Setting Up a Secure Network Using pfSense

Tyler Thompson

I've been working in IT for a number of years and my knowledge in networking mainly comes from work experience and self-teaching, so I'm really only scratching the surface of the subject with this paper. There are many different types of networking devices that can be used as a firewall/routing solution, but I set up my home network with pfSense. When I began looking for a new routing solution, I was looking for something that gave me full control of the security and traffic flow of my network. I chose pfSense because it is capable of providing business-grade network management for free and it was the best I could find after trying many others.

What is pfSense?

pfSense is an open-source firewall/router operating system based on FreeBSD. It can be installed on a normal consumer-grade computer, such as a desktop. It provides terminal and web interfaces to the user and is capable of providing services normally only found on business-grade networking devices. See the links below for more details.

https://www.pfsense.org/
https://en.wikipedia.org/wiki/PfSense

Basic Home Network

Figure 1 shows the network topology of a typical home network. This diagram divides all the "moving parts" of the network into separate devices. In practice, it is typical for some of these devices to be combined into one device. For example, newer modems Comcast supplies have the modem, firewall, router, switch, and wifi access point all in one device. These combination type devices are also typically slower since the resources of the device are being used to run many functions. If you have ever had a Comcast modem, you know how terrible they are.

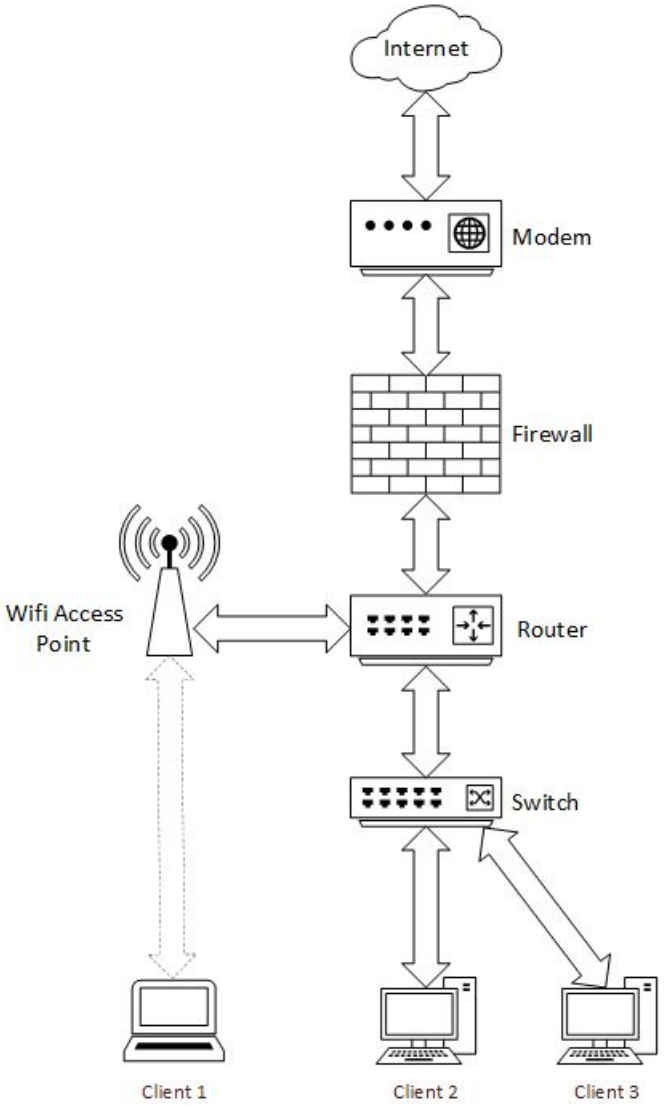

Figure 1 - Typical Home Network Topology

In this setup, pfSense would act as both the router and the firewall. Since the wifi access point will provide the same functionality as the switch, choosing one over the other is optional.

Multi-Site Network

For this paper, I will be using my home network setup as an example of a basic multi-site network. This is not the only way to do multiple sites, but it works well for my setup and provides a high level of security for my server network. Figure 2 shows the network topology of my home network, including its link to my apartment. Alpha Site is my parent's house in Port

Huron, MI where I have an additional Server Network running. Beta Site is my apartment in Kalamazoo, MI and is a typical setup and previously discussed.

