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.

|  |  |
| --- | --- |
|  | **Para cada ejercicio**, se tienen que proporcionar los **comandos utilizados con sus correspondientes salidas**, las **capturas de pantalla de Wireshark** **realizadas**, y la **información requerida de manera específica**.  **Activar el portapapeles bidireccional en las máquinas** (menú Dispositivos) para copiar la salida de los comandos. Las capturas de pantalla se realizarán usando también Virtualbox (menú Ver).  Las **credenciales de la máquina virtual** son: usuario cursoredes, con contraseña 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.

|  |
| --- |
| *Adjuntar la salida del comando ss correspondiente a los servidores.*  [cursoredes@localhost ~]$ ss -tln  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 :::\*  [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.

|  |
| --- |
| *Adjuntar la salida del comando ss correspondiente a la conexión y una captura de pantalla de Wireshark.*  [cursoredes@localhost ~]$ ss -tn  State Recv-Q Send-Q Local Address:Port Peer Address:Port  ESTAB 0 0 192.168.0.2:40668 192.168.0.1:7777 |

***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.

|  |
| --- |
| *Adjuntar la salida del comando ss correspondiente a la conexión.*  [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 \*:\*  TIME-WAIT 0 0 192.168.0.2:40672 192.168.0.1:7777  LISTEN 0 100 ::1:25 :::\*  LISTEN 0 128 :::111 :::\*  LISTEN 0 128 :::22 :::\*  LISTEN 0 128 ::1:631 :::\*  [cursoredes@localhost ~]$ ss -tano  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 \*:\*  TIME-WAIT 0 0 192.168.0.2:40672 192.168.0.1:7777 timer:(timewait,49sec,0)  LISTEN 0 100 ::1:25 :::\*  LISTEN 0 128 :::111 :::\*  LISTEN 0 128 :::22 :::\*  LISTEN 0 128 ::1:631 :::\* |

***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). 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 que bloquee un mensaje del acuerdo TCP de forma que el servidor se quede en el estado SYN-RCVD. 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 (con Ctrl+C) en el cliente. Usar la opción -o de ss para determinar cuántas retransmisiones se realizan y con qué frecuencia.

|  |
| --- |
| *Adjuntar los comandos iptables utilizados y la salida del comando ss correspondiente a las conexiones.*  (VM1)[cursoredes@localhost ~]$ sudo iptables -A OUTPUT -p tcp --tcp-flags ALL SYN,ACK -j DROP  (VM2)[cursoredes@localhost ~]$ nc 192.168.0.1 7777  Ncat: Connection timed out.  (VM2)[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 1 192.168.0.2:40680 192.168.0.1:7777  (VM1)[[cursoredes@localhost ~]$ sudo iptables -D OUTPUT 1 or iptables -F OUTPUT  (VM2)[cursoredes@localhost ~]$ sudo iptables -A OUTPUT -p tcp --tcp-flags ALL ACK -j DROP  (VM1)[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 10 \*:7777 \*:\*  SYN-RECV 0 0 192.168.0.1:7777 192.168.0.2:40722  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 :::\*  [cursoredes@localhost ~]$ ss -tano  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 192.168.0.2:40722 timer:(on,40sec,1)  LISTEN 0 100 ::1:25 :::\*  LISTEN 0 128 :::111 :::\*  LISTEN 0 128 :::22 :::\*  LISTEN 0 128 ::1:631 :::\* |

***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.

|  |
| --- |
| *Adjuntar 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 al mensaje SYN+ACK del servidor con un mensaje RST que liberaría los recursos, bloquear los mensajes SYN+ACK en el atacante con iptables.
* (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*).
* (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).

|  |
| --- |
| *Adjuntar los comandos iptables y hping3 utilizados. Describir el comportamiento de la máquina con y sin el mecanismo SYN cookies.*  (VM1)[cursoredes@localhost ~]$ sudo sysctl net.ipv4.tcp\_syncookies=1  (VM2)[cursoredes@localhost ~]$ sudo iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -j DROP  (VM2)[cursoredes@localhost ~]$ sudo hping3 --flood --rand-source -p 22 --syn 192.168.0.1  (VM1)[cursoredes@localhost ~]$ sudo sysctl net.ipv4.tcp\_syncookies=0  Se puede observar que si se activa el mecanismo SYN cookies evita el ataque TCP SYN *flood no permitiendo el envío de paquetes para saturar al servidor (VM1) desde el atacante (VM2)* |

