

---

## Wireshark Traffic Analysis Report

**Internship:** Cyber Security Internship

**Task Number:** 5

**Name:** Dhruvik Variya

**Date:** 30 June 2025

**Tool Used:** Wireshark

---

### Objective

To perform live packet capture on an active network interface using Wireshark and analyze network traffic by identifying key protocols such as **ICMP**, **DNS**, and **TCP**. This task aims to improve hands-on skills in traffic analysis and protocol behavior observation.

---

### Tools & Environment

- **Operating System:** Windows
  - **Software:** Wireshark GUI (Dark Theme enabled)
  - **Capture File Format:** .pcapng
  - **Network Activity Simulated:**
    - ping to generate ICMP traffic
    - Browsing websites like YouTube and Google for DNS and TCP traffic
- 

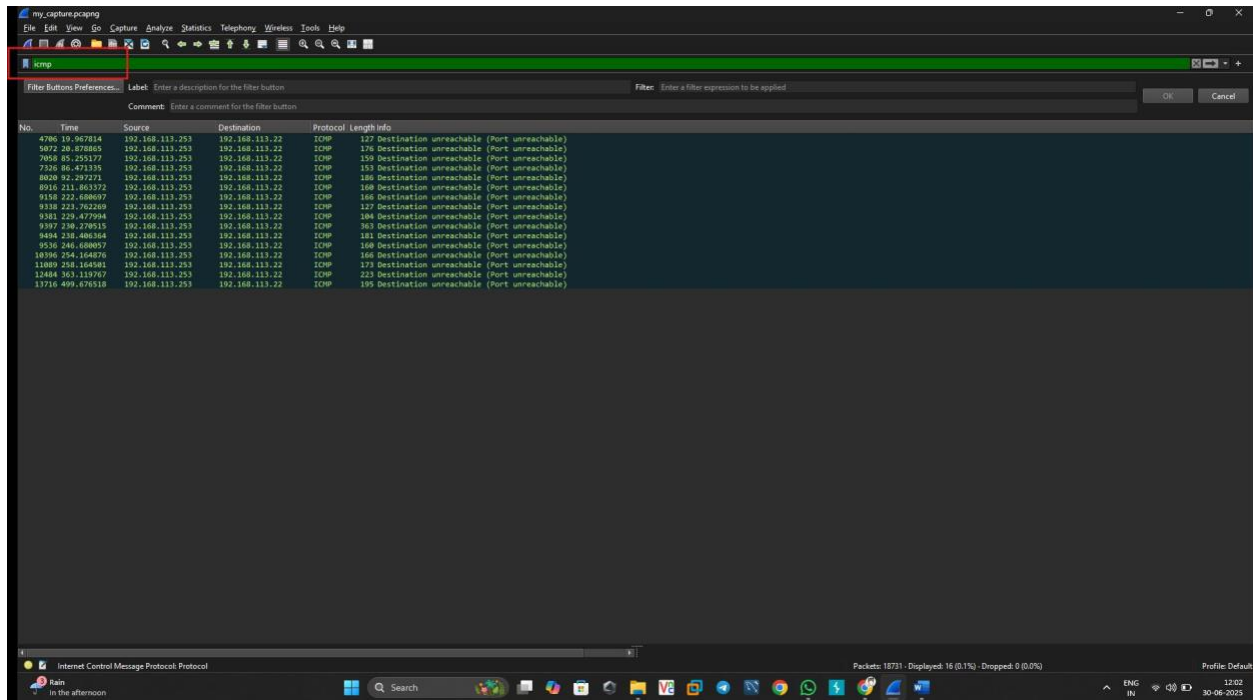
### Protocol Analysis

---

#### 1. ICMP – Internet Control Message Protocol

- **Wireshark Filter Used:** icmp
- **Captured Behavior:**
  - All ICMP packets show: Destination Unreachable (Port unreachable)
  - **Source IP:** 192.168.113.253
  - **Destination IP:** 192.168.113.22
- **Explanation:**
  - ICMP is used for diagnostics and error messages.

