Experiment No. 9

Title: Network Sniffing - Wireshark

Batch: A3 **Roll No.:16010421073** **Experiment No.:9**

Aim: To perform network sniffing using wire shark tool

Resources needed: Wire shark tool

Theory

Wireshark is a network packet analyzer. Any network packet analyzer will try to capture network packets and will try to display that packet data as detailed as possible in human readable format. Wireshark is an open source software project, and is released under the GNU General Public License (GPL). We can freely use Wireshark on any number of computers, without worrying about license keys. In addition, all source code is freely available under the GPL. Because of that, it is very easy for people to add new protocols to Wireshark, either as plug-in, or built into the source code. In the past, such tools were either very expensive, proprietary. However, with the advent of Wire-shark, all that has changed. Wireshark is perhaps one of the best open source packet analyzers available today.

What Wireshark is not......

Here are some things Wireshark does not provide:
1. Wireshark isn’t an intrusion detection system. It will not warn us when someone does strange things on our network that he/she isn’t allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.
2. Wireshark will not manipulate things on the network, it will only "measure" things from it. Wireshark doesn’t send packets on the network or do other active things.

Applications of Wireshark:
Here are some applications. Many people use Wireshark for doing following things,

- Network administrators use it to troubleshoot network problems.
- Network security engineers use it to examine security problems (Network Forensics.)
- Developers use it to debug protocol implementations.
- People use it to learn network protocol internals.

Beside these examples Wireshark can be helpful in many other situations too.

Features of Wireshark:
The following are some of the many features Wireshark has:

- Available for UNIX and Windows operating systems.
- Capture live packet data from a chosen network interface.
- Open files containing packet data captured with tcpdump/WinDump and a number of other packet capture programs.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Save packet data captured.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.

- Create various statistics.

  ……and a lot more!

Most important menus are : 1) Capture 2) Analyze 3) Statistics
Students are expected to explore all these menus and sub-menus in details.

Wireshark can capture traffic from many different network media types including wireless LAN as well. Which media types are supported, depends on many things like the operating system we are using and the hardware support.

Physical interfaces supported:

- ATM - capture ATM traffic
- Bluetooth- capture Bluetooth traffic .
- Cisco HDLC links - capture on synchronous links using Cisco HDLC encapsulation.
- Ethernet- capture on different topologies, including switched networks.
- Framerelay – captures framerelay traffic.
- IrDA capture IrDA traffic - currently limited to Linux.
- PPP links - capture on dial-up lines, ISDN connections and PPP-over-Ethernet (PPPoe, e.g. ADSL)
- Tokenring - capture on Tokenring adapters, promiscuous mode and switched networks
- USB- capture of raw USB traffic
- WLAN- capture on 802.11 (WLAN, Wi-Fi) interfaces, including "monitor mode" , raw 802.11 headers and radio information

Virtual interfaces :

- Loopbak - capture traffic from a machine to itself, including the IP address 127.0.0.1
- Pipes - use UNIX pipes to capture from other applications (even remote!)
- VLAN – capture VLAN traffic, including VLAN tags.

In addition to this, Wireshark can do following things.

- Import files from many other capture programs.
- Wireshark can open packets captured from a large number of other capture programs.
- Export files for many other capture programs.
- Wireshark can save packets captured in a large number of formats of other capture programs.
- Can be used as a protocol decoder.

Procedure / Approach /Algorithm / Activity Diagram:

1. Go to the official website of Wire shark (www.wireshark.org) and download the stable version of Wire shark for 64 bit windows operating system.
2. After successful installation you will get the blue icon of Wire shark on the desktop.
3. Click on the icon and start the software.
4. Choose an interface and start capturing the packets.
5. Study the packet details of all the protocols.
6. Understand colour code in details.

7. Perform the statistics for a particular protocol. (Every student should perform for different protocol).

Implementation:

Task1: Design your own registration and login pages (along with user database of registered users)

Task2: Run wire shark and capture the login page request data using wire shark and locate the captured password.