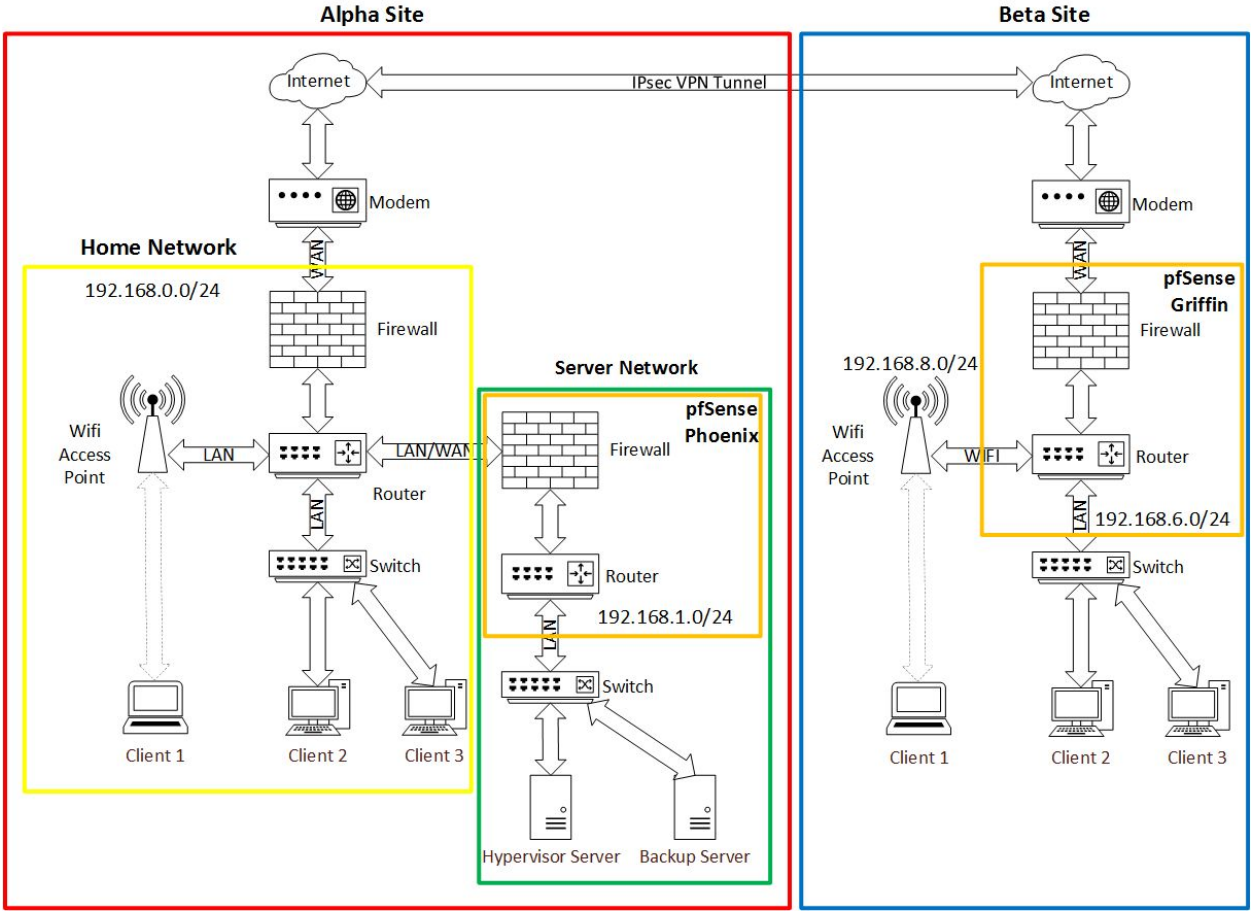

Figure 2 - Multi-Site Network Topology

Notice that there are two routers at Alpha Site. This is not necessary, but I set it up this way to create a security barrier (firewall) between the Server Network and the Home Network. This kind of setup could be used for a small data center, where the Home Network would be the office and the Server Network the server room. It also allows me to perform maintenance on the Server Network without disturbing the Home Network.

The two pfSense router/firewalls are named Phoenix and Griffin, where Phoenix is at Alpha Site and Griffin is at Beta Site. These two routers are linked through an encrypted IPSec VPN tunnel. This means that both sites see the network at the other site as if it were a local network. It allows me to access all devices at both sites from any other device at either site, securely.

At Alpha Site, the Home Network is all run on a single consumer-grade routing device that isn't designed to handle complex routing. Because of this, the Phoenix router is set up as a Demilitarized Zone (DMZ) on the Home Network. This means that all traffic originating from outside Alpha Site will be forwarded to Phoenix, allowing Phoenix to act as the main firewall for outside traffic. This is ideal for many reasons, Phoenix now has full port forwarding/NAT capabilities at Alpha Site, it can detect and log outside attacks, and it takes the load off the consumer-grade router running the Home Network. Essentially, the DMZ forces Phoenix to be the "front door" for network traffic at Alpha Site.

