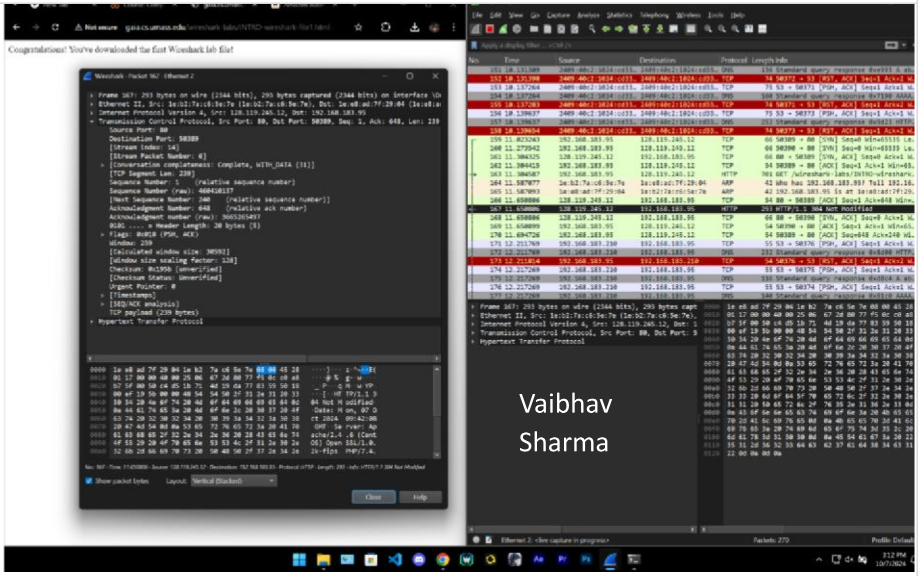
Experiment No: 11	
Name	Vaibhav Sharma
PRN	22070126125
Date of Performance	16th October 2024
Title	To get familiar with the packet sniffer tool “Wireshark” and conduct the packet capturing and packet analysis for various tasks related to HTTP protocol.
Objective	The aim of this experiment is to use the Wireshark packet sniffer tool to capture and analyze packets, specifically focusing on the HTTP protocol to understand how web traffic is transmitted and received.
Setup	- Wireshark was installed and configured to monitor the Ethernet interface. - The goal was to capture and filter HTTP packets and analyze key protocol details such as TCP flags, sequence numbers, acknowledgment numbers, and HTTP headers.

Procedure	1. Initiating Packet Capture: - A packet capture session was started on the Ethernet interface. - Filters like http and tcp were applied to focus the capture on HTTP traffic. 2. Generating HTTP Traffic: - A web page request was initiated using a browser, which triggered HTTP GET requests. - Packets between my local machine (192.168.183.95) and a server (128.119.245.12) on port 80 were captured and analyzed.
Analysis	- Frame 167 in the capture window shows a TCP segment associated with an HTTP request. - Source IP: 128.119.245.12 - Destination IP: 192.168.183.95 - Protocol: TCP - Total Packet Length: 293 bytes - Source Port: 80 (HTTP) - Destination Port: 58389 (client) - Sequence Number: 464041137 - Acknowledgment Number: 3666525497 - TCP Flags: PSH, ACK - Window Size: 239 The TCP payload contains 239 bytes of data, which is part of an HTTP response.
Screenshot	

Congratulations! You've downloaded the first Wireshark lab file!

Not secure gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html
Wireshark · Packet 167 · Ethernet 2
Vaibhav
Sharma

```
> Frame 167: 293 bytes on wire (2344 bits), 293 bytes captured (2344 bits) on interface \D
> Ethernet II, Src: 1e:b2:7a:c6:5e:7e (1e:b2:7a:c6:5e:7e), Dst: 1e:e8:ad:7f:29:04 (1e:e8:a
> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.183.95
v Transmission Control Protocol, Src Port: 80, Dst Port: 58389, Seq: 1, Ack: 648, Len: 239
    Source Port: 80
    Destination Port: 58389
    [Stream index: 14]
    [Stream Packet Number: 6]
  > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 239]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 464041137
    [Next Sequence Number: 240    (relative sequence number)]
    Acknowledgment Number: 648    (relative ack number)
    Acknowledgment number (raw): 3666525497
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window: 239
    [Calculated window size: 30592]
    [Window size scaling factor: 128]
    Checksum: 0x1950 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (239 bytes)
> Hypertext Transfer Protocol
```

Observation	The captured packet represents an HTTP response with minimal content due to the 304 Not Modified status. This status is commonly used to optimize performance by reducing unnecessary data transfer when the cached version of a resource is still valid. Key TCP Details: - Sequence Number: This ensures that packets are reassembled in the correct order. - Acknowledgment Number: This confirms receipt of previous data packets. - Flags (PSH, ACK): These control the flow of data between client and server.
Self-assessment Q&A	Q: What is Wireshark used for in network analysis? Ans: Wireshark captures and analyzes network traffic, allowing users to inspect packet-level data for various protocols, including HTTP, to troubleshoot and study network behavior. Q: How does Wireshark capture HTTP traffic? Ans: Wireshark captures HTTP traffic by monitoring packets sent over port 80 or 443, allowing users to view headers, content, and other protocol-specific data for analysis. Q: What can be learned from HTTP packet analysis using Wireshark? Ans: HTTP packet analysis reveals request/response details, headers, status codes, and payloads, helping understand web communication, troubleshooting, and security analysis.
Conclusion	This experiment demonstrated how to use Wireshark to capture, filter, and analyze network traffic, focusing on the HTTP protocol. It provided insights into how web browsers and servers communicate over TCP, and how HTTP responses like 304 Not Modified help save bandwidth. The packet details, such as sequence and acknowledgment numbers, flags, and HTTP headers, were key elements in understanding the structure of network communication.