HowTo ASIX VPN

Virtual Private Network OpenVPN

Curs 2018-2019

Objectius

En aquest manual es treballen els següents conceptes clau:

- OpenVPN
- Establiment d'un túnel no xifrat host to host.
- Establiment d'un túnel Host to Host amb xifrat simètric (pre-shared key)
- Establiment d'un túnel Host to Host amb xifrat TLS
- Establiment de tràfic xifrat amb VPN Network to Network.
- Generar subxarxes amb Docker

- Servei OpenVPN, serveis personalitzats amb el template openvpn@.service
  - Generar un servidor multiclient. Connetar diversos clients.
  - Generar diversos serveis template.
- Applet gràfic de xarxes openvpn.

Documentació

- Pàgina de documentació de OpenVPN
- Exemples (els 4 dels exercicis) de la documentació OpenVPN.
- Man pages de l'ordre openvpn (inclou els 4 exemples implementats).
- Fedora project: openvpn

Pendent

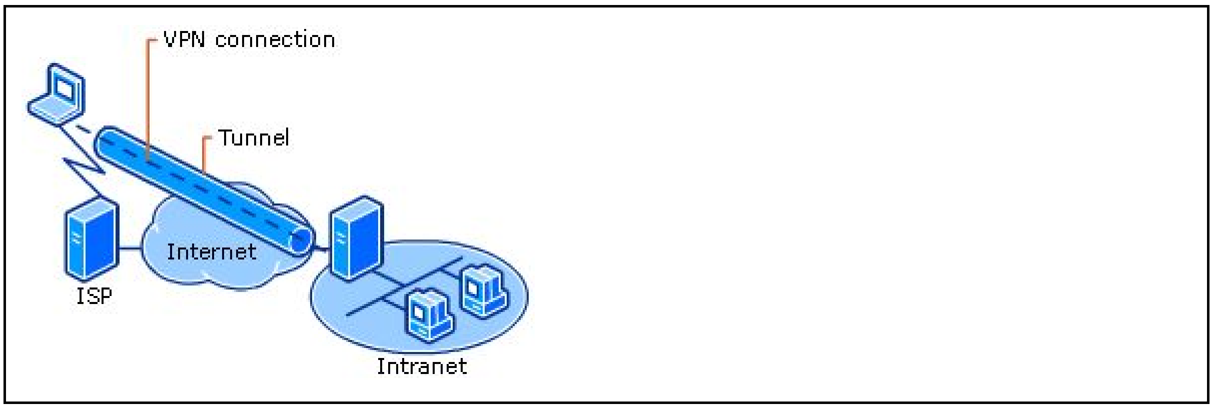
- VPN públiques per tràfic obscur a internet
- thor.
- docker network com a eina per fer exemples de xarxes diferents.
- iptables mascarade/NAT com a eina per fer xarxes diferents

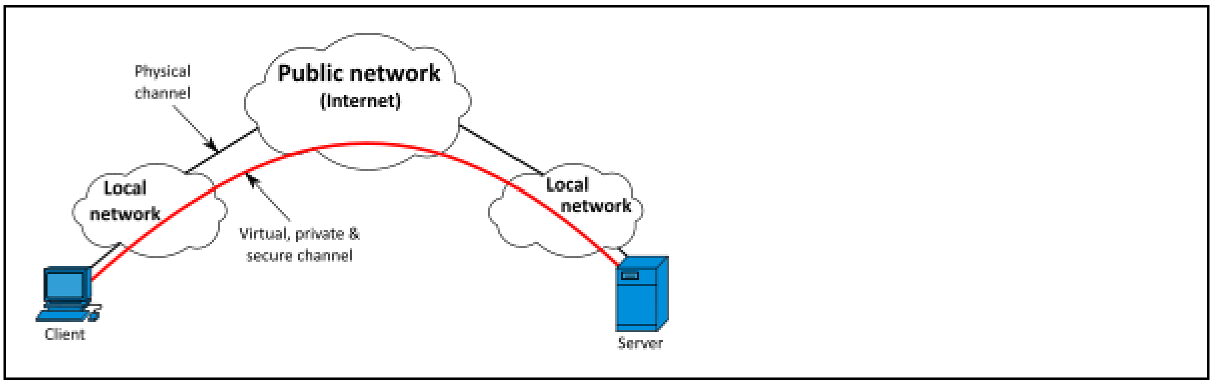
Implementació de túnels amb OpenVPN

Descripció general

Crear un túnel VPN és com tenir un aparell màgic que us obre una porta al rebedor de casa que quan la travesseu us porta per art de màgia / ciència a un altre edifici en una altra habitació. Estem tips de veure-ho, les naus espaials entren per un túnel del hiperespai i apareixen en una altra galàxia.

Observeu els dos dibuixos, en el primer s'ha generat un túnel còsmic que conecta des del portàtil (per exemple a Alderaan) a la intranet (per exemplea Jaku) i a l'inrevés. En el segon dibuix la línia vermella mostra aquest "camí de cuc" per n viatgen les naus espaials per saltar a l'hiperespai i aparèixer a l'altre extrem.





Malauradament com passa amb els reis mags (i alb els altres!) la màgia desapareix quan coneixem la realitat de les coses. Aquesta porta interestel·lar que apareix a cada extrem simplement és una nova interfície de xarxa, de tipus TUN (de tunel).

Tot el tràfic passa per aquesta interfícia i s'encipta i s'envia com a destí a la interfície TUN de l'altre costat. Ara bé, en realitat el tràfic no viatgja per l'hiperespai (el túnel del primer gràfic o la línia vermella del segon) sinó que viatja per internet com tot l'altre tràfic, però xifrat.

Per explicar-ho encara més 'kutre': fem aparèixer una porta TUN en cada extrem del rebedor de casa i del destí. En passar aquesta porta ens posen una caputxa al cap per xifrar-nos i sortim caminant per la porta del carrer normal, viatgem per el carrer normal (però xifrats) i quan arribem al destí en passar per la porta màgica TUN del destí ens treuen la caputxa (i ens diuen que hem viatjat per l'hiperespai o que hem de deixar de prendre segons quines substàncies…).

