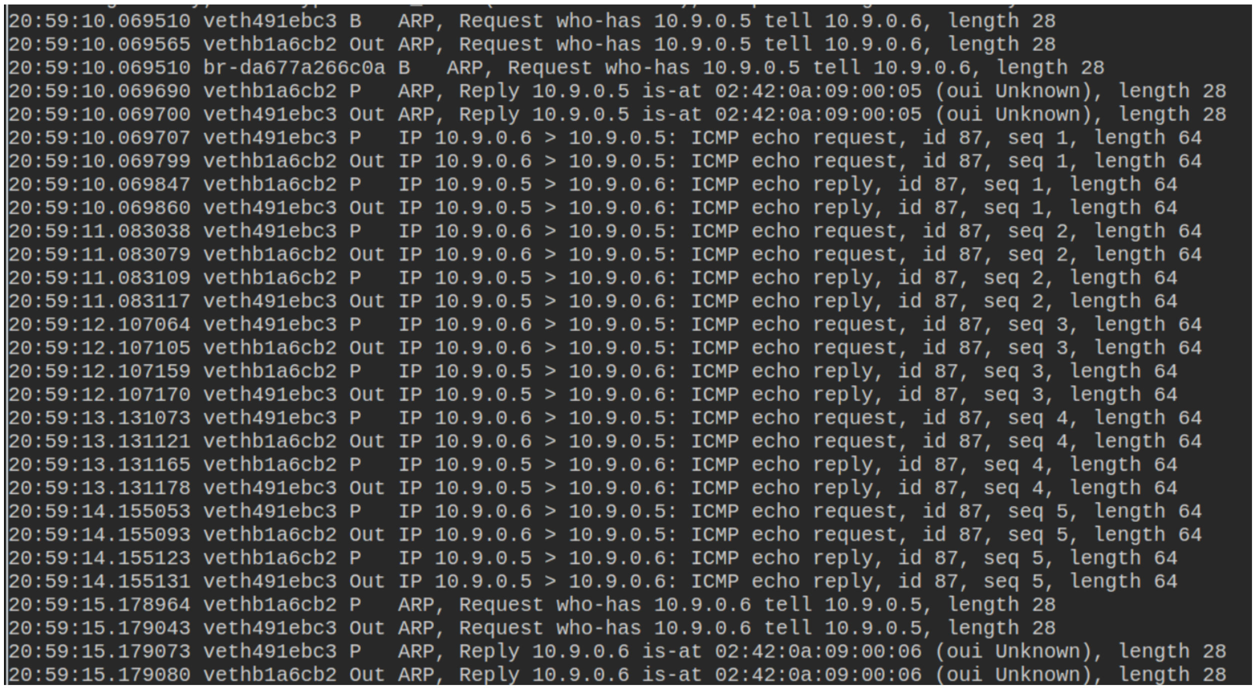
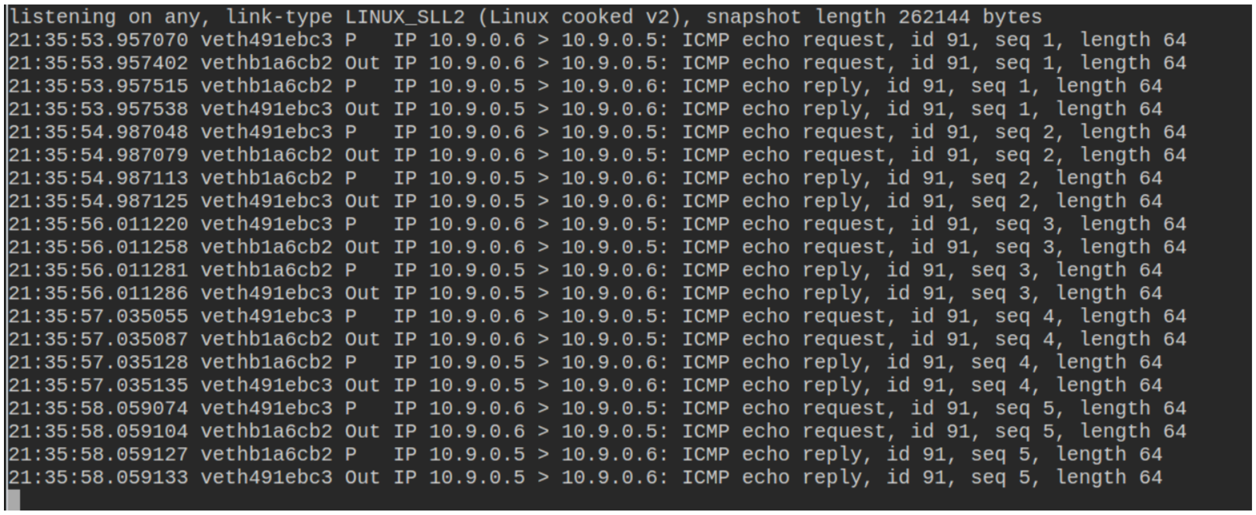
Task 1: Packet Sniffing
1.Capturing all the packets between the user and the victim:



```
20:59:10.069510 veth491ebc3 B   ARP, Request who-has 10.9.0.5 tell 10.9.0.6, length 28
20:59:10.069565 vethb1a6cb2 Out ARP, Request who-has 10.9.0.5 tell 10.9.0.6, length 28
20:59:10.069510 br-da677a266c0a B   ARP, Request who-has 10.9.0.5 tell 10.9.0.6, length 28
20:59:10.069690 vethb1a6cb2 P   ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05 (oui Unknown), length 28
20:59:10.069700 veth491ebc3 Out ARP, Reply 10.9.0.5 is-at 02:42:0a:09:00:05 (oui Unknown), length 28
20:59:10.069707 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 1, length 64
20:59:10.069799 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 1, length 64
20:59:10.069847 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 1, length 64
20:59:10.069860 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 1, length 64
20:59:11.083038 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 2, length 64
20:59:11.083079 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 2, length 64
20:59:11.083109 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 2, length 64
20:59:11.083117 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 2, length 64
20:59:12.107064 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 3, length 64
20:59:12.107105 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 3, length 64
20:59:12.107159 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 3, length 64
20:59:12.107170 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 3, length 64
20:59:13.131073 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 4, length 64
20:59:13.131121 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 4, length 64
20:59:13.131165 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 4, length 64
20:59:13.131178 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 4, length 64
20:59:14.155053 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 5, length 64
20:59:14.155093 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 87, seq 5, length 64
20:59:14.155123 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 5, length 64
20:59:14.155131 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 87, seq 5, length 64
20:59:15.178964 vethb1a6cb2 P   ARP, Request who-has 10.9.0.6 tell 10.9.0.5, length 28
20:59:15.179043 veth491ebc3 Out ARP, Request who-has 10.9.0.6 tell 10.9.0.5, length 28
20:59:15.179073 veth491ebc3 P   ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:06 (oui Unknown), length 28
20:59:15.179080 vethb1a6cb2 Out ARP, Reply 10.9.0.6 is-at 02:42:0a:09:00:06 (oui Unknown), length 28
```

