Tema 3 – Taller sobre VPNs: strongSwan (IPsec) y OpenVPN (TLS)

1. Conceptos básicos de enrutamiento y cortafuegos

1.1 Red de pruebas

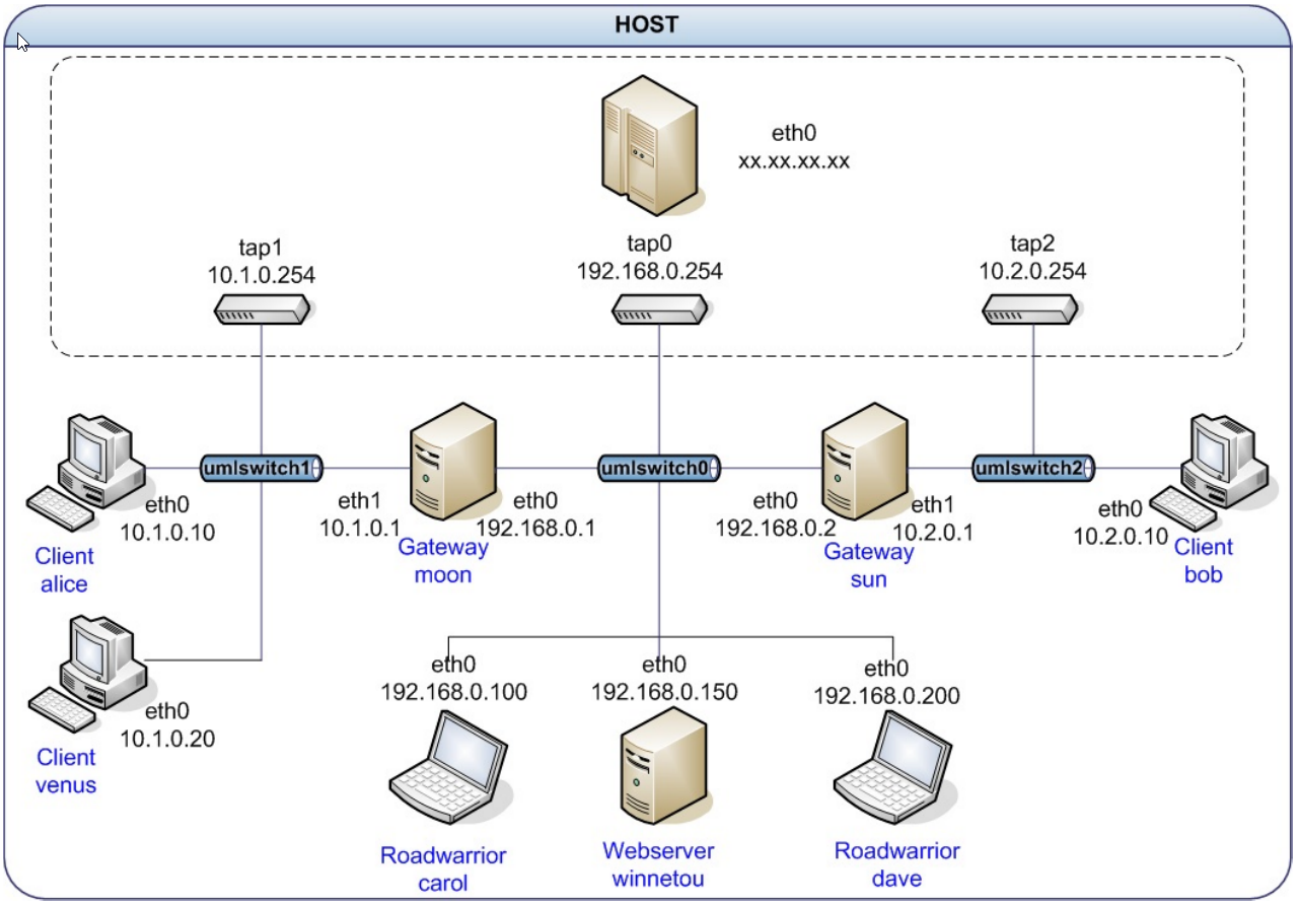

Figura 1 Diagrama de la red del entorno de pruebas de strongSwan.

