

Wireshark Network Traffic Analysis Report

Task Name: Task 5 – Capture and Analyze Network Traffic Using Wireshark

Target Site: “http://demo.testfire.net” **Tool**

Used: Wireshark (on Kali Linux) **Capture**

File: “traffic_<files>.pcap”

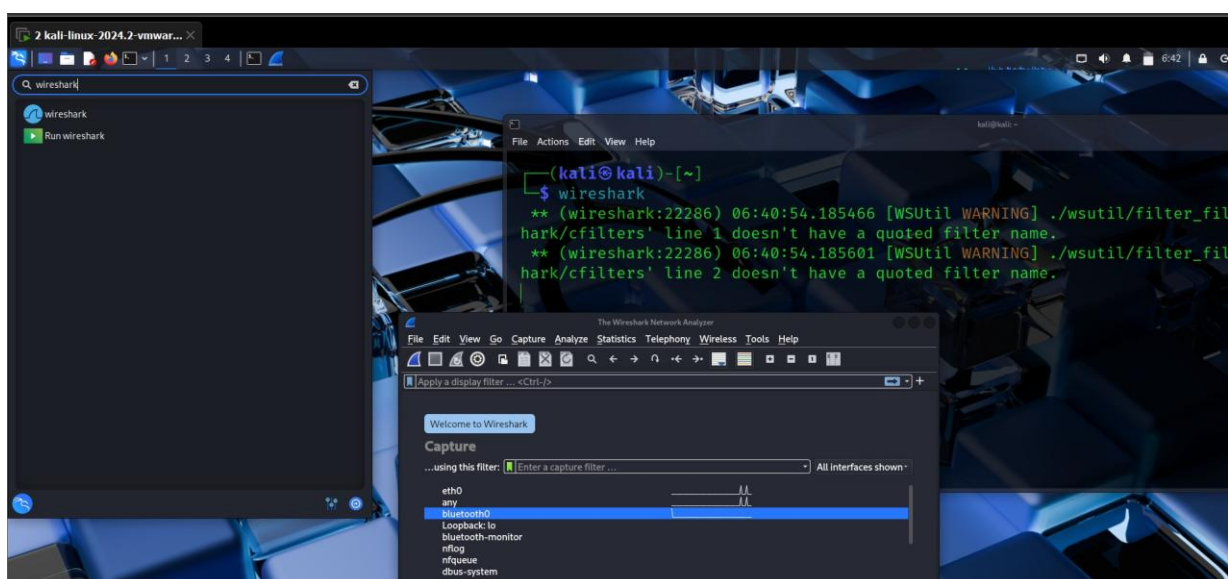
Objective:

The objective of this task is to perform live packet capturing using Wireshark, analyze the network packets and identifying basic protocols and traffic types.

1. Setting up Wireshark

Wireshark is a network packet analyzer tool that is publicly available and comes pre-installed in Kali Linux by default.

We can start Wireshark by typing wireshark in the Kali terminal or by opening it from the applications menu. Once it starts, it asks us to choose a network interface to capture traffic from. For this task, we selected the eth0 interface.



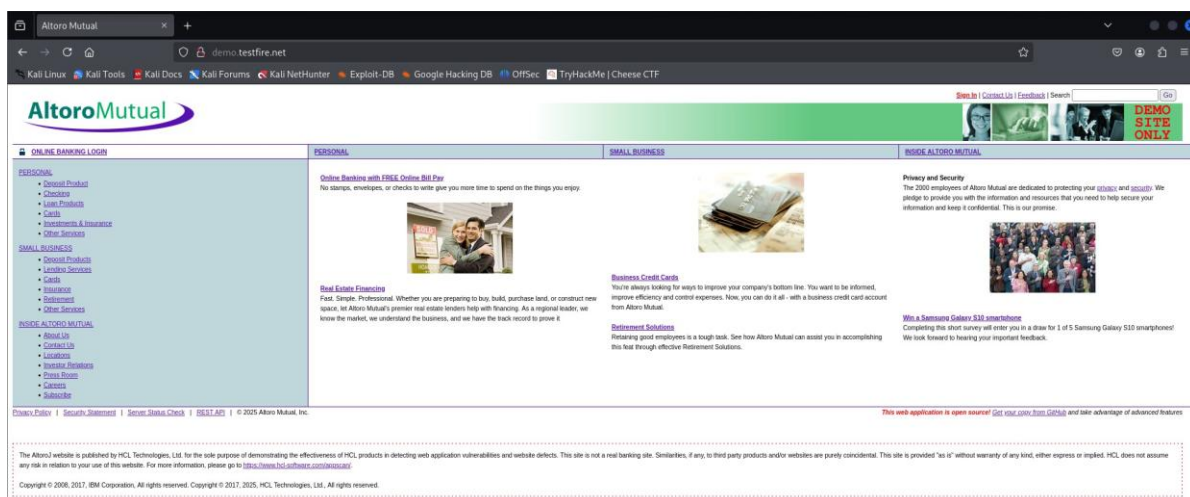
2. Capturing the Packets for a particular domain.

- Packets are small pieces of data that move across a network. They carry part of the message along with source and destination info. At the end, they're reassembled to complete the data.
- The configurations are set, now to hit the search engine with our demo site "demo.testfire.net" and view its network's traffic in Wireshark and capture the packets for our analysis.
- After selecting the interface (eth0), a window appears where live network packets start scrolling as they are captured. We can watch the traffic being captured in real time.