In this scenario, we observe an ARP request sent from the user to the attacker, asking for the IP address of the destination, which is 10.9.0.5.
This is followed by an ARP response, which includes the MAC address of the destination: 02:42:0a:09:00:05.
Next, we see an ICMP request. But first, what is an ICMP request? ICMP (Internet Control Message Protocol) is a network layer protocol used for error reporting, diagnostics, and network management. It enables devices to communicate issues such as unreachable destinations or packet timeouts. ICMP is essential for tools like ping (which uses ICMP Echo Requests/Replies to test connectivity) and traceroute (which maps network paths using ICMP Time Exceeded messages).
This process repeats itself every second.

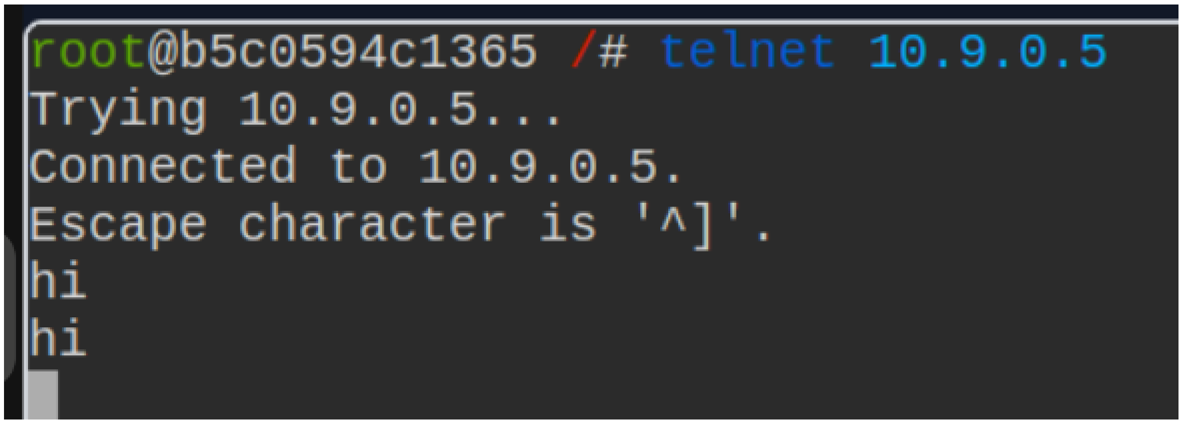
2.After applying a filter to capture only ICMP requests, we can focus on the relevant traffic.



```
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
21:35:53.957070 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 1, length 64
21:35:53.957402 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 1, length 64
21:35:53.957515 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 1, length 64
21:35:53.957538 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 1, length 64
21:35:54.987048 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 2, length 64
21:35:54.987079 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 2, length 64
21:35:54.987113 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 2, length 64
21:35:54.987125 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 2, length 64
21:35:56.011220 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 3, length 64
21:35:56.011258 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 3, length 64
21:35:56.011281 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 3, length 64
21:35:56.011286 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 3, length 64
21:35:57.035055 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 4, length 64
21:35:57.035087 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 4, length 64
21:35:57.035128 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 4, length 64
21:35:57.035135 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 4, length 64
21:35:58.059074 veth491ebc3 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 5, length 64
21:35:58.059104 vethb1a6cb2 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 91, seq 5, length 64
21:35:58.059127 vethb1a6cb2 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 5, length 64
21:35:58.059133 veth491ebc3 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 91, seq 5, length 64
```

