Wireguard - an extremely simple yet fast and modern VPN

Contents

About Wireguard

Wireguard is an extremely simple yet fast and modern VPN that utilizes state-of-the-art cryptography. It can be a useful replacement for IPSec or OpenVPN.

Official website: https://www.wireguard.io/

Installation

Wireguard is packaged in DebianUnstable as DebianPackage: wireguard which pulls in DebianPackage: wireguard-dkms and DebianPackage: wireguard-tools.

The packages also work on DebianJessie and DebianStretch, by following the Wireguard installation instructions which boil down to:

```
# echo "deb http://deb.debian.org/debian/ unstable main" > /etc/apt/sources.list.d/unstable-wireguard.list
# printf 'Package: *\nPin: release a=unstable\nPin-Priority: 150\n' > /etc/apt/preferences.d/limit-unstable
# apt update
# apt install wireguard
```

Configuration

Step 1 - Generating Keypairs

To generate key pairs, use:

```
# wg genkey | tee wg-private.key | wg pubkey > wg-public.key
```

Make sure that you protect the `wg-private.key` file, e.g. via appropriate file permissions.

Step 2 - Alternative A - Manual Configuration

For testing purposes, it may be sufficient to use `wg-quick` directly - see the DebianMan: wg-quick(8) man page (upstream version) and quickstart instructions.

Step 2 - Alternative B - /etc/network/interfaces

The following configuration examples focus on using /etc/network/interfaces as much as possible.

For a server, configuration based on `/etc/network/interfaces` is often the preferred way.

Point-to-point tunnel

This example builds a simple point-to-point tunnel between two machines.

```
# /etc/network/interfaces
auto wg-p2p
iface wg-p2p inet static
        address 10.88.88.1
        netmask 255.255.255.0
        pre-up ip link add $IFACE type wireguard
        pre-up wg setconf $IFACE /etc/wireguard/$IFACE.conf
        post-down ip link del $IFACE
iface wg-p2p inet6 static
        address 2001:db8:1234:5678::1
        netmask 64
```

```
# /etc/wireguard/wg-p2p.conf
[Interface]
PrivateKey = <paste the private key of the local host here>
ListenPort = <enter a port number to use for Wireguard UDP data, 51820 seems common>

[Peer]
Endpoint = <remote IP>:<remote port>
PublicKey = <paste the public key of the remote host here>
AllowedIPs = 0.0.0.0/0, ::/0
```

You can then simply add routes through the tunnel, either statically, or dynamically using e.g. OSPF or BGP. For static routes:

```
# ip route add 2001:db8:4242::/48 dev wg-demo
# ip route add 192.168.42.0/24 dev wg-demo
```

VPN client with default route

This allows a "client" to connect to a server, and redirect its default route through the tunnel. This example uses wg-quick, make sure you understand what it does to your routing tables!

```
# /etc/network/interfaces
auto wg-client
iface wg-client inet static
        address 10.88.88.1
        netmask 255.255.255.0
        pre-up wg-quick up $IFACE
        post-down wg-quick down $IFACE
```

```
# /etc/wireguard/wg-client.conf
[Interface]
PrivateKey = <paste the private key of the local host here>
ListenPort = <enter a port number to use for Wireguard UDP data, 51820 seems common>

[Peer]
Endpoint = <server IP>:<server port>
PublicKey = <paste the public key of the remote host here>
AllowedIPs = 0.0.0.0/0, ::/0
```

Step 2 - Alternative C - systemd

systemd has native support for setting up Wireguard interfaces since version 237 (available in DebianUnstable).

First, create a DebianMan: systemd.netdev(5) file ending in `.netdev` and place it in `/etc/systemd/network`, for example as `/etc/systemd/network/wg0.netdev`:

```
[NetDev]
Name=wg0
Kind=wireguard
Description=Wireguard test

[WireGuard]
PrivateKey=<paste the private key of the local host here>
ListenPort=<enter a port number to use for Wireguard UDP data, 51820 seems common>

[WireGuardPeer]
PublicKey=<paste the public key of the remote host here>
AllowedIPs=0.0.0.0/0
AllowedIPs=::/0
Endpoint=<remote IP or hostname>:<remote port>
```

Note that the above example assumes that you are setting up a "client" to connect to a server. If you are instead setting up a server, you probably want much more restricted `AllowedIPs` entries. Also, on a server you would typically have several `WireGuardPeer` sections.

The `.netdev` file contains security-sensitive data (the private key) and should have appropriate file permissions:

```
# chown root.systemd-network /etc/systemd/network/wg0.netdev
# chmod 0640 /etc/systemd/network/wg0.netdev
```

Second, create a matching DebianMan: systemd.network(5) file ending in `.network` and place it in `/etc/systemd/network`, for example as `/etc/systemd/network/wg0.network`:

```
[Match]
Name=wg0

[Network]
Address=10.88.88.1/24
Address=2001:db8:1234:5678::1
```

Now tell systemd to reload its configuration and start DebianMan: systemd-networkd(8):

```
# systemctl daemon-reload
# systemctl start systemd-networkd
```

Step 3 - Check the end result

You can check the status of your new interface by using e.g.:

```
# ip addr show dev wg0
# wg show wg0
```

And, if you've let systemd create the interface, by using:

```
# networkctl status wg0
```

Mobile clients Configuration

Wireguard has a user space implementation for mobile devices available via the Wireguard app - note that at the moment of writing (2018 July 28) it is available only on Android. The client config can be configured in 3 ways:

1. Create configuration manually

This is self-explanatory, you actually create the config on the mobile device then transfer the relevant keys to the server's config.

2. Create configuration from archive

Here you have to create a .zip archive of the client configuration file, transfer it to the device then import it into the app.

3. Import by reading a QR code (most secure method)

The mobile client as of version 0.0.20180724 supports QR code based input. In the Debian repositories there is a package called qrencode that is capable of generating qr codes even in a terminal/console using UTF8 characters. The syntax is:

```
# qrencode -t ansiutf8 < client.conf
```

This will generate a QR code that is readable by the mobile client.

The advantage of this approache is that there is no need to transfer sensitive information via data channels that can potentially be compromised and there is no need of any other supplementary software besides a terminal or console.