HTTP Traffic Analysis Using Wireshark

(Portfolio Project – Network Traffic Analysis & Packet Inspection)

Overview

This project demonstrates a full HTTP packet analysis using Wireshark. I captured network traffic while visiting a webpage, then analyzed the HTTP GET request and the server's HTTP 200 OK response. This exercise highlights my ability to perform packet-level inspection, extract protocol metadata, measure response times, and interpret how multiple network layers interact.

Tools Used

- Wireshark
- Windows 10
- Microsoft Edge Browser

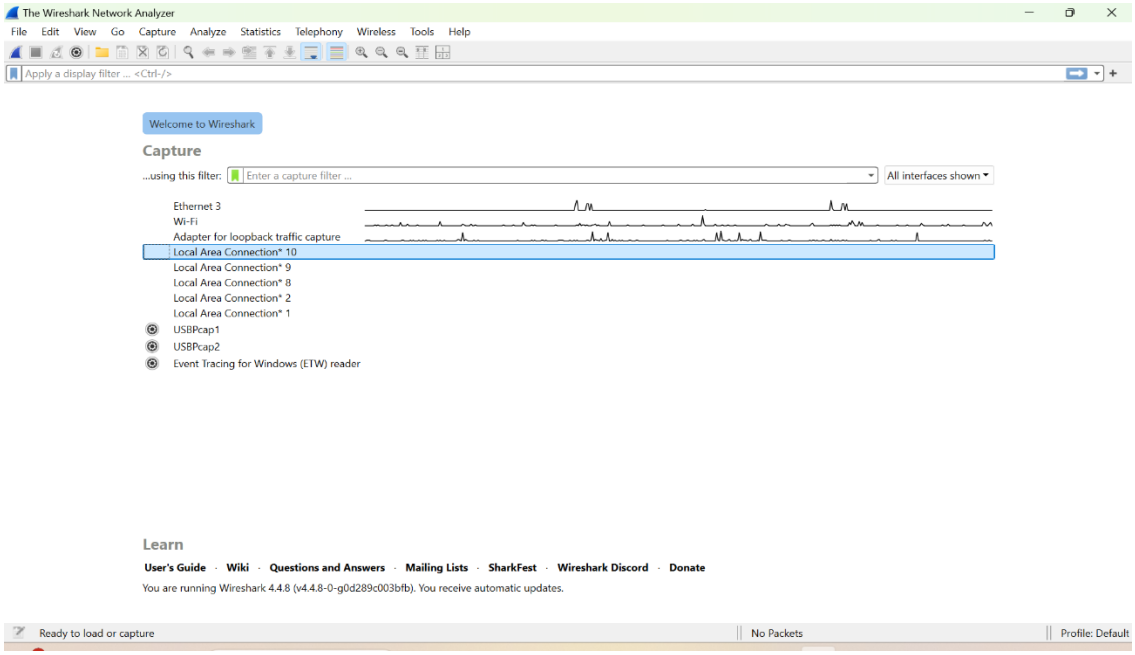
Objectives

- Capture real network packets
- Filter and analyze HTTP traffic
- Examine protocol layers (Ethernet → IP → TCP → HTTP)
- Identify client/server IPs, ports, browser metadata
- Measure HTTP response time
- Produce clear evidence-based documentation