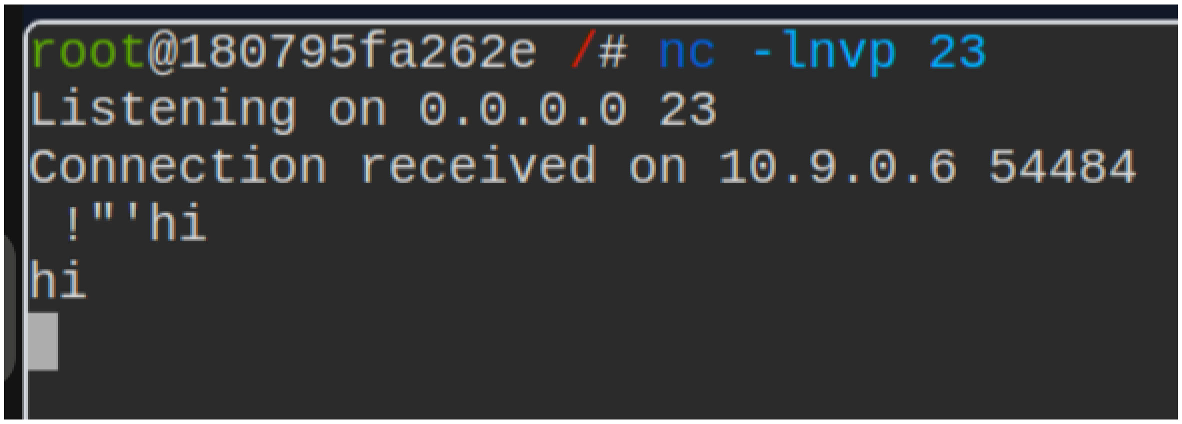
3.Shifting Focus to Port 23 (Telnet):

On the user machine, we execute the Telnet command using the IP address 10.9.0.5. By running the command telnet 10.9.0.5, we now established a connection on the 23 port.



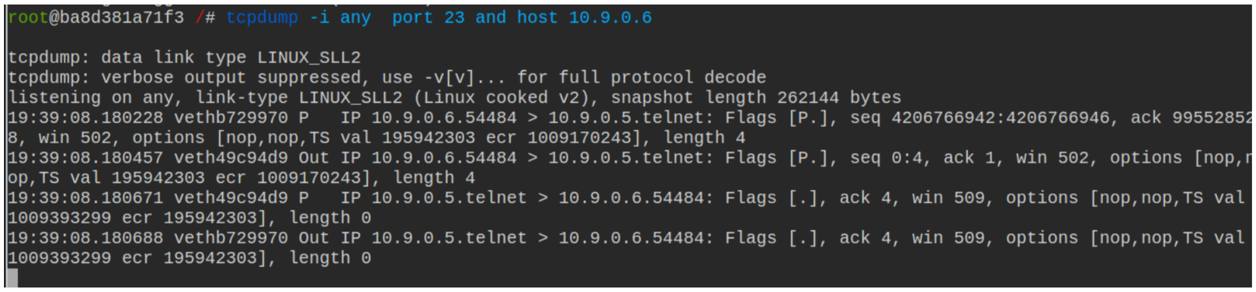
```
root@b5c0594c1365 /# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
hi
hi
```

From the victim's machine we listen for the communication coming from the victim on the port 23 to the user.



