Task 5 : Capture and Analyze Network Traffic Using Wireshark.

Step 1 :-

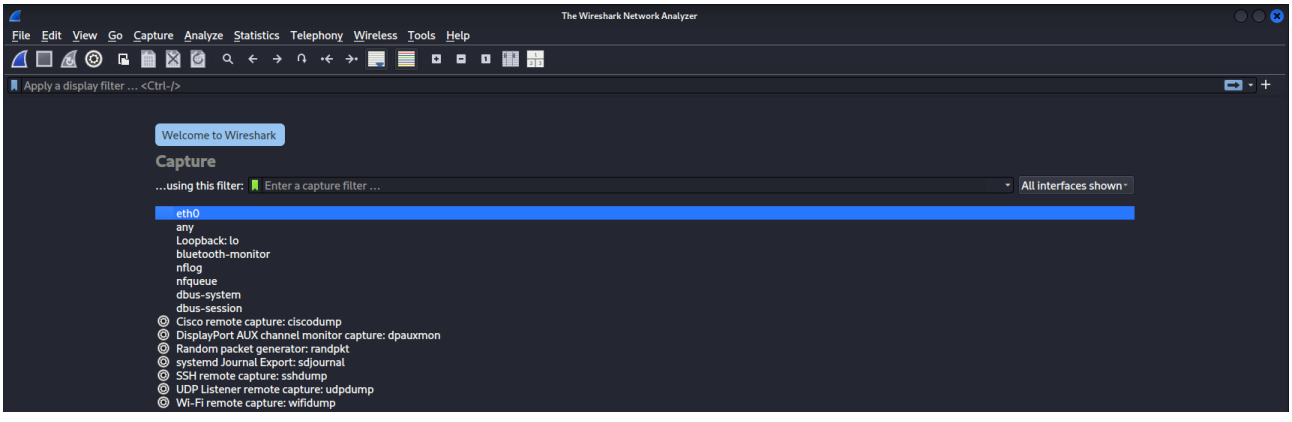
Install Wireshark on Linux Machine.

```
┌──(root㉿kali)-[/home/kali]
└─# apt install wireshark -y
Upgrading:
  libwireshark-data  libwireshark18  libwiretap15  libwsutil16  tshark  wireshark  wireshark-common

Summary:
  Upgrading: 7, Installing: 0, Removing: 0, Not Upgrading: 866
  Download size: 27.4 MB
  Space needed: 104 kB / 54.3 GB available

Get:1 http://kali.download/kali kali-rolling/main amd64 libwsutil16 amd64 4.4.9-1 [127 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 libwiretap15 amd64 4.4.9-1 [267 kB]
Get:4 http://kali.download/kali kali-rolling/main amd64 tshark amd64 4.4.9-1 [175 kB]
Get:5 http://kali.download/kali kali-rolling/main amd64 wireshark amd64 4.4.9-1 [4,626 kB]
Get:6 http://kali.download/kali kali-rolling/main amd64 wireshark-common amd64 4.4.9-1 [598 kB]
Get:3 http://mirrors.esto.network/kali kali-rolling/main amd64 libwireshark18 amd64 4.4.9-1 [21.2 MB]
Get:7 http://mirrors.esto.network/kali kali-rolling/main amd64 libwireshark-data all 4.4.9-1 [449 kB]
Fetched 27.4 MB in 19s (1,419 kB/s)
Preconfiguring packages ...
(Reading database ... 417523 files and directories currently installed.)
Preparing to unpack .../0-libwsutil16_4.4.9-1_amd64.deb ...
Unpacking libwsutil16:amd64 (4.4.9-1) over (4.4.7-1+b1) ...
Preparing to unpack .../1-libwiretap15_4.4.9-1_amd64.deb ...
Unpacking libwiretap15:amd64 (4.4.9-1) over (4.4.7-1+b1) ...
Preparing to unpack .../2-libwireshark18_4.4.9-1_amd64.deb ...
Unpacking libwireshark18:amd64 (4.4.9-1) over (4.4.7-1+b1) ...
Preparing to unpack .../3-tshark_4.4.9-1_amd64.deb ...
Unpacking tshark (4.4.9-1) over (4.4.7-1+b1) ...
Preparing to unpack .../4-wireshark_4.4.9-1_amd64.deb ...
Unpacking wireshark (4.4.9-1) over (4.4.7-1+b1) ...
Preparing to unpack .../5-wireshark-common_4.4.9-1_amd64.deb ...
Unpacking wireshark-common (4.4.9-1) over (4.4.7-1+b1) ...
Preparing to unpack .../6-libwireshark-data_4.4.9-1_all.deb ...
Unpacking libwireshark-data (4.4.9-1) over (4.4.7-1) ...
Setting up libwireshark-data (4.4.9-1) ...
Setting up libwsutil16:amd64 (4.4.9-1) ...
Setting up libwiretap15:amd64 (4.4.9-1) ...
Setting up libwireshark18:amd64 (4.4.9-1) ...
Setting up wireshark-common (4.4.9-1) ...
Setting up wireshark (4.4.9-1) ...
Setting up tshark (4.4.9-1) ...
Processing triggers for mailcap (3.74) ...
Processing triggers for kali-menu (2025.3.2) ...
Processing triggers for desktop-file-utils (0.28-1) ...
Processing triggers for hicolor-icon-theme (0.18-2) ...
Processing triggers for libc-bin (2.41-12) ...
Processing triggers for man-db (2.13.1-1) ...
Processing triggers for shared-mime-info (2.4-5+b3) ...
```

