Control de acceso a los servicios II: Iptables.

Autor: Enrique V. Bonet Esteban

Introducción.

Un cortafuegos es un sistema que, colocado entre la red interna de una organización y la red externa, proporciona una manera simple de controlar el tráfico entre ambas redes. Existen dos estrategias básicas para definir el funcionamiento de un cortafuego:

- Permiso predeterminado: El cortafuego posee un determinado conjunto de condiciones que tendrán como resultado que los datos no pasen a la red interna. Cualquier ordenador o protocolo no cubierto dentro de las condiciones estará autorizado de forma predeterminada a entrar en la red interna.

- Negación predeterminada: El cortafuegos posee un conjunto de condiciones y ordenadores a los cuales se les permite pasar a la red interna. Cualquier otro ordenador o protocolo será denegado en su intento de acceso.

La ventaja principal del permiso predeterminado es que es más fácil de configurar, pues simplemente se bloquean los protocolos “peligrosos”. Con la negación predeterminada, los protocolos son habilitados a medida que lo solicitan los usuarios, siendo necesario un continuo seguimiento del cortafuego.

Los cortafuegos, además de permitir o denegar los accesos a la red, pueden ser utilizados para otros fines, como puede ser el cifrar de forma automática las comunicaciones entre dos redes, realizar estadísticas de acceso desde la red interna hacia la red externa y viceversa, etc[1].

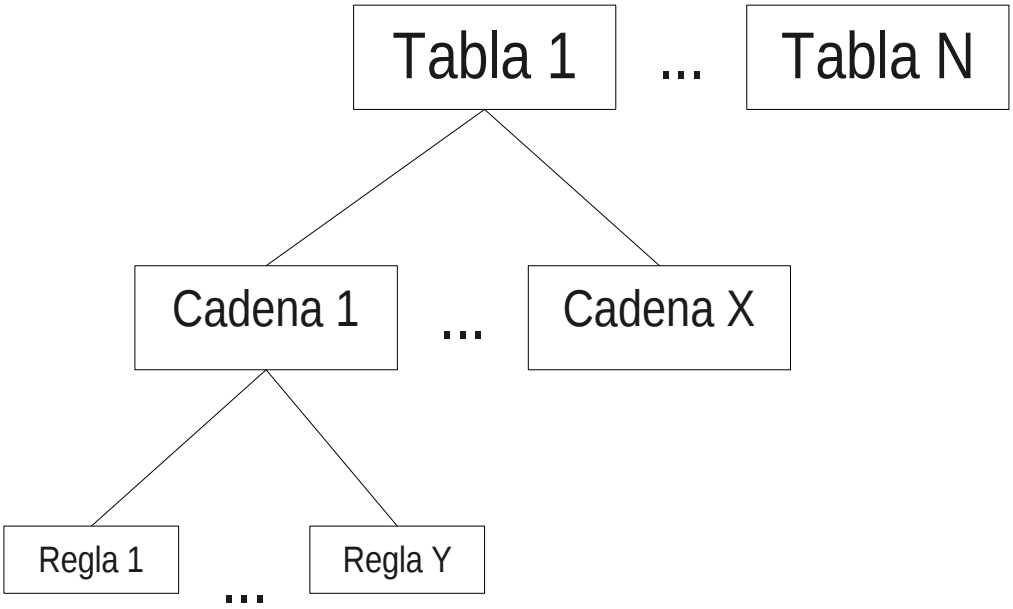
Las iptables del kernel de Linux.

En versiones anteriores al kernel 2.4 de Linux existía un conjunto de reglas, conocido como ipchains, que permitían realizar filtrado y enmascaramiento de direcciones IP. Sin embargo, este conjunto de reglas no tenía la posibilidad de realizar el seguimiento de conexiones, etc., por lo que era difícil escribir reglas para ciertos casos que pueden producirse en un cortafuegos.

Por ello, a partir de la versión 2.4 del kernel de Linux, surgió un nuevo y complejo sistema de tablas, cadenas y reglas que permiten no solo filtrar paquetes entrantes y salientes, sino 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 del kernel de Linux.

La estructura de tablas, cadenas y reglas puede parecer de entrada complicada, pero en realidad es bastante sencilla y puede observarse en la siguiente figura:

[1] Una breve explicación teórica de las distintas arquitecturas de cortafuegos existentes se encuentra en el apéndice A.



En ella podemos ver como iptables posee tablas, estando compuesta cada tabla por un conjunto de cadenas, las cuales a su vez poseen en su interior un conjunto de reglas.

El actual kernel de Linux en Fedora 19 (versión 3.12) posee predefinidas cinco tablas 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 o 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.
- *security*: Permite a módulos de seguridad de Linux, como SELinux, implementar reglas Mandatory Access Control que permitan filtrar paquetes.

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:

- *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.