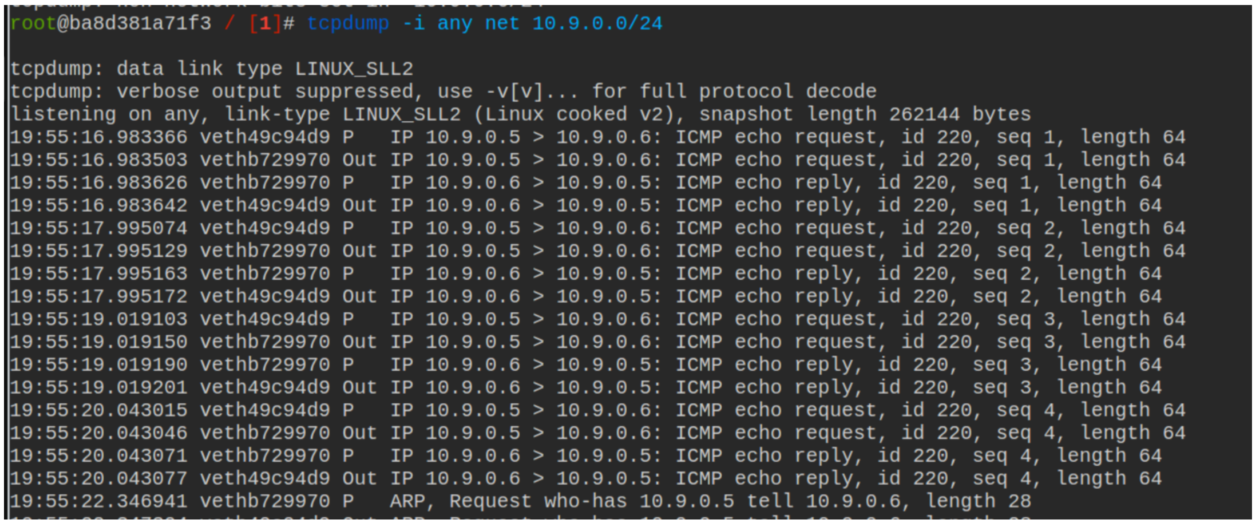
```
root@180795fa262e /# nc -lnvp 23
Listening on 0.0.0.0 23
Connection received on 10.9.0.6 54484
 !"'hi
hi
```

This is the command that I used to listen to the traffic on port 23 from the attacker Vm.



```
root@ba8d381a71f3 /# tcpdump -i any  port 23 and host 10.9.0.6

tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
19:39:08.180228 vethb729970 P   IP 10.9.0.6.54484 > 10.9.0.5.telnet: Flags [P.], seq 4206766942:4206766946, ack 99552852
8, win 502, options [nop,nop,TS val 195942303 ecr 1009170243], length 4
19:39:08.180457 veth49c94d9 Out IP 10.9.0.6.54484 > 10.9.0.5.telnet: Flags [P.], seq 0:4, ack 1, win 502, options [nop,n
op,TS val 195942303 ecr 1009170243], length 4
19:39:08.180671 veth49c94d9 P   IP 10.9.0.5.telnet > 10.9.0.6.54484: Flags [.], ack 4, win 509, options [nop,nop,TS val
1009393299 ecr 195942303], length 0
19:39:08.180688 vethb729970 Out IP 10.9.0.5.telnet > 10.9.0.6.54484: Flags [.], ack 4, win 509, options [nop,nop,TS val
1009393299 ecr 195942303], length 0
```

4.This is the fourth which is listening according to the subnet mask

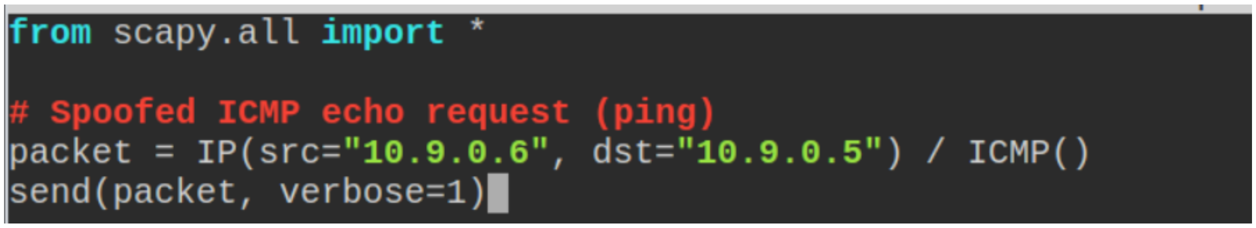


```
root@ba8d381a71f3 / [1]# tcpdump -i any net 10.9.0.0/24

tcpdump: data link type LINUX_SLL2
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on any, link-type LINUX_SLL2 (Linux cooked v2), snapshot length 262144 bytes
19:55:16.983366 veth49c94d9 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo request, id 220, seq 1, length 64
19:55:16.983503 vethb729970 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo request, id 220, seq 1, length 64
19:55:16.983626 vethb729970 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo reply, id 220, seq 1, length 64
19:55:16.983642 veth49c94d9 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo reply, id 220, seq 1, length 64
19:55:17.995074 veth49c94d9 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo request, id 220, seq 2, length 64
19:55:17.995129 vethb729970 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo request, id 220, seq 2, length 64
19:55:17.995163 vethb729970 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo reply, id 220, seq 2, length 64
19:55:17.995172 veth49c94d9 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo reply, id 220, seq 2, length 64
19:55:19.019103 veth49c94d9 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo request, id 220, seq 3, length 64
19:55:19.019150 vethb729970 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo request, id 220, seq 3, length 64
19:55:19.019190 vethb729970 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo reply, id 220, seq 3, length 64
19:55:19.019201 veth49c94d9 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo reply, id 220, seq 3, length 64
19:55:20.043015 veth49c94d9 P   IP 10.9.0.5 > 10.9.0.6: ICMP echo request, id 220, seq 4, length 64
19:55:20.043046 vethb729970 Out IP 10.9.0.5 > 10.9.0.6: ICMP echo request, id 220, seq 4, length 64
19:55:20.043071 vethb729970 P   IP 10.9.0.6 > 10.9.0.5: ICMP echo reply, id 220, seq 4, length 64
19:55:20.043077 veth49c94d9 Out IP 10.9.0.6 > 10.9.0.5: ICMP echo reply, id 220, seq 4, length 64
19:55:22.346941 vethb729970 P   ARP, Request who-has 10.9.0.5 tell 10.9.0.6, length 28
```

