upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Práctica 5: Seguridad perimetral: filtrado de paquetes

1- Introducción

Cuando hablamos de seguridad perimetral, nos estamos refiriendo a asegurar la frontera entre nuestra red interna y el resto de Internet. El objetivo es restringir o controlar qué datos entran a nuestra organización o salen de ella. La principal ventaja de este tipo de seguridad es que permite al administrador concentrarse en los puntos de entrada. No es necesario, por tanto, que se preocupe de todos y cada uno de los sistemas del interior.

En esta práctica nos iniciaremos en la seguridad perimetral de nuestras redes, conoceremos el concepto de *firewall* y nos familiarizaremos con la utilización de la herramienta *netfilter*/*iptables*, que nos permite implementar *firewalls* en Linux.

2- Seguridad Perimetral: firewalls

El ejemplo más destacable de seguridad perimetral lo constituyen los cortafuegos (*firewalls*). Un cortafuegos es un sistema o grupo de sistemas que hacen cumplir una política de control de acceso entre dos redes, realizando un filtrado de paquetes y/o contenidos. El filtrado funciona de un modo muy sencillo: analiza la cabecera de cada paquete y lo acepta o rechaza según unas reglas predefinidas que hayamos establecido.

La arquitectura de filtrado que emplea Linux se denomina *netfilter* (www.netfilter.org) y está programada en su núcleo (bien como módulo o estáticamente) desde las versiones 2.4.x del mismo. Adicionalmente disponemos de la herramienta *iptables*, un proceso de usuario que nos permite interactuar con el núcleo para que filtre todos los paquetes que le pidamos, y que emplearemos en la realización de esta práctica. Con anterioridad ipfwadmin (en los kernels 2.0.x) e ipchains (en los kernels 2.2.x) eran las encargadas de las tareas de filtrado (se recuerda que las distribuciones actuales implementan el kernel 2.6, por lo tanto *iptables*).

En el apartado siguiente comentaremos con más profundidad la herramienta *iptables* pero, por el momento, veamos un ejemplo de tabla de reglas:

Origen	Destino	Servicio	Acción
ANY	130.20.1.1	80 (HTTP), 21 (FTP)	ACCEPT
212.34.34.34	130.20.1.0/24	22 (SSH)	ACCEPT
101.0.0.1	ANY	ANY	DROP

Según la anterior tabla de reglas, nuestro cortafuegos permitiría ser atravesado por:

1. Paquetes que provengan de cualquier red, cuyos puertos de destino sean 80 o 21 (servicios web y ftp), les permitimos acceder al servidor que implementa dichos servicios.
2. Paquetes cuya IP de origen sea 212.34.34.34 (equipo del administrador de sistemas en su domicilio) y vayan dirigidos al puerto 22 de cualquier equipo de nuestra red (lo que podría sugerir una posible administración remota).

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

3. Niega el acceso (tira el paquete) a cualquier paquete que tenga como dirección origen la 101.0.0.1 (puede ser porque sepamos que desde esa dirección se nos está atacando).

¿Y si llegase un paquete que no coincidiera con ninguna de las anteriores reglas? Entonces se aplicaría lo que se conoce como política por defecto. Generalmente es recomendable una política de denegación por defecto, esto es, denegar todo e ir añadiendo las reglas necesarias para permitir el tráfico o servicios que deseemos (denegado todo lo que no esté aceptado explícitamente). Por supuesto también existe la política contraria, que es permitir todo y denegar explícitamente el tráfico que queramos restringir.

Los cortafuegos son una herramienta potente y tremendamente útil con un grave, aunque subjetivo, inconveniente: dan una falsa sensación de seguridad total que conduce, a menudo, a descuidar otros aspectos que también son importantes. Evidentemente, de nuevo, el problema no está en los *firewall* sino en el factor humano. Los cortafuegos tienen limitaciones sobradamente conocidas y documentadas, pero hay quien tiende a creer que son la panacea de la seguridad cuando no es así: el filtrado, por sí sólo no es suficiente.

