TCP Attacks (CS 915) Post-Lab Assignment Report

Hanzhi Zhang - 5525549

1. Attack 1: SYN flooding attack. Describe the results of the attack with screenshots and explain why the attack works as you observe.

Task 1A: Launching the SYN flooding attack in C

```
root@0d4b62047ca5:/# netstat -nat | grep SYN_RECV | wc -l
97
```

```
root@VM:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

The attack is successful, the VM is not able to telnet the server.

Task 1B: Launch SYN flooding attack using Python

```
root@0d4b62047ca5:/# netstat -tna | grep SYN_RECV | wc -l
92
root@0d4b62047ca5:/# netstat -tna | grep SYN_RECV | wc -l
97
root@0d4b62047ca5:/# netstat -tna | grep SYN_RECV | wc -l
95
root@0d4b62047ca5:/# netstat -tna | grep SYN_RECV | wc -l
97
root@0d4b62047ca5:/# netstat -tna | grep SYN_RECV | wc -l
91
```

```
root@VM:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
0d4b62047ca5 login:
```

The Python program is not fast enough to win the legitimate telnet packet. We see there are open slots left in the connection queue, from time to time. The VM takes advantage of this and established connection with the server.

```
root@VM:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
```

Attack is successful with 7 instances of Python program running at the same time.

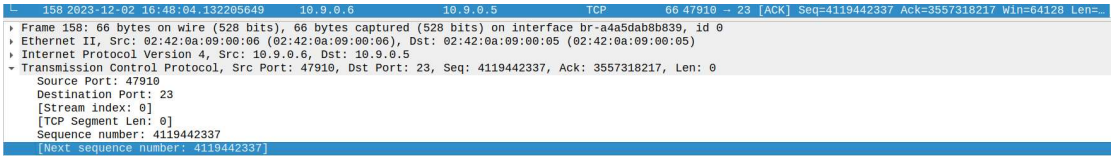
```
root@0d4b62047ca5:/# netstat -tna | grep SYN_RECV | wc -l
61
```

Attack also works with just one Python program but TCP queue size reduced to 80, we see the capacity now becomes 64 (compared with 97 for size 128), consistent with our knowledge that one fourth of the space in the backlog queue is reserved.

Both the C and the Python attack program intend to perform the SYN Flooding Attack by inserting a lot of TCB records to consume the space in server's queue. Both programs keep generating and sending out TCP packets with random (source) IP address. Because these source IP's are fake, no ACK packets will be received, the 3-way handshake will never be established, so all half-open connections will stay in the queue before they are removed after a number of retransmissions. If the attack program can send packets at a higher rate than these expired connections are removed from the queue, there will not be open slots in the queue to accept another machine's new request for connection.

2. Attack 2: TCP RESET attack. Describe the results with screenshots and explain how you modify the parameters in the code and why the attack works.

Task 2A: Launch TCP Reset Attack



Next sequence number for the last TCP packet is 4119442337.
Src: 10.9.0.6, Dst: 10.9.0.5, Source Port: 47910, Destination Port: 23.

```
ip  = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=47910, dport=23, seq=4119442337, flags="R")
```

```
seed@83624abbfa85:~$ Connection closed by foreign host.
root@7750849bbe0e:/#
```

Attack was successful, telnet connection has been closed.

Task 2B: Launching the attack automatically

```
[12/02/23]seed@VM:~/.../Lab 6$ ifconfig
br-a4a5dab8b839: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

Change the iface field in reset_auto.py accordingly.

```
seed@83624abbfa85:~$ hConnection closed by foreign host.
root@7750849bbe0e:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
83624abbfa85 login: Connection closed by foreign host.
root@7750849bbe0e:/#
```

With reset_auto.py running in the background, not only is the original telnet connection taken down, but every time the client tries to establish a new connection with the server the new connection is instantly attacked and closed too.

Using Wireshark on attacker machine to sniff the traffic, we can retrieve the destination port, source port and next sequence number needed to forge a TCP RST packet. We take

the last TCP packet from client to sever and spoof the next (also client to server), using the same src, dst and the “next” sequence, and we only do this once.

As for reset_auto.py, it keeps sniffing and spoofing RST packets using the TCP packets it sniffed. For each TCP packet from server to client captured, it “answers” with a RST where src and dst are reversed and the ack number is used as the new sequence.

In both cases, the server received an RST packet with the right sequence number it was expecting to get, it then immediately broke the connection.

3. Attack 3: TCP Session hijacking attack. Describe the results with screenshots and explain how you modify the parameters in the code and why the attack works.

Task 3A: Injecting a malicious command
We want to inject data in an established connection between user1 and victim, we know from previous attacks the src IP, dest IP and dest port for this session.

src = "10.9.0.6", dst = "10.9.0.5", dport = 23, iface='br-a4a5dab8b839'.

```
seed@83624abbfa85:~$ 1234567890
```

The telnet window freezes.

```
root@83624abbfa85:/# ls -l /tmp/
total 0
-rw-rw-r-- 1 seed seed 0 Dec  3 00:55 success
```

A new file called success is created under /tmp on the victim.

Task 3B: Session Hijacking with Reverse Shell

```
[12/02/23]seed@VM:~/.../Lab 6$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 51360
seed@83624abbfa85:~$
```

The attacker’s listening program entered the reverse shell.

Suppose the victim has received data up to the sequence number, x, the spoofed packet uses sequence number x+10, and stored in the victim’s buffer at position x+10, leaving 10 spaces. Once these spaces are filled (user1 types 10 characters), the victim considers everything received before (and including) the spoofed packet in order, then the injected touch command is executed, creating a new file on the victim’s machine.

The reverse shell command starts an interactive bash shell on the victim’s machine, and redirects its input and output device to a TCP connection to the attacker’s server. That is to say the shell runs on victim machine, but receives its input and send its output both to the attacker. Attacker then can have remote access to the victim’s machine.

4. Question: In the TCP session hijacking attack, the user's session window freezes after the TCP session is hijacked. Explain what's happening to the user's session.

By the time the injected command is triggered, the last packet that the victim considers part of its "normal" traffic is the spoofed packet with sequence number x+10. The next packet it receives is from user1, also with the sequence number x+10. The victim takes this packet as duplicate and discards it. User1 receives an ACK from victim, but since it was a "reply" to the spoofed TCP packet, both the sequence and ack number will be wrong for user1. User1 regards packets as out of order and send its packet with sequence number x+10 again. Now the victim and user1 are in a deadlock, user1 keeps receiving invalid ACK's and resending its packet; victim keeps receiving and dropping duplicated packets while resending its ACK. From user1's perspective, the program freezes.