Project Implementation

1. Launching Wireshark

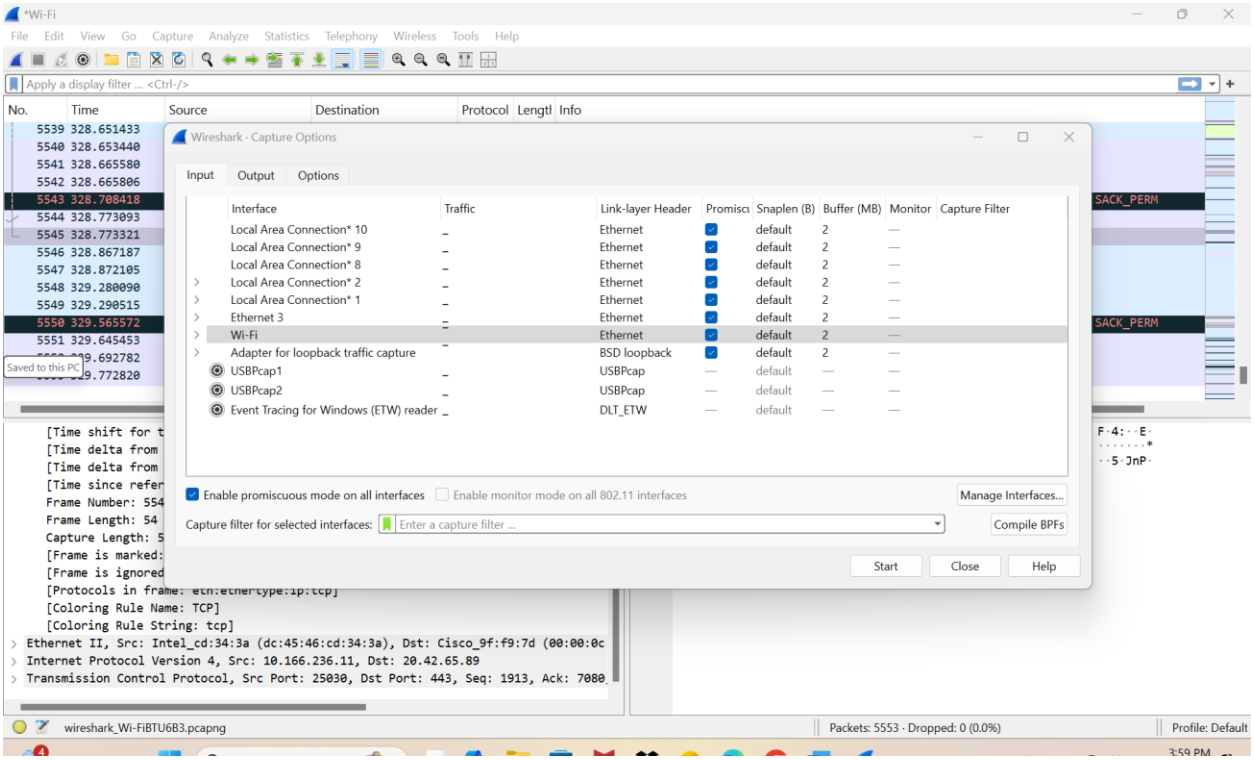
I opened Wireshark and viewed the list of available network interfaces.



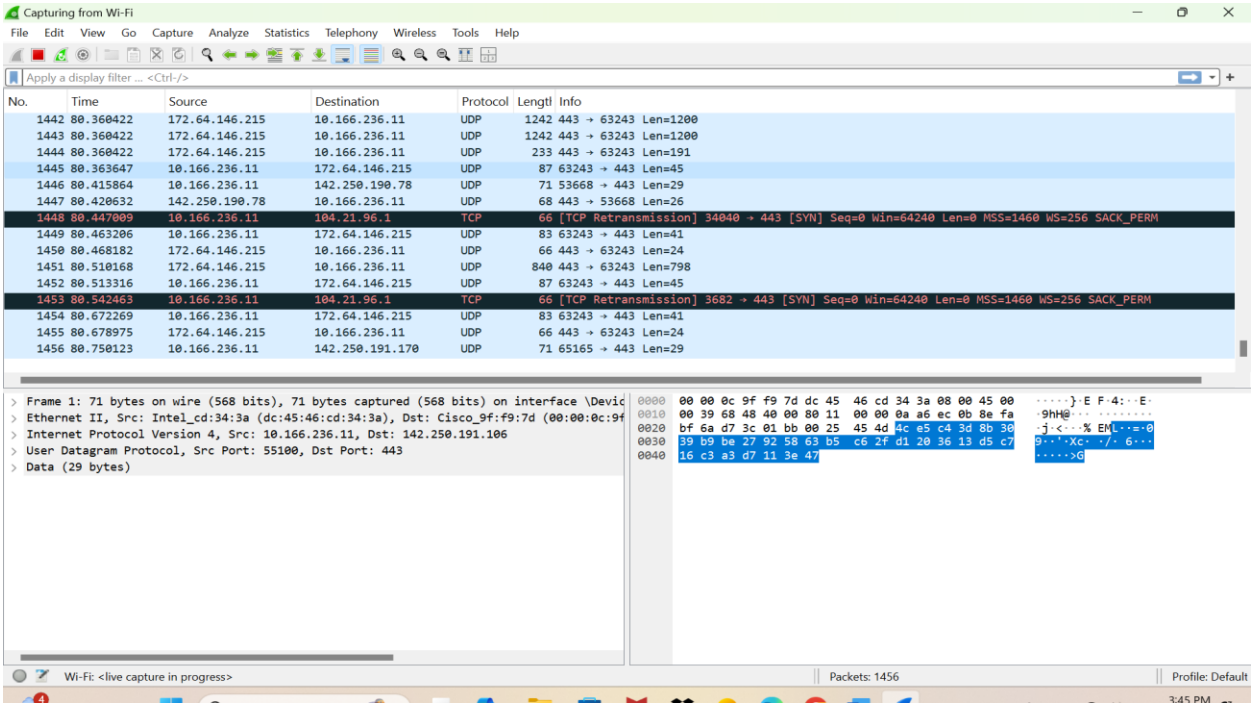
Screenshots/wireshark-startup.png

2. Selecting the Correct Interface

I selected the **Wi-Fi interface** where network activity would occur.