Veámoslo con un sencillo ejemplo:

> Uno de los principales puntos débiles del filtrado es que la toma de decisiones sólo se basa en el análisis de las cabeceras. En realidad *netfilter/iptables* no comprenden qué información transmite el paquete y, por tanto, no pueden saber si realmente se trata del tipo de información que indican las cabeceras; es decir, no pueden validar el contenido. Dicho esto, suponga que trabaja en una empresa en la que sólo le permiten acceder al contenido web de Internet (dispone de acceso a Internet pero sólo a través del puerto 80), pero usted desea poder conectarse de forma remota, por *ssh*, con su ordenador de casa para ver qué tal van sus "descargas". Si el administrador de la red de empresa sólo ha activado un sistema de filtrado de paquetes, basta con que en su casa inicie el servicio de *ssh* en el puerto 80, en lugar del 22 por defecto, para poder conectarse. Igualmente, gracias a la opción de redirección de puertos que posee *ssh* (-L), podríamos conectar con cualquier otro servicio que queramos.

Para complementar al sistema de filtrado de paquetes existen los llamados *firewalls de aplicación o firewalls de proxy*, que bloquean o reenvían el tráfico dirigido a servicios específicos (*http*, *ftp*, *ssh*, *telnet*, etc.). Un *proxy* analiza los datos del paquete, y no sólo la cabecera, lo que, en casos como el anterior, nos permitiría detectar que el tráfico que circula por el puerto 80 no corresponde a una conversación *http* válida, descartando los paquetes de la misma.

La suma de filtrado de paquetes y *proxies* de aplicación hace que nuestro cortafuegos sea mucho más robusto y seguro… contra la gran mayoría de atacantes, pero el sistema seguirá presentando vulnerabilidades. En el caso de nuestra conexión por *ssh* al ordenador de casa, una pequeña herramienta de libre distribución como es httptunnel nos permite burlar al *proxy*, tunelando nuestra conexión a través de *http* y haciéndole creer que los paquetes son simples peticiones *http* estándar.

En resumen, siempre existirá alguien capaz de romper la seguridad de nuestros sistemas, pero no luchamos contra él/ella, sino contra una gran mayoría, con muchos menos conocimientos. Para combatir contra él/ella, será vital la monitorización de la actividad en nuestro cortafuegos, la realización de auditorías, el uso de herramientas como los *honeypots* (que nos permiten estudiar las técnicas que emplean nuestros atacantes y que veremos en la práctica siguiente) y, como último recurso, si nuestro sistema se viera comprometido sólo nos quedaría desconectarlo, antes de comprometer también la información sensible que éste pueda contener.

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

Departamento de
Automática y Computación
Automatika eta
Konputazio Saila

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

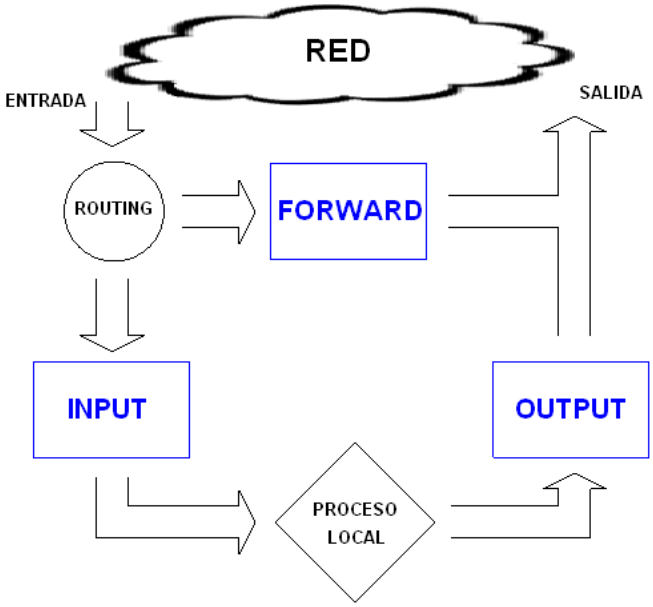
3- Iptables

La estructura de *netfilter* nos permite definir reglas para decirle al núcleo qué hacer con los paquetes que atraviesan los interfaces de red. Las reglas se agrupan en **cadenas** (cada cadena es una lista ordenada de reglas), y las cadenas se agrupan en **tablas** (cada una asociada a un tipo de procesado de paquetes diferente).

