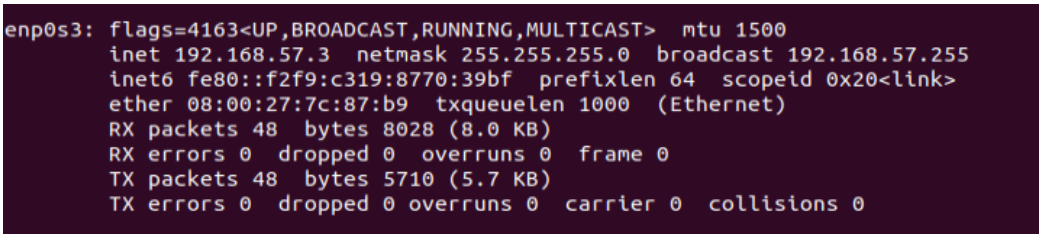
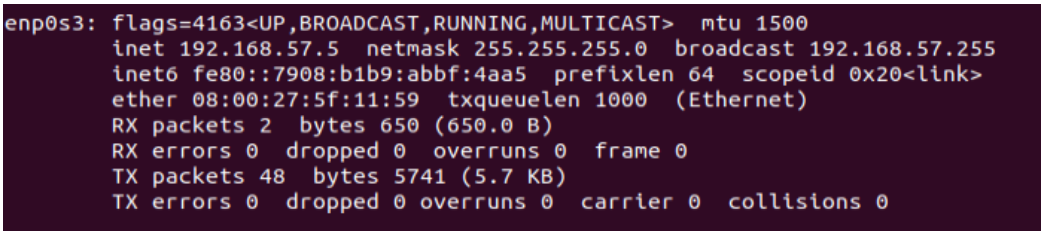
Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

DOCUMENTACIÓN PRÁCTICA 2

- **Check the IPs for both machines (ifconfig -a). Verify that both have a different address, and note the address of the VM (usually 192.168.57.3) acting as a server. Ping from client to server machine, using the address just seen. You should get a successful response.**

  En las siguientes imágenes se muestran las configuraciones del servidor y el cliente, respectivamente.

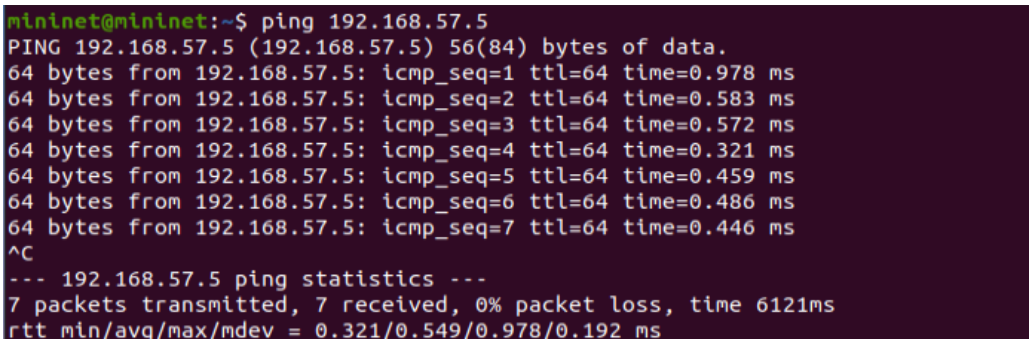




  A partir de las dos imágenes anteriores podemos extraer las IPs físicas de ambas máquinas:

  - Server: 192.168.57.3
  - Client: 192.168.57.5

  Una vez hemos comprobado que ambas máquinas tienen distintas IPs hemos realizado un ping desde el servidor hacia el cliente y desde el cliente hacia el servidor para comprobar la conexión entre ellas.



Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

```
mininet@mininet:~$ ping 192.168.57.3
PING 192.168.57.3 (192.168.57.3) 56(84) bytes of data.
64 bytes from 192.168.57.3: icmp_seq=1 ttl=64 time=0.698 ms
64 bytes from 192.168.57.3: icmp_seq=2 ttl=64 time=0.777 ms
64 bytes from 192.168.57.3: icmp_seq=3 ttl=64 time=0.605 ms
64 bytes from 192.168.57.3: icmp_seq=4 ttl=64 time=0.371 ms
64 bytes from 192.168.57.3: icmp_seq=5 ttl=64 time=0.571 ms
64 bytes from 192.168.57.3: icmp_seq=6 ttl=64 time=0.781 ms
64 bytes from 192.168.57.3: icmp_seq=7 ttl=64 time=0.523 ms
^C
--- 192.168.57.3 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6060ms
rtt min/avg/max/mdev = 0.371/0.618/0.781/0.136 ms
```

- **Edit file server.conf with a text editor e.g gedit (sudo gedit /etc/openvpn/server/server.conf from the terminal window, enable "Display line numbers" in the Preferences menu to see the file lines).**

  A partir de la instrucción *sudo gedit /etc/openvpn/server/server.conf* hemos podido editar el fichero *server.conf* y verificar y editar las líneas de código que se nos proponía en el enunciado. A continuación se muestra como han quedado las líneas 78, 79, 80, 85 y 101 del archivo en cuestión:

```
78 ca /etc/openvpn/easy-rsa/pki/ca.crt
79 cert /etc/openvpn/easy-rsa/pki/issued/server.crt
80 key /etc/openvpn/easy-rsa/pki/private/server.key  # This file should be kept secret
81
82 # Diffie hellman parameters.
83 # Generate your own with:
84 #   openssl dhparam -out dh2048.pem 2048
85 dh /etc/openvpn/easy-rsa/pki/dh.pem
86
87 # Network topology
88 # Should be subnet (addressing via IP)
89 # unless Windows clients v2.0.9 and lower have to
90 # be supported (then net30, i.e. a /30 per client)
91 # Defaults to net30 (not recommended)
92 ;topology subnet
93
94 # Configure server mode and supply a VPN subnet
95 # for OpenVPN to draw client addresses from.
96 # The server will take 10.8.0.1 for itself,
97 # the rest will be made available to clients.
98 # Each client will be able to reach the server
99 # on 10.8.0.1. Comment this line out if you are
100 # ethernet bridging. See the man page for more info.
101 server 10.8.0.0 255.255.255.0
```

- **Change dir to /etc/openvpn/server and start the OpenVPN server (sudo openvpn server.conf); expect a line similar to "Initialization Sequence Completed".**
  Para poder llegar al directorio simplemente nos hemos tenido que mover a través de las carpetas con la instrucción cd y la carpeta de destino.

```
mininet@mininet:/etc/openvpn/server$ sudo openvpn server.conf
Thu Apr 21 19:30:20 2022 OpenVPN 2.4.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Jul 19 2021
Thu Apr 21 19:30:20 2022 library versions: OpenSSL 1.1.1f  31 Mar 2020, LZO 2.10
Thu Apr 21 19:30:20 2022 Diffie-Hellman initialized with 2048 bit key
Thu Apr 21 19:30:20 2022 ROUTE: default_gateway=UNDEF
Thu Apr 21 19:30:20 2022 TUN/TAP device tun0 opened
Thu Apr 21 19:30:20 2022 TUN/TAP TX queue length set to 100
Thu Apr 21 19:30:20 2022 /sbin/ip link set dev tun0 up mtu 1500
Thu Apr 21 19:30:20 2022 /sbin/ip addr add dev tun0 local 10.8.0.1 peer 10.8.0.2
Thu Apr 21 19:30:20 2022 /sbin/ip route add 10.8.0.0/24 via 10.8.0.2
Thu Apr 21 19:30:20 2022 Could not determine IPv4/IPv6 protocol. Using AF_INET
Thu Apr 21 19:30:20 2022 Socket Buffers: R=[212992->212992] S=[212992->212992]
Thu Apr 21 19:30:20 2022 UDPv4 link local (bound): [AF_INET][undef]:1194
Thu Apr 21 19:30:20 2022 UDPv4 link remote: [AF_UNSPEC]
Thu Apr 21 19:30:20 2022 MULTI: multi_init called, r=256 v=256
Thu Apr 21 19:30:20 2022 IFCONFIG POOL: base=10.8.0.4 size=62, ipv6=0
Thu Apr 21 19:30:20 2022 IFCONFIG POOL LIST
Thu Apr 21 19:30:20 2022 Initialization Sequence Completed
```

Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

- **On the other VM, edit file /etc/openvpn/client/client.conf. Review the value of "remote" directive, where the IP address of the server must be specified, and the rest of relevant parameters:**

  Siguiendo el mismo procedimiento que en el ejercicio 11, hemos accedido mediante la instrucción *sudo gedit /etc/openvpn/client/client.conf* al fichero *server.conf.* Una vez ubicados hemos podido verificar y editar las líneas de código 42, 88, 89 y 90, tal y como se muestran en las siguientes imágenes.

```
39 # The hostname/IP and port of the server.
40 # You can have multiple remote entries
41 # to load balance between the servers.
42 remote 192.168.57.3 1194
43 ;remote my-server-2 1194
```

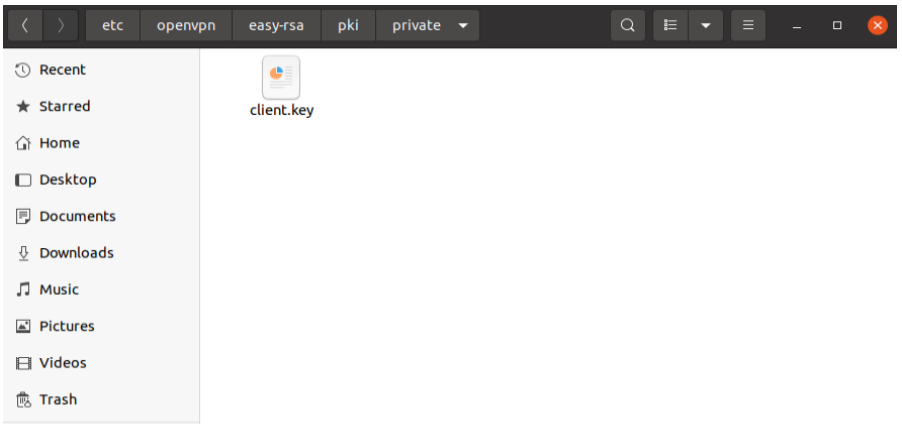
```
82 # SSL/TLS parms.
83 # See the server config file for more
84 # description.  It's best to use
85 # a separate .crt/.key file pair
86 # for each client.  A single ca
87 # file can be used for all clients.
88 ca /etc/openvpn/easy-rsa/pki/ca.crt
89 cert /etc/openvpn/easy-rsa/pki/issued/client.crt
90 key /etc/openvpn/easy-rsa/pki/private/client.key
```

- **To ensure no private key other than the own one is accessed, remove the rest of the keys.**

  Simplemente para remover todas las *keys* menos la propia hemos ejecutado las siguientes líneas de código en la VM del cliente.

