



Answer Sheet
Assignment – A02

Web Server and HTTP Message Inspection

Name : Alvaro Austin
Student ID : 2106752180

Preliminary Task

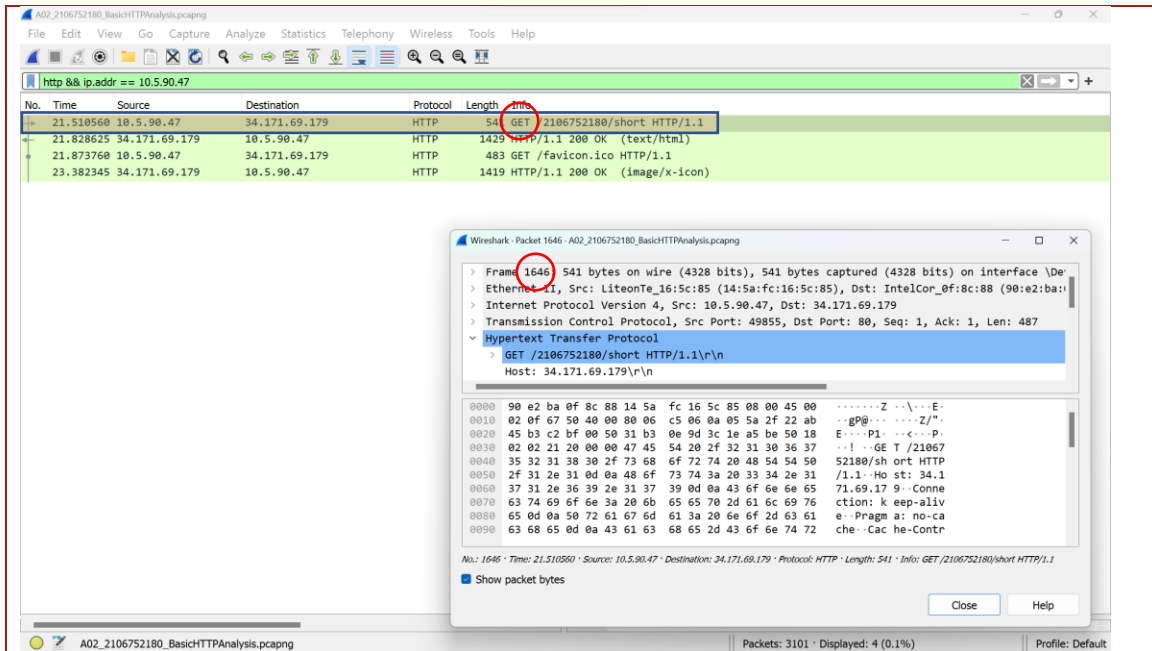
Task	Screenshot																		
Local computer private IP address	<pre>C:\Users\notal>ipconfig Windows IP Configuration Ethernet adapter vEthernet (WSL): Connection-specific DNS Suffix . : Link-local IPv6 Address : fe80::cf93:8715:65ec:9a64%54 IPv4 Address. : 172.27.16.1 Subnet Mask : 255.255.240.0 Default Gateway : Ethernet adapter Ethernet: Connection-specific DNS Suffix . : Link-local IPv6 Address : fe80::2788:71bd:5367:cbe3%9 Autoconfiguration IPv4 Address. . : 169.254.59.232 Subnet Mask : 255.255.0.0 Default Gateway : Wireless LAN adapter Local Area Connection* 1: Media State : Media disconnected Connection-specific DNS Suffix . : Wireless LAN adapter Local Area Connection* 2: Media State : Media disconnected Wireless LAN adapter Local Area Connection* 2: Media State : Media disconnected Connection-specific DNS Suffix . : Wireless LAN adapter Wi-Fi: Connection-specific DNS Suffix . : ui.ac.id Link-local IPv6 Address : fe80::4d39:a525:c40f:2b06%19 IPv4 Address. : 10.5.90.47 Subnet Mask : 255.255.248.0 Default Gateway : 10.5.88.1 C:\Users\notal></pre>																		
GCP instance External IP address	<table><tr><th><input type="checkbox"/></th><th>Status</th><th>Name ↑</th><th>Zone</th><th>Recommendations</th><th>In use by</th><th>Internal IP</th><th>External IP</th><th>Connect</th></tr><tr><td><input type="checkbox"/></td><td>✔</td><td>vm-1-alvaro-2106752180</td><td>us-central1-a</td><td></td><td></td><td>10.128.0.3 (nic0)</td><td>34.171.69.179 (nic0)</td><td>SSH ▾ ⋮</td></tr></table>	<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect	<input type="checkbox"/>	✔	vm-1-alvaro-2106752180	us-central1-a			10.128.0.3 (nic0)	34.171.69.179 (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect											
<input type="checkbox"/>	✔	vm-1-alvaro-2106752180	us-central1-a			10.128.0.3 (nic0)	34.171.69.179 (nic0)	SSH ▾ ⋮											

Notes: All answers must be supported by relevant screenshot(s). You **must highlight part of the screenshot** or **describe which part of the screenshot (fields, value, etc.) specifically** to support your answer. Answers with screenshots that have no highlight/description **will not be scored**.

[37 Points] Basic HTTP Analysis

- [7] Find the frame that contains an HTTP request. Which machine acts as the source of this message (is it your local machine or your web server)? Explain how you come to that answer from the information available.

Screenshot:



Explanation:

Question:

1. Find the frame that contains an HTTP request.

Answer: Frame 1646, as you can see the one that is circled in red. It is the frame that contains a GET request.

2. Which machine acts as the source of this message (local/webserver)

Answer: the machine that acts as the source of this message is **my local machine**. Reason:

- My private IP address is 10.5.90.47
- Web server IP address is 34.171.69.179

As you can see the source of GET request is on IP address of 10.5.90.47 which means my private IP address from my local machine.

3. Explain how I come up with that logic.

Answer: There are 4 HTTP protocol from my wireshark. You can see that out of 4 protocol, there are only 2 request happened to my webserver (34.171.69.179). However, out of 2 request, the first request is the only request that was responded by the server in **text/html** (message), so favicon.ico (it's a request that I marked as a pretty irrelevant information in this case. Hence I can infer through that reasoning, that the first request is the source of the message.

2. [6] Which HTTP version is used to make this request? What about the response's HTTP version? Is it the same?

