

# VPN

## **Conceptes:**

Una VPN és una “Virtual Private Network”. Utilitzant un medi públic i obert, la VPN crea un túnel privat (“private”) entre un servidor/xarxa fiable i un client/xarxa extern, de manera que aquest sembli (“virtual”) que formi part de la mateixa xarxa privada que la del servidor/xarxa fiable. És a dir, ofereix seguretat i confidencialitat al tràfic d'informació sensible al llarg de xarxes desprotegides com ara Internet. A la VPN hi intervenen conceptes com autenticació i encriptació, que s'han de negociar entre les dues parts, i que assegurin la confidencialitat i integritat de les dades que viatgen pel túnel.

Una VPN es pot realitzar des d'un client (un portàtil, per exemple) fins un servidor corporatiu (el qual, opcionalment, pot donar accés a la resta de recursos de la seva LAN i pot permetre o no la connexió simultània de múltiples clients), o bé es pot realitzar entre dos dispositius iguals -normalment dos routers-, els quals tindran cadascun d'ells una LAN per darrera, (permetent en aquest cas un túnel segur entre dues xarxes com si fossin una sola).

En tots dos casos, la idea és generar en els dos sistemes dels extrems una tarja de xarxa virtual (que s'anomenarà *tunX*), la qual tindrà una IP privada pertanyent a una xarxa diferent de les xarxes a les que les eventuais tarjes de xarxa reals poguessin pertànyer en ambdós costats. Les tarjes *tunX* es comuniquen entre elles mitjançant un túnel segur mentre que les tarjes reals continuen mantenint en paral·lel la seva comunicació no segura.

Encara que es parli de "client" i "servidor", per a les VPNs no existeixen aquests conceptes, ja que ambdós extrems són iguals. No obstant, s'usarà aquesta terminologia perquè quan s'utilitzen certificats TLS sí que hi ha una màquina client i una altra servidora.

## Altres tipus de VPN

**IPSec** (capa 3): Representa un conjunt de protocols de comunicació dissenyats per autenticar i encriptar les dades transmeses per una VPN (oferint confidencialitat -és a dir, secret-, integritat -és a dir, protecció a la manipulació dels paquets- i assegurament de les comunicacions passades). IPSec fa servir diferents mètodes per oferir cadascuna d'aquestes característiques: per exemple, pot oferir AES per la confidencialitat, SHA-512 (HMAC) per la integritat, etc. En qualsevol cas, IPSec és dependent de la infraestructura de mòduls del kernel de tots els elements de capa 3 involucrats a la comunicació (dispositius finals i routers)

Tiene dos métodos de trabajo principales: el modo tunnel, donde todos los paquetes IP son encapsulados en un nuevo paquete y enviados a través del túnel siendo desempaquetados en el otro extremo (así se protegen las direcciones IP del emisor y receptor además de los metadatos del paquete); o el modo transporte, donde sólo la carga útil de la sección de datos es cifrada y encapsulada.

**PPTP** (capa 2): protocolo inseguro propuesto por Microsoft. Encapsula datagramas de cualquier protocolo de red en datagramas IP usando una versión extendida del protocolo GRE (permitiendo así que cualquier protocolo pueda ser enrutado a través de una red IP, como Internet). Normalmente funciona conectándose via PPP a un servidor RAS que ofrece el servicio PPTP. Para la autenticación, PPTP tiene dos opciones: CHAP ó PAP (texto plano). La conexión de control se realiza en el puerto Tcp 1723.

**L2TP** (capa 2): protocolo evolución de L2F propuesto por Cisco. Encapsula tramas PPP sobre cualquier medio, no necesariamente redes IP. En el caso de IP se usa Udp, puerto 1701. Permite el uso de conexiones simultáneas. Como no ofrece mecanismos de seguridad, para su uso deberá ser combinado con otros mecanismos generalmente de capa 3.

