

Task 5: Capture and Analyze Network Traffic Using Wireshark

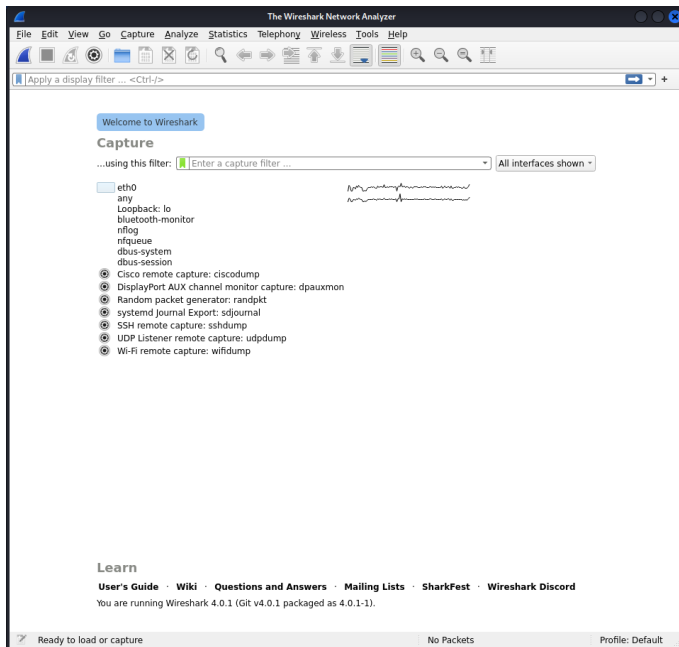
Task Objective : To capture live network packets on a Linux system using Wireshark and analyze the traffic to identify common protocols like HTTP, DNS, TCP, and ICMP.

1. Install Wireshark.

```
$sudo apt update
sudo apt install wireshark -y
[sudo] password for kishan:
Get:1 https://dl.google.com/linux/chrome/deb stable InRelease [1,825 B]
Hit:2 https://brave-browser-apt-release.s3.brave.com stable InRelease
Get:3 https://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1,211 B]
Get:4 https://deb.parrot.sh/parrot lory InRelease [29.8 kB]
Get:5 https://deb.parrot.sh/direct/parrot lory-security InRelease [29.5 kB]
Get:6 https://deb.parrot.sh/parrot lory-backports InRelease [29.7 kB]
Get:7 https://deb.parrot.sh/parrot lory/main amd64 Packages [19.2 MB]
Get:8 https://deb.parrot.sh/direct/parrot lory-security/main amd64 Packages [539 kB]
Get:9 https://deb.parrot.sh/parrot lory-backports/main amd64 Packages [722 kB]
Fetched 20.6 MB in 14s (1,472 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
26 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
wireshark is already the newest version (4.0.17-0+deb12u1).
wireshark set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 26 not upgraded.
```

2. Start capturing on your active network interface.

The GUI of wireshark will be like this. Now I will select eth0. This is a wired (Wi-Fi) interface connected on PCI bus 1, slot 0.



3. Browse a website or ping a server to generate traffic.

After starting the capture, a new terminal window was opened and the following command was run to ping Google, which generates ICMP packets.