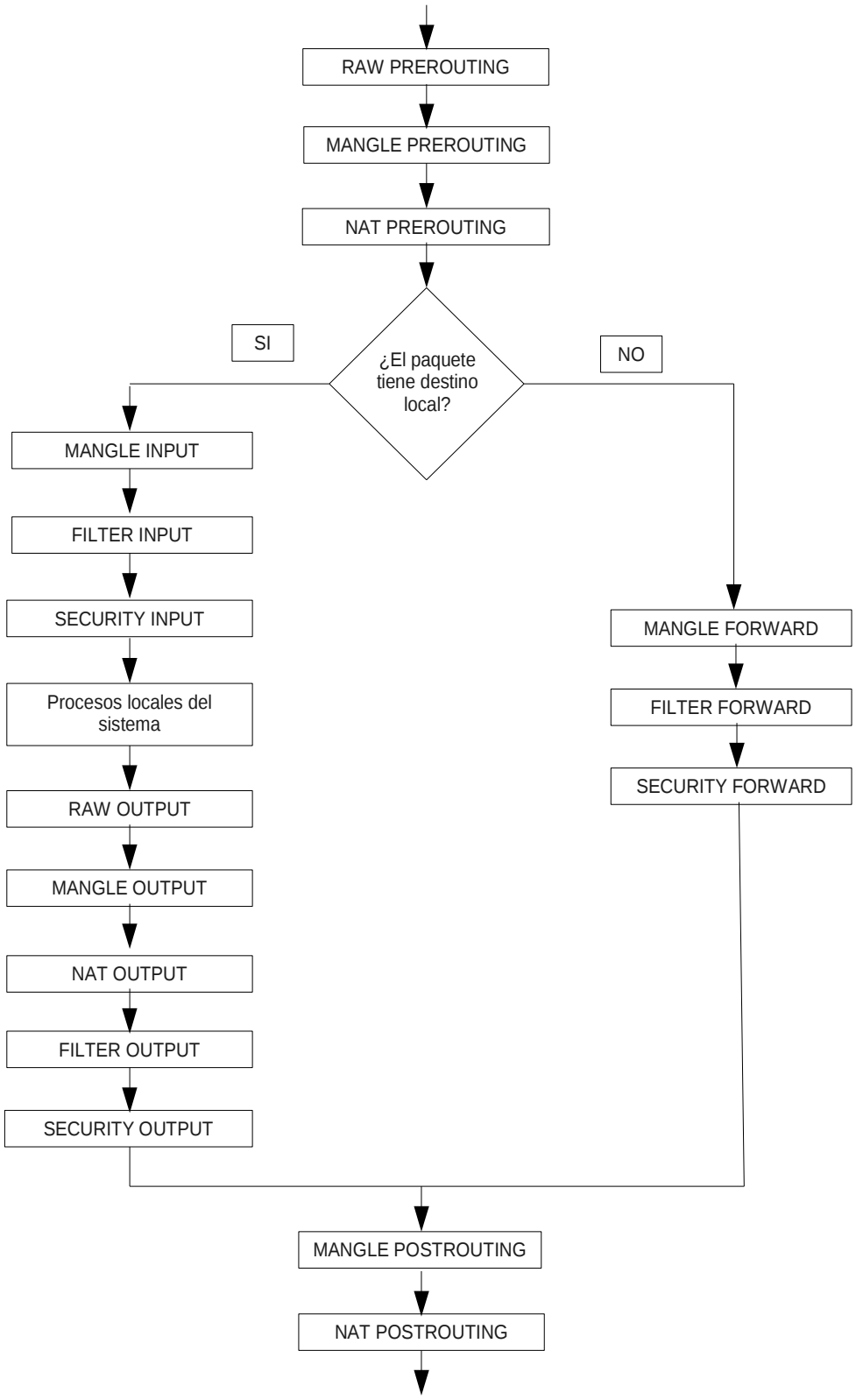
La tabla *raw* tiene las siguientes dos cadenas predefinidas:

- *PREROUTING*: Se aplica a los paquetes recibidos por cualquier dispositivo de red.
- *OUTPUT*: Se aplica a los paquetes generados por un proceso local.

Por último, la tabla *security* posee las siguientes tres 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.

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 o 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, si suponemos que la IP de nuestro ordenador es 192.168.0.1, podemos reenviar el tráfico destinado al puerto TCP 80 de nuestro ordenador a otro ordenador, de IP 192.168.0.2, mediante la regla:

```
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[2].

- *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 aplicada 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*.

[2] Esta acción debe estar soportada explícitamente por el kernel.

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:

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>* indica la acción a realizar, como puede ser añadir una regla, borrar una regla, etc., en 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 complicado.

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 encuentran a continuación:

Comando	Descripción
-A	Añade la regla especificada al final de la cadena especificada.
-C	Chequea una regla y verifica su validez en la cadena especificada. Permite al usuario chequear una regla antes de que sea añadida 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	Borrar 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 (valor -v), que se de en valores exactos y no abreviados con K (miles), M (millones), etc., (valor -x), y que se de en valor numérico de direcciones IP y puertos (valor -n).
-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.

Comando	Descripción
-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 validos son los siguientes[3]:

```
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 puede ser llamada desde alguna de las cadenas predefinidas existentes en la tabla, o desde otra cadena definida previamente por el usuario en la misma 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*, pues al no especificar ninguna tabla se utiliza la tabla por defecto.

