INTERNATIONAL CYBERSECURITY AND DIGITAL FORENS ACADEMY

INT312 - Basic Networking Skills for Digital Forensics

Ayilara Busari Dare

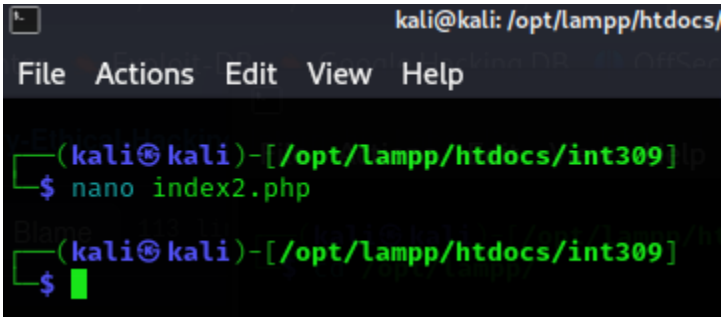
IDEAS/24/28133

INT312 - Basic Networking Skills for Digital Forensics – Lab 1

Network forensics is a specialized area of digital forensics that focuses on the monitoring, analysis, and investigation of network traffic for evidence of security incidents or malicious activities. It involves capturing, recording, and analyzing data packets to understand network events, identify compromised systems, and tracing attack vectors.

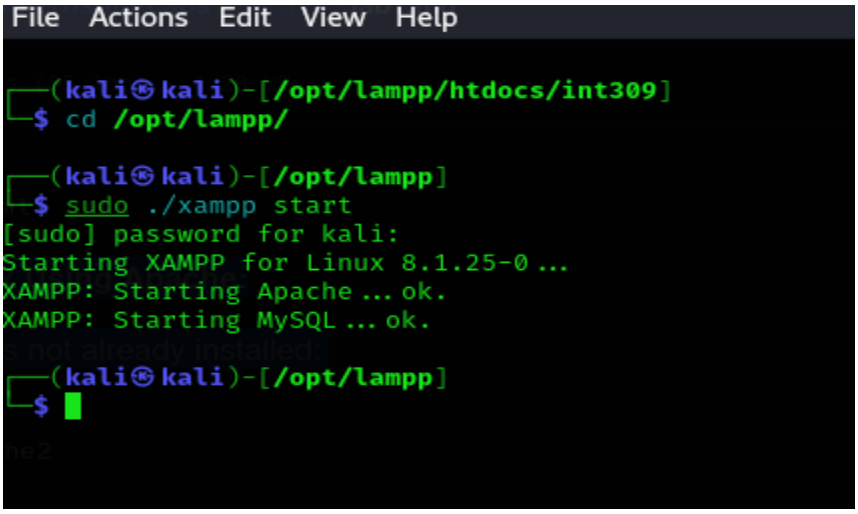
Lab Title: Creating a Simple Website and Capturing Network Traffic

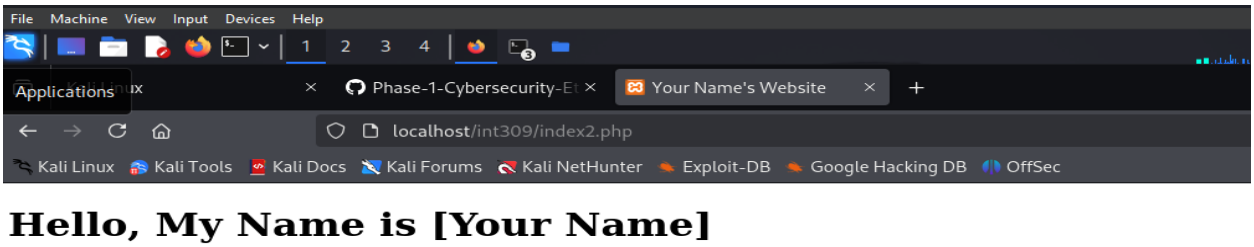
Part 1: Create a Simple Website



2. Hosting the Website on Linux Using Apache:

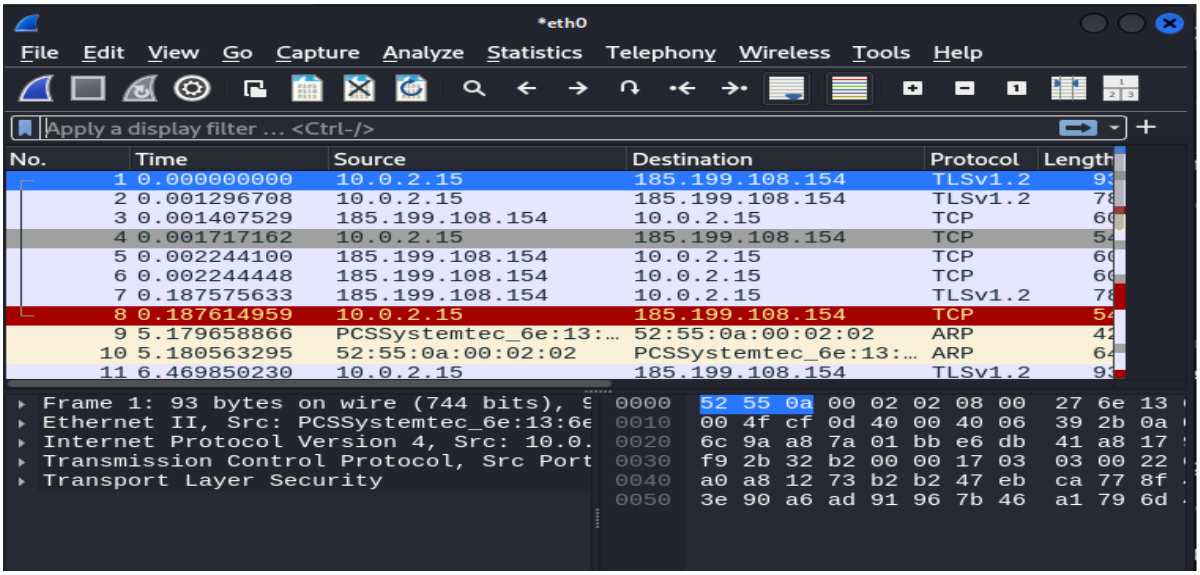
- **Step 1:** Install Apache if it's not installed yet





Part 2: Capture Network Traffic

1. **Set Up Wireshark:**
   - Open Wireshark and select the appropriate network interface (e.g., Wi-Fi or Ethernet) to capture traffic.
   - Start capturing packets before accessing your website.
2. **Access Your Website:**
   - In a web browser, navigate to your hosted website (e.g., http://localhost/index.html).

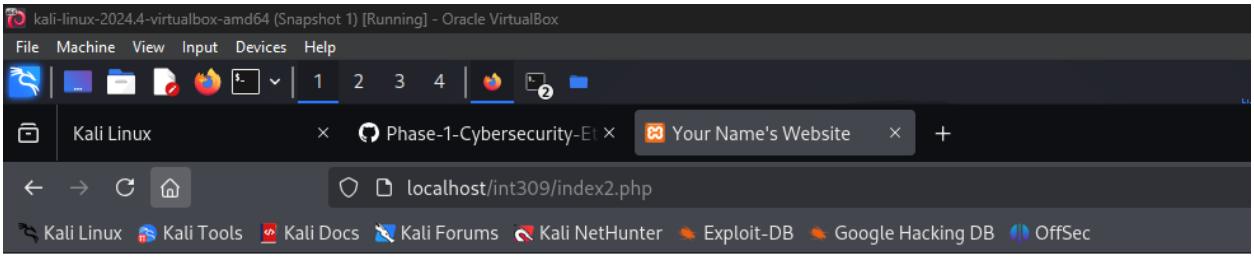


Part 3: Capture Screenshots with Required Information

1. **Information to Capture:**

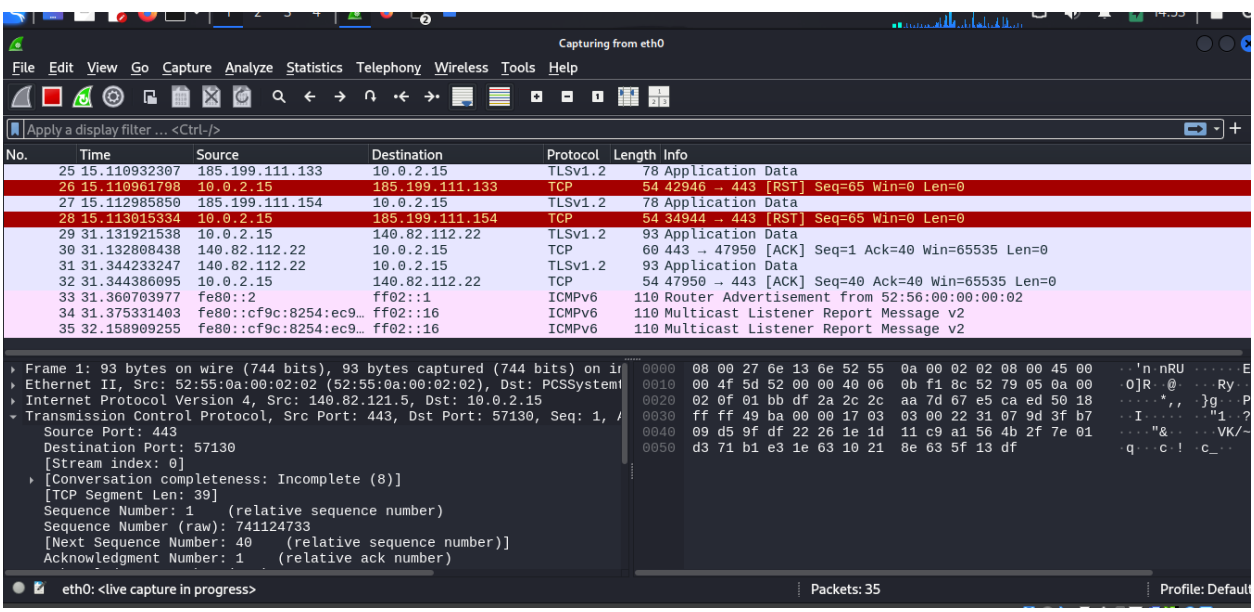
Take screenshots of the following data from Wireshark and your web browser:

- **The website displays your name.**



Hello, My Name is [Ayilara Busari Dare]

- **Ports used (sender and receiver):** Identify the source and destination ports in the TCP packets.

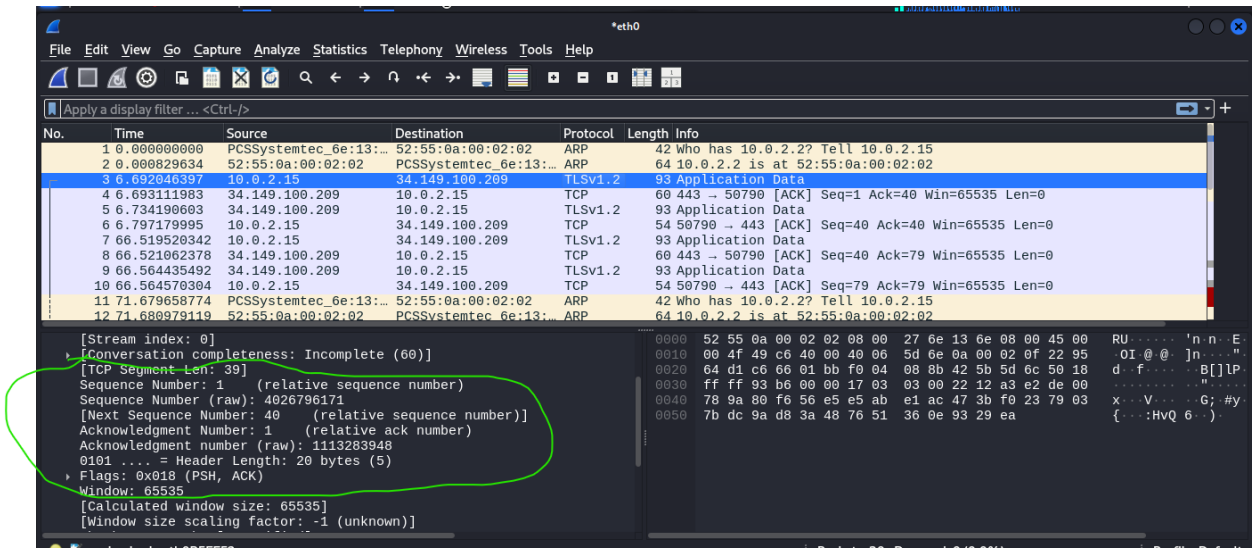


- **Initial sequence numbers (sender and receiver):** Locate the sequence numbers in the TCP handshake (SYN, SYN-ACK, ACK) packets.

