How to easily configure WireGuard

9-11 minutes

WireGuard is pretty great!

You might have noticed the buzz around WireGuard lately. WireGuard is a very simple VPN that uses state-of-the-art cryptography, and the buzz comes from both the fact that it's simple and good at what it does, and the fact that it's so good that it's going to be included in the Linux kernel by default. Linus Torvalds himself said that he loves it, which took the software world by storm, as we weren't aware that Linus was capable of love or any emotion other than perkele.

The only problem I've found with WireGuard is a lack of documentation, or rather a lack of documentation where you expect it. The quickstart guide, the first thing I look at, mentions a configuration file that it never tells you how to write, and it also assumes you're more familiar with networking than I am.

Since the initial conditions at the creation of the universe set things up so WireGuard would eventually be underdocumented, I am going against Creation itself and showing you how to easily configure and run it. Let's begin!

Short overview

A guard that looks kinda wiry, and makes sure you don't subconsciously find this post dry and boring.

At its core, all WireGuard does is create an interface from one computer to another. It doesn't really let you access other computers on either end of the network, or forward all your traffic through the VPN server, or anything like that. It just connects two computers, directly, quickly and securely.

To do anything other than that, you'll need to configure your network for it, which is "out of scope" for the WireGuard docs, but which I consider very much in the scope of a VPN. Luckily, WireGuard comes with a helper script, `wg-quick`, which will do pretty much everything the average user needs. Keep in mind that the configuration files for `wg-quick` aren't compatible with the `wg` executable, but `wg-quick` is all we'll need, so that shouldn't matter.

Let's start setting everything up.

Installation and setup

To install WireGuard, see the installation page, it should be a pretty simple process. After you've installed it, you will need to generate a private and a public key for each computer you want accessing the

VPN. Due to WireGuard's design, both computers on either end of a connection will need to have each other's public key.

Let's generate these keys. On each of the computers that will be in the VPN, create the directory /etc/wireguard/ and run these commands *on each of the computers* in the directory you just created:

```
$ umask 077  # This makes sure credentials don't leak in a race condition.
$ wg genkey | tee privatekey | wg pubkey > publickey
```

This will generate two files, privatekey and publickey on each of the computers. The publickey file is for telling the world, the privatekey file is secret and should stay on the computer it was generated on. You need to paste the contents of these files in the config file, I'm afraid WireGuard doesn't support referencing them by path yet.

If your VPN server is behind a NAT, you'll also need to open a UDP port of your choosing (51820 by default). Do this for any computer you want to connect *to* (computers that you'll connect *from* don't need a port open, as far as I know, but correct me if I'm wrong).

Common scenarios

After you've done the above, you're ready to configure WireGuard. Go to /etc/wireguard/ and create a file called wg0.conf on each of your computers. We'll go over some common scenarions along with the configuration for each.

Just a single connection

If you just want a single connection between two computers (say, to connect your laptop to your home server), the configuration is pretty simple. On the server, enter the following:

```
[Interface]
Address = 192.168.2.1
PrivateKey = <server's privatekey>
ListenPort = 51820
```

```
[Peer]
PublicKey = <client's publickey>
AllowedIPs = 192.168.2.2/32
```

That's all you need for the server. Keep in mind that `<server's privatekey>` is your home server's `privatekey` file's contents (**not** the path to the file, the actual **contents**, a long line of gibberish), and `<client's publickey>` is similarly the contents of your laptop's `publickey` file.

For the client, here's the configuration:

```
[Interface]
Address = 192.168.2.2
PrivateKey = <client's privatekey>
ListenPort = 21841

[Peer]
PublicKey = <server's publickey>
Endpoint = <server's ip>:51820
AllowedIPs = 192.168.2.0/24

# This is for if you're behind a NAT and
# want the connection to be kept alive.
PersistentKeepalive = 25
```

Similarly, replace the keys with the appropriate strings you generated.