Task 2:

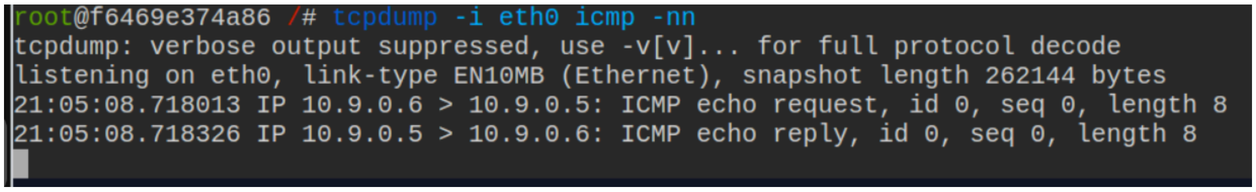
What We Are Trying to Do Now:
We are attempting to trick the victim by sending a ping request from the attacker's machine. The attacker spoofs the source IP address, making it appear as if the ping request is coming from the user's IP address (10.9.0.6). This deception causes the victim to respond to the user's IP address, even though the user never actually sent any ping requests. In reality, the user's IP address was spoofed by the attacker.

Here is the python script used for spoofing :



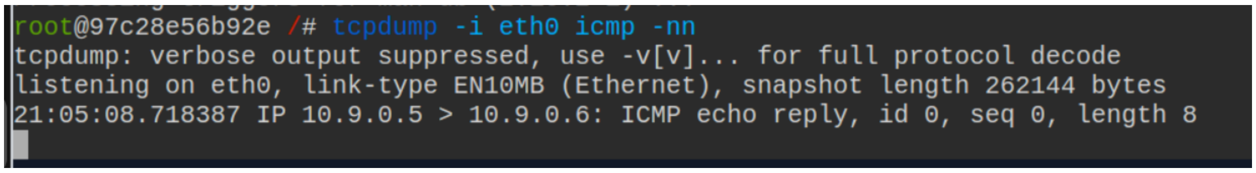
```
from scapy.all import *

# Spoofed ICMP echo request (ping)
packet = IP(src="10.9.0.6", dst="10.9.0.5") / ICMP()
send(packet, verbose=1)
```

The screenshot below is for the victim VM



```
root@f6469e374a86 /# tcpdump -i eth0 icmp -nn
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
21:05:08.718013 IP 10.9.0.6 > 10.9.0.5: ICMP echo request, id 0, seq 0, length 8
21:05:08.718326 IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 0, seq 0, length 8
```

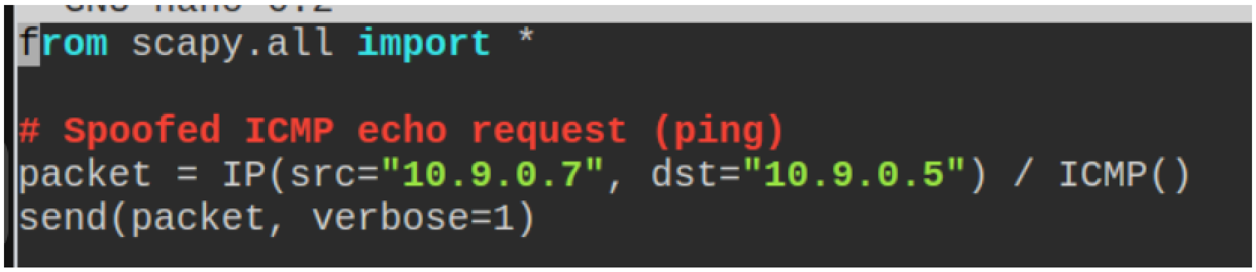
The screenshot below is for the user VM



```
root@97c28e56b92e /# tcpdump -i eth0 icmp -nn
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
21:05:08.718387 IP 10.9.0.5 > 10.9.0.6: ICMP echo reply, id 0, seq 0, length 8
```