Questions:

1. **What is the difference between Burp suite and Wire shark tools?**
   Ans:
   Burp Suite and Wireshark are both popular tools used in the field of cybersecurity and network analysis, but they serve different purposes and have distinct functionalities. Here are the key differences between Burp Suite and Wireshark:

   1. Purpose:
   - Burp Suite: Burp Suite is a cybersecurity tool primarily used for web application security testing, including web vulnerability scanning, penetration testing, and security assessment of web applications.
   - Wireshark: Wireshark, on the other hand, is a network protocol analyzer. It's used for monitoring and analyzing network traffic at the packet level. Wireshark is not specifically designed for web application security testing but for broader network analysis.

   2. Scope:
   - Burp Suite: Burp Suite is typically used for assessing the security of web applications, focusing on issues like cross-site scripting (XSS), SQL injection, security misconfigurations, and other web-related vulnerabilities.
   - Wireshark: Wireshark is used to capture and analyze network traffic, which can be applied to a wide range of network-related tasks, including troubleshooting, debugging, and understanding network protocols.

   3. User Interface:
   - Burp Suite: Burp Suite has a user-friendly graphical interface tailored for web application testing. It provides various tools and features for intercept.

2. **Suggest the methods and/or security mechanisms to protect the password being leaked using tools like wireshark.**
   Ans: Protecting passwords from being leaked when using network analysis tools like Wireshark is crucial for maintaining the security of your systems and data. Here are some methods and security mechanisms to help prevent password leaks in network traffic:

   1. Use HTTPS: Whenever possible, use HTTPS for your web applications and services. HTTPS encrypts the data transferred between the client and server, including passwords. It ensures that the data, including login credentials, is secure in transit.

   2. Implement Strong Authentication: Use strong authentication methods, such as multi-factor authentication (MFA). Even if a password is intercepted, an attacker would still need additional authentication factors to gain access.

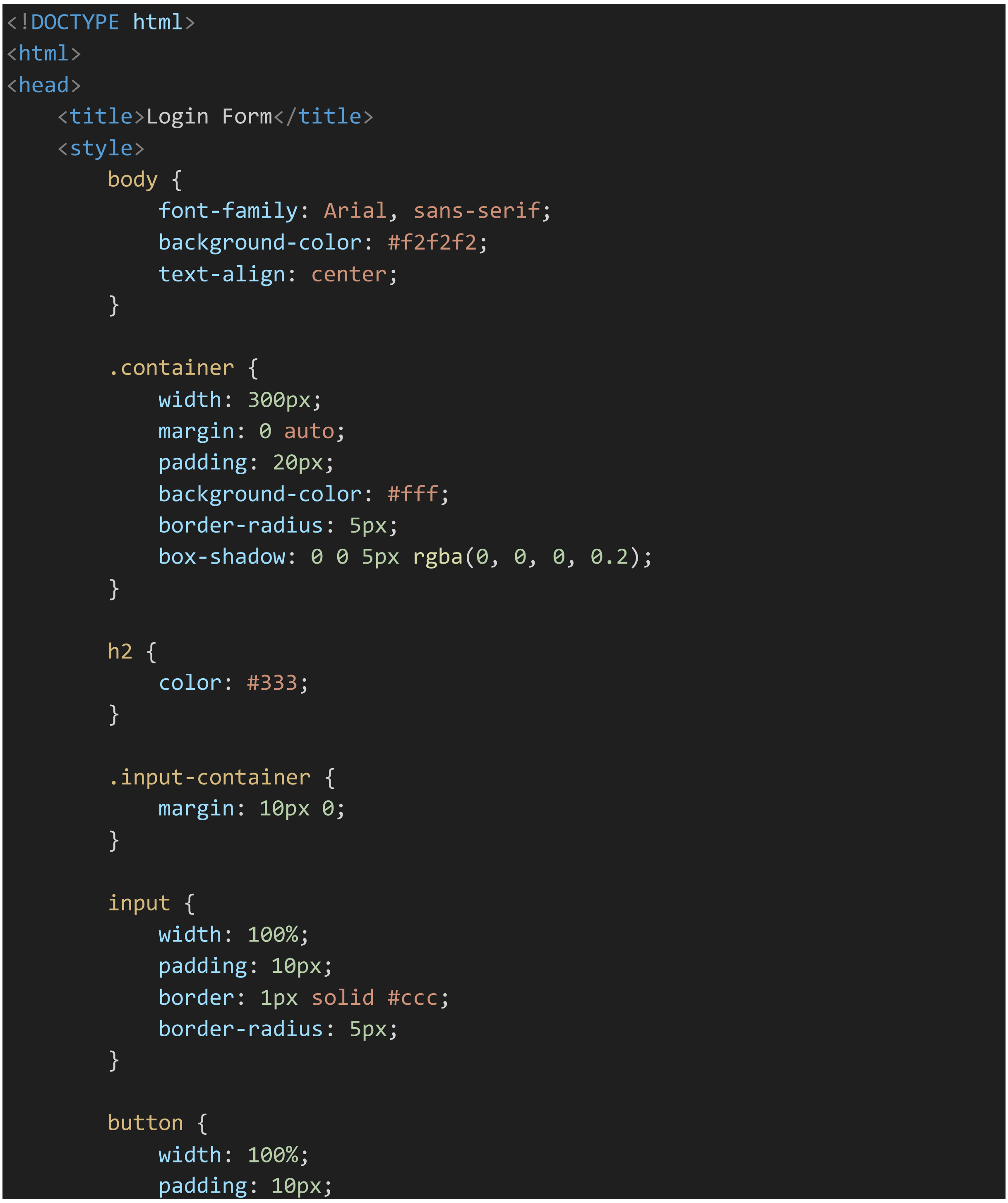
3. Hashed Passwords: Store passwords securely on the server using strong, one-way cryptographic hash functions. Don't store plain text passwords. When a user logs in, hash their input and compare it to the stored hash.