Netfilter posee tres tablas: *FILTER*, responsable del filtrado de paquetes (o sea, de permitir o bloquear el paso de los mismos) por la que pasarán todos los paquetes; *NAT*, que establece las reglas para reescribir las direcciones y puertos en los paquetes; y *MANGLE*, que nos permite manipular los paquetes, para ajustar distintas opciones, como por ejemplo, el TOS (Type of Service).

Cada una de ellas tiene sus propias cadenas prefinidas, aunque podemos crear nuevas cadenas, relacionarlas, anidar unas dentro de otras, etc., para conseguir los objetivos de filtrado que necesitemos en nuestro cortafuegos.

Por ejemplo, la tabla *FILTER*, posee tres cadenas predefinidas: INPUT, por la que pasan todos los paquetes que tienen por destino nuestro sistema; OUTPUT, por la que pasan todos los paquetes que tienen como origen nuestro sistema; y FORWARD, por la que pasan todos los paquetes que no tienen nuestro sistema como origen o destino pero son enrutados por él (para que pueda funcionar será necesario tener activado *ip_forwarding*). El proceso de filtrado podría resumirse en el siguiente diagrama de estados (para una versión completa de lo que ocurre con los paquetes cuando atraviesan *netfilter* ver los enlaces de interés al final de la práctica):



Como ya hemos comentado, cada cadena es una lista ordenada de reglas. Cuando un paquete alcanza una cadena, se comprueba secuencialmente si éste se ajusta a alguna de las reglas; la primera con la que encaje se ejecuta (independientemente de que a posteriori haya otras con las que pudiera encajar, por eso hay que tener cuidado en el orden) aplicando las acciones indicadas (ACCEPT, DROP) y si no se ajusta a ninguna, se aplica la política por defecto (en sistemas conscientes de la seguridad, DROP)

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Para facilitar el manejo y creación de tablas, cadenas y reglas existen varias herramientas, pero nosotros vamos a utilizar directamente *iptables*. En Linux, a partir de *man iptables* podemos obtener, como siempre, una información exhaustiva, pero vamos a ver un breve resumen con las claves principales de *iptables*.

Cadenas predefinidas

- Tabla FILTER
  - INPUT Los paquetes que llegan a nuestro sistema
  - OUTPUT Los paquetes que salen de nuestro sistema
  - FORWARD Los paquetes que atraviesan nuestro sistema
- Tabla NAT
  - PREROUTING Para alterar los paquetes entrantes antes del encaminamiento
  - POSTROUTING Para alterar los paquetes salientes tras el encaminamiento

Acciones

- ACCEPT Acepta el paquete
- DROP Elimina el paquete
- LOG Genera LOG de la conexión (aceptada o denegada)
- Otras más específicas como:QUEUE, RETURN, REJECT, MASQUERADE, etc.

Operaciones sobre cadenas

- -L Listar las reglas de una cadena
- -P Establecer la política por defecto de una cadena
- -N Crear una cadena nueva
- -F Borrar todas las reglas de una cadena
- -Z Poner a cero todos los contadores (paquetes y bytes) de una cadena
- -X Borrar una cadena vacía

Operaciones sobre reglas

- -A Añadir una nueva regla a una cadena (la añade al final)
- -I Insertar una nueva regla en una posición determinada
- -D Eliminar reglas, de una posición concreta o que coincidan con una serie de parámetros

Opciones básicas

- -s Especifica una dirección de origen
- -d Especifica una dirección de destino
- -p Especifica un protocolo
- -i Especifica una interfaz de entrada
- -o Especifica una interfaz de salida
- -j Especifica la acción a ejecutar sobre el paquete
- --sport Puerto de origen (con las extensiones TCP y UDP)
- --dport Puerto de destino origen (con las extensiones TCP y UDP)
- -m state –state Estado del paquete, intentando una nueva conexión, paquete de una conexión ya establecida, etc. Las opciones son:
  - -m state --state NEW

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

- -m state --state ESTABLISHED
- -m state --state RELATED
- -m state --state INVALID

Un sinfín de ellas más, y aún más que podemos añadir mediante los módulos y extensiones. Un módulo muy interesante y que tendremos que utilizar en la práctica es el módulo *state*.