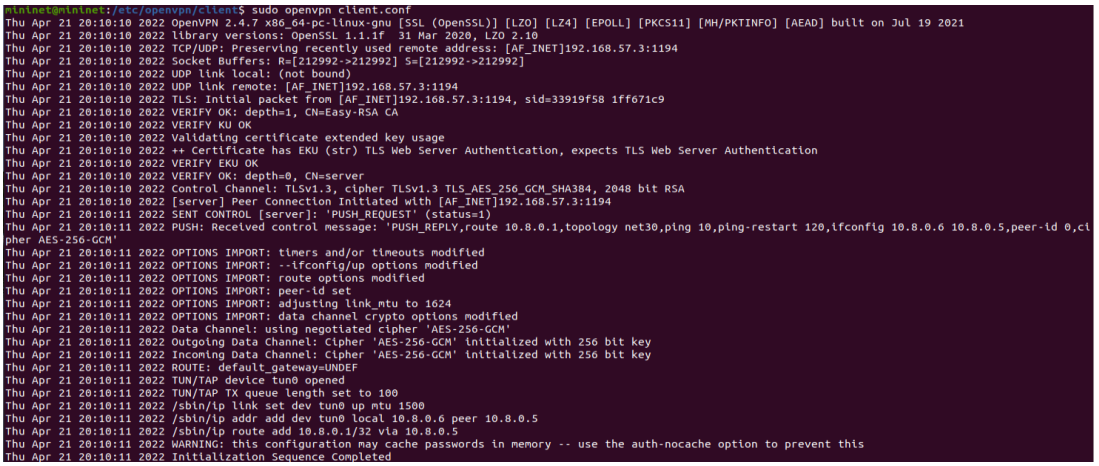
  - sudo rm /etc/openvpn/easy-rsa/pki/private/ca.key
  - sudo rm /etc/openvpn/easy-rsa/pki/private/server.key
  - sudo rm /etc/openvpn/easy-rsa/pki/private/client2.key

  Hemos tenido que añadir *sudo* para poder darle los permisos necesarios para eliminar los archivos en cuestión. En la siguiente captura se muestra que el único archivo que queda en /etc/openvpn/easy-rsa/pki/private es client.key.

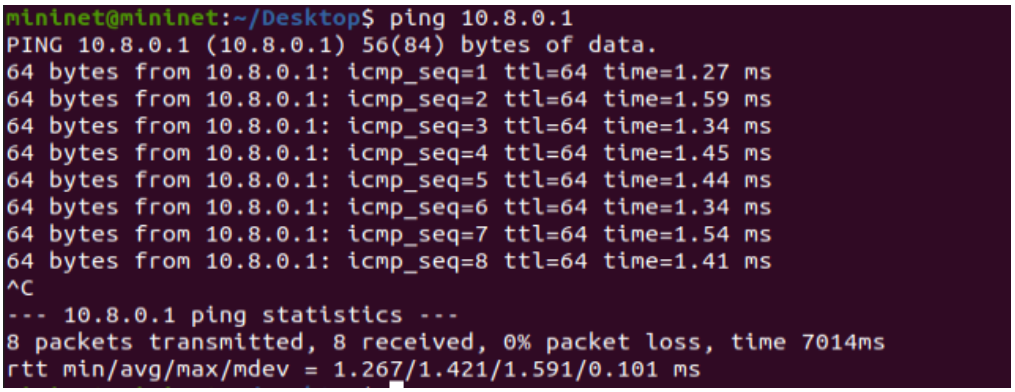


Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

- **Start the client: cd to /etc/openvpn/client and do sudo openvpn client.conf.**



- **Open a second terminal window on client and ping address of the server on the VPN (probably 10.8.0.1). If you get a response, you have just succeeded in setting up your VPN!**



- **Stop ping and open a new terminal on server. Execute a network capture over the physical interface (sudo tcpdump -n -i enp0s3) and ping again from client. Check what you can see on the network capture, and repeat after on the virtual interface (tun0). Try to explain the results in your own words.**

  - sudo tcdump -n -i enp0s3

    En el siguiente par de imágenes se muestra la información del cliente y el servidor, respectivamente, a partir de hacer un ping con la interfaz física (enp0s3). Como estamos en el caso de la interfaz física la IP que hemos tenido que dar al cliente es la física del servidor. De esta manera vemos que el cliente envía ficheros ICMP que son recibidos por el servidor.

Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

```
mininet@mininet:~/Desktop$ ping 192.168.57.3
PING 192.168.57.3 (192.168.57.3) 56(84) bytes of data.
64 bytes from 192.168.57.3: icmp_seq=1 ttl=64 time=0.817 ms
64 bytes from 192.168.57.3: icmp_seq=2 ttl=64 time=0.924 ms
64 bytes from 192.168.57.3: icmp_seq=3 ttl=64 time=0.786 ms
64 bytes from 192.168.57.3: icmp_seq=4 ttl=64 time=0.743 ms
64 bytes from 192.168.57.3: icmp_seq=5 ttl=64 time=0.613 ms
64 bytes from 192.168.57.3: icmp_seq=6 ttl=64 time=0.726 ms
64 bytes from 192.168.57.3: icmp_seq=7 ttl=64 time=0.646 ms
64 bytes from 192.168.57.3: icmp_seq=8 ttl=64 time=0.828 ms
64 bytes from 192.168.57.3: icmp_seq=9 ttl=64 time=0.953 ms
64 bytes from 192.168.57.3: icmp_seq=10 ttl=64 time=0.751 ms
64 bytes from 192.168.57.3: icmp_seq=11 ttl=64 time=0.917 ms
64 bytes from 192.168.57.3: icmp_seq=12 ttl=64 time=0.914 ms
64 bytes from 192.168.57.3: icmp_seq=13 ttl=64 time=0.810 ms
64 bytes from 192.168.57.3: icmp_seq=14 ttl=64 time=1.02 ms
64 bytes from 192.168.57.3: icmp_seq=15 ttl=64 time=0.796 ms
64 bytes from 192.168.57.3: icmp_seq=16 ttl=64 time=1.03 ms
64 bytes from 192.168.57.3: icmp_seq=17 ttl=64 time=0.784 ms
64 bytes from 192.168.57.3: icmp_seq=18 ttl=64 time=0.876 ms
64 bytes from 192.168.57.3: icmp_seq=19 ttl=64 time=0.492 ms
```

```
20:23:59.048290 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 45, length 64
20:23:59.048355 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 45, length 64
20:24:00.049461 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 46, length 64
20:24:00.049531 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 46, length 64
20:24:01.050645 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 47, length 64
20:24:01.050725 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 47, length 64
20:24:02.052029 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 48, length 64
20:24:02.052094 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 48, length 64
20:24:03.053780 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 49, length 64
20:24:03.053826 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 49, length 64
20:24:04.055727 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 50, length 64
20:24:04.055803 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 50, length 64
20:24:05.056794 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 51, length 64
20:24:05.056859 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 51, length 64
20:24:06.058227 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 52, length 64
20:24:06.058296 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 52, length 64
20:24:07.038158 IP 192.168.57.5.53811 > 192.168.57.3.1194: UDP, length 40
20:24:07.038575 IP 192.168.57.3.1194 > 192.168.57.5.53811: UDP, length 40
20:24:07.073169 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 53, length 64
20:24:07.073225 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 53, length 64
20:24:08.074347 IP 192.168.57.5 > 192.168.57.3: ICMP echo request, id 4, seq 54, length 64
20:24:08.074411 IP 192.168.57.3 > 192.168.57.5: ICMP echo reply, id 4, seq 54, length 64
```

