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**.

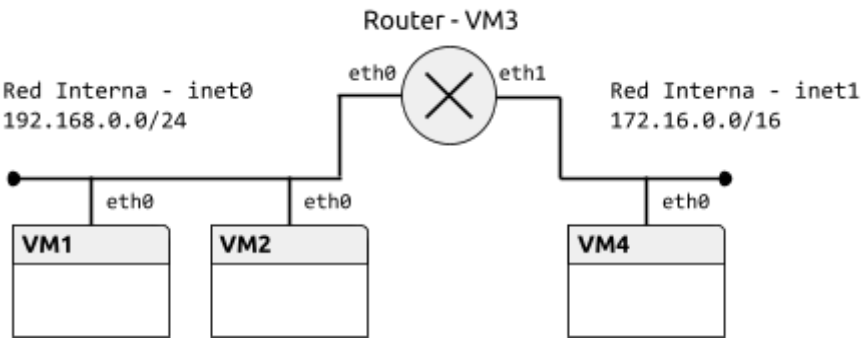
La contraseña del usuario `cursoredes` es `cursoredes`.

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.



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`.

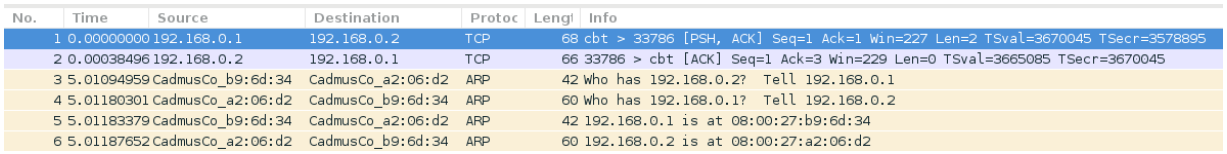
```
ss -tln
State    Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN   0      100    127.0.0.1:25                *:*
LISTEN   0      10     192.168.0.1:7776               *:*
LISTEN   0      10        *:7777                   *:*
LISTEN   0      128       *:111                    *:*
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                  :::*

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.

```
ss -tn
```

```
State      Recv-Q Send-Q Local Address:Port          Peer Address:Port
ESTAB      0      0      192.168.0.2:33786          192.168.0.1:7777
```



No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	192.168.0.1	192.168.0.2	TCP	68	cbt > 33786 [PSH, ACK] Seq=1 Ack=1 Win=227 Len=2 TSval=3670045 TSecr=3578895
2	0.00038496	192.168.0.2	192.168.0.1	TCP	66	33786 > cbt [ACK] Seq=1 Ack=3 Win=229 Len=0 TSval=3665085 TSecr=3670045
3	5.01094959	CadmusCo_b9:6d:34	CadmusCo_a2:06:d2	ARP	42	Who has 192.168.0.2? Tell 192.168.0.1
4	5.01180301	CadmusCo_a2:06:d2	CadmusCo_b9:6d:34	ARP	60	Who has 192.168.0.1? Tell 192.168.0.2
5	5.01183379	CadmusCo_b9:6d:34	CadmusCo_a2:06:d2	ARP	42	192.168.0.1 is at 08:00:27:b9:6d:34
6	5.01187652	CadmusCo_a2:06:d2	CadmusCo_b9:6d:34	ARP	60	192.168.0.2 is at 08:00:27:a2:06:d2

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.