El módulo *state* (-m state --state) permite el seguimiento de conexiones controlando el estado del paquete y por lo tanto nos permite tomar decisiones a partir del estado del paquete. Los valores posibles son:

- -m state --state NEW: significa que el paquete inicia una nueva conexión o asociado con una conexión que aun no ha tenido paquetes en ambas direcciones
- -m state --state ESTABLISHED: que se asocia con una conexión que ya ha tenido tráfico en ambas direcciones
- -m state --state RELATED: indica que el paquete comienza una nueva conexión, pero está asociado con una conexión previa existente, por ejemplo una transferenia FTP o un error ICMP
- -m state --state INVALID: significa que el paquete no se puede identifcar por alguna razón, como por ejemplo falta de memoria o errores ICMP que no corresponden a ninguna conexión conocida

Por defecto el estado es NEW si no indicamos lo contrario

Todas las opciones pueden invertirse o negarse, empleando el símbolo [!]. De este modo, la opción *–i eth0* significa "todos los paquetes cuyo interfaz de entrada sea eth0", mientras que *–i ! eth0* significa "todos los paquetes cuyo interfaz de entrada NO sea eth0".

Ejemplos básicos

- Establecemos políticas por defecto

```
$ sudo iptables –P INPUT DROP
$ sudo iptables –P OUTPUT ACCEPT
```

- Impedimos el tráfico entrante y permitimos el saliente:

```
$ sudo iptables -A INPUT -j DROP
$ sudo iptables -A OUTPUT -j ACCEPT
```

- Aceptamos que se conecten a nuestros servidores web(80) y dns (53):

```
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
$ sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
```

- Aceptamos solicitudes del tipo ICMP Request:

```
$ sudo iptables -A INPUT -p icmp -j ACCEPT
```

- Impedimos el acceso de determinadas IPs a nuestra máquina:

```
$ sudo iptables -A INPUT -s 195.76.238.0/24 -j DROP
```

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

```
$ sudo iptables -A INPUT -s 217.116.8.112/29 -j DROP
$ sudo iptables -A INPUT -s 217.116.0.144 -j DROP
$ sudo iptables -A INPUT -s 195.76.172.0/24 -j DROP
$ sudo iptables -A INPUT -s 155.201.0.0/16 -j DROP
```

- Permitir paquetes salientes de conexiones establecidas

```
$ sudo iptables -A OUTPUT -m state --state ESTABLISHED,RELATED  -j ACCEPT
$ sudo iptables -A FORWARD  -m state --state ESTABLISHED,RELATED  -j ACCEPT
```

Como ve, se ha repetido la misma instrucción para varias redes e IPs.

***Nota**: 0.0.0.0/0 hace referencia a cualquier red.*

Scripts IPTables

También se puede observar que *iptables*, al ser un comando, requiere introducir una a una todas las reglas y en el orden requerido. Esto hace que para un cambio pueda ser complicado y costoso recrear toda la estructura de reglas una y otra vez. Para ello se utilizan *scripts shell* añadiendo en dichos scripts las reglas que queremos introducir en el orden establecido. El administrador mantiene el script y lo ejecuta cuando ha realizado cambios en él. Habitualmente el script lo primero que hace es eliminar todas las reglas y volver a añadirlas mediante comandos *iptables*.

Script de ejemplo

A continuación veremos un pequeño *script,* muy básico, de *iptables*.

La primera línea establece el shell con el que se va a ejecutar el script (en este caso /bin/bash). Observar que lo primero que hace el script es limpiar las reglas ya existentes para volver a generar las reglas. Otro aspecto importante es que se establece una política por defecto para cada cadena.

```
==^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^==
#!/bin/bash

#Limpiamos las tablas (solo queremos validar nuestras reglas)
sudo iptables –F
sudo iptables –X
sudo iptables –Z

# Permitimos que se conecten a nuetros servidores web y ssh
sudo iptables -A INPUT -p TCP --dport 80 -j ACCEPT
sudo iptables -A INPUT -p TCP --dport 22 -j ACCEPT

# Permitimos la comunicación con el servidor dns
iptables -A INPUT -p udp --dport 53 -j ACCEPT

# Permitimos todo el tráfico del loopback
sudo iptables –A INPUT –i lo –j ACCEPT
```

```
sudo iptables –A OUTPUT –o lo –j ACCEPT
# Política por defecto. Denegamos todas las entradas
#permitimos todas las salidas y filtramos tráfico de tránsito
sudo iptables -P INPUT DROP
sudo iptables -P OUTPUT ACCEPT
sudo iptables -P FORWARD DROP
==^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^=^==
```

Bastará con ejecutar el script para que se apliquen las reglas correspondientes:

```
$ ./mi_iptables.sh          (suponiendo que mi_iptables.sh sea el nombre del script)
```

Nota: *En adelante utilice scripts para trabajar con iptables.*

4- Practicando con iptables

Vamos a ir probando todo lo que hemos visto a nivel teórico hasta ahora. Para ello vais a necesitar configurar adecuadamente los equipos PC-A, PC-B y PC-C.

