Objective

The objective of this lab is to capture live network traffic using Wireshark on Kali Linux, filter and identify various network protocols, and analyze the traffic to understand communication between network devices.

Tools and Environment

- **Wireshark** – GUI-based network protocol analyzer
- **Kali Linux** – Security-focused Linux distribution
- **Firefox** – Used to generate HTTP traffic
- **Terminal (Bash)** – Used to generate ICMP traffic (ping)

Procedure

1. Install and Launch Wireshark

Wireshark was installed (if not already) and launched using the following command:

sudo wireshark

Wireshark was granted permission to capture traffic on the active interface.

---

2. Identify and Select Network Interface

Used:

ip a

to identify the active interface (eth0 in this case). Selected this interface in Wireshark for live packet capture.

---

3. Start Packet Capture

Live capture was started by clicking the capture icon in Wireshark on the selected interface.

---

4. Generate Network Traffic

Traffic was manually generated using:

- **Ping (ICMP):**

ping google.com

- **Web browsing (HTTP/DNS):**
    - Visited https://example.com via Firefox to generate DNS and HTTP traffic.

---

5. Stop Capture

After ~60 seconds, packet capture was stopped.

---

6. Apply Protocol Filters

Applied filters to identify and analyze:

- DNS traffic: dns
- HTTP traffic: http
- ICMP traffic: icmp
- TCP handshakes and sessions: tcp

Analysis and Findings

1. DNS (Domain Name System)

- **Port:** UDP 53
- **Observation:** DNS query for example.com and google.com.
- **Packet Example:**

less

Standard query 0x37d1 A google.com

Response: A record for google.com -> 142.250.190.78

---

2. HTTP (HyperText Transfer Protocol)

- **Port:** TCP 80
- **Observation:** HTTP GET request when accessing example.com.
- **Packet Example:**

makefile

GET / HTTP/1.1

Host: example.com

User-Agent: Mozilla/5.0 (Linux x86_64; rv:89.0)

---

3. ICMP (Internet Control Message Protocol)

- **No Port** (network layer protocol)
- **Observation:** Echo requests and replies using the ping command.
- **Packet Example:**

Echo (ping) request id=0x01 seq=1

Echo (ping) reply id=0x01 seq=1

---

4. TCP (Transmission Control Protocol)

- **Observation:** TCP 3-way handshake observed prior to HTTP data exchange.
- **Packet Sequence:**

arduino

1. SYN (Client -> Server)

2. SYN-ACK (Server -> Client)

3. ACK (Client -> Server)

Capture File

- File Name: network_traffic.pcap
- Size: ~500 KB (varies based on traffic)
- Protocols Included: DNS, HTTP, TCP, ICMP

Key Takeaways

- Wireshark effectively captures and dissects network communication.
- DNS requests precede HTTP traffic as domain names are resolved.
- HTTP GET requests clearly show human-readable URLs and headers.

- ICMP packets are useful for diagnostic purposes (reachability and latency).
- TCP ensures reliable communication through handshakes and ACKs.

Conclusion

This activity demonstrated how to:

- Use Wireshark to capture real-time network traffic
- Apply filters to isolate protocol-specific packets
- Interpret packet contents and understand network behavior
- Identify and analyze at least 3 core Internet protocols

References

- Wireshark Official Documentation: https://www.wireshark.org/docs/
- Protocol Port Numbers: IANA Service Name and Port Number Registry