

INTERNATIONAL CYBERSECURITY AND DIGITAL FORENSICS ACADEMY

INT312 - Basic Networking Skills for Digital Forensics

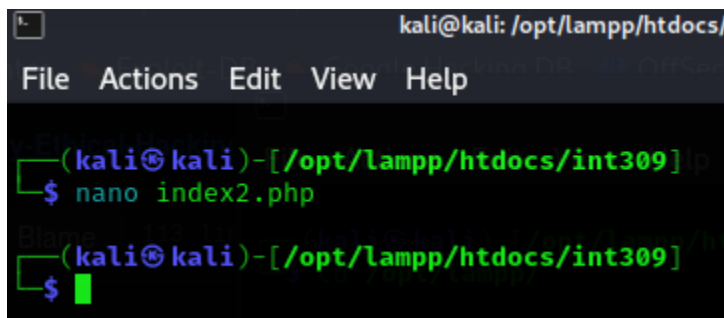
Ayilara Busari Dare
IDEAS/24/28133

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

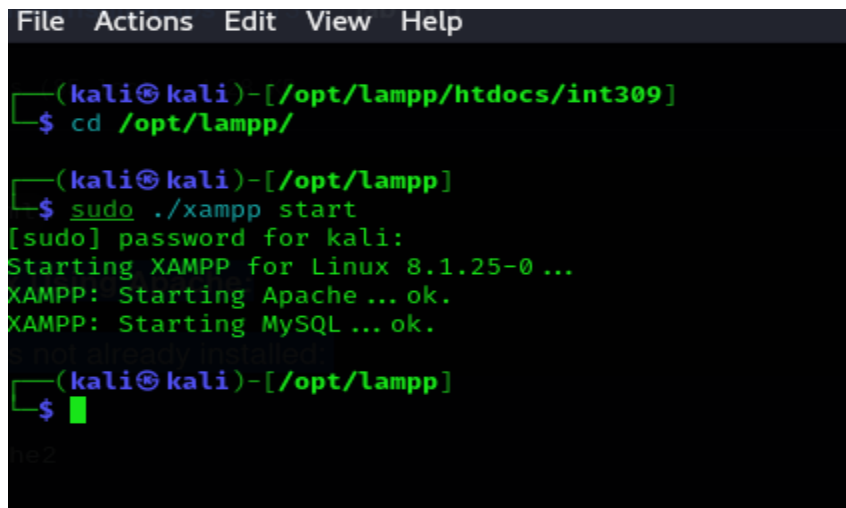
Part 1: Create a Simple Website



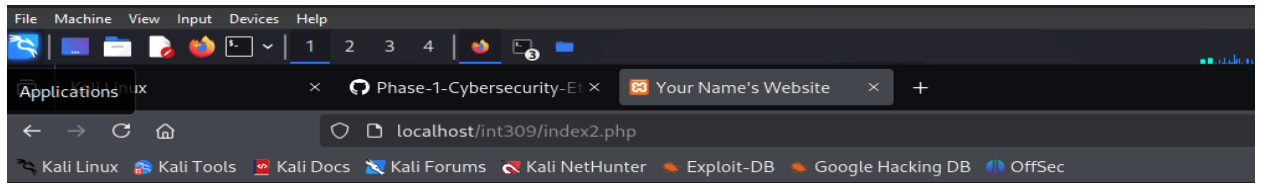
```
kali@kali: /opt/lampp/htdocs/
File Actions Edit View Help
(kali@kali)-[/opt/lampp/htdocs/int309]
$ nano index2.php
(kali@kali)-[/opt/lampp/htdocs/int309]
$
```

2. Hosting the Website on Linux Using Apache:

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



```
File Actions Edit View Help
(kali@kali)-[/opt/lampp/htdocs/int309]
$ cd /opt/lampp/
(kali@kali)-[/opt/lampp]
$ sudo ./xampp start
[sudo] password for kali:
Starting XAMPP for Linux 8.1.25-0 ...
XAMPP: Starting Apache ... ok.
XAMPP: Starting MySQL ... ok.
not already installed
(kali@kali)-[/opt/lampp]
$
```



Hello, My Name is [Your Name]

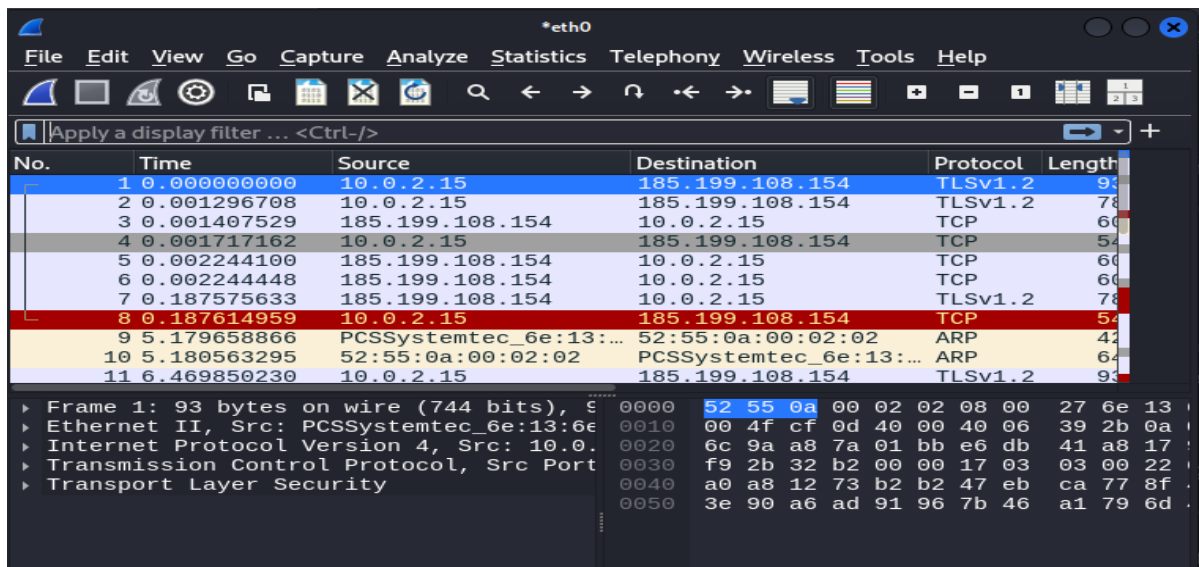
Part 2: Capture Network Traffic

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>).



1. Information to Capture:

- **The website displays your name.**



-
- The image displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains icons for various functions like opening files, saving, and zooming. The main window is divided into three panes:
- Packet List:** Shows a list of captured packets. The selected packet is 25, a TLSv1.2 Application Data packet from 185.199.111.133 to 10.0.2.15.
 - Packet Details:** Shows the hierarchical structure of the selected packet. It includes Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol. The TCP segment length is 39 bytes.
 - Packet Bytes:** Shows the raw data of the selected packet in hexadecimal and ASCII format.
- The status bar at the bottom indicates "eth0: <live capture in progress>" and "Packets: 35".

- 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 it has received cumulated data and is ready for the next segment.