4. Salting Passwords: Add a unique salt value to each password before hashing it. Salting ensures that even if two users have the same password, their hashed values will differ, making it harder for attackers to use precomputed tables (rainbow tables) to crack passwords.

5. Credential Policies: Enforce strong password policies that require users to choose complex, unique passwords. Educate users about password best practices.

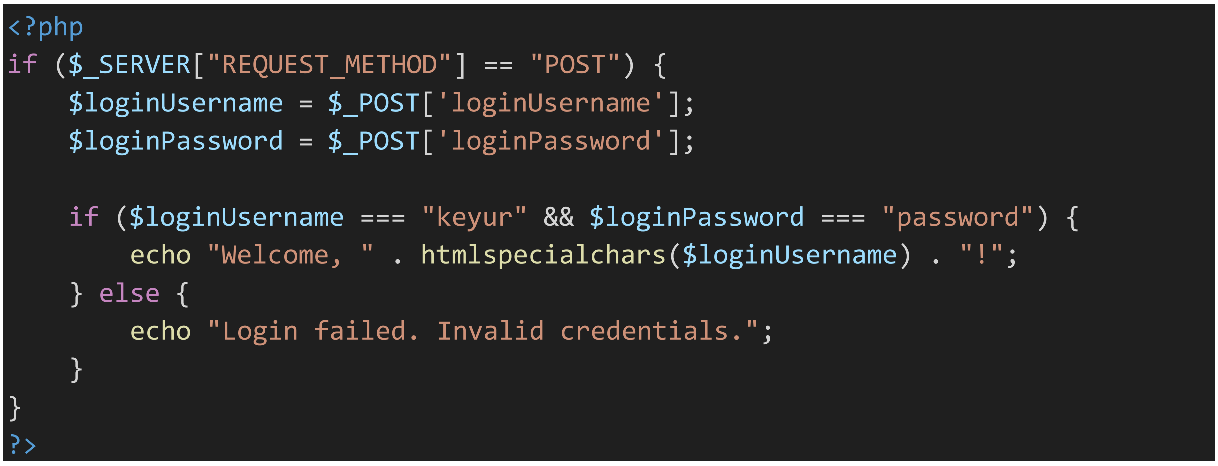
By implementing these measures, you can significantly reduce the risk of passwords being leaked through tools like Wireshark and enhance the overall security of your network and systems.

