LAB- 03

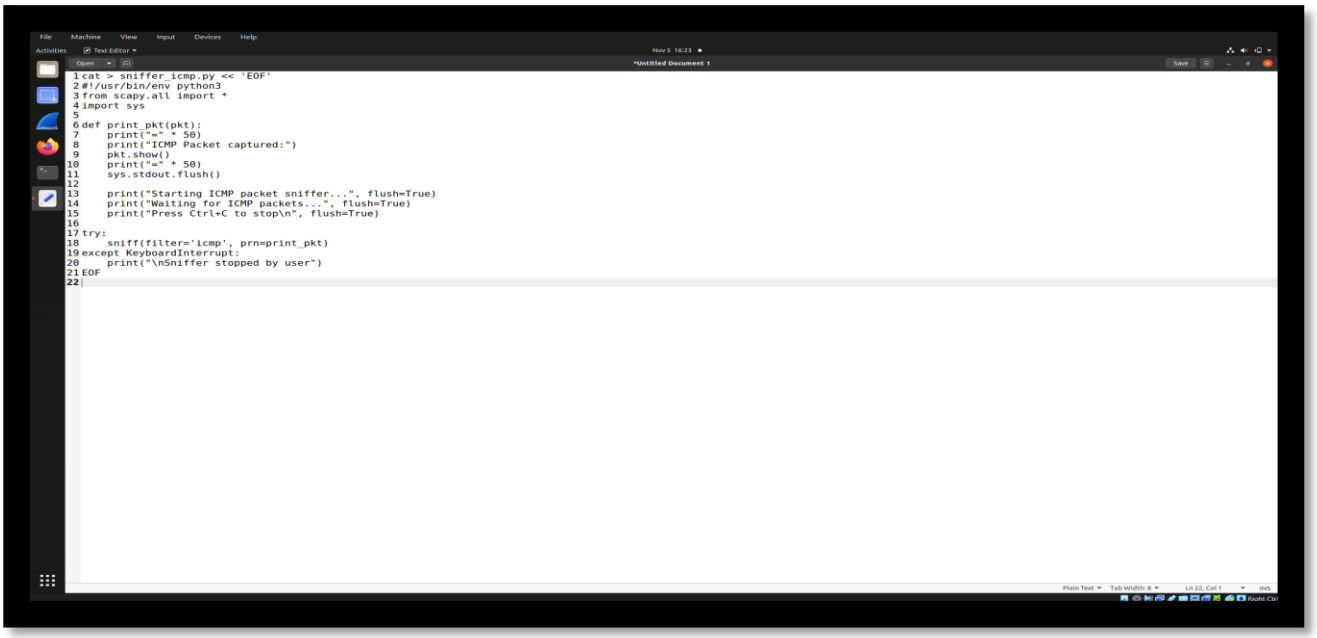
By :- Faraz Ahmed

Task 1.1

- Firstly, we use command “ifconfig” to find all interface in the network as highlighted in screenshot 1. It has enp0s3 (highlighted) which is our main network interface that has IP address 10.0.2.15.

Screenshot 1: Executing command “ifconfig” to find interface in the network.

- Now we will create sniffer_icmp.py file which will contain code shown in screenshot 2.

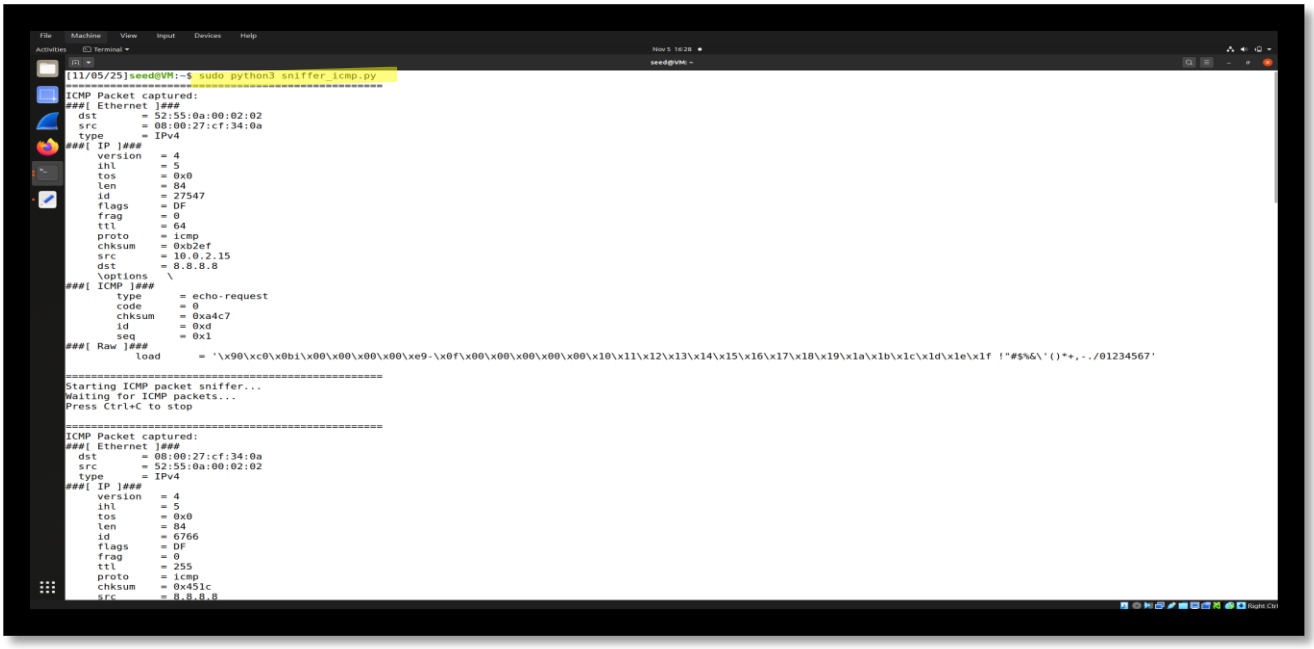


Screenshot 2: “Sniffer_icmp.py” python code which is newly created.

- Then we will execute command “sudo python3 sniffer_icmp.py” (highlighted in screenshot 4) which will start ICMP packet capture and then on another terminal and “ping -c 3 8.8.8.8” to start an ICMP ping as highlighted in screenshot 3.

Screenshot 3: Command “ping -c 3 8.8.8.8” to start an ICMP ping.

- After running ICMP packet, we can see some activity in ICMP packet capture as observed in screenshot 4.



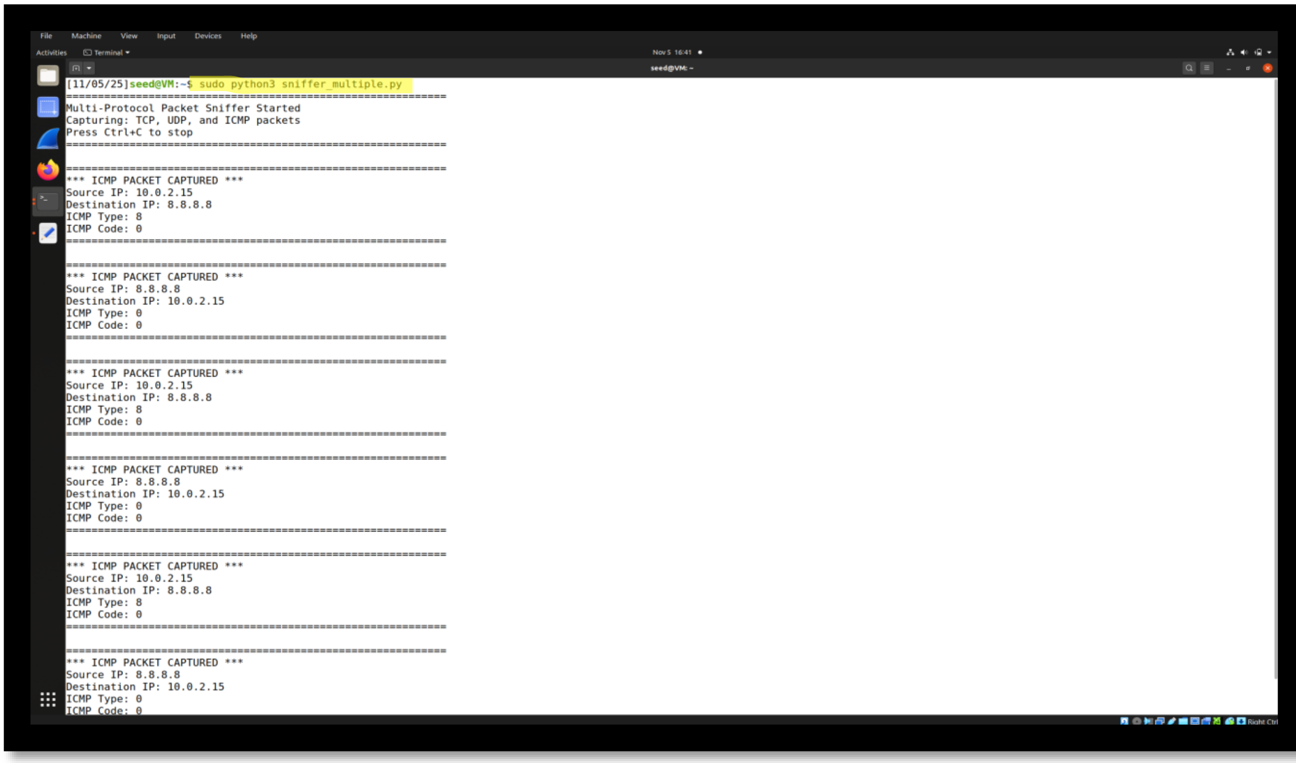
Screenshot 4: ICMP activity logged in ICMP packet capture.

- So, now we will capture three different packets from three different protocols like ICMP, TCP and UDP. So, for that we will use python code to create “sniffer_multiple.py” as observed in screenshot 5.

Screenshot 5: Python code in “sniffer_multiple.py”.

