

Práctica 1.2. TCP y NAT

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 verá cómo configurar NAT con iptables.



Activar el **portapapeles bidireccional** (menú Dispositivos) en las máquinas.

Usar la opción de Virtualbox (menú Ver) para realizar **capturas de pantalla**.

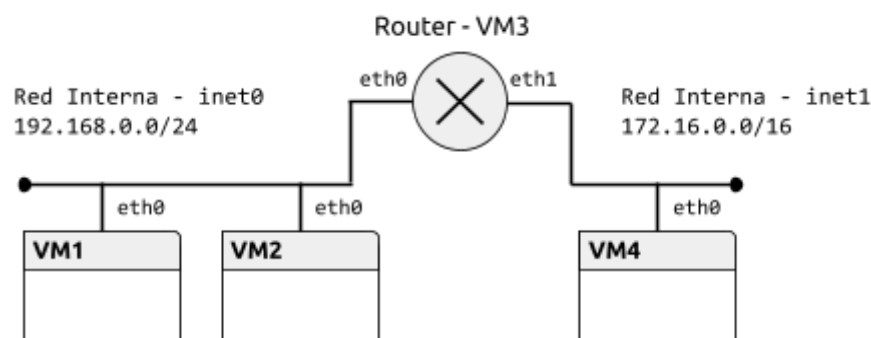
La contraseña del usuario cursoredes es cursoredes.

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 siguiente figura, igual a la empleada en la práctica anterior.



Antes de crear el entorno **eliminar las máquinas virtuales de ASOR de VirtualBox**, junto con todos sus archivos. Después **importar el servidor** usando /mnt/DiscoVMs/ASOR/ASOR-FE.ova. Finalmente **crear la topología con vtopo1**.

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
Router - VM3	192.168.0.3/24 (eth0) 172.16.0.3/16 (eth1)	Activar el <i>forwarding</i> de paquetes
VM4	172.16.0.4/16	Añadir Router como encaminador por defecto

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 de nc y ss. 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 VM1 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 VM1 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 VM1 con localhost como dirección destino usando el comando nc localhost 7776.

State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
LISTEN	0	100	127.0.0.1:25	*.*
LISTEN	0	10	192.168.0.1:7776	*.*
LISTEN	0	10	*:7777	*.*
LISTEN	0	128	*:111	*.*
LISTEN	0	128	*:22	*.*
LISTEN	0	128	127.0.0.1:631	*.*
LISTEN	0	100	:::1:25	:::*
LISTEN	0	10	:::7777	:::*
LISTEN	0	128	:::111	:::*
LISTEN	0	128	:::22	:::*
LISTEN	0	128	:::1:631	:::*

Ejercicio 3. (ESTABLISHED) En VM2, iniciar una conexión 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.

```
State Recv-Q Send-Q Local Address:Port Peer Address:Port
ESTAB 0 0 192.168.0.2:59500 192.168.0.1:7777
```

No.	Time	Source	Destination	Protoc	Length	Info
1	0.00000000	192.168.0.2	192.168.0.1	TCP	68	59500 > cbt [PSH, ACK] Seq=1 Ack=1 Win=229 Len=2 TSval=1183866 TSecr=1092033
2	0.00088611	192.168.0.1	192.168.0.2	TCP	66	cbt > 59500 [ACK] Seq=1 Ack=3 Win=227 Len=0 TSval=1185615 TSecr=1183866

Ejercicio 4. (TIME-WAIT) Cerrar la conexión en el cliente (con Ctrl+C) y comprobar el estado de la conexión usando `ss -tan`. Usar la opción `-o` de `ss` para observar el valor del temporizador TIME-WAIT.

```
State Recv-Q Send-Q Local Address:Port Peer Address:Port
FIN-WAIT-2 0 0 192.168.0.1:7777 192.168.0.2:59500
```

```
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port
tcp FIN-WAIT-2 0 0 192.168.0.1:cbt 192.168.0.2:59500 timer:(timewait,30sec,0)
```

```
TIME-WAIT 0 0 192.168.0.2:47750 192.168.0.1:7777
```

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`). Usando esta opción:

- 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 `ss -tan` en el cliente.
- Borrar la regla anterior y fijar otra en el cliente (VM2) 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.

```
vm1> iptables -A INPUT -p tcp --tcp-flags ALL SYN -j DROP
```

```
vm2:
State Recv-Q Send-Q Local Address:Port Peer Address:Port
...
SYN-SENT 0 1 192.168.0.2:59504 192.168.0.1:7777
...
```

```
vm1> iptables -D INPUT -p tcp --tcp-flags ALL SYN -j DROP
```

```
vm2> iptables -A INPUT -p tcp --tcp-flags ACK ACK -j DROP
```

```
vm1:
State Recv-Q Send-Q Local Address:Port Peer Address:Port
...
SYN-RECV 0 0 192.168.0.1:7777 192.168.0.2:47738
```

...

```
vm2> iptables -D INPUT -p tcp --tcp-flags ACK ACK -j DROP
```

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.