  TCP Sequence (seq) and Acknowledgement (ack) numbers help enable ordered reliable data transfer for TCP streams. The seq number is sent by the TCP client,

indicating how much data has been sent for the session (also known as the byte-order number). The ack number is sent by the TCP server, indicating that is has received cumulated data and is ready for the next segment.

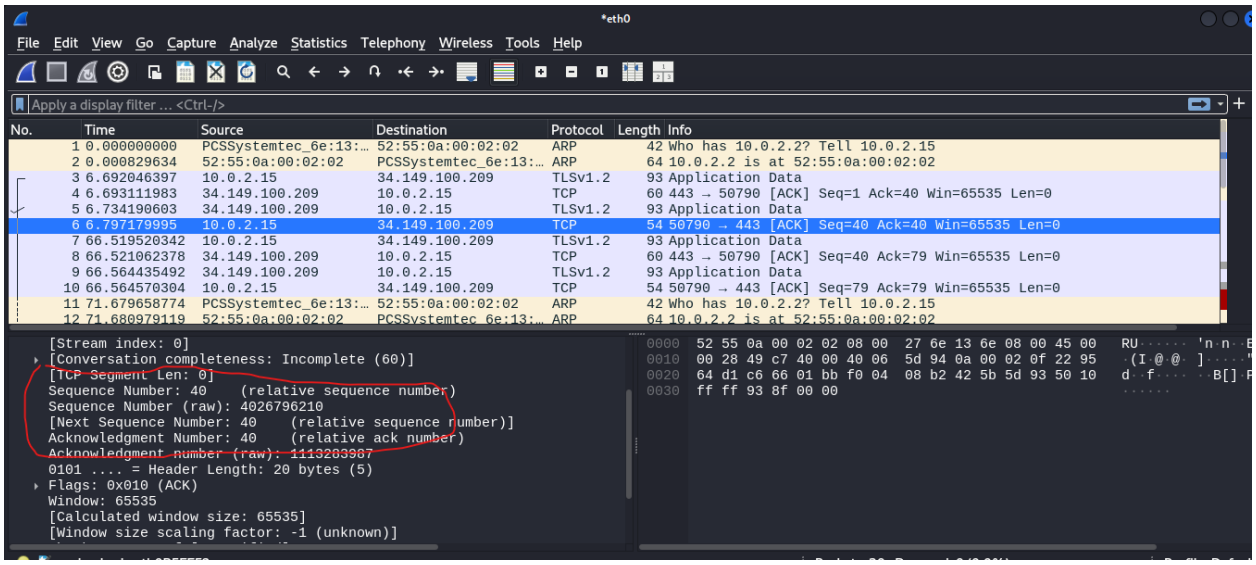
Client sends seq=1 and tcp segment length=39



The client sends the first segment with seq=1 and the length of the segment is 39 bytes. The server responds with an ack=40 which tells the client that the next expected segment will have a sequence number is 1.

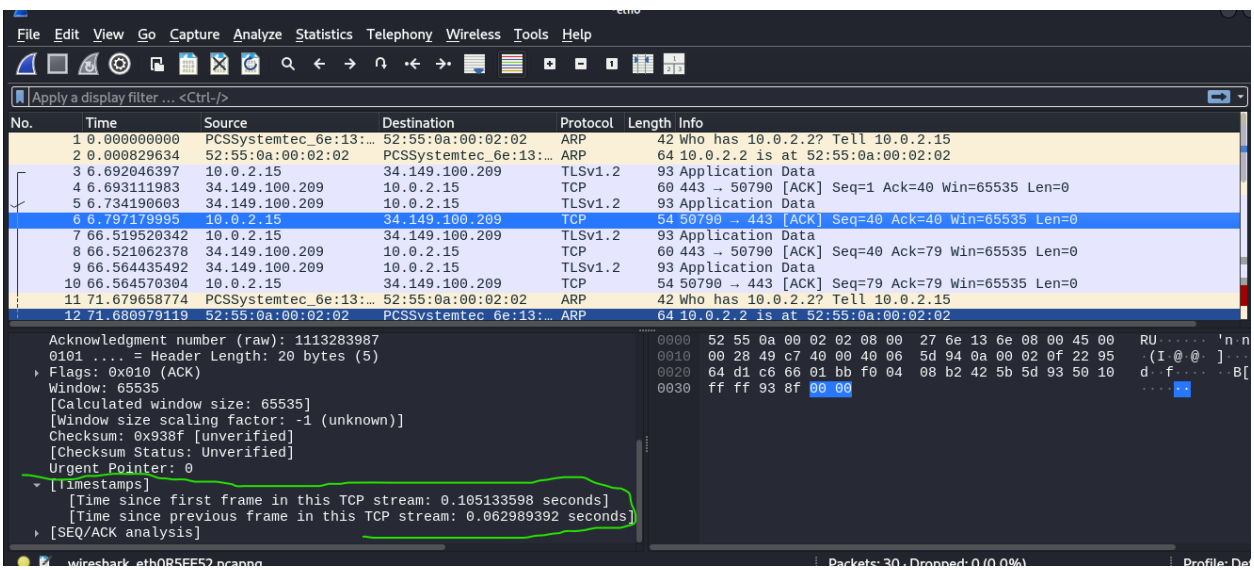
The next segment the client sends is seq=40 and the length is now 0 bytes. In turn, the server responds with ack=40. This cycle continues until the end of the TCP session.

Server responds with ack=40



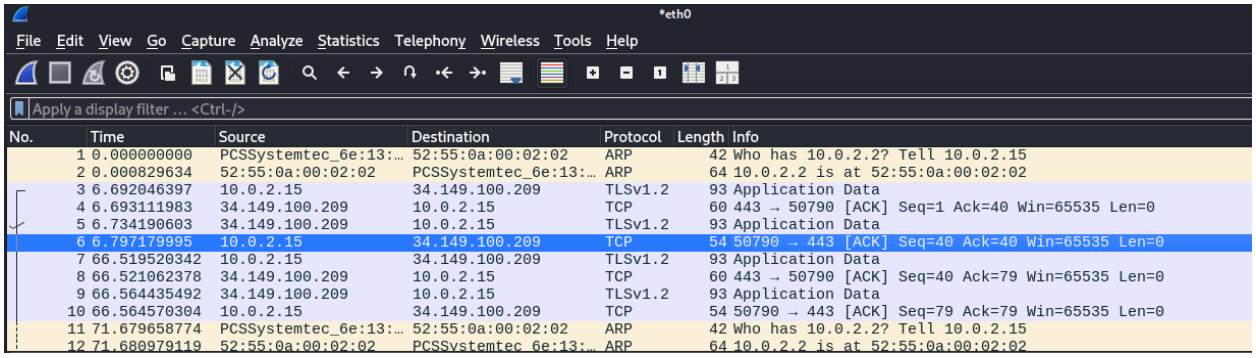
- **Timestamps of the TCP handshake:** Note the timestamps of the SYN, SYN-ACK, and ACK packets.

  Time stamps of the tcp handshake between client and server to establish a connection, the time since first frame in this tcp stream is 0.105 seconds and the time since previous frame in the tcp stream is 0.062 seconds



- **IP addresses (sender and receiver):** Identify your machine's IP address and the localhost address.

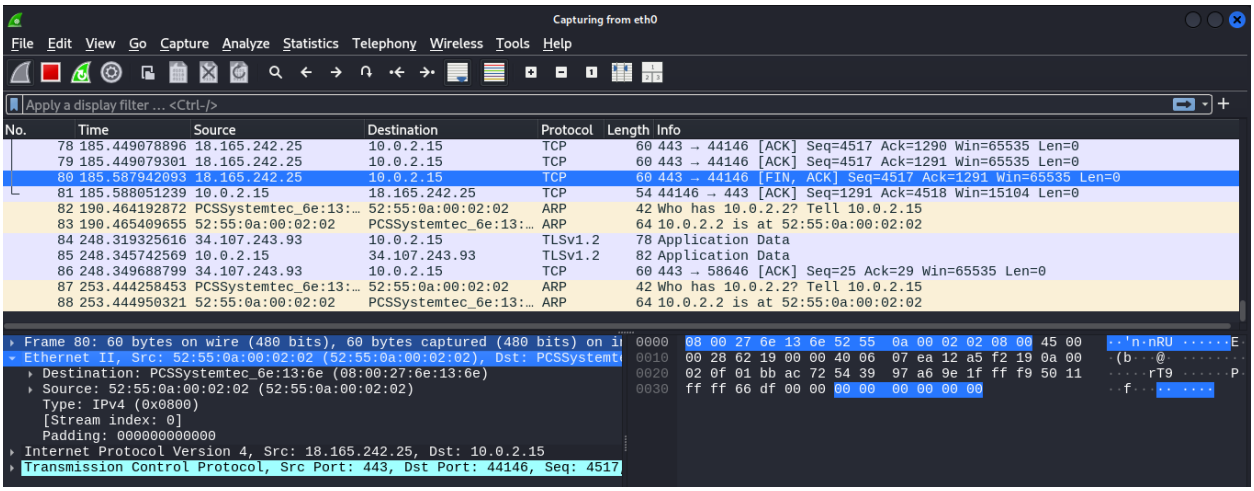
  The sender (machine IP address) is 10.0.2.15 and receiver IP address (localhost address) is 34.149.108.209



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PCSSystemtec_6e:13:…	52:55:0a:00:02:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
2	0.000829634	52:55:0a:00:02:02	PCSSystemtec_6e:13:…	ARP	64	10.0.2.2 is at 52:55:0a:00:02:02
3	6.692046397	10.0.2.15	34.149.100.209	TLSv1.2	93	Application Data
4	6.693111983	34.149.100.209	10.0.2.15	TCP	60	443 → 50790 [ACK] Seq=1 Ack=40 Win=65535 Len=0
5	6.734190603	34.149.100.209	10.0.2.15	TLSv1.2	93	Application Data
6	6.797179995	10.0.2.15	34.149.100.209	TCP	54	50790 → 443 [ACK] Seq=40 Ack=40 Win=65535 Len=0
7	66.519520342	10.0.2.15	34.149.100.209	TLSv1.2	93	Application Data
8	66.521062378	34.149.100.209	10.0.2.15	TCP	60	443 → 50790 [ACK] Seq=40 Ack=79 Win=65535 Len=0
9	66.564435492	34.149.100.209	10.0.2.15	TLSv1.2	93	Application Data
10	66.564570304	10.0.2.15	34.149.100.209	TCP	54	50790 → 443 [ACK] Seq=79 Ack=79 Win=65535 Len=0
11	71.679658774	PCSSystemtec_6e:13:…	52:55:0a:00:02:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
12	71.680979119	52:55:0a:00:02:02	PCSSystemtec_6e:13:…	ARP	64	10.0.2.2 is at 52:55:0a:00:02:02

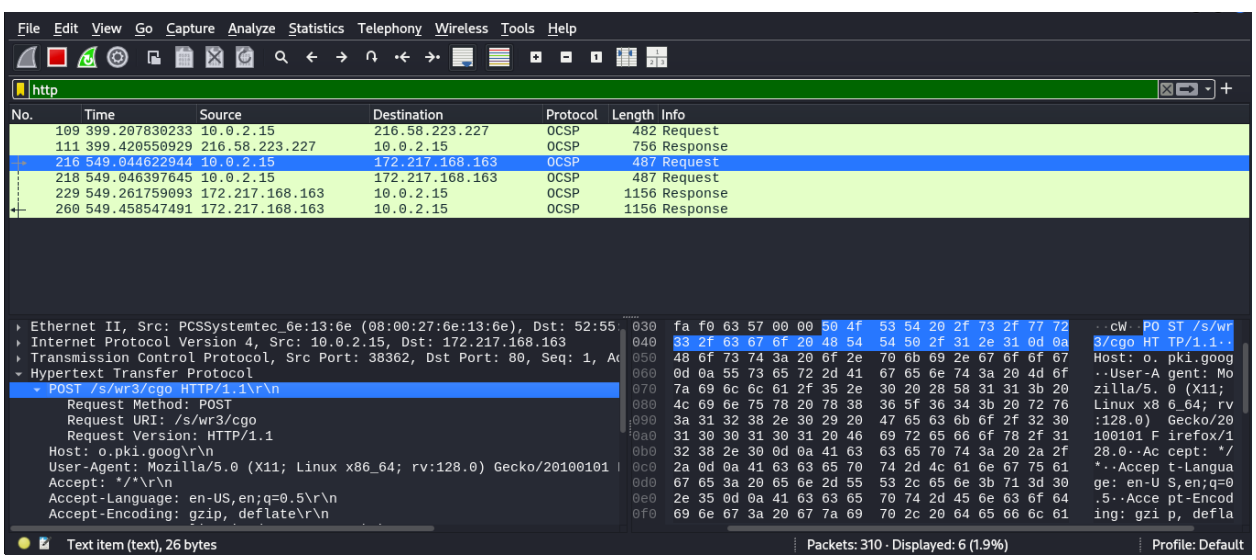
- **MAC addresses (sender and receiver):** Capture the Ethernet frame details to find the MAC addresses.

