SIT202: Computer Networks and Communication
Leaning Evidence for Active Class Task 2

Name: Kenisha Corera
Student ID: C23020001

Members in this group activity task:

1. Nishad – C23110001
2. Kenisha – C23020001
3. Shekaina – C23110002
4. Raaid – C23020004
5. Pavithran – C22060015

Activity 1

1. Why do we need application layer protocols?
   - Applications need application layer protocols in order to communicate reliably, effectively, and securely over a network. Consequently, this improves the performance and usability of networked systems.
2. Discuss some examples of application layer protocols.
   - The World Wide Web's data communication system is based on the HTTP (Hypertext Transfer Protocol). From a web server to a web browser, it is used to transfer hypertext documents, such as HTML. In HTTP, requests are sent by clients, usually web browsers, to servers, who then provide the requested resource in return. Because there is no session data stored between queries, each request made using this protocol is stateless.

   - HTTPS (Hypertext transfer protocol secure) – HTTPS is a secure version of HTTP, used for safe communication between a web server and a browser. It combines HTTP with SSL/TLS to encrypt data, protecting sensitive information like passwords and payment details from interception. When using HTTPS, users see a padlock icon in their browser, indicating a secure connection. HTTPS also authenticates the server, ensuring users are communicating with the legitimate server.

   - File Transfer Protocol, or FTP, is a computer network protocol that facilitates file transfers between a client and a server. Users can upload, download, delete, rename, move, and copy files, and it gives them access to and control over files on a remote server..

   - Simple Mail Transfer Protocol (SMTP): SMTP is a network protocol used to transmit emails. Email clients can communicate with email servers and each other over this text-based protocol. To move email from the sender's email client to the recipient's email server, SMTP collaborates with the Mail Transfer Agent (MTA).

   - DNS (Domain name System) – DNS translates human-readable domain names (like www.example.com) into IP addresses that computers use to identify each other on the network. When a user enters a domain name into their browser, a DNS server translates this name into the

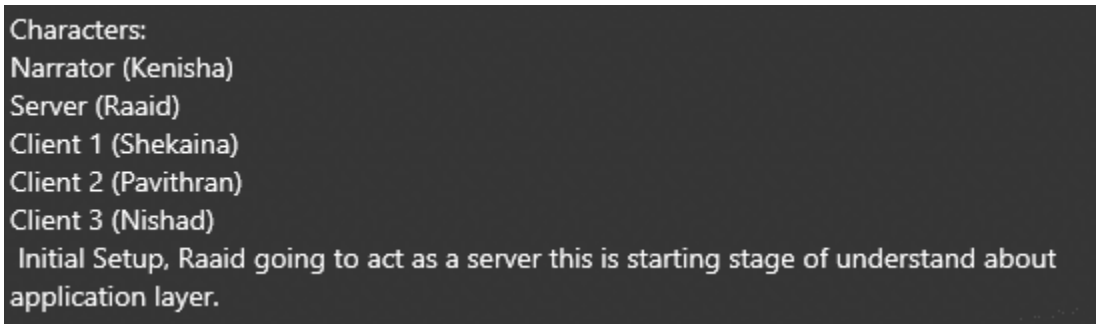
corresponding IP address, allowing the browser to locate and access the website. DNS is crucial for the usability of the internet, as it enables users to use easily memorable domain names instead of numerical IP addresses.

- SNMP, or Simple Network Management Protocol, is a tool for managing and keeping an eye on networks. Network performance management, problem detection and resolution, and network expansion planning are all made possible by it for administrators. The Management Information Base (MIB), an agent, and a manager make up SNMP. While the agents are the devices under the manager's control, the manager is in charge of the network. Details regarding the network devices' status are available in the MIB.

3. Let's consider one of the popular application layer protocols, HTTP to learn the principal operation of HTTP. To carry on this discussion, we can do a small role play.
    a. HTTP is a client-server protocol. Can you identify the key features of HTTP?

    ✓ The World Wide Web's cornerstone for data exchange is HTTP (Hypertext Transfer Protocol). Here are a few essential HTTP characteristics;
        - HTTP operates based on a request/response model where a client (such as a web browser) sends a request to a server, and the server responds with the requested resources.
        - HTTP is a stateless protocol, meaning each request from a client to a server is independent and does not require the server to retain session information or status between requests.
        - headers in HTTP provide essential information about the request or response, such as content type, content length, server details, client details.

    b. Identify a group member who could act as a web server and other three members can act as clients. So, your network has one server and three clients.



    c. Let's assume each client wants to access a web page stored in the web server and we use HTTP to communicate in the application layer level. You can now act as a server-client model communicating the right messages between each entity in sequence to get the information that each client wants. For example, Client 1 could say: "Client 1 initiating TCP connection with the Server". Then, the server replies with the correct response. You need to note down the correct messages in sequence.

Server: Alright team, let's start with the activity on understanding the application layer using HTTP. I'll be the web server, and the rest of you will be clients. Client 2, later you'll act as the proxy server. Ready?

Kenisha CICRA
Clients: Ready! Server and clients I'll introduce the clients, client 1 is Shekaina, Client 2 is pavithran and the last client 3 is Nishad

Shekaina CICRA
Client 1: Ok, I'm initiating TCP connection with the server

Server: Ok, I'm acknowledging TCP connection from Client 1. Connection established.

Shekaina CICRA
Client 1: I'm sending HTTP GET request for page1.html

Server: I'm responding with HTTP 200 OK and the content of page1.html.

Pavithran AIR
Client 2: Ok, I'm initiating TCP connection with the Server.

Server: Ok, I'm acknowledging TCP connection from Client 2. Connection established.

Pavithran AIR
Client 2: I'm sending HTTP GET request for page2.html.

Server: I'm responding with HTTP 200 OK and the content of page2.html.

Nishad
Client 3: Ok, I'm initiating TCP connection with the Server.

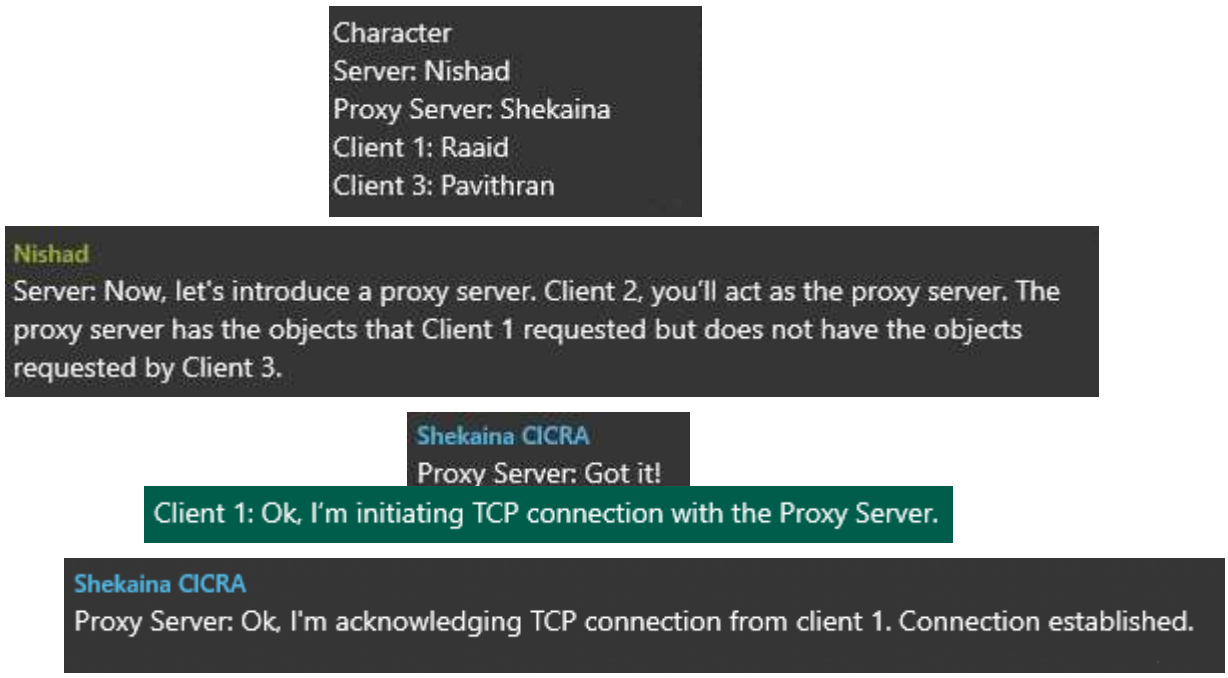
Server: Ok, I'm acknowledging TCP connection from Client 3. Connection established.

Nishad
Client 3: I'm sending HTTP GET request for page3.html.

Server: I'm responding with HTTP 200 OK and the content of page3.html.

d. You can maintain a timeline diagram as follows to show the messages that exchanged between the server and clients.

e. Once you have done the above steps (a to d), now assume a Web cache which is also called a proxy server introduced to the network as shown in the following diagram



Client 1: I'm sending HTTP GET request for page1.html.

Shekaina CICRA
Proxy Server: I'm responding with cached content of page1.html

Pavithran AIR
Client 3: Ok, I'm initiating TCP connection with the Proxy Server.

Shekaina CICRA
Proxy Server: Ok, I'm acknowledging TCP connection from client 3. Connection established.