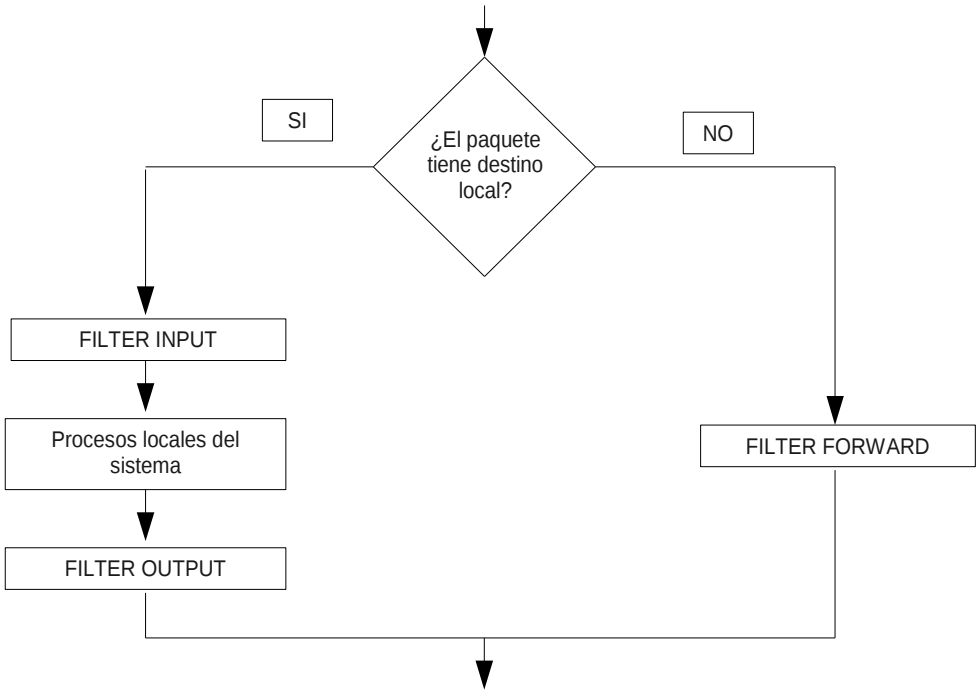
La tabla filter de iptables.

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)[4]. 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:

[3] En estos ejemplos se utilizan parámetros, etc., que todavía no hemos explicado. Son puestos únicamente como ejemplos ilustrativos de cómo utilizar los comandos de iptables y en este punto no pueden ser entendidos en su totalidad.

[4] 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.



Donde puede observarse que todo paquete es examinado por una cadena de reglas, pudiendo decidirse sobre si dicho paquete es o no admitido con destino a un proceso local (*INPUT*), enviado o no desde un proceso local a la red (*OUTPUT*) o reenviado a través de los interfaces de red del ordenador (*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.

Las acciones a realizar sobre el paquete que encaja con una regla pueden ser una de las vistas con anterioridad (ACCEPT, DROP, QUEUE o RETURN) o una de las siguientes acciones, que son extensiones de las acciones generales vistas con anterioridad:

- *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*[5]. 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

[5] 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.

principales opciones son --*log-level* <*nivel*>, que indica el nivel del log[6] 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 durante su inserción (comando -I), añadido (comando –A) o reemplazo (comando -R). Acepta los valores *PKTS* o *BYTES* para especificar que solo se inicialice el contador especificado (paquetes o bytes).

- -*j*: Especifica o bien una acción que debe ejecutarse cuando el paquete coincide con la regla[7], o bien el nombre de una cadena, definida por el usuario mediante el comando -N de iptables, y a cuyas reglas será pasado el análisis de los paquetes. Así, por ejemplo, el comando:

  ```
  iptables -A INPUT -j ACCEPT
  ```

  Que añade, al final de la cadena INPUT, una regla que indica que se acepten todos los paquetes que lleguen a esta regla.

  Mientras que el comando:

  ```
  iptables -A INPUT -j MI_CADENA
  ```

  Especifica que se añada, al final de la cadena INPUT, una regla que indica que los paquetes que lleguen a esta regla sean enviados para su análisis a las reglas de la cadena MI_CADENA, que obviamente ha debido ser definida previamente por el usuario.

  Es necesario resaltar que una regla no tiene que tener obligatoriamente un parámetro -j, pues si una cadena no tiene un parámetro -j, el paquete pasa a ser comprobado por la siguiente regla, siendo el contador de está regla incrementado en una unidad, pues el paquete cumplió la regla.