Client sends seq=1 and tcp segment length=39

The image shows a Wireshark packet capture of a TCP session. The packet list pane shows several packets, with packet 3 highlighted. The packet details pane for packet 3 shows the following information:

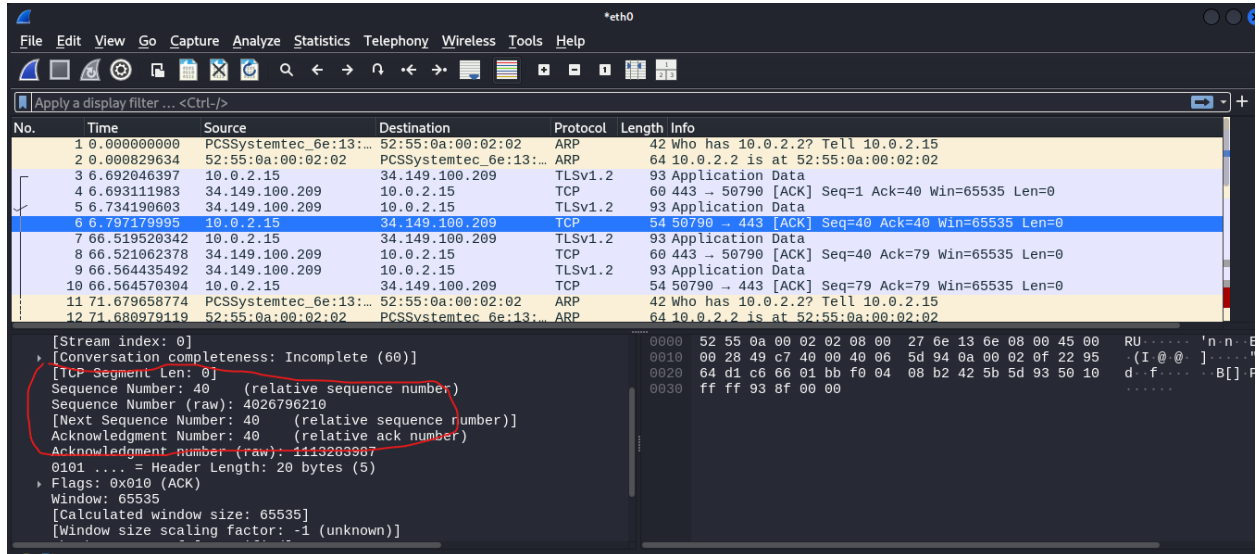
- [Stream index: 0]
- [Conversation completeness: Incomplete (60)]
- [TCP Segment Len: 39]
- Sequence Number: 1 (relative sequence number)
- [Next Sequence Number: 40 (relative sequence number)]
- Acknowledgment Number: 1 (relative ack number)
- Acknowledgment number (raw): 1113283948
- 0101 = Header Length: 20 bytes (5)
- Flags: 0x018 (PSH, ACK)
- Window: 65535
- [Calculated window size: 65535]
- [Window size scaling factor: -1 (unknown)]

The packet bytes pane shows the raw data of the packet, including the TCP header and application data.

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.

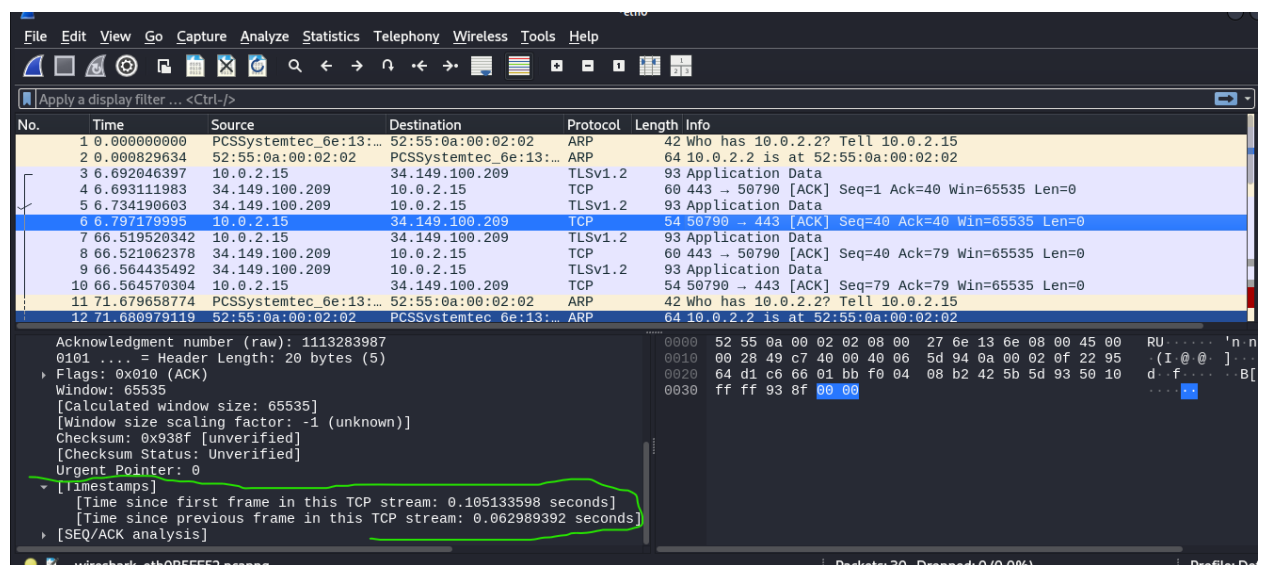
The next segment the client sends is seq=40 and the length is now 0 bytes. In turn, the server responds with ack=40. This cycle continues until the end of the TCP session.

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

Frame 80: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0	0000	08 00 27 6e 13 6e 52 55 0a 00 02 02 08 00 45 00	...
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)	0010	00 28 02 19 00 00 00 06 07 ea 12 a5 f2 19 0a 00	(b @
Destination: PCSSystemtec_6e:13:6e (08:00:27:6e:13:6e)	0020	02 0f 01 bb ac 72 54 39 97 a6 9e 1f ff f9 50 11	...rt9
Source: 52:55:0a:00:02:02 (52:55:0a:00:02:02)	0030	ff ff 66 df 00 00 00 00 00 00 00 00 00 00	...f
Type: IPv4 (0x0800)			
[Stream index: 0]			
Padding: 000000000000			
Internet Protocol Version 4, Src: 18.165.242.25, Dst: 10.0.2.15			
Transmission Control Protocol, Src Port: 443, Dst Port: 44146, Seq: 4517			

- **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

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)	0300	fa f0 63 57 00 00 50 4f 53 54 20 2f 73 2f 77 72	cw PO ST /s/wr
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 172.217.168.163	0400	83 2f 83 67 6f 20 48 54 54 50 2f 31 2e 31 0d 0a	8/cgo HT TP/1.1
Transmission Control Protocol, Src Port: 38362, Dst Port: 80, Seq: 1, Ack: 38362	0500	48 6f 73 74 3a 20 6f 2e 70 0b 69 2e 67 6f 6f 67	Host: o.pki.goog
Hypertext Transfer Protocol	0600	0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f	User-Agent: Mo
POST /s/wr3/cgo HTTP/1.1	0700	74 69 6c 6c 61 2f 35 2e 30 20 28 58 31 31 3b 20	zilla/5.0 (X11;
Request Method: POST	0800	4c 69 6e 75 78 20 78 38 36 5f 36 34 3b 20 72 76	Linux x86_64; rv
Request URI: /s/wr3/cgo	0900	3a 31 32 38 2e 30 29 20 47 65 63 6b 6f 2f 32 30	:128.0) Gecko/20
Request Version: HTTP/1.1	0a00	31 30 30 31 30 31 20 46 69 72 65 66 6f 78 2f 31	100101 Firefox/1
Host: o.pki.goog	0b00	32 38 2e 30 0d 0a 41 63 63 65 70 74 3a 20 2a 2f	28.0-Accept: /*
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/1.1	0c00	2a 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61	*-Accept-Language
Accept: */*	0d00	67 65 3a 20 65 6e 2d 55 53 2c 65 6e 3b 71 3d 30	ge: en-US,en;q=0
Accept-Language: en-US,en;q=0.5	0e00	2e 35 0d 0a 41 63 63 65 70 74 2d 45 6e 63 6f 64	.5-Accept-Encode
Accept-Encoding: gzip, deflate	0f00	69 6e 67 3a 20 67 7a 69 70 2c 20 64 65 66 6c 61	ing: gzip, defla

