CS 331: COMPUTER NETWORKS

ASSIGNMENT-1

NAMES: YALLA SAI TEJA(23110366)
RAMAVATH CHANTI(23110270)

INTRODUCTION

Purpose: State the goal of the assignment, which was to understand network packet manipulation and protocol behaviour.

Task 1: Mention that you built a custom DNS client-server application to parse a .pcap file, add custom headers to DNS queries, and resolve them using a custom server.

Task 2: Mention that you conducted a comparative analysis of the traceroute utility on both Linux and Windows operating systems by capturing and inspecting the network traffic.

Task 1: Custom DNS Resolver

Methodology

- **Client (client.py):** Explain that the client was designed to read a specific .pcap file (6.pcap in your case). Describe its process:
    1. It uses the Scapy library to efficiently read the file packet by packet.
    2. It filters for DNS queries by checking for UDP packets with the DNS QR flag set to 0.
    3. For each filtered query, it generates an 8-byte custom header in the format HHMMSSID.
    4. Finally, it sends these constructed payloads (Custom Header + Original DNS Packet) to the server over a UDP socket.
- **Server (server.py):** Explain that the server listens for incoming UDP packets on a specific port (127.0.0.1:5300). Describe its process:
    1. Upon receiving a payload, it separates the 8-byte custom header from the original DNS packet data.
    2. It parses the DNS data to extract the domain name being queried.
    3. It uses Python's socket.gethostbyname() to perform a real DNS lookup for that domain.
    4. It logs the timestamp, the original query, the custom header, and the resolved IP address into a CSV file named dns_resolver.log.

Results

- This is where you show your proof. State that the system worked as expected.

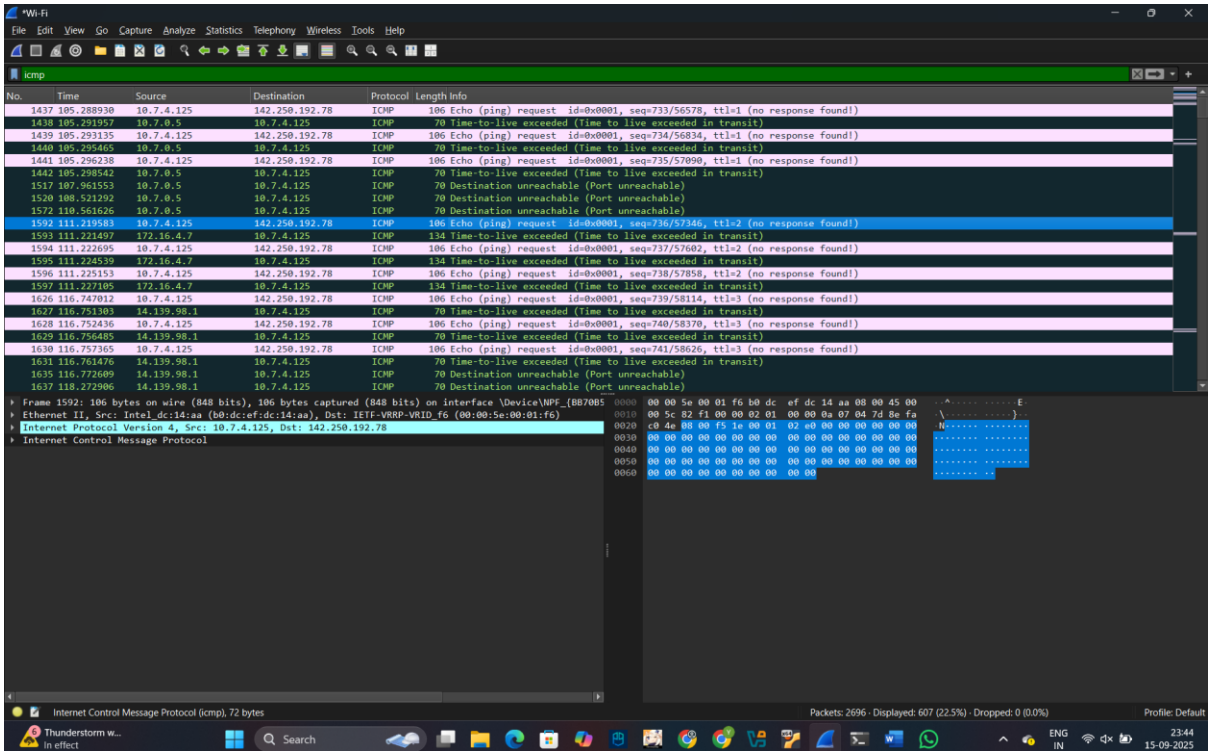
- **Include a screenshot of the final dns_resolver.log file.** This is your primary evidence for this task. Make sure the screenshot shows the header row and a good portion of the data, including both successfully resolved domains and the "Not Found" entries for local hostnames.

Task 2: Traceroute Protocol Behavior

This is where you answer the five questions, using the evidence you collected.

