Práctica 1.2. Conceptos Avanzados de TCP

Objetivos

En esta práctica estudiaremos el funcionamiento del protocolo TCP. Además veremos algunos parámetros que permiten ajustar el comportamiento de las aplicaciones TCP. Finalmente se consideran algunas aplicaciones del filtrado de paquetes mediante iptables.

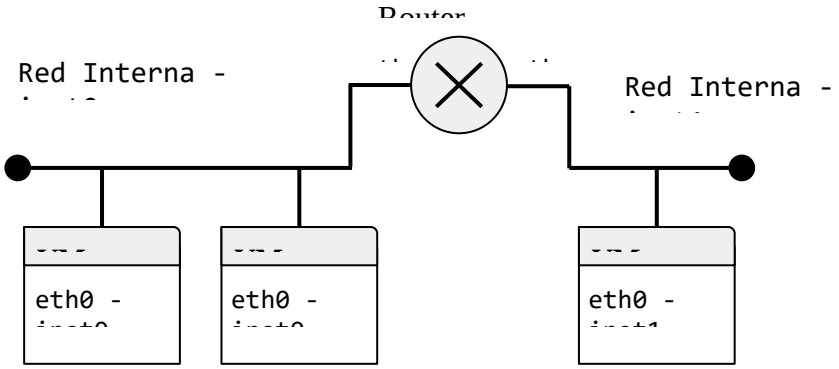
Contenidos

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

Preparación del entorno para la práctica

Configuraremos la topología de red que se muestra en la Figura 1, igual a la empleada en la práctica anterior. En este caso, las direcciones consideradas son IPv4.

Figura 1: Topología y direccionamiento de las redes usadas en la práctica



El siguiente fichero muestra el archivo de configuración de la topología:

```
$ cat pr1.topol
netprefix inet
machine 1 0 0
machine 2 0 0
machine 3 0 0 1 1
machine 4 0 1
```

Finalmente configurar la red de todas las máquinas de la red según la siguiente tabla. Después de configurar todas las máquinas comprobar la conectividad con la orden `ping`.

Máquina	**Dirección IPv4**	**Comentarios**
VM1	`192.168.0.1/24`	Añadir Router como encaminador por defecto
VM2	`192.168.0.2/24`	Añadir Router como encaminador por defecto

VM3 - Router	`192.168.0.3/24 (eth0)` `172.16.0.3/24 (eth1)`	Activar el *forwarding* de paquetes
VM4	`172.16.0.1/24`	Añadir Router como encaminador por defecto

Estados de una conexión TCP

En esta práctica 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 la herramienta `netstat`.

Ejercicio 1. Consultar las páginas de manual para `nc` y `netstat`. 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 VM1 usando el comando `nc`. Comprobar el estado de la conexión en el servidor con el comando netstat.

`nc -l -p 7777 (Añadir "&" al final del comando para ejecutar en segundo plano)`

`netstat -l (para comprobar que el servidor está LISTEN)`
netstat -tln (t para indicar que solo queremos protocolo TCP, l para listar y n para que lo indique con números).

Ejercicio 3. (ESTABLISHED) En VM2, iniciar una conexión cliente al servidor arrancado en el ejercicio anterior. Comprobar el estado de la conexión e identificar los parámetros (dirección IP y puerto) usando el comando `netstat`:

- Iniciar el servidor usando la opción `-s`. Comprobar si es posible la conexión desde VM1 usando como dirección destino `localhost`. Observar la diferencia con el comando anterior (sin opción `-s`) usando `netstat`.

nc -l -s 192.168.0.1 -p 7777& (No escucha en localhost, conexion rechazada)

Si lo haces con la opcion -s escucha en todas las ip (direccion de broadcast), todo 0.0.0.0

- Iniciar el servidor e intercambiar un único carácter con el cliente. Con ayuda de `wireshark`, observar los mensajes intercambiados (especialmente los números de secuencia, confirmación y flags TCP) y determinar cuántos bytes (y número de mensajes) han sido necesarios.

nc [direcciónIpServidor] [Puerto]
nc 192.168.0.1 7777 (En este caso concreto)
Msg intercambiado: "l".
Nº bytes necesario y nº msgs necesarios:

- TCP msg → Flags: SYN. Seq=0. Bytes: 66
- TCP msg → FLags: SYN, ACK. Seq=0, Ack=1. Bytes: 66
- TCP msg → Flags: ACK. Seq=1, Ack=1. Bytes: 66
- TCP msg → Flags: PSH, ACK. Seq=1, Ack=1. Bytes: 66 + 2 Bytes de datos ("l" + "/0")
- TCP msg → Flags: ACK. Seq=1, Ack=3. Bytes: 66
- ARP msg → 42 bytes.
- ARP msg → 60 bytes.