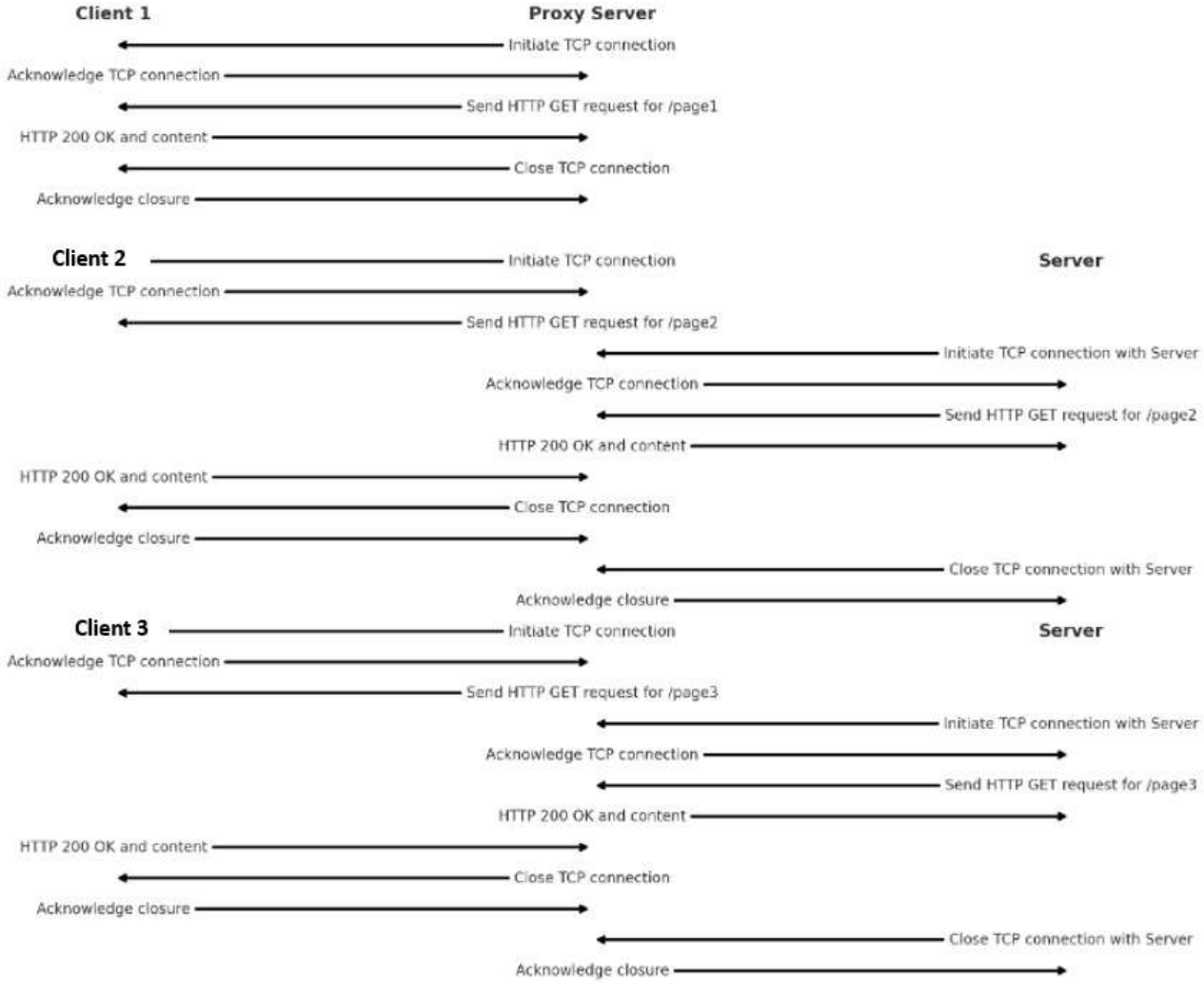
Pavithran AIR
Client 3: I'm sending HTTP GET request for page3.html.

Shekaina CICRA
Proxy Server: I'm forwarding the request to the original server.

Nishad
Server: Ok, I'm acknowledging TCP connection from the Proxy Server. Connection established. I'm responding with HTTP 200 OK and the content of page3.html to the

Shekaina CICRA
Proxy server(client 2): I'm forwarding the content of page3.html to client 3.

Kenisha CICRA
All note down the sequence of messages, ending point complete successfully

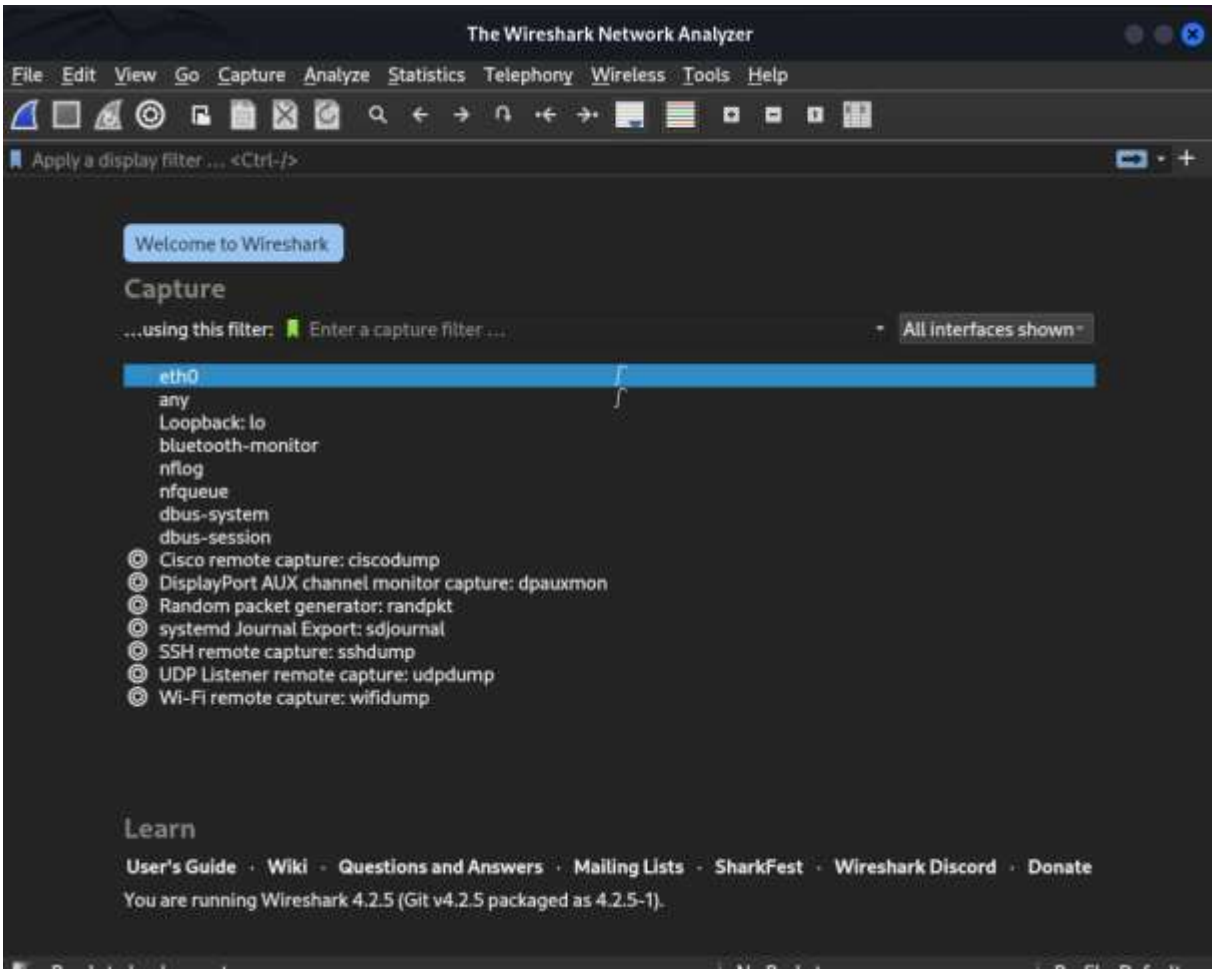

Client 1
Proxy Server
Initiate TCP connection
Acknowledge TCP connection
Send HTTP GET request for /page1
HTTP 200 OK and content
Close TCP connection
Acknowledge closure
Client 2
Initiate TCP connection
Server
Acknowledge TCP connection
Send HTTP GET request for /page2
Initiate TCP connection with Server
Acknowledge TCP connection
Send HTTP GET request for /page2
HTTP 200 OK and content
HTTP 200 OK and content
Close TCP connection
Acknowledge closure
Close TCP connection with Server
Acknowledge closure
Client 3
Initiate TCP connection
Server
Acknowledge TCP connection
Send HTTP GET request for /page3
Initiate TCP connection with Server
Acknowledge TCP connection
Send HTTP GET request for /page3
HTTP 200 OK and content
HTTP 200 OK and content
Close TCP connection
Acknowledge closure
Close TCP connection with Server
Acknowledge closure

Activity 2

a. Open your web browser and clear the browser's cache if you have not done that already

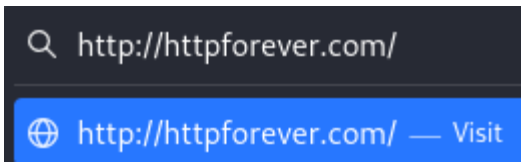
(Already completed this step)

b. Open the Wireshark packet sniffer and start the packet capture (Wireshark captures the packets by default when you open it for the first time)

