Assignment-1

CS 331: Computer Networks

Shardul Junagade (23110297)   Rishabh Jogani (23110276)

Repository: cn-assignment-1

September 15, 2025

Contents

Task 1: Custom DNS Resolver

1.1 Aim

The aim of this task was to implement a custom DNS resolver system using both the Scapy and dpkt Python libraries. The objective was to parse DNS query packets from a given PCAP file, add a custom header, send them to a server for resolution based on predefined rules, and log the resolved IP addresses for reporting. This task helped us understand packet parsing, custom protocol design, and time-based routing logic.

1.2 Introduction

In this assignment, we developed a client-server system to process DNS queries captured in a PCAP file. The client parsed DNS query packets, prepended a custom header containing a timestamp and sequence ID, and sent these packets to a custom server. The server extracted the header, applied time-based routing rules to select an IP address, and responded with the resolved result. We implemented the solution using two different packet processing libraries: Scapy and dpkt, to demonstrate versatility and robustness.

The custom header format was `HHMMSSID`, where `HH` is the hour, `MM` is the minute, `SS` is the second, and `ID` is the sequence number of the DNS query. The server used this header to determine the time slot and select an IP address from a pool, as specified in a JSON rules file.

1.3 Procedure

1.3.1 PCAP Parsing and DNS Query Extraction

We first parsed the provided PCAP file to extract DNS query packets. For this, we implemented two separate clients:

- **Scapy-based client** (`client_scapy.py`)**:** Used Scapy's `PcapReader` and DNS layers to filter and extract DNS queries. Example code snippet:

```
for pkt in PcapReader(pcap_path):
    if (pkt.haslayer(DNS) and pkt[DNS].qr == 0):
        qname = pkt[DNS].qd.qname.decode()
        if not qname.endswith(".local.") and not qname.startswith("_"):
            dns_pkts.append(pkt)
```

- **dpkt-based client** (`client_dpkt.py`)**:** Used dpkt's PCAP and DNS parsing utilities:

```
with open(pcap_path, "rb") as f:
    pcap = dpkt.pcap.Reader(f)
    for ts, buf in pcap:
        eth = dpkt.ethernet.Ethernet(buf)
        ip = eth.data
        udp = ip.data
```

```
            if isinstance(udp, dpkt.udp.UDP) and udp.dport == 53:
                dns = dpkt.dns.DNS(udp.data)
                ...
```

1.3.2 Custom Header Construction

For each DNS query, we constructed an 8-byte custom header in the format `HHMMSSID`, where:

- `HH`: Hour (24-hour format)
- `MM`: Minute
- `SS`: Second
- `ID`: Sequence number of the DNS query (starting from 00)

This header was prepended to the raw DNS packet bytes. Example (Scapy):

```
hhmmss = datetime.now().strftime("%H%M%S")
sid = f"{i:02d}"
header = hhmmss + sid
payload = header.encode('ascii') + dns_bytes
```

1.3.3 Client-Server Communication

The client sent the modified packet (header + DNS query) to the server over UDP. The server was implemented in two variants:

- **Scapy-based server (`server_scapy.py`):** Used Scapy to parse incoming DNS queries and extract the domain name.
- **dpkt-based server (`server_dpkt.py`):** Used dpkt for the same purpose.

Example server code (Scapy):

```
header = data[:8].decode('ascii', errors='ignore')
dns_bytes = data[8:]
dns_query = DNS(dns_bytes)
qname_bytes = dns_query.qd.qname if dns_query.qd else b""
qname_str = qname_bytes.decode(errors='ignore')
```

1.3.4 Server-side IP Resolution

Upon receiving a packet, the server extracted the custom header and used the hour and sequence ID to select an IP address from a pool, according to time-based rules defined in a JSON file (`rules.json`). The rules specified different IP pool ranges and hash functions for morning, afternoon, and night slots. The logic was as follows:

```
hour = int(header[0:2])
session_id = int(header[6:8])
if 4 <= hour <= 11:
    slot_cfg = time_rules.get("morning")
elif 12 <= hour <= 19:
    slot_cfg = time_rules.get("afternoon")
else:
    slot_cfg = time_rules.get("night")
index = ip_pool_start + (session_id % hash_mod)
```

The rules and pool configuration were as described in the assignment and the provided rules documentation.

1.3.5 Response and Logging

The server responded with a plain text message in the format `header|domain|resolved_ip`. The client received this response and logged the results in a CSV file for reporting. Both implementations (Scapy and dpkt) followed this protocol.

1.4 Results

The system successfully parsed DNS queries from the PCAP file, sent them to the server with the custom header, and received resolved IP addresses based on the specified rules. The results were saved in CSV files (`dpkt_dns_report.csv` and `scapy_dns_report.csv`), each containing the custom header, domain name, and resolved IP address for every query.

- Both Scapy and dpkt implementations produced consistent results, demonstrating the correctness of the approach.
- The server correctly applied time-based routing rules and handled edge cases (e.g., out-of-bounds indices).
- The code was modular, well-commented, and robust against malformed packets.

Sample output (CSV):

Table 1.1: DNS Resolution Results

Custom header value (HHMMSSID)	Domain name	Resolved IP address
20464700	netflix.com	192.168.1.11
20464701	linkedin.com	192.168.1.12
20464702	example.com	192.168.1.13
20464703	google.com	192.168.1.14
20464704	facebook.com	192.168.1.15
20464705	amazon.com	192.168.1.11

DNS Resolution Rule Example:

- If the hour is 20 (night slot), the pool start is 10, and hash mod is 5. For ID 00: $10 + (00\%5) = 10 \rightarrow$ 192.168.1.11
- For ID 01: $10 + (01\%5) = 11 \rightarrow$ 192.168.1.12
- For ID 04: $10 + (04\%5) = 14 \rightarrow$ 192.168.1.15

Repository: `https://github.com/ShardulJunagade/cn-assignment-1`

Task 2: Pydriller Tool

2.1 Aim

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

2.2 References

[1] Git Documentation

[2] GitHub Guides