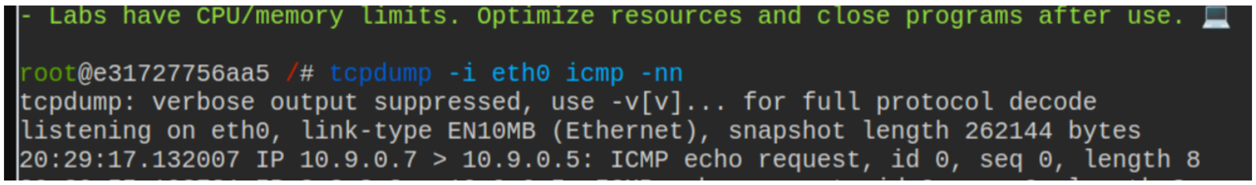
As we see it receives the the ping from the victim that was originally fooled that the ping come from the user

So lets now repeat the Task suing an Ip that I just created by my mind



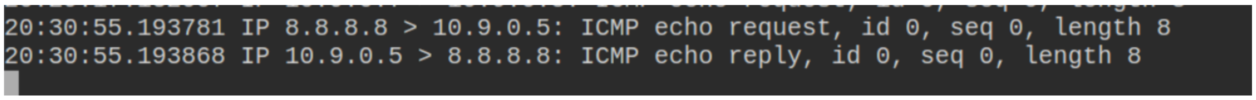
```
from scapy.all import *

# Spoofed ICMP echo request (ping)
packet = IP(src="10.9.0.7", dst="10.9.0.5") / ICMP()
send(packet, verbose=1)
```

Then we can see the request in the victims VM for the request but no response is sent back



```
- Labs have CPU/memory limits. Optimize resources and close programs after use. 💻

root@e31727756aa5 /# tcpdump -i eth0 icmp -nn
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
20:29:17.132007 IP 10.9.0.7 > 10.9.0.5: ICMP echo request, id 0, seq 0, length 8
```

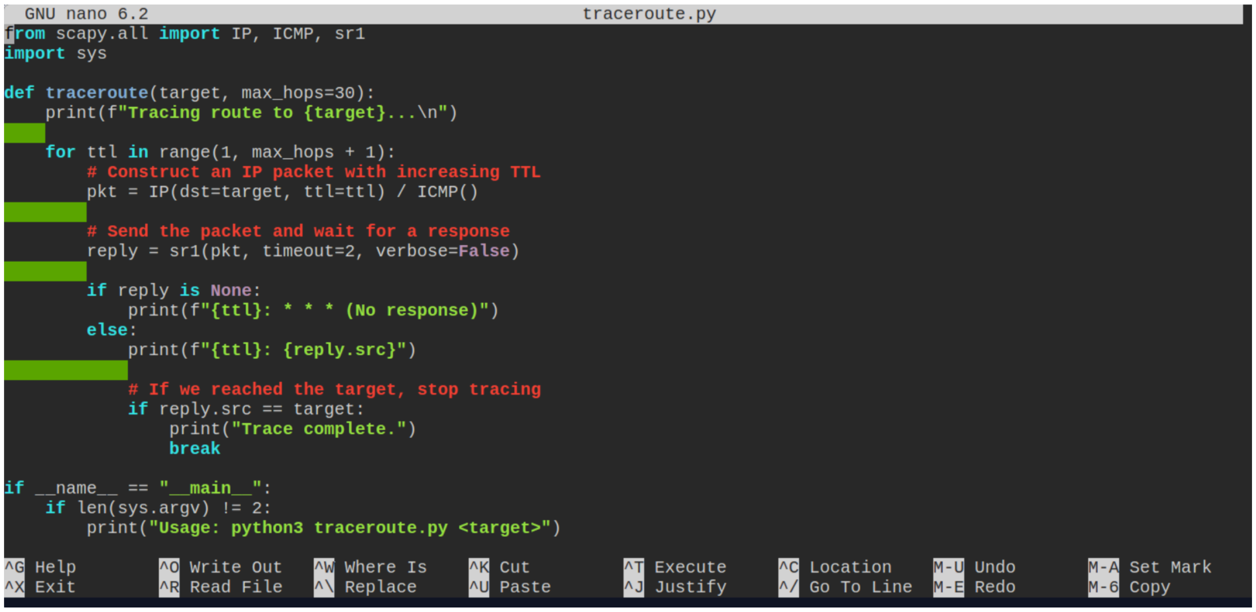
Trying again but putting google IP as a src IP



