

---

# Cybersecurity Internship Report — Task 5

**Intern Name:** Sai Aditya

**Organization:** Elevate Labs

**Task Title:** Capture and Analyze Network Traffic Using Wireshark

**Date:** August 11, 2025

---

## Objective

To capture live network packets using Wireshark, identify at least three different protocols, and summarize their key characteristics and relevance in network communication.

## Tools Used

- Operating System: Kali Linux
- Packet Analyzer: Wireshark
- Network Interface: eth0 (wired) / wlan0 (wireless)
- Target IP: 192.168.150.133

## Scenario

A network capture was performed on the active network interface to monitor and analyze real-time traffic. During the capture session, common activities such as website browsing, DNS lookups, and ICMP pings were carried out to generate traffic. The capture lasted approximately 60 seconds and was then filtered to examine specific protocols.

## Disclaimer

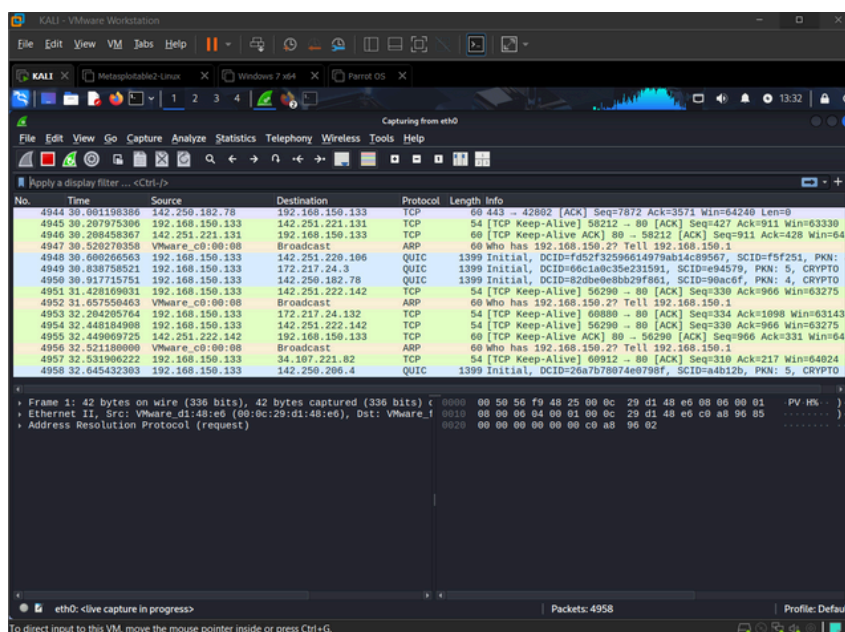
This activity was conducted in a controlled environment solely for educational purposes. No unauthorized traffic interception or monitoring of third-party networks was performed.

---

## Steps Performed

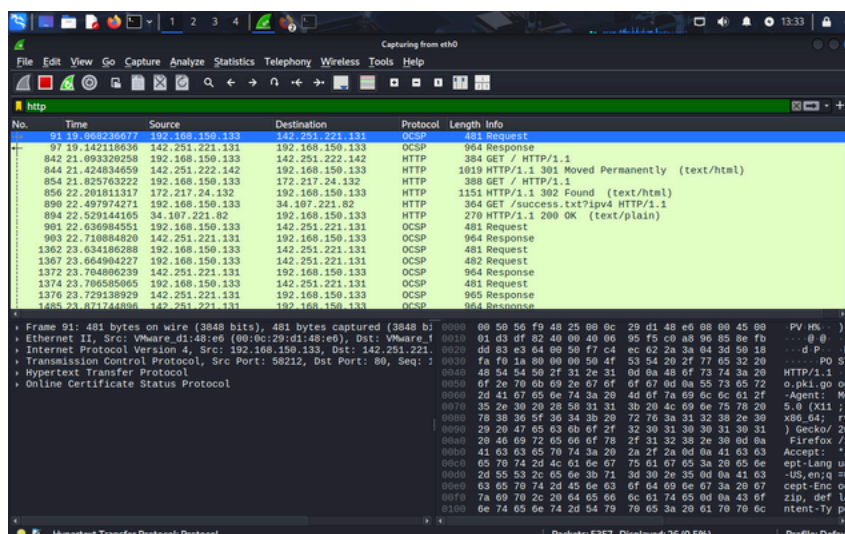
- Installed and launched Wireshark on Kali Linux.
- Selected the active network interface (wlan0).
- Started packet capture.
- Generated network activity by browsing example.com and pinging 8.8.8.8.
- Stopped capture after ~60 seconds.
- Applied protocol filters for HTTP, DNS, and TLS.
- Analyzed packet details and noted source/destination IPs, ports, and protocol functions.
- Saved the capture file as task5\_capture.pcap.

