Sin cerrar las conexiones anteriores, intenta ahora abrir nuevas conexiones a la dirección loopback. ¿Cuantas conexiones nuevas puedes abrir? A la vista del comportamiento observado explica el significado de los parámetros *'per_source'* e *'instances'*.

4.- Configuración de los TCP Wrappers

Los TCP Wrappers o envolventes de acceso (wrapper = envoltura) son un sistema de control que se usa para filtrar el acceso por red a servicios en hosts con sistemas operativos tipo UNIX (Linux o BSD por ejemplo). A pesar de su nombre los TCP Wrappers se utilizan no solo para tráfico TCP sino también UDP e incluso ICMP pues se puede restringir por ejemplo el acceso al servicio pingd (ping daemon) que es el encargado de responder a los pings recibidos por el host.

La configuración de los TCP Wrappers se guarda en dos ficheros, */etc/hosts.allow* y */etc/hosts.deny* que especifican respectivamente los permisos y las denegaciones que se aplican a los diferentes servicios. Puedes ver el contenido de estos ficheros, por ejemplo mediante los comandos:

```
more /etc/hosts.allow
more /etc/hosts.deny
```

De momento estos ficheros tienen todas sus líneas comentadas, es decir no contienen ninguna regla (una línea está comentada si comienza por el símbolo “#”).

Los TCP Wrappers funcionan con la filosofía de ‘permiso por defecto’, es decir si algo no esta expresamente prohibido está implícitamente permitido. Además los TCP Wrappers procesan todas las reglas del hosts.allow por el orden en que aparecen y luego, también por orden, las del hosts.deny. Si el acceso solicitado cumple una regla se le aplica lo que corresponda (permitir si la regla está en el hosts.allow y denegar si está en el hosts.deny) y ya no se analiza nada más. Así por ejemplo, si en el hosts.allow ponemos la siguiente regla:

```
ALL : ALL
```

Ya dará igual lo que pongamos detrás de ella y todo lo que pongamos en el hosts.deny, pues cualquier acceso cumplirá esa regla, será permitido y no se evaluarán las reglas que haya detrás o en el hosts.deny..

Normalmente los TCP Wrappers controlan todos los servicios que controla xinetd y alguno más. Así por ejemplo en nuestro caso el servicio sshd (Secure Shell Daemon) que escapaba al control de xinetd, sí es controlado por los TCP Wrappers.

Como algunas de las pruebas de los TCP Wrappers las haremos con el sshd vamos en primer lugar a arrancar este servicio, ya que como hemos comentado no es controlado en nuestro caso por el xinetd. Para ello, ejecutaremos el comando:

```
service sshd start
```

Una vez arrancado, comprueba que funciona correctamente ejecutando el comando:

```
ssh <dirección_IP>
```

donde *<dirección_IP>* puede ser cualquiera de las direcciones IP de tu ordenador (eth0 o loopback).

También harás pruebas con el servicio *telnet*, por lo que debes deshacer algunos cambios realizados en el telnet en la parte de xinetd. Para ello es suficiente con que suprimas o comentes en el fichero telnet del xinetd las líneas que hubiera con los parámetros *'only_from'* o *'access_times'*, el resto no es necesario quitarlo.

Comprueba ahora que el Telnet funciona correctamente intentando una conexión a tu ordenador (eth0 o loopback).

Antes hemos comprobado que los TCP Wrappers no tenían ninguna regla configurada, por lo que no aplicaban ninguna restricción, pues como ya hemos dicho funcionan bajo el principio de "*permiso por defecto*", esto es, si no hay regla se permite el acceso.

Ahora escribe en el fichero */etc/hosts.deny* la siguiente línea:

```
ALL : ALL
```

Intenta acceder ahora al ssh o telnet, ¿Qué sucede?.

Escribe ahora en el fichero */etc/hosts.allow* la siguiente línea:

```
in.telnetd : ALL
```

¿Qué sucede ahora si intentas utilizar el servicio telnet?. ("*in.telnetd*" es el nombre del fichero ejecutable que corresponde al servidor Telnet, ver apéndice B).

Comprueba que no existe el fichero */tmp/telnet.log* y en caso de que exista bórralo. Modifica ahora la línea del fichero */etc/hosts.allow* para que ponga lo siguiente:

```
In.telnetd : ALL : spawn (/bin/echo `date` %c >> /tmp/telnet.log) &
```

(ojo a las comillas de *'date'* y al espacio que hay detrás de *spawn*).

¿Qué sucede ahora cuando realizas una conexión telnet a tu ordenador?. ¿Qué indican las líneas del fichero */tmp/telnet.log*?.

Cambia el fichero hosts.allow poniendo:

```
in.telnetd : IP_de_tu_ordenador
```

Abre el wireshark, y captura y analiza los paquetes de entrada y salida de tu ordenador cuando otro ordenador intenta hacer un telnet al tuyo.

Por último escribe en el fichero */etc/hosts.allow* la línea:

```
sshd : <IP ordenador>
```

Donde <*IP ordenador*> corresponde a la dirección IP de la interfaz Ethernet de tu ordenador.

¿Qué sucede ahora si realizas una conexión ssh a la eth0 de tu propio ordenador? ¿Y si lo intentas a la dirección loopback?.

Comprueba que no existe el fichero */tmp/denegado.log* y en caso de que exista bórralo. Modifica la línea del fichero */etc/hosts.deny* para que ponga lo siguiente:

```
ALL : ALL : spawn (/bin/echo `date` %d %c >> /tmp/denegado.log) &
```

Intenta ahora acceder de nuevo por ssh a la dirección loopback. ¿Qué información aparece en el fichero */tmp/denegado.log*?.

El fichero denegado.log registra los intentos fallidos de conexión a cualquier servicio activo en el host, pero para que el intento se registre es preciso que el servicio esté activo en el host (ssh o Telnet en nuestro caso), si se intenta acceder a un servicio inactivo o inexistente los TCP Wrappers no lo registran.

Una vez terminadas las pruebas restaura la configuración del TCP Wrappers a su estado inicial, comentando o borrando todas las reglas de los ficheros */etc/hosts.allow* y */etc/hosts.deny*.

5.- Configuración del cortafuegos.

El cortafuegos de Linux se configura mediante el fichero */etc/sysconfig/iptables*.

Vamos a proceder a continuación a configurar y activar el cortafuegos. En primer lugar escribe las siguientes líneas, en lugar de las que hubiera, en el fichero */etc/sysconfig/iptables. (*Es muy importante que estas líneas se escriban correctamente, pues en caso contrario pueden dejar de funcionar las ventanas gráficas que funcionan sobre el servidor X. Para mayor comodidad puedes usar copy/paste):

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -j REJECT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -j REJECT
COMMIT
```

Ahora activa el cortafuegos ejecutando el comando:

```
service iptables start
```

Ejecuta ahora algún comando de red, por ejemplo un *ping* a tu interfaz Ethernet (dirección 147.156.xxx.yyy) y a la interfaz loopback (dirección 127.0.0.1). ¿Qué sucede?.¿Puedes explicar el comportamiento?

Puedes comprobar que reglas tiene definidas el cortafuegos y el número de veces que cada una de ellas se ha aplicado con el comando:

```
iptables -L -v
```

Lanza otra vez el ping de antes y vuelve a ejecutar el comando anterior. Observa cómo cambian los contadores de paquetes.

Añade ahora las dos líneas marcadas en negrita en la configuración siguiente, de forma que el fichero quede así:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -j REJECT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -j REJECT
COMMIT
```

¿Qué sucede ahora si ejecutas el “ping” a la Ethernet? ¿Y si haces ping a otro ordenador, por ejemplo a 147.156.1.1 (gong.uv.es)? ¿Qué sucede si ejecutas “ping gong.uv.es” en vez de a la dirección IP?. Comprueba el valor de los contadores de las reglas y explica las anteriores respuestas.

Modifica ahora el fichero de configuración añadiendo las dos líneas marcadas en negrita:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p udp --sport 53 --dport 1024: -j ACCEPT
-A INPUT -j REJECT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p udp --sport 1024: --dport 53 -j ACCEPT
-A OUTPUT -j REJECT
COMMIT
```

Ejecuta ahora el comando “ping gong.uv.es” y explica que sucede. Prueba ahora a hacer “ssh lab3inf005.uv.es” (el ordenador del profesor) ¿Qué sucede? Explícalo.

Añade ahora la línea marcada en negrita:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p udp --sport 53 --dport 1024: -j ACCEPT
```

```
-A INPUT -p tcp ! --syn -j ACCEPT
-A INPUT -j REJECT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p udp --sport 1024: --dport 53 -j ACCEPT
-A OUTPUT -p tcp -j ACCEPT
-A OUTPUT -j REJECT
COMMIT
```

Repite ahora el "ssh lab3inf005.uv.es". Explica el resultado. Ejecuta ahora el ssh hacia tu propio ordenador. ¿Funciona?. ¿Por qué?.

