Wireguard

WireGuard (https://www.wireguard.com) és una tecnologia VPN per a IPv4 i IPv6. Les seves principals característiques són:

* La majoria del seu codi resideix al nucli Linux (tot i que hi ha implementacions multiplataforma). Això el converteix en una solució molt ràpida en comparació amb altres alternatives

* Opera al sistema generant una interfície de xarxa virtual (sovint anomenada *wg0* però això no és obligatori) que pot ser gestionada mitjançant la comanda *ip* estàndard. De fet, per "connectar-s'hi" a la VPN aquesta interfície haurà de tenir assignada una adreça IP pròpia, independent de la IP real que tingui la interfície de xarxa real subjacent sobre la qual aquesta tarja virtual operarà.

* Utilitza un port UDP personalitzable per escoltar i establir connexions entre màquines de la VPN. Així doncs, només en el cas que a la xarxa hi hagi un tallafocs que talli l'ús del protocol UDP no es podrà implementar el túnel VPN

* Obliga a utilitzar només un conjunt reduït de primitives criptogràfiques modernes, no configurables. En concret, són els algoritmes de xifratge/autenticació/integritat/etc següents: Curve25519, HKDF, ChaCha20, Poly1305, BLAKE2 i SipHash24, a més del protocol Noise (http://www.noiseprotocol.org), tecnologia bàsica dissenyada per implementar-hi sobre ella protocols criptogràfics de xarxa més específics com és Wireguard però també l'usat a WhatsApp o a altres xarxes com I2P.

* Compta amb un esquema d'autenticació similar al de SSH, pel qual el servidor VPN i cada client tenen el seu propi parell de claus asimètric: autoritzar un client nou és tan senzill com afegir la seva clau pública al fitxer de configuració del servidor.

NOTA: WireGuard es pot configurar per utilitzar claus precompartides, però només com a capa mixta addicional de seguretat a sobre de les claus asimètriques existents.

Tot i que a Wireguard (a diferència d´altres sol·lucions VPN com OpenVPN) no existeix els conceptes de "servidor" i "client" VPN pròpiament dits (ja que s´usa el terme més horitzontal "peer"), sí que és veritat que, a la pràctica, es distingeixen dues configuracions diferents segons el rol que agafi cada extrem del túnel VPN: el "servidor" sol ser l´ordinador remot que fa de passarel·la o bé a Internet o bé a una xarxa LAN corporativa i el "client" (o clients, per què hi pot haver molts connectats al mateix servidor) sol ser l'ordinador on l'usuari en qüestió hi està treballant físicament.

NOTA: En el cas d' usar el servidor VPN per accedir a Internet, normalment aquest tindrà una IP pública directament accessible des de qualsevol lloc (tot i que també podria ser un ordinador domèstic, funcionant llavors, això sí, darrera d´un DNAT implementat en el nostre "router" de casa). En el cas d´usar el servidor VPN per accedir a una LAN corporativa, normalment aquest estarà ubicat darrera del "router" corporatiu, així que llavors caldrà realitzar sí o sí un DNAT en ell per tal de redireccionar el trànsit que li arribi per un determinat port a la IP local -la real, no la VPN!- del servidor intern i així fer-lo accessible des del client remot (a no ser que el servidor estigui implementat en el propi sistema del "router", que també podria ser).

1.-Instal·lació i configuració comuna (per client i servidor)

A Ubuntu caldrà instal·lar els següents paquets (dels repositoris oficials) per començar a treballar amb Wireguard: "**wireguard**" (el mòdul del kernel pròpiament dit), "**wireguard-tools**" (les comandes de terminal que ens permetran configurar-lo) i, opcionalment "**wireguard-dkms**" (per si volem que s´actualitzi el mòdul Wireguard automàticament en actualitzar-se el kernel subjacent). A Fedora només caldrà instal·lar el paquet "wireguard-tools" perquè el mòdul Wireguard ja ve incorporat dins del paquet "kernel-core" (i, per tant, s´actualitzarà amb ell). Per més sistemes, veieu https://www.wireguard.com/install. Un cop instal·lat/s, el que haurem de fer a ambdós extrems del túnel VPN (és a dir, al client -o clients- i al servidor) és el següent:

1.-Crear una clau privada: *wg genkey > priv.key*

2.-Crear, a partir de l'anterior, una clau pública: *wg pubkey < priv.key > pub.key*

NOTA: Les dues passes anteriors es podrien realitzar en un sol pas així: *wg genkey | tee priv.key | wg pubkey > pub.key*
NOTA: Les claus privades haurien de tenir permisos 700 i tenir com a propietari "root" i grup "root". D´altra banda, sol ser habitual guardar-les sota la carpeta "/etc/wireguard"

2.-Configuració del servidor

Cal fer els següents passos:

0.-Si volem que el servidor VPN serveixi com "trampolí" dels clients per accedir a Internet, cal:

*****Assegurar-se de què no només el port usat per Wireguard estigui obert pel tallafocs del sistema, sinó també les eventuals cadenes FORWARD usades per la interfície *wg0*
*****Assegurar-se de què SNAT està activat (amb la regla MASQUERADE adient) a la tarja real que està de cara a l´exterior ("Internet").
*Assegurar-se de què l´IP-Forwarding està activat

1.-Editar l´arxiu **"/etc/wireguard/wg0.conf"** de forma semblant a la mostrada a continuació (el propietari d'aquest fitxer ha de ser "root" obligatòriament, i els seus permisos han de ser 700):

```
[Interface]
PrivateKey = SERVER_PRIVATE_KEY
Address=192.168.3.1/24
ListenPort=12345
[Peer]
PublicKey = CLIENT_PUBLIC_KEY
AllowedIPs = CLIENT.VPN.IP/32
[Peer]
...
```

NOTA: La directiva *Address=* representa l´adreça IP (opcionalment amb màscara CIDR) a assignar a la interfície *wg0*. Recordem que les IPs assignades a les interfícies *wgX* han de ser de tipus privat però les IPs reals de les tarjes reals subjacents, en canvi, seran normalment IPs públiques. D´altra banda, si no s´indica cap directiva *ListenPort=*, Wireguard triarà un port d´escolta a l´atzar (això en el cas dels clients no és problema perquè són ells els qui comencen la comunicació contra el port definit d'un servidor, el qual només els ha de respondre)
NOTA: La secció *[Peer]* representa l´extrem remot del túnel VPN, el qual s'identifica per la seva clau pública. En el cas de la configuració del servidor, poden haver-hi tantes seccions *[Peer]* diferents com diferents clients es vulgui que s´hi puguin connectar (en el cas del client, normalment només hi haurà una sola secció *[Peer]*, corresponent al servidor remot)
NOTA: La directiva *AllowedIP=* , des del punt de vista del servidor, representa les adreces IP VPN (amb màscara i separades per comes si n´hi ha més d´una) que el peer client en qüestió podrà utilitzar com a adreces IP d'origen (i, on la resposta del servidor serà encaminada); si un paquet arriba al costat del servidor amb una IP d'origen que no es troba a la llista d'"AllowedIPs", s'eliminarà.