## OpenVPN (basat en SSL/TLS, de capa 7):

A continuació es presenten diferents receptes per construir túnels de diferents tipus entre una màquina "A" i una altra "B":

### \*Túnel sense seguretat:

\*En A: **sudo openvpn --remote *ip.real.B* --dev *tun0* --ifconfig *ip.virtual.A* *ip.virtual.B* [--verb 9] &**

\*En B: **sudo openvpn --remote *ip.real.A* --dev *tun0* --ifconfig *ip.virtual.B* *ip.virtual.A* [--verb 9] &**

\*Provar el túnel: **ping *ip.virtual.B*** (en A); **ping *ip.virtual.A*** (en B)

**NOTA:** Les IPs virtuals han de ser de tipus privat (normalment se solen escollir de la xarxa 10.0.0.0/8 si aquesta no és usada a cap LAN real de les possibles involucrades, però això no és pas cap norma. Les IPs reals, en canvi, poden ser perfectament IPs públiques. De fet, es podrien haver escrit, en comptes d'elles, els noms DNS corresponents.-

**NOTA:** El túnel entre A i B s'estableix obrint el port 1194(UDP) a cada extrem.

**NOTA:** És molt interessant observar com, en executar les comandes *openvpn...* internament es crea el dispositiu *tun0* mitjançant les comandes **ip link set dev *tun0* up && ip address add dev *tun0* local 10.0.0.1 peer 10.0.0.2** (això seria en A)

També es podria haver escrit tot el conjunt de paràmetres anteriors que defineixen el túnel dins d'un fitxer a cada extrem i indicar aquest a l'execució de la comanda *openvpn* amb el paràmetre **--config /ruta/arxiu** (o fins i tot escrivint **/ruta/arxiu** a seques si no s'especifica cap paràmetre més). Això és vàlid en general per qualsevol tipus de túnel que volguem construir, ja sigui amb seguretat o sense. Aquest arxiu (que sol ubicar-se dins de la carpeta */etc/openvpn*) hauria de tenir el següent contingut en aquest cas:

\*Fitxer en A:

```
dev tun0
remote ip.real.B
ifconfig ip.virtual.A ip.virtual.B
#comp-lzo      (si es vol usar compressió)
```

\*Fitxer en B:

```
dev tun0
remote ip.real.A
ifconfig ip.virtual.B ip.virtual.A
#comp-lzo      (si es vol usar compressió)
```

El més habitual, però, no és iniciar OpenVPN manualment com hem fet abans sinó fer-ho com un servei Systemd. Concretament, executant **systemctl start openvpn@nomFitxer.service**, **systemctl enable openvpn@nomFitxer.service**, etc. Fixeu-vos com s'indica el nom (sense extensió) del fitxer de configuració del túnel a gestionar després de l'arroba i abans del ".service".

### \*Túnel amb clau simètrica:

\*Generar una clau compartida (en A o B, és igual): **openvpn --genkey --secret *shared.key***

\*Copiar la clau a l'altre extrem (utilitzant un canal segur, com ara SSH)

\*En A: **sudo openvpn --remote *ip.real.B* --dev *tun0* --ifconfig *ip.virtual.A* *ip.virtual.B* --secret *shared.key* &**

\*En B: **sudo openvpn --remote *ip.real.A* --dev *tun0* --ifconfig *ip.virtual.B* *ip.virtual.A* --secret *shared.key* &**

\*Provar el túnel: **ping *ip.virtual.B*** (en A); **ping *ip.virtual.A*** (en B)

**NOTA:** La generació de la clau simètrica es podria haver fet mitjançant qualsevol altre software adient, com *openssl* o similar

Podem igualment utilitzar un fitxer de directives en comptes d'escriure tot el "xoriço" de paràmetres (un cop creada i copiada la clau simètrica, òbviament): Bàsicament, el que ens caldrà serà afegir la següent línia a l'arxiu de cada extrem:

```
secret /ruta/shared.key
```

**NOTA:** Normalment, la ruta on es guarda la clau compartida és */etc/openvpn* però no té perquè ser aquesta carpeta. En qualsevol cas, si no es vol indicar explícitament la ruta a la directiva anterior (i de fet, a qualsevol altra directiva que apuntin a fitxers) es pot afegir a la invocació de la comanda *openvpn* el paràmetre **--cd /ruta/carpeta/a/usar/per/defecte**

Aquest túnel, no obstant, té diferents inconvenients:

\*Només es poden realitzar connexions punt a punt (és a dir, un client<->un servidor)

\*La clau està disponible en text pla a cada extrem (a més de què caldrà haver-la intercanviat de forma segura)

\*El crackejament de la clau permet el desxifrat no només de les sessions futures sinó també de les passades

#### \*Túnel amb parell certificat/clau privada:

Les passes bàsiques són:

**0.-**Convertir-se en la nostra pròpia "Autoritat Certificadora" (CA) per tal de poder emetre certificats no autosignats. La màquina CA no cal que sigui el servidor, pot ser-ne una tercera màquina que faci exclusivament aquesta funció

