Práctica 1.2. TCP y NAT

Objetivos

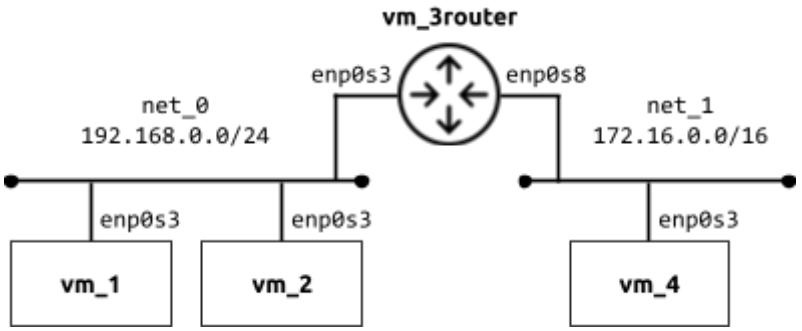
En esta práctica estudiaremos el funcionamiento del protocolo TCP y cómo configurar NAT con `iptables`. Además, veremos una introducción a la seguridad en TCP y estudiaremos algunos parámetros que permiten ajustar el comportamiento del protocolo.

Contenidos

Preparación del entorno para la práctica
Estados de una conexión TCP
Traducción de direcciones (NAT) y reenvío de puertos (*port forwarding*)
Introducción a la seguridad en TCP
Opciones y parámetros de TCP

Preparación del entorno para la práctica

Configuraremos la topología de red que se muestra en la siguiente figura, igual a la empleada en la práctica anterior.



El contenido del fichero de configuración de la topología debe ser el siguiente:

```
prefix vm
netprefix net
machine 1 0 0
machine 2 0 0
machine 3router 0 0 1 1
machine 4 0 1
```

Configurar la red de todas las máquinas de la red según la siguiente tabla:

Máquina	Dirección IPv4	Acciones adicionales
vm_1	`192.168.0.1/24`	Establecer vm_3router como encaminador por defecto
vm_2	`192.168.0.2/24`	Establecer vm_3router como encaminador por defecto
vm_3router	`192.168.0.3/24` (`enp0s3`) `172.16.0.3/16` (`enp0s8`)	Activar el reenvío de paquetes
vm_4	`172.16.0.4/16`	Establecer vm_3router como encaminador por defecto

Finalmente, comprobar la conectividad con la orden ping.

Estados de una conexión TCP

En esta parte usaremos la herramienta Netcat, que permite leer y escribir en conexiones de red. Netcat es muy útil para investigar y depurar el comportamiento de la red en la capa de transporte, ya que permite especificar un gran número de los parámetros de la conexión. Además para ver el estado de las conexiones de red usaremos el comando `ss` (similar a `netstat`, pero más moderno y completo).

Ejercicio 1. Consultar las páginas de manual `nc(1)` y `ss(8)`. En particular, consultar las siguientes opciones de `ss`: `-a`, `-l`, `-n`, `-t` y `-o`. Probar algunas de las opciones para ambos programas para familiarizarse con su comportamiento.

Ejercicio 2. (LISTEN) Abrir un servidor TCP en el puerto 7777 en vm_1 usando el comando `nc -l 7777`. Comprobar el estado de la conexión en el servidor con el comando `ss -tln`. Abrir otro servidor en el puerto 7776 en vm_1 usando el comando `nc -l 192.168.0.1 7776`. Observar la diferencia entre ambos servidores usando `ss`. Comprobar que no es posible la conexión desde vm_1 con `localhost` como dirección destino usando el comando `nc localhost 7776`.

Ejercicio 3. (ESTABLISHED) En vm_2, conectar un cliente al primer servidor arrancado en el ejercicio anterior usando el comando `nc 192.168.0.1 7777`.

- Comprobar el estado de la conexión e identificar los parámetros (dirección IP y puerto) con el comando `ss -tn`.
- Iniciar una captura con Wireshark. Intercambiar un único carácter con el cliente y observar los mensajes intercambiados (especialmente los números de secuencia, confirmación y flags TCP) y determinar cuántos bytes (y número de mensajes) han sido necesarios.

Ejercicio 4. (TIME-WAIT) Cerrar la conexión en el cliente (con `Ctrl+C`) y comprobar el estado de la conexión y el valor del temporizador TIME-WAIT con el comando `ss -tano`.

Ejercicio 5. (SYN-SENT y SYN-RECV) El comando `iptables` permite filtrar paquetes según los flags TCP del segmento con la opción `--tcp-flags` (consultar la página de manual `iptables-extensions(8)`). Usando esta opción:

- Fijar una regla en el servidor (vm_1) que bloquee un mensaje del acuerdo TCP de forma que el cliente (vm_2) se quede en el estado SYN-SENT. Comprobar el resultado con `ss -tan` en el cliente.
- Borrar la regla anterior y fijar otra en el cliente (vm_2) que bloquee un mensaje del acuerdo TCP de forma que el servidor se quede en el estado SYN-RECV. Comprobar el resultado con `ss -tan` en el servidor. Además, esta regla debe dejar al servidor también en el estado LAST-ACK después de cerrar la conexión en el cliente. Usar la opción `-o` de `ss` para determinar cuántas retransmisiones se realizan y con qué frecuencia. Borrar la regla al terminar.