```
ss -tn
State      Recv-Q Send-Q Local Address:Port          Peer Address:Port
CLOSE-WAIT 0      0      192.168.0.2:33786          192.168.0.1:7777


timer:(timewait,45sec,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:

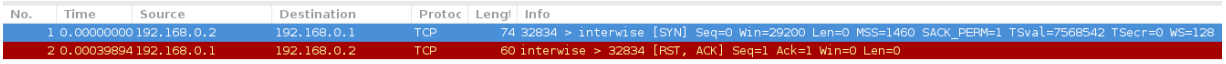
- Fijar una regla en el servidor (VM1) que bloquee un mensaje del acuerdo TCP de forma que el cliente (VM2) se quede en el estado SYN-SENT. Comprobar el resultado con `ss -tan` en el cliente.
- Borrar la regla anterior y fijar otra en el cliente (VM2) que bloquee un mensaje del acuerdo TCP de forma que 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.

```
a)
VM1:
iptables -A INPUT -p tcp --destination-port 7777 --tcp-flags ALL SYN -j DROP

VM2:
$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           192.168.0.2:33812             192.168.0.1:7777
LISTEN     0      100           ::1:25                      :::*
LISTEN     0      128           :::111                      :::*
LISTEN     0      128           :::22                       :::*
LISTEN     0      128           ::1:631                     :::*

b)
VM2:
sudo iptables -A OUTPUT -p tcp --dport 7777 --tcp-flags ALL ACK -j DROP

VM1:
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                 192.168.0.2:33814
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 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.

VM2:
nc 192.168.0.1 7778



No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	192.168.0.2	192.168.0.1	TCP	74	32834 > interwise [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=7568542 TSecr=0 WS=128