Hardware

Most consumer-grade routers these days have single-core processors and less than 1GB of memory. Since the routers function if fairly basic, there isn't much need for a lot of hardware resources. However, as features are added, the need for more hardware increases. I designed my routers hardware to be completely overkill so I could do as much with them as a business could with theirs. CPU: AMD Sempron Dual Core, Memory: 4GB, Disk: 256 SSD. With the extra resources, running services, such as ntop traffic monitor, doesn't impact the performance of the internet connection.

In order to build a router out of consumer-grade desktop parts, you will need at least two network ports. Typically desktop motherboards come with an on-board network port, so you will have to add at least one more through a PCI card. Figure 3 shows an example of a 1GB ethernet PCI network card. You can get them with multiple ports as well, or even with an alternate transmission method, such as fiber. The one you pick should be based on the needs of your network. At a minimum, you need one connection to run the Wide Area Network (WAN) and one to run the Local Area Network (LAN) interfaces.

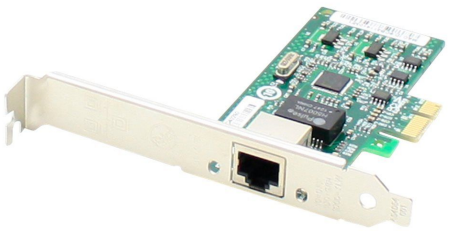

Figure 3 - Network PCI Card

In my setup, I use the on-board network port to run the WAN and an added PCI card to run the LAN. You could also use multiple cards or cards with multiple ports if you wanted to set up bridging or load balancing. I am actually using both of these methods at Alpha and Beta site, however, it isn't important to understanding a basic hardware setup.

Installing pfSense

Installing pfSense is just as easy as installing Windows or Ubuntu. You download an iso from pfSense, burn it to a USB drive and then boot your router to the USB. To get going quickly, you could just through an extra PCI network card in a standard desktop, boot to your pfSense USB drive, and your good to go. pfSense has a pretty good getting started guide.

https://www.pfsense.org/getting-started/

VPN Tunneling

There are different forms of Virtual Private Network (VPN) protocols and software, but the concept is all the same. You are on the outside of a firewall and you want to access the network behind it through a secure connection. VPN will encrypt all data sent between a server and a client, no matter what the data is or where it's going. Usually, this is referred to as a tunnel. All the data sent from the client has to go through the tunnel to the server before arriving at its destination.

pfSense provides multiple VPN services that are intended for slightly different configurations. I am using two of them, IPSec and OpenVPN. IPSec is used as a persistent link between Phoenix and Griffin while OpenVPN is used as a temporary link between Phoenix and a remote client. I use OpenVPN when I need to connect to my network from a random computer outside of my network.

