**ACS 56000: Lab2**

-By Vikramadithya Ivaturi

**Task 1.1A**

**Code:**

#!/usr/bin/python3

from scapy.all import \*

print("SNIFFING PACKETS.........")

def print\_pkt(pkt):

pkt.show()

print("\n")

print("----------------------------------")

pkt = sniff(iface='br-a77fad32c8dc', filter='icmp', prn=print\_pkt)

**Output:**

Running the code with root privileges

Text, letter

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application

Description automatically generated with medium confidenceText

Description automatically generated with low confidence

Graphical user interface

Description automatically generated with low confidence

Text

Description automatically generated with low confidence

A picture containing text

Description automatically generated

Running the code without root privileges

Text, letter

Description automatically generated

The process of sniffing packets requires root privileges, so, it cannot run on user’s ID.

**Task 1.1B**

1. Capture only the ICMP packet

The previous program we are capturing only ICMP packets, so the same code ca be considered.

1. Capture any TCP packet that comes from a particular IP and with a destination port number 23.

In the below picture, we are sending a TCP packet to 10.9.0.6 (machine B) from machine A saying ‘Hello!’

Graphical user interface, text, application

Description automatically generated

In the below picture, we are listening on the port 23 in machine B, and got the message ‘Hello!’.

Graphical user interface, text, application, website

Description automatically generated

Below screenshot gives the listening output on the bridge.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

1. Capture packets comes from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to.

**Code:**

#!/usr/bin/python3

from scapy.all import \*

print("SNIFFING PACKETS.........")

def print\_pkt(pkt):

#print("Source IP:", pkt[IP].src)

#print("Destination IP:", pkt[IP].dst)

#print("Protocol:", pkt[IP].proto)

pkt.show()

print("\n")

print("----------------------------------")

pkt = sniff(iface='br-4e745a9aba02', filter='net 128.230.0.0/16', prn=print\_pkt)

**Output:**

Pinged one of the IPs in the subnet to test the program.

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**1.2. send a spoofed ICMP packet to an IP address**

**Code:**

#!/usr/bin/python3

from scapy.all import \*

print("SENDING SPOOFED ICMP PACKET.........")

ip = IP(src="8.8.4.4", dst="1.2.3.4")

icmp = ICMP()

pkt = ip/icmp

pkt.show()

send(pkt,verbose=0)

**Output:**

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**1.3. implement a traceroute functionality in python using scapy.**

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

a = IP()

a.dst = sys.argv[1]

ttl = 3

b = ICMP()

for i in range(201):

a.ttl = i+1

p = a/b

resp = sr1(p, timeout=2, verbose=0)

#print(resp[ICMP].type )

if resp is None:

print("No reply")

elif resp[ICMP].type == 0 :

print("%d hops away: " % (a.ttl), resp[IP].src)

print("Done", resp[IP].src)

else :

print("%d hops away: " % (a.ttl), resp[IP].src)

if resp[IP].src == a.dst:

break

**Output:**

Text

Description automatically generated

**1.4. Sniffing then spoofing**

**Code:**

#!/usr/bin/python3

from scapy.all import \*

def spoof\_pkt(pkt):

if ICMP in pkt and pkt[ICMP].type == 8:

print("Original Packet.........")

print("Source IP : ", pkt[IP].src)

print("Destination IP :", pkt[IP].dst)

ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl, ttl = 50)

icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)

data = pkt[Raw].load

newpkt = ip/icmp/data

print("Spoofed Packet.........")

print("Source IP : ", newpkt[IP].src)

print("Destination IP :", newpkt[IP].dst)

send(newpkt,verbose=0)

pkt = sniff(iface='br-4e745a9aba02', filter='icmp',prn=spoof\_pkt)

**Output:**

**Case-1: pinging 1.2.3.4**

We have executed the above code in the below terminal,

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Here, we are pinging a non-existent IP address that is out of our subnet. We have sent 6 packets from client, and we have received 6 packets from our program.

Text

Description automatically generated

**Case-2: pinging 10.9.0.99**

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

The reason for this error is due to the existence of the above IP in our current subnet. When we send a packet to an IP in the current subnet, the system looks up the IP in the ARP table and checks if the MAC address of the IP is already present, if not then, it requests for the MAC address, which is non-existent.

**Case-3: pinging 8.8.8.8**

Text

Description automatically generated

Text

Description automatically generated

In the above screenshot, the duplicate entries with 50 as it’s ttl is a response that has been sent from the program.