Experiment No. 9

Title: Network Sniffing - Wireshark

Batch: B2 **Roll No.:16010421119** **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?
2. Suggest the methods and/or security mechanisms to protect the password being leaked using tools like wireshark.

Result:

Task 1:

//auth.html

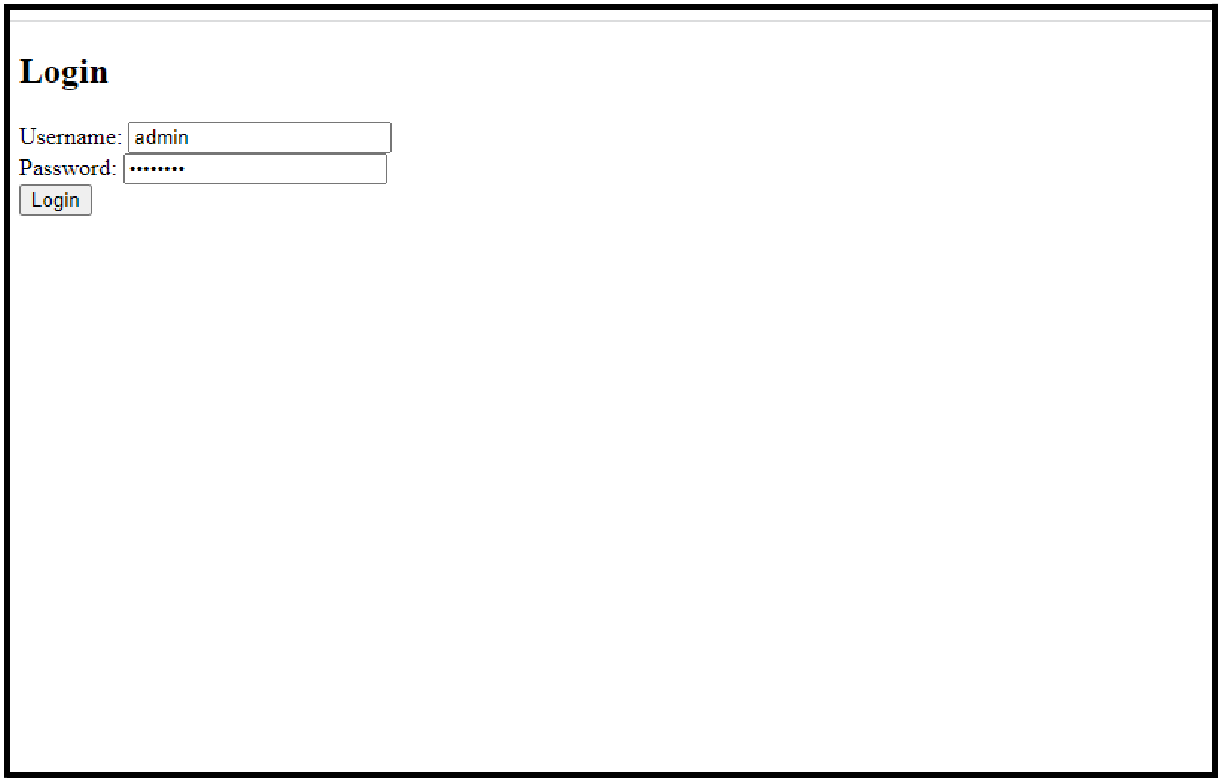
```
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
</head>
<body>
  <h2>Login</h2>
  <form action="login.php" method="POST">
```

```
    <label for="username">Username:</label>
    <input type="text" name="username" required><br>

    <label for "password">Password:</label>
    <input type="password" name="password" required><br>

    <input type="submit" value="Login">
  </form>
</body>
</html>
```

