PRN: 21070126039 Name: Jainil Patel Batch: AI/ML A2

CN LAB 7

Packet Capturing and Analysis with Wireshark

Aim

Objective of this lab is to get familiar with the packet sniffer tool “Wireshark” and conduct the packet capturing and packet analysis for various tasks related to HTTP protocol.

Theory

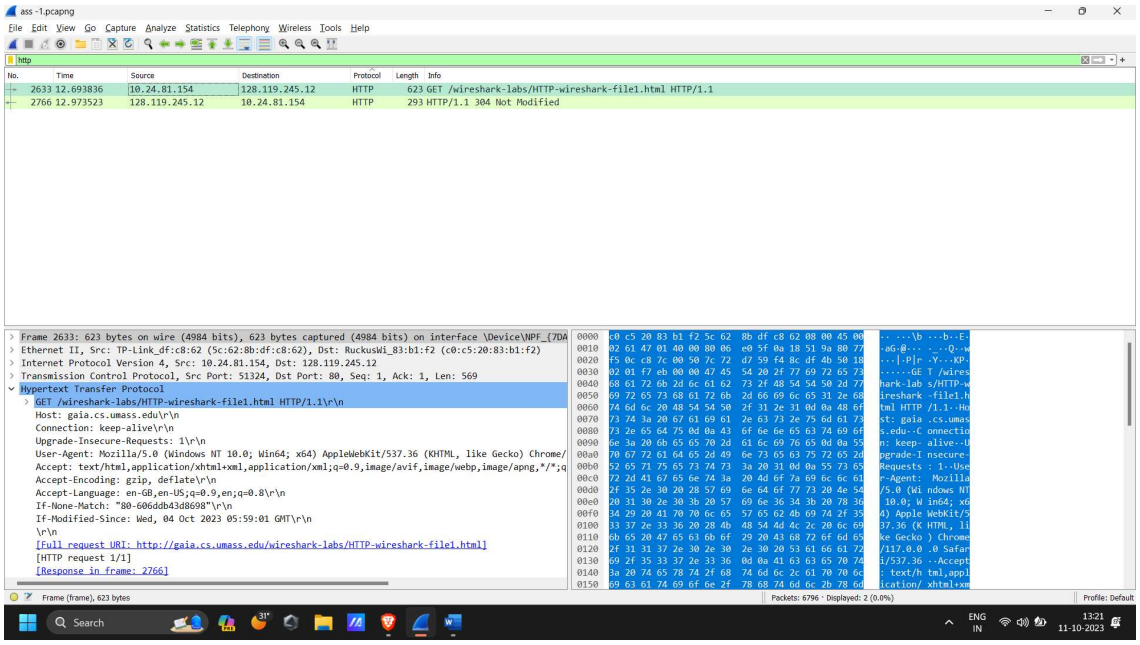
Wireshark is a powerful and widely-used network protocol analyzer that allows users to capture and inspect the traffic on a computer network. It is an open-source tool that can be used for various network analysis tasks, including troubleshooting network issues, monitoring network performance, and analyzing network security.

When it comes to Wireshark and the HTTP protocol, here's a brief overview:

