ENCOM

About

Raspberry Pi, Coding, Linux

Build your own IPv6 VPN with Wireguard

Saturday, January 9, 2021

> Finally, no `/etc/network/interfaces` and `wg-quick`

Update Mar 23 ‘21: Improve the stability of IPv6 tunnel.

I like IPv6, but Cox’s IPv6 network is suboptimal.

I like the idea of a VPN providing millions of IPv6 addresses to its clients.

Wireguard, it seems, is the obvious choice for creating an IPv6 VPN.

Yes, I did set up Wireguard servers before using Debian, `/etc/network/interfaces` and `wg-quick`. It was not the best experience. Since I was using Hurricane Electric’s IPv6 tunnel broker to provide public IPv6 addresses to clients, I need to write the tunnel configuration in `/etc/network/interfaces`, and then set up Wireguard with `wg-quick`.

Well, I mean it works, right?

Unfortunately I needed some brand new kernel modules on my Wireguard server, `mptcp` to be exact. (to combine bandwidth from multiple interface…) And Debian’s kernel does not have multipath kernel support compiled. How convenient, Debian.

Well, I could compile a custom kernel with `mptcp` enabled, like any other conscious sysadmin would do. But I was feeling adventurous, and took the risk to attempting to set up another Wireguard server on Arch Linux. The first thing I noticed, well, was that `/etc/network/interfaces` was completely empty. Apparently Ubuntu and Arch Linux can use something called `systemd-networkd` instead.

Some Background Information

Wireguard

Wireguard is for setting up VPNs (virtual private networks).

I am using it to proivide public IPv6 addresses and prefixes to devices behind NAT without IPv6 addresses. And also to hide my devices’ real IP, yes.

Hurricane Electric’s IPv6 Tunnel

Well, it turns out you need IPv6 prefix larger than /64 to provide each client with individual /64 IPv6 prefixes. Any prefix smaller than /64 would not work, sadly.

And conveniently, most VPS providers only provide /64 IPv6 prefixes.

The good news is that Hurricane Electric provides IPv6 tunnel with /48 prefix for free, with no bandwidth limits.

Anyway the sign up link is here .

systemd-networkd

The heck is this.

Well, it is a systemd unit for setting up networking interfaces, used by Ubuntu, Arch Linux, etc. And it is much easier to use than Debian's `networking.service` .

In `systemd-networkd` , you write interface configurations in `/etc/systemd/network/your-interface.network` , and virtual interface configurations in `/etc/systemd/network/your-netdev.netdev` .

And *to my surprise*, Wireguard and IPv6 tunnel are all virtual interfaces! Wow!

That means `systemd-networkd` would work.

Actually Setting Up

Yes, I am using Arch Linux. For some reason it is quite stable and lightweight on my servers.

Enable IP Forwarding

First, enable IPv4 and IPv6 forwarding.

Create `/etc/sysctl.d/20-ip-forward.conf` and input:

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.accept_ra = 2
```

This is for IPv4 NAT and forwarding public IPv6 addresses.

Then, run `sudo sysctl --system` to apply new configuration.

Install Wireguard and systemd-networkd

Wireguard is included in most Linux distributions, you just need to install `wireguard-tools` .

```
sudo pacman -S wireguard-tools
```

systemd-networkd is included in systemd, so if your distribution uses systemd, you have it. Just need to enable it (It should already be enabled though).

```
sudo systemctl enable --now systemd-networkd
```

IPv6 Tunnel Configuration

You should already obtained IPv6 tunnel configuration from before.

It looks like this:

- IPv6 Tunnel Endpoints
  - Server IPv4 Address: `Tunnel's IPv4 endpoint`
  - Server IPv6 Address: `Tunnel's IPv6 endpoint`
  - Client IPv4 Address: `Your server's IPv4`
  - Client IPv6 Address: `Obtained IPv6`
- Routed IPv6 Prefixes
  - Routed /64: `Obtained /64 prefix`
  - Routed /48: `Obtained /48 prefix`
- DNS Resolvers:
  - Anycast IPv6 Caching Nameserver: `IPv6 DNS`

I could never get those configurations right on the first try.

Create `/etc/systemd/network/30-he.network` and input:

```
[Match]
Name=he-ipv6

[Network]
Address=<Obtained IPv6, with "/64" suffix>
Gateway=<Tunnel's IPv6 endpoint, without "/64" suffix>
DNS=<IPv6 DNS>
```

Then, create `/etc/systemd/network/30-he.netdev` and input:

```
[Match]
```

```
[NetDev]
Name=he-ipv6
Kind=sit
MTUBytes=1480

[Tunnel]
Local=<Your server's IPv4>
Remote=<Tunnel's IPv4 endpoint>
TTL=255
```

In your existing network connection config file, for example `/etc/systemd/network/20-wired.network`, insert `Tunnel=he-ipv6` in its `[Network]` section.

```
[Match]
Name=ens18

[Network]
DHCP=ipv4
Tunnel=he-ipv6
```

Wireguard Configuration

First generating Wireguard Server's public key and private key.

```
sudo -i
cd /etc/wireguard/
umask 077; wg genkey | tee privatekey | wg pubkey > publickey
```

Your public key and private key are now stored in `/etc/wireguard/publickey` and `/etc/wireguard/privatekey`.

```
cat /etc/wireguard/publickey
<Your Wireguard public key>
cat /etc/wireguard/privatekey
<Your Wireguard private key>
```

Create `/etc/systemd/network/99-wg0.network` and input:

```
[Match]
Name=wg0

[Network]
IPMasquerade=true

[Address]
Address=10.64.0.1/16

[Address]
Address=<Obtained /48 prefix>::1/48
```

`10.64.0.1/64` is your NATed IPv4 subnet.

You can use the entire /48 IPv6 prefix, so might as well use it.

Create `/etc/systemd/network/99-wg0.netdev` and input:

```
[NetDev]
Name=wg0
Kind=wireguard
Description=WireGuard tunnel wg0

[WireGuard]
ListenPort=<Wireguard server port>
PrivateKey=<Your Wireguard private key>

[WireGuardPeer]
PublicKey=<Your Wireguard client's public key>
AllowedIPs=10.64.10.0/24
AllowedIPs=<Obtained /48 prefix>:100::/56
```

Now restart `systemd-networkd` to apply settings, or just reboot.

```
sudo systemctl restart systemd-networkd
```

Try setting up your Wireguard client, you should be able to use the whole /56 IPv6 prefix.

Yeah, you basically just created an IPv6 VPN!

Plus, I had no idea `systemd-networkd` was that convenient. Never would I touch `/etc/network/interfaces` again.

I am also testing multipath TCP in conjunction with Wireguard. If succeeded, it means you can combine network bandwidth from multiple networks, along with automatic fail-over and roaming. It is going to take some time, however.

#Wireguard #VPN #IPv6 #systemd #systemd-networkd #Linux

Dedicated IPv6 address per Shadowsocks Instance

OTG Ethernet: Connect to Raspberry Pi 4 via USB-C

Thoughts?

0 Responses

👍 Upvote 😝 Funny 😍 Love 😲 Surprised 😤 Angry 😥 Sad

0 Comments ENCOM 🔒 Disqus' Privacy Policy 1 Login

♡ Recommend Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

✉ Subscribe Ⓓ Add Disqus to your siteAdd DisqusAdd ⚠ Do Not Sell My Data DISQUS

Powered by Hugo & Notepadium