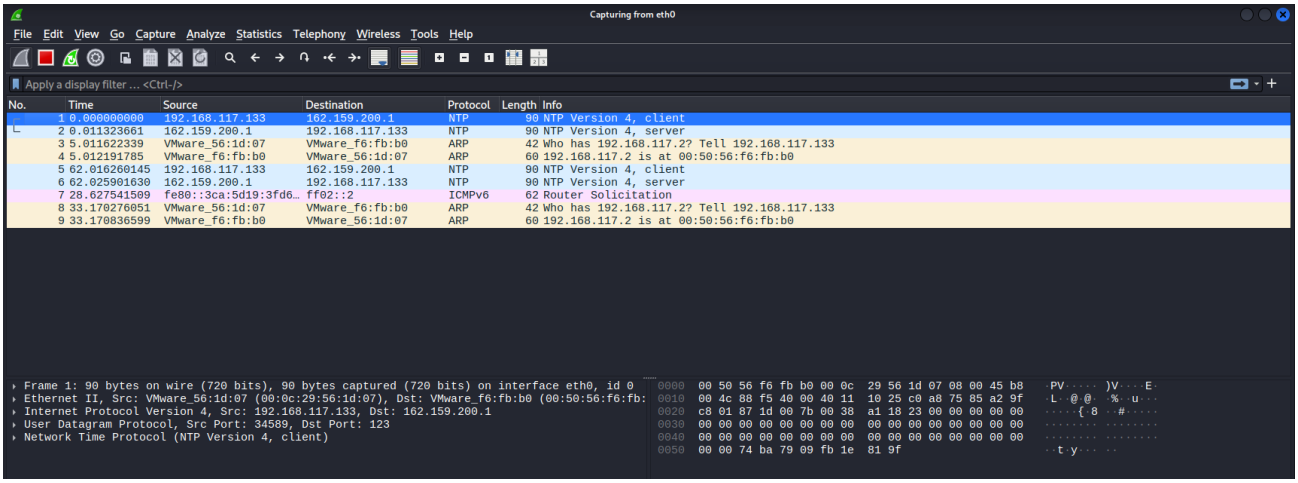
Step 2 :-

Start capturing the data packet's on Active Network.

Open the Wireshark –



Capturing the data packets.