Descripció dels components

```
[root@profeN2I ~]# rpm -ql openvpn
/etc/openvpn
/usr/include/openvpn-plugin.h
/usr/lib/systemd/system/openvpn@.service
/usr/lib/tmpfiles.d/openvpn.conf
/usr/lib64/openvpn
/usr/lib64/openvpn/plugins
/usr/lib64/openvpn/plugins/openvpn-plugin-auth-pam.so
/usr/lib64/openvpn/plugins/openvpn-plugin-down-root.so
/usr/sbin/openvpn
/usr/share/doc/openvpn
/usr/share/doc/openvpn/AUTHORS
/usr/share/doc/openvpn/COPYING
/usr/share/doc/openvpn/COPYRIGHT.GPL
/usr/share/doc/openvpn/INSTALL
/usr/share/doc/openvpn/PORTS
/usr/share/doc/openvpn/README
/usr/share/doc/openvpn/README.auth-pam
/usr/share/doc/openvpn/README.down-root
/usr/share/doc/openvpn/contrib
/usr/share/doc/openvpn/contrib/OCSP_check
/usr/share/doc/openvpn/contrib/OCSP_check/OCSP_check.sh
/usr/share/doc/openvpn/contrib/README
/usr/share/doc/openvpn/contrib/multilevel-init.patch
/usr/share/doc/openvpn/contrib/openvpn-fwmarkroute-1.00
/usr/share/doc/openvpn/contrib/openvpn-fwmarkroute-1.00/README
/usr/share/doc/openvpn/contrib/openvpn-fwmarkroute-1.00/fwmarkroute.down
/usr/share/doc/openvpn/contrib/openvpn-fwmarkroute-1.00/fwmarkroute.up
/usr/share/doc/openvpn/contrib/pull-resolv-conf
/usr/share/doc/openvpn/contrib/pull-resolv-conf/client.down
/usr/share/doc/openvpn/contrib/pull-resolv-conf/client.up
/usr/share/doc/openvpn/sample
/usr/share/doc/openvpn/sample/Makefile
/usr/share/doc/openvpn/sample/Makefile.am
/usr/share/doc/openvpn/sample/Makefile.in
/usr/share/doc/openvpn/sample/sample-config-files
```

```
/usr/share/doc/openvpn/sample/sample-config-files/README
/usr/share/doc/openvpn/sample/sample-config-files/client.conf
/usr/share/doc/openvpn/sample/sample-config-files/firewall.sh
/usr/share/doc/openvpn/sample/sample-config-files/home.up
/usr/share/doc/openvpn/sample/sample-config-files/loopback-client
/usr/share/doc/openvpn/sample/sample-config-files/loopback-server
/usr/share/doc/openvpn/sample/sample-config-files/office.up
/usr/share/doc/openvpn/sample/sample-config-files/openvpn-shutdown.sh
/usr/share/doc/openvpn/sample/sample-config-files/openvpn-startup.sh
/usr/share/doc/openvpn/sample/sample-config-files/roadwarrior-client.conf
/usr/share/doc/openvpn/sample/sample-config-files/roadwarrior-server.conf
/usr/share/doc/openvpn/sample/sample-config-files/server.conf
/usr/share/doc/openvpn/sample/sample-config-files/static-home.conf
/usr/share/doc/openvpn/sample/sample-config-files/static-office.conf
/usr/share/doc/openvpn/sample/sample-config-files/tls-home.conf
/usr/share/doc/openvpn/sample/sample-config-files/tls-office.conf
/usr/share/doc/openvpn/sample/sample-config-files/xinetd-client-config
/usr/share/doc/openvpn/sample/sample-config-files/xinetd-server-config
/usr/share/doc/openvpn/sample/sample-keys
/usr/share/doc/openvpn/sample/sample-keys/.gitignore
/usr/share/doc/openvpn/sample/sample-keys/README
/usr/share/doc/openvpn/sample/sample-keys/ca.crt
/usr/share/doc/openvpn/sample/sample-keys/ca.key
/usr/share/doc/openvpn/sample/sample-keys/client-ec.crt
/usr/share/doc/openvpn/sample/sample-keys/client-ec.key
/usr/share/doc/openvpn/sample/sample-keys/client-pass.key
/usr/share/doc/openvpn/sample/sample-keys/client.crt
/usr/share/doc/openvpn/sample/sample-keys/client.key
/usr/share/doc/openvpn/sample/sample-keys/client.p12
/usr/share/doc/openvpn/sample/sample-keys/dh2048.pem
/usr/share/doc/openvpn/sample/sample-keys/gen-sample-keys.sh
/usr/share/doc/openvpn/sample/sample-keys/openssl.cnf
/usr/share/doc/openvpn/sample/sample-keys/server-ec.crt
/usr/share/doc/openvpn/sample/sample-keys/server-ec.key
/usr/share/doc/openvpn/sample/sample-keys/server.crt
/usr/share/doc/openvpn/sample/sample-keys/server.key
/usr/share/doc/openvpn/sample/sample-plugins
/usr/share/doc/openvpn/sample/sample-plugins/defer
/usr/share/doc/openvpn/sample/sample-plugins/defer/README
/usr/share/doc/openvpn/sample/sample-plugins/defer/build
/usr/share/doc/openvpn/sample/sample-plugins/defer/simple.c
/usr/share/doc/openvpn/sample/sample-plugins/defer/simple.def
/usr/share/doc/openvpn/sample/sample-plugins/defer/winbuild
/usr/share/doc/openvpn/sample/sample-plugins/log
/usr/share/doc/openvpn/sample/sample-plugins/log/build
/usr/share/doc/openvpn/sample/sample-plugins/log/log.c
/usr/share/doc/openvpn/sample/sample-plugins/log/log_v3.c
/usr/share/doc/openvpn/sample/sample-plugins/log/winbuild
/usr/share/doc/openvpn/sample/sample-plugins/simple
/usr/share/doc/openvpn/sample/sample-plugins/simple/README
```

```
/usr/share/doc/openvpn/sample/sample-plugins/simple/build
/usr/share/doc/openvpn/sample/sample-plugins/simple/simple.c
/usr/share/doc/openvpn/sample/sample-plugins/simple/simple.def
/usr/share/doc/openvpn/sample/sample-plugins/simple/winbuild
/usr/share/doc/openvpn/sample/sample-scripts
/usr/share/doc/openvpn/sample/sample-scripts/auth-pam.pl
/usr/share/doc/openvpn/sample/sample-scripts/bridge-start
/usr/share/doc/openvpn/sample/sample-scripts/bridge-stop
/usr/share/doc/openvpn/sample/sample-scripts/ucn.pl
/usr/share/doc/openvpn/sample/sample-scripts/verify-cn
/usr/share/doc/openvpn/sample/sample-windows
/usr/share/doc/openvpn/sample/sample-windows/sample.ovpn
/usr/share/man/man8/openvpn.8.gz
/var/run/openvpn
```

```
[root@profeN2I ~]# cat /usr/lib/systemd/system/openvpn@.service
[Unit]
Description=OpenVPN Robust And Highly Flexible Tunneling Application On %I
After=network.target

[Service]
PrivateTmp=true
Type=forking
PIDFile=/var/run/openvpn/%i.pid
ExecStart=/usr/sbin/openvpn --daemon --writepid /var/run/openvpn/%i.pid --cd
/etc/openvpn/ --config %i.conf

[Install]
WantedBy=multi-user.target
```

```
[root@profeN2I ~]# cat /usr/lib/tmpfiles.d/openvpn.conf
D /var/run/openvpn 0710 root openvpn -
```

```
[root@hp01 ~]# tree /etc/openvpn/
/etc/openvpn/
├── keys
│   ├── ca.crt
│   ├── dh2048.pem
│   ├── server.crt
│   └── server.key
├── myserver.conf
├── roadwarrior-server.conf
└── sample.server.conf
```

Exemples VPN manuals (ordre openvpn)

Aquesta secció està extreta dels exemples del Man openvpn. Implementa un a un els quatre exemples de la documentació i man de OpenVPN:

- Host-2-Host no encriptat.
- Host-2-Host xifrat simètric (pre-shared key)
- Host-2-Host TLS (xifrat asimètric claus Pública/Secreta)
- Network-2-Network

Descripció del model que es treballa:

VPN Address Setup:

For purposes of our example, our two machines will be called *may.kg* and *june.kg*. If you are constructing a VPN over the internet, then replace may.kg and june.kg with the internet hostname or IP address that each machine will use to contact the other over the internet.

Now we will choose the tunnel endpoints. Tunnel endpoints are private IP addresses that only have meaning in the context of the VPN. Each machine will use the tunnel endpoint of the other machine to access it over the VPN. In our example, the tunnel endpoint for *may.kg* will be *10.4.0.1* and for *june.kg*, *10.4.0.2*.

Once the VPN is established, you have essentially created a secure alternate path between the two hosts which is addressed by using the tunnel endpoints. You can control which network traffic passes between the hosts (a) over the VPN or (b) independently of the VPN, by choosing whether to use (a) the VPN end-point address or (b) the public internet address, to access the remote host. For example if you are on may.kg and you wish to connect to june.kg via ssh without using the VPN (since ssh has its own built-in security) you would use the command ssh june.kg. However in the same scenario, you could also use the command telnet 10.4.0.2 to create a telnet session with june.kg over the VPN, that would use the VPN to secure the session rather than ssh.

You can use any address you wish for the tunnel endpoints but make sure that they are private addresses (such as those that begin with 10 or 192.168) and that they are not part of any existing subnet on the networks of either peer, unless you are bridging. If you use an address that is part of your local subnet for either of the tunnel endpoints, you will get a weird feedback loop

Exemple 1: Túnel Host to Host [no xifrat]

Consisteix en fer una VPN entre dos hosts (may.jk i june.jk), estiguin on estiguin en una mateixa LAN o separats per internet. Podeu usar dos ordinadors de l’aula, o dos de casa o un ordinador de casa i un a AWS EC2 (caldrà obrir els ports apropiats de VPN). Quan els dos ordinadors es comuniquen per la seva interfície pública el tràfic és pel canal normal, quan es comuniquen a través de les interfícies TUN el tràfic viatja pel túnel (que no està xifrat).

En aquest exercici simplement observem l'establiment d'un túnel entre dos hosts, però no és un túnel xifrat sinó en text pla. No aporta res a nivell de seguretat, però podem observar com s'estableixen els túnels.

Assegureu-vos de:

- Observar l'aparició de les noves interfícies de túnel.
- Observar les rutes.
- Fer ping, telnet, ssh i accedir a algun servei daytyme o echo de l'altre peer.

Example 1: A simple tunnel without security

On may:

```
openvpn --remote june.kg --dev tun1 --ifconfig 10.4.0.1 10.4.0.2 --verb 9
```

On june:

```
openvpn --remote may.kg --dev tun1 --ifconfig 10.4.0.2 10.4.0.1 --verb 9
```

Now verify the tunnel is working by pinging across the tunnel.

On may:

```
ping 10.4.0.2
```

On june:

```
ping 10.4.0.1
```

The --verb 9 option will produce verbose output, similar to the tcpdump(8) program. Omit the --verb 9 option to have OpenVPN run quietly.

Exemple 2: Túnel Host to Host [pre-shared key]

Consisteix en fer una VPN entre dos hosts (may.jk i june.jk), estiguin on estiguin en una mateixa LAN o separats per internet. Podeu usar dos ordinadors de l'aula, o dos de casa o un ordinador de casa i un a AWS EC2 (caldrà obrir els ports apropiats de VPN). Quan els dos ordinadors es comuniquen per la seva interfície pública el tràfic és insegur. Quan es comuniquen entre ells utilitzant les interfcíes dels túnels el tràfic és segur.

En aquest exemple el mecanisme de xifrat és utilitzar com a *Key* un *pre-shared-secret*. És a dir, un secret compartit per els dos extrems de túnel. En configurar el túnel s'ha acordat un secret que saben les dues parts (per exemple jupiter) i que és la clau d'encriptació del tràfic segur. Així doncs, en aquest exemple NO s'utilitzen certificats digitals.

Assegureu-vos de:

- Observar l'aparició de les noves interfícies de túnel.
- Observar les rutes.
- Fer ping, telnet, ssh i accedir a algún servei daytime o echo de l'altre peer.
- Malauradament amb wireshark no veurem la diferència entre tràfic segur i insegur si analitzem el tràfic en un dels nodes extrem. Només tindria sentit su utilitzem

wireshark en un node router a mig camí (per on passi el tràfic cap a internet). Llavors podem veure quan el tràfic és insegur i quan és segur.

Example 2: A tunnel with static-key security (i.e. using a pre-shared secret)

First build a static key on may.

```
        openvpn −−genkey −−secret key
```

This command will build a random key file called key (in ascii format). Now copy key to june over a secure medium such as by using the scp(1) program.

On may:

```
        openvpn −−remote june.kg −−dev tun1 −−ifconfig 10.4.0.1 10.4.0.2 −−verb 5
        −−secret key
```

On june:

```
        openvpn −−remote may.kg −−dev tun1 −−ifconfig 10.4.0.2 10.4.0.1 −−verb 5
        −−secret key
```

Now verify the tunnel is working by pinging across the tunnel.

