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**.

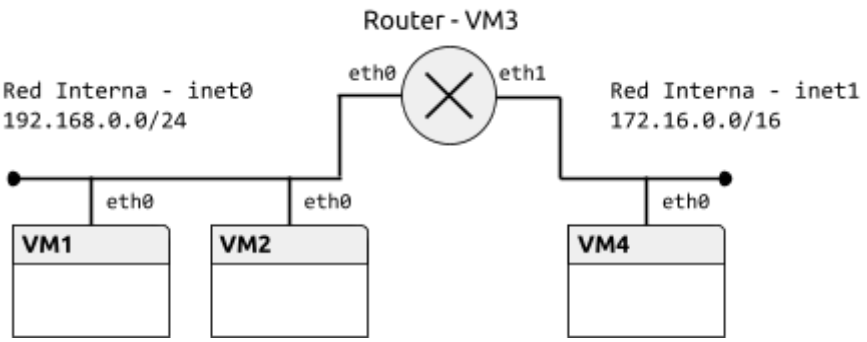
La contraseña del usuario `cursoredes` es `cursoredes`.

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 `vtopol`**.

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`.

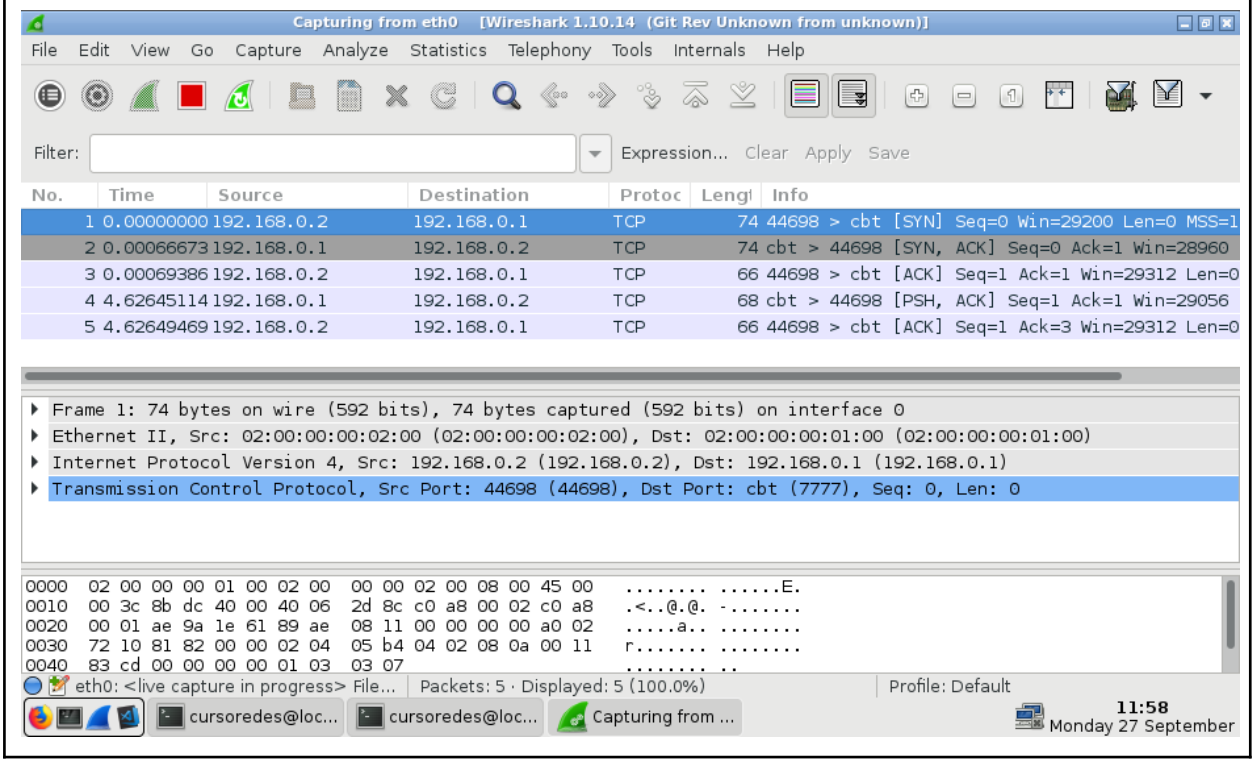
```
State    Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN   0      100    127.0.0.1:25                *:*
LISTEN   0      10        *:7777                   *:*
LISTEN   0      128       *:111                    *:*
LISTEN   0      128       *:22                     *:*
LISTEN   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                  :::*

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                  :::*
```

```
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.

