



**NAME: SHUBHAM MAWASKAR**

## **TASK1: BASIC NETWORK ANIFFER**

**Project Name:** Build a network sniffer in Python that captures and analyzes network traffic.

### **Table of Contents**

<b>No</b>	<b>Title of Contents</b>	<b>Page No.</b>
<b>1</b>	<b>Summary</b>	
<b>2</b>	<b>Technology Used</b>	
<b>3</b>	<b>Objectives of Study</b>	
<b>4</b>	<b>Analysis Details</b>	
<b>5</b>	<b>Conclusion</b>	

### **SUMMARY:**

The packet capture effectively showcases the role of essential protocols in maintaining a secure network environment, confirming that the client-server interaction is safeguarded by industry-standard cryptographic techniques, ensuring the integrity, confidentiality, and reliability of the exchanged data.

The packet capture provided illustrates the step-by-step process of how a secure HTTPS connection is established between a client device and a remote web server. It highlights key network protocols involved in this process and the roles they play in ensuring reliable and secure communication. This report analyses a packet capture from a network session between a client device (192.168.100.7) and a remote server (18.66.41.41). The packet capture involves multiple protocols working together to establish a secure HTTPS connection. The traffic flow shows standard steps such as DNS resolution, TCP handshake, and TLS encryption.

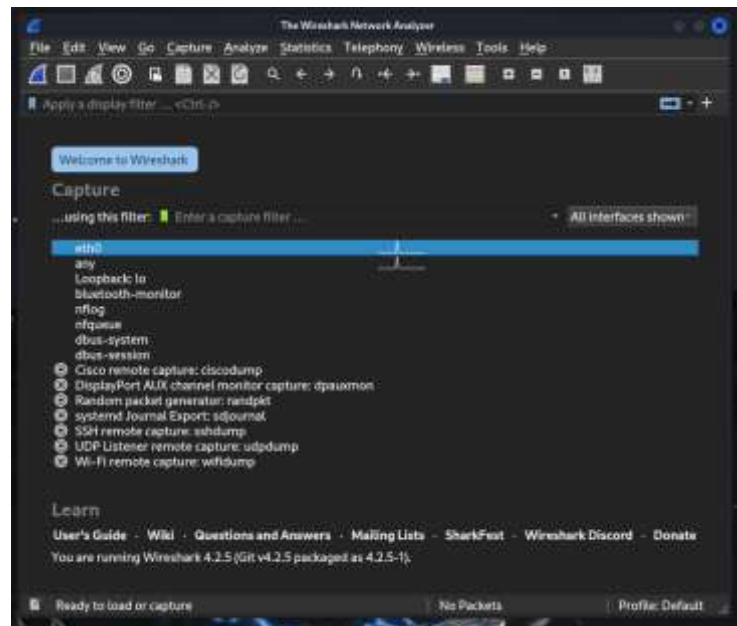
## TECHNOLOGY USED:

- **OS:** Kali Linux



- **SOFTWARE:** Wireshark

```
root@kali: /home/kali
File Actions Edit View Help
(kali@kali)-[~]
└─$ sudo su
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
root@kali: /home/kali
└─$ sudo wireshark
** (wireshark:2517) 03:48:23.481225 [GUI WARNING] -- QStandardPaths: XDG_RUN
TIME_DIR not set, defaulting to '/tmp/runtime-root'
** (wireshark:2517) 03:49:27.888547 [Capture MESSAGE] -- Capture Start ...
** (wireshark:2517) 03:49:27.956571 [Capture MESSAGE] -- Capture started
** (wireshark:2517) 03:49:27.956674 [Capture MESSAGE] -- file: "/tmp/wiresha
rk_eth0MW30V2.pcapng"
```



## OBJECTIVE of STUDIES:

Gather Information about which Protocol, Network and Data Flow are using this analysis for generate the analysis report.

### 1. Protocol Observed:

- **DNS (Domain Name System)** is used to translate domain names into IP addresses so the client can communicate with the correct server.
- **TCP (Transmission Control Protocol)** is responsible for establishing a reliable connection between the client and server using a 3-way handshake.
- **TLSv1.3 (Transport Layer Security)** provides encryption for the data exchanged, ensuring that the communication remains secure and private.
- **Ethernet II** manages local data transmission over the physical network.
- **IPv4 (Internet Protocol)** routes packets across networks using IP addresses.
- **HTTPS (HTTP over TLS)** is used to ensure that all web traffic between the client and server is encrypted.

