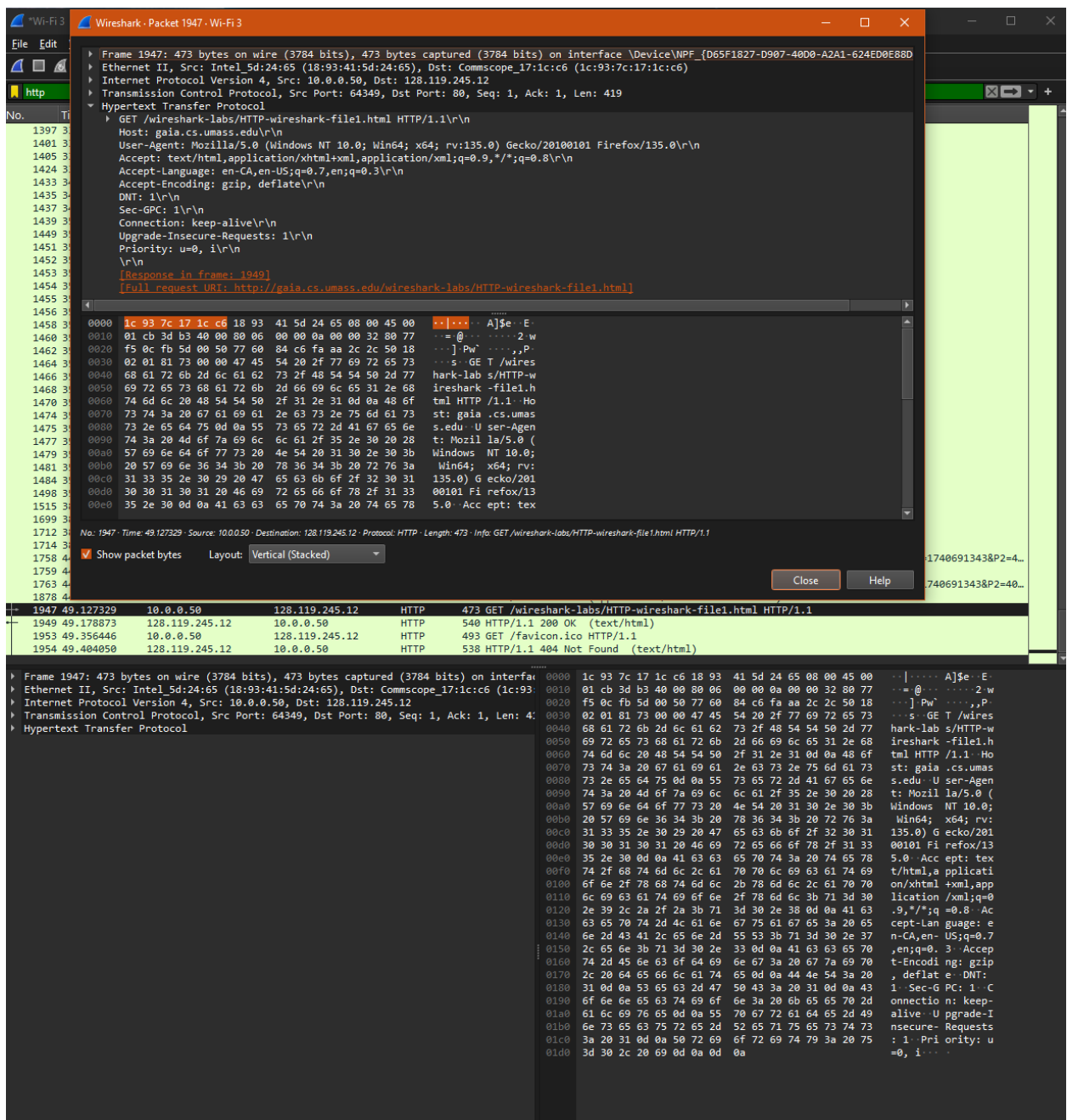


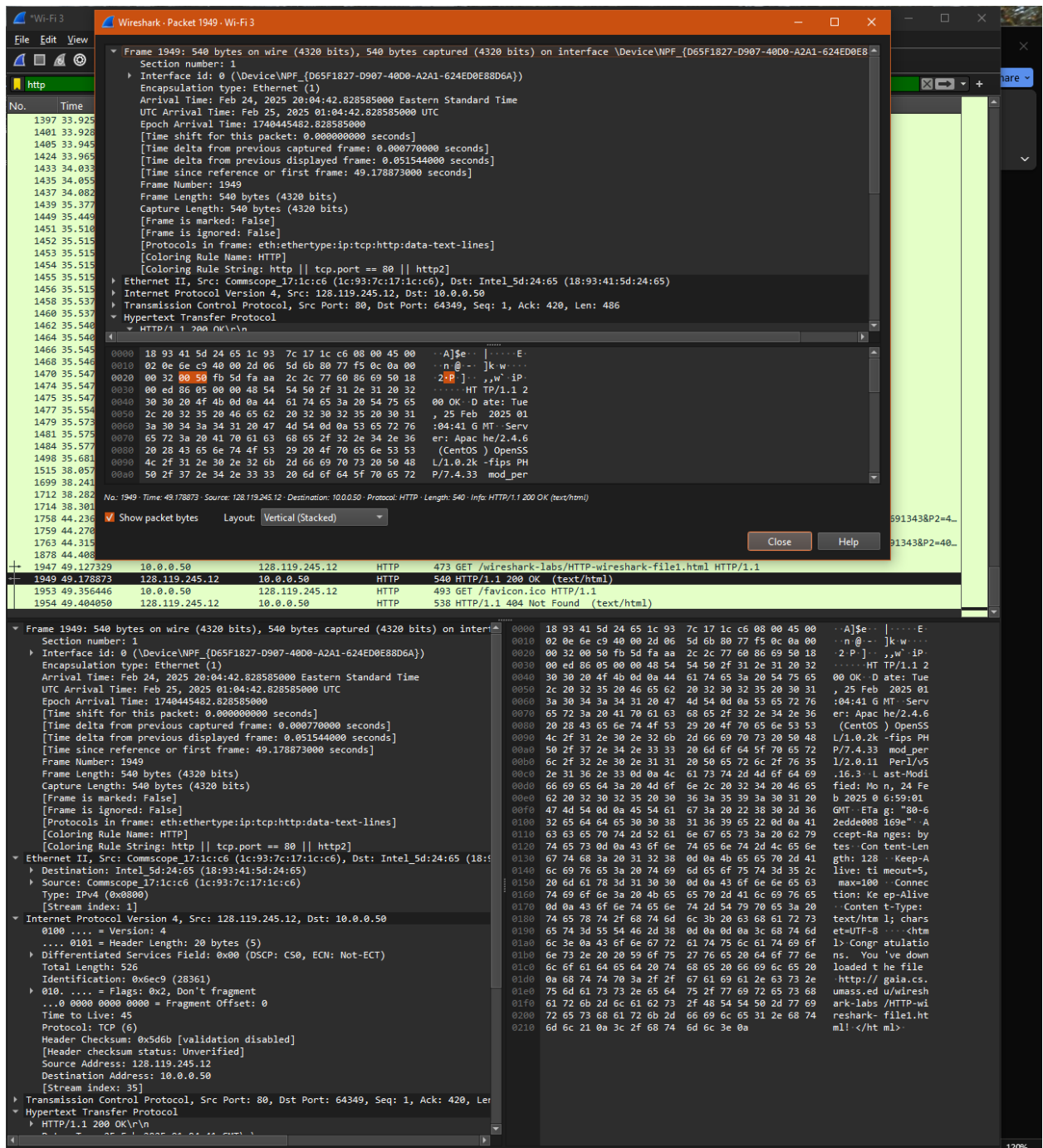
2025-03-11

## Task 1

1.

[illegible]





It is important to access the HTTP version and not the HTTPS version because we're using Wireshark to capture and analyze HTTP messages. Wireshark monitors the network traffic where HTTP traffic is transmitted in plaintext. This process makes it easy for Wireshark to

capture and display the GET request and the HTTP response along with other headers and messages. HTTPS has a secure connection (SSL), which means that it is encrypted. Therefore, capturing traffic in HTTPS is futile since even if we can capture it, the packets won't be available for view. In everyday use, however, HTTPS is preferable because it encrypts the communication between the browser and the web server. Thus, all sensitive information such as payments or login credentials are protected. In HTTP connection it's known that anyone can monitor the network and can have a look at the data.