  Ethernet II, Src: 52:55:0a:00:02:02 (52:55:0a:00:02:02), Dst: PCSSystemtec_6e:13:6e (08:00:27:6e:13:6e)

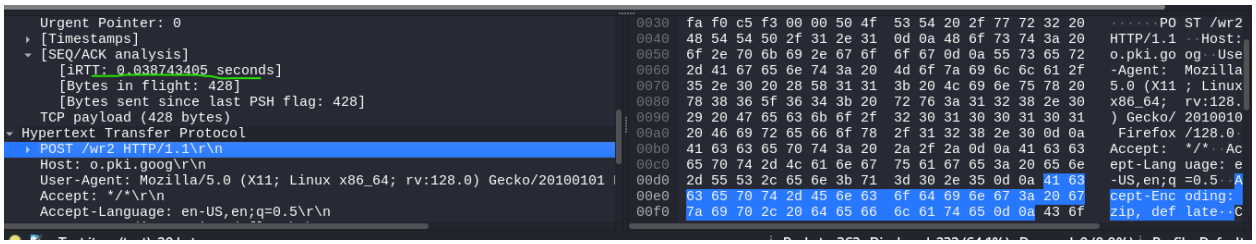


No.	Time	Source	Destination	Protocol	Length	Info
78	185.449078896	18.165.242.25	10.0.2.15	TCP	60	443 → 44146 [ACK] Seq=4517 Ack=1290 Win=65535 Len=0
79	185.449079301	18.165.242.25	10.0.2.15	TCP	60	443 → 44146 [ACK] Seq=4517 Ack=1291 Win=65535 Len=0
80	185.587942093	18.165.242.25	10.0.2.15	TCP	60	443 → 44146 [FIN, ACK] Seq=4517 Ack=1291 Win=65535 Len=0
81	185.588051239	10.0.2.15	18.165.242.25	TCP	54	44146 → 443 [ACK] Seq=1291 Ack=4518 Win=15104 Len=0
82	190.464192872	PCSSystemtec_6e:13:…	52:55:0a:00:02:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
83	190.465409655	52:55:0a:00:02:02	PCSSystemtec_6e:13:…	ARP	64	10.0.2.2 is at 52:55:0a:00:02:02
84	248.319325616	34.107.243.93	10.0.2.15	TLSv1.2	78	Application Data
85	248.345742569	10.0.2.15	34.107.243.93	TLSv1.2	82	Application Data
86	248.349688799	34.107.243.93	10.0.2.15	TCP	60	443 → 58646 [ACK] Seq=25 Ack=29 Win=65535 Len=0
87	253.444258453	PCSSystemtec_6e:13:…	52:55:0a:00:02:02	ARP	42	Who has 10.0.2.2? Tell 10.0.2.15
88	253.444950321	52:55:0a:00:02:02	PCSSystemtec_6e:13:…	ARP	64	10.0.2.2 is at 52:55:0a:00:02:02

- **Timestamps of the HTTP request:** Note the timestamp of the HTTP GET request packet.



No.	Time	Source	Destination	Protocol	Length	Info
109	399.207830233	10.0.2.15	216.58.223.227	OCSP	482	Request
111	399.420550929	216.58.223.227	10.0.2.15	OCSP	756	Response
216	549.044622944	10.0.2.15	172.217.168.163	OCSP	487	Request
218	549.046397645	10.0.2.15	172.217.168.163	OCSP	487	Request
229	549.261759093	172.217.168.163	10.0.2.15	OCSP	1156	Response
260	549.458547491	172.217.168.163	10.0.2.15	OCSP	1156	Response

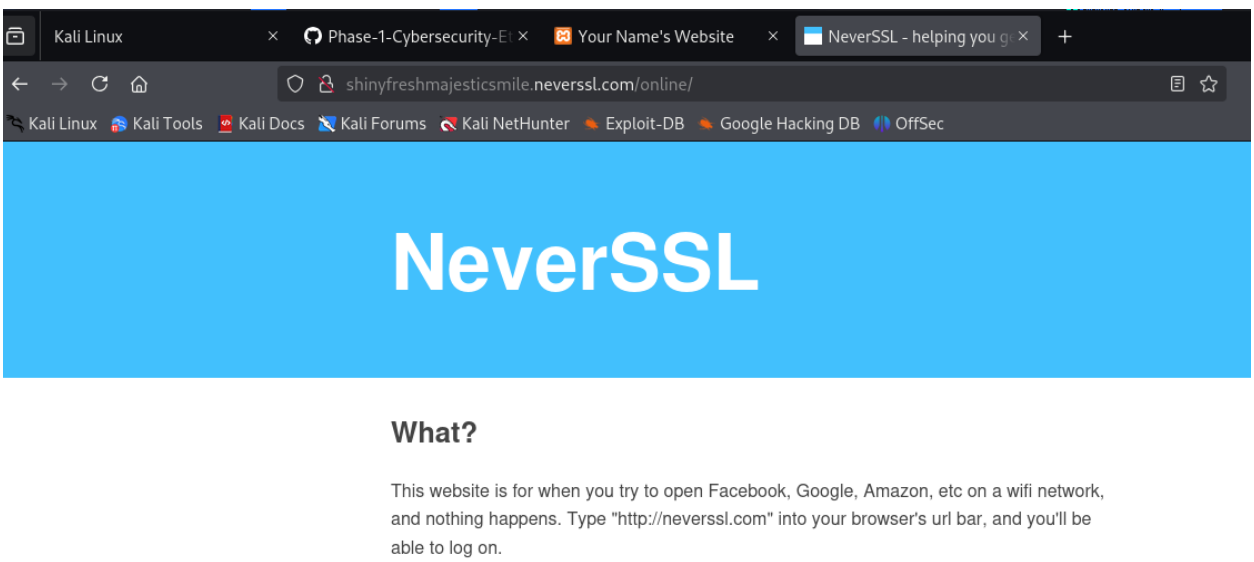


2. **Repeat the Process with External Site:**
   - Access the website: http://shinyfreshmajesticsmile.neverssl.com/online/.
   - Capture the same information as above for this website.

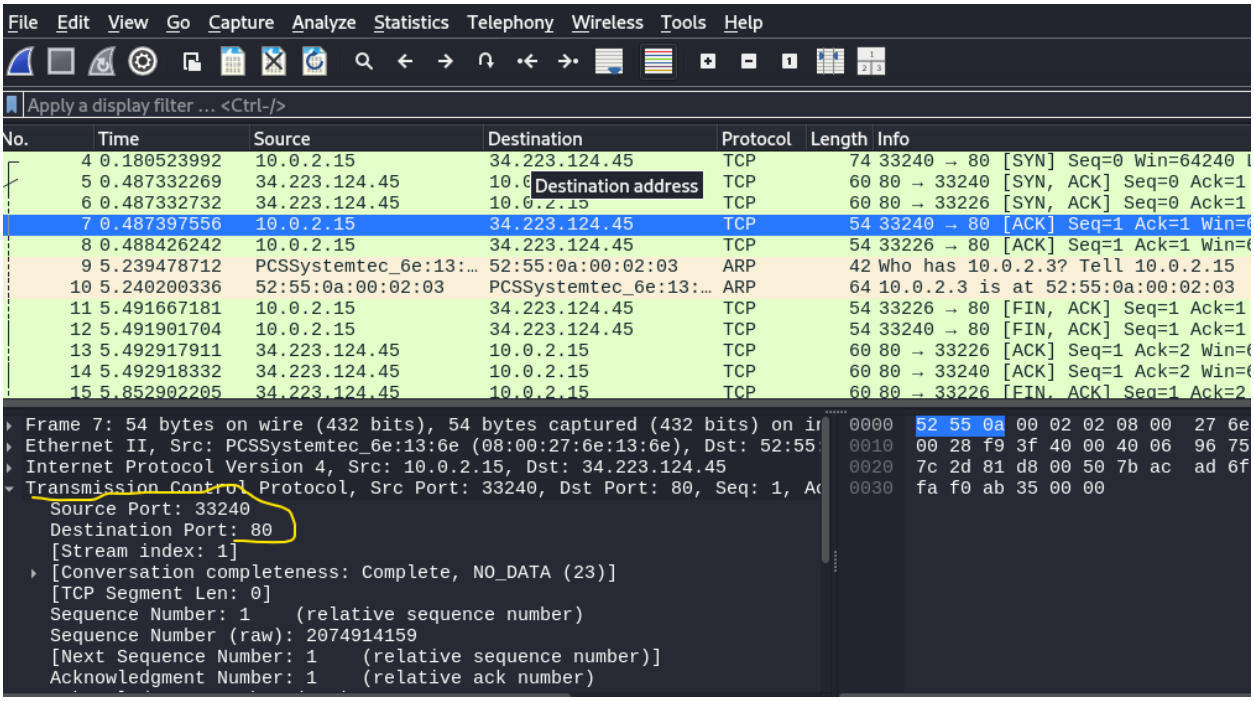
Ans:

- **The website displays your name.**

The name of website display is NeverSSl-helping you get online



- **Ports used (sender and receiver):** Identify the source and destination ports in the TCP packets.



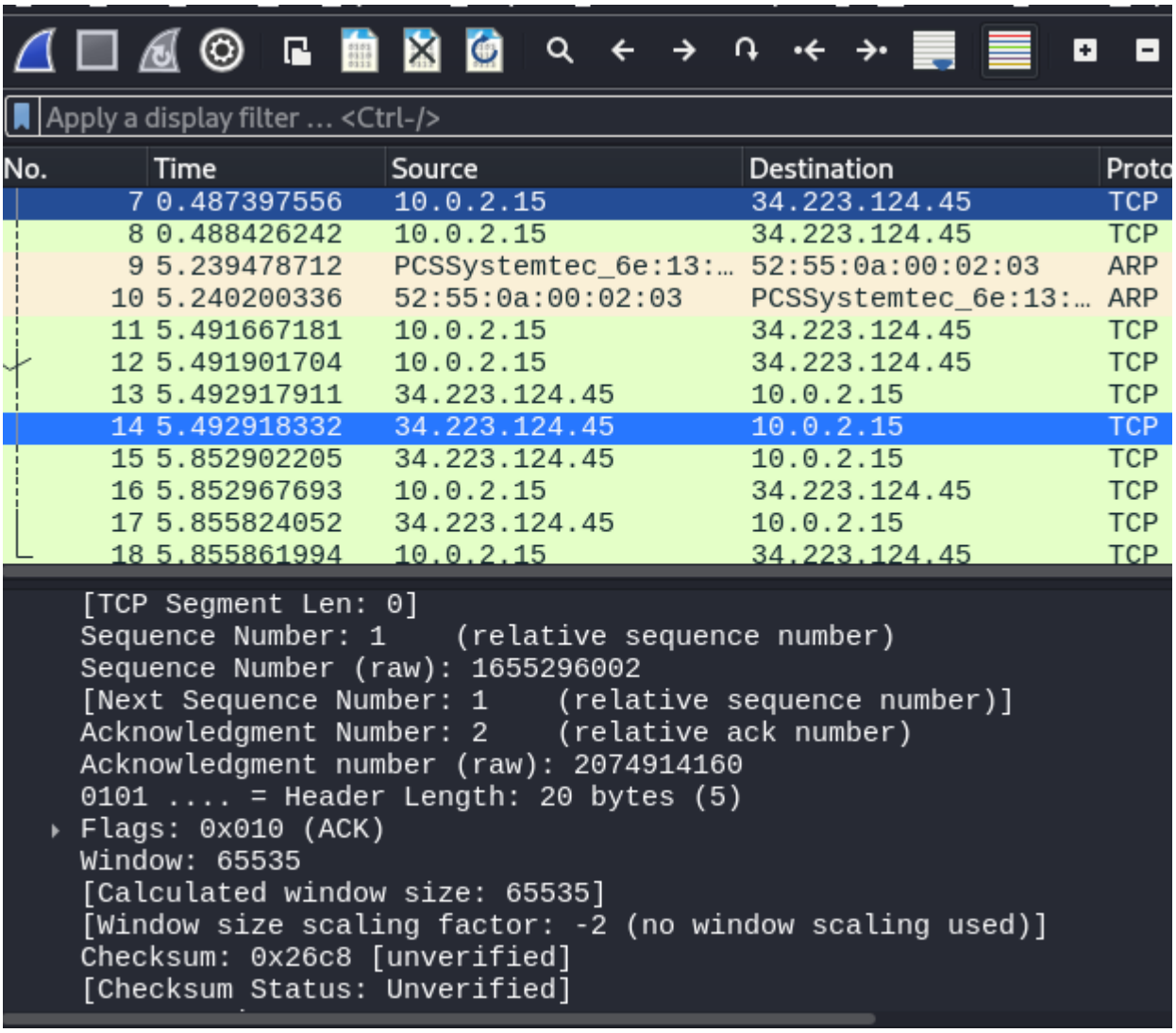
- **Initial sequence numbers (sender and receiver):** Locate the sequence numbers in the TCP handshake (SYN, SYN-ACK, ACK) packets.