//authlogin.php

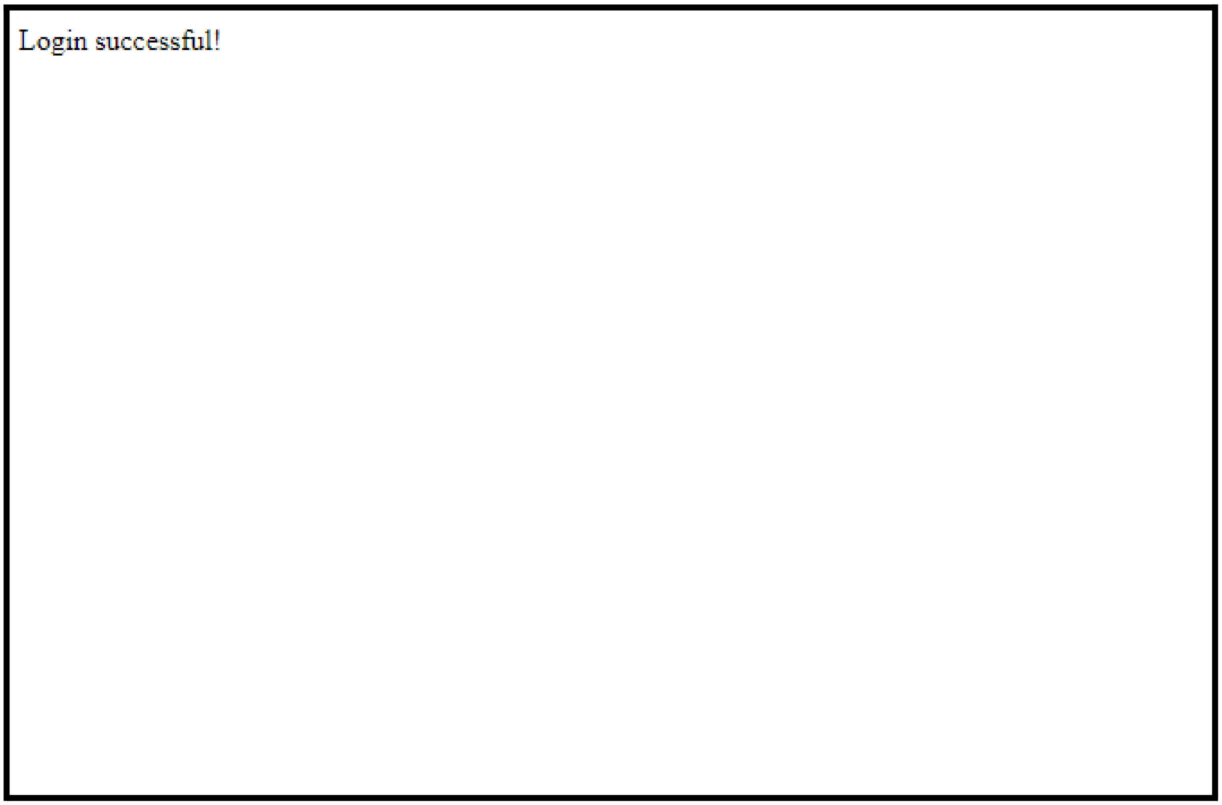
```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  session_start();

  // Establish a MySQL connection (replace with your own database
  credentials)
  $conn = new mysqli("localhost", "username", "password", "mydb");

  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }

  $username = $_POST["username"];
  $password = $_POST["password"];

  $stmt = $conn->prepare("SELECT id, password FROM users WHERE
  username = ?");
  $stmt->bind_param("s", $username);

  if ($stmt->execute()) {
    $stmt->store_result();

    if ($stmt->num_rows == 1) {
      $stmt->bind_result($id, $stored_password);
      $stmt->fetch();

      if ($password === $stored_password) {
        $_SESSION['user_id'
        ] = $id; echo
        "Login
        successful!";
      } else {
        echo "Login failed. Incorrect password.";
```

```
}
    } else {
      echo "Login failed. User not found.";
    }
  } else {
    echo "Login failed: " . $stmt->error;
  }

  $stmt->close();
  $conn->close();
}
?>
```



