

## Task-5

### Capturing and Analysing Network Traffic Using Wireshark:

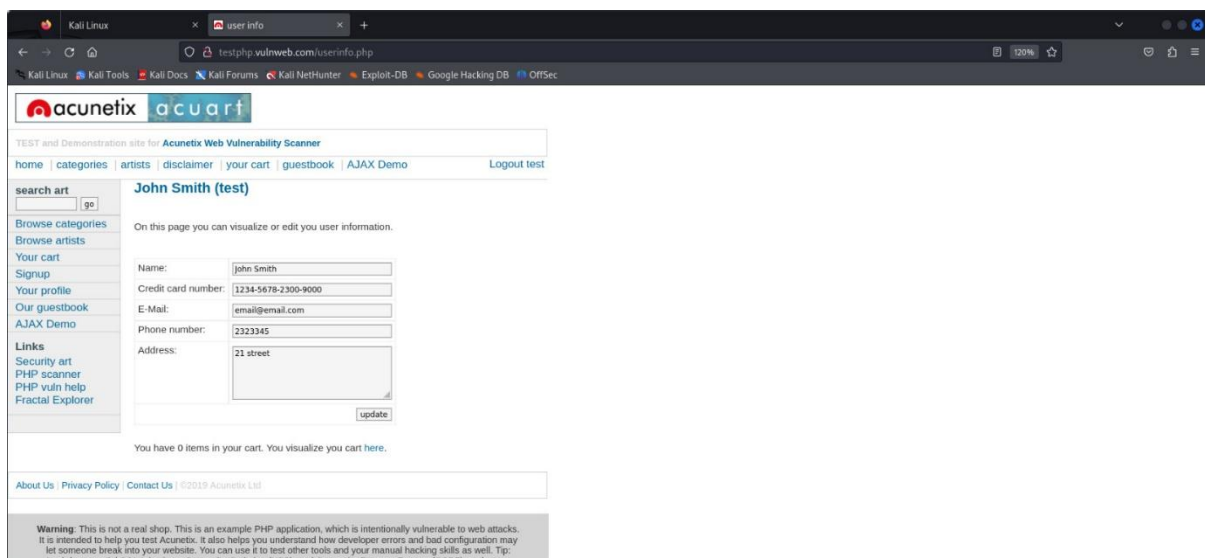
Capturing live network packets using Wireshark and identifying common protocols like HTTP, DNS, and TCP.

**Wireshark:** Wireshark open-source network protocol analyser. It allows you to capture, inspect, and analyse network traffic in real time or from saved capture files. With Wireshark, you can see what's happening on your network at a microscopic level, making it an essential tool for:

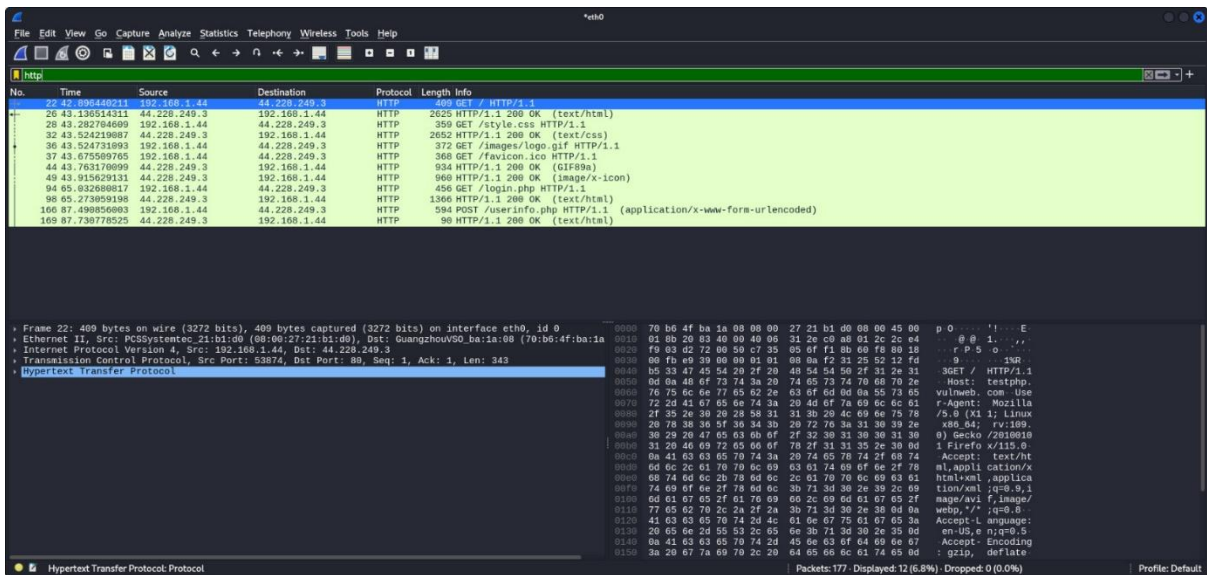
- Network troubleshooting
- Protocol analysis
- Security investigations
- Learning and teaching about network protocols