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol
4	0.180523992	10.0.2.15	34.223.124.45	TCP
5	0.487332269	34.223.124.45	10.0.2.15	TCP
6	0.487332732	34.223.124.45	10.0.2.15	TCP
7	0.487397556	10.0.2.15	34.223.124.45	TCP
8	0.488426242	10.0.2.15	34.223.124.45	TCP
9	5.239478712	PCSSystemtec_6e:13:…	52:55:0a:00:02:03	ARP
10	5.240200336	52:55:0a:00:02:03	PCSSystemtec_6e:13:…	ARP
11	5.491667181	10.0.2.15	34.223.124.45	TCP
12	5.491901704	10.0.2.15	34.223.124.45	TCP
13	5.492917911	34.223.124.45	10.0.2.15	TCP
14	5.492918332	34.223.124.45	10.0.2.15	TCP
15	5.852902205	34.223.124.45	10.0.2.15	TCP

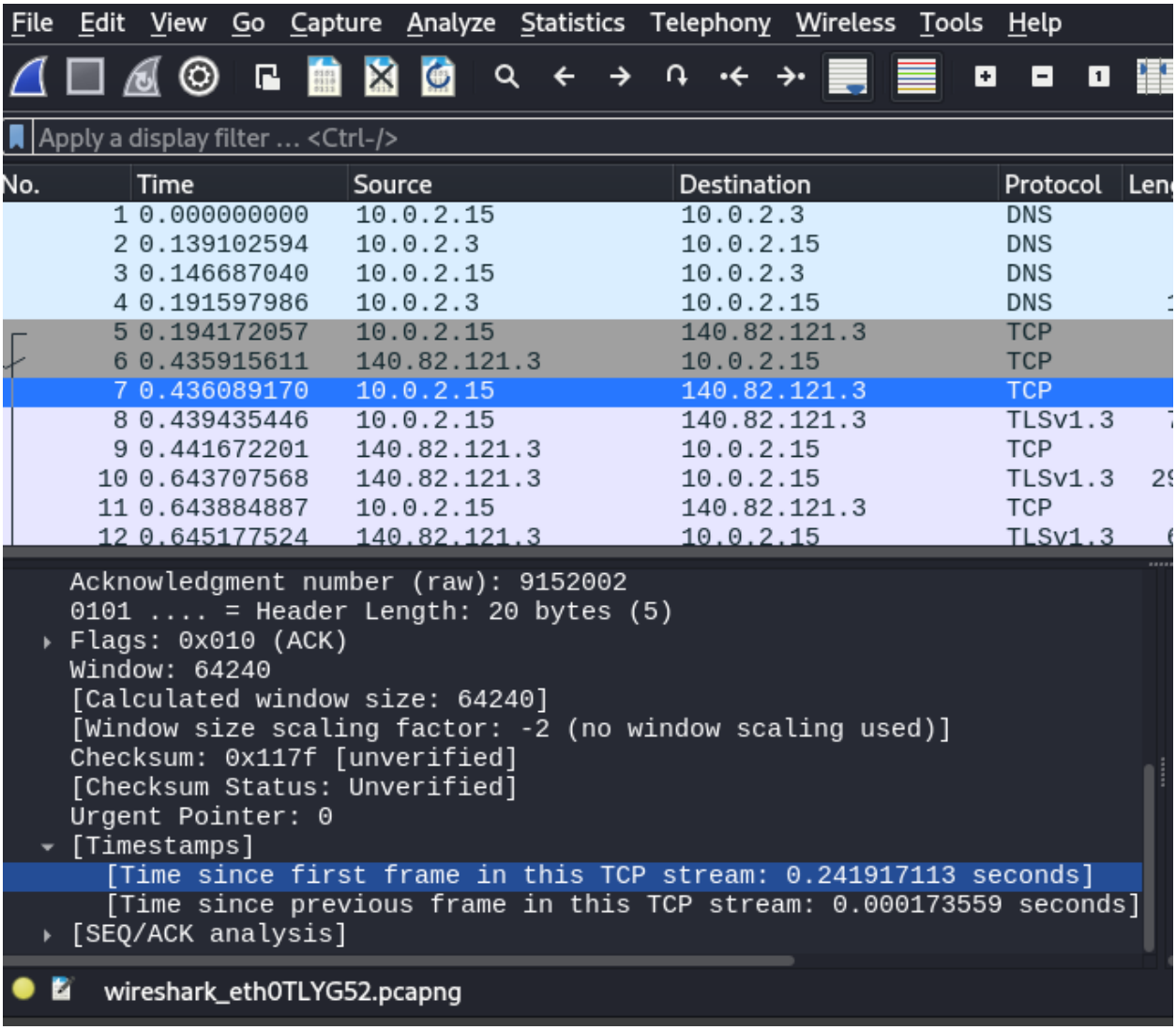
```
    [Stream index: 1]
  ▸ [Conversation completeness: Complete, NO_DATA (23)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2074914159
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1655296002
    0101 .... = Header Length: 20 bytes (5)
  ▸ Flags: 0x010 (ACK)
    Window: 64240
    [Calculated window size: 64240]
    [Window size scaling factor: -2 (no window scaling used)]
```



No.	Time	Source	Destination	Proto
7	0.487397556	10.0.2.15	34.223.124.45	TCP
8	0.488426242	10.0.2.15	34.223.124.45	TCP
9	5.239478712	PCSSystemtec_6e:13:…	52:55:0a:00:02:03	ARP
10	5.240200336	52:55:0a:00:02:03	PCSSystemtec_6e:13:…	ARP
11	5.491667181	10.0.2.15	34.223.124.45	TCP
12	5.491901704	10.0.2.15	34.223.124.45	TCP
13	5.492917911	34.223.124.45	10.0.2.15	TCP
14	5.492918332	34.223.124.45	10.0.2.15	TCP
15	5.852902205	34.223.124.45	10.0.2.15	TCP
16	5.852967693	10.0.2.15	34.223.124.45	TCP
17	5.855824052	34.223.124.45	10.0.2.15	TCP
18	5.855861994	10.0.2.15	34.223.124.45	TCP

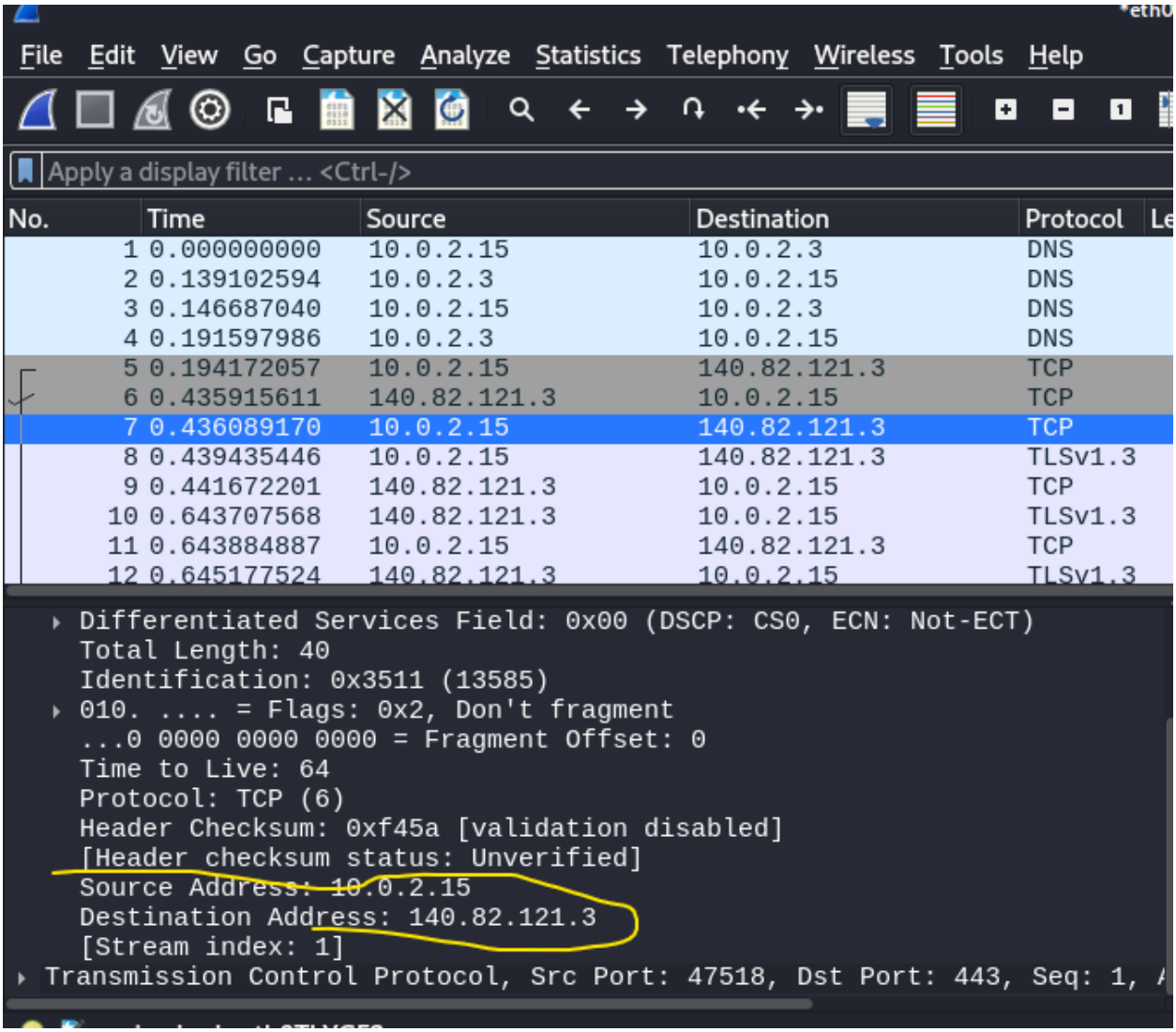
```
[TCP Segment Len: 0]
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 1655296002
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 2    (relative ack number)
Acknowledgment number (raw): 2074914160
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 65535
[Calculated window size: 65535]
[Window size scaling factor: -2 (no window scaling used)]
Checksum: 0x26c8 [unverified]
[Checksum Status: Unverified]
```

Timestamps of the TCP handshake: Note the timestamps of the SYN, SYN-ACK, and ACK packets