2. Our browser is running HTTP version 1.1. The server is running version 1.1.
3. Our browser indicates that it can accept Canadian and American languages.
4. 20:04:42.777041000 and 20:04:42.828585000  
RTT = 0.051544000 seconds or 51.544 milliseconds.
5. The status code returned is 200 OK.
6. Last modified : Mon, 24 Feb 2025 06:59:01 GMT.
7. Header missing is TCP segment data (#bytes) and ETag (identifier for specific version of resource)

## TASK 2

8. We only see it when not removing the cache, If-Modified-Since: Mon, 24 Feb 2025

06:59:01 GMT

The image shows a Wireshark packet capture of an HTTP GET request. The packet list pane on the left shows a single packet (No. 80) at time 17.590268, source 10.0.0.50, and destination 128.119.245.12. The packet details pane on the right shows the structure of the packet, including Ethernet II, Internet Protocol Version 4, and Hypertext Transfer Protocol. The packet bytes pane at the bottom shows the raw data of the packet, including the GET request line and headers.

**Packet Details:**

- Frame 80: 559 bytes on wire (4472 bits), 559 bytes captured (4472 bits) on interface \Device\NPF\_{D65F1827-D907-480B-A2A1-624ED0E8BD}
- Ethernet II, Src: Intel 5d:24:65 (18:93:41:5d:24:65), Dst: Commscope\_17:1c:c6 (1c:93:7c:17:1c:c6)
- Internet Protocol Version 4, Src: 10.0.0.50, Dst: 128.119.245.12
- Transmission Control Protocol, Src Port: 52973, Dst Port: 80, Seq: 1, Ack: 1, Len: 505
- Hypertext Transfer Protocol
  - GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
  - Host: gaia.cs.umass.edu
  - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101 Firefox/135.0
  - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8
  - Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
  - Accept-Encoding: gzip, deflate
  - DNT: 1
  - Sec-GPC: 1
  - Connection: keep-alive
  - Upgrade-Insecure-Requests: 1
  - If-Modified-Since: Mon, 24 Feb 2025 06:59:01 GMT
  - If-None-Match: "173-62edde0808eccc"
  - Priority: u=0, i=1

**Packet Bytes:**

0000 1c 93 7c 17 1c c6 18 93 41 5d 24 65 08 00 45 00 ..|....A]se..E..  
0010 02 21 3d c5 40 00 80 06 00 00 0a 00 00 32 80 77 ..!=@.....2w  
0020 f5 0c ce ed 00 50 4e bd 1b 2e aa a5 50 19 50 18 .....PN.....P.P.  
0030 02 01 81 c9 00 00 47 45 54 20 2f 77 69 72 65 73 .....GET /wires  
0040 68 61 72 6b 2d 6c 61 62 73 2f 48 54 54 50 2d 77 hark-lab s/HTTP-w  
0050 69 72 65 73 68 61 72 6b 2d 66 69 6c 65 32 e 68 ireshark -file2.h  
0060 74 6d 6c 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f tml HTTP /1.1 Ho  
0070 73 74 3a 20 67 61 69 61 2e 63 73 2e 75 6d 61 73 st: gaia .cs.umas  
0080 73 2e 65 64 75 0d 0a 55 73 65 72 2d 41 67 65 6e s.edu U ser-Agen  
0090 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 t: Mozil la/5.0 (  
00a0 57 69 6e 64 6f 77 73 20 4e 54 20 31 30 2e 30 3b Windows NT 10.0;  
00b0 20 57 69 6e 36 34 3b 20 78 36 34 3b 20 72 76 3a Win64; x64; rv:  
00c0 31 33 35 2e 30 29 20 47 65 63 6b 6f 2f 32 30 31 135.0) G ecko/201  
00d0 30 30 31 30 31 20 46 69 72 65 66 6f 78 2f 31 33 00101 Fi refex/13  
00e0 35 2e 30 0d 0a 41 63 63 65 70 74 3a 20 74 65 78 5.0 Acc ept: tex

When we delete the cache, we do not see it.

