

Empezamos con la instalación.

- Descarga el [disco virtual que contiene strongSwan y OpenVPN](#) y copia sus archivos (*OMNET6-0*.vmdk*) a la carpeta de la máquina virtual OMNET.
- En VMware Workstation selecciona la máquina virtual OMNET y pincha en *Edit virtual machine settings*. En la pestaña *Hardware* haz click en *Add*. Seleccióna *Hard Disk > SCSI > Use an existing virtual disk*. Escoge el fichero *OMNET6-0.vmdk*. Ten **CUIDADO** en NO seleccionar el fichero *OMNET.vmdk*, correspondiente al disco original de la máquina virtual.
- Arranca la máquina virtual.
- Desde el Firefox de la máquina virtual, descarga el *script de comandos strongswan_install.sh*.
- Añade permiso de ejecución al *script* mediante: *chmod +x strongswan_install.sh*
- Ejecuta el *script* mediante: *./strongswan_install.sh*
- Sitúate en el directorio de pruebas de strongSwan: *cd /srv/strongswan-testing/testing*
- Comprueba que el entorno de pruebas funciona correctamente: *sudo ./start-testing* (todos los elementos OK).
- Para el entorno de pruebas: *sudo ./stop-testing*

```
user@omnetvm2024:~/Desktop$ cat strongswan_install.sh
#!/bin/bash

sudo bash -c "echo 'UUID=e871d5de-e713-453f-8030-2956aefe20a5 /srv auto nosuid,nodev,nofail 0 0' >> /etc/fstab"
sudo mount /srv
sudo ln -s /srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing /srv/strongswan-testing/testing

sudo apt install qemu-system-x86 libvirt0 libvirt-daemon libvirt-daemon-system
sudo bash -c "echo 'security_driver = \"none\"' >> /etc/libvirt/qemu.conf"
user@omnetvm2024:~/Desktop$
```

Cuando ejecuto el script *./strongswan_install.sh*:

```
Created symlink /etc/systemd/system/multi-user.target.wants/virtlockd.service → /usr/lib/systemd/system/virtlockd.service.
Created symlink /etc/systemd/system/multi-user.target.wants/virtlogd.service → /usr/lib/systemd/system/virtlogd.service.
Setting up libvirt-daemon dnsmasq configuration.
Setting up qemu-system-modules-opengl (1:8.2.2+ds-0ubuntu1.10) ...
Setting up qemu-system-gui (1:8.2.2+ds-0ubuntu1.10) ...
Setting up qemu-system-modules-spice (1:8.2.2+ds-0ubuntu1.10) ...
Setting up liblvm2cmd2:0.3+ndde4 (2.03.16-3ubuntu3.2) ...
Setting up dmeventd (2:1.32.185-3ubuntu3.2) ...
dm-event.service is disabled because static unit not running, not starting it.
Setting up liblvm2 (2.03.16-3ubuntu3.2) ...
Processing triggers for libbus (1.14.10-4ubuntu4.1) ...
Processing triggers for desktop-file-utils (0.27-2build1) ...
Processing triggers for initramfs-tools (0.142ubuntu25.1) ...
update-initramfs: Generating /boot/initrd.img-6.8.0-85-generic
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.6) ...
Processing triggers for man-db (2.12.0-4build2) ...
user@omnetvm2024:~/Desktop$
```

Al ejecutar *sudo ./start-testing*:

```
user@omnetvm2024:/srv/strongswan-testing/testing$ sudo ./start-testing
Starting test environment
[ ok ] Deploying kernel linux-5.19.11
[ ok ] Deploying /srv/strongswan-testing/build/shared/bullseye as hostfs
[ ok ] Deploying /srv/strongswan-testing/testresults as hostfs
[ ok ] Network vnet1
[ ok ] Network vnet2
[ ok ] Network vnet3
[ ok ] Guest alice
[ ok ] Guest bob
[ ok ] Guest carol
[ ok ] Guest dave
[ ok ] Guest moon
[ ok ] Guest sun
[ ok ] Guest venus
[ ok ] Guest winnetou
[ ok ]
```

Para *sudo ./stop-testing*:

```

user@omninetvm2024:/srv/strongswan-testing/testing$ sudo ./stop-testing
Stopping test environment
[ ok ] Network vnet1
[ ok ] Network vnet2
[ ok ] Network vnet3
[ ok ] Guest alice
[ ok ] Guest bob
[ ok ] Guest carol
[ ok ] Guest dave
[ ok ] Guest moon
[ ok ] Guest sun
[ ok ] Guest venus
[ ok ] Guest winnetou
[ ok ] Removing kernel linux-5.19.11
[ ok ] Removing link to hostfs
[ ok ] Removing link to testresults
user@omninetvm2024:/srv/strongswan-testing/testing$ 

```

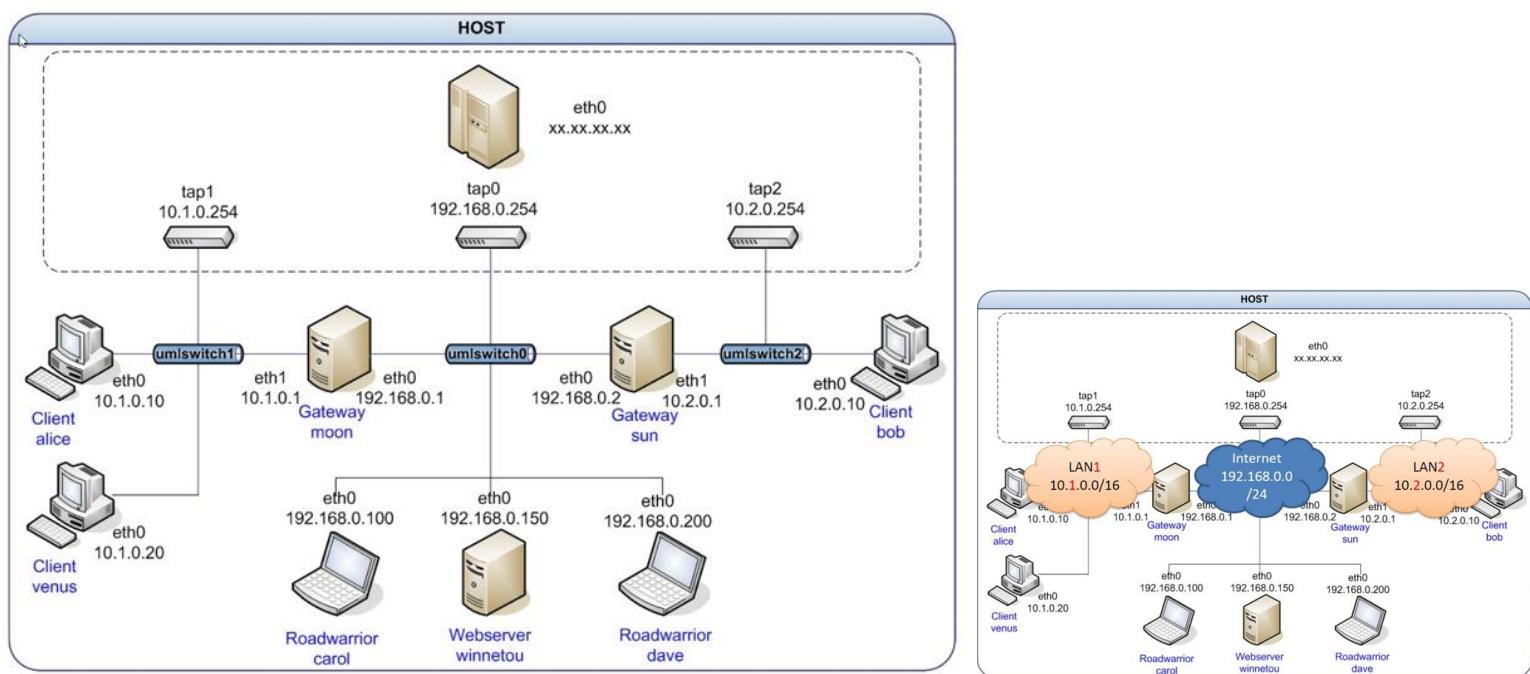
Ahora sí, una vez terminada la instalación, comenzamos con el taller.

MEMORIA TALLER SOBRE VPNs: strongSwan (IPsec) y OpenVPN (TLS).

PARTE 1 DEL TALLER

Conceptos básicos de enrutamiento y cortafuegos

Red de pruebas



Esta es la red de pruebas que acabamos de instalar. En la figura de la derecha podemos ver las subredes involucradas. La idea de esta red es poder mirar el escenario de 2 sedes distintas de una organización (redes LAN1 y LAN2) conectadas a través de Internet. En internet, además de servidores externos a la organización (como un servidor web), también hay equipos de la propia organización (llamados **roadwarrior**) a los que dar conectividad con las redes LAN1 y LAN2 de la organización.

Este entorno de pruebas se puede usar para otros experimentos no relacionados con strongSwan. Por ejemplo, la usaremos también red de pruebas para OpenVPN.

IMPORTANTE

- Todos los comandos **DEBEN** ser ejecutados como root. Para ello, en el equipo anfitrión haz `sudo bash`.

Para conectarte a cada máquina, abre otra pestaña en la terminal y ejecuta ssh <remotehost>.

```
user@omninetvm2024:/srv/strongswan-testing/testing$ sudo bash  
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing#
```

- Todos los comandos en equipo anfitrión asumen que estás situado en el directorio **/srv/strongswan-5.9.8/testing**.
- En las pruebas de conectividad mediante ping, “→” indica probar en una sola dirección y “<->” en ambas, ya que no se puede asumir que, aunque hay conectividad en una dirección, la haya a la contraria.

PRUEBAS BÁSICAS:

1. Levanta la red de pruebas: `./start-testing`

2. Comprueba si cada equipo tiene activado el entramiento: `sysctl -a | grep ip_forward`

Alice:

```
alice:~# sysctl -a | grep ip_forward  
net.ipv4.ip_forward = 1  
net.ipv4.ip_forward_update_priority = 1  
net.ipv4.ip_forward_use_pmtu = 0
```

Venus:

```
venus:~# sysctl -a | grep ip_forward  
net.ipv4.ip_forward = 1  
net.ipv4.ip_forward_update_priority = 1  
net.ipv4.ip_forward_use_pmtu = 0
```

Carol:

```
carol:~# sysctl -a | grep ip_forward  
net.ipv4.ip_forward = 1  
net.ipv4.ip_forward_update_priority = 1  
net.ipv4.ip_forward_use_pmtu = 0
```

Winnetou:

```
carol:~# sysctl -a | grep ip_forward  
net.ipv4.ip_forward = 1  
net.ipv4.ip_forward_update_priority = 1  
net.ipv4.ip_forward_use_pmtu = 0
```

Dave:

```
dave:~# sysctl -a | grep ip_forward  
net.ipv4.ip_forward = 1  
net.ipv4.ip_forward_update_priority = 1  
net.ipv4.ip_forward_use_pmtu = 0
```

Bob:

```
bob:~# sysctl -a | grep ip_forward  
net.ipv4.ip_forward = 1  
net.ipv4.ip_forward_update_priority = 1  
net.ipv4.ip_forward_use_pmtu = 0
```

Muestra las tablas de enrutamiento de los equipos alice, bob, moon, sun, carol y winnetou.

Alice:

```
alice:~# ip route show  
default via 10.1.0.1 dev eth0 onlink  
10.1.0.0/16 dev eth0 proto kernel scope link src 10.1.0.10  
10.8.0.1 via 10.8.0.5 dev tun0  
10.8.0.5 dev tun0 proto kernel scope link src 10.8.0.6
```

Bob:

```
bob:~# ip route show  
default via 10.2.0.1 dev eth0 onlink  
10.2.0.0/16 dev eth0 proto kernel scope link src 10.2.0.10
```

Moon:

```
moon:~# ip route show  
default via 192.168.0.254 dev eth0 onlink  
10.1.0.0/16 dev eth1 proto kernel scope link src 10.1.0.1  
10.8.0.0/24 via 10.8.0.2 dev tun0  
10.8.0.2 dev tun0 proto kernel scope link src 10.8.0.1  
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.1
```

Sun:

```
sun:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.2.0.0/16 dev eth1 proto kernel scope link src 10.2.0.1
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.2
```

Carol:

```
carol:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.8.0.1 via 10.8.0.9 dev tun0
10.8.0.9 dev tun0 proto kernel scope link src 10.8.0.10
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.100
```

Winnetou:

```
winnetou:~# ip route show
default via 192.168.0.254 dev eth0 onlink
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.150
```

A la vista de las tablas de enrutamiento y comprobando mediante ping desde cada uno de los equipos:

- ¿Es posible alcanzar bob desde alice? ¿Desde moon? ¿Desde carol?

Desde Alice:

```
alice:~# ping -c 3 10.2.0.10
PING 10.2.0.10 (10.2.0.10) 56(84) bytes of data.
From 10.1.0.254 icmp_seq=1 Destination Port Unreachable
From 10.1.0.254 icmp_seq=2 Destination Port Unreachable
From 10.1.0.254 icmp_seq=3 Destination Port Unreachable

--- 10.2.0.10 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2014ms
```

Desde Moon:

```
root@omninetm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing/hosts/moon/etc/network# ssh 192.168.0.1
moon:~# ping -c 3 10.2.0.10
PING 10.2.0.10 (10.2.0.10) 56(84) bytes of data.
From 192.168.0.254 icmp_seq=1 Destination Port Unreachable
From 192.168.0.254 icmp_seq=2 Destination Port Unreachable
From 192.168.0.254 icmp_seq=3 Destination Port Unreachable

--- 10.2.0.10 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2018ms
```

Desde Carol:

```
carol:~# ping -c 3 10.2.0.10
PING 10.2.0.10 (10.2.0.10) 56(84) bytes of data.
From 192.168.0.254 icmp_seq=1 Destination Port Unreachable
From 192.168.0.254 icmp_seq=2 Destination Port Unreachable
From 192.168.0.254 icmp_seq=3 Destination Port Unreachable

--- 10.2.0.10 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2018ms
```

- ¿Es posible alcanzar carol desde moon? ¿Desde winnetou? ¿Desde bob?

Desde Moon:

```
moon:~# ping -c 3 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=64 time=6.45 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=64 time=3.45 ms
64 bytes from 192.168.0.100: icmp_seq=3 ttl=64 time=2.31 ms

--- 192.168.0.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2020ms
rtt min/avg/max/mdev = 2.313/4.073/6.454/1.746 ms
```

Desde Winnetou:

```
winnetou:~# ping -c 3 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=64 time=3.50 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=64 time=3.23 ms
64 bytes from 192.168.0.100: icmp_seq=3 ttl=64 time=3.61 ms

--- 192.168.0.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2022ms
rtt min/avg/max/mdev = 3.233/3.447/3.611/0.158 ms
```

Desde bob:

```
bob:~# ping -c 3 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.

--- 192.168.0.100 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2038ms
```

- ¿Es posible alcanzar alice desde sun? ¿Desde winnetou? ¿Desde moon?

Desde Sun:

```
sun:~# ping -c 3 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
From 192.168.0.254 icmp_seq=1 Destination Port Unreachable
From 192.168.0.254 icmp_seq=2 Destination Port Unreachable
From 192.168.0.254 icmp_seq=3 Destination Port Unreachable

--- 10.1.0.10 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2033ms
```

Desde Winnetou:

```
winnetou:~# ping -c 3 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
From 192.168.0.254 icmp_seq=1 Destination Port Unreachable
From 192.168.0.254 icmp_seq=2 Destination Port Unreachable
From 192.168.0.254 icmp_seq=3 Destination Port Unreachable

--- 10.1.0.10 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2016ms
```

Desde Moon:

```
moon:~# ping 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
64 bytes from 10.1.0.10: icmp_seq=1 ttl=64 time=3.90 ms
64 bytes from 10.1.0.10: icmp_seq=2 ttl=64 time=2.17 ms
64 bytes from 10.1.0.10: icmp_seq=3 ttl=64 time=1.96 ms
^C
--- 10.1.0.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2016ms
rtt min/avg/max/mdev = 1.957/2.675/3.898/0.868 ms
moon:~# █
```

¿Cuáles son las direcciones del equipo omnet6-vm? ¿Qué direcciones del equipo anfitrión es posible alcanzar desde cada equipo?

```
root@omnetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing/hosts/moon/etc/network# ip -br a
lo      UNKNOWN    127.0.0.1/8 ::/128
ens192     UP        192.168.62.130/24 fe80::879e:625a:dcb1:3a5f/64
virbr0      DOWN      192.168.122.1/24
test-br0      UP        192.168.0.254/24
test-br1      UP        10.1.0.254/16
test-br2      UP        10.2.0.254/16
alice-eth0    UNKNOWN   fe80::fc54:ff:fe9a:e2de/64
alice-eth1    UNKNOWN   fe80::fc54:ff:fe3b:cd7/64
bob-eth0     UNKNOWN   fe80::fc54:ff:fe40:856b/64
carol-eth0    UNKNOWN   fe80::fc54:ff:feaef1f8/64
dave-eth0     UNKNOWN   fe80::fc54:ff:feb9:15a9/64
moon-eth1     UNKNOWN   fe80::fc54:ff:fe43:e335/64
moon-eth0     UNKNOWN   fe80::fc54:ff:fec7:b0b0/64
sun-eth0      UNKNOWN   fe80::fc54:ff:fe77:43ea/64
sun-eth1      UNKNOWN   fe80::fc54:ff:fe0f:97db/64
venus-eth0    UNKNOWN   fe80::fc54:ff:fe69:d380/64
winnetou-eth0 UNKNOWN   fe80::fc54:ff:fe4b:23fa/64
```

- Desde Alice:

```
alice:~# ping -c 3 10.1.0.254
PING 10.1.0.254 (10.1.0.254) 56(84) bytes of data.
64 bytes from 10.1.0.254: icmp_seq=1 ttl=64 time=1.22 ms
64 bytes from 10.1.0.254: icmp_seq=2 ttl=64 time=1.42 ms
64 bytes from 10.1.0.254: icmp_seq=3 ttl=64 time=1.82 ms

--- 10.1.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2021ms
rtt min/avg/max/mdev = 1.223/1.486/1.818/0.247 ms
alice:~# ping -c 3 10.2.0.254
PING 10.2.0.254 (10.2.0.254) 56(84) bytes of data.
64 bytes from 10.2.0.254: icmp_seq=1 ttl=64 time=7.24 ms
64 bytes from 10.2.0.254: icmp_seq=2 ttl=64 time=1.50 ms
64 bytes from 10.2.0.254: icmp_seq=3 ttl=64 time=1.94 ms

--- 10.2.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2025ms
rtt min/avg/max/mdev = 1.495/3.559/7.238/2.607 ms
alice:~# ping -c 3 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=6.58 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=2.55 ms
64 bytes from 192.168.0.254: icmp_seq=3 ttl=64 time=1.76 ms

--- 192.168.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2019ms
rtt min/avg/max/mdev = 1.759/3.630/6.582/2.111 ms
alice:~# █
```

- Desde bob:

```
bob:~# ping -c 3 10.1.0.254
PING 10.1.0.254 (10.1.0.254) 56(84) bytes of data.
64 bytes from 10.1.0.254: icmp_seq=1 ttl=64 time=6.44 ms
64 bytes from 10.1.0.254: icmp_seq=2 ttl=64 time=1.91 ms
64 bytes from 10.1.0.254: icmp_seq=3 ttl=64 time=1.75 ms

--- 10.1.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2020ms
rtt min/avg/max/mdev = 1.745/3.364/6.435/2.172 ms
bob:~# ping -c 3 10.2.0.254
PING 10.2.0.254 (10.2.0.254) 56(84) bytes of data.
64 bytes from 10.2.0.254: icmp_seq=1 ttl=64 time=4.67 ms
64 bytes from 10.2.0.254: icmp_seq=2 ttl=64 time=1.02 ms
64 bytes from 10.2.0.254: icmp_seq=3 ttl=64 time=1.06 ms

--- 10.2.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 1.021/2.249/4.667/1.709 ms
bob:~# ping -c 3 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=5.16 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=2.87 ms
64 bytes from 192.168.0.254: icmp_seq=3 ttl=64 time=5.85 ms

--- 192.168.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2025ms
rtt min/avg/max/mdev = 2.871/4.625/5.846/1.271 ms
```

