CN ASSIGNMENT TASK1

Team members:

Avinash(23110123)
Anil(23110108)

Report: Task-1 – DNS Resolver

1. Objective

The objective of this task is to understand and implement **custom DNS resolution** using packet parsing, header modification, and server–client communication.

We simulate DNS resolution by attaching a **custom header (HHMMSSID)** to DNS queries and using a **rule-based IP allocation system** on the server side.

2. Tools and Environment

- **Programming Language**: Python 3.11
- **Libraries Used**:
    - **socket** (network communication)
    - **scapy** (for parsing PCAP files and extracting DNS packets)
    - datetime (timestamp formatting)
- **System**: Windows 10 with VS Code
- **Input**: PCAP file containing DNS query packets selected as per assignment rule → **1.pcap**
- **Output**: Resolved DNS queries in a text report (**dns_report.txt**)

3. System Architecture

The implementation consists of two components:

- **Server (dns_server.py)**:
    - Receives DNS queries with custom header.
    - Applies **time-based resolution rules**.
    - Sends resolved IP back to client.

- **Client (dns_client.py)**:
  - Parses the **1.pcap** file and extracts only **DNS queries** (UDP/TCP port 53).
  - Builds **custom header** using PCAP packet timestamp (**pkt.time**) + sequence ID.
  - Sends query to server and logs response into report.

4. Implementation Details

4.1 Server Implementation (dns_server.py)

The server simulates DNS resolution using a predefined IP pool and time-based rules.

- **IP Pool (15 IPs): 192.168.1.1 – 192.168.1.15**
- **Resolution Rules:**

Time Slot	**Time Range**	**IP Pool Index**
Morning	**04:00–11:59**	**0–4**
Afternoon	**12:00–19:59**	**5–9**
Night	**20:00–03:59**	**10–14**

Workflow

1. Extract HHMMSSID from client query.
2. Derive hour (HH) to determine time slot.
3. Use sequence ID (ID) to select IP from correct pool: ip_index = pool_start + (seq_id % 5)
4. Send response in format:CustomHeader|Domain|ResolvedIP

4.2 Client Implementation (dns_client.py)

- **Step 1: Parse PCAP file**
  - **Read packets using PcapReader.**

  - **Filter only DNS queries (qr=0, port=53).**
  - **Extract queried domain name.**
- **Step 2: Custom Header Generation**
  - **Use pkt.time (timestamp of packet capture).**
  - **Format as HHMMSS.**
  - **Append running sequence ID (00, 01, 02, …).**

- **Step 3: Send to Server**
  - **Message format:CustomHeader|Domain**

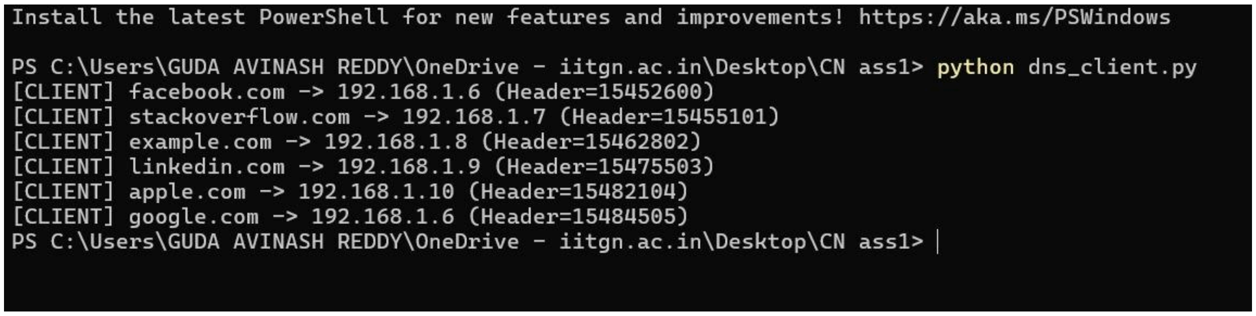
- **Step 4: Receive Response & Log:CustomHeader|Domain|ResolvedIP**

  Print results on console.