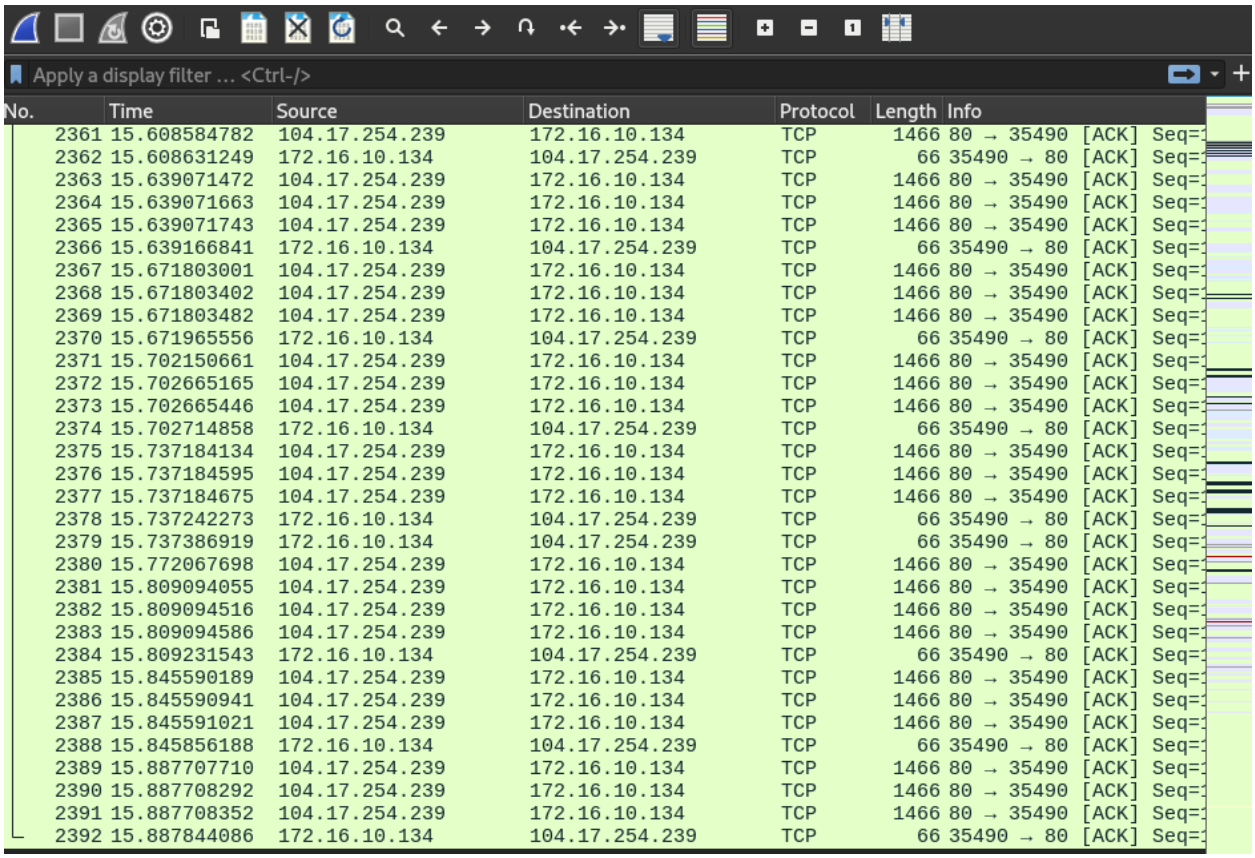


c. Enter the URL with http (not https) into your browser. If you are unsure what to use, you can use http://www.columbia.edu/~fdc/sample.html.

The webpage used for this activity is, http://httpforever.com/



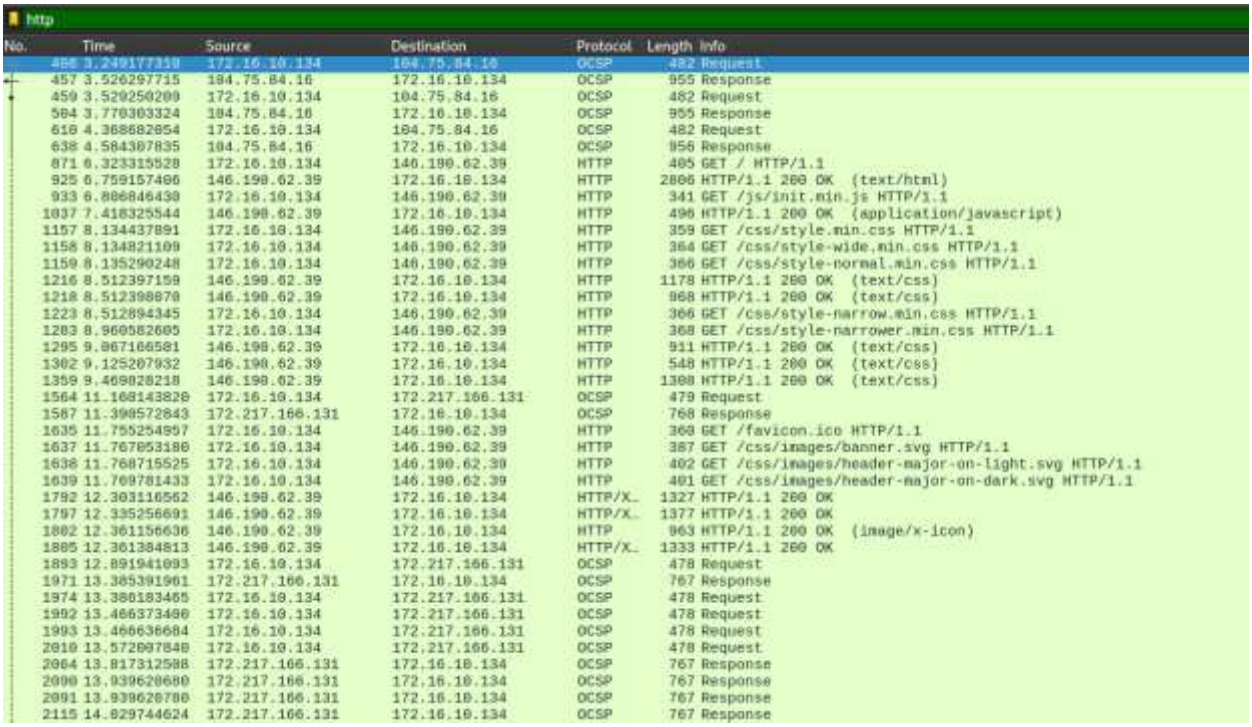
d. Stop Wireshark packet capture once the page is loaded in your browser.



Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
2361	15.608584782	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2362	15.608631249	172.16.10.134	104.17.254.239	TCP	66	35490 → 80 [ACK] Seq=1
2363	15.639071472	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2364	15.639071663	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2365	15.639071743	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2366	15.639166841	172.16.10.134	104.17.254.239	TCP	66	35490 → 80 [ACK] Seq=1
2367	15.671803001	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2368	15.671803402	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2369	15.671803482	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2370	15.671965556	172.16.10.134	104.17.254.239	TCP	66	35490 → 80 [ACK] Seq=1
2371	15.702150661	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2372	15.702665165	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2373	15.702665446	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2374	15.702714858	172.16.10.134	104.17.254.239	TCP	66	35490 → 80 [ACK] Seq=1
2375	15.737184134	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2376	15.737184595	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2377	15.737184675	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2378	15.737242273	172.16.10.134	104.17.254.239	TCP	66	35490 → 80 [ACK] Seq=1
2379	15.737386919	172.16.10.134	104.17.254.239	TCP	66	35490 → 80 [ACK] Seq=1
2380	15.772067698	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2381	15.809094055	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2382	15.809094516	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2383	15.809094586	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2384	15.809231543	172.16.10.134	104.17.254.239	TCP	66	35490 → 80 [ACK] Seq=1
2385	15.845590189	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2386	15.845590941	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2387	15.845591021	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2388	15.845856188	172.16.10.134	104.17.254.239	TCP	66	35490 → 80 [ACK] Seq=1
2389	15.887707710	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2390	15.887708292	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2391	15.887708352	104.17.254.239	172.16.10.134	TCP	1466	80 → 35490 [ACK] Seq=1
2392	15.887844086	172.16.10.134	104.17.254.239	TCP	66	35490 → 80 [ACK] Seq=1

e. We want to analyse HTTP. Therefore, you want to filter out “http” responses. To do that, enter “http” in the filter.