### Capturing HTTP and TCP Packets for testphp.vulnweb.com website:

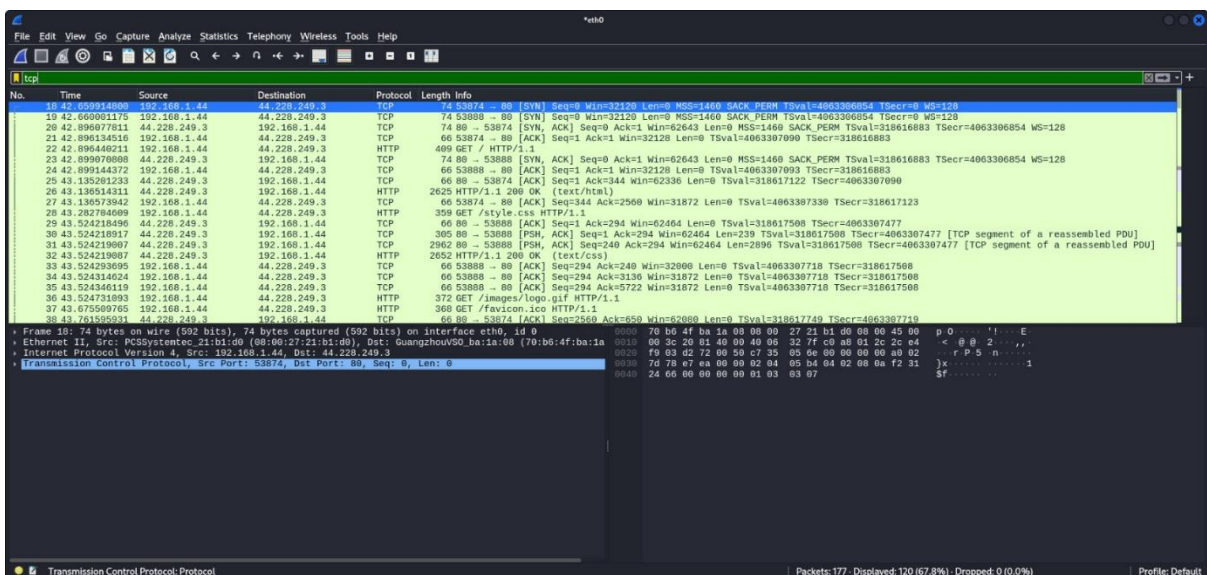
Launch the Wireshark tool and start capturing packets. Then, go to your web browser, type "testphp.vulnweb.com," and log in with the username "test." After that, return to Wireshark, stop the capture, and analyse the captured HTTP and TCP data packets.



-test Login-



## -HTTP Packet-



## -TCP Packet-

## HTTP Packet Analysis:

### GET request:

- It's an HTTP GET request sent from your system (IP: 192.168.1.44) to the web server (IP: 44.228.249.3) on port 80.
- This request asks for a resource from testphp.vulnweb.com using HTTP/1.1.

### Response: 200 (ok)

### Post request:

- It's an HTTP POST request sent from your computer (IP: 192.168.1.44) to the web server (IP: 44.228.249.3) on port 80.
- The browser is submitting form data to /userinfo.php on the site.

**Response:**200 (ok)

### TCP Packet Analysis:

computer (192.168.1.44) is communicating with the web server (44.228.249.3) over TCP on port 80.

These packets show the TCP three-way handshake used to set up the connection:

- SYN packet: starts the connection.
- SYN-ACK packet: server replies to acknowledge.
- ACK packet: completes the handshake.

The first TCP packet (No. 18) is a **SYN** from your computer to the server, trying to start a connection.

The next packets show **SYN-ACK** and **ACK** to finish the handshake.

Following packets are **ACK** and **PSH, ACK**, which carry data like HTTP requests and responses.

