Tailscale is magic; even more so with NixOS

Published 2020-09-17 on Farid Zakaria's Blog — Permalink

Our adventure into NixOS continues; this time let’s look into how we can **harden** our NixOS machines by putting them within a VPN. We will be using tailscale to setup our VPN.

Restricting your machines; especially SSH for servers; behind a VPN is a great way to add a layer of security without having to mess with various checklists like making sure *password based logins* are disabled.

First off, tailscale delivers on it’s promise of *“VPN made easy”*; it’s **magic**. Few often do I get to use a product that delivers on the ease of use so wonderfully that it comes off as magic. It has turned the act of setting up a VPN into a *brain dead* process; made even easier when coupled with NixOS.

For the purpose of following along; we will be using my NixOS AWS EC2 machine. It had a SecurityGroup with **SSH 22** open previously so that I could access it.

Simply create a new module **vpn.nix** with the following contents and perform a `nixos-rebuild switch`

```
{ config, pkgs, ... }: {

  # enable the tailscale daemon; this will do a variety of tasks:
  # 1. create the TUN network device
  # 2. setup some IP routes to route through the TUN
  services.tailscale = { enable = true; };

  # Let's open the UDP port with which the network is tunneled through
  networking.firewall.allowedUDPPorts = [ 41641 ];

  # Disable SSH access through the firewall
  # Only way into the machine will be through
  # This may cause a chicken & egg problem since you need to register a machine
  # first using `tailscale up`
  # Better to rely on EC2 SecurityGroups
  # services.openssh.openFirewall = false;

  # Let's make the tailscale binary available to all users
  environment.systemPackages = [ pkgs.tailscale ];
}
```

> Specifically for EC2, I had to also add **41641** as an ingress rule to my SecurityGroup.

After the rebuild, let’s register the machine. Let’s now register the machine & connect to the VPN network.

```
sudo tailscale up
```

The above will provide a HTTPS link for you to login through your registered identity provider (i.e. Google); and register the machine.

After running this, you checkout the changes that were done.

1. Confirm that *tailscaled* is running properly

   ```
   ❯ systemctl is-active tailscaled
   active
   ```

2. Checkout the new network device created.

```
❯ ip link show dev tailscale0
6: tailscale0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1280 qdisc pfifo_fast
link/none
```

> Interestingly, the MTU is set to 1280 which is very conservative for the necessary encapsulation headers to tunnel traffic. #246

3. Checkout the route table created

```
❯ ip route show table 52
100.87.7.89 dev tailscale0 scope link
100.100.100.100 dev tailscale0 scope link
100.101.102.103 dev tailscale0 scope link
```

`100.100.100.100` is the DNS server you can use.

`100.101.102.103` is the default IRC server they make available when you create an account.

`100.87.7.89` is another machine in my VPN.

Now remove the ability to SSH through the default network interface. If you have access to the server directly, go ahead and uncomment the line above.

```
services.openssh.openFirewall = false;
```

If you are on a cloud provider (i.e. AWS), simply remove the SSH ingress rule.

> Unfortunately, I could not think through a way to disable the firewall for a clouded server. On a fresh NixOS rebuild, tailscale will not have been registered but it will disable SSH. Have an idea ? Reach out to me.

Wow! That's it, we're done! We've setup a full VPN between any machines with the help of tailscale.

Now the strength of our SSH is directly related to our identity provider. For instance, I've setup MFA for my Google account.

tailscale & *NixOS* are a great match.

Extra credit

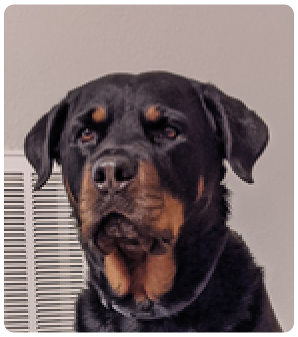
> The following is only really useful for those running home-networks; please don't use it in any production setting.

Now that access via *ssh* is only allowed through the VPN; which requires it's own authentication – we can make further simplifications.

If we wanted to, we could now make *sshd* accept *password based logins* & have our user have an *empty password* for a real simple login experience.

At first this would have the drawback that any user with access to the VPN could access any host & as any user. We can however make use of tailscale's ACL feature to restrict access only to the correct users.

I leave the implementation as an exercise.



I'm a software engineer, father and wishful amateur surfer. If you've come seeking my political views; you've found the wrong Fareed.

linkedin
fmzakari

github
fzakaria

email
farid.m.zakaria@gmail.com

pgp
D1B232E7

Web friendly version of my resume.

Projects

Click here for some personal projects I've worked on.

Old Blog

Click here for the archive of my old blog posts from Wordpress.

Recent Posts

2022-09-12
Making RUNPATH redundant for Nix
2022-03-15
Shrinkwrap: Taming dynamic shared objects
2022-01-06
Computing all output paths for every attribute in Nixpkgs

License

Improve this page @ 8d230c3