Wireguard

Introduction

WireGuard (https://www.wireguard.com) is a network tunnel (VPN) technology for IPv4 and IPv6. Its main characteristics are:

* Currently most of its code resides in the Linux kernel (but cross platform implementations are under way). This make it a very fast solution comparing to other alternatives

* It operates in the system as a virtual network interface (often named *wg0* but that's not mandatory). This means that the interface can be managed using the standard *ip* tool. In fact, it has to have an own **VPN IP** (besides the **real IP** the underlying real network interface has)

* It uses a customizable UDP port to listen and establish connections

* It features an authentication scheme similar to that of SSH, whereby the VPN server and each client have their own asymmetric key pair. Authorizing a new client is as simple as adding their public key in the server configuration file. Note that WireGuard can be configured to use pre-shared keys but only as an additional mixed layer of security on top of the existing asymmetric keys.

* It forces to use only a reduced set of modern cryptographic primitives, not configurable. Specifically, they are: Curve25519, HKDF, ChaCha20, Poly1305, BLAKE2s and SipHash24

WireGuard introduces the three important concepts:

*"Peer" : Remote host. It's identified by its public key.

*EndPoint": Pair of **real IP** address (or hostname) and port of a peer.

NOTA: It is automatically updated to the most recent source IP address and port of correctly authenticated packets from the peer. This means that a peer that is for example jumping between mobile networks (and whose external IP address changes) will still be able to receive incoming traffic because its endpoint will be updated whenever he sends an authenticated message to the server. This is possible because the peer is identified by its public key.

*"AllowedIP": From the server's point of view, the AllowedIPs are **VPN IP** addresses that a peer is allowed to use as source IP addresses (if a packet arrives to server's side with a source IP that is not in its list of AllowedIPs, it will be simply dropped). For the client, they work as a sort of routing table, determining which peer a packet should be encrypted for.

Installation

Commands to execute in Ubuntu (as root):
add-apt-repository ppa:wireguard/wireguard
apt update && apt install wireguard

Commands to execute in Fedora (as root):
dnf install https://download1.rpmfusion.org/free/fedora/rpmfusion-free-release-$(rpm -E %fedora).noarch.rpm
dnf install wireguard

Common setup (client and server)

1.-Generate private key: *wg genkey > priv.key*
2.-Generate public key: *wg pubkey < priv.key > pub.key*

Los dos pasos anteriores se podrían realizar en uno solo así: *wg genkey | tee priv.key | wg pubkey > pub.key*

Server setup

1.-Make sure that port used by WireGuard and suitable FORWARD chains for wg0 are opened on your firewall
2.-Make sure SNAT is activated with suitable Iptables's MASQUERADE rule for Internet's facing interface.
3.-Make sure IP forwarding is enabled
4.-Edit "/etc/wireguard/wg0.conf" file (its owner must be root with 700 permissions).

```
[Interface]
PrivateKey = *SERVER_PRIVATE_KEY*
Address=192.168.3.1/24
ListenPort=12345
#SaveConfig = true
[Peer]
PublicKey = *CLIENT_PUBLIC_KEY*
AllowedIPs = *CLIENT_VPN_IP/32*
[Peer]
...
```

NOTA:The SaveConfig = true entry in the config file tells WireGuard to automatically update the config file when new clients are added as explained below. The overwriting of the contents of the /etc/wireguard/wg0.conf file will happen whenthe service wg-quick shuts down. So any changes made to the configuration file while the service is running will be overwritten when wg-quick stores its active configuration.

NOTA2: The "Address" directive represents the IP address (optionally with CIDR mask) to be assigned to the interface Recordeu 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.

NOTA3: Para no tener creada constantemente la regla MASQUERADE (mediante *systemctl enable netfilter-persistent && netfilter-persistent save*, por ejemplo), una alternativa podría ser crearla en el momento de activar la tarjeta wg0 y destruirla en el momento de desactivarla. Esto se puede conseguir añadiendo dentro de la sección [Interface] un par de líneas como:

```
PostUp   = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o tarjetaInternet -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -o tarjetaInternet -j MASQUERADE
```

5.-Activate VPN service: *wg-quick up wg0* (It can be shutdown with *wg-quick down wg0*). Another way is executing: *systemctl {start|enable} wg-quick@wg0*
6.-The different connected peers now can be viewed using the commands *wg show* and *wg showconf*

Client setup

1.-Edit "/etc/wireguard/wg0.conf" file (its owner must be root with 700 permissions).

```
[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
[Peer]
...
```

NOTA: Here we have set up a VPN that will route all traffic through the tunnel (using the 0.0.0.0/0, ::/0 AllowedIPs on the client's config) but this can be reduced to a specific range only (thus allowing only specific IPs to be routed through the tunnel). Also multiple configurations might co-exist on each client such that different setups can be used. Simply create different configuration files within /etc/wireguard/ and use them with sudo wg-quick up NAME where name refers to the config file's name without its .conf extension.