En comptes d'usar directament OpenSSL, per fer això es pot fer servir un "wrapper" ofert pel propi projecte OpenVPN anomenat "easy-rsa" (disponible a través d'un paquet homònim o bé des de <https://github.com/OpenVPN/easy-rsa> ), el qual facilita aquest procés. Si optem per aquesta solució, les comandes concretes a utilitzar (sempre com a root<sup>a</sup>) són:

```
cd /usr/share/easy-rsa  
source ./vars  
./clean-all  
./build-ca
```

La primera comanda inicialitza les variables d'entorn necessàries per realitzar el procés sense problemes (en realitat, la comanda "vars" no és més que un script que es pot editar sense problemes; per exemple, es poden establir respostes per defecte a les preguntes que realitzaran les comandes anteriors si s'editen les directives KEY\_\* del final del fitxer, com ara KEY\_COUNTRY, KEY\_ORG, KEY\_EMAIL, KEY\_SIZE, KEY\_DIR...

La segona comanda esborra els possibles certificats i claus velles que pogueren existir de processos anteriors

La tercera comanda crea dins de la carpeta "/usr/share/easy-rsa/keys" dos fitxers anomenats "ca.crt" i "ca.key", corresponents al certificat de la CA i a la clau privada de la CA, respectivament, després d'haver-se contestat interactivament una sèrie de preguntes (la més important de les quals, la que demana el "Common Name", haurà de respondre's amb la IP de la màquina que farà de CA). Una comanda alternativa seria *./pktool --initca*

**1.-**Crear al servidor el parell certificat/clau privada d'ell mateix (suposarem que és la màquina A). Ara sí parlem de servidor perquè aquest serà capaç d'acceptar múltiples connexions de diferents clients, a diferència del cas anterior, on només es podien fer connexions punt a punt. Al certificat l'anomenarem "miserver.crt" i a la clau privada "miserver.key", per exemple.

```
./build-key-server miserver
```

Apareixeran una sèrie de preguntes que es podran respondre amb el que desitgem o deixar els valors per defecte. En qualsevol cas, és important indicar a la pregunta de "Common Name" la IP real verdadera (o el nom DNS corresponent) de la màquina servidora. Al final s'obtingran, dins de la carpeta "/usr/share/easy-rsa/keys", els fitxers "miserver.crt" i "miserver.key". Una comanda alternativa a l'anterior es *./pktool --server miserver*.

El fitxer "miserver.crt" està signat per la CA que és el nostre propi servidor. No obstant, si es desitja signar el certificat del servidor per una altra CA, la comanda anterior també genera el fitxer "miserver.csr", el qual és una petició de signatura de certificat que s'haurà d'enviar a la CA escollida per a què aquesta la signi i retorni el corresponent "miserver.crt".

**2.-**Crear al servidor diferents parells certificat/clau privada per cadascun dels clients (suposarem que són la màquina B, C,...) que volguem autoritzar la seva connexió al servidor. A la pregunta de "Common Name" haurem d'introduir la IP real del client en qüestió. Als certificats dels clients els anomenarem "client1.crt", "client2.crt", etc i a les claus privades "client1.key", "client2.key", etc

```
./build-key client1  
./build-key client2
```

Obtenim els fitxers "client1.crt" (certificat signat per la CA del propi servidor) i "client1.key" (la seva clau privada). Una comanda alternativa a l'anterior és `./pktool client1`. Si desitjéssim generar més certificats de clients en un altre moment, hauríem d'executar abans de `./build-key` la comanda `source ./vars`. Si es vol que els parells generats estiguin protegits per contrasenya es pot fer servir la comanda `./build-key-pass client1`

Hem de generar a més els paràmetres Diffie-Hellman necessaris per a què l'intercanvi secret de claus entre els dos extrems (que no han tingut contacte previ) es realitzi correctament. Amb la següent comanda obtindrem l'arxiu "dh2048.pem"

```
./build-dh
```

3.-Copiar (o moure) els fitxers "miserver.crt", "miserver.key", "ca.crt" i "dh2048.pem" a la carpeta /etc/openssh local del servidor ...

```
cp keys/ca.crt /etc/openssh  
cp keys/miserver.{key,crt} /etc/openssh  
cp keys/dh2048.pem /etc/openssh
```