La Figura 1 muestra la red emulada por el entorno de pruebas de strongSwan. La Figura 2 muestra las subredes involucradas. La idea de esta red es contemplar el escenario de dos 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.

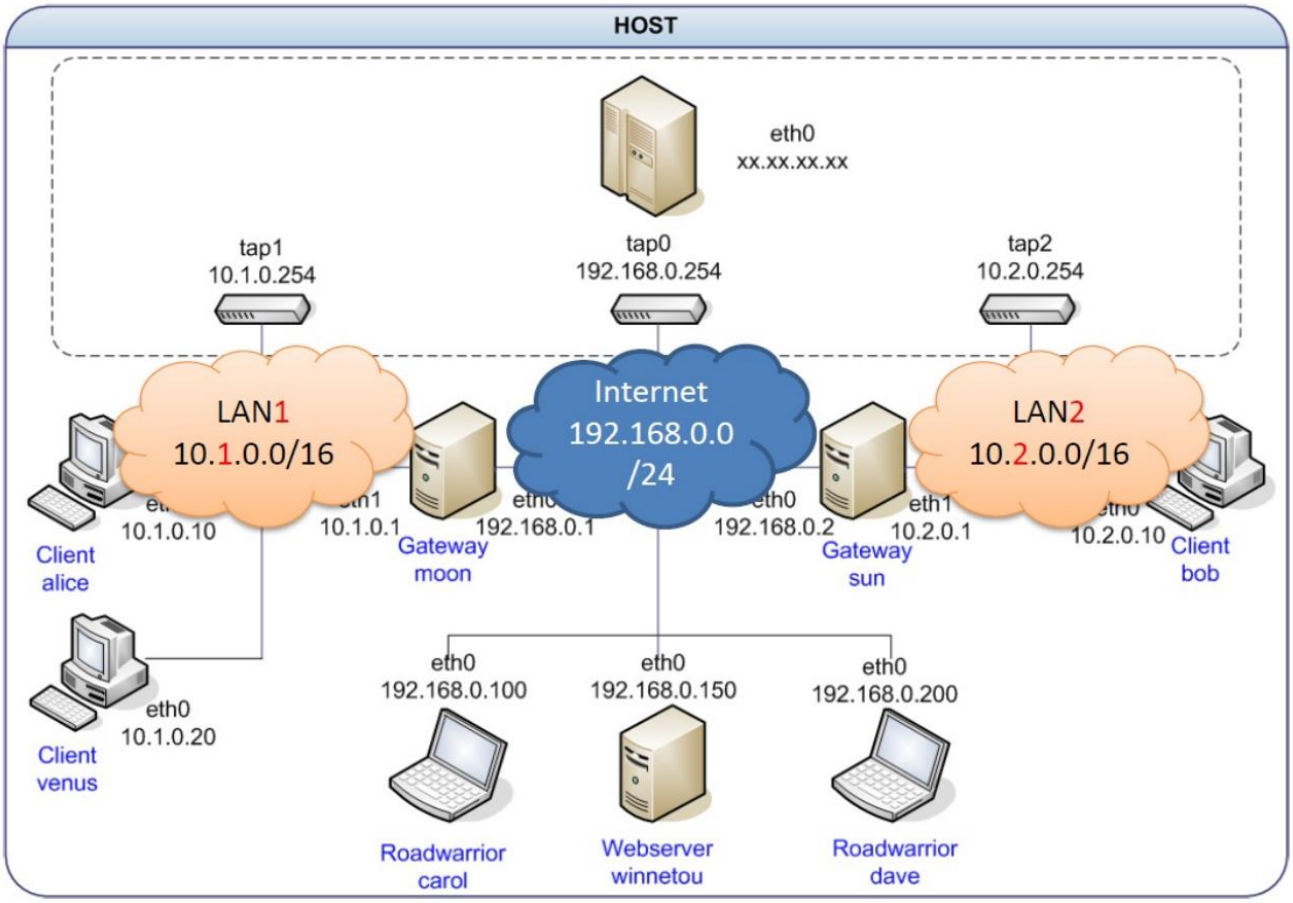

Figura 2 Subredes de la red del entorno de pruebas de strongSwan.

La red del entorno de pruebas de strongSwan se puede usar para otros experimentos no relacionados con strongSwan. En nuestro caso, la usaremos también como red de pruebas para OpenVPN.