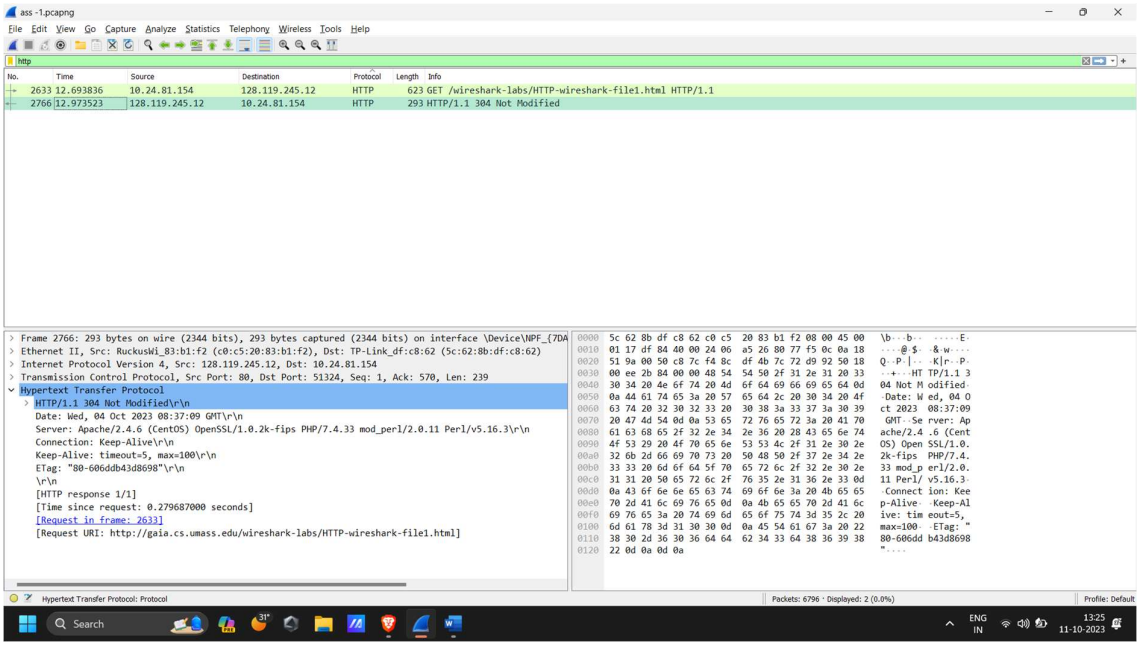
- **HTTP Protocol**: HTTP, which stands for HyperText Transfer Protocol, is the foundation of data communication on the World Wide Web. It is an application layer protocol that governs how data is formatted and transmitted between a web client (such as a web browser) and a web server. HTTP is the protocol responsible for requesting and delivering web pages, images, videos, and other resources on the internet.
- **Using Wireshark with HTTP:**
    - **Packet Capture:** Wireshark captures network traffic by intercepting packets as they are transmitted across a network interface. This includes HTTP traffic.
    - **Filtering:** Wireshark provides various filtering options, allowing you to isolate and analyze HTTP traffic specifically. You can use display filters like "http" to focus on HTTP-related packets.
    - **Packet Inspection:** Once you have captured HTTP packets, Wireshark allows you to inspect the details of each packet. This includes the HTTP request and response headers, which contain important information such as the requested URL, status codes, cookies, and user-agent data.
    - **Follow Stream:** Wireshark also offers a feature called "Follow TCP Stream," which assembles all the packets belonging to an HTTP transaction, making it easier to view the entire conversation between the client and server.

(I)Getting basic information on HTTP protocol:

Request:



Response:



From the packet listing window look at the HTTP GET/ response message and investigate the details by answering to the following questions:

Q.1 Please note down the IP address of your machine and the destination machine (gaia.cs.umass).

Source Machine (Your Machine): 10.24.81.154

Destination Machine (gaia.cs.umass.edu): 128.119.245.12

Q.2 What do you observe in the HTTP request message.

Request Method: GET

Request URI: /wireshark-labs/HTTP-wireshark-file1.html

Request Version: HTTP/1.1

Host: gaia.cs.umass.edu

Connection: keep-alive

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/117.0.0.0 Safari/537.36

Accept: Various content types including text/html, application/xhtml+xml, and more.

Accept-Encoding: gzip, deflate

Accept-Language: en-GB, en-US, en

If-None-Match: "80-606ddb43d8698"

If-Modified-Since: Wed, 04 Oct 2023 05:59:01 GMT

Q.3 Write down the details of the HTTP response message such as status code,

content length and file modified last time.