***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). **Nota:** La versión de nc instalada no soporta rangos de puertos.
* Con ayuda de Wireshark, observar los paquetes intercambiados.

|  |
| --- |
| *Adjuntar los comandos nc utilizados y su salida.*  *[cursoredes@localhost ~]$ nc -v -z 192.168.0.1 7775*  *Ncat: Version 7.50 ( https://nmap.org/ncat )*  *Ncat: Connection refused.*  *[cursoredes@localhost ~]$ nc -v -z 192.168.0.1 7776*  *Ncat: Version 7.50 ( https://nmap.org/ncat )*  *Ncat: Connection refused.*  *[cursoredes@localhost ~]$ nc -v -z 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 0.02 seconds.*  *[cursoredes@localhost ~]$ nc -v -z 192.168.0.1 7778*  *Ncat: Version 7.50 ( https://nmap.org/ncat )*  *Ncat: Connection refused.*  *[cursoredes@localhost ~]$ nc -v -z 192.168.0.1 7779*  *Ncat: Version 7.50 ( https://nmap.org/ncat )*  *Ncat: Connection refused.*  *[cursoredes@localhost ~]$ nc -v -z 192.168.0.1 7780*  *Ncat: Version 7.50 ( https://nmap.org/ncat )*  *Ncat: Connection refused.* |

***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:

Comando: $ sysctl -a

|  |  |  |
| --- | --- | --- |
| **Parámetro del kernel** | **Propósito** | **Valor por defecto** |
| net.ipv4.tcp\_window\_scaling | Permite aumentar el tamaño de la ventana, si los dos extremos lo soportan. | 1 |
| net.ipv4.tcp\_timestamps | Permiten identificar, sin apenas coste, cuándo se generó el mensaje. | 1 |
| net.ipv4.tcp\_sack | En caso de que los paquetes se pierdan, sólo será necesario retransmitir los perdidos, en lugar de toda la secuencia a partir del perdido. | 1 |

***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.

|  |
| --- |
| *Adjuntar una captura de pantalla de Wireshark donde se muestren las opciones TCP.* |

***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 | Tiempo a partir del cual se empiezan a mandar señales keep\_alive | 7200 |
| net.ipv4.tcp\_keepalive\_probes | Número máximo de señales keep\_alive que se enviarán antes de terminar la conexión | 9 |
| net.ipv4.tcp\_keepalive\_intvl | Número de segundos entre las señales keep\_alive | 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 los dos interfaces.
* (VM1) Comprobar la conexión entre VM1 y VM4 con la orden ping.
* (Router) Analizar con Wireshark el tráfico intercambiado, especialmente los puertos y direcciones IP origen y destino en ambas redes

|  |
| --- |
| *Adjuntar el comando iptables utilizado y una captura de pantalla de Wireshark.*  (Router) [cursoredes@localhost ~]$ sudo iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE  (VM1) [cursoredes@localhost ~]$ ping 172.16.0.4 |

***Ejercicio 13*.** ¿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.

|  |
| --- |
| *Adjuntar la salida del comando conntrack y responder a la pregunta.*  [root@localhost ~]# sudo conntrack -Lo  conntrack v1.4.4 (conntrack-tools): 0 flow entries have been shown.  (No se porque no me salen las entradas y/o ejecutar comando “$ sudo cat /proc/net/nf\_conntrack”, pero he estado mirando en internet que debería suceder con la máquina virtual 4 que se conecta a la red 192.168.0.0/24 a través de eth1)  En el caso de VM1, las direcciones de origen y de destino son las correspondientes a las que hemos configurado previamente, pero para VM4 la dirección de origen y destino de los paquetes resulta ser la del router eth1 (172.16.0.3). |

***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 los dos interfaces.
* (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.

|  |
| --- |
| *Adjuntar el comando iptables utilizado y una captura de pantalla de Wireshark.*  (Router) [cursoredes@localhost ~]$ sudo iptables -t nat -F  (Router) [cursoredes@localhost ~]$ sudo iptables -t nat -A PREROUTING -d 172.16.0.3 -p tcp --dport 80 -j DNAT --to 192.168.0.1:7777  (VM1) [cursoredes@localhost ~]$ nc -l 7777  (VM4) [cursoredes@localhost ~]$ nc 172.16.0.3 80 |