Práctica 1.2. Conceptos Avanzados de TCP

Realizado por: Estíbaliz Busto Pérez de Mendiguren

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`.

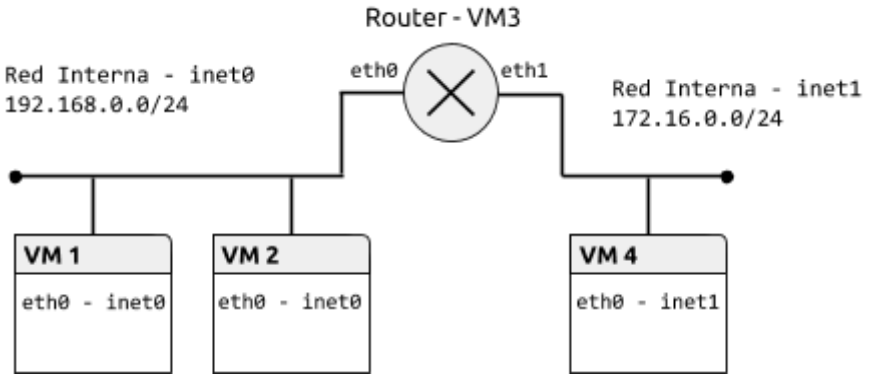
Contenidos

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

```
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

En VM1:
$sudo ip link set eth0 up

$sudo ip a add 192.168.0.1/24 dev eth0

$sudo ip route add default via 192.168.0.3

$ping -c 1 172.16.0.1

En VM2:
$sudo ip link set eth0 up

$sudo ip a add 192.168.0.2/24 dev eth0

$sudo ip route add default via 192.168.0.3

En VM3:

$sudo ip link set eth0 up

$sudo ip link set eth1 up

$sudo ip a add 192.168.0.3/24 dev eth0

$sudo ip a add 172.16.0.2/24 dev eth1

$sudo sysctl net.ipv4.ip_forward=1

En VM2:
$sudo ip link set eth0 up

$sudo ip a add 172.16.0.1/24 dev eth0

$sudo ip route add default via 172.16.0.2

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`. En particular, consultar las siguientes opciones de `netstat`: `-a`, `-l`, `-n`, `-t` y `-o`. Probar algunas de las opciones para ambos programas para familiarizarse con su comportamiento.

- **Nc:** herramienta para el análisis de red.

- **Netcat**: herramienta de red que permite a través de intérprete de comandos y con una sintaxis sencilla abrir puertos TCP/UDP en un HOST.

Ejercicio 2. (LISTEN) Abrir un servidor TCP en el puerto 7777 en VM1 usando el comando `nc -l 7777`. Comprobar el estado de la conexión en el servidor con el comando `netstat`.

En VM1:

$sudo nc -l -p 7777

En otro terminal de VM1:

$sudo netstat -l

Ejercicio 3. (ESTABLISHED) En VM2, iniciar una conexión cliente al 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) usando el comando `netstat -ltn`.

En VM2:

$sudo netstat -t -l -a

- Reiniciar el servidor en VM1 usando el comando `nc -l 192.168.0.1 7777`. Comprobar que no es posible la conexión desde VM1 usando como dirección destino `localhost`. Observar la diferencia con el comando anterior (sin la dirección IP) usando `netstat`.

En VM2:

$sudo nc -l 192.168.0.1 -p 7777

- 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.

En VM2:

$sudo nc 192.168.0.1 7777
Para TCP 66 bytes.

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

En VM2:

$sudo netstat -o -> timewait (52,75/0/0)

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

- Fijar una regla en el servidor (VM1) que bloquee un mensaje del acuerdo TCP de forma que el cliente (VM2) se quede en el estado SYN-SENT. Comprobar el resultado con el comando `netstat` en el cliente.

En VM1:

$sudo iptables -A INPUT -p tcp --syn -j DROP
$sudo nc -l -p 7777

En VM2:

$sudo nc 192.168.0.1 7777
$sudo netstat -t

- Borrar la regla anterior y fijar otra en el cliente que bloquee un mensaje del acuerdo TCP de forma que el servidor se quede 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.

En VM2:

$sudo iptables -A OUTPUT -p tcp --tcp-flags ALL ACK -j DROP

En VM1:

$sudo netstat -t

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

$sudo netstat -o

Retransmisiones 6
Frecuencia ascendente

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.

En VM2:

$sudo nc 192.168.0.1 7778

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 TCP SYN *flood* y técnicas de exploración de puertos.

Ejercicio 1. El ataque TCP SYN *flood* consiste en saturar un servidor mediante el envío masivo de mensajes SYN.

- (Cliente VM2) Para evitar que el atacante responda al mensaje SYN+ACK del servidor con un mensaje RST que liberaría los recursos, bloquear los mensajes SYN+ACK en el atacante con `iptables`.
  $sudo `iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -j DROP`

- (Cliente VM2) Para enviar paquetes TCP con los datos de interés usaremos el comando `hping3` (estudiar la página de manual). En este caso, enviar mensajes SYN al puerto 22 del servidor (`ssh`) lo más rápido posible (*flood*).
  $sudo `hping3 --flood --syn -p 22 192.168.0.1`

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

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

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.
  $sudo 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.

$sudo nc -z -v 192.168.0.1 7775

$sudo nc -z -v 192.168.0.1 7776
nc: inverse lookup failed for 192.168.0.1: Fallo temporal en la resolución del nombre
nc: cannot connect to 192.168.0.1 7776: Conexión rehusada
nc: unable to connect to address 192.168.0.1, service 7776
$sudo nc -z -v 192.168.0.1 7777

- 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`	Aumenta si se puede, el límite del tamaño de la ventana de recepción. Se usa para el control de flujo.	=1, activado
Marcas de tiempo	`net.ipv4.tcp_timestamps`	Determina en qué orden se envían los paquetes en TCP impidiendo que los números de secuencia sean un problema.	=1, activado con offset aleatorio
ACKs selectivos	`net.ipv4.tcp_sack`	Que solo se retransmitan los bytes perdidos y no toda la secuencia a partir del que se ha perdido.	=1, activado

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.

```
$sudo -w net.ipv4.tcp_sack=0
```

Los siguientes parámetros permiten configurar el temporizador *keepalive*:

Parámetro del kernel	Propósito	Valor por defecto
`net.ipv4.tcp_keepalive_time`	Tiempo que una conexión puede estar en silencio.	7200 segundos = 2 horas.
`net.ipv4.tcp_keepalive_probes`	Número máximo de señales keep_alive para ver si la conexión sigue viva.	9 Número de pruebas.
`net.ipv4.tcp_keepalive_intvl`	Número de segundos entre las señales keep_alive.	75 segundos.

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`.
  $sudo `iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE`

- (VM1) Comprobar la conexión entre VM1 y VM4 con la orden `ping`.
  $ping 172.16.0.1

- (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 con las respuestas? Comprueba la salida del comando `conntrack -L` o alternativamente el fichero `/proc/net/nf_conntrack`.
  Ip origen pasa a ser la ip del router.

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

- (VM1) Arrancar el servidor con `nc` en el puerto 7777.
  $sudo nc -l -p 7777
- (VM3 - Router) Usando el comando `iptables` reenviar las conexiones al puerto 80 de Router al puerto 7777 de VM1.
  $sudo `iptables -t nat -A PREROUTING -d 172.16.0.3 -p tcp --dport 80 -j DNAT --to 192.168.0.1:7777`

- (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.
  VM4 piensa que está conectada con VM3, pero los mensajes llegan a VM1. VM1 ve la Ip destino y la de origen.