Screenshot:

Wireshark packet capture showing HTTP traffic. The filter is 'http && ip.addr == 10.5.90.47'. The table below shows the captured packets:

No.	Time	Source	Destination	Protocol	Length	Info
21.510560	10.5.90.47	34.171.69.179	10.5.90.47	HTTP	541	GET /2106752180/short HTTP/1.1
21.828625	34.171.69.179	10.5.90.47	10.5.90.47	HTTP	1429	HTTP/1.1 200 OK (text/html)
21.873760	10.5.90.47	34.171.69.179	10.5.90.47	HTTP	483	GET /favicon.ico HTTP/1.1
23.382345	34.171.69.179	10.5.90.47	10.5.90.47	HTTP	1419	HTTP/1.1 200 OK (image/x-icon)

Explanation:

I will provide the important things to know:

- Red Circle: denote the request
- Blue Circle: denote the response

As you can see both the red and blue circle points at HTTP/1.1. So HTTP version for the request is HTTP/1.1 and for the response is HTTP/1.1 as well. Both request and response has the same HTTP version, which is HTTP/1.1

3. [6] What is the HTTP request type used? Is this request considered safe (no side effect)?

Screenshot:

Wireshark packet capture showing HTTP traffic. The filter is 'http && ip.addr == 10.5.90.47'. The table below shows the captured packets:

No.	Time	Source	Destination	Protocol	Length	Info
21.510560	10.5.90.47	34.171.69.179	10.5.90.47	HTTP	541	GET /2106752180/short HTTP/1.1
21.828625	34.171.69.179	10.5.90.47	10.5.90.47	HTTP	1429	HTTP/1.1 200 OK (text/html)
21.873760	10.5.90.47	34.171.69.179	10.5.90.47	HTTP	483	GET /favicon.ico HTTP/1.1
23.382345	34.171.69.179	10.5.90.47	10.5.90.47	HTTP	1419	HTTP/1.1 200 OK (image/x-icon)

Packets: 3101 · Displayed: 4 (0.1%)

Explanation:

The request type that is used in both for requesting data from webserver and favicon.ico is GET request. But the request that we truly focus on is the first request.

GET request generally doesn't pose a side effect and is considered safe. Request is considered safe when they didn't modify any data on the server. GET request according to HTTP/1.1 specification shouldn't cause side effect because they shouldn't cause any changes to the state of the server. However GET request could pose a security threat because when we transmitted data through GET request, the information will be contained in URL parameters hence could cause a security threat.

In conclusion, GET request is considered safe because if used correctly in HTTP/1.1 it will not modify data and change the state of the server.

4. [6] What language/locales does your local machine accept? What language does your local machine prefer among the acceptable languages (if any)?

Screenshot:

The screenshot shows a Wireshark packet capture of an HTTP GET request. The packet list at the top shows a GET request to /2106752180/short. The packet details pane shows the request headers, with 'Accept-Language: en-US,en;q=0.9' circled in red. The packet bytes pane shows the raw data.

```
Host: 34.171.69.179\r\n
Connection: keep-alive\r\n
Pragma: no-cache\r\n
Cache-Control: no-cache\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,ima
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9\r\n
\r\n
[Full request URI: http://34.171.69.179/2106752180/short]
Full request URI: http://34.171.69.179/2106752180/short HTTP
```

Explanation:

Q1: What language does my local machine accept:

A1: Based on the screenshot and my analysis through the details of request sent to the server, my local machine accepts and expects a language code of **en-US, en**. Those language code means:

- **en-US: English used in the United States**
- **en: English language in general, without any specific regional or national variant.**

In conclusion, my local machine accepts English language, which is divided into 2 parts: English used in US and English in general.

Q2: What language does my local machine prefer among acceptable language (en-US or en, code based)

A2:

Note: According to the HTTP/1.1 specification (RFC 7231), if a quality value is not specified explicitly for a language, it defaults to q=1.0.

Based on note above, You can infer that en-US has a default q value of = 1.0 and en has a default q value of 0.9 based on the screenshot above.

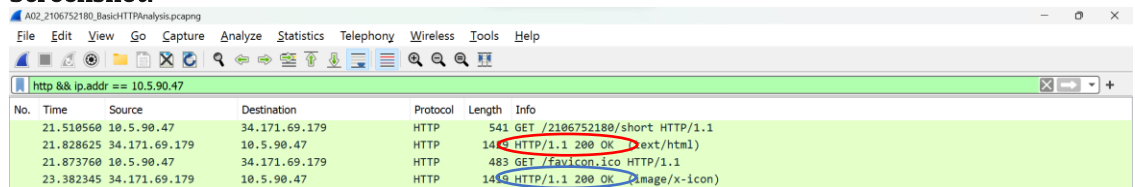
q, in this case refers to relative quality of each language preference from 0 – 1 (highest).

Based on all this information, we can conclude that my local machine preferred **en-US** over any language in this case **en**. Because **en-US** has q value of 1.0 compared to **en** with q value of 0.9.

To conclude, **en-US** or English used in United States is the language that my machine preferred over any other acceptable language (**en** or english).

5. [6] What is the status code and phrase from the HTTP response in the packet found? What is the meaning of that status code and phrase?

Screenshot:



The screenshot shows a Wireshark packet capture window with the title 'A02_2106752180_BasicHTTPAnalysis.pcapng'. The packet list pane shows four packets, all of which are HTTP responses with status code 200 OK. The first packet is a GET request for '/2106752180/short' with a content type of 'text/html'. The second packet is a GET request for '/favicon.ico' with a content type of 'image/x-icon'. The third and fourth packets are also GET requests for '/favicon.ico' with a content type of 'image/x-icon'. The packet details pane for the first packet shows the status code 200 and the phrase 'OK'.

No.	Time	Source	Destination	Protocol	Length	Info
21.510560	10.5.90.47	34.171.69.179	10.5.90.47	HTTP	541	GET /2106752180/short HTTP/1.1
21.828625	34.171.69.179	10.5.90.47	10.5.90.47	HTTP	146	HTTP/1.1 200 OK (text/html)
21.873760	10.5.90.47	34.171.69.179	10.5.90.47	HTTP	483	GET /favicon.ico HTTP/1.1
23.382345	34.171.69.179	10.5.90.47	10.5.90.47	HTTP	146	HTTP/1.1 200 OK (image/x-icon)

Explanation:

As you can see from the screenshot, both request was responded from the webserver as status code of 200 and phrase OK.