The image shows a Wireshark packet capture window titled "Wi-Fi 3". The main pane displays a list of captured packets. The selected packet is No. 4, a GET request for "/wireshark-labs/HTTP-wireshark-file2.html" from 10.0.0.50 to 128.119.245.12. The packet details pane shows the structure of the HTTP request, including the Host, User-Agent, Accept, Accept-Language, Accept-Encoding, DNT, Sec-GPC, Connection, Upgrade-Insecure-Requests, and Priority. The packet bytes pane shows the raw data of the request, including the GET method, the request URI, and the HTTP version.

Wireshark - Packet 4 - Wi-Fi 3

Frame 4: 473 bytes on wire (3784 bits), 473 bytes captured (3784 bits) on interface \Device\NPF\_{D65F1827-D907-40D0-A2A1-624ED0E88D6A}

Ethernet II, Src: Intel 5d:24:65 (18:93:41:5d:24:65), Dst: Commscope\_17:1c:c6 (1c:93:7c:17:1c:c6)

Internet Protocol Version 4, Src: 10.0.0.50, Dst: 128.119.245.12

Transmission Control Protocol, Src Port: 53046, Dst Port: 80, Seq: 1, Ack: 1, Len: 419

Hypertext Transfer Protocol

GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n

Host: gaia.cs.umass.edu\r\n

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:135.0) Gecko/20100101 Firefox/135.0\r\n

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8\r\n

Accept-Language: en-CA,en-US;q=0.7,en;q=0.3\r\n

Accept-Encoding: gzip, deflate\r\n

DNT: 1\r\n

Sec-GPC: 1\r\n

Connection: keep-alive\r\n

Upgrade-Insecure-Requests: 1\r\n

Priority: u=0, i\r\n

\r\n

[Response in frame 6]

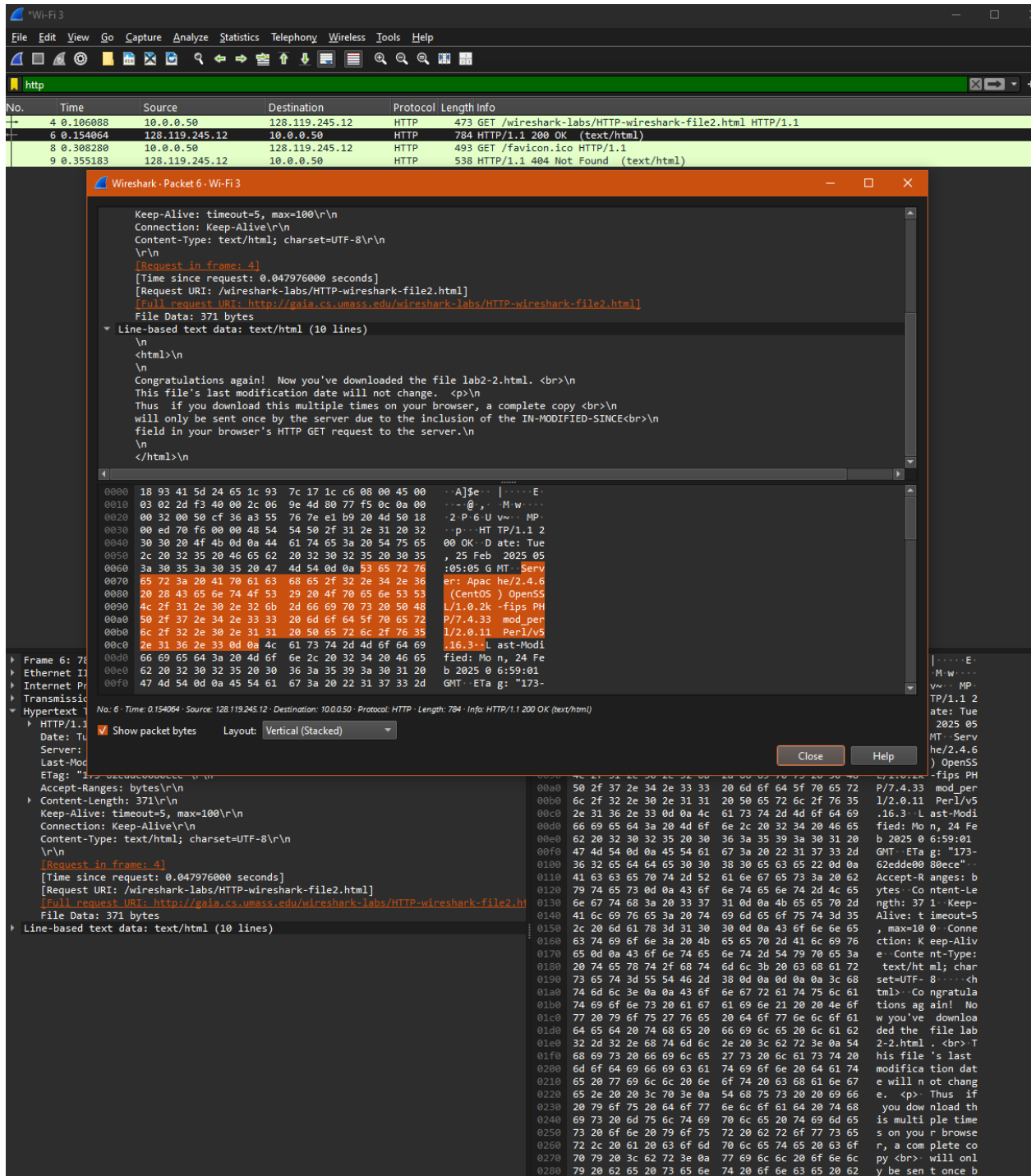
[Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html]