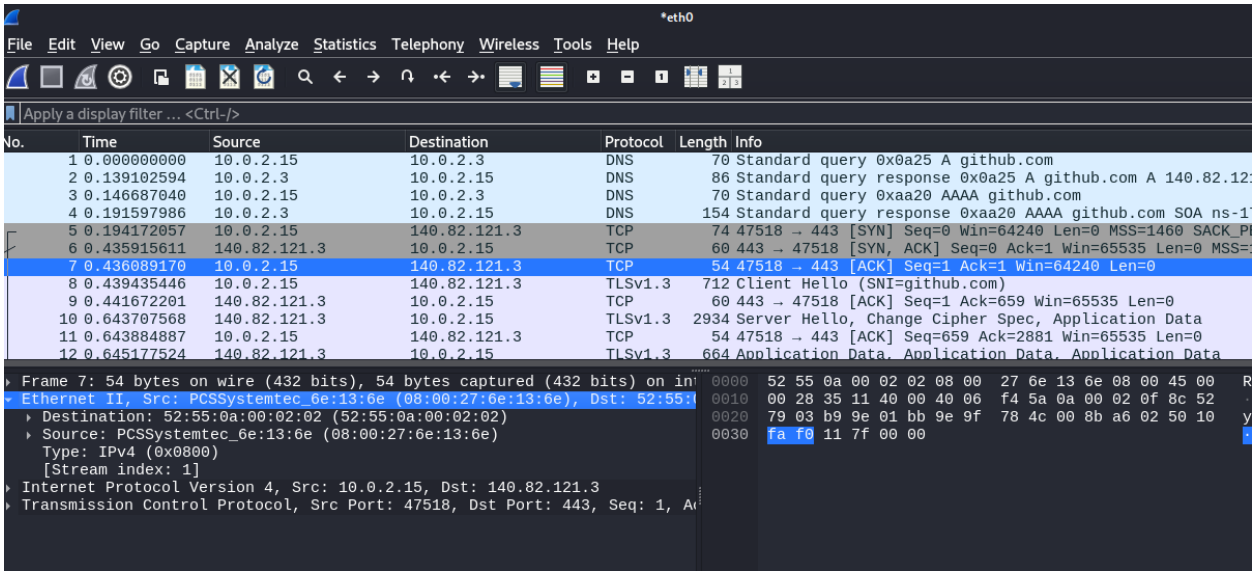


- **IP addresses (sender and receiver):** Identify your machine's IP address and the localhost address.

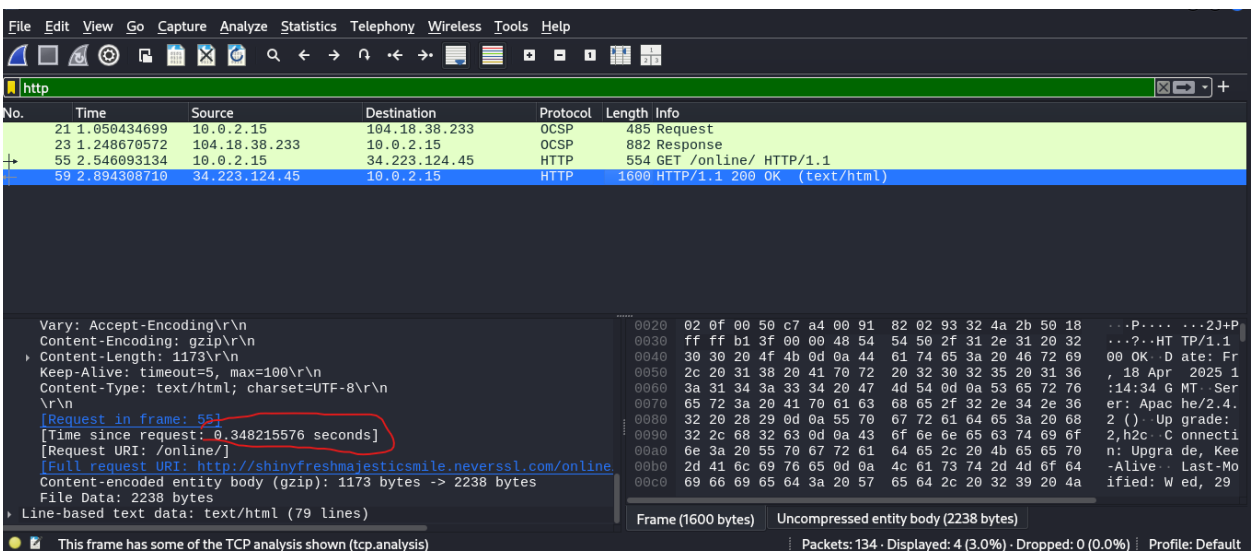
  The sender (machine IP address) is 10.0.2.15 and receiver IP address (localhost address) is 340.81.121.3



- **MAC addresses (sender and receiver):** Capture the Ethernet frame details to find the MAC addresses.
  Ethernet II, Src: PCSSystemtec_6e:13:6e (08:00:27:6e:13:6e), Dst: 52:55:0a:00:02:02 (52:55:0a:00:02:02)



- **Timestamps of the HTTP request:** Note the timestamp of the HTTP GET request packet.



INT312 - Basic Networking Skills for Digital Forensics – Lab 2: Title: Capturing and Analyzing HTTP Traffic with Embedded Images

Part 2: Captured HTTP Traffic Overview

Part 1: Capture HTTP Traffic

1. **Set Up Wireshark:**

- Open Wireshark and select the appropriate network interface (e.g., Wi-Fi or Ethernet).

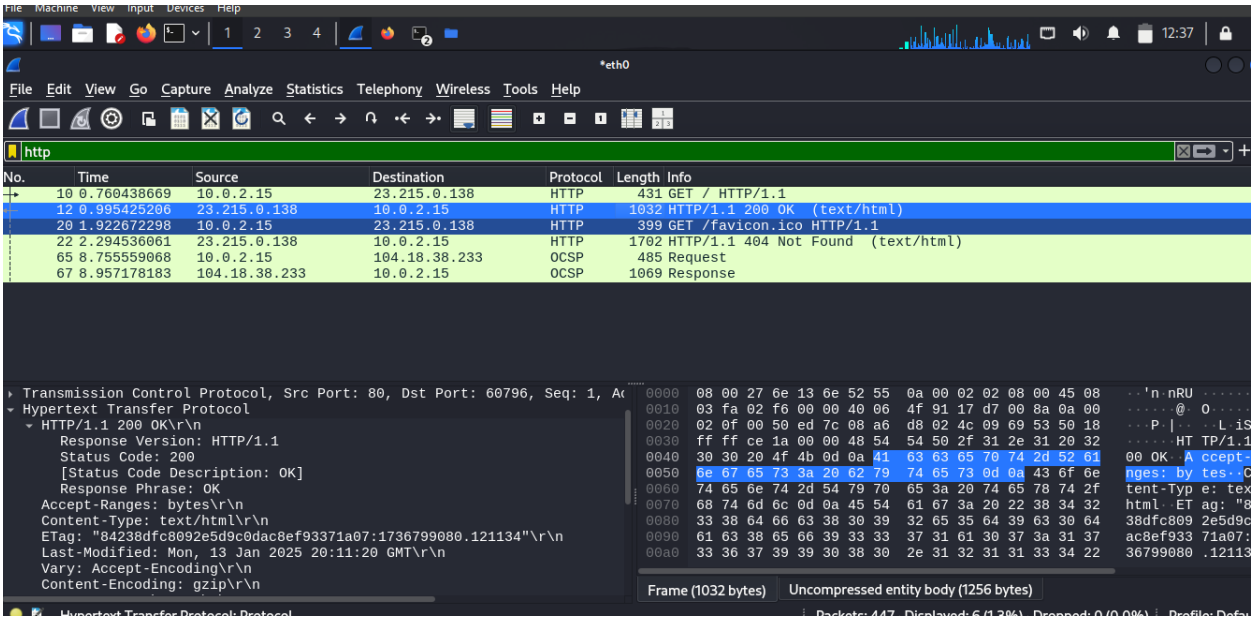
- Start capturing packets before making an HTTP request to a website that contains an embedded image.

2. Access a Website with an Embedded Image:

- Open a web browser and navigate to a website with an embedded image (you may choose any website that you like or use the example below): http://example.com

3. **Stop the Capture:**

- After the page loads, stop the packet captured in Wireshark.

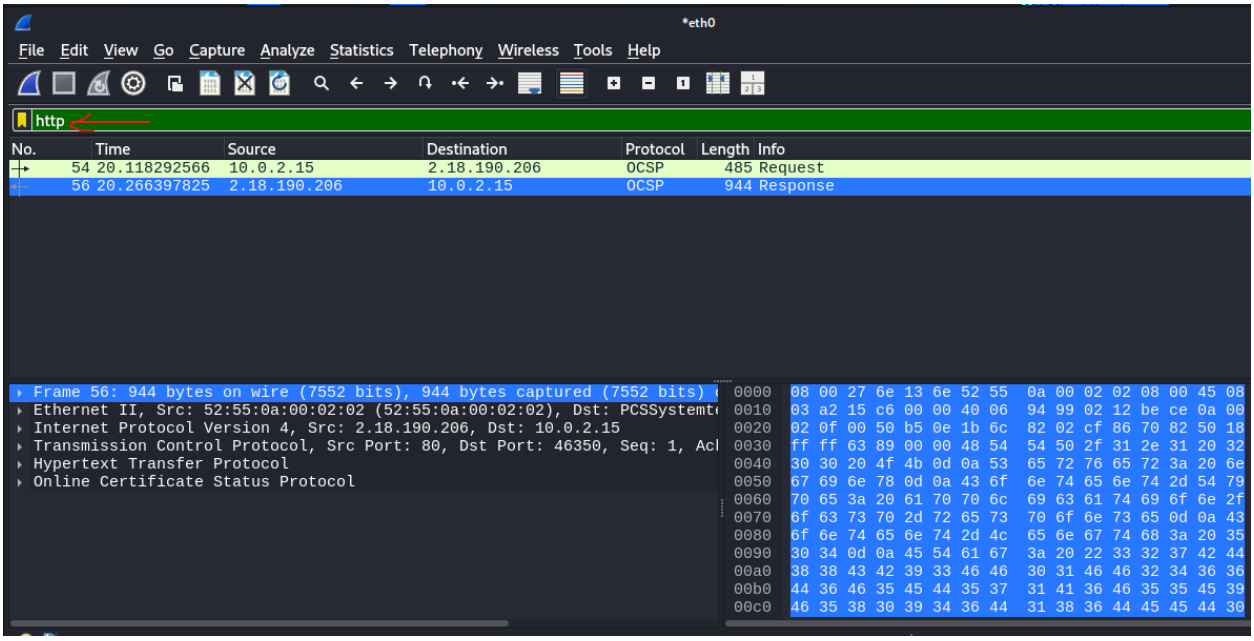


The Wireshark was open to capture traffic on the eth0 network, after this an http request was made using example.com which contain an embedded file (PDF), after the web page load and some traffic were captured by Wireshark, it then stop for analysis

Part 2: Captured HTTP Traffic Overview

1. **Identify the HTTP GET Request:**

- In Wireshark, filter the traffic to show only HTTP packets: http
- Look for the second HTTP GET request in the list of captured packets.



2. Analyze the Second HTTP GET Request/Response:

- Click on the second HTTP GET request packet.
- Analyze the following details:

Request Line: Check the URL being requested.

- **Request Line:**
  - **Request Method**: GET
    - This indicates that the client is requesting data from the server, specifically for the resource at the specified URI.
  - **Request URI**: /
    - The URI indicates that the client is requesting the root resource of the server.
  - **Request Version**: HTTP/1.1
    - Specifies the HTTP version being used for the request.

Headers: Review relevant headers such as User-Agent, Accept, Host, etc.

- **Headers:**
  - **Host**: example.com

        - Specifies the domain name of the server (this is mandatory in HTTP/1.1).
    - **User-Agent**: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
        - Provides identification of the client software making the request (browser type and version, which can help the server tailor responses appropriately).
    - **Accept**: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
        - Indicates the types of content the client is willing to accept. The q values indicate the priority of each type; the higher the q value, the more preferred the type.
    - **Accept-Language**: en-US,en;q=0.5
        - Specifies the preferred languages for the response, in this case, primarily English (US) with a fallback to other English variants.
    - **Accept-Encoding**: gzip, deflate
        - Specifies the content encodings the client can understand. This indicates that the client supports gzip and deflate compression, meaning the server can send compressed responses.
    - **Connection**: keep-alive
        - Requests that the server maintain the connection open for further requests, which can increase performance by reducing latency for subsequent requests.
    - **Upgrade-Insecure-Requests**: 1
        - This indicates a request to upgrade to secure connections (HTTPS) when possible, suggesting that the client is requesting the server to serve resources securely.
    - **Priority**: u=0, i

        - This header suggests a priority setting, likely influencing how network resources are allocated, where u=0 indicates low urgency and i indicates importance.