ping google.com -c 5

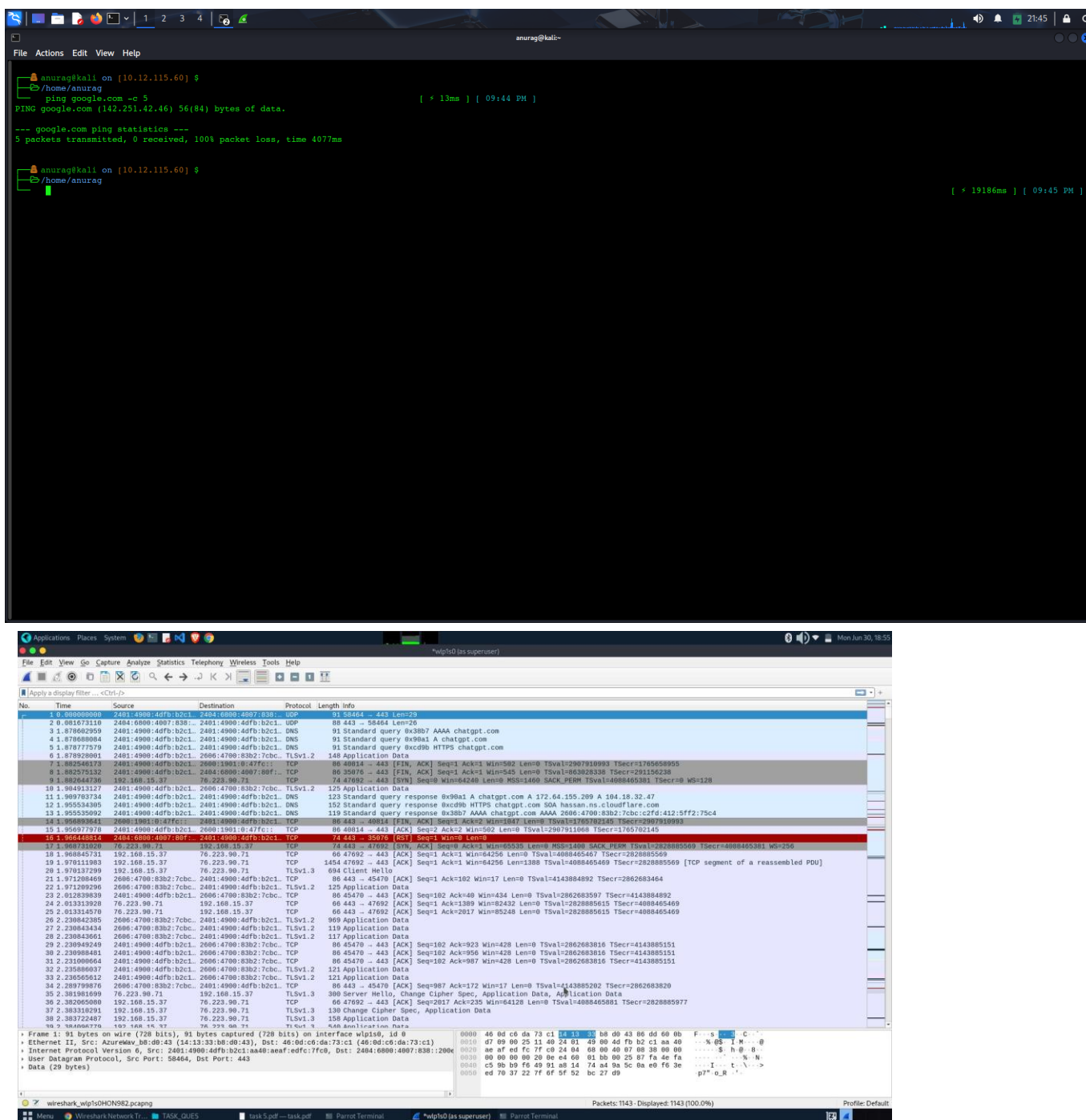
Then, to generate HTTP, HTTPS, and DNS traffic, a website was accessed using

curl https://example.com

This simulates real-world browsing activity and helps capture various protocol packets like DNS (for name resolution), TCP (for reliable transport), and HTTP/HTTPS (for web content).