NOTAS

- Todos los comandos debes ejecutarlos 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>`.
- 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 en la contraria.
- Referencias sobre *iptables*
  - http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m6/cortafuegos_iptables.html
  - http://www.brennan.id.au/06-Firewall_Concepts.html
  - https://www.karlrupp.net/en/computer/nat_tutorial

1.2 Pruebas básicas

- Levanta la red de pruebas:

```
$ ./start-testing
```

- Comprueba si cada equipo tiene activado el enrutamiento:

```
host$ sysctl -a | grep ip_forward
```

- **Comandos de utilidad**
  - `ip route`
  - `ip address show`
  - `ip route show`
  - `iptables -L [-v] [--line-number]`
  - `iptables -S`
  - `ping`
  - `traceroute`

- Muestra las tablas de enrutamiento de los equipos *alice*, *bob*, *moon*, *sun*, *carol* y *winnetou*.
- 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*?
  - ¿Es posible alcanzar *carol* desde *moon*? ¿Desde *winnetou*? ¿Desde *bob*?
  - ¿Es posible alcanzar *alice* desde *sun*? ¿Desde *winnetou*? ¿Desde *moon*?
- ¿Cuáles son las direcciones del equipo *omnet6-vm*? ¿Qué direcciones del equipo anfitrión es posible alcanzar desde cada equipo?

1.3 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*
- 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*
- Borra esa ruta

```
carol$ ip route del 10.1.0.0/16
```

- Añada rutas en *moon* y *sun* para conectar las redes LAN1 y LAN2

```
moon$ ip route add 10.2.0.0/16 via 192.168.0.2
sun$ ip route add 10.1.0.0/16 via 192.168.0.1
```

  Ten en cuenta que son necesarias las dos rutas: solo con una no tenemos camino de vuelta.

- Borra esas rutas en *moon* y *sun*

```
moon$ ip route del 10.2.0.0/16
sun$ ip route del 10.1.0.0/16
```

2. Conceptos básicos de cortafuegos

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

```
omnetvm2024$ route -n
omnetvm2024$ iptables -L -v
```

  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:

```
omnetvm2024$ iptables -t nat -L -v
```

- ¿Qué reglas *iptables* aíslan el tráfico de cada una de las tres redes?
- Prueba a añadir como primera regla de la cadena FORWARD un "aceptar todo", y comprueba si cambia en algo la conectividad entre los equipos de distintas redes:

```
omnetvm2024$ iptables -I FORWARD 1 -j ACCEPT
```

- Eimina la regla anterior:

```
omnetvm2024$ iptables -D FORWARD 1
```

2.1 NAT básico

Source NAT (MASQUERADE)

- Habilita SNAT (tipo MASQUERADE) en *moon* para permitir la conexión desde la red 10.1.0.0/16 a la red 192.168.0.0/24:

```
moon$ iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

- Muestra las iptables (chains) de la tabla NAT:

```
moon$ iptables -t nat -L -v
```

- 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`.
  - ¿Cuál es la dirección IP origen de *alice* que ve *carol*? ¿Y *alice* de *carol*?
- Si ejecutamos `netstat -4an` en el router *moon*, ¿vemos alguna conexión que muestre el NAT que realiza?
- 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
```

- Elimina la regla de MASQUERADE:

```
moon$ iptables -t nat -D POSTROUTING -o eth0 -s 10.1.0.10 -j MASQUERADE
```

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

```
moon$ iptables -t nat -A POSTROUTING -o eth0 -s 10.1.0.10 -p tcp --dport 80 -j
MASQUERADE
```

- Compruébalo con `alice$ ssh -p 80 winnetou`
- Comprueba ahora esas posibles conexiones mediante `ping` desde cada uno de los equipos.
- Elimina la regla de MASQUERADE:

```
moon$ iptables -t nat -D POSTROUTING -o eth0 -s 10.1.0.10 -p tcp --dport 80 -j
MASQUERADE
```

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
```

- Comprobamos `ssh` *carol* --> *alice* vía DNAT: `carol$ ssh moon -p 10022`
- Elimina la regla de DNAT

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

2.2 Pruebas de velocidad

- Realiza una prueba de velocidad *alice* --> *moon*:

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

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

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

- ¿Hay alguna diferencia de velocidad?

3. IPsec: strongSwan

Los escenarios que vamos a ver a continuación forman parte del conjunto de tests de 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 *rw*). 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:

```
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
```

En el caso del *gateway moon* hay que aplicar las mismas reglas de *mangling*.

Si no se efectúa este *mangling*, **no funcionan las conexiones TCP desde los clientes agnósticos de IPsec aunque funcione el `ping`**. Ver:

https://wiki.strongswan.org/projects/strongswan/wiki/ForwardingAndSplitTunneling#MTUMSS-issues

https://www.zeitgeist.se/2013/11/26/mtu-woes-in-ipsec-tunnels-how-to-fix/

3.1 IPsec: escenario ikev2/esp-alg-null

- Documentación
- Carga el escenario:

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

- Abre una terminal con *alice*, *moon* y *carol*.
- Comprueba que no es posible alcanzar *alice* desde *carol* y viceversa.
- Muestra las tablas de enrutamiento y las *iptables* de cada equipo.
- Inicia IPsec en *moon* y *carol*

```
moon$ systemctl start strongswan
carol$ systemctl start strongswan
```

- 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.

- **Contenido de */etc/swanctl/swanctl.conf* en *moon*:**

```
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
   }
}
```

- **Contenido de */etc/ipsec.conf* en *carol*:**

```
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
   }
}
```

- En el caso de *moon*:
  - ¿Qué interfaz (dirección IP) de moon utiliza la conexión *rw*?
  - ¿Qué versión de IKE soporta?
  - ¿Qué propuesta de protocolo de cifrado, integridad y grupo de intercambio de claves Diffie-Hellman admite para la asociación de seguridad IKE?
  - ¿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?
- En el caso de *carol*:

  - ¿Qué interfaz (dirección IP) local y remota utiliza la conexión *home*?
  - ¿Qué versión de IKE soporta?
  - ¿Qué propuesta de protocolo de cifrado, integridad y grupo de intercambio de claves Diffie-Hellman admite para la asociación de seguridad IKE?
  - ¿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?

- Carga las conexiones:

```
moon$ expect-connection rw
carol$ expect-connection home
```

- Muestra las tablas de enrutamiento y las *iptables* de cada equipo. ¿Ha cambiado algo?
- Levanta ahora la conexión IPsec *home* desde *carol*:

```
carol$ swanctl --initiate --child home
```

- En caso de fallo, puedes examinar el fichero */var/log/daemon.log* de *carol* y *moon*.
- Comprueba que ahora sí es posible la conectividad *carol* --> *alice*
- 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`.
- Comprueba si es posible hacer un `ping` *carol* --> *moon* o *moon* --> *carol*.
- Muestra las conexiones IPsec en *alice*, *moon* y *carol* con el comando `swanctl –list-sas`
  - ¿Cuál es el algoritmo de *encriptado* usado en la IKE SA? ¿Y en la ESP SA?
  - ¿Cuál es el algoritmo de *integridad* usado en la IKE SA? ¿Y en la ESP SA?
  - ¿Cuál es algoritmo usado para el Diffie-Hellman en IKE?
  - ¿Cuánto falta para se haga un *rekeying* de la IKE SA? ¿Y de la ESP SA?
  - ¿Cuáles son los SPI de la IKE SA? ¿Cuál corresponde a *carol* y cuál a *moon*?
  - ¿Cuáles son los SPI de la ESP SA? ¿Cuál corresponde a *carol* y cuál a *moon*?
- Mide la velocidad *alice* --> *carol* y viceversa con `iperf` tanto con TCP como con UDP.
- Para el servicio strongswan:

```
moon$ systemctl stop strongswan
carol$ systemctl stop strongswan
```

3.2 IPsec: escenario host2host

3.2.1 host2host ESP modo túnel

- Documentación
- Carga el escenario:

```
omnetvm2024$ scripts/load-testconfig ikev2/host2host-cert
```

- Abre una terminal en *moon* y en *sun*
- 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*.
- Inicia strongSwan en *moon* y *sun*:

```
moon$ systemctl start strongswan
sun$ systemctl start strongswan
```

- **Contenido de */etc/swanctl/swanctl.conf* en *moon*:**