- -*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

[6] Para ver los posibles niveles de log consultar la página de manual de syslog.conf.
[7] Las acciones validas son tanto las acciones por defecto (*ACCEPT*, *DROP*, *QUEUE* y *RETURN*) como las acciones extendidas (*REJECT* y *LOG*).

existentes de especificación de la máscara: 192.168.0.0/255.255.0.0 o 192.168.0.0/16. Por ejemplo, el comando:

```
iptables -A INPUT -d 192.168.0.0/24 -j ACCEPT
```

Añade una regla, al final de la cadena INPUT, indicando que se admitan todos los paquetes de entrada que tengan como destino la subred 192.168.0.0/24.

La opción *!* puede preceder al parámetro[8], e indica que se aplique la regla a los ordenadores que no correspondan a la especificación del ordenador, etc., realizada. Por ejemplo, el comando:

```
iptables -I OUTPUT 3 ! -d 147.156.222.65 -j DROP
```

Indica que se añada, en tercer lugar de las reglas de la cadena OUTPUT, la regla que indica que se elimine todo el tráfico cuyo destino no se la IP 147.156.222.65.

- -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.

- -*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. Por ejemplo, el comando:

  ```
  iptables -A OUTPUT -f -j DROP
  ```

  Indica que se añada, al final de las reglas de la cadena OUTPUT, la regla que indica que se deniegue la salida de todo paquete que este fragmentado.

- -*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*. Por ejemplo, el comando:

  ```
  iptables -I INPUT 1 -i lo -j ACCEPT
  ```

  Indica que se añada, en primer lugar de las reglas de la cadena INPUT, una regla que especifica que todo el tráfico entrante del interfaz de loopback (lo) sea aceptado.

  Este parámetro admite la opción *!*, que precede al parámetro[9] y especifica que no se aplique la regla al dispositivo de red especificado. Por ejemplo, el comando:

  ```
  iptables -A FORWARD ! -i eth0 -j DROP
  ```

[8] La opción ! puede preceder a la dirección de red, etc., pero se muestra un aviso indicando que se está utilizando una sintaxis antigua y que puede no ser aceptada en posteriores versiones.
[9] La opción ! puede preceder al dispositivo de red indicado, pero se muestra un aviso indicando que se esta utilizando una sintaxis antigua y que puede no ser aceptada en posteriores versiones.

Indica que se añada, al final de las reglas de la cadena FORWARD, una regla que indica que si el tráfico entrante no proviene del interfaz eth0 sea eliminado.

Además, este parámetro admite el símbolo +, que es utilizado como comodín y permite especificar de forma simultanea un grupo de dispositivos de red, como por ejemplo *eth+*, que especificaría *eth0*, *eth1*, etc.

- -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. Por ejemplo, el comando:

  ```
  iptables -A OUTPUT -p icmp -j ACCEPT
  ```

  Indica que se añada, al final de la cadena OUTPUT, una regla que indica que si el protocolo de transporte es ICMP se acepte la salida del paquete.

  Este parámetro posee como opción !, que precede al parámetro[10] e indica que no se aplique esta regla a ese protocolo especificado.

- *-m*: Especifica que se va a utilizar una extensión de los parámetros básicos aquí descritos. Las extensiones de los parámetros básicos pueden corresponder a un determinado protocolo (TCP, UDP e ICMP principalmente) o a otras condiciones. 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*, aunque su uso es posible y no acarrea ningún error de sintaxis, etc. Así, por ejemplo, los comandos:

  ```
  iptables -A INPUT -m tcp -p tcp ...
  ```

  Y

  ```
  iptables -A INPUT -p tcp ...
  ```

  Son equivalentes.

  A continuación veremos las extensiones a los tres protocolos de transporte más utilizados (TCP, UDP e ICMP), así como algunas extensiones generales.

[10] La opción ! puede preceder al protocolo, pero se muestra un aviso indicando que se esta utilizando una sintaxis antigua y que puede no ser aceptada en posteriores versiones.

Extensiones del protocolo 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[11] (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 *:*. Por ejemplo, el comando:

  ```
  iptables -I INPUT 3 -p tcp --dport 80 -j ACCEPT
  ```

  Indica que se coloque en tercer lugar de las reglas de la cadena INPUT, una regla que indica que si el protocolo de transporte es TCP y el puerto destino es el 80 (www) el paquete sea aceptado.

  La extensión puede tener delante de el símbolo *!* que indica que se aplique a los puertos que no están especificados en esta regla[12].

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

- --*syn*: Se aplica a los paquetes TCP que inician una comunicación en este protocolo[13]. Por ejemplo, el comando:

  ```
  iptables -A INPUT -p tcp --syn -j MI_CADENA
  ```

  Indica que se añada al final de la cadena INPUT, una regla que indica que si un paquete posee protocolo de transporte TCP y establece una conexión, se pase su análisis a las reglas de la cadena indicada por MI_CADENA.

  Si se coloca el carácter *!* delante de la extensión se indica que se aplique a los paquetes TCP que no inician la comunicación. Así, por ejemplo, el comando:

  ```
  iptables -A INPUT -p tcp ! --syn -j ACCEPT
  ```

  Indica que se añada al final de la cadena OUTPUT, una regla que indica que si un paquete posee protocolo de transporte TCP y no establece una conexión, se acepte su salida.