2.-Activar la tarja configurada a l´arxiu anterior (el nom de la qual ha de ser el mateix que el de l'arxiu però sense l´extensió ".conf") executant la comanda *sudo wg-quick up wg0* (per desactivar-la es podria fer *sudo wg-quick down wg0*).

NOTA: Una altra manera alternativa d´activar la tarja és posant en marxa un servei VPN associat, amb la comanda: *sudo systemctl {start|enable} wg-quick@wg0* Si ens fixem, de fet, en el contingut de la plantilla Systemd (*systemctl cat wg-quick@*) veurem que a les seves línies *ExecStart=* i *ExecStop=* s´executen les mateixes comandes indicades al paràgraf anterior. Fer "enable" pot ser interessant si volem que la interfície estigui automàticament disponible en iniciar el sistema.
NOTA: En realitat, la comanda *wg-quick* no deixa de ser un script Bash que embolcalla l´execució de dues comandes més fonamentals amb paràmetres concrets : *ip link/address/route/rule* i *wg* . Concretament, *wg-quick up* crea i activa la interfície "wg0" (amb *ip link*) i, a partir de llegir l´arxiu de configuració (amb *wg setconf*), li assigna la IP allà indicada (amb *ip address*) i les rutes adients (amb *ip route/rule*), a més d'executar possibles comandes "pre/post" (que de seguida veurem). Igualment, *wg-quick down* opcionalment guarda la configuració actual ubicada en RAM en el fitxer de configuració per una eventual posterior posada en marxa, elimina la interfície "wg0" (amb *ip link*) i executa opcionalment també possibles comandes "pre/post".

A partir de llavors, es podran veure tots els *peers* connectats en un moment donat amb la comanda *sudo wg show [wg0]* i *sudo wg showconf wg0*

NOTA: La comanda *wg* té més opcions interessants, com ara la ja mencionada *sudo wg setconf wg0 /ruta/fitxer.conf* (la qual llegeix el fitxer de configuració indicat i l´aplica a la interfície indicada...és la comanda que fa servir la comanda *wg-quick up* internament) o també *sudo wg syncconf wg0 /ruta/fitxer.conf* (similar a *wg setconf* però on llegeix només fa els canvis que siguin explícitament diferent entre l´arxiu indicat i la configuració actual, prèviament repassada, no trencant així les sessions VPN actuals -però essent un procés menys eficient-). De totes formes, cal destacar sobre tot la comanda *sudo wg set wg0 característica1 valor1 característica2 valor2 ...*, la qual permet modificar "al vol" qualsevol opció de la tarja *wg0* que haguem pogut tenir definida prèviament a l´arxiu de configuració.
NOTA: D´altra banda, l´script *wg-quick* també té altres opcions interessants, com ara *sudo wg-quick save wg0* (la qual guarda la configuració actual de la tarja indicada present a la memòria RAM -que, recordem, pot haver sigut modificada mitjançant *wg set*- en el seu fitxer de configuració en disc) o també *sudo wg-quick strip wg0* (la qual obté com a sortida la configuració de la tarja "wg0" però eliminant les directives de configuració pròpies de *wg-quick*, per a què quedin només les enteses per la comanda *wg* tal qual; veieu següent nota)
NOTA: A l´arxiu "/etc/wireguard/wg0.conf" es poden afegir unes poques directives més que no són enteses per la comanda *wg* sinó per l´script *wg-quick* Totes aquestes s´han d´indicar sota la secció *[Interface]* i en el cas del servidor en podem destacar les següents (per més informació, consulteu *man wg* i *man wg-quick*):

* Directiva *SaveConfig=true :* Indica a Wireguard que, en apagar-se la tarja Wireguard en qüestió, automàticament actualitzi el seu fitxer de configuració amb la configuració actual que hi hagi a la RAM. Això evita haver d´executar manualment la comanda *wg-quick save* Aquesta directiva és interessant si, per exemple, en un servidor VPN s´afegeixen nous clients VPN "al vol" (en comptes d'escriure'ls al fitxer), així, per exemple: *sudo wg set wg0 peer CLIENT_PUBLIC_KEY allowed-ips CLIENT.VPN.IP/32* o també així (suposant que al fitxer "nouclient.conf" hi apareixen com a mínim les línies *PublicKey=* i *AllowedIP=* adients sota la secció *[Peer]*): *cat "nouclient.conf" | sudo wg addconf wg0 /dev/fd/0*
* Directives *PreUp/PostUp/PreDown/PostDown=...* : Indica a Wireguard les comandes que haurà d´executar (via Bash) abans/després d´activar/desactivar la tarja en qüestió. Aquestes directives són interessants quan es vol configurar algun servidor DNS personalitzat per la tarja, o alguna regla de tallafocs específica, etc. Concretament, per exemple, per no tenir creada constantement la regla MASQUERADE a la configuració del tallafocs del sistema, una alternativa podria ser crear-la en el moment d'activar la tarja *wg0* i destruir-la en el moment de desactivar-la; això es pot aconseguir afegint dins de la secció *[Interface]* un parell de línies com:
PostUp = nft add rule inet filter forward iifname wg0 accept
PostUp = nft add rule inet nat postrouting oifname enp0s8 masquerade
PostDown = nft delete rule inet filter forward iifname wg0 accept
PostDown = nft delete rule inet nat postrouting oifname enp0s8 masquerade
Aquestes directives es poden repetir (s´executaran en l´ordre en què apareguin a l´arxiu de configuració) i es pot escriure el valor *%i* com a equivalent del nom de la tarja Wireguard en qüestió.