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help						
Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
4	0.180523992	10.0.2.15	34.223.124.45	TCP	74	33240 → 80 [SYN] Seq=0 Win=64240
5	0.487332269	34.223.124.45	10.0.2.15	TCP	60	80 → 33240 [SYN, ACK] Seq=0 Ack=1
6	0.487332732	34.223.124.45	10.0.2.15	TCP	60	80 → 33226 [SYN, ACK] Seq=0 Ack=1
7	0.487397556	10.0.2.15	34.223.124.45	TCP	54	33240 → 80 [ACK] Seq=1 Ack=1 Win=0
8	0.488426242	10.0.2.15	34.223.124.45	TCP	54	33226 → 80 [ACK] Seq=1 Ack=1 Win=0
9	5.239478712	PCSSystemtec_6e:13:...	52:55:0a:00:02:03	ARP	42	Who has 10.0.2.3? Tell 10.0.2.15
10	5.240200336	52:55:0a:00:02:03	PCSSystemtec_6e:13:...	ARP	64	10.0.2.3 is at 52:55:0a:00:02:03
11	5.491667181	10.0.2.15	34.223.124.45	TCP	54	33226 → 80 [FIN, ACK] Seq=1 Ack=1
12	5.491901704	10.0.2.15	34.223.124.45	TCP	54	33240 → 80 [FIN, ACK] Seq=1 Ack=1
13	5.492917911	34.223.124.45	10.0.2.15	TCP	60	80 → 33226 [ACK] Seq=1 Ack=2 Win=0
14	5.492918332	34.223.124.45	10.0.2.15	TCP	60	80 → 33240 [ACK] Seq=1 Ack=2 Win=0
15	5.852902205	34.223.124.45	10.0.2.15	TCP	60	80 → 33226 [FIN, ACK] Seq=1 Ack=2

▶ Frame 7: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface	0000	52 55 0a 00 02 02 08 00 27 6e
▶ Ethernet II, Src: PCSSystemtec_6e:13:6e (08:00:27:6e:13:6e), Dst: 52:55:0a:00:02:03	0010	00 28 f9 3f 40 00 40 06 96 75
▶ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 34.223.124.45	0020	7c 2d 81 d8 00 50 7b ac ad 6f
▶ Transmission Control Protocol, Src Port: 33240, Dst Port: 80, Seq: 1, Ack: 1, Win: 0, Len: 0	0030	fa f0 ab 35 00 00

Source Port: 33240	
Destination Port: 80	
[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)	

- **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

The image shows a Wireshark network traffic capture. The top pane displays a list of 12 packets. The bottom pane shows the detailed view of the selected packet (No. 7), which is a TCP acknowledgment.

No.	Time	Source	Destination	Protocol	Length
1	0.000000000	10.0.2.15	10.0.2.3	DNS	
2	0.139102594	10.0.2.3	10.0.2.15	DNS	
3	0.146687040	10.0.2.15	10.0.2.3	DNS	
4	0.191597986	10.0.2.3	10.0.2.15	DNS	
5	0.194172057	10.0.2.15	140.82.121.3	TCP	
6	0.435915611	140.82.121.3	10.0.2.15	TCP	
7	0.436089170	10.0.2.15	140.82.121.3	TCP	
8	0.439435446	10.0.2.15	140.82.121.3	TLSv1.3	
9	0.441672201	140.82.121.3	10.0.2.15	TCP	
10	0.643707568	140.82.121.3	10.0.2.15	TLSv1.3	29
11	0.643884887	10.0.2.15	140.82.121.3	TCP	
12	0.645177524	140.82.121.3	10.0.2.15	TLSv1.3	6

Detailed View of Packet 7 (TCP):

- Acknowledgment number (raw): 9152002
- 0101 = Header Length: 20 bytes (5)
- Flags: 0x010 (ACK)
- Window: 64240
- [Calculated window size: 64240]
- [Window size scaling factor: -2 (no window scaling used)]
- Checksum: 0x117f [unverified]
- [Checksum Status: Unverified]
- Urgent Pointer: 0
- [Timestamps]
 - [Time since first frame in this TCP stream: 0.241917113 seconds]
 - [Time since previous frame in this TCP stream: 0.000173559 seconds]
- [SEQ/ACK analysis]

File: wireshark_eth0TLYG52.pcapng

- 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

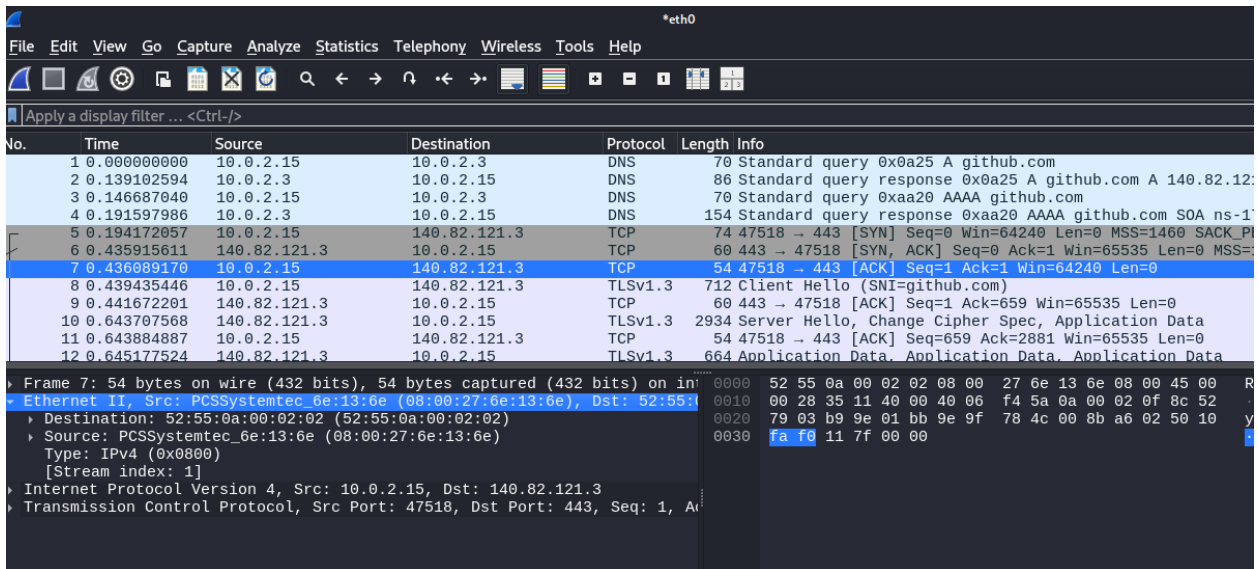
The image shows a Wireshark interface with a packet capture list and a packet details pane. The packet list shows 12 packets. Packet 7 is selected, and the details pane shows the structure of a TCP packet. The source and destination IP addresses are highlighted with a yellow circle.