On may:

```
        ping 10.4.0.2
```

On june:

```
        ping 10.4.0.1
```

Exemple 3: Túnel Host to Host [TLS Public/Secret Key]

En aquest exemple es farà un túnel host to host igual que l'anterior però utilitzant com a mecanisme de xifrat del tràfic certificats digitals, és a dir, Clau pública / clau privada. El resultat funcional és el mateix que el de l'exemple anterior, però ara no cal cap secret compartit entre els dos peers. Així si, cal disposar del conjunt de certificats apropiats.

Com que el model de tràfic segur TLS/SSL està dissenyat per actuar amb els rols de client/serer aqui triarem arbitràriament un dels hosts de client i l'altre de server (by the face!). Per tant un tindrà els certificats de client i l'altre els certificats de server.

Atenció: Problema de certificats. Tal i com us descriu la documentació OpenVPN proporcionava d'exemple amb el seu paquet d'instal·lació un conjunt prefabricat de certificats i claus privades de client, servidor, CA i diffie-hellman. Advertia, però, que NO s'utilitzesis mai en producció, ja que tothom tenia les claus privades i per tant el tràfic era del tot insegur.

Malauradament la naturalesa humana és tan idiota que al final els desenvolupadors de OpenVPN van decidir de NO incorporar aquests fitxers en les noves distribucions.

Així doncs, si volem practicar l'exemple podem buscar els certificats/claus antics en versions antigues de OpenVPN. Podem fer-ho consultant els que hi ha a la web de ASIX-M11 (espero que hi siguin), al repositori públic (se siente confinament!) o generant amb docker un fedora:24 i instal·lant-hi OpenVPN (una versió antiga).

Més endavant, un cop estudiat Certificats Digitals TLS/SSL fabricarem els nostres propis certificats de l'roganització edt i implementarem OpenVPN amb certificats propis.

una última solució és anar a un proveïdor de certificats tipus LetsEncrypt i obtenir-ne.

Assegureu-vos de:

- Observar l'aparició de les noves interfícies de túnel.
- Observar les rutes.
- Fer ping, telnet, ssh i accedir a algún servei daytime o echo de l'altre peer.

Example 3: A tunnel with full TLS-based security

For this test, we will designate *may* as the *TLS client* and *june* as the *TLS server*. Note that client or server designation only has meaning for the TLS subsystem. It has no bearing on OpenVPN's peer-to-peer, UDP-based communication model.

First, build a separate *certificate/key* pair for both may and june (see above where −−cert is discussed for more info). Then construct Diffie Hellman parameters (see above where −−dh is discussed for more info).

You can also use the included test files *client.crt*, *client.key*, *server.crt*, *server.key* and *ca.crt*. The .crt files are certificates/public-keys, the .key files are private keys, and ca.crt is a certification authority who has signed both client.crt and server.crt. For Diffie Hellman parameters you can use the included file dh1024.pem. Note that all client, server, and certificate authority certificates and keys included in the OpenVPN distribution are totally insecure and should be used for testing only.

On may:
```
openvpn −−remote june.kg −−dev tun1 −−ifconfig 10.4.0.1 10.4.0.2 −−tls−client
−−ca ca.crt −−cert client.crt −−key client.key −−reneg−sec 60 −−verb 5
```

On june:
```
openvpn −−remote may.kg −−dev tun1 −−ifconfig 10.4.0.2 10.4.0.1 −−tls−server
−−dh dh1024.pem −−ca ca.crt −−cert server.crt −−key server.key −−reneg−sec 60
−−verb 5
```

Now verify the tunnel is working by pinging across the tunnel.

On may:
```
ping 10.4.0.2
```
On june:
```
ping 10.4.0.1
```

Notice the −−reneg−sec 60 option we used above. That tells OpenVPN to renegotiate the data channel keys every minute. Since we used −−verb 5 above, you will see status information on each new key negotiation.

For production operations, a key renegotiation interval of 60 seconds is probably too frequent. Omit the −−reneg−sec 60 option to use OpenVPN's default key renegotiation interval of one hour.

Exemple 4: Túnel Network to Network

En aquest exemple es tracta de connectar dues xarxes separades a través de una connexió VPN. Imaginem que tenim una seu a Sydney (o a casa) i una altra a Anchorage (song) (o a AWS EC2) i voleu que el tràfic entre aquestes dues xarxes sigui segur a través de una VPN.

Aconseguir-ho és ben senzill, de fet des del punt de vista de OpenVPN és el mateix que l'exemple anterior host-2host. El router de Sydney i el router de Anchorage estableixen una VPN entre ells dos (exactament igual que en l'exercici anterior).

El fet de que la conexió sigui network-2-network es gestiona des de les ordres d'enrutament d e tràfic en cada un dels dos routers. Consisteix en dir des de Sydney que tota la seva xarxa local si vol enviar dades a la xarxa de Anchorage ho ha de fer per el túnel VPN (en lloc de per el gw de internet). Així doncs és una simple ordre ip route que convertirà el túnel a N2N. Des de l'altre extrem cal fer el mateix, establir la ruta que envia del tràfic des de Anchorage cap a Sydney a través del túnel.

En resum, el mateix exercici que abans però afegint-hi ordres d'enrutament de les xarxes.

Per practicar aquest exercici podem plantejar-nos l'escenari següent:

- Un host que farà de router de una subseu formada per una subxarxa docker *Sydney* amb tres containers docker per exemple amb *nethosts* (serveis echo, daytime, etc).
- Un altre host que farà de router de la subxarxa docker *Anchorage* amb tres containers docker amb serveis de xarxa.
- Es tracta d'enrutar tràfic d'una subxarxa a l'altra. Podeu validar-ho iniciant una sessió interactiva amb docker exec a un dels hosts de cada subxarxa per verificar la connectivitat a serveis de l'altra xarxa.
- Aquest model el podeu fer usant dos hosts de casa, dos AMIs de dins de AWS EC2 o unde casa i una AMI (però estigueu al cas de l'obertura de ports al firewall del AWS EC2).

Routing:

Assuming you can ping across the tunnel, the next step is to route a real subnet over the secure tunnel. Suppose that may and june have two network interfaces each, one connected to the internet, and the other to a private network. Our goal is to securely connect both private networks. We will assume that may's private subnet is 10.0.0.0/24 and june's is 10.0.1.0/24.

First, ensure that IP forwarding is enabled on both peers. On Linux, enable routing:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

and enable TUN packet forwarding through the firewall:

```
iptables −A FORWARD −i tun+ −j ACCEPT
```

On may:

```
route add −net 10.0.1.0 netmask 255.255.255.0 gw 10.4.0.2
```

On june:

```
route add −net 10.0.0.0 netmask 255.255.255.0 gw 10.4.0.1
```

Now any machine on the 10.0.0.0/24 subnet can access any machine on the 10.0.1.0/24 subnet over the secure tunnel (or vice versa).
In a production environment, you could put the route command(s) in a script and execute with the −−up option.

Servei OpenVPN

En aquest apartat treballarem amb el servei OpenVPN gestionable a través de l'ordre *systemctl* com un servei més del sistema. Podrem fer start, stop, etc per activar / desactivar el servei.

Està estructurat seguint el model client (o clients) servidor. Un host executarà arbitràriament el rol de servidor. Aquest host ha de tenir el fitxer de *configuració de servidor* i el *certificat digital de servidor* (amb els drets per a fer la tasca de servidor VPN). Un altre (o altres) host actuarà com a client VPN per establir un túnel entre el client i el servidor. Aquest host client ha de tenir el fitxer de *configuració client* i el *certificat digital de client*.

Amb aquesta configuració tenim un model host-2-host. Primerament es fa el *systemctl start del servidor*, que es queda escoltant connexions i a continuació els clients que volen s'hi connecten fent ells també el *systemctl start client*.

Pot haver-hi *múltiples clients*. Cada un d'ells amb la seva configuració i certificat propi (almenys en teoria). En aquest model tenim Nhosts-2-server, hi ha una VPN de cada host al server. Aquest és el servei que ofereixen els servidors VPN de internet que permeten als clients conectar-si i llavors ells enruten el tràfic cap a l'exterior proporcionant confidencialitat.

El model anteior de un servidor que accepta múltiples clients el podem imaginar com un model d'estrella on cada client conecta al server. Per exemple el servidor és al seu central i cada client és un empleat des de casa (confinat potser!).

Podem fer que la connexió també sigui *client-2-client* permetent que els clients es comuniquin també entre ells, sempre a través del servidor. Cada client estableix una connexió al servidor i el servidor permet / redirigeix el tràfic entre clients. Per poder implementar quest model cal:

❏ configurar el servidor per permetre tràfic client-client
❏ si els clients que utilitzem tenen el mateix certificat digital de client (és el que ens passarà fins que no en fabriquem de propis i diferents per a cada client) cal permetre certificats duplicats.

Tricks: Per treballar en aquest apartat es trobarem amb algunes dificultats de configuració que com sabem en informàtica depenen de les versions, el temps i les bruixes, per tant, paciència. Assegureu-vos en especial de verificar que:

- el tipus de cipher és el mateix a client i server
- la compressió (tipus i si sí o si no).
- si es permet o no client-client i si es permeten certificats client duplicats.
- no us feu un embolic amb la gestió dels noms de servei ab arguments, noms de servei que tenen el caràcter @ per indicar que reben arguments.

Assegureu-vos de consultar:

- el fitxer de configuració .service per defecte. La versió del servidor i la versió del client.
- el contingut del paquet OpenVPN.