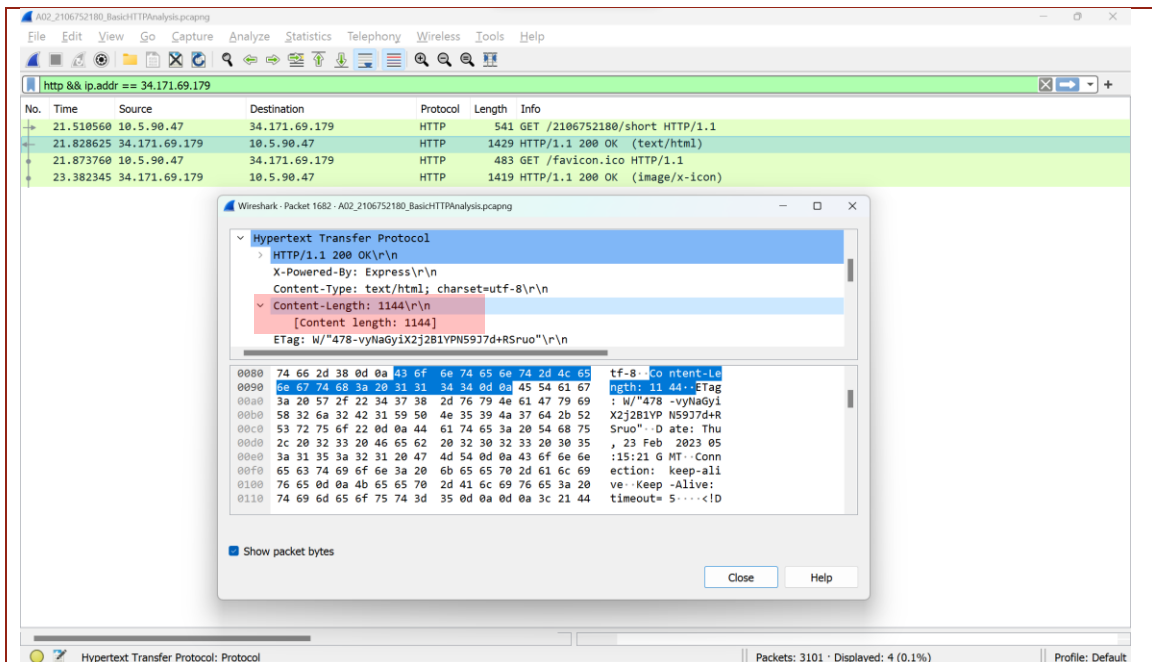
The meaning of status code **200** is the request has **succeeded** (basically means that the request has successfully transferred).

The meaning of phrase **OK** is to provides additional context and confirms that the request was proceeded without any errors .

In conclusion, the status code and phrase is **200** and **OK** respectively. The meaning of them as mentioned above is “The request has succeed/proceed without error” and “Additional context for the status code” respectively. Both main request and favicon request also gives different output which text/html and image/x-icon.

6. [6] What is the size of the HTTP response message body to the /short request?

Screenshot:



Explanation:

There are 4 HTTP protocol sent, out of 4 HTTP protocol, there are only 2 response sent by my webserver, which to /short or /favicon.ico. In the question mentioned /short, so you could find details on /short response. To find out about the size of the HTTP response message body is to see the content length (in bytes) in header field. In this case the **size of my HTTP response is 1144 bytes**. You could see it being highlighted in red.

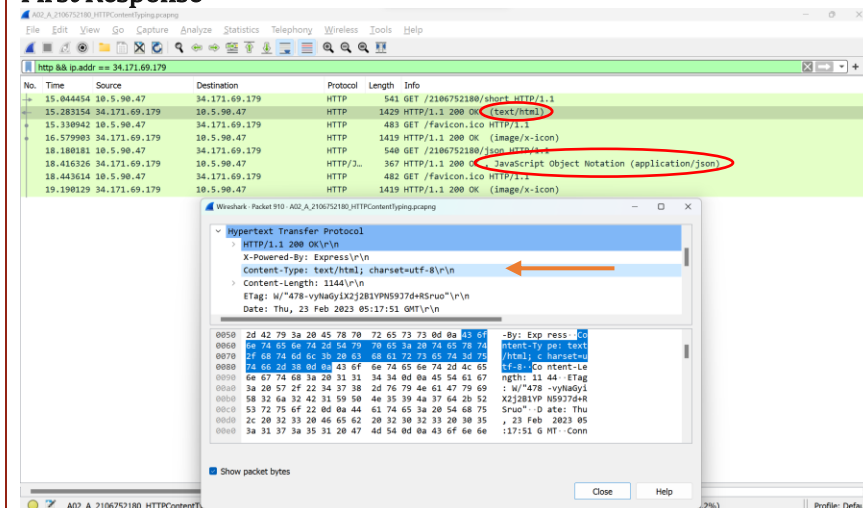
[14 Points] HTTP Content Typing

1. [7] Compare the HTTP response from the two different requests. You may notice that your browser may render the response differently. How does your browser know about the content type it received? (Hint: check the HTTP response).

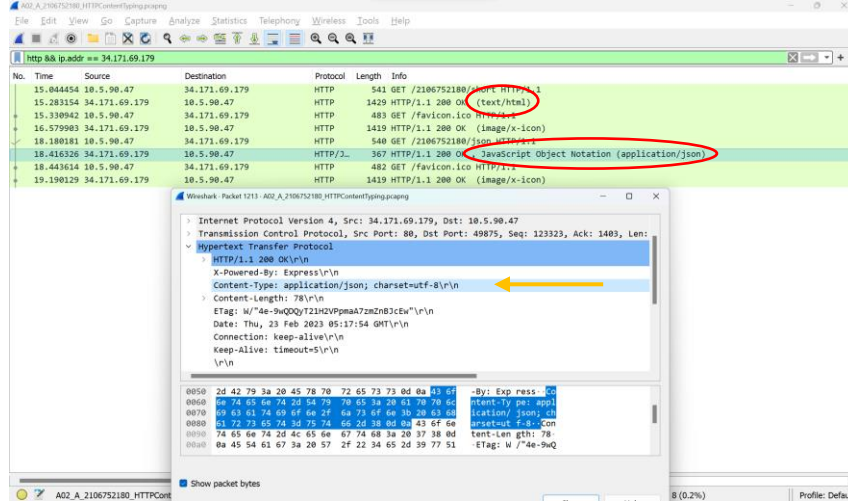
Screenshot:

Response Screenshot:

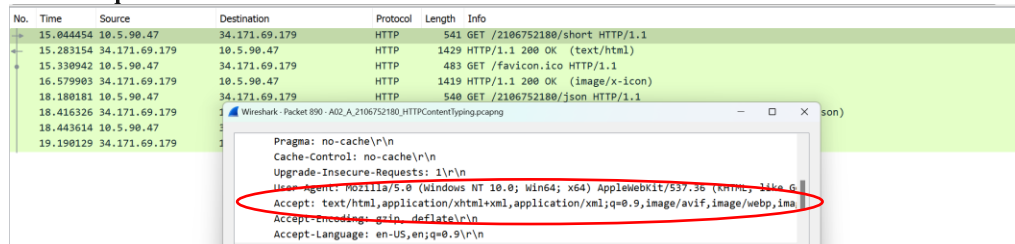
First Response



Third Response



Request Screenshot: First Request



Explanation:

When comparing between 2 different screenshot (the difference lies on the details of each response), we could see that there are difference on rendering the response. In this case, we will look up at first response (text/html) and third response(application/json).