- Now, we will ping can of the three protocols one by one and observe the packets which are captured by running “sudo python3 sniffer_multiple.py”. First, we will run ICMP ping as highlighted in screenshot 6 and note all ICMP packets captured in screenshot 7.

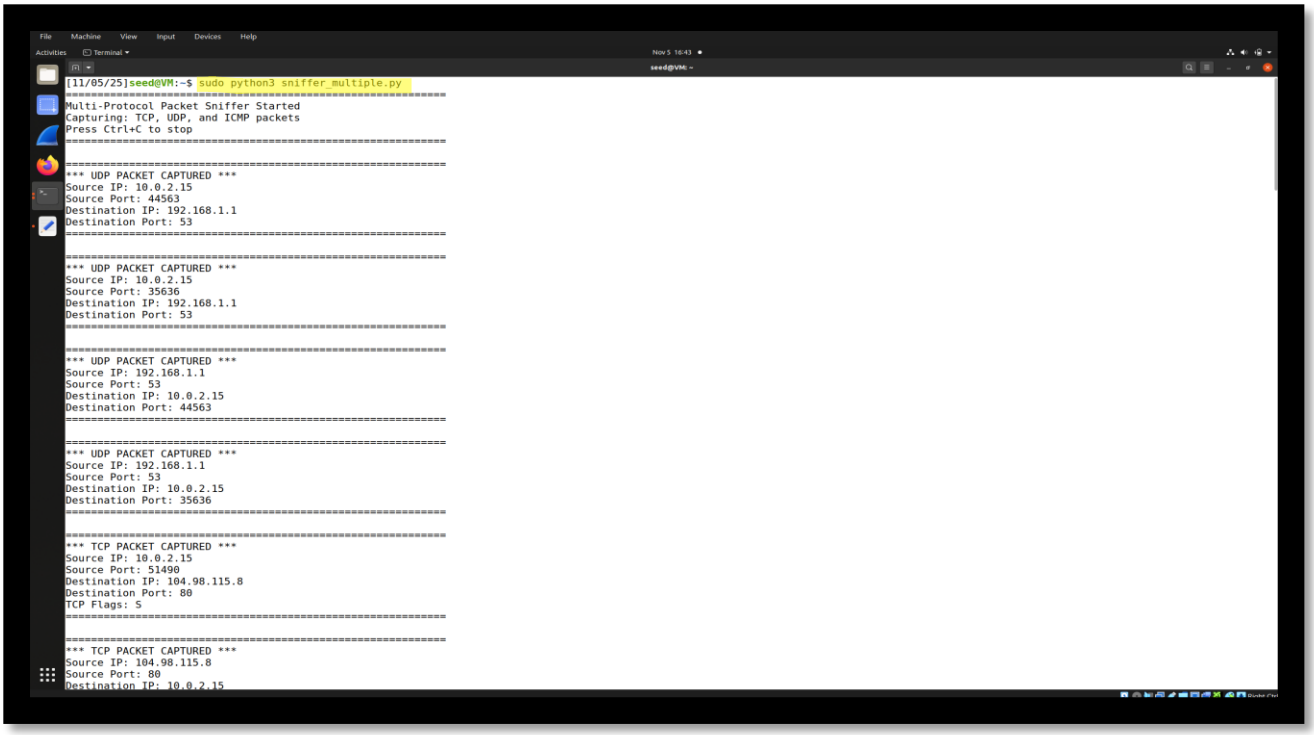
Screenshot 6: ICMP ping using “ping -c 3 8.8.8.8”.



<u>Screenshot 7: All captured ICMP packets from ICMP ping.</u>

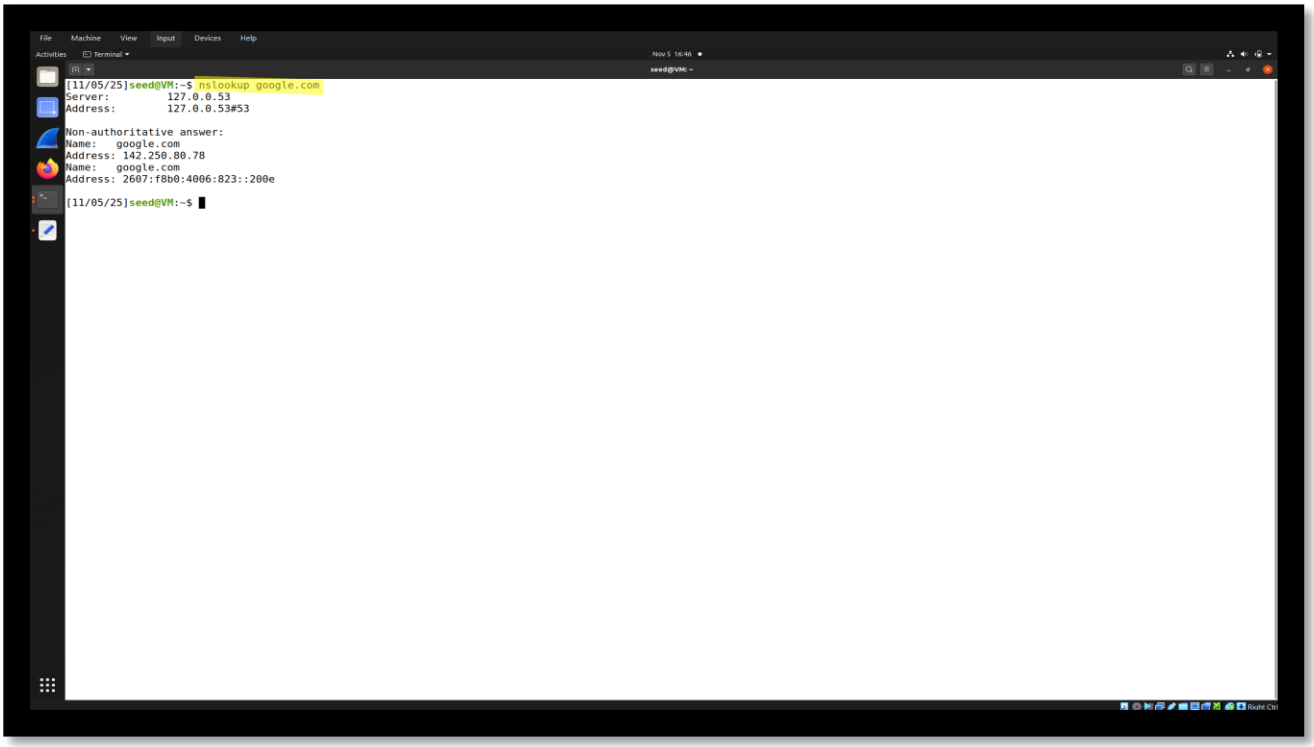
- Second, we will run TCP ping using "curl http://www.example.com" as highlighted in screenshot 8 and then observe all TCP packets captured in screenshot 9.

<u>Screenshot 8: TCP ping using "curl http://www.example.com".</u>

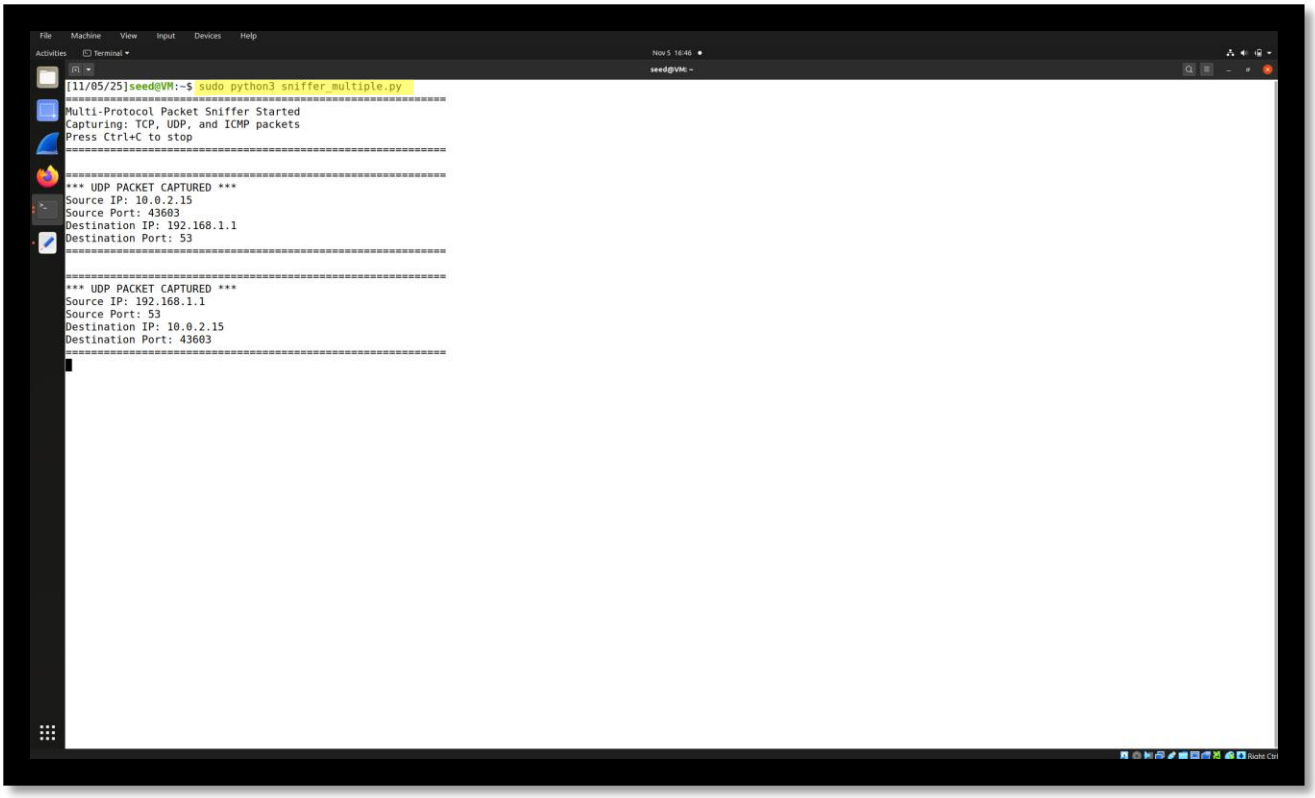


Screenshot 9: All captured TCP packets from TCP ping.