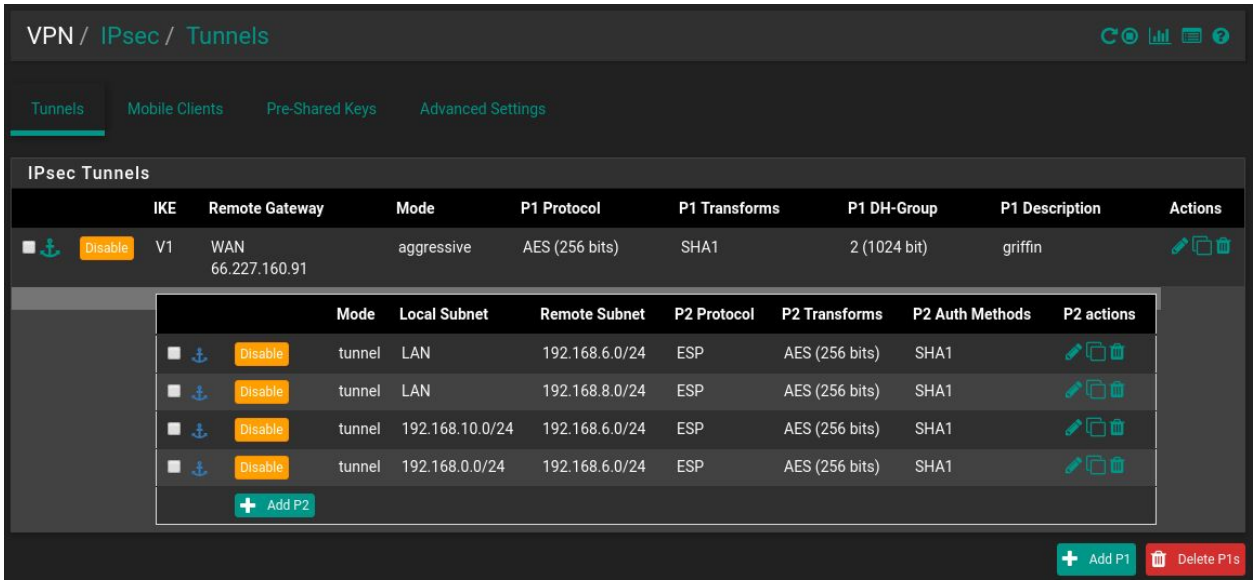

Figure 4 - Phoenix IPSec VPN Configuration

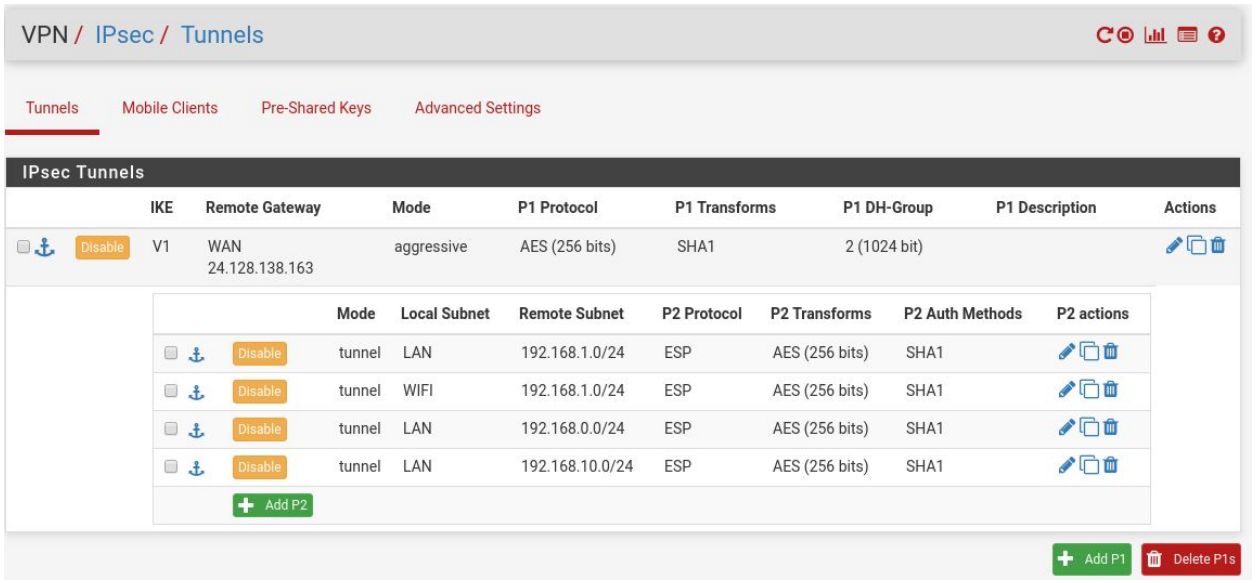

Figure 5 - Griffin IPSec VPN Configuration

Figure 4 shows the IPSec VPN configuration on Phoenix. This is the configuration Phoenix needs in order to talk to Griffin over the internet. Notice the Remote Gateway is set to 66.227.160.91. This is the public IP address of Beta Site, or the IP address obtained by the modem from the internet service provider (ISP). Similarly in figure 5, Griffin is set up with the Remote Gateway set to the public IP address of Alpha Site. In IPSec this is known as the Phase 1 configuration and is responsible for the transmission of data between the two routers (Transport Mode). Transport Mode will encrypt the entire payload with no regard for securing routing information.

IPSec then has a Phase 2 configuration that is responsible for routing (Tunneling Mode). Tunneling Mode encrypts data packets, so routing information is also encrypted. With the combination of Phase 1 and 2 protocols, the entire data stream is encrypted. My configuration for Phase 2 can be seen in figures 4 and 5. Basically, this grants access to a remote subnet to access a local subnet. This configuration will not bypass the firewall, so we will have to configure the firewall to allow access as well. More information on IPSec can be found below.

https://en.wikipedia.org/wiki/IPsec
https://docs.netgate.com/pfsense/en/latest/vpn/ipsec/configuring-a-site-to-site-ipsec-vpn.html

Static Routes

Static routes are important when you have multiple routers and want devices on different subnets to be able to talk to each other. It is also important to securing your network and understanding who has access to what.

In my setup, I want anyone on the Home Network to be able to access the Server Network. The only problem is the Home Network doesn't know what to do with traffic addressed to the Server Network. This is where a static route comes in. On the Home Network, a static route is created to direct traffic destined for the Server Network through Phoenix. For example, the Home Network has the static route 192.168.1.0/24 -> 192.168.0.2 where 192.168.0.2 is the WAN IP address of Phoenix and 192.168.1.0/24 is the Server Network subnet. All clients on the 192.168.0.0/24 subnet would then be able to access the 192.168.1.0/24 subnet.