```
State      Recv-Q Send-Q Local Address:Port          Peer Address:Port
ESTAB      0      0      192.168.0.2:44696          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.

```
[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      192.168.0.2:44700           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 -o -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     192.168.0.2:44702          192.168.0.1:7777          timer:(timewait,57sec,0)
LISTEN    0    100      ::1:25               :::*
LISTEN    0    128       :::111              :::*
LISTEN    0    128       :::22               :::*
LISTEN    0    128      ::1:631              :::*
```

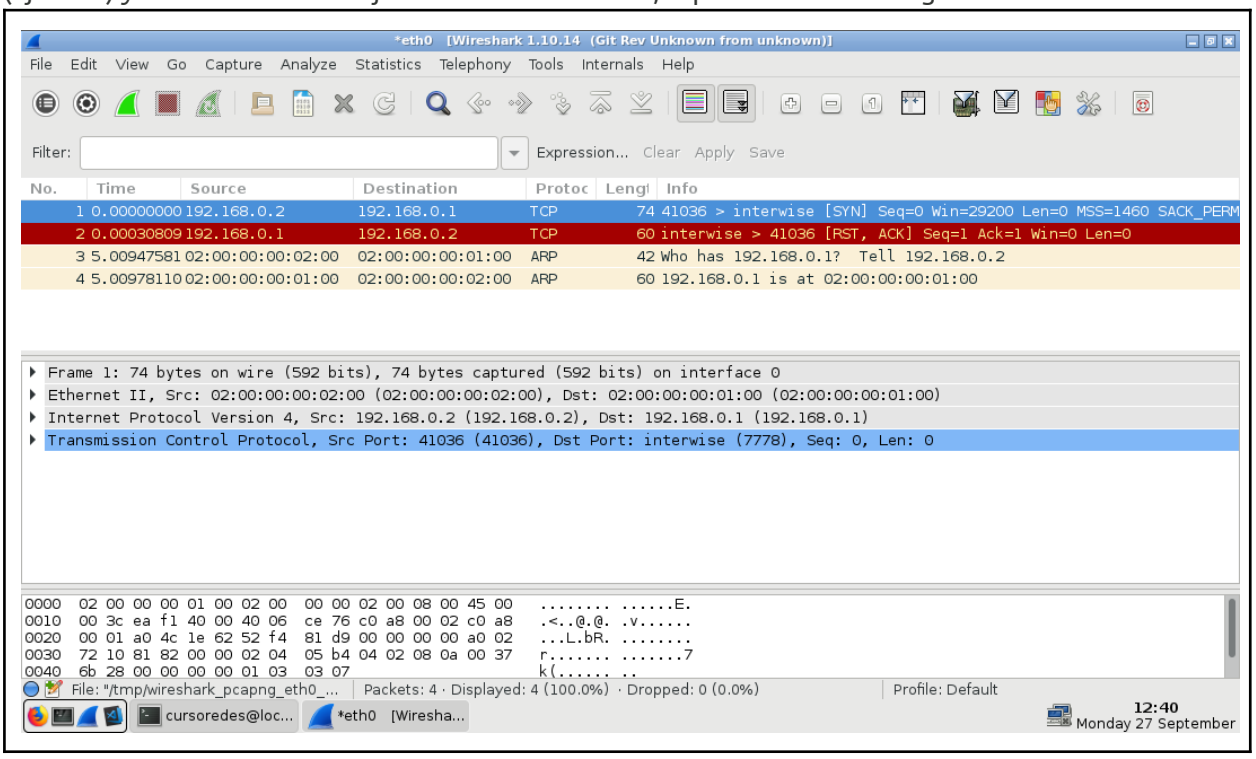
Ejercicio 5. (SYN-SENT y SYN-RECV) El comando `iptables` permite filtrar paquetes según los flags TCP del segmento con la opción `--tcp-flags` (consultar la página de manual `iptables-extensions`). Usando esta opción:

- 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.

```
sudo iptables -A OUTPUT -p tcp --tcp-flags ALL SYN,ACK -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     192.168.0.2:44706          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 -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           192.168.0.2:44710
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 -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        192.168.0.2:44716
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.



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`).

```
iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -j DROP

[cursoredes@localhost ~]$ sudo hping3 --syn -p 22 --flood 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 ---
36417040 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

Resultado del  ataque:
[cursoredes@localhost ~]$ ss -tan
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:8201
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:7112
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:6309
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:8436
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:8401
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:8220
.
.
.


SYN cookies activado:
[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                     *:*
.
.
.
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:9963
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:12572
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:9205
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:7823
SYN-RECV  0     0     192.168.0.1:22              192.168.0.2:14222
LISTEN     0      128    127.0.0.1:631                *:*
ESTAB      0      0      192.168.0.1:22             192.168.0.3:53378
LISTEN     0      100       ::1:25                   :::*
LISTEN     0      128       :::111                   :::*
LISTEN     0      128       :::22                    :::*
LISTEN     0      128       ::1:631                  :::*

SYN cookies desactivado:
[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      192.168.0.3:53380            192.168.0.1:22
LISTEN     0      100       ::1:25                   :::*
LISTEN     0      128       :::111                   :::*
LISTEN     0      128       :::22                    :::*
LISTEN     0      128       ::1:631                  :::*