The browser knows how to render because when the browser make a request for a resource, they include a HTTP header that specifies the file type, that is called “Accept” header.

You could see in the screenshot above (the **request screenshot**), they have a header called “**Accept**” that tells what type of file the browser able to render.

Note: Below are the main reason, above only for supporting

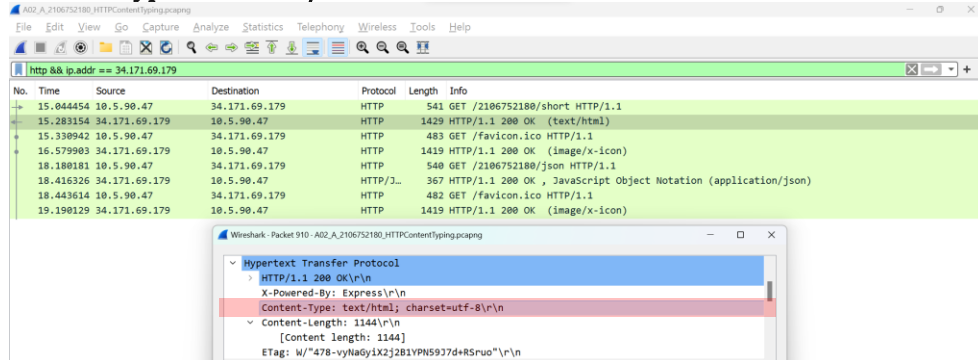
Then when server responds to the request, it includes HTTP response header that specifies the type. The header is called **Content-Type**. Through content-type, the browser are able to display/render the information it provides.

In conclusion, because the **Content-Type** that allows browser to render based on the value.
In the screenshot it was highlighted with arrows surrounding the text.

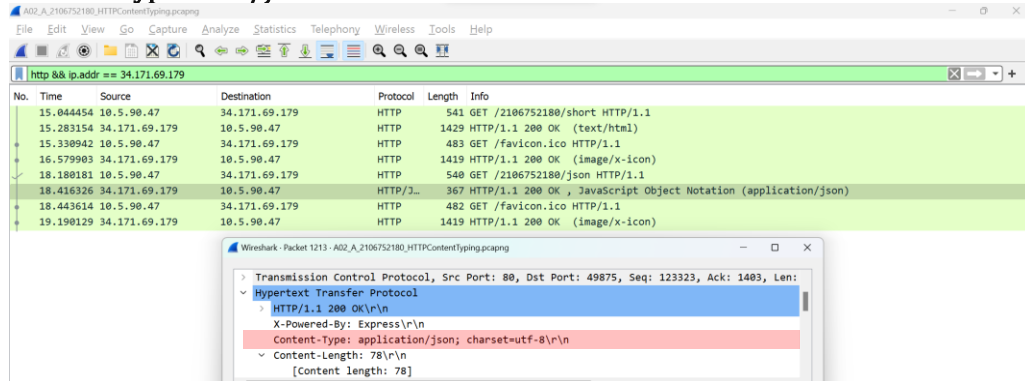
2. [7] What is the content type from the URL `/short`? What about the content type from the URL `/json`?

Screenshot:

Content-Type from URL `/short`



Content-Type from `/json`



Explanation:

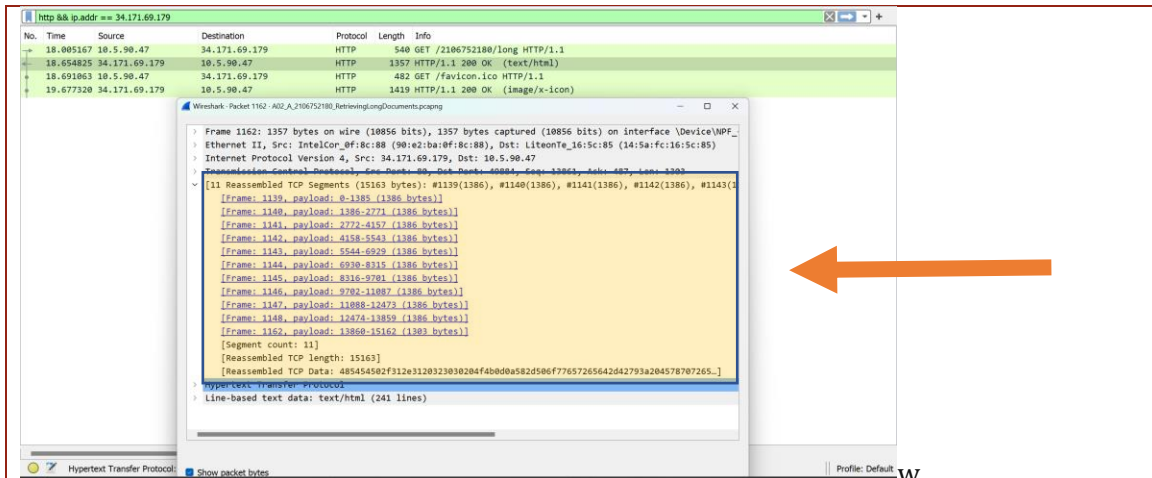
The Content-Type of URL `/short` could be seen in the response of the request from `"/short"`. In this case, Content-Type of URL `/short` is **text/html; charset=utf-8**.

The content-type of URL `/json` could be seen in the response of the request from `"/json"`. In this case, Content-Type of URL `/json` is **application/json; charset=utf-8**.

[14 Points] Retrieving Long Documents

1. [7] Can the entire HTTP response message fit in one frame? If yes, which frame (mention the frame number) contains the whole response? If not, mention **all** frame numbers that participates in bringing the HTTP response.

Screenshot:



Explanation:

No in the case of requesting data to /long, entire HTTP response message cannot fit in one frame. As you can see from the screenshot, there needs to be 11 frames to bringing the whole HTTP response.

All of those frame numbers are:

1. Frame 1139, payload: 0 – 1385
2. Frame 1140, payload: 1386-2771
3. Frame 1141, payload: 2772 – 4157
4. Frame 1142, payload: 4158 – 5543
5. Frame 1143, payload: 5544 – 6929
6. Frame 1144, payload: 6930 – 8315
7. Frame 1145, payload: 8316 – 9701
8. Frame 1146, payload: 9702 – 11087
9. Frame 1147, payload: 11088 – 12473
10. Frame 1148, payload: 12474 – 13859
11. Frame 1162, payload: 13860 – 15162

It needs to be divided into 11 frames with each frames have the maximum of 1386 bytes.