This is the interface where i have pressed the commands

This is the traffic captured

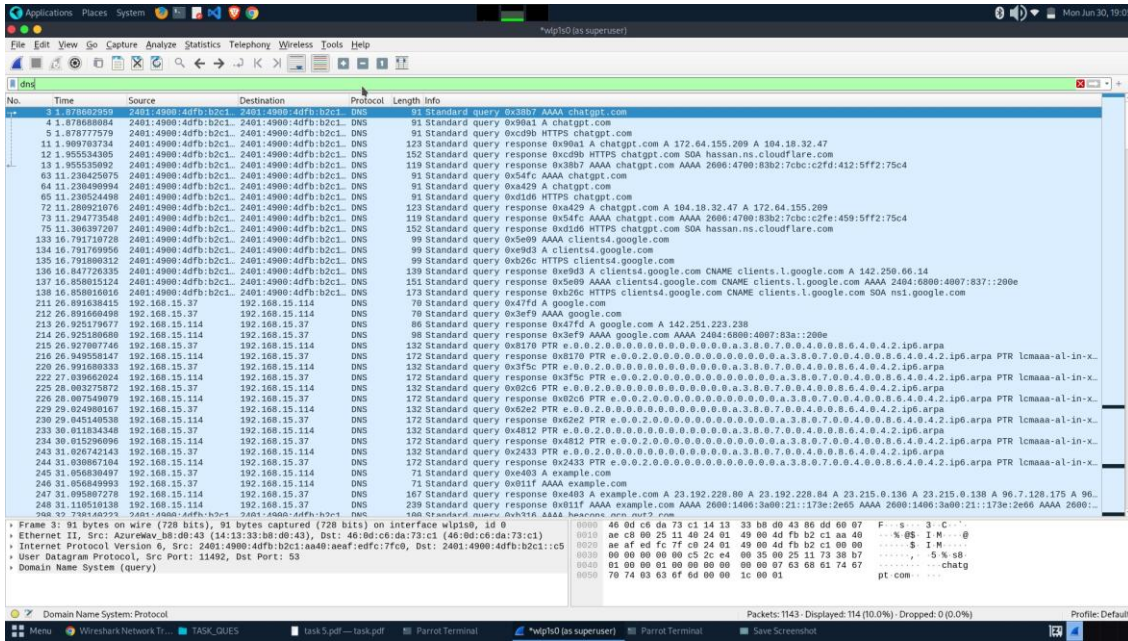


4. Stop capture after a minute.

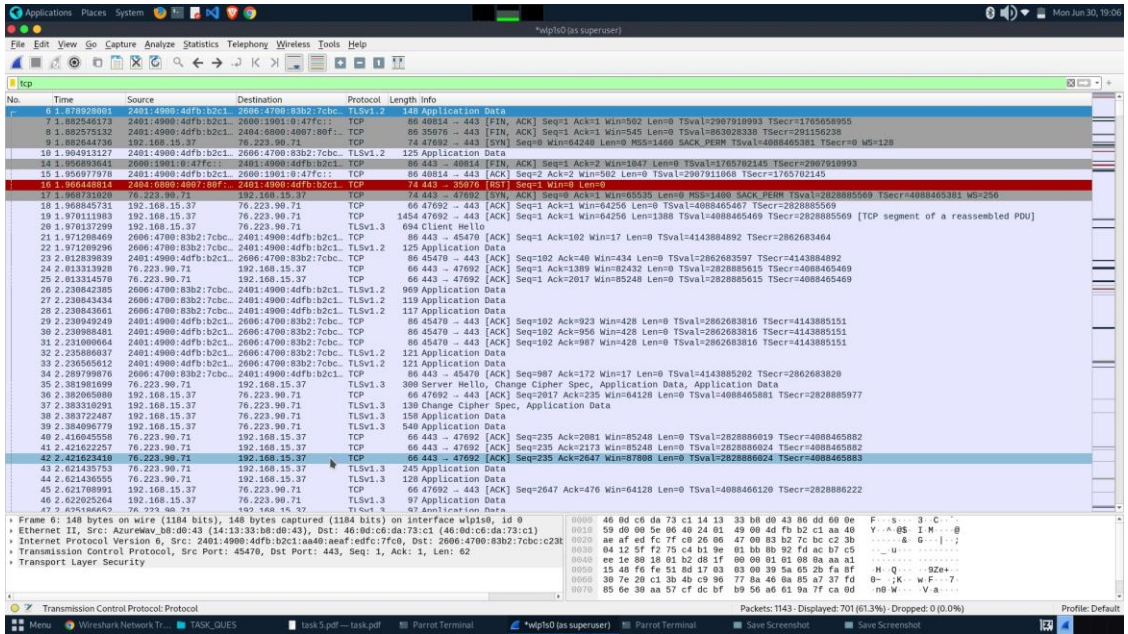
I have stopped it after a min

5. Filter captured packets by protocol (e.g., HTTP, DNS, TCP).

Used the Display Filter bar at the top of Wireshark to isolate specific types of packets.