3.-Configuració del client

Cal fer els següents passos:

1.-Editar l´arxiu **"/etc/wireguard/wg0.conf"** de forma semblant a la mostrada a continuació (el propietari d´aquest fitxer ha de ser "root" obligatòriament, i els seus permisos han de ser 700):

```
[Interface]
PrivateKey = CLIENT_PRIVATE_KEY
Address=192.168.3.2/24
[Peer]
PublicKey = SERVER_PUBLIC_KEY
Endpoint = SERVER.REAL.IP:12345
AllowedIPs = 0.0.0.0/0
#PersistenKeepalive=25
```

NOTA: La directiva *Endpoint=* val l´adreça IP real (o nom DNS) + port on escolta el Wireguard del *peer* servidor Fixeu-vos que a la configuració del client no ha calgut indicar aquesta directiva (tot i que es podria haver indicat igualment, ja que Wireguard és realment un protocol punt-a-punt) perquè com que és el client qui comença la comunicació, el servidor, un cop rebi el primer paquet, ja sabrà a quina IP i port del client ha de contestar. De fet, és preferible que al client no aparegui aquesta directiva en el cas que sigui propens a saltar entre xarxes mòbils (i per tant, canvir l'adreça IP) ja que d'aquesta manera el servidor VPN podrà fer-ne seguiment del tràfic intercanviat de forma dinàmica sense talls de connexió (ja que el client estarà sempre identificat per la seva clau pública).
NOTA: La directiva *AllowedIPs=* , des del punt de vista del client, funciona com una taula d'encaminament, determinant quines IPs de destí són a les que s´enviaran els paquets xifrats via el túnel VPN (i des d´on s´espera rebre'n la resposta) En l´exemple anterior hem configurat el client per a què encamini tot el trànsit a través del túnel (això ho fem indicant el valor 0.0.0.0/0,::/0), però es podria reduir a una IP de xarxa específica (fent llavors que només el tràfic dirigit a les IPs d'aquesta xarxa s´encamini pel túnel) i/o a IPs concretes de màquines

concretes (es poden afegir vàries IPs separades per comes). Així doncs, bàsicament la secció *[Peer]* es pot interpretar així: tots els paquets destinats a *AllowedIPs=* es xifren amb *PublicKey=* i s'envien a *EndPoint=* (el qual pot ser fixe o dinàmic)
NOTA: De fet, cal entendre la directiva *AllowedIPs=* com una taula d'enrutament pels enviaments i una ACL per les rebudes de paquets: quan un extrem vol enviar un paquet a una IP, comprovarà el valor d'aquesta directiva i si la IP de destí hi apareix, serà enviat per la interfície Wireguard; quan un extrem rep un paquet, comprovarà el valor d'aquesta directiva de nou i si la IP d'origen no hi apareix, serà descartat.
NOTA: La directiva *PersistentKeepalive=* representa un interval de temps (en segons) que indica la freqüència amb què s'envia un paquet UDP buit al *peer* amb el propòsit de mantenir la connexió establerta (evitant que s'arribi a algun eventual timeout, per exemple definit pel tallafocs) Si s'estableix a 0, val "off" o no s´estableix explícitament, aquesta opció estarà desactivada. En general es recomanda no activar-la (per no gastar ample de banda ni energia) a no ser que el servidor sigui intern d'una LAN (és a dir, no estigui públicament accessible)
NOTA: A l´arxiu "/etc/wireguard/wg0.conf" es poden afegir unes poques directives més que no són enteses per la comanda *wg* sinó per l´script *wg-quick* Totes aquestes s´han d´indicar sota la secció *[Interface]* i en el cas dels clients en podem destacar la següent (per més informació, consulteu *man wg* i *man wg-quick*):

* Directiva *DNS=ip.serv.dns* : Indica l'adreça IP del servidor DNS que la interfície Wireguard farà servir (després de ser enrutada cap el servidor VPN segons el destí marcat a la directiva *AllowedIPs=*) per resoldre noms. Seria equivalent a executar manualment la comanda *sudo resolvectl dns wg0 ip.serv.dns* Es pot indicar vàries vegades per indicar més d'un servidor DNS possible a utilitzar (en ordre). En tot cas, si es fa servir aquesta directiva, llavors el valor de la directiva *EndPoint=* pot ser un nom DNS en lloc d'una adreça IP

2.-Activar la tarja configurada a l´arxiu anterior (el nom de la qual ha de ser el mateix que el de l'arxiu però sense l´extensió ".conf") executant la comanda *sudo wg-quick up wg0* (per desactivar-la es podria fer *sudo wg-quick down wg0*). O també, com ja hem vist, amb *sudo systemctl {start|enable} wg-quick@wg0*

NOTA: Poden coexistir diverses configuracions a cada client de manera que es puguin utilitzar configuracions diferents. Per això simplement cal crear diferents fitxers de configuració a la carpeta "/etc/wireguard" i utilitzar-los mitjançant l´execució de la comanda *sudo wg-quick up NOM*, on "NOM" fa referència al nom del fitxer de configuració sense la seva extensió ".conf"

A partir de llavors, es podran veure tots els peers connectats en un moment donat amb la comanda *sudo wg show [wg0]* i *sudo wg showconf wg0*

Un cop configurat el servidor i client (o clients!), ambdós extrems es podran fer "ping" mitjançant les seves respectives interfícies *wg0*. A més, si el tallafocs del servidor estigués correctament configurat, el/s client/s podran accedir a través d´ell al món exterior (o a la LAN corporativa, segons fos el cas).

EXERCICIS:

En el següent escenari, simularem l'accés "a Internet" a través d'un servidor VPN. Primer, però, primer haurem de muntar la infraestructura de xarxa necessària. Això consistirà concretament en dissenyar un laboratori de tres màquines virtuals: un client VPN que representarà el nostre ordinador de casa, un servidor VPN que imaginarem que està allotjat en algun núvol d'Internet que tinguem contractat i finalment, una tercera màquina que representarà qualsevol destí d'Internet (en el nostre cas, farem que sigui un simple servidor web).

0.-a) Arrenca una màquina virtual (Ubuntu o Fedora, és igual, però millor Server per no consumir massa recursos) i instal.la-hi el paquet "wireguard-tools" i, si calgués, també el paquet "nftables", "tshark" i "curl". Apaga-la i, en el cas de fer servir VirtualBox, clona-la (de forma "enllaçada" i reinicialitzant la MAC de la tarja de xarxa) un cop, i tot seguit, clona-la de nou un altre cop (també de forma "enllaçada i reinicialitzant la MAC de la tarja de xarxa). Hauràs de tenir finalment, doncs, tres màquines amb el mateix software instal·lat.