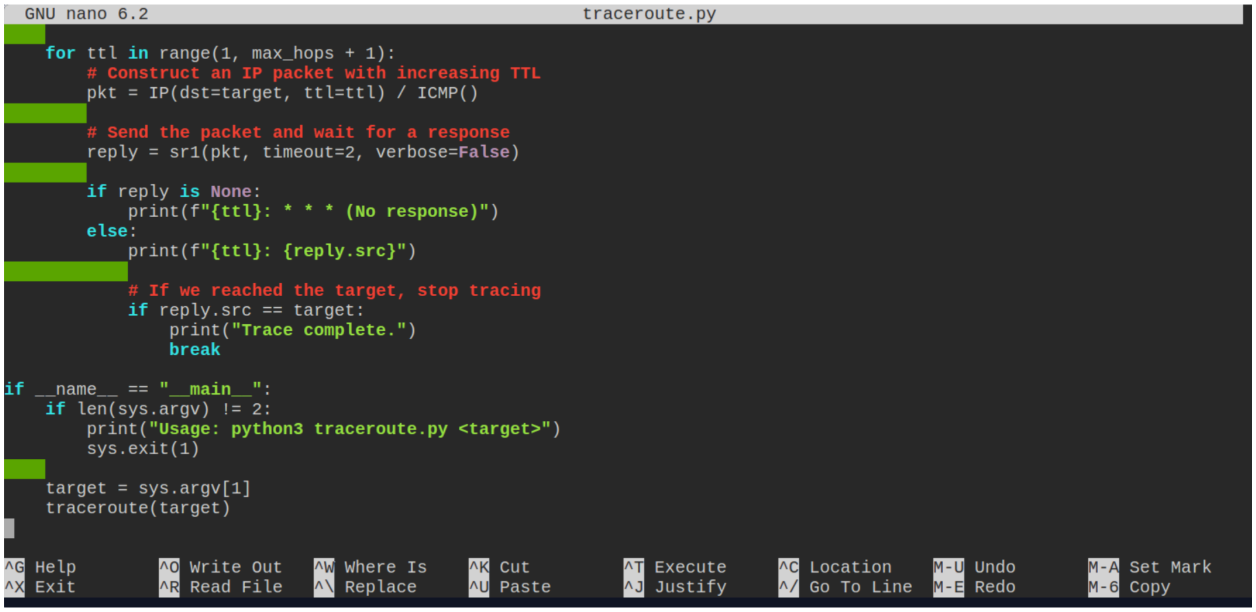
```
20:30:55.193781 IP 8.8.8.8 > 10.9.0.5: ICMP echo request, id 0, seq 0, length 8
20:30:55.193868 IP 10.9.0.5 > 8.8.8.8: ICMP echo reply, id 0, seq 0, length 8
```

Task 3 :

So I write a python script that trace a packet that is going to google and this is the code

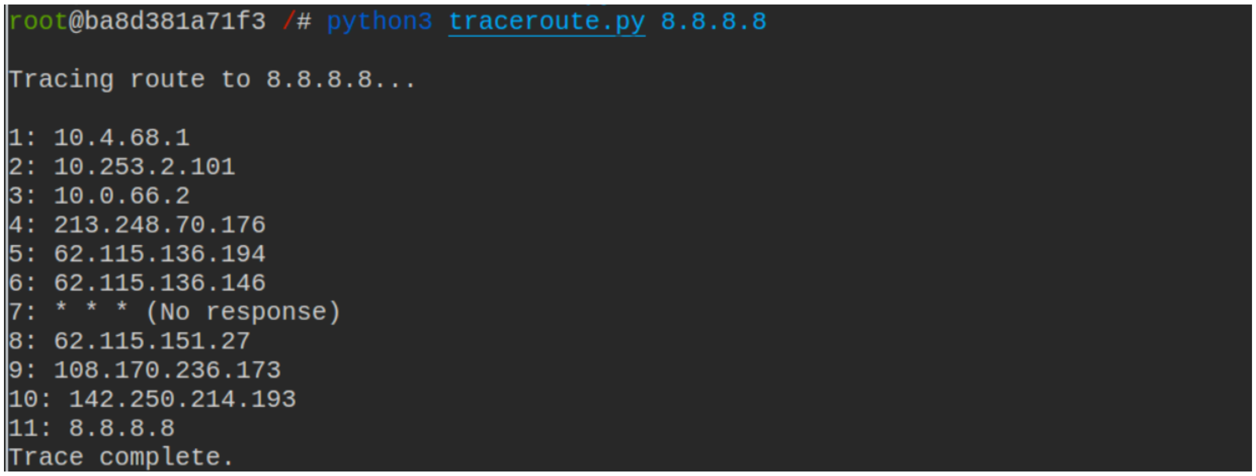


```
  GNU nano 6.2                                          traceroute.py
from scapy.all import IP, ICMP, sr1
import sys

def traceroute(target, max_hops=30):
    print(f"Tracing route to {target}...\n")

    for ttl in range(1, max_hops + 1):
        # Construct an IP packet with increasing TTL
        pkt = IP(dst=target, ttl=ttl) / ICMP()

        # Send the packet and wait for a response
        reply = sr1(pkt, timeout=2, verbose=False)

        if reply is None:
            print(f"{ttl}: * * * (No response)")
        else:
            print(f"{ttl}: {reply.src}")

            # If we reached the target, stop tracing
            if reply.src == target:
                print("Trace complete.")
                break

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python3 traceroute.py <target>")

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```



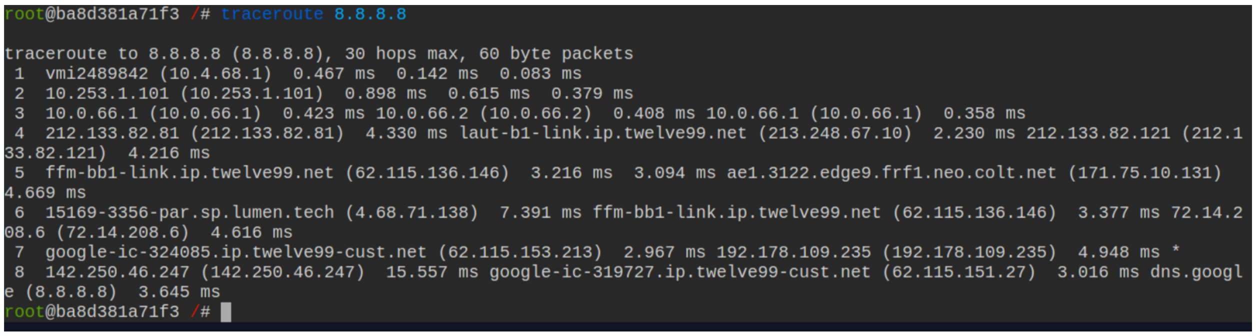
```
  GNU nano 6.2                                          traceroute.py