dns → filters DNS queries and responses



tcp → shows only TCP traffic

No.	Time	Source	Destination	Protocol	Length	Info
57.6	0.02080457	fe80::f86f:a5b7:4fe4:4f04	2401:4900:4dfb:b2c1::c5	ICMPv6	80	Neighbor Solicitation for 2401:4900:4dfb:b2c1::c5 from 14:13:33:b8:d0:43
58	6.932183776	2401:4900:4dfb:b2c1::	fe80::f86f:a5b7:4fe4:4f04	ICMPv6	78	Neighbor Advertisement 2401:4900:4dfb:b2c1::c5 (rtr, sol)
61.7	0.08072353	fe80::440d:c0ff:fed.	2401:4900:4dfb:b2c1::	ICMPv6	80	Neighbor Solicitation for 2401:4900:4dfb:b2c1::aa40:aaef:edfc:7fc0 from 46:0d:c6:da:73:c1
62.7	0.08130662	2401:4900:4dfb:b2c1::	fe80::440d:c0ff:fed.	ICMPv6	78	Neighbor Advertisement 2401:4900:4dfb:b2c1::aa40:aaef:edfc:7fc0 (sol)
67.11	0.02727336	fe80::440d:c0ff:fed.	2401:4900:4dfb:b2c1::	ICMPv6	80	Neighbor Solicitation for fe80::f86f:a5b7:4fe4:4f04 from 46:0d:c6:da:73:c1
68.11	0.02805841	fe80::f86f:a5b7:4fe4:4f04	2401:4900:4dfb:b2c1::	ICMPv6	78	Neighbor Advertisement fe80::f86f:a5b7:4fe4:4f04 (sol)
217.26	0.95121808	2401:4900:4dfb:b2c1::	2404:6800:4007:83a::	ICMPv6	118	Echo (ping) request id=0xfce, seq=1, hop limit=64 (reply in 219)
219.26	0.951199318	2404:6800:4007:83a::	2401:4900:4dfb:b2c1::	ICMPv6	118	Echo (ping) reply id=0xfce, seq=1, hop limit=117 (request in 217)
223.27	0.952195597	2401:4900:4dfb:b2c1::	2404:6800:4007:83a::	ICMPv6	118	Echo (ping) request id=0xfce, seq=2, hop limit=64 (reply in 224)
224.28	0.95234431	2404:6800:4007:83a::	2401:4900:4dfb:b2c1::	ICMPv6	118	Echo (ping) reply id=0xfce, seq=2, hop limit=117 (request in 223)
227.28	0.954143543	2401:4900:4dfb:b2c1::	2404:6800:4007:83a::	ICMPv6	118	Echo (ping) request id=0xfce, seq=3, hop limit=64 (reply in 228)
228.29	0.954812162	2404:6800:4007:83a::	2401:4900:4dfb:b2c1::	ICMPv6	118	Echo (ping) reply id=0xfce, seq=3, hop limit=117 (request in 227)
231.29	0.954213914	2401:4900:4dfb:b2c1::	2404:6800:4007:83a::	ICMPv6	118	Echo (ping) request id=0xfce, seq=4, hop limit=64 (reply in 232)
232.30	0.11333512	2404:6800:4007:83a::	2401:4900:4dfb:b2c1::	ICMPv6	118	Echo (ping) reply id=0xfce, seq=4, hop limit=117 (request in 231)
238.30	0.370679029	fe80::440d:c0ff:fed.	2401:4900:4dfb:b2c1::	ICMPv6	80	Neighbor Solicitation for 2401:4900:4dfb:b2c1::aa40:aaef:edfc:7fc0 from 46:0d:c6:da:73:c1
239.30	0.370143005	2401:4900:4dfb:b2c1::	fe80::440d:c0ff:fed.	ICMPv6	78	Neighbor Advertisement 2401:4900:4dfb:b2c1::aa40:aaef:edfc:7fc0 (sol)
241.30	0.956792839	2401:4900:4dfb:b2c1::	2404:6800:4007:83a::	ICMPv6	118	Echo (ping) request id=0xfce, seq=5, hop limit=64 (reply in 242)
242.31	0.926076165	2404:6800:4007:83a::	2401:4900:4dfb:b2c1::	ICMPv6	118	Echo (ping) reply id=0xfce, seq=5, hop limit=117 (request in 241)
576.53	0.362769914	fe80::f86f:a5b7:4fe4:4f04	2401:4900:4dfb:b2c1::	ICMPv6	80	Neighbor Solicitation for 2401:4900:4dfb:b2c1::c5 from 14:13:33:b8:d0:43
577.53	0.84428088	2401:4900:4dfb:b2c1::	fe80::f86f:a5b7:4fe4:4f04	ICMPv6	78	Neighbor Advertisement 2401:4900:4dfb:b2c1::c5 (rtr, sol)
579.53	0.925457549	fe80::440d:c0ff:fed.	2401:4900:4dfb:b2c1::	ICMPv6	80	Neighbor Solicitation for 2401:4900:4dfb:b2c1::aa40:aaef:edfc:7fc0 from 46:0d:c6:da:73:c1
580.53	0.925515966	2401:4900:4dfb:b2c1::	fe80::440d:c0ff:fed.	ICMPv6	78	Neighbor Advertisement 2401:4900:4dfb:b2c1::aa40:aaef:edfc:7fc0 (sol)
603.59	0.43431194	fe80::440d:c0ff:fed.	2401:4900:4dfb:b2c1::	ICMPv6	80	Neighbor Solicitation for fe80::f86f:a5b7:4fe4:4f04 from 46:0d:c6:da:73:c1
604.59	0.43501084	fe80::f86f:a5b7:4fe4:4f04	2401:4900:4dfb:b2c1::	ICMPv6	78	Neighbor Advertisement fe80::f86f:a5b7:4fe4:4f04 (sol)
606.59	0.60693594	fe80::440d:c0ff:fed.	ff02::1	ICMPv6	142	Router Advertisement from 46:0d:c6:da:73:c1
607.59	0.676302742	fe80::f86f:a5b7:4fe4:4f04	ff02::1	ICMPv6	110	Multicast Listener Report Message v2
608.59	0.209546114	fe80::f86f:a5b7:4fe4:4f04	ff02::1	ICMPv6	110	Multicast Listener Report Message v2
1022.61	0.930221158	2401:4900:4dfb:b2c1::	2401:4900:4dfb:b2c1::	ICMPv6	110	Destination Unreachable (Port unreachable)
1022.61	0.931985171	2401:4900:4dfb:b2c1::	2401:4900:4dfb:b2c1::	ICMPv6	210	Destination Unreachable (Port unreachable)
1084.65	5.53758214	2401:4900:4dfb:b2c1::	2401:4900:4dfb:b2c1::	ICMPv6	220	Destination Unreachable (Port unreachable)