Status Code: 304 (Not Modified)

Response Phrase: Not Modified

Response Version: HTTP/1.1

Date: Wed, 04 Oct 2023 08:37:09 GMT

Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3

Connection: Keep-Alive

Keep-Alive: timeout=5, max=100

ETag: "80-606ddb43d8698"

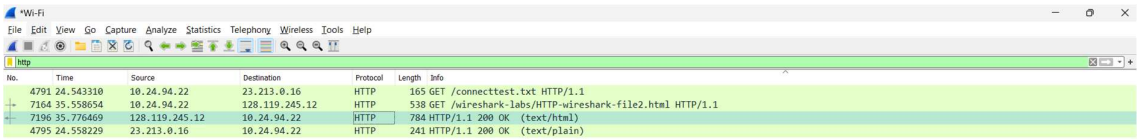
Content Length : Frame 2766: 293 bytes on wire (2344 bits), 293 bytes captured (2344 bits)

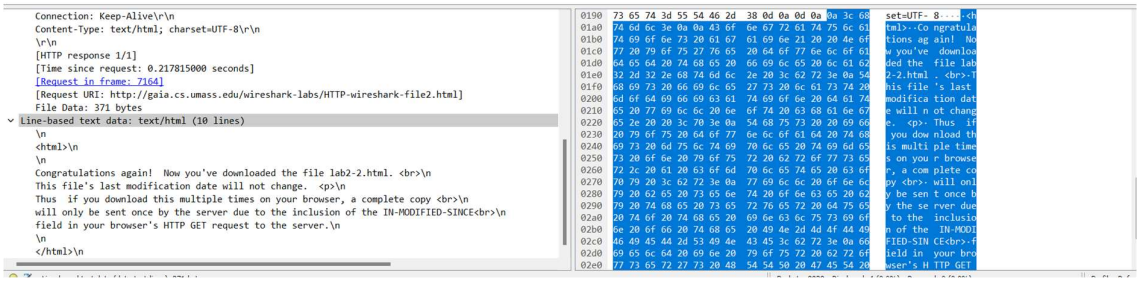
(II) GET request/response interaction:

Q.4 Write down your interesting observations for the GET request and response messages.

Observations for GET Request and Response Messages:

Based on the provided packet capture data, here are some interesting observations for the GET request and response messages:





1. GET Request 1 (Line 1):

   Request Source IP: 10.24.94.22

   Request Destination IP: 23.213.0.16

   Requested Resource: /connecttest.txt

   HTTP Version: HTTP/1.1

   Request Method: GET

   Request Message Length: 165 bytes

   The GET request is sent from the source to the destination, requesting the "/connecttest.txt" resource.

2. GET Request 2 (Line 2):

   Request Source IP: 10.24.94.22

   Request Destination IP: 128.119.245.12

   Requested Resource: /wireshark-labs/HTTP-wireshark-file2.html

   HTTP Version: HTTP/1.1

   Request Method: GET

   Request Message Length: 538 bytes

   The GET request is sent from the source to a different destination, requesting the "/wireshark-labs/HTTP-wireshark-file2.html" resource.

3. HTTP Response 1 (Line 3):

   Response Source IP: 128.119.245.12

   Response Destination IP: 10.24.94.22

   Response Status: 200 OK

   Response Content Type: text/html

Response Message Length: 784 bytes

The first response is for the second GET request. It indicates a successful request with a status of "200 OK," and the content type is "text/html."

4. HTTP Response 2 (Line 4):

Response Source IP: 23.213.0.16

Response Destination IP: 10.24.94.22

Response Status: 200 OK

Response Content Type: text/plain

Response Message Length: 241 bytes

The second response is for the first GET request. It also indicates a successful request with a status of "200 OK," but the content type is "text/plain."

These observations reveal that two different GET requests were made from the source IP 10.24.94.22 to two different destination IPs, and both requests received successful responses with different content types. The first request was for a text/plain resource, and the second request was for a text/html resource.

(III) Getting long document from server:

Procedure: To perform this task, ensure that cache is cleared. Check again your web

browser history and clear it. (Follow the instruction advised in the previous task to

clear the cache).