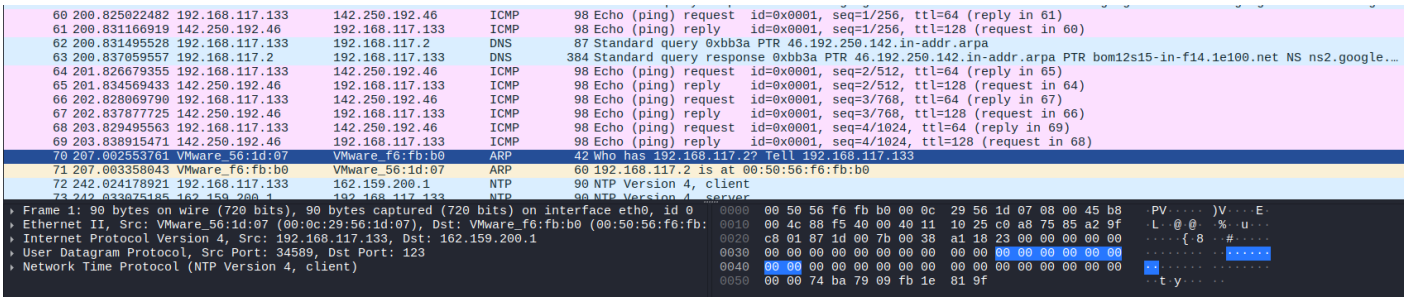


Step 3 :-

Generate the Network Traffics.

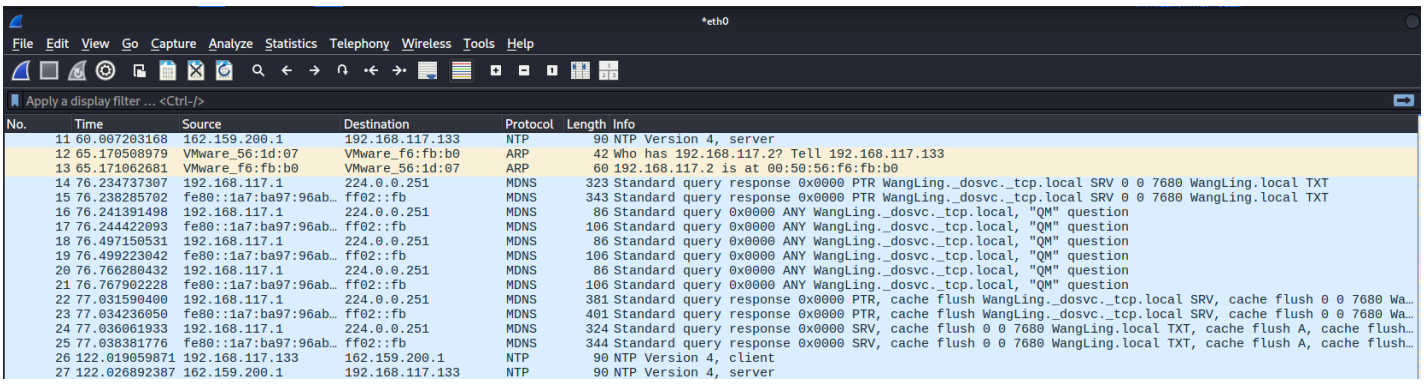
```
┌──(root㉿kali)-[/home/kali]
└─# ping -c 4 google.com

PING google.com (142.250.192.46) 56(84) bytes of data.
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=1 ttl=128 time=6.22 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=2 ttl=128 time=7.93 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=3 ttl=128 time=9.85 ms
64 bytes from bom12s15-in-f14.1e100.net (142.250.192.46): icmp_seq=4 ttl=128 time=9.46 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 6.217/8.364/9.847/1.430 ms
```

ICMP, DNS, ARP, NTP, etc packets are captured.



Step 4 :-

Let’s stop the scan.