- **Locate the corresponding HTTP response packet.**
    - **Status Code:** Check if the response is successful (200 OK).
    - **Headers:** Review the Content-Type and Content-Length headers.
    - **Payload:** If the response contains an image, identify the details related to the image.

Response Line:

- **HTTP Version**: HTTP/1.1
- **Status Code**: 200
- **Status Code Description**: OK
- **Response Phrase**: OK

This indicates that the request to the server was successful, and the server is returning the requested resource.

- **Headers:**
    - **Accept-Ranges**: bytes
        - This informs the client that the server supports a range of requests for bytes, allowing clients to request specific portions of a resource.
    - **Content-Type**: text/html
        - Specifies that the type of content being returned is HTML.
    - **ETag**: "84238dfc8092e5d9c0dac8ef93371a07:1736799080.121134"
        - An identifier for a specific version of the resource. Clients can use this to determine if the resource has changed.
    - **Last-Modified**: Mon, 13 Jan 2025 20:11:20 GMT
        - It indicates the last time the resource was modified. Clients can use this information for caching.

- **Vary**: Accept-Encoding
    - It indicates that the server might return different versions of the resource depending on the Accept-Encoding header sent by the client.
- **Content-Encoding**: gzip
    - Specifies that the content is compressed using gzip, which reduces the size of the response.
- **Content-Length**: 648
    - It indicates the size of the response body when compressed.
- **Cache-Control**: max-age=2882
    - Specifies that the response is fresh for 2882 seconds (about 48 minutes) from the time it was sent, allowing clients to cache the response during this time.
- **Date**: Fri, 18 Apr 2025 16:32:28 GMT
    - The date and time when the response was generated.
- **Connection**: keep-alive
    - It indicates that the server would like to keep the connection open for further requests, improving performance by avoiding the overhead of opening new connections.

Payload:

4bd37348641015472ea1a7d46e6eeb572829c7b1343c5a86b543c6129cb9f92854d71de64ef1669e17c0af26b1dd0f5fa05dbc12f20542f2c4f595b6798b1b415ee71785e7b97928518f3bf847a7e7f9f804f690f405bce95ac5b7f21074722bf561583bfeaee0546f0681

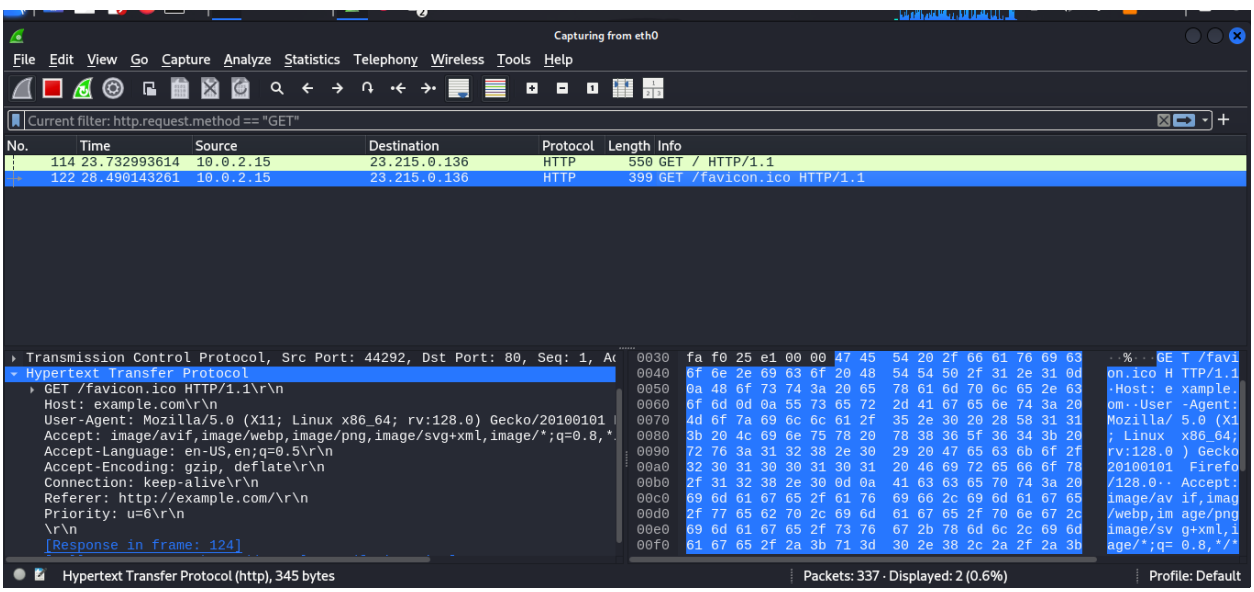
- This hexadecimal string represents the payload data contained in the packet. The content of this payload can be encoded information relevant to the application, connection state, acknowledgments, or QUIC streams, depending on the context of its use. The hexadecimal can be converted to binary to determine the embedded files in the payload whether is PDF, JPG or PNG etc.

Additional Details:

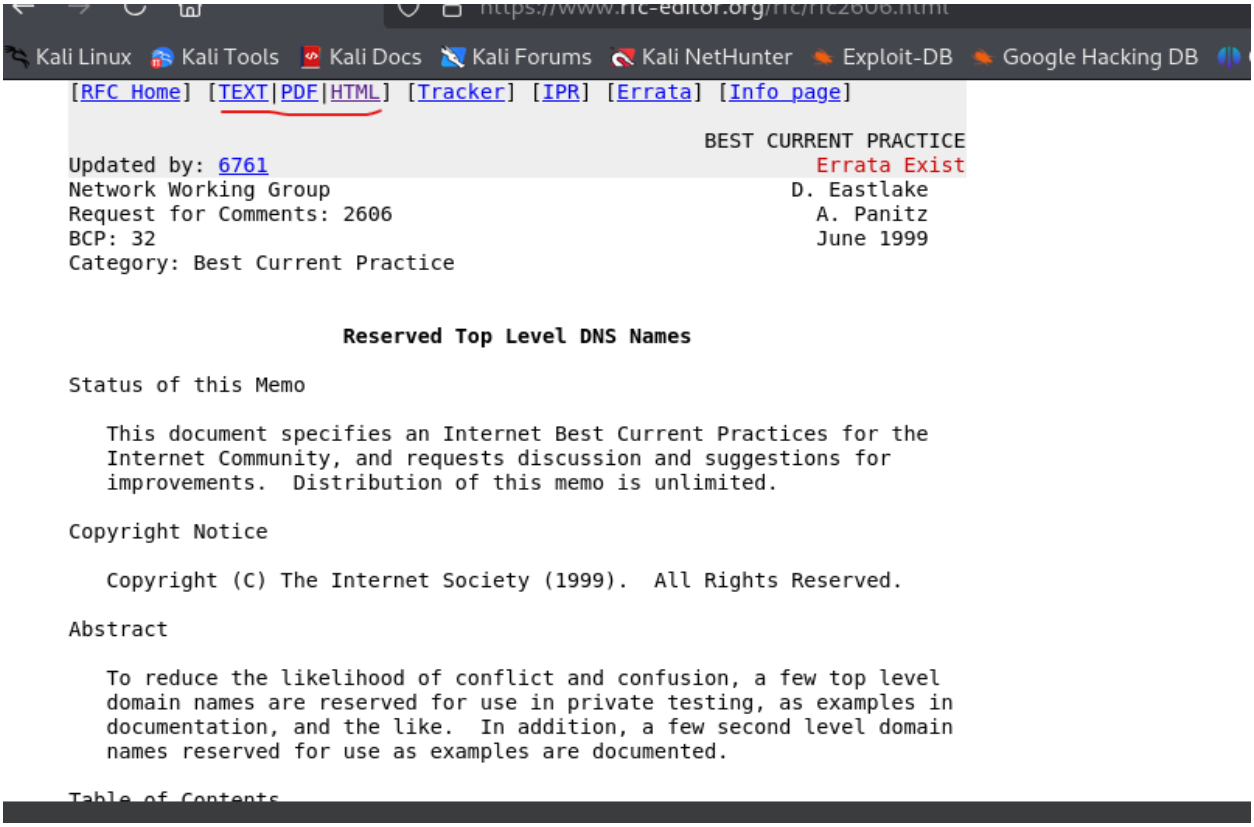
- **Request Information:**
  - **Request URI**: /
    - This indicates that the root resource of the server was requested.
  - **Full Request URI**: http://example.com/
    - Provides the complete address of the requested resource.
- **Content-Encoded Entity Body:**
  - The message notes that the returned content is encoded with gzip. The original size of the content before encoding was 1256 bytes, and after compression, the size is 648 bytes. This demonstrates effective compression, reducing bandwidth usage.

3. Capture Screenshots:

- Take screenshots of:
  - The Wireshark packet details for both the GET request and response.



  - The page displaying the embedded image in the browser.



Part 3: Extract the Image from HTTP Traffic

1. **Extract the Image File:**

- Use the following command to extract the image file from the captured HTTP traffic:

wget https://github.com/frankwxu/digital-forensics-lab/blob/main/Illegal_Possession_Images/lab_files/traffic/image.log

- Once downloaded, you will analyze the contents of image2.log.

2. **Analyze the Extracted Log File:**

- Open the image2.log file and examine its contents.

```
kali@kali: ~
File  Actions  Edit  View  Help
<meta name="viewport" content="width=device-width">

<meta name="description" content="Free hands-on digital forensics labs
for students and faculty - digital-forensics-lab/Illegal_Possession_Images/la
b_files/traffic/image2.log at main · frankwxu/digital-forensics-lab">

<link rel="search" type="application/opensearchdescription+xml" href="/
opensearch.xml" title="GitHub">

<link rel="fluid-icon" href="https://github.com/fluidicon.png" title="Git
Hub">
<meta property="fb:app_id" content="1401488693436528">
<meta name="apple-itunes-app" content="app-id=1477376905, app-argument=ht
tps://github.com/frankwxu/digital-forensics-lab/blob/main/Illegal_Possession_
Images/lab_files/traffic/image2.log" />

<meta name="twitter:image" content="https://opengraph.githubassets.com/
4c98cb8331d9c95224fdeecea81dca07e0e7543d629670c54040e7c8507d6c10/frankwxu/dig
ital-forensics-lab" /><meta name="twitter:site" content="@github" /><meta nam
e="twitter:card" content="summary_large_image" /><meta name="twitter:title" c
ontent="digital-forensics-lab/Illegal_Possession_Images/lab_files/traffic/ima
ge2.log at main · frankwxu/digital-forensics-lab" /><meta name="twitter:descr
iption" content="Free hands-on digital forensics labs for students and facult
y - frankwxu/digital-forensics-lab" />
<meta property="og:image" content="https://opengraph.githubassets.com/4c98c
b8331d9c95224fdeecea81dca07e0e7543d629670c54040e7c8507d6c10/frankwxu/digital-
forensics-lab" /><meta property="og:image:alt" content="Free hands-on digital
 forensics labs for students and faculty - frankwxu/digital-forensics-lab" />
<meta property="og:image:width" content="1200" /><meta property="og:image:hei
ght" content="600" /><meta property="og:site_name" content="GitHub" /><meta p
roperty="og:type" content="object" /><meta property="og:title" content="digit
```