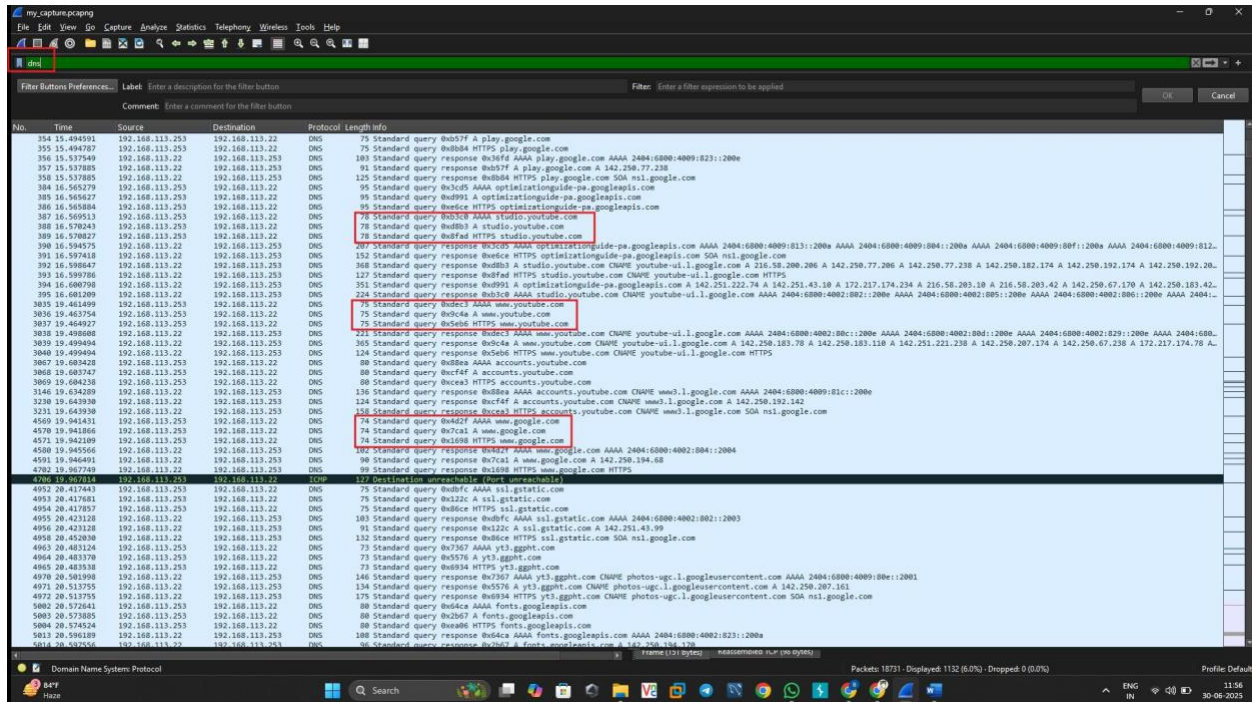
- These ICMP packets likely resulted from trying to reach services on closed or blocked ports.
- **Use Case:**
  - Helps detect unreachable hosts or services
  - Common in ping and traceroute utilities
- 



## 2. DNS – Domain Name System

- **Wireshark Filter Used:** dns
- **Captured Behavior:**
  - Standard query and response packets observed.
  - Query targets included:
    - www.google.com
    - www.youtube.com
    - studio.youtube.com
    - optimizationguide-pa.googleapis.com
  - Various **record types** captured:

- AAAA: IPv6 Address
- CNAME: Canonical Name (redirect)
- A: IPv4 Address
- **Explanation:**
  - DNS is critical for resolving human-readable domains to IP addresses.
  - It's usually the first step in browsing activity.
- **Use Case:**
  - Essential for network forensics to track what websites or services are being accessed.
  - Attackers also exploit DNS for tunneling or exfiltration.



### 3. TCP – Transmission Control Protocol

- **Wireshark Filter Used:** tcp
- **Captured Behavior:**
  - TCP 3-way handshakes: SYN, SYN-ACK, ACK
  - Many connections observed on **port 443** (HTTPS)
  - Protocol stack includes:
    - TLSv1.3 handshakes

- Application Data (encrypted traffic)

## • Explanation:

- TCP ensures reliable delivery of data using acknowledgement and retransmission.
- Encrypted communication is common on port 443.

## • Use Case:

- Useful for detecting secure traffic, session reassembly, and performance debugging.
- In real-world forensics, TCP reassembly can reveal full HTTP content or downloads.

The image shows a Wireshark packet capture of TCP traffic. The filter is set to 'tcp'. The packet list shows a series of TCP segments from 2049:40c1:15:28:c8:11 to 2049:40c1:15:28:c8:11. The packet details show the TCP header and application data. The packet bytes show the raw data, including a 'Server Hello' message.

