🛡️ *Cybersecurity Internship – Task 5 Report*

Task Title: Capture and Analyze Network Traffic Using Wireshark
Intern Name: Golla venkatesham

🎯 *Objective*

To capture live network traffic using Wireshark, identify key protocols, and analyze packet behavior for cybersecurity awareness and documentation.

🧰 *Tools Used*

- *Wireshark v4.4.9*
- *Windows Command Prompt (ping google.com)*
- *Web Browser (for generating HTTP/DNS traffic)*

📝 *Steps Followed*

1. *Installed and launched Wireshark.*
2. *Identified active network interface using ipconfig.*

3. *Started live capture on the correct interface.*
4. *Generated traffic by:*
    - *Running ping google.com (ICMP)*
    - *Browsing websites (HTTP/DNS)*
5. *Stopped capture after 2 minutes.*
6. *Applied filters: icmp, dns, http, tcp*
7. *Saved capture as task5_capture.pcapng*

🔍 *Protocols Observed*

ping google.com

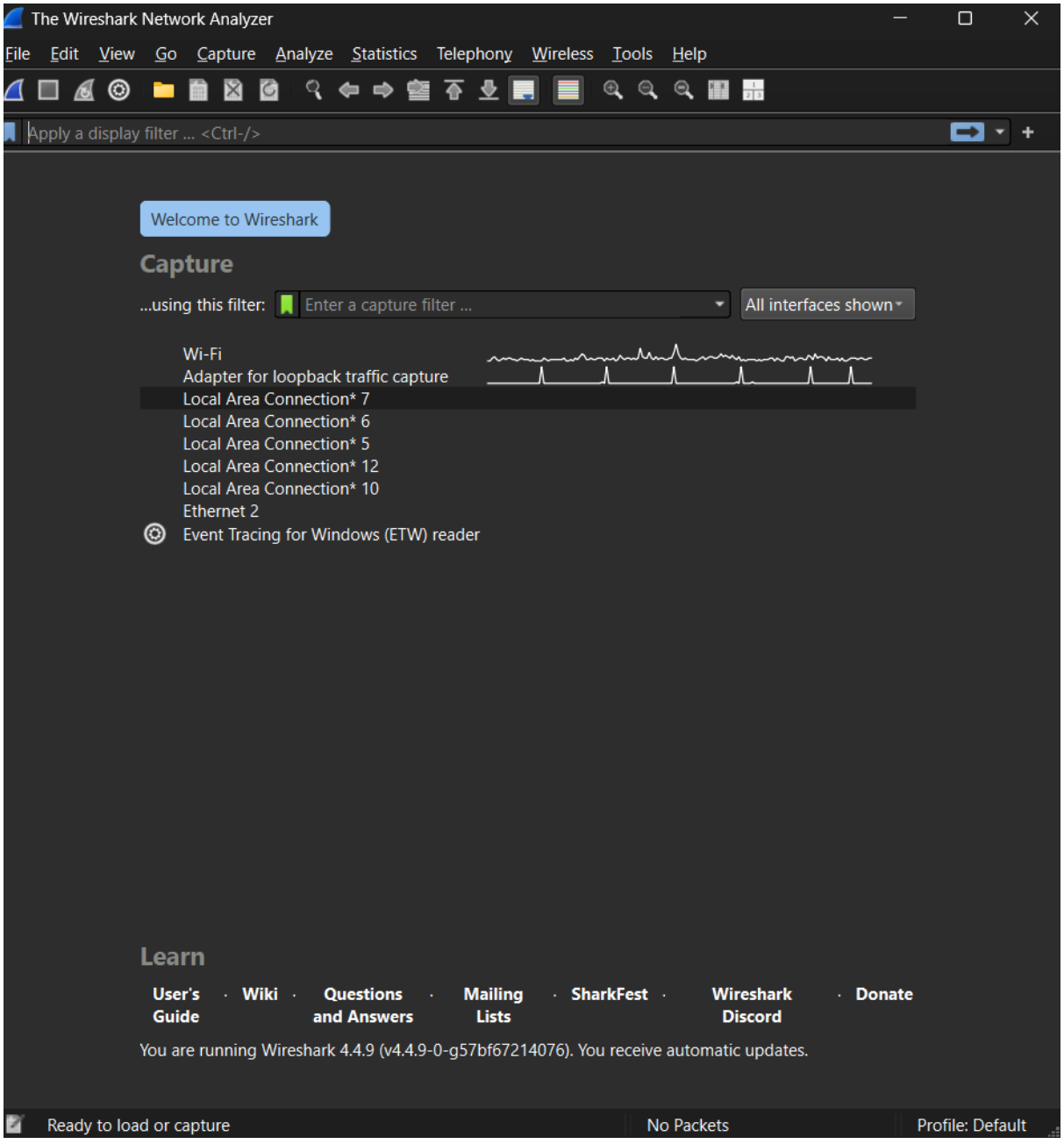
google.com

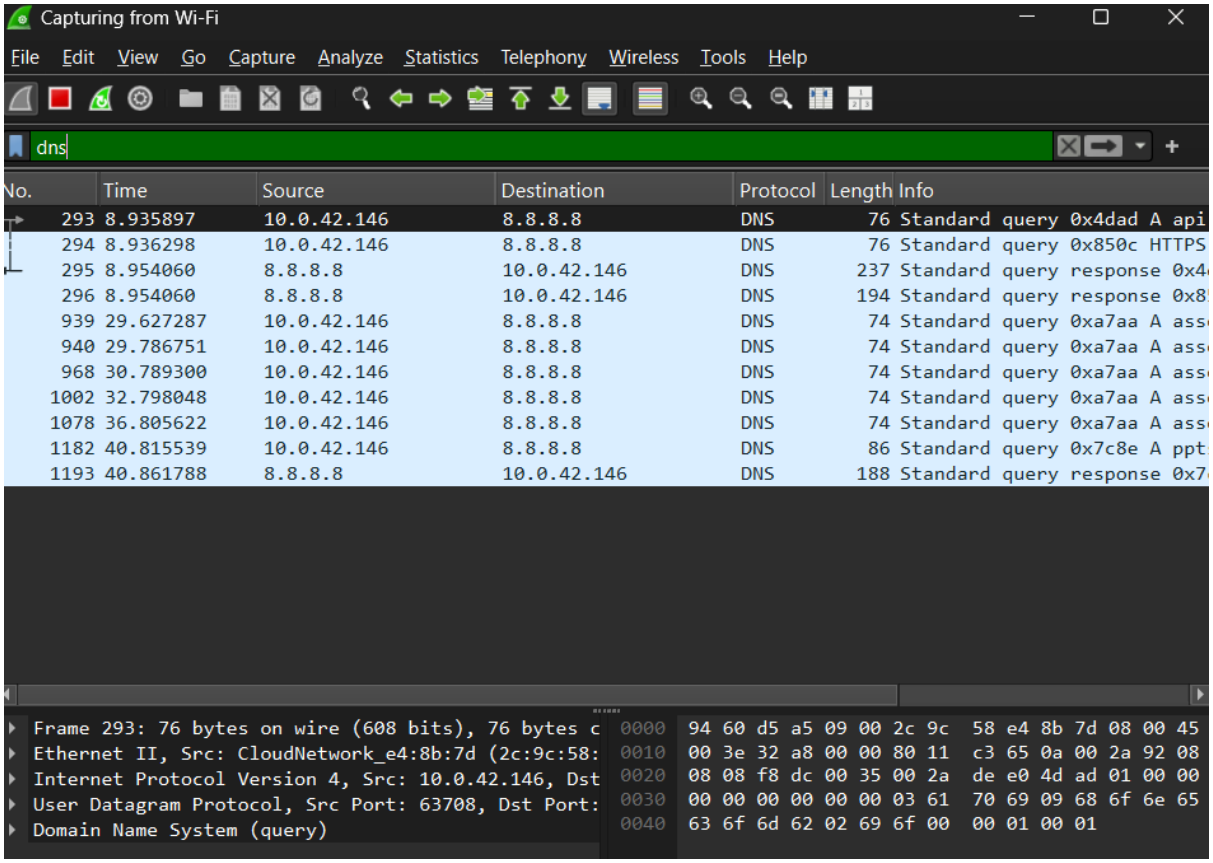
example.com

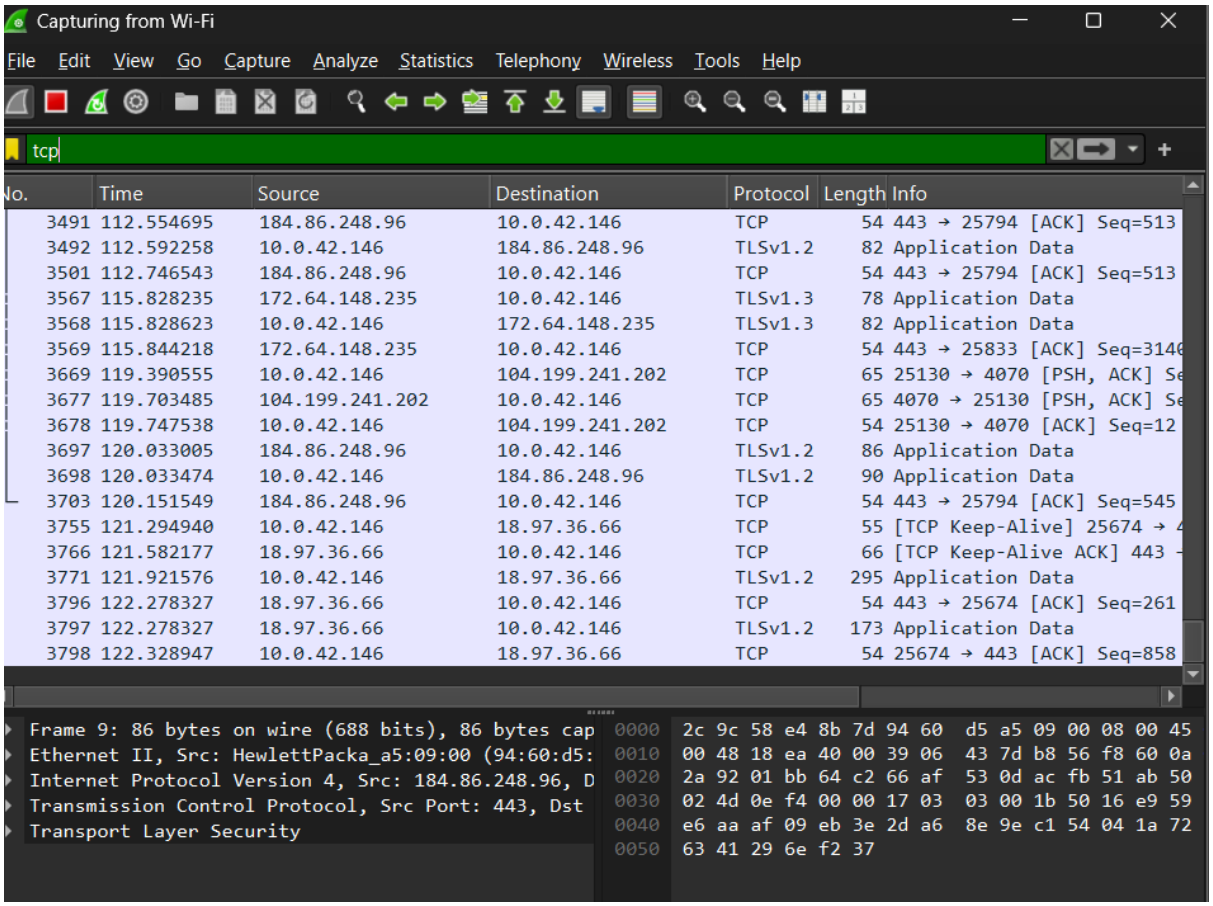
📸 *Screenshots*

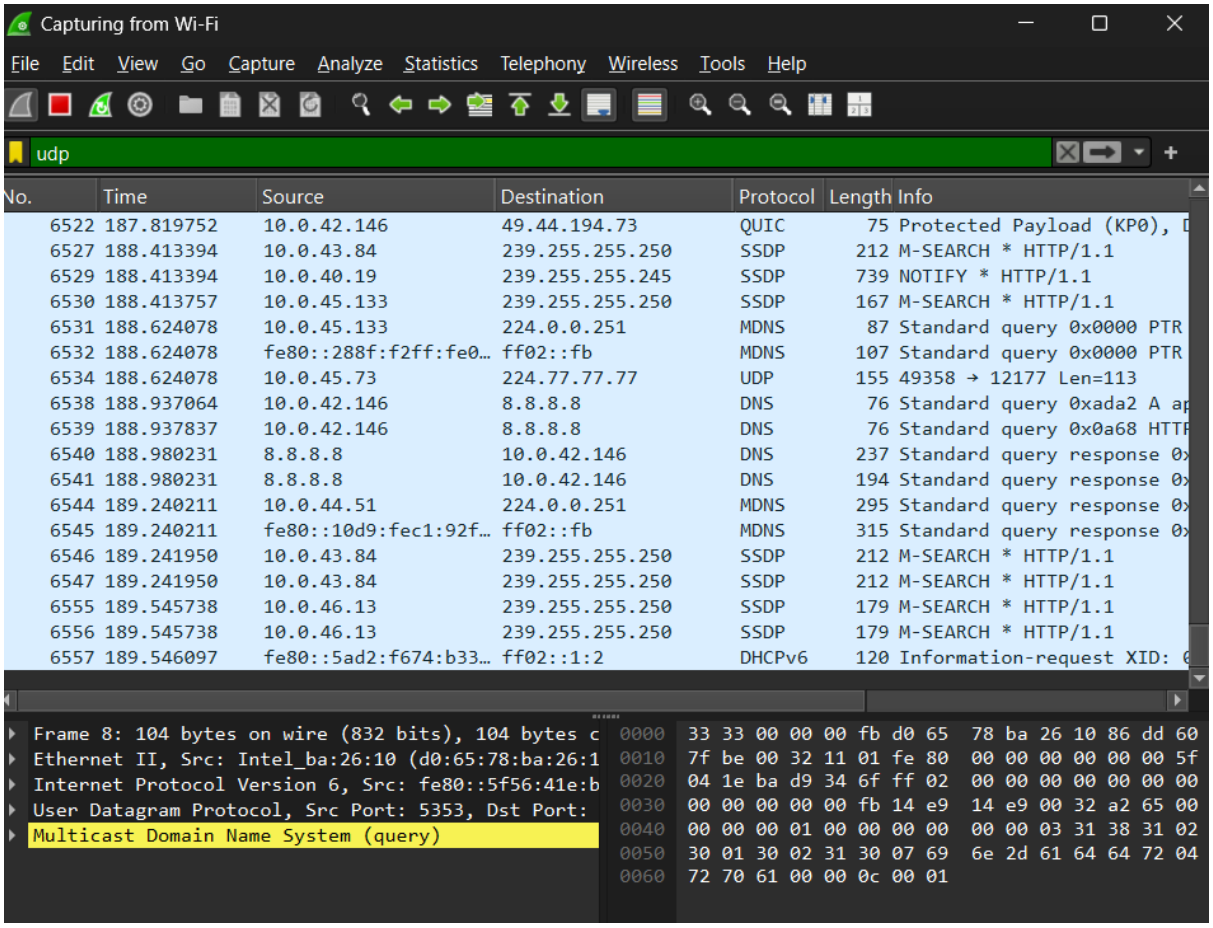
(Insert screenshots here showing filtered packets and expanded packet details. You can

label them like “Figure 1: ICMP Packet”, “Figure 2: DNS Query”, etc.)




Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
293 8.935897 10.0.42.146 8.8.8.8 DNS 76 Standard query 0x4dad A api
294 8.936298 10.0.42.146 8.8.8.8 DNS 76 Standard query 0x850c HTTPS
295 8.954060 8.8.8.8 10.0.42.146 DNS 237 Standard query response 0x4
296 8.954060 8.8.8.8 10.0.42.146 DNS 194 Standard query response 0x8
939 29.627287 10.0.42.146 8.8.8.8 DNS 74 Standard query 0xa7aa A ass
940 29.786751 10.0.42.146 8.8.8.8 DNS 74 Standard query 0xa7aa A ass
968 30.789300 10.0.42.146 8.8.8.8 DNS 74 Standard query 0xa7aa A ass
1002 32.798048 10.0.42.146 8.8.8.8 DNS 74 Standard query 0xa7aa A ass
1078 36.805622 10.0.42.146 8.8.8.8 DNS 74 Standard query 0xa7aa A ass
1182 40.815539 10.0.42.146 8.8.8.8 DNS 86 Standard query 0x7c8e A ppt
1193 40.861788 8.8.8.8 10.0.42.146 DNS 188 Standard query response 0x7
Frame 293: 76 bytes on wire (608 bits), 76 bytes c
Ethernet II, Src: CloudNetwork_e4:8b:7d (2c:9c:58:
Internet Protocol Version 4, Src: 10.0.42.146, Dst
User Datagram Protocol, Src Port: 63708, Dst Port:
Domain Name System (query)
0000 94 60 d5 a5 09 00 2c 9c 58 e4 8b 7d 08 00 45
0010 00 3e 32 a8 00 00 80 11 c3 65 0a 00 2a 92 08
0020 08 08 f8 dc 00 35 00 2a de e0 4d ad 01 00 00
0030 00 00 00 00 00 00 03 61 70 69 09 68 6f 6e 65
0040 63 6f 6d 62 02 69 6f 00 00 01 00 01