    for ttl in range(1, max_hops + 1):
        # Construct an IP packet with increasing TTL
        pkt = IP(dst=target, ttl=ttl) / ICMP()

        # Send the packet and wait for a response
        reply = sr1(pkt, timeout=2, verbose=False)

        if reply is None:
            print(f"{ttl}: * * * (No response)")
        else:
            print(f"{ttl}: {reply.src}")

            # If we reached the target, stop tracing
            if reply.src == target:
                print("Trace complete.")
                break

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python3 traceroute.py <target>")
        sys.exit(1)

    target = sys.argv[1]
    traceroute(target)


^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```

After that I ran the script heading to the google server as shown in the photo



```
root@ba8d381a71f3 /# python3 traceroute.py 8.8.8.8

Tracing route to 8.8.8.8...

1: 10.4.68.1
2: 10.253.2.101
3: 10.0.66.2
4: 213.248.70.176
5: 62.115.136.194
6: 62.115.136.146
7: * * * (No response)
8: 62.115.151.27
9: 108.170.236.173
10: 142.250.214.193
11: 8.8.8.8
Trace complete.
```

After that I ran the build in trace route option and it give me this



```
root@ba8d381a71f3 /# traceroute 8.8.8.8

traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  vmi2489842 (10.4.68.1)  0.467 ms  0.142 ms  0.083 ms
 2  10.253.1.101 (10.253.1.101)  0.898 ms  0.615 ms  0.379 ms
 3  10.0.66.1 (10.0.66.1)  0.423 ms 10.0.66.2 (10.0.66.2)  0.408 ms 10.0.66.1 (10.0.66.1)  0.358 ms
 4  212.133.82.81 (212.133.82.81)  4.330 ms laut-b1-link.ip.twelve99.net (213.248.67.10)  2.230 ms 212.133.82.121 (212.1
33.82.121)  4.216 ms
 5  ffm-bb1-link.ip.twelve99.net (62.115.136.146)  3.216 ms  3.094 ms ae1.3122.edge9.frf1.neo.colt.net (171.75.10.131)
4.669 ms
 6  15169-3356-par.sp.lumen.tech (4.68.71.138)  7.391 ms ffm-bb1-link.ip.twelve99.net (62.115.136.146)  3.377 ms 72.14.2
08.6 (72.14.208.6)  4.616 ms
 7  google-ic-324085.ip.twelve99-cust.net (62.115.153.213)  2.967 ms 192.178.109.235 (192.178.109.235)  4.948 ms *
 8  142.250.46.247 (142.250.46.247)  15.557 ms google-ic-319727.ip.twelve99-cust.net (62.115.151.27)  3.016 ms dns.googl
e (8.8.8.8)  3.645 ms
root@ba8d381a71f3 /#
```

We can slightly see differences in the routes.

Code 1 :

```python
from scapy.all import *

# Spoofed ICMP echo request (ping)
packet = IP(src="10.9.0.6", dst="10.9.0.5") / ICMP()
send(packet, verbose=1)
```

Code 2:

```python
from scapy.all import IP, ICMP, sr1
import sys

def traceroute(target, max_hops=30):
    print(f"Tracing route to {target}...\n")

    for ttl in range(1, max_hops + 1):
        # Construct an IP packet with increasing TTL
        pkt = IP(dst=target, ttl=ttl) / ICMP()

        # Send the packet and wait for a response
        reply = sr1(pkt, timeout=2, verbose=False)

        if reply is None:
            print(f"{ttl}: * * * (No response)")
        else:
            print(f"{ttl}: {reply.src}")

            # If we reached the target, stop tracing
            if reply.src == target:
                print("Trace complete.")
                break

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: python3 traceroute.py <target>")
        sys.exit(1)
    target = sys.argv[1]
    traceroute(target)
```