- sudo tcdump -n -i tun0

  En este caso, nos encontramos en una interfaz virtual (tun0), por lo tanto tendremos que realizar el mismo proceso que anteriormente pero dándole al cliente la IP virtual del servidor (10.8.0.1). En las siguientes imágenes vemos el ping realizado a través de dicha interfaz.

```
mininet@mininet:~/Desktop$ ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=1.46 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=1.27 ms
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=1.35 ms
64 bytes from 10.8.0.1: icmp_seq=4 ttl=64 time=1.66 ms
64 bytes from 10.8.0.1: icmp_seq=5 ttl=64 time=1.50 ms
64 bytes from 10.8.0.1: icmp_seq=6 ttl=64 time=1.52 ms
64 bytes from 10.8.0.1: icmp_seq=7 ttl=64 time=1.41 ms
64 bytes from 10.8.0.1: icmp_seq=8 ttl=64 time=1.48 ms
64 bytes from 10.8.0.1: icmp_seq=9 ttl=64 time=1.47 ms
64 bytes from 10.8.0.1: icmp_seq=10 ttl=64 time=1.44 ms
64 bytes from 10.8.0.1: icmp_seq=11 ttl=64 time=1.43 ms
64 bytes from 10.8.0.1: icmp_seq=12 ttl=64 time=1.27 ms
64 bytes from 10.8.0.1: icmp_seq=13 ttl=64 time=1.76 ms
64 bytes from 10.8.0.1: icmp_seq=14 ttl=64 time=1.04 ms
64 bytes from 10.8.0.1: icmp_seq=15 ttl=64 time=1.41 ms
64 bytes from 10.8.0.1: icmp_seq=16 ttl=64 time=1.46 ms
64 bytes from 10.8.0.1: icmp_seq=17 ttl=64 time=1.26 ms
64 bytes from 10.8.0.1: icmp_seq=18 ttl=64 time=1.30 ms
64 bytes from 10.8.0.1: icmp_seq=19 ttl=64 time=1.35 ms
64 bytes from 10.8.0.1: icmp_seq=20 ttl=64 time=0.908 ms
```

```
mininet@mininet:~/Desktop$ sudo tcpdump -n -i tun0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
20:25:29.631693 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 5, seq 9, length 64
20:25:29.631798 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 5, seq 9, length 64
20:25:30.634322 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 5, seq 10, length 64
20:25:30.634391 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 5, seq 10, length 64
20:25:31.636097 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 5, seq 11, length 64
20:25:31.636136 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 5, seq 11, length 64
20:25:32.638193 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 5, seq 12, length 64
20:25:32.638232 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 5, seq 12, length 64
20:25:33.640532 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 5, seq 13, length 64
20:25:33.640568 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 5, seq 13, length 64
20:25:34.641645 IP 10.8.0.6 > 10.8.0.1: ICMP echo request, id 5, seq 14, length 64
20:25:34.641680 IP 10.8.0.1 > 10.8.0.6: ICMP echo reply, id 5, seq 14, length 64
```

  Como podemos observar, en este caso también recibimos paquetes ICMP, tal como era de esperar.

Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

- **Check the network interfaces on both nodes (ifconfig -a) and verify the presence of the new virtual interface associated to the VPN.**

  Abriendo la configuración de ambas máquinas observamos que la interfaz virtual (tun0) ha aparecido. La primera imagen corresponde al cliente y la segunda al servidor.

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.8.0.6  netmask 255.255.255.255  destination 10.8.0.5
        inet6 fe80::4ffc:1222:ffe6:9d2c  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 100  (UNSPEC)
        RX packets 609  bytes 50940 (50.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 611  bytes 51036 (51.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.8.0.1  netmask 255.255.255.255  destination 10.8.0.2
        inet6 fe80::c978:fa4a:30cd:61a0  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 100  (UNSPEC)
        RX packets 603  bytes 50652 (50.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 612  bytes 51084 (51.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- **Check also the routes defined on each node (netstat -r).**

  A través de ejecutar *netstat -r* en los terminales de ambas VM observamos las siguientes rutas (la primera imagen corresponde al cliente y la segunda al servidor). Donde podemos observar que están creadas las interfaces virtuales tun0.

```
mininet@mininet:~/Desktop$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.8.0.1        10.8.0.5        255.255.255.255 UGH       0 0          0 tun0
10.8.0.5        0.0.0.0         255.255.255.255 UH        0 0          0 tun0
link-local      0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
172.17.0.0      0.0.0.0         255.255.0.0     U         0 0          0 docker0
192.168.57.0    0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
```

```
mininet@mininet:~/Desktop$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
10.8.0.0        10.8.0.2        255.255.255.0   UG        0 0          0 tun0
10.8.0.2        0.0.0.0         255.255.255.255 UH        0 0          0 tun0
link-local      0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
172.17.0.0      0.0.0.0         255.255.0.0     U         0 0          0 docker0
192.168.57.0    0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
```

- **Stop the client and server and repeat the two points above. Spot the differencies between the two situations.**

  Si ahora paramos la VPN tanto en el cliente como en el servidor y volvemos a comprobar las listas anteriores obtenemos los siguientes resultados:

```
mininet@mininet:~/Desktop$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
link-local      0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
172.17.0.0      0.0.0.0         255.255.0.0     U         0 0          0 docker0
192.168.57.0    0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
```

```
mininet@mininet:~/Desktop$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
link-local      0.0.0.0         255.255.0.0     U         0 0          0 enp0s3
172.17.0.0      0.0.0.0         255.255.0.0     U         0 0          0 docker0
192.168.57.0    0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
```

En estos podemos observar que una vez parados la interfaz virtual ha desaparecido y solo tenemos información de las interfaces físicas.