## 2. Network Architecture:

- The **client** device is part of a private network, using a **DNS server** to resolve domain names.
- A **router/gateway** connects the private network to the public internet, allowing the client to reach external services.
- The **remote server** is a public IP on the internet, providing secure web services via HTTPS.

## 3. Packet Flow:

- **DNS queries** and responses allow the client to resolve the domain name to an IP address.
- The **TCP handshake** ensures a reliable connection by exchanging SYN, SYN-ACK, and ACK messages.
- The **TLS handshake** follows, where the client and server negotiate encryption methods, ensuring all further data is secure.
- Once the handshake is complete, **encrypted HTTPS traffic** flows between the client and server.

## ANALYSIS DETAILS:

### 1. Protocol Analysis:

- **DNS (Domain Name System):**
  - **Layer:** Application Layer (Layer 7)
  - **Role:** Resolves the domain name app.link to an IP address, allowing the client to locate the server.
  - **Packets:** Packets 0, 1, 3 show DNS queries and responses between the client and DNS server (172.16.1.40).

➤ **TCP (Transmission Control Protocol):**

- **Layer:** Transport Layer (Layer 4)
- **Role:** Manages the establishment of a reliable connection using the TCP 3-way handshake.
- **Packets:** Packets 10-12 represent the 3-way handshake: SYN (Packet 10), SYN-ACK (Packet 11), and ACK (Packet 12).

➤ **TLSv1.3 (Transport Layer Security):**

- **Layer:** Application Layer (Layer 7)
- **Role:** Secures communication between the client and server using encryption. The TLS handshake negotiates encryption parameters.
- **Packets:** Packets 12 (Client Hello) and 13 (Server Hello) show the start of the TLS handshake, which ensures secure encrypted communication.

➤ **Ethernet II:**

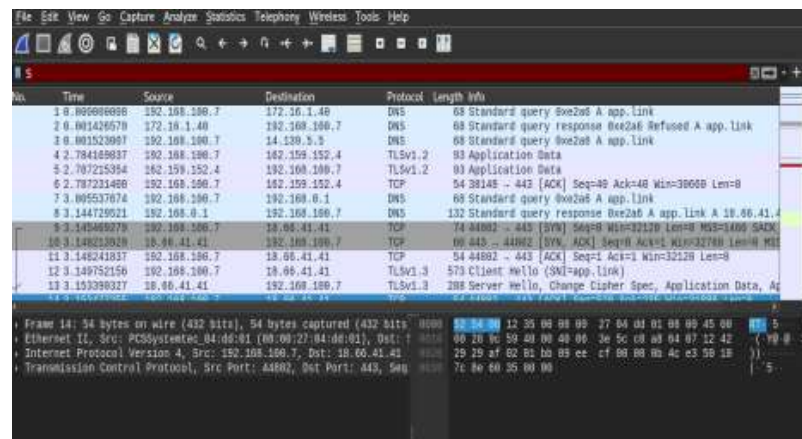
- **Layer:** Data Link Layer (Layer 2)
- **Role:** Provides local network frame delivery, encapsulating packets for transmission across the local area network (LAN).
- **Presence:** Ethernet II headers are present in every packet in the capture.

➤ **IPv4 (Internet Protocol):**

- **Layer:** Network Layer (Layer 3)
- **Role:** Routes packets between the client and the server using IP addresses, enabling cross-network communication.
- **Presence:** Present in all packets, showing the source (192.168.100.7) and destination (18.66.41.41) IP addresses.

## ➤ HTTPS (HTTP over TLS):

- **Layer:** Application Layer (Layer 7)
- **Role:** Securely transmits web traffic over an encrypted connection (TLS). The raw HTTPS data is encrypted and therefore unreadable after the TLS handshake.
- **Implied Traffic:** Following Packet 13, the encrypted application data would be transmitted.