Ejercicio 4. (TIMEWAIT) Cerrar la conexión en el servidor (con Ctl+C) y comprobar el estado de la conexión con `netstat`. Usar la opción `-o` para determinar el valor del temporizador TIMEWAIT.

netstat -o → timewait (14,66/0/0)

Ejercicio 5. (SYN-SENT y SYN-RCVD) El comando `iptables` permite filtrar paquetes según los flags TCP del segmento (opción `--tcp-flags`):

- Fijar una regla que permita filtrar las conexiones en el servidor (VM1) de forma que deje al cliente en el estado SYN-SENT. Comprobar el resultado con el comando `netstat` en el cliente (VM2).

En VM1:

- `iptables -A OUTPUT -p TCP --tcp-flags ALL SYN,ACK -j DROP (en VM1)`
- `iptables -A [APPEND] OUTPUT [DeSalida] -p [protocol] --tcp-flags [TCPFLAGS] -j DROP [Descartar]`

EN VM2:

- `nc 192.168.0.1 7777& (en VM2)`
- `netstat -t (en VM2)`

- Fijar una regla que permita filtrar las conexiones en el cliente (VM2) de forma que deje al servidor en el estado SYN-RCVD. Además esta regla debe dejar al servidor también en el estado LAST-ACK después de cerrar la conexión (con Ctrl+C) en el cliente.

- `iptables -A OUTPUT -p TCP --tcp-flags ALL ACK -j DROP (en VM2)`
- `netstat -t (en VM1)`

- Usando el programa `wireshark`, estudiar los mensajes intercambiados especialmente los números de secuencia , confirmación y flags TCP.

- Con ayuda de `netstat` (usando la opción `-o`) determinar cuántas retransmisiones se realizan y con qué frecuencia.

Nota: La regla debe ser lo suficientemente restrictiva para afectar sólo a las conexiones al servidor. Después de cada ejercicio eliminar las reglas de filtrado.

Para comprobar el listado de reglas aplicadas usar

- `Iptables -L`

Para borrar una regla usar la opción -D (DELETE)

- `iptables -D [DELETE] ...`

Ejercicio 6. Finalmente, intentar una conexión a un puerto cerrado del servidor (ej. 7778) y, con ayuda de la herramienta `wireshark`, observar los mensajes TCP intercambiados, especialmente los flags TCP.

Introducción a la seguridad en el protocolo 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 SYN *flood* y técnicas de exploración de puertos.

Ejercicio 1. El ataque SYN *flood* consiste en saturar un servidor por el envío masivo de mensajes con el *flag* SYN.

- (Cliente VM2) Para evitar que el atacante responda al mensaje SYN+ACK del servidor con un mensaje RST que liberaría los recursos, bloquearemos los mensajes SYN+ACK en el atacante con `iptables`.

`iptables -A INPUT -p TCP --tcp-flags ALL SYN,ACK -j DROP (en VM2)`

- (Cliente VM2) Para enviar paquetes TCP con los datos de interés usaremos el comando `hping3` (estudiar la página de manual). En este caso, enviaremos mensajes SYN al puerto 23 del servidor (`telnet`) lo más rápido posible (*flood*).

```
hping3 -p 23 -S --flood 192.168.0.1 (en VM2)
```

- (Servidor VM1) Estudiar el comportamiento de la máquina, en términos del número de paquetes recibidos. Comprobar si es posible la conexión al servicio `telnet` (23).

```
telnet localhost (no dará conexión, hacerlo justo después del anterior punto)

netstat -t | wc -l => cuenta el número de peticiones en la conexión
```

- (Servidor VM1) Activar la defensa SYN *cookies* en el servidor con el comando `sysctl` (parámetro `net.ipv4.tcp_syncookies`). Comprobar si es posible la conexión al servicio `telnet` (23).

```
sysctl net.ipv4.tcp_syncookies=1
```

Ejercicio 2. (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.

