OpenVPN

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 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)

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:

Es necessita tenir un certificat de servidor que sigui reconegut pels clients. Per això caldrà crear primer una CA (és a dir, una clau privada de CA que signarà la clau pública de la CA generant així un certificat arrel que haurà de ser enviat als clients per a què acceptin el certificat del servidor) i segon el propi certificat del servidor en sí (és a dir, una clau pública signada amb la clau privada de la CA) a més de la clau privada del servidor. Noteu, però, que OpenVPN utilitza una estratègia d'autenticació bidireccional: no només el client ha de confiar en el certificat del servidor sinó que el servidor ha de confiar en el certificat del client. Això vol dir que caldrà generar tants certificats de clients com clients n'hi hagi, i tots ells hauran d'estar signats amb la mateixa clau privada de la CA (per a què tothom els reconegui).Les passes bàsiques a la pràctica 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" oferit 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[a]) 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 procesos 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 *./pkitool --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 conexions 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'obtindran, dins de la carpeta "/usr/share/easy-rsa/keys", els fitxers "miserver.crt" i "miserver.key". Una comanda alternativa a l'anterior es *./pkitool --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 *./pkitool 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*

2BIS.-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/openvpn local del servidor ...

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

...i copiar (o moure) a la carpeta /etc/openvpn 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/openvpn
scp keys/client1.{key,crt} usuariClient@ip.Client:/etc/openvpn

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/openvpn/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 conexió. Aquesta opció inclou les opcions "ping" i
                        #"ping-restart". També està "ping-exit"
```

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

5.-Configurar el túnel a cada client amb un fitxer similar a aquest (l'anomenarem **"/etc/openvpn/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'inicïi automàticament a cada arranc de la màquina fent *systemctl enable openvpn@client1.service*.

6.-Probar el túnel de la manera usual (interesant veure el que mostra la comanda "traceroute")

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

NOTA: 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/revoke-full cliente1
```

NOTA: 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 crear aquesta clau cal executar la comanda...:

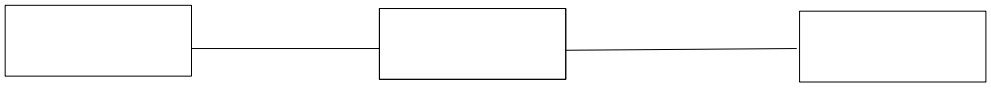
```
openvpn –genkey –secret ta.key
```

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

NOTA: El NetworkManager és capaç de gestionar conexions VPN sempre i quan s'hagi instal.lat els plugin "networkmanager-openvpn" i "networkmanager-openvpn-gnome" (també estan el "networkmanager-pptp" i el "networkmanager-vpnc" -per vpn de Cisco-, etc)

*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 3r 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

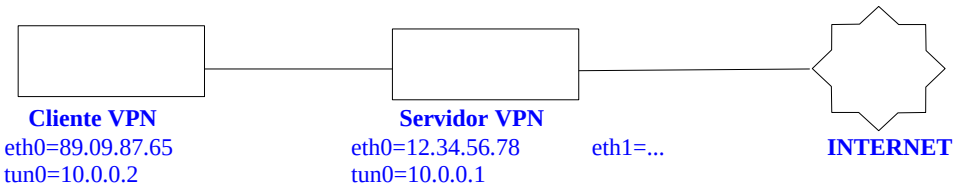
O más general, para cualquier cliente VPN posible que se pudiera conectar:

ip route add 10.0.0.0/24 via 192.168.0.1

NOTA: Las rutas establecidas de esta manera (con el comando "ip route", tanto en el punto 2 como en éste) son temporales. Para hacer que aparezcan fijas en el ordenador, existen varios métodos, pero tal vez el más sencillo es utilizar la directiva *up* de los ficheros de configuración de OpenVPN, la cual sirve para indicar un shell script que se ejecutará en el momento de activar la interfaz tunX, con lo que en este shell script podríamos escribir dicho comando "ip route"

*Permitir el acceso a Internet del servidor VPN:

Supongamos que tenemos esta configuración de red (se supone que 89.09.87.65 y 12.34.56.78 son ips públicas otorgadas por un ISP):



Para permitir a un cliente acceder a Internet a través 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 eth0, conectada al cliente) sean pasadas internamente a su interfaz conectada al resto de Internet (en este caso, eth1)

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

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)

2.-En el servidor hay que activar el SNAT. Esto se puede conseguir simplemente ejecutando:

iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

NOTA: Se supone que el tráfico FORWARD entre eth0 y eth1 está aceptado, y más concretamente las conexiones en estado ESTABLISHED,RELATED

3.-En el cliente hay que tener como puerta de enlace por defecto la IP real del servidor VPN. Esto se puede conseguir temporalmente ejecutando:

ip route add default via 12.34.56.78