Añade ahora la línea marcada en negrita:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -p udp --sport 53 --dport 1024: -j ACCEPT
-A INPUT -p tcp ! --syn -j ACCEPT
-A INPUT -p tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p udp --sport 1024: --dport 53 -j ACCEPT
-A OUTPUT -p tcp -j ACCEPT
-A OUTPUT -j REJECT
COMMIT
```

¿Puedes ejecutar ahora el ssh hacia tu propio ordenador?. ¿Y el telnet?. Explica las respuestas.

Abre el wireshark, y captura y analiza los paquetes de entrada y salida de tu ordenador cuando otro ordenador intenta hacer un telnet al tuyo.

El conjunto de reglas del iptables actúa como las ACLs de los routers. Podemos considerar que el conjunto de reglas INPUT es una ACL en sentido entrante y el conjunto de reglas OUTPUT es una ACL en sentido saliente. De la misma forma que ocurre en los routers el orden de las reglas es crucial, siendo lo normal poner en primer lugar las más específicas y en último las más genéricas.

Muchas funciones de control se pueden ejercer en cualquiera de las tres herramientas que hemos visto en esta práctica. La elección de una u otra depende de la comodidad para el usuario o de con que herramienta se sienta él más seguro. Sin embargo debemos destacar dos diferencias importantes que tiene el iptables respecto al xinetd y los TCP Wrappers:

- Iptables actúa a nivel de paquetes, sin ser consciente de la conexión TCP como tal. Esto significa que si por ejemplo establecemos una regla para impedir las conexiones entrantes a un servidor telnet con iptables el efecto será inmediato, bloqueando las conexiones telnet en curso en el momento que activamos el nuevo fichero de configuración. En cambio con xinetd o TCP Wrappers no se dejarán entrar nuevos clientes, pero los que ya están dentro podrán seguir trabajando sin limitaciones.

- Iptables es un cortafuegos y como tal está diseñado para controlar el tráfico entrante y saliente, tanto de procesos clientes como servidores. En cambio xinetd y TCP Wrappers están pensados para el control de servidores, por lo que solo pueden controlar el tráfico entrante a procesos servidores.

Una vez terminada la práctica debes apagar el ordenador ordenadamente. Es decir, no de olvides de devolver borrar todos los ficheros que hayas modificado, restaurando los originales eque has guardado como .cop

Apéndice A: Configuración del servidor xinetd.

La configuración por defecto de los servicios controlados por *xinetd* se introduce en el fichero */etc/xinetd.conf*. Las entradas del fichero son de la forma:

```
service <nombre del servicio>
{
<atributo> <operador> <valor>...
...
}
```

donde <*nombre del servicio*> identifica el nombre del servicio que configura esta entrada (ftp, telnet, etc.), <*atributo*> especifica el atributo que se esta configurando, <*operador*> puede ser '=', '+=' y '-=', que asignan, añaden y eliminan valores del atributo respectivamente[1], y <*valor*> es el valor dado al atributo. Los atributos, así como los valores que pueden tomar y una breve descripción de los mismos se encuentran a continuación:

Atributo	**Descripción**
Id	Identifica de forma univoca el servicio. Este atributo tiene por defecto el nombre del servicio y, de forma general, solo es necesario cuando un mismo servicio posee diferentes protocolos y necesita ser descrito con diferentes entradas en el fichero de configuración.
Type	Identifica el tipo de servicio y puede ser una combinación de los valores RPC (servicio RPC), INTERNAL (servicio proporcionado por el propio *xinetd*), TCPMUX/TCPMUXPLUS (servicio que debe ser arrancado de acuerdo al protocolo descrito en el RFC 1078 en un puerto TCPMUX bien conocido) y UNLISTED (servicio no listado en los ficheros estándar de servicios del ordenador).
Flags	Identifica el modo de funcionamiento del servicio y es una combinación de los valores INTERCEPT (interceptar los paquetes o aceptar conexiones para verificar que provienen de ordenadores validos), NORETRY (no reintentar en caso de que falle la llamada a la creación de un proceso hijo mediante *fork*), IDONLY (aceptar conexiones solo cuando el ordenador remoto identifique al usuario remoto), NAMEINARGS (colocar el nombre del servicio como argumento primero en la llamada al servidor, tal y como sucede con *inetd*), NODELAY (permite que si el servicio es de tipo TCP, pueda configurarse la opción TCP_NODELAY en el socket), KEEPALIVE (permite que si el servicio es de tipo TCP, pueda configurarse la opción SO_KEEPALIVE en el socket), NOLIBWRAP (desactiva la llamada interna al TCPWrapper, con lo cual la llamada debe ser realizada de forma explícita como sucedía con *inetd*), SENSOR (reemplaza el servicio con un sensor que detecta los accesos al puerto especificado), IPv4 (especifica que el servicio es de tipo IPv4, esto es, AF_INET) y por último IPv6 (especifica que el servicio es de tipo IPv6, esto es, AF_INET6).
Disable	Es un valor booleano ("yes" o "no") que indica si el servicio esta habilitado o deshabilitado.
socket_type	Especifica el tipo de socket, sus valores son *stream*, *dgram*, *raw* y *seqpacket* (secuencia de datagramas).
protocol	Especifica el protocolo que emplea el servicio. Sus valores posibles son cualquier protocolo de transporte especificado en */etc/protocols*.
wait	Sus valores posibles son "yes" o "no" e indica si *xinetd* debe esperar la finalización del servidor de ese servicio antes de lanzar otro servidor (valor "yes") o no (valor "no").
user	Determina el UID con el que se ejecutara el proceso. Dicho UID debe existir en el fichero */etc/password*.
group	Determina el GID del proceso servidor. Si el GID no existe se utiliza el GID del usuario.
instances	Indica el número de servidores que pueden estar activos simultáneamente. El valor por defecto es sin límite.
nice	Determina la prioridad con la que se ejecuta el servidor.
server	Indica el nombre del programa que ejecuta este servicio.

[1] La mayoría de atributos solo admiten el operador '='.

Atributo	Descripción
server_args	Determina los argumentos que se pasaran al servidor.
only_from	Indica que ordenadores están autorizados a ejecutar este servicio en particular. Las formas más comunes de especificación son mediante una dirección IP concreta (147.156.222.65), un rango de direcciones IP especificado en formato dirección/rango de la mascara (147.156.222.0/23), el nombre de un ordenador (*glup.irobot.uv.es*) o el nombre de un dominio (*.irobot.uv.es*).
no_access	Determina que ordenadores no están autorizados a ejecutar este servicio en particular. El formato de especificación es igual al de *only_from*.
access_times	Indica el intervalo de horas en que el servicio esta disponible. El formato es hh:mm-hh:mm, donde hh va de 0 a 23 y mm de 0 a 59[2].
log_type	Determina el tipo de log que utiliza el servicio, existen dos formas, SYSLOG y FILE. SYSLOG tiene la sintaxis SYSLOG syslog_facility [syslog_level] y especifica que el log será enviado al fichero de log del sistema con la facilidad especificada por syslog_ facility (daemon, auth, authpriv, user, mail, lpr, new, uucp, ftp, local0-7) y el nivel especificado por syslog_level (emerg, alert, crit, err, warning, notice, info, debug), si el nivel no esta presente se asume info. Por su parte FILE tiene la sintaxis FILE file [soft_limit [hard_limit]] e indica que la salida será grabada en el fichero especificado por file, teniendo dicho fichero un limite soft y hard de forma similar a como sucede con los limites soft y hard en las cuotas de los usuarios.
log_on_success	Indica la información que será almacenada en el log cuando el servidor empieza y cuando termina. Puede ser cualquier combinación de los valores PID (identificador del proceso servidor), HOST (dirección del ordenador remoto), USERID (identificador del usuario), EXIT (código de terminación del servidor) y DURATION (duración del servicio).
log_on_failure	Indica la información que será almacenada cuando la petición es rechazada. Sus valores son una combinación de HOST (dirección del ordenador remoto), USERID (identificador del usuario) y ATTEMPT (guarda que ha sucedido un fallo, esta opción esta implícita en las dos anteriores).
rpc_versión	Determina la versión de RPC para un servicio RPC. La versión puede ser un número o un rango en el formato número-número.
rpc_number	Determina el número para un UNLISTED RPC.
env	Indica una lista de strings en formato 'nombre=valor'. Esos strings serán añadidos a las variables de ambiente antes de arrancar el servidor.
passenv	Determina la lista de las variables de ambiente de *xinetd* que deben ser pasadas al proceso servidor.
port	Determina el puerto del servicio[3].
redirect	Permite a los servicios TCP ser redirigidos a otro ordenador. Cuando *xinetd* recibe una conexión TCP a ese puerto, establece una conexión con el ordenador y puerto especificado y envía todos los datos entre los dos ordenadores. La sintaxis es *redirect = <dirección IP> <puerto>*[4].
bind	Permite al servicio ser asignado a una determinada dirección IP o interface específico del ordenador, esto permite, por ejemplo, que el servicio este disponible para un interface de la intranet y no para el interface que da acceso a Internet, su sintaxis es *bind = <dirección IP o interface>*.
interface	Es un sinónimo de *bind*.
banner	Indica el nombre de un fichero que será mostrado en el ordenador remoto cuando una conexión a este servicio es solicitada.
banner_success	Indica el nombre de un fichero que será mostrado en el ordenador remoto cuando una conexión a este servicio es aceptada.
banner_fail	Indica el nombre de un fichero que será mostrado en el ordenador remoto cuanto una conexión a este servicio es rechazada.

[2] Toda petición del servicio es aceptada en ese intervalo de tiempo, no limitando en cualquier caso la duración del servicio, que puede exceder de dicho rango.
[3] Si este atributo es especificado para un servicio listado en el fichero /etc/services, debe ser igual al valor indicado en ese fichero.
[4] La dirección IP puede ser sustituida por el nombre del ordenador, en cuyo caso xinetd en el arranque determina la dirección IP de ese ordenador mediante el DNS.

Atributo	Descripción
per_source	Especifica el número de instancias permitidas de este servicio por dirección IP. Su valor es un entero o UNLIMITED si no se desea limitarlo.
cps	Especifica el número máximo de conexiones por segundo que pueden ser recibidas por este servicio. Sus argumentos son dos enteros, el primero indica el número máximo de conexiones que pueden ser recibidas y el segundo el intervalo en segundos en que el servicio estará deshabilitado si se sobrepasa el valor anterior.
max_load	Es un número en coma flotante que indica la carga (porcentaje de CPU) máxima para este servicio. En caso de que dicho valor se sobrepase el servicio dejará de aceptar conexiones.
groups	Puede tomar los valores "yes" o "no" e indica si el servidor es ejecutado con los permisos de los grupos a los que el UID del servidor tiene acceso o no.
umask	Especifica la máscara del servicio en formato octal. La máscara por defecto es 022.
enabled	Toma como argumento la lista de los *id* que deben ser habilitados. Aquellos que no se encuentren en esta lista serán deshabilitados.
include	Indica el nombre de un fichero que será tomado como nuevo fichero de configuración.
includedir	Indica el nombre de un directorio cuyos ficheros serán añadidos como configuración de *xinetd*. De estos ficheros se excluye todos aquellos que contienen un punto en su nombre o terminan con una tilde (~).
rlimit_as	Determina el limite de memoria del servicio, el límite se especificá como un entero seguido de K (kilobytes) o M (megabytes) o UNLIMITED para indicar que no existe límite.
rlimit_cpu	Especificá el máximo número de segundos que el servidor puede utilizar de CPU. El límite se especificá como un entero o UNLIMITED si no existe.
rlimit_data	Especificá el tamaño máximo de los datos que el servidor puede utilizar. El límite se especificá como un entero indicando los bytes o UNLIMITED si no existe.
rlimit_rss	Especificá el tamaño máximo del programa que debe permanecer residente. Un tamaño pequeño hace a este servicio candidato a ser volcado a disco cuando la cantidad de memoria disponible es baja.. El tamaño se especificá como un entero que indica el número de bytes o UNLIMITED si no existe.
rlimit_stack	Especificá el tamaño máximo de la pila que el servidor puede utilizar. El límite se especifica como un entero indicando los bytes o UNLIMITED si no existe.
deny_time	Especificá el tiempo de denegación de acceso a todos los servicios para una IP que ha sido indicada por el *flag* de SENSOR. Los valores posibles son FOREVER, NEVER y un valor numérico. FOREVER causa que la dirección IP no tenga acceso a los servicios hasta que *xinetd* sea restaurado, NEVER permite que la dirección IP continué teniendo acceso y el valor numérico indica el número de minutos en que le será denegado el acceso[5].

No es necesario especificar todos los atributos para cada servicio. Los obligatorios son los siguientes:

- *socket_type*.
- *user* (solo para servicios no internos).
- *server* (solo para servicios no internos).
- *wait*.
- *protocol* (solo para servicios RPC o no listados).
- *rpc_version* (solo para servicios RPC).
- *rpc_number* (solo para servicios RPC no listados).
- *port* (solo para servicios no listados).

El resto adoptará valores por defecto si no se indica ningún valor.

Además, el fichero de configuración */etc/xinetd.conf* puede contener una entrada (y solo una) con los atributos asignados por defecto a todos los servicios[6]. El formato de dicha entrada es:

[5] Esta opción permite en muchos casos detener ataques de denegación de servicio desde una dirección IP.