Result: Task 1 implementation code and Task 2 screenshots.
Code:
Index.php



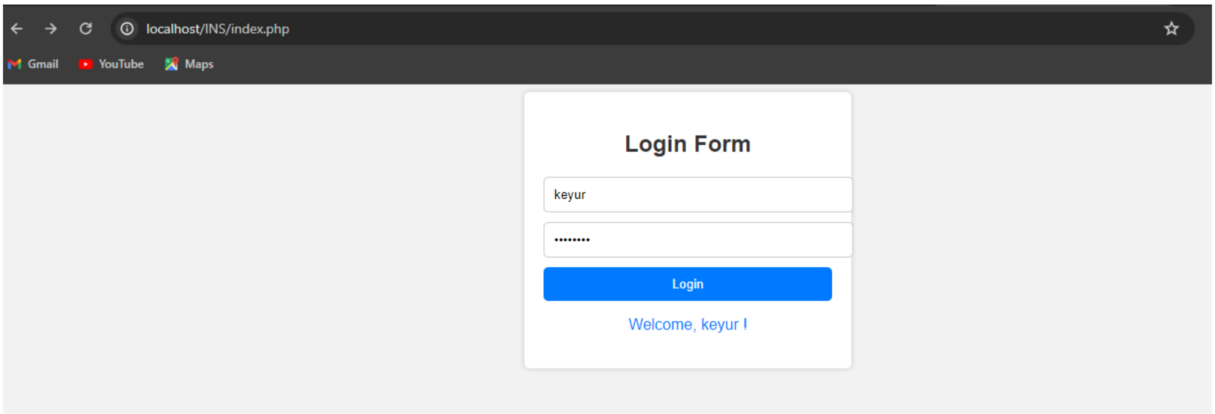
```
<!DOCTYPE html>
<html>
<head>
    <title>Login Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f2f2f2;
            text-align: center;
        }

        .container {
            width: 300px;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 5px;
            box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
        }

        h2 {
            color: #333;
        }

        .input-container {
            margin: 10px 0;
        }

        input {
            width: 100%;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        button {
            width: 100%;
            padding: 10px;
            background-color: #007BFF;
            color: #fff;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        #responseMessage {
            color: #007BFF;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Login Form</h2>
        <div id="loginForm">
            <form action="login.php" method="post">
                <div class="input-container">
                    <input type="text" id="loginUsername" name="loginUsername"
placeholder="Username" required>
                </div>
                <div class="input-container">
                    <input type="password" id="loginPassword"
name="loginPassword" placeholder="Password" required>
                </div>
                <button type="submit" id="loginButton">Login</button>
            </form>
        </div>

        <p id="responseMessage"></p>
    </div>

    <script>
        document.getElementById('loginButton').addEventListener('click',
function () {
            var loginUsername =
document.getElementById('loginUsername').value;
            var loginPassword =
document.getElementById('loginPassword').value;

            document.getElementById('responseMessage').textContent = "Welcome,
" + loginUsername + "!";
        });
    </script>
</body>
</html>
```

Login.php



```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $loginUsername = $_POST['loginUsername'];
    $loginPassword = $_POST['loginPassword'];

    if ($loginUsername === "keyur" && $loginPassword === "password") {
        echo "Welcome, " . htmlspecialchars($loginUsername) . "!";
    } else {
        echo "Login failed. Invalid credentials.";
    }
}
?>
```

Output:

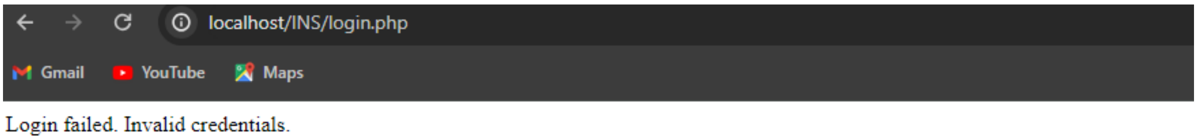


Login Form

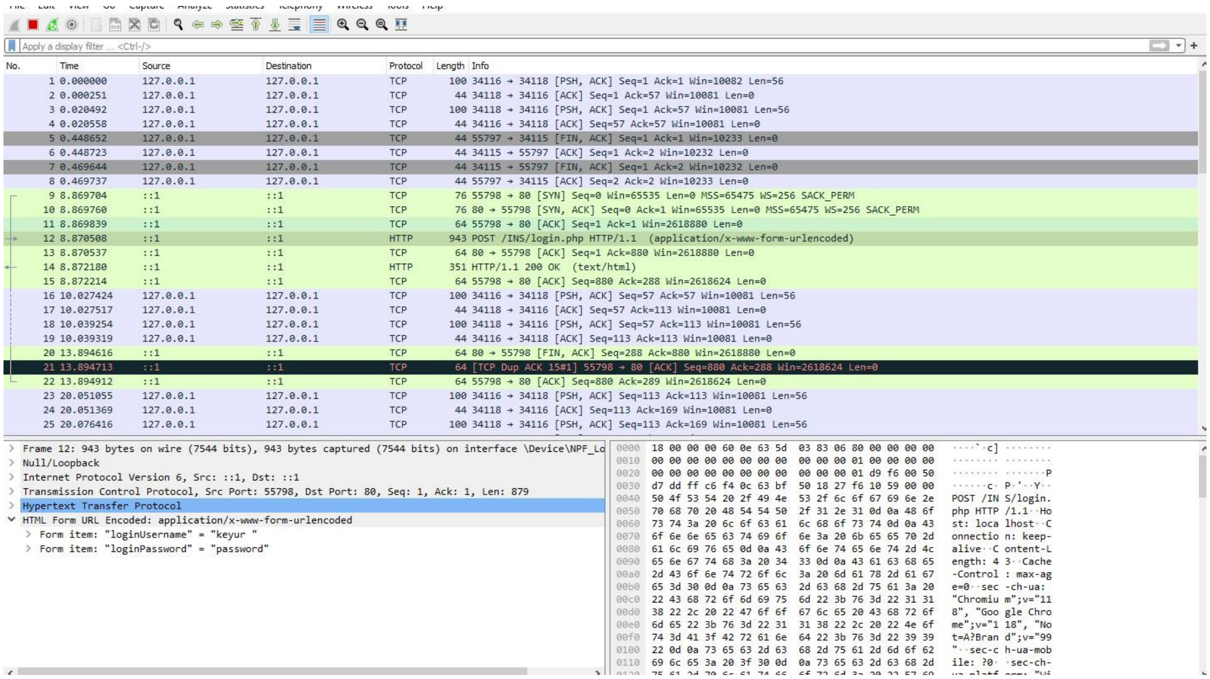
keyur

••••••••

Welcome, keyur !



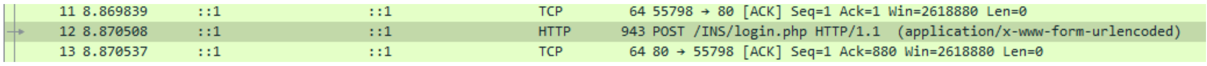
Login failed. Invalid credentials.