- Identify the bytes corresponding to the image data. You may need to locate the Content-Type header to confirm the image format (e.g., JPEG).

```
<react-app
  app-name="react-code-view"
  initial-path="/frankwxu/digital-forensics-lab/blob/main/Illegal_Possession_
Images/lab_files/traffic/image2.log"
    style="display: block; min-height: calc(100vh - 64px);"
  data-attempted-ssr="true"
  data-ssr="true"
  data-lazy="false"
  data-alternate="false"
  data-data-router-enabled="false"
>
  <script type="application/json" data-target="react-app.embeddedData">{"payl
oad":{"allShortcutsEnabled":false,"fileTree":{"Illegal_Possession_Images/lab_
files/traffic":{"items":[{"name":"basic.log","path":"Illegal_Possession_Image
s/lab_files/traffic/basic.log","contentType":"file"},{"name":"building_202011
08_221645.jpg","path":"Illegal_Possession_Images/lab_files/traffic/building_2
0201108_221645.jpg","contentType":"file"},{"name":"image.html","path":"Illega
l_Possession_Images/lab_files/traffic/image.html","contentType":"file"},{"nam
e":"image.log","path":"Illegal_Possession_Images/lab_files/traffic/image.log"
,"contentType":"file"},{"name":"image2.log","path":"Illegal_Possession_Images
/lab_files/traffic/image2.log","contentType":"file"},{"name":"smtp.pcap","pat
h":"Illegal_Possession_Images/lab_files/traffic/smtp.pcap","contentType":"fil
e"}],"totalCount":6},"Illegal_Possession_Images/lab_files":{"items":[{"name":
"SYN_Flood","path":"Illegal_Possession_Images/lab_files/SYN_Flood","contentTy
pe":"directory"},{"name":"traffic","path":"Illegal_Possession_Images/lab_file
s/traffic","contentType":"directory"},{"name":"wlan_decrypt","path":"Illegal_
Possession_Images/lab_files/wlan_decrypt","contentType":"directory"},{"name":
"f0335017_She_died_in_February_at_the_age_of_74.doc","path":"Illegal_Possessi
on_Images/lab_files/f0335017_She_died_in_February_at_the_age_of_74.doc","cont
entType":"file"}],"totalCount":4},"Illegal_Possession_Images":{"items":[{"nam
e":"USB_image","path":"Illegal_Possession_Images/USB_image","contentType":"di
```


- Use tools like xxd or hexdump to visualize the raw data in the log file.

```
00025dd0: 0a20 2064 6174 612d 6c61 7a79 3d22 6661  .  data-lazy="fa
00025de0: 6c73 6522 0a20 2064 6174 612d 616c 7465  lse".  data-alte
00025df0: 726e 6174 653d 2266 616c 7365 220a 2020  rnate="false".
00025e00: 6461 7461 2d64 6174 612d 726f 7574 6572  data-data-router
00025e10: 2d65 6e61 626c 6564 3d22 6661 6c73 6522  -enabled="false"
00025e20: 0a3e 0a20 200a 2020 3c73 6372 6970 7420  .>.  .  <script
00025e30: 7479 7065 3d22 6170 706c 6963 6174 696f  type="applicatio
00025e40: 6e2f 6a73 6f6e 2220 6461 7461 2d74 6172  n/json" data-tar
00025e50: 6765 743d 2272 6561 6374 2d61 7070 2e65  get="react-app.e
00025e60: 6d62 6564 6465 6444 6174 6122 3e7b 2270  mbeddedData">{"p
00025e70: 6179 6c6f 6164 223a 7b22 616c 6c53 686f  ayload":{"allSho
00025e80: 7274 6375 7473 456e 6162 6c65 6422 3a66  rtcutsEnabled":f
00025e90: 616c 7365 2c22 6669 6c65 5472 6565 223a  alse,"fileTree":
00025ea0: 7b22 496c 6c65 6761 6c5f 506f 7373 6573  {"Illegal_Posses
00025eb0: 7369 6f6e 5f49 6d61 6765 732f 6c61 625f  sion_Images/lab_
00025ec0: 6669 6c65 732f 7472 6166 6669 6322 3a7b  files/traffic":{
00025ed0: 2269 7465 6d73 223a 5b7b 226e 616d 6522  "items":[{"name"
00025ee0: 3a22 6261 7369 632e 6c6f 6722 2c22 7061  :"basic.log","pa
00025ef0: 7468 223a 2249 6c6c 6567 616c 5f50 6f73  th":"Illegal_Pos
00025f00: 7365 7373 696f 6e5f 496d 6167 6573 2f6c  session_Images/l
00025f10: 6162 5f66 696c 6573 2f74 7261 6666 6963  ab_files/traffic
00025f20: 2f62 6173 6963 2e6c 6f67 222c 2263 6f6e  /basic.log","con
00025f30: 7465 6e74 5479 7065 223a 2266 696c 6522  tentType":"file"
00025f40: 7d2c 7b22 6e61 6d65 223a 2262 7569 6c64  },{"name":"build
00025f50: 696e 675f 3230 3230 3131 3038 5f32 3231  ing_20201108_221
00025f60: 3634 352e 6a70 6722 2c22 7061 7468 223a  645.jpg","path":
00025f70: 2249 6c6c 6567 616c 5f50 6f73 7365 7373  "Illegal_Possess
00025f80: 696f 6e5f 496d 6167 6573 2f6c 6162 5f66  ion_Images/lab_f
00025f90: 696c 6573 2f74 7261 6666 6963 2f62 7569  iles/traffic/bui
00025fa0: 6c64 696e 675f 3230 3230 3131 3038 5f32  lding_20201108_2
00025fb0: 3231 3634 352e 6a70 6722 2c22 636f 6e74  21645.jpg","cont
00025fc0: 656e 7454 7970 6522 3a22 6669 6c65 227d  entType":"file"}
00025fd0: 2c7b 226e 616d 6522 3a22 696d 6167 652e  ,{"name":"image.
00025fe0: 6874 6d6c 222c 2270 6174 6822 3a22 496c  html","path":"Il
```

The hexadecimal values show:

1. **Filename**:
    - The sequence building_20201108_645.jpg appears to be a filename, likely an image file, where:
        - building is likely a prefix or part of a descriptive name.
        - 20201108 represents a date formatted as YYYYMMDD (November 8, 2020).
        - 645 is potentially a sequence number or identifier associated with this specific image.

2. **File Extension**:
   - The `.jpg` extension indicates that this file is a JPEG image, which is a widely used format for digital images.
3. **Contextual Portion**:
   - The following part (`, "path":`) suggests that this filename is part of a data structure (possibly in JSON format). It might denote a property named "path", indicating where this image file is located within a file system or web server.

```
┌──(kali㉿kali)-[~]
└─$ xxd image2.log
00000000: 0a0a 0a0a 0a0a 3c21 444f 4354 5950 4520  ......<!DOCTYPE
00000010: 6874 6d6c 3e0a 3c68 746d 6c0a 2020 6c61  html>.<html.  la
00000020: 6e67 3d22 656e 220a 2020 0a20 2064 6174  ng="en".  .  dat
00000030: 612d 636f 6c6f 722d 6d6f 6465 3d22 6175  a-color-mode="au
00000040: 746f 2220 6461 7461 2d6c 6967 6874 2d74  to" data-light-t
00000050: 6865 6d65 3d22 6c69 6768 7422 2064 6174  heme="light" dat
00000060: 612d 6461 726b 2d74 6865 6d65 3d22 6461  a-dark-theme="da
00000070: 726b 220a 2020 6461 7461 2d61 3131 792d  rk".  data-a11y-
00000080: 616e 696d 6174 6564 2d69 6d61 6765 733d  animated-images=
00000090: 2273 7973 7465 6d22 2064 6174 612d 6131  "system" data-a1
000000a0: 3179 2d6c 696e 6b2d 756e 6465 726c 696e  1y-link-underlin
000000b0: 6573 3d22 7472 7565 220a 2020 0a20 203e  es="true".  .  >
000000c0: 0a0a 0a0a 2020 3c68 6561 643e 0a20 2020  ....  <head>.
000000d0: 203c 6d65 7461 2063 6861 7273 6574 3d22   <meta charset="
000000e0: 7574 662d 3822 3e0a 2020 3c6c 696e 6b20  utf-8">.  <link
000000f0: 7265 6c3d 2264 6e73 2d70 7265 6665 7463  rel="dns-prefetc
00000100: 6822 2068 7265 663d 2268 7474 7073 3a2f  h" href="https:/
00000110: 2f67 6974 6875 622e 6769 7468 7562 6173  /github.githubas
00000120: 7365 7473 2e63 6f6d 223e 0a20 203c 6c69  sets.com">.  <li
00000130: 6e6b 2072 656c 3d22 646e 732d 7072 6566  nk rel="dns-pref
00000140: 6574 6368 2220 6872 6566 3d22 6874 7470  etch" href="http
00000150: 733a 2f2f 6176 6174 6172 732e 6769 7468  s://avatars.gith
00000160: 7562 7573 6572 636f 6e74 656e 742e 636f  ubusercontent.co
00000170: 6d22 3e0a 2020 3c6c 696e 6b20 7265 6c3d  m">.  <link rel=
00000180: 2264 6e73 2d70 7265 6665 7463 6822 2068  "dns-prefetch" h
00000190: 7265 663d 2268 7474 7073 3a2f 2f67 6974  ref="https://git
```

NT312 - Basic Networking Skills for Digital Forensics – Lab 3

Lab Title: Lab on Packet Sniffing and Interception & Lab on DNS Spoofing and ARP Poisoning

---

Part One – Web Traffic Capture Using Login Page

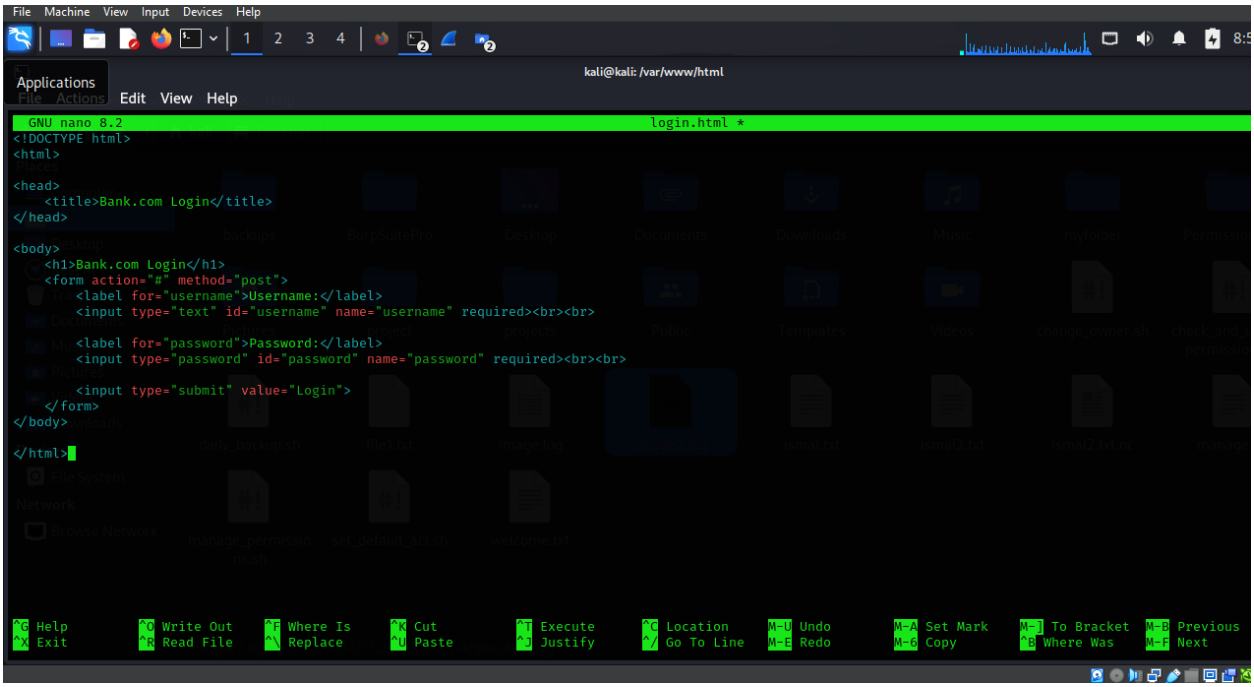
Step 1: Create a Basic HTML Login Form

1. Open your terminal in Kali Linux.

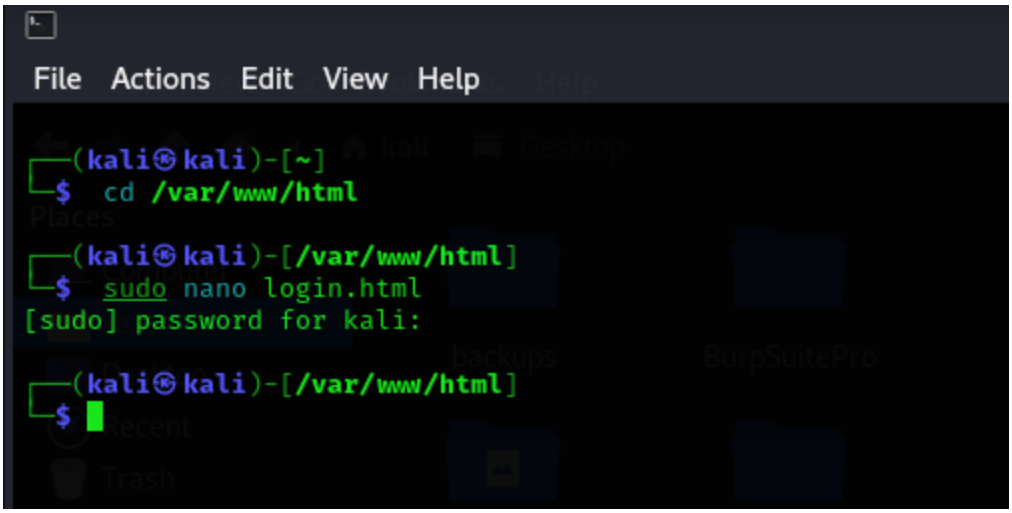
2. Navigate to the default web directory: cd /var/www/html