Additionally, two more static routes are set up to route data between Alpha and Beta Site. The IPSec configuration handles the tunnel between the two sites, but in order for data to actually be routed from one to the other, a static route has to be put in each router. Figure 6 shows the static route on Phoenix that routes data destined for the 192.168.6.0/24 subnet through the WAN address of Griffin (192.168.6.1). Similarly, figure 7 shows the static route on Griffin that routes data destined for the 192.168.1.0/24 subnet through Phoenix (192.168.1.1).

https://en.wikipedia.org/wiki/Static_routing

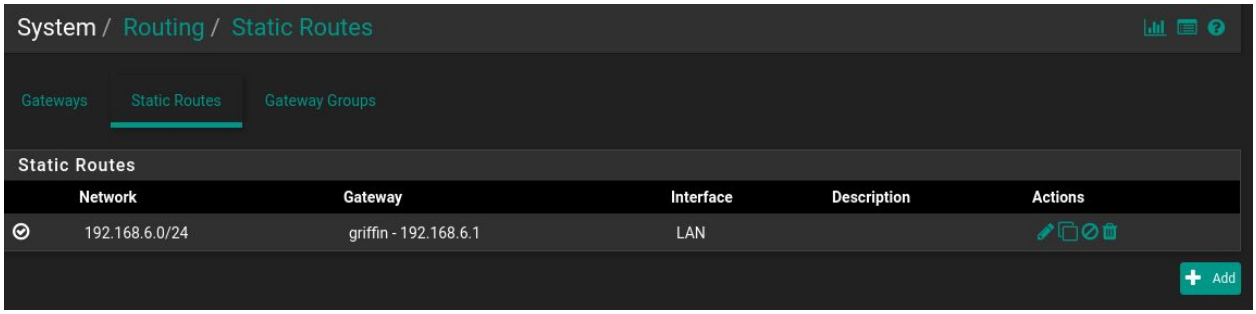

Figure 6 - Phoenix Static Route for Griffin Access

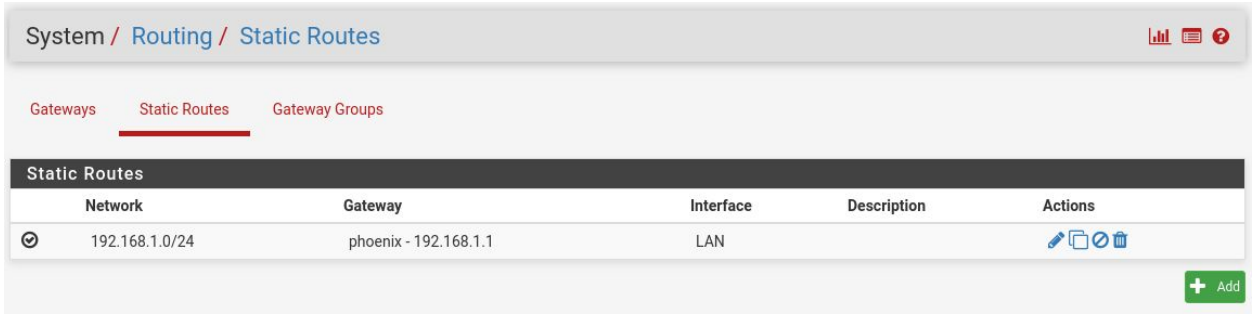

Figure 7 - Griffin Static Route for Phoenix Access

Firewall Rules

A firewall works like a bouncer at a club. Anyone can leave at any time, but if you want to come in, you have to go through the bouncer. Now look back at figure 2 and notice that Phoenix is hooked up to the Home Network in the same fashion as the switch and wifi access point. This means that as far as the Home Network is concerned, Phoenix is just another client. It also means that all the servers attached to Phoenix are seen as the same client and can access anything on the Home Network (The bouncer lets anyone leave at any time).

A firewall can enforce rules dependant on a network interface. Meaning that you can have different rules for your WAN interface vs your LAN interface. Typically all traffic is allowed between clients on a LAN interface and the main firewall in on the WAN interface. In most consumer-grade routers, the firewall only supports the WAN interface.

Because of the static route I setup, clients on the Home Network can access clients on the Server Network, however, they have to go through the firewall on Phoenix (The Bouncer). I'm not concerned with security between the Home and Server Networks, so I allow all traffic from 192.168.0.0/24 to access all subnets on the Server Network as seen in Figure 8. Optionally you

could create this rule with restrictions on which clients can have access to what servers and on what ports.

Figure 8 - Phoenix Firewall Subnet Hole

Similarly, these rules need to be set up between the subnets at Beta Site and the Server Network. IPSec and static routes allow the data to be routed between the two sites, but it still has to go through the bouncer (firewall).

If you're running a typical home network setup similar to Figure 1 and you want to host a server, it is likely you want to set up Port Forwarding, NAT, or just open a port in the firewall. Port Forwarding and NAT are not the same things, but they can produce the same results. See the links below for more information on the two.