Configure el equipo PC-A y PC-B en la misma red y asegúrese que se comunican entre sí (el ping es nuestro mejor aliado). Comprobar además que hay otros servicios activos en dichos equipos, como por ejemplo un servidor Web o el servicio SSH.

Cortafuegos de Servidor

En primer lugar vamos a probar a configurar en PC-B un cortafuegos de servidor, es decir, el cortafuegos establecerá qué se puede hacer en ese servidor y desde donde. Para ello vamos a probar a establecer inicialmente una política por defecto de denegación de todo el tráfico entrante y una política de aceptación de todo el tráfico saliente. Anteriormente se ha descrito como hacerlo.

Comprueba que ya no es posible conectarse al servidor Web de PC-B desde el PC-A ni tampoco establecer una conexión ssh

Ahora autoriza uno a uno los servicios http (puerto TCP 80) y ssh (Puerto TCP 22).

Checkpoint 5.1: Comprueba el funcionamiento y muéstraselo al profesor

Desconfigure el iptables en el equipo PC-B. Establezca una política por defecto de aceptar para el tráfico entrante y saliente. *Compruébelo antes de continuar*.

Activando *ip_forwarding*

Otra de las grandes capacidades que tiene el cortafuegos de Linux es la posibilidad de utilizarlo como cortafuegos de red, es decir, es capaz de analizar y filtrar todo el tráfico que se transmite entre varias redes para las cuales un equipo con Linux es router. Para ello es necesario que el kernel de Linux tenga activado el *ip_forwarding*. *ip_forwarding* permite enrutar un paquete que llegue al equipo tal y como lo hace un router dedicado sólo a ello.

Para comprobar si está activado el enrutado en vuestros equipos basta con comprobar el contenido del siguiente “fichero” (en realidad no es un fichero ya que se encuentra en el filesystem */proc* que es un filesystem virtual que representa el estado del kernel).

```
$ cat /proc/sys/net/ipv4/ip_forwarding
```

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

Si el resultado es 0 el enrutado no está activo, en cambio si el resultado es 1 el enrutado ip Sí está activado.

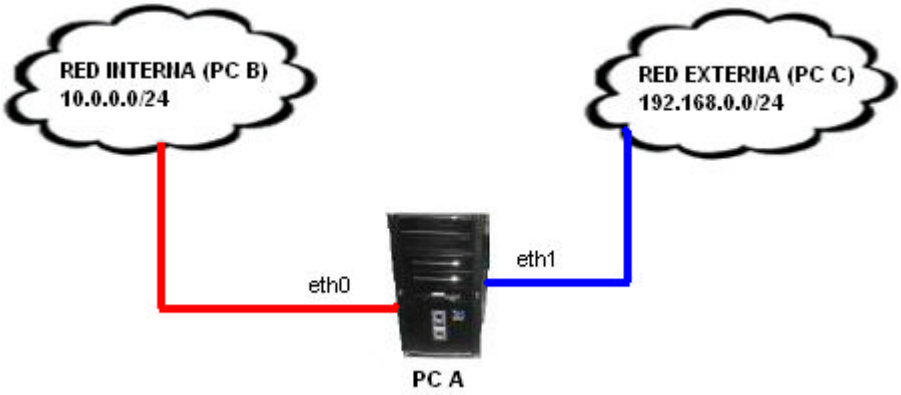
Para activar el enrutado basta con cambiar dinámicamente el contenido de dicho "fichero". Las distribuciones disponen de ficheros de configuración de la red así como de herramientas gráficas que permiten hacerlo, pero en esencia están realizando el cambio de dicho parámetro en el kernel. El comando para hacerlo sería:

```
$ echo 1 > /proc/net/sys/ipv4/ip_forwarding
```

Si lo intentáis se producirá un error ya que no disponéis de permiso para modificar tal parámetro y tampoco *sudo* os dará tal capacidad así que para estos equipos se ha creado un comando que nos permite activar y desactivar el enrutado. El comando es *forwarding* y no lo vais a encontrar en las distribuciones habituales de Linux.

```
$ sudo forwarding si
$ sudo forwarding no
```

Suponga que tenemos la estructura de red que se muestra a continuación.