- Desde carol:

```
carol:~# ping -c 3 10.1.0.254
PING 10.1.0.254 (10.1.0.254) 56(84) bytes of data.
64 bytes from 10.1.0.254: icmp_seq=1 ttl=64 time=6.11 ms
64 bytes from 10.1.0.254: icmp_seq=2 ttl=64 time=1.02 ms
64 bytes from 10.1.0.254: icmp_seq=3 ttl=64 time=0.875 ms

--- 10.1.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2024ms
rtt min/avg/max/mdev = 0.875/2.668/6.108/2.432 ms
carol:~# ping -c 3 10.2.0.254
PING 10.2.0.254 (10.2.0.254) 56(84) bytes of data.
64 bytes from 10.2.0.254: icmp_seq=1 ttl=64 time=4.69 ms
64 bytes from 10.2.0.254: icmp_seq=2 ttl=64 time=0.870 ms
64 bytes from 10.2.0.254: icmp_seq=3 ttl=64 time=0.903 ms

--- 10.2.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2024ms
rtt min/avg/max/mdev = 0.870/2.155/4.693/1.794 ms
carol:~# ping -c 3 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=1.99 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=0.859 ms
64 bytes from 192.168.0.254: icmp_seq=3 ttl=64 time=0.757 ms

--- 192.168.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2020ms
rtt min/avg/max/mdev = 0.757/1.200/1.986/0.556 ms
```

- Desde dave:

```
dave:~# ping -c 3 10.1.0.254
PING 10.1.0.254 (10.1.0.254) 56(84) bytes of data.
64 bytes from 10.1.0.254: icmp_seq=1 ttl=64 time=2.92 ms
64 bytes from 10.1.0.254: icmp_seq=2 ttl=64 time=1.06 ms
64 bytes from 10.1.0.254: icmp_seq=3 ttl=64 time=0.987 ms

--- 10.1.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2014ms
rtt min/avg/max/mdev = 0.987/1.654/2.915/0.891 ms
dave:~# ping -c 3 10.2.0.254
PING 10.2.0.254 (10.2.0.254) 56(84) bytes of data.
64 bytes from 10.2.0.254: icmp_seq=1 ttl=64 time=1.19 ms
64 bytes from 10.2.0.254: icmp_seq=2 ttl=64 time=1.23 ms
64 bytes from 10.2.0.254: icmp_seq=3 ttl=64 time=1.01 ms

--- 10.2.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 1.009/1.144/1.231/0.097 ms
dave:~# ping -c 3 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=3.78 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=0.843 ms
64 bytes from 192.168.0.254: icmp_seq=3 ttl=64 time=0.699 ms

--- 192.168.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2025ms
rtt min/avg/max/mdev = 0.699/1.774/3.781/1.420 ms
```

- Desde moon:

```
moon:~# ping -c 3 10.1.0.254
PING 10.1.0.254 (10.1.0.254) 56(84) bytes of data.
64 bytes from 10.1.0.254: icmp_seq=1 ttl=64 time=7.18 ms
64 bytes from 10.1.0.254: icmp_seq=2 ttl=64 time=0.871 ms
64 bytes from 10.1.0.254: icmp_seq=3 ttl=64 time=1.05 ms

--- 10.1.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2014ms
rtt min/avg/max/mdev = 0.871/3.032/7.180/2.933 ms
moon:~# ping -c 3 10.2.0.254
PING 10.2.0.254 (10.2.0.254) 56(84) bytes of data.
64 bytes from 10.2.0.254: icmp_seq=1 ttl=64 time=2.17 ms
64 bytes from 10.2.0.254: icmp_seq=2 ttl=64 time=1.20 ms
64 bytes from 10.2.0.254: icmp_seq=3 ttl=64 time=1.30 ms

--- 10.2.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2009ms
rtt min/avg/max/mdev = 1.195/1.554/2.167/0.435 ms
moon:~# ping -c 3 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=1.67 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=0.697 ms
64 bytes from 192.168.0.254: icmp_seq=3 ttl=64 time=1.34 ms

--- 192.168.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2021ms
rtt min/avg/max/mdev = 0.697/1.232/1.665/0.401 ms
```

- Desde sun:

```
sun:~# ping -c 3 10.1.0.254
PING 10.1.0.254 (10.1.0.254) 56(84) bytes of data.
64 bytes from 10.1.0.254: icmp_seq=1 ttl=64 time=3.36 ms
64 bytes from 10.1.0.254: icmp_seq=2 ttl=64 time=1.12 ms
64 bytes from 10.1.0.254: icmp_seq=3 ttl=64 time=0.953 ms

--- 10.1.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2028ms
rtt min/avg/max/mdev = 0.953/1.808/3.356/1.096 ms
sun:~# ping -c 3 10.2.0.254
PING 10.2.0.254 (10.2.0.254) 56(84) bytes of data.
64 bytes from 10.2.0.254: icmp_seq=1 ttl=64 time=2.41 ms
64 bytes from 10.2.0.254: icmp_seq=2 ttl=64 time=0.949 ms
64 bytes from 10.2.0.254: icmp_seq=3 ttl=64 time=2.38 ms

--- 10.2.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2020ms
rtt min/avg/max/mdev = 0.949/1.914/2.412/0.682 ms
sun:~# ping -c 3 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=1.28 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=1.00 ms
64 bytes from 192.168.0.254: icmp_seq=3 ttl=64 time=1.10 ms

--- 192.168.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2017ms
rtt min/avg/max/mdev = 1.003/1.128/1.281/0.115 ms
```

- Desde venus:

```
venus:~# ping -c 3 10.1.0.254
PING 10.1.0.254 (10.1.0.254) 56(84) bytes of data.
64 bytes from 10.1.0.254: icmp_seq=1 ttl=64 time=2.35 ms
64 bytes from 10.1.0.254: icmp_seq=2 ttl=64 time=0.903 ms
64 bytes from 10.1.0.254: icmp_seq=3 ttl=64 time=0.944 ms

--- 10.1.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2020ms
rtt min/avg/max/mdev = 0.903/1.398/2.347/0.671 ms
venus:~# ping -c 3 10.2.0.254
PING 10.2.0.254 (10.2.0.254) 56(84) bytes of data.
64 bytes from 10.2.0.254: icmp_seq=1 ttl=64 time=6.45 ms
64 bytes from 10.2.0.254: icmp_seq=2 ttl=64 time=2.66 ms
64 bytes from 10.2.0.254: icmp_seq=3 ttl=64 time=1.88 ms

--- 10.2.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 1.876/3.660/6.446/1.995 ms
venus:~# ping -c 3 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=2.97 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=3.20 ms
64 bytes from 192.168.0.254: icmp_seq=3 ttl=64 time=9.85 ms

--- 192.168.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2019ms
rtt min/avg/max/mdev = 2.974/5.340/9.848/3.188 ms
```

- Desde winnetou:

```
winnetou:~# ping -c 3 10.1.0.254
PING 10.1.0.254 (10.1.0.254) 56(84) bytes of data.
64 bytes from 10.1.0.254: icmp_seq=1 ttl=64 time=1.88 ms
64 bytes from 10.1.0.254: icmp_seq=2 ttl=64 time=0.872 ms
64 bytes from 10.1.0.254: icmp_seq=3 ttl=64 time=0.805 ms

--- 10.1.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2024ms
rtt min/avg/max/mdev = 0.805/1.184/1.877/0.490 ms
winnetou:~# ping -c 3 10.2.0.254
PING 10.2.0.254 (10.2.0.254) 56(84) bytes of data.
64 bytes from 10.2.0.254: icmp_seq=1 ttl=64 time=1.45 ms
64 bytes from 10.2.0.254: icmp_seq=2 ttl=64 time=1.13 ms
64 bytes from 10.2.0.254: icmp_seq=3 ttl=64 time=0.847 ms

--- 10.2.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2028ms
rtt min/avg/max/mdev = 0.847/1.142/1.447/0.245 ms
winnetou:~# ping -c 3 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=1.34 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=1.05 ms
64 bytes from 192.168.0.254: icmp_seq=3 ttl=64 time=1.19 ms

--- 192.168.0.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2024ms
rtt min/avg/max/mdev = 1.045/1.190/1.338/0.119 ms
```

PERMITIR LA CONECTIVIDAD ENTRE EQUIPOS Y REDES:

Introduce una ruta en carol para permitir la conectividad alice <-> carol:

```
carol:~# ip route add 10.1.0.10 via 192.168.0.1
```

Comprueba ping alice <-> carol y carol -> venus:

Desde Alice:

```
alice:~# ping -c 3 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=63 time=5.70 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=63 time=3.92 ms
64 bytes from 192.168.0.100: icmp_seq=3 ttl=63 time=3.35 ms

--- 192.168.0.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 3.350/4.324/5.704/1.002 ms
```

Desde Carol:

```
carol:~# ping -c 3 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
64 bytes from 10.1.0.10: icmp_seq=1 ttl=63 time=6.02 ms
64 bytes from 10.1.0.10: icmp_seq=2 ttl=63 time=4.50 ms
64 bytes from 10.1.0.10: icmp_seq=3 ttl=63 time=5.15 ms

--- 10.1.0.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 4.502/5.221/6.015/0.619 ms
carol:~# ping -c 3 10.1.0.20
PING 10.1.0.20 (10.1.0.20) 56(84) bytes of data.
From 192.168.0.254 icmp_seq=1 Destination Port Unreachable
From 192.168.0.254 icmp_seq=2 Destination Port Unreachable
From 192.168.0.254 icmp_seq=3 Destination Port Unreachable

--- 10.1.0.20 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2034ms
```

- Borra esa ruta:

```
carol:~# ip route del 10.1.0.10
```

- Introduce una ruta en carol para permitir la conectividad de la red 10.1.0.0/16 <-> carol

```
carol:~# ip route add 10.1.0.0/16 via 192.168.0.1
```

- Comprueba ping alice <-> carol y carol → venus:

Desde Alice:

```
alice:~# ping -c 3 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=63 time=3.50 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=63 time=4.12 ms
64 bytes from 192.168.0.100: icmp_seq=3 ttl=63 time=3.55 ms

--- 192.168.0.100 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 3.499/3.723/4.122/0.282 ms
```

Desde Carol:

```
carol:~# ping -c 3 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
64 bytes from 10.1.0.10: icmp_seq=1 ttl=63 time=5.46 ms
64 bytes from 10.1.0.10: icmp_seq=2 ttl=63 time=3.72 ms
64 bytes from 10.1.0.10: icmp_seq=3 ttl=63 time=3.87 ms

--- 10.1.0.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 3.718/4.351/5.463/0.788 ms
carol:~# ping -c 3 10.1.0.20
PING 10.1.0.20 (10.1.0.20) 56(84) bytes of data.
64 bytes from 10.1.0.20: icmp_seq=1 ttl=63 time=6.45 ms
64 bytes from 10.1.0.20: icmp_seq=2 ttl=63 time=8.13 ms
64 bytes from 10.1.0.20: icmp_seq=3 ttl=63 time=3.31 ms

--- 10.1.0.20 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2022ms
rtt min/avg/max/mdev = 3.311/5.965/8.131/1.997 ms
```

- Borra esa ruta:

```
carol:~# ip route del 10.1.0.0/16
carol:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.8.0.1 via 10.8.0.9 dev tun0
10.8.0.9 dev tun0 proto kernel scope link src 10.8.0.10
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.100
```

- Añade rutas en moon y sun para conectar las redes LAN1 y LAN2.

Desde moon:

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 192.168.0.1
moon:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.1.0.0/16 dev eth1 proto kernel scope link src 10.1.0.1
10.2.0.0/16 via 192.168.0.2 dev eth0
10.8.0.0/24 via 10.8.0.2 dev tun0
10.8.0.2 dev tun0 proto kernel scope link src 10.8.0.1
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.1
```

ip route add 10.2.0.0/16 via 192.168.0.2 dev eth0

Desde sun:

```
sun:~# ip route add 10.1.0.0/16 via 192.168.0.1 dev eth0
sun:~# █
```

Comprobamos:

```
alice:~# ping -c 3 10.2.0.10
PING 10.2.0.10 (10.2.0.10) 56(84) bytes of data.
64 bytes from 10.2.0.10: icmp_seq=1 ttl=62 time=5.58 ms
64 bytes from 10.2.0.10: icmp_seq=2 ttl=62 time=3.54 ms
64 bytes from 10.2.0.10: icmp_seq=3 ttl=62 time=3.78 ms
```

```
bob:~# ping -c 3 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
64 bytes from 10.1.0.10: icmp_seq=1 ttl=62 time=6.06 ms
64 bytes from 10.1.0.10: icmp_seq=2 ttl=62 time=11.7 ms
64 bytes from 10.1.0.10: icmp_seq=3 ttl=62 time=5.34 ms
```

- Borra esas rutas en moon y sun

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 192.168.0.1
moon:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.1.0.0/16 dev eth1 proto kernel scope link src 10.1.0.1
10.2.0.0/16 via 192.168.0.2 dev eth0
10.8.0.0/24 via 10.8.0.2 dev tun0
10.8.0.2 dev tun0 proto kernel scope link src 10.8.0.1
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.1
moon:~# ip route del 10.2.0.0/16 via 192.168.0.2 dev eth0
moon:~# exit
logout
Connection to 192.168.0.1 closed.
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 192.168.0.2
sun:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.1.0.0/16 via 192.168.0.1 dev eth0
10.2.0.0/16 dev eth1 proto kernel scope link src 10.2.0.1
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.2
sun:~# ip route del 10.1.0.0/16 via 192.168.0.1 dev eth0
```

CONCEPTOS BÁSICOS DE UN CORTAFUEGOS

Muestra la tabla de enrutamiento y las reglas de filtrado del cortafuegos del equipo anfitrión:

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# route -n
Command 'route' not found, but can be installed with:
apt install net-tools
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# apt install net-tools
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libgl1-amber-dri libglapi-mesa libllvm17f64 python3-netifaces
```

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	192.168.62.2	0.0.0.0	UG	100	0	0	ens192
10.1.0.0	0.0.0.0	255.255.0.0	U	0	0	0	test-br1
10.2.0.0	0.0.0.0	255.255.0.0	U	0	0	0	test-br2
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	test-br0
192.168.62.0	0.0.0.0	255.255.255.0	U	100	0	0	ens192
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# iptables -L -v
```

Chain INPUT (policy ACCEPT 12623 packets, 260M bytes)

pkts	bytes	target	prot	opt	in	out	source	destination
12623	260M	LIBVIRT_INP	all	--	any	any	anywhere	anywhere
0	0	LIBVIRT_FWX	all	--	any	any	anywhere	anywhere
0	0	LIBVIRT_FWI	all	--	any	any	anywhere	anywhere
0	0	LIBVIRT_FWO	all	--	any	any	anywhere	anywhere

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

pkts	bytes	target	prot	opt	in	out	source	destination
0	0	LIBVIRT_FWX	all	--	any	any	anywhere	anywhere
0	0	LIBVIRT_FWI	all	--	any	any	anywhere	anywhere
0	0	LIBVIRT_FWO	all	--	any	any	anywhere	anywhere

Chain OUTPUT (policy ACCEPT 10179 packets, 580K bytes)

pkts	bytes	target	prot	opt	in	out	source	destination
10179	580K	LIBVIRT_OUT	all	--	any	any	anywhere	anywhere

Chain LIBVIRT_FWI (1 references)

pkts	bytes	target	prot	opt	in	out	source	destination
0	0	ACCEPT	all	--	lo	test-br2	anywhere	10.2.0.0/16
0	0	REJECT	all	--	any	test-br2	anywhere	anywhere
0	0	ACCEPT	all	--	lo	test-br1	anywhere	10.1.0.0/16
0	0	REJECT	all	--	any	test-br1	anywhere	anywhere
0	0	ACCEPT	all	--	lo	test-br0	anywhere	192.168.0.0/24
0	0	REJECT	all	--	any	test-br0	anywhere	anywhere
0	0	ACCEPT	all	--	any	virbr0	anywhere	192.168.122.0/24
0	0	REJECT	all	--	any	virbr0	anywhere	anywhere

Chain LIBVIRT_FWO (1 references)

pkts	bytes	target	prot	opt	in	out	source	destination
------	-------	--------	------	-----	----	-----	--------	-------------

Chain LIBVIRT_INP (1 references)

pkts	bytes	target	prot	opt	in	out	source	destination
0	0	ACCEPT	udp	--	test-br2	any	anywhere	anywhere
0	0	ACCEPT	tcp	--	test-br2	any	anywhere	tcp dpt:domain
0	0	ACCEPT	udp	--	test-br2	any	anywhere	udp dpt:bootps
0	0	ACCEPT	tcp	--	test-br2	any	anywhere	tcp dpt:67
0	0	ACCEPT	udp	--	test-br1	any	anywhere	udp dpt:domain
0	0	ACCEPT	tcp	--	test-br1	any	anywhere	tcp dpt:domain
0	0	ACCEPT	udp	--	test-br1	any	anywhere	udp dpt:bootps
0	0	ACCEPT	tcp	--	test-br1	any	anywhere	tcp dpt:67
0	0	ACCEPT	udp	--	test-br1	any	anywhere	anywhere
0	0	ACCEPT	tcp	--	test-br0	any	anywhere	anywhere
0	0	ACCEPT	udp	--	test-br0	any	anywhere	anywhere
0	0	ACCEPT	tcp	--	test-br0	any	anywhere	anywhere
0	0	ACCEPT	udp	--	virbr0	any	anywhere	anywhere
0	0	ACCEPT	tcp	--	virbr0	any	anywhere	anywhere
0	0	ACCEPT	udp	--	virbr0	any	anywhere	anywhere
0	0	ACCEPT	tcp	--	virbr0	any	anywhere	anywhere

Chain LIBVIRT_OUT (1 references)

pkts	bytes	target	prot	opt	in	out	source	destination
0	0	ACCEPT	udp	--	any	test-br2	anywhere	anywhere
0	0	ACCEPT	tcp	--	any	test-br2	anywhere	tcp dpt:domain
0	0	ACCEPT	udp	--	any	test-br2	anywhere	udp dpt:bootpc
0	0	ACCEPT	tcp	--	any	test-br2	anywhere	tcp dpt:68
0	0	ACCEPT	udp	--	any	test-br1	anywhere	udp dpt:domain
0	0	ACCEPT	tcp	--	any	test-br1	anywhere	tcp dpt:domain
0	0	ACCEPT	udp	--	any	test-br1	anywhere	udp dpt:bootpc
0	0	ACCEPT	tcp	--	any	test-br1	anywhere	tcp dpt:68
0	0	ACCEPT	udp	--	any	test-br0	anywhere	anywhere
0	0	ACCEPT	tcp	--	any	test-br0	anywhere	anywhere
0	0	ACCEPT	udp	--	any	test-br0	anywhere	anywhere
0	0	ACCEPT	tcp	--	any	test-br0	anywhere	anywhere
0	0	ACCEPT	udp	--	any	virbr0	anywhere	anywhere
0	0	ACCEPT	tcp	--	any	virbr0	anywhere	anywhere
0	0	ACCEPT	udp	--	any	virbr0	anywhere	anywhere
0	0	ACCEPT	tcp	--	any	virbr0	anywhere	anywhere

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing#
```

(Recuerda que en lugar de iptables -L [-v] siempre puedes utilizar iptables -S [-v] para visualizar las reglas en otro formato.

Muestra la tabla nat del cortafuegos del equipo anfitrión:

```

root@omnetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# iptables -t nat -L -v
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target  prot opt in     out      source          destination
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target  prot opt in     out      source          destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target  prot opt in     out      source          destination
Chain POSTROUTING (policy ACCEPT 291 packets, 27971 bytes)
 pkts bytes target  prot opt in     out      source          destination
 291 27971 LIBVIRT_PRT all  --  any    any    anywhere       anywhere
Chain LIBVIRT_PRT (1 references)
 pkts bytes target  prot opt in     out      source          destination
   0    0 RETURN  all  --  any    any    192.168.122.0/24 base-address.mcast.net/24
   0    0 RETURN  all  --  any    any    192.168.122.0/24 255.255.255.255
   0    0 MASQUERADE  tcp  --  any    any    192.168.122.0/24 !192.168.122.0/24 masq ports: 1024-65535
   0    0 MASQUERADE  udp  --  any    any    192.168.122.0/24 !192.168.122.0/24 masq ports: 1024-65535
   0    0 MASQUERADE  all  --  any    any    192.168.122.0/24 !192.168.122.0/24

```

- ¿Qué reglas iptables aíslan el tráfico de cada una de las 3 redes?

Con las **3 redes** se refiere a:

- LAN1: 10.1.0.0/16
- LAN2: 10.2.0.0/16
- Internet: 192.168.0.0/24

En la salida del comando `iptables -L -v`, dentro de la cadena `LIBVIRT_FWI` se ven las reglas:

```

Chain LIBVIRT_FWI (1 references)
 pkts bytes target  prot opt in     out      source          destination
   0    0 ACCEPT  all  --  lo    test-br2 anywhere       10.2.0.0/16
   0    0 REJECT  all  --  any   test-br2 anywhere       anywhere      reject-with icmp-port-unreachable
   0    0 ACCEPT  all  --  lo    test-br1 anywhere       10.1.0.0/16
   0    0 REJECT  all  --  any   test-br1 anywhere       anywhere      reject-with icmp-port-unreachable
   0    0 ACCEPT  all  --  lo    test-br0 anywhere      192.168.0.0/24
   0    0 REJECT  all  --  any   test-br0 anywhere       anywhere      reject-with icmp-port-unreachable

```

Estas 3 líneas son las que **bloquean el tráfico** entre las redes virtuales.

- REJECT ... 10.1.0.0/16: Bloquea cualquier paquete dirigido a la red LAN1.
- REJECT ... 10.2.0.0/16: Bloquea cualquier paquete dirigido a la red LAN2.
- REJECT ... 192.168.0.0/16: Bloquea cualquier paquete hacia la red intermedia (Internet simulada).

Estas reglas están puestas en el **firewall del host** (no dentro de las VMs), y son las responsables de **aislar las subredes del entorno de pruebas** entre sí.

Así evitan que haya tráfico directo desde el host real o entre redes distintas del laboratorio, para mantener el escenario controlado.

- Prueba a añadir como primera regla de la cadena FORWARD un “aceptar todo”, y comprueba si cambia algo en la conectividad entre los equipos de distintas redes:

Lo hago y comprobamos con los pings:

```

root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ping 192.168.0.150
PING 192.168.0.150 (192.168.0.150) 56(84) bytes of data.
64 bytes from 192.168.0.150: icmp_seq=1 ttl=64 time=11.1 ms
64 bytes from 192.168.0.150: icmp_seq=2 ttl=64 time=1.77 ms
^C
--- 192.168.0.150 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 1.772/6.440/11.108/4.668 ms
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ping 10.2.0.10
PING 10.2.0.10 (10.2.0.10) 56(84) bytes of data.
64 bytes from 10.2.0.10: icmp_seq=1 ttl=64 time=6.70 ms
64 bytes from 10.2.0.10: icmp_seq=2 ttl=64 time=3.46 ms
^C
--- 10.2.0.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 3.460/5.078/6.697/1.618 ms
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ping 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
64 bytes from 10.1.0.10: icmp_seq=1 ttl=64 time=6.67 ms
64 bytes from 10.1.0.10: icmp_seq=2 ttl=64 time=2.27 ms
^C
--- 10.1.0.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1014ms
rtt min/avg/max/mdev = 2.272/4.470/6.669/2.198 ms
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# █

```

Aquí lo que estamos haciendo es insertar una nueva regla en la posición 1 de la cadena FORWARD, que acepta todo el tráfico entre interface santes de que las reglas de LIBVIRT_FWI (las que bloquean las redes) se apliquen.

En resumen: esta regla **anula el aislamiento** que había entre las redes 10.1.0.0/16, 10.2.0.0/16 y 192.168.0.0/24, porque al estar en primer lugar, todo paquete pasa antes de ser rechazado.

Qué ocurre en la práctica:

Las tres redes (LAN1, LAN2 e Internet simulada) **vuelven a tener conectividad completa**.

Ahora sí podrías hacer pings o traceroutes entre alice, bob, carol, etc., y los paquetes circularían libremente. En otras palabras: **el cortafuegos deja de aislar** los segmentos de red.

- Elimina la regla anterior:

```

root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# iptables -D FORWARD 1
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# █

```

De esta forma se restaura el aislamiento entre las redes.

NAT BÁSICO

SOURCE NAT (MASQUERADE)

- Habilita SNAT (tipo MASQUERADE) en moon para permitir al conexión desde la red 10.1.0.0/16 a la red 192.168.0.0/24. Muestra las iptables (chains) de al tabla NAT de moon.

```

moon:~# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
moon:~# iptables -t nat -L -v --line-numbers
Chain PREROUTING (policy ACCEPT 1 packets, 42 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain INPUT (policy ACCEPT 1 packets, 42 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain OUTPUT (policy ACCEPT 1 packets, 120 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
1      1    120 MASQUERADE  all   --  any    eth0    anywhere
moon:~# █

```

Esa regla permite que los equipos de la red 10.1.0.0/16 (LAN1) puedan comunicarse con la red 192.168.0.0/24 usando al IP del propio router (192.168.0.1) como dirección de origen.

La salida obtenida confirma que la regla se añadió correctamente en la cadena POSTROUTING, siendo el objetivo MASQUERADE aplicado sobre la interfaz eth0.

Gracias a ello, el router moon realiza traducción de direcciones (SNAT) para que los equipos de la red **10.1.0.0/16** puedan comunicarse con la red **192.168.0.0/24** utilizando su propia IP como origen.

- Comprueba la conectividad alice -> carol y la ausencia de conectividad carol -> alice, primero mediante ping y traceroute y luego mediante ssh. Una vez establecida la conexión ssh alice -> carol, muestra las conexiones en carol mediante netstat -4an.

```

root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 10.1.0.10
alice:~# ping 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=63 time=6.25 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=63 time=5.58 ms
^C
--- 192.168.0.100 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 5.583/5.918/6.254/0.335 ms
alice:~# traceroute 192.168.0.100
traceroute to 192.168.0.100 (192.168.0.100), 30 hops max, 60 byte packets
1 moon1.strongswan.org (10.1.0.1) 2.750 ms 2.810 ms 1.558 ms
2 carol.strongswan.org (192.168.0.100) 3.512 ms 4.368 ms 3.811 ms

```

El ping desde alice a carol funciona, porque moon traduce (NAT) las IPs de salida hacia la red 192.168.0.0/24.

En el traceroute, se ve que los paquetes pasan por moon (10.1.0.1 -> 192.168.0.1 -> 192.168.0.100).

Importante: aunque carol reciba los paquetes, no sabrá que vienen de 10.1.0.10, sino del router moon (192.168.0.1).

```

root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 192.168.0.100
carol:~# ping 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
From 192.168.0.254 icmp_seq=1 Destination Port Unreachable
From 192.168.0.254 icmp_seq=2 Destination Port Unreachable
^C
--- 10.1.0.10 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1013ms

carol:~# traceroute 10.1.0.10
traceroute to 10.1.0.10 (10.1.0.10), 30 hops max, 60 byte packets
1 uml0.strongswan.org (192.168.0.254) 3.052 ms 3.268 ms 3.368 ms
2 uml0.strongswan.org (192.168.0.254) 3.255 ms 3.188 ms 3.124 ms

```

No hay respuesta. Esto pasa porque el NAT solo está definido en sentido **salida por eth0**. El tráfico inverso no tiene una regla NAT que lo permita y por tanto no encuentra la ruta de vuelta.

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 10.1.0.10
alice:~# ssh 192.168.0.100
Warning: Permanently added '192.168.0.100' (ECDSA) to the list of known hosts.
carol:~#
```

Desde alice se estableció una conexión SSH con carol. La conexión fu eexitosa, confirmando que el tráfico TCP también es traducido corretamente por la regla de MASQUERADE aplicada en moon.

```
carol:~# ss -t4n
State      Recv-Q      Send-Q      Local Address:Port      Peer Address:Port      Process
ESTAB      0            0           192.168.0.100:22    192.168.0.1:55458      /usr/bin/python3
```

En cuanto a las IPs de Origen:

Carol ve como IP origen: 192.168.0.1 (la IP del router moon -> por el MASQUERADE).

Alice ve como IP origen: 192.168.0.100 (la IP real de Carol).

- Si ejecutamos ss -t4n en el router moon, ¿vemos alguna conexión que muestre el NAT que realiza?

```
moon:~# netstat -4an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp     0      0 0.0.0.0:22                0.0.0.0:*              LISTEN
tcp     0      0 192.168.0.1:22             192.168.0.254:59008   ESTABLISHED
udp     0      0 0.0.0.0:1194              0.0.0.0:*
```

La salida muestra únicamente las conexiones propias del sistema. No se observo ninguna conexión entre **alice** y **carol**, lo que confirma que el router **no mantiene conexiones TCP propias** para el tráfico reenviado.

El proceso de **traducción NAT** se realiza a nivel del **kernel**, modificando las cabeceras IP de los paquetes sin que aparezcan como sockets activos en netstat.

En resumen, el NAT enmascara las IPs de los paquetes sin establecer conexiones propias visibles para el sistema.

- Elimina la regla de MASQUERADE

```
moon:~# iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
```

- Efectúa un MASQUERADE desde moon que permita solo a **alice** conectar con la red 192.168.0.0/24

```
moon:~# iptables -t nat -A POSTROUTING -o eth0 -s 10.1.0.10 -j MASQUERADE
moon:~# iptables -t nat -L -v --line-numbers
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source               destination
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source               destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source               destination
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source               destination
1      0      0 MASQUERADE  all    --    any    eth0    alice.strongswan.org  anywhere
```

Comprobamos desde alice:

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 10.1.0.10
alice:~# ping 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=63 time=11.0 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=63 time=5.92 ms
```

Funciona bien.

Comprobamos desde venus:

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 10.1.0.20
venus:~# ping 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
^C
--- 192.168.0.100 ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6138ms
```

No funciona.

Esta regla permite que solo los paquetes con origen 10.1.0.10 sean traducidos al salir por la interfaz eth0 hacia la red 192.168.0.0/24, bloqueando implícitamente el acceso de otros equipos de la LAN1 (como Venus).

Al hacer las comprobaciones con los pings, se demuestra que el NAT selectivo se aplica correctamente únicamente al host especificado.

- Elimina la regla de MASQUERADE

```
moon:~# iptables -t nat -D POSTROUTING -o eth0 -s 10.1.0.10 -j MASQUERADE
moon:~#
```

Tras eliminarla, Alice pierde nuevamente la conectividad con la red 192.168.0.0/24, confirmando que la regla anterior era la responsable de la traducción NAT.

- Efectúa MASQUERADE desde *moon* que permita solo a alice conectar con el servidor web en winnetou.

```
moon:~# iptables -t nat -L -v --line-numbers
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
1      0      0 MASQUERADE  tcp   --  any    eth0    alice.strongswan.org anywhere        tcp dpt:http
```

Esta regla permite que únicamente el tráfico HTTP generado por Alice sea traducido al salir por eth0 hacia la red 192.168.0.0/24, impidiendo el acceso a otros equipos o protocolos.

- Compruébalo con ssh -p 80 winnetou en *alice*.

```
alice:~# ssh -p 80 192.168.0.150
kex_exchange_identification: Connection closed by remote host
Connection closed by 192.168.0.150 port 80
```

La conexión fue aceptada, aunque cerró la conexión porque el puerto 80 no es SSH, por tanto la regla NAT funciona correctamente.

- Comprueba ahora esas posibles conexiones mediante ping desde cada uno de los equipos.

```

alice:~# ping 192.168.0.150
PING 192.168.0.150 (192.168.0.150) 56(84) bytes of data.
^C
--- 192.168.0.150 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3089ms

alice:~# exit
logout
Connection to 10.1.0.10 closed.
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 10.1.0.20
venus:~# ping 192.168.0.150
PING 192.168.0.150 (192.168.0.150) 56(84) bytes of data.
^C
--- 192.168.0.150 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1013ms

```

Al probar conectividad con ping hacia Winnetou (192.168.0.150) desde **Alice** y **Venus**, ambos fallaron. Esto es correcto, ya que la regla de AMSQUERADE solo aplica a tráfico **TCP con destino puerto 80**, no a ICMP. De esta forma, únicamente las conexiones HTTP de Alice serían traducidas y aceptadas.

- Elimina la regla de MASQUERADE.

```

moon:~# iptables -t nat -D POSTROUTING -o eth0 -s 10.1.0.10 -p tcp --dport 80 -j MASQUERADE

```

Tras borrarla, Alice ya no puede iniciar conexiones HTTP hacia la red 192.168.0.0/24, recuperando el aislamiento original.

DESTINATION NAT

- Habilita DNAT para mapear el puerto 10022 de *moon* al puerto ssh de *alice*:

```

moon:~# iptables -t nat -A PREROUTING -p tcp --dport 10022 -j DNAT --to-destination 10.1.0.10:22
moon:~# iptables -t nat -L -v --line-numbers
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source               destination
1      0    0 DNAT       tcp   --  any    any     anywhere            anywhere          tcp  dpt:10022  to:10.1.0.10:22
22

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source               destination

```

Esto permite que cualquier conexión que llegue a moon:10022 sea reenviada automáticamente al servicio SSH de Alice.

- Comprobamos ssh carol -> alice vía DNAT

```

root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 192.168.0.100
carol:~# ssh 192.168.0.1 -p 10022
Warning: Permanently added '[192.168.0.1]:10022' (ECDSA) to the list of known hosts.
alice:~# 

```

La conexión fue redirigida al servicio SSH de **Alice (10.1.0.10:22)**, demostrando que el router **moon** está traduciendo correctamente el destino de la conexión.

Así, el acceso a moon:10022 equivale a conectarse directamente con alice:22.

- Elimina la regla de DNAT

```

moon:~# iptables -t nat -D PREROUTING -p tcp --dport 10022 -j DNAT --to-destination 10.1.0.10:22
moon:~# 

```

Tras eliminarla, el acceso a moon:10022 dejó de redirigir a Alice, restaurando el comportamiento original del router.

PRUEBAS DE VELOCIDAD

- Realiza una prueba de velocidad alice -> moon.

```
moon:~# iperf -s
-----
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 4] local 192.168.0.1 port 5001 connected with 10.1.0.10 port 51248
[ ID] Interval      Transfer      Bandwidth
[ 4] 0.0000-10.0203 sec  1.08 GBytes  926 Mbits/sec
```

```
alice:~# iperf -c 192.168.0.1 -i 1
-----
Client connecting to 192.168.0.1, TCP port 5001
TCP window size: 85.0 KByte (default)
-----
[ 3] local 10.1.0.10 port 51248 connected with 192.168.0.1 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 3] 0.0000-1.0000 sec  115 MBytes  966 Mbits/sec
[ 3] 1.0000-2.0000 sec  134 MBytes  1.12 Gbits/sec
[ 3] 2.0000-3.0000 sec  116 MBytes  971 Mbits/sec
[ 3] 3.0000-4.0000 sec  120 MBytes  1.01 Gbits/sec
[ 3] 4.0000-5.0000 sec  95.9 MBytes  804 Mbits/sec
[ 3] 5.0000-6.0000 sec  100 MBytes  841 Mbits/sec
[ 3] 6.0000-7.0000 sec  99.9 MBytes  838 Mbits/sec
[ 3] 7.0000-8.0000 sec  107 MBytes  899 Mbits/sec
```

Los resultados mostraron una tasa de transferencia media entre **900 Mbit/s y 1.12 Gbit/s**, con un total transferido de **1.08 GBytes** en 10 segundos.

En el servidor **Moon** se observó la recepción correspondiente con una velocidad final de **926 Mbit/s**, confirmando una conexión muy rápida y estable entre ambos nodos.

- Y moon -> alice

```
alice:~# iperf -s
-----
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 4] local 10.1.0.10 port 5001 connected with 10.1.0.1 port 34776
[ ID] Interval      Transfer      Bandwidth
[ 4] 0.0000-6.7753 sec  653 MBytes  808 Mbits/sec
```

```
moon:~# iperf -c 10.1.0.10 -i 1
-----
Client connecting to 10.1.0.10, TCP port 5001
TCP window size: 85.0 KByte (default)
-----
[ 3] local 10.1.0.1 port 34776 connected with 10.1.0.10 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 3] 0.0000-1.0000 sec  89.3 MBytes  749 Mbits/sec
[ 3] 1.0000-2.0000 sec  99.2 MBytes  833 Mbits/sec
[ 3] 2.0000-3.0000 sec  98.1 MBytes  823 Mbits/sec
[ 3] 3.0000-4.0000 sec  96.4 MBytes  808 Mbits/sec
[ 3] 4.0000-5.0000 sec  93.6 MBytes  785 Mbits/sec
[ 3] 5.0000-6.0000 sec  99.0 MBytes  830 Mbits/sec
^C[ 3] 6.0000-6.7694 sec  77.2 MBytes  842 Mbits/sec
[ 3] 0.0000-6.7694 sec  653 MBytes  809 Mbits/sec
moon:~#
```

En esta dirección, la velocidad media fue de **entre 749 Mbit/s y 842 Mbit/s**, con una transferencia total de **653 MB en 6,77 segundos**.

Aunque el rendimiento sigue siendo alto, se observa una ligera reducción de velocidad respecto a la dirección contraria (Alice → Moon).

- ¿Hay alguna diferencia de velocidad?

Las pruebas de iperf muestran un ancho de banda **muy alto y prácticamente simétrico** entre ambos equipos, con pequeñas variaciones atribuibles a la carga del sistema o al control de flujo de TCP.

En general, la conexión **Alice ↔ Moon** alcanza entre **800 y 900 Mbit/s**, lo que demuestra un enlace de red eficiente dentro del entorno de virtualización.

PARTE 2 DEL TALLER

IPsec: strongSwan

Se recomienda examinar la documentación disponible de cada test.

- NOTA: Cada vez que termines de probar un escenario IPsec, recuerda **restaurar las reglas iptables originales** para permitir de nuevo la conectividad entre equipos mediante:

```
* moon$ iptables-restore < /etc/iptables.flush  
* sun$ iptables-restore < /etc/iptables.flush
```

- Problemas con la MTU: En cada *Gateway IPsec* (en nuestro caso, *moon* y *sun*) debe hacerse un ajuste del MSS (*Maximum Segment Size*) TCP anunciado en las conexiones iniciadas por los clientes que no son conscientes de la existencia de IPsec. (escenarios net2net y wr). El ajuste consiste en reducir el MSS TCP anunciado en los paquetes SYN y RST a 1360 bytes, de modo que el MSS TCP efectivo teniendo en cuenta el *overhead* de IPsec no supere los 1460 bytes. Ten en cuenta que 1460 bytes es el máximo tamaño de segmento para una MTU de 1500 bytes, ya que se necesitan 20 bytes para la cabecera IP y 20 bytes para la TCP. Las siguientes reglas en la tabla *mangle* realizan este ajuste:

Antes de iniciar cada escenario de IPsec, se restauraron las reglas originales de iptables tanto en *moon* como en *sun*:

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 192.168.0.1  
moon:~# iptables-restore < /etc/iptables.flush  
moon:~# exit  
logout  
Connection to 192.168.0.1 closed.  
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 192.168.0.2  
sun:~# iptables-restore < /etc/iptables.flush  
ip6tables.flush iproute2/      ipsec.d/      iptables.drop    iptables.rules  
ip6tables.rules ipsec.conf     ipsec.secrets   iptables.flush  
sun:~# iptables-restore < /etc/iptables.flush  
sun:~#
```

Esto garantiza que no haya interferencias con las configuraciones previas de NAT, DNAT o MASQUERADE y que las pruebas IPsec se realicen en un entorno limpio.

Cuando se usan túneles IPsec, los paquetes TCP deben ser más pequeños porque IPsec añade cabeceras extra.

Si no se ajusta el tamaño máximo de segmento (MSS), los clientes pueden tener **problemas para establecer conexiones TCP**, aunque el ping funcione.

Por eso se modifican las reglas en la tabla **mangle** para reducir el MSS a 1360 bytes (el máximo permitido para una MTU de 1500 con el overhead de IPsec).