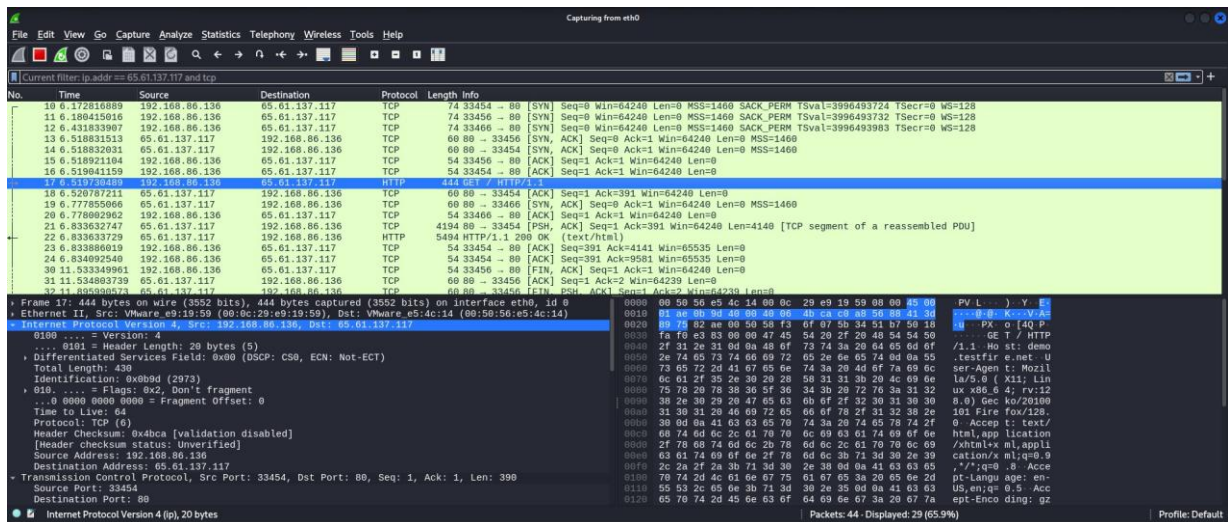
The Wireshark window is divided into three panels :-

- The top panel shows the list of captured packets.
- The middle panel displays details and related protocols of the selected packet.
- The bottom panel shows the raw data (bytes) inside the packet.

Demo site for our network analysis:-



Wireshark capturing the network packets :-

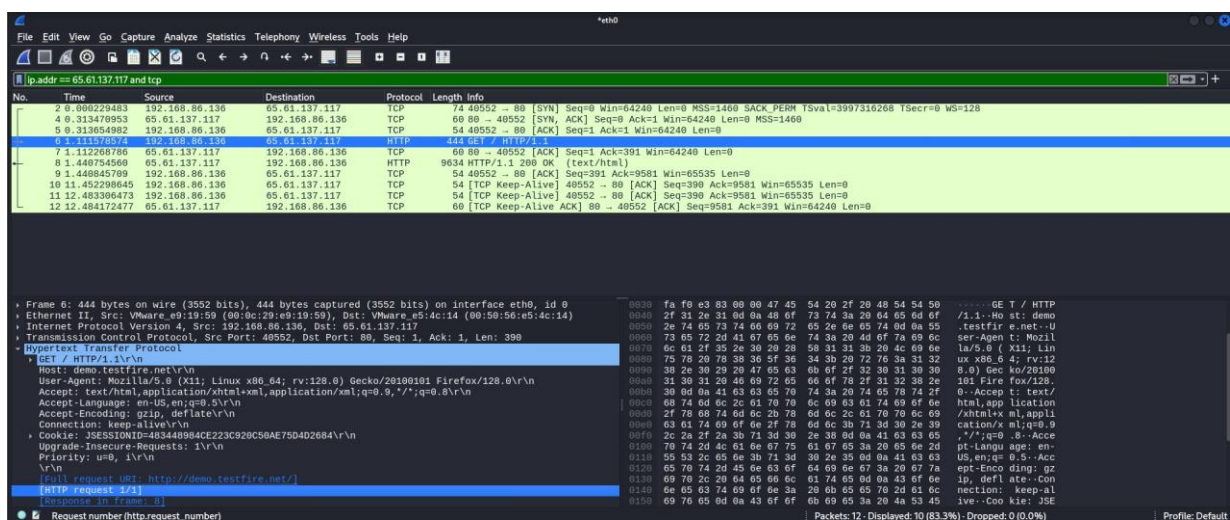


2. Traffic Filtering

Wireshark captures a large number of network requests, so to analyze a specific one, we need to use special keywords to filter the traffic like :-

- “ip.addr □□ 65.61.67.117” – shows all packets to or from that IP.
- “tcp.port □□ 80” – shows traffic on port 80 (HTTP).
- “ip.addr □□ 65.61.67.117 and tcp.port □□ 80” – traffic from/to that IP on port 80.

Network being filtered with specified key words :-



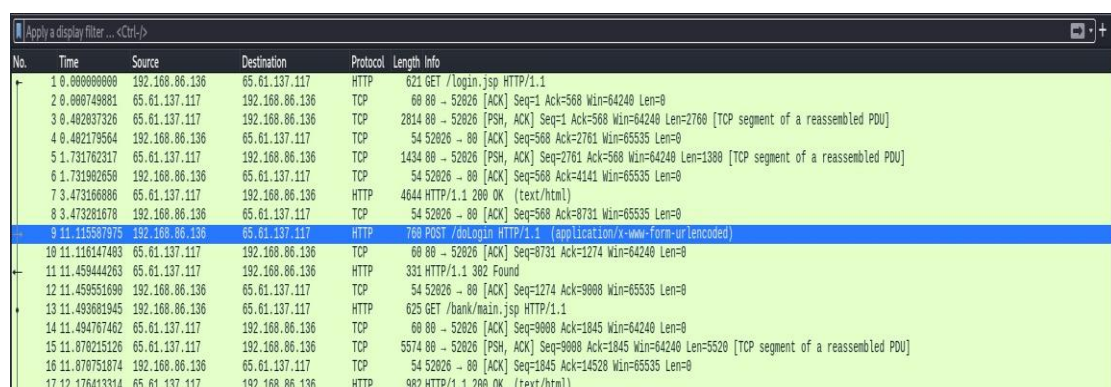
3. Packet List Overview

The Packet List Panel is the top section in Wireshark where all captured packets are shown in a list. Each row represents one packet, and each column shows specific details about it.

The fields listed in packet list panel are described as follows :-

- **No.** – This is the packet number, assigned in the order packets are captured.
- **Time** – Shows when the packet was captured. The time starts from when Wireshark begins capturing and helps measure the gap between packets.
- **Source** – Shows the IP address of the device that sent the packet.
- **Destination** – Shows the IP address of the device that is receiving the packet.
- **Protocol** – Tells which network protocol is used, such as TCP, HTTP, DNS, or ICMP. This shows what kind of communication the packet is part of.
- **Length** – Shows the size of the packet in bytes.
- **Info** – Gives a brief summary of what the packet is doing, such as request type, port numbers, or flags.

