Introduction: Novice to Network Admin

Introduction

If you're just looking to get started reading about how I setup everything you can skip down to the Goals section or go straight to Guest Setup to get started.

Background

From when I first started using computers as a kid I treated all things related to networking as a black box. I had a rudimentary understanding of IP addresses but had no real idea how data got from my computer to a server other than the high level concept of "*my computer is sending data to that address*". It is similar to how most of us don't really know how the US Postal Service works. We have a vague notion that we drop a letter in a mailbox to be picked up and then "*my letter gets delivered to the address on the envelope.*" In reality how mail gets picked up, sorted, tracked, bundled, routed, and delivered is much more involved than we ever think about. Up until about three years ago my knowledge hadn't progressed much past knowing network packets existed and for a computer to get on the Internet it had to have an IP address, a subnet mask (not that I really understood what this was), and have a router address.

Like most computer users I never delved into networking. I'd plug my router (always an Apple AirPort into the modem and go with mostly the defaults. I used *Linux* for years but strictly as a server for web apps I programmed and never ventured very far off that path.

That all started to change in 2018 when I decided to build a server (really just a fast PC at the time) to play around with. It also coincided with me starting a new job that put me adjacent to some networking topics that started to peak my interest. Pretty soon I was self hosting a several applications accessible from the Internet and wanted to take the plunge into setting up a DMZ to try and keep my local network safe from the ever increasing number of things I was making publically accessible. Looking into how best to cordon off my applications I saw that putting public applications on a different subnet and often a VLAN is the best practice. But my AirPort didn't

support that so I went searching for a router/firewall that would work better. The general consensus at the time seemed to be to use a Unifi Security Gateway or run pfSense on an old PC. Not having a PC to run it on I first tried virtualizing my router by running it as a virtual machine on my "server." For a newbie this was more complicated than necessary and had the risk that if not done correctly could expose my internal network to the Internet. I liked *pfSense* but I quickly found out the downside of running your gateway to the Internet on your server is that when your server has a problem you likely won't have the Internet to fix it which is quite frustrating.

So I bought a mini PC (Protecli Vault) and started running pfSense on there and was happy for a year. I continued to self-host more applications and eventually got to self-hosting DNS with Pi-hole. Pretty soon I ran into my old problem of when the server is down there is no DNS and so the Internet pretty much stops working. I briefly considered buying buyin a Raspberry Pi which would have been a great solution but I decided to treat the mini PC as my "network infrastructure server" and instead of just running pfSense on there I'd use Proxmox VE and virtualize both *pfSense* and *Pi-hole* the same way I had been virtualizing Pi-hole on my server.

This worked great except I eventually noticed that my maximum download speeds were slower than they had been when only *pfSense* was running on the min PC. After a bunch of testing and attempted workarounds I realized I ~~needed~~ *wanted* a new plan. So once again I bought a mini PC that was pretty much just a more powerful version of the one I already had to try and fix the problem. I was disappointed to see that my upgrade helped but wasn't enough to overcome the overhead that came along with virutalizing *pfSense*. Additionally I was starting to push into things that *pfSense* didn't support yet like using Wireguard.

The next step was to investigate if a *Linux* based firewall would perform better while virtualized. The answer turns out to be no since both *pfSense* and *Linux* both implement *Virtio* network drivers that work very similarly and the real problem seems to just be the result of the additional layers a packet has to travel up through. Each packet coming in must be processed by the hypervisor kernel then redirected to the virtualized router kernel and if the packet was destined somewhere else had to go through the same layers in reverse to exit. That seems to be jus too much for a device with modest CPU and memory performance.

Then it dawned on me that I could get around those layers of virtualization by using the same containerization I had been using for servers I had virtualized to run all my self-hosted projects. I prefer to run my guest machines as LXC (Linux Container) guests instead of virtual machines. A *LXC* guest uses the same *Linux* Kernel as the host Operating System so there are no additional layers of virtualization overhead to deal with. Pretty neat!

When I was looking at *Linux* based firewalls I came across VyOS which is based on Debian Linux and allowed me to peek behind the curtain of the types of tools you use for a *Linux* firewall. The seed of an idea had been planted that would use all the knowledge I gained over the last 3 years tinkering with my home network-- *Linux*, *Proxmox*, *Wireguard*, and virtualization.

I could run a *LXC* guest with *Debian* that would have zero virtualization overhead and provide all the functionality I needed. Best of all I'd be able to customize the firewall/router because it is just a *Linux* machine!

So I made a list of features I had been using on *pfSense* and *VyOS* to see what I'd have to implement.

Goals

- Basic *Linux* install in *LXC* guest
- Firewall protection of my local network from the Internet
- VLAN separation for added isolation between my subnets and the Internet
- DHCP to provide each subnet with IPv4 address assignment and local DNS resolution
- Recursive DNS for added security, privacy and removing reliance on external entities
- IPv6 stack support (DHCPv6, Router Advertisements (NDP), Prefix Delegation)
- *Wireguard* support

Extras

- Create bonded Ethernet interface to remove bandwidth bottleneck between router and switch
- Add intrusion prevension system like Snort
- Add bandwidth monitor and graphing

Revision #11
Created 2021-03-30 18:54:04 UTC by dustin@swigg.net
Updated 2021-12-02 07:48:10 UTC by dustin@swigg.net