No.	Time	Source	Destination	Protocol
1	0.000000000	10.0.2.15	10.0.2.3	DNS
2	0.139102594	10.0.2.3	10.0.2.15	DNS
3	0.146687040	10.0.2.15	10.0.2.3	DNS
4	0.191597986	10.0.2.3	10.0.2.15	DNS
5	0.194172057	10.0.2.15	140.82.121.3	TCP
6	0.435915611	140.82.121.3	10.0.2.15	TCP
7	0.436089170	10.0.2.15	140.82.121.3	TCP
8	0.439435446	10.0.2.15	140.82.121.3	TLSv1.3
9	0.441672201	140.82.121.3	10.0.2.15	TCP
10	0.643707568	140.82.121.3	10.0.2.15	TLSv1.3
11	0.643884887	10.0.2.15	140.82.121.3	TCP
12	0.645177524	140.82.121.3	10.0.2.15	TLSv1.3

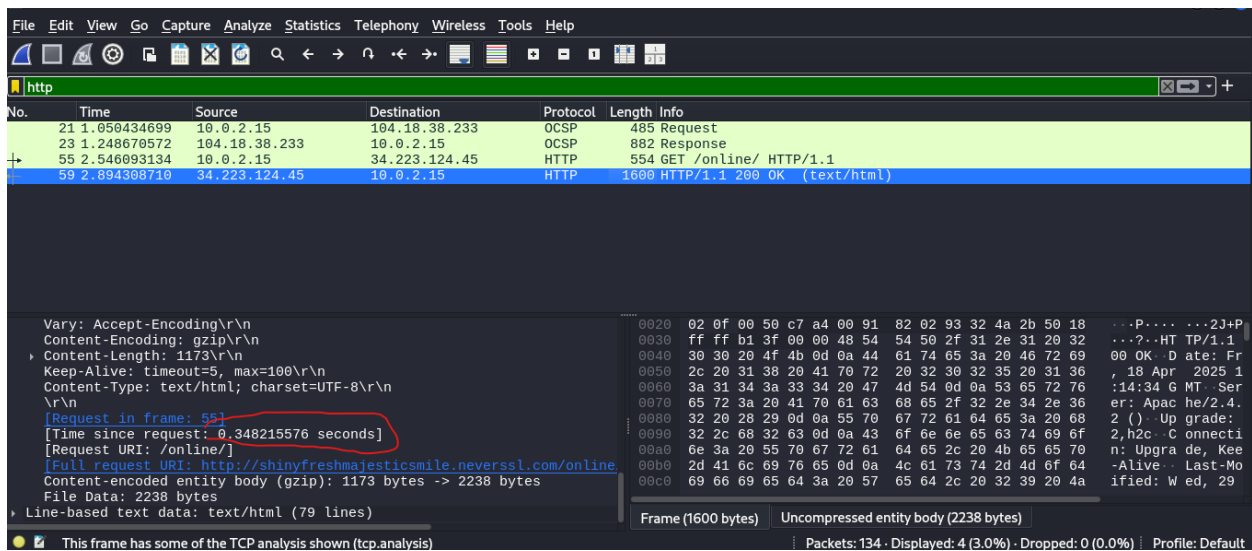
```

    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x3511 (13585)
    010. .... = Flags: 0x2, Don't fragment
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: TCP (6)
    Header Checksum: 0xf45a [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.0.2.15
    Destination Address: 140.82.121.3
    [Stream index: 1]
    Transmission Control Protocol, Src Port: 47518, Dst Port: 443, Seq: 1, /
  
```

- 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).

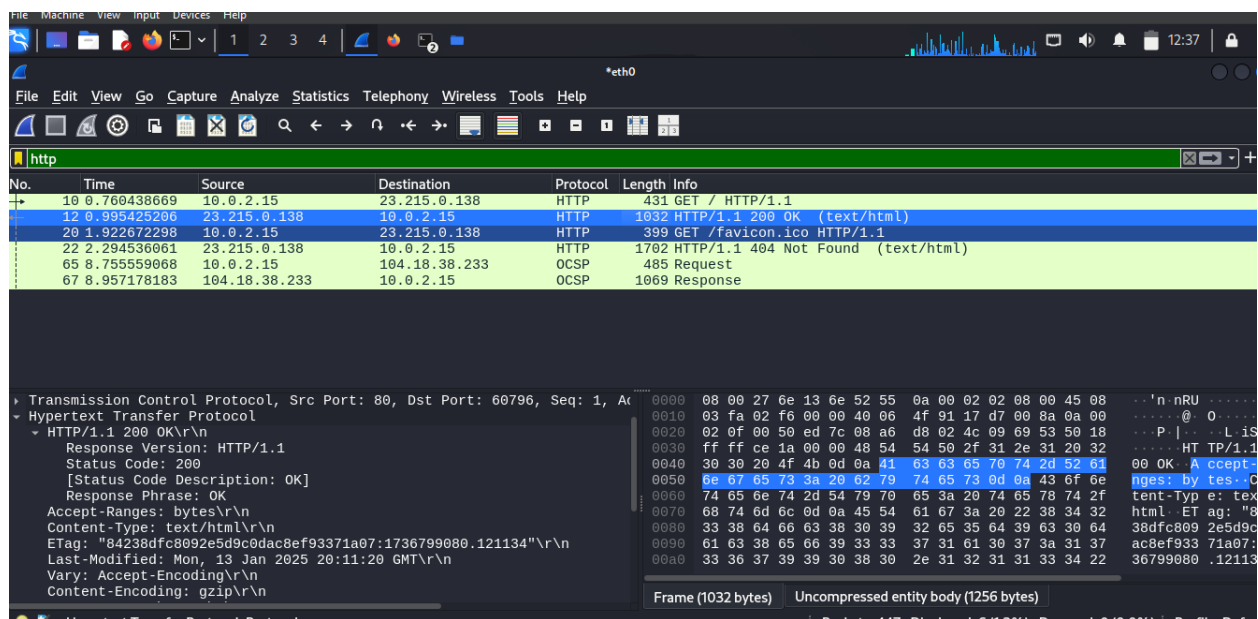
- Start capturing packets before making an HTTP request to a website that contains an embedded image.

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.