Screenshots/wireshark-interfaces1.png

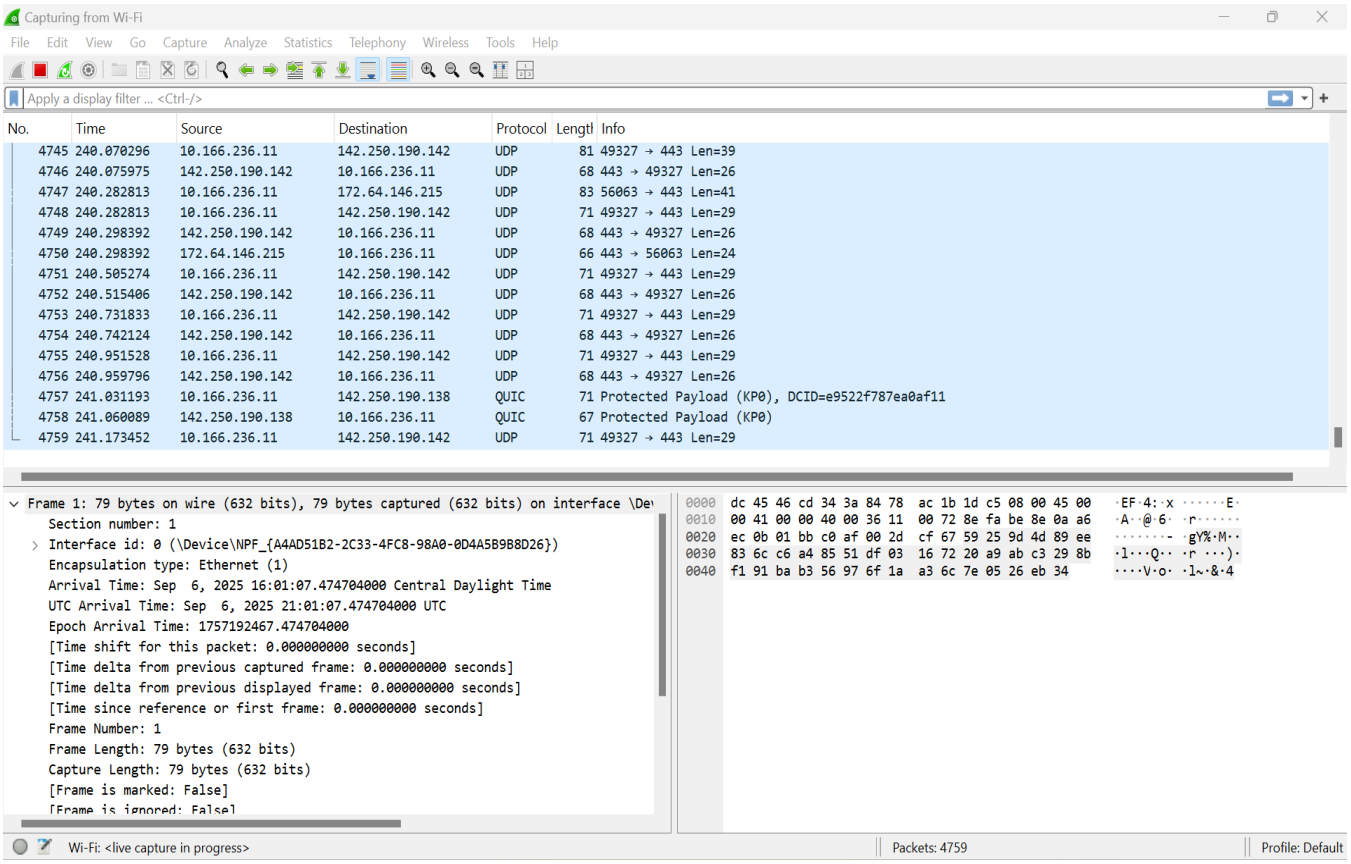


Screenshots/wireshark-interfaces2.png

3. Starting the Packet Capture

Once the interface was selected, I began capturing traffic.