- --*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, separadas por un espacio, estando los elementos de cada lista separados por

[11] Si se especifica mediante esta forma, el servicio de red debe encontrarse definido en el fichero */etc/services*.

[12] La opción ! puede preceder al puerto, pero se muestra un aviso indicando que se esta utilizando una sintaxis antigua y que puede no ser aceptada en posteriores versiones.

[13] 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.

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 tomar 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, el comando:

```
iptables -A INPUT -p tcp SYN,RST,ACK SYN -j MI_CADENA
```

Indica que se añada al final de la cadena INPUT una regla que indica que si un paquete posee protocolo de transporte TCP y de los bits de bandera SYN, RST y ACK posee activo el bit de bandera SYN (valor 1) y no activos los bits de bandera RST y ACK (valor 0), el paquete se pase para su análisis a las reglas de la cadena indica por MI_CADENA[14].

Si se especifica antes de la opción 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 las no especificadas para que la regla se cumpla.

- --*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 del protocolo 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 *:*. Por ejemplo, el comando:

```
iptables -I INPUT 3 -p udp --dport 1024: -j ACCEPT
```

Indica que se añada en la tercera posición de las reglas de la cadena INPUT, una regla que indica que si el protocolo de transporte es UDP y el puerto destino es mayor de 1024 se acepte el paquete.

La extensión puede tener delante de la extensión el símbolo *!* e indica que se aplique a los puertos que no están especificados en esta regla[15].

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

[14] Este comando no es más que la especificación de las banderas a analizar para detectar una conexión TCP, por lo que la opción --syn no es más que una forma abreviada, pero muy usada, de utilizar los flags para comprobar una conexión TCP.

[15] La opción ! puede preceder al puerto, pero se muestra un aviso indicando que se esta utilizando una sintaxis antigua y que puede no ser aceptada en posteriores versiones.

Extensiones del protocolo 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[16]. Permite la utilización del símbolo *!* para indicar los paquetes ICMP que no sean del tipo especificado.

Extensiones generales.

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 obligatorio especificar *-m <extensión>*, pudiendo utilizarse varias entradas *-m* en la misma regla para indicar el uso de distintas extensiones[17].

Aunque el conjunto de extensiones existentes es muy amplio[18], 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*. La opción *--mac-source* puede ir precedida del símbolo *!*, lo cual indica que la regla se aplique a las direcciones MAC que no correspondan con la especificada.

- *recent*: Permite crear una lista dinámica de direcciones IP origen de los paquetes y buscar ocurrencias de direcciones IP en la lista. La extensión posee las siguientes opciones:

    - *--name <nombre>*: Nombre de la lista que será utilizada. Si no se especifica ninguna lista se supone la lista *DEFAULT*.

    - *--set*: Añade la dirección IP de origen del paquete a la lista especificada, o si ya se encuentra en la lista, actualiza su entrada. Devuelve siempre verdad (o mentira si delante de la opción se coloca el símbolo *!*).

    - *--rcheck*: Mira si la dirección IP de origen del paquete ya se encuentra en la lista especificada. Devuelve verdad si se encuentra en la lista o mentira en caso contrario. Los valores contrarios son devueltos si se precede la opción del símbolo *!*.

[16] 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.
[17] Obviamente, y por lo dicho aquí, estas extensiones pueden utilizarse junto con las extensiones ligadas a un protocolo específico.
[18] Una lista de las extensiones disponibles puede encontrarse en el manual de iptables, así como una lista más completa en la URL http://www.netfilter.org.

- --*update*: Mira si la dirección IP de origen del paquete se encuentra en la lista especificada, actualiza la entrada y devuelve verdad si la dirección IP se encuentra en la lista, devolviendo mentira en caso contrario. Si se precede la opción con el símbolo *!* se devuelven los valores contrarios.

- --*remove*: Comprueba si la dirección IP de origen del paquete se encuentra en la lista especificada, borrándola y devolviendo el valor verdad, o mentira en caso de que no se encuentre la dirección IP de origen en la lista. Si se precede del símbolo *!* se devuelven los valores contrarios.

- --*seconds* <*segundos*>: Esta opción debe ser usada junto con las opciones *rcheck* o *update* e indica que la regla se cumple si la dirección IP de origen del paquete esta en la lista y fue recibida con anterioridad dentro del número de segundos especificados. Si se precede la opción del símbolo *!* la regla se cumple en caso contrario.

- --*hitcount* <*ocurrencias*>: Esta opción debe ser usada junto con las opciones *rcheck* o *update* e indica que la regla se cumple si la dirección IP de origen del paquete esta en la lista y ha sido recibida un número de veces mayor o igual que el indicado. Si se precede la opción con el símbolo *!* la regla se cumple en caso contrario. Esta opción suele utilizarse junto con la opción *seconds* para indicar un periodo temporal de las ocurrencias.