Packet Panel List :-



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.86.136	65.61.137.117	HTTP	621	GET /login.jsp HTTP/1.1
2	0.000749881	65.61.137.117	192.168.86.136	TCP	60	80 → 52026 [ACK] Seq=1 Ack=568 Win=64240 Len=0
3	0.492837326	65.61.137.117	192.168.86.136	TCP	2814	80 → 52026 [PSH, ACK] Seq=1 Ack=568 Win=64240 Len=2768 [TCP segment of a reassembled PDU]
4	0.492179564	192.168.86.136	65.61.137.117	TCP	54	52026 → 80 [ACK] Seq=568 Ack=2761 Win=65535 Len=0
5	1.731762317	65.61.137.117	192.168.86.136	TCP	1434	80 → 52026 [PSH, ACK] Seq=2761 Ack=568 Win=64240 Len=1388 [TCP segment of a reassembled PDU]
6	1.731982658	192.168.86.136	65.61.137.117	TCP	54	52026 → 80 [ACK] Seq=568 Ack=4141 Win=65535 Len=0
7	3.473168886	65.61.137.117	192.168.86.136	HTTP	4644	HTTP/1.1 200 OK (text/html)
8	3.473281678	192.168.86.136	65.61.137.117	TCP	54	52026 → 80 [ACK] Seq=568 Ack=8731 Win=65535 Len=0
9	11.115387975	192.168.86.136	65.61.137.117	HTTP	768	POST /doLogin HTTP/1.1 (application/x-www-form-urlencoded)
10	11.116147483	65.61.137.117	192.168.86.136	TCP	60	80 → 52026 [ACK] Seq=8731 Ack=1274 Win=64240 Len=0
11	11.459444263	65.61.137.117	192.168.86.136	HTTP	331	HTTP/1.1 302 Found
12	11.459551698	192.168.86.136	65.61.137.117	TCP	54	52026 → 80 [ACK] Seq=1274 Ack=9008 Win=65535 Len=0
13	11.493681945	192.168.86.136	65.61.137.117	HTTP	625	GET /bank/main.jsp HTTP/1.1
14	11.494767402	65.61.137.117	192.168.86.136	TCP	60	80 → 52026 [ACK] Seq=9008 Ack=1845 Win=64240 Len=0
15	11.870215126	65.61.137.117	192.168.86.136	TCP	5574	80 → 52026 [PSH, ACK] Seq=9008 Ack=1845 Win=64240 Len=5520 [TCP segment of a reassembled PDU]
16	11.870751874	192.168.86.136	65.61.137.117	TCP	54	52026 → 80 [ACK] Seq=1845 Ack=14528 Win=65535 Len=0
17	12.176413314	65.61.137.117	192.168.86.136	HTTP	982	HTTP/1.1 200 OK (text/html)

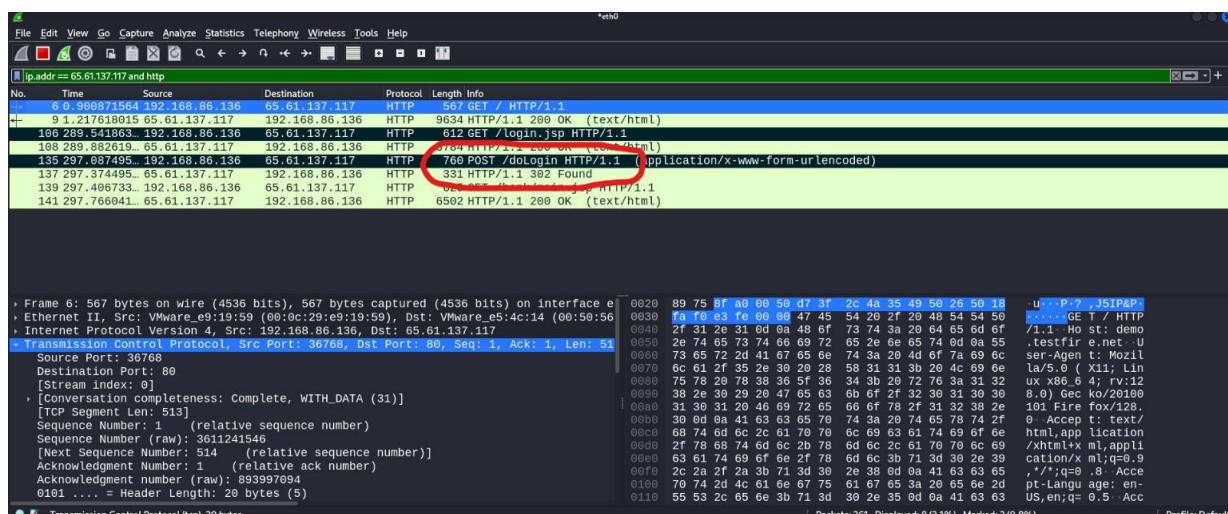
4. Capturing different Protocol's packets



HTTP (Hyper Text Transfer Protocol) :-

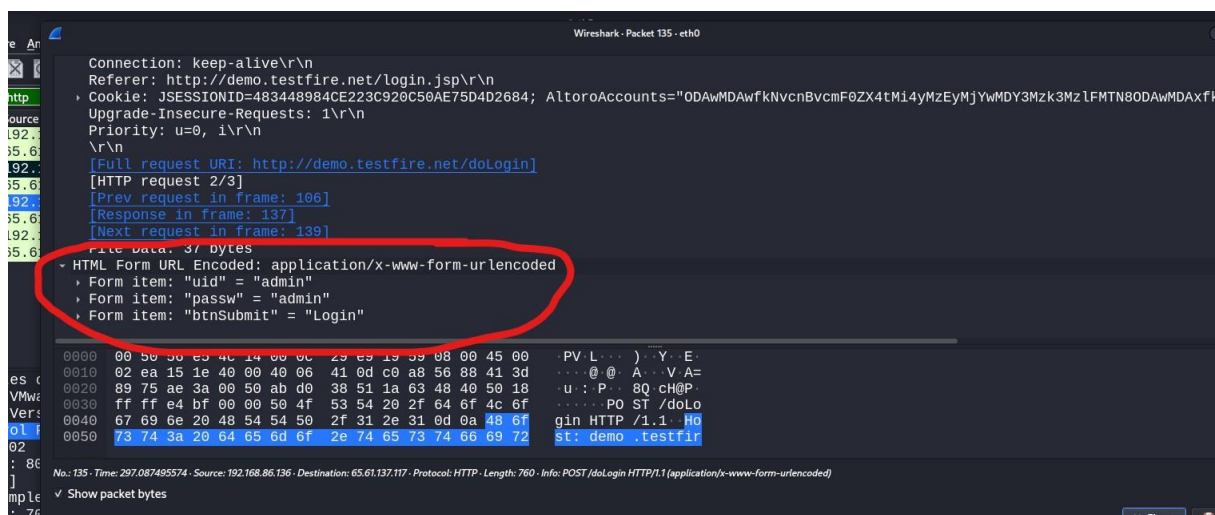
The basic protocol used for communication between our **web browser** and **web servers**. As we all know that this protocol is not safe as it does not provide with Encryption layer.