Wireshark immediately displayed live packets in real time.



Screenshots/wireshark-live-capture.png

4. Generating HTTP Traffic

I navigated to:

http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html

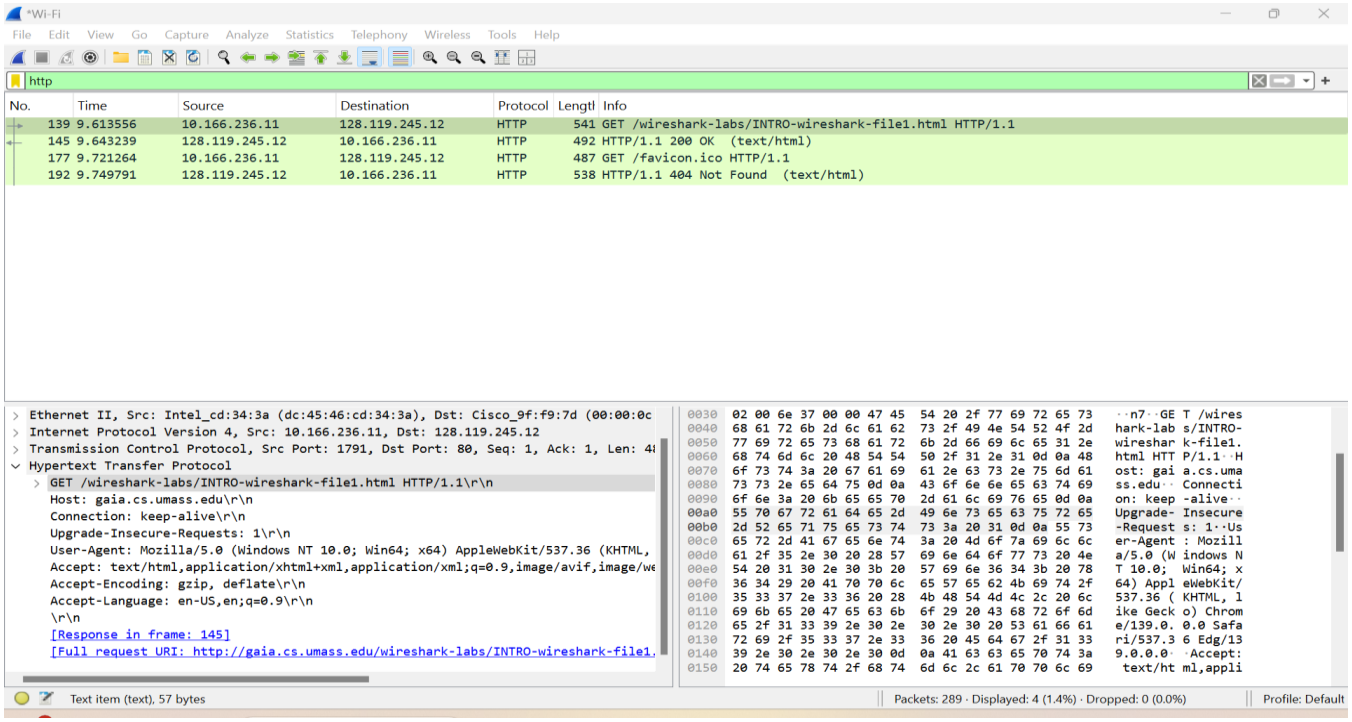
This produced a clear HTTP GET and HTTP OK exchange between my computer and the server.

5. Filtering HTTP Packets

I applied the filter:

http

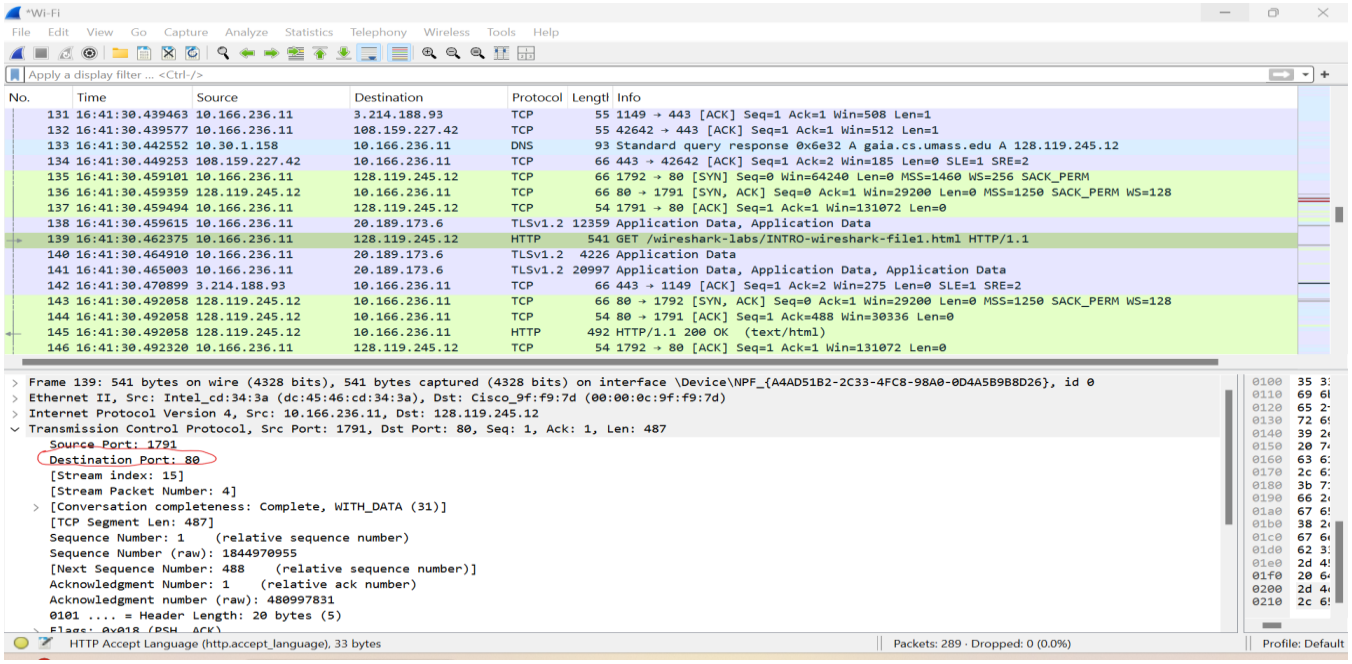
This isolated all HTTP messages for easier analysis.



Screenshots/http-filter.png

6. Identifying the HTTP GET Request

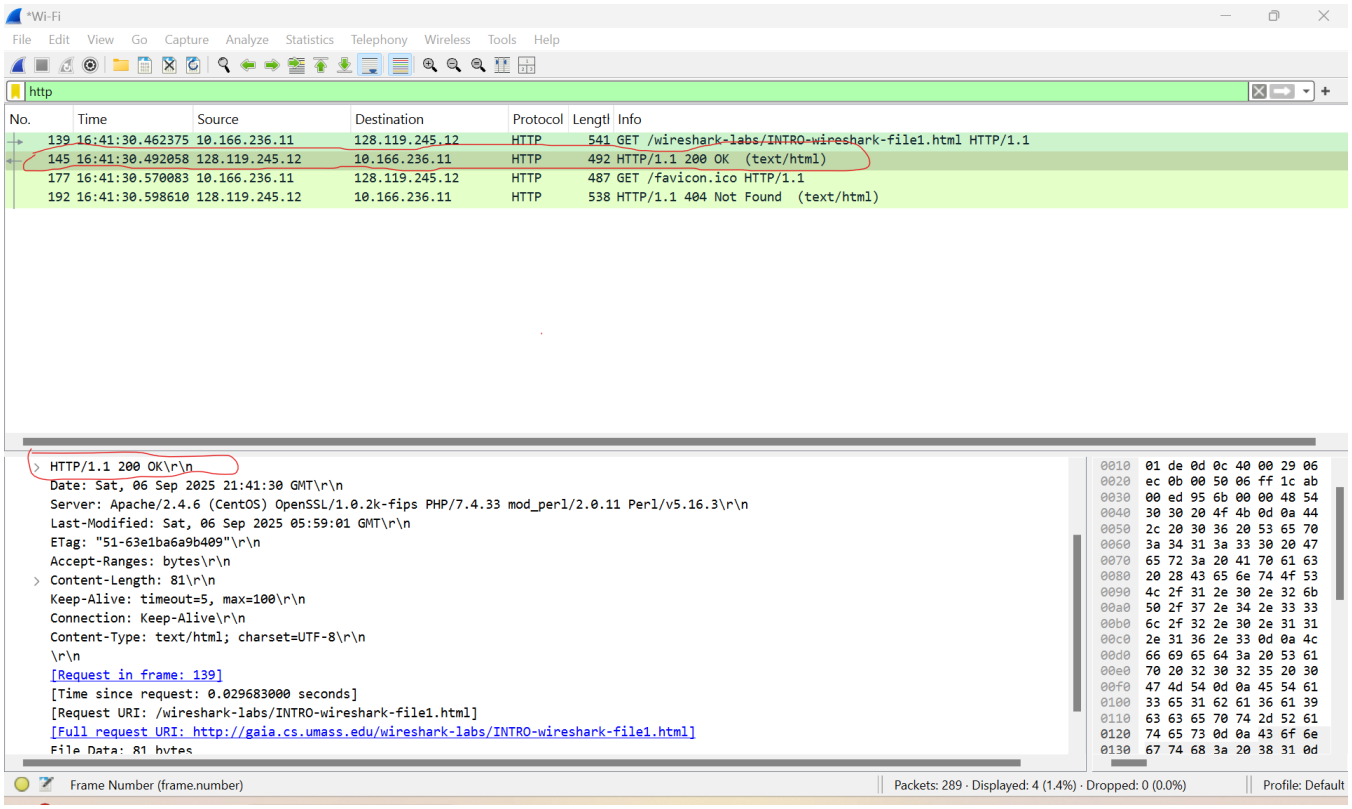
I located and expanded the GET request within Wireshark, reviewing its structure across the protocol layers.