```
defaults
{
      <atributo> = <valor>...
      ...
}
```

donde el campo *atributo* puede tomar los valores siguientes:

- *log_type* (efecto acumulativo).
- *bind*.
- *per_source*.
- *umask*.
- *log_on_success* (efecto acumulativo).
- *log_on_failure* (efecto acumulativo).
- *only_from* (efecto acumulativo).
- *passenv* (efecto acumulativo).
- *instances*.
- *disabled* (efecto acumulativo).
- *enabled* (efecto acumulativo).
- *banner*.
- *banner_success*.
- *banner_fail*.
- *per_source*.
- *groups*.
- *cps*.
- *max_load*.

Los atributos con efecto acumulativo pueden ser especificados múltiples veces, con los valores especificados cada vez acumulados, esto es, el operador '=' tiene el mismo efecto que el operador '+='. Un ejemplo de fichero */etc/xinetd.conf* es el siguiente:

```
defaults
{
      instances       = 60
      log_type        = SYSLOG authpriv
      log_on_success  = HOST PID
      log_on_failure  = HOST
      cps             = 25 30
}

includedir /etc/xinetd.d
```

En este fichero podemos ver que se indica que por defecto un servicio puede tener 60 instancias como máximo, que el tipo de log es SYSLOG y que el fichero que se utilizará es el de *authpriv*. Además, indicamos que la información en caso de aceptar una conexión será la IP del ordenador remoto y el PID del proceso servidor y en caso de rechazar la solicitud de conexión la IP del ordenador remoto. Por último, especificamos que a cada servidor se le

[6] Estos atributos serán modificados si existe la entrada correspondiente en la configuración del servicio, serán modificados de la forma que corresponda a cada uno de ellos.

pueden efectuar 25 solicitudes por segundo y, una vez sobrepasado ese límite, el servicio quedará desactivado durante 30 segundos.

Por último, podemos ver que aparece una línea *includedir /etc/xinetd.d* que indica que se incluya como configuración de *xinetd* la información contenida en los ficheros de ese directorio[7]. Algunos ejemplos de ficheros en el directorio */etc/xinetd.d* podrían ser los siguientes:

```
# Fichero daytime-tcp
# Servicio interno TCP que devuelve el día y la hora.
service daytime
{
      type            = INTERNAL
      id              = daytime-stream
      socket_type     = stream
      protocol        = tcp
      user            = root
      wait            = no
      disable         = yes
}

# Fichero daytime-udp
# Servicio interno UDP que devuelve el día y la hora.
service daytime
{
      disable         = yes
      type            = INTERNAL UNLISTED
      id              = daytime-dgram
      socket_type     = dgram
      protocol        = udp
      user            = root
      wait            = yes
      port            = 13
}

# Fichero rsync
# Sincroniza de forma remota archivos y directorios.
service rsync
{
      disable         = yes
      socket_type     = stream
      wait            = no
      user            = root
      server          = /usr/bin/rsync
      server_args     = --daemon
      log_on_failure  += USERID
}

# Fichero telnet.
# Servidor de telnet que utiliza la transmisión de usuarios y
# contraseñas sin cifrado para la autentificación.
service telnet
{
      flags           = REUSE[8]
      socket_type     = stream
      wait            = no
      user            = root
```

[7] Téngase en cuenta la excepción que sobre ciertos ficheros establece *includedir*.

[8] La bandera *REUSE* es obsoleta, pues en la actualidad todos los servicios tienen implícitamente la bandera *REUSE* activada.

```
        server          = /usr/sbin/in.telnetd
        log_on_failure     += USERID
        disable         = yes
}
# Fichero tftp
# Servidor de tftp utilizado para arranque remoto de sistema,
# descarga de ficheros de configuracion, etc.
service tftp
{
        socket_type     = dgram
        protocol        = udp
        wait            = yes
        user            = root
        server          = /usr/sbin/in.tftpd
        server_args     = -s /tftpboot
        disable         = yes
        per_source      = 11
        cps             = 100 2
        flags           = IPv4
}
```

Analizando los ficheros podemos ver que en los servicios de *daytime* hemos definido identificadores (atributo *id*) para los servicios (*daytime-tcp* y *daytime-udp*), pues existen servicios de igual nombre con versiones para TCP y UDP. Además, en *daytime-udp*, ha sido necesario definir el puerto de escucha.

Podemos ver como hay servicios internos y externos. En el primer grupo se encuentran por ejemplo el daytime-tcp o el daytime-udp ("type = INTERNAL"). Esto significa que el servicio es suministrado a los clientes por el propio proceso xinetd, sin necesidad de arrancar para ello ningún proceso adicional. En cambio otros servicios más complejos, como el Telnet o el TFTP, son externos, es decir requieren arrancar un programa servidor específico del servicio, indicándose en el atributo "server" el fichero que se debe ejecutar para proveer dicho servicio

También podemos ver en los ejemplos como en el servicio de *rsync* utilizamos la opción *server_args* para indicarle que argumentos han de pasarse al servidor en el momento de ser lanzado. Por último, podemos ver que el servidor de *tftp* posee la opción *cps = 100 2*, lo cual le indica que admita como máximo 100 conexiones simultáneas y si este número se sobrepasa este 2 segundos desactivado antes de comprobar si puede aceptar nuevas conexiones.

Apéndice B: Configuración del envolvente de acceso (TCP Wrappers).

Vamos a ver aquí en detalle la sintaxis que deben cumplir los ficheros de configuración de los TCP Wrappers, */etc/hosts.allow* y */etc/hosts.deny*. Los TCP Wrappers se aplican envolventes de acceso se aplican además de los permisos que se hayan establecido en el xinetdservicios a los que se permite acceso y desde que ordenadores y los servicios a los que se deniega el acceso y desde que ordenadores[9].

Las reglas de funcionamiento de los envolventes en cuanto al tratamiento de dichos ficheros ya fueron explicadas con anterioridad. En este punto veremos la sintaxis de esos ficheros, etc., así como un ejemplo de los mismos.

La sintaxis de las reglas de acceso es la siguiente:

```
<lista de servicios>: <lista de clientes> [: spawn <comando de shell>]
```

Donde <*lista de servicios*> es una lista de uno o más servicios separados por espacios, <*lista de clientes*> es uno o más nombres de ordenador, direcciones IP o patrones separados por espacios; y <*comando de shell*> es opcional e indica una acción a ejecutar si una regla se cumple. La lista de servicios o la lista de clientes pueden ser simplificadas utilizando patrones, pues permiten especificar grupos de servicios o grupos de clientes de forma sencilla.

El patrón más utilizado es el carácter punto (.) al principio de una cadena de caracteres o al final de una especificación de direcciones IP. Así, si escribimos *.irobot.uv.es*, nos estamos refiriendo a todo ordenador del dominio del Instituto de Robótica, mientras que si escribimos *147.156.* nos referimos a todo ordenador cuya dirección IP comienza por 147.156, esto es, ordenadores de la Universidad de Valencia[10]. Además del carácter punto pueden utilizarse como comodines los caracteres asterisco (*) e interrogación (?), con las mismas funcionalidades que poseen en el interprete de comandos del sistema operativo[11].

Además de los caracteres anteriores, existen unas palabras clave que pueden ser usadas en las de acceso en lugar de especificar la lista de clientes. Estas palabras clave pueden verse en la tabla siguiente:

Palabra	**Descripción**
ALL	Especifica todos los ordenadores.
LOCAL	Especifica todos los ordenadores de nuestra red local, esto es, que no contienen el carácter '.' en su nombre.
KNOWN	Especifica todos los ordenadores cuyo nombre o dirección IP son conocidos.
UNKNOW	Especifica todos los ordenadores cuyo nombre o dirección IP es desconocidos.
PARANOID	Especifica todos los ordenadores cuyo nombre no corresponde con su dirección IP.

[9] El envolvente de acceso general limita también los programas que pueden ejecutar xinetd, con lo cual debe configurarse teniendo en cuenta esta doble limitación.

[10] Siempre que sea posible, es conveniente utilizar las direcciones IP en vez de los nombres, pues para este último caso es necesario usar un DNS que puede ser objeto de modificación por parte de un intruso antes de intentar el acceso a un servicio de otro ordenador.

[11] Esto es, el carácter comodín * sustituye a cero o más caracteres, mientras que el carácter comodín ? sustituye solo a un carácter que además debe estar presente.

Las palabras KNOWN, UNKNOW y PARANOID deben ser utilizadas con precaución pues un error o alteración del DNS puede producir que ordenadores o usuarios no autorizados obtengan acceso a los servicios.

Las palabras clave anteriores contienen un operador, el operador EXCEPT, que permite que listas separadas sean combinadas en la misma línea. Cuando EXCEPT es utilizada entre dos listas, la lista de servicios se aplica a todas las entradas contenidas en la primera lista excepto a aquellas contenidas en la segunda lista. Para entender mejor el uso de estas palabras clave consideremos el siguiente ejemplo de línea en el fichero *hosts.allow*:

```
vsftpd: .irobot.uv.es EXCEPT amparo.irobot.uv.es glup.irobot.uv.es
```

En dicha línea estamos indicando que permitimos el uso del servicio de FTP proporcionado por *vsftpd* a todos los ordenadores del dominio del Instituto de Robótica excepto a los ordenadores de nombre *amparo* y *glup*[12].

Los nombres que se utilizan para identificar los servicios en los TCP Wrappers son en general diferentes de los utilizados en xinetd. En xinetd se utiliza el nombre asignado a cada servicio en el atributo “id”, en cambio en los TCP Wrappers se utiliza el nombre del programa ejecutable que ofrece el servicio, que es el nombre que en la configuración de xinetd aparece en el atributo “server”. Ahora bien, en el caso de los servicios internos de xinetd, que no tienen asociado un fichero ejecutable propio, se utiliza el identificador definido en xinetd. Así por ejemplo daytime (servicio interno de xinetd) utiliza el mismo nombre en TCP Wrappers, pero Telnet utiliza “in.telnetd”, tftp usa “in.tftpd” y FTP usa “vsftpd”.

Además de lo expuesto con anterioridad, la palabra clave ALL puede ser también utilizada en la lista de servicios, significando todos los servicios. Su uso podemos verlo en el siguiente ejemplo, que muestra además un uso más complicado del operador EXCEPT:

```
ALL: ALL EXCEPT in.telnetd: amparo.irobot.uv.es
```

Que permite a todos los ordenadores utilizar todos los servicios excepto al ordenador *amparo* al que no autoriza a utilizar el servicio de telnet.

Por último, el comando *spawn* permite ejecutar un comando en caso de que sea cierta una regla. Así, por ejemplo, podemos escribir en un fichero información sobre el ordenador, etc., al que se autoriza o deniega el uso de un servicio. Un ejemplo de regla que permite esto es:

```
in.telnetd: .irobot.uv.es : spawn (/bin/echo `date` %c >> /var/log/telnet.log) &
```

Que escribiría la hora (ejecución del comando date) y la dirección IP y usuario del ordenador que intento acceder.

Como ha podido verse en el ejemplo anterior, existen una serie de caracteres que indican que debe escribirse información obtenida a través de la conexión o intento de conexión. Estos caracteres se encuentran en la tabla siguiente:

[12] Es necesario tener en cuenta que la no autorización expresa a esos dos ordenadores no implica la denegación de utilización de ese servicio, esto debería hacerse constar de forma implícita en el fichero /etc/hosts.deny.

Carácter	Descripción
%a	La dirección IP del cliente.
%A	La dirección IP del servidor.
%c	Proporciona una variedad de información como el nombre del usuario y el nombre del ordenador, o el nombre del usuario y la dirección IP.
%d	El nombre del servicio solicitado.
%h	El nombre del cliente (o dirección IP si el nombre no existe).
%H	El nombre del servidor (o dirección IP si el nombre no existe).
%n	El nombre del cliente. Si no existe se escribe *unknow*. Si el nombre del cliente y su dirección IP no coinciden se escribe *paranoid*.
%N	El nombre del servidor. Si no existe se escribe *unknow*. Si el nombre del cliente y su dirección IP no coinciden se escribe *paranoid*.
%p	El identificador del proceso del servicio.
%s	Proporciona una variedad de información como el identificador del proceso y el nombre o dirección IP del servidor.
%u	El nombre del cliente. Si no existe se escribe *unknow*.

Un ejemplo de ficheros *hosts.allow* y *hosts.deny* es el siguiente:

```
# Fichero hosts.allow
# Servicio FTP (21/tcp) permitido a todo el mundo
vsftpd: ALL
# Servicio SSH (22/tcp) permitido a todo el mundo
sshd: ALL
# Servicio SMTP (25/tcp) permitido solo a Robótica
sendmail: .irobot.uv.es