- Third, we will run UDP ping using “nslookup google.com” as highlighted in screenshot 10 and then observe all UDP packets captured in screenshot 11.



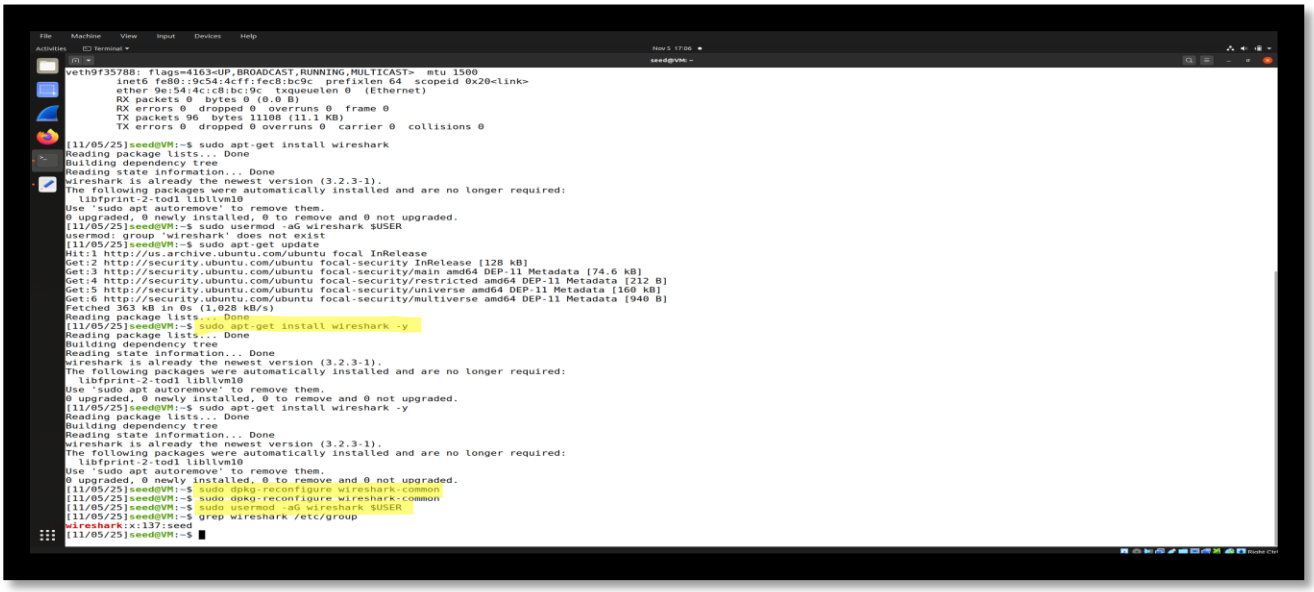
Screenshot 10: UDP ping using “nslookup google.com”.



Screenshot 11: All captured UDP packets from UDP ping.

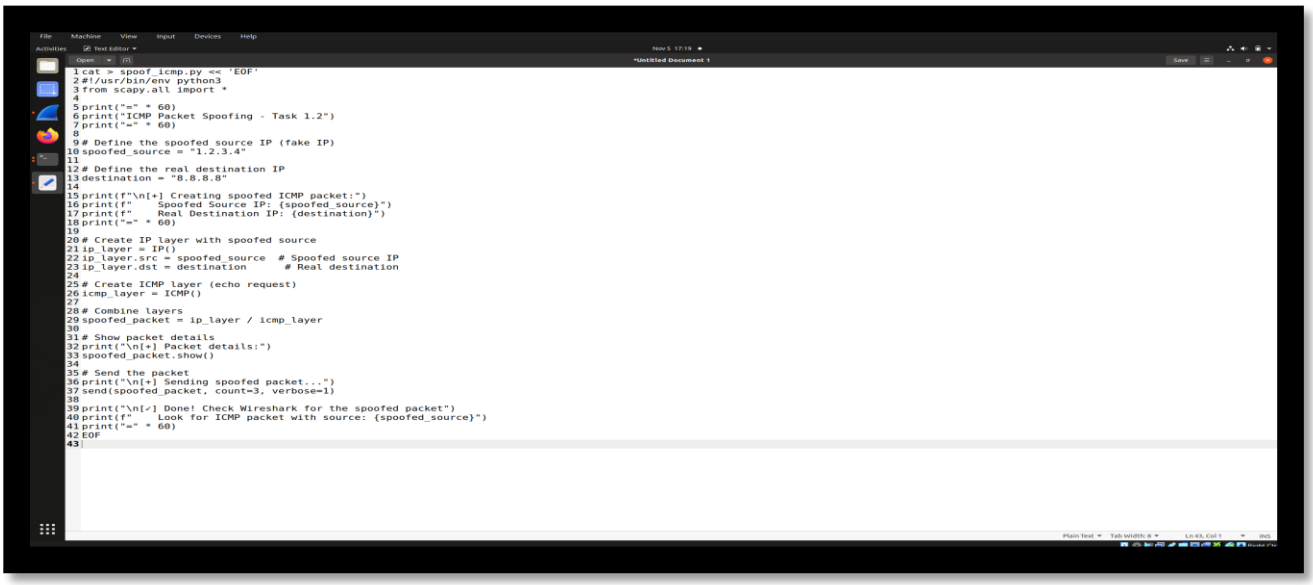
Task 1.2

- Firstly, we have to install wireshark using “sudo apt-get install wireshark” and then set it up using “sudo dpkg-reconfigure wireshark-common” and select YES and add user to wireshark group using “sudo usermod -aG wireshark $USER” as highlighted in screenshot 12.



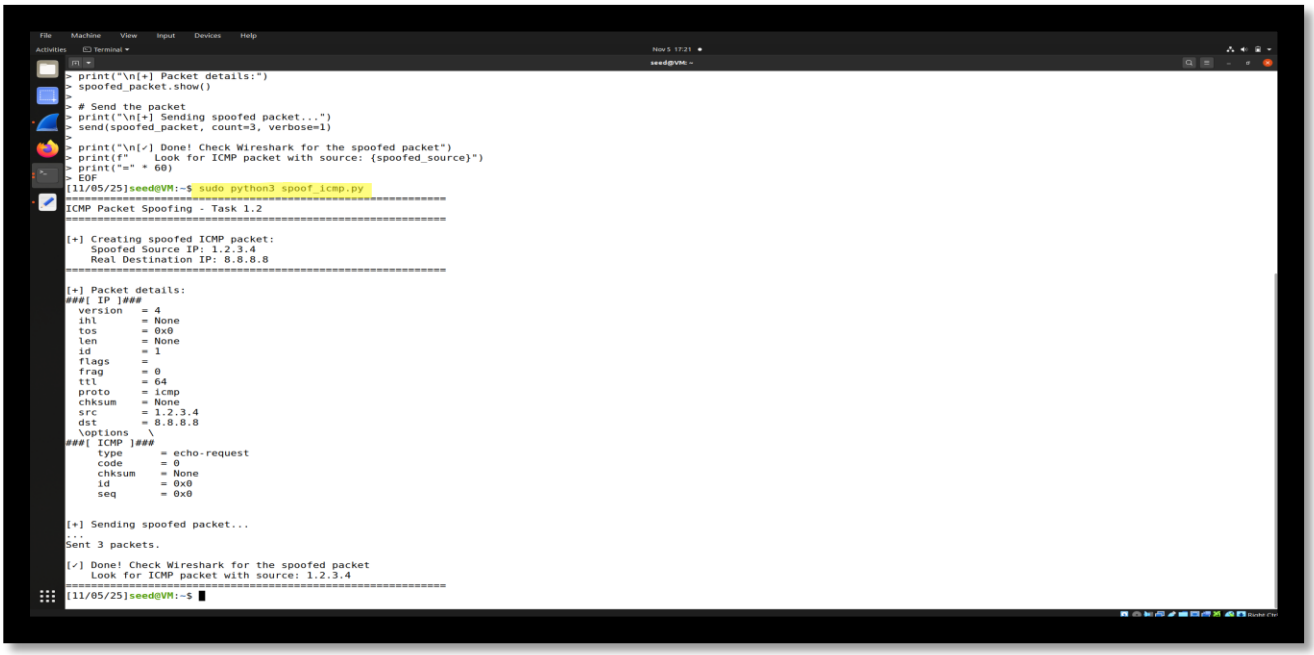
Screenshot 12: Commands used to install and setup Wireshark in the machine.

- Then we will create a new ICMP spoofing script of “spoof_icmp.py” which can be observed in screenshot 13.