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp
No. Time Source Destination Protocol Length Info
3491 112.554695 184.86.248.96 10.0.42.146 TCP 54 443 → 25794 [ACK] Seq=513
3492 112.592258 10.0.42.146 184.86.248.96 TLSv1.2 82 Application Data
3501 112.746543 184.86.248.96 10.0.42.146 TCP 54 443 → 25794 [ACK] Seq=513
3567 115.828235 172.64.148.235 10.0.42.146 TLSv1.3 78 Application Data
3568 115.828623 10.0.42.146 172.64.148.235 TLSv1.3 82 Application Data
3569 115.844218 172.64.148.235 10.0.42.146 TCP 54 443 → 25833 [ACK] Seq=3140
3669 119.390555 10.0.42.146 104.199.241.202 TCP 65 25130 → 4070 [PSH, ACK] Se
3677 119.703485 104.199.241.202 10.0.42.146 TCP 65 4070 → 25130 [PSH, ACK] Se
3678 119.747538 10.0.42.146 104.199.241.202 TCP 54 25130 → 4070 [ACK] Seq=12
3697 120.033005 184.86.248.96 10.0.42.146 TLSv1.2 86 Application Data
3698 120.033474 10.0.42.146 184.86.248.96 TLSv1.2 90 Application Data
3703 120.151549 184.86.248.96 10.0.42.146 TCP 54 443 → 25794 [ACK] Seq=545
3755 121.294940 10.0.42.146 18.97.36.66 TCP 55 [TCP Keep-Alive] 25674 → 4
3766 121.582177 18.97.36.66 10.0.42.146 TCP 66 [TCP Keep-Alive ACK] 443 -
3771 121.921576 10.0.42.146 18.97.36.66 TLSv1.2 295 Application Data
3796 122.278327 18.97.36.66 10.0.42.146 TCP 54 443 → 25674 [ACK] Seq=261
3797 122.278327 18.97.36.66 10.0.42.146 TLSv1.2 173 Application Data
3798 122.328947 10.0.42.146 18.97.36.66 TCP 54 25674 → 443 [ACK] Seq=858
Frame 9: 86 bytes on wire (688 bits), 86 bytes cap
Ethernet II, Src: HewlettPacka_a5:09:00 (94:60:d5:
Internet Protocol Version 4, Src: 184.86.248.96, D
Transmission Control Protocol, Src Port: 443, Dst
Transport Layer Security
0000 2c 9c 58 e4 8b 7d 94 60 d5 a5 09 00 08 00 45
0010 00 48 18 ea 40 00 39 06 43 7d b8 56 f8 60 0a
0020 2a 92 01 bb 64 c2 66 af 53 0d ac fb 51 ab 50
0030 02 4d 0e f4 00 00 17 03 03 00 1b 50 16 e9 59
0040 e6 aa af 09 eb 3e 2d a6 8e 9e c1 54 04 1a 72
0050 63 41 29 6e f2 37



🧠 *Key Learnings*

- *ICMP packets matched ping results with TTL and round-trip times.*
- *DNS queries revealed how domain names are resolved.*
- *TCP packets showed handshake and data flow.*
- *HTTP packets displayed GET requests and responses.*

🌟 *Bonus Insight*

*Wireshark is like **Hanuman** in Ramayana—carrying packets safely across the network jungle!* 🐒✨

Understanding protocols is like decoding the language of the internet—each packet tells a story.

☑ *Conclusion*

This task provided hands-on experience in packet analysis and protocol behavior. Wireshark proved essential for visualizing and understanding network traffic, making it a powerful tool for cybersecurity diagnostics.

Once you paste this into Word or Docs and add your screenshots, export it as a PDF and upload it to your GitHub repo. Want help writing the GitHub description or README next? I'm ready!