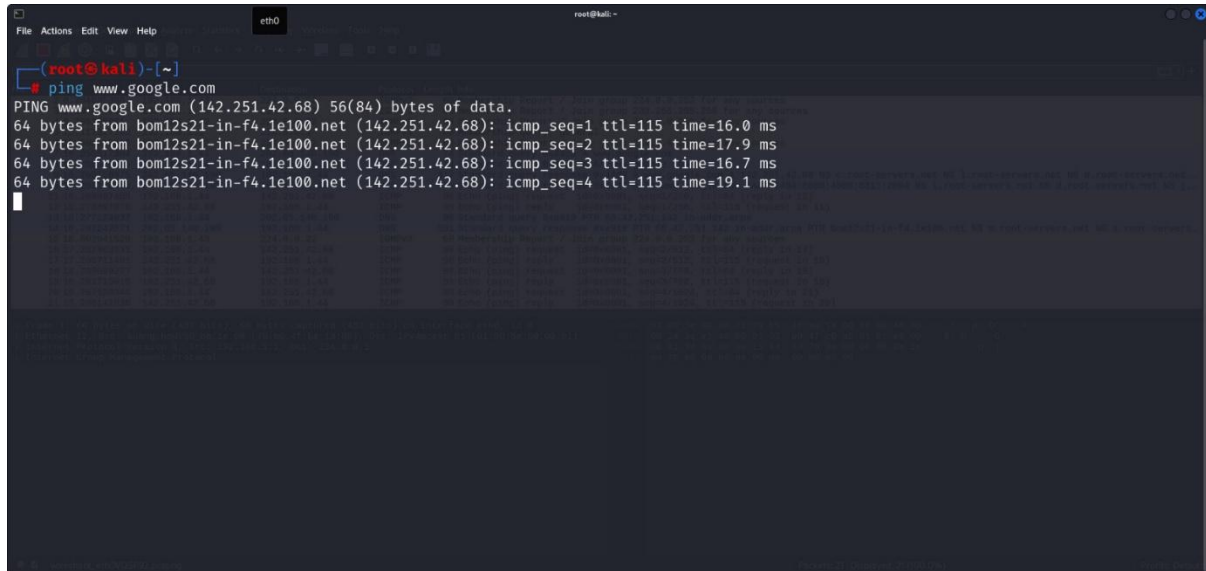
These packets confirm a **successful TCP connection** was established.

After the handshake, your browser started sending HTTP requests over this connection.

The TCP handshake (SYN → SYN-ACK → ACK) is required for reliable data transfer. Without it, HTTP traffic wouldn't work.

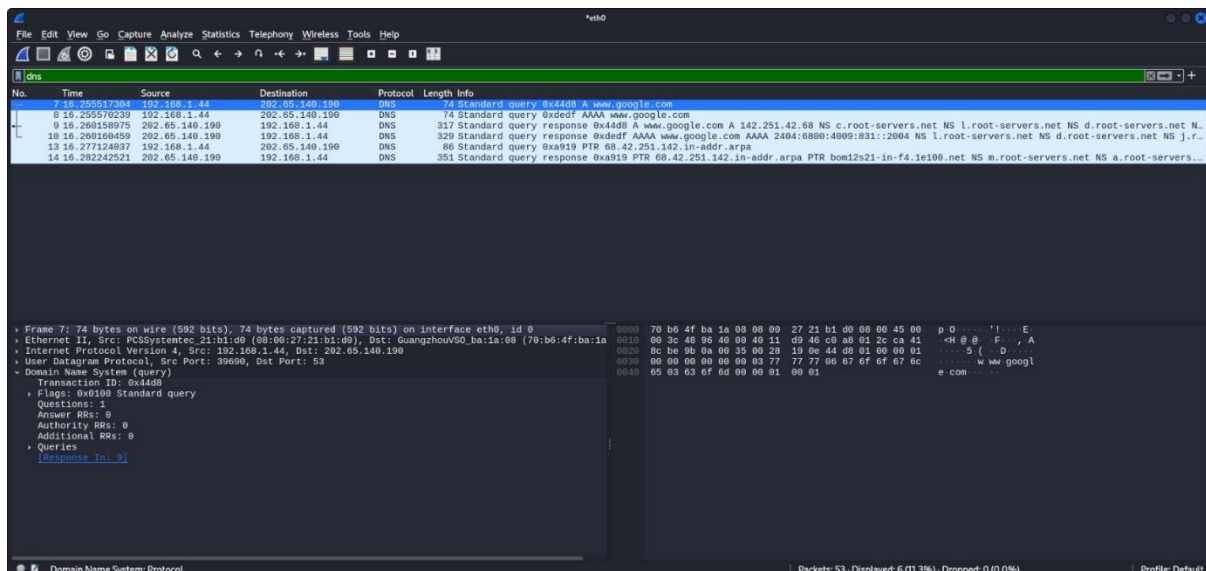
## Capturing DNS Packet for [www.google.com](http://www.google.com):