Start Wireshark and enter the following link to retrieve a long file from UMass

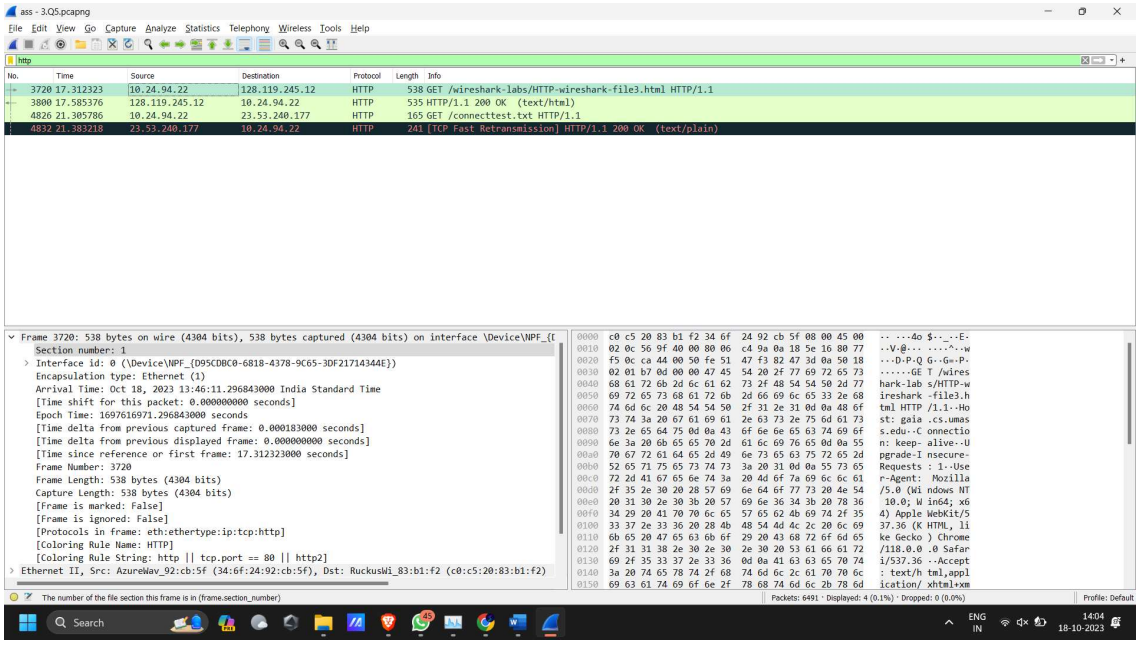
server:

http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html.

Stop the packet capture and filter the packets by entering ‘http’ in the packet filter

window. Based on the above activity, answer the following questions:

Q.5 As you are retrieving a long document, how many request packets are sent from the client to the server?



There is one request packet sent from the client to the server to retrieve the long document.

Packet Information:

Packet Number: 3720

Source: 10.24.94.22

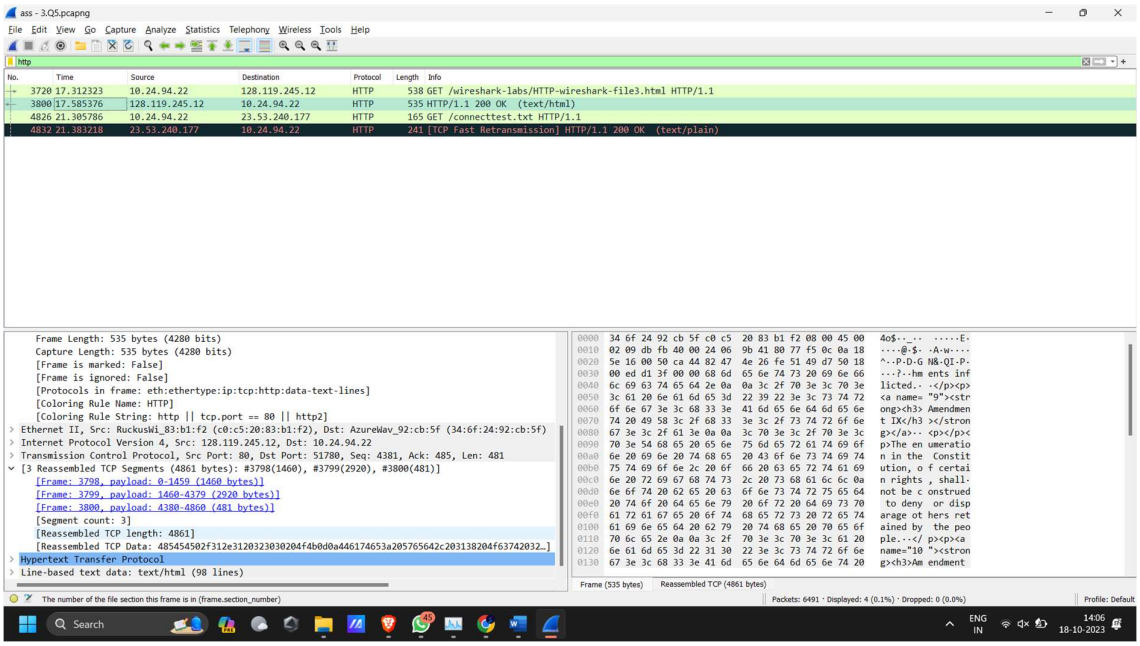
Destination: 128.119.245.12

Protocol: HTTP

Request Method: GET

Requested Resource: /wireshark-labs/HTTP-wireshark-file3.html

Q.6 Write down your understanding of how the HTTP long file is supported by underlying TCP. (also give info where TCP was used here)



HTTP uses the underlying TCP (Transmission Control Protocol) to establish a reliable connection between the client and the server. TCP ensures that data is transmitted in a reliable and ordered manner. In the provided packet information, we can see how TCP is used:

In Packet 3720:

Source Port: 51780

Destination Port: 80

TCP Seq: 1

TCP Ack: 1

Len: 484

In Packet 3800 (HTTP Response):

Source Port: 80

Destination Port: 51780

TCP Seq: 4381

TCP Ack: 485

Len: 481

Reassembled TCP Segments

In Packet 4826:

Source Port: 51784

Destination Port: 80

TCP Seq: 1

TCP Ack: 1

Len: 111

In Packet 4832 (HTTP Response):

Source Port: 80

Destination Port: 51784

TCP Seq: 1

TCP Ack: 112

Len: 187

TCP ensures that the long document is divided into manageable segments for transmission and reassembles these segments at the receiving end. It also handles acknowledgment of received data and retransmission in case of packet loss, making it a reliable choice for transferring large files over HTTP.

Q.7 Inspect the packet which contains the status code and phrase of the response message.

The packet that contains the status code and phrase of the response message is Packet 3800 (HTTP Response). Here's the information:

Packet Number: 3800

Source: 128.119.245.12

Destination: 10.24.94.22

Protocol: HTTP

HTTP Response: HTTP/1.1 200 OK (text/html)

Length: 535 bytes

Status Code: 200

Status Phrase: OK

Content Type: text/html

This packet indicates a successful HTTP response with a status code of 200 and a status phrase of "OK." It also specifies that the content type is "text/html."

(IV) Getting a password protected document from the server:

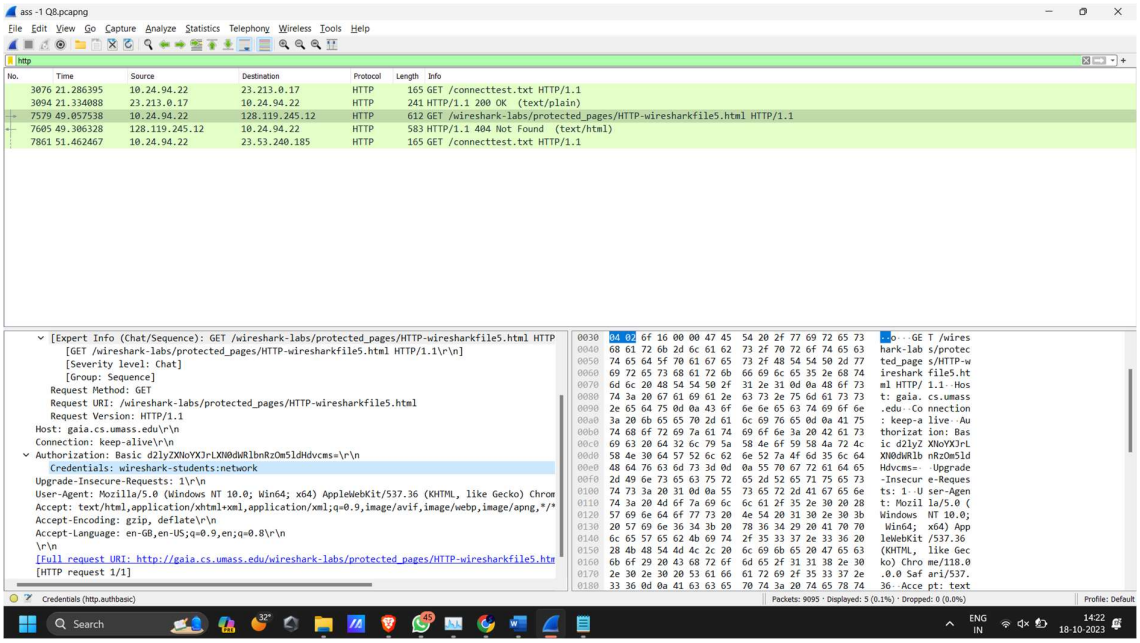
Procedure: In this task, you are trying to access a secured file stored on UMass

server. Username is: wireshark-students and password is: network.

Start the packet capture and enter the following URL on the browser running on your machine:

http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wiresharkfile5.html.

Q.8 Write down your interesting observations for the request and response messages while performing this task.



Request 1:

Packet Number: 3076

Source: 10.24.94.22

Destination: 23.213.0.17

Protocol: HTTP

Request Method: GET

Requested Resource: /connecttest.txt

User-Agent: Microsoft NCSI

Host: www.msftconnecttest.com

Response 1:

Packet Number: 3094

Source: 23.213.0.17

Destination: 10.24.94.22

Protocol: HTTP

Status Code: 200

Status Phrase: OK

Content-Length: 22

Date: Wed, 18 Oct 2023 08:44:38 GMT

Content-Type: text/plain

Response Data: "Microsoft Connect Test"

Request 2:

Packet Number: 7579

Source: 10.24.94.22

Destination: 128.119.245.12

Protocol: HTTP

Request Method: GET

Requested Resource: /wireshark-labs/protected_pages/HTTP-wiresharkfile5.html

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36

Authorization: Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms= (Credentials: wireshark-students:network)

Response 2:

Packet Number: 7605

Source: 128.119.245.12

Destination: 10.24.94.22

Protocol: HTTP

Status Code: 404 (Not Found)

Status Phrase: Not Found

Date: Wed, 18 Oct 2023 08:45:06 GMT

Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3

Content-Type: text/html; charset=iso-8859-1

Response Data: HTML response indicating "404 Not Found"

Request 3:

Packet Number: 7861

Source: 10.24.94.22

Destination: 23.53.240.185

Protocol: HTTP

Request Method: GET

Requested Resource: /connecttest.txt

User-Agent: Microsoft NCSI

Host: www.msftconnecttest.com

Interesting Observations:

Request 1 is made to check connectivity using Microsoft NCSI to the URL www.msftconnecttest.com, and it receives a successful response (Status Code 200).

Request 2 includes an Authorization header with Basic authentication credentials, where the credentials are "wireshark-students:network."

Response 2 indicates a "404 Not Found" error for the requested resource.

Request 3 is another connectivity check to www.msftconnecttest.com.

Conclusion:

In my experience with Wireshark, being able to locate and analyze network packets was incredibly helpful for understanding how network communication works. When I could spot the status codes in HTTP responses, it was especially valuable because it let me know whether requests were successful or if there were issues to address. This was crucial for troubleshooting. Additionally, delving into response messages provided insights into how the server was behaving, which was vital for diagnosing problems, improving performance, and ensuring smooth data transfers. Wireshark remains a must-have tool for network professionals, as it offers a detailed view of network traffic, enabling us to make well-informed decisions.