# Fichero hosts.deny
ALL : ALL : spawn (/bin/echo `date` %h %d >> /var/log/deny.log) &
```

Como puede verse, el fichero *hosts.allow* permite explícitamente el servicio de FTP y SSH a todos los ordenadores de Internet, mientras que limita el uso del servicio de SMTP a los ordenadores del dominio *.irobot.uv.es* (en este ejemplo se supone que el programa sendmail está controlado por los TCP Wrappers).

Por su parte, el fichero *hosts.deny* niega todos los servicios a todos los ordenadores, de forma que si un ordenador no se encuentra entre los autorizados de forma explícita en *hosts.allow*, verá denegado su acceso a cualquier servicio[13]. La denegación incluye la escritura en un fichero de log de la fecha, el nombre o dirección IP del cliente que solicito el servicio (parámetro '%c') y el nombre del servicio solicitado (parámetro '%d').

[13] Una política de seguridad tan restrictiva permite controlar la seguridad del sistema y evitar posibles problemas de seguridad por errores en la configuración.

Apéndice C: El cortafuegos del kernel de Linux (iptables).

A partir de la versión 2.4 del kernel de Linux, surgió un nuevo y complejo sistema de tablas, cadenas y reglas que permiten filtrar paquetes entrantes y salientes, realizar un enmascaramiento de los mismos, modificar los campos de la cabecera de los paquetes, etc. Todo este complejo sistema es lo que se conoce como **iptables**.

iptables posee cuatro tablas predefinidas que son:

- *filter*: Es la tabla por defecto y la que se encarga de filtrar los paquetes de red.
- *nat*: Es la tabla usada para alterar, en los paquetes de entrada ó salida que establecen conexiones, las direcciones origen y/o destino de estos paquetes.
- *mangle*: Permite realizar alteraciones locales del origen o destino de los paquetes, lo que permite, por ejemplo, balancear el tráfico que accede a un servicio a un conjunto de ordenadores.
- *raw*: Permite configurar excepciones en el seguimiento de los paquetes de las conexiones[14].

Cada una de esas tablas posee un grupo de cadenas predefinidas, las cuales corresponden a las acciones que se ejecutan sobre los paquetes de red en el filtrado.

La tabla *filter* posee las siguientes cadenas predefinidas:

- *INPUT*: Se aplica a los paquetes destinados a un proceso local.
- *OUTPUT*: Se aplica a los paquetes generados de forma local por un proceso y que van a ser enviados por la red.
- *FORWARD*: Se aplica a los paquetes recibidos por un dispositivo de red y que van a ser reenviados por otro dispositivo de red del ordenador sin ser procesados por algún proceso local.

A su vez, la tabla *nat* posee las siguientes cadenas predefinidas:

- *PREROUTING*: Se aplica a los paquetes recibidos por un dispositivo de red antes de ser procesados.
- *OUTPUT*: Se aplica a los paquetes generados por un proceso local antes de ser enviados.
- *POSTROUTING*: Se aplica a los paquetes antes de que salgan a la red.

  Por su parte, la tabla mangle tiene las siguientes cinco cadenas predefinidas:

[14] La tabla *raw* aparece en los núcleos 2.6.X de Linux.

- *PREROUTING*: Se aplica a los paquetes recibidos por un dispositivo de red antes de ser enrutados.

- *INPUT*: Se aplica a los paquetes destinados a un proceso local.

- *OUTPUT*: Se aplica a los paquetes generados de forma local por un proceso antes de ser enrutados.

- *FORWARD*: Se aplica a los paquetes que son reenviados a través de dos dispositivos de red del ordenador sin la intervención de ningún proceso local.

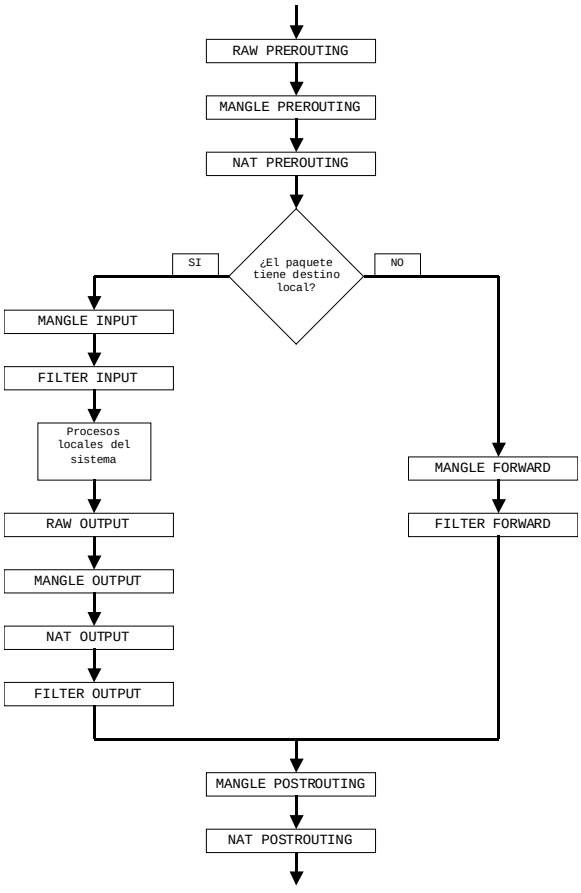
- *POSTROUTING*: Se aplica a los paquetes antes de salir a la red.

Por último, la tabla *raw* tiene las siguientes dos cadenas predefinidas:

- *PREROUTING*: Se aplica a los paquetes recibidos por cualquier dispositivo de red.

- *OUTPUT*: Se aplica a los paquetes generados localmente por un proceso local.

La estructura de tablas y cadenas indicada puede parecer confusa, pues no queda claro en que punto actúa cada de las cadenas de las tablas y por tanto, como afectan a los paquetes entrantes ó salientes del ordenador. Por ello, es conveniente observar la siguiente figura:



En ella, podemos ver el orden en que actúa cada una de las cadenas de las tablas y, por tanto, como afecta su acción al resto de cadenas de las tablas, pues un paquete modificado por una

regla de una cadena de una tabla aparece, para el resto de reglas que la analicen posteriormente, con ese cambio y sin posibilidad de conocer el contenido inicial del paquete.

Así, el siguiente ejemplo, permite reenviar el tráfico destinado al puerto 80 TCP de nuestro ordenador (192.168.0.1) a otro ordenador de IP 192.168.0.2:

```
iptables -t nat -A PREROUTING -p tcp -d 192.168.0.1 --dport 80  -j DNAT --to-destination 192.168.0.2:80
```

A partir de que esta regla actúe sobre un paquete, el resto de reglas, incluido el examen sobre si el paquete es local o no, verán el paquete como destinado a la dirección IP 192.168.0.2, en lugar de la 192.168.0.1 que es la dirección con la que llegó inicialmente el paquete a nuestro ordenador.

Este ejemplo esta más allá del alcance de estos apuntes, por lo que no incidiremos más en el mismo o ejemplos similares de reenvío de puertos, balanceo de carga, etc., que pueden realizarse fácilmente con las iptables, centrándonos únicamente en su funcionamiento como cortafuegos.

Acciones existentes por defecto en iptables.

Por defecto, iptables puede realizar sobre un paquete de red una de las siguientes acciones:

- *ACCEPT*: Que indica que el paquete no debe ser analizado por el resto de reglas y tablas y debe permitirse su continuación hasta el destino.

- *DROP*: Que especifica que el paquete debe ser rechazado sin enviar ningún tipo de mensaje a la dirección de origen del paquete.

- *QUEUE*: Indica que el paquete debe ser enviado para su análisis a un módulo en el espacio del usuario[15].

- *RETURN*: Devuelve el paquete a la regla siguiente a la que ocasiono la llamada a la regla que contiene esta acción. Suele ser utilizada en las reglas que se encuentran en las cadenas definidas por los usuarios.

Si ninguna de las reglas puede ser aplica al paquete, esté se comportará con el modo por defecto definido en la tabla.

Además de las reglas anteriores, existen extensiones a las mismas, las cuales dependen de la tabla en la que se encuentre la regla. Así, en un ejemplo anterior hemos visto la acción *DNAT*, que es una acción extendida existente para reglas de las cadenas de la tabla *nat*.

Comandos de iptables.

Cada tabla posee por defecto un comportamiento predefinido, basado en el propósito de la tabla. Dicho comportamiento puede ser alterado mediante la ejecución de comandos, los cuales poseen la siguiente estructura general:

[15] Esta regla debe estar soportada explícitamente por el kernel.

iptables [-t <nombre de tabla>] <comando> <nombre de la cadena> <parámetro 1> <opción 1>...<parámetro N> <opción N>

donde <*nombre de tabla*> permite indicar sobre que tabla se ejecuta el comando. Si no se especifica se ejecuta sobre la tabla por defecto (tabla *filter*). El campo <*comando*> es el centro de la orden, indicando una acción específica como puede ser añadir una regla, borrar una regla, etc., de la cadena especificada por <*nombre de la cadena*>. Los parámetros y opciones siguientes definen la regla, que acción debe realizar, etc., sobre los paquetes que estén de acuerdo con ella.

La complejidad, etc., de los comandos varía según el objetivo del comando. Si el comando pretende eliminar una regla, puede ser tan sencillo como indicar la cadena y la posición de la regla en la cadena, mientras que si se trata de filtrar paquetes de una subred con una gran variedad de opciones, etc., puede ser muy complicada.

Los comandos admitidos por *iptables* son sensibles al contexto, siendo especificados por una letra mayúscula, excepto el comando de ayuda que es especificado por una letra minúscula. Los comandos existentes y una descripción de los mismos se encuentra a continuación:

Comando	**Descripción**
-A	Añade la regla especificada al final de la cadena especificada.
-C	Chequea una regla antes de que sea añadida por el usuario a la cadena especificada.
-D	Borra una regla de la cadena especificada. Puede especificarse por un número que indique su posición, comenzando a contar siempre en 1, o bien escribir la regla completa a borrar.
-E	Renombra una cadena definida por el usuario. Esta acción no afecta a la estructura de la tabla donde se encuentra la cadena.
-F	Borra todas la reglas de la cadena especificada. Si no se especifica la cadena, todas las reglas de todas las cadenas son borradas.
-h	Proporciona información de ayuda.
-I	Inserta una regla en la cadena en la posición indicada. Si no se indica ninguna posición la regla es insertada al principio de la cadena.
-L	Lista todas las reglas. Los valores -v, -x y -n, permiten especificar que la salida sea más extensa, que se de en valores exactos y no abreviados con K (miles), M (millones), etc., y que se de en valor numérico de direcciones IP y puertos, respectivamente.
-N	Crea una nueva cadena con el nombre especificado por el usuario.
-P	Asigna la política por defecto a una cadena, de forma que si un paquete no corresponde a ninguna regla, esta será la acción por defecto a aplicar.
-R	Reemplaza la regla situada en la posición indicada de la cadena por la regla especificada. Como en la opción –D empieza a contar en 1.
-X	Borra una cadena especificada por el usuario. Borrar una cadena predefinida de una tabla no esta permitido.
-Z	Inicializa a cero el contador de bytes y paquetes en todas las cadenas de una tabla.

Algunos ejemplos de comandos válidos son los siguientes:

```
iptables -t nat -L -v
```

que permite obtener los datos sobre las reglas definidas en todas las cadenas de la tabla *nat*.