No.	Time	Source	Destination	Protocol	Length	Info
27	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	22908 → 443 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
35	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	49411 → 443 [ACK] Seq=1 Ack=1 Win=253 Len=0
36	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	80	443 → 49411 [ACK] Seq=1 Ack=2 Win=889 Len=0 SRE=2
37	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	76	7000 → 1500 [ACK] Seq=1 Ack=1 Win=308 Len=0
38	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	5228 → 5228 [ACK] Seq=1 Ack=1 Win=253 Len=0
39	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	75	22888 → 5228 [ACK] Seq=1 Ack=1 Win=253 Len=0
40	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	75	22888 → 5228 [ACK] Seq=1 Ack=1 Win=253 Len=0
41	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	75	22888 → 5228 [ACK] Seq=1 Ack=1 Win=253 Len=0
42	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	86	5228 → 22888 [ACK] Seq=1 Ack=2 Win=1845 Len=0 SLE=1 SRE=2
43	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	86	5228 → 22888 [ACK] Seq=1 Ack=2 Win=1845 Len=0 SLE=1 SRE=2
44	0.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	86	5228 → 22888 [ACK] Seq=1 Ack=2 Win=1845 Len=0 SLE=1 SRE=2
400	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	86	22888 → 443 [SYN] Seq=1 Win=5535 Len=0 MSS=1340 WS=256 SACK_PERM
401	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	86	22888 → 443 [SYN] Seq=1 Win=5535 Len=0 MSS=1340 WS=256 SACK_PERM
406	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	86	443 → 22888 [SYN, ACK] Seq=1 Ack=1 Win=5535 Len=0 MSS=1340 SACK_PERM WS=256
407	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	22888 → 443 [ACK] Seq=1 Ack=1 Win=5280 Len=0
408	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1374	22888 → 443 [ACK] Seq=1 Ack=1 Win=5280 Len=1300 [TCP PDU reassembled in 409]
409	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	534	Client Hello [SHA256, youtube.com]
410	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	86	443 → 22888 [SYN, ACK] Seq=1 Ack=1 Win=5280 Len=0
411	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	22888 → 443 [ACK] Seq=1 Ack=1 Win=5280 Len=0
412	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1374	22888 → 443 [ACK] Seq=1 Ack=1 Win=5280 Len=1300 [TCP PDU reassembled in 413]
413	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	566	Client Hello [SHA256, youtube.com]
415	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	443 → 22888 [ACK] Seq=1 Ack=1761 Win=267520 Len=0
416	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	Server Hello
417	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	Change Cipher Spec
418	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	443 → 22888 [ACK] Seq=2441 Ack=1761 Win=267520 Len=1220 [TCP PDU reassembled in 423]
419	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	443 → 22888 [PSH, ACK] Seq=3661 Ack=1761 Win=267520 Len=1220 [TCP PDU reassembled in 423]
420	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	443 → 22888 [ACK] Seq=4881 Ack=1761 Win=267520 Len=1220 [TCP PDU reassembled in 423]
421	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	443 → 22888 [PSH, ACK] Seq=6181 Ack=1761 Win=267520 Len=1220 [TCP PDU reassembled in 423]
422	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	22888 → 443 [ACK] Seq=1793 Ack=7321 Win=5280 Len=0
423	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	627	Application Data
424	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	148	Change Cipher Spec, Application Data
425	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	148	Application Data
426	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1374	22888 → 443 [ACK] Seq=1927 Ack=7874 Win=64768 Len=1300 [TCP PDU reassembled in 428]
427	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1374	22888 → 443 [ACK] Seq=3227 Ack=7874 Win=64768 Len=1300 [TCP PDU reassembled in 428]
428	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	118	Application Data
429	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	443 → 22888 [ACK] Seq=1381 Win=268832 Len=0
430	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	443 → 22888 [ACK] Seq=1381 Win=267776 Len=0
431	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	Server Hello
432	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	Change Cipher Spec
433	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	443 → 22888 [ACK] Seq=2441 Ack=1793 Win=267776 Len=1220 [TCP PDU reassembled in 437]
434	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	443 → 22888 [PSH, ACK] Seq=3661 Ack=1793 Win=267776 Len=1220 [TCP PDU reassembled in 437]
435	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	443 → 22888 [ACK] Seq=4881 Ack=1793 Win=267776 Len=1220 [TCP PDU reassembled in 437]
436	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1294	443 → 22888 [PSH, ACK] Seq=6181 Ack=1793 Win=267776 Len=1220 [TCP PDU reassembled in 437]
437	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	22888 → 443 [ACK] Seq=1793 Ack=7321 Win=5280 Len=0
443	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	148	Change Cipher Spec, Application Data
444	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	148	Application Data
445	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	443 → 22888 [ACK] Seq=7874 Ack=6563 Win=264960 Len=0
446	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1840	Application Data, Application Data
447	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	1840	Application Data
448	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	22888 → 443 [ACK] Seq=4561 Ack=8871 Win=5280 Len=0
449	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	185	Application Data
450	16.000000	2049:40c1:15:28:c8:11	2049:40c1:15:28:c8:11	TCP	74	22888 → 443 [ACK] Seq=1793 Ack=7321 Win=5280 Len=0

## Files Submitted

- `my_capture.pcapng` — Captured packet file
  - `README.md` — Protocol summary (for GitHub)
  - Screenshots:
    - `icmp.png`
    - `dns.png`
    - `tcp.png`
- 

## Conclusion & Learning Outcomes

Through this task, I gained:

- Practical experience with **live packet capturing**.
  - Ability to **filter and isolate protocols**.
  - Understanding of **how protocols behave in real time**.
  - Insight into network diagnostics, layered protocol behavior, and encrypted vs. unencrypted traffic.
-