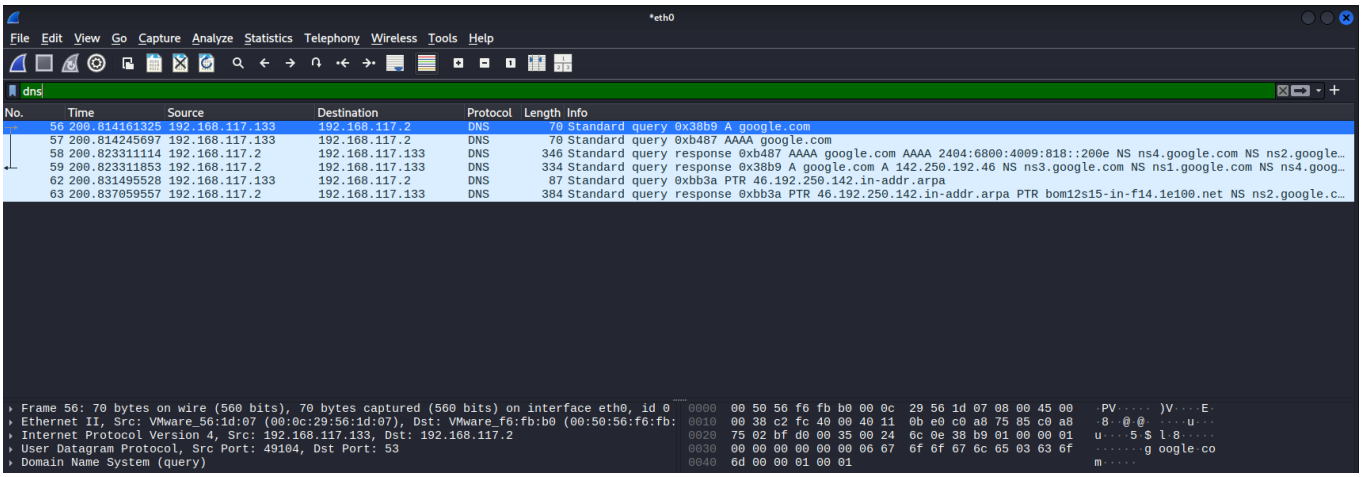


Step 5 :-

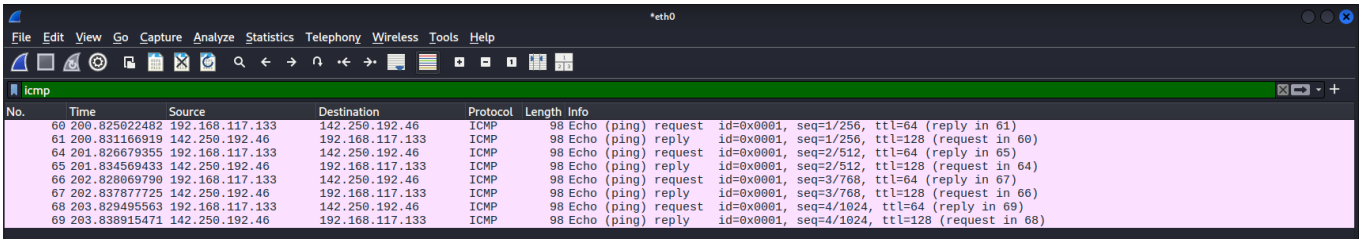
Filter the Captured Packets.

As, http, dns, tcp, icmp, etc.