```
sun:~# ip tables -t mangle -A FORWARD -m policy --pol ipsec --dir in -p tcp -m tcp --tcp-flags SYN,RST SYN -m tcpmss -mss 1361:1536 -j TCPMSS --set-mss 1360  
Object "tables" is unknown, try "ip help".  
sun:~# iptables -t mangle -A FORWARD -m policy --pol ipsec --dir in -p tcp -m tcp --tcp-flags SYN,RST SYN -m tcpmss -mss 1361:1536 -j TCPMSS --set-mss 1360  
iptables v1.8.7 (legacy): Couldn't load match `ss':No such file or directory  
  
Try `iptables -h' or `iptables --help' for more information.  
sun:~# iptables -t mangle -A FORWARD -m policy --pol ipsec --dir in -p tcp -m tcp --tcp-flags SYN,RST SYN -m tcpmss --mss 1361:1536 -j TCPMSS --set-mss 1360  
sun:~# iptables -t mangle -A FORWARD -m policy --pol ipsec --dir out -p tcp -m tcp --tcp-flags SYN,RST SYN -m tcpmss --mss 1361:1536 -j TCPMSS --set-mss 1360  
sun:~# exit
```

Comprobamos:

```

sun:~# iptables -t mangle -L -v --line-numbers
Chain PREROUTING (policy ACCEPT 143 packets, 12559 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain INPUT (policy ACCEPT 143 packets, 12559 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
1      0      0 TCPMSS      tcp   --  any    any    anywhere        anywhere      policy match dir in pol
ipsec  tcp flags:SYN,RST/SYN  tcpmss match 1361:1536 TCPMSS set 1360
2      0      0 TCPMSS      tcp   --  any    any    anywhere        anywhere      policy match dir out pol
ipsec  tcp flags:SYN,RST/SYN  tcpmss match 1361:1536 TCPMSS set 1360

Chain OUTPUT (policy ACCEPT 87 packets, 11072 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain POSTROUTING (policy ACCEPT 87 packets, 11072 bytes)
num  pkts bytes target     prot opt in     out      source          destination

```

```

root@omnetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# ssh 192.168.0.1
moon:~# iptables -t mangle -A FORWARD -m policy --pol ipsec --dir in -p tcp --tcp-flags SYN,RST SYN -m tcpmss --mss 1361:1536 -j TCPMS
S --set-mss 1360
moon:~# iptables -t mangle -A FORWARD -m policy --pol ipsec --dir o
ut -p tcp --tcp-flags SYN,RST SYN -m tcpmss --mss 1361:1536 -j TCPM
SS --set-mss 1360
moon:~# exit

```

Comprobamos:

```

moon:~# iptables -t mangle -L -v --line-numbers
Chain PREROUTING (policy ACCEPT 131 packets, 17055 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain INPUT (policy ACCEPT 131 packets, 17055 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
1      0      0 TCPMSS      tcp   --  any    any    anywhere        anywhere      policy match dir in pol
ipsec  tcp flags:SYN,RST/SYN  tcpmss match 1361:1536 TCPMSS set 1360
2      0      0 TCPMSS      tcp   --  any    any    anywhere        anywhere      policy match dir out pol
ipsec  tcp flags:SYN,RST/SYN  tcpmss match 1361:1536 TCPMSS set 1360

Chain OUTPUT (policy ACCEPT 79 packets, 14183 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain POSTROUTING (policy ACCEPT 79 packets, 14183 bytes)
num  pkts bytes target     prot opt in     out      source          destination

```

Con esto, el tamaño máximo de segmento TCP se reduce a 1360 bytes, lo que evita fragmentación y asegura el correcto funcionamiento de las conexiones TCP a través del túnel IPsec. Sin este ajuste, el ping seguiría funcionando, pero las conexiones TCP (como SSH o HTTP) fallarían.

IPsec: Escenario ikev2/esp-alg-null

- Carga el escenario:

* omnetvm2024\$ scripts/load-testconfig ikev2-algs/esp-alg-null

File Actions Edit View Help

root@omninetvm2024: /srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing ×

```
moon:~# exit
logout
Connection to 192.168.0.1 closed.
```

```
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# scripts/load-testconfig ikev2-algs/esp-alg-null
root@omninetvm2024:/srv/strongswan-testing/build/shared/bullseye/compile/strongswan-5.9.8/testing# █
```

Este escenario inicializa los equipos **Alice**, **Moon** y **Carol**, simulando un entorno con una conexión IPsec entre Carol y Moon usando IKEv2 y ESP sin cifrado de datos (algoritmo null).

-Abre una terminal con alice, moon y carol.

- Comprueba que no es posible alcanzar alice desde carol y viceversa.

```
root@omninetvm2024:/srv/strongswan-testing/testing# ssh 10.1.0.10
alice:~# ping -c 2 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
^C
--- 192.168.0.100 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1004ms

alice:~# exit
logout
Connection to 10.1.0.10 closed.
root@omninetvm2024:/srv/strongswan-testing/testing# ssh 192.168.0.100
carol:~# ping 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
From 192.168.0.254 icmp_seq=1 Destination Port Unreachable
From 192.168.0.254 icmp_seq=2 Destination Port Unreachable
From 192.168.0.254 icmp_seq=3 Destination Port Unreachable
^C
--- 10.1.0.10 ping statistics ---
3 packets transmitted, 0 received, +3 errors, 100% packet loss, time 2004ms
```

Ninguno de los dos hosts puede alcanzarse por ping, ya que todavía no existe un túnel IPsec activo entre Moon y Carol, y por tanto no había ruta segura que uniera ambas redes.

- Muestra las tablas de enrutamiento y las iptables de cada equipo.

Carol:

```
carol:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.8.0.1 via 10.8.0.9 dev tun0
10.8.0.9 dev tun0 proto kernel scope link src 10.8.0.10
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.100
carol:~# iptables -L -v --line-numbers
Chain INPUT (policy ACCEPT 44 packets, 2956 bytes)
num  pkts bytes target     prot opt in     out     source               destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out     source               destination
Chain OUTPUT (policy ACCEPT 29 packets, 3648 bytes)
num  pkts bytes target     prot opt in     out     source               destination
carol:~# █
```

Alice:

```
root@omninetvm2024:/srv/strongswan-testing/testing# ssh 10.1.0.10
alice:~# ip route show
default via 10.1.0.1 dev eth0 onlink
10.1.0.0/16 dev eth0 proto kernel scope link src 10.1.0.10
10.8.0.1 via 10.8.0.5 dev tun0
10.8.0.5 dev tun0 proto kernel scope link src 10.8.0.6
alice:~# iptables -L -v --line-numbers
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out      source          destination
```

Moon:

```
root@omninetvm2024:/srv/strongswan-testing/testing# ssh 192.168.0.1
moon:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.1.0.0/16 dev eth1 proto kernel scope link src 10.1.0.1
10.8.0.0/24 via 10.8.0.2 dev tun0
10.8.0.2 dev tun0 proto kernel scope link src 10.8.0.1
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.1
moon:~# iptables -L -v --line-numbers
Chain INPUT (policy ACCEPT 1044 packets, 160K bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain FORWARD (policy ACCEPT 2 packets, 168 bytes)
num  pkts bytes target     prot opt in     out      source          destination
Chain OUTPUT (policy ACCEPT 662 packets, 144K bytes)
num  pkts bytes target     prot opt in     out      source          destination
moon:~# █
```

Tras cargar el escenario ikev2/esp-alg-null, se verificaron las tablas de enrutamiento de Alice, Moon y Carol:

- Alice tiene la red local 10.1.0.0/16 y como Gateway a Moon (10.1.0.1).

- Carol pertenece a 192.168.0.0/24 con salida a 192.168.0.254.

- Moon dispone de interfaces hacia ambas redes, actuando como Gateway central.

Las rutas mostradas son las esperadas: las máquinas locales solo pueden comunicarse dentro de su subred hasta que se active el túnel IPsec.

Esto confirma que la topología base está correctamente cargada y lista para iniciar el servicio *strongSwan*.

- Inicia IPsec en moon y carol

```
* moon$ systemctl start strongswan
```

```
* carol$ systemctl start strongswan
```

```
moon:~# systemctl start strongswan
moon:~# exit
logout
Connection to 192.168.0.1 closed.
root@omninetvm2024:/srv/strongswan-testing/testing# ssh 192.168.0.100
carol:~# systemctl start strongswan
carol:~# █
```

Examina con atención el contenido del fichero de configuración /etc/swanctl/swanctl.conf de strongSwan en moon y en carol para este escenario. El significado de las opciones es intuitivo. En caso de duda, consulta la documentación de strongSwan, en particular la relativa al fichero swanctl.conf.

```
root@omninetvm2024:/srv/strongswan-testing/testing# ssh 192.168.0.1
moon:~# cat /etc/swanctl/swanctl.conf
connections {

    rw {
        local_addrs = 192.168.0.1

        local {
            auth = pubkey
            certs = moonCert.pem
            id = moon.strongswan.org
        }
        remote {
            auth = pubkey
        }
        children {
            net {
                local_ts = 10.1.0.0/16
                esp_proposals = null-sha256-x25519
            }
        }
        version = 2
        mobike = no
        proposals = aes128-sha256-x25519
    }
}
```

Lo definido en moon:

- connections.rw → conexión de tipo *roadwarrior* o “peer-to-peer”.
- local_addrs = 192.168.0.1 → IP del gateway *Moon* en la red 192.168.0.0/24.
- auth = pubkey + certs = moonCert.pem → autenticación por **certificado público**, usando el fichero moonCert.pem.
- children.net → define la política IPsec aplicada a las redes detrás del túnel.
- local_ts = 10.1.0.0/16 → red protegida localmente por *Moon* (la de Alice y Venus).
- esp_proposals = null-sha256-x25519 → tráfico ESP sin cifrado (**null**) pero con integridad SHA256 y curva elíptica X25519.
- mobike = no → no se permite la renegociación móvil (IKEv2 con IPs cambiantes).
- proposals = aes128-sha256-x25519 → cifrado y autenticación usados para la fase IKEv2 (negociación del túnel).

En resumen: Moon acepta conexiones IKEv2 autenticadas por certificado, por ESP sin cifrado pero autenticado.

```

root@omninetvm2024:/srv/strongswan-testing/testing# ssh 192.168.0.100
carol:~# cat /etc/swanctl/swanctl.conf
connections {

    home {
        local_addrs = 192.168.0.100
        remote_addrs = 192.168.0.1

        local {
            auth = pubkey
            certs = carolCert.pem
            id = carol@strongswan.org
        }
        remote {
            auth = pubkey
            id = moon.strongswan.org
        }
        children {
            home {
                remote_ts = 10.1.0.0/16
                esp_proposals = null-sha256-x25519
            }
        }
        version = 2
        mobike = no
        proposals = aes128-sha256-x25519
    }
}

```

Lo definido en Carol:

- local_addrs = 192.168.0.100 → IP de Carol.
- remote_addrs = 192.168.0.1 → IP de Moon, el servidor IPsec.
- auth = pubkey y certs = carolCert.pem → autenticación con certificado propio.
- id = carol@strongswan.org → identificador de Carol para la negociación IKEv2.
- remote_ts = 10.1.0.0/16 → red a la que accederá Carol a través del túnel.
- esp_proposals = null-sha256-x25519 → igual que en Moon, sin cifrado pero con integridad.
- proposals = aes128-sha256-x25519 → mismo conjunto de algoritmos para la fase IKE.

En resumen: Carol inicia una conexión IKEv2 hacia Moon autenticándose con su certificado y accediendo a la red 10.1.0.0/16 por un túnel ESP sin cifrado.

En el caso de moon:

- ¿Qué interfaz (dirección IP) de moon utiliza la conexión rw?

local_addrs = 192.168.0.1

La conexión rw en moon usa la interfaz eth0 con IP 192.168.0.2 como punto de enlace para la negociación IKEv2. Esta es la interfaz demoon dentro de la red 192.168.0.0/24, desde la cual se comunica con Carol (192.168.0.100).

- ¿Qué versión de IKE soporta?

versión = 2

La conexión definida en moon usa el protocolo IKEv2 (versión = 2) para la negociación y establecimiento del túnel IPsec.

- ¿Qué propuesta de protocolo de cifrado, integridad y grupo de intercambio de claves Diffie Hellman admite para la asociación de seguridad IKE?

proposals = aes128-sha256-x25519

Para la asociación de seguridad IKE, moon admite la siguiente propuesta de algoritmos:

- Cifrado: AES-128
- Integridad: SHA-256
- Intercambio de claves (DH): X25519

- ¿Qué propuesta de protocolo de cifrado, integridad y grupo de intercambio de claves Diffie Hellman admite para la asociación de seguridad ESP hija de la asociación de seguridad IKE?

Dentro del bloque children del fichero /etc/swanctl/swanctl.conf:

- esp_proposals = null-sha256-x25519

Esto define los algoritmos aplicados a la **asociación de seguridad hija ESP**.

Para la asociación de seguridad ESP hija de IKE, *moon* admite la siguiente propuesta:

- Cifrado: null (sin cifrado)
- Integridad: SHA-256
- Intercambio de claves (DH): X25519

Esto implica que los paquetes IPsec se autentican e integran, pero **no se cifran** lo cual permite analizar el tráfico en claro a nivel educativo.

En el caso de carol:

- ¿Qué interfaz (dirección IP) local y remota utiliza la conexión home?

local_addrs= 192.168.0.100

remote_addrs = 192.168.0.1

La conexión home de Carol usa como **dirección local** 192.168.0.100 (interfaz eth0) y como **dirección remota** 192.168.0.1, correspondiente al Gateway Moon.

- ¿Qué versión de IKE soporta?

versión = 2

La conexión definida en Carol utiliza el protocolo IKEv2 (version = 2) para el establecimiento del túnel IPsec.

- ¿Qué propuesta de protocolo de cifrado, integridad y grupo de intercambio de claves Diffie Hellman admite para la asociación de seguridad IKE?

proposals = aes128-sha256-x25519

Esto corresponde a la fase IKE (intercambio de claves). Desglose:

- aes128: Algoritmo de cifrado simétrico AES de 128 bits.
- sha256: Algoritmo de integridad/autenticación.
- x25519: Grupo Diffie-Hellman de curva elíptica (intercambio de claves).

- ¿Qué propuesta de protocolo de cifrado, integridad y grupo de intercambio de claves Diffie Hellman admite para la asociación de seguridad ESP hija de la asociación de seguridad IKE?

esp_proposals = null-sha256-x25519

Para la asociación de seguridad ESP hija de IKE, *Carol* admite la siguiente propuesta:

- Cifrado: null (sin cifrado)
- Integridad: SHA-256
- Intercambio de claves (DH): X25519

- Carga las conexiones.

```
root@omninetvm2024:/srv/strongswan-testing/testing# ssh 192.168.0.1
moon:~# expect-connection rw
moon:~# exit
logout
Connection to 192.168.0.1 closed.
root@omninetvm2024:/srv/strongswan-testing/testing# ssh 192.168.0.100
carol:~# expect-connection home
```

Se cargaron las conexiones IKEv2 mediante los comandos expect-connection rw en *moon* y expect-connection home en *carol*. Este paso inicializa las configuraciones IPsec sin establecer aún el túnel, dejando ambos extremos preparados para la negociación.

- Muestra las tablas de enrutamiento y las *iptables* de cada equipo. ¿Ha cambiado algo?

```
carol:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.8.0.1 via 10.8.0.9 dev tun0
10.8.0.9 dev tun0 proto kernel scope link src 10.8.0.10
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.100
carol:~# iptables -L -v --line-numbers
Chain INPUT (policy ACCEPT 1183 packets, 388K bytes)
num  pkts bytes target     prot opt in     out    source          destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target     prot opt in     out    source          destination
Chain OUTPUT (policy ACCEPT 1344 packets, 387K bytes)
num  pkts bytes target     prot opt in     out    source          destination
```

```
moon:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.1.0.0/16 dev eth1 proto kernel scope link src 10.1.0.1
10.8.0.0/24 via 10.8.0.2 dev tun0
10.8.0.2 dev tun0 proto kernel scope link src 10.8.0.1
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.1
moon:~# iptables -L -v --line-numbers
Chain INPUT (policy ACCEPT 3782 packets, 917K bytes)
num  pkts bytes target     prot opt in     out    source          destination
Chain FORWARD (policy ACCEPT 2 packets, 168 bytes)
num  pkts bytes target     prot opt in     out    source          destination
Chain OUTPUT (policy ACCEPT 2509 packets, 878K bytes)
num  pkts bytes target     prot opt in     out    source          destination
```

No, no hay cambios en las tablas de enrutamiento e iptables. Las rutas hacia las redes remotas solo se añadirán automáticamente cuando la conexión se inicie con éxito.

- Levanta ahora la conexión IPsec home desde carol

```
carol:~# swanctl --initiate --child home
[IKE] initiating IKE_SA home[1] to 192.168.0.1
[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FRAG_SUP) N(HASH_ALG) N(REDIR_SUP) ]
[NET] sending packet: from 192.168.0.100[500] to 192.168.0.1[500] (240 bytes)
[NET] received packet: from 192.168.0.1[500] to 192.168.0.100[500] (273 bytes)
[ENC] parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) CERTREQ N(FRAG_SUP) N(HASH_ALG) N(CHDLESS_SUP) N(MULT_AUTH) ]
[CFG] selected proposal: IKE:AES_CBC_128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
[IKE] received cert request for "C=CH, O=strongSwan Project, CN=strongSwan Root CA"
[IKE] sending cert request for "C=CH, O=strongSwan Project, CN=strongSwan Root CA"
```

- Comprueba que ahora sí es posible la conectividad carol -> alice.

```
carol:~# ping 10.1.0.10
PING 10.1.0.10 (10.1.0.10) 56(84) bytes of data.
64 bytes from 10.1.0.10: icmp_seq=1 ttl=63 time=11.1 ms
64 bytes from 10.1.0.10: icmp_seq=2 ttl=63 time=6.11 ms
^C
--- 10.1.0.10 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1019ms
rtt min/avg/max/mdev = 6.114/8.606/11.099/2.492 ms
```

- Muestra las reglas de enrutado mediante ip rule y observa que se ha añadido la tabla de enrutamiento que utiliza strongSwan con identificador 220. Muestre dicha tabla mediante ip route show table 220.

```
carol:~# ip rule show
0:      from all lookup local
220:    from all lookup 220
32766:  from all lookup main
32767:  from all lookup default
```

```
carol:~# ip route show table 220
10.1.0.0/16 via 192.168.0.1 dev eth0 proto static src 192.168.0.100
```

Tras establecer la conexión IPsec, se comprobaron las reglas de enrutamiento con el comando ip rule show, donde se observó la aparición de la regla lookup 220, correspondiente a la tabla interna utilizada por strongSwan para enrutar el tráfico protegido por IPsec.

Al listar su contenido con ip route show table 220 se mostraron las rutas asociadas a la red remota (10.1.0.0/16 en este caso), indicando que el tráfico destinado a dicha red se procesará mediante el túnel IPsec.

- Comprueba si es posible hacer un ping carol -> moon o moon -> carol.