3. [7] Which frame contains the header of the HTTP response? (You may need to inspect the raw contents from the frame).

Screenshot:

Wireshark · Packet 1162 · A02_A_2106752180_RetrievingLongDocuments.pcapng

> [Timestamps]
 > [SEQ/ACK analysis]
 TCP payload (1303 bytes)
 TCP segment data (1303 bytes)

▼ [11 Reassembled TCP Segments (15163 bytes): #1139(1386), #1140(1386), #1141(1386), #1142(1386), #1143(1386), #1144(1386)]

[Frame: 1139, payload: 0-1385 (1386 bytes)] ←
 [Frame: 1140, payload: 1386-2771 (1386 bytes)]
 [Frame: 1141, payload: 2772-4157 (1386 bytes)]
 [Frame: 1142, payload: 4158-5543 (1386 bytes)]
 [Frame: 1143, payload: 5544-6929 (1386 bytes)]
 [Frame: 1144, payload: 6930-8315 (1386 bytes)]
 [Frame: 1145, payload: 8316-9701 (1386 bytes)]
 [Frame: 1146, payload: 9702-11087 (1386 bytes)]
 [Frame: 1147, payload: 11088-12473 (1386 bytes)]

0000 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK
 0010 0a 58 2d 50 6f 77 65 72 65 64 2d 42 79 3a 20 45 .X-Powered-By: E
 0020 78 70 72 65 73 73 0d 0a 43 6f 6e 74 65 6e 74 2d xpress- Content-
 0030 54 79 70 65 3a 20 74 65 78 74 2f 68 74 6d 6c 3b Type: text/html;
 0040 20 63 68 61 72 73 65 74 3d 75 74 66 2d 38 0d 0a charset=utf-8
 0050 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 Content- Length:
 0060 31 34 39 33 30 0d 0a 45 54 61 67 3a 20 57 2f 22 14930- E Tag: W/"
 0070 33 61 35 32 2d 56 57 42 61 68 51 31 36 73 64 45 3a52-VwB ahQ16sdE
 0080 4e 77 30 74 54 57 45 42 35 7a 6e 42 6f 78 35 4d Nw0tTWEB 5znBoxSM
 0090 22 0d 0a 44 61 74 65 3a 20 54 68 75 2c 20 32 33 ".Date: Thu, 23
 00a0 20 46 65 62 20 32 30 32 33 20 30 35 3a 31 39 3a Feb 2023 05:19:
 00b0 30 37 20 47 4d 54 0d 0a 43 6f 6e 6e 65 63 74 69 07 GMT- Connecti
 00c0 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d 0a on: keep-alive-
 00d0 4b 65 65 70 2d 41 6c 69 76 65 3a 20 74 69 6d 65 Keep-Alive: time
 00e0 6f 75 74 3d 35 0d 0a 0d 0a 3c 21 44 4f 43 54 59 out=5- <!DOCTYPE
 00f0 50 45 20 68 74 6d 6c 3e 0a 3c 68 74 6d 6c 20 6c PE html> <html l
 0100 61 6e 67 3d 22 65 6e 22 3e 0a 20 20 3c 68 65 61 ang="en" >> <hea
 0110 64 3e 0a 20 20 20 20 3c 74 69 74 6c 65 3e 4c 6f d> < title>Lo
 0120 6e 67 20 7c 20 43 53 55 49 20 43 6f 6d 70 4e 65 ng | CSU I CompNe

Frame (1357 bytes) Reassembled TCP (15163 bytes)

No.: 1162 · Time: 18.654825 · Source: 34.171.69.179 · Destination: 10.5.90.47 · Protocol: HTTP · Length: 1357 · Info: HTTP/1.1 200 OK (text/html)

Show packet bytes

Explanation:

Frame 1139 contains the header of HTTP response. We got this detail from raw contents of each frame marked by the red box in screenshot. On there, you can see that there is Header such as Content-Type, etc.

[14 Points] Content with Embedded Objects

- [7] How many HTTP requests are made from the source? What is the point of each request (what content is being requested from each request)?

Screenshot:

No.	Time	Source	Destination	Protocol	Length	Info
14..	19.561880	10.5.90.47	34.171.69.179	HTTP	542	GET /2106752180/images HTTP/1.1
15..	19.979796	34.171.69.179	10.5.90.47	HTTP	196	HTTP/1.1 200 OK (text/html)
15..	19.994141	10.5.90.47	34.171.69.179	HTTP	510	GET /images/llana-B05sm7QMH_U-unsplash.jpg HTTP/1.1
15..	19.995831	10.5.90.47	34.171.69.179	HTTP	516	GET /images/asher-zhang-87J_sPcUaNA-unsplash.jpg HTTP/1.1
16..	20.391731	10.5.90.47	34.171.69.179	HTTP	517	GET /images/nicola-powys-cGOVLxbYB2o-unsplash.jpg HTTP/1.1
36..	22.271598	34.171.69.179	10.5.90.47	HTTP	673	HTTP/1.1 200 OK (JPEG image)
37..	22.298992	34.171.69.179	10.5.90.47	HTTP	640	HTTP/1.1 200 OK (JPEG image)
37..	22.315054	34.171.69.179	10.5.90.47	HTTP	1305	HTTP/1.1 200 OK (JPEG image)
37..	22.322344	10.5.90.47	34.171.69.179	HTTP	484	GET /favicon.ico HTTP/1.1
38..	22.568039	34.171.69.179	10.5.90.47	HTTP	1419	HTTP/1.1 200 OK (image/x-icon)

Explanation:

There are 5 request that are made from the source. In the screenshot above, the color teal determines the request that are made from the source. Those 5 requests are:

1. GET Request for HTML file (index.html) that were rendered the first time we loaded the server. The server will give a response in **text/html** to tell the browser that the content should be treated as HTML and displayed as webpage.
2. GET Request for JPEG or JFIF image for **image/ilana.....** It shouldn't be combined with text/html because it typically larger in size and take longer to load. Separating them could speedup the loading time of a webpage. It basically will render the first image
3. GET Request for JPEG or JFIF image for **image/asher.....** It shouldn't be combined with text/html because it typically larger in size and take longer to load. Separating them could speedup the loading time of a webpage. It basically will render the second image
4. GET Request for JPEG or JFIF image for **image/nicola.....** It shouldn't be combined with text/html because it typically larger in size and take longer to load. Separating them could speedup the loading time of a webpage. It basically will render the third image
5. GET Request for favicon.ico, this is for user to look for favicon.ico file. Favicon.ico file is used to show small icon (basically logo, like youtube logo, scele logo, etc) in address bar next to website title (name).