http

No.	Time	Source	Destination	Protocol	Length	Info
455	3.249177319	172.16.10.134	104.75.84.16	OCSP	482	Request
457	3.526297715	104.75.84.16	172.16.10.134	OCSP	955	Response
459	3.529250209	172.16.10.134	104.75.84.16	OCSP	482	Request
504	3.770303324	104.75.84.16	172.16.10.134	OCSP	955	Response
616	4.368882054	172.16.10.134	104.75.84.16	OCSP	482	Request
638	4.584307835	104.75.84.16	172.16.10.134	OCSP	956	Response
871	6.323315528	172.16.10.134	146.190.62.39	HTTP	405	GET / HTTP/1.1
925	6.759157496	146.190.62.39	172.16.10.134	HTTP	2806	HTTP/1.1 200 OK (text/html)
933	6.806846439	172.16.10.134	146.190.62.39	HTTP	341	GET /js/init.min.js HTTP/1.1
1037	7.418325544	146.190.62.39	172.16.10.134	HTTP	496	HTTP/1.1 200 OK (application/javascript)
1157	8.134437891	172.16.10.134	146.190.62.39	HTTP	359	GET /css/style.min.css HTTP/1.1
1158	8.134821109	172.16.10.134	146.190.62.39	HTTP	364	GET /css/style-wide.min.css HTTP/1.1
1159	8.135290248	172.16.10.134	146.190.62.39	HTTP	366	GET /css/style-normal.min.css HTTP/1.1
1216	8.512397159	146.190.62.39	172.16.10.134	HTTP	1178	HTTP/1.1 200 OK (text/css)
1218	8.512398870	146.190.62.39	172.16.10.134	HTTP	968	HTTP/1.1 200 OK (text/css)
1223	8.512894345	172.16.10.134	146.190.62.39	HTTP	366	GET /css/style-narrow.min.css HTTP/1.1
1283	8.960582605	172.16.10.134	146.190.62.39	HTTP	368	GET /css/style-narrower.min.css HTTP/1.1
1295	9.067166581	146.190.62.39	172.16.10.134	HTTP	911	HTTP/1.1 200 OK (text/css)
1302	9.125207932	146.190.62.39	172.16.10.134	HTTP	548	HTTP/1.1 200 OK (text/css)
1359	9.469028218	146.190.62.39	172.16.10.134	HTTP	1308	HTTP/1.1 200 OK (text/css)
1564	11.168143828	172.16.10.134	172.217.166.131	OCSP	479	Request
1567	11.398572843	172.217.166.131	172.16.10.134	OCSP	768	Response
1635	11.755254957	172.16.10.134	146.190.62.39	HTTP	360	GET /favicon.ico HTTP/1.1
1637	11.767053180	172.16.10.134	146.190.62.39	HTTP	387	GET /css/images/banner.svg HTTP/1.1
1638	11.768715525	172.16.10.134	146.190.62.39	HTTP	402	GET /css/images/header-major-on-light.svg HTTP/1.1
1639	11.769781433	172.16.10.134	146.190.62.39	HTTP	401	GET /css/images/header-major-on-dark.svg HTTP/1.1
1792	12.303116562	146.190.62.39	172.16.10.134	HTTP/X…	1327	HTTP/1.1 200 OK
1797	12.335256691	146.190.62.39	172.16.10.134	HTTP/X…	1377	HTTP/1.1 200 OK
1802	12.361156636	146.190.62.39	172.16.10.134	HTTP	963	HTTP/1.1 200 OK (image/x-icon)
1805	12.361384813	146.190.62.39	172.16.10.134	HTTP/X…	1333	HTTP/1.1 200 OK
1893	12.891941093	172.16.10.134	172.217.166.131	OCSP	478	Request
1971	13.385391961	172.217.166.131	172.16.10.134	OCSP	767	Response
1974	13.386183465	172.16.10.134	172.217.166.131	OCSP	478	Request
1992	13.466373480	172.16.10.134	172.217.166.131	OCSP	478	Request
1993	13.466636684	172.16.10.134	172.217.166.131	OCSP	478	Request
2010	13.572087848	172.16.10.134	172.217.166.131	OCSP	478	Request
2064	13.817312588	172.217.166.131	172.16.10.134	OCSP	767	Response
2090	13.939620680	172.217.166.131	172.16.10.134	OCSP	767	Response
2091	13.939620780	172.217.166.131	172.16.10.134	OCSP	767	Response
2115	14.029744624	172.217.166.131	172.16.10.134	OCSP	767	Response

f. Now you can analyse HTTP, requests, responses, and sequences.

(Analyzed and answered in questions below)

g. By analysing the HTTP responses, you should be able to answer the following questions,

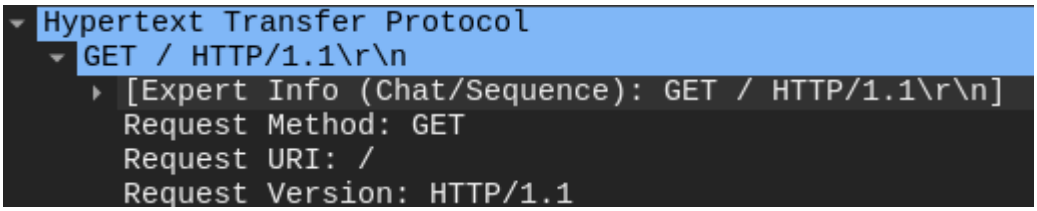
a. What is the sequence of HTTP message exchange?

HTTP/1.1 200 OK (text/html)
GET /js/init.min.js HTTP/1.1
HTTP/1.1 200 OK (application/javascript)
GET /css/style.min.css HTTP/1.1
GET /css/style-wide.min.css HTTP/1.1
GET /css/style-normal.min.css HTTP/1.1
HTTP/1.1 200 OK (text/css)
HTTP/1.1 200 OK (text/css)
GET /css/style-narrow.min.css HTTP/1.1
HTTP/1.1 200 OK (text/css)
HTTP/1.1 200 OK (text/css)
HTTP/1.1 200 OK (text/css)
Request
Response
GET /favicon.ico HTTP/1,1
GET /css/images/banner,svg HTTP/1.1
GET /css/images.header-major-on-light.svg HTTP/1.1
GET /css/images/header-major-on-dark.svg HTTP/1.1
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK (image/x-icon)
HTTP/1,1 200 OK

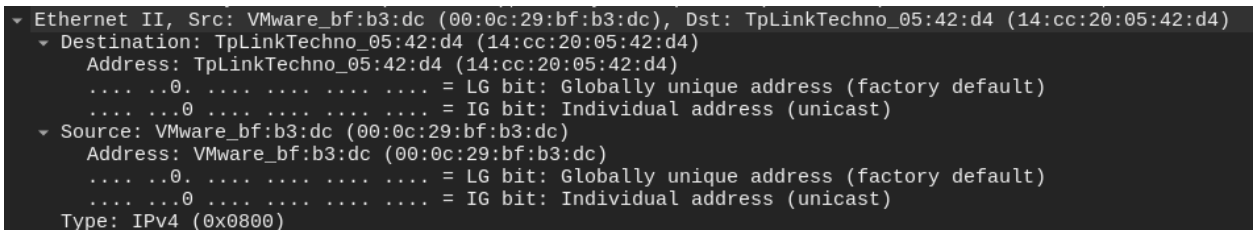
b. Are we using persistence/ non-persistence connection?

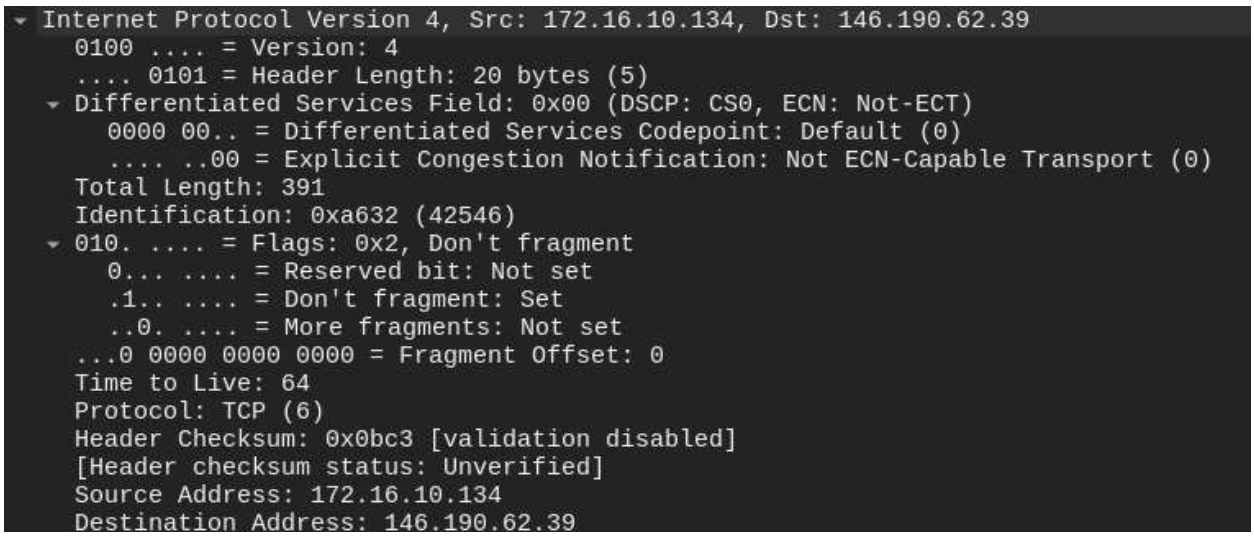
This website is using persistence connection because there are Multiple Requests and Responses between source and destination addresses. Which means that the connection is being reused which is prevalent in persistent connections.

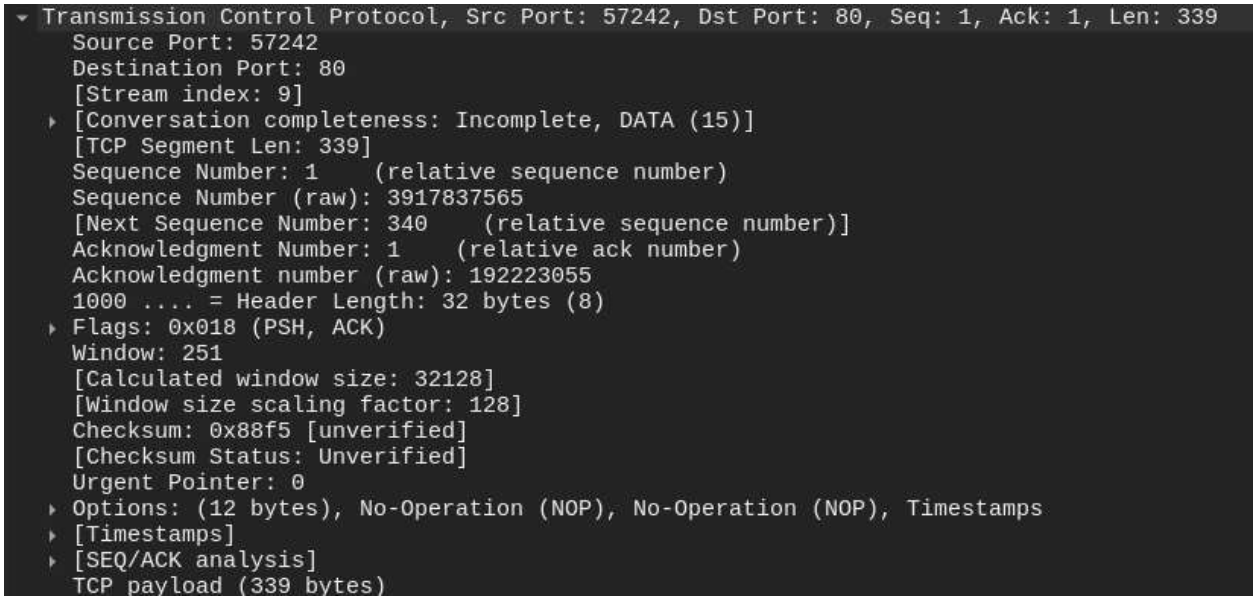
c. What are details you can find in HTTP GET message?