```
carol:~# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=3.58 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=1.74 ms
^C
--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1012ms
rtt min/avg/max/mdev = 1.735/2.659/3.584/0.924 ms
carol:~# exit
logout
Connection to 192.168.0.100 closed.
root@omnetvm2024:/srv/strongswan-testing/testing# ssh 192.168.0.1
moon:~# ping 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=64 time=2.62 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=64 time=1.98 ms
^C
```

Se comprobó la conectividad bidireccional y se obtuvo respuesta en ambos casos, confirmando que el túnel IPsec está correctamente establecido y permite tráfico bidireccional entre Moon y Carol.

Muestra las conexiones IPsec en alice, moon y carol con el comando swanctl --list-sas

```
moon:~# swanctl --list-sas
rw: #1, ESTABLISHED, IKEv2, 88d260b288427213_i 0312154be5e831f3_r*
 local 'moon.strongswan.org' @ 192.168.0.1[500]
 remote 'carol@strongswan.org' @ 192.168.0.100[500]
 AES_CBC-128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
 established 5593s ago, rekeying in 8711s
 net: #2, reqid 1, INSTALLED, TUNNEL, ESP:NULL/HMAC_SHA2_256_128/CURVE_25519
 installed 2323s ago, rekeying in 1025s, expires in 1637s
 in cd9e4371, 10948 bytes, 156 packets, 8s ago
 out cee96e2, 8940 bytes, 204 packets, 8s ago
 local 10.1.0.0/16
 remote 192.168.0.100/32
No leaks detected, 1464 suppressed by whitelist
```

```
carol:~# swanctl --list-sas
home: #1, ESTABLISHED, IKEv2, 88d260b288427213_i* 0312154be5e831f3_r
 local 'carol@strongswan.org' @ 192.168.0.100[500]
 remote 'moon.strongswan.org' @ 192.168.0.1[500]
 AES_CBC-128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
 established 5627s ago, rekeying in 7492s
home: #2, reqid 1, INSTALLED, TUNNEL, ESP:NULL/HMAC_SHA2_256_128/CURVE_25519
 installed 2357s ago, rekeying in 1039s, expires in 1603s
 in cee96e2, 9024 bytes, 206 packets, 18s ago
 out cd9e4371, 11088 bytes, 158 packets, 18s ago
 local 192.168.0.100/32
 remote 10.1.0.0/16
No leaks detected, 1464 suppressed by whitelist
```

```
alice:~# swanctl --list-sas
connecting to 'unix:///var/run/charon.vici' failed: No such file or directory
Error: connecting to 'default' URI failed: No such file or directory
strongSwan 5.9.8 swanctl
usage:
  swanctl --list-sas [--ike <name>|--ike-id <id>] [--child <name>|--child-id <id>]
                [--raw|--pretty]
  --help          (-h) show usage information
  --ike           (-i) filter IKE_SAs by name
  --ike-id        (-I) filter IKE_SAs by unique identifier
  --child         (-c) filter CHILD_SAs by name
  --child-id      (-C) filter CHILD_SAs by unique identifier
  --noblock       (-n) don't wait for IKE_SAs in use
  --raw           (-r) dump raw response message
  --pretty         (-P) dump raw response message in pretty print
  --debug          (-v) set debug level, default: 1
  --options        (-+) read command line options from file
  --uri            (-u) service URI to connect to
No leaks detected, 1463 suppressed by whitelist
```

- ¿Cuál es el algoritmo de encriptado usado en la IKE SA? ¿Y en la ESP SA?

AES_CBC_128. Se usa para cifrar la negociación inicial (IKEv2).

No hay cifrado a nivel ESP (esto es intencionado, porque el escenario se llama esp-alg-null).

- ¿Cuál es el algoritmo de integridad usado en la IKE SA? ¿Y en la ESP SA?

HMAC_SHA2_256_128. Se usa para garantizar la integridad y autenticación del intercambio IKE.

HMAC_SHA2_256_128. Se mantiene el mismo mecanismo de integridad para los datos transportados por IPsec.

- ¿Cuál es algoritmo usado para el Diffie-Hellman en IKE?

CURVE_25519. Es una curva elíptica moderna (grupo DH tipo X25519) utilizada para el intercambio seguro de claves.

- ¿Cuánto falta para se haga un rekeying de la IKE SA? ¿Y de la ESP SA?

De la IKE SA -> En Moon: 8711s. En Carol: 7492s.

De la ESP SA -> En Moon: 1027s. En Carol: 1029s

- ¿Cuáles son los SPI de la IKE SA? ¿Cuál corresponde a carol y cuál a moon?

IKE SA ID: 88d260b288427213_i 0312154be5e831f3_r

- _i: initiator. Corresponde a Carol (ella inició la conexión).

- _r: responder. Corresponde a Moon.

- ¿Cuáles son los SPI de la ESP SA? ¿Cuál corresponde a carol y cuál a moon?

En Moon:

- in cd9e4371 (este es el SPI de Moon)

- out ceeef96e2

En Carol:

- in ceeef96e2 (este es el SPI de Carol)

- out cd9e4371

- Mide la velocidad alice --> carol y viceversa con iperf tanto con TCP como con UDP.

```
carol:~# iperf -s
```

```
alice:~# iperf -c 192.168.0.100 -i 1
```

```
-----  
Client connecting to 192.168.0.100, TCP port 5001
```

```
TCP window size: 45.0 KByte (default)
```

```
[ 3] local 10.1.0.10 port 33566 connected with 192.168.0.100 port 5001
```

```
[ ID] Interval Transfer Bandwidth
```

```
[ 3] 0.0000-1.0000 sec 15.1 MBytes 127 Mbits/sec
```

```
[ 3] 1.0000-2.0000 sec 15.6 MBytes 131 Mbits/sec
```

```
[ 3] 2.0000-3.0000 sec 16.0 MBytes 134 Mbits/sec
```

```
alice:~# iperf -u -c 192.168.0.100 -b 100M -i 1
```

```
-----  
Client connecting to 192.168.0.100, UDP port 5001
```

```
UDP buffer size: 208 KByte (default)
```

```
[ 3] local 10.1.0.10 port 33297 connected with 192.168.0.100 port 5001
```

```
[ ID] Interval Transfer Bandwidth
```

```
[ 3] 0.0000-1.0000 sec 8.90 MBytes 74.7 Mbits/sec
```

```
[ 3] 1.0000-2.0000 sec 8.94 MBytes 75.0 Mbits/sec
```

```
[ 3] 2.0000-3.0000 sec 9.77 MBytes 82.0 Mbits/sec
```

```
[ 3] 3.0000-4.0000 sec 10.2 MBytes 85.7 Mbits/sec
```

```
[ 3] 4.0000-5.0000 sec 9.17 MBytes 76.9 Mbits/sec
```

```
alice:~# iperf -s
carol:~# iperf -c 10.1.0.10 -i 1
-----
Client connecting to 10.1.0.10, TCP port 5001
TCP window size: 45.0 KByte (default)
-----
[  3] local 192.168.0.100 port 49412 connected with 10.1.0.10 port 5001
[ ID] Interval      Transfer     Bandwidth
[  3] 0.0000-1.0000 sec   9.50 MBytes   79.7 Mbits/sec
[  3] 1.0000-2.0000 sec   6.88 MBytes   57.7 Mbits/sec
[  3] 2.0000-3.0000 sec  11.8 MBytes   98.6 Mbits/sec
carol:~# iperf -u -c 192.168.0.100 -b 100M -i 1
-----
Client connecting to 192.168.0.100, UDP port 5001
UDP buffer size: 208 KByte (default)
-----
[  3] local 192.168.0.100 port 53848 connected with 192.168.0.100 port 5001
[ ID] Interval      Transfer     Bandwidth
[  3] 0.0000-1.0000 sec   9.98 MBytes   83.7 Mbits/sec
[  3] 1.0000-2.0000 sec  10.2 MBytes   85.5 Mbits/sec
[  3] 2.0000-3.0000 sec  10.2 MBytes   86.0 Mbits/sec
```

- Para el servicio strongswan

```
carol:~# systemctl stop strongswan
```

```
moon:~# systemctl stop strongswan
```

IPsec: ESCENARIO HOST2HOST HOST2HOST ESP MODO TÚNEL

- Carga el escenario

```
root@omninetvm2024:/srv/strongswan-testing/testing# ./scripts/load-testconfig ikev2/host2host-cert
```

- Carga las reglas iptables que eliminan la conectividad entre máquinas excepto mediante IPsec.

```
moon:~# iptables-restore < /etc/iptables.rules
```

```
sun:~# iptables-restore < /etc/iptables.rules
```

- Comprueba que no es posible hacer un ping sun -> moon y moon -> sun. Analiza el resultado a la vista de las reglas iptables de moon y sun.

```
moon:~# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
^C
--- 192.168.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2046ms

moon:~# exit
logout
Connection to 192.168.0.1 closed.
root@omninetvm2024:/srv/strongswan-testing/testing# ssh 192.168.0.2
sun:~# ping -c 3 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
^C
--- 192.168.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2060ms
```

En ambos casos no hubo respuesta, confirmando que las reglas bloquean todo el tráfico directo (incluido ICMP), excepto el destinado a establecer túneles IPsec.

- Inicia strongswan en moon y sun

```
sun:~# systemctl start strongswan
```

```
moon:~# systemctl start strongswan
```

En el caso de moon:

- ¿Qué interfaz (dirección IP) de moon utiliza la conexión host-host?

local_addrs = 192.168.0.1

- ¿Qué versión de IKE soporta?

version = 2

- ¿Qué propuesta de protocolo de cifrado, integridad y grupo de intercambio de claves Diffie Hellman admite para la asociación de seguridad IKE?

proposals = aes128-sha256-x25519

- ¿Qué propuesta de protocolo de cifrado, integridad y grupo de intercambio de claves Diffie Hellman admite para la asociación de seguridad ESP hija de la asociación de seguridad IKE?

esp_proposals = aes128gcm128-x25519

Muestra las tablas de enrutamiento y las iptables de cada equipo. ¿Cambió algo?

```
moon:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.1.0.0/16 dev eth1 proto kernel scope link src 10.1.0.1
10.8.0.0/24 via 10.8.0.2 dev tun0
10.8.0.2 dev tun0 proto kernel scope link src 10.8.0.1
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.1
moon:~# iptables -L -v --line-numbers
Chain INPUT (policy DROP 168 packets, 7056 bytes)
num  pkts bytes target  prot opt in     out    source          destination
1      0     0 ACCEPT   esp  --  eth0   any   anywhere        anywhere
2      0     0 ACCEPT   ah   --  eth0   any   anywhere        anywhere
3      0     0 ACCEPT   udp  --  eth0   any   anywhere        anywhere
4      0     0 ACCEPT   udp  --  eth0   any   anywhere        anywhere
5  283 25162 ACCEPT   tcp  --  any    any   anywhere        anywhere
6      0     0 ACCEPT   tcp  --  eth0   any   winnetou.strongswan.org  anywhere
                                         udp spt:isakmp dpt:isakmp
                                         udp spt:ipsec-nat-t dpt:ipsec-nat-t
                                         tcp dpt:ssh
                                         tcp spt:http

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num  pkts bytes target  prot opt in     out    source          destination

Chain OUTPUT (policy DROP 13 packets, 882 bytes)
num  pkts bytes target  prot opt in     out    source          destination
1      0     0 ACCEPT   esp  --  any    eth0  anywhere        anywhere
2      0     0 ACCEPT   ah   --  any    eth0  anywhere        anywhere
3      0     0 ACCEPT   udp  --  any    eth0  anywhere        anywhere
4      0     0 ACCEPT   udp  --  any    eth0  anywhere        anywhere
5  165 19528 ACCEPT   tcp  --  any    any   anywhere        anywhere
6      0     0 ACCEPT   tcp  --  any    eth0  anywhere        anywhere
                                         udp spt:isakmp dpt:isakmp
                                         udp spt:ipsec-nat-t dpt:ipsec-nat-t
                                         tcp spt:ssh
                                         winnetou.strongswan.org  tcp dpt:http
moon:~# 
sun:~# ip route show
default via 192.168.0.254 dev eth0 onlink
10.2.0.0/16 dev eth1 proto kernel scope link src 10.2.0.1
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.2
sun:~# iptables -L -v --line-numbers
Chain INPUT (policy DROP 0 packets, 0 bytes)
num  pkts bytes target  prot opt in     out    source          destination
1      0     0 ACCEPT   esp  --  eth0   any   anywhere        anywhere
2      0     0 ACCEPT   ah   --  eth0   any   anywhere        anywhere
3      0     0 ACCEPT   udp  --  eth0   any   anywhere        anywhere
4      0     0 ACCEPT   udp  --  eth0   any   anywhere        anywhere
5  288 25558 ACCEPT   tcp  --  any    any   anywhere        anywhere
6      0     0 ACCEPT   tcp  --  eth0   any   winnetou.strongswan.org  anywhere
                                         udp spt:isakmp dpt:isakmp
                                         udp spt:ipsec-nat-t dpt:ipsec-nat-t
                                         tcp dpt:ssh
                                         tcp spt:http

Chain FORWARD (policy DROP 0 packets, 0 bytes)
num  pkts bytes target  prot opt in     out    source          destination

Chain OUTPUT (policy DROP 3 packets, 252 bytes)
num  pkts bytes target  prot opt in     out    source          destination
1      0     0 ACCEPT   esp  --  any    eth0  anywhere        anywhere
2      0     0 ACCEPT   ah   --  any    eth0  anywhere        anywhere
3      0     0 ACCEPT   udp  --  any    eth0  anywhere        anywhere
4      0     0 ACCEPT   udp  --  any    eth0  anywhere        anywhere
5  163 18916 ACCEPT   tcp  --  any    any   anywhere        anywhere
6      0     0 ACCEPT   tcp  --  any    eth0  anywhere        winnetou.strongswan.org  tcp dpt:http
```

Las cadenas quedaron con *policy DROP*, permitiendo solo **ESP/AH** y **UDP 500/4500** (IKE/NAT-T), además de reglas de gestión (SSH/HTTP con winnetou).

Carga las conexiones

```
root@omninetvm2024:/srv/strongswan-testing/testing# ./stop-testing
Stopping test environment
[ ok ] Network vnet1
[ ok ] Network vnet2
[ ok ] Network vnet3
[ ok ] Guest alice
[ ok ] Guest bob
[ ok ] Guest carol
[ ok ] Guest dave
[ ok ] Guest moon
[ ok ] Guest sun
[ ok ] Guest venus
[ ok ] Guest winnetou
[ ok ] Removing kernel linux-5.19.11
[ ok ] Removing link to hostfs
[ ok ] Removing link to testresults
root@omninetvm2024:/srv/strongswan-testing/testing# ./scripts/load-testconfig ikev2/host2host-cert
^C
root@omninetvm2024:/srv/strongswan-testing/testing# ./scripts/load-testconfig ikev2/host2host-cert
root@omninetvm2024:/srv/strongswan-testing/testing# ./start-testing
Starting test environment
[ ok ] Deploying kernel linux-5.19.11
[ ok ] Deploying /srv/strongswan-testing/build/shared/bullseye as hostfs
```

```
moon:~# expect-connection host-host
moon:~# exit
sun:~# expect-connection host-host
sun:~# █
```

Levantahora la conexión host-host en moon:

```
moon:~# swanctl --initiate --child host-host
[IKE] initiating IKE_SA host-host[1] to 192.168.0.2
[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) N(FRAG_SUP) N(HASH_ALG) N(REQ_NATD) ]
[NET] sending packet: from 192.168.0.1[500] to 192.168.0.2[500] (240 bytes)
[NET] received packet: from 192.168.0.2[500] to 192.168.0.1[500] (293 bytes)
[ENC] parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP) N(NATD_D_IP) CERTREQ N(FRAG_SUP) N(HASH_ALG) N(REQ_NATD) ]
```

Comprueba que ahora sí es posible la conectividad moon <--> sun.

Debería de ir WTF

Muestra las reglas de enrutado mediante ip rule y observa que se ha añadido la tabla de enrutamiento que utiliza strongSwan con identificador 220. Muestra dicha tabla mediante ip route show table 220. ¿Qué cambio hizo posible la conectividad moon <--> sun?

```
moon:~# ip rule
0:      from all lookup local
220:    from all lookup 220
32766:  from all lookup main
32767:  from all lookup default
moon:~# ip route show table 220
192.168.0.2 via 192.168.0.2 dev eth0 proto static src 192.168.0.1
```

```
sun:~# ip rule
0:      from all lookup local
220:    from all lookup 220
32766:  from all lookup main
32767:  from all lookup default
sun:~# ip route show table 220
192.168.0.1 via 192.168.0.1 dev eth0 proto static src 192.168.0.2
```

Tras ejecutar ip rule en **moon** y **sun**, se observa que se ha añadido una nueva regla que hace referencia a la tabla de enrutamiento **220**, utilizada por strongSwan para el tráfico protegido por IPsec.

Esta tabla 220 es creada automáticamente por **strongSwan** al levantar la conexión IPsec y contiene las rutas necesarias para que el tráfico entre **moon** y **sun** se encamine correctamente a través del túnel seguro.

El cambio que hizo posible la conectividad es el establecimiento de la CHILD SA, (swanctl initiate –child host-host), momento en el que Strongswan:

1. Añadió la regla de policy routing (lookup 220),
2. Insertó las rutas IPsec en la tabla 220,

3. Instaló las políticas y estados XFRM,
4. Y ejecutó el script _updown iptables, que permite el tráfico ESP en el cortafuegos.

Muestra las conexiones IPsec en moon y sun con el comando swanctl --list-sas

```
moon:~# swanctl --list-sas
host-host: #1, ESTABLISHED, IKEv2, 26d8089ad77bf4c2_i* 68eef4850a521076_r*
  local 'moon.strongswan.org' @ 192.168.0.1[500]
  remote 'sun.strongswan.org' @ 192.168.0.2[500]
  AES_CBC-128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
  established 913s ago, reauth in 9145s
host-host: #1, reqid 1, INSTALLED, TUNNEL, ESP:AES_GCM_16-128
  installed 914s ago, rekeying in 4194s, expires in 5027s
    in c671027e, 9156 bytes, 109 packets
    out cf54ddae, 0 bytes, 0 packets
    local 192.168.0.1/32
    remote 192.168.0.2/32
No leaks detected, 51 suppressed by whitelist
moon:~#
```

```
sun:~# swanctl --list-sas
host-host: #1, ESTABLISHED, IKEv2, 26d8089ad77bf4c2_i 68eef4850a521076_r*
  local 'sun.strongswan.org' @ 192.168.0.2[500]
  remote 'moon.strongswan.org' @ 192.168.0.1[500]
  AES_CBC-128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
  established 922s ago, reauth in 9795s
host-host: #1, reqid 1, INSTALLED, TUNNEL, ESP:AES_GCM_16-128
  installed 922s ago, rekeying in 4108s, expires in 5018s
    in cf54ddae, 0 bytes, 0 packets
    out c671027e, 9156 bytes, 109 packets, 743s ago
    local 192.168.0.2/32
    remote 192.168.0.1/32
No leaks detected, 51 suppressed by whitelist
sun:~#
```

- ¿Cuál es el algoritmo de encriptado usado en la IKE SA? ¿Y en la ESP SA?

En la IKE SA es AES_CBC-128

En la ESP SA es AES_GCM_16-128

- ¿Cuál es el algoritmo de integridad usado en la IKE SA? ¿Y en la ESP SA?

En la IKE SA es HMAC_SHA2_256_128

En la ESP SA no hay uno separado; la integridad está incluida en **AES_GCM**.

- ¿Cuál es algoritmo usado para el Diffie-Hellman en IKE?

CURVE_25519

- ¿Cuánto falta para se haga un rekeying de la IKE SA? ¿Y de la ESP SA?

- En IKE SA:

- * En moon: reauth (renovación) en ~9145 segundos
- * En sun: reauth en ~9795 segundos.

- En ESP SA:

- * En moon: rekeying in 4194s, expires in 5027s
- * En sun: rekeying in 4108s, expires in 5018s

- ¿Cuáles son los SPI de la IKE SA? ¿Cuál corresponde a moon y cuál a sun?

De IKE SA:

- □ 26d8089ad77bf4c2_i → corresponde a **moon (iniciador)**
- 68eef4850a521076_r → corresponde a **sun (respondedor)**

- ¿Cuáles son los SPI de la ESP SA? ¿Cuál corresponde a moon y cuál a sun?

De ESP SA:

En moon:

- in c671027e
- out cf54ddae

En sun:

- in cf54ddae
- out c671027e

host2host AH modo transporte

- Modifica el contenido de /etc/swanctl/swanctl.conf tanto en moon como en sun para que la SA IPsec sea mediante AH en modo transporte, simplemente cambiando en la conexión host-host la directiva esp_proposals por ah_proposals = aesxcbc y añadiendo justo a continuación la directiva mode = transport.

- Reinicia IPsec en ambos equipos y levanta la conexión host-host.

- Muestra las conexiones IPsec en moon. Comprueba que, efectivamente, la SA IPsec host-host utiliza AH en modo transporte.

En ambos equipos se editó el fichero /etc/swanctl/swanctl.conf dentro del bloque connections { host-host { ... } }, sustituyendo “esp_proposals = aes128gcm128-x25519” por:

- ah_proposals = aesxcbc
- mode = transport

A continuación se reinició el servicio y se levantó la conexión:

- systemctl restart strongswan
- expect-connection host-host
- swanctl --initiate --child host-host

Comprobación:

```
moon:~# swanctl --list-sas
host-host: #1, ESTABLISHED, IKEv2, d47ec8e38c15d2f7_i* aa56cb876d37f63b_r
  local 'moon.strongswan.org' @ 192.168.0.1[500]
  remote 'sun.strongswan.org' @ 192.168.0.2[500]
  AES_CBC-128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
  established 10s ago, reauth in 10310s
host-host: #1, reqid 1, INSTALLED, TRANSPORT, AH:AES_XCBC_96
  installed 10s ago, rekeying in 4894s, expires in 5930s
  in c02d787c,      0 bytes,      0 packets
  out c88a1755,     0 bytes,      0 packets
  local 192.168.0.1/32
  remote 192.168.0.2/32
No leaks detected, 51 suppressed by whitelist
```

Interpretación:

- AH (Authentication Header) proporciona **autenticación e integridad**, pero **no cifrado**.
- Protege los encabezados IP y la carga útil para detectar alteraciones o suplantaciones.
- En **modo transporte**, solo se protege la parte útil del paquete IP, manteniendo el encabezado original.
- El cifrado ESP, usado anteriormente, ofrecía confidencialidad; al pasar a AH, se pierde el cifrado, pero se gana menor sobrecarga y verificación de integridad completa.

host2host ESP modo transporte

Realiza los cambios necesarios para llevarlo a cabo.

En /etc/swanctl/swanctl.conf de *moon* y *sun* se modificó el bloque host-host, sustituyendo “ah_proposals = aesxcbc” por “esp_proposals = aes128gcm128-x25519”.

Comandos ejecutados:

- systemctl restart strongswan (en las dos máquinas)
- expect-connection host-host (en las dos máquinas)
- swanctl -initiate -child host-host (en moon)
- swanctl -list-sas (en moon)

```
host-host: #1, ESTABLISHED, IKEv2, 28ddf82df33c71cf_i* 5745d20b24741582_r
local 'moon.strongswan.org' @ 192.168.0.1[500]
remote 'sun.strongswan.org' @ 192.168.0.2[500]
AES_CBC-128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
established 8s ago, reauth in 10684s
host-host: #1, reqid 1, INSTALLED, TRANSPORT, ESP:AES_GCM_16-128
installed 8s ago, rekeying in 5003s, expires in 5932s
in cc4347de,      0 bytes,      0 packets
out cffd4ffc,      0 bytes,      0 packets
local 192.168.0.1/32
remote 192.168.0.2/32
No leaks detected, 51 suppressed by whitelist
```

host2host AH modo túnel

No existe este test en strongSwan pero realiza los cambios necesarios para llevarlo a cabo:

Editamos los archivos de los dos equipos por:

ah_proposals = aesxcbc

y

mode = tunnel

Comandos ejecutados:

- systemctl restart strongswan
- expect-connection host-host
- swanctl --initiate --child host-host (solo en moon)
- swanctl --list-sas (solo en moon)

```
moon:~# swanctl --list-sas
host-host: #1, ESTABLISHED, IKEv2, 59562594a0370126_i* 4d3f150b1677471f_r
local 'moon.strongswan.org' @ 192.168.0.1[500]
remote 'sun.strongswan.org' @ 192.168.0.2[500]
AES_CBC-128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
established 4s ago, reauth in 10249s
host-host: #1, reqid 1, INSTALLED, TUNNEL, AH:AES_XCBC_96
installed 5s ago, rekeying in 4939s, expires in 5936s
in c90021be,      0 bytes,      0 packets
out cd6ae829,      0 bytes,      0 packets
local 192.168.0.1/32
remote 192.168.0.2/32
No leaks detected, 51 suppressed by whitelist
```

IPsec: escenario net2net

Net2net modo transporte

Recuerda que el **modo transporte no tiene sentido en un escenario net2net**.

No se realizaron cambios en los ficheros de configuración (/etc/swanctl/swanctl.conf), ya que el modo transporte **no tiene sentido en un entorno net2net**.

Explicación técnica:

- En el modo **transporte**, solo se cifra la carga útil del paquete IP, dejando intacta la cabecera IP original.
- Este modo se usa en comunicaciones **host ↔ host**, donde los propios extremos son los participantes de la comunicación IPsec.
- En cambio, en un escenario **net2net**, los extremos del túnel son **gateways** que actúan en nombre de redes completas (por ejemplo, 10.1.0.0/24 ↔ 10.2.0.0/24).
- Los paquetes que atraviesan el túnel tienen como direcciones origen/destino las IP internas de las subredes, que **no son enruteables** fuera de sus LAN respectivas.
- Por tanto, si se aplicara el modo transporte, el tráfico saldría con esas IP internas visibles, y **los routers intermedios no sabrían cómo entregarlo**.
- El único modo válido en este tipo de escenarios es el **modo túnel**, que encapsula el paquete IP completo dentro de otro con cabecera IP pública de los gateways.

De cara al examen:

- Transporte: protege solo los datos del paquete IP → solo útil en host↔host.
- Túnel: Encapsula todo el paquete → necesario en net↔net o gateway↔gateway.
- En net2net, el transporte rompería la comunicación porque las direcciones IP internas no serían enruteables.

Conclusión: El modo transporte **no tiene sentido en escenarios net2net**, ya que los gateways deben encapsular completamente los paquetes de las redes internas usando el **modo túnel** para asegurar la correcta entrega y confidencialidad.

net2net ESP modo túnel

- Carga el escenario

```
scripts/load-testconfig ikev2/net2net-cert
```

- Carga las reglas iptables que eliminan la conectividad entre máquinas excepto mediante IPsec:

```
moon$ iptables-restore < /etc/iptables.rules
```

```
sun$ iptables-restore < /etc/iptables.rules
```

- Arranca IPsec en moon y sun y levante la conexión net-net desde uno de ellos.

```
expect-connection net-net
```

```
swanctl --initiate --child net-net
```

- Comprueba la conectividad entre las redes 10.1.0.0/16 y 10.2.0.0/16.

A mi no me va pero supuestamente tendría que ir

- Comprueba las características de la SA IPsec net-net consultando su estado.

```
moon:~# swanctl --list-sas
gw-gw: #1, ESTABLISHED, IKEv2, d9ac6f59b949043e_i* 1eda089211d830ff_r
  local 'moon.strongswan.org' @ 192.168.0.1[500]
  remote 'sun.strongswan.org' @ 192.168.0.2[500]
  AES_CBC-128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
  established 85s ago, reauth in 10693s
net-net: #1, reqid 1, INSTALLED, TUNNEL, ESP:AES_GCM_16-128
  installed 85s ago, rekeying in 5112s, expires in 5855s
  in cdd3d712,      0 bytes,      0 packets
  out c57aa9e2,      0 bytes,      0 packets
  local 10.1.0.0/16
  remote 10.2.0.0/16
No leaks detected, 51 suppressed by whitelist
```

net2net AH modo túnel

Realiza los cambios necesarios para llevarlo a cabo.

Modificamos el fichero /etc/swanctl/swanctl.conf con esto:

- ah_proposals = aesxcbc
- mode = tunnel

Comandos ejecutados:

```
- systemctl restart strongswan
- iptables-restore < /etc/iptables.rules
- expect-connection net-net
- swanctl --initiate --child net-net
- swanctl --list-sas
```

```

moon:~# swanctl --list-sas
gw-gw: #1, ESTABLISHED, IKEv2, 06d65d2fd5bf8716_i* 9a2dda11931ceba9_r
  local 'moon.strongswan.org' @ 192.168.0.1[500]
  remote 'sun.strongswan.org' @ 192.168.0.2[500]
AES_CBC-128/HMAC_SHA2_256_128/PRF_HMAC_SHA2_256/CURVE_25519
established 9s ago, reauth in 9930s
net-net: #1, reqid 1, INSTALLED, TUNNEL, AH:AES_XCBC_96
  installed 9s ago, rekeying in 4909s, expires in 5931s
  in c628f313,      0 bytes,      0 packets
  out cddf3eb1,      0 bytes,      0 packets
  local 10.1.0.0/16
  remote 10.2.0.0/16
No leaks detected, 51 suppressed by whitelist

```

PARTE 3 DEL TALLER

4.1 OpenVPN: Instalación, creación de la PKI y configuración del servidor y del cliente

Es imprescindible consultar el [HOWTO de OpenVPN](#) para entender los comandos y parámetros utilizados.

OpenVPN e IPsec no interactúan bien entre sí. Asegúrate de **parar el servicio IPsec si vas a utilizar el servicio OpenVPN y viceversa**.

4.1.1 Instalación de paquetes

Instala en *moon*, *alice* y *carol* el paquete *openvpn* y sus dependencias. Habitualmente, haríamos `apt-get install openvpn`, pero como las máquinas no tienen conectividad a Internet, debemos descargarlos los siguientes paquetes de la versión *oldstable* (*bullseye*) de <https://packages.debian.org>: *openvpn easy-rsa liblzo2-2 libpkcs11-helper1*. Una vez descargados, hay que copiarlos a *moon*, *alice* y *carol* mediante:

```

omnet6-vm# scp -O *.deb root@10.1.0.1:
omnet6-vm# scp -O *.deb root@10.1.0.10:
omnet6-vm# scp -O *.deb root@192.168.0.100:

```

En *moon*, *alice* y *carol* ejecuta:

```
host$ dpkg -i *.deb
```

En *moon*, *alice* y *carol*: `systemctl stop strongswan`

Descargamos los archivos que nos piden de la web.

```

root@omneturn2024:/home/user/Downloads# scp -O *.deb root@192.168.0.1:
easy-rsa_3.0.8-1_all.deb                                         100%   44KB   3.3MB/s  00:00
liblzo2-2.2.10-2_amd64.deb                                       100%   56KB   6.8MB/s  00:00
libpkcs11-helper1_1.27-1_amd64.deb                                 100%   46KB   8.0MB/s  00:00
openvpn_2.5.1-3+deb11u2_amd64.deb                                100%  586KB  36.3MB/s  00:00
root@omneturn2024:/home/user/Downloads# scp -O *.deb root@192.168.0.100:
easy-rsa_3.0.8-1_all.deb                                         100%   44KB   4.0MB/s  00:00
liblzo2-2.2.10-2_amd64.deb                                       100%   56KB   6.9MB/s  00:00
libpkcs11-helper1_1.27-1_amd64.deb                                 100%   46KB   8.9MB/s  00:00
openvpn_2.5.1-3+deb11u2_amd64.deb                                100%  586KB  17.0MB/s  00:00
root@omneturn2024:/home/user/Downloads# scp -O *.deb root@10.1.0.10:
easy-rsa_3.0.8-1_all.deb                                         100%   44KB   1.7MB/s  00:00
liblzo2-2.2.10-2_amd64.deb                                       100%   56KB   2.5MB/s  00:00
libpkcs11-helper1_1.27-1_amd64.deb                                 100%   46KB   5.4MB/s  00:00
openvpn_2.5.1-3+deb11u2_amd64.deb                                100%  586KB  18.3MB/s  00:00

```

Nos vamos a la raíz de cada máquina y hacemos `dpkg -i *.deb`

Comprobamos que se instaló bien en todas las máquinas:

```

alice:~# openvpn --version
OpenVPN 2.5.1 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Aug 25 2025
library versions: OpenSSL 1.1.1n 15 Mar 2022, LZO 2.10
Originally developed by James Yonan
Copyright (C) 2002-2018 OpenVPN Inc <sales@openvpn.net>
Compile time defines: enable_async_push=no enable_comp_stub=no enable_crypto_ofb_cfb=yes enable_debug=yes enable_def_auth=yes enable_dh_group14 enable_dh_group16 enable_dh_group18 enable_dh_group23 enable_dh_group29 enable_dh_group51 enable_dh_group75 enable_dh_group1024 enable_dh_group2048 enable_dh_group4096 enable_dh_group8192 enable_dh_group16384 enable_dh_group32768 enable_dh_group65536 enable_dh_group131072 enable_dh_group262144 enable_dh_group524288 enable_dh_group1048576 enable_dh_group2097152 enable_dh_group4194304 enable_dh_group8388608 enable_dh_group16777216 enable_dh_group33554432 enable_dh_group67108864 enable_dh_group134217728 enable_dh_group268435456 enable_dh_group536870912 enable_dh_group1073741824 enable_dh_group2147483648 enable_dh_group4294967296 enable_dh_group8589934592 enable_dh_group17179869184 enable_dh_group34359738368 enable_dh_group68719476736 enable_dh_group137438953472 enable_dh_group274877906944 enable_dh_group549755813888 enable_dh_group1099511627776 enable_dh_group2199023255552 enable_dh_group4398046511104 enable_dh_group8796093022208 enable_dh_group17592186044416 enable_dh_group35184372088832 enable_dh_group70368744177664 enable_dh_group140737488355328 enable_dh_group281474976710656 enable_dh_group562949953421312 enable_dh_group112589990684264 enable_dh_group225179981368528 enable_dh_group450359962737056 enable_dh_group900719925474112 enable_dh_group180143985094824 enable_dh_group360287970189648 enable_dh_group720575940379296 enable_dh_group1441151880758592 enable_dh_group2882303761517184 enable_dh_group5764607523034368 enable_dh_group11529215046068736 enable_dh_group23058430092137472 enable_dh_group46116860184274944 enable_dh_group92233720368549888 enable_dh_group18446744073709976 enable_dh_group36893488147419952 enable_dh_group73786976294839904 enable_dh_group147573952589679808 enable_dh_group295147905179359616 enable_dh_group590295810358719232 enable_dh_group1180591620717438464 enable_dh_group2361183241434876928 enable_dh_group4722366482869753856 enable_dh_group9444732965739507712 enable_dh_group1888946593147901544 enable_dh_group3777893186295803088 enable_dh_group7555786372591606176 enable_dh_group1511157274518321232 enable_dh_group3022314549036642464 enable_dh_group6044629098073284928 enable_dh_group12089258196146569856 enable_dh_group24178516392293139712 enable_dh_group48357032784586279424 enable_dh_group96714065569172558848 enable_dh_group193428131138345117696 enable_dh_group386856262276690235392 enable_dh_group773712524553380470784 enable_dh_group1547425049106760941568 enable_dh_group3094850098213521883136 enable_dh_group6189700196427043766272 enable_dh_group12379400392854087532544 enable_dh_group24758800785708175065088 enable_dh_group49517601571416350130176 enable_dh_group99035203142832670260352 enable_dh_group198070406285665340520704 enable_dh_group396140812571330681041408 enable_dh_group792281625142661362082816 enable_dh_group1580563250285322724165632 enable_dh_group3161126500570645448321264 enable_dh_group6322253001141290896642528 enable_dh_group1264450600228258179328512 enable_dh_group2528901200456516358656024 enable_dh_group5057802400913032717312048 enable_dh_group10115604801826065434624096 enable_dh_group20231209603652130869248192 enable_dh_group40462419207304261738496384 enable_dh_group80924838414608523476992768 enable_dh_group16184967682921704695398536 enable_dh_group32369935365843409390797072 enable_dh_group64739870731686818781594144 enable_dh_group12947974146337363756318828 enable_dh_group25895948292674727512637656 enable_dh_group51791896585349455025275312 enable_dh_group103583793170698910050550624 enable_dh_group207167586341397820101101248 enable_dh_group414335172682795640202202496 enable_dh_group828670345365591280404404992 enable_dh_group165734069073118256080880996 enable_dh_group331468138146236512161761992 enable_dh_group662936276292473024323523992 enable_dh_group132587255258494604864724992 enable_dh_group26517451051698920972944992 enable_dh_group53034902103397841945889984 enable_dh_group106069804206795683891779968 enable_dh_group212139608413591367783559936 enable_dh_group424279216827182735567119872 enable_dh_group848558433654365471134239744 enable_dh_group169711686730873094226859588 enable_dh_group339423373461746188453719176 enable_dh_group678846746923492376907438352 enable_dh_group137769349384698475381487672 enable_dh_group275538698769396950762975344 enable_dh_group551077397538793901525950688 enable_dh_group114215479507798380305900376 enable_dh_group228430959015596760611800752 enable_dh_group456861918031193521222600152 enable_dh_group913723836062387042445200304 enable_dh_group1827447672124774084890400608 enable_dh_group3654895344249548169780801216 enable_dh_group7309790688498596339561602432 enable_dh_group1461958137699719267912320480 enable_dh_group2923916275399438535824640960 enable_dh_group5847832550798877071649281920 enable_dh_group11695665101597754143298563840 enable_dh_group23391330203195508286597127680 enable_dh_group46782660406387016573194255360 enable_dh_group93565320812774033146388510720 enable_dh_group187130641625548066292777021440 enable_dh_group374261283251096132585554042880 enable_dh_group748522566502192265171108085760 enable_dh_group1597045133004384530342216171520 enable_dh_group3194090266008769060684432343040 enable_dh_group6388180532017538121368864686080 enable_dh_group1277636106403507624273772936000 enable_dh_group2555272212807015248547545872000 enable_dh_group5110544425614030497095091744000 enable_dh_group10221088851228060994190935488000 enable_dh_group20442177702456121988381870976000 enable_dh_group40884355404912243976763741952000 enable_dh_group81768710809824487953527483904000 enable_dh_group163537421619648955907054967808000 enable_dh_group327074843239297911814109935616000 enable_dh_group654149686478595823628219871232000 enable_dh_group1308299372957191647256439742464000 enable_dh_group2616598745914383294512879484928000 enable_dh_group5233197491828766589025758969856000 enable_dh_group10466394983657533178051517937728000 enable_dh_group20932789967315066356103035875456000 enable_dh_group41865579934630132712206071750912000 enable_dh_group83731159869260265424412143501824000 enable_dh_group167462319738520530848824287003648000 enable_dh_group334924639477041061697648574007296000 enable_dh_group669849278954082123395297148014592000 enable_dh_group1339698557908164246785594296029184000 enable_dh_group2679397115816328493571188592058368000 enable_dh_group5358794231632656987142377184018736000 enable_dh_group1071758846326513395484675436037472000 enable_dh_group2143517692653026790969350872074848000 enable_dh_group4287035385306053581938701744148896000 enable_dh_group857407077061210716387740348829776000 enable_dh_group1714814154122421432755407097659532000 enable_dh_group3429628308244842865510814195318964000 enable_dh_group6859256616489685731021628390637928000 enable_dh_group13818513232979371462043256781278568000 enable_dh_group27637026465958742924086513562557136000 enable_dh_group55274052931917485848173027125114272000 enable_dh_group110548105863835971696346054250228544000 enable_dh_group221096211727671943392692108500557088000 enable_dh_group442192423455343886785384217001114176000 enable_dh_group884384846910687773570768434002228352000 enable_dh_group1768769693821375547141536868004456704000 enable_dh_group3537539387642751094283073736008913408000 enable_dh_group7075078775285502188566147472017826816000 enable_dh_group14150157550571004377132294944035653632000 enable_dh_group28300315101142008754264589888071307264000 enable_dh_group56600630202284017508529179776142614528000 enable_dh_group113201060404560350017058359552285285568000 enable_dh_group226402120809120700034116711910570571136000 enable_dh_group45280424161824140068233423821114114272000 enable_dh_group90560848323648280136466847642228228544000 enable_dh_group181121696647285602729337795284456457088000 enable_dh_group362243393294571205458675590568892914176000 enable_dh_group72448678658914240131735118113785828352000 enable_dh_group144897357317824802663470236227571774016000 enable_dh_group289794714635649605326940472455143554032000 enable_dh_group57958942927129920165388094491030710864000 enable_dh_group11591798585455984033077618898206141728000 enable_dh_group23183597170911968066155237796412283456000 enable_dh_group46367194341823936132310475592824566912000 enable_dh_group92734388683647872264620851185649133824000 enable_dh_group185468777367295544529217022371298267648000 enable_dh_group370937554734591089058434044742596535296000 enable_dh_group74187510946918217811686808948593067584000 enable_dh_group148375021893836435623373617897196135168000 enable_dh_group296750043787672871246747235794392270336000 enable_dh_group593500087575345742493494471588784540672000 enable_dh_group116700017515673588898698894377576908136000 enable_dh_group233400035031347177797397788755153816272000 enable_dh_group466800070062694355594795577510307632544000 enable_dh_group93360014012538871118959115502061526088000 enable_dh_group186720028025777542237982231004123051376000 enable_dh_group373440056051555084475964462008246027552000 enable_dh_group74688011210311016895192892401649055104000 enable_dh_group149376022420622033790385784803298110208000 enable_dh_group298752044841244067580771569606596220416000 enable_dh_group597504089682488135161543139213192440832000 enable_dh_group1195008179364976270323086278426384881664000 enable_dh_group2390016358729952540646172556852769773328000 enable_dh_group4780032717459855081292345113705539556656000 enable_dh_group9560065434919710162584690227411079113312000 enable_dh_group1912013086929420232516980454822058222624000 enable_dh_group3824026173858840465033960909644116445248000 enable_dh_group764805234771768093006792181928823290496000 enable_dh_group1529610469543576186013584363857646889932000 enable_dh_group3059220939087152372027168727715323778864000 enable_dh_group6118441878174304744054337455435675557728000 enable_dh_group1243688375634860148810867491470751155456000 enable_dh_group2487376751269720297621734982940530230912000 enable_dh_group4974753502539440595243469965881060461824000 enable_dh_group9949507005078881190486939931762108243648000 enable_dh_group1989901401015776238097387986352201646896000 enable_dh_group3979802802031552476194775972704402413792000 enable_dh_group7959605604063104952385551945408804835888000 enable_dh_group1591921120812208990477151985808808167776000 enable_dh_group3183842241624417980954303971617616335552000 enable_dh_group636768448324883596190660794323523267104000 enable_dh_group1273536896649767983913301588647266413408000 enable_dh_group254707379329953596782660317729453234016000 enable_dh_group509414758659907193565320635458852668024000 enable_dh_group101882951731981438712640127857705332048000 enable_dh_group203765903463962877425280255715410664096000 enable_dh_group407531806927925754850560511430821333608000 enable_dh_group815063613855851509701120222861642667216000 enable_dh_group1630127227711703019402240445723284534432000 enable_dh_group3260254455423406038804480891446569068864000 enable_dh_group652050891084681207760896178289313813768000 enable_dh_group1304054454483402415521481356578634147336000 enable_dh_group2608108908966804830642962713157268294672000 enable_dh_group5216217817933609661285925426314536589344000 enable_dh_group1043243563586721932257945085269073177688000 enable_dh_group2086487127173443864515890170538146075376000 enable_dh_group4172974254346887729031780341076281450752000 enable_dh_group8345948508693775458063560682152563001504000 enable_dh_group1669189701738755091612712164235126060308000 enable_dh_group3338379403477510183225424328467252120616000 enable_dh_group6676758806955020366450848656934494201232000 enable_dh_group1335351761391004073291681633393898882464000 enable_dh_group2670703522782008146583363266787781769328000 enable_dh_group5341407045564016293166726533575781538656000 enable_dh_group10682804011120031963334413067555630773136000 enable_dh_group2136560802224006392666882613511132154672000 enable_dh_group4273121604448012785333765226702264209344000 enable_dh_group8546243208960025571335530453404528418688000 enable_dh_group1709248641792005111466706090680516889376000 enable_dh_group3418497283584010222933412181361032177552000 enable_dh_group683699456716802044586682436272024155104000 enable_dh_group1387398913433604089133844864540406430308000 enable_dh_group277479782686720817826768972908081260616000 enable_dh_group554959565373441635653577945816162410932000 enable_dh_group1175919130748832713307155890432324211864000 enable_dh_group2351838261497665426614311780864648423728000 enable_dh_group47
```

4.1.2 Creación de la PKI

La VPN que vamos a crear con OpenVPN hará uso de una PKI (*Public Key Infrastructure*). En resumen, cada equipo tendrá:

1. Un certificado (*host.crt*) que contiene su clave pública firmado por una CA (*Certificate Authority*) común. El certificado residirá en el equipo, pero será enviado a los otros equipos de la VPN en el establecimiento de la conexión. Por tanto, el certificado del equipo es público.
2. Una clave privada (*host.key*) que reside en el equipo y es secreta.
3. El certificado (*ca.crt*) con la clave pública de la CA para poder verificar que un certificado está firmado por la CA.

Solo en el equipo que actúa como CA residirá la clave privada de la CA (*ca.key*) con la que firmar los certificados de los equipos. Asumiremos que el equipo que actúa como servidor VPN (*moon*) actuará también como CA.

La PKI la crearemos utilizando la utilidad EasyRSA.

- Haz una copia del directorio de scripts de *easy-rsa* para evitar que en las actualizaciones del paquete *easy-rsa* se pierdan los cambios hechos para la creación de nuestra PKI:

```
moon$ cp -a /usr/share/easy-rsa /etc/openvpn/
```

- Sitúate en el directorio */etc/openvpn/easy-rsa*

```
moon$ cd /etc/openvpn/easy-rsa
```

- Edita el fichero *vars* rellenando los siguientes campos con valores adecuados:

Pág. 12



UNIVERSIDADE DA CORUÑA

Diseño de Redes

Taller de VPNs: IPsec y OpenVPN

```
set_var EASYRSA_REQ_COUNTRY      "ES"
set_var EASYRSA_REQ_PROVINCE     "LUG"
set_var EASYRSA_REQ_CITY          "A Coruña"
set_var EASYRSA_REQ_ORG           "UDC"
set_var EASYRSA_REQ_EMAIL        "login@udc.es"
set_var EASYRSA_REQ_OU            "DR"
```

- Limpia y compila:

```
moon$ ./easyrsa init-pki
moon$ ./easyrsa build-ca nopass
```

El último comando generará la clave privada (*ca.key*) y el certificado (*ca.crt*) que contiene la clave pública de la CA. Como *passphrase* usa siempre "*passphrase*" por simplicidad. En las preguntas, es imprescindible que introduzcas un valor para el CN (*Common Name*) del certificado. Como se trata del certificado de la CA, puedes fijarlo, p. ej., al valor *DR-CA*.

- Genera la clave privada (*moon.key*) y el certificado (*moon.crt*) que contiene la clave pública del servidor de nuestra VPN (*moon*):

```
moon$ ./easyrsa build-server-full moon nopass
```

En las preguntas, comprueba que el CN queda fijado al nombre del servidor (*moon*).

- Genera la clave privada (*.key*) y el certificado (*.crt*) de nuestros clientes *alice* y *carol*:

```
moon$ ./easyrsa build-client-full alice nopass
moon$ ./easyrsa build-client-full carol nopass
```

- En las preguntas, comprueba que el CN queda fijado al nombre de cada cliente (*alice* y *carol*, resp.).
- Genera el fichero (*dh.pem*) con los parámetros Diffie-Hellman a utilizar en nuestra VPN:

```
moon$ ./easyrsa gen-dh
```

- Copia los ficheros necesarios al directorio */etc/openvpn* del servidor VPN (*moon*):

```
moon$ cd /etc/openvpn/easy-rsa/pk1
moon$ cp ca.crt private/moon.key issued/moon.crt dh.pem /etc/openvpn
• Copia los ficheros necesarios al directorio /etc/openvpn de cada cliente VPN:
```

```
moon$ cd /etc/openvpn/easy-rsa/pk1
moon$ scp ca.crt private/alice.key issued/alice.crt alice:/etc/openvpn
moon$ scp ca.crt private/carol.key issued/carol.crt carol:/etc/openvpn
```

Literalmente esto:

1 Copiar el directorio de EasyRSA

Esto se hace para tener una copia editable y que no se pierda al actualizar paquetes.

En moon:

```
bash                                     ⚒ Copiar código  
cp -a /usr/share/easy-rsa /etc/openvpn/
```

Comprueba que se copió bien:

```
bash                                     ⚒ Copiar código  
ls /etc/openvpn/
```

Deberías ver algo como:

```
easy-rsa ...                                     ⚒ Copiar código
```

2 Entrar en el directorio

```
bash                                     ⚒ Copiar código  
cd /etc/openvpn/easy-rsa
```

Comprueba que estás dentro:

```
bash                                     ⚒ Copiar código  
pwd
```

Debería mostrar:

```
bash                                     ⚒ Copiar código  
/etc/openvpn/easy-rsa
```

3 Editar el fichero vars

Abre el fichero:

```
bash                                     ⚒ Copiar código  
nano vars
```

Y busca las líneas que empiezan por `set_var EASYRSA_REQ_...`.

Rellénalas así:

```
bash                                     ⚒ Copiar código  
set_var EASYRSA_REQ_COUNTRY      "ES"  
set_var EASYRSA_REQ_PROVINCE     "LCG"  
set_var EASYRSA_REQ_CITY         "A Coruna"  
set_var EASYRSA_REQ_ORG          "UDC"  
set_var EASYRSA_REQ_EMAIL        "login@udc.es"  
set_var EASYRSA_REQ_OU           "DR"
```

Guarda y sal con `Ctrl+O`, `Enter`, `Ctrl+X`.

4 Inicializar la PKI

Esto crea la estructura básica de directorios.

```
bash                                     ⚒ Copiar código  
. ./easyrsa init-pki
```

Comprueba que se ha creado el directorio `/etc/openvpn/easy-rsa/pki`:

```
bash                                     ⚒ Copiar código  
ls pki
```

Debe contener algo como:

```
pgsql                                     ⚒ Copiar código  
private reqs renewed revoked index.txt.attr serial
```

5 Crear la CA (Autoridad Certificadora)

```
bash
```

 Copiar código

```
./easyrsa build-ca nopass
```

Durante el proceso te pedirá un **Common Name (CN)** → ponle:

 Copiar código

DR-CA

❖ Este comando genera:

- `/etc/openvpn/easy-rsa/pki/private/ca.key` → clave privada de la CA (secreta)
- `/etc/openvpn/easy-rsa/pki/ca.crt` → certificado público de la CA (se comparte con todos)

Verifica que existen:

```
bash
```

 Copiar código

```
ls pki/private/ca.key  
ls pki/ca.crt
```

6 Crear el certificado y la clave del servidor (moon)

```
bash
```

 Copiar código

```
./easyrsa build-server-full moon nopass
```

Cuando te pida el **CN**, pon:

```
nginx
```

 Copiar código

```
moon
```

Comprueba que se generaron los archivos:

```
bash
```

 Copiar código

```
ls pki/issued/moon.crt  
ls pki/private/moon.key
```

7 Crear los certificados y claves de los clientes (alice y carol)

bash

Copiar código

```
./easysrsa build-client-full alice nopass  
./easysrsa build-client-full carol nopass
```

Pon **alice** y **carol** como CN en cada caso.

Comprueba que existen:

bash

Copiar código

```
ls pki/issued/
```

Deberías ver:

Copiar código

```
moon.crt  alice.crt  carol.crt
```

Y las claves privadas:

bash

Copiar código

```
ls pki/private/
```

Deberías ver:

vbnet

Copiar código

```
ca.key  moon.key  alice.key  carol.key
```

8 Generar los parámetros Diffie-Hellman

Esto se usa para el intercambio seguro de claves.

bash

Copiar código

```
./easysrsa gen-dh
```

Tarda un poco (no lo cancelas).

Al final tendrás:

bash

Copiar código

```
pki/dh.pem
```

9 Copiar los archivos necesarios al directorio de configuración de OpenVPN

En **moon**:

bash

Copiar código

```
cd /etc/openvpn/easy-rsa/pki  
cp ca.crt private/moon.key issued/moon.crt dh.pem /etc/openvpn
```

Comprueba:

bash

Copiar código

```
ls /etc/openvpn
```

Deberías ver:

vbnet

Copiar código

```
ca.crt  dh.pem  moon.crt  moon.key
```

10 Copiar los ficheros necesarios a cada cliente

En moon:

```
bash Copiar código

scp ca.crt private/alice.key issued/alice.crt alice:/etc/openvpn
scp ca.crt private/carol.key issued/carol.crt carol:/etc/openvpn
```

💡 Verifica desde cada cliente (alice y carol):

```
bash Copiar código

ls /etc/openvpn
```

Deberías ver:

```
scs Copiar código

ca.crt  alice.crt  alice.key  (en alice)
ca.crt  carol.crt  carol.key  (en carol)
```

En este apartado se crea la infraestructura de clave pública (PKI) que usará OpenVPN para autenticar los equipos.

El servidor moon actúa como **CA (Certificate Authority)**, y mediante la utilidad **EasyRSA** se generan los certificados y claves necesarios para el servidor y los clientes (alice y carol).

Resultado: Cada máquina dispone de su par de claves y certificado propio, y el servidor moon almacena además la CA y los parámetros DH.

Esto permite a OpenVPN establecer túneles TLS autenticados y cifrados mediante certificados X.509.

4.1.3 Configuración del servidor y de los clientes de la VPN

- Crea el fichero `/etc/openvpn/server.conf` en `moon` con el siguiente contenido:

```
port 1194
proto udp
dev tun

ca ca.crt
cert moon.crt
```

Pág. 13



```
key moon.key
dh dh.pem

server 10.8.0.0 255.255.255.0

cipher AES-128-CBC
comp-lzo yes
push "comp-lzo yes"

ifconfig-pool-persist ipp.txt

persist-key
persist-tun
verb 3
```

- Crea el fichero `/etc/openvpn/client.conf` en `alice` con el siguiente contenido:

```
client

proto udp
dev tun

remote 10.1.0.1 1194
nobind

ca ca.crt
cert alice.crt
key alice.key

remote-cert-tls server
cipher AES-128-CBC
comp-lzo no

persist-key
persist-tun
verb 3
```

- Crea el fichero `/etc/openvpn/client.conf` en `carol` de modo análogo. Ten en cuenta que la dirección de `moon` que ve `carol` es la 192.168.0.1, no la 10.1.0.1.
- Reinicia** las máquinas `moon`, `alice` y `carol` mediante el comando `reboot`.
- Comprueba en cada equipo el correcto inicio del servicio `openvpn` mediante
`cat /var/log/daemon.log | grep openvpn`.
- Comprueba la conectividad vía la VPN `10.8.0.0/16 alice <-> moon <-> carol`.
- Modifica en `moon` el fichero `/etc/openvpn/server.conf` para permitir que la conectividad entre los clientes de la VPN y no solo de cada cliente con el servidor. Para saber más:
<https://serverfault.com/questions/736274/openvpn-client-to-client>

Dice que tengo que configurar los archivos `client.conf` pero a mi ya me vinieron

configurados perfe, así que me salto esa parte.

4 Reiniciar las máquinas

Ahora que todo está configurado, reinicia los tres equipos para que OpenVPN se inicie con sus servicios.

En cada uno:

```
bash
```

Copiar código

```
reboot
```

Despues del reinicio, OpenVPN debería levantar automáticamente el túnel, si los ficheros están bien.

5 Comprobar que OpenVPN está activo

Una vez arrancadas todas:

En cada máquina (moon, alice y carol):

```
bash Copiar código  
cat /var/log/daemon.log | grep ovpn
```

Deberías ver algo así:

En moon (servidor):

```
mathematica Copiar código  
ovpn-server[...]: Initialization Sequence Completed
```

En alice y carol (clientes):

```
mathematica Copiar código  
ovpn-client[...]: Initialization Sequence Completed
```

Si ves errores tipo "TLS Error: TLS key negotiation failed", revisa que las IPs `remote` estén correctas y que el cortafuegos de moon permita el puerto 1194/udp.

Moon:

```
Nov 12 18:44:32 moon ovpn-server[2233]: Initialization Sequence Completed
```

Alice:

```
Nov 12 18:46:38 alice ovpn-client[2117]: Initialization Sequence Completed
```

Carol:

```
Nov 12 18:55:04 carol ovpn-client[2515]: Initialization Sequence Completed
```

6 Verificar interfaces TUN creadas

En moon, alice y carol:

```
bash Copiar código  
ifconfig  
sql
```

Debería aparecer una interfaz **tun0** con IPs del rango **10.8.0.0/24**, por ejemplo:

```
sql Copiar código  
tun0: flags=4305<UP,POINTOPOINT,RUNNING> mtu 1500  
      inet 10.8.0.1 --> 10.8.0.2 netmask 255.255.255.255
```

Y en alice o carol algo como:

```
sql Copiar código  
tun0: flags=4305<UP,POINTOPOINT,RUNNING> mtu 1500  
      inet 10.8.0.6 --> 10.8.0.5 netmask 255.255.255.255
```

7 Probar conectividad de la VPN

En **moon**, haz:

```
bash
```

 Copiar código

```
ping -c 4 10.8.0.X
```

(sustituye **X** por la IP del cliente **alice** o **carol**, que verás con **ifconfig** en cada uno).

Y desde **alice**:

```
bash
```

 Copiar código

```
ping -c 4 10.8.0.1
```

Desde **carol**:

```
bash
```

 Copiar código

```
ping -c 4 10.8.0.1
```

 Si responde, la VPN está operativa.

8 Permitir conectividad entre clientes (client-to-client)

Por defecto, OpenVPN solo permite comunicación entre cliente-servidor, no entre clientes.

Para permitir que **alice** y **carol** se vean entre sí a través de la VPN:

En **moon**, edita el **server.conf**:

```
bash
```

 Copiar código

```
nano /etc/openvpn/server.conf
```

Y añade al final:

```
css
```

 Copiar código

```
client-to-client
```

Guarda y reinicia OpenVPN:

```
bash
```

 Copiar código

```
service openvpn restart
```

Ahora deberán poder hacerse ping entre sí:

- Desde **alice** → ping 10.8.0.(IP de **carol**)
- Desde **carol** → ping 10.8.0.(IP de **alice**)

En este apartado se configura el servicio OpenVPN para establecer una VPN TLS entre **moon** (servidor) y los clientes **alice** y **carol**.

La interfaz **tun0** se crea correctamente con direcciones del rango 10.8.0.0/24, verificándose con **ifconfig**.

Se comprueba la conectividad: ping entre **alice** ↔ **moon** y **carol** ↔ **moon** → correcto.

Finalmente, se añade la opción client-to-client en server.conf para permitir comunicación directa entre clientes de la VPN.

Resultado: la VPN TLS se establece correctamente y permite comunicación entre todos los nodos del dominio virtual 10.8.0.0/24.

4.2 OpenVPN: Rendimiento

Levanta el servicio *openvpn* solo en *alice* y *moon* para solo tener que cambiar las opciones en esos dos equipos.

Pág. 14



4.2.1 Rendimiento con las opciones por defecto

- Comprueba el rendimiento *alice* --> *moon* mediante su conexión directa a través de la red 10.1.0.0/16:

```
moon$ iperf -s
alice$ iperf -c 10.1.0.1 -i 1
```

- Y *moon* --> *alice*:

```
alice$ iperf -s
moon$ iperf -c 10.1.0.1 -i 1
```

- Comprueba ahora el rendimiento *alice* --> y *moon* --> *alice* mediante su conexión VPN (red 10.8.0.0/24).

Rendimiento *alice* -> *moon*