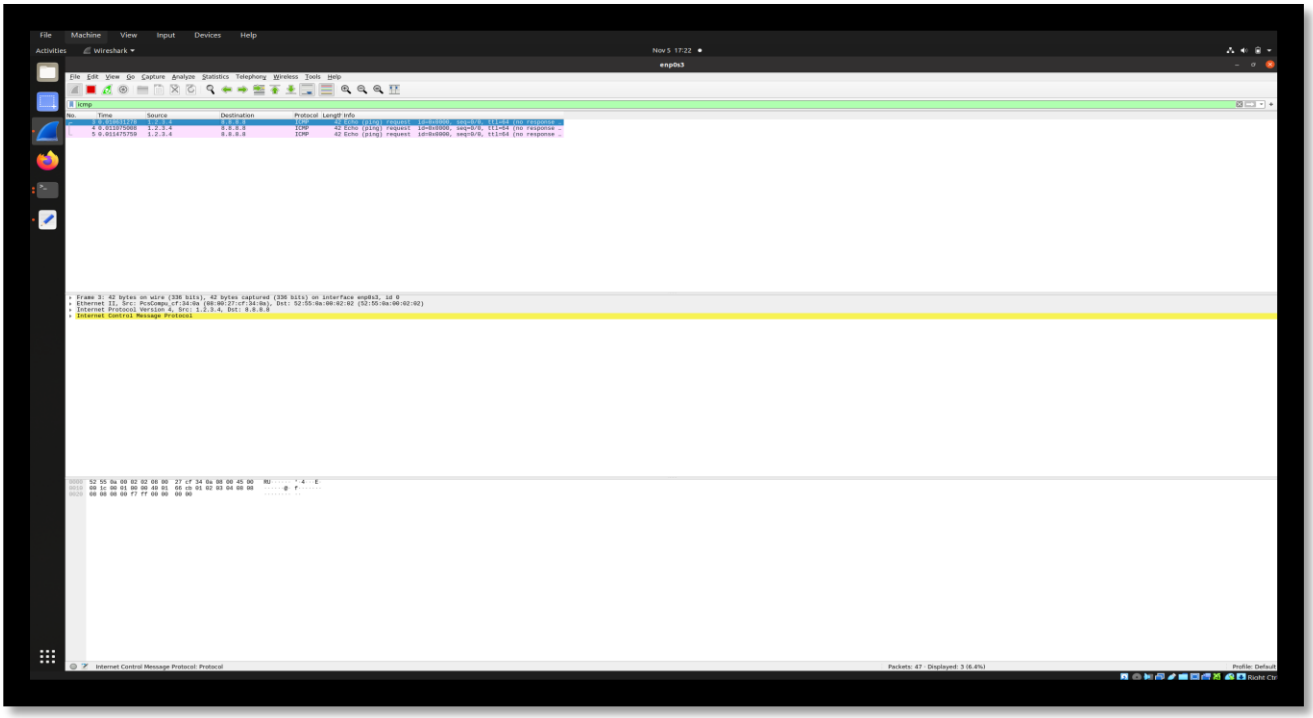
Screenshot 13: New ICMP spoofing script of “spoof_icmp.py”.

- Then we will run the spoofing script using “sudo python3 spoof_icmp.py” (highlighted in screenshot 14) which will send 3 packets as observed in screenshot 14.



Screenshot 14: Running “sudo python3 spoof_icmp.py” to run spoofing script.

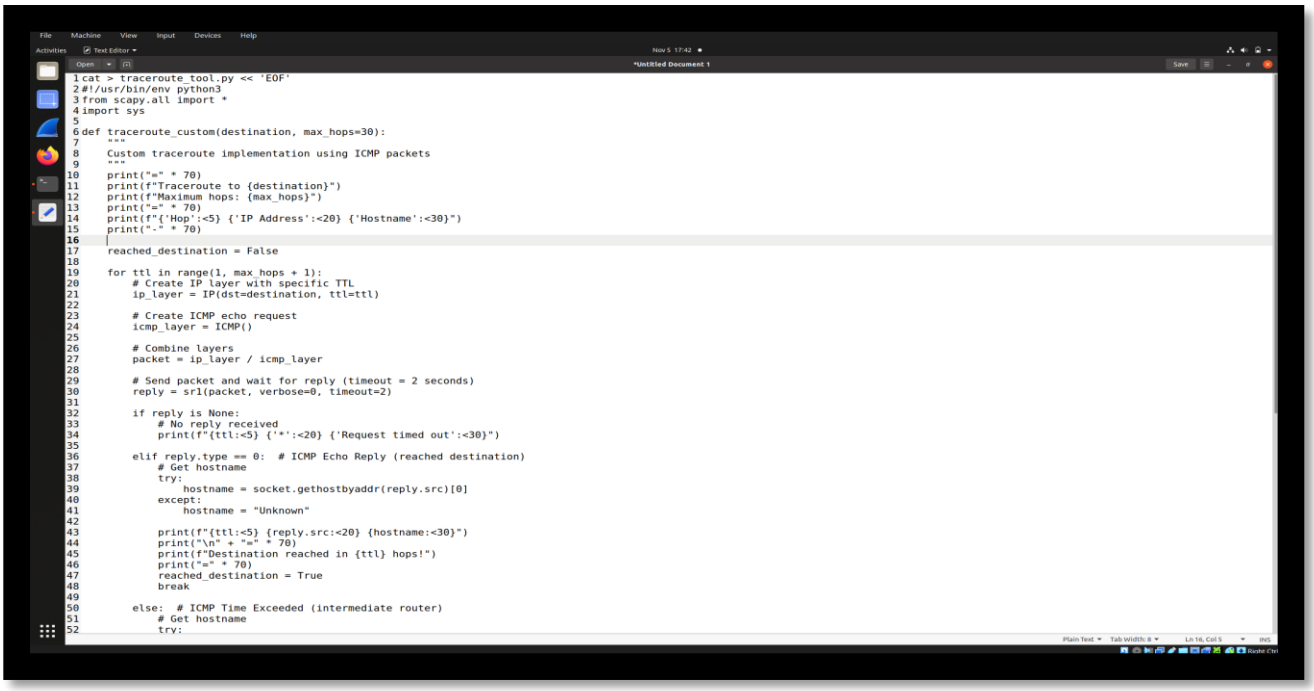
- Then we will open Wireshark using “sudo wireshark &” and observe all three generated packets as shown in screenshot 15.



Screenshot 15: All three generated packets in Wireshark.

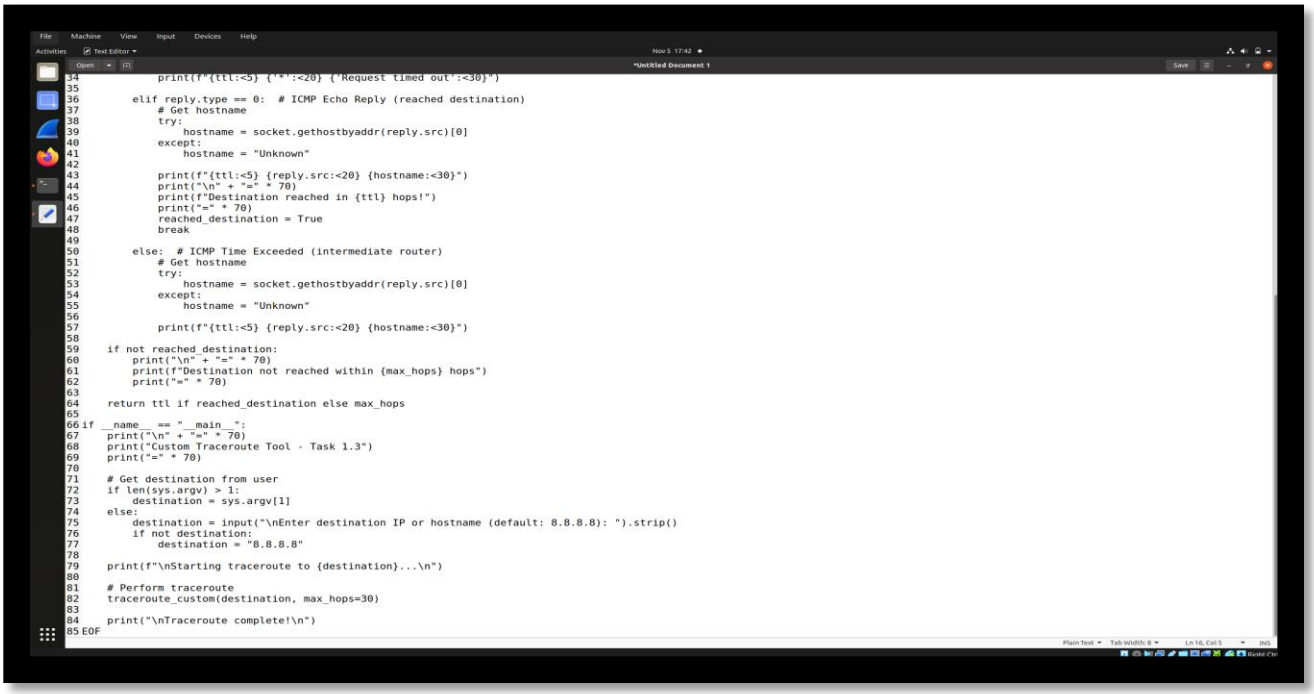
Task 1.3

- For this first, we will create an automated traceroute script “traceroute_tool.py” as observed in screenshot 16 and 17.



```
cat > traceroute_tool.py << 'EOF'
#!/usr/bin/env python3
from scapy.all import *
import sys

def traceroute_custom(destination, max_hops=30):
    """
    Custom traceroute implementation using ICMP packets
    """
    print("=" * 70)
    print(f"Traceroute to {destination}")
    print(f"Maximum hops: {max_hops}")
    print("=" * 70)
    print(f"{'Hop':<5} {'IP Address':<20} {'Hostname':<30}")
    print("-" * 70)

    reached_destination = False

    for ttl in range(1, max_hops + 1):
        # Create IP layer with specific TTL
        ip_layer = IP(dst=destination, ttl=ttl)

        # Create ICMP echo request
        icmp_layer = ICMP()

        # Combine layers
        packet = ip_layer / icmp_layer

        # Send packet and wait for reply (timeout = 2 seconds)
        reply = sr1(packet, verbose=0, timeout=2)

        if reply is None:
            # No reply received
            print(f"{ttl:<5} {'*':<20} {'Request timed out':<30}")

        elif reply.type == 0:  # ICMP Echo Reply (reached destination)
            # Get hostname
            try:
                hostname = socket.gethostbyaddr(reply.src)[0]
            except:
                hostname = "Unknown"

            print(f"{ttl:<5} {reply.src:<20} {hostname:<30}")
            print("\n" + "=" * 70)
            print(f"Destination reached in {ttl} hops!")
            print("=" * 70)
            reached_destination = True
            break

        else:  # ICMP Time Exceeded (intermediate router)
            # Get hostname
            try:
```