> NOTA: En el cas de fer servir IsardVDI, no existeix l'opció de clonar màquines, així que hauràs de crear tres màquines (això sí, a partir d'una mateixa plantilla, Ubuntu o Fedora preferiblement Server) i llavors instal·lar en cadascuna els paquets indicats (i apagar-les)

b) Edita la primera màquina virtual per a què tingui dues tarjes de xarxa (llegiu les notes següents per saber el seu mode de funcionament concret segons l'eina que estigueu fent servir, VirtualBox o IsardVDI); aquesta màquina serà el servidor VPN.

> NOTA: Si estàs fent servir VirtualBox, el mode de les dues tarjes de xarxes ha de ser la següent:
> * Tarja *enp0s3*: Xarxa interna que anomenarem "VPN"
> * Tarja *enp0s8*: Xarxa interna que anomenarem "Internet"
> NOTA: Si estàs fent servir IsardVDI, el mode de les dues tarjes de xarxes ha de ser la següent:
> * Tarja *enp1s0*: Xarxa interna "Personal1"
> * Tarja *enp2s0*: Xarxa interna "Personal2"

bII) Edita la segona màquina virtual per a què tingui la seva única tarja de xarxa en el mode de funcionament indicat a les notes següents segons l'eina que estigueu fent servir, VirtualBox o IsardVDI); aquesta màquina serà el client VPN.

> NOTA: Si estàs fent servir VirtualBox, el mode de la tarja de xarxa *enp0s3* ha de ser: Xarxa interna anomenada "VPN"
> NOTA: Si estàs fent servir IsardVDI, el mode de la tarja dexarxa *enp1s0* ha de ser: Xarxa interna "Personal1"

bIII) Edita la tercera màquina virtual per a què tingui la seva única tarja de xarxa en el mode de funcionament indicat a les notes següents segons l'eina que estigueu fent servir, VirtualBox o IsardVDI); aquesta màquina srepresentarà un servidor qualsevol d´Internet (per exemple HTTP).

> NOTA: Si estàs fent servir VirtualBox, el mode de la tarja de xarxa *enp0s3* ha de ser: Xarxa interna anomenada "Internet"
> NOTA: Si estàs fent servir IsardVDI, el mode de la tarja dexarxa *enp1s0* ha de ser: Xarxa interna "Personal2"

c) Arrenca el servidor VPN i assigna la IP 8.0.0.1/8 a la seva tarja *enp0s3* (o *enp1s0*) i la IP 16.16.0.1/16 a la seva tarja *enp0s8* (o *enp2s0*); simularem, en tot cas, que ambdues tarjes tenen una adreça pública. Això ho pots fer directament amb les comandes *sudo ip address add 8.0.0.1/8 dev enp0s3* i *sudo ip address add 16.16.0.1/16 dev enp0s8* per anar ràpid

d) Encara al servidor VPN, activa-hi l´"IP-Forward" per a què, quan calgui, el tràfic d'una tarja del servidor pugui passar internament a l'altra i viceversa (per defecte aquesta possibilitat no està activada). Per això, hauràs d'afegir la següent línia al final de l'arxiu "/etc/sysctl.d/99-sysctl.conf" (a Ubuntu ja ve escrita però caldrà descomentar-la; a Fedora cal escriure-la a mà), i tot seguit executa la comanda *sudo sysctl -p*:

> *net.ipv4.ip_forward=1*

> NOTA: Pots comprovar que l'"IP-forward" s'hagi activat observant si sortida de *sysctl -n net.ipv4.ip_forward* és 1

e) Arrenca el client VPN i assigna a la seva tarja enp0s3 la IP 8.0.0.2/8 (ho pots fer igualment de forma temporal amb *sudo ip address add 8.0.0.2/8 dev enp0s3*). Comprova tot seguit que ja es facin "ping" entre aquesta màquina i el servidor VPN

> NOTA: Aquestes IPs representen, en teoria, les IPs públiques de cada extrem a Internet. A la pràctica, la IP pública d´un client real no sol ser fixa, però això no hauria de ser inconvenient mentre la del servidor sí que ho sigui. I si no ho fos, tampoc passaria res mentre tingués un nom DNS que apuntés de forma actualitzada a l´adreça IP real en cada moment

f) Arrenca la tercera màquina (que anomenarem "servidor HTTP") i assigna a la seva tarja enp0s3 la IP 16.16.0.2/16 (ho pots fer igualment de forma temporal amb *sudo ip address add 16.16.0.2/16 dev enp0s3*). Comprova tot seguit que es facin "ping" entre aquesta màquina i el servidor VPN

g) Fes que la porta d'enllaç del client VPN sigui la IP de la tarja enp0s3 del servidor VPN (això ho pots fer directament amb la comanda *sudo ip route add default via 8.0.0.1 dev enp0s3*). Igualment, fes que la porta d'enllaç del "servidor HTTP" sigui la IP de la tarja enp0s8 del servidor VPN (això ho pots fer amb la comanda *sudo ip route add default via 16.16.0.1 dev enp0s3*) Comprova finalment que es facin "ping" entre aquesta màquina i el "servidor HTTP" (el qual, insistim, representarà qualsevol servidor d'Internet).

> NOTA: En el cas que el nostre "servidor HTTP" fos un servidor realment públic d'Internet, s'hi podria accedir perfectament des del client amb la configuració que tenim feta només fent que la tarja *enp0s8* del nostre servidor VPN estigués en mode "adaptador pont": d'aquesta forma s'hi podria fer "ping", de fet, a qualsevol adreça IP d'Internet (per exemple, a 8.8.8.8 o a 1.1.1.1, etc). No es podria fer resolució de noms, però, perquè, tot i tenir a la màquina client el servei *systemd-resolved* encès, aquest no hauria obtingut cap servidor DNS de referència per configurar, ja que no se n´ha indicat cap ni mitjançant eventuals línies *DNS=* en arxius ".network" o bé a través d´un inexistent servidor DHCP per la tarja en mode "xarxa interna" o bé fent ús de la directiva *DNS=* de l'arxiu "/etc/systemd/resolved.conf". Podríem sol·lucionar momentàniament aquest problema executant la comanda ***sudo resolvectl dns enp0s3 8.8.8.8*** (i comprovar-ho amb *resolvectl dns*)

h) Executa al servidor VPN la comanda *tshark -i enp0s8 -Y "icmp"* i executa al client la comanda *ping 16.16.0.2*. ¿Què veus? ¿I si mentre està funcionant el "ping" anterior executes al servidor en canvi la comanda *tshark -i enp0s3 -Y "icmp"*?