```
iptables -t mangle -N MI_CADENA
```

que crea una nueva cadena, de nombre *MI_CADENA* en la tabla *mangle*. Esta cadena deberá ser llamada desde alguna de las cadenas por defecto de la tabla en la que se define.

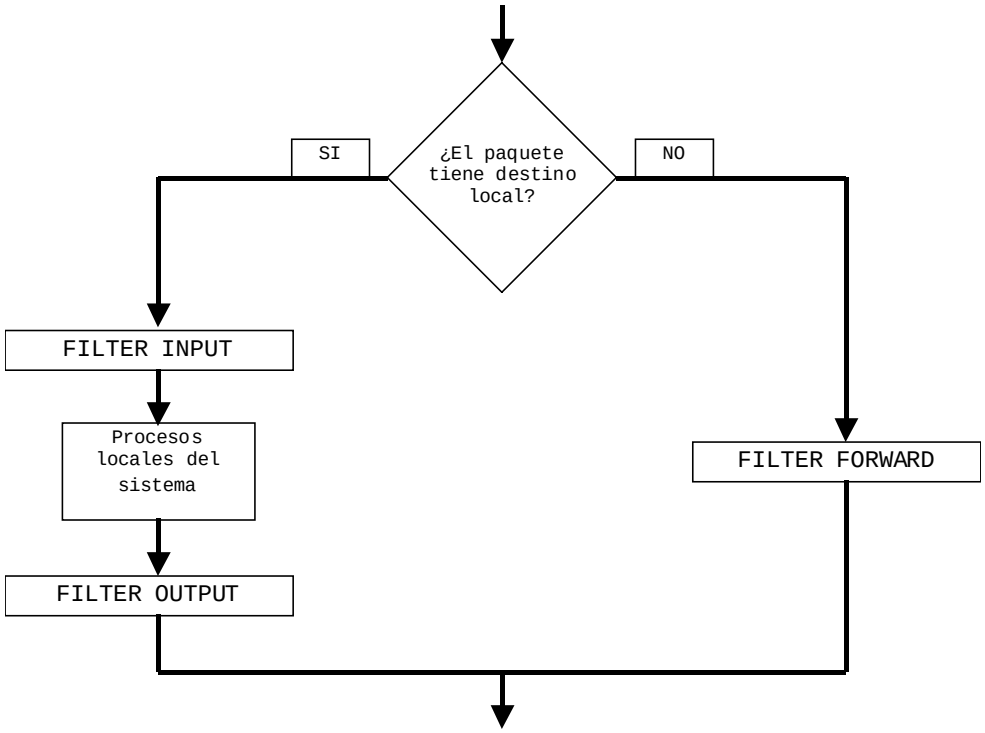
```
iptables -I INPUT 3 -p tcp -s 147.156.0.0/16 -j ACCEPT
```

que añade la regla especificada (*-p tcp -s 147.156.0.0/16 -j ACCEPT*) en la tercera posición de la cadena *INPUT* de la tabla *filter* (tabla por defecto).

La tabla filter de iptables.

En primer lugar, comentar que la mayoría de lo comentado a partir de este punto para la tabla *filter* es perfectamente valido para el resto de tablas existentes. Sin embargo, y con el fin de simplificar el contenido y hacerlo más pedagógico, omitiremos cualquier referencia a la otras tablas aunque lo indicado pueda utilizarse en las mismas, centrándonos únicamente en la tabla *filter*.

La tabla *filter* de iptables es la encargada de filtrar los paquetes que tienen como origen o destino el ordenador, decidiendo si estos son admitidos o reenviados o bien son rechazados (filtrados)[16]. Si nos atenemos a unas iptables donde solo son utilizadas las cadenas de la tabla filter, la figura vista con anterioridad puede simplificarse en la siguiente:



Como puede observarse en la figura todo paquete es examinado por una cadena de reglas, pudiendo decidirse sobre si dicho paquete es o no admitido (*INPUT*), enviado a la red (*OUTPUT*) o reenviado (*FORWARD*).

La estructura y propósito de las reglas varía, pero de forma general, tratan de identificar paquetes que tienen como origen o destino una dirección o conjunto de direcciones IP. Cuando un paquete encaja con una de las reglas definidas en una de las tablas, es marcado para realizar sobre él una acción concreta de las vistas con anterioridad o una de las siguientes acciones, que son extensiones de las acciones generales vistas con anterioridad:

[16] Aunque las otras tablas y sus cadenas pueden realizar también funciones de filtrado, pero la configuración más normal es que el filtrado se realice en las cadenas de la tabla filter.

- *REJECT*: Especifica que el paquete debe ser rechazado, enviando un paquete ICMP de error al origen del paquete. Posee la opción *--reject-with <tipo>*, que especifica el tipo de mensaje de error ICMP a enviar al origen. El tipo debe ser uno de los valores *icmp-net-unreachable*, *icmp-host-unreachable*, *icmp-port-unreachable*, *icmp-proto-unreachable*, *icmp-net-prohibited*, *icmp-host-prohibited* ó *icmp-admin-prohibited*[17]. Si la opción no es especificada, se devuelve por defecto el mensaje de error *icmp-port-unreachable*.

- *LOG*: Especifica que se almacene información sobre el paquete en el log del sistema. Esta extensión no termina la ejecución de las reglas, pues tan solo incluye la información en el log y continua ejecutando la regla siguiente. Sus principales opciones son *--log-level <nivel>*, que indica el nivel del log[18] y *--log-prefix <cadena>*, donde *<cadena>* es una cadena de texto de hasta 29 caracteres que se antepone al mensaje que generará iptables y que permite identificar los mensajes en el fichero de log.

Una vez vista la estructura de las iptables con la tabla *filter*, veremos a continuación como es posible especificar las reglas de las cadenas. Como se comentó con anterioridad, la mayoría de lo aquí descrito sirve para cualquiera de las tablas existentes, pero alguno de los elementos aquí descritos pueden servir solo para la tabla *filter*.

Parámetros de especificación de reglas.

Los parámetros que permiten especificar y construir las reglas son:

- *-c*: Inicializa el contador de una determinada regla. Acepta los valores *PKTS* o *BYTES* para especificar el contador a inicializar (paquetes o bytes).

- *-d*: Selecciona el nombre del ordenador, dirección IP o red, que es el destino del paquete. Cuando se especifica una red, puede hacerse mediante las dos formas existentes de especificación de la máscara: 192.168.0.0/255.255.0.0 o 192.168.0.0/16. La opción *!* puede preceder a la especificación del ordenador, etc., e indica que se aplique la regla a los ordenadores que no correspondan a la especificación del ordenador, etc., realizada.

- *-f*: Aplica la regla solo a paquetes que estén fragmentados. Puede ir precedida del símbolo *!* que indica que solo debe aplicarse a paquetes no fragmentados.

- *-i*: Identifica el dispositivo de red de entrada, como *ppp0* ó *eth0*, al que se debe aplicar la regla. Si ningún dispositivo de red es especificado se toma que todos los dispositivos de red existentes son afectados. Este parámetro solo puede utilizarse con las cadenas *INPUT* ó *FORWARD*. Este parámetro admite las opciones *!*, que precede al nombre del dispositivo de red y especifica que no se aplique la regla al dispositivo de red especificado y +, que es utilizado como comodín y permite especificar de

[17] Si el kernel no soporta los mensajes ICMP de tipo *icmp-admin-prohibited*, la opción REJECT se comporta exactamente igual que la opción DROP, por lo que no envía ningún tipo de respuesta.
[18] Para ver los niveles de los posibles consultar la página de manual de syslog.conf.

forma simultanea un grupo de dispositivos de red, como por ejemplo *eth+*, que especificaría *eth0*, *eth1*, etc.

- *-j*: Especifica una acción concreta cuando el paquete coincide con la regla[19]. Si no se especificá ninguna acción, el paquete pasa a ser comprobado por la siguiente regla, pero el contador de está regla es incrementado en una unidad, pues el paquete cumplió la regla.

- *-m*: Especifica que se va a utilizar una extensión de los parámetros básicos aquí descritos.

- *-o*: Identifica el dispositivo de red de salida para una regla. Solo puede aplicarse a las cadenas *OUTPUT* o *FORWARD*. Su comportamiento y opciones existentes son las mismas que las del parámetro *–i* visto con anterioridad.

- *-p*: Selecciona el protocolo IP al que se aplicará la regla, por ejemplo *tcp*, *udp*, *icmp*, etc., o *all* para todos los protocolos soportados. Cualquier protocolo existente en el fichero */etc/protocols* puede ser indicado en este parámetro. Si esta opción es omitida se presupone la opción *all* para la regla. Posee como opción *!*, que precede al protocolo e indica que no se aplique esta regla a ese protocolo especificado.

- *-s*: Selecciona el nombre del ordenador, dirección IP o red, que es el origen del paquete. La especificación se realiza igual que lo visto con anterioridad para el parámetro *–d*.

Como se ha indicado expresamente, la opción *-m* permite especificar extensiones, las cuales pueden o no corresponder a un protocolo determinado. Si la extensión depende de un protocolo en concreto, y este se ha especificado en la regla mediante la opción *-p*, es posible utilizar las extensiones de este protocolo sin necesidad de especificar la opción *-m*. A continuación veremos las extensiones a los tres protocolos de transporte más utilizados (TCP, UDP e ICMP), así como algunas extensiones generales.

Extensiones de TCP (-p tcp).

Las extensiones disponibles con el protocolo TCP son:

- *--dport*: Especifica el puerto de destino del paquete. El puerto puede especificarse como un nombre de servicio de red[20] (www, smtp, etc.), numero de puerto, o rango de números de puerto. El rango de números de puertos se especifica como dos números separados por el símbolo *:*. La extensión puede tener delante de los puertos el símbolo *!* e indica que se aplique a los puertos que no están especificados en esta regla[21].

- *--sport*: Especifica el puerto de origen del paquete. Sus valores, sintaxis, etc., son idénticas a la opción *--dport*.

[19] Las acciones validas son tanto las acciones por defecto (*ACCEPT*, *DROP*, *QUEUE* y *RETURN*) como las acciones extendidas (*REJECT* y *LOG*).
[20] Si se especifica mediante esta forma, el servicio de red debe encontrarse definido en el fichero /etc/services.
[21] El símbolo *!* debe ir detrás justo de la extensión, es decir, de la forma *--dport ! 25*, por ejemplo.

- --*syn*: Se aplica a los paquetes TCP que inician una comunicación en este protocolo[22]. Si se coloca el carácter *!* delante de la opción se indica que se aplique a los paquetes TCP que no inician la comunicación.

- --*tcp-flags*: Permite especificar los paquetes a los que se aplicará según el valor de los bits de bandera que indican las opciones de TCP. Los bits de bandera son *ACK*, *FIN*, *PSH*, *RST*, *SYN* y *URG* y se especifican mediante dos listas cuyos elementos se separan por comas. La primera lista indica las banderas a examinar y la segunda especifica el valor que deben tener las banderas para que se cumpla la regla, de forma que si una bandera se encuentra en la segunda lista debe tener un valor 1 y un valor 0 en caso contrario. Por supuesto, toda bandera que se encuentre en la segunda lista debe encontrarse en la primera lista. Por ejemplo, la opción --*tcp-flags ACK,FIN,SYN SYN* indica paquetes que tengan activos el bit de *SYN* (valor 1) y no activados los bits de *ACK* y *FIN* (valor 0). Si se especifica detrás el carácter *!*, el sentido de la opción es invertido de forma que las banderas indicadas en la segunda lista deberán estar a 0 y a 1 los no especificados.

- --*tcp-option*: Permite especificar si el paquete contiene una opción TCP concreta de las posibles opciones de la cabecera TCP. Esta opción puede también invertirse utilizando el símbolo *!*.

Extensiones de UDP (-p udp).

Las extensiones disponibles con el protocolo UDP son:

- --*dport*: Especifica el puerto de destino del paquete. El puerto puede especificarse como un nombre de servicio de red (DNS, etc.), numero de puerto, o rango de números de puerto. El rango de números de puertos se especifica como dos números separados por el símbolo *:*. La extensión puede tener detrás el símbolo *!* e indica que se aplique a los puertos que no están especificados en esta regla.

- --*sport*: Especifica el puerto de origen del paquete. Sus valores, sintaxis, etc., son idénticas a la opción --*dport*.

Extensiones de ICMP (-p icmp).

El protocolo ICMP solo posee una extensión permitida, esta es:

- --*icmp-type*: Que especifica el nombre o número del tipo de ICMP que debe cumplir esta regla[23]. Permite la utilización del símbolo *!* para indicar los paquetes ICMP que no sean del tipo especificado.

Extensiones generales.

[22] El protocolo TCP inicia el establecimiento de conexión mediante el envío y la respuesta de un paquete que no contiene datos y que posee el flag de SYN a 1, indicando cada paquete la petición de conexión y la aceptación de dicha conexión, respectivamente.

[23] El comando iptables –p icmp –h permite obtener una lista completa de los tipos de paquetes ICMP soportados y que pueden especificarse en la regla.

Existen algunas extensiones añadidas a las opciones anteriores y que no van unidas al uso de ningún protocolo particular. El uso de las extensiones esta ligado al parámetro de especificación de reglas *-m*, de forma que para poder usar una extensión es necesario especificar *-m <extensión>*, pudiendo utilizarse varias entradas *-m* en la misma regla para indicar el uso de distintas extensiones.

Aunque el conjunto de extensiones existentes es muy amplio[24], aquí solo veremos un pequeño número de las mismas, que corresponden a las extensiones que suelen ser más utilizadas, y que son las siguientes:

- *mac*: Es solo valida en las cadenas *INPUT* y *FORWARD* y permite especificar, mediante la opción *--mac-source* la dirección MAC de la que provienen los paquetes de red. La dirección MAC se especifican en formato *XX:XX:XX:XX:XX:XX*, y puede ir precedida del símbolo *!*, lo cual indica que la regla se aplique a las direcciones MAC que no correspondan con la especificada.

- *state*: Permite, mediante el seguimiento de una conexión, controlar el acceso de un paquete en función del estado de la conexión. La única opción que posee esta extensión es *--state <estado>*, donde *<estado>* especifica el estado de la conexión. Los valores posibles para *<estado>* son *INVALID*, que indica que el paquete esta asociado a una conexión desconocida; *ESTABLISHED*, que indica que el paquete esta asociado a una conexión ya establecida y que envía paquetes en ambas direcciones; *NEW*, que indica que el paquete desea establecer una nueva conexión o bien que no esta asociado a una conexión que envía paquetes en ambas direcciones; o *RELATED*, que indica que el paquete inicia una nueva conexión que esta asociada a otra, como puede ser el envío de datos en una conexión *FTP* o un error *ICMP*.

- *time*: Permite especificar valores temporales para las reglas. Posee múltiples opciones, no siendo necesario especificar todas ellas, pues los valores por defecto de las mismas permiten que la regla funcione sin especificar ningún valor. Las opciones más interesantes son aquellas que permiten definir un rango temporal de validez dentro de un día o los días de la semana. El rango de validez se especifica mediante las opciones *--timestart <valor>* y *--timestop <valor>*, que indican el tiempo inicial y final de validez de la regla. El parámetro valor se especifica en formato hh:mm, siendo el valor por defecto 00:00 para *timestart* y 23:59 para *timestop*. Por su parte, el día de la semana se especifica mediante la opción *--days <lista de días>*, donde la *<lista de días>* es una lista que puede contener uno o más valores, separados por comas, de los días de la semana (Mon, Tue, Wed, Thu, Fri, Sat y Sun). Si no se especifica ningún día el valor por defecto es todos los días.

Guardando la configuración de las iptables.

Las reglas, etc., creadas con el comando *iptables* son almacenadas solamente en la memoria RAM del sistema, lo cual ocasiona que al inicializar el sistema las reglas creadas se pierdan. Por ello, si se desea que las reglas creadas puedan ser almacenadas y utilizadas la siguiente vez que se inicialice el ordenador, han de ser almacenadas en el fichero

[24] Consultar el manual de iptables para una lista completa de las extensiones disponibles así como sus opciones.

/etc/sysconfig/iptables, el cual contiene las reglas por defecto que son leídas en el arranque del cortafuegos del sistema.

Si se observa cualquier fichero */etc/sysconfig/iptables*, se podrá ver que tiene un formato similar al aquí descrito en la especificación de las reglas. Dicho formato podemos verlo en el siguiente ejemplo:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -i eth0 -p udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -j ACCEPT
-A INPUT -p tcp --syn -j REJECT
-A INPUT -p udp -j REJECT
COMMIT
```