2	0.00039894	192.168.0.1	192.168.0.2	TCP	60	interwise > 32834 [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`).

VM2:Iptables -A INPUT -p tcp –tcp-flags ALL SYN,ACK -j DROP

VM2:$ hping3 --flood -p 22 --syn 182.168.0.1
VM1: $ ssh localhost

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.

VM2:

nc -z -v 192.168.0.1 7775

Ncat: Version 7.50 (https://nmap.org/ncat)

Ncat: Connection refused.

[cursoredes@localhost ~]$ nc -z -v 192.168.0.1 7776

Ncat: Version 7.50 (https://nmap.org/ncat)

Ncat: Connection refused.

[cursoredes@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 0.03 seconds.

[cursoredes@localhost ~]$ nc -z -v 192.168.0.1 7778

Ncat: Version 7.50 (https://nmap.org/ncat)

Ncat: Connection refused.

[cursoredes@localhost ~]$ nc -z -v 192.168.0.1 7779

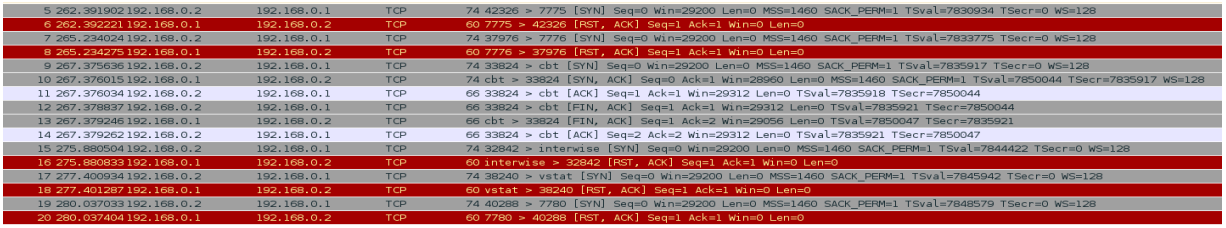
Ncat: Version 7.50 (https://nmap.org/ncat)

Ncat: Connection refused.

[cursoredes@localhost ~]$ nc -z -v 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 (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`	Aumentar tamaño de RW a >64kB	1
`net.ipv4.tcp_timestamps`	Poner marcas de tiempo para determinar el orden de paquetes	1
`net.ipv4.tcp_sack`	Activar los Selective ACK	1

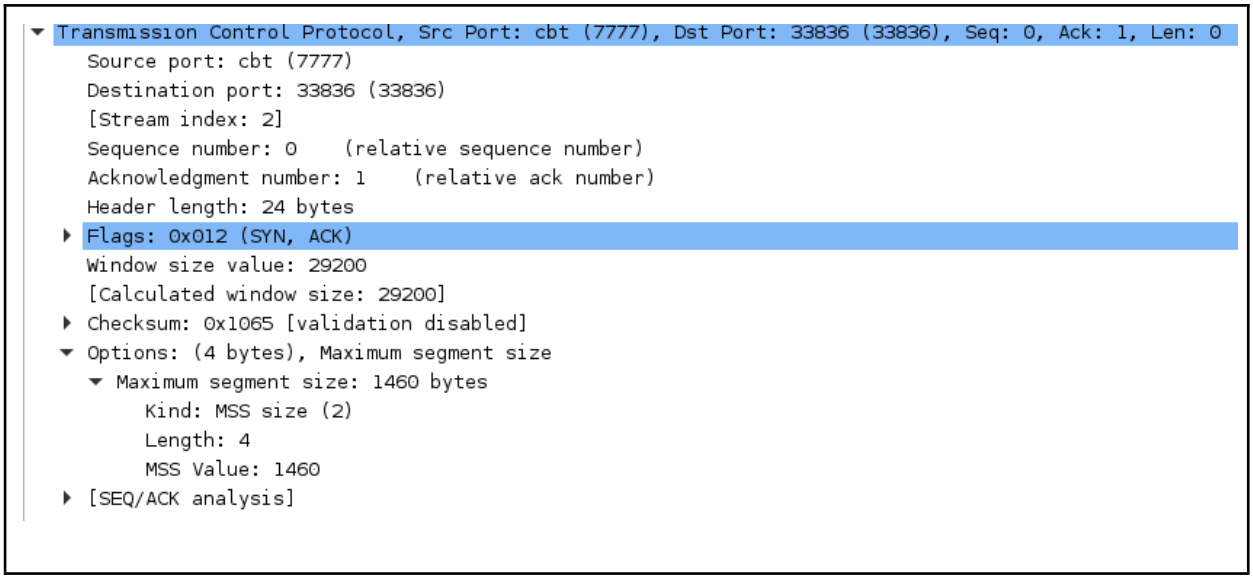
Ejercicio 10. Iniciar una captura de Wireshark. Abrir el servidor en el puerto 7777 y realizar una conexión desde la VM cliente. Estudiar el valor de las opciones que se intercambian durante la conexión. Variar algunos de los parámetros anteriores (ej. no usar ACKs selectivos) y observar el resultado en una nueva conexión.

Sin cambios:

```
▼ Transmission Control Protocol, Src Port: 33832 (33832), Dst Port: cbt (7777), Seq: 1, Ack: 1, Len: 0
    Source port: 33832 (33832)
    Destination port: cbt (7777)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header length: 32 bytes
  ▶ Flags: 0x010 (ACK)
    Window size value: 229
    [Calculated window size: 29312]
    [Window size scaling factor: 128]
  ▶ Checksum: 0x817a [validation disabled]
  ▼ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▶ No-Operation (NOP)
    ▶ No-Operation (NOP)
    ▶ Timestamps: TSval 8607827, TSecr 8621954
  ▶ [SEQ/ACK analysis]
```

Con cambios:

```
sudo sysctl net.ipv4.tcp_window_scaling=0
net.ipv4.tcp_window_scaling = 0
[cursoredes@localhost ~]$ sudo sysctl net.ipv4.tcp_timestamps=0
net.ipv4.tcp_timestamps = 0
[cursoredes@localhost ~]$ sudo sysctl net.ipv4.tcp_sack=0
net.ipv4.tcp_sack = 0
```



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 para revisar si una conexión sigue viva	7200
`net.ipv4.tcp_keepalive_probes`	Número de pruebas para ver si la conexión sigue viva una vez pasado el tiempo de comprobación	9
`net.ipv4.tcp_keepalive_intvl`	Intervalo de tiempo entre pruebas	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

VM3:
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
VM1:
ping 172.16.0.4 -c 1
PING 172.16.0.4 (172.16.0.4) 56(84) bytes of data.
64 bytes from 172.16.0.4: icmp_seq=1 ttl=63 time=0.765 ms

--- 172.16.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 0.765/0.765/0.765/0.000 ms

eth0:

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	192.168.0.1	172.16.0.4	ICMP	98	Echo (ping) request id=0x11f7, seq=1/256, ttl=64 (reply in 2)
2	0.00037701	172.16.0.4	192.168.0.1	ICMP	98	Echo (ping) reply id=0x11f7, seq=1/256, ttl=63 (request in 1)
3	5.01258425	CadmusCo_48:35:c1	CadmusCo_b9:6d:34	ARP	42	Who has 192.168.0.1? Tell 192.168.0.3
4	5.01322002	CadmusCo_b9:6d:34	CadmusCo_48:35:c1	ARP	60	192.168.0.1 is at 08:00:27:b9:6d:34

eth1:

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	172.16.0.3	172.16.0.4	ICMP	98	Echo (ping) request id=0x11f7, seq=1/256, ttl=63 (reply in 2)
2	0.00032227	172.16.0.4	172.16.0.3	ICMP	98	Echo (ping) reply id=0x11f7, seq=1/256, ttl=64 (request in 1)
3	5.01138361	CadmusCo_5f:b1:c7	CadmusCo_5d:d6:4f	ARP	60	Who has 172.16.0.3? Tell 172.16.0.4
4	5.01142826	CadmusCo_5d:d6:4f	CadmusCo_5f:b1:c7	ARP	42	172.16.0.3 is at 08:00:27:5d:d6:4f

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?

VM3:
$sudo conntrack -L
icmp 1 29 src=192.168.0.1 dst=172.16.0.4 type=8 code=0 id=4655 src=172.16.0.4 dst=172.16.0.3 type=0 code=0 id=4655 mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 1 flow entries have been shown.

Se utilizan directamente las direcciones IP.

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.

VM3:
iptables -t nat -A PREROUTING -d 172.16.0.3 -p tcp --dport 80 -j DNAT --to 192.168.0.1:7777
VM1:
nc -l 777
VM4:
nc 172.16.0.3 80
VM3:
eth0:

37	169.904563	CadmusCo_5d:d6:4f	CadmusCo_5f:b1:c7	ARP	42	Who has 172.16.0.4? Tell 172.16.0.3
38	169.904821	CadmusCo_5f:b1:c7	CadmusCo_5d:d6:4f	ARP	60	172.16.0.4 is at 08:00:27:5f:b1:c7
39	201.877107	172.16.0.4	172.16.0.3	TCP	74	45874 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10187680 TSecr=0 WS=128
40	201.877485	172.16.0.3	172.16.0.4	TCP	74	http > 45874 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10232511 TSecr=10187680 WS=128
41	201.877784	172.16.0.4	172.16.0.3	TCP	66	45874 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=10187681 TSecr=10232511
42	206.880304	CadmusCo_5d:d6:4f	CadmusCo_5f:b1:c7	ARP	42	Who has 172.16.0.4? Tell 172.16.0.3
43	206.881058	CadmusCo_5f:b1:c7	CadmusCo_5d:d6:4f	ARP	60	172.16.0.4 is at 08:00:27:5f:b1:c7

eth1:

39	201.877107	172.16.0.4	172.16.0.3	TCP	74	45874 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=10187680 TSecr=0 WS=128
40	201.877485	172.16.0.3	172.16.0.4	TCP	74	http > 45874 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=10232511 TSecr=10187680 WS=128
41	201.877784	172.16.0.4	172.16.0.3	TCP	66	45874 > http [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=10187681 TSecr=10232511
42	206.880304	CadmusCo_5d:d6:4f	CadmusCo_5f:b1:c7	ARP	42	Who has 172.16.0.4? Tell 172.16.0.3
43	206.881058	CadmusCo_5f:b1:c7	CadmusCo_5d:d6:4f	ARP	60	172.16.0.4 is at 08:00:27:5f:b1:c7