...i copiar (o moure) a la carpeta /etc/openssh de cada client el seu parell certificat/clau privada corresponent més el fitxer "ca.crt" utilitzant un canal segur, com ara SSH, per exemple així:

```
scp keys/ca.crt usuariClient@ip.Client:/etc/openssh  
scp keys/client1.{key,crt} usuariClient@ip.Client:/etc/openssh
```

En qualsevol cas, el propietari de tots aquests fitxers ha de ser l'usuari "root" del sistema en qüestió

4.-Configurar el túnel al servidor amb un fitxer similar a aquest (l'anomenarem "/etc/openssh/servidor.conf")..:

```
dev tun0  
local ip.real.Servidor #El servidor només serà funcional a la tarja real associada a la IP indicada  
ca ca.crt  
cert miserver.crt  
key miserver.key #Aquest fitxer hauria de ser secret  
dh dh2048.pem  
user nobody #El servidor s'executarà amb privilegis de l'usuari i grup "nobody" ...  
group nogroup #...això es fa per seguretat perquè "nobody" és un usuari quasi sense permisos  
server 10.0.0.0 255.255.255.0 #Assigna als clients que es connectin una IP dins de la xarxa indicada  
#(a mode de "servidor DHCP"). El servidor s'autoassigna la primera IP  
#automàticament (és a dir, en aquest cas la "10.0.0.1"). En el cas d'usar  
#dispositiu tap en comptes de tun, la directiva és server-bridge  
client-to-client #Permet que els clients connectats al servidor es puguin veure entre sí  
persist-key #Augmenta la resistència del link (no es tornen a llegir les claus a cada inici del servidor)  
persist-tun #Augmenta la resistència del link (no es torna a regenerar tunX a cada inici del servidor)  
keepalive 10 120 #Emet un "ping" al client als 10s d'inactivitat i, si als 120s es continua sense  
# rebre res, es torna a obrir la connexió. Aquesta opció inclou les opcions "ping" i  
# "ping-restart". També està "ping-exit"
```

... i iniciar-lo mitjançant l'execució de **systemctl start openssh@servidor.service** (també es pot fer que s'iniciï automàticament a cada arranc de la màquina fent **systemctl enable openssh@servidor.service** o al contrari, **systemctl mask openssh@servidor.service**, etc )

5.-Configurar el túnel a cada client amb un fitxer similar a aquest (l'anomenarem "/etc/openssh/client1.conf")..:

```
dev tun0  
remote ip.real.Servidor 1194  
ca ca.crt  
cert client1.crt  
key client1.key #Aquest fitxer hauria de ser secret  
client #Equivalent a tls-client més pull
```

***persist-key***  
***persist-tun***

... i iniciar-lo mitjançant l'execució de **systemctl start openvpn@client1.service** . També es pot fer que s'iniciï automàticament a cada arranc de la màquina fent **systemctl enable openvpn@client1.service**.

**6.-Probar el túnel de la manera usual** (interessant veure el que mostra la comanda “traceroute”)

Altres arxius d'exemple es poden trobar a la carpeta `"/usr/share/doc/openvpn/examples/sample-config-files"`

Si es volgués treure l'accés a un client concret, es pot executar el següent al servidor (com a root):

***source /usr/share/openvpn/easy-rsa/vars***  
***/usr/share/easy-rsa/revoked-full client1***

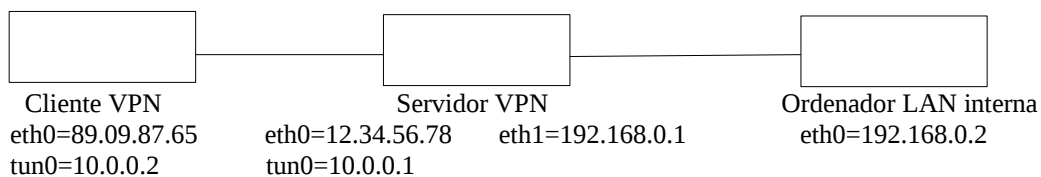
D'altra banda, es pot augmentar encara més la seguretat creant una clau TLS-AUTH addicional per a què totes les negociacions TLS verifiquin la integritat dels paquets, fent que qualsevol paquet UDP que no incorpori aquesta clau sigui bloquejat. D'aquesta manera podem protegir la comunicació d'atacs DoS, escaneig de ports, etc ja que el servidor tallarà la connexió en el moment d'alguna fallada. Per rear aquesta clau cal executar la comanda...