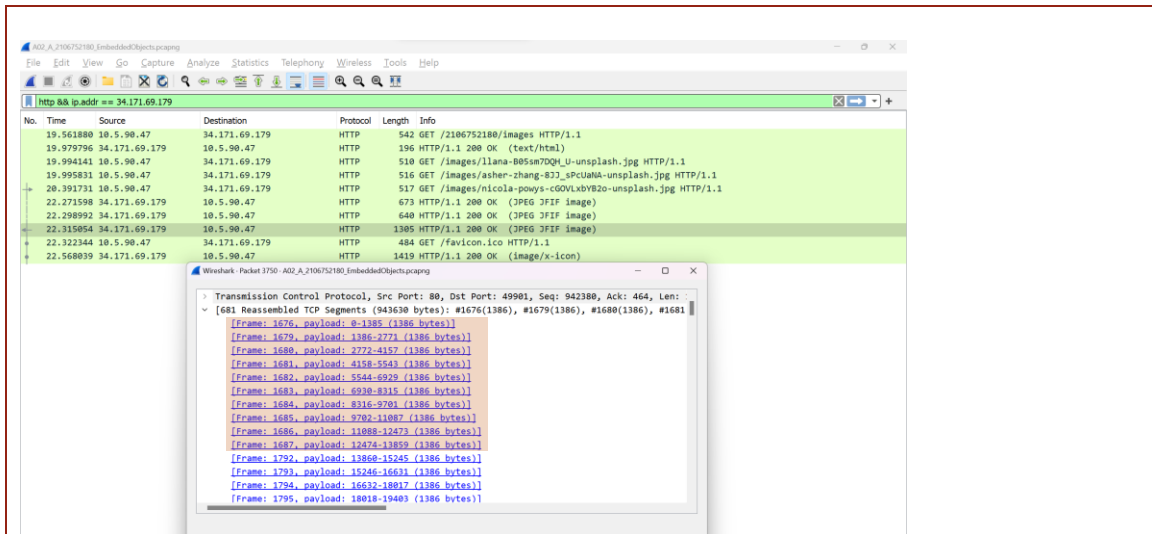
2. [7] Among the images, can you determine whether the images were downloaded serially or parallelly? Explain!

Screenshot:

The screenshot displays a Wireshark packet capture of an HTTP session. The packet list shows the following requests:

No.	Time	Source	Destination	Protocol	Length	Info
19	561880	10.5.90.47	34.171.69.179	HTTP	542	GET /2106752180/images HTTP/1.1
19	979796	34.171.69.179	10.5.90.47	HTTP	196	HTTP/1.1 200 OK (text/html)
19	994141	10.5.90.47	34.171.69.179	HTTP	510	GET /images/ilana-805sm7DQH_U-unsplash.jpg HTTP/1.1
19	995831	10.5.90.47	34.171.69.179	HTTP	516	GET /images/asher-zhang-873_sPcuuNA-unsplash.jpg HTTP/1.1
20	391731	10.5.90.47	34.171.69.179	HTTP	517	GET /images/nicola-powys-G0VLvWR2o-unsplash.jpg HTTP/1.1
22	271598	34.171.69.179	10.5.90.47	HTTP	673	HTTP/1.1 200 OK (JPEG JFIF image)
22	298992	34.171.69.179	10.5.90.47	HTTP	640	HTTP/1.1 200 OK (JPEG JFIF image)
22	315054	34.171.69.179	10.5.90.47	HTTP	1305	HTTP/1.1 200 OK (JPEG JFIF image)
22	322344	10.5.90.47	34.171.69.179	HTTP	484	GET /favicon.ico HTTP/1.1
22	568039	34.171.69.179	10.5.90.47	HTTP	1305	HTTP/1.1 200 OK (JPEG JFIF image)

The packet details pane shows the raw bytes of the image responses, indicating they are downloaded serially. The raw bytes are displayed in a hex dump format, showing the structure of the JPEG files.



Explanation:

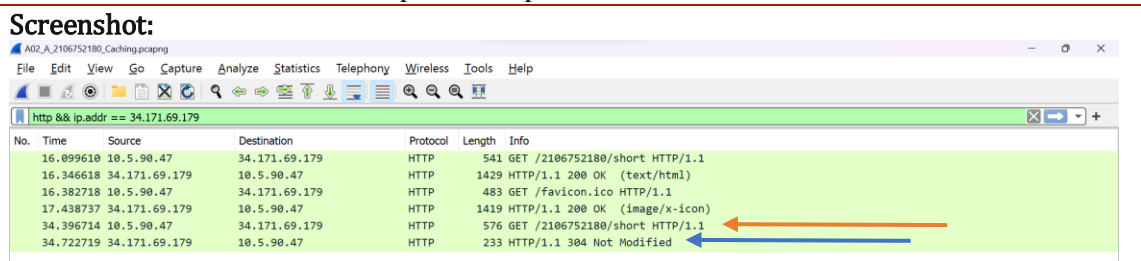
Yes, these images were downloaded parallelly. Based on screenshot above, the first screenshot shows the detail of frames for ilana (first image) response. As you can see on that first screenshot, There is a leap between frames from 1673 to 1704. On second screenshot shows detail of frames asher-zhang (second image) response. As you can see, there was frames between 1673 and 1704 which was 1674 and 1675 (highlighted in blue). On third screenshot shows detail of frames nicola-pows response. As you can see, there was frames between 1675 (from second screenshot) – 1687.

From all of this we can conclude that, 3 images were downloaded parallelly because there were some frames that doesn't finished all it's job but already occupied by another image frames. This shows that all of the image were trying to be downloaded (so it could be rendered) parallelly.