HTTP Login request packet captured of demo site :-

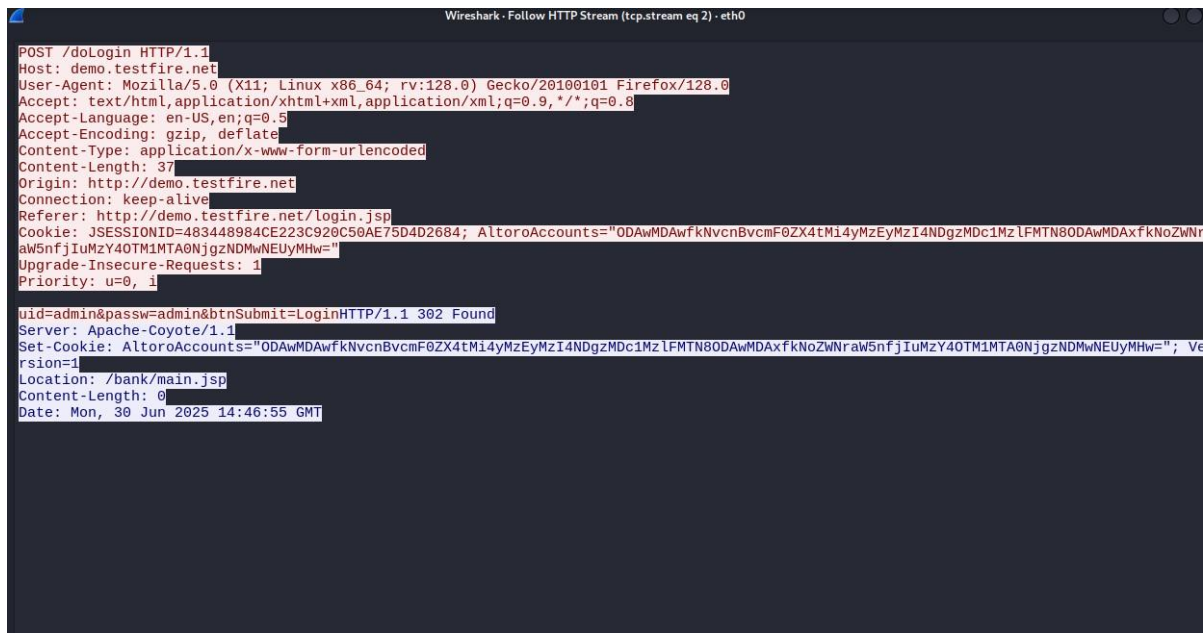


As we can see, I have marked the HTTP login packets captured in Wireshark. This shows that a login attempt was made on our demo site. Since HTTP does not provide any encryption, as mentioned earlier, it can easily expose our confidential data to anyone monitoring the network.

Lets analyze the “/doLogin packet” :-



As mentioned earlier the request package exposes our confidential info.



```
Wireshark · Follow HTTP Stream (tcp.stream eq 2) · eth0

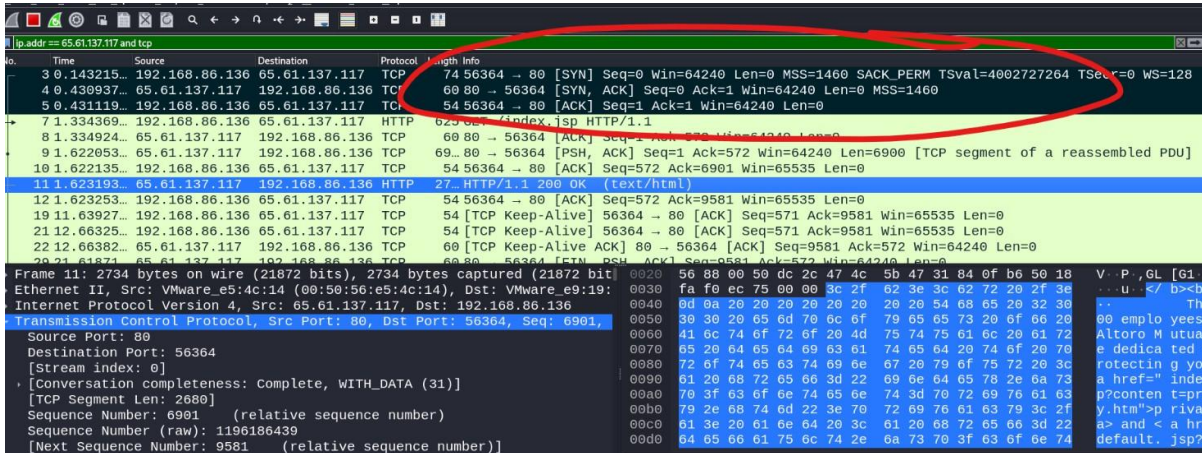
POST /doLogin HTTP/1.1
Host: demo.testfire.net
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Origin: http://demo.testfire.net
Connection: keep-alive
Referer: http://demo.testfire.net/login.jsp
Cookie: JSESSIONID=483448984CE223C920C50AE75D4D2684; AltoroAccounts="ODAwMDAwfKNvcnBvcnF0ZX4tM14yMzEyMzI4NDgzMdc1MzlfMTN8ODAwMDAxfkNoZWNr
aw5nfjIuMzY4OTM1MTA0NjgzNDMwNEUyMHw="
Upgrade-Insecure-Requests: 1
Priority: u=0, i

uid=admin&passw=admin&btnSubmit=LoginHTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Set-Cookie: AltoroAccounts="ODAwMDAwfKNvcnBvcnF0ZX4tM14yMzEyMzI4NDgzMdc1MzlfMTN8ODAwMDAxfkNoZWNr
aw5nfjIuMzY4OTM1MTA0NjgzNDMwNEUyMHw="; Version=1
Location: /bank/main.jsp
Content-Length: 0
Date: Mon, 30 Jun 2025 14:46:55 GMT
```

TCP (Transmission Control Protocol) :-

It is used to send data across a network in a reliable way. TCP breaks the data into packets, makes sure all packets arrive, and puts them back together in the correct order. If any packets are lost or damaged, TCP will resend them.