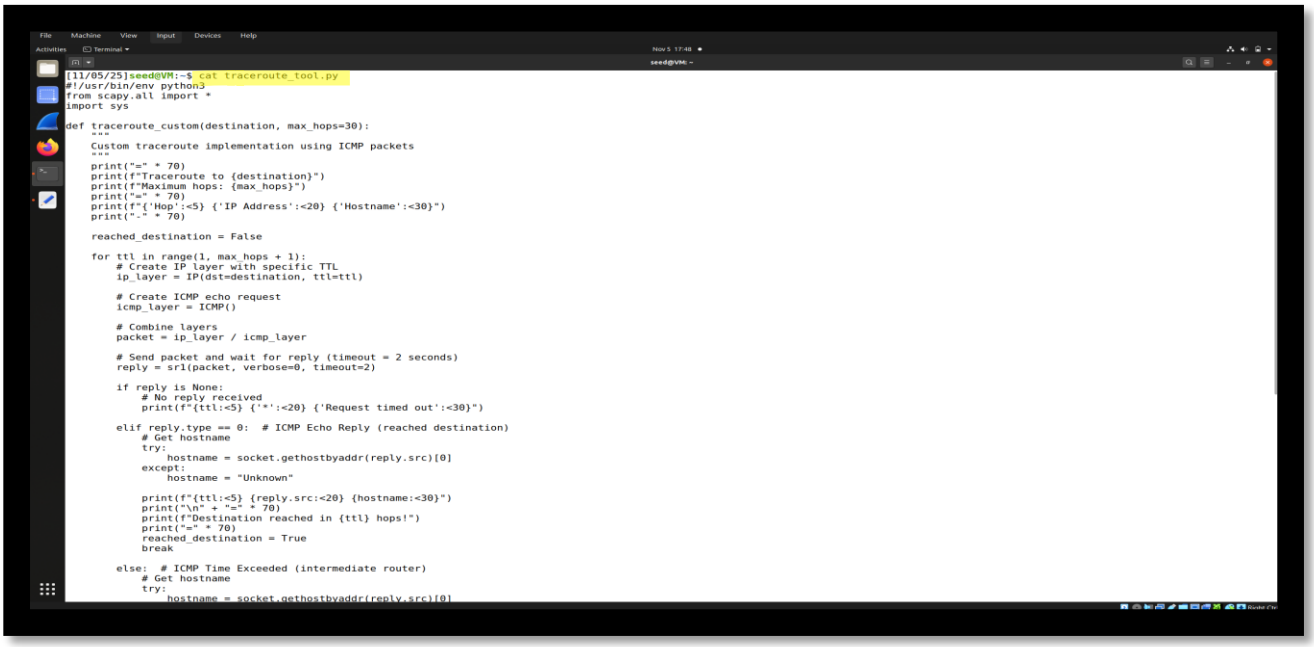
Screenshot 16: Create the Automated Traceroute Script- “traceroute_tool.py”.



```
            print(f"{ttl:<5} {'*':<20} {'Request timed out':<30}")

        elif reply.type == 0:  # ICMP Echo Reply (reached destination)
            # Get hostname
            try:
                hostname = socket.gethostbyaddr(reply.src)[0]
            except:
                hostname = "Unknown"

            print(f"{ttl:<5} {reply.src:<20} {hostname:<30}")
            print("\n" + "=" * 70)
            print(f"Destination reached in {ttl} hops!")
            print("=" * 70)
            reached_destination = True
            break

        else:  # ICMP Time Exceeded (intermediate router)
            # Get hostname
            try:
                hostname = socket.gethostbyaddr(reply.src)[0]
            except:
                hostname = "Unknown"

            print(f"{ttl:<5} {reply.src:<20} {hostname:<30}")

    if not reached_destination:
        print("\n" + "=" * 70)
        print(f"Destination not reached within {max_hops} hops")
        print("=" * 70)

    return ttl if reached_destination else max_hops

if __name__ == "__main__":
    print("\n" + "=" * 70)
    print("Custom Traceroute Tool - Task 1.3")
    print("=" * 70)

    # Get destination from user
    if len(sys.argv) > 1:
        destination = sys.argv[1]
    else:
        destination = input("\nEnter destination IP or hostname (default: 8.8.8.8): ").strip()
        if not destination:
            destination = "8.8.8.8"

    print(f"\nStarting traceroute to {destination}...\n")

    # Perform traceroute
    traceroute_custom(destination, max_hops=30)

    print("\nTraceroute complete!\n")
EOF
```

Screenshot 17: Create the Automated Traceroute Script- “traceroute_tool.py”.

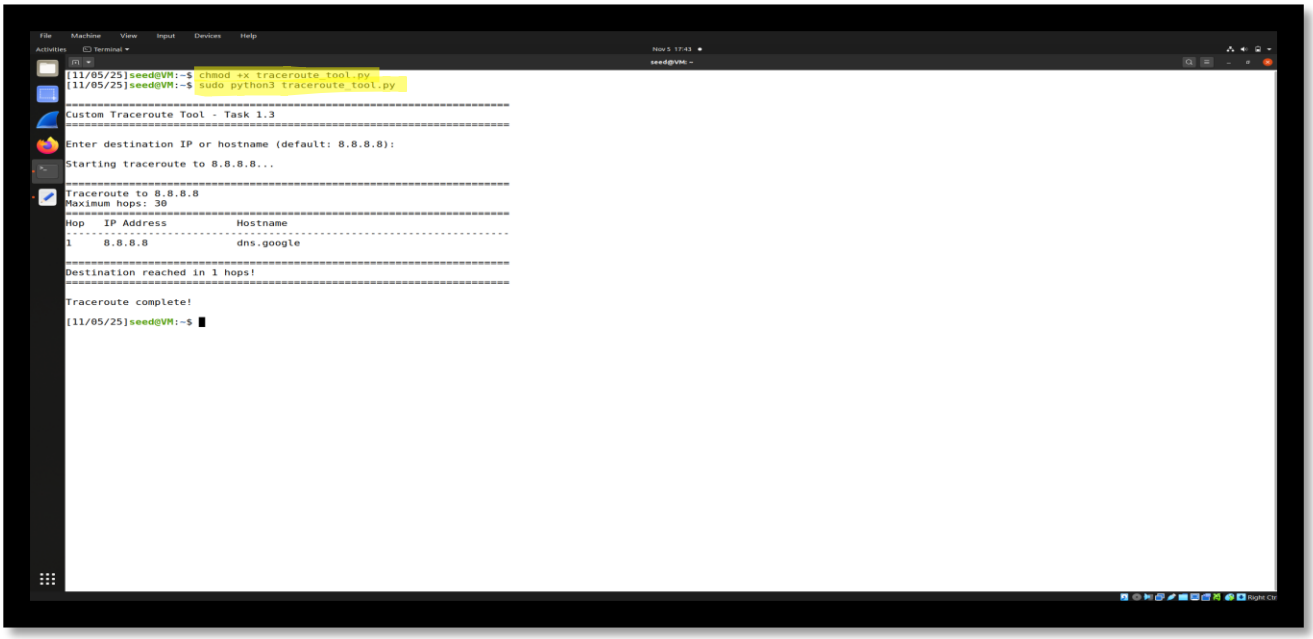
- So, we can verify the code once using “cat traceroute_tool.py” as highlighted in screenshot 18.



```
[11/05/25]seed@VM:~$ cat traceroute_tool.py
#!/usr/bin/env python3
from scapy.all import *
import sys

def traceroute_custom(destination, max_hops=30):
    """
    Custom traceroute implementation using ICMP packets
    """
    print("=" * 70)
    print(f"Traceroute to {destination}")
    print(f"Maximum hops: {max_hops}")
    print("=" * 70)
    print(f"{'Hop':<5} {'IP Address':<20} {'Hostname':<30}")
    print("-" * 70)

    reached_destination = False

    for ttl in range(1, max_hops + 1):
        # Create IP layer with specific TTL
        ip_layer = IP(dst=destination, ttl=ttl)

        # Create ICMP echo request
        icmp_layer = ICMP()

        # Combine layers
        packet = ip_layer / icmp_layer

        # Send packet and wait for reply (timeout = 2 seconds)
        reply = sr1(packet, verbose=0, timeout=2)

        if reply is None:
            # No reply received
            print(f"{ttl:<5} {'*':<20} {'Request timed out':<30}")

        elif reply.type == 0:  # ICMP Echo Reply (reached destination)
            # Get hostname
            try:
                hostname = socket.gethostbyaddr(reply.src)[0]
            except:
                hostname = "Unknown"

            print(f"{ttl:<5} {reply.src:<20} {hostname:<30}")
            print("\n" + "=" * 70)
            print(f"Destination reached in {ttl} hops!")
            print("=" * 70)
            reached_destination = True
            break

        else:  # ICMP Time Exceeded (intermediate router)
            # Get hostname
            try:
                hostname = socket.gethostbyaddr(reply.src)[0]
```

Screenshot 18: Verifying code using “cat traceroute_tool.py”.

- After that we make it executable using “chmod +x traceroute_tool.py” (highlighted below) and also use this to run the tool by using command “sudo python3 traceroute_tool.py” as highlighted in screenshot 19 and enter default google’s dns 8.8.8.8. This is how we will run the overall tool.