- --*rttl*: Esta opción debe ser usada juntamente con las opciones *rcheck* o *update* e indica que se cumpla la regla solo si se encuentra en la lista especificada y el TTL del paquete recibido se corresponde con el TTL del paquete que añadió esta dirección IP, con la opción *set*, a la lista. Su uso permite evitar que usuarios malintencionados envíen paquetes con direcciones IP de origen falsas para confundir las reglas y generar ataques de denegación de servicio.

- *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 esta relacionado con una conexión ya existente, aunque no pertenece a la misma, como puede ser el establecimiento de una conexión para el envío de datos en una conexión *FTP* o un mensaje de 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 */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 udp -j REJECT
-A INPUT -p tcp --syn -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[19]:

```
systemctl save iptables.service
```

[19] 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.

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[20].

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.

Los módulos de seguimiento de las conexiones se especifican en la línea *IPTABLES_MODULES*, que consiste en una lista de módulos, separados por espacios, que el sistema debe cargar junto con las iptables para poder realizar el seguimiento de las conexiones existentes. Un ejemplo sencillo es el de un servidor FTP en modo pasivo, donde el servidor envía al cliente, a través de la conexión de control, el puerto al cual el cliente debe conectarse para establecer la conexión de datos. El cortafuegos debe realizar el seguimiento de la conexión de control y guardar la información del cliente y el puerto que se le ha indicado, pues es posible permitir, mediante la opción de estado *RELATED*, que el cliente indicado establezca una conexión con dicho puerto.

El conjunto de módulos existentes es muy amplio, encontrándose los mismos dentro de los subdirectorios que se encuentran a partir del directorio */lib/modules/<versión del kernel>/kernel/net*. 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 *nf_conntrack_ftp.ko* (o *ip_conntrack_ftp.ko* para versiones del kernel anteriores a la 2.6.20), debiendo contener, en cualquier caso, la línea *IPTABLES_MODULES* del fichero */etc/sysconfig/iptables-config* la entrada:

```
IPTABLES_MODULES="ip_conntrack_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.

[20] 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.

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -p udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -j ACCEPT
-A INPUT -p udp -j REJECT
-A INPUT -p tcp --syn -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 se 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*)[21].

- La tercera regla indica que todo el tráfico de entrada udp (*-p udp*) sea rechazado (*-j REJECT*).

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

El resto de tráfico de red es permitido, pues la política por defecto es aceptar (*ACCEPT*) el tráfico de red entrante.

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 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
```

[21] Esta regla, permite la configuración mediante DHCP, la cual a su vez modifica el cortafuegos para permitir 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.

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*)[22].

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

- Las siguientes cinco reglas permiten que cualquier dispositivo de red[23], 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 22*, *--dport 25*, *--dport 80*, *--dport 110* y *--dport 995*)[24].

- Por último, 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.

El resto de tráfico de red, igual que en el ejemplo anterior, es permitido, pues la política por defecto es aceptar (*ACCEPT*) el tráfico de red entrante.

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 udp -j REJECT
```

[22] 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.

[23] El ordenador sobre el que se encuentran estas reglas posee un único dispositivo de red ethernet (eth0).

[24] Con estas cinco reglas se da acceso a los servicios TCP de SSH, sendmail, HTTP POP3 y POP3s, respectivamente.

```
-A INPUT -p tcp --syn -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*)[25].

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*), pues consideramos confiable toda nuestra Intranet, por lo que no limitamos su tráfico.

- 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 nombres mediante el uso del servicio de DNS.

- Las dos reglas siguientes indican, al igual que en los anteriores ejemplos, que se rechace todo el tráfico UDP y que se rechace el tráfico TCP que intente establecer una conexión.

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[26], 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*).

[25] De esta forma, todo el tráfico de red proveniente de la Intranet, y que alcanza Internet por el modem, posee como dirección IP de origen la asignada al modem, pues *MASQUERADE* es un caso particular de enmascaramiento de dirección IP de origen (*SNAT*).

[26] 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.

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

Ejercicios.

1- Deseamos configurar el cortafuegos de un servidor, de dirección IP 147.156.222.65, que ejecuta los servicios de FTP, SSH y HTTP, de forma que permita el acceso del protocolo ICMP, del protocolo UDP a cualquier puerto, y a los servicios TCP de FTP (modo activo y pasivo) y HTTP desde cualquier ordenador de Internet, y al servicio de SSH solo desde los ordenadores de nuestra red de clase B.

2- Un servidor posee dos interfaces de red, eth0 y eth1, de forma que eth0 recibe el tráfico de red de una Intranet y eth1 recibe el tráfico de red proveniente de Internet. El servidor ejecuta los servicios de FTP (en modo activo y pasivo), DNS, HTTP (solo en modo no seguro) y SSH. Configurar el cortafuegos de forma que permita a los ordenadores de la Intranet acceder a todos los servicios ofrecidos por el servidor y a Internet solo permita el acceso a los servicios de FTP y HTTP.