TCP 3-way handshake request package captured :-



```
Time      Source            Destination        Protocol  Length  Info
0.000000  65.61.137.117    192.168.86.136    TCP      60      656364 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4002727264 TSecr=0 WS=128
0.000000  192.168.86.136  65.61.137.117    TCP      60      80 → 56364 [SYN, ACK] Seq=1 Ack=1 Win=64240 Len=0 MSS=1460
0.000000  65.61.137.117    192.168.86.136    TCP      60      80 → 56364 [ACK] Seq=1 Ack=572 Win=65535 Len=0
7.1334369  192.168.86.136  65.61.137.117    HTTP      625      200 OK (text/html)
11.1623193  65.61.137.117    192.168.86.136    HTTP      27      HTTP/1.1 200 OK (text/html)
12.1623253  192.168.86.136  65.61.137.117    TCP      54      56364 → 80 [ACK] Seq=572 Ack=9581 Win=65535 Len=0
19.1163927  192.168.86.136  65.61.137.117    TCP      54      [TCP Keep-Alive] 56364 → 80 [ACK] Seq=571 Ack=9581 Win=65535 Len=0
21.1266325  192.168.86.136  65.61.137.117    TCP      54      [TCP Keep-Alive] 56364 → 80 [ACK] Seq=571 Ack=9581 Win=65535 Len=0
22.1266382  65.61.137.117    192.168.86.136    TCP      60      [TCP Keep-Alive ACK] 80 → 56364 [ACK] Seq=9581 Ack=572 Win=64240 Len=0
23.2161871  65.61.137.117    192.168.86.136    TCP      60      80 → 56364 [FIN, PSH, ACK] Seq=9581 Ack=572 Win=64240 Len=0

Frame 11: 2734 bytes on wire (21872 bits), 2734 bytes captured (21872 bits) on interface 0
Ethernet II, Src: VMware_e5:4c:14 (00:50:56:e5:4c:14), Dst: VMware_e9:19:14 (00:50:56:e9:19:14)
Internet Protocol Version 4, Src: 65.61.137.117, Dst: 192.168.86.136
Transmission Control Protocol, Src Port: 80, Dst Port: 56364, Seq: 6901, Len: 0
  Source Port: 80
  Destination Port: 56364
  [Stream index: 0]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 2680]
  Sequence Number: 6901 (relative sequence number)
  Sequence Number (raw): 1196186439
  [Next Sequence Number: 9581 (relative sequence number)]
```

The TCP 3-way handshake is the process used to start a reliable connection between two devices on a network.

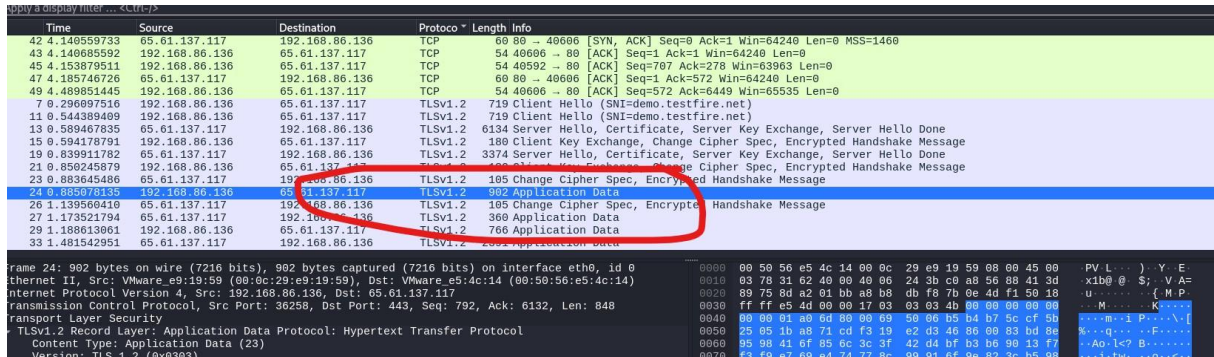
1. First, the sender sends a message called SYN to request a connection
2. The receiver replies with SYN-ACK to accept the request
3. The sender then sends an ACK to confirm the connection is ready

This three-step process makes sure both sides are ready to send and receive data safely.

HTTPS(Hyper Text Transfer Protocol Secure) :-

HTTPS stands for Hypertext Transfer Protocol Secure. It is the secure version of HTTP and uses encryption to keep data safe while it travels between your browser and a website. This helps protect things like passwords and personal information from being seen or stolen by attackers.