The image shows a Wireshark packet capture of an HTTPS connection. The packet list on the left shows packets 1 through 14. Packet 13 is selected, showing a TLSv1.3 packet from 192.168.100.7 to 18.66.41.41. The packet details pane on the right shows the structure of the TLSv1.3 packet, including the Client Hello, Server Hello, Change Cipher Spec, and Application Data. The packet bytes pane at the bottom shows the raw data of the selected packet, which is encrypted.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.100.7	172.16.1.40	DNS	68	Standard query 0xe2a0 A app.link
2	0.001426578	172.16.1.40	192.168.100.7	DNS	68	Standard query response 0xe2a0 Refused A app.link
3	0.001523967	192.168.100.7	14.139.3.3	DNS	68	Standard query 0xe2a0 A app.link
4	2.784169837	192.168.100.7	162.159.152.4	TLSv1.2	93	Application Data
5	2.787225354	162.159.152.4	192.168.100.7	TLSv1.2	93	Application Data
6	2.787231488	192.168.100.7	162.159.152.4	TCP	54	3814E -> 443 [ACK] Seq=40 Ack=48 Win=38068 Len=0
7	3.005537674	192.168.100.7	192.168.0.1	DNS	68	Standard query 0xe2a0 A app.link
8	3.144725021	192.168.0.1	192.168.100.7	DNS	132	Standard query response 0xe2a0 A app.link A 18.66.41.41
9	3.145469279	192.168.100.7	18.66.41.41	TCP	74	44802 -> 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1460 SACK
10	3.146213928	18.66.41.41	192.168.100.7	TCP	60	4482 -> 44802 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS
11	3.146241037	192.168.100.7	18.66.41.41	TCP	54	44802 -> 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0
12	3.149752156	192.168.100.7	18.66.41.41	TLSv1.3	573	Client Hello (3N1-app.link)
13	3.153398327	18.66.41.41	192.168.100.7	TLSv1.3	288	Server Hello, Change Cipher Spec, Application Data, Ar
14	3.155779022	192.168.100.7	18.66.41.41	TCP	54	44802 -> 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0 MSS

## 2. Network Architecture Analysis:

- **Client Device (192.168.100.7):** This device is located within a private network, as indicated by its private IP address.
- **DNS Server (172.16.1.40):** An internal DNS server that resolves domain names for the client. This server is also part of the private network.
- **Router/Gateway:** Likely present between the client and the internet, performing Network Address Translation (NAT) to allow the client's private IP to interact with public servers.
- **Remote Server (18.66.41.41):** A public web server accessed by the client over HTTPS for secure communication.

The image shows a Wireshark packet capture of a network session. The main pane displays a list of 13 packets. Packets 0, 1, and 3 are DNS queries from 192.168.100.7 to 172.16.1.40. Packets 2, 4, 6, 7, and 8 are DNS responses from 172.16.1.40 to 192.168.100.7. Packets 9, 10, 11, and 12 are TCP packets forming a handshake between 192.168.100.7 and 18.66.41.41. Packet 13 is a TLSv1.3 Client Hello from 192.168.100.7 to 18.66.41.41. The bottom pane shows the details of the selected packet (13), including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol fields.

No.	Time	Source	Destination	Protocol	Length	Info
0	0.000000000	192.168.100.7	172.16.1.40	DNS	68	Standard query 0xe2a6 A app.link
1	0.001426578	172.16.1.40	192.168.100.7	DNS	68	Standard query response 0xe2a6 Refused A app.link
2	0.001523997	192.168.100.7	14.130.5.5	DNS	68	Standard query 0xe2a6 A app.link
4	2.784169837	192.168.100.7	162.159.152.4	TLSv1.2	93	Application Data
5	2.787215354	162.159.152.4	192.168.100.7	TLSv1.2	93	Application Data
6	2.787231488	192.168.100.7	162.159.152.4	TCP	54	38148 → 443 [ACK] Seq=49 Ack=48 Win=38608 Len=0
7	3.005537674	192.168.100.7	192.168.0.1	DNS	68	Standard query 0xe2a6 A app.link
8	3.144729521	192.168.0.1	192.168.100.7	DNS	132	Standard query response 0xe2a6 A app.link A 18.66.41.41
9	3.145489279	192.168.100.7	18.66.41.41	TCP	74	44882 → 443 [SYN] Seq=8 Win=32128 Len=0 MSS=1460 SACK
10	3.146213929	18.66.41.41	192.168.100.7	TCP	66	443 → 44882 [SYN, ACK] Seq=1 Ack=1 Win=32128 Len=0 MSS
11	3.146241837	192.168.100.7	18.66.41.41	TCP	54	44882 → 443 [ACK] Seq=1 Ack=1 Win=32128 Len=0
12	3.149752150	192.168.100.7	18.66.41.41	TLSv1.3	573	Client Hello (SN=app.link)
13	3.153398327	18.66.41.41	192.168.100.7	TLSv1.3	208	Server Hello, Change Cipher Spec, Application Data, A