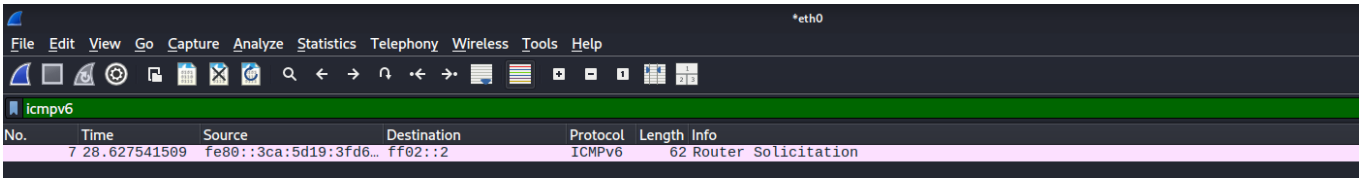
Filtered for dns :-



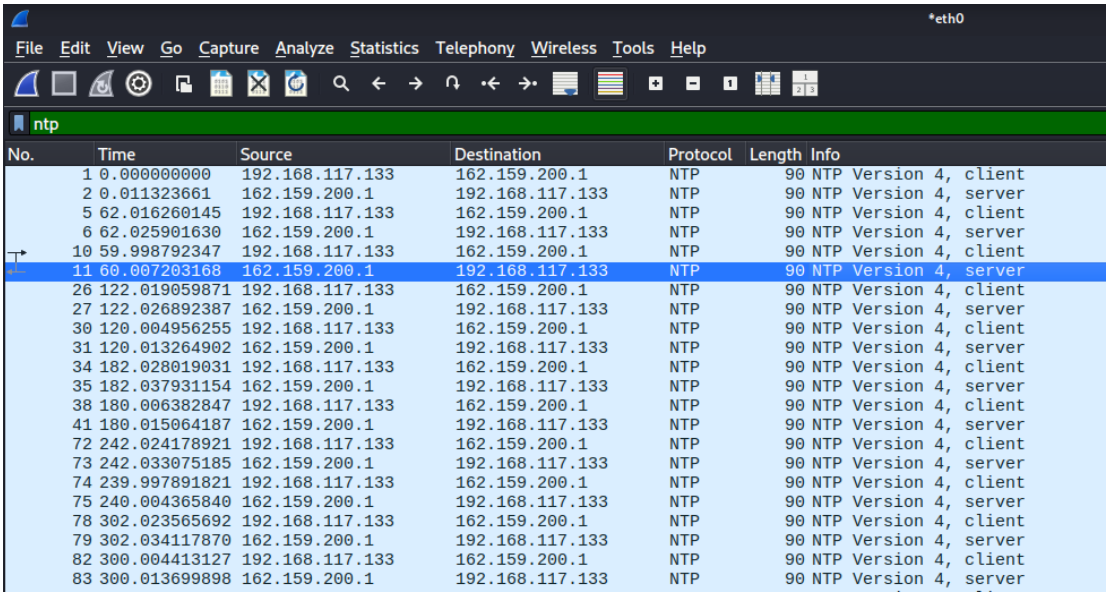
Filtered for icmp :-



Filtered for icmp v6 :-



Filtered for ntp :-



Step 6 :-

Identify atleast 3 different protocols.

