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](#_14a3ftqman5y)

[Estados de una conexión TCP](#_z8kpop4tqxmq)

[Introducción a la seguridad en el protocolo TCP](#_72odsoledxbm)

[Opciones y parámetros TCP](#_rngdrx1kptex)

[Traducción de direcciones (NAT) y reenvío de puertos (*port forwarding*)](#_pgwskimsb2ry)

# 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:

| 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 *forwarding* 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.

| *LISTEN 0 10 192.168.0.1:7776 \*:\**  *LISTEN 0 10 \*:7777 \*:\**  *[cursoredes@localhost ~]$ nc localhost 7776*  *Ncat: Connection refused.* |
| --- |

***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.

| *[cursoredes@localhost ~]$ ss -tn*  *State Recv-Q Send-Q Local Address:Port Peer Address:Port*  *ESTAB 0 0 192.168.0.2:40502 192.168.0.1:7777*  *1 0.000000000 192.168.0.2 192.168.0.1 TCP 68 Seq=1 Ack=1 Flags: 0x018 (PSH, ACK)*  *2 0.000355195 192.168.0.1 192.168.0.2 TCP 66 Seq=1 Ack=3 Flags: 0x010 (ACK)*  *Nº mensajes: 2*  *Nª Bytes: 134* |
| --- |

***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.

| *TIME-WAIT 0 0 192.168.0.2:40502 192.168.0.1:7777*  *[cursoredes@localhost ~]$ ss -o state TIME-WAIT*  *Netid Recv-Q Send-Q Local Address:Port Peer Address:Port*  *tcp 0 0 192.168.0.2:40506 192.168.0.1:cbt timer:(timewait,40sec,0)* |
| --- |

***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.

| *[root@localhost ~]# iptables -A INPUT -p tcp --tcp-flags ALL SYN -j DROP*  *[cursoredes@localhost ~]$ ss -tan*  *State Recv-Q Send-Q Local Address:Port Peer Address:Port*  *LISTEN 0 100 127.0.0.1:25 \*:\**  *LISTEN 0 128 \*:111 \*:\**  *LISTEN 0 128 \*:22 \*:\**  *LISTEN 0 128 127.0.0.1:631 \*:\**  *SYN-SENT 0 1 192.168.0.2:40510 192.168.0.1:7777*  *LISTEN 0 100 ::1:25 :::\**  *LISTEN 0 128 :::111 :::\**  *LISTEN 0 128 :::22 :::\**  *LISTEN 0 128 ::1:631 :::\**  *Ahorita borramos la regla con:*  *[root@localhost ~]# iptables -D INPUT 1*  *[root@localhost ~]# iptables -A OUTPUT -p tcp --tcp-flags ALL ACK -j DROP*  *[root@localhost ~]# ss -tan*  *State Recv-Q Send-Q Local Address:Port Peer Address:Port*  *LISTEN 0 100 127.0.0.1:25 \*:\**  *LISTEN 0 10 \*:7777 \*:\**  *SYN-RECV 0 0 192.168.0.1:7777 192.168.0.2:40516*  *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 :::\**  *[root@localhost ~]# ss -tan*  *State Recv-Q Send-Q Local Address:Port Peer Address:Port*  *LISTEN 0 100 127.0.0.1:25 \*:\**  *LISTEN 0 128 \*:111 \*:\**  *LISTEN 0 128 \*:22 \*:\**  *LISTEN 0 128 127.0.0.1:631 \*:\**  *LAST-ACK 0 1 192.168.0.1:7777 192.168.0.2:40522*  *LISTEN 0 100 ::1:25 :::\**  *LISTEN 0 128 :::111 :::\**  *LISTEN 0 128 :::22 :::\**  *LISTEN 0 128 ::1:631 :::\**  *6 retransmisiones cada una de ellas con más tiempo de retardo que la anterior* |
| --- |