Ahora solo llega a enviar el SYN pero el server no responde por lo que no llega a pedir ni la contraseña
```

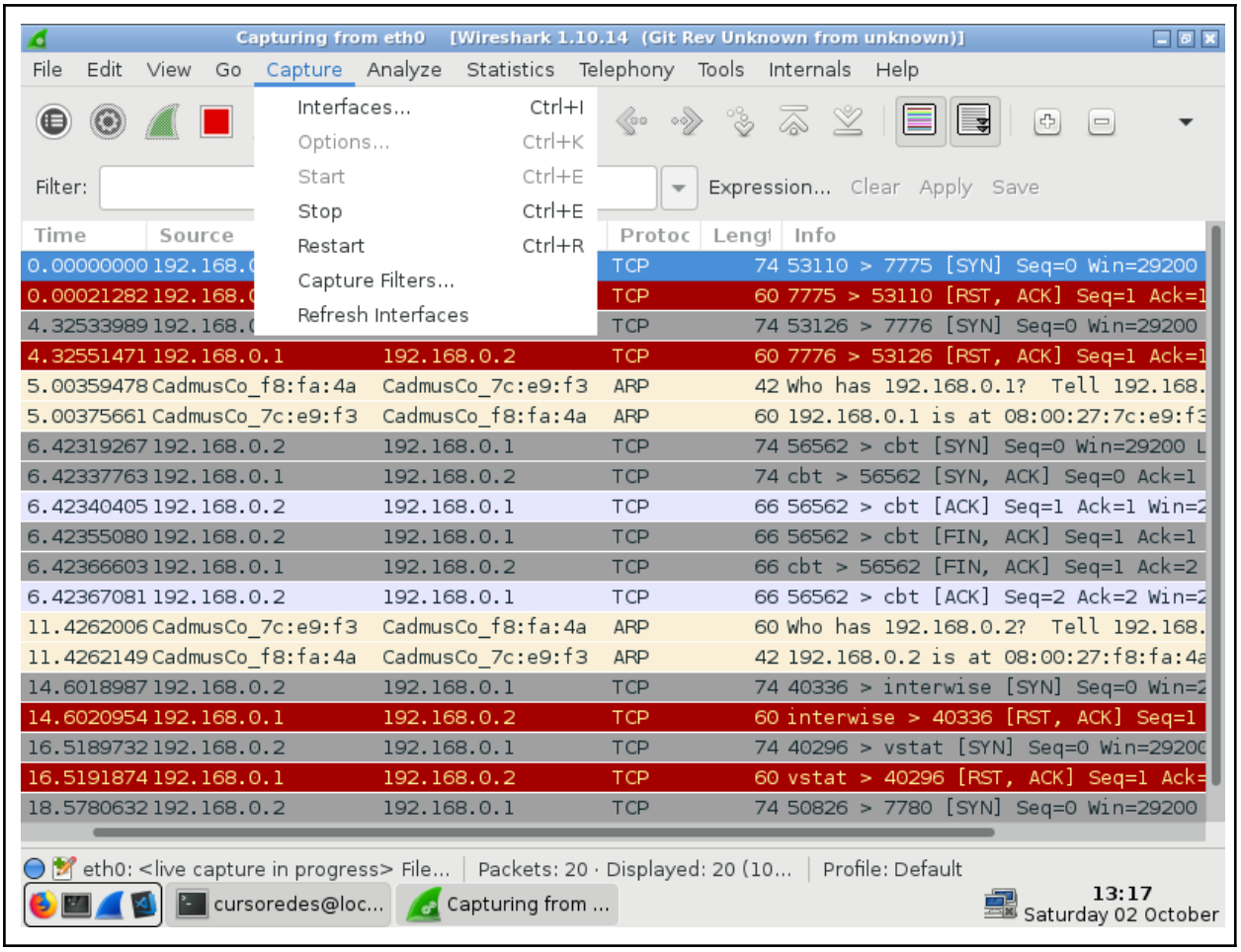
Nota: Wireshark no debe estar activo cuando se envían paquetes lo más rápido posible (*flooding*).

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 -l 7777 (VM1)

(VM2)
[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 0.01 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

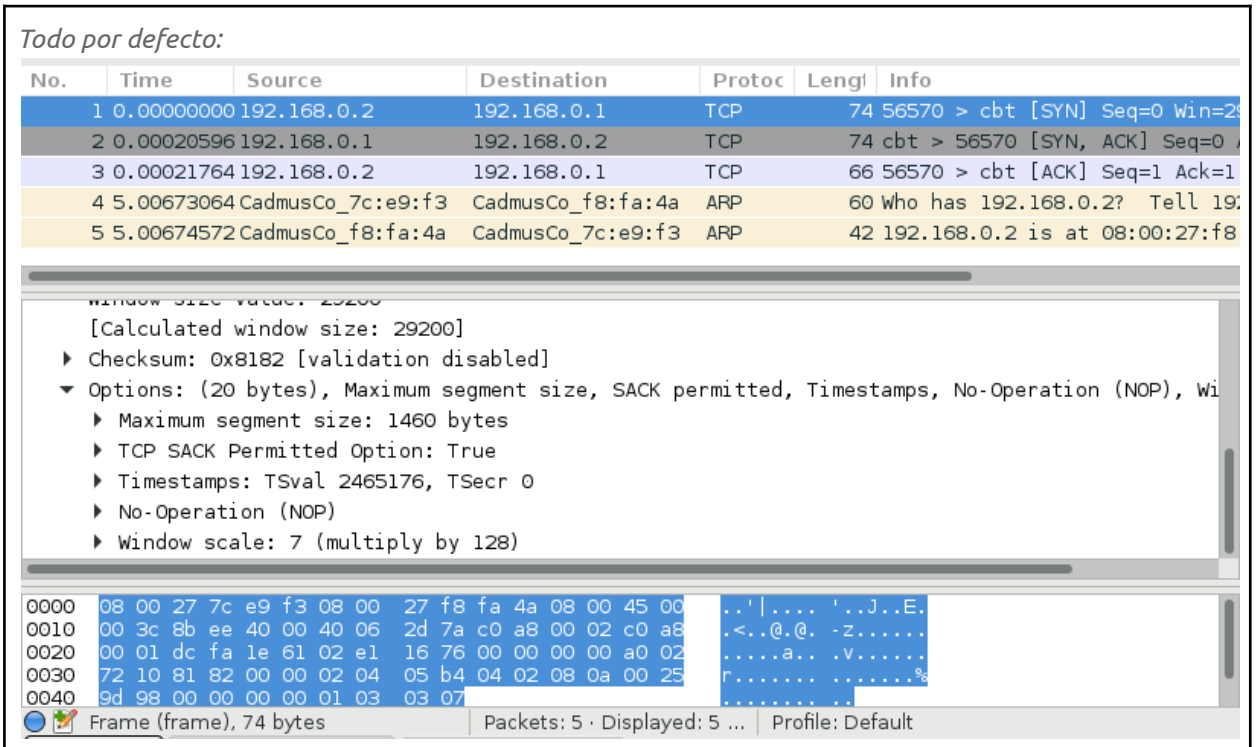
El comportamiento de la conexión TCP se puede controlar con varias opciones que se incluyen en la cabecera en los mensajes SYN y que son configurables en el sistema operativo por medio de parámetros del kernel.