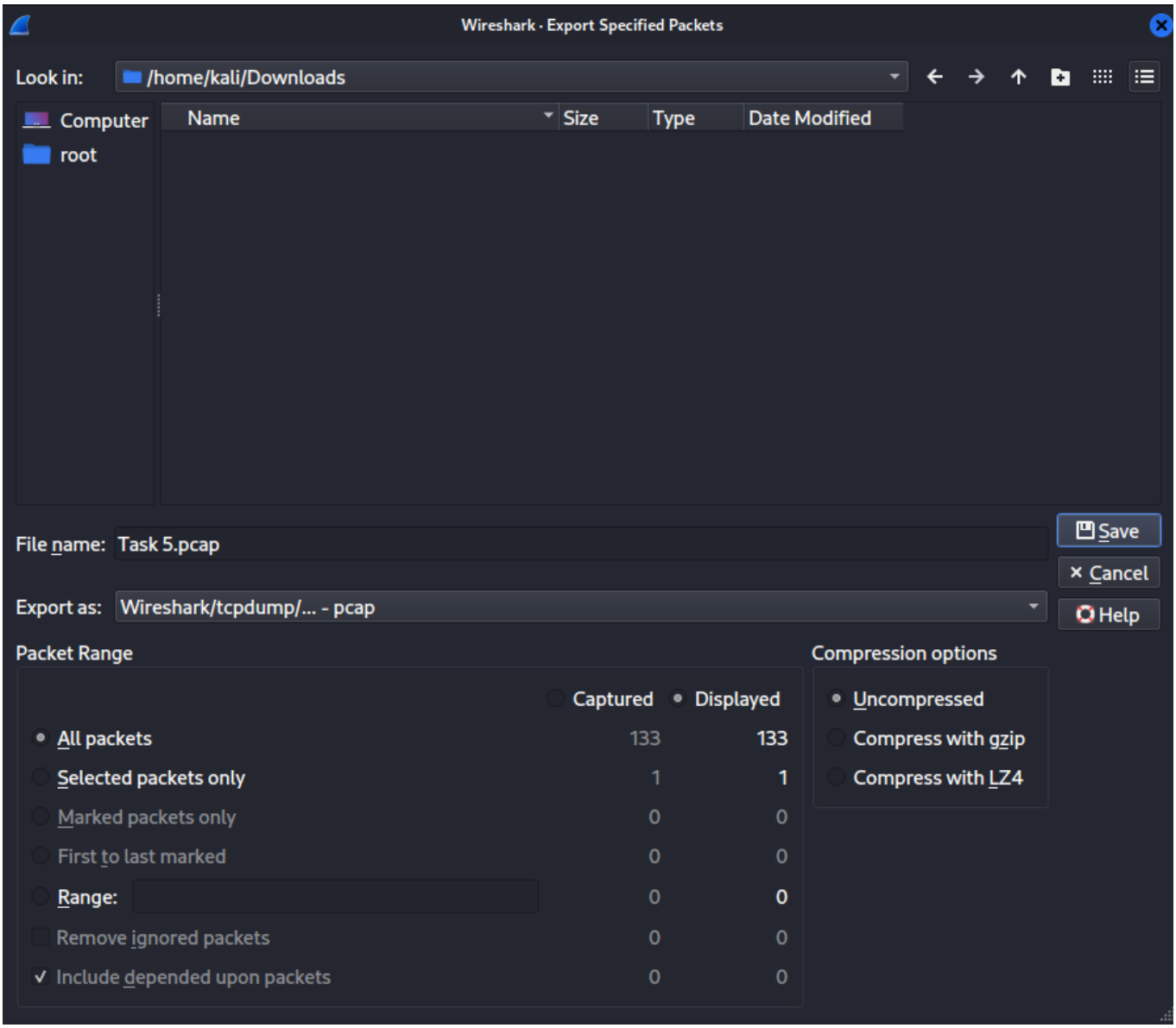
Protocol	Full Name	Layer (OSI Model)	Purpose / Function	Example Packet Details (Typical)	Notes for Wireshark
NTP	Network Time Protocol	Application Layer	Synchronizes the system clock with time servers (e.g., time.google.com)	**Src:** 192.168.1.5**Dst:** 216.239.35.0**Protocol:** UDP (port 123)	Filter: ntp or udp.port == 123
DNS	Domain Name System	Application Layer	Translates domain names (e.g., google.com) into IP addresses	**Src:** 192.168.1.5**Dst:** 8.8.8.8**Protocol:** UDP (port 53)	Filter: dns or udp.port == 53
ICMP	Internet Control Message Protocol (IPv4)	Network Layer	Used for diagnostics (e.g., ping) — echo request and reply messages	**Src:** 192.168.1.5**Dst:** 142.250.64.78 (Google)**Type:** Echo (8) / Echo Reply (0)	Filter: icmp
ICMPv6	Internet Control Message Protocol for IPv6	Network Layer	Same as ICMP but for IPv6; used for neighbor discovery and ping6	**Src:** fe80::1**Dst:** fe80::abcd:1234**Type:** Echo Request (128) / Reply (129)	Filter: icmpv6
ARP	Address Resolution Protocol	Data Link Layer	Resolves IPv4 addresses to MAC addresses on local network	**Src MAC:** 08:00:27:aa:bb:cc**Dst MAC:** ff:ff:ff:ff:ff:ff **Operation:** Who has 192.168.1.1? Tell 192.168.1.5	Filter: arp

Step 7 :-

Export the capture as a .pcap file

File → Export Specified Packets...

And select for the PCAP file format.



Step 8 :-

Summarize the findings and packet details.

I captured network traffic for 60 seconds on interface wlan0 while browsing websites and pinging Google.
I identified multiple protocols including DNS, TCP, HTTP, ICMP, NTP, etc.
The DNS requests showed hostname lookups for domains visited, HTTP packets revealed GET requests for web pages, and ICMP packets were generated by the ping command.