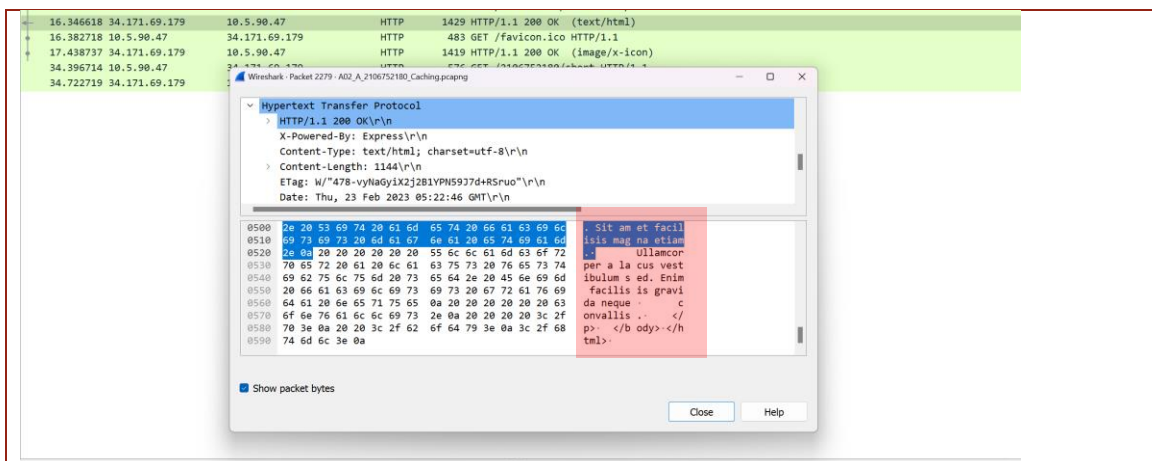
In conclusion, this images were downloaded parallelly because in serial each image would have to wait for another image TCP Segment to be finished to start their own TCP segments. Hence it was downloaded parallelly.

[21 Points] Persistent HTTP (Caching)

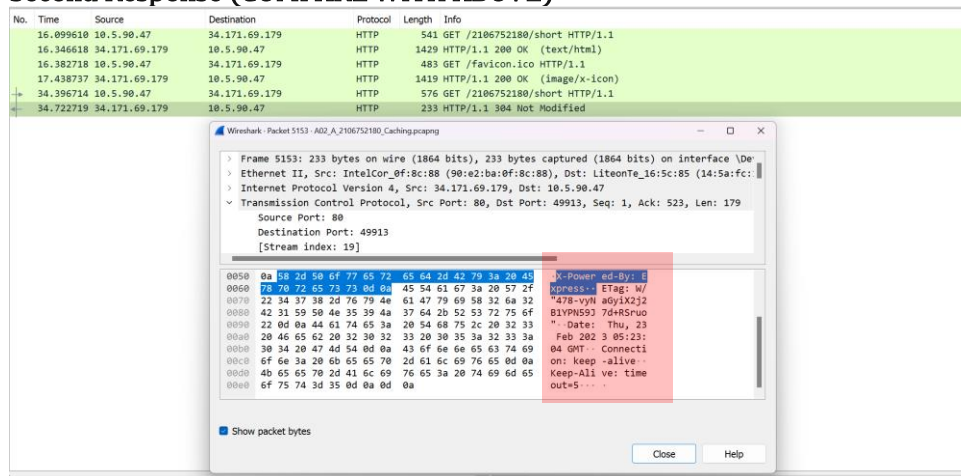
- [7] What is the status code and phrase from the **second** HTTP response? Is a payload sent from the server in that HTTP response? Explain!



First Response



Second Response (COMPARE WITH ABOVE)



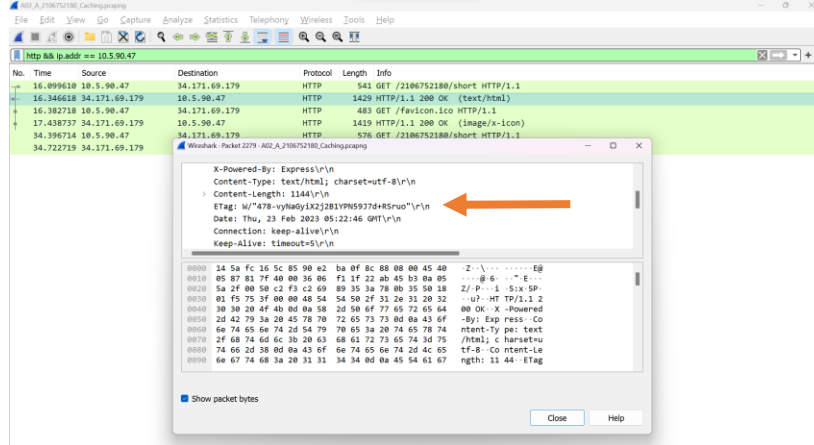
Explanation:

Based on teaching assistant answer, favicon.ico HTTP request **will be ignored**. If we take that to account then **second HTTP request** should be the one with **orange arrow pointing at it**. Hence, the **second response** could be seen as the **blue arrow** pointing at it. The status code and phrase of the second response is 304 and “Not Modified” respectively. This response happened because when we refresh a webpage, browser send HTTP request (GET) to the server to get the latest version of the page. If there was no modified then the server doesn’t have to download the entire page again.

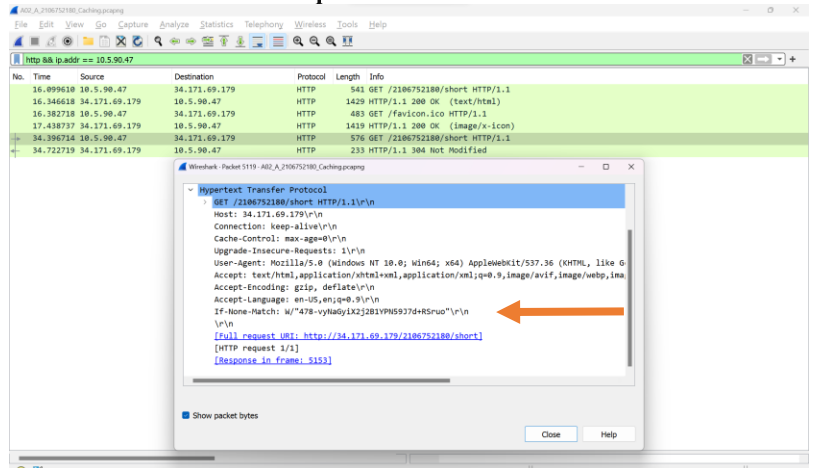
There was no payload sent from the server. In the screenshot above, I gave a comparison between first response and second response. In the first response it returns text/html but in second response there was no text/html in the package details (both marked in red highlight). This is also backed by there were **no Content-Type and Content-Length header** in second response.

- [7] There are a few ways that client and server can negotiate about caching. Which field in the **second** HTTP request and response are compared to decide whether the server should send a payload?