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

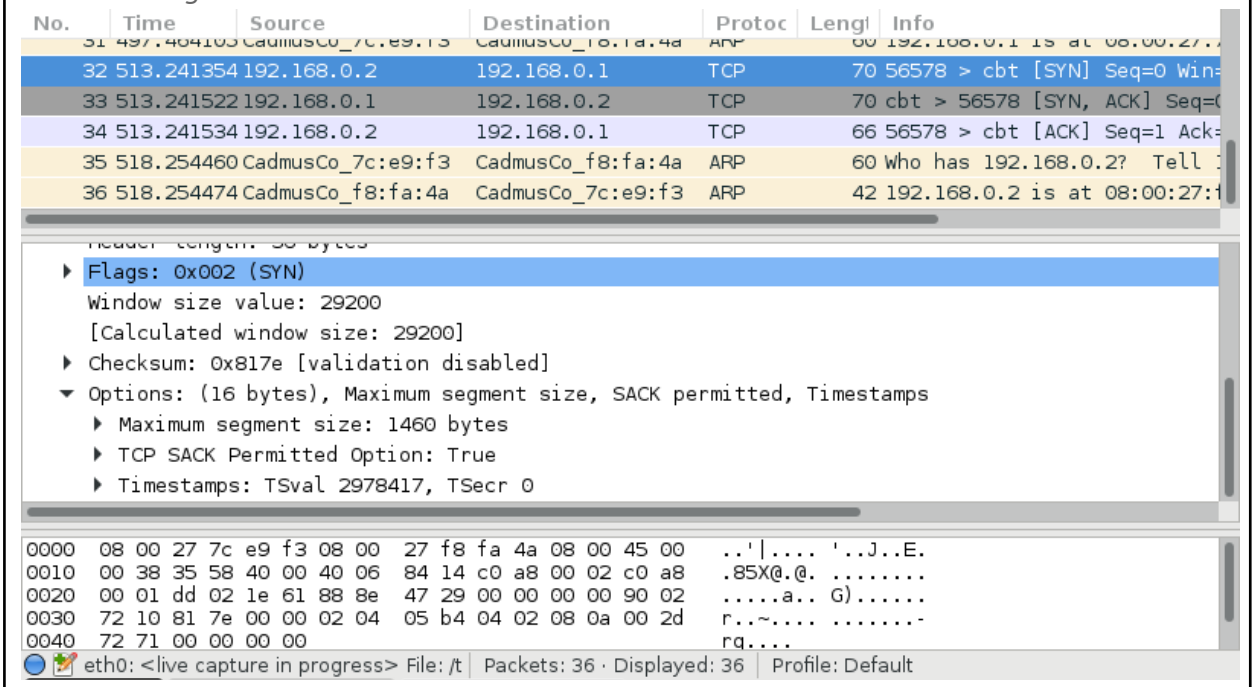
Parámetro del kernel	**Propósito**	**Valor por defecto**
`net.ipv4.tcp_window_scaling`	Enables window scaling, change value of the receive window size allowed in TCP	1
`net.ipv4.tcp_timestamps`	Enable timestamps in TCP	1
`net.ipv4.tcp_sack`	Enable select acknowledgments	1

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.

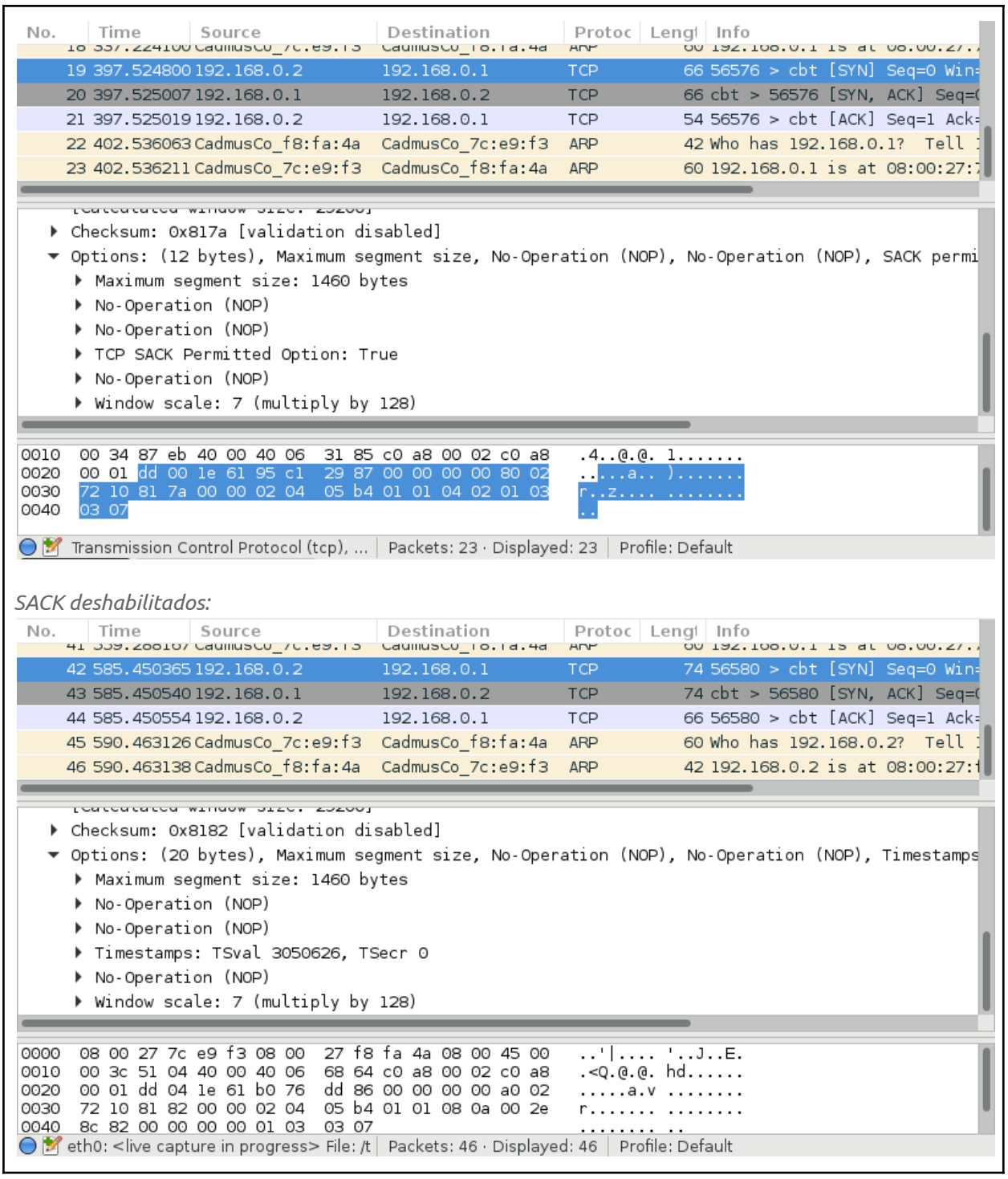
Todo por defecto:



Window scaling deshabilitado:



Timestamps deshabilitados:



SACK deshabilitados:

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`	How often TCP sends out keepalive messages when keepalive is enabled.	2 horas
`net.ipv4.tcp_keepalive_probes`	How many keepalive probes TCP sends out, until it decides that the	9

	connection is broken.	
`net.ipv4.tcp_keepalive_intvl`	How frequently the probes are sent out. Multiplied by tcp_keepalive_probes it is time to kill not responding connection, after probes started.	75 segundos

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 eth1 -j MASQUERADE

eth0:

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	192.168.0.1	172.16.0.1	ICMP	98	Echo (ping) request id=0x246
2	0.00020511	172.16.0.1	192.168.0.1	ICMP	98	Echo (ping) reply id=0x246