3- Un ordenador dispone de dos interfaces de red, uno público (Internet) de IP 147.156.222.65 y otro privado (Intranet) de IP 192.168.0.1. El ordenador ejecuta los servicios de DNS, FTP, SSH, HTTP y HTTPS. Configurar el cortafuegos (iptables) de forma que:

- Se permitan las consultas DNS a nuestro servidor desde la Intranet y el acceso de las respuestas de DNS de otros servidores de Internet a nuestro servidor.

- Se permitan las conexiones, tanto activas como pasivas, al servidor de FTP, pero solo desde la Intranet.

- Se permitan las conexiones SSH al servidor tanto desde Internet como desde la Intranet.

- Se permita el acceso a los servicios de HTTP y HTTPS pero solo desde Internet.

4- Un ordenador de dirección de red 147.156.222.65 ejecuta los servicios de SSH en el puerto 22 y de HTTP en los puertos 80 y 8080. Configurar el cortafuegos de Linux de forma que permita tan solo que el ordenador pueda enviar o recibir cualquier paquete ICMP, pueda enviar consultas y recibir respuestas de los servidores DNS y permita el acceso al servicio de HTTP del puerto 80 a cualquier ordenador de Internet, limitando el acceso al servicio HTTP del puerto 8080 a la subred 147.156.0.0/16. Por último, debe permitir el acceso al servidor de SSH a cualquier ordenador de la subred 147.156.222.0/23, limitando el acceso al resto de ordenadores de Internet entre las 8 y las 20 horas y de lunes a viernes, utilizando para ello la extensión *time*.

5- Un ordenador ejecuta un servidor de SSH por el que pueden acceder los usuarios del mismo desde Internet. Últimamente hemos observado que se producen, desde ordenadores de Internet, múltiples intentos de conexión al servidor SSH intentando entrar al mismo mediante la prueba sistemática de usuarios y contraseñas (ataque de fuerza bruta). Configurar el cortafuegos de forma que permita tan solo 3 conexiones por

minuto al servidor de SSH desde la misma dirección IP, utilizando para ello la extensión *recent*.

Apéndice A: Tipos de cortafuegos.

De forma general, existen distintos de cortafuegos, según su complejidad. En cualquier caso, todos ellos se construyen en función de dos componentes distintos:

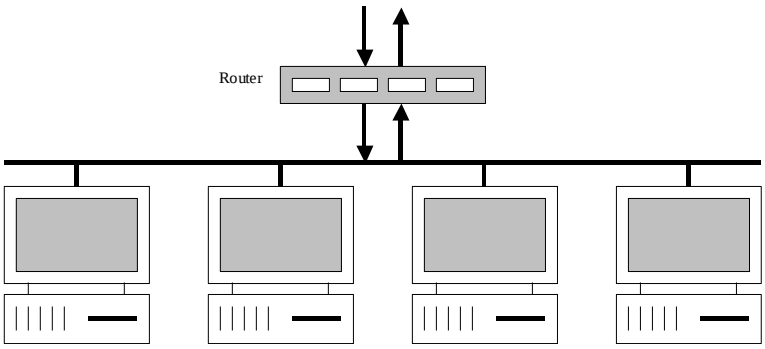
- Router[27]. Dispositivos de comunicación que restringen el flujo de paquetes entre las redes.

- Nodos bastión. Ordenadores dentro del perímetro del cortafuegos y configurados para recibir las conexiones externas y manejarlas de forma apropiada[28].

Existen varios esquemas básicos de configuración de un cortafuegos, desde el más sencillo, en el cual se utiliza un solo router, hasta el más complicado en el cual son utilizadas dos routers y un nodo bastión. Veremos a continuación, de forma sucinta, estos modelos teóricos de cortafuegos.

Cortafuegos de un solo router.

El cortafuegos más sencillo que puede construirse consta de un solo router, según puede verse en la figura adjunta.



En este cortafuegos el router se encarga de filtrar todos los paquetes provenientes de la red exterior hacia la red interior y viceversa. Sus reglas de filtrado son muy sencillas:

- Se bloquean todos los servicios que no se usan, así como todos los paquetes que explícitamente asignan opciones de enrutamiento IP en el origen.

- Se permiten las conexiones TCP para los ordenadores que ofrecen servicios a la red externa y se bloquean las conexiones para todos los demás ordenadores.

[27] Aunque emplearemos la palabra router por la fácil interpretación de la acción que lleva a cabo, cualquier ordenador que desempeñe una función de router, por ejemplo un ordenador con dos interfaces de red, puede ser también utilizado para configurar un cortafuegos.
[28] De forma teórica los usuarios de la red interna no deben tener cuentas en los ordenadores utilizados como nodos bastión.