Fitxers del servei i de configuració

```
[root@profeN2I ~]# cat /usr/lib/systemd/system/openvpn@.service
[Unit]
Description=OpenVPN Robust And Highly Flexible Tunneling Application On %I
After=network.target

[Service]
PrivateTmp=true
Type=forking
PIDFile=/var/run/openvpn/%i.pid
ExecStart=/usr/sbin/openvpn --daemon --writepid /var/run/openvpn/%i.pid --cd
/etc/openvpn/ --config %i.conf

[Install]
WantedBy=multi-user.target
```

```
/user/lib/systemd/system/docker.service
        --fixed-cidr=172.17.0.0/16
# systemctl daemon-reload
# systemctl restart docker.service
```

```
[root@hp01 ~]# tree /etc/openvpn/
/etc/openvpn/
├── keys
│   ├── ca.crt
│   ├── dh2048.pem
│   ├── server.crt
│   └── server.key
├── myserver.conf
├── roadwarrior-server.conf
└── sample.server.conf
```

```
[root@hp01 keys]# ll /usr/share/doc/openvpn/sample/sample-keys/
-rw-r--r-- 1 root root 2195 18 mai  2015 ca.crt
-rw-r--r-- 1 root root 3272 18 mai  2015 ca.key
-rw-r--r-- 1 root root 5934 18 mai  2015 client.crt
-rw-r--r-- 1 root root 4798 18 mai  2015 client-ec.crt
-rw-r--r-- 1 root root  237 18 mai  2015 client-ec.key
```

```
-rw-r--r-- 1 root root 1704 18 mai  2015 client.key
-rw-r--r-- 1 root root 4533 18 mai  2015 client.p12
-rw-r--r-- 1 root root 1766 18 mai  2015 client-pass.key
-rw-r--r-- 1 root root  424 18 mai  2015 dh2048.pem
-rw-r--r-- 1 root root 2883 17 jul  2015 gen-sample-keys.sh
-rw-r--r-- 1 root root 4312 18 mai  2015 openssl.cnf
-rw-r--r-- 1 root root  737 18 mai  2015 README
-rw-r--r-- 1 root root 6391 18 mai  2015 server.crt
-rw-r--r-- 1 root root 5254 18 mai  2015 server-ec.crt
-rw-r--r-- 1 root root  237 18 mai  2015 server-ec.key
-rw-r--r-- 1 root root 1704 18 mai  2015 server.key
```

```
[root@hp01 keys]# ll /usr/share/doc/openvpn/sample/sample-config-files/
-rw-r--r-- 1 root root  3441 17 jul  2015 client.conf
-rw-r--r-- 1 root root  3562 18 mai  2015 firewall.sh
-rw-r--r-- 1 root root    62 18 mai  2015 home.up
-rw-r--r-- 1 root root   642 17 jul  2015 loopback-client
-rw-r--r-- 1 root root   645 17 jul  2015 loopback-server
-rw-r--r-- 1 root root    62 18 mai  2015 office.up
-rw-r--r-- 1 root root    63 18 mai  2015 openvpn-shutdown.sh
-rw-r--r-- 1 root root   776 18 mai  2015 openvpn-startup.sh
-rw-r--r-- 1 root root   131 18 mai  2015 README
-rw-r--r-- 1 root root   820  4 ago  2015 roadwarrior-client.conf
-rw-r--r-- 1 root root  1498  4 ago  2015 roadwarrior-server.conf
-rw-r--r-- 1 root root 10441 17 jul  2015 server.conf
-rw-r--r-- 1 root root  1742 18 mai  2015 static-home.conf
-rw-r--r-- 1 root root  1688 18 mai  2015 static-office.conf
-rw-r--r-- 1 root root  1937 18 mai  2015 tls-home.conf
-rw-r--r-- 1 root root  1948 18 mai  2015 tls-office.conf
-rw-r--r-- 1 root root   199 18 mai  2015 xinetd-client-config
-rw-r--r-- 1 root root   989 18 mai  2015 xinetd-server-config
```

Podeu observar els fitxers d'exemple de configuració servidor a l'apèndix d'aquest document.

Implementació Multi-client

Servidor Multi-client

Per implementar un servidor openvpn multi-client:

- Usarem TLS, per tal cal disposar a mà dels elements: ca.crt, server.crt, server.key i dh2048.pem.
- Generarem un fitxer de configuració del servidor usant com a model **server.conf** (és el mateix que sample-server.conf). Farem una còpia per personalitzar el servei i anomenarem el fitxer: **hisxserver.conf**.
- [cal que la ubicació dels fitxers sigui localitzable pel servei]

- Com que ‘tunejarem’ el servei el copiem a */etc/systemd/system* i caldrà fer el ***daemon-reload*** un cop modificat.
- Engegar el servei tenint en compte que és un template.

Generar els fitxers de treball:

```
# cp /etc/openvpn/sample-server.conf  hisxserver.conf

# cp /usr/lib/systemd/system/openvpn@service /etc/systemd/system/.
```

Tunejar el template per fer un echo de l’argument rebut (es veu al journald)

```
[root@hp01 keys]# cat /etc/systemd/system/openvpn@.service
[Unit]
Description=OpenVPN Robust And Highly Flexible Tunneling Application On %I hisx
After=network.target

[Service]
PrivateTmp=true
Type=forking
PIDFile=/var/run/openvpn/%i.pid
ExecStartPre=/usr/bin/echo servei %i %I
ExecStart=/usr/sbin/openvpn --daemon --writepid /var/run/openvpn/%i.pid --cd
/etc/openvpn/ --config %i.conf

[Install]
WantedBy=multi-user.target
```

Fitxer de configuració del servei personalitzat:

```
[root@hp01 keys]# cat /etc/openvpn/hisxserver.conf

#################################################
# Sample OpenVPN 2.0 config file for            #
# multi-client server.                          #
#                                               #
# This file is for the server side              #
# of a many-clients <-> one-server              #
# OpenVPN configuration.                        #
#                                               #
# OpenVPN also supports                         #
# single-machine <-> single-machine             #
# configurations (See the Examples page         #
# on the web site for more info).               #
#                                               #
# This config should work on Windows            #
# or Linux/BSD systems.  Remember on            #
# Windows to quote pathnames and use            #
# double backslashes, e.g.:                     #
# "C:\\Program Files\\OpenVPN\\config\\foo.key" #
#                                               #
# Comments are preceded with '#' or ';'        #
#################################################

# Which local IP address should OpenVPN
# listen on? (optional)
;local a.b.c.d

# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one.  You will need to
# open up this port on your firewall.
port 1194

# TCP or UDP server?
;proto tcp
proto udp

# "dev tun" will create a routed IP tunnel,
# "dev tap" will create an ethernet tunnel.
# Use "dev tap0" if you are ethernet bridging
# and have precreated a tap0 virtual interface
# and bridged it with your ethernet interface.
# If you want to control access policies
# over the VPN, you must create firewall
# rules for the the TUN/TAP interface.
# On non-Windows systems, you can give
# an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel if you
# have more than one.  On XP SP2 or higher,
# you may need to selectively disable the
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key).  Each client
# and the server must have their own cert and
# key file.  The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys.  Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca ca.crt
cert server.crt
key server.key  # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
#   openssl dhparam -out dh2048.pem 2048
dh dh2048.pem

# Network topology
# Should be subnet (addressing via IP)
# unless Windows clients v2.0.9 and lower have to
# be supported (then net30, i.e. a /30 per client)
# Defaults to net30 (not recommended)
;topology subnet

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0

# Maintain a record of client <-> virtual IP address
# associations in this file.  If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt

# Configure server mode for ethernet bridging.
# You must first use your OS's bridging capability
# to bridge the TAP interface with the ethernet
# NIC interface.  Then you must manually set the
# IP/netmask on the bridge interface, here we
# assume 10.8.0.4/255.255.255.0.  Finally we
# must set aside an IP range in this subnet
# (start=10.8.0.50 end=10.8.0.100) to allocate
# to connecting clients.  Leave this line commented
# out unless you are ethernet bridging.
```

```
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100

# Configure server mode for ethernet bridging
# using a DHCP-proxy, where clients talk
# to the OpenVPN server-side DHCP server
# to receive their IP address allocation
# and DNS server addresses.  You must first use
# your OS's bridging capability to bridge the TAP
# interface with the ethernet NIC interface.
# Note: this mode only works on clients (such as
# Windows), where the client-side TAP adapter is
# bound to a DHCP client.
;server-bridge

# Push routes to the client to allow it
# to reach other private subnets behind
# the server.  Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
;push "route 192.168.10.0 255.255.255.0"
;push "route 192.168.20.0 255.255.255.0"

# To assign specific IP addresses to specific
# clients or if a connecting client has a private
# subnet behind it that should also have VPN access,
# use the subdirectory "ccd" for client-specific
# configuration files (see man page for more info).

# EXAMPLE: Suppose the client
# having the certificate common name "Thelonious"
# also has a small subnet behind his connecting
# machine, such as 192.168.40.128/255.255.255.248.
# First, uncomment out these lines:
;client-config-dir ccd
;route 192.168.40.128 255.255.255.248
# Then create a file ccd/Thelonious with this line:
#   iroute 192.168.40.128 255.255.255.248
# This will allow Thelonious' private subnet to
# access the VPN.  This example will only work
# if you are routing, not bridging, i.e. you are
# using "dev tun" and "server" directives.

# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
# Then add this line to ccd/Thelonious:
```

```
#   ifconfig-push 10.9.0.1 10.9.0.2

# Suppose that you want to enable different
# firewall access policies for different groups
# of clients.  There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
#       group, and firewall the TUN/TAP interface
#       for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
#       modify the firewall in response to access
#       from different clients.  See man
#       page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# or bridge the TUN/TAP interface to the internet
# in order for this to work properly).
;push "redirect-gateway def1 bypass-dhcp"

# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses.  CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
# The addresses below refer to the public
# DNS servers provided by opendns.com.
;push "dhcp-option DNS 208.67.222.222"
;push "dhcp-option DNS 208.67.220.220"

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

# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names.  This is recommended
# only for testing purposes.  For production use,
# each client should have its own certificate/key
# pair.
#
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn

# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120

# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
#   openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
;tls-auth ta.key 0 # This file is secret

# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
;cipher BF-CBC        # Blowfish (default)
;cipher AES-128-CBC   # AES
;cipher DES-EDE3-CBC  # Triple-DES

# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
comp-lzo

# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100

# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nobody
```

```
# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log

# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it.  Use one
# or the other (but not both).
;log      openvpn.log
;log-append  openvpn.log

# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 3

# Silence repeating messages.  At most 20
# sequential messages of the same message
# category will be output to the log.
;mute 20

# Notify the client that when the server restarts so it
# can automatically reconnect.
#explicit-exit-notify 1
```