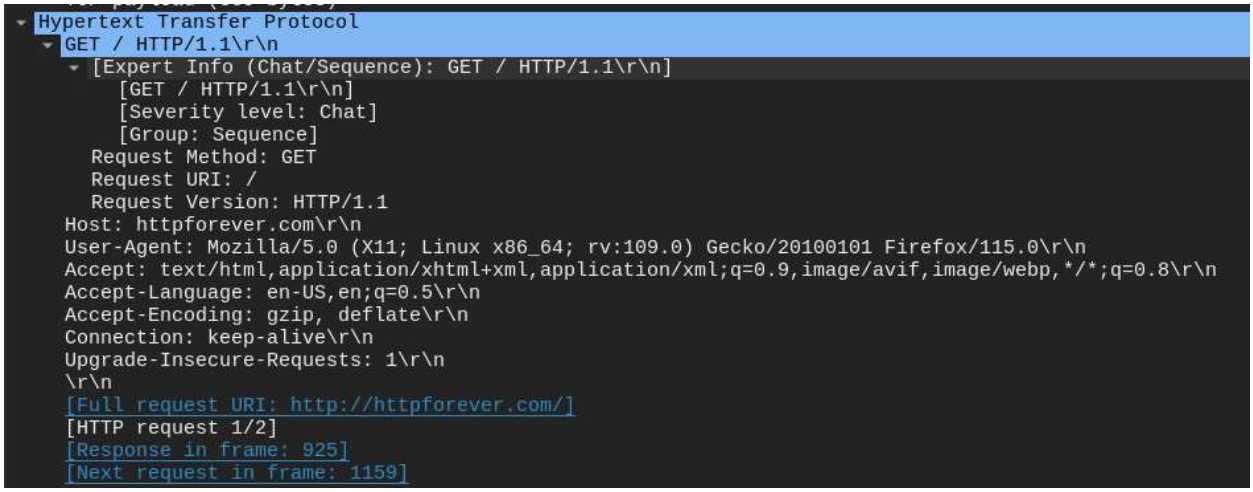


d. Check the packet details in the middle Wireshark packet details pane. Can you identify the details in Ethernet II / Internet Protocol Version 4 / Transmission Control Protocol / Hypertext Transfer Protocol frames?









e. What is the HTTP version your browser used?

- Version 1.1 of HTTP is in use.

f. Identify the response message received from the server?

- All was well with the message from the server.

g. What is version of HTTP that the server is running?

- The server is running HTTP version 1.1.

h. Can you identify the IP address of your computer?

- 172.16.10.134

i. Can you identify the IP address of http://www.columbia.edu/ server?

- Given that we utilized the httpforever website, the webserver's IP address is 146.190.62.39.

j. What is the status code returned from the server to your browser?

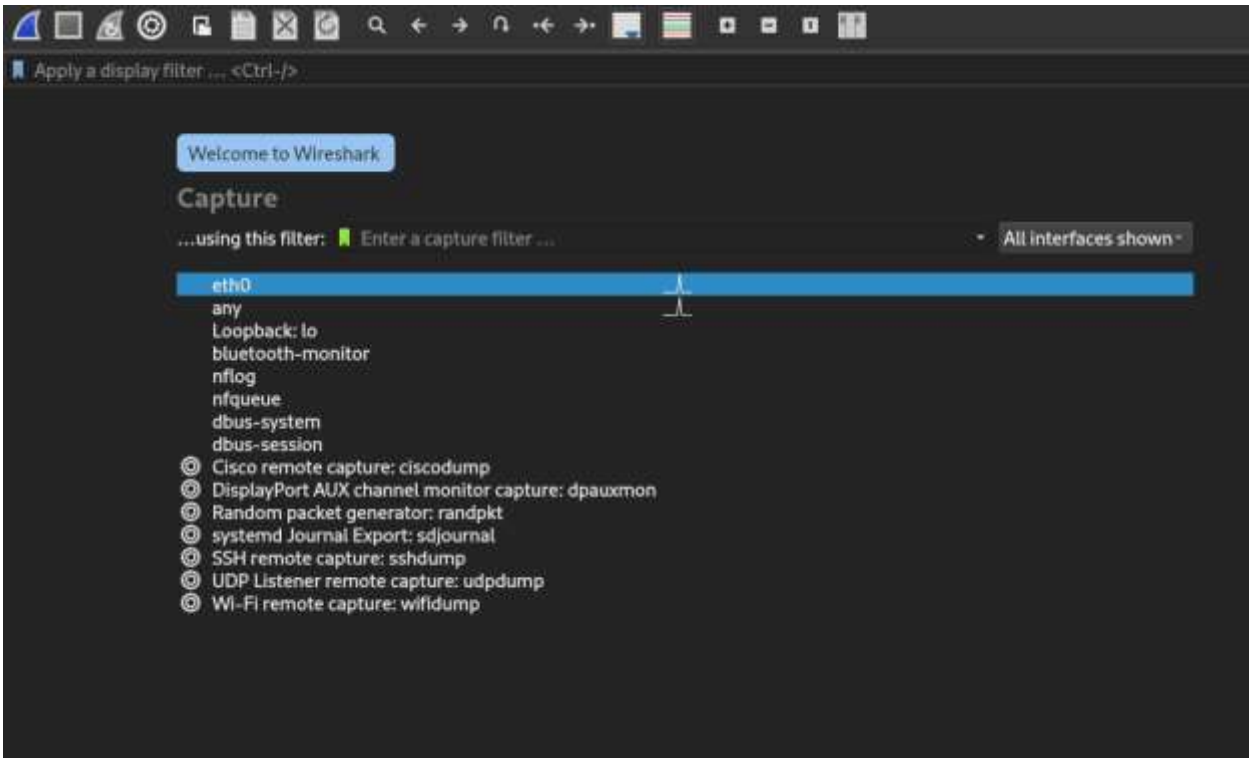
- The server has sent the browser a status code of 200 (OK).

k. When was the HTML file that you are accessing last modified at the server?

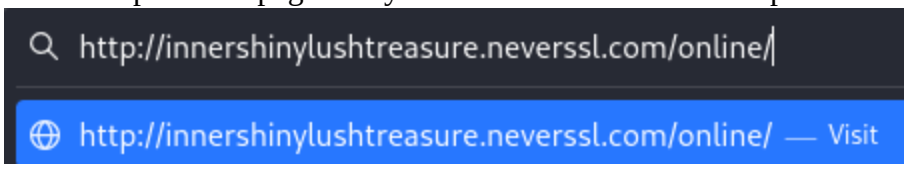
- The last update to the HTML file was made on November 16, 2022.

l. Repeat the analysis by accessing a different web page of your choice.

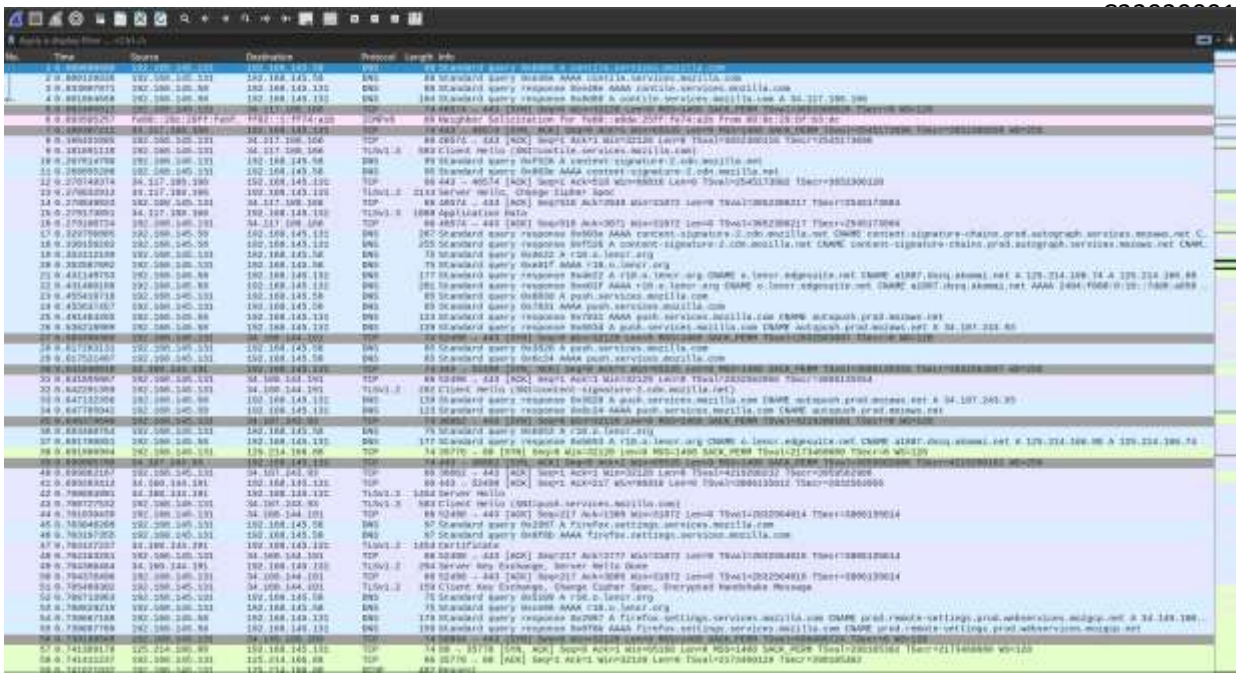
The HTTP webpage I'll be analyzing for this question is http://innershinylushtreasure.neverssl.com/online/
I shall begin by first open wireshark and begin capturing packets.