No. 4 Time: 0.106088 Source: 10.0.0.50 Destination: 128.119.245.12 Protocol: HTTP Length: 473 Info: GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1

Show packet bytes Layout: Vertical (Stacked)

Close Help

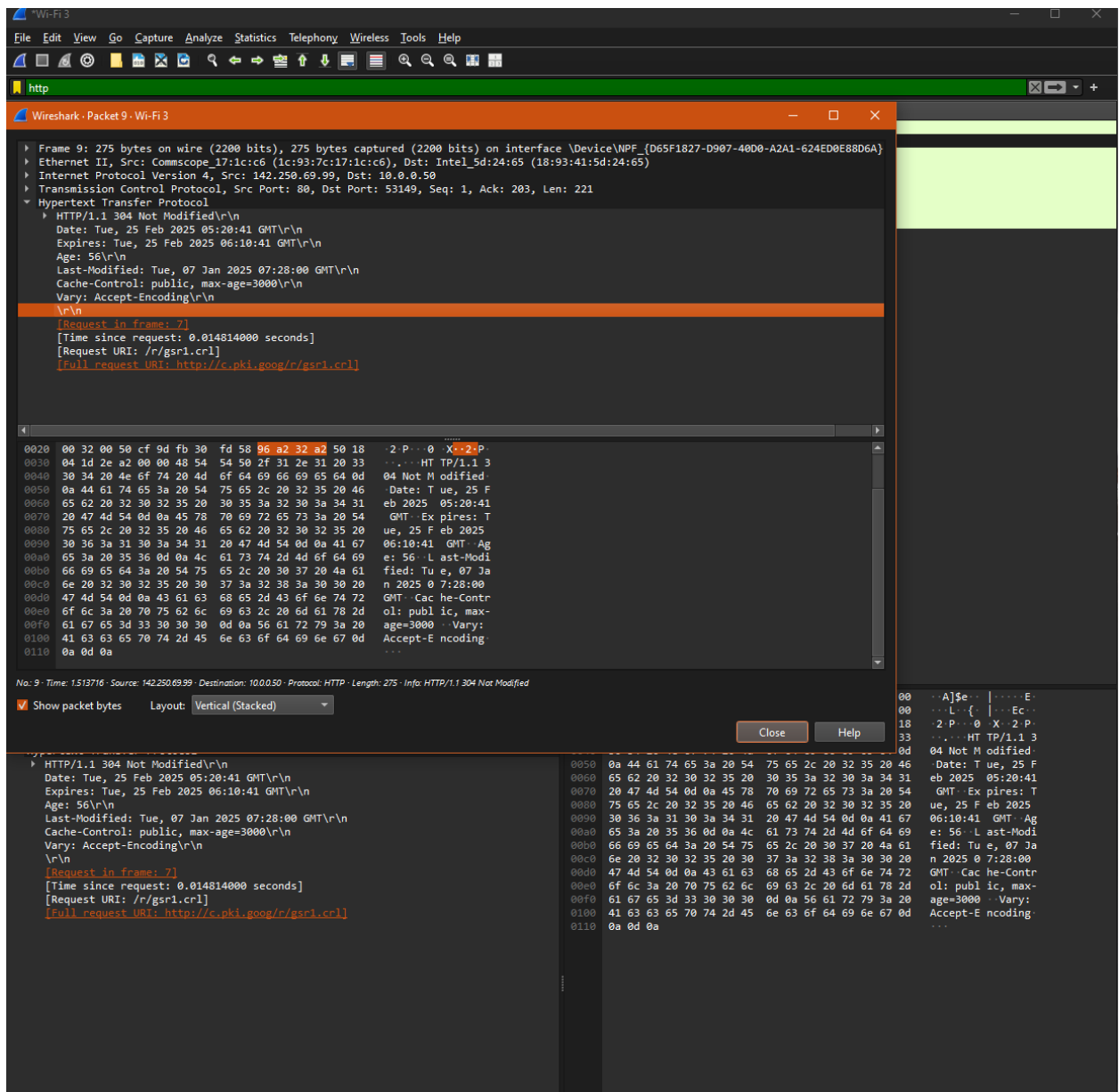
9. When we open the content-length and line-based text data, we can see all the details.