3. Create a new HTML file (e.g., login.html): sudo nano login.html

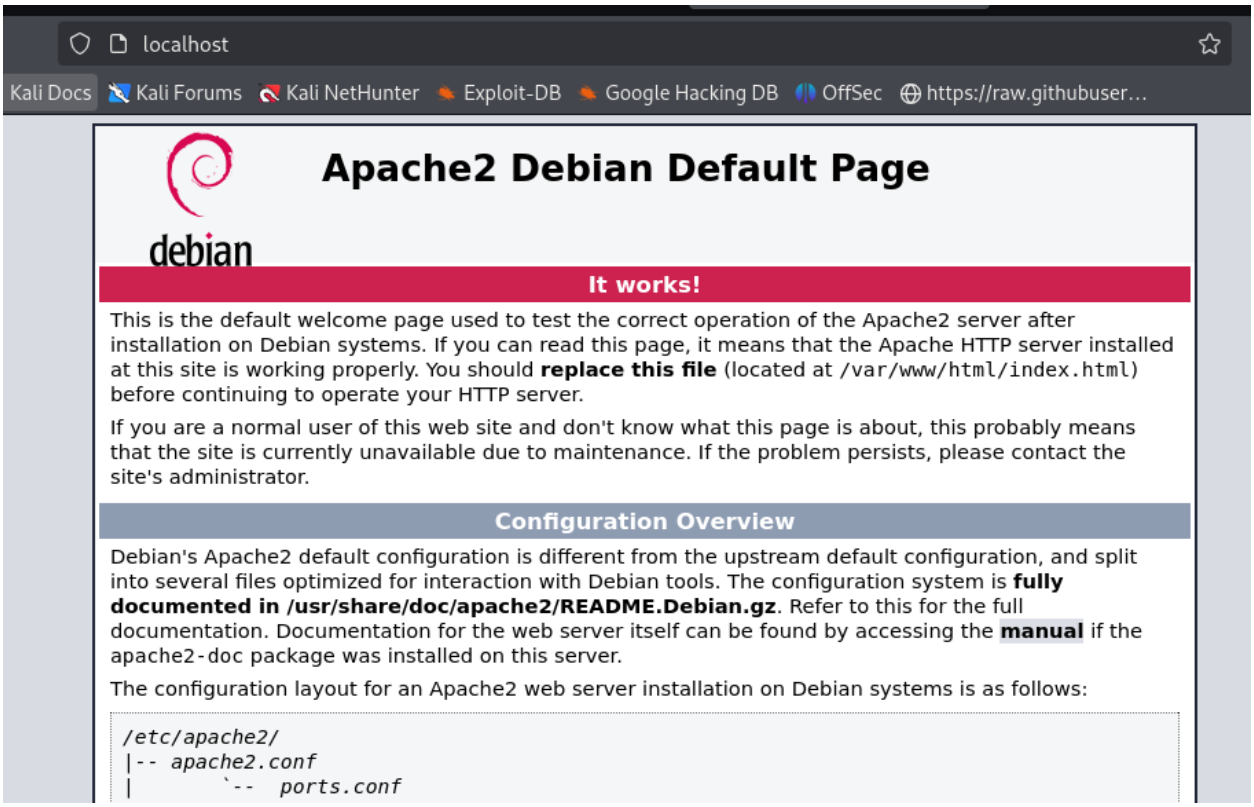
4. Paste the following HTML code:



5. Save and exit (CTRL + O, then Enter, then CTRL + X).



Step 2: Start the Apache Web Server sudo systemctl start apache2

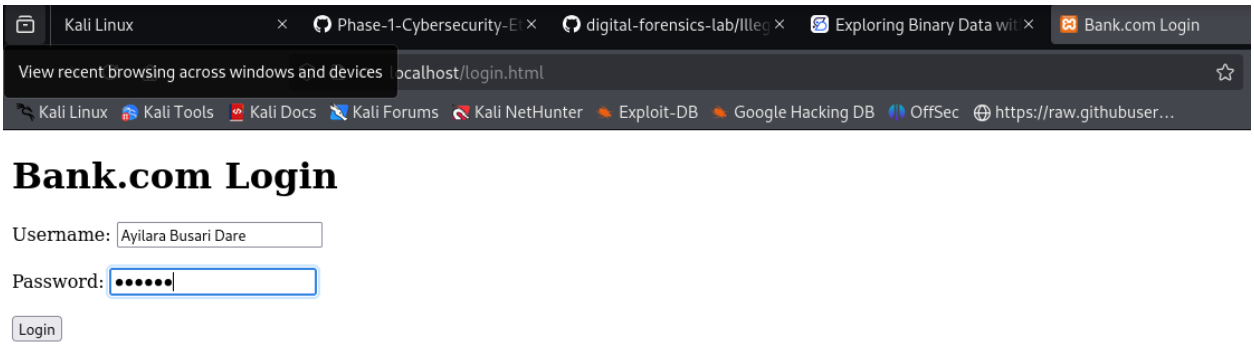


Step 3: Access the Login Page in a Browser

1. Open Firefox or another browser in Kali.

2. Navigate to: http://127.0.0.1t/login.html

3. Enter test credentials (e.g., username: test, password: 1234) and click Login.



Step 4: Capture the Login Traffic with Wireshark

1. Open Wireshark.

2. Choose the network interface (e.g., eth0, wlan0, or lo for localhost).

3. Start capturing packets.

4. While capturing, go back to the browser and submit the login form.

5. Return to Wireshark and stop the capture.

The packet was filter using http.request.method == "POST" and look for packet that contains username and password.

Part Two – Log File Analysis for Spoofing Attacks

Step 1: Download the Log Files

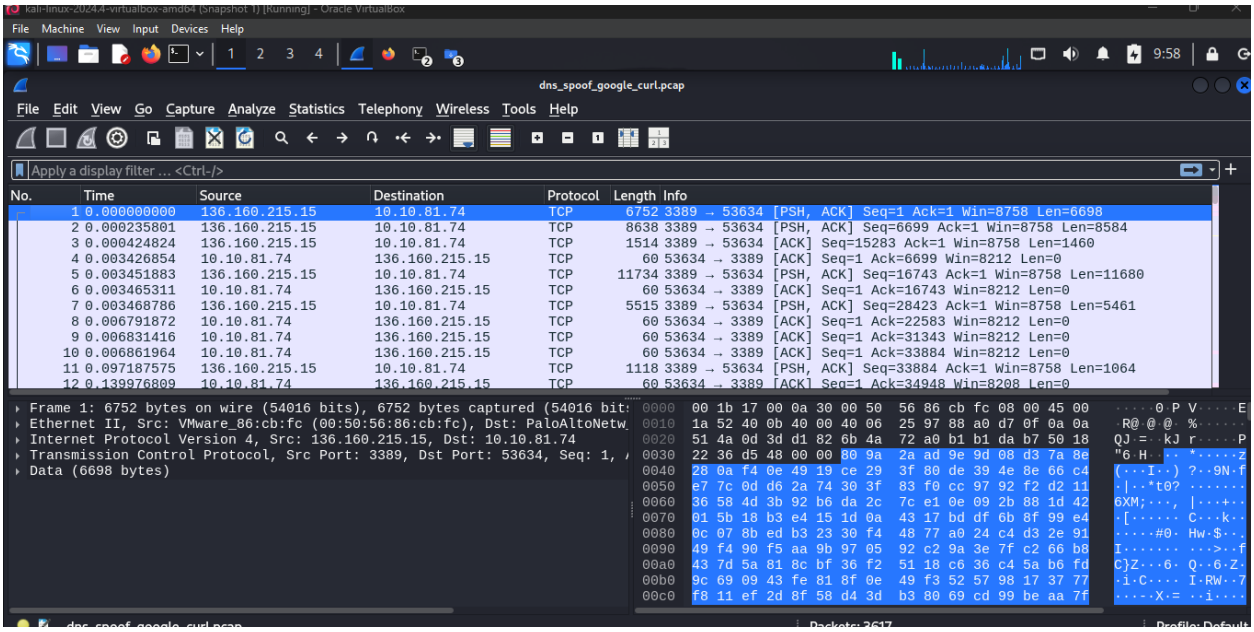
- DNS Spoofing Log: Download from Google Drive
- ARP Poisoning Log: Download from Google Drive

Step 2: Open the Log Files in Wireshark

1. Launch Wireshark.

2. Go to File > Open.

3. Select the downloaded .pcap files one at a time.



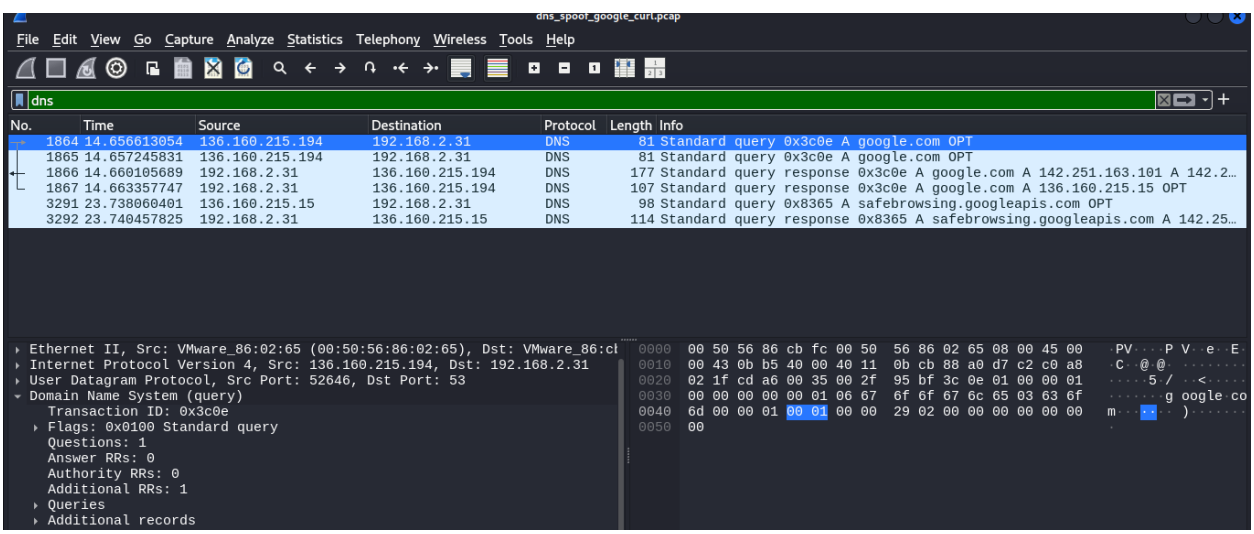
Step 3: Analyze DNS Spoofing Log

1. Apply the following display filters:

2. To view DNS queries:dns

3. To detect spoofing, look for:

- Multiple responses to the same query
- Responses from unexpected IPs
- Inconsistent or fake DNS answers



The addresses you received appear to be consistent with Google's allocation of IP addresses. By performing these checks using reliable tools, you can verify the authenticity of DNS responses.

Step 4: Analyze ARP Poisoning Log

1. Apply ARP filters: arp

2. Look for:

ARP replies without requests (gratuitous ARPs)

Multiple MAC addresses claiming to be the same IP

Inconsistent MAC-IP pairings