Tal com es pot veure al darrer apartat de l'exercici anterior, tant el tràfic que travessa la tarja enp0s3 com la tarja enp0s8 del servidor viatja en clar. A l'exercici següent aconseguirem implementar una VPN entre el client i el servidor de tal forma que el tràfic que arribi a la tarja enp0s3 estigui xifrat (una altra cosa és el tràfic que surti per enp0s8, la qual ja no forma part de la VPN). En tot cas, si hi hagués algun possible "espia" observant el tràfic entre el client i el servidor VPN (per exemple, algú connectat a la mateixa LAN que el client), no hi veuria res: només a partir del servidor VPN i més enllà hi hauria la possibilitat de veure tal qual la comunicació com és (i és per això que es recomana fer servir HTTPS i en general, qualsevol altre protocol que porti el xifrat incorporat, més enllà del que pugui implementar el canal que estigui fent servir)

1.-a) Al servidor VPN crea les seves claus pública i privada, tal com diu l´apartat "Configuració comuna" de la teoria. Igualment, al client VPN crea igualment les seves claus pública i privada.

b) Edita al servidor VPN l'arxiu "/etc/wireguard/wg0.conf" per a què tingui el següent contingut i tot seguit executa *sudo wg-quick up wg0* (o bé, ja que serà el servidor, es pot alternativament fer *sudo systemctl –now enable wg-quick@wg0)*. En tot cas, comprova amb *sudo wg show* que la configuració és correcta. També pots observar si apareix la nova tarja "wg0" a la sortida de la comanda *ip -c a* :

```
[Interface]
PrivateKey = SERVER_PRIVATE_KEY
Address=192.168.3.1/24
ListenPort=12345
[Peer]
PublicKey = CLIENT_PUBLIC_KEY
AllowedIPs = 192.168.3.2/32
```

> NOTA: Fixa´t que no hi ha estipulat cap sistema de transmissió de les claus públiques entre els extrems...Es pot fer de qualsevol manera "out of band": mitjançant email, *ncat*, *scp*, pendrive, etc, etc

c) Ara edita al client l'arxiu "/etc/wireguard/wg0.conf" per a què tingui el següent contingut i tot seguit executa *sudo wg-quick up wg0*. Comprova amb *sudo wg show* que la configuració és correcta. També pots observar si apareix la nova tarja "wg0" a la sortida de la comanda *ip -c a* :

[Interface]
PrivateKey = CLIENT_PRIVATE_KEY
Address=192.168.3.2/24
[Peer]
PublicKey = SERVER_PUBLIC_KEY
Endpoint = SERVER_REAL_IP:12345
AllowedIPs = 0.0.0.0/0

d) Comprova que la tarja wg0 del client pot fer "ping" a la del servidor executant (a la màquina client) la comanda *ping -I 192.168.3.2 192.168.3.1*.

dII) Comprova que també pugui fer "ping" al "servidor HTTP" que representa Internet executant la comanda *ping -I 192.168.3.2 16.16.0.2*

NOTA: En el cas de tenir accessible algun servidor DNS (no és el cas perquè el laboratori que hem confeccionat només conté màquines amb tarjes en mode "xarxa interna" -així que els servidors DNS públics d'Internet no estan disponibles- i tampoc no hi hem implementat cap servidor DNS propi), des del client VPN es podria fer "ping" igualment a noms de domini, sempre i quan, això sí, en aquest client s'hi hagués indicat el/s servidor/s DNS concrets a fer servir per la interfície *wg0*, o bé explícitament executant la comanda *sudo resolvectl dns wg0 8.8.8.8* o bé afegint la directiva ***DNS=*** corresponent sota la secció *[Interface]* de l'arxiu "wg0.conf" del client.

dIII) Comprova, executant *ping 16.16.0.2* , que, de fet, no cal ni indicar la interfície d'origen que sigui "wg0" perquè ja ho és per defecte (gràcies a les regles d'enrutament introduïdes per *wg-quick* a partir de l´indicat a la directiva *AllowedIPs=*)

e) Executa a la màquina servidor HTTP", ara sí, la comanda *python -m http.server 4321* Aquesta comanda posa en marxa un servidor HTTP bàsic en el port indicat (per defecte, si no s'especifica, serà el 8000). Tot seguit, al client VPN, executa la comanda *curl http://16.16.0.2:4321* per accedir a la pàgina inicial d´aquest servidor HTTP. No hauries de tenir cap problema en accedir a aquesta pàgina, però el més interessant aquí és veure quina és la IP recollida, com a origen de les peticions HTTP: ¿quina adreça IP veus? Per què? Si no haguéssim implementat la VPN, només la redirecció de paquets, ¿es veuria alguna diferència?

NOTA: Si estiguéssim fent servir l'Apache2 com a servidor HTTP, aquesta dada (la IP origen de les peticions), entre unes quantes més, es guardarà dins de l'arxiu "/var/log/apache2/access.log" (a Ubuntu) o "/var/log/httpd/access_log" (a Fedora)

f) Executa al servidor VPN la comanda *tshark -i wg0 -Y "icmp or http"* i executa al client la comanda *ping 16.16.0.2* i també *curl http://16.16.0.2:4321*. ¿Què veus? ¿Què veus en canvi, si mentre està funcionant les comandes anteriors executes al servidor la comanda *tshark -i enp0s3 -Y "wg"* ¿Per què?

NOTA: Per més informació sobre el reconeixement del protocol Wireguard per part del Wireshark, veieu https://wiki.wireshark.org/WireGuard

g) Executa al client VPN la comanda *ip route get 1.1.1.1* Aquesta comanda indica quina és la ruta que seguirà el paquet per intentar arribar a la IP de destí indicada. ¿Com interpretes el resultat obtingut?