Engegar el servei

```
[root@hp01 system]# systemctl daemon-reload

[root@hp01 system]# systemctl start  openvpn@hisxserver.service

[root@hp01 system]# systemctl status  openvpn@hisxserver.service
● openvpn@hisxserver.service - OpenVPN Robust And Highly Flexible Tunneling
Application On hisxserver hisx
   Loaded: loaded (/etc/systemd/system/openvpn@.service; disabled)
   Active: active (running) since dc 2017-03-22 17:46:25 CET; 5s ago
  Process: 5774 ExecStart=/usr/sbin/openvpn --daemon --writepid /var/run/openvpn/%i.pid
--cd /etc/openvpn/ --config %i.conf (code=exited, status=0/SUCCESS)
  Process: 5772 ExecStartPre=/usr/bin/echo servei %i %I (code=exited,
status=0/SUCCESS)
 Main PID: 5776 (openvpn)
   CGroup: /system.slice/system-openvpn.slice/openvpn@hisxserver.service
           └─5776 /usr/sbin/openvpn --daemon --writepid /var/run/openvpn/hisxserver.pid
--cd /etc/openvpn/ --config his...

mar 22 17:46:25 hp01 openvpn[5776]: do_ifconfig, tt->ipv6=0,
tt->did_ifconfig_ipv6_setup=0
mar 22 17:46:25 hp01 openvpn[5776]: /usr/sbin/ip link set dev tun0 up mtu 1500
mar 22 17:46:25 hp01 openvpn[5776]: /usr/sbin/ip addr add dev tun0 local 10.8.0.1 peer
10.8.0.2
mar 22 17:46:26 hp01 openvpn[5776]: /usr/sbin/ip route add 10.8.0.0/24 via 10.8.0.2
mar 22 17:46:26 hp01 openvpn[5776]: UDPv4 link local (bound): [undef]
mar 22 17:46:26 hp01 openvpn[5776]: UDPv4 link remote: [undef]
mar 22 17:46:26 hp01 openvpn[5776]: MULTI: multi_init called, r=256 v=256
mar 22 17:46:26 hp01 openvpn[5776]: IFCONFIG POOL: base=10.8.0.4 size=62, ipv6=0
mar 22 17:46:26 hp01 openvpn[5776]: IFCONFIG POOL LIST
mar 22 17:46:26 hp01 openvpn[5776]: Initialization Sequence Completed
```

Observar el journalct -f o indicat la unit:

```
# journalctl -u openvpn@hisixserver
mar 22 17:44:56 hp01 systemd[1]: Unit openvpn@hisxserver.service entered failed state.
zmar 22 17:44:56 hp01 systemd[1]: openvpn@hisxserver.service failed.
mar 22 17:46:25 hp01 echo[5772]: servei hisxserver hisxserver
mar 22 17:46:25 hp01 openvpn[5774]: OpenVPN 2.3.8 x86_64-redhat-linux-gnu [SSL
(OpenSSL)] [LZO] [EPOLL] [PKCS11] [MH] [IPv6] built on Aug  4 2015
mar 22 17:46:25 hp01 openvpn[5774]: library versions: OpenSSL 1.0.1k-fips 8 Jan 2015,
LZO 2.08
mar 22 17:46:25 hp01 openvpn[5776]: NOTE: your local LAN uses the extremely common
subnet address 192.168.0.x or 192.168.1.x.  Be aware that this might create routing
conflicts if you connect to the VPN server from public locations such as internet cafes that
use the same subnet.
mar 22 17:46:25 hp01 openvpn[5776]: Diffie-Hellman initialized with 2048 bit key
mar 22 17:46:25 hp01 openvpn[5776]: WARNING: file 'server.key' is group or others
accessible
mar 22 17:46:25 hp01 openvpn[5776]: Socket Buffers: R=[212992->131072]
S=[212992->131072]
mar 22 17:46:25 hp01 openvpn[5776]: ROUTE_GATEWAY 192.168.1.1/255.255.255.0
IFACE=wls1 HWADDR=00:26:c7:08:4d:18
mar 22 17:46:25 hp01 openvpn[5776]: TUN/TAP device tun0 opened
mar 22 17:46:25 hp01 openvpn[5776]: TUN/TAP TX queue length set to 100
mar 22 17:46:25 hp01 openvpn[5776]: do_ifconfig, tt->ipv6=0,
tt->did_ifconfig_ipv6_setup=0
mar 22 17:46:25 hp01 openvpn[5776]: /usr/sbin/ip link set dev tun0 up mtu 1500
mar 22 17:46:25 hp01 NetworkManager[587]: <info>  (tun0): carrier is OFF
mar 22 17:46:25 hp01 NetworkManager[587]: <info>  (tun0): new Tun device (driver:
'unknown' ifindex: 7)
mar 22 17:46:25 hp01 NetworkManager[587]: <info>  (tun0): exported as
/org/freedesktop/NetworkManager/Devices/6
mar 22 17:46:25 hp01 openvpn[5776]: /usr/sbin/ip addr add dev tun0 local 10.8.0.1 peer
10.8.0.2
mar 22 17:46:25 hp01 NetworkManager[587]: <info>  (tun0): link connected
mar 22 17:46:26 hp01 openvpn[5776]: /usr/sbin/ip route add 10.8.0.0/24 via 10.8.0.2
mar 22 17:46:26 hp01 openvpn[5776]: UDPv4 link local (bound): [undef]
mar 22 17:46:26 hp01 openvpn[5776]: UDPv4 link remote: [undef]
mar 22 17:46:26 hp01 openvpn[5776]: MULTI: multi_init called, r=256 v=256
mar 22 17:46:26 hp01 openvpn[5776]: IFCONFIG POOL: base=10.8.0.4 size=62, ipv6=0
mar 22 17:46:26 hp01 openvpn[5776]: IFCONFIG POOL LIST
mar 22 17:46:26 hp01 openvpn[5776]: Initialization Sequence Completed
```

```
# ip a tun0
Command "tun0" is unknown, try "ip address help".
[root@hostedt openvpn]# ip a s tun0
392: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UNKNOWN group default qlen 100
    link/none
    inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::3edd:6fac:a7b1:f34/64 scope link flags 800
       valid_lft forever preferred_lft forever
```

```
# ip r
default via 192.168.1.1 dev enp0s25  proto static  metric 100
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
192.168.1.0/24 dev enp0s25  proto kernel  scope link  src 192.168.1.33  metric 100
```

Host Client

Per connectar un client openvpn a un servidor multi-client cal:

- Generar un servei openvpn template en el host client (com s'ha fet amb el servidor). Podem personalitzar-lo o usar l'estàndard.
- Crear un fitxer de configuració personalitzat.Per exemple usar **client.conf** (proporcionat als exemples) i editar-lo o fer-ne una còpia amb un nom personalitzat segons la finalitat del túnel vpn. Per exemple aquí usarem **hisxclient.conf**.

Crear un fitxer de configuració client personalitzat:

```
# cp /usr/share/doc/openvpn/sample/sample-config-files/client.conf /etc/openvpn/.
# cp /etc/openvpn/client.conf /etc/openvpn/hisxclient.conf
```

Editar el fitxer client per personalitzar-lo:

```
[root@hp01 keys]# cat /etc/openvpn/hisxclient.conf
```

```
##############################################
# Sample client-side OpenVPN 2.0 config file #
# for connecting to multi-client server.     #
#                                            #
# This configuration can be used by multiple #
# clients, however each client should have   #
# its own cert and key files.                #
#                                            #
# On Windows, you might want to rename this  #
# file so it has a .ovpn extension           #
##############################################

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client

# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel
# if you have more than one.  On XP SP2,
# you may need to disable the firewall
# for the TAP adapter.
;dev-node MyTap

# Are we connecting to a TCP or
# UDP server?  Use the same setting as
# on the server.
;proto tcp
proto udp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote i27 1194              #i27 és el host remot
;remote my-server-2 1194

# Choose a random host from the remote
# list for load-balancing.  Otherwise
# try hosts in the order specified.
;remote-random
```

```
# Keep trying indefinitely to resolve the
# host name of the OpenVPN server.  Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
;user nobody
;group nobody

# Try to preserve some state across restarts.
persist-key
persist-tun

# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
z# server, put the proxy server/IP and
# port number here.  See the man page
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]

# Wireless networks often produce a lot
# of duplicate packets.  Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.
ca ca.crt
cert client.crt
key client.key

# Verify server certificate by checking that the
# certicate has the correct key usage set.
# This is an important precaution to protect against
# a potential attack discussed here:
#  http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the keyUsage set to
#   digitalSignature, keyEncipherment
# and the extendedKeyUsage to
#   serverAuth
# EasyRSA can do this for you.
remote-cert-tls server

# If a tls-auth key is used on the server
# then every client must also have the key.
;tls-auth ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
;cipher x
cipher AES-256-CBC

# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
comp-lzo                        # Atenció o tots dos la tene o no la tenen!

# Set log file verbosity.
verb 3

# Silence repeating messages
;mute 20
```