Launch Wireshark and start capturing. Then, go to the terminal and type the command ping www.google.com. After that, stop Wireshark and analyze the captured DNS packets.



```
(root@kali)~# ping www.google.com
PING www.google.com (142.251.42.68) 56(84) bytes of data:
64 bytes from bom12s21-in-f4.1e100.net (142.251.42.68): icmp_seq=1 ttl=115 time=16.0 ms
64 bytes from bom12s21-in-f4.1e100.net (142.251.42.68): icmp_seq=2 ttl=115 time=17.9 ms
64 bytes from bom12s21-in-f4.1e100.net (142.251.42.68): icmp_seq=3 ttl=115 time=16.7 ms
64 bytes from bom12s21-in-f4.1e100.net (142.251.42.68): icmp_seq=4 ttl=115 time=19.1 ms
```

-www.google.com-

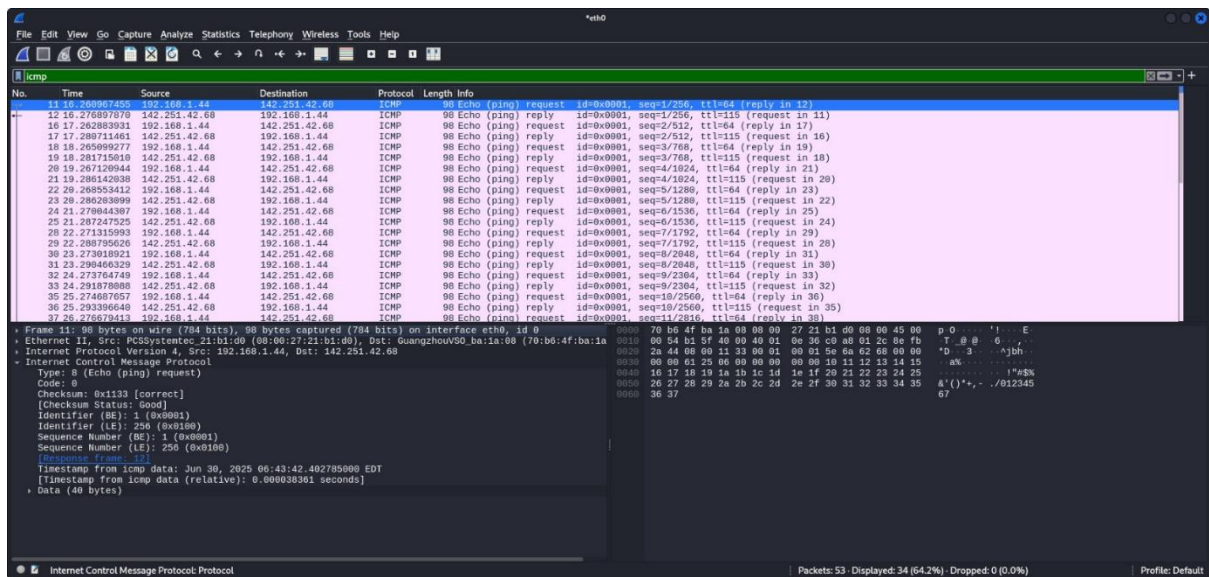


-DNS Packet-

## DNS Packet Analysis:

Your computer (192.168.1.44) sends a **DNS request** packet to the DNS server (202.65.140.190) asking for the IP address of `www.google.com`. This is the **DNS query**. The server then sends back a **DNS response** packet with the answer, providing the IP address of `www.google.com`. This allows your computer to know where to send further packets, like pings or web requests, to reach Google.

## ICMP Packet:



When you run `ping www.google.com`, your computer sends **ICMP Echo Request** packets to the IP address of `google.com`. The server then replies with **ICMP Echo Reply** packets.

- **ICMP Echo Request:**

- Sent by your computer to check if the destination (`google.com`) is reachable.
- Identified by **Type: 8** (Echo Request) in the packet details.

- **ICMP Echo Reply:**

- Sent back by the destination server if it is reachable.
  - Identified by **Type: 0** (Echo Reply) in the packet details.
-