```
connections {

   host-host {
      local_addrs = 192.168.0.1
      remote_addrs = 192.168.0.2

      local {
         auth = pubkey
         certs = moonCert.pem
         id = moon.strongswan.org
      }
      remote {
         auth = pubkey
         id = sun.strongswan.org
      }
      children {
         host-host {
            updown = /usr/local/libexec/ipsec/_updown iptables
            rekey_time = 5400
            rekey_bytes = 500000000
            rekey_packets = 1000000
            esp_proposals = aes128gcm128-x25519
         }
      }
      version = 2
      mobike = no
      reauth_time = 10800
      proposals = aes128-sha256-x25519
   }
}
```

- **Contenido de */etc/swanctl/swanctl.conf* en *sun*:**

```
connections {

   host-host {
      local_addrs = 192.168.0.2
      remote_addrs = 192.168.0.1

      local {
         auth = pubkey
         certs = sunCert.pem
         id = sun.strongswan.org
```

```
            }
            remote {
               auth = pubkey
               id = moon.strongswan.org
            }
            children {
               host-host {
                  updown = /usr/local/libexec/ipsec/_updown iptables
                  rekey_time = 5400
                  rekey_bytes = 500000000
                  rekey_packets = 1000000
                  esp_proposals = aes128gcm128-x25519
               }
            }
            version = 2
            mobike = no
            reauth_time = 10800
            proposals = aes128-sha256-x25519
         }
      }
```

- En el caso de *moon*:
  - ¿Qué interfaz (dirección IP) de *moon* utiliza la conexión *host-host*?
  - ¿Qué versión de IKE soporta?
  - ¿Qué propuesta de protocolo de cifrado, integridad y grupo de intercambio de claves Diffie-Hellman admite para la asociación de seguridad IKE?
  - ¿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?
- Muestra las tablas de enrutamiento y las *iptables* de cada equipo. ¿Cambió algo?
- Carga las conexiones:

```
moon$ expect-connection host-host
sun$ expect-connection host-host
```

- Levanta ahora la conexión *host-host* en *moon*:

```
moon$ swanctl --initiate --child host-host
```

- Comprueba que ahora sí es posible la conectividad *moon* <--> *sun*
- 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*?
- Muestra las conexiones IPsec en *moon* y *sun* con el comando `swanctl –list-sas`
  - ¿Cuál es el algoritmo de *encriptado* usado en la IKE SA? ¿Y en la ESP SA?
  - ¿Cuál es el algoritmo de *integridad* usado en la IKE SA? ¿Y en la ESP SA?
  - ¿Cuál es algoritmo usado para el Diffie-Hellman en IKE?
  - ¿Cuánto falta para se haga un *rekeying* de la IKE SA? ¿Y de la ESP SA?
  - ¿Cuáles son los SPI de la IKE SA? ¿Cuál corresponde a *moon* y cuál a *sun*?
  - ¿Cuáles son los SPI de la ESP SA? ¿Cuál corresponde a *moon* y cuál a *sun*?

- Mide la velocidad *moon* --> *sun* y viceversa con `iperf` tanto con TCP como con UDP.

3.2.2 host2host AH modo transporte

- Documentación
- 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.

3.2.3 host2host ESP modo transporte

- Documentación
- Realiza los cambios necesarios para llevarlo a cabo.

3.2.4 host2host AH modo túnel

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

3.3 IPsec: escenario net2net

3.3.1 net2net modo transporte

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

3.3.2 net2net ESP modo túnel

- Documentación
- Carga el escenario:

```
omnetvm2024$ 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.
- Comprueba la conectividad entre las redes 10.1.0.0/16 y 10.2.0.0/16.
- Comprueba las características de la SA IPsec *net-net* consultando su estado.

3.3.3 net2net AH modo túnel

- Documentación
- Realiza los cambios necesarios para llevarlo a cabo.

4. VPN TLS: OpenVPN

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 interaccionan 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 descargarnos 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 roo@10.1.0.10:
omnet6-vm# scp -O *.deb root@192.168.0.100:
```

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

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

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:

```
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"
```

- 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/pki
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/pki
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
```

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
```

```
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 ovpn`
- Comprueba la conectividad vía la VPN 10.8.0.0/16 *alice* <--> *moon* y *carol* <--> *moon*.
- 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

4.2 OpenVPN: Rendimiento

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

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).

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
```

```
$ 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?

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
```