Opening a port on the WAN interface of your router is about the most dangerous thing you can do, so I don't recommend it if you aren't sure what you're doing. When you open a port, it allows any data originating from outside of your network, destined for that port, through. This would be like a bouncer at a club letting anyone named Tom in, without any regard for who they actually are or where they are going. Port Forwarding or NAT is much safer and preferred over just opening ports on a WAN interface.

Port Forwarding requires a port to be opened in the firewall, but the port isn't just freely open. All traffic on the port is re-routed to a specified address on the LAN network. In my setup, I am running a GitLab virtual server on the hypervisor at Alpha Site. I want GitLab to be accessible from anywhere on the internet, so I set up a Port Forwarding rule to redirect all traffic on port 80 and 443 to the GitLab server (192.168.1.103). This means any traffic originating from the outside of Alpha and Beta Site, destined for port 80 or 443, will be redirected to GitLab. Figure 9 shows this configuration on Phoenix.

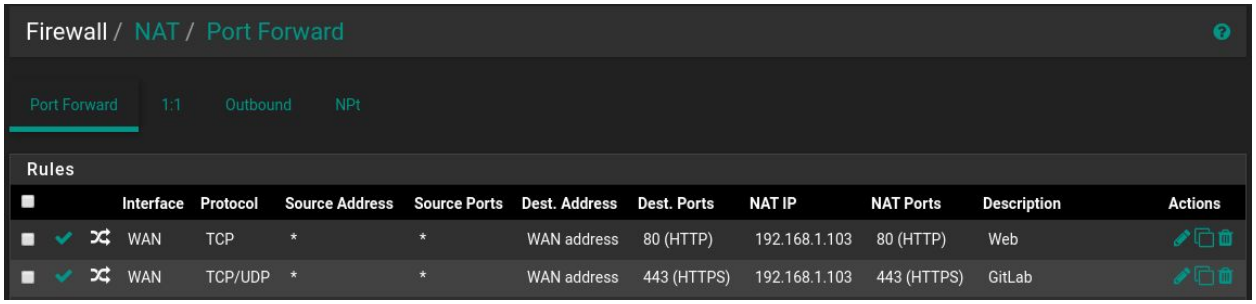

Figure 9 - Phoenix Port Forwarding Record

Using Port Forwarding does not mean you are secure. It just means the security for that port on the network is now the responsibility of the server you are forwarding the data to. In my setup, GitLab is responsible for the security of all data on ports 80 and 443 (http and https). I handle this by re-directing all traffic on port 80 to 443 and using a valid SSL cert on port 443 for https. The web site GitLab provides on port 443 defaults to a login page. The SSL cert encrypts the connection between a web browser and GitLab, so when you login, your credentials are not transmitted in clear text across the internet for everyone to see.

In summary, if you are going to open a hole in your firewall, you better be sure there is another device handling the security for that port. Below are some resources for understanding and setting up NAT/Port Forwarding.

https://en.wikipedia.org/wiki/Network_address_translation
https://en.wikipedia.org/wiki/Port_forwarding
https://docs.netgate.com/pfsense/en/latest/nat/forwarding-ports-with-pfsense.html

Traffic Monitoring

One of the features pfSense contains is the ability to install additional software packages through the same package manager that's used by FreeBSD. I installed ntopng on both Phoenix and Griffin which is just one of the tools made by ntop (https://www.ntop.org/). This is the real reason I designed the hardware of the routers to have additional memory and disk space. I set up ntopng to save a year of network traffic information to disk. This way I can analyze trends over time and keep a record of potential attacks.

ntopng has a lot of features and different ways of looking at your traffic to help you understand what is going on on your network. The easiest way to look at live data streams and active connections is through the Active Flows view.

Figure 10 shows the top active flows on the WAN interface of Phoenix. Active Flows are recent data streams that are being allowed through the firewall. This is just one of the basic features of ntopng, but it is good for understanding the top talkers on your network. In Figure 10 you can see that the biggest talker on Phoenix's WAN port is the IPSec tunnel to Griffin. This is expected since the tunnel requires a persistent connection, there is always overhead traffic even when it's not in use.