10. Like question 9, it only shows when the cache is not cleared. Otherwise, we can see it with those details If-Modified-Since: Mon, 24 Feb 2025 06:59:01.



11. The packet info reads “not modified” and the status code is 304. The server did not explicitly return the contents of the file this time as it was mentioned previously that if the cache is not cleared, the server will only send one copy of the file due to the inclusion of the If-Modified-Since.





### TASK 3

12. Our browser sent 1 HTTP GET requests.

13. 200 OK

14. 14 segments were needed to carry the single HTTP response.

15. Overhead refers to the additional resources used in a network protocol. Examples are TCP,

HTTP, etc. Every TCP segment contains a header that holds a certain number of bytes.

Those headers contain crucial information to manage and maintain the connection. Some

headers are destination port or sequence numbers.

```

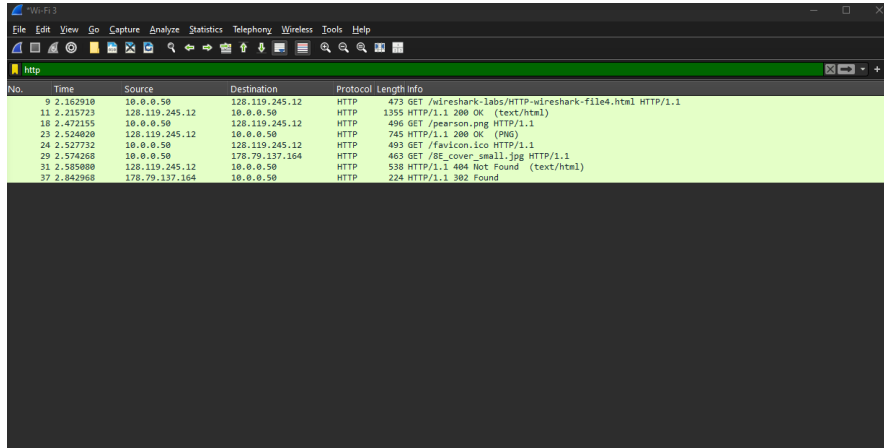
From 201.320.100.0 to 10.0.0.1 [4200 bits], 330 bytes captured (4200 bits) on Interface eth0:0/0 [DSCP:EF, DSCP:EF, 4800-A241-5241E8B0DA], 18.0
Ethernet II, Src: Composable_77:1c:0c:1e (1a:93:a1:56:24:65), Dst: Intel_E1:24:65 (1a:93:a1:56:24:65)
Internet Protocol Version 4, Src: 201.320.100.1, Dst: 10.0.0.1
Transmission Control Protocol, Src Port: 80, Dst Port: 3330, Seq: 4301, Ack: 430, Len: 401
Stream Name:
Destination Port: 3330
Stream Index: 0
Stream Packet Number: 9
Conversation completeness: Incomplete, DATA (15)
TCP Segment Len: 401
Sequence Number (raw): 98033213
Next Sequence Number: 4302 (relative sequence number)
Acknowledgment Number: 430 (relative ack number)
Acknowledgment number (raw): 1424230076
E101... 0 window length: 20 bytes (5)
Flags: 0x02 (PSH, ACK)
Window: 137
[calculated window size: 30336]
Window size scaling factor: 132
Checksum: 0x6ff9 [verified]
Checksum Status: Verified
Urgent Pointer: 0
[Timestamp]
[SIG/ACK analysis]
TCP payload (401 bytes)
TCP segment data (401 bytes)
[3 Resampled TCP segments (4001 bytes): #25(2020), #27(1400), #30(401)]
Hypertext Transfer Protocol
HTTP/1.1 200 OK/1
Server: Apache/2.4.6 (CentOS) OpenSsl/1.0.2-fips PM2/7.4.33 mod_per/2.0.11 Perl/5.16.3/vln
Last-Modified: Mon, 28 Feb 2023 06:50:01 GMT/vln
Etag: "133a-4b0e0e0e0a35/vln"
Accept-Ranges: bytes/vln
Content-Length: 4500/vln