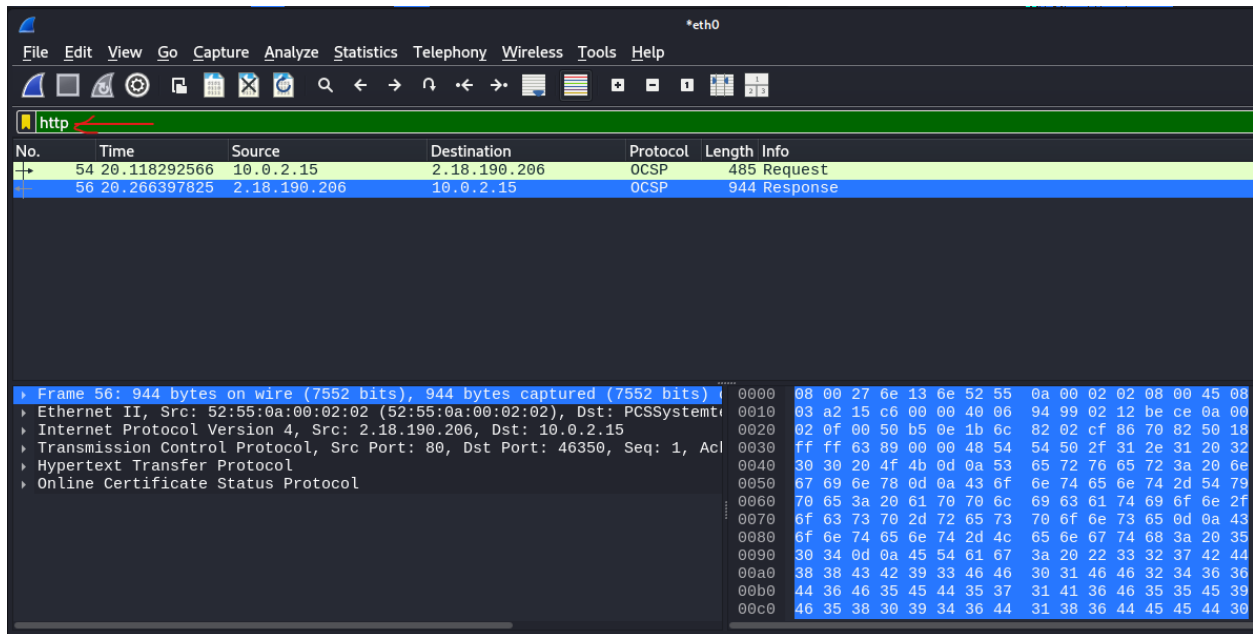


The Wireshark was open to capture traffic on the eth0 network, after this an http request was made using example.com which contain an embedded file (PDF), after the web page load and some traffic were captured by Wireshark, it then stop for analysis

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:

4bd37348641015472ea1a7d46e6eeb572829c7b1343c5a86b543c6129cb9f92854d71de64ef1669e17c0af26b1dd0f5fa05dbc12f20542f2c4f595b6798b1b415ee71785e7b97928518f3bf847a7e7f9f804f690f405bce95ac5b7f21074722bf561583bfeaae0546f0681

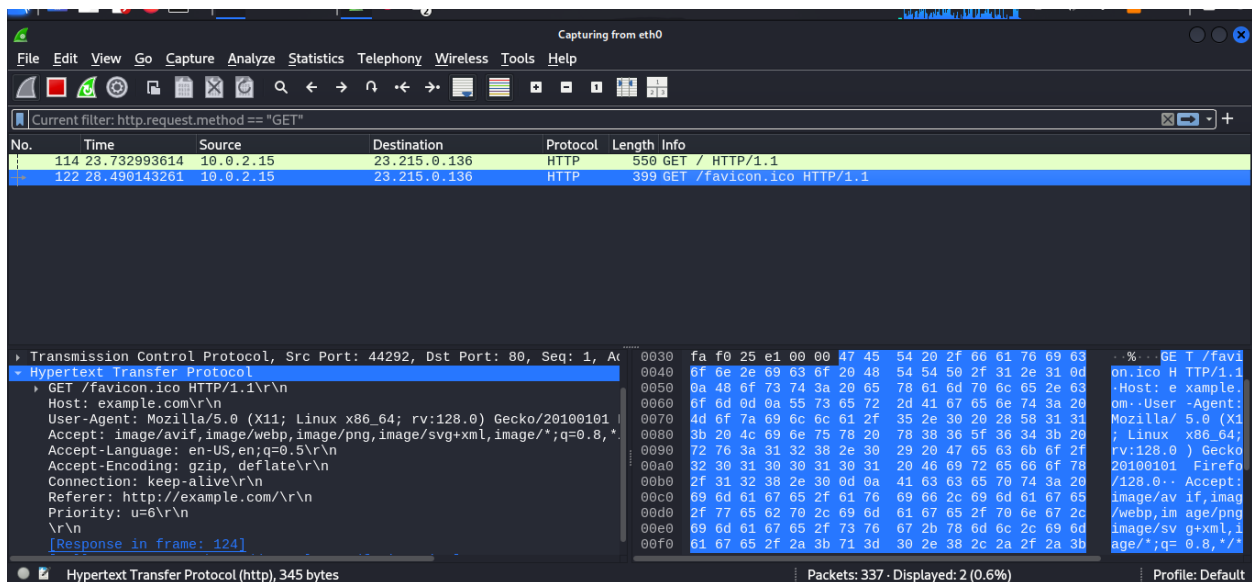
- This hexadecimal string represents the payload data contained in the packet. The content of this payload can be encoded information relevant to the application, connection state, acknowledgments, or QUIC streams, depending on the context of its use. The hexadecimal can be converted to binary to determine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

https://www.rfc-editor.org/rfc/rfc2606.html

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

[RFC Home] [TEXT] [PDF] [HTML] [Tracker] [IPR] [Errata] [Info page]

Updated by: 6761	BEST CURRENT PRACTICE
Network Working Group	Errata Exist
Request for Comments: 2606	D. Eastlake
BCP: 32	A. Panitz
Category: Best Current Practice	June 1999

Reserved Top Level DNS Names

Status of this Memo

This document specifies an Internet Best Current Practices for the Internet Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

To reduce the likelihood of conflict and confusion, a few top level domain names are reserved for use in private testing, as examples in documentation, and the like. In addition, a few second level domain names reserved for use as examples are documented.