No.	Time	Source	Destination	Protoc	Length	Info
1	0.00000000	192.168.0.2	192.168.0.1	TCP	74	40546 > interwise [SYN] Seq=0 win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=6988257 TSecr=0 WS=128
2	0.00019964	192.168.0.1	192.168.0.2	TCP	60	interwise > 40546 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

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 7. 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 con un mensaje RST (que liberaría la conexión), bloquear con iptables los mensajes SYN+ACK del servidor.
- (Cliente VM2) Usar el comando `hping3` (estudiar la página de manual) para enviar mensajes SYN al puerto 22 del servidor (ssh) lo más rápido posible (*flood*).
- (Servidor VM1) Estudiar el comportamiento de la máquina, en términos del número de paquetes recibidos. Comprobar si es posible la conexión al servicio ssh desde Router.

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

```
sudo iptables -A INPUT -p tcp --tcp-flags ACK ACK -j DROP
sudo hping3 -S --flood -V -p 22 192.168.0.1
```

vm1 recibe demasiados paquetes.

La conexión desde router no funciona correctamente:

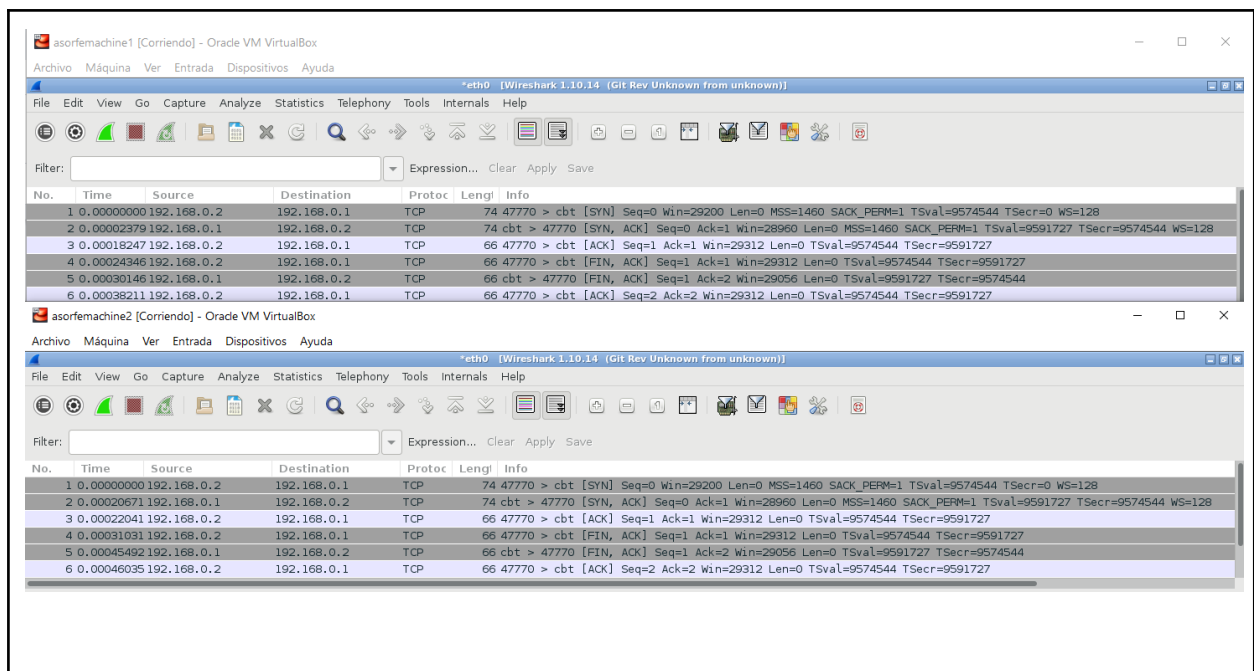
```
SSH-2.0-OpenSSH_7.4
Protocol mismatch.
Ncat: Broken pipe.
```

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

Ejercicio 8. (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.
- (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`).
- Con ayuda de Wireshark, observar los paquetes intercambiados.

```
nc -vz 192.168.0.1 7775-7780
```