I shall now load up the webpage in my browser so wireshark can capture traffic.



Once the website has loaded, I shall end packet capturing and filter results by 'http' (as seen in the next page).



http

No.	Time	Source	Destination	Protocol	Length	Info
59	0.741621032	192.168.145.131	125.214.166.88	OCSP	482	Request
70	0.800815334	125.214.166.88	192.168.145.131	OCSP	956	Response
74	0.810319218	192.168.145.131	125.214.166.88	OCSP	482	Request
88	0.892633881	125.214.166.88	192.168.145.131	OCSP	955	Response
89	0.892944416	192.168.145.131	125.214.166.88	OCSP	482	Request
103	0.947012983	125.214.166.88	192.168.145.131	OCSP	955	Response
122	1.001038552	192.168.145.131	125.214.166.88	OCSP	482	Request
126	1.047066482	125.214.166.88	192.168.145.131	OCSP	956	Response
213	4.584089429	192.168.145.131	34.223.124.45	HTTP	432	GET /online/ HTTP/1.1
216	5.202572631	34.223.124.45	192.168.145.131	HTTP	1612	HTTP/1.1 200 OK (text/html)
222	6.119178326	192.168.145.131	34.223.124.45	HTTP	407	GET /favicon.ico HTTP/1.1
224	6.428223348	34.223.124.45	192.168.145.131	HTTP	509	HTTP/1.1 200 OK (PNG)

As you can see above. The communication between server and client has been captured just like in the previous instance.

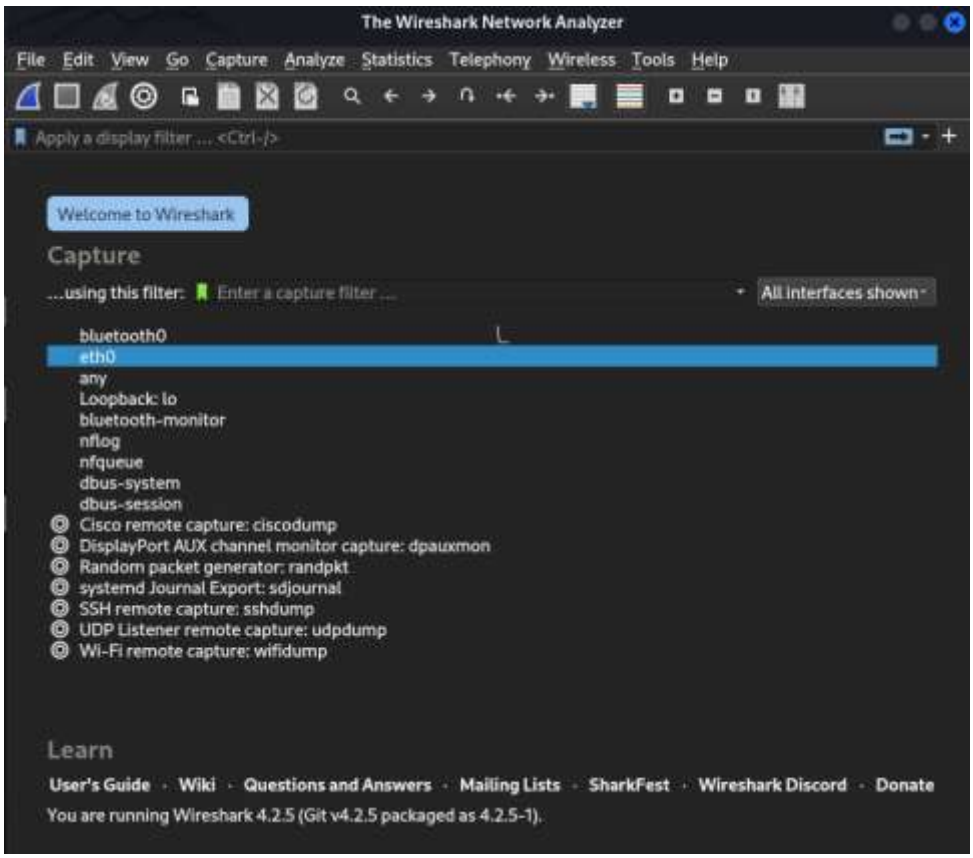
The sequence of HTTP message exchange in this instance is as follows,

- GET /online/ HTTP/1.1
- HTTP/1.1 200 OK (text/html)
- GET /favicon.ico HTTP/1.1
- HTTP/1.1 200 OK (PNG)

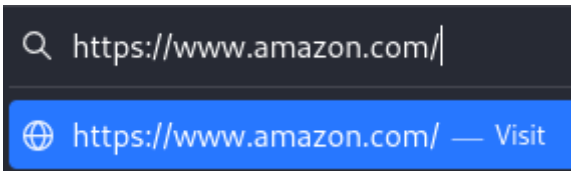
We can also see that both browser and server use HTTP version 1.1 and the status code returned from the server to the browser is 200 (OK). The Client IP address is 192.168.145.131 and the Server IP address is 34.223.124.45 and finally the last modified date of the HTML file was on 29th June 2022

h. What happens when you use “https”? Can you analyse the responses? Discuss your answer with your group members.

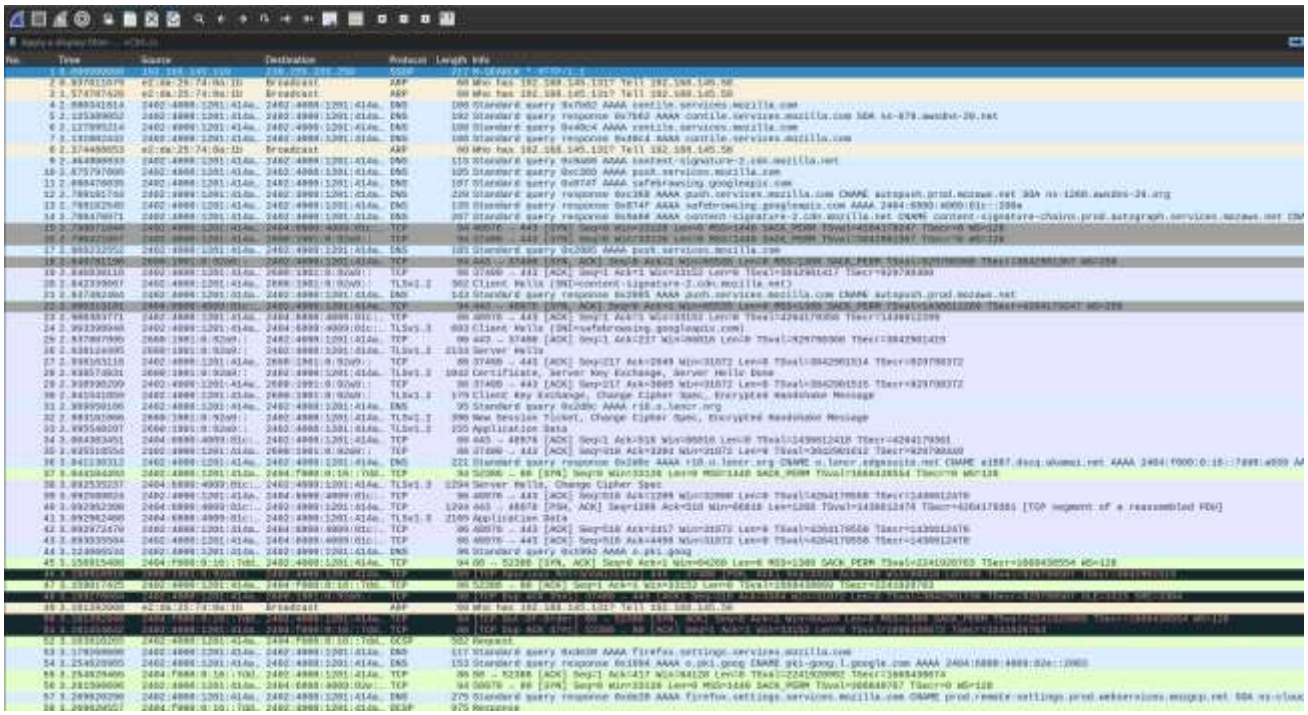
- To begin, we open Wireshark and begin capturing packets as instructed.