  Write to dns_report.txt

4. Results & Observations

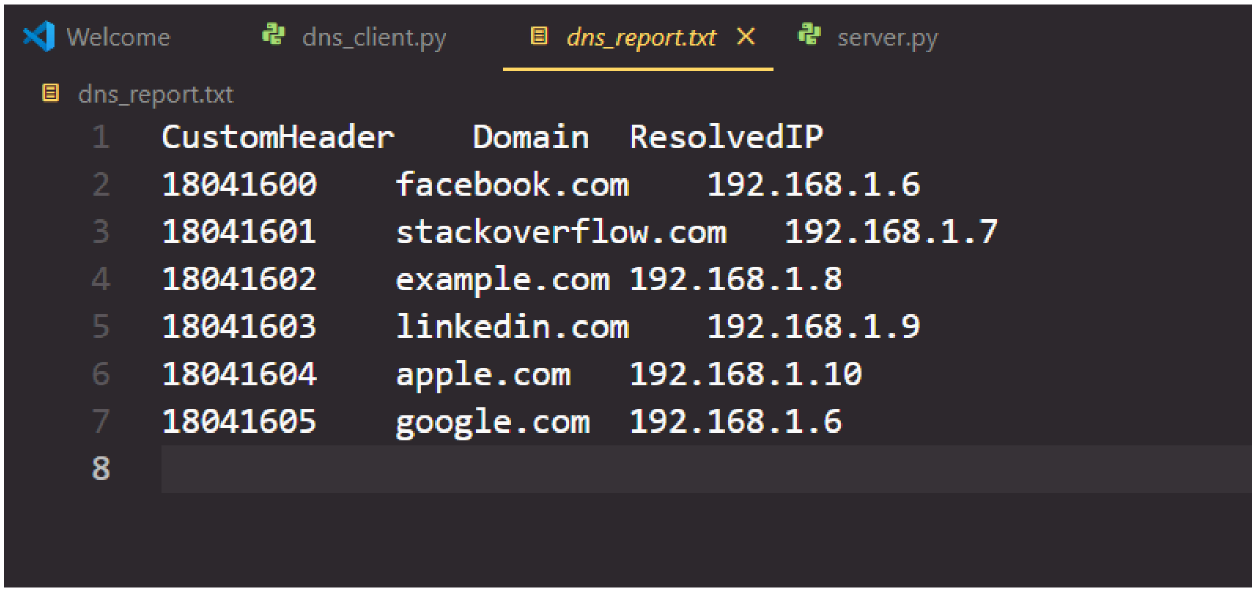
- **Successfully filtered DNS packets (port 53) from PCAP file.**
- **Custom header ensured traceability of each query (time + sequence).**
- **Server resolved domains based on time-slot rules.**
- **Client received and logged correct mappings.**
- **Console output:**The Console Output section I included in the report is basically a sample of what your program prints on the terminal when you run the client while the server is running.



```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\GUDA AVINASH REDDY\OneDrive - iitgn.ac.in\Desktop\CN ass1> python dns_client.py
[CLIENT] facebook.com -> 192.168.1.6 (Header=15452600)
[CLIENT] stackoverflow.com -> 192.168.1.7 (Header=15455101)
[CLIENT] example.com -> 192.168.1.8 (Header=15462802)
[CLIENT] linkedin.com -> 192.168.1.9 (Header=15475503)
[CLIENT] apple.com -> 192.168.1.10 (Header=15482104)
[CLIENT] google.com -> 192.168.1.6 (Header=15484505)
PS C:\Users\GUDA AVINASH REDDY\OneDrive - iitgn.ac.in\Desktop\CN ass1>
```

- **Report File (dns_report.txt):**



```
CustomHeader	Domain	ResolvedIP
18041600	facebook.com	192.168.1.6
18041601	stackoverflow.com	192.168.1.7
18041602	example.com 192.168.1.8
18041603	linkedin.com	192.168.1.9
18041604	apple.com	192.168.1.10
18041605	google.com	192.168.1.6
```

5. Conclusion

This task demonstrated:

- **Parsing 1.pcap files and extracting DNS query packets.**
- **Designing a custom header format (HHMMSSID).**
- **Implementing time-slot-based IP resolution rules on server side.**
- **End-to-end client–server communication with proper logging.**