icmp → shows ping request/reply

No Http traffic was captured

6. Identify at least 3 different protocols in the capture.

I have ICMP , TCP and DNS packets that were captured on my wire shark i will say what are those and what will they do shortly

1. ICMP (Internet Control Message Protocol)

- **Purpose:** Used for network diagnostics.
- **Example:** ping google.com sends ICMP Echo Requests and receives Echo Replies.
- **Use Case:** Helps check if a host is reachable and measures round-trip time.

2. TCP (Transmission Control Protocol)

- **Purpose:** Ensures reliable and ordered data delivery between devices.
- **Example:** Used in protocols like HTTP, HTTPS, FTP.
- **Use Case:** Establishes a connection (3-way handshake), ensures all data reaches correctly.

3. DNS (Domain Name System)

- **Purpose:** Resolves human-readable domain names into IP addresses.
- **Example:** When accessing example.com, your system sends a DNS query to find its IP.
- **Use Case:** First step in web browsing — without DNS, the browser can't find the server.

7. Export the capture as a .pcap file.

I have exported the capture

8. Summarize your findings and packet details.

During the network capture, three main types of packets were identified: ICMP, TCP, and DNS. Each of these serves a distinct role in network communication:

ICMP (Internet Control Message Protocol)

- Used for diagnostic and error-reporting functions.
- Commonly seen in tools like ping.
- Helps determine if a host is reachable and measures latency.

TCP (Transmission Control Protocol)

- A connection-oriented protocol that ensures reliable data transfer.
- Used in web browsing (HTTP/HTTPS), file transfers (FTP), emails, etc.
- Establishes a connection using a 3-way handshake before transmitting data.

DNS (Domain Name System)

- Resolves domain names (like google.com) to their respective IP addresses.
- Works before any website or online service can be accessed.