- **Now open a new terminal on first client node and ping the IP addresses of server node and second client node. Can the client see both nodes? You probably noticed that clients cannot ping each other over the VPN addresses. Think for a moment about how the nodes are connected over the virtual tun0 interfaces… there must be something in the configurations that prevents this. Time to google a bit for a solution. Notice that only client to client communication is requested.**

```
mininet@mininet:~/Desktop$ ping 192.168.57.6
PING 192.168.57.6 (192.168.57.6) 56(84) bytes of data.
64 bytes from 192.168.57.6: icmp_seq=1 ttl=64 time=0.754 ms
64 bytes from 192.168.57.6: icmp_seq=2 ttl=64 time=0.482 ms
64 bytes from 192.168.57.6: icmp_seq=3 ttl=64 time=0.953 ms
64 bytes from 192.168.57.6: icmp_seq=4 ttl=64 time=0.523 ms
^C
--- 192.168.57.6 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.482/0.678/0.953/0.189 ms
mininet@mininet:~/Desktop$ ping 10.8.0.10
ping: connect: Network is unreachable
```

Como se puede observar en la imagen, al hacer ping entre los dos clientes, en el caso de hacerlo a través de la IP física no hay problema alguno y se ven correctamente, en cambio al realizarlo a través de la IP que proporciona la VPN vemos que no hay visibilidad entre ellos ya que está opción está desactivada en las opciones del server.conf.

- **Once you have found the change involved, apply it to the configuration file that needs it and restart the affected process.**

```
203 # Uncomment this directive to allow different
204 # clients to be able to "see" each other.
205 # By default, clients will only see the server.
206 # To force clients to only see the server, you
207 # will also need to appropriately firewall the
208 # server's TUN/TAP interface.
209 client-to-client
```

Dentro de server.conf hemos buscado la opción de client-to-client y hemos descomentado la línea 209 que previamente estaba comentada ya que está opción estaba deshabilitada.

Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

- **Verify that now you can indeed ping from client to client**.

```
mininet@mininet:~/Desktop$ ping 10.8.0.10
PING 10.8.0.10 (10.8.0.10) 56(84) bytes of data.
64 bytes from 10.8.0.10: icmp_seq=1 ttl=64 time=2.43 ms
64 bytes from 10.8.0.10: icmp_seq=2 ttl=64 time=0.963 ms
64 bytes from 10.8.0.10: icmp_seq=3 ttl=64 time=1.29 ms
64 bytes from 10.8.0.10: icmp_seq=4 ttl=64 time=2.30 ms
^C
--- 10.8.0.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.963/1.745/2.429/0.631 ms
```

Una vez guardados los cambios previos observamos que hay visibilidad entre clientes a través de la IP virtual/proporcionada por la VPN.

- **While running a continuous ping from client1 to client2, start a network capture in a terminal window on the server node, using sudo tcpdump -n -i any port 1194 or icmp; this must output a trace of each packet involving openvpn (port 1194) or the ping command (icmp). Do the same on a terminal window on client2 node.**