HTTPS Login request packet captured of demo site :-

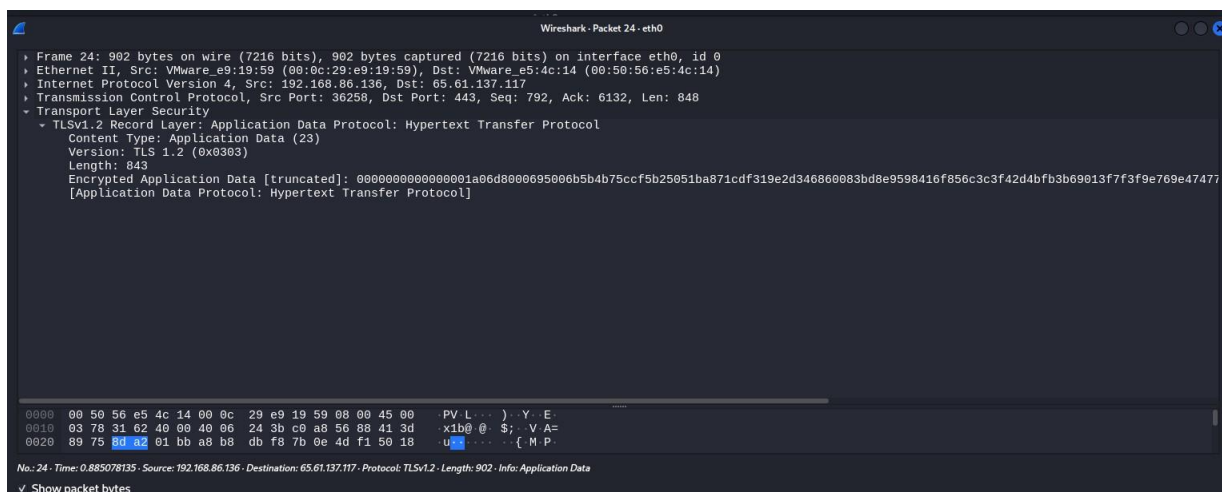


Time	Source	Destination	Protocol	Length	Info
42.4.148559733	65.61.137.117	192.168.86.136	TCP	60	80 → 40606 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
43.4.140895592	192.168.86.136	65.61.137.117	TCP	54	40606 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
45.4.153879511	192.168.86.136	65.61.137.117	TCP	54	40992 → 80 [ACK] Seq=707 Ack=276 Win=63963 Len=0
47.4.185746726	65.61.137.117	192.168.86.136	TCP	60	80 → 40606 [ACK] Seq=1 Ack=572 Win=64240 Len=0
49.4.489851445	192.168.86.136	65.61.137.117	TCP	54	40606 → 80 [ACK] Seq=572 Ack=6449 Win=65535 Len=0
7.0.296897516	192.168.86.136	65.61.137.117	TLSv1.2	719	Client Hello (SNI=demo.testfire.net)
11.0.544389489	192.168.86.136	65.61.137.117	TLSv1.2	719	Client Hello (SNI=demo.testfire.net)
13.0.589467835	65.61.137.117	192.168.86.136	TLSv1.2	6134	Server Hello, Certificate, Server Key Exchange, Server Hello Done
15.0.594178791	192.168.86.136	65.61.137.117	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
19.0.839911782	65.61.137.117	192.168.86.136	TLSv1.2	3374	Server Hello, Certificate, Server Key Exchange, Server Hello Done
21.0.859245879	192.168.86.136	65.61.137.117	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
23.0.883645486	65.61.137.117	192.168.86.136	TLSv1.2	902	Application Data
24.0.885878135	192.168.86.136	65.61.137.117	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
26.1.139560410	65.61.137.117	192.168.86.136	TLSv1.2	300	Application Data
27.1.173521794	65.61.137.117	192.168.86.136	TLSv1.2	766	Application Data
29.1.188613061	192.168.86.136	65.61.137.117	TLSv1.2	766	Application Data
33.1.481542951	65.61.137.117	192.168.86.136	TLSv1.2	766	Application Data

Frame 24: 902 bytes on wire (7216 bits), 902 bytes captured (7216 bits) on interface eth0, id 0
Ethernet II, Src: VMware_e9:19:59 (08:0c:29:e9:19:59), Dst: VMware_e5:4c:14 (08:50:56:e5:4c:14)
Internet Protocol Version 4, Src: 192.168.86.136, Dst: 65.61.137.117
Transmission Control Protocol, Src Port: 36258, Dst Port: 443, Seq: 792, Ack: 6132, Len: 848
Transport Layer Security
- TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
Content Type: Application Data (23)
Version: TLS 1.2 (0x0303)
Length: 843
Encrypted Application Data [truncated]: 0000000000000001a06d0800695006b5b4b75ccf5b25051ba871cdf319e2d34686083bd8e9598416f856c3c3f42d4bfb3b69013f7f3f9e769e47477
[Application Data Protocol: Hypertext Transfer Protocol]

We can see that the HTTPS request made is being encrypted by adding a TLS (Transport layer security), which encodes out data from unauthorized monitoring as seen below.

Lets analyze this packet :-



Time	Source	Destination	Protocol	Length	Info
42.4.148559733	65.61.137.117	192.168.86.136	TCP	60	80 → 40606 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
43.4.140895592	192.168.86.136	65.61.137.117	TCP	54	40606 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
45.4.153879511	192.168.86.136	65.61.137.117	TCP	54	40992 → 80 [ACK] Seq=707 Ack=276 Win=63963 Len=0
47.4.185746726	65.61.137.117	192.168.86.136	TCP	60	80 → 40606 [ACK] Seq=1 Ack=572 Win=64240 Len=0
49.4.489851445	192.168.86.136	65.61.137.117	TCP	54	40606 → 80 [ACK] Seq=572 Ack=6449 Win=65535 Len=0
7.0.296897516	192.168.86.136	65.61.137.117	TLSv1.2	719	Client Hello (SNI=demo.testfire.net)
11.0.544389489	192.168.86.136	65.61.137.117	TLSv1.2	719	Client Hello (SNI=demo.testfire.net)
13.0.589467835	65.61.137.117	192.168.86.136	TLSv1.2	6134	Server Hello, Certificate, Server Key Exchange, Server Hello Done
15.0.594178791	192.168.86.136	65.61.137.117	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
19.0.839911782	65.61.137.117	192.168.86.136	TLSv1.2	3374	Server Hello, Certificate, Server Key Exchange, Server Hello Done
21.0.859245879	192.168.86.136	65.61.137.117	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
23.0.883645486	65.61.137.117	192.168.86.136	TLSv1.2	902	Application Data
24.0.885878135	192.168.86.136	65.61.137.117	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
26.1.139560410	65.61.137.117	192.168.86.136	TLSv1.2	300	Application Data
27.1.173521794	65.61.137.117	192.168.86.136	TLSv1.2	766	Application Data
29.1.188613061	192.168.86.136	65.61.137.117	TLSv1.2	766	Application Data
33.1.481542951	65.61.137.117	192.168.86.136	TLSv1.2	766	Application Data