***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.

| *Copiar una captura de pantalla de Wireshark.* |
| --- |

# 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).

| *Copiar los comandos iptables y hping3 utilizados. Describir el comportamiento de la máquina con y sin el mecanismo SYN cookies.*  ***1)***  *[root@localhost ~]#* ***iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -j DROP***  ***2)***  *[root@localhost ~]#* ***hping3 --flood -p 22 -S 192.168.0.1***  *HPING 192.168.0.1 (eth0 192.168.0.1): S set, 40 headers + 0 data bytes*  *hping in flood mode, no replies will be shown*  *^C*  *--- 192.168.0.1 hping statistic ---*  *3256497 packets transmitted, 0 packets received, 100% packet loss*  *round-trip min/avg/max = 0.0/0.0/0.0 ms*  ***3)***  *[root@localhost ~]# nc 192.168.0.1 22*  *SSH-2.0-OpenSSH\_7.4*  *Desactivando el servicio con el comando mostrado a continuación deja de ser posible conectarse:*  *[root@localhost ~]# sysctl net.ipv4.tcp\_syncookies=0*  *[root@localhost ~]# nc 192.168.0.1 22*  ***Ncat: Connection timed out.*** |
| --- |

**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.

| *Copiar los comandos nc utilizados y su salida.*  ***1)***  *[root@localhost ~]# nc -l 7777*  ***2)***  *[root@localhost ~]#* ***nc -z -v 192.168.0.1 7777***  *Ncat: Version 7.50 ( https://nmap.org/ncat )*  *Ncat: Connected to 192.168.0.1:7777.*  *Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.*  *Para el resto de puertos:*  *Ncat: Version 7.50 ( https://nmap.org/ncat )*  *Ncat: Connection refused.*  *[cursoredes@localhost ~]$*  ***3)*** |
| --- |

***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 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** |
| --- | --- | --- |
| net.ipv4.tcp\_window\_scaling | Incrementa el tamaño de la ventana de recepción a un valor superior a 64K | Activado |
| net.ipv4.tcp\_timestamps | Guarda una secuencia de caracteres indicando la temporalidad en la que fue enviado el mensaje | Activado |
| net.ipv4.tcp\_sack | Permite la confirmación de un segmento en específico fuera de orden | Activado |

***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.

| *Copiar una captura de pantalla de Wireshark donde se muestren las opciones TCP.*  *Timestamp:*    *Window Scale:*    *Desactivando timestamp obtenemos lo siguiente:* |
| --- |

***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** |
| --- | --- | --- |
| net.ipv4.tcp\_keepalive\_time | Se utiliza para que una vez expirado se manden señales para cerrar la conexión en caso de que no exista tráfico de datos | 7200 s |
| net.ipv4.tcp\_keepalive\_probes | Proporcionan un método para eliminar sockets inactivos y notificar a las aplicaciones sobre pares que no responden a través de una conexión TCP | 9 sondas |
| net.ipv4.tcp\_keepalive\_intvl | El intervalo entre las sondas keepalive subsiguientes, independientemente de lo que haya intercambiado la conexión mientras tanto | 75 s |

# 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

| ***Router***  *[root@localhost ~]#* ***iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE***  ***VM1***  *[root@localhost ~]#* ***ping 172.16.0.4***  *Podemos observar en la captura de paquetes de la interfaz eth0 que aparecen ambas direcciones públicas y privadas:*    *Por otra parte las direcciones Ip en la interfaz eth1 son todas públicas indicando que la traducción mediante NAPT-Masquerade son correctas:* |
| --- |

***Ejercicio 13*.** Comprueba la salida del comando conntrack -L o, alternativamente, 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?

| *[root@localhost ~]#* ***conntrack -L***  *icmp 1 29 src=192.168.0.1 dst=172.16.0.4 type=8 code=0 id=4975 src=172.16.0.4 dst=172.16.0.3 type=0 code=0 id=4975 mark=0 use=1*  *conntrack v1.4.4 (conntrack-tools): 1 flow entries have been shown.*  *Se utiliza como parámetro el número de identificación del ICMP recibido (4975)* |
| --- |

***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.

| *[root@localhost ~]#* ***iptables -t nat -A PREROUTING -d 172.16.0.3 -p tcp --dport 80 -i eth1 -j DNAT --to 192.168.0.1:7777*** |
| --- |