Configura los 3 equipos para simular éste escenario y activa el enrutado en el equipo PC-A. Utilizar 2 VLANs diferentes para simular las dos redes. Comprueba el funcionamiento.

Nota: Otra vez el ping puede ser nuestro fiel aliado, aunque también es interesante comprobar que se puede acceder desde PC-B a los servicios http y al servicio ssh de PC-C y viceversa. También podréis acceder desde PC-B y PC-C a los servicios http y ssh de PC-A.

Cortafuegos de Red

Utilizando el escenario anterior vamos a activar el cortafuegos en el PC-A para controlar el tráfico entre B y C. Lo que haremos en este apartado es implementar un sencillo cortafuegos que controle los accesos a nuestra red interna. Sólo el equipo PC-A dispondrá de cortafuegos, los equipos PC-B y PC-C no deben tener reglas de filtrado en el kernel.

Nota: Es muy importante asegurarse de que no arrastramos reglas de posibles pruebas anteriores, además de ir escribiendo las reglas que habéis ido añadiendo, y en el orden en el que están, para una mejor verificación de los checkpoints.

Realice las siguientes tareas: (*Tenga en cuenta el estado de los paquetes)*

1. Modifique el comportamiento del cortafuegos para permitir conexiones desde la red interna o desde el propio cortafuegos a máquinas externas e impedir conexiones desde máquinas

Campus de Arrosadía
Arrosadiko Campusa
31006 Pamplona - *Iruñea*
Tfno. 948 169113, Fax. 948 168924
Email: ayc@unavarra.es

externas (a máquinas de la red interna o al cortafuegos). Las conexiones al cortafuegos estarán permitidas desde la red interna.

2. Suponga que en el PC B (red interna) hemos instalado el servidor web del departamento. Haga las modificaciones necesarias para que se puedan realizar conexiones web desde máquinas externas al servidor departamental, pero sólo a él.
3. Suponga que el equipo PC C (red externa) es el equipo que tiene el administrador para realizar control remoto a los equipos de la red interna. Queremos permitir en el cortafuegos que desde el equipo PC C se pueda establecer una conexión ssh al servidor departamental para realizar dicho control remoto, pero sólo desde ese equipo, no se debe permitir tráfico ssh desde otro origen.

Checkpoint 5.2: Muestre el resultado al profesor

Lo que se pretende ahora es que la máquina que actúa como cortafuegos realice también funciones de traducción de direcciones (NAT). Nuestra organización va a contar con una única dirección IP pública (la asociada al interfaz externo de la máquina cortafuegos). Las demás máquinas utilizarán esa misma dirección en las comunicaciones que tengan con el exterior. Añada las reglas correspondientes para que se produzca el NAT

Checkpoint 5.3: Muestre el resultado al profesor

Suponga ahora que volvemos a encontrarnos con un escenario de NAT. Ahora lo que se pretende es redirigir las conexiones web y ssh que van dirigidas a la IP pública (la asociada al interfaz externo de la máquina del cortafuegos) a una máquina de la red interna (en nuestro caso el PC-B).

Checkpoint 5.4: Muestre al profesor cómo lo haría.

Direcciones de interés

- Página oficial de netfilter: http://www.netfilter.org/
- Tutorial de iptables (con versión en español): http://iptables-tutorial.frozentux.net/
- Para una versión más completa del diagrama de estados que representa el recorrido de los paquetes al atravesar el sistema *netfilter*:
  - http://dmiessler.com/images/DM_NF.PNG
  - http://www.shorewall.net/images/Netfilter.png
- Herramientas gráficas de configuración y gestión de cortafuegos, gratuitas y muy potentes: Shorewall http://www.shorewall.net/
- Firewall Builder http://www.fwbuilder.org/
- Lista de Números de Puertos y su uso: http://es.wikipedia.org/wiki/Lista_de_n%C3%BAmeros_de_puerto