  What do you see on each window, and which addresses are involved? How do you interpret these results? What is "real", and what is "virtual" in this setup?

```
mininet@mininet:~/Desktop$ sudo tcpdump -n -i any port 1194 or icmp
[sudo] password for mininet:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
17:08:32.727626 IP 192.168.57.5.60783 > 192.168.57.3.1194: UDP, length 108
17:08:32.727821 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:08:32.728708 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
17:08:32.728832 IP 192.168.57.3.1194 > 192.168.57.5.60783: UDP, length 108
17:08:33.730093 IP 192.168.57.5.60783 > 192.168.57.3.1194: UDP, length 108
17:08:33.730357 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:08:33.731200 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
17:08:33.731412 IP 192.168.57.3.1194 > 192.168.57.5.60783: UDP, length 108
17:08:34.731500 IP 192.168.57.5.60783 > 192.168.57.3.1194: UDP, length 108
17:08:34.731694 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:08:34.732130 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
17:08:34.732193 IP 192.168.57.3.1194 > 192.168.57.5.60783: UDP, length 108
17:08:35.733512 IP 192.168.57.5.60783 > 192.168.57.3.1194: UDP, length 108
17:08:35.733760 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:08:35.734263 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
17:08:35.734405 IP 192.168.57.3.1194 > 192.168.57.5.60783: UDP, length 108
17:08:36.736161 IP 192.168.57.5.60783 > 192.168.57.3.1194: UDP, length 108
17:08:36.736365 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:08:36.737060 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
17:08:36.737190 IP 192.168.57.3.1194 > 192.168.57.5.60783: UDP, length 108
^C
20 packets captured
20 packets received by filter
0 packets dropped by kernel
```

Sara Soriano - 240007
Rubén Vera - 241456
Eneko Treviño - 241679

```
mininet@mininet:~/Desktop$ sudo tcpdump -n -i any port 1194 or icmp
[sudo] password for mininet:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
17:09:57.606492 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:09:57.606880 IP 10.8.0.6 > 10.8.0.10: ICMP echo request, id 5, seq 113, length 64
17:09:57.606993 IP 10.8.0.10 > 10.8.0.6: ICMP echo reply, id 5, seq 113, length 64
17:09:57.607063 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
17:09:58.608711 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:09:58.608984 IP 10.8.0.6 > 10.8.0.10: ICMP echo request, id 5, seq 114, length 64
17:09:58.609012 IP 10.8.0.10 > 10.8.0.6: ICMP echo reply, id 5, seq 114, length 64
17:09:58.609045 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
17:09:59.609987 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:09:59.610222 IP 10.8.0.6 > 10.8.0.10: ICMP echo request, id 5, seq 115, length 64
17:09:59.610247 IP 10.8.0.10 > 10.8.0.6: ICMP echo reply, id 5, seq 115, length 64
17:09:59.610274 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
17:10:00.613380 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:10:00.613553 IP 10.8.0.6 > 10.8.0.10: ICMP echo request, id 5, seq 116, length 64
17:10:00.613583 IP 10.8.0.10 > 10.8.0.6: ICMP echo reply, id 5, seq 116, length 64
17:10:00.613618 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
17:10:01.615368 IP 192.168.57.3.1194 > 192.168.57.6.44978: UDP, length 108
17:10:01.615504 IP 10.8.0.6 > 10.8.0.10: ICMP echo request, id 5, seq 117, length 64
17:10:01.615521 IP 10.8.0.10 > 10.8.0.6: ICMP echo reply, id 5, seq 117, length 64
17:10:01.615541 IP 192.168.57.6.44978 > 192.168.57.3.1194: UDP, length 108
^C
20 packets captured
20 packets received by filter
0 packets dropped by kernel
```

IP’s Cliente 1:192.168.57.5//10.8.0.5

IP’s cliente 2: 192.168.57.6//10.8.0.10

IP’s server: 192.168.57.3//10.8.0.1

Al realizar un ping del cliente 1 al cliente 2 a través de la ip virtual, primeramente el cliente 1 solicita al servidor el ping, el trackeador ve la petición de ping en el servidor y envía el ping al cliente 2 mediante la IP física, es decir hace de puente para poder realizarse el ping a través del puerto 1194. Y luego, en el cliente 2 vemos tanto el tráfico mediante IP’s físicas como el tráfico mediante las IP’s virtuales, solo que no ve como se hace el bridge, simplemente ve que envia el hecho y le llega una reply.