WireGuard

From the **WireGuard (https://www.wireguard.com/)** project homepage:

> Wireguard is an extremely simple yet fast and modern VPN that utilizes state-of-the-art cryptography. It aims to be faster, simpler, leaner, and more useful than IPSec, while avoiding the massive headache. It intends to be considerably more performant than OpenVPN. WireGuard is designed as a general purpose VPN for running on embedded interfaces and super computers alike, fit for many different circumstances. Initially released for the Linux kernel, it plans to be cross-platform and widely deployable. It is currently under heavy development, but already it might be regarded as the most secure, easiest to use, and simplest VPN solution in the industry.

Warning: *WireGuard has not undergone proper degrees of security auditing and the protocol is still subject to change.* **[1] (https://www.wireguard.com/#work-in-progress)**

Contents

Installation

Install the **wireguard-dkms (https://www.archlinux.org/packages/?name=wireguard-dkms)** and **wireguard-tools (https://www.archlinux.org/packages/?name=wireguard-tools)** packages.

Usage

To create a public and private key

```
$ wg genkey | tee privatekey | wg pubkey > publickey
```

Below commands will demonstrate how to setup a basic tunel between two peers with the following settings:

	Peer A	Peer B
External IP address	10.10.10.1/24	10.10.10.2/24
Internal IP address	10.0.0.1/24	10.0.0.2/24
wireguard listening port	UDP/48574	UDP/39814

Peer A setup

This peer will listen on UDP port 48574 and will accept connection from peer B by linking its public key with both its inner and outer IPs addresses.

```
$ ip link add dev wg0 type wireguard
$ ip addr add 10.0.0.1/24 dev wg0
$ wg set wg0 listen-port 48574 private-key ./privatekey
$ wg set wg0 peer [Peer B public key] persistent-keepalive 25 allowed-ips 10.0.0.2/32 endpoint 10.10.10.2:39814
$ ip link set wg0 up
```

Peer B setup

As with Peer A, whereas the wireguard daemon is listening on the UDP port 39814 and accept connection from peer A only.

```
$ ip link add dev wg0 type wireguard
$ ip addr add 10.0.0.2/24 dev wg0
$ wg set wg0 listen-port 39814 private-key ./privatekey
$ wg set wg0 peer [Peer A public key] persistent-keepalive 25 allowed-ips 10.0.0.1/32 endpoint 10.10.10.1:48574
$ ip link set wg0 up
```

Basic checkups

Invoking the wg command without parameter will give a quick overview of the current configuration.

As an example, when Peer A has been configured we are able to see its identity and its associated peers:

```
peer-a$ wg
interface: wg0
  public key: UguPyBThx/+xMXeTbRYkKlP0Wh/QZT3vTLPOVaaXTD8=
  private key: (hidden)
  listening port: 48574

peer: 9jalV3EEBnVXahro0pRMQ+cHlmjE33Slo9tddzCVtCw=
  endpoint: 10.10.10.2:39814
  allowed ips: 10.0.0.2/32
```

At this point one could reach the end of the tunnel:

```
peer-a$ ping 10.0.0.2
```

Persistent configuration

The config can be saved by utilizing `showconf`

```
$ wg showconf wg0 > /etc/wireguard/wg0.conf
$ wg setconf wg0 /etc/wireguard/wg0.conf
```

Setup a VPN server

Wireguard comes with a tool to quickly create and tear down VPN servers and clients, `wg-quick`. Note that the config file used here is not a valid config file that can be used with `wg setconf`.

Server

/etc/wireguard/wg0server.conf

```
[Interface]
Address = 10.200.100.1/24
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
ListenPort = 51820
PrivateKey = [SERVER PRIVATE KEY]

[Peer]
PublicKey = [CLIENT PUBLIC KEY]
AllowedIPs = 10.200.100.2/32  # This denotes the clients IP.
```

Bring this interface up by using `wg-quick up wg0server`, and use `wg-quick down wg0server` to bring it down.

Client

/etc/wireguard/wg0.conf

```
[Interface]
Address = 10.200.100.2/32  # The client IP from wg0server.conf
PrivateKey = [CLIENT PRIVATE KEY]
DNS = 10.200.100.1

[Peer]
PublicKey = [SERVER PUBLICKEY]
AllowedIPs = 0.0.0.0/0
Endpoint = [SERVER ENDPOINT]:51820
PersistentKeepalive = 25
```

Bring this interface up by using `wg-quick up wg0`, and use `wg-quick down wg0` to bring it down.

To bring this up automatically one can use `systemctl enable wg-quick@wg0`

If you use **NetworkManager**, it may be necessary to also enable NetworkManager-wait-online.service `systemctl enable NetworkManager-wait-online.service`

or if you're using systemd-networkd, to enable systemd-networkd-wait-online.service `systemctl enable systemd-networkd-wait-online.service`

to wait until devices are network ready before attempting wireguard connection.

Troubleshooting

DKMS module not available

If the following command does not list any module after you installed **wireguard-dkms (https://www.archlinux.org/packages/?name=wireguard-dkms)**,

```
$ modprobe wireguard && lsmod | grep wireguard
```

or if creating a new link returns

```
# ip link add dev wg0 type wireguard
RTNETLINK answers: Operation not supported
```

you probably miss the linux headers.

These headers are available in **linux-headers (https://www.archlinux.org/packages/?name=linux-headers)** or **linux-lts-headers (https://www.archlinux.org/packages/?name=linux-lts-headers)** depending of the kernel installed on your system.

Tips and tricks

Store private keys in encrypted form

It may be desirable to store private keys in encrypted form, such as through use of **pass (https://www.archlinux.org/packages/?name=pass)**. Just replace the PrivateKey line under [Interface] in your config file with:

```
PostUp = wg set %i private-key <(su user -c "export PASSWORD_STORE_DIR=/path/to/your/store/; pass WireGuard/private-keys/%i")
```

where user is your username. See the `wg-quick(8)` man page for more details.

Retrieved from "https://wiki.archlinux.org/index.php?title=WireGuard&oldid=507638"
