Program arguments:

`source_ip`: The IP address the victim **(target)** is looking for (the target IP address to which the victim sends ARP requests).

- This is the IP address requested in the victim's ARP query. Your program will impersonate the device with this address.

`source_mac`: The MAC address the attacker impersonates (fake MAC address sent in the ARP reply).

- This is the MAC address your program sends in the ARP reply to the victim, replacing the real MAC address of the device with `source_ip`. Ideally, this should be your machine's MAC address, so the victim routes traffic to your device.

`target_ip`**:** The victim's IP address.

- This is the IP address of the device sending the ARP request. The program waits for ARP requests from this address and responds with a spoofed ARP reply.

`target_mac`**:** The victim's MAC address.

- This is the MAC address of the device sending the ARP request. It helps the program identify the recipient of the spoofed ARP reply.

Program Execution Example

Imagine a network with two devices:

1. **Device A (Victim):** IP `10.0.0.20`, MAC `AA:BB:CC:DD:EE:FF`
2. **Device B (Router):** IP `10.0.0.1`, MAC `11:22:33:44:55:66`

You want to spoof ARP entries on Device **A** to think that the router's IP (`10.0.0.1`) is your machine's MAC (`FF:EE:DD:CC:BB:AA`).

Command execution: sudo ./ft_malcolm `10.0.0.1` `FF:EE:DD:CC:BB:AA` `10.0.0.20` `AA:BB:CC:DD:EE:FF`

Explanation:

- `10.0.0.1` :The IP address you will spoof **(the router's IP).**
- `FF:EE:DD:CC:BB:AA`: Your MAC address, to be associated with the **router's** IP `10.0.0.1`.
- `10.0.0.20`: The target IP address **(Device A)**.
- `AA:BB:CC:DD:EE:FF`: The target MAC address **(Device A)**.

Program Behavior:

1. **Wait for ARP Requests:**
    - The program waits for **Device A** to send an ARP request to find the MAC address for `10.0.0.1`**.**
    - ARP requests are broadcast messages visible to all devices on the network.
2. **Reply with Spoofed Information:**
    - Upon detecting an ARP request, the program sends a spoofed ARP reply to **Device A**, claiming:
        i. The IP address `10.0.0.1` **(Router)** is associated with the MAC address `FF:EE:DD:CC:BB:AA` **(your computer).**
    - **Device A** updates its ARP table with this new information, associating `10.0.0.1` with your MAC.
3. **Program Exit:**
    - After sending the spoofed ARP reply, the program terminates.
    - **Device A** will now send traffic intended for the router to your machine instead.

    - he code and using raw sockets for direct access to packets.
    - **Raw Sockets**:

        - Allow direct access to network packets at the IP level, bypassing higher-level protocol handling like TCP/UDP.
        - Enable faster ARP responses by reducing system delays in packet handling.

2. **Send Multiple ARP Replies:**
    - Instead of a single ARP reply, send multiple responses in quick succession. This increases the chance of overwriting the router's legitimate reply.
3. **Periodic ARP Table Updates:**
    - Continuously send spoofed ARP replies at intervals to ensure the victim's ARP table remains updated with your MAC address.

Potential Issues and Solutions

Race Condition:

- The actual router may respond to ARP requests faster than your program, resulting in the victim's ARP table not being updated with your MAC.

Solutions:

1. **Reduce Reply Latency:**
    - Ensure the program replies as quickly as possible by optimizing the code and using raw sockets for direct access to packets.
    - **Raw Sockets**:
        - Allow direct access to network packets at the IP level, bypassing higher-level protocol handling like TCP/UDP.
        - Enable faster ARP responses by reducing system delays in packet handling.
2. **Send Multiple ARP Replies:**

    - Instead of a single ARP reply, send multiple responses in quick succession. This increases the chance of overwriting the router's legitimate reply.

3. **Periodic ARP Table Updates:**
    - Continuously send spoofed ARP replies at intervals to ensure the victim's ARP table remains updated with your MAC address.

Testing the Program

1. **Monitor Traffic:**
    a. Record traffic for analysis: `sudo tcpdump -i eth0 -w traffic.pcap`
    b. Filter traffic from a specific host: `tcpdump -i eth0 -X host 172.23.0.3 -w traffic.pcap icmp`
2. **Send ARP Requests:**
    a. From the victim, ping the router `ping -c 1 172.23.0.2` and verify the program intercepts traffic.
3. **Network Discovery:**
    a. Scan local network devices: `nmap -sn 192.168.1.37/24`
4. **Flush ARP Cache:**
    a. Clear ARP table entries: `ip -s -s neigh flush all` or `arp -d 172.23.0.3`

IPv4 Decimal Conversion

- **Formula for decimal representation:**

  Decimal = $(A \times 256^3) + (B \times 256^2) + (C \times 256^1) + (D \times 256^0)$

- **Example:**

  IPv4: 192.168.1.10

  Decimal = $(192 \times 256^3) + (168 \times 256^2) + (1 \times 256^1) + (10 \times 256^0) = 3221225472 + 11010048 + 256 + 10 = 3232235786$

Hostname resolution:

- Add to /etc/hosts IP and its corresponding hostname