**Lab 2**

Sanchit Dass

# **Task 1.1A: Sniffing Packets**

This task involves the sniffing of packets. The packet needs to be intercepted and then printed on the console.

**Source code:**

*#!/usr/bin/python3*

*from scapy.all import \**

*def print\_pkt(pkt):*

*pkt.show()*

*pkt = sniff(iface="br-d053d20513b7", filter="icmp", prn=print\_pkt)*

I executed the above script using root privileges and was able to intercept a simple ICMP request/response pair.

* Sending a ping request from 10.9.0.6 to 10.9.0.5  
  Graphical user interface, text, application, chat or text message

  Description automatically generated
* Capturing these packets using the script:  
  Graphical user interface

  Description automatically generated  
    
  Graphical user interface

  Description automatically generated

Running the same program without root privileges gives the following error:

Text

Description automatically generated

This can be explained by the fact that the script needs to set the promiscuous mode which can only be done with administrator privileges.

# **Task 1.1B: Using BPF**

1. **Capture only the ICMP packet.**  
   This task has already been completed in 1.1A above.
2. **Capture any TCP packet that comes from a particular IP and with a destination port number 23.**Captured all telnet (port 23) packets sent from 10.9.0.5  
     
   **Source Code:***#!/usr/bin/python3  
     
   from scapy.all import \*  
     
   def print\_pkt(pkt):  
    pkt.show()  
     
   pkt = sniff(iface=”br-d053d20513b7”, filter=”tcp and src host 10.9.0.5 and dst port 23”, prn=print\_pkt)*  
   Sending the telnet request:  
   A screenshot of a computer

   Description automatically generated with medium confidence  
   Capturing the packet:  
   Text

   Description automatically generated
3. **Capture packets on a particular subnet.**  
   **Source code:***#!/usr/bin/python3  
     
   from scapy.all import \*  
     
   def print\_pkt(pkt):  
    pkt.show()  
     
   pkt = sniff(iface=”br-d053d20513b7”, filter=”net 128.230.0.0/16”, prn=print\_pkt)*  
     
   **Graphical user interface

   Description automatically generated**

# **Task 1.2: Spoofing ICMP packets**

This task involves spoofing an ICMP packet from a non-existent host (in this case - 1.2.2.1).

This screenshot below captures the spoofed packet received by 10.9.0.5:

**Graphical user interface, text, application, chat or text message

Description automatically generated  
  
Source code:**

*#!/usr/bin/python3*

*from scapy.all import \**

*ip = IP(src="1.2.2.1", dst="10.9.0.5")*

*icmp = ICMP()*

*pkt = ip/icmp*

*send(pkt)*

# **Task 1.3: Traceroute**

This task involves the creation of a traceroute program. This program estimates the number of hops between the host and destination using ICMP packets and manipulation of the TTL value.

Here is a screenshot of the route taken by my VM to reach 8.8.4.4 (Google DNS):

A screen shot of a computer

Description automatically generated with low confidence

**Source Code:**

*#!/usr/bin/env python3*

*from scapy.all import \**

*a = IP()*

*a.dst = sys.argv[1]*

*ttl = 1*

*while True:*

*a.ttl = ttl*

*b = ICMP()*

*p = a/b*

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

*if resp is None:*

*print("No reply")*

*break*

*elif resp[ICMP].type == 0 :*

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

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

*break*

*else :*

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

*ttl += 1*

# **Task 1.4: Sniffing-and-then-spoofing**

The final task combines the ideas of sniffing and spoofing to intercept an echo request and then send a spoofed reply. All the pings were sent from 10.9.0.5:

1. ping 1.2.3.4 (non-existent host on the internet)  
   A screenshot of a computer

   Description automatically generated with medium confidence  
   The host receives a reply from 1.2.3.4 even though it does not exist on the network!
2. ping 8.8.4.4 (google DNS)  
   Graphical user interface, text, application, chat or text message

   Description automatically generated  
   We receive two replies, one from the actual server and one spoofed reply (indicated by DUP).
3. ping 10.9.0.99 (non-existent host on the local network)  
   Graphical user interface, text, application, chat or text message

   Description automatically generated  
   The destination host is unreachable as the source host doesn’t have an entry for 10.9.0.99 in its ARP table. ARP (Address Resolution Protocol) is a link layer protocol which uses an ICMP broadcast message to identify hosts on the network. It essentially stores the IP to MAC address mapping for each host on the network. To successfully spoof this address, we need to poison the cache of the host machine first.

**Source Code:**

*#!/usr/bin/python3*

*from scapy.all import \**

*def spoof\_pkt(pkt):*

*if ICMP in pkt:*

*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, code=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)*

*data = pkt[ICMP].load*

*newpkt = ip/icmp/data*

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

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

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

*# newpkt.show()*

*send(newpkt,verbose=1)*

*pkt = sniff(iface='br-4177281c6d27', filter='icmp[icmptype] == icmp-echo and src host 10.9.0.5',prn=spoof\_pkt)*