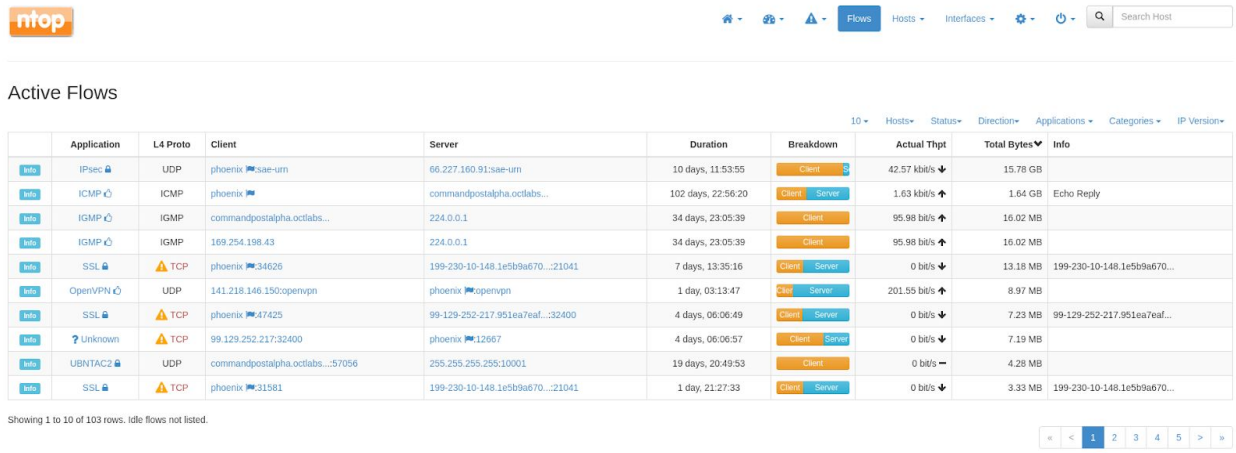

Figure 10 - ntopng Active Flows

Figure 11 shows an alert ntopng picked up about a flood attack. This was an attack I did myself in order to see how ntopng recorded it. I wrote a script to create 25 threads that all just tried to access the GitLab web site at the same time. ntopng picked it up right away.

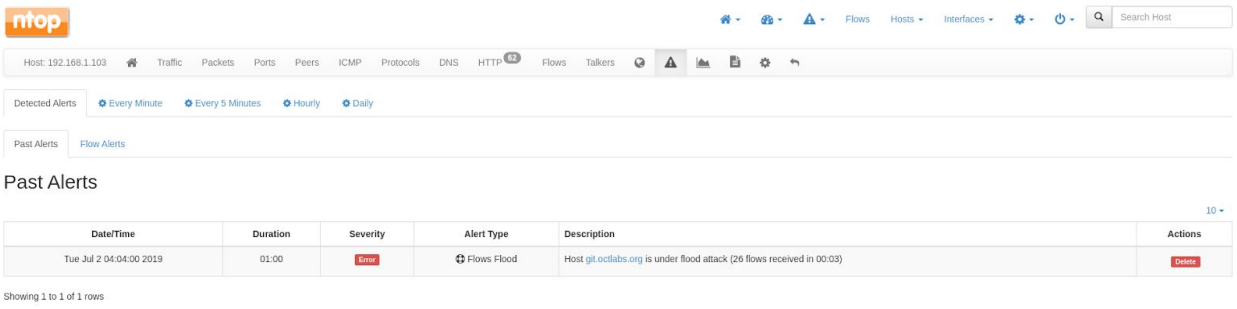

Figure 11 - ntopng Flood Attack Warning

ntopng is capable of a lot more than I know how to use, so if you'd like to learn more, see the links below.

https://www.ntop.org/products/traffic-analysis/ntop/
https://techexpert.tips/pfsense/ntopng-installation-pfsense/

Blocking an IP Address

If you start monitoring your internet traffic you will probably get to a point where you want to totally block connections to a certain IP address. In order to fully block a connection, two firewall rules need to be set up. The first one will block all incoming data on the WAN interface for the specified IP address (see Figure 12). The second one will block all outgoing data from the LAN interface to the specified IP address (see Figure 13). Figures 12 and 13 show an example I configured on Griffin to block a known ad server.