Engegar el client:

```
[root@asus01 openvpn]# systemctl daemon-reload

[root@asus01 openvpn]# systemctl start openvpn@hisxclient.service

[root@asus01 openvpn]# systemctl -l status  openvpn@hisxclient.service
● openvpn@hisxclient.service - OpenVPN Robust And Highly Flexible Tunneling
Application On hisxclient
   Loaded: loaded (/etc/systemd/system/openvpn@.service; disabled)
   Active: active (running) since Wed 2017-03-22 17:58:01 CET; 6s ago
  Process: 2972 ExecStart=/usr/sbin/openvpn --daemon --writepid /var/run/openvpn/%i.pid
--cd /etc/openvpn/ --config %i.conf (code=exited, status=0/SUCCESS)
  Process: 2970 ExecStartPre=/usr/bin/echo client %i %I (code=exited,
status=0/SUCCESS)
 Main PID: 2974 (openvpn)
   CGroup: /system.slice/system-openvpn.slice/openvpn@hisxclient.service
           └─2974 /usr/sbin/openvpn --daemon --writepid /var/run/openvpn/hisxclient.pid --cd
/etc/openvpn/ --config hisxclient.conf

Mar 22 17:58:03 asus01 openvpn[2974]: OPTIONS IMPORT: --ifconfig/up options
modified
Mar 22 17:58:03 asus01 openvpn[2974]: OPTIONS IMPORT: route options modified
Mar 22 17:58:03 asus01 openvpn[2974]: ROUTE_GATEWAY 192.168.1.1/255.255.255.0
IFACE=wlp3s0 HWADDR=00:13:02:33:8e:4e
Mar 22 17:58:03 asus01 openvpn[2974]: TUN/TAP device tun0 opened
Mar 22 17:58:03 asus01 openvpn[2974]: TUN/TAP TX queue length set to 100
Mar 22 17:58:03 asus01 openvpn[2974]: do_ifconfig, tt->ipv6=0,
tt->did_ifconfig_ipv6_setup=0
Mar 22 17:58:03 asus01 openvpn[2974]: /usr/sbin/ip link set dev tun0 up mtu 1500
Mar 22 17:58:03 asus01 openvpn[2974]: /usr/sbin/ip addr add dev tun0 local 10.8.0.6 peer
10.8.0.5
Mar 22 17:58:03 asus01 openvpn[2974]: /usr/sbin/ip route add 10.8.0.1/32 via 10.8.0.5
Mar 22 17:58:03 asus01 openvpn[2974]: Initialization Sequence Completed
```

Atenció Tricks:

- Del fitxer de configuració *openvpn@hisxserver.service* assegurar-se de comentar la última línia: *#explicit-exit-notify 1*
- En el client *openvpn@hisxclient.service* afegir el tipus de cipher que s'ha definit en el servidor: *cipher AES-256-CBC*
- Atenció que client i server tinguin o no configurada la compressió (o tots dos si o tots dos no): *;comp-lzo*

Verificar connectivitat

Observar la ip de cada extrem del túnel (10.8.0.1 i 10.8.0.6) i fer ping entre els extrems. Si hi ha subxarxa-A i subxarxa-B establir rutes per accedir de un cantó a l'altre (per exemple amb xarxes docker) i fer test de connectivitat.
Des del client ping al server:

```
# ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=5.32 ms
```

Examinar el túnel

En el server:

```
[root@hp01 ~]# ip a s dev tun0
7: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UNKNOWN group default qlen 100
        link/none
        inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
        valid_lft forever preferred_lft forever

[root@hp01 ~]# ip r
default via 192.168.1.1 dev wls1  proto static  metric 1024
```

```
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
192.168.1.0/24 dev wls1  proto kernel  scope link  src 192.168.1.36
```

En el client:

```
[root@asus01 ~]# ip a s dev tun0
7: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UNKNOWN group default qlen 100
        link/none
        inet 10.8.0.6 peer 10.8.0.5/32 scope global tun0
        valid_lft forever preferred_lft forever

[root@asus01 ~]# ip r
default via 192.168.1.1 dev wlp3s0  proto static  metric 1024
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
192.168.1.0/24 dev wlp3s0  proto kernel  scope link  src 192.168.1.38
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1
```

Exemple multiclients que es veuen entre ells

Cas a)

- si engeguem un server i dos clients que usen el mateix certificat de client es genera un problema, en engegar-ne un aturem l'altre, comparteixen ídem adreça ip 10.8.0.6.

Cas b)

- configurem les directives per permetre múltiples repeticions del mateix certificat: *duplicate-cn*
- també configurem que els clients es vegin entre ells amb la directiva: *client-to-client*
- ara tenim un server 10.8.0.1 i dos clients 10.8.0.6 i 10.8.0.10. Entre els tres es veuen.

VPN Usant AWS

aula / AWS

Xarxa local de l'aula (docker) accedint a xarxa local (docker) de AWS

En el host de AWS

- Engegar el servei servidor OpenVPN
- Observar la interfície tun
- Crear/verificar la xarxa netB (amb adreces ip de subxarxa diferents de netA).
- Engegar containers de ldap i kerberos usant netB.
- Obrir el firewall del host AWS port 1194/udp

Engegar el servei OpnVPN server i verificar interfície tun:

```
[fedora@ip-172-31-20-75 openvpn]$ sudo systemctl start openvpn-server@server.service

[fedora@ip-172-31-20-75 openvpn]$ sudo systemctl status openvpn-server@server.service
● openvpn-server@server.service - OpenVPN service for server
  Loaded: loaded (/usr/lib/systemd/system/openvpn-server@.service; disabled; vendor preset: disabled)
  Active: active (running) since Thu 2019-03-21 15:26:31 UTC; 14s ago
        Docs: man:openvpn(8)
        https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
        https://community.openvpn.net/openvpn/wiki/HOWTO
 Main PID: 891 (openvpn)
  Status: "Initialization Sequence Completed"
        Tasks: 1 (limit: 4915)
  CGroup: /system.slice/system-openvpn\x2dserver.slice/openvpn-server@server.service
          └─891 /usr/sbin/openvpn --status /run/openvpn-server/status-server.log --status-version 2
--suppress-timestamps --cipher AES-256-GCM --ncp-

Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: Socket Buffers:
R=[212992->212992] S=[212992->212992]
Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: UDPv4 link local (bound):
[AF_INET][undef]:1194
Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: UDPv4 link remote:
[AF_UNSPEC]
Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: MULTI: multi_init called,
r=256 v=256
Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: IFCONFIG POOL:
base=10.8.0.4 size=62, ipv6=0
Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: ifconfig_pool_read(),
in='Test-Client,10.8.0.4', TODO: IPv6
Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: succeeded ->
ifconfig_pool_set()
Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: IFCONFIG POOL LIST
Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: Test-Client,10.8.0.4
Mar 21 15:26:31 ip-172-31-20-75.eu-west-2.compute.internal openvpn[891]: Initialization Sequence
Completed

[fedora@ip-172-31-20-75 openvpn]$ ip address show  tun0
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN
group default qlen 100
        link/none
        inet 10.8.0.1 peer 10.8.0.2/32 scope global tun0
        valid_lft forever preferred_lft forever
        inet6 fe80::2f34:7fd8:8fde:4058/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
```

Crear/verificar netB i engegar containers:

```
[fedora@ip-172-31-20-75 openvpn]$ docker network create netB
5eba7532d8664935dad215f4e5da7198d6bdd1ad5191f0bfd5cfd83b7efb1c38

[fedora@ip-172-31-20-75 openvpn]$ docker network inspect netB
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"

[fedora@ip-172-31-20-75 openvpn]$ docker run --rm --name ldap -h ldap --network netB -d
edtasixm06/ldapserver:18group
95bc9bd9b6a3b98112f488cdc11cd8d82976a395fd272e40ad5c1467189500a4

[fedora@ip-172-31-20-75 openvpn]$ docker run --rm --name kserver -h kserver --network netB -d
edtasixm11/k18:kserver
d0937b2781af905f9026c96d72447ed4f031acc546783129c8f390b74399679f

[fedora@ip-172-31-20-75 openvpn]$ docker ps
CONTAINER ID   IMAGE                      COMMAND                   CREATED            STATUS
PORTS          NAMES
d0937b2781af   edtasixm11/k18:kserver            "/opt/docker/startup…"   10 seconds ago     Up 7
seconds                  kserver
95bc9bd9b6a3   edtasixm06/ldapserver:18group   "/opt/docker/startup…"   37 seconds ago     Up 35
seconds        389/tcp         ldap

[fedora@ip-172-31-20-75 openvpn]$ docker network inspect netB
          "Name": "ldap",
          "IPv4Address": "172.18.0.2/16",
          "Name": "kserver",
          "IPv4Address": "172.18.0.3/16",

[fedora@ip-172-31-20-75 openvpn]$ nmap 172.18.0.2
PORT     STATE SERVICE
389/tcp open  ldap

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
[fedora@ip-172-31-20-75 openvpn]$ nmap 172.18.0.3
PORT     STATE SERVICE
88/tcp  open  kerberos-sec
464/tcp open  kpasswd5
749/tcp open  kerberos-adm
```

Obrir el port 1194/udp al firewall

```
Custom UDP rule  1194  Anywhere Openvpn
```

En el host de l'aula

- Engegar el servei servidor OpenVPN (configurar adreça IP servidor)
- Observar la interfície tun
- Verificar connectivitat client/server del tunel
- Crear/verificar la xarxa netA (amb adreces ip de subxarxa diferents de netB).
- Engegar containers de ldap i kerberos usant netA.