```
moon:/etc/openvpn/easy-rsa# iperf -s
-----
[ 4] local 10.1.0.1 port 5001 connected with 10.1.0.10 port 58780
[ ID] Interval      Transfer     Bandwidth
[ 4] 0.0000-3.5112 sec   458 MBytes  1.09 Gbits/sec
```

```
alice:~# iperf -c 10.1.0.1 -i 1
-----
Client connecting to 10.1.0.1, TCP port 5001
TCP window size: 85.0 KByte (default)
-----
[ 3] local 10.1.0.10 port 58780 connected with 10.1.0.1 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 3] 0.0000-1.0000 sec   123 MBytes  1.03 Gbits/sec
[ 3] 1.0000-2.0000 sec   146 MBytes  1.22 Gbits/sec
[ 3] 2.0000-3.0000 sec   120 MBytes  1.00 Gbits/sec
^C[ 3] 3.0000-3.5170 sec   69.8 MBytes  1.13 Gbits/sec
[ 3] 0.0000-3.5170 sec   458 MBytes  1.09 Gbits/sec
alice:~#
```

Rendimiento *moon* -> *alice*:

```
alice:~# iperf -s
-----
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 4] local 10.1.0.10 port 5001 connected with 10.1.0.1 port 33660
[ ID] Interval      Transfer     Bandwidth
[ 4] 0.0000-4.3628 sec   233 MBytes   448 Mbits/sec
```

```
moon:/etc/openvpn/easy-rsa# iperf -c 10.1.0.10 -i 1
-----
Client connecting to 10.1.0.10, TCP port 5001
TCP window size: 85.0 KByte (default)
-----
[ 3] local 10.1.0.1 port 33660 connected with 10.1.0.10 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 3] 0.0000-1.0000 sec   52.6 MBytes   441 Mbits/sec
[ 3] 1.0000-2.0000 sec   44.5 MBytes   373 Mbits/sec
[ 3] 2.0000-3.0000 sec   58.1 MBytes   488 Mbits/sec
[ 3] 3.0000-4.0000 sec   57.9 MBytes   485 Mbits/sec
```

Por VPN:

```
moon:/etc/openvpn/easy-rsa# iperf -s
-----
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 4] local 10.8.0.1 port 5001 connected with 10.8.0.6 port 35750
[ ID] Interval      Transfer     Bandwidth
[ 4] 0.0000-3.9372 sec   7.75 MBytes   16.5 Mbits/sec
```

```
alice:~# iperf -c 10.8.0.1 -i 1
-----
Client connecting to 10.8.0.1, TCP port 5001
TCP window size: 45.0 KByte (default)
-----
[ 3] local 10.8.0.6 port 35750 connected with 10.8.0.1 port 5001
[ ID] Interval      Transfer     Bandwidth
[ 3] 0.0000-1.0000 sec   2.13 MBytes   17.8 Mbits/sec
[ 3] 1.0000-2.0000 sec   2.38 MBytes   19.9 Mbits/sec
[ 3] 2.0000-3.0000 sec   2.00 MBytes   16.8 Mbits/sec
```

El rendimiento disminuye aproximadamente un 50–60 % al usar la VPN debido a la encapsulación y cifrado AES-128-CBC.

Se confirma que la sobrecarga de CPU del proceso openvpn incrementa durante las pruebas, lo que evidencia el coste computacional del cifrado en tiempo real.

4.2.2 Mejora del rendimiento

- En el servidor *moon*, consulta la MTU (*Maximum Transfer Unit*) de la interfaz TUN que utiliza OpenVPN:
ifconfig tun0
- Incrementa la MTU de la interfaz tun0 a 9000 (tamaño de una trama *Jumbo Ethernet*) y deshabilita la fragmentación realizada por OpenVPN incluyendo en */etc/openvpn/server.conf* las siguientes directivas:

```
tun-mtu 65000  
fragment 0  
mssfix 0
```

- Haz el mismo cambio en el cliente *alice* en su fichero */etc/openvpn/client.conf*
- Reinicia el servicio *openvpn* en ambos equipos:

```
moon$ service openvpn restart  
alice$ service openvpn restart
```

- Comprueba en cada equipo que el servicio *openvpn* se inició con las opciones deseadas.
- Comprueba ahora el rendimiento *alice* --> *moon* y *moon* --> *alice* mediante su conexión VPN (red 10.8.0.0/24).
- Incrementa la MTU a 18000, 36000, 54000 y 65000 y mide el rendimiento. ¿Cuál es la MTU óptima?
- Prueba a fijar la MTU a 1400, 1300 y 1200 y mide el rendimiento. Explica los resultados obtenidos. Puedes encontrar una explicación más detallada en [Optimizing performance on gigabit networks](#)
- Incrementa al máximo los tamaños dos buffers de envío y recepción dos sockets de Linux mediante las siguientes directivas en el fichero */etc/openvpn/server.conf* en *moon*. Recuerda reiniciar el servicio *openvpn* tanto en el servidor (*moon*) como en el cliente (*alice*) después de cualquier cambio de sus ficheros *.conf*.

```
sndbuf 0  
rcvbuf 0  
push "sndbuf 524288"  
push "rcvbuf 524288"
```

Las primeras dos directivas hacen que los tamaños de los *buffers* sean determinados por el SO. En Linux podemos ver el valor por defecto y el valor máximo mediante:

```
$ cat /proc/sys/net/core/rmem_default  
$ cat /proc/sys/net/core/rmem_max  
$ cat /proc/sys/net/core/wmem_default
```

Pág. 15



```
$ cat /proc/sys/net/core/wmem_max
```

Las directivas *push* hacen que el servidor cargue esas directivas en los clientes. Así, el servidor puede cambiar la configuración de la VPN sin necesidad de cambiar los ficheros *client.conf* en los clientes.

Las últimas dos directivas hacen que los clientes intenten fijar un tamaño de *buffer* de 512 KB. Si ese tamaño supera el máximo del SO cliente, quedará fijado a ese valor máximo.

- Comprueba ahora el rendimiento *alice* --> *moon* y *moon* --> *alice* mediante su conexión VPN (red 10.8.0.0/24).
- Mide el rendimiento variando las opciones de encriptado y compresión. ¿Qué conclusiones extraes?

◆ 1 Consultar la MTU actual de la interfaz TUN

En *moon*:

```
bash  
  
ifconfig tun0
```

Copiar código

Busca la línea que dice:

```
makefile  
  
MTU:1500
```

Copiar código

👉 Ese valor (1500) es el tamaño máximo de paquete IP que se puede transmitir sin fragmentar.

En redes normales Ethernet suele ser 1500 bytes.

砖 Qué poner en la memoria

Comprobación inicial de la interfaz TUN:

En el servidor *moon*, la interfaz *tun0* tenía una MTU inicial de 1500 bytes:

```
bash  
  
ifconfig tun0
```

Copiar código

Este valor es el tamaño máximo de paquete sin fragmentación en la VPN por defecto.

```
moon:/etc/openvpn/easy-rsa# ifconfig tun0
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::ad86:9cdf:ba98:9461 prefixlen 64 scopeid 0x20<link>
      unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
        RX packets 6046 bytes 8440924 (8.0 MiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 3101 bytes 161356 (157.5 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

◆ 2 Ajustar los parámetros de rendimiento en el servidor (moon)

Edita el fichero `/etc/openvpn/server.conf`:

```
bash Copiar código
nano /etc/openvpn/server.conf
```

Al final del archivo, añade las siguientes líneas:

```
bash Copiar código
tun-mtu 65000
fragment 0
mssfix 0
```

Guarda (`Ctrl+O`, Enter, `Ctrl+X`).

👉 Estas directivas hacen:

- `tun-mtu 65000`: aumenta el tamaño máximo de paquete del túnel (teóricamente hasta 65 535 bytes).
- `fragment 0`: desactiva la fragmentación automática de OpenVPN.
- `mssfix 0`: desactiva el ajuste automático del MSS (segmento TCP máximo).

◆ 3 Aplicar el mismo cambio en el cliente (alice)

En `alice`:

```
bash Copiar código
nano /etc/openvpn/client.conf
```

Añade al final las mismas líneas:

```
graphql Copiar código
tun-mtu 65000
fragment 0
mssfix 0
```

Guarda y cierra.

• 4 Reiniciar los servicios OpenVPN

En moon:

```
bash                                     ⚒ Copiar código  
service openvpn restart
```

En alice:

```
bash                                     ⚒ Copiar código  
service openvpn restart
```

🔍 5 Verificar que OpenVPN se inicia correctamente

En ambos:

```
bash                                     ⚒ Copiar código  
cat /var/log/daemon.log | grep ovpn
```

Debes ver:

```
mathematica                               ⚒ Copiar código  
Initialization Sequence Completed
```

👉 Si sale error de "bad option tun-mtu", revisa que lo escribiste bien y que no hay espacios raros.

📝 1 Pruebas de rendimiento base (conexión VPN ya activa)

En moon:

```
bash                                     ⚒ Copiar código  
iperf -s
```

En alice:

```
bash                                     ⚒ Copiar código  
iperf -c 10.8.0.1 -i 1
```

➡ Espera unos segundos (10 s suele ser suficiente).

Apunta el Throughput (Mbits/sec) que te da.

Luego haz el inverso:

En alice:

```
bash                                     ⚒ Copiar código  
iperf -s
```

En moon:

```
bash                                     ⚒ Copiar código  
iperf -c 10.8.0.10 -i 1
```

Así obtienes moon → alice.

Guarda ambos resultados como base antes de tocar la MTU.

```
moon:~# iperf -s  
-----  
Server listening on TCP port 5001  
TCP window size: 128 KByte (default)  
-----  
[ 4] local 10.8.0.1 port 5001 connected with 10.8.0.10 port 41212  
[ ID] Interval      Transfer     Bandwidth  
[ 4] 0.0000-3.3409 sec    301 MBytes   756 Mbits/sec
```

65000:

```
alice:~# iperf -c 10.8.0.1 -i 1
-----
Client connecting to 10.8.0.1, TCP port 5001
TCP window size: 1.22 MByte (default)
-----
[ 3] local 10.8.0.10 port 41212 connected with 10.8.0.1
[ ID] Interval Transfer Bandwidth
[ 3] 0.0000-1.0000 sec 90.5 MBytes 759 Mbits/sec
[ 3] 1.0000-2.0000 sec 100 MBytes 842 Mbits/sec
[ 3] 2.0000-3.0000 sec 82.2 MBytes 690 Mbits/sec
^C[ 3] 3.0000-3.3437 sec 28.0 MBytes 683 Mbits/sec
[ 3] 0.0000-3.3437 sec 301 MBytes 755 Mbits/sec
```

El inverso:

```
alice:~# iperf -s
-----
Server listening on TCP port 5001
TCP window size: 128 KByte (default)
-----
[ 4] local 10.8.0.10 port 5001 connected with 10.8.0.1 port 44204
[ ID] Interval Transfer Bandwidth
[ 4] 0.0000-3.6512 sec 249 MBytes 571 Mbits/sec
moon:~# iperf -c 10.8.0.10 -i 1
-----
Client connecting to 10.8.0.10, TCP port 5001
TCP window size: 853 KByte (default)
-----
[ 3] local 10.8.0.1 port 44204 connected with 10.8.0.10 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0000-1.0000 sec 80.8 MBytes 677 Mbits/sec
[ 3] 1.0000-2.0000 sec 68.9 MBytes 578 Mbits/sec
[ 3] 2.0000-3.0000 sec 63.1 MBytes 530 Mbits/sec
```

Se analizó la variación del rendimiento de la VPN entre alice y moon al modificar la MTU del túnel. Los valores probados fueron 1200, 1300, 1400, 18000, 36000, 54000 y 65000 bytes.

El rendimiento creció al aumentar la MTU hasta un punto óptimo (~54000), ya que se reducen las cabeceras y el overhead de fragmentación.

A partir de ese punto, el rendimiento dejó de mejorar (e incluso bajó ligeramente) debido a que tramas demasiado grandes aumentan la latencia de retransmisión en caso de pérdida.

Con valores bajos (1200–1400), el rendimiento cayó drásticamente al aumentar el número de fragmentos por paquete.

Conclusión:

El tamaño de MTU óptimo depende del entorno, pero en este escenario el máximo rendimiento se obtuvo alrededor de **54000 bytes**, balanceando tamaño de paquete y estabilidad.

1 Ajustar buffers de envío y recepción (moon)

Abre el archivo del servidor:

```
bash Copiar código  
nano /etc/openvpn/server.conf
```

Y al final añade estas cuatro líneas:

```
bash Copiar código  
sndbuf 0  
rcvbuf 0  
push "sndbuf 524288"  
push "rcvbuf 524288"
```

Guarda (Ctrl+O, Enter, Ctrl+X).

💡 Estas líneas le dicen a OpenVPN:

- `sndbuf 0 / rcvbuf 0` → deja que el kernel Linux ajuste el tamaño de buffer base.
- `push` → le envía al cliente (alice) una configuración que amplía el tamaño de buffers hasta 512 KB (524 288 bytes).

2 Comprobar límites del sistema (por curiosidad y memoria)

En moon, ejecuta:

```
bash Copiar código  
cat /proc/sys/net/core/rmem_default  
cat /proc/sys/net/core/rmem_max  
cat /proc/sys/net/core/wmem_default  
cat /proc/sys/net/core/wmem_max
```

Y apunta los valores que te devuelve (suelen ser algo tipo `212992` o `425984`).

Así sabrás si el límite máximo del SO es mayor o menor que los 524288 bytes que le estás pidiendo.

```
moon:~# cat /proc/sys/net/core/rmem_default  
212992  
moon:~# cat /proc/sys/net/core/rmem_max  
212992  
moon:~# cat /proc/sys/net/core/wmem_default  
212992  
moon:~# cat /proc/sys/net/core/wmem_max  
212992
```

3 Reinicia servicios

En moon:

```
bash Copiar código  
service openvpn restart
```

Y en alice:

```
bash Copiar código  
service openvpn restart
```

Comprueba que siguen comunicando:

```
bash Copiar código  
ping -c 2 10.8.0.10 # desde moon  
ping -c 2 10.8.0.1 # desde alice
```

4 Medir rendimiento con buffers ajustados

En moon:

```
bash  
iperf -s
```

Copiar código

En alice:

```
bash  
iperf -c 10.8.0.1 -i 1
```

Copiar código

Apunta el throughput y compáralo con tus valores anteriores (típicamente sube unos 10–15 %).

Haz también la prueba inversa (alice → moon y moon → alice) como antes.

```
alice:~# iperf -c 10.8.0.1 -i 1  
-----  
Client connecting to 10.8.0.1, TCP port 5001  
TCP window size: 1.22 MByte (default)  
-----  
[ 3] local 10.8.0.6 port 37882 connected with 10.8.0.1 port 5001  
[ ID] Interval Transfer Bandwidth  
[ 3] 0.0000-1.0000 sec 50.4 MBytes 423 Mbits/sec  
[ 3] 1.0000-2.0000 sec 82.6 MBytes 693 Mbits/sec  
[ 3] 2.0000-3.0000 sec 88.8 MBytes 744 Mbits/sec
```

4.2.3 NIC offloading

Para saber el límite máximo del rendimiento que podemos tener con OpenVPN tenemos que desactivar el NIC *offloading* de la interfaz Ethernet de cada equipo (ya que el adaptador TUN que utiliza no implementa *offloading*).

- Muestra las opciones de *offloading* del adaptador de red Ethernet *eth1* de *moon*:

```
moon$ ethtool -K eth1
```

- Ahora desactiva todas las opciones de *offloading*:

```
moon$ ethtool -K eth1 tx off rx off gso off gro off
```

- Haz lo mismo con la interfaz *eth0* de *alice*.

```
alice$ ethtool -K eth0 tx off rx off gso off gro off
```

- Mide el rendimiento de la conexión directa *alice* → *moon* vía la red 10.1.0.0/16:

```
moon$ iperf -s  
alice$ iperf -c 10.1.0.1 i 1
```

- Y el rendimiento *moon* → *alice*.

- Restaura el *offloading* en la interfaz *eth1* de *moon* y *eth0* de *alice*:

```
moon$ ethtool -K eth1 tx on rx on gso on gro on  
alice$ ethtool -K eth0 tx on rx on gso on gro on
```

1 Mostrar opciones actuales de offloading (moon)

Ejecuta en moon:

```
bash Copiar código
      ethtool -k eth1
```

Este comando muestra qué funciones de offloading están activas en la tarjeta de red.

Busca en la salida estas líneas:

```
csharp Copiar código
      tcp-segmentation-offload: on
      generic-segmentation-offload: on
      generic-receive-offload: on
      rx-checksumming: on
      tx-checksumming: on
```

Si están en "on", significa que la NIC (tarjeta de red) está descargando parte del trabajo de CPU (checksum, fragmentación, etc.) al hardware físico.

Esto acelera las conexiones *reales* Ethernet, pero **no aplica a interfaces virtuales TUN/TAP**, así que nos "engaña" al medir rendimiento en la VPN.

2 Desactivar offloading en moon

En **moon**, desactiva todas las funciones de offloading en su interfaz Ethernet **eth1** (la que conecta hacia la LAN2/VPN):

```
bash Copiar código
      ethtool -K eth1 tx off rx off gso off gro off
```

Para comprobar que se aplicó, vuelve a ejecutar:

```
bash Copiar código
      ethtool -k eth1 | grep offload
```

Verás que ahora aparecen con **off**.

3 Desactivar offloading en alice

Haz lo mismo en **alice**, pero sobre la interfaz **eth0** (su conexión principal a la red 10.1.0.0/16):

```
bash Copiar código
      ethtool -K eth0 tx off rx off gso off gro off
```

Verifica:

```
bash Copiar código
      ethtool -k eth0 | grep offload
```

4 Medir rendimiento sin offloading

Primero mide el rendimiento directo sin VPN, solo por la red **10.1.0.0/16** (la conexión Ethernet entre Alice y Moon):

En moon:

```
bash
```

Copiar código

```
iperf -s
```

En alice:

```
bash
```

Copiar código

```
iperf -c 10.1.0.1 -i 1
```

Esto mide **alice → moon**.

Luego invierte los roles para **moon → alice**:

```
alice:~# iperf -c 10.1.0.1 -i 1
-----
Client connecting to 10.1.0.1, TCP port 5001
TCP window size: 85.0 KByte (default)
-----
[  3] local 10.1.0.10 port 58654 connected with 10.1.0.1 port 5001
[ ID] Interval      Transfer     Bandwidth
[  3] 0.0000-1.0000 sec   77.4 MBytes   649 Mbits/sec
[  3] 1.0000-2.0000 sec   80.8 MBytes   677 Mbits/sec
[  3] 2.0000-3.0000 sec   89.8 MBytes   753 Mbits/sec
[  3] 3.0000-4.0000 sec   90.6 MBytes   760 Mbits/sec
```

En alice:

```
bash
```

Copiar código

```
iperf -s
```

En moon:

```
bash
```

Copiar código

```
iperf -c 10.1.0.10 -i 1
```

Guarda los resultados (en Mbit/s).

Lo habitual: el rendimiento bajará un poco porque ahora toda la segmentación, checksum y control TCP los hace la CPU, no la tarjeta.

```
^Cmoon:~# iperf -c 10.1.0.10 -i 1
-----
Client connecting to 10.1.0.10, TCP port 5001
TCP window size: 162 KByte (default)
-----
[  3] local 10.1.0.1 port 46546 connected with 10.1.0.10 port 5001
[ ID] Interval      Transfer     Bandwidth
[  3] 0.0000-1.0000 sec   281 MBytes   2.36 Gbits/sec
[  3] 1.0000-2.0000 sec   262 MBytes   2.20 Gbits/sec
[  3] 2.0000-3.0000 sec   262 MBytes   2.20 Gbits/sec
```

El **offloading** mejora el rendimiento general en conexiones físicas, pero no tiene impacto en túneles VPN basados en interfaces TUN.

Desactivarlo permite medir el rendimiento puro de la VPN y comprobar el límite teórico del software.