### 3. Packet Flow and Analysis:

- **DNS Resolution:** The client sends DNS queries to the internal DNS server, which resolves the domain name to the server's IP (Packets 0, 1, 3).
- **TCP Handshake:** A 3-way handshake is initiated by the client, ensuring reliable communication with the remote server. The TCP handshake is completed with SYN, SYN-ACK, and ACK packets (Packets 10-12).
- **TLS Handshake:** The TLSv1.3 handshake begins, where the client and server exchange cryptographic information to establish a secure, encrypted connection (Packets 12-13). Once completed, this enables secure communication between the two parties.
- **Encrypted HTTPS Traffic:** After the TLS handshake, encrypted data is exchanged between the client and the server, though it cannot be analyzed as it is encrypted.

This image is a duplicate of the one above, showing the same Wireshark packet capture. It details the DNS resolution process, the TCP 3-way handshake, and the start of the TLSv1.3 handshake between a client (192.168.100.7) and a server (18.66.41.41).

No.	Time	Source	Destination	Protocol	Length	Info
0	0.000000000	192.168.100.7	172.16.1.40	DNS	68	Standard query 0xe2a6 A app.link
1	0.001426578	172.16.1.40	192.168.100.7	DNS	68	Standard query response 0xe2a6 Refused A app.link
2	0.001523997	192.168.100.7	14.130.5.5	DNS	68	Standard query 0xe2a6 A app.link
4	2.784169837	192.168.100.7	162.159.152.4	TLSv1.2	93	Application Data
5	2.787215354	162.159.152.4	192.168.100.7	TLSv1.2	93	Application Data
6	2.787231488	192.168.100.7	162.159.152.4	TCP	54	38148 → 443 [ACK] Seq=49 Ack=48 Win=38608 Len=0
7	3.005537674	192.168.100.7	192.168.0.1	DNS	68	Standard query 0xe2a6 A app.link
8	3.144729521	192.168.0.1	192.168.100.7	DNS	132	Standard query response 0xe2a6 A app.link A 18.66.41.41
9	3.145489279	192.168.100.7	18.66.41.41	TCP	74	44882 → 443 [SYN] Seq=8 Win=32128 Len=0 MSS=1460 SACK
10	3.146213929	18.66.41.41	192.168.100.7	TCP	66	443 → 44882 [SYN, ACK] Seq=1 Ack=1 Win=32128 Len=0 MSS
11	3.146241837	192.168.100.7	18.66.41.41	TCP	54	44882 → 443 [ACK] Seq=1 Ack=1 Win=32128 Len=0
12	3.149752150	192.168.100.7	18.66.41.41	TLSv1.3	573	Client Hello (SN=app.link)
13	3.153398327	18.66.41.41	192.168.100.7	TLSv1.3	208	Server Hello, Change Cipher Spec, Application Data, A

## **CONCLUSION:**

The analysis of the packet capture provides a clear breakdown of the communication process between a client device and a remote web server. The capture highlights the involvement of multiple key protocols, including DNS, TCP, TLS, IPv4, and HTTPS, and demonstrates how they work together to establish a secure and reliable HTTPS connection.

This secure communication process begins with DNS resolution, followed by the TCP 3-way handshake, and finally, the TLS handshake that encrypts subsequent traffic. The successful establishment of a secure HTTPS connection ensures that the data transmitted between the client and server remains confidential and protected from unauthorized access.