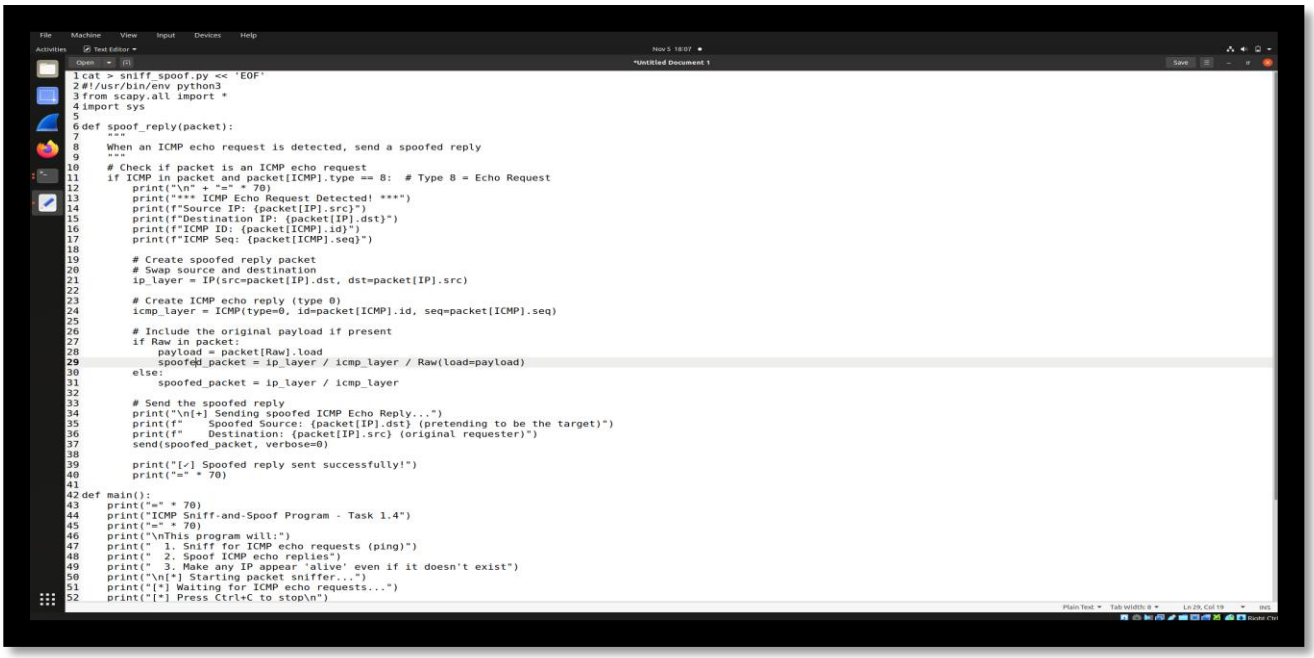


```
[11/05/25]seed@VM:~$ chmod +x traceroute_tool.py
[11/05/25]seed@VM:~$ sudo python3 traceroute_tool.py

======================================================================
Custom Traceroute Tool - Task 1.3
======================================================================

Enter destination IP or hostname (default: 8.8.8.8):

Starting traceroute to 8.8.8.8...

======================================================================
Traceroute to 8.8.8.8
Maximum hops: 30
======================================================================
Hop   IP Address           Hostname
----------------------------------------------------------------------
1     8.8.8.8              dns.google

======================================================================
Destination reached in 1 hops!
======================================================================

Traceroute complete!

[11/05/25]seed@VM:~$
```

Screenshot 19: Running the tool using “sudo python3 traceroute_tool.py” command.

Task 1.4

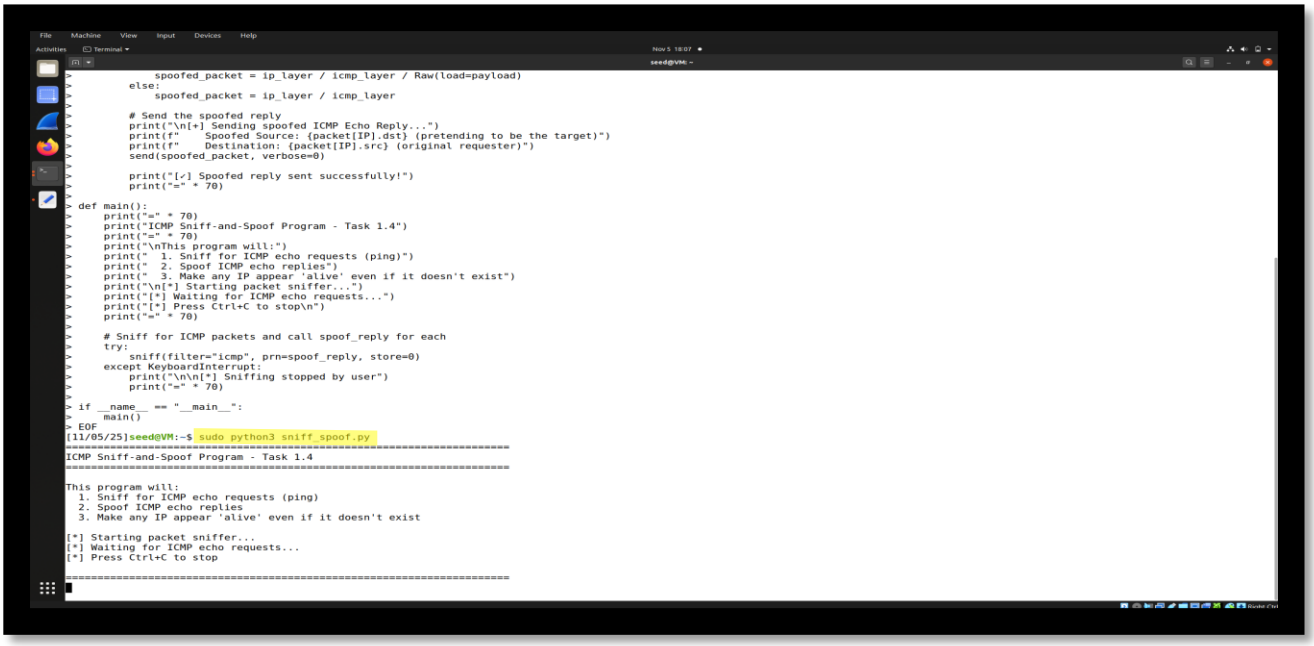
- Firstly, we will create the sniff and spoof program by creating file “sniff_spoof.py” and implementing code as observed in screenshot 20.



```
cat > sniff_spoof.py << 'EOF'
#!/usr/bin/env python3
from scapy.all import *
import sys

def spoof_reply(packet):
    """
    When an ICMP echo request is detected, send a spoofed reply
    """
    # Check if packet is an ICMP echo request
    if ICMP in packet and packet[ICMP].type == 8:  # Type 8 = Echo Request
        print("\n" + "=" * 70)
        print("*** ICMP Echo Request Detected! ***")
        print(f"Source IP: {packet[IP].src}")
        print(f"Destination IP: {packet[IP].dst}")
        print(f"ICMP ID: {packet[ICMP].id}")
        print(f"ICMP Seq: {packet[ICMP].seq}")

        # Create spoofed reply packet
        # Swap source and destination
        ip_layer = IP(src=packet[IP].dst, dst=packet[IP].src)

        # Create ICMP echo reply (type 0)
        icmp_layer = ICMP(type=0, id=packet[ICMP].id, seq=packet[ICMP].seq)

        # Include the original payload if present
        if Raw in packet:
            payload = packet[Raw].load
            spoofed_packet = ip_layer / icmp_layer / Raw(load=payload)
        else:
            spoofed_packet = ip_layer / icmp_layer

        # Send the spoofed reply
        print("\n[+] Sending spoofed ICMP Echo Reply...")
        print(f"    Spoofed Source: {packet[IP].dst} (pretending to be the target)")
        print(f"    Destination: {packet[IP].src} (original requester)")
        send(spoofed_packet, verbose=0)

        print("[✓] Spoofed reply sent successfully!")
        print("=" * 70)

def main():
    print("=" * 70)
    print("ICMP Sniff-and-Spoof Program - Task 1.4")
    print("=" * 70)
    print("\nThis program will:")
    print("  1. Sniff for ICMP echo requests (ping)")
    print("  2. Spoof ICMP echo replies")
    print("  3. Make any IP appear 'alive' even if it doesn't exist")
    print("\n[*] Starting packet sniffer...")
    print("[*] Waiting for ICMP echo requests...")
    print("[*] Press Ctrl+C to stop\n")
```

Screenshot 20: Code used to create sniff and spoof program- “sniff_spoof.py”.

- Then we will run this program using command “sudo python3 sniff_spoof.py” as highlighted in screenshot 21.