- (Servidor VM1) Abrir un servidor en el puerto 7777.

```
nc  -l -p 7777 &
```

- (Cliente VM2) Explorar de uno en uno, el rango de puertos 7775-7780 usando `nc`, en este caso usar las opciones de exploración (`-z`) y de salida detallada (`-v`). **Nota:** La versión de `nc` instalada en la VM no soporta rangos de puertos. Por tanto, se debe hacer manualmente, o bien, incluir la sentencia de exploración de un puerto en un bucle para automatizar el proceso.

```
nc -zv 192.168.0.1 [puerto]
```

- 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. Estas estrategias (SYN *stealth*, ACK *stealth*, FIN-ACK *stealth*…) son más rápidas que la anterior y se basan en el funcionamiento del protocolo TCP. Estudiar la página de manual de `nmap` (PORT SCANNING TECHNIQUES) y emplearlas para explorar los puertos del servidor. Comprobar con `wireshark` los mensajes intercambiados.

Opciones y parámetros 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. La siguiente tabla incluye alguna de las opciones y las variables asociadas del kernel:

Opción TCP	Parámetro del kernel	Propósito	Valor por defecto
Escalado de la ventana	`net.ipv4.tcp_window_scaling`	Incrementa el límite del tamaño de la ventana de recepción (hasta un máximo de casi 1 GB) Se usa para control de flujo	`net.ipv4.tcp_window_scaling = 1`
Marcas de tiempo	`net.ipv4.tcp_timestamps`	Impide que los números de secuencia TCP sean un problema proporcionando un método alternativo para determinar el orden de un segmento.	`net.ipv4.tcp_timestamps=1`
ACKs selectivos	`net.ipv4.tcp_sack`	Permite al receptor especificar qué bytes se han perdido y cuáles no, por lo que el emisor sólo tiene que retransmitir los bytes perdidos.	`net.ipv4.tcp_sack=1`

Ejercicio 1. Con ayuda del comando `sysctl` y la bibliografía recomendada completar la tabla anterior.

Ejercicio 2. Abrir el servidor en el puerto 7777 y realizar una conexión desde la VM cliente. Con ayuda de `wireshark` 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.

Los siguientes parámetros permiten configurar el temporizador *keepalive*:

Parámetro del kernel	Propósito	Valor por defecto
`net.ipv4.tcp_keepalive_time`	Es el tiempo que una conexión puede estar en silencio.	`net.ipv4.tcp_keepalive_time=7200`
`net.ipv4.tcp_keepalive_probes`	El número de sondas no reconocidas para enviar antes de considerar la conexión muerta.	`net.ipv4.tcp_keepalive_probes=9`
`net.ipv4.tcp_keepalive_intvl`	El intervalo entre las sondas de las subsecuencias.	`net.ipv4.tcp_keepalive_intvl=75`

(encontrado en http://tldp.org/HOWTO/TCP-Keepalive-HOWTO/usingkeepalive.html)

Ejercicio 3. Con ayuda del comando `sysctl` y la bibliografía recomendada completar la tabla anterior.

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

En esta sección supondremos que la red que conecta Router (VM3) con VM4 es pública y que no puede encaminar el tráfico `192.168.0.0/24`. Además, asumiremos que la IP de Router es dinámica.

Ejercicio 1. Configurar la traducción de direcciones dinámica en Router:

- (VM3 - Router) Configurar Router para que haga SNAT (*masquerade*) sobre la interfaz `eth1` usando el comando `iptables`.

iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

- (VM1) Comprobar la conexión entre VM1 y VM4 con la orden `ping`.
- (VM4 y VM1) Usando `wireshark`, determinar la IP origen y destino de los ICMP de Echo request y Echo reply en ambas redes. ¿Qué parámetro se utiliza, en lugar del puerto origen, para relacionar las solicitudes de Echo con las respuestas?

Se utiliza como ip origen la ip del router en lugar de la de la maquina.

Ejercicio 2. Acceso a un servidor en la red privada:

- (VM1) Arrancar el servidor con `nc` en el puerto 7777.

nc -l -p 7777&

- (VM3 - Router) Usando el comando `iptables` reenviar las conexiones del puerto 80 de Router al puerto 7777 de VM1.

iptables -t nat -A PREROUTING -D [direccionIpRouter] -p tcp --dport <puerto receptor> -j DNAT --to <ip final>:<puerto de ip final>

- (VM4) Conectarse al puerto 80 de Router con `nc` y comprobar el resultado en VM1. Analizar el tráfico intercambiado con `wireshark`, especialmente los puertos y direcciones IP origen y destino en ambas redes.

nc [direccionIpRouter] [Puerto]