Dicho formato es sencillo. En la primera línea podemos ver que se especifica la tabla sobre la que se aplican las reglas (tabla *filter*). A continuación vienen la especificación de la acciones por defecto que se aplicarán sobre cada una de las cadenas de reglas que posee esa tabla, en nuestro caso *INPUT*, *FORWARD* y *OUTPUT*, seguidas de una pareja de valores que en nuestro ejemplo son *[0:0]*. Dicha pareja de valores indica el valor inicial que tienen los contadores de paquetes y bytes de esa regla al inicializar las *iptables*. A continuación pueden verse la inserción de reglas en las cadenas de esa tabla y por último la palabra *COMMIT*, que indica el final de especificación de reglas para esa tabla.

Aunque dicho formato es sencillo, existe la posibilidad de guardar en cualquier momento el estado de las *iptables* en el fichero de configuración mediante la ejecución, como usuario *root*, del comando[25]:

```
service iptables save
```

Dicho comando hace que se ejecute el programa */sbin/iptables-save* y se escriba la configuración actual en memoria de las *iptables* en el fichero */etc/sysconfig/iptables*. De esta forma, la siguiente vez que el ordenador se inicialice, tendrá la configuración del cortafuegos que escribimos con los comandos vistos con anterioridad[26].

El fichero iptables-config.

En el directorio */etc/sysconfig*, además del fichero *iptables* que contiene la especificación de las reglas para el cortafuegos, existe otro fichero, llamado *iptables-config*, que además de especificar el comportamiento de las iptables en el momento de ser cargadas, salvadas, etc., permite indicar los módulos que deben ser cargados para realizar el seguimiento de las conexiones.

[25] Una buena opción es, antes de ejecutar dicho comando, realizar una copia del antiguo fichero de iptables, con el fin de poder restaurar las mismas.