Table of Contents

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

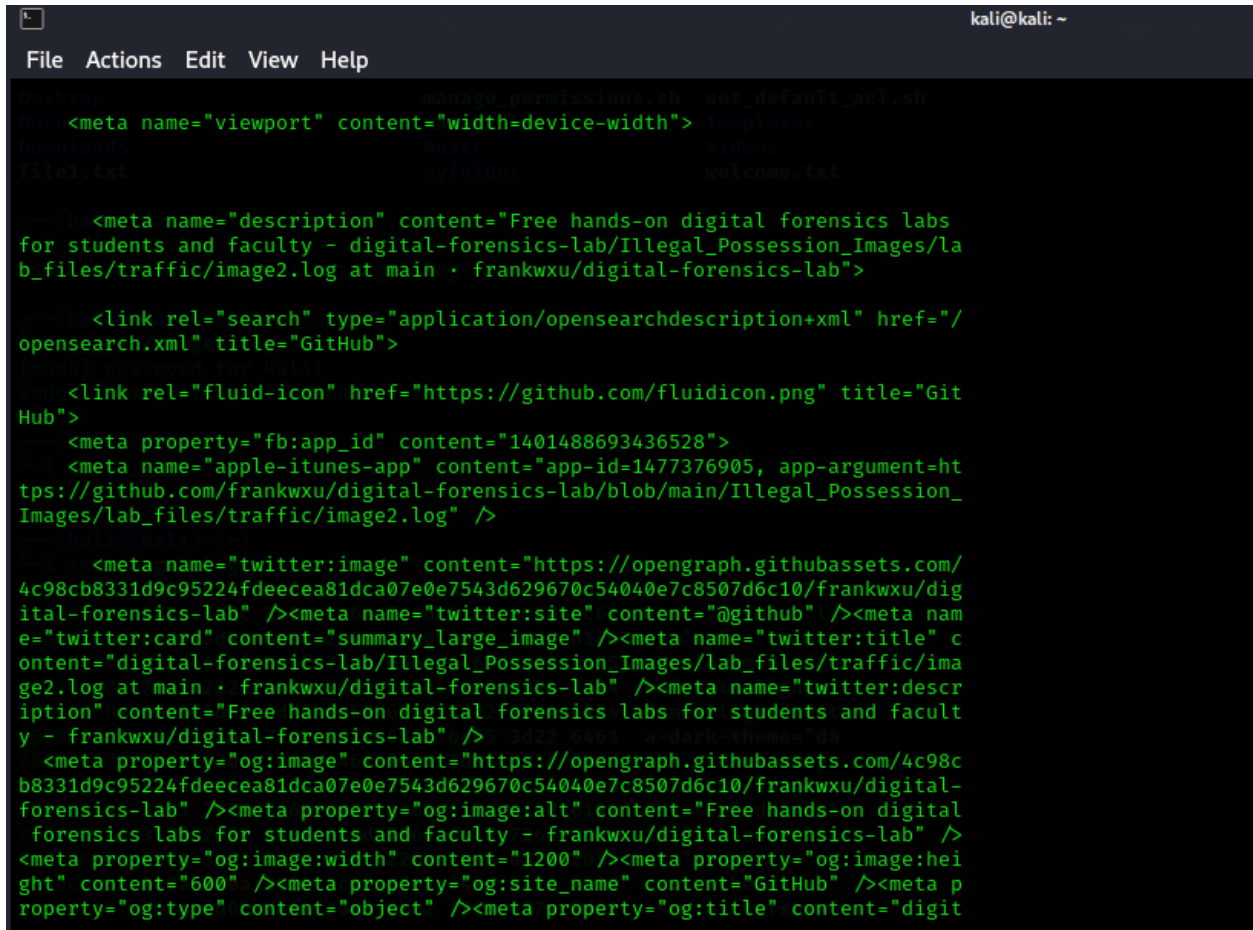
```
(kali@kali)-[~]
$ wget https://github.com/frankwxu/digital-forensics-lab/blob/main/Illegal_Possession_Images/lab_files/traffic/image2.log
--2025-04-20 05:56:39-- https://github.com/frankwxu/digital-forensics-lab/blob/main/Illegal_Possession_Images/lab_files/traffic/image2.log
Resolving github.com (github.com) ... 140.82.121.3
Connecting to github.com (github.com)|140.82.121.3|:443 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: unspecified [text/html]
Saving to: 'image2.log'

image2.log  [100%] 212.17K  117KB/s  in 1.8s
2025-04-20 05:56:42 (117 KB/s) - 'image2.log' saved [217261]
```

- 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

Desktop  manage_permissions.sh  set_default_acl.sh
Downloads  templates
file1.txt  music  Videos
myfolder  welcome.txt

<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/lab_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).

sudo ./images_log_command_not_found

```
<react-app [all] [1-1]
  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"0a 0a0a 3c21 444f 4354 5950 4520 .....<!DOCTYPE
  data-lazy="false" 3e0a 3c68 746d 6c0a 2020 6c61 html>,<html. la
  data-alternate="false" 220a 2020 0a20 2064 6174 ng="en", . . dat
  data-data-router-enabled="false" 6465 3d22 6175 a-color-mode="au
> 000000: 746f 2220 6461 7461 2d6c 6967 6874 2d74 to" data-light-t
000001: 6865 6d65 3d22 6c69 6768 7422 2064 6174 home="light" dat
<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.

```
File Actions Edit View Help
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-a1ly-
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
```

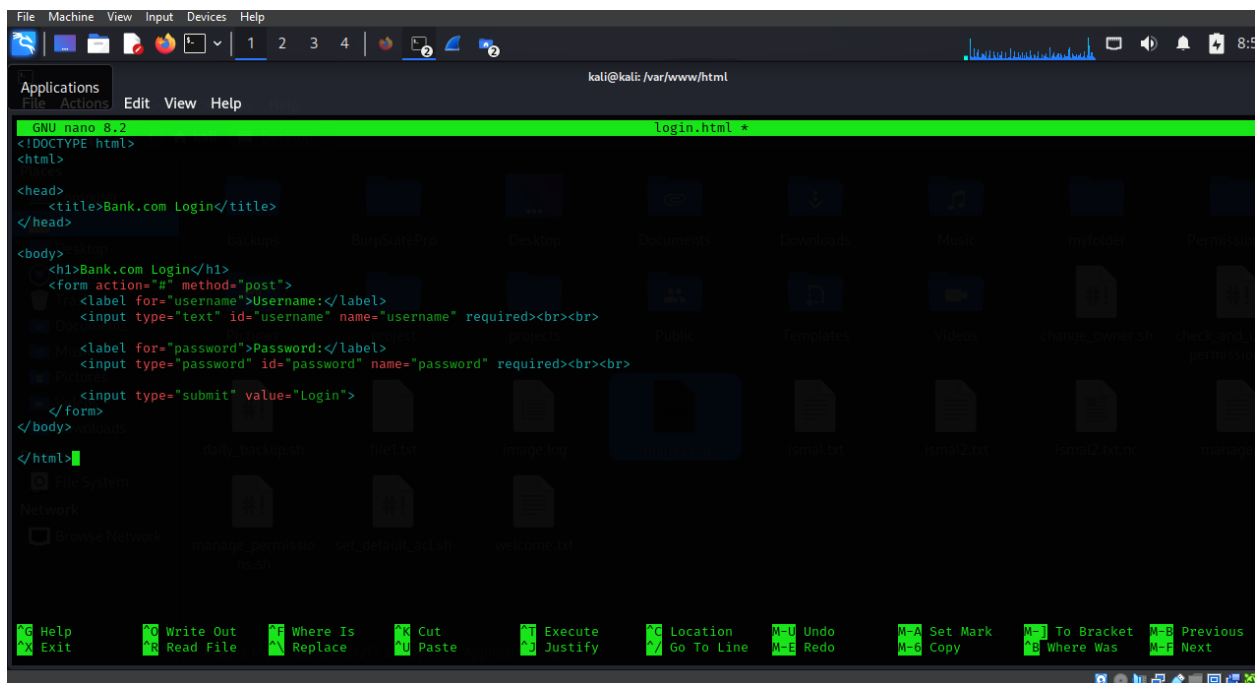
NT312 - Basic Networking Skills for Digital Forensics – Lab 3

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:

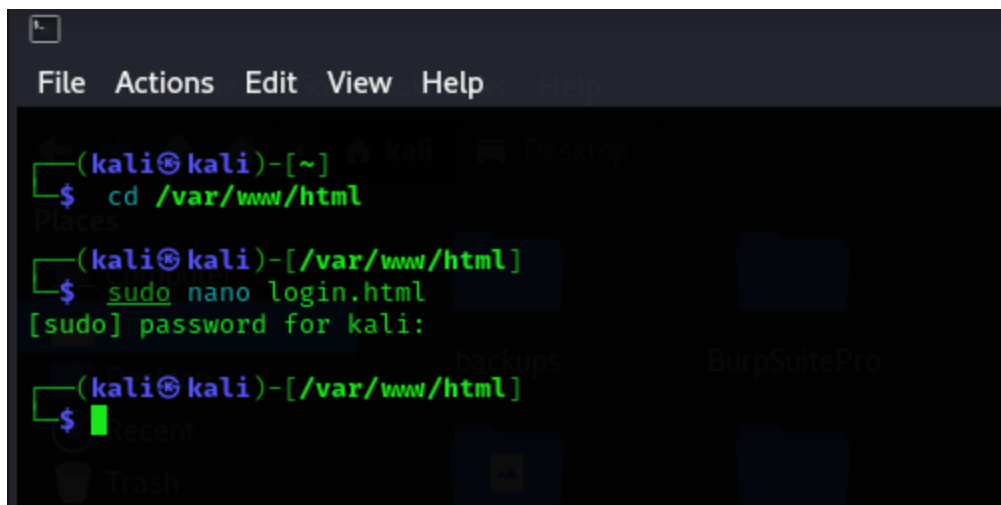