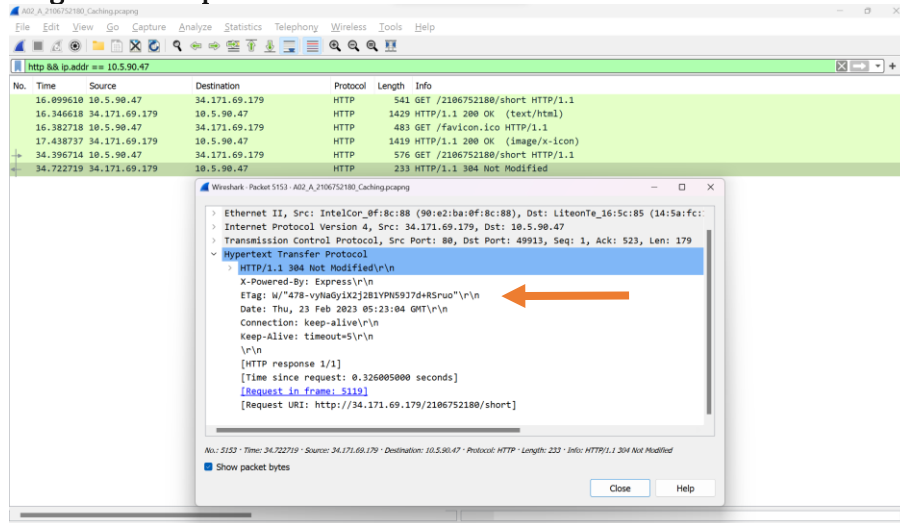
Screenshot: Etag first response:



If None match second request:



Etag second response:



Explanation:

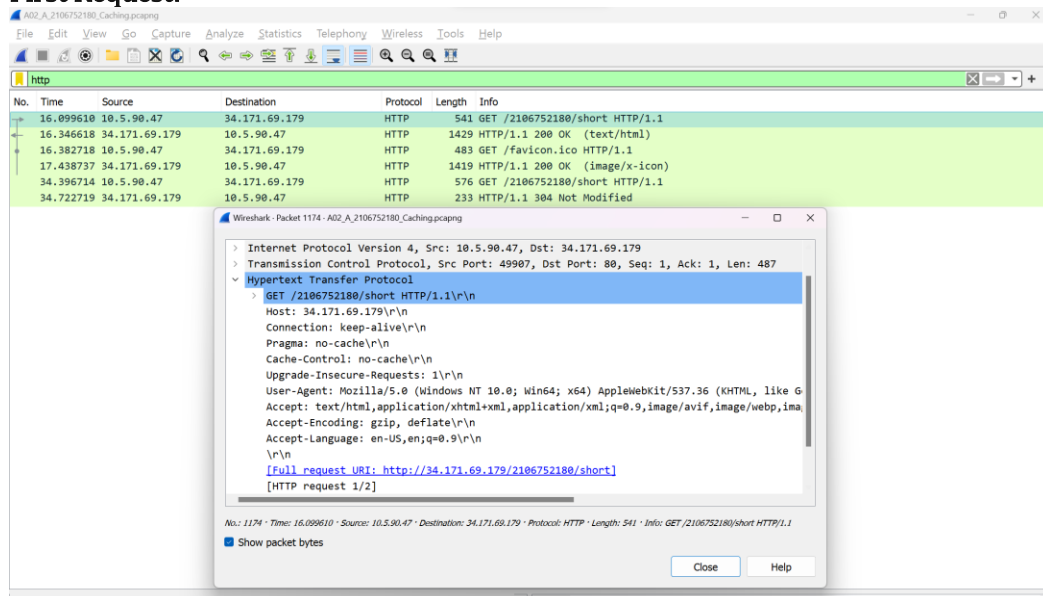
Field in second HTTP request and response that could be used to decide whether the server should send a payload is **ETag** and **If None Match** Header (highlighted by orange arrow). The "If-None-Match" header is an HTTP request header that is used to check if the content of a requested resource has been changed since the last time it was accessed. The server checks previous ETag value against the current ETag value for the requested resource. If the two values match means that the content of the resource has not changed. That's why the server can decide to sent the payload or not.

Worth mention, in the second response, there were also no content-type or content-length. This is probably the cause of no payload.

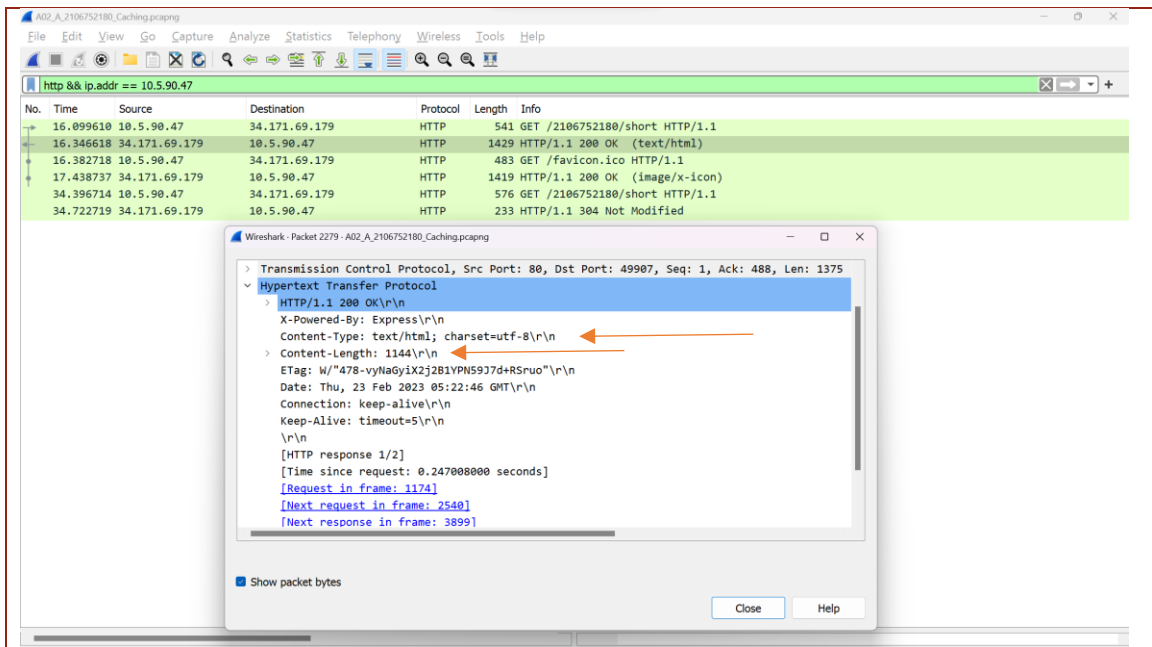
- [7] Where do the value of the attribute from the second HTTP request that you have identified in number 2 come from? Specifically, which attribute becomes the reference to fill that field?

Screenshot:

First Request:



First Response:



Explanation:

The value of ETag attribute from second HTTP request comes from the first request response of **Content-Type** and **Content-Length**. These two become reference of the ETag value then encrypted. The screenshot above shows first request and response. As you can see in the first response, there were **Content-Type** and **Content-Length** header. This is backed because in the first request, there were no ETag value hence after first request, the response shows ETag from **Content-Type** and **Content-Length** as references.