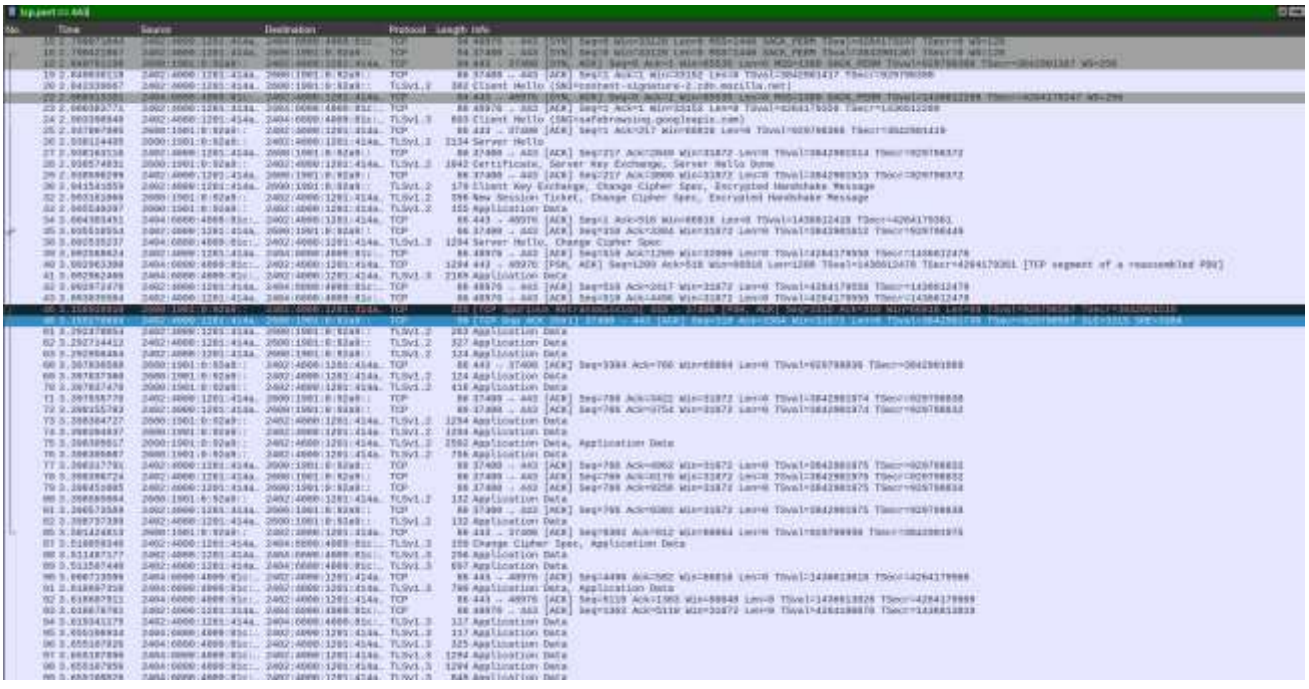
- Then we load up the website on the browser. For this example, we chose https://www.amazon.com/



- Once the website has loaded successfully, we end packet capturing.



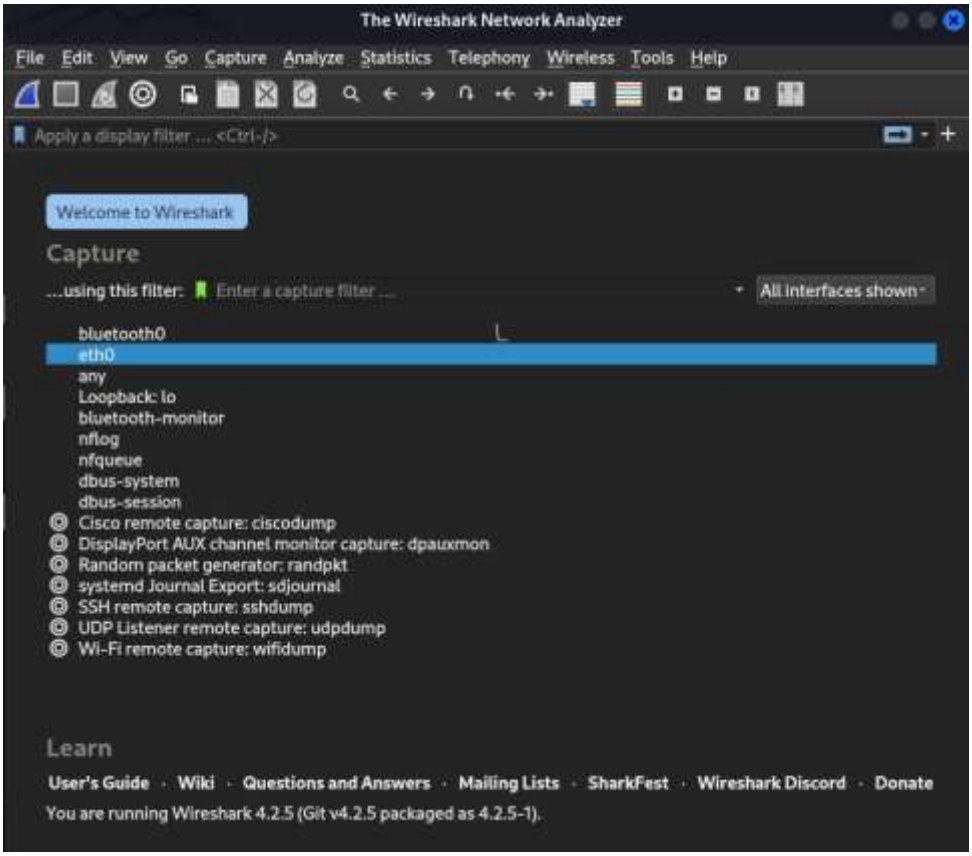
- Since HTTPS uses TCP port 443 we can filter HTTPS traffic by using the filter ‘tcp.port == 443’



- As mentioned in the Above and Beyond Task. Wireshark cannot analyze HTTPS due to its encryption. So the only possible method of displaying HTTPS traffic is by decrypting it using SSL and TLS packets.

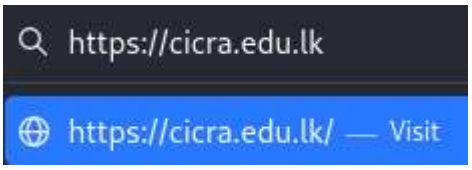
Above and Beyond Tasks

1. Open your web browser and clear the browser's cache. Open the Wireshark packet sniffer and start the packet capture

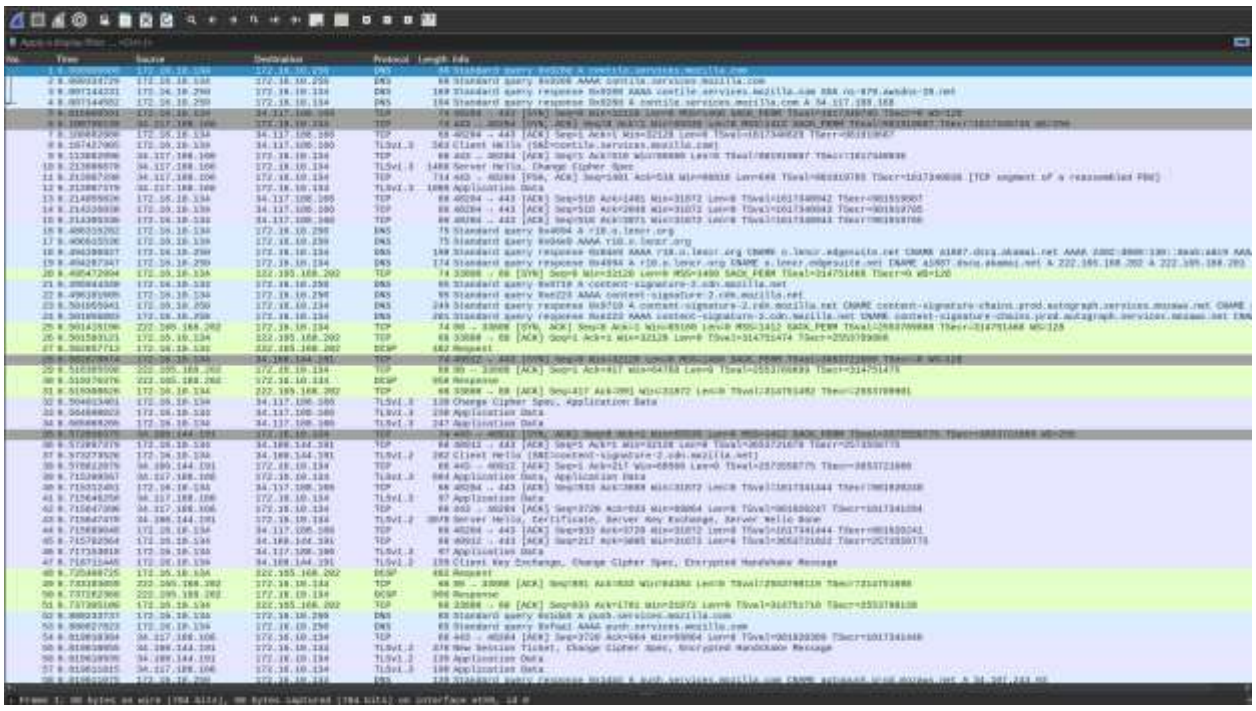


2. Enter a URL with HTTPS in your browser.

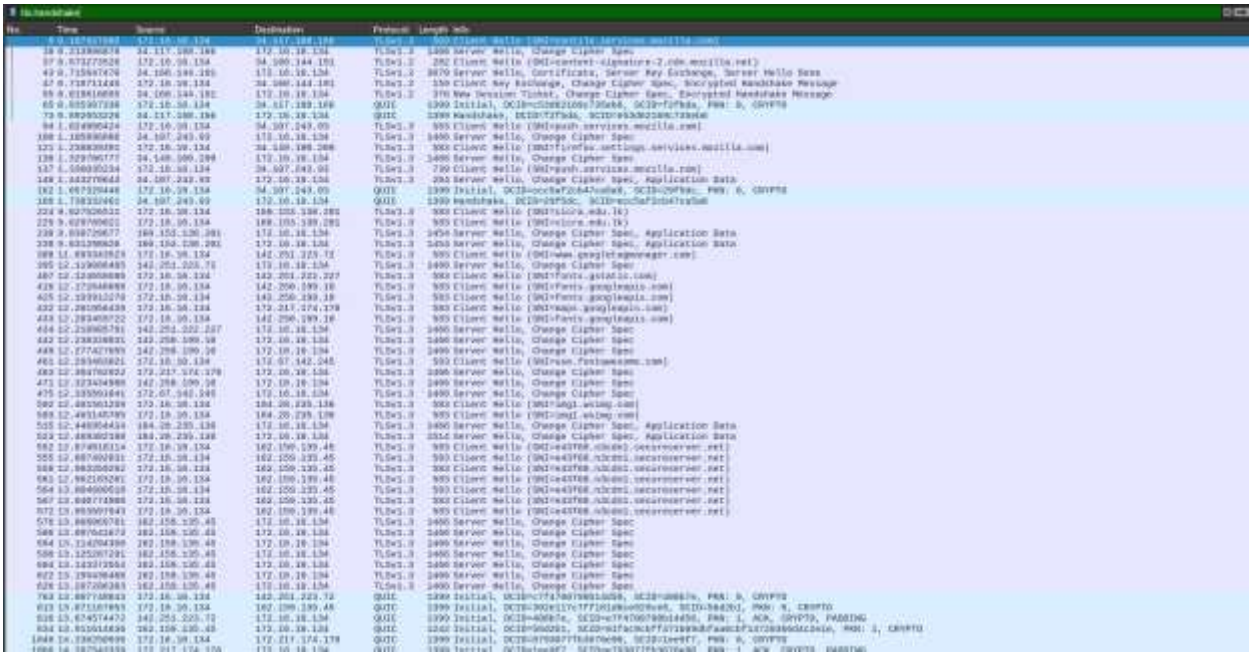
   The webpage used for this activity is, https://cicra.edu.lk/



