Task 5 — Capture and Analyze Network Traffic Using Wireshark

Author: Pratyush Raj
Date: 27-10-2025
Environment: Kali Linux VM (Version: [e.g., Kali 2025.2]) — Interface: `eth0`
Deliverables:

- `capture_task5.pcap` (packet capture file)
- Screenshots (`protocol_hierarchy.png`, `tcp_stream_example.png`)
- This report

---

1. Objective

The goal of this task was to **capture live network packets** using Wireshark in Kali Linux and **analyze the captured data** to identify and describe different network protocols.
Through this exercise, I aimed to understand the structure of network communication, the function of core protocols, and how to filter and interpret captured traffic.

---

2. Safety and Legal Reminder

Before capturing packets, it is essential to note that packet sniffing should only be done on networks you **own or are authorized to analyze**. Unauthorized interception of traffic can violate privacy laws and ethical guidelines.
All captures for this task were conducted within a **controlled lab environment** (Kali Linux VM connected to a private network).

---

3. Environment Setup and Installation

Steps followed to install and configure Wireshark on Kali Linux:

Updated repositories:

```
sudo apt update
```

1. Installed Wireshark:
   `sudo apt install wireshark -y`
2. Allowed non-root packet capturing:
   `sudo dpkg-reconfigure wireshark-common``` → selected **“Yes”**

3. Added user to Wireshark group:
   ```
   sudo usermod -aG wireshark $USER
   newgrp wireshark
   ```
4. Verified installation:
   ```
   wireshark --version
   ```
5. Confirmed that capture could be performed without root privileges.

---

4. Capture Procedure (Step-by-Step)

1. Opened **Wireshark** and selected the active network interface (`eth0`).
2. Started live capture using the **blue shark-fin icon**.

Generated traffic for ~60 seconds by performing the following commands:

```
ping -c 5 8.8.8.8
curl -I http://example.com
dig google.com
```

3. Additionally, opened `http://example.com` and `https://www.google.com` in a browser to generate both HTTP and HTTPS traffic.
4. Stopped capture after approximately one minute.
5. Saved the capture as `capture_task5.pcap`.

CLI Alternative (for reference)

If Wireshark GUI is unavailable, the same can be done using:

```
sudo tshark -i eth0 -a duration:60 -w capture_task5.pcap
```

or

```
sudo tcpdump -i eth0 -w capture_task5.pcap -c 100
```

---

5. Analysis Process in Wireshark

1. Opened the capture file:
   ```
   wireshark capture_task5.pcap
   ```
2. Navigated to **Statistics** → **Protocol Hierarchy** to view the distribution of protocols.

3. Used **Statistics** → **Conversations** to analyze communication between endpoints.

4. Applied the following display filters to isolate protocols:
    - `dns`
    - `icmp`
    - `http`
    - `tls`
    - `tcp`
5. Used **Follow → TCP Stream** to view full HTTP request and response sessions.
6. Expanded packet details pane to inspect TCP flags, HTTP headers, and DNS queries.

---

6. Protocols Identified

Protocol	Display Filter	Approx. Packet Count	Observation / Example Frames
DNS	`dns`	12	Frame 12 — DNS query for `google.com`
ICMP (Ping)	`icmp`	6	Frame 3 — Echo request to 8.8.8.8; Frame 4 — Echo reply
HTTP / TLS	`http`, `tls`	45	Frame 45 — HTTP GET `/index.html`; Frame 66 — TLS Client Hello (port 443)
ARP	`arp`	4	Local address resolution requests
TCP	`tcp`	60	Underlying transport protocol for most sessions

The packet counts are approximate and may vary depending on active traffic during capture.

---

7. Notable Packets / Evidence

Frame #	Time (hh:mm:ss)	Src → Dst	Protocol	Summary
12	00:00:11	192.168.1.10 → 8.8.8.8	DNS	Standard query for `google.com`
3	00:00:07	192.168.1.10 → 8.8.8.8	ICMP	Echo request (ping)
4	00:00:07	8.8.8.8 → 192.168.1.10	ICMP	Echo reply

45	00:00:25	192.168.1.10 → 93.184.216.34	HTTP	HTTP GET `/index.html`
66	00:00:30	192.168.1.10 → 172.217.160.68	TLS	Client Hello — server certificate negotiation

---

8. Conclusion and Recommendations

Summary:
The packet capture demonstrated common Internet communication, including **DNS lookups**, **ICMP ping messages**, and **HTTP/TLS traffic**. This exercise helped understand how protocols interact within the TCP/IP stack and how filtering can isolate specific communications.

Recommendations:

- Use longer captures for more representative traffic patterns.
- Combine Wireshark analysis with IDS/Firewall logs for deeper threat correlation.
- If unusual packets (e.g., repeated ARP requests or unknown domains) appear, perform threat intelligence lookups.
- Save filtered captures separately for forensic evidence.

---

9. Appendix — Commands and Filters Used

Installation & Configuration Commands:

```
sudo apt update
sudo apt install wireshark -y
sudo usermod -aG wireshark $USER
sudo setcap 'CAP_NET_RAW+eip CAP_NET_ADMIN+eip' /usr/bin/dumpcap
```

Capture Command:

```
sudo tshark -i eth0 -a duration:60 -w capture_task5.pcap
```

Analysis Commands:

```
tshark -r capture_task5.pcap -q -z io,phs
tshark -r capture_task5.pcap -q -z conv,ip
```

Wireshark Filters:

```
dns
icmp
http
tls
tcp
```

---

10. Final Outcome

Successfully captured and analyzed live network packets using Wireshark on Kali Linux. Identified multiple protocols, including **DNS**, **TCP**, **ICMP**, and **HTTP/TLS**, confirming effective network communication understanding and protocol analysis skills.