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*...) 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 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 9. Con ayuda del comando `sysctl` y la bibliografía recomendada, completar la siguiente tabla con parámetros que permiten modificar algunas opciones de TCP:

Parámetro del kernel	Propósito	Valor por defecto
<code>net.ipv4.tcp_window_scaling</code>	Cambiar el tamaño máximo de la ventana de recepción. 0 fijo, 1 variable	1 según el manual
<code>net.ipv4.tcp_timestamps</code>	Permite configurar la visibilidad de timestamps: 0 no, 1 sí, 2 sí (sin offset aleatorio)	1 según el manual
<code>net.ipv4.tcp_sack</code>	Booleano, permitir el envío de SACKs	1 según el manual

Ejercicio 10. Iniciar una captura de Wireshark. Abrir el servidor en el puerto 7777 y realizar una conexión desde la VM cliente. 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.

Todo realizado en el cliente:
`sudo sysctl net.ipv4.tcp_sack=1`

```
sudo sysctl net.ipv4.tcp_timestamps=0
```

```
▼ Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale
  ▶ Maximum segment size: 1460 bytes
  ▶ No-Operation (NOP)
  ▶ No-Operation (NOP)
  ▼ TCP SACK Permitted Option: True
    Kind: SACK Permission (4)
    Length: 2
  ▶ No-Operation (NOP)
  ▶ Window scale: 7 (multiply by 128)
```

```
sudo sysctl net.ipv4.tcp_sack=0
```

```
sudo sysctl net.ipv4.tcp_timestamps=1
```

```
▼ Options: (20 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), Timestamps, No-Operation (NOP), Window scale
  ▶ Maximum segment size: 1460 bytes
  ▶ No-Operation (NOP)
  ▶ No-Operation (NOP)
  ▶ Timestamps: TSval 11428818, TSecr 0
  ▶ No-Operation (NOP)
  ▶ Window scale: 7 (multiply by 128)
```

Ejercicio 11. Con ayuda del comando `sysctl` y la bibliografía recomendada, completar la siguiente tabla con parámetros que permiten configurar el temporizador *keepalive*:

Parámetro del kernel	Propósito	Valor por defecto
<code>net.ipv4.tcp_keepalive_time</code>	Los segundos que debe seguir idle una conexión antes de que el formato tcp empiece a mandar <i>keepalive probes</i>	7200
<code>net.ipv4.tcp_keepalive_probes</code>	Máximo número a mandar de <i>keepalive probes</i> sin respuesta antes de rendirse y terminar conexión	9
<code>net.ipv4.tcp_keepalive_intvl</code>	Segundos entre envíos de <i>keepalive probes</i>	75

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

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

Ejercicio 12. Configurar la traducción de direcciones dinámica en Router:

- (Router) Usando `iptables`, configurar Router para que haga SNAT (*masquerade*) sobre la interfaz `eth1`. Iniciar una captura de Wireshark en cada interfaz de red.
- (VM1) Comprobar la conexión con VM4 usando la orden `ping`.
- (Router) Analizar con Wireshark el tráfico intercambiado, especialmente los puertos y direcciones IP origen y destino en ambas redes

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4
```

3 0.00028130 1.2.3.4

172.16.0.4

ICMP

98 Echo

Ejercicio 13. Comprueba la salida del comando `conntrack -L` o, alternatively, el contenido del fichero `/proc/net/nf_conntrack` en Router mientras se ejecuta el ping del ejercicio anterior. ¿Qué parámetro se utiliza, en lugar del puerto origen, para relacionar las solicitudes con las respuestas?

```
icmp 1 29 src=192.168.0.1 dst=172.16.0.4 type=8 code=0 id=5443 [UNREPLIED] src=172.16.0.4
dst=1.2.3.4 type=0 code=0 id=5443 mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 1 flow entries have been shown.
```

Ejercicio 14. Acceso a un servidor en la red privada:

- (Router) Usando `iptables`, reenviar las conexiones (DNAT) del puerto 80 de Router al puerto 7777 de VM1. Iniciar una captura de Wireshark en cada interfaz de red.
- (VM1) Arrancar el servidor en el puerto 7777 con `nc`.
- (VM4) Conectarse al puerto 80 de Router con `nc` y comprobar el resultado en VM1.
- (Router) Analizar con Wireshark el tráfico intercambiado, especialmente los puertos y direcciones IP origen y destino en ambas redes.

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 \
-j DNAT --to 192.168.0.1:7777
```