***openvpn --genkey --secret ta.key***

...i als fitxers de configuració del túnel, tant al del servidor com als dels clients, haurem d'escriure la línia ***tls-crypt ta.key***

\*Permitir el acceso a la LAN del servidor VPN:

Supongamos que tenemos esta configuración de red (se supone que 89.09.87.66 y 12.34.56.78 son ips públicas otorgadas por un ISP):



Para permitir a un cliente acceder a la red LAN del servidor , hay que hacer tres cosas:

**1.-En el servidor** hay que activar el IP Forwarding para que las peticiones que le lleguen por la interfaz tun0 (asociada a la interfaz externa, en este caso, eth0) le pasen internamente a la interfaz interna (en este caso, eth1) y de allí al ordenador de su LAN deseado:

***echo 1 > /proc/sys/net/ipv4/ip\_forward*** (o bien ***sysctl -w net.ipv4.ip\_forward=1***)

En ambos casos esta configuración es temporal hasta que la máquina servidora se reinicie. Si se desea hacer el forward permanente, hay que escribir la siguiente línea en el archivo `/etc/sysctl.conf`: ***net.ipv4.ip\_forward=1*** y activar dicho cambio haciendo ***sysctl -p /etc/sysctl.conf*** (o reiniciando la máquina)

**2.-En el fichero de configuración del servidor** hay que "publicitar" a los clientes que cuando adquieran la IP privada también podrán acceder a la red 192.168.0.0/24. Esto se consigue añadiendo una simple directiva tal como esta:

**push "route 192.168.0.0 255.255.255.0"**

Habiendo hecho los dos primeros pasos, ya tenemos el tráfico tunelado que va desde el cliente hasta el “3r ordenador” pero la comunicación todavía no es completa porque este 3r ordenador no es capaz de saber a dónde tiene que devolver las respuestas. Así pues, el tercer paso es:

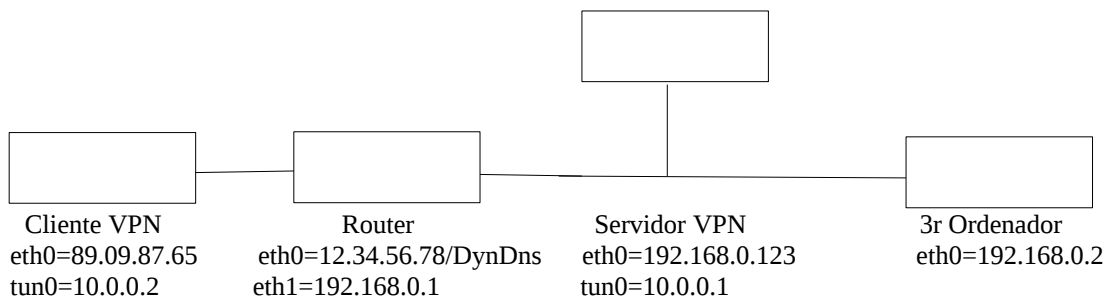
3.-En el 3º ordenador (de hecho, a cada uno de los ordenadores de la LAN interna del servidor) añadir una ruta dentro de su tabla de rutas que diga que todos los paquetes que vayan a parar al cliente VPN tengan que pasar por el servidor:

***ip route add 10.0.0.2/32 via 192.168.0.1***

O más general, para cualquier cliente VPN posible que se pudiera conectar:

***ip route add 10.0.0.0/24 via 192.168.0.1***

Lo más probable es que nos encontremos con que nuestro servidor VPN no está conectado directamente a la red Internet (es decir, él no tiene ip pública propiamente), ni tan siquiera tenga dos tarjetas de red: lo más habitual es tener un router, de esta manera:



En este caso, la configuración sería muy similar a la explicada, pero además deberíamos configurar el router para que hiciera un DNAT y así conseguir hacer que todas las peticiones del cliente VPN dirigidas al puerto 1194 se redirigieran al servidor VPN (en este caso, con la ip 192.168.0.123), de forma que el router sea en este caso un mero transmisor “correveydile”. Esto se puede hacer de forma muy sencilla mediante el panel de control web que ofrecen la mayoría de modelos de router

También existe la posibilidad de que el propio router ya incorpore dentro de sí mismo un servidor VPN, con lo que nos podríamos ahorrar la instalación y configuración del servidor OpenVPN en un PC, excepto para generar los certificados y claves, los cuales se deberán copiar en la configuración del router.