Configurar adreça IP del servidor al client.conf:

```
/etc/openvpn/client/client.conf
remote 18.130.232.31 1194
```

Engegar el túnel:

```
[root@hostedt openvpn]# systemctl start openvpn@client.service

[root@hostedt openvpn]# systemctl status openvpn@client.service
● openvpn@client.service - OpenVPN Robust And Highly Flexible Tunneling Application On client
```

```
   Loaded: loaded (/etc/systemd/system/openvpn@.service; disabled; vendor preset: disabled)
   Active: active (running) since dj 2019-03-21 16:55:33 CET; 3s ago
  Process: 18187 ExecStart=/usr/sbin/openvpn --daemon --writepid /var/run/openvpn/%i.pid --cd
/etc/openvpn/ --config %i.conf (code=exited, status=0/SU
  Process: 18184 ExecStartPre=/usr/bin/echo servei %i %I (code=exited, status=0/SUCCESS)
 Main PID: 18189 (openvpn)
           Tasks: 1 (limit: 512)
   CGroup: /system.slice/system-openvpn.slice/openvpn@client.service
            └─18189 /usr/sbin/openvpn --daemon --writepid /var/run/openvpn/client.pid --cd /etc/openvpn/
--config client.conf

mar 21 16:55:33 hostedt openvpn[18189]: WARNING: file 'client.key' is group or others accessible
mar 21 16:55:33 hostedt openvpn[18189]: UDPv4 link local: [undef]
mar 21 16:55:33 hostedt openvpn[18189]: UDPv4 link remote: [AF_INET]18.130.232.31:1194
mar 21 16:55:33 hostedt openvpn[18189]: [Test-Server] Peer Connection Initiated with
[AF_INET]18.130.232.31:1194
mar 21 16:55:35 hostedt openvpn[18189]: TUN/TAP device tun0 opened
mar 21 16:55:35 hostedt openvpn[18189]: do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
mar 21 16:55:35 hostedt openvpn[18189]: /usr/sbin/ip link set dev tun0 up mtu 1500
mar 21 16:55:35 hostedt openvpn[18189]: /usr/sbin/ip addr add dev tun0 local 10.8.0.6 peer 10.8.0.5
mar 21 16:55:35 hostedt openvpn[18189]: WARNING: this configuration may cache passwords in memory
-- use the auth-nocache option to prevent this
mar 21 16:55:35 hostedt openvpn[18189]: Initialization Sequence Completed

[root@hostedt openvpn]# ip address show tun0
400: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN
group default qlen 100
           link/none
           inet 10.8.0.6 peer 10.8.0.5/32 scope global tun0
           valid_lft forever preferred_lft forever
           inet6 fe80::fae6:357b:d38b:a65e/64 scope link flags 800
           valid_lft forever preferred_lft forever
```

Verificar connectivitat client/server del tunel:

```
[root@hostedt openvpn]# ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=31.4 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=31.6 ms
```

```
[fedora@ip-172-31-20-75 openvpn]$ ping 10.8.0.6
PING 10.8.0.6 (10.8.0.6) 56(84) bytes of data.
64 bytes from 10.8.0.6: icmp_seq=1 ttl=64 time=32.3 ms
64 bytes from 10.8.0.6: icmp_seq=2 ttl=64 time=31.7 ms
```

Crear la subxara i engegar els containers:

```
[root@hostedt ~]# docker network create --subnet=172.19.0.0/16 netA
[root@hostedt ~]# docker network inspect netA
                      "Subnet": "172.19.0.0/16"

[root@hostedt ~]# docker run --rm --name nethost -h nethost --network netA -d edtasixm11/net18:nethost
9556cb820d4aa2442b3be57a281f885e1dd2e53d0794397876172 6de862e8d25

[root@hostedt ~]# nmap 172.19.0.2
PORT     STATE SERVICE
7/tcp    open  echo
13/tcp   open  daytime
19/tcp   open  chargen
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
110/tcp  open  pop3
143/tcp  open  imap
993/tcp  open  imaps
995/tcp  open  pop3s
2007/tcp open  dectalk
2013/tcp open  raid-am
```

```
2022/tcp open  down

[root@hostedt ~]# telnet 172.19.0.2 13
Trying 172.19.0.2...
Connected to 172.19.0.2.
Escape character is '^]'.
21 MAR 2019 16:05:39 UTC
Connection closed by foreign host.
```

Enrutament entre xarxes

- Afegir l'enrutament entre xarxes
- Activar forwarding als hosts
- Verificar la connectivitat

En el host de l'aula:

```
[root@hostedt ~]# ip route add 172.18.0.0/16 via 10.8.0.6

[root@hostedt ~]# ip route show
default via 192.168.1.1 dev enp0s25  proto static  metric 100
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
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1 linkdown
172.18.0.0/16 via 10.8.0.6 dev tun0
172.19.0.0/16 dev br-b776627b415e  proto kernel  scope link  src 172.19.0.1
192.168.1.0/24 dev enp0s25  proto kernel  scope link  src 192.168.1.33  metric 100
192.168.122.0/24 dev virbr0  proto kernel  scope link  src 192.168.122.1 linkdown

[root@hostedt ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Ara hi ha connectivitat d'anada fins al host AWS (router)

```
[root@hostedt ~]# ping 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=31.8 ms

[root@hostedt ~]# ping 172.18.0.1
PING 172.18.0.1 (172.18.0.1) 56(84) bytes of data.
64 bytes from 172.18.0.1: icmp_seq=1 ttl=64 time=32.2 ms

[root@hostedt ~]# ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2) 56(84) bytes of data.
^C
```

Configuracio de la ruta de tornada AWS / aula:

```
[fedora@ip-172-31-20-75 openvpn]$ sudo ip route add 172.19.0.0/16 via 10.8.0.2

[fedora@ip-172-31-20-75 openvpn]$ sudo ip r
default via 172.31.16.1 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
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
172.18.0.0/16 dev br-5eba7532d866 proto kernel scope link src 172.18.0.1
172.19.0.0/16 via 10.8.0.2 dev tun0
172.31.16.0/20 dev eth0 proto kernel scope link src 172.31.20.75

[fedora@ip-172-31-20-75 openvpn]$ sudo echo 1 > /proc/sys/net/ipv4/ip_forward
```

Verificar la connectivitat:

```
[fedora@ip-172-31-20-75 openvpn]$ ping 10.8.0.6
PING 10.8.0.6 (10.8.0.6) 56(84) bytes of data.
64 bytes from 10.8.0.6: icmp_seq=1 ttl=64 time=31.6 ms
```

```
[fedora@ip-172-31-20-75 openvpn]$ ping 172.19.0.1
PING 172.18.0.1 (172.19.0.1) 56(84) bytes of data.
64 bytes from 172.19.0.1: icmp_seq=1 ttl=64 time=0.033 ms

[fedora@ip-172-31-20-75 openvpn]$ ping 172.18.0.2
^C
```

Punyeta! Encara no va?

Observar que la connectivitat del host de l'aula simplement arriba a l'altre extrem del túnel, però no es propaga als hosts de la xarxa destí. Això és un problema de la configuració de ***iptables***, el firewall actual tant al host de l'aula com al de AWS tenen regles per *isolar* la xarxa *DOCKER* de l'exterior.

Podem veure les regles a un host i altre amb l'ordre:

```
iptables -L
```

Per arreglar-ho (atenció a les configuracions on obrim masses coses de iptables!) podem activar que es permet tot el tràfic *forward* de la interfície *tun*:

```
[root@hostedt ~]# iptables -A FORWARD -i tun+ -j ACCEPT
```

```
[fedora@ip-172-31-20-75 openvpn]$ sudo iptables -A FORWARD -i tun+ -j ACCEPT
```

Connectivitat establerta. Des del host de l'aula:

```
[root@hostedt ~]# ping 172.18.0.1
PING 172.18.0.1 (172.18.0.1) 56(84) bytes of data.
64 bytes from 172.18.0.1: icmp_seq=1 ttl=64 time=32.5 ms

[root@hostedt ~]# ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2) 56(84) bytes of data.
64 bytes from 172.18.0.2: icmp_seq=1 ttl=63 time=31.9 ms

[root@hostedt ~]# ping 172.18.0.3
PING 172.18.0.3 (172.18.0.3) 56(84) bytes of data.
64 bytes from 172.18.0.3: icmp_seq=1 ttl=63 time=32.1 ms

[root@hostedt ~]# nmap 172.18.0.2
PORT     STATE SERVICE
389/tcp open  ldap

[root@hostedt ~]# nmap 172.18.0.3
PORT     STATE SERVICE
88/tcp  open  kerberos-sec
464/tcp open  kpasswd5
749/tcp open  kerberos-adm
```

Des del host AWS:

```
pot ser que el tràfic output estigui filtrat, però no el forward
```

Utilització de la connectivitat entre xarxes

Des del host de l'aula:

```
[root@hostedt ~]# ldapsearch  -x -LLL -h 172.18.0.2:389 -b 'dc=edt,dc=org' dn
dn: dc=edt,dc=org
```

```
dn: ou=maquines,dc=edt,dc=org
…

[root@hostedt ~]# docker run --network netA -it edtasixm11/k18:khost

[root@39ab79b9040f docker]# cat /etc/hosts
172.18.0.3 kserver.edt.org kserver

[root@39ab79b9040f docker]# kinit pere
Password for pere@EDT.ORG:

[root@39ab79b9040f docker]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: pere@EDT.ORG
Valid starting       Expires              Service principal
03/21/19 16:49:23  03/22/19 16:49:23  krbtgt/EDT.ORG@EDT.ORG
```

Des del host AWS:

```
[fedora@ip-172-31-20-75 openvpn]$ docker run --rm --network netB -it fedora:27
/bin/bash

<pendent de configurar iptables>
```

Client / AWS EC2 Server / Client

Pràctica final

Crear una topologia amb dos clients locals i un servidor OpenVPN a AWS EC2. Els clients han d'establir un túnel VPN amb el servidor de AWS i s'han de poder comunicar entre ells a través del túnel.

Com a clients podeu usar dos hosts de casa o de l'aula. També podeu usar dos hosts de docker però que estiguin en xarxes diferents, que NO puguin tenir connectivitat entre ells si no és a través del túnel.

Et voilà!

Applet gràfic VPN

```
<underconstruction>
```

Eines d'internet per a tràfic obscur

```
<underconstruction>
```

Apèndix: Fitxers de configuració

Per defecte en la versió OpenVPN de fedora:24 hi ha el directori de samples amb keys i configs. Podeu observar a continuació el llistat total del fitxer d'exemple i després un de simplificat.

Fitxer sample de server.conf

```
[root@hp01 ~]# cat /etc/openvpn/sample.server.conf
#################################################
# Sample OpenVPN 2.0 config file for            #
# multi-client server.                          #
#                                               #
# This file is for the server side              #
# of a many-clients <-> one-server              #
# OpenVPN configuration.                        #
#                                               #
# OpenVPN also supports                         #
# single-machine <-> single-machine             #
# configurations (See the Examples page         #
# on the web site for more info).               #
#                                               #
# This config should work on Windows            #
# or Linux/BSD systems.  Remember on            #
# Windows to quote pathnames and use            #
# double backslashes, e.g.:                     #
# "C:\\Program Files\\OpenVPN\\config\\foo.key" #
#                                               #
# Comments are preceded with '#' or ';'         #
#################################################

# Which local IP address should OpenVPN
# listen on? (optional)
;local a.b.c.d

# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one.  You will need to
# open up this port on your firewall.
port 1194

# TCP or UDP server?
;proto tcp
proto udp
```

```
# "dev tun" will create a routed IP tunnel,
# "dev tap" will create an ethernet tunnel.
# Use "dev tap0" if you are ethernet bridging
# and have precreated a tap0 virtual interface
# and bridged it with your ethernet interface.
# If you want to control access policies
# over the VPN, you must create firewall
# rules for the the TUN/TAP interface.
# On non-Windows systems, you can give
# an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel if you
# have more than one.  On XP SP2 or higher,
# you may need to selectively disable the
# Windows firewall for the TAP adapter.
# Non-Windows systems usually don't need this.
;dev-node MyTap

# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key).  Each client
# and the server must have their own cert and
# key file.  The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys.  Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca ca.crt
cert server.crt
key server.key  # This file should be kept secret

# Diffie hellman parameters.
# Generate your own with:
#   openssl dhparam -out dh2048.pem 2048
dh dh2048.pem

# Network topology
# Should be subnet (addressing via IP)
# unless Windows clients v2.0.9 and lower have to
# be supported (then net30, i.e. a /30 per client)
# Defaults to net30 (not recommended)
;topology subnet

# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0

# Maintain a record of client <-> virtual IP address
# associations in this file.  If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt

# Configure server mode for ethernet bridging.
# You must first use your OS's bridging capability
# to bridge the TAP interface with the ethernet
# NIC interface.  Then you must manually set the
# IP/netmask on the bridge interface, here we
# assume 10.8.0.4/255.255.255.0.  Finally we
# must set aside an IP range in this subnet
# (start=10.8.0.50 end=10.8.0.100) to allocate
# to connecting clients.  Leave this line commented
# out unless you are ethernet bridging.
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100

# Configure server mode for ethernet bridging
# using a DHCP-proxy, where clients talk
# to the OpenVPN server-side DHCP server
# to receive their IP address allocation
# and DNS server addresses.  You must first use
# your OS's bridging capability to bridge the TAP
# interface with the ethernet NIC interface.
# Note: this mode only works on clients (such as
# Windows), where the client-side TAP adapter is
# bound to a DHCP client.
;server-bridge

# Push routes to the client to allow it
# to reach other private subnets behind
# the server.  Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
;push "route 192.168.10.0 255.255.255.0"
;push "route 192.168.20.0 255.255.255.0"

# To assign specific IP addresses to specific
# clients or if a connecting client has a private
# subnet behind it that should also have VPN access,
# use the subdirectory "ccd" for client-specific
# configuration files (see man page for more info).

# EXAMPLE: Suppose the client
# having the certificate common name "Thelonious"
# also has a small subnet behind his connecting
# machine, such as 192.168.40.128/255.255.255.248.
# First, uncomment out these lines:
;client-config-dir ccd
;route 192.168.40.128 255.255.255.248
# Then create a file ccd/Thelonious with this line:
#   iroute 192.168.40.128 255.255.255.248
# This will allow Thelonious' private subnet to
# access the VPN.  This example will only work
# if you are routing, not bridging, i.e. you are
# using "dev tun" and "server" directives.

# EXAMPLE: Suppose you want to give
# Thelonious a fixed VPN IP address of 10.9.0.1.
# First uncomment out these lines:
;client-config-dir ccd
;route 10.9.0.0 255.255.255.252
# Then add this line to ccd/Thelonious:
#   ifconfig-push 10.9.0.1 10.9.0.2

# Suppose that you want to enable different
# firewall access policies for different groups
# of clients.  There are two methods:
# (1) Run multiple OpenVPN daemons, one for each
#        group, and firewall the TUN/TAP interface
#        for each group/daemon appropriately.
# (2) (Advanced) Create a script to dynamically
#        modify the firewall in response to access
#        from different clients.  See man
#        page for more info on learn-address script.
;learn-address ./script

# If enabled, this directive will configure
# all clients to redirect their default
# network gateway through the VPN, causing
# all IP traffic such as web browsing and
# and DNS lookups to go through the VPN
# (The OpenVPN server machine may need to NAT
# or bridge the TUN/TAP interface to the internet
# in order for this to work properly).
;push "redirect-gateway def1 bypass-dhcp"

# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses.  CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
# The addresses below refer to the public
# DNS servers provided by opendns.com.
;push "dhcp-option DNS 208.67.222.222"
;push "dhcp-option DNS 208.67.220.220"

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

# Uncomment this directive if multiple clients
# might connect with the same certificate/key
# files or common names.  This is recommended
# only for testing purposes.  For production use,
# each client should have its own certificate/key
# pair.
#
# IF YOU HAVE NOT GENERATED INDIVIDUAL
# CERTIFICATE/KEY PAIRS FOR EACH CLIENT,
# EACH HAVING ITS OWN UNIQUE "COMMON NAME",
# UNCOMMENT THIS LINE OUT.
;duplicate-cn

# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120

# For extra security beyond that provided
# by SSL/TLS, create an "HMAC firewall"
# to help block DoS attacks and UDP port flooding.
#
# Generate with:
```

```
#   openvpn --genkey --secret ta.key
#
# The server and each client must have
# a copy of this key.
# The second parameter should be '0'
# on the server and '1' on the clients.
;tls-auth ta.key 0 # This file is secret

# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
;cipher BF-CBC          # Blowfish (default)
;cipher AES-128-CBC   # AES
;cipher DES-EDE3-CBC  # Triple-DES

# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
comp-lzo

# The maximum number of concurrently connected
# clients we want to allow.
;max-clients 100

# It's a good idea to reduce the OpenVPN
# daemon's privileges after initialization.
#
# You can uncomment this out on
# non-Windows systems.
;user nobody
;group nobody

# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun

# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log

# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it.  Use one
# or the other (but not both).
;log      openvpn.log
;log-append  openvpn.log

# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 3

# Silence repeating messages.  At most 20
# sequential messages of the same message
# category will be output to the log.
;mute 20
```

Fitxer d'exemple simplificat/personalitzat server.conf

```
[root@hp01 ~]# cat /etc/openvpn/myserver.conf
#############################################
# Sample OpenVPN config file for
# 2.0-style multi-client udp server
#
# Adapted from http://openvpn.sourceforge.net/20notes.html
#
# tun-style tunnel

port 1194
dev tun

# Use "local" to set the source address on multi-homed hosts
#local [IP address]

# TLS parms
tls-server
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/server.crt
key /etc/openvpn/keys/server.key
dh /etc/openvpn/keys/dh2048.pem

# Tell OpenVPN to be a multi-client udp server
mode server

# The server's virtual endpoints
ifconfig 10.8.0.1 10.8.0.2
```

```
# Pool of /30 subnets to be allocated to clients.
# When a client connects, an --ifconfig command
# will be automatically generated and pushed back to
# the client.
ifconfig-pool 10.8.0.4 10.8.0.255

# Push route to client to bind it to our local
# virtual endpoint.
push "route 10.8.0.1 255.255.255.255"

# Push any routes the client needs to get in
# to the local network.
push "route 192.168.0.0 255.255.255.0"

# Push DHCP options to Windows clients.
push "dhcp-option DOMAIN example.com"
push "dhcp-option DNS 192.168.0.1"
push "dhcp-option WINS 192.168.0.1"

# Client should attempt reconnection on link
# failure.
keepalive 10 60

# Delete client instances after some period
# of inactivity.
inactive 600

# Route the --ifconfig pool range into the
# OpenVPN server.
route 10.8.0.0 255.255.255.0

# The server doesn't need privileges
user openvpn
group openvpn

# Keep TUN devices and keys open across restarts.
persist-tun
persist-key

verb 4
```