Screenshots/http-get-details.png

7. Inspecting the HTTP 200 OK Response

I then examined the corresponding HTTP OK response returned by the server.



Screenshots/http-get-ok.png

Findings

1. Protocols Observed

During the capture, the following protocols appeared:

- DNS
- TCP
- TLSv1.2
- HTTP

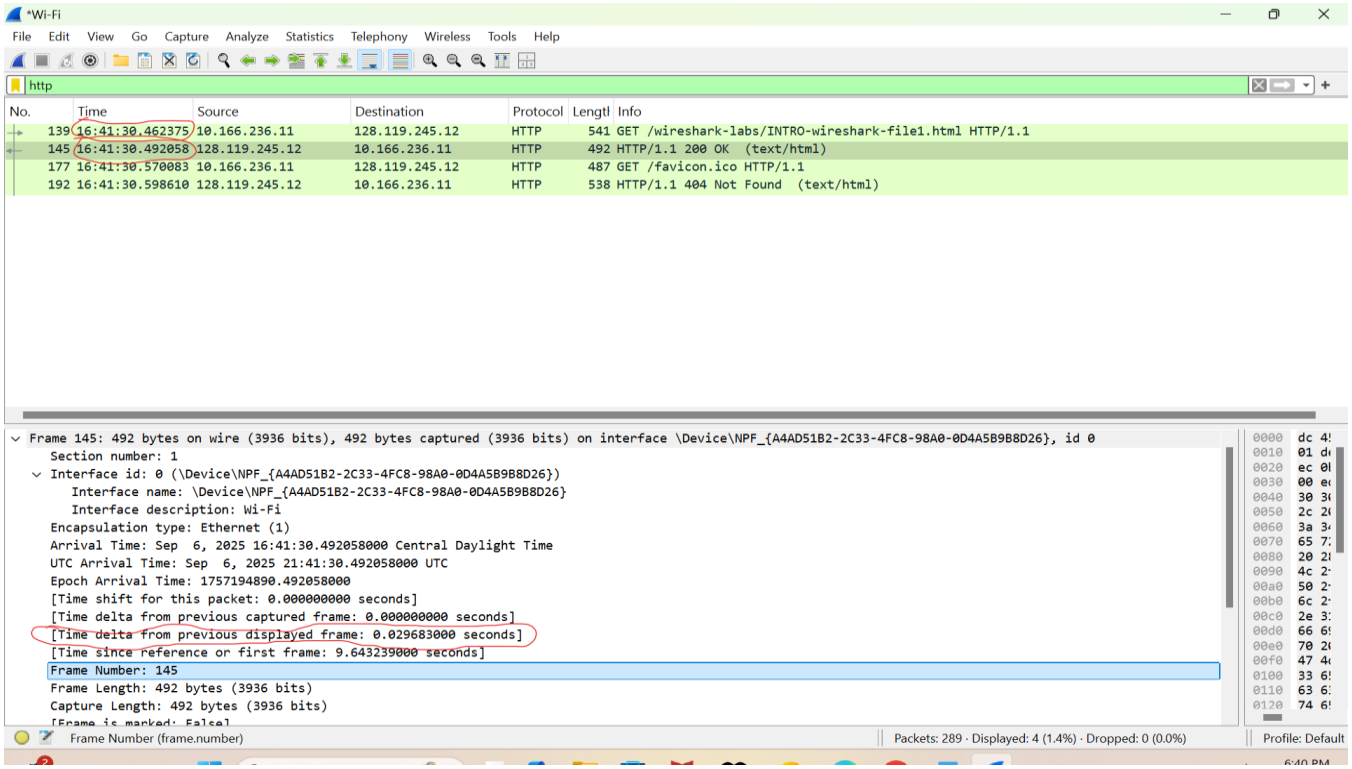
2. HTTP Response Time Measurement

Using Wireshark timestamps:

- **GET request sent:** 16:41:30.462375
- **HTTP OK received:** 16:41:30.492058

Response Time = 0.029683 seconds

This matches the “Time delta from previous displayed frame” within the packet analysis.

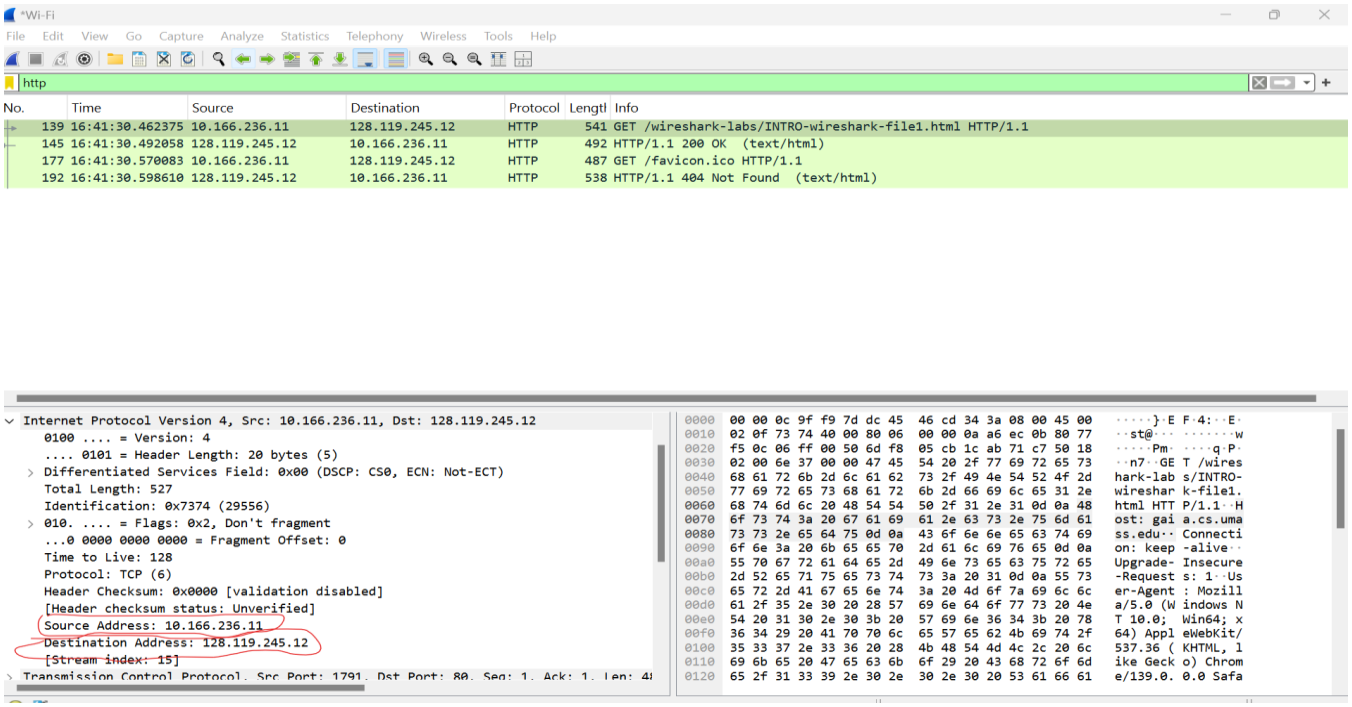


Screenshots/http-time-calc.png

3. IP Addresses Identified

- **Client IP:** 10.166.236.11
- **Server IP (gaia.cs.umass.edu):** 128.119.245.12

These were extracted from the IP header fields.