NOTA: El fet d'indicar, mitjançant la directiva *AllowedIPs=*, que, com fem en aquest cas, tot el trànsit generat pel client volem que vagi a parar al servidor VPN (i les seves corresponents respostes provinguin del mateix lloc) fa que a partir d´ara la porta d'enllaç per defecte definida al client VPN deixi de tenir cap importància, ja que a partir d´ara el client delegarà aquesta tasca en la porta d´enllaç que tingui definida el servidor VPN remot. Per tant, podríem executar la comanda *sudo ip route del default via 8.0.0.1 dev enp0s3* al client VPN i tot continuaria funcionant de la mateixa manera
NOTA: Una conseqüència del fet que tot el tràfic provinent del client VPN s'enruti directament al servidor VPN i sigui aquest el qui llavors faci la feina real d'encaminament és que la resolució DNS de les peticions demanades pel client VPN les fa realment el servidor VPN (contra el servidor DNS configurat al client, sí) un cop ha travessat la petició en qüestió el túnnel VPN (sense haver sigut processada). Això es pot comprovar si executem al client per exemple la comanda *dig +short www.hola.com* mentre mantenim la comanda *tshark -i wg0 -Y "dns"* en marxa al servidor VPN. De totes formes, per estar segur que les peticions DNS travessen el túnel i no les fa el client pel seu compte fora d'ell (el que implicaria la pèrdua de privacitat) es pot anar a https://ipleak.net i comprovar el resultat del test que allà s'hi fa

2.-a) No és necessari fer servir cap fitxer de configuració per *wg0* per generar la interfície. Per comprovar-ho, desactiva la tarja "wg0" del client VPN (executant *sudo wg-quick down wg0*), executa les següents ordres una després de l'altra (com a "root") i digues què veus si tornes llavors a executar *sudo wg show wg0* :

ip link add wg0 type wireguard
ip address add 192.168.3.2/24 dev wg0
wg set wg0 private-key privclient.key
wg set wg0 peer SERVER_PUBLIC_KEY endpoint SERVER.REAL.IP:12345 allowed-ips 0.0.0.0/0
ip link set up dev wg0
ip route change default via 192.168.3.1 dev wg0

b) ¿Tornes a fer "ping" amb 16.16.0.2? ¿Per què?

c) ¿Què passa si ara executes *sudo wg-quick save wg0*? (comprova-ho executant *sudo wg showconf wg0*)

d) Desactiva de nou la targa "wg0" del client VPN i canvia ara el valor de la directiva *AllowedIPs=* del seu arxiu de configuració per a què ara valgui *16.16.0.0/16,1.1.1.1* Torna a activar la tarja i intenta tornar a fer "ping" a 16.16.0.2. ¿Veus algun canvi? ¿Per què?

En el següent escenari, simularem l'accés a un servidor intern de la nostra LAN a través d'un servidor VPN. El client VPN continuarà representant el nostre ordinador de casa, un servidor VPN seguirà estant accessible públicament però en aquest cas imaginarem que pertany a la nostra infraestructura pròpia (és a dir, és el "router" de la oficina on volem accedir) i finalment, la tercera màquina serà també un servidor HTTP però en aquest cas, com hem dit intern de la LAN de la oficina. Primer de tot, però, haurem de canviar algunes coses de la infraestructura de xarxa establerta als exercicis anteriors:

0BIS.-a) L'adreça IP del client VPN en aquest exercici pot seguir essent la mateixa que la d'exercicis anteriors per simplicitat (és a dir, 8.0.0.2/8), i la de la tarja *enp0s3/enp1s0* del servidor VPN també, però per ser una mica més realista en l'escenari, ara canviarem les IPs de la tarja *enp0s8/enp2s0* del servidor VPN i la de la tarja *enp0s3/enp1s0* del "servidor HTTP" per a que siguin IPs privades. Concretament, al servidor VPN executa la comanda *sudo ip address add 172.16.0.1/16 dev enp0s8* i al "servidor HTTP" la comanda *sudo ip address add 172.16.0.2/16 dev enp0s3*

NOTA: Recorda que els rangs d'IPs privades possibles només són tres: 10.0.0.0/8-10.255.255.255/8 ; 172.16.0.0/16-172.31.255.255/16 i 192.168.0.0/24-192.168.255.255/24 La resta d'IPs possibles són públiques
NOTA: Si es vol, al VirtualBox es pot canviar el nom de la xarxa interna on està cada xarxa de "VPN" a "Internet" i viceversa per coherència, però no cal

aII) Fes que la porta d'enllaç del client VPN sigui la IP de la tarja enp0s3 del servidor VPN (recorda, això es pot fer directament amb la comanda *sudo ip route add default via 8.0.0.1 dev enp0s3*). Igualment, fes que la porta d'enllaç del "servidor HTTP" intern de la LAN sigui la IP de la tarja enp0s8 del servidor VPN (executa, per tant, la comanda *sudo ip route add default via 172.16.0.1 dev enp0s3*)

b) Assegura´t que l'"IP-Forwarding" estigui activat al servidor VPN (d'exercicis anteriors). Recorda que això ho pots esbrinar fàcilment observant la sortida de la comanda *sysctl -n net.ipv4.ip_forward*

Fent els passos anteriors, en exercicis anteriors ja hauria de ser suficient per a què el client VPN i el servidor HTTP es poguessin fer "ping" entre elles. No obstant, el fet que ara el servidor HTTP tingui una adreça IP privada complica l'assumpte perquè si no fem res més, ni aquesta màquina podrà accedir a l'exterior ni des de l'exterior podran contactar amb ella. Aquí es on entra el tallafocs integrat en el "router" de la oficina, o el que és el mateix, en nostre servidor VPN.

NOTA: En aquest sentit, per simplificar, aquí hem fet que el servidor VPN tingui dues tarjes: una de cara a "Internet" amb la seva IP "pública" i una altra connectada a la xarxa interna privada. No obstant, a la realitat aquest servidor sol estar dins de la LAN amb una sola tarja de xarxa mentre que l´admissió de les connexions provinents de l'exterior es fa amb un "router" configurat amb DNAT per redirigir directament les peticions rebudes per un port determinat a la IP real -és a dir, "no VPN"!- (i el port triat) del servidor VPN local

Concretament hem d'aconseguir el següent:

* A través del "router" de l'oficina(que serà també el servidor VPN), el servidor intern de la LAN ha de poder accedir a Internet malgrat tenir una IP privada (com qualsevol altre ordinador d'aquella LAN). Això s'aconsegueix mitjançant una tècnica anomenada SNAT, implementada a nivell de tallafocs

* A través del "router" de l'oficina (que serà també el servidor VPN), el servidor intern de la LAN ha d'estar accessible *des d'*Internet tot i tenir una IP privada. Això s'aconsegueix mitjançant una tècnica anomenada DNAT, implementada a nivell de tallafocs

Això no té res a veure amb si tenim implementada una VPN o no: és una qüestió de redirecció de paquets

c) Configura el tallafocs del sistema Nftables del servidor VPN per tal que incorpori les regles adients que permetran que aquest servidor pugui aplicar tant el SNAT (concretament, la regla "masquerade" dins de la cadena "postrouting") com el DNAT (concretament, la regla homònima de la cadena "prerouting") a la tarja que està connectada al món exterior (en aquest cas, doncs, enp0s3). Concretament:

* En el cas d´usar Fedora, atura el servei de tallafocs que ve en marxa per defecte (així: *sudo systemctl stop firewalld*). Tot seguit afegeix les següents línies al final de l´arxiu "/etc/sysconfig/nftables.conf" i reinicia el servei Nftables (*sudo systemctl start nftables)*:

```
flush ruleset
table inet nat {
        chain postrouting { type nat hook postrouting priority 100;
                            oifname enp0s3 masquerade
        }
        chain prerouting { type nat hook prerouting priority -100;
                           iifname enp0s3 tcp dport 4321 dnat ip to 172.16.0.2:4321
        }
}
```

NOTA: Les paraules "ip to" després de la paraula "dnat" en la comanda anterior són necessàries perquè la taula emprada és de tipus "inet" (i no pas "ip/ip6", que és on no caldria indicar aquestes paraules després de "dnat")

* En el cas d´usar Ubuntu, atura el servei de tallafocs que ve en marxa per defecte (així: *sudo systemctl stop ufw*). Tot seguit afegeix les següents línies al final de l´arxiu "/etc/nftables.conf" i reinicia el servei Nftables (*sudo systemctl start nftables)*:

```
table inet nat {
        chain postrouting { type nat hook postrouting priority 100;
                            oifname enp0s3 masquerade
        }
        chain prerouting { type nat hook prerouting priority -100;
                           iifname enp0s3 tcp dport 4321 dnat ip to 172.16.0.2:4321
        }
}
```

NOTA: Òbviament, a les regles hauràs d'indicar el nom de la tarja adient; en IsardVDI aquesta serà *enp1s0*
NOTA: Pots comprovar que les cadenes i regles anteriors s'han afegit bé observant la sortida de *sudo nft list ruleset*

d) Posa en marxa de nou al servidor HTTP la comanda *python -m http.server 4321*. Tot seguit comprova, executant en el client VPN la comanda *curl http://8.0.0.1:4321*, que pots accedir a la pàgina web inicial del servidor HTTP intern (i, per tant, que el DNAT funciona). Observa les darreres línies aparegudes a la sortida de la comanda Python: ¿quina és la IP origen que queda registrada?

NOTA: Recorda que en el cas del servidor Apache2, caldria mirar a l´arxiu "/var/log/apache2/access.log" (a Ubuntu) o "/var/log/httpd/access_log" (a Fedora)

Hem aconseguit que el client VPN (el qual representa que està connectat directament a Internet) pugui accedir a un servidor intern corporatiu d´una determinada LAN mitjançant la tècnica DNAT estàndard. Però ara ho hem de fer d´una forma segura (és a dir, a través de realment un túnel VPN) perquè tal com està ara mateix establerta la infraestructura (és a dir, sense VPN), qualsevol "espia" podria veure tot el tràfic intercanviat entre el client VPN i el servidor VPN (el qual representa el primer "lloc segur" perquè ja forma part de la xarxa interna de la nostra organització...a partir d'ell fins arribar al servidor HTTP intern la infraestructura representa que ja és privada). A continuació afegirem, doncs, el túnel.

3.-a) Al servidor VPN sustitueix la regla del tallafocs *iifname enp0s3 tcp dport 80 dnat ip to 172.16.0.2:80* per aquesta altra: *iifname **wg0** tcp dport 80 dnat ip to 172.16.0.2:80* i reinicia de nou el servei Nftables. D'aquesta manera, hauràs activat el DNAT (només!!) per aquesta nova tarja *wg0* en lloc de la *enp0s3* de l'exercici anterior.

b) Al servidor VPN no cal que canvïis res de la configuració escrita a l´arxiu "/etc/wireguard/wg0.conf". Només cal que executis *sudo wg-quick up wg0* i prou **.** Igualment, al client VPN no cal que tampoc canvïis res de la configuració escrita a l´arxiu "/etc/wireguard/wg0.conf". Només cal que executis *sudo wg-quick up wg0* i prou

c) Comprova, executant al client VPN la comanda *curl http://192.168.3.1*, que pots accedir, a través del servidor VPN a la pàgina web inicial del servidor HTTP intern. Observa les darreres línies aparegudes a la sortida de la comanda Python: ¿quina és la IP origen que queda registrada?

> NOTA: En aquest cas, com que no estem fent un SNAT del client VPN (és a dir, substituïnt la IP d´origen per la del servidor VPN, no tindrem la possibilitat d'"emmascarar" la nostra IP d´orígen; això vol dir que el servidor HTTP intern sí que veurà la nostra IP (la de tipus VPN, això sí)

d) Confirma que, efectivament, el trànsit entre client i servidor VPN està xifrat executant, al servidor, la comanda *tshark -i enp0s3* i tornant a repetir la comanda *curl http://192.168.3.1* en el client. ¿Quina sortida treu el Tshark? ¿Per què penses, tot i així, que mantenir una connexió HTTP en lloc de HTTPS, a pesar de la VPN, és molt mala idea? Pista: executa *tshark -i wg0* i ho entendràs

En el següent escenari, simularem la connexió a un servidor VPN de múltiples clients VPN (i l'eventual xarxa formada per tots ells)

4.-a) Canvia la configuració de xarxa de les tres màquines utilitzades als exercicis anteriors així:

> * Elimina la segona tarja de xarxa de la màquina servidor VPN (queda't doncs, amb *enp0s3/enp1s0*)
> * Fes que el nom de la xarxa on pertanyi la tarja de xarxa de la màquina-servidor HTTP sigui el mateix que el de les altres dues màquines (és a dir, que totes estiguin dins de la xarxa "Internet", per exemple, o "Personal1")

b) Arrenca totes tres màquines i assigna a la tarja de cadascuna una adreça IP pública d'una xarxa diferent (és a dir, per exemple, 1.0.0.1/8, 2.0.0.1/8 i 3.0.0.1/8). Comprova que la màquina-servidor HTTP no podrà fer "ping" a la resta (ni viceversa, és clar).

c) Fes que la màquina-servidor HTTP ara sigui un segon client VPN i crea-hi el corresponent parell de claus pública i privada Wireguard. Tot seguit, edita-hi l'arxiu "/etc/wireguard/wg0.conf" per a què tingui el següent contingut, i finalment executa *sudo wg-quick up wg0*:

```
[Interface]
PrivateKey = NEW_CLIENT_PRIVATE_KEY
Address=192.168.3.3/24
[Peer]
PublicKey = SERVER_PUBLIC_KEY
Endpoint = SERVER.REAL.IP:12345
AllowedIPs = 0.0.0.0/0
```

d) Arrenca el "primer" client VPN que has anat fent servir als exercicis anteriors. No cal que canviïs res en la seva configuració, tan sols executa *sudo wg-quick up wg0*.