- **2.1. What protocol does Windows tracert use by default, and what protocol does Linux traceroute use by default?**
    - **Answer:** State clearly that Windows tracert uses **ICMP** by default, while the standard Linux traceroute uses **UDP**. Mention that you used the -I flag on Linux to force it to use **ICMP** for a successful trace.
    - **Evidence:** Include a screenshot of your **Wireshark capture from Windows**, filtered by icmp, showing the "Echo (ping) request" packets. For Linux, include the output from your **analyzer.py script**, which also shows "ICMP Type: 8 (Echo Request)" packets.
- **2.2. Provide at least two reasons why a router might not reply, resulting in * * *.**
    - **Answer:**
        1. **Firewall Configuration:** The router (or a firewall in front of it) is configured to block the incoming traceroute probes or, more commonly, the outgoing "ICMP Time Exceeded" replies.
        2. **Router Deprioritization:** The router is configured to deprioritize responding to these types of ICMP messages to prevent being used in network scanning or DDoS attacks, and the reply simply times out.
    - **Evidence:** Mention your initial attempt to run traceroute google.com on Linux, which resulted in all * * *, as a prime example of this behavior.
- **2.3. In Linux traceroute, which field in the probe packets changes between successive probes?**
    - **Answer:** The **Time To Live (TTL)** field in the IP header is the key field that is intentionally changed.
    - **Evidence:** Include a screenshot of the **output from your analyzer.py script**. **Highlight** the lines showing the TTL incrementing from TTL: 1 to TTL: 2, etc. This is direct proof.
- **2.4. At the final hop, how is the response different compared to the intermediate hops?**
    - **Answer:** An intermediate router responds with an **ICMP Type 11 (Time-to-live exceeded)** message. The final destination, however, responds with an **ICMP Type 0 (Echo reply)** message, indicating the probe successfully reached its target.
    - **Evidence:** Include a screenshot from **Wireshark (Windows)**. Highlight a "Time-to-live exceeded" packet and contrast it with the final "Echo (ping) reply" packet. Similarly, point to the output of your analyzer.py script, which explicitly labels the final packet as "ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION".

- **2.5. How would a firewall blocking UDP but allowing ICMP affect the results?**
    - **Answer:** This would cause the default **Linux traceroute to fail** (showing * * *) because it relies on UDP. However, the **Windows tracert command would work perfectly** because it uses ICMP by default. A modified Linux command (traceroute -I) would also succeed.
    - **Evidence:** Reference your own experience in this assignment as the proof. Explain that your first Linux attempt failed, but the ICMP-based trace (-I) and the Windows trace both worked.



```
set-iitgn-vm@set-iitgn-vm:~/Desktop/CN_assignment$ traceroute google.com
traceroute to google.com (142.251.222.78), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  0.245 ms  0.218 ms  0.458 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
set-iitgn-vm@set-iitgn-vm:~/Desktop/CN_assignment$ traceroute -I google.com
traceroute to google.com (142.250.70.78), 30 hops max, 60 byte packets
 1  pnbomb-ab-in-f14.1e100.net (142.250.70.78)  13.315 ms  13.296 ms  13.339 ms
```

```
set-iitgn-vm@set-iitgn-vm:~/Desktop/CN_assignment$ python3 analyzer.py
--- Analyzing 'traceroute_capture.pcap' ---
Your IP was determined to be: 10.0.2.15
------------------------------
Packet #9: YOU -> 142.250.70.78 | TTL: 1  | ICMP Type: 8 (Echo Request)
Packet #10: YOU -> 142.250.70.78 | TTL: 1  | ICMP Type: 8 (Echo Request)
Packet #11: YOU -> 142.250.70.78 | TTL: 1  | ICMP Type: 8 (Echo Request)
Packet #12: YOU -> 142.250.70.78 | TTL: 2  | ICMP Type: 8 (Echo Request)
Packet #13: YOU -> 142.250.70.78 | TTL: 2  | ICMP Type: 8 (Echo Request)
Packet #14: YOU -> 142.250.70.78 | TTL: 2  | ICMP Type: 8 (Echo Request)
Packet #15: YOU -> 142.250.70.78 | TTL: 3  | ICMP Type: 8 (Echo Request)
Packet #16: YOU -> 142.250.70.78 | TTL: 3  | ICMP Type: 8 (Echo Request)
Packet #17: YOU -> 142.250.70.78 | TTL: 3  | ICMP Type: 8 (Echo Request)
Packet #18: YOU -> 142.250.70.78 | TTL: 4  | ICMP Type: 8 (Echo Request)
Packet #19: YOU -> 142.250.70.78 | TTL: 4  | ICMP Type: 8 (Echo Request)
Packet #20: YOU -> 142.250.70.78 | TTL: 4  | ICMP Type: 8 (Echo Request)
Packet #21: YOU -> 142.250.70.78 | TTL: 5  | ICMP Type: 8 (Echo Request)
Packet #22: YOU -> 142.250.70.78 | TTL: 5  | ICMP Type: 8 (Echo Request)
Packet #23: YOU -> 142.250.70.78 | TTL: 5  | ICMP Type: 8 (Echo Request)
Packet #24: YOU -> 142.250.70.78 | TTL: 6  | ICMP Type: 8 (Echo Request)
Packet #25: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #26: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #27: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #29: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #30: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #32: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #33: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #34: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #35: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #36: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #37: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #38: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #39: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #40: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #41: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
Packet #42: 142.250.70.78 -> YOU | ICMP Type: 0 (Echo Reply) <-- FINAL DESTINATION
set-iitgn-vm@set-iitgn-vm:~/Desktop/CN_assignment$
```