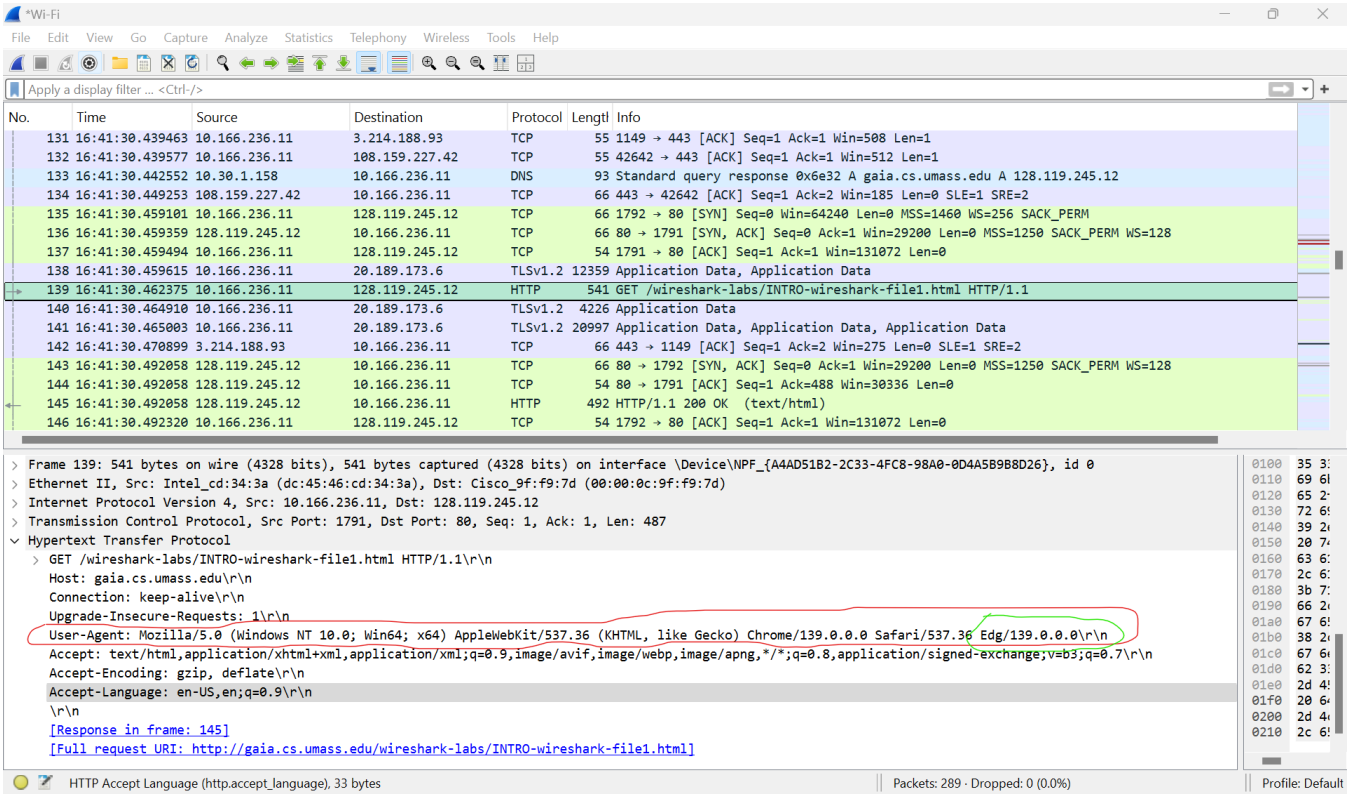


Screenshots/ip-identification.png

4. Browser Information (User-Agent)

From the HTTP GET request header, the User-Agent field showed:

Browser: Microsoft Edge



Screenshots/browser-information.png

5. Destination TCP Port

The HTTP GET request was sent to:

Port 80 (standard port for HTTP traffic)

Skills Demonstrated

- Packet capture and protocol inspection
- HTTP GET/Response analysis
- Filtering and isolating traffic
- Response-time calculation
- Metadata extraction (IP, ports, browser)
- Understanding of TCP/IP and application-layer behavior
- Professional documentation and reporting

Conclusion

This project provided practical experience in network traffic analysis using Wireshark. By capturing and analyzing a real HTTP exchange, I demonstrated an understanding of protocol interactions across multiple layers, the structure of HTTP communication, and essential skills used in SOC investigations, blue-team network monitoring, and cybersecurity analysis.