Ejercicio 6. Iniciar una captura con Wireshark. Intentar una conexión a un puerto cerrado del servidor (ej. 7778) y observar los mensajes TCP intercambiados, especialmente los flags TCP.

Traducción de direcciones (NAT) y reenvío de puertos (*port forwarding*)

En esta sección supondremos que la red que conecta vm_3router con vm_4 es pública y que no puede encaminar el tráfico `192.168.0.0/24`. Además, asumiremos que la dirección IP de vm_3router es dinámica.

Ejercicio 7. Configurar la traducción de direcciones dinámica en vm_3router:

- Usando `iptables` en vm_3router, traducir las direcciones (SNAT/*masquerade*) en la interfaz `enp0s8`. Iniciar una captura de Wireshark en cada interfaz de red.
- En vm_1, comprobar la conexión con vm_4 usando la orden `ping`.
- Analizar con Wireshark el tráfico intercambiado, especialmente las direcciones IP origen y destino en ambas redes

Ejercicio 8. Comprueba la salida del comando `conntrack -L` en vm_3router mientras se ejecuta el `ping` del ejercicio anterior. ¿Qué parámetro se utiliza, en lugar del puerto origen, para relacionar las solicitudes con las respuestas?

Ejercicio 9. Acceso a un servidor en la red privada:

- Usando `iptables` en vm_3router, reenviar las conexiones (DNAT) del puerto 80 de vm_3router al puerto 7777 de vm_1. Iniciar una captura de Wireshark en cada interfaz de red.
- En vm_1, arrancar el servidor en el puerto 7777 con `nc`.
- En vm_4, conectarse al puerto 80 de vm_3router con `nc` y comprobar el resultado en vm_1.
- Analizar con Wireshark el tráfico intercambiado, especialmente los puertos y direcciones IP origen y destino en ambas redes.

Introducción a la seguridad en TCP

Diferentes aspectos del protocolo TCP pueden aprovecharse para comprometer la seguridad del sistema. En este apartado vamos a estudiar dos: ataques DoS basados en TCP SYN *flood* y técnicas de exploración de puertos.

Ejercicio 10. El ataque TCP SYN *flood* consiste en saturar un servidor mediante el envío masivo de mensajes SYN.

- Para evitar que el propio atacante responda con un mensaje RST (que liberaría la conexión), bloquear con `iptables` en vm_2 los mensajes SYN+ACK de entrada.
- Usar el comando `hping3` (consultar la página de manual `hping3(8)`) en vm_2 para enviar mensajes SYN al puerto 22 (`ssh`) de vm_1 lo más rápido posible (*flood*).
- Estudiar el comportamiento de vm_1 en términos del número de paquetes recibidos. Comprobar si es posible la conexión al servicio `ssh` desde vm_3router.

Repetir el ejercicio desactivando el mecanismo SYN *cookies* en vm_1 con el comando `sysctl` (parámetro `net.ipv4.tcp_syncookies`).

Nota: Wireshark no debe estar activo cuando se envían paquetes lo más rápido posible (*flooding*).

Ejercicio 11. (Técnica CONNECT) Netcat permite explorar puertos usando la técnica CONNECT que intenta establecer una conexión a un puerto determinado. En función de la respuesta (SYN+ACK o RST), es posible determinar si hay un proceso escuchando.

- Abrir con `nc` un servidor en el puerto 7777 de vm_1.
- Explorar el rango de puertos 7775-7780 con `nc` en vm_2 usando las opciones de exploración (`-z`) y de salida detallada (`-v`).
- Con ayuda de Wireshark, observar los paquetes intercambiados.

Opcional. La herramienta Nmap permite realizar diferentes tipos de exploración de puertos, que emplean estrategias más eficientes (SYN *stealth*, ACK *stealth*, FIN-ACK *stealth*…). Estas estrategias se basan en el funcionamiento del protocolo TCP. Estudiar la página de manual `nmap(1)` (sección PORT SCANNING TECHNIQUES) y emplearlas para explorar los puertos del servidor. Comprobar con Wireshark los mensajes intercambiados.

Opciones y parámetros de TCP

El comportamiento de la conexión TCP se puede controlar con varias opciones que se incluyen en la cabecera en los mensajes SYN y que son configurables en el sistema operativo por medio de parámetros del kernel.

Ejercicio 12. Con ayuda del comando `sysctl`, la página del manual `tcp(7)` y la bibliografía recomendada, completar la siguiente tabla con parámetros que permiten modificar algunas opciones de TCP:

Parámetro del kernel	Propósito	Valor por defecto
`net.ipv4.tcp_window_scaling`		
`net.ipv4.tcp_timestamps`		
`net.ipv4.tcp_sack`		

Ejercicio 13. Iniciar una captura de Wireshark. Abrir un servidor con `nc` en el puerto 7777 de vm_1 y conectar un cliente desde vm_2. Estudiar el valor de las opciones que se intercambian durante la conexión. Variar algunos de los parámetros anteriores (ej. no usar ACKs selectivos) y observar el resultado en una nueva conexión.

Ejercicio 14. Con ayuda del comando `sysctl`, la página del manual `tcp(7)` y la bibliografía recomendada, completar la siguiente tabla con parámetros que permiten configurar el temporizador *keepalive*:

Parámetro del kernel	Propósito	Valor por defecto
`net.ipv4.tcp_keepalive_time`		
`net.ipv4.tcp_keepalive_probes`		
`net.ipv4.tcp_keepalive_intvl`		