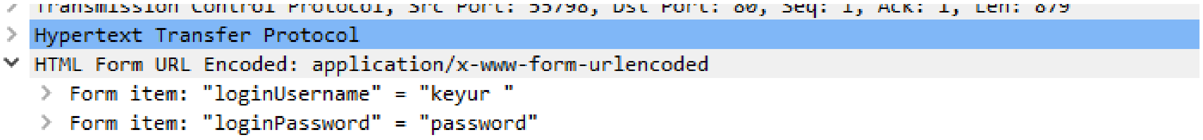
```
No.   Time        Source      Destination  Protocol  Length Info
    1 0.000000    127.0.0.1   127.0.0.1    TCP        100 34116 → 34118 [PSH, ACK] Seq=1 Ack=1 Win=10082 Len=56
    2 0.000251    127.0.0.1   127.0.0.1    TCP         44 34118 → 34116 [ACK] Seq=1 Ack=57 Win=10081 Len=0
    3 0.020492    127.0.0.1   127.0.0.1    TCP        100 34118 → 34116 [PSH, ACK] Seq=1 Ack=57 Win=10081 Len=56
    4 0.020558    127.0.0.1   127.0.0.1    TCP         44 34116 → 34118 [ACK] Seq=57 Ack=57 Win=10081 Len=0
    5 0.448652    127.0.0.1   127.0.0.1    TCP         44 55797 → 34115 [FIN, ACK] Seq=1 Ack=1 Win=10233 Len=0
    6 0.448723    127.0.0.1   127.0.0.1    TCP         44 34115 → 55797 [ACK] Seq=1 Ack=2 Win=10232 Len=0
    7 0.469644    127.0.0.1   127.0.0.1    TCP         44 34115 → 55797 [FIN, ACK] Seq=1 Ack=2 Win=10232 Len=0
    8 0.469737    127.0.0.1   127.0.0.1    TCP         44 55797 → 34115 [ACK] Seq=2 Ack=2 Win=10233 Len=0
    9 8.869704    ::1         ::1          TCP         76 55798 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65475 WS=256 SACK_PERM
   10 8.869760    ::1         ::1          TCP         76 80 → 55798 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65475 WS=256 SACK_PERM
   11 8.869839    ::1         ::1          TCP         64 55798 → 80 [ACK] Seq=1 Ack=1 Win=2618880 Len=0
   12 8.870508    ::1         ::1          HTTP       943 POST /INS/login.php HTTP/1.1  (application/x-www-form-urlencoded)
   13 8.870537    ::1         ::1          TCP         64 80 → 55798 [ACK] Seq=1 Ack=880 Win=2618880 Len=0
   14 8.872180    ::1         ::1          HTTP       351 HTTP/1.1 200 OK  (text/html)
   15 8.872214    ::1         ::1          TCP         64 55798 → 80 [ACK] Seq=880 Ack=288 Win=2618624 Len=0
   16 10.027424   127.0.0.1   127.0.0.1    TCP        100 34116 → 34118 [PSH, ACK] Seq=57 Ack=57 Win=10081 Len=56
   17 10.027517   127.0.0.1   127.0.0.1    TCP         44 34118 → 34116 [ACK] Seq=57 Ack=113 Win=10081 Len=0
   18 10.039254   127.0.0.1   127.0.0.1    TCP        100 34118 → 34116 [PSH, ACK] Seq=57 Ack=113 Win=10081 Len=56
   19 10.039319   127.0.0.1   127.0.0.1    TCP         44 34116 → 34118 [ACK] Seq=113 Ack=113 Win=10081 Len=0
   20 13.894616   ::1         ::1          TCP         64 80 → 55798 [FIN, ACK] Seq=288 Ack=880 Win=2618880 Len=0
   21 13.894713   ::1         ::1          TCP         64 [TCP Dup ACK 15#1] 55798 → 80 [ACK] Seq=880 Ack=288 Win=2618624 Len=0
   22 13.894912   ::1         ::1          TCP         64 55798 → 80 [ACK] Seq=880 Ack=289 Win=2618624 Len=0
   23 20.051055   127.0.0.1   127.0.0.1    TCP        100 34116 → 34118 [PSH, ACK] Seq=113 Ack=113 Win=10081 Len=56
   24 20.051369   127.0.0.1   127.0.0.1    TCP         44 34118 → 34116 [ACK] Seq=113 Ack=169 Win=10081 Len=0
   25 20.076416   127.0.0.1   127.0.0.1    TCP        100 34118 → 34116 [PSH, ACK] Seq=113 Ack=169 Win=10081 Len=56