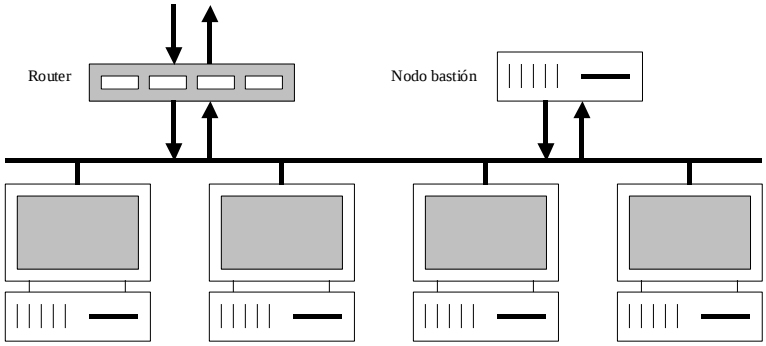
- Se permite a las computadoras de la red interna iniciar conexiones TCP de salida.

Las principales ventajas de este cortafuegos son su simplicidad y bajo coste, además de su flexibilidad, pues es posible cambiar su configuración de forma sencilla y rápida.

Sus principales inconvenientes son la limitación en los registros que es posible almacenar, lo cual dificulta la comprobación de intentos de intrusión, el hecho de que si la seguridad del cortafuegos se compromete, toda la red interna queda comprometida y por último, la no protección contra el contenido de algunas conexiones como el correo electrónico o el FTP.

Cortafuegos de un router y un nodo bastión.

El nivel de seguridad del cortafuegos aumenta si se instala un nodo bastión en la arquitectura del cortafuegos, según puede verse en la figura siguiente.



Igual que en el esquema de cortafuegos anterior, el router se encarga de filtrar todos los paquetes provenientes de la red exterior hacia la red interior y viceversa. Por su parte, el nodo bastión consiste en una computadora que ejecuta el servicio de correo electrónico, Web y en general, cualquier programa de usuario que deba permitir el acceso externo. La programación de esta configuración es más compleja. El router debe:

- Bloquear los servicios que no se desea que crucen el cortafuegos así como los paquetes que tienen enrutamiento IP de origen.

- Bloquear todos los paquetes que tienen como destino la red interna.

- Dejar pasar únicamente los paquetes cuya dirección IP de origen o destino corresponda con la dirección IP del nodo bastión.

Por su parte, el nodo bastión debe:

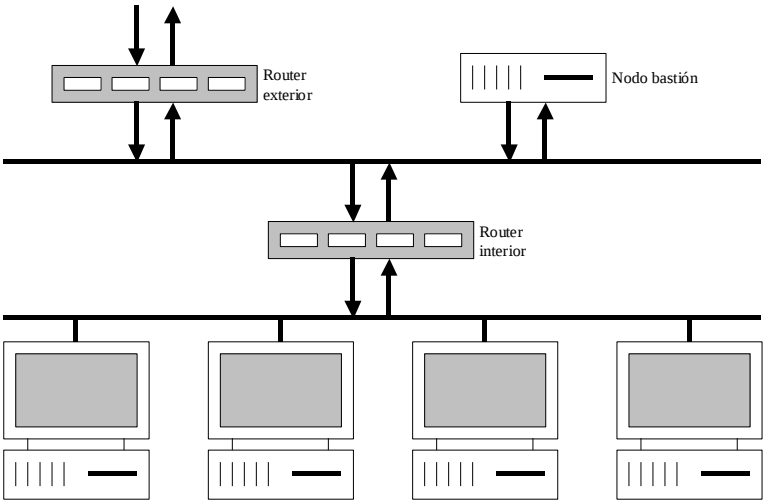
- Ejecutar los servidores que permitan a los usuarios de la red interna usar los servicios de la red externa.

- Actuar como servidor de correo, Web, etc., para la red externa.

Con esta configuración, el router solo deja pasar paquetes entre la red externa y el nodo bastión en cualquiera de los dos sentidos, por lo cual los usuarios internos deben comunicarse con la red externa a través del nodo bastión. De igual forma, los usuarios externos deben establecer primero una comunicación con el nodo bastión antes de poder entrar a cualquier otro ordenador de la red interna.

Cortafuegos de dos router y un nodo bastión.

La seguridad del cortafuegos anterior puede ser mejorada mediante la instalación de otro router que aislé la red interna del nodo bastión, según puede verse en la figura siguiente:



El router interno ofrece una seguridad extra a la red interna. En caso de que un intruso logre saltar la seguridad del router externo e instalarse en el nodo bastión, el router interno protege al resto de ordenadores de la red interna de una agresión desde el nodo bastión. Para ello, la programación del router externo y del nodo bastión es similar al caso anterior, solo que el router externo además de bloquear los paquetes dirigidos a la red interna, excepto el nodo bastión, debe también bloquear los paquetes dirigidos al router interno. Por su parte, el router interno debe:

- Bloquear paquetes para los servicios que no ofrece el nodo bastión y que por tanto no deben cruzar el cortafuegos, así como paquetes con enrutamiento IP de origen.

- Bloquear paquetes dirigidos al router externo.

- Permitir el paso de paquetes con IP origen o destino el nodo bastión y solo para los servicios deseados.

- Bloquear cualquier otro paquete IP.