```
Total Length: 84
Identification: 0xbeb4 (48820)
▸ Flags: 0x00
Fragment offset: 0
Time to live: 63
Protocol: ICMP (1)
▸ Header checksum: 0x503a [validation disabled]
Source: 172.16.0.1 (172.16.0.1)
Destination: 192.168.0.1 (192.168.0.1)
```

eth1:

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	172.16.0.2	172.16.0.1	ICMP	98	Echo (ping) request id=0x246
2	0.00016120	172.16.0.1	172.16.0.2	ICMP	98	Echo (ping) reply id=0x246

```
▸ Flags: 0x00
Fragment offset: 0
Time to live: 64
Protocol: ICMP (1)
▸ Header checksum: 0x63d1 [validation disabled]
Source: 172.16.0.1 (172.16.0.1)
Destination: 172.16.0.2 (172.16.0.2)
```

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?

[cursoredes@localhost ~]$ sudo conntrack -L
icmp 1 29 src=192.168.0.1 dst=172.16.0.1 type=8 code=0 id=9385 src=172.16.0.1 dst=172.16.0.2 type=0 code=0 id=9385 mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 1 flow entries have been shown.

Las solicitudes se relacionan con las respuestas gracias al id.

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 -j DNAT --to 192.168.0.1:7777

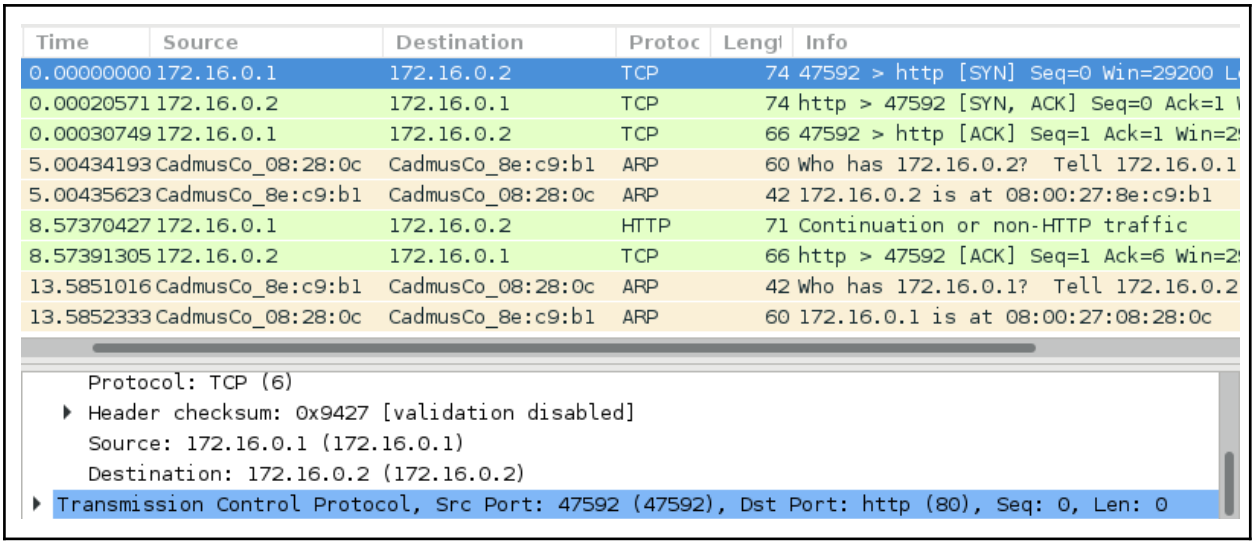
También ha sido enviado un mensaje para probar la conexión

eth0:

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	172.16.0.1	192.168.0.1	TCP	74	47592 > cbt [SYN] Seq=0 Win=2
2	0.00016485	192.168.0.1	172.16.0.1	TCP	74	cbt > 47592 [SYN, ACK] Seq=0
3	0.00028033	172.16.0.1	192.168.0.1	TCP	66	47592 > cbt [ACK] Seq=1 Ack=1
4	5.00908763	CadmusCo_77:49:cc	CadmusCo_7c:e9:f3	ARP	42	Who has 192.168.0.1? Tell 19
5	5.00921998	CadmusCo_7c:e9:f3	CadmusCo_77:49:cc	ARP	60	192.168.0.1 is at 08:00:27:7c
6	8.57369851	172.16.0.1	192.168.0.1	TCP	71	47592 > cbt [PSH, ACK] Seq=1
7	8.57387124	192.168.0.1	172.16.0.1	TCP	66	cbt > 47592 [ACK] Seq=1 Ack=6
8	13.5744348	CadmusCo_7c:e9:f3	CadmusCo_77:49:cc	ARP	60	Who has 192.168.0.3? Tell 19
9	13.5744470	CadmusCo_77:49:cc	CadmusCo_7c:e9:f3	ARP	42	192.168.0.3 is at 08:00:27:77