## Screenshots



### Overview of Wireshark

This image shows a Wireshark capture in progress on your Kali Linux system, specifically on the wlan0 interface. It displays a variety of network packets with different protocols, including TCP, QUIC, and ARP. This screenshot demonstrates the initial phase of capturing live network traffic before any filters are applied.



### http filter

This image shows Wireshark with an http filter applied. You can see several packets related to HTTP, including GET and POST requests. This screenshot visually confirms the capture of unencrypted web traffic, showing the transfer of web page data between a client and a server.

No.	Time	Source	Destination	Protocol	Length	Info
46	10.561522386	192.168.150.2	192.168.150.133	DNS	193	Standard query response 0x8add AAAA content-signature-2.cdn
69	18.664812159	192.168.150.133	192.168.150.2	DNS	87	Standard query 0x248a A safebrowsing.googlelepis.com
70	18.665476881	192.168.150.133	192.168.150.2	DNS	87	Standard query 0x5075 AAAA safebrowsing.googlelepis.com
71	18.652468530	192.168.150.2	192.168.150.133	DNS	163	Standard query response 0x248a A safebrowsing.googlelepis.com
72	18.711388443	192.168.150.2	192.168.150.133	DNS	115	Standard query response 0x5075 AAAA safebrowsing.googlelepis.com
84	18.962781936	192.168.150.133	192.168.150.2	DNS	78	Standard query 0x8683 A o.pki.goog
85	18.963295215	192.168.150.133	192.168.150.2	DNS	78	Standard query 0x2499 AAAA o.pki.goog
86	19.033224267	192.168.150.2	192.168.150.133	DNS	121	Standard query response 0x8683 A o.pki.goog CNAME pki-goog
87	19.033214172	192.168.150.2	192.168.150.133	DNS	113	Standard query response 0x2499 AAAA o.pki.goog CNAME pki-goog
93	19.071195344	192.168.150.133	192.168.150.2	DNS	85	Standard query 0x05d A push.services.mozilla.com
94	19.071562195	192.168.150.133	192.168.150.2	DNS	85	Standard query 0x79b1 AAAA push.services.mozilla.com
95	19.134217267	192.168.150.2	192.168.150.133	DNS	166	Standard query response 0x79b1 AAAA push.services.mozilla.com
96	19.134260627	192.168.150.2	192.168.150.133	DNS	161	Standard query response 0x05d A push.services.mozilla.com
358	19.526613450	192.168.150.133	192.168.150.2	DNS	97	Standard query 0x3513 A firefox.settings.services.mozilla.com
359	19.527109487	192.168.150.133	192.168.150.2	DNS	97	Standard query 0xb728 AAAA firefox.settings.services.mozilla.com
363	20.741674384	192.168.150.133	192.168.150.2	DNS	78	Standard query 0x188f A nnonline.com

Frame 87: 133 bytes on wire (1064 bits), 133 bytes captured (1064 bits) on interface eth0  
 Ethernet II, Src: VMware\_d1:48:60:00:00:00, Dst: VMware\_d1:48:60:00:00:00  
 Internet Protocol Version 4, Src: 192.168.150.2, Dst: 192.168.150.133  
 User Datagram Protocol, Src Port: 53, Dst Port: 48758  
 Domain Name System (response)

dns  
filter

This screenshot displays Wireshark filtered for dns traffic. The packets visible are DNS queries and responses, which are used to resolve domain names to IP addresses. This image provides visual evidence of how the computer performs DNS lookups during network activity.