```

```
> Frame 12: 943 bytes on wire (7544 bits), 943 bytes captured (7544 bits) on interface \Device\NPF_Lo
> Null/Loopback
> Internet Protocol Version 6, Src: ::1, Dst: ::1
> Transmission Control Protocol, Src Port: 55798, Dst Port: 80, Seq: 1, Ack: 1, Len: 879
> Hypertext Transfer Protocol
v HTML Form URL Encoded: application/x-www-form-urlencoded
   > Form item: "loginUsername" = "keyur "
   > Form item: "loginPassword" = "password"
```

```
0000  18 00 00 00 60 0e 63 5d  03 83 06 80 00 00 00 00   ····`·c] ········
0010  00 00 00 00 00 00 00 00  00 00 00 01 00 00 00 00   ········ ········
0020  00 00 00 00 00 00 00 00  00 00 00 01 d9 f6 00 50   ········ ·······P
0030  d7 dd ff c6 f4 0c 63 bf  50 18 27 f6 10 59 00 00   ······c· P·'··Y··
0040  50 4f 53 54 20 2f 49 4e  53 2f 6c 6f 67 69 6e 2e   POST /IN S/login.
0050  70 68 70 20 48 54 54 50  2f 31 2e 31 0d 0a 48 6f   php HTTP /1.1··Ho
0060  73 74 3a 20 6c 6f 63 61  6c 68 6f 73 74 0d 0a 43   st: loca lhost··C
0070  6f 6e 6e 65 63 74 69 6f  6e 3a 20 6b 65 65 70 2d   onnectio n: keep-
0080  61 6c 69 76 65 0d 0a 43  6f 6e 74 65 6e 74 2d 4c   alive··C ontent-L
0090  65 6e 67 74 68 3a 20 34  33 0d 0a 43 61 63 68 65   ength: 4 3··Cache
00a0  2d 43 6f 6e 74 72 6f 6c  3a 20 6d 61 78 2d 61 67   -Control : max-ag
00b0  65 3d 30 0d 0a 73 65 63  2d 63 68 2d 75 61 3a 20   e=0··sec -ch-ua: 
00c0  22 43 68 72 6f 6d 69 75  6d 22 3b 76 3d 22 31 31   "Chromiu m";v="11
00d0  38 22 2c 20 22 47 6f 6f  67 6c 65 20 43 68 72 6f   8", "Goo gle Chro
00e0  6d 65 22 3b 76 3d 22 31  31 38 22 2c 20 22 4e 6f   me";v="1 18", "No
00f0  74 3d 41 3f 42 72 61 6e  64 22 3b 76 3d 22 39 39   t=A?Bran d";v="99
0100  22 0d 0a 73 65 63 2d 63  68 2d 75 61 2d 6d 6f 62   "··sec-c h-ua-mob
0110  69 6c 65 3a 20 3f 30 0d  0a 73 65 63 2d 63 68 2d   ile: ?0· ·sec-ch-
```



```
   11 8.869839    ::1         ::1          TCP         64 55798 → 80 [ACK] Seq=1 Ack=1 Win=2618880 Len=0
   12 8.870508    ::1         ::1          HTTP       943 POST /INS/login.php HTTP/1.1  (application/x-www-form-urlencoded)
   13 8.870537    ::1         ::1          TCP         64 80 → 55798 [ACK] Seq=1 Ack=880 Win=2618880 Len=0
```



```
> Transmission Control Protocol, Src Port: 55798, Dst Port: 80, Seq: 1, Ack: 1, Len: 879
> Hypertext Transfer Protocol
v HTML Form URL Encoded: application/x-www-form-urlencoded
   > Form item: "loginUsername" = "keyur "
   > Form item: "loginPassword" = "password"
```

Outcomes:

CO4 : Understand Security issues related to Software, Web and Networks.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved**)**

Thus we performed network sniffing using wireshark tool and captured to login page request data and also password.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. **https://www.wireshark.org/ _(software)**
2. **https://en.wikipedia.org/wiki/Wireshark**
3. https://www.wireshark.org/docs/
4. https://www.youtube.com/watch?v=UBfSgjUCEi0