```
    Protocol: TCP (6)
  ▸ Header checksum: 0x8090 [validation disabled]
    Source: 172.16.0.1 (172.16.0.1)
    Destination: 192.168.0.1 (192.168.0.1)
▸ Transmission Control Protocol, Src Port: 47592 (47592), Dst Port: cbt (7777), Seq: 0, Len: 0
```

eth1:


Time Source Destination Protoc Lengt Info
0.00000000 172.16.0.1 172.16.0.2 TCP 74 47592 > http [SYN] Seq=0 Win=29200 L
0.00020571 172.16.0.2 172.16.0.1 TCP 74 http > 47592 [SYN, ACK] Seq=0 Ack=1
0.00030749 172.16.0.1 172.16.0.2 TCP 66 47592 > http [ACK] Seq=1 Ack=1 Win=2
5.00434193 CadmusCo_08:28:0c CadmusCo_8e:c9:b1 ARP 60 Who has 172.16.0.2? Tell 172.16.0.1
5.00435623 CadmusCo_8e:c9:b1 CadmusCo_08:28:0c ARP 42 172.16.0.2 is at 08:00:27:8e:c9:b1
8.57370427 172.16.0.1 172.16.0.2 HTTP 71 Continuation or non-HTTP traffic
8.57391305 172.16.0.2 172.16.0.1 TCP 66 http > 47592 [ACK] Seq=1 Ack=6 Win=2
13.5851016 CadmusCo_8e:c9:b1 CadmusCo_08:28:0c ARP 42 Who has 172.16.0.1? Tell 172.16.0.2
13.5852333 CadmusCo_08:28:0c CadmusCo_8e:c9:b1 ARP 60 172.16.0.1 is at 08:00:27:08:28:0c
Protocol: TCP (6)
Header checksum: 0x9427 [validation disabled]
Source: 172.16.0.1 (172.16.0.1)
Destination: 172.16.0.2 (172.16.0.2)
Transmission Control Protocol, Src Port: 47592 (47592), Dst Port: http (80), Seq: 0, Len: 0