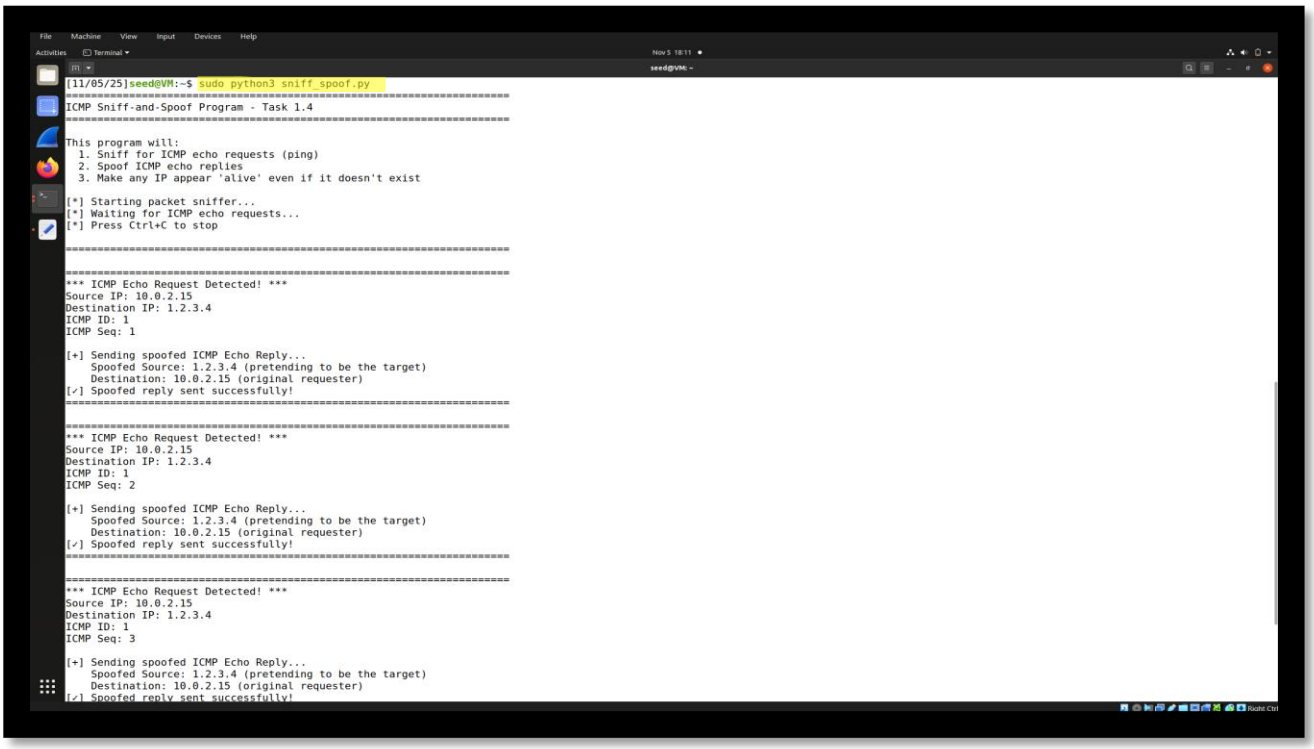


```
>             spoofed_packet = ip_layer / icmp_layer / Raw(load=payload)
>         else:
>             spoofed_packet = ip_layer / icmp_layer
>
>         # Send the spoofed reply
>         print("\n[+] Sending spoofed ICMP Echo Reply...")
>         print(f"    Spoofed Source: {packet[IP].dst} (pretending to be the target)")
>         print(f"    Destination: {packet[IP].src} (original requester)")
>         send(spoofed_packet, verbose=0)
>
>         print("[✓] Spoofed reply sent successfully!")
>         print("=" * 70)
>
> def main():
>     print("=" * 70)
>     print("ICMP Sniff-and-Spoof Program - Task 1.4")
>     print("=" * 70)
>     print("\nThis program will:")
>     print("  1. Sniff for ICMP echo requests (ping)")
>     print("  2. Spoof ICMP echo replies")
>     print("  3. Make any IP appear 'alive' even if it doesn't exist")
>     print("\n[*] Starting packet sniffer...")
>     print("[*] Waiting for ICMP echo requests...")
>     print("[*] Press Ctrl+C to stop\n")
>     print("=" * 70)
>
>     # Sniff for ICMP packets and call spoof_reply for each
>     try:
>         sniff(filter="icmp", prn=spoof_reply, store=0)
>     except KeyboardInterrupt:
>         print("\n\n[*] Sniffing stopped by user")
>         print("=" * 70)
>
> if __name__ == "__main__":
>     main()
> EOF
[11/05/25]seed@VM:~$ sudo python3 sniff_spoof.py
======================================================================
ICMP Sniff-and-Spoof Program - Task 1.4
======================================================================

This program will:
  1. Sniff for ICMP echo requests (ping)
  2. Spoof ICMP echo replies
  3. Make any IP appear 'alive' even if it doesn't exist

[*] Starting packet sniffer...
[*] Waiting for ICMP echo requests...
[*] Press Ctrl+C to stop

======================================================================
```

Screenshot 21: To run the program using command “sudo python3 sniff_spoof.py”.

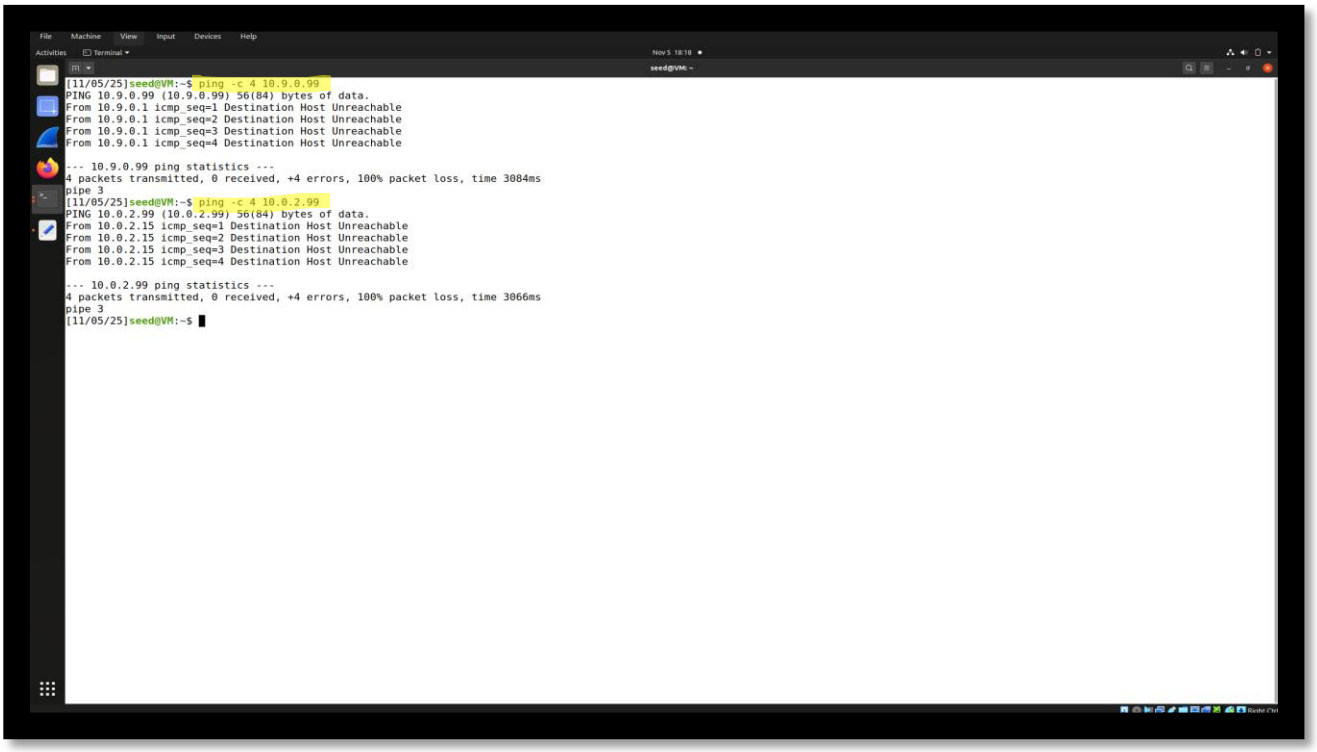
- Then, we will test three different scenarios, first one is of testing non-existing host on the internet by using “ping -c 4 1.2.3.4” as highlighted in screenshot 22 and we can observe the result of it in screenshot 23.

Screenshot 22: Executing ping command “ping -c 4 1.2.3.4”.

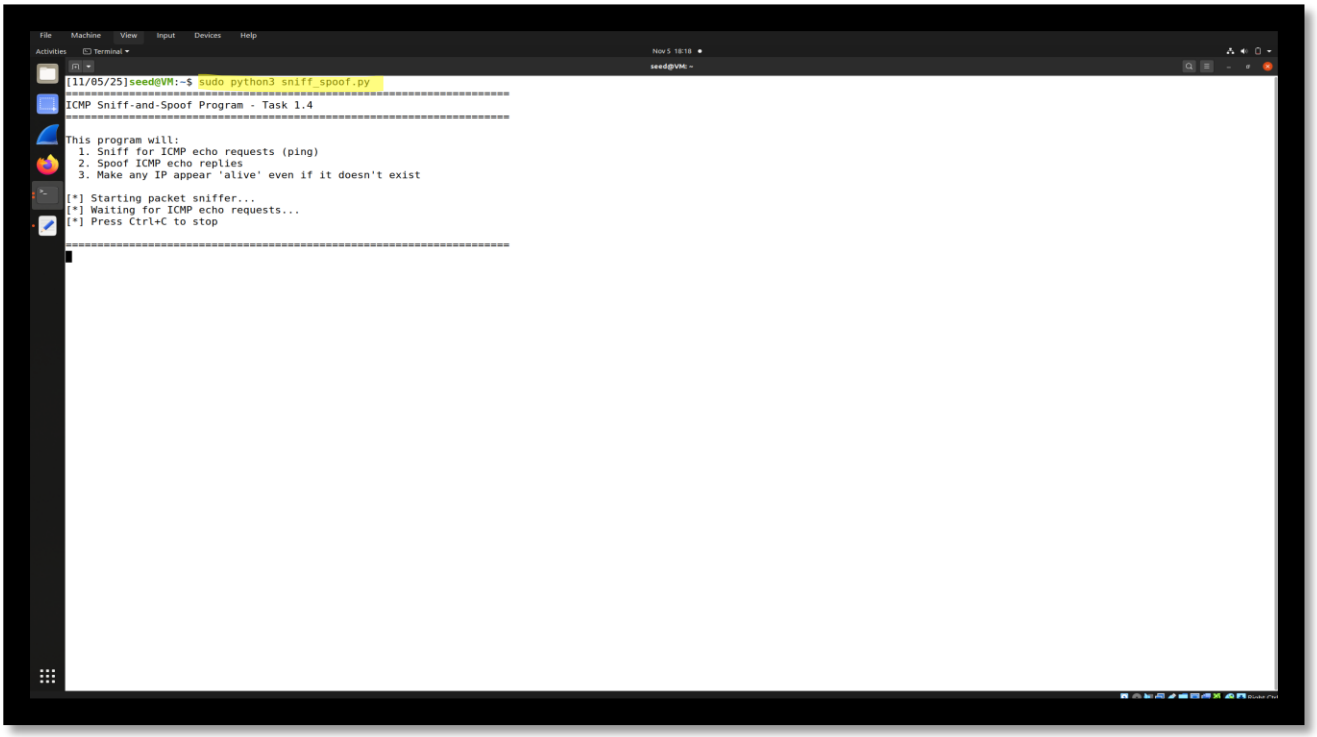


Screenshot 23: Executing “sudo python3 sniff spoof.py” to observe result from ping.

- Second one is of testing non-existing host on the LAN by using “ping -c 4 10.0.2.99” as highlighted in screenshot 24 and we can observe the result of it in screenshot 25.