```
File Machine View Input Devices Help
Applications
File Actions Edit View Help
GNU nano 8.2 login.html *
<!DOCTYPE html>
<html>

<head>
  <title>Bank.com Login</title>
</head>

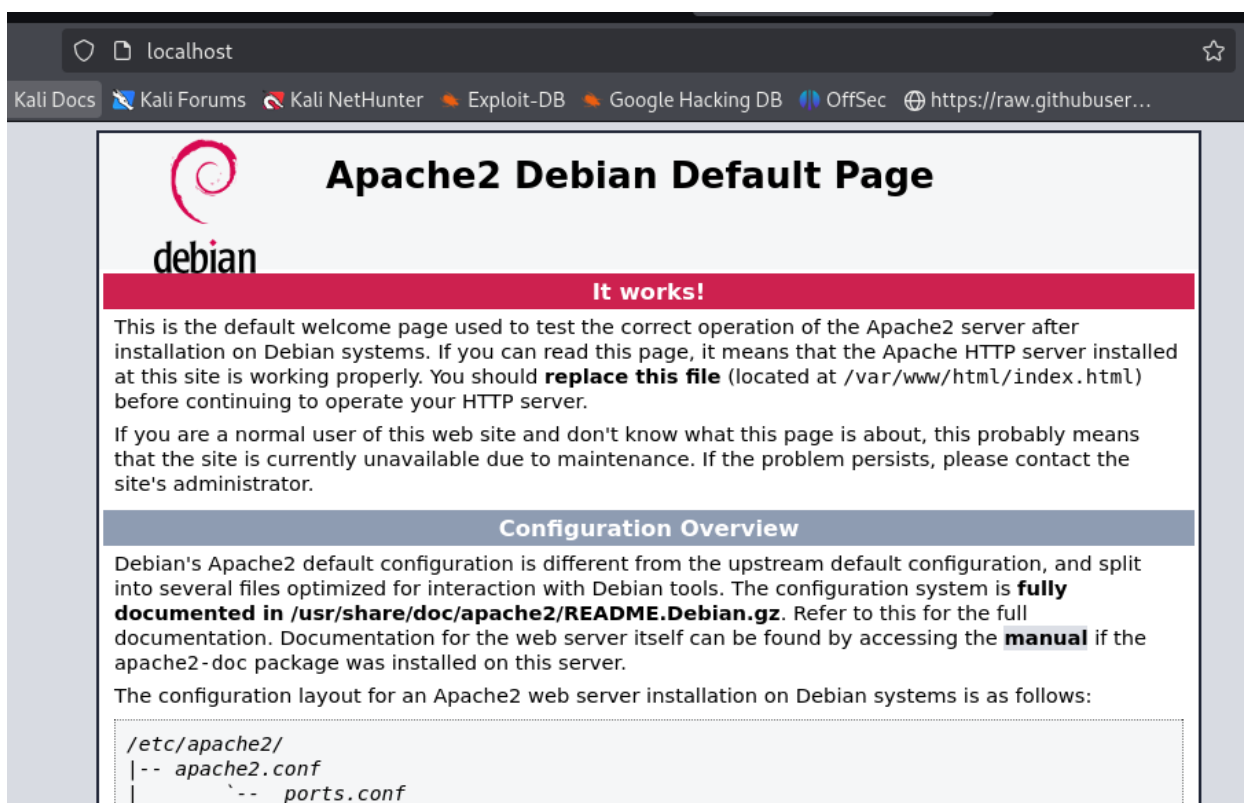
<body>
  <h1>Bank.com Login</h1>
  <form action="#" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required<br><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required<br><br>
    <input type="submit" value="Login">
  </form>
</body>
</html>
Help Write Out Where Is Cut Execute Location M-U Undo M-A Set Mark M-J To Bracket M-B Previous
Exit Read File Replace Paste Justify Go To Line M-E Redo M-G Copy M-K Where Was M-F Next
```

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



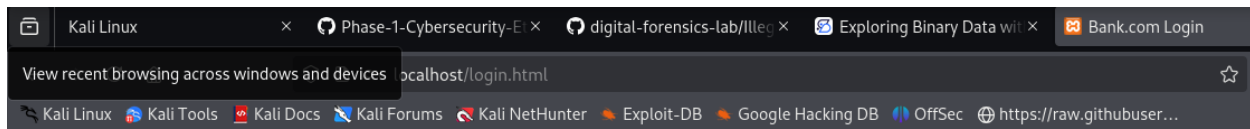
```
(kali㉿kali)-[~]
$ cd /var/www/html
(kali㉿kali)-[/var/www/html]
$ sudo nano login.html
[sudo] password for kali:
(kali㉿kali)-[/var/www/html]
$
```

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://127.0.0.1/login.html`
3. Enter test credentials (e.g., username: test, password: 1234) and click Login.



Bank.com Login

Username:

Password:

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

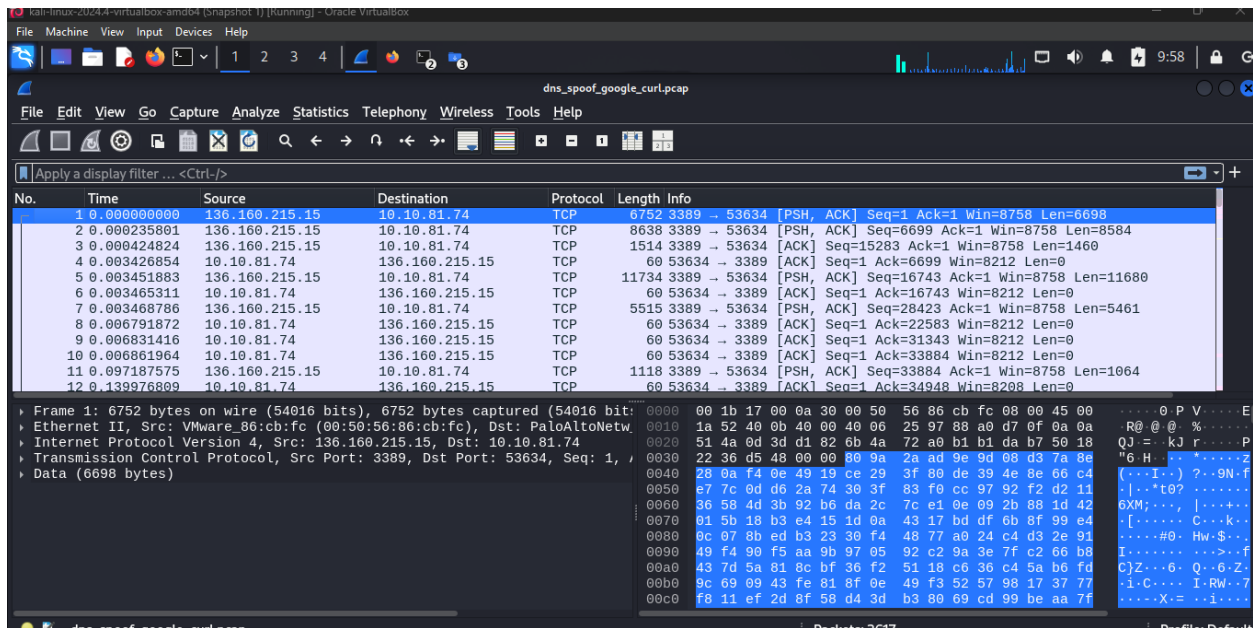
Step 1: Download the Log Files

- 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.



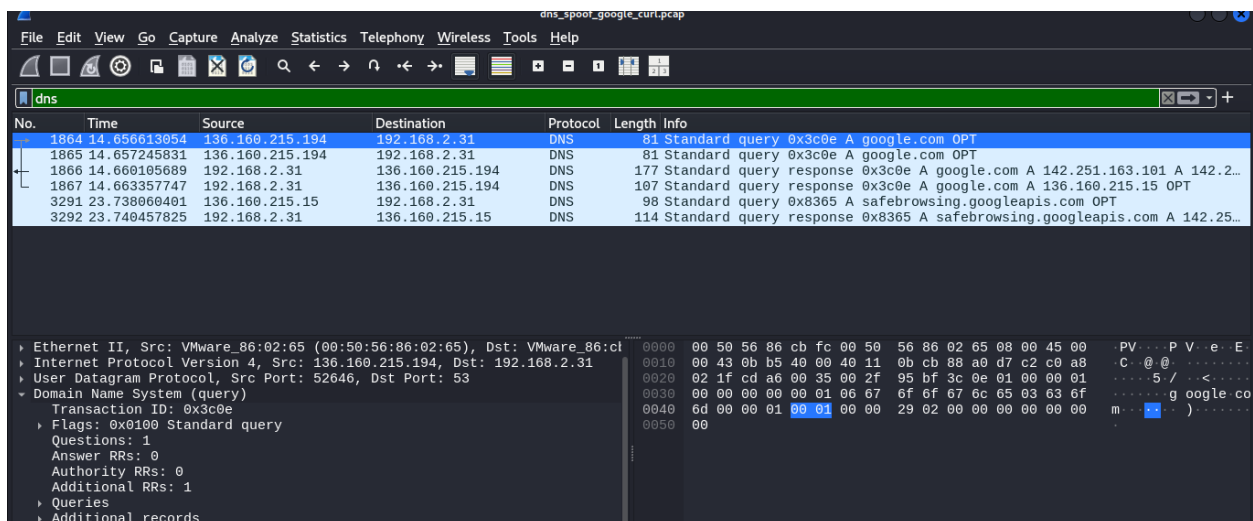
Step 3: Analyze DNS Spoofing Log

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

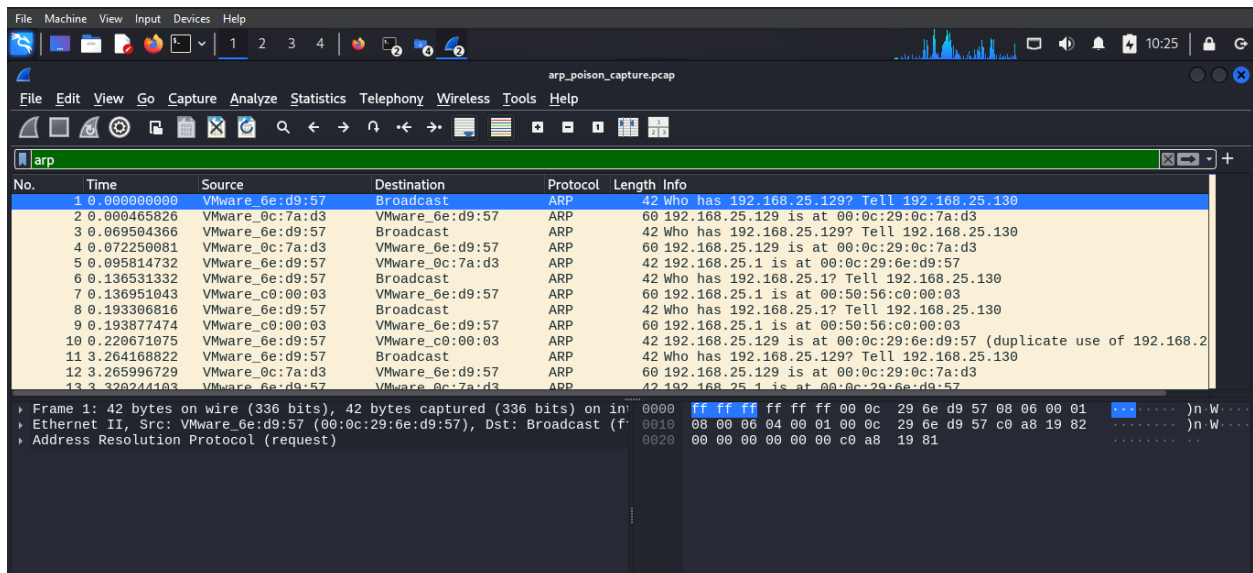
1. Apply ARP filters: arp

2. Look for:

ARP replies without requests (gratuitous ARPs)

Multiple MAC addresses claiming to be the same IP

Inconsistent MAC-IP pairings



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	VMware_6e:d9:57	Broadcast	ARP	42	Who has 192.168.25.129? Tell 192.168.25.130
2	0.000465826	VMware_6e:7a:d3	VMware_6e:d9:57	ARP	60	192.168.25.129 is at 00:0c:29:0c:7a:d3
3	0.000904366	VMware_6e:d9:57	Broadcast	ARP	42	Who has 192.168.25.129? Tell 192.168.25.130
4	0.072250081	VMware_6e:7a:d3	VMware_6e:d9:57	ARP	60	192.168.25.129 is at 00:0c:29:0c:7a:d3
5	0.095814732	VMware_6e:d9:57	VMware_6e:7a:d3	ARP	42	192.168.25.1 is at 00:0c:29:6e:d9:57
6	0.136531332	VMware_6e:d9:57	Broadcast	ARP	42	Who has 192.168.25.1? Tell 192.168.25.130
7	0.136531043	VMware_c0:00:03	VMware_6e:d9:57	ARP	60	192.168.25.1 is at 00:50:56:c0:00:03
8	0.193306816	VMware_6e:d9:57	Broadcast	ARP	42	Who has 192.168.25.1? Tell 192.168.25.130
9	0.193877474	VMware_c0:00:03	VMware_6e:d9:57	ARP	60	192.168.25.1 is at 00:50:56:c0:00:03
10	0.220671075	VMware_6e:d9:57	VMware_c0:00:03	ARP	42	192.168.25.129 is at 00:0c:29:6e:d9:57 (duplicate use of 192.168.25.129)
11	3.264168822	VMware_6e:d9:57	Broadcast	ARP	42	Who has 192.168.25.129? Tell 192.168.25.130
12	3.265996729	VMware_6e:7a:d3	VMware_6e:d9:57	ARP	60	192.168.25.129 is at 00:0c:29:0c:7a:d3
13	3.320244103	VMware_6e:d9:57	VMware_6e:7a:d3	ARP	42	192.168.25.1 is at 00:0c:29:6e:d9:57

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
Ethernet II, Src: VMware_6e:d9:57 (00:0c:29:0c:d9:57), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)