e) Arrenca el servidor VPN i desactiva el seu tallafocs (*sudo systemctl stop nftables*); l´"IP-Forwarding" tant és que estigui activat o no. Tot seguit, edita el seu arxiu "/etc/wireguard/wg0.conf" per a què tingui el següent contingut (les novetats estan marcades en negreta) i finalment executa *sudo wg-quick up wg0:*

[Interface]
PrivateKey = SERVER_PRIVATE_KEY
Address=192.168.3.1/24
ListenPort=12345
[Peer]
PublicKey = CLIENT_PUBLIC_KEY
AllowedIPs = 192.168.3.2/32
[Peer]
PublicKey = NEW_CLIENT_PUBLIC_KEY
AllowedIPs = 192.168.3.3/32

NOTA: La configuració del servidor també es podria fer, recordem, "al vol" amb les comandes *wg set + wg-quick save*

f) Confirma que totes tres màquines es poden fer "ping" utilitzant la seva IP "privada" de la xarxa VPN, independentment de si la seva IP "real" és d'una altra xarxa diferent o no i sense haver de fer cap tipus de redirecció ni NAT!. Executa des del primer client VPN la comanda *curl http://192.168.3.3* ¿què veus?

NOTA: Fixeu-vos que, en un cas real, fer l´apartat anterior implicaria que clients separats per una WAN com Internet es podrien veure entre sí (i amb el servidor) com si estiguessin dins d´una LAN. De fet, la VPN per definició és una LAN. Això ens pot permetre treballar en xarxa independentment de la nostra ubicació física. Recordem, en aquest sentit, que per a què una VPN Wireguard funcioni només cal que el servidor tingui una IP pública fixa: la dels clients pot ser dinàmica perfectament; això actualment és un requisit força assequible si implementem el servidor en un IaaS d´Internet (tipus AWS o similar) mentre que els clients poden ser ordinadors estàndard darrera d´una connexió a Internet domèstica via NAT

g) Per evitar que els clients VPN es puguin veure entre sí (hi ha situacions en que només interessa que cada client es connecti al servidor independentment i ja està) la manera més fàcil és iniciar el servei del tallafocs Nftables de la màquina servidora VPN i tot seguit afegir la regla marcada en negreta següent (en aquest cas, afegirem la taula, cadena i regla necessàries directament des del terminal de comandes en lloc de des de d'un fitxer, per veure una forma alternativa; recordeu, però, que fent-ho així aquestes regles només estaran actives mentre el servei es mantingui en marxa):

sudo nft flush ruleset
sudo nft add table inet filter
sudo nft add chain inet filter forward { type filter hook forward priority 0 \; }
sudo nft add rule inet filter ***forward iifname wg0 oifname wg0 reject*** *with icmp type admin-prohibited*

Executa les comandes anteriors i tot seguit comprova com, des d´un client es pot continuar fent "pings" al servidor però no pas a l'altre client (i viceversa).

NOTA: Fixeu-vos que la regla anterior prohibeix el trànsit Wireguard<->Wireguard però no pas el trànsit Wireguard<->Ethernet; això fa que cada client podria seguir tenint accés a Internet travessant el servidor VPN (si aquest implementés "masquerading" sobre la tarja Ethernet)
NOTA: Si es vol relaxar la regla anterior per determinats clients, es pot afegir abans la següent regla addicional (on 192.168.3.3 representa un client i 192.168.3.0/24 la xarxa VPN): *sudo nft add rule inet filter forward iifname wg0 ip saddr 192.168.3.3/32 ip daddr 192.168.3.0/24 accept*
NOTA: Una forma alternativa de controlar la visibilitat entre clients VPN és mitjançant l´ús de diverses tarjes wgX definides al servidor i aplicar les regles "Forwarding" adients entre elles, tal com s´explica a https://jrs-s.net/2018/08/05/routing-between-wg-interfaces-with-wireguard, però no és una sol·lució que aporti res en especial a la indicada en els apartats anteriors

h) ¿Què passaria si a la línia *AllowedIps=* d´algun client no hi hagués el valor "0.0.0.0/0" sinó el valor "192.168.3.0/8"? ¿I si hi hagués el valor "153.223.4.0/24,192.168.3.0/8", per exemple (on "153.223.4.0/24" representa la IP pública d´una xarxa qualsevol?

5.-a) ¿Què explica aquest article: https://gist.github.com/joepie91/5a9909939e6ce7d09e29 ?

b) Explica, en aquest sentit, la motivació d'aquest altre article: https://drew2a.medium.com/replace-your-vpn-provider-by-setting-up-wireguard-on-digitalocean-6954c9279b17

> NOTA: Un "droplet" vol dir simplement "màquina virtual en el núvol" segons la literatura comercial de l'empresa DigitalOcean (https://www.digitalocean.com/pricing/droplets) . Altres empreses que ofereixen un recurs similar són Vultr, ScaleWay, Hetzner o Linode, entre altres. En aquest sentit, aquestes empreses ofereixen guies d'instal·lació i ús de Wireguard a les màquines virtuals del seus clients, com https://www.vultr.com/marketplace/apps/wireguard#getting-started o https://www.scaleway.com/en/docs/tutorials/install-wireguard o https://docs.hetzner.com/cloud/apps/list/wireguard o https://www.linode.com/marketplace/apps/linode/wireguard-vpn , respectivament, tot i que la guia pròpia de DigitalOcean és la més complerta, de fet: https://www.digitalocean.com/community/tutorials/how-to-set-up-wireguard-on-ubuntu-22-04

6.-a) Vés a https://play.google.com/store/apps/details?id=com.wireguard.android&hl=es ¿Què t'hi trobes?

b) Vés a https://github.com/pivpn/pivpn ¿Què t'hi trobes?

c) ¿En què pot ser d'ajuda aquesta pàgina web: https://www.wireguardconfig.com (prova de generar una configuració per veure el resultat obtingut)?

> NOTA: Es poden generar codis QR nativament a partir d'un determinat fitxer de text (com pot ser una determinada configuració Wireguard) amb la comanda *qrencode -t ansiutf8 < clientwg.conf* (del paquet homònim). Això és molt interessant si es fan servir com a clients aplicacions de mòbil, les quals poden importar la imatge QR de forma molt senzilla (més que haver d'indicar manualment tots els paràmetres escrits "a mà")