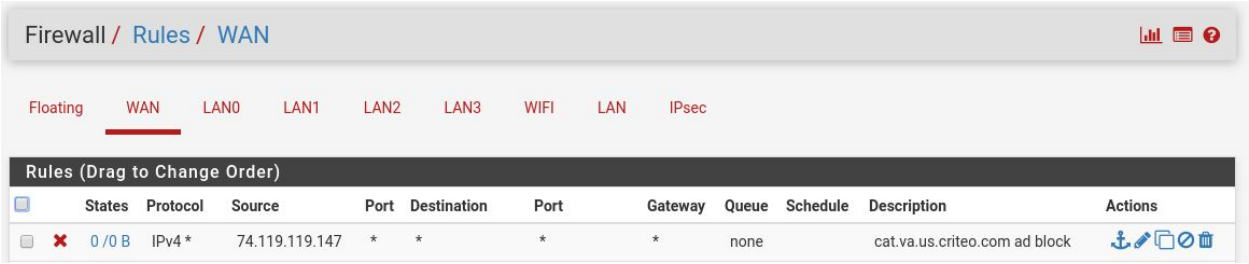

Figure 12 - Firewall Rule WAN

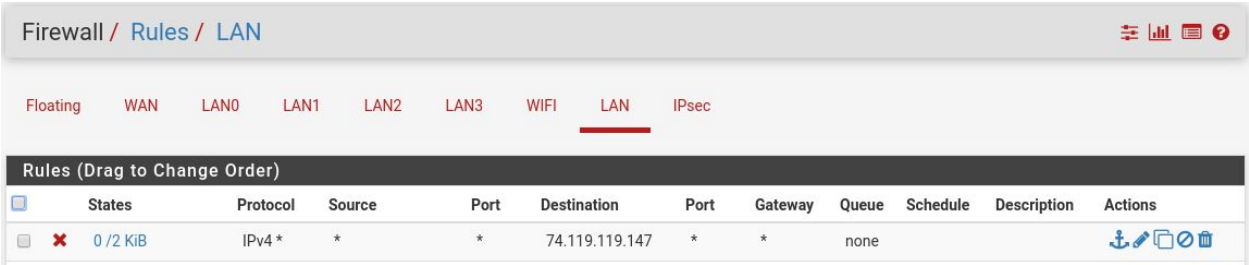

Figure 13 - Firewall Rule LAN

Ad Blocking

Blocking ads in your web browser, smart TV, Android Apps, and more can be done at the routing level. All ads pull their content from a remote server with a public IP address. Blocking

the public IP of the server an ad is pulling its content from will prevent the ad from showing. This isn't quite enough though. Ad companies are aware some of their servers may not be accessible from all locations in the world and make their ads pull from a list of servers that could be consistently changing. Attempting to block all of the IP addresses required to effectively removes any ads would be difficult to do by hand.

Instead of trying to block ads manually, the pfSense community came up with pfBlockerNG. pfBlockerNG is an extra software package (like ntopng) that is installable on pfSense. It automatically downloads lists of IP addresses that are known ad or malware sources and automatically blocks traffic to them. The list of blocked IP addresses is usually referred to as "the blacklist". pfBlockerNG updates the blacklist from configured feeds. Each feed is a list of IP addresses provided by a community or organization which they believe to be malicious. Figure 14 shows pfBlockerNG on Griffin where I configured a few automatic feeds for the blacklist

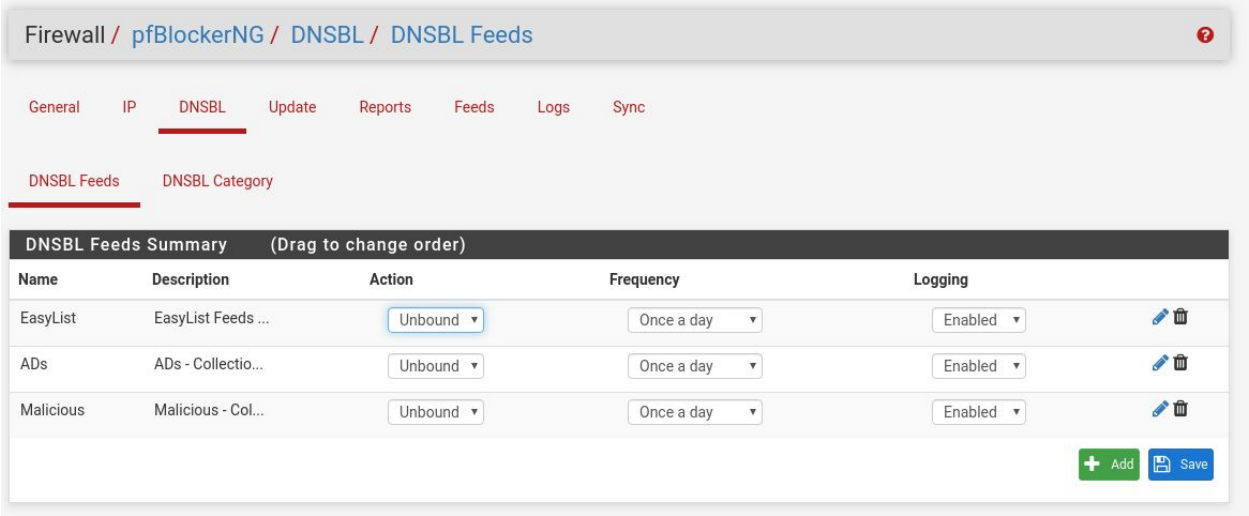

Figure 14 - pfBlockerNG Feed Summary

Using tools like ntopng and pfBlockerNG is really where you get your moneys worth when building a pfSense router. pfSense also has a good community forum and documentation, so it is a good place to start if you're trying to learn more about networking.