Frame 24: 902 bytes on wire (7216 bits), 902 bytes captured (7216 bits) on interface eth0, id 0
Ethernet II, Src: VMware_e9:19:59 (08:0c:29:e9:19:59), Dst: VMware_e5:4c:14 (08:50:56:e5:4c:14)
Internet Protocol Version 4, Src: 192.168.86.136, Dst: 65.61.137.117
Transmission Control Protocol, Src Port: 36258, Dst Port: 443, Seq: 792, Ack: 6132, Len: 848
Transport Layer Security
- TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
Content Type: Application Data (23)
Version: TLS 1.2 (0x0303)
Length: 843
Encrypted Application Data [truncated]: 0000000000000001a06d0800695006b5b4b75ccf5b25051ba871cdf319e2d34686083bd8e9598416f856c3c3f42d4bfb3b69013f7f3f9e769e47477
[Application Data Protocol: Hypertext Transfer Protocol]

```
Wireshark - Follow TCP Stream (tcp.stream eq 0) - eth0

...U.p#.h.....}.....E.J.%...l.R.....qK.u...m!"<|...$......+/.0.
.../5...%.....demo.testfire.net.....
...#.....h2.http/1.1.....
...3.k.i.....{...6~.~!g..8...F...A.<...k.....C.....>Y.k*!F...Y.IU.;W.@R(H.CS(...Se3..40B.+.....
...@.....
...X.l...@...<...{=..5.6.m.e..8.....2.s0..^..L7...u...6...d.|
...e...R...PA.%;...P...;^C.C.Q>c.V..|P*.....
...3...w...{.<wJQs.R.{vT..pm9;./...)
...k...u...n...iu...C...Z...!x.....#...!...|...>...
...X.gP...=...C.....E;...Cw.9.0.....Q...i.@4.+RC4...Q.....Q..hb.[v:W.9p.QML.CM.J.....\...|!hb.[.....r.kc.y.s.]x
.../.....00...0.....P...u.v.....qm0
...*H...
...0.1.0.....GB1.0...U...Greater Manchester1.0...U...Salford1.0...U...
...Setigo Limited1705...U...Setigo RSA Domain Validation Secure Server CA0...
250521000000Z...
260621235959Z0.1.0...U...demo.testfire.net0..."}0
...*H...
...0...
...b0..2...@...+R.....#...-xE...m..h..2.f..v?r.)I...{Q...4.Yc...t.R."c0...1...93n.
...A.X<.(...b...v...5...[.a.D.{...}l..6TK.ad0.)s...^..U..6...FL...$].g-E7.$R.7.i.c...V>n.Q..b...~..s...&=.E.2oS...
...+y...-z.X.....:C...e.....0...0...U..#..0.....^T..w.....a..0...U.....*...<...0...U.....0...U.....
...0...U..X...0...+.....01.U...B0@04.+.....1...0%0#...+.....https://setigo.com/CPS0...g...0...+.....x0v00...+.....0
...Http://crt.setigo.com/SetigoRSADomainValidationSecureServerCA.crt0#...+.....0...http://ocsp.setigo.com0...
...+.....y.....0...k.i.w...d.UX...C.h7.Bw...?.....6nF.?.....H0F.!.....0...V.....%Q...+y...!^..U...+Z+.9N.|..."}@..
...L.4.v...{.0...ox^M...-r1...|pA-%L.....G0E..7^..b.=w.\R2...V.Z..5...|dF.TA.!..Z/S.%8...j.?..b0,p;..Q.....v..W....>
...3...q2%!.%a!N...G0E..N...>]...00..P.VeJ.^.....a..
...p.!...^..n.'Y:x"....@{(...%...N..A0...U...0...demo.testfire.net0
...*H...
...K.R.(?...u8...%...
...S.../...
...%...q.ok.02.L...ha...szg..(#K...vmmk.X./..rd>:y.....N...L.H.&a.J.ma^...5.9...2...{.i:.....;{=4J.C..0:1.Tz...'
...Z...
...8.F..p..i..a^*
...B...$J0...J...1.X...Dw...f{M2'a..5(.g.dF...e...YyG.....0...0.....}{Q&v...t...S
...
...*H...
...0.1.0...US1.0...U...
3 client pkts, 3 server pkts, 5 turns.

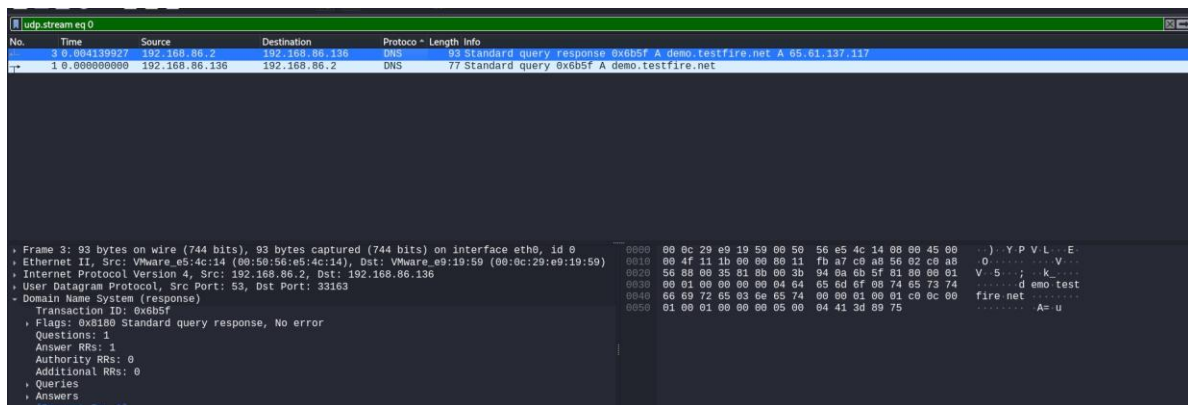
Entire conversation (8,076 bytes) Show data as ASCII Stream 0

Find: Filter Out This Stream Print Save as Back x Close Help
```