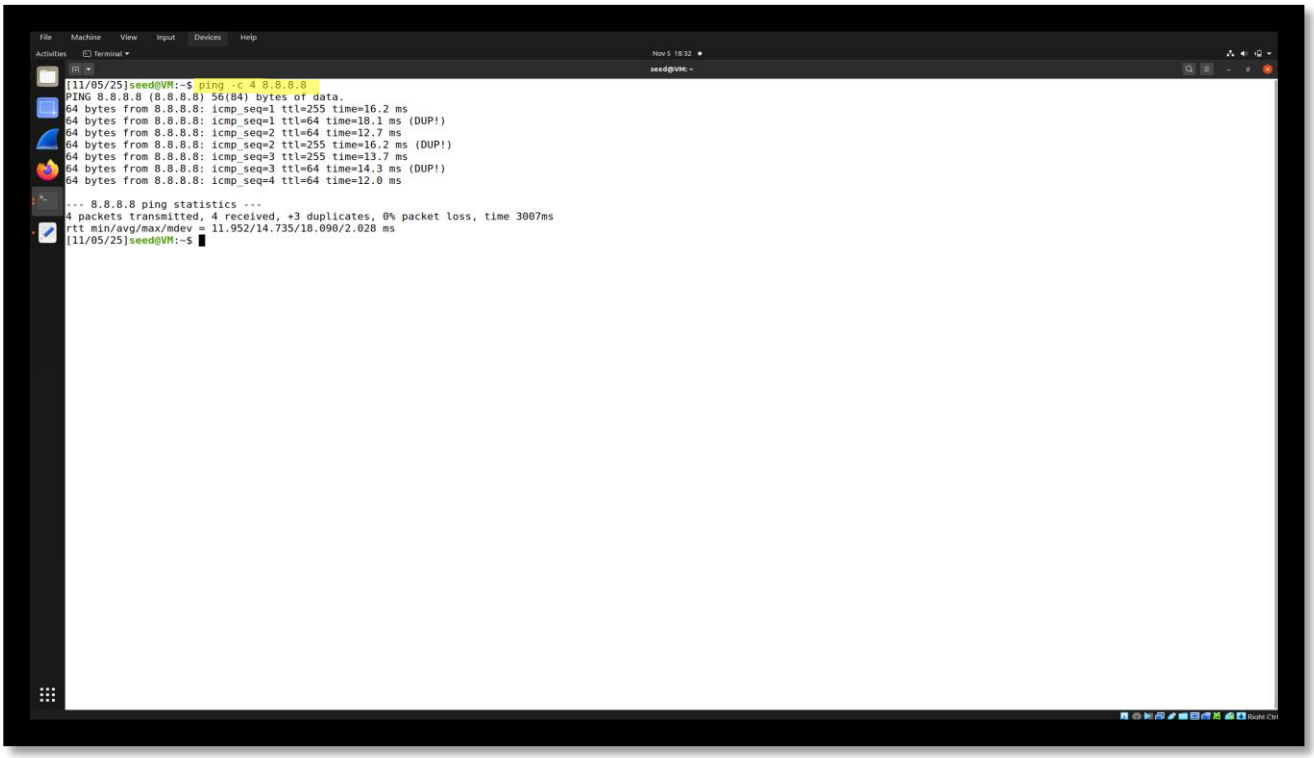


Screenshot 24: Executing ping command “ping -c 4 10.0.2.99”.

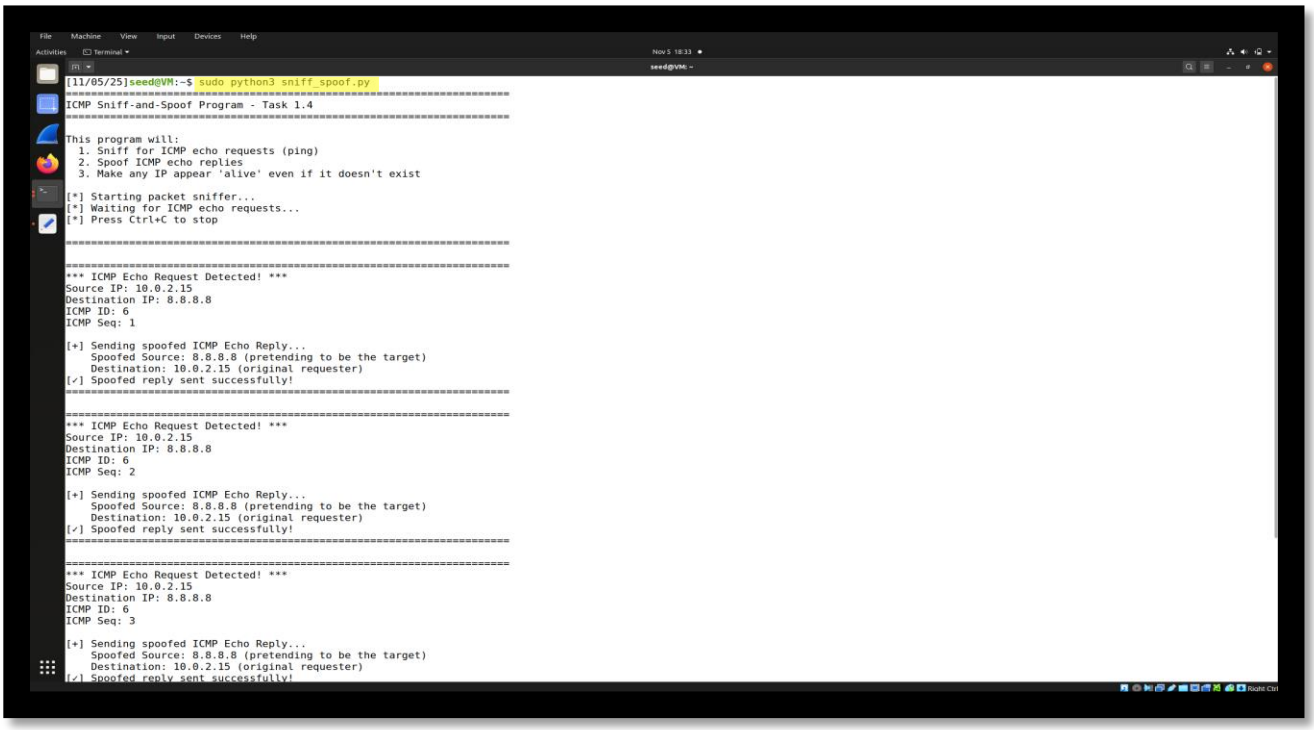


Screenshot 25: Executing “sudo python3 sniff_spoof.py” to observe result from ping.

- Third one is of testing existing host on the internet by using “ping -c 4 8.8.8.8” as highlighted in screenshot 26 and we can observe the result of it in screenshot 27.

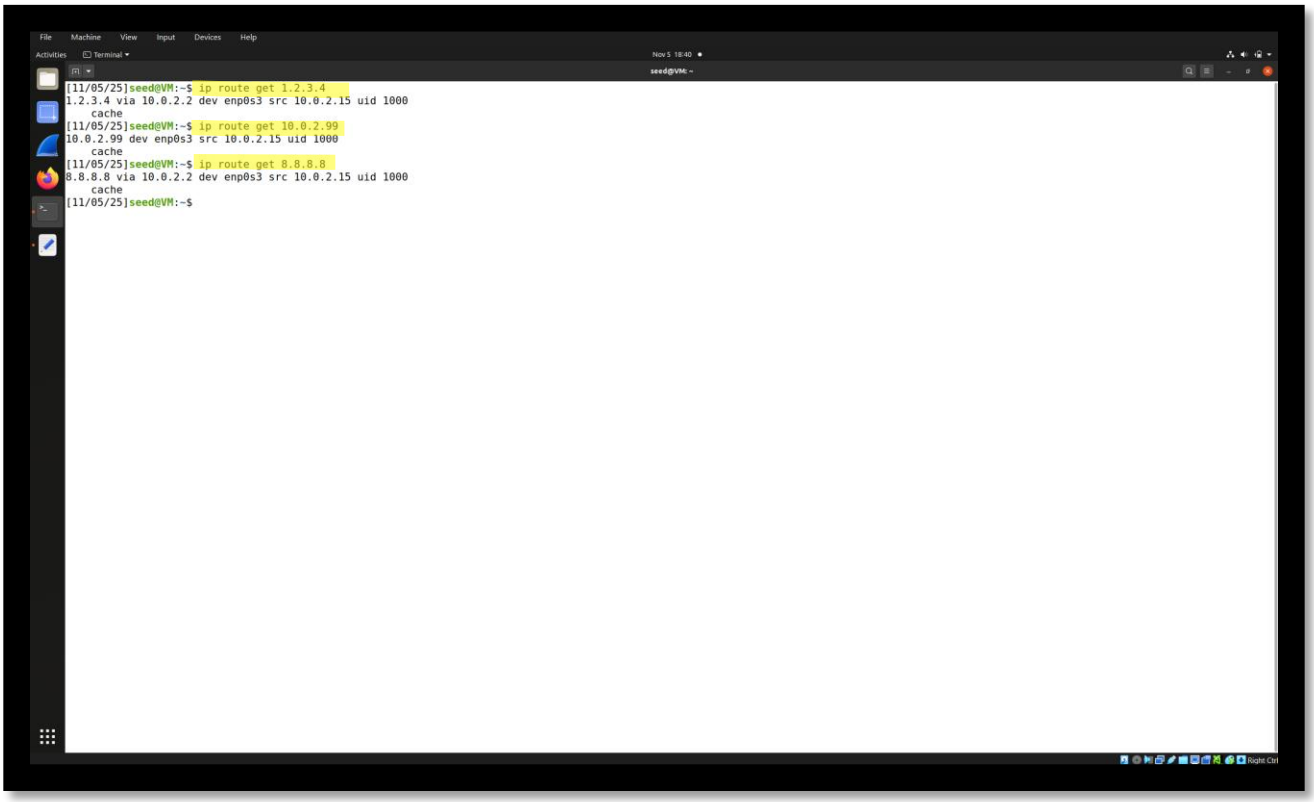


Screenshot 26: Executing ping command “ping -c 4 8.8.8.8”.



Screenshot 27: Executing “sudo python3 sniff_spoof.py” to observe result from ping.

- After that to verify the path where the packet to this IP address, we utilize “ip route get ….” which is highlighted in screenshot 28.



Screenshot 28: Utilizing “ip route get ….” to verify the full route of packets.