As you can see, the addresses I picked for each computer are `192.168.2.1` and `192.168.2.2`, because that subnet was free in my setup. If there's an interface with that subnet on either computer, you should pick another one, such as `192.168.3.x`, to avoid conflicts.

After writing the two files, run `wg-quick up wg0` on the server and then on the client. The two machines should now be connected if you entered the server's IP in the config and configured the port

correctly, and you should be able to ping 192.168.2.1 from the VPN client and see the responses. To close the connection again, just run wg-quick down wg0.

I repeat that *this setup only lets you access the server's interface from the client*, it won't forward any of your traffic over the server or let you access any other machines on the server's LAN.

Accessing your home LAN

To actually access the server's LAN, you'll need to make a slight modification to the configuration. On the server's config file, at the end of the the [Interface] section, add these two lines:

```
PostUp   = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j ACCEPT; iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -j ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
```

This assumes that your LAN interface is called eth0. If it isn't, change the lines above to the actual name.

On the client, add your LAN's subnet under AllowedIPs. For example, if your subnet is 192.168.1.x, change AllowedIPs to look like this:

```
AllowedIPs = 192.168.2.0/24, 192.168.1.0/24
```

Briefly, the AllowedIPs setting acts as a routing table when sending, and an ACL when receiving. When a peer tries to send a packet to an IP, it will check AllowedIPs, and if the IP appears in the list, it will send it through the WireGuard interface. When it receives a packet over the interface, it will check AllowedIPs again, and if the packet's source address is not in the list, it will be dropped.

You might need to enable IP forwarding on the server for this to work, but it's a simple process for Linux. Here's a good guide. If you're running an OS X or Windows server, you don't deserve nice things.

Then run `wg-quick up wg0` as above, and you should be able to ping the other computers in the LAN from the client, as if you were home. Try `ping 192.168.1.1` or any other computer in your LAN to verify.

Do note that *this won't forward any other traffic* through your server, so it won't proxy your web browsing or anything like that. It will just let you talk to other machines on your server's LAN.

For traffic forwarding, see below:

Forwarding all your traffic through

If you want to forward *all* your traffic through the VPN, WireGuard can easily do that as well. Keep in mind that, if you're doing this to avoid ISP tracking, it won't work against your server's ISP. All your traffic will just look like it's coming from your server, but if that's at your house, all your torrents and porn downloading is just going to look like it's coming from there, even though you're at a net cafe in Cambodia.

To forward all the traffic through, simply change the `AllowedIPs` line on the client to this:

```
AllowedIPs = 0.0.0.0/0, ::/0
```

Here's the entire client config again:

```
[Interface]
Address = 192.168.2.2
PrivateKey = <client's privatekey>
ListenPort = 21841

[Peer]
PublicKey = <server's publickey>
Endpoint = <server's ip>:51820
AllowedIPs = 0.0.0.0/0, ::/0
```

This will make the `wg0` interface responsible for routing all IP addresses (hence the `0.0.0.0/0`), and should route all your traffic

over your server. You can check by loading one of the what is my IP websites and seeing that your server's IP is what's detected.

Securing and running on startup

After you're done, run the following to make the directory and files readable only by administrators (it *does* contain secret keys, after all):

```
$ sudo chown -R root:root /etc/wireguard/
$ sudo chmod -R og-rwx /etc/wireguard/*
```

After you've created and secured the file, you can easily set WireGuard to initialize the VPN on startup if your OS is using systemd:

```
$ sudo systemctl enable wg-quick@wg0.service
```

Similarly, to start or stop the service:

```
$ sudo systemctl start wg-quick@wg0.service
$ sudo systemctl stop wg-quick@wg0.service
```

Epilogue

I hope this has been useful! It's the guide I wish existed before I spent three hours trying to configure WireGuard, and hopefully you can just copy the configs and have it work right away.

If you need the configuration for IPv6, I'm afraid you're going to have to experiment yourself, as my ISP does not support it, but feel free to let me know what should be added and I can amend the article.

As always, tweet or toot any comments to me, or leave a comment below.

Thanks!