Keep-Alive: timeout=60, max=100/vln
Connection: keep-alive/vln
Content-Type: text/html; charset=UTF-8/vln
vln/vln
[time since request: 0.00027000 seconds]
[request url: http://www.lanhttp-vln-tracker-files.html]
File Data: 4500 bytes
Line-based text data: text/html (90 lines)

```

## TASK 4

16. Our browser sent 3 GET requests. They were sent to 128.119.245.12 (gaia server) and 178.79.137.164 (French server).

17.



The image shows a Wireshark packet capture window with the 'http' filter applied. The packet list shows several HTTP GET requests. The first two requests are to 128.119.245.12, and the third is to 178.79.137.164. The packet details pane shows the structure of an HTTP GET request.

No.	Time	Source	Destination	Protocol	Length	Info
9	2.162910	10.0.0.50	128.119.245.12	HTTP	473	GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
11	2.215723	128.119.245.12	10.0.0.50	HTTP	1355	HTTP/1.1 200 OK (text/html)
18	2.472155	10.0.0.50	128.119.245.12	HTTP	496	GET /pearson.png HTTP/1.1
23	2.524820	128.119.245.12	10.0.0.50	HTTP	745	HTTP/1.1 200 OK (PNG)
24	2.527732	10.0.0.50	128.119.245.12	HTTP	463	GET /favicon.ico HTTP/1.1
29	2.574268	10.0.0.50	178.79.137.164	HTTP	463	GET /SE_cover_small.jpg HTTP/1.1
31	2.585888	128.119.245.12	10.0.0.50	HTTP	538	HTTP/1.1 404 Not Found (text/html)
37	2.642968	178.79.137.164	10.0.0.50	HTTP	224	HTTP/1.1 302 Found

Our browser downloaded the two images in parallel. After the browser sent the first GET request to the server at the 128 address, another GET request to the same address followed shortly after. Then, almost at the same time, a GET request was sent to the 178 address. This shows that the browser was requesting the images from two different servers at the same time. GET request to the 178 address was made quickly after the one to the 128 address, indicating that both requests were made almost simultaneously. This suggests that the browser didn't wait for one image to finish downloading before it started requesting the next. Instead, it fetched them in parallel, which is a common practice to speed up page loading by getting resources from different servers at once. Also, when refreshing the page, we can notice an overlap in GET coming from the server inside dev tools. This further proves that it was downloaded in parallel.