3. Stop packet capture and you can start analysing the packets. Explain the operation and handshake process of TLS using the screen captures of Wireshark.



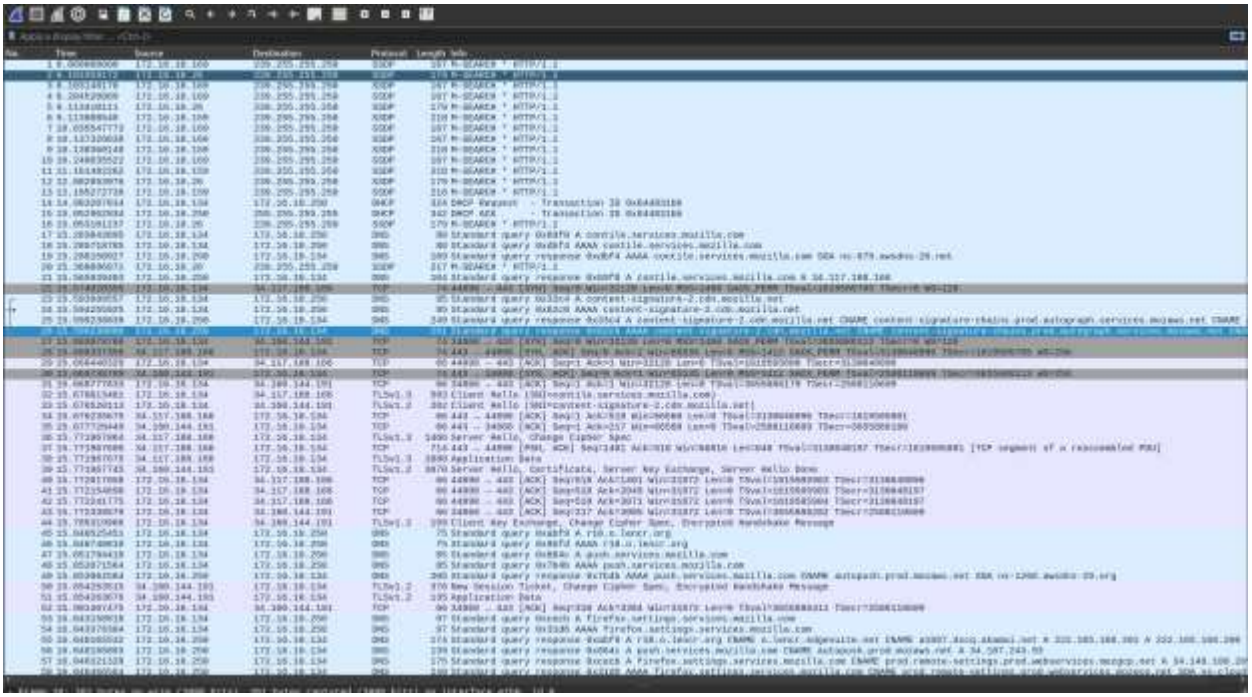
4. You need to clearly identify the message sequence and protocols used (including transport layer protocols) before your browser sends the first HTTP GET message to the relevant web server.



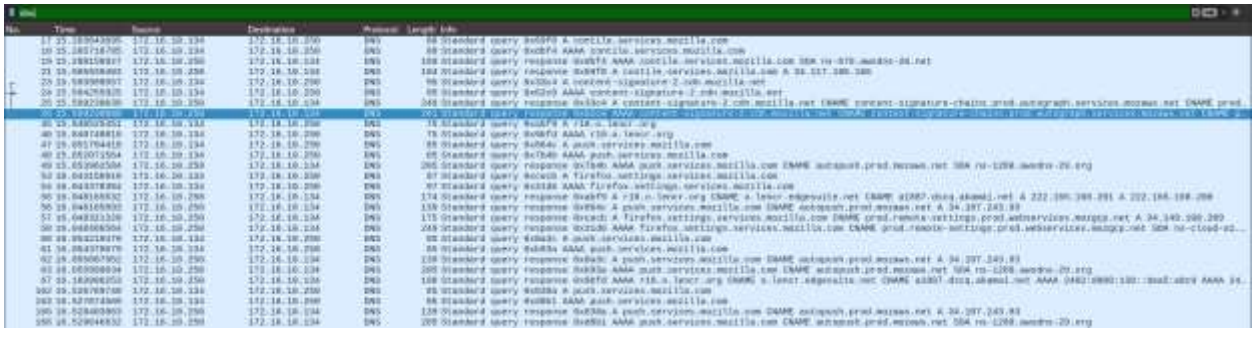
As you can see in the previous page by filtering with the keyword “tls.handshake” we can witness the entire handshake process of TLS. Its sequence is as follows,

- Client Hello
- Server Hello
- Certificate
- Server Key Exchange
- Server Hello Done
- Client Key Exchange
- Change Cipher Spec
- Encrypted Handshake Message
- Handshake

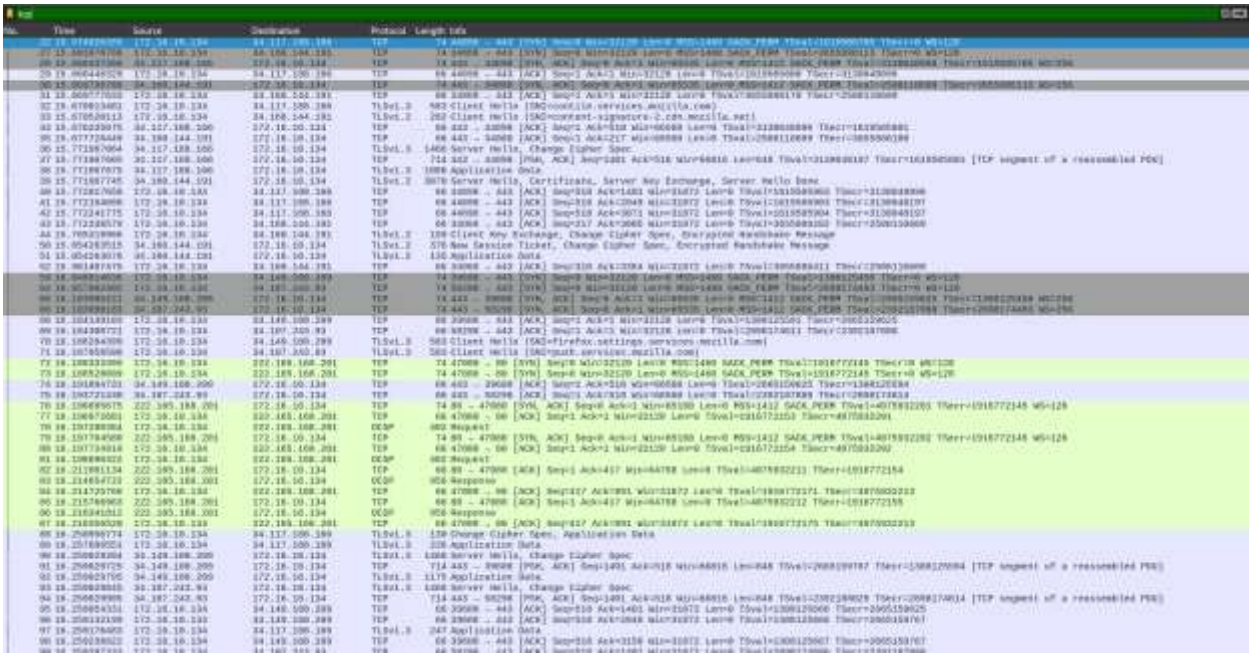
Below are the results from traffic captured before opening a link in a web browser.



If we filter by DNS protocol,



We can see the query and response packets between the browser and the web server.



By filtering using the TCP protocol as seen above, we can notice the three-way handshake, which establishes a TCP connection between your browser and the web server.

We can see the ‘SYN’ (which is sent from the client browser to initiate a connection), ‘SYN-ACK’ (which is sent from the server to confirm the ‘SYN’ packet and finally the ‘ACK’ sent from the client confirming the ‘SYN-Ack’ packet.

5. Can you analyse HTTPS in Wireshark? Explain your answer. If yes, provide evidence on how we can do that. If not, is there any alternative method we could use to analyse HTTPS?

By default, Wireshark cannot analyse HTTPS due to its encryption but you can display the SSL and TLS packets and decrypt them to monitor HTTPS traffic.