[26] La ejecución del comando crea un fichero iptables que posee como diferencia principal el que delante de cada regla se guarda, en el mismo formato que para las cadenas por defecto, los contadores de paquetes y bytes que han sido procesados por dicha regla.

Los módulos de seguimiento de las conexiones se especifican en la línea *IPTABLES_MODULES* y consiste en un lista de módulos, separados por espacios, que el sistema debe cargar junto con las iptables.

El conjunto de módulos existentes es muy amplio, encontrándose los mismos en el directorio */lib/modules/<versión del kernel>/kernel/net/ipv4/netfilter*. De estos módulos, el que permite realizar el seguimiento de las conexiones FTP por ejemplo, para permitir el establecimiento de conexiones a los puertos en el modo pasivo del servidor, es el módulo *ip_conntract_ftp.ko*, debiendo en este caso contener la línea *IPTABLES_MODULES* del fichero */etc/sysconfig/iptables-config* la entrada:

```
IPTABLES_MODULES="ip_conntract_ftp"
```

Algunos ejemplos de cortafuegos en Linux.

Veremos a continuación algunos ejemplos de cortafuegos configurados en Linux. Estos ejemplos, además de servir de ejemplo de uso de lo descrito con anterioridad, servirán para ilustrar tres casos básicos de configuración de un cortafuegos de Linux: Un cliente Linux, un servidor Linux y un servidor Linux con dos interfaces que actúa como puerta de acceso a una subred.

Cliente Linux.

El ejemplo de cliente Linux es muy sencillo. Corresponde a un ordenador que actúa como estación de trabajo y que requiere arrancar la red mediante DHCP.

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -i eth0 -p udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -j ACCEPT
-A INPUT -p tcp --syn -j REJECT
-A INPUT -p udp -j REJECT
COMMIT
```

En este ejemplo podemos ver que las reglas especificadas son muy sencillas.

- La primera regla permite (*-j ACCEPT*) que sea aceptado todo el tráfico de entrada proveniente de la red de loopback (*-i lo*).

- La segunda regla indica que el dispositivo de red *eth0* acepte (*-j ACCEPT*) el tráfico de entrada udp (*-p udp*) con origen en cualquier ordenador (*-s 0/0*) y puertos de origen 67 o 68 (*--sport 67:68*) y con destino cualquier ordenador (*-d 0/0*) y puertos de destino 67 o 68 (*--dport 67:68*)[27].

- La tercera regla indica que todo el tráfico de entrada tcp (*-p tcp*) que quiera iniciar una conexión tcp (*--syn*), sea rechazado (*-j REJECT*).

[27] Esta regla, que de forma explícita es la que permite la configuración mediante DHCP, permite, de forma implícita, que los servidores de nombres (DNS) que proporciona la configuración mediante DHCP queden autorizados a enviar paquetes UDP de resolución de nombres, esto es, paquetes UDP con origen en el servidor de nombre y puerto de origen 53.

- Por último, la cuarta regla indica que todo el tráfico de entrada udp (*-p udp*) sea rechazado (*-j REJECT*).

Servidor Linux.

El siguiente ejemplo corresponde a un ordenador que actúa como servidor Linux de correo, Web, DNS, etc., y que debe permitir el acceso a los servicios que ofrece.

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT[0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p udp --dport 53 -j ACCEPT
-A INPUT -p udp --sport 53 -j ACCEPT
-A INPUT -p udp -j REJECT
-A INPUT -p tcp --dport 21 --syn -j ACCEPT
-A INPUT -p tcp --dport 22 --syn -j ACCEPT
-A INPUT -p tcp --dport 25 --syn -j ACCEPT
-A INPUT -p tcp --dport 80 --syn -j ACCEPT
-A INPUT -p tcp --dport 110 --syn -j ACCEPT
-A INPUT -p tcp --dport 995 --syn -j ACCEPT
-A INPUT -p tcp --syn -j REJECT
COMMIT
```

En este segundo ejemplo podemos ver que las reglas, a pesar de ser más numerosas, son tan sencillas como en el ejemplo anterior.

- La primera regla permite (*-j ACCEPT*), igual que en el ejemplo anterior, que sea aceptado todo el tráfico de entrada proveniente de la red de loopback (*-i lo*).

- Las dos siguientes reglas permiten que el ordenador funcione como servidor de nombres (DNS), pues habilitan (*-j ACCEPT*) la recepción de paquetes udp (*-p udp*) que tengan como destino el puerto 53 (*--dport 53*) o como origen el puerto 53 (*--sport 53*)[28].

- La siguiente regla indica que se rechace todo el tráfico UDP que no haya cumplido alguna de las reglas anteriores.

- Las siguientes seis reglas permiten que cualquier dispositivo de red[29], pues este no se encuentra especificado, admita (*-j ACCEPT*) los paquetes tcp (*-p tcp*) que quieran iniciar una conexión (*--syn*) con los puertos especificados como destino (*--dport 21*, *--dport 22*, *--dport 25*, *--dport 80*, *--dport 110* y *--dport 995*)[30].

[28] Es necesario habilitar también la recepción de paquetes UDP con puerto de origen 53 pues, en caso de que la petición corresponda a un ordenador de otro dominio, nuestro servidor DNS deberá enviar una consulta al servidor de DNS de ese otro dominio, el cual responderá desde el puerto 53, debiendo por tanto aceptar dichos paquetes UDP con origen en el puerto 53 para recibir esas respuestas.
[29] El ordenador sobre el que se encuentran estas reglas posee un único dispositivo de red ethernet (eth0).
[30] Con estas seis reglas se da acceso a los servicios TCP de FTP en modo activo, SSH, sendmail, HTTP POP3 y POP3s, respectivamente.

- La última regla indica que se rechace todo el tráfico TCP que quiera establecer una conexión y que no haya cumplido alguna de las reglas anteriores.

Servidor Linux como puerta de acceso a una subred.

El último ejemplo corresponde a un ordenador que actuá como puerta de acceso de una red interna a Internet. Dicho ordenador posee dos dispositivos de red, un modem (dispositivo ppp0) y una tarjeta ethernet (eth0), limitando el acceso de los ordenadores de Internet a la red interna y permitiendo a los ordenadores de la red interna acceder a Internet.

```
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -o ppp0 -j MASQUERADE
COMMIT
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -i eth0 -j ACCEPT
-A INPUT -p udp --sport 53 -j ACCEPT
-A INPUT -p tcp --syn -j REJECT
-A INPUT -p udp -j REJECT
-A FORWARD -i ppp0 -o eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FORWARD -i eth0 -o ppp0 -j ACCEPT
-A FORWARD -j DROP
COMMIT
```

En primer lugar, podemos ver como este ejemplo implica la modificación de dos tablas, las tablas *nat* y *filter*.

En la tabla *nat*, además de establecer como política por defecto de sus tres cadenas de reglas el que todo sea aceptado, añadimos una regla:

- La regla añadida indica que a todo el tráfico que vaya a ser enviado (*POSTROUTING*) por el modem (*-o ppp0*), se le modifique la dirección IP y se le asigne la dirección IP que le ha sido proporcionada al modem (*-j MASQUERADE*)[31].

Por su parte, en la tabla *filter* se establecen las políticas por defecto en sus cadenas de reglas y se añaden las siguientes reglas:

- En las dos primeras reglas permiten (*-j ACCEPT*), que sea aceptado todo el tráfico de entrada proveniente de la red de loopback (*-i lo*) y de la Intranet (*-i eth0*).

- En la siguiente regla aceptamos (*-j ACCEPT*) todo el tráfico udp (*-p udp*) proveniente del puerto 53 (*--sport 53*). De esta forma permitimos la resolución de nombre mediante el uso del servicio de DNS.

[31] De esta forma, todo el tráfico de red proveniente de la Intranet, y que alcanza Internet por el modem, es como si tuviera como origen el ordenador que tiene físicamente el modem.

- Por último, las dos reglas siguientes indican, al igual que en los anteriores ejemplos, que se rechace todo el tráfico TCP y UDP que no haya cumplido alguna de las reglas anteriores.

Aunque aparentemente el cortafuegos ya esta configurado correctamente, esto no es así, pues debemos todavía configurar las reglas de la tabla *filter* que afectan al reenvió de paquetes[32], esto es, las reglas que afectan a la cadena *FORWARD*. Las tres reglas establecidas son las siguientes:

- La primera regla indica que se acepten (*-j ACCEPT*) todos los paquetes que tengan como origen el modem (*-i ppp0*) y como destino la Intranet (*-o eth0*), siempre que su estado (*-m state --state ESTABLISHED,RELATED*) corresponda a una conexión ya establecida (*ESTABLISHED*) o a una nueva conexión cuya apertura esta relacionada con otra conexión ya existente (*RELATED*).

- La segunda regla indica que se acepten (*-j ACCEPT*) los paquetes que tengan como origen la Intranet (*-i eth0*) y destino Internet, esto es, el modem (*-o ppp0*).

- Por último, la tercera regla indica que todo lo que no cumpla las reglas anteriores de reenvió sea rechazado (*-j DROP*).

[32] Estas reglas permitirán asegurar los ordenadores existentes en el interior de la Intranet, tal y como sucede en el diseño de un cortafuegos de un solo router.