No.	Time	Source	Destination	Protocol	Length	Info
1115	121.554891369	172.217.24.132	192.168.150.133	TLSv1.3	388	Application Data, Application Data
1117	121.629374677	172.217.24.132	192.168.150.133	TLSv1.3	869	Application Data, Application Data, Application Data, Applic
1118	121.630761136	192.168.150.133	172.217.24.132	TLSv1.3	93	Application Data
1120	121.739534988	192.168.150.133	172.217.24.132	TCP	1514	43270 -> 443 [ACK] Seq=13222 Ack=718970 Win=65535 Len=1460 [T
1121	121.739729258	192.168.150.133	172.217.24.132	TLSv1.3	236	Application Data
1124	121.741997251	192.168.150.133	142.250.182.78	TLSv1.3	386	Application Data
1125	121.742286246	192.168.150.133	142.250.182.78	TLSv1.3	929	Application Data
1128	121.748935369	192.168.150.133	172.217.24.132	TLSv1.3	252	Application Data
1130	121.749349367	192.168.150.133	172.217.24.132	TLSv1.3	85	Application Data
1132	121.760841320	192.168.150.133	172.217.24.132	TLSv1.3	89	Application Data
1134	121.857109882	172.217.24.132	192.168.150.133	TLSv1.3	347	Application Data, Application Data, Application Data
1135	121.858342417	192.168.150.133	172.217.24.132	TLSv1.3	93	Application Data
1137	121.878553334	142.250.182.78	192.168.150.133	TLSv1.3	517	Application Data, Application Data
1138	121.883464670	142.250.182.78	192.168.150.133	TLSv1.3	124	Application Data, Application Data
1140	121.884389611	192.168.150.133	142.250.182.78	TLSv1.3	93	Application Data

Frame 1: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface eth0  
 Ethernet II, Src: VMware\_d1:48:60:00:00:00, Dst: VMware\_d1:48:60:00:00:00  
 Internet Protocol Version 4, Src: 192.168.150.133, Dst: 172.217.24.132  
 Transmission Control Protocol, Src Port: 50526, Dst Port: 443, Seq: 13222, Win: 65535, Len: 1460  
 Transport Layer Security

tls  
filter

This image shows Wireshark filtered for tls traffic. The packets in the capture are related to Transport Layer Security (TLS), which is used to encrypt web traffic for confidentiality and integrity. This screenshot demonstrates that secure, encrypted communication is also being captured and analyzed.

No.	Time	Source	Destination	Protocol	Length	Info
70	115.268999283	192.168.150.133	34.107.221.82	HTTP	364	GET /success.txt?ip=v4 HTTP/1.1
72	115.338314491	34.107.221.82	192.168.150.133	HTTP	270	HTTP/1.1 200 OK (text/plain)
87	118.686121936	192.168.150.133	142.250.206.68	HTTP	380	GET / HTTP/1.1
89	119.099269564	142.250.206.68	192.168.150.133	HTTP	1150	HTTP/1.1 302 Found (text/html)

Frame 70: 364 bytes on wire (2912 bits), 364 bytes captured (2912 bits) on interface eth0  
 Ethernet II, Src: VMware\_d1:48:60:00:00:00, Dst: VMware\_d1:48:60:00:00:00  
 Internet Protocol Version 4, Src: 192.168.150.133, Dst: 34.107.221.82  
 Transmission Control Protocol, Src Port: 49842, Dst Port: 80, Seq: 11526899283, Win: 0, Len: 364  
 Hypertext Transfer Protocol  
 GET /success.txt?ip=v4 HTTP/1.1  
 Host: detectportal.firefox.com  
 User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:128.0) Gecko/20100101 Firefox/128.0  
 Accept: \*/\*  
 Accept-Language: en-US,en;q=0.5  
 Accept-Encoding: gzip, deflate  
 Connection: keep-alive  
 Priority: u=0  
 Pragma: no-cache  
 Cache-Control: no-cache  
 \r\n

packet  
details

This image shows the packet details pane in Wireshark for an HTTP GET request. It displays the structure of the request, including the method (GET), the URL (/success.txt?ip=v4), the host (detectportal.firefox.com), and various headers like User-Agent, Accept, and Cache-Control.

---

## Protocols Identified

PROTOCOL	PURPOSE	EXAMPLE FROM CAPTURE
HTTP	Transfers web page data between client and server	GET request to example.com
DNS	Resolves domain names to IP addresses	Query for example.com
TLS	Encrypts web traffic for confidentiality and integrity	Encrypted HTTPS session to example.com

## Key Learnings

- Wireshark provides in-depth visibility into network communications.
- Protocol filters help isolate and study specific traffic types.
- Packet-level analysis can aid in troubleshooting network issues and detecting suspicious activity.

## Conclusion

This task demonstrated how Wireshark can be used to capture and analyze network traffic effectively. By applying filters for HTTP, DNS, and TLS, it was possible to identify key protocols and their functions within normal network activity. The exercise reinforced packet analysis skills and highlighted the role of encrypted communication in protecting data confidentiality.

---