NOTA2: PersistentKeepalive directive represents a time interval (in seconds) indicating how often to send an empty UDP packet to the peer for the purpose of keeping a stateful firewall or NAT mapping valid persistently. If set to 0 or "off", this option is disabled. By default or when unspecified, this option is off.

2.-Same as Server's step nº 5
3.-Same as Server's step nº 6

A partir de aquí ambos extremos se podrán hacer ping mediantes las IPs de sus respectivas interfícies wg0 y, si el cortafuegos del servidor está correctamente configurado, el cliente podrá acceder a través de él al mundo exterior.

Every new client must perform the client setup detailed above. Then simply add the new client on the server executing: *wg set wg0 peer CLIENT_PUBLIC_KEY allowed-ips CLIENT_VPN_IP/32*
Thanks to the config entry SaveConfig = true, the new added client will be saved in the config file next time WireGuard is brought down without worrying about manually editing it.
NOTA: Alternativamente, se puede ejecutar *wg setconf wg0 myconfig.conf*

EXERCICIS:

0.-a) Prepara una màquina virtual Ubuntu amb dues tarjes de xarxa: la primera en mode "xarxa interna" i la segona en mode "NAT". Aquesta màquina serà el servidor VPN. Arrenca-la i instal.la-hi el Wireguard. Ara clona aquesta màquina (de forma "enllaçada" i reinicialitzant la MAC) i a la màquina clon elimina la segona tarja de xarxa per tal de què només en tingui una, en mode "xarxa interna". Aquesta màquina serà el client VPN

b) Assigna la IP 10.0.0.1/8 a la tarja enp0s3 del servidor VPN i la IP 10.0.0.2/8 a la tarja enp0s3 del client VPN (ho pots fer directament amb la comanda *ip address add* ... o bé amb un arxiu .network). Comprova que es fan pings entre elles

c) Activa l'IP-Forward al servidor VPN (*echo 1 > /proc/sys/net/ipv4/ip_forward* o millor de forma permanent a l'arxiu sysctl.conf) i també el "Masquerading" per la tarja enp0s8 (*iptables -t nat -A POSTROUTING -o enp0s8 -j MASQUERADE*). Assigna com a porta d'enllaç del client VPN la IP de la tarja enp0s3 del servidor VPN. (amb *ip route add...* o bé en l'arxiu .network corresponent) Comprova que des de la màquina client es pot fer ping, per exemple, a la IP 8.8.8.8

1.-a) Segueix els passos indicats a l'apartat "Common setup", tant a la màquina servidor com client, per crear les respectives claus públiques i privades.

b) Edita al servidor l'arxiu "/etc/wireguard/wg0.conf" amb el següent contingut...:

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

...i seguidament executa *systemctl start wg-quick@wg0* . Comprova amb *wg show* que la configuració s'ha llegit bé

c) Ara Edita al client l'arxiu "/etc/wireguard/wg0.conf" amb el següent contingut...:

```
[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
```

...i seguidament executa *systemctl start wg-quick@wg0* . Comprova amb *wg show* que la configuració s'ha llegit bé

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*. Comprova que també pugui fer "ping" a qualsevol IP d'Internet (com per exemple 8.8.8.8) executant la comanda *ping -I 192.168.3.2 8.8.8.8* . Comprova que també es poden fer "ping" a noms de domini executant la comanda *ping -I 192.168.3.2 elpuig.xeill.net*.

e) De fet, comprova que no cal ni indicar la interfície d'origen que sigui wg0 perquè ja ho és per defecte. Concretament, executa al servidor la comanda *tcpdump -i wg0* i executa al client la comanda *ping elpuig.xeill.net.* ¿Què veus?

f) ¿Quins passos hauries de fer si volguessis connectar un altre client al mateix servidor Wireguard?

2.-Atura el servei wg-quick@wg0 del client i ara executa les següents ordres una després de l'altra:

ip link add wg0 type wireguard
ip address add 192.168.3.2/24 dev wg0
*wg set wg0 private-key priv.key peer *SERVER_PUBLIC_KEY* endpoint *SERVER_REAL_IP:12345* allowed-ips 0.0.0.0/0*
ip link wg0 set up dev wg0

¿Tornes a fer "ping" amb qualsevol domini d'Internet?