DNS(Domain name system)) :-

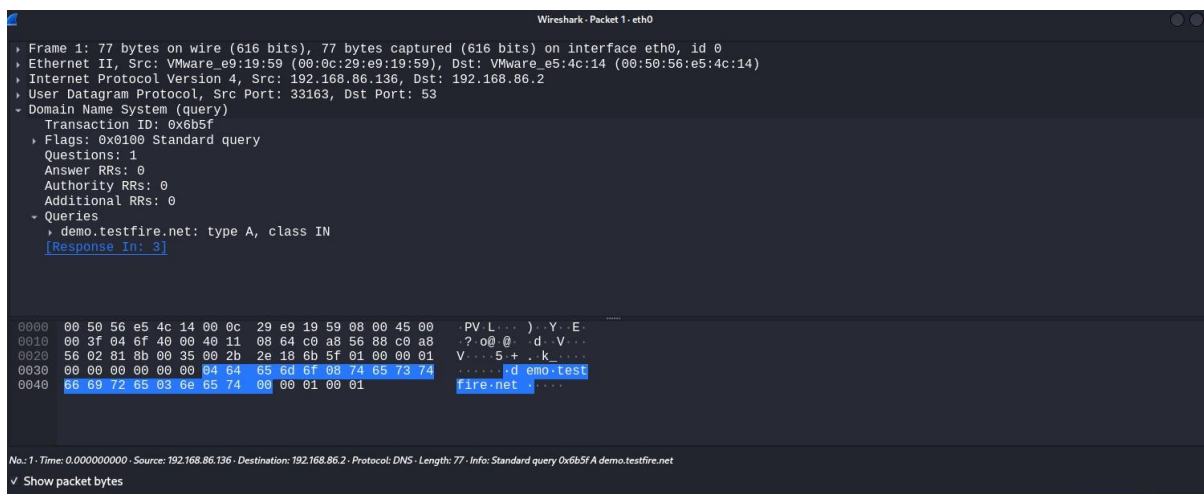
DNS stands for Domain Name System. It works like the phonebook of the internet. When we type a website name like google.com, DNS finds the correct IP address for that site so our browser can connect to it. Without DNS, we would have to remember the number based IP addresses of every website.

DNS request packet captured :-



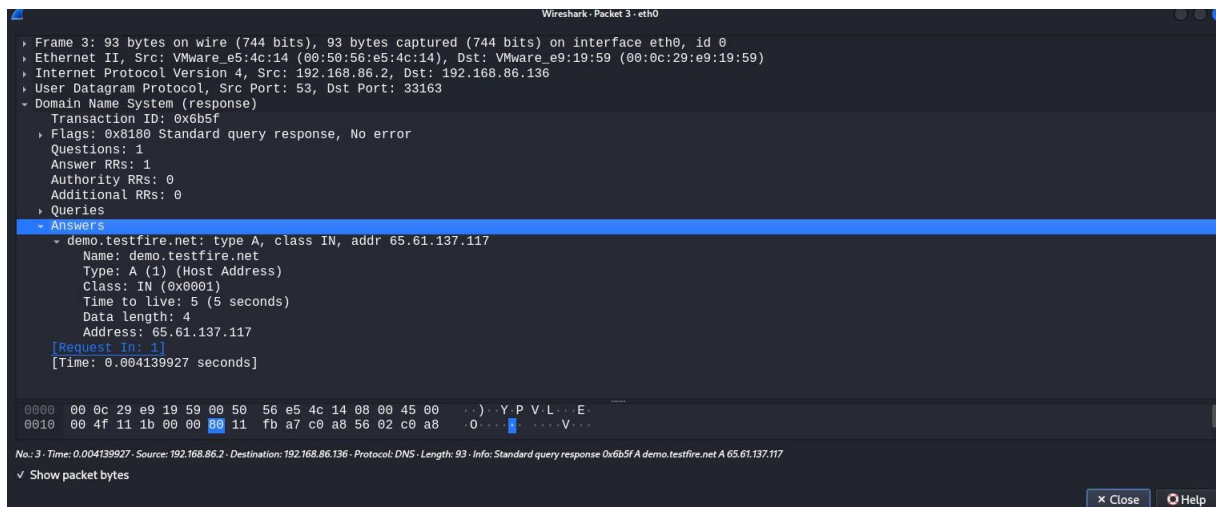
As mentioned earlier, the DNS protocol helps us find the IP address for the website or service we are trying to reach. In the figure above, we can see two DNS packets. One is making a request to find the IP address, and the other is responding with the answer to that request.

Lets analyze 1st packet :-



The analysis of the first packet shows that the request is asking for the domain name "demo.testfire.net", which is similar to sending a query to find its IP address.

lets analyze 2nd packet :-



In response to the first packet, this packet replies with the IP address "65.61.137.117", similar to answering a question that was asked by the 1st packet.

5. Packet Analysis

❖ HTTP Packet Analysis

Packet No: 106, 108, 135

Source IP: 192.168.82.136

Destination IP: 65.61.137.117

Info: /login.jsp, /doLogin.jsp

Purpose: These packets show requests made to the login page and form submission, indicating a user is trying to log into the website using the HTTP protocol.

❖ TCP Handshake Analysis

Packets: 3 (SYN), 4 (SYN-ACK), 5 (ACK)

Source Port: 40952

Destination Port: 80

Purpose: These three packets show the standard TCP handshake used to establish a stable and reliable connection between the client and the web server.

❖ HTTPS Packet Analysis

Packet No: 106, 108, 135

Source IP: 192.168.82.136

Destination IP: 65.61.137.117

Info: /login.jsp, /doLogin.jsp

Purpose: These packets represent secure web requests. If HTTPS was in use, the data would be encrypted, protecting sensitive login information during transmission.

❖ DNS Packet Analysis

Packet No: 1, 3

Query: demo.testfire.net

Response: 65.61.137.117

Purpose: These packets show the DNS resolution process where the domain name is translated into an IP address before the connection is made.

6. Observation

- The site uses HTTP instead of HTTPS, which means data like usernames and passwords can be seen in plain text over the network.
- Multiple HTTP requests were made to pages such as /login.jsp and /index.jsp, showing user activity on the site.
- DNS queries were made before the HTTP connections, which is normal as the system first resolves the domain name to an IP address.
- A complete TCP 3-way handshake was observed before any data was exchanged, confirming that a proper connection was established.

7. Conclusion

By using Wireshark, we were able to watch how our system talked to the website demo.testfire.net step by step. We saw the process start with a DNS request to find the website's IP address, then a proper connection setup using the TCP handshake, followed by the actual page visits using HTTP.

The DNS packets showed that the domain demo.testfire.net was linked to the IP address 65.61.137.117. Then, a full connection was made using the usual 3-step handshake method. After that, multiple HTTP requests were sent to pages like /login.jsp and /doLogin.jsp, showing that a login attempt took place. Since the site used HTTP instead of HTTPS, all the information sent was visible in plain text, including sensitive details like login data.

Overall, this task helped us understand how data moves through a network and how tools like Wireshark can help us see what's really happening behind the scenes. It also made it clear why using secure websites (with HTTPS) is important to keep our data safe.