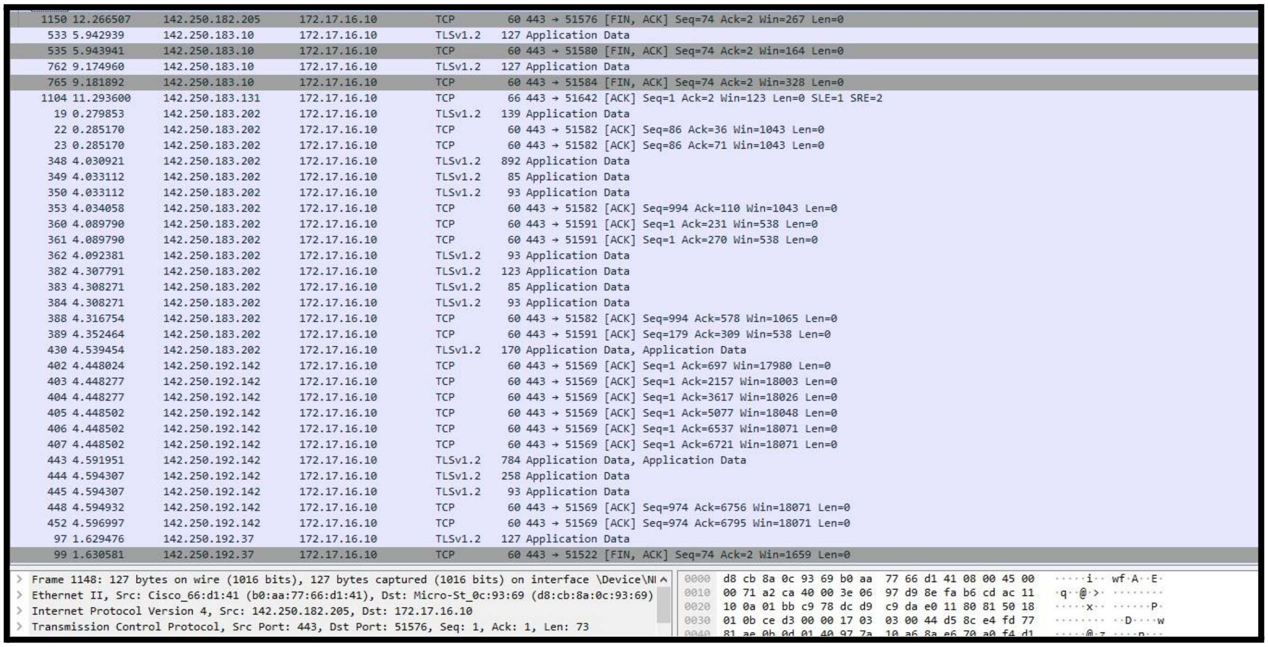
Login

Username: admin
Password: ••••••••

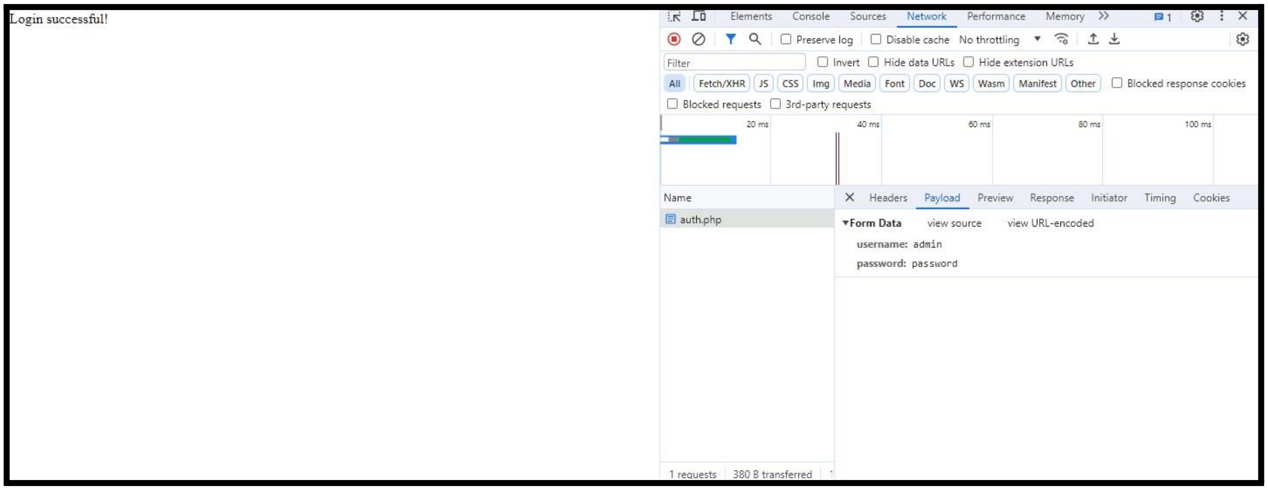


Login successful!

Task 2:



```
1150 12.266507    142.250.182.205    172.17.16.10       TCP        60 443 → 51576 [FIN, ACK] Seq=74 Ack=2 Win=267 Len=0
 533 5.942939     142.250.183.10     172.17.16.10       TLSv1.2   127 Application Data
 535 5.943941     142.250.183.10     172.17.16.10       TCP        60 443 → 51580 [FIN, ACK] Seq=74 Ack=2 Win=164 Len=0
 762 9.174960     142.250.183.10     172.17.16.10       TLSv1.2   127 Application Data
 765 9.181892     142.250.183.10     172.17.16.10       TCP        60 443 → 51584 [FIN, ACK] Seq=74 Ack=2 Win=328 Len=0
1104 11.293600    142.250.183.131    172.17.16.10       TCP        66 443 → 51642 [ACK] Seq=1 Ack=2 Win=123 Len=0 SLE=1 SRE=2
  19 0.279853     142.250.183.202    172.17.16.10       TLSv1.2   139 Application Data
  22 0.285170     142.250.183.202    172.17.16.10       TCP        60 443 → 51582 [ACK] Seq=86 Ack=36 Win=1043 Len=0
  23 0.285170     142.250.183.202    172.17.16.10       TCP        60 443 → 51582 [ACK] Seq=86 Ack=71 Win=1043 Len=0
 348 4.030921     142.250.183.202    172.17.16.10       TLSv1.2   892 Application Data
 349 4.033112     142.250.183.202    172.17.16.10       TLSv1.2    85 Application Data
 350 4.033112     142.250.183.202    172.17.16.10       TLSv1.2    93 Application Data
 353 4.034058     142.250.183.202    172.17.16.10       TCP        60 443 → 51582 [ACK] Seq=994 Ack=110 Win=1043 Len=0
 360 4.089790     142.250.183.202    172.17.16.10       TCP        60 443 → 51591 [ACK] Seq=1 Ack=231 Win=538 Len=0
 361 4.089790     142.250.183.202    172.17.16.10       TCP        60 443 → 51591 [ACK] Seq=1 Ack=270 Win=538 Len=0
 362 4.092381     142.250.183.202    172.17.16.10       TLSv1.2    93 Application Data
 382 4.307791     142.250.183.202    172.17.16.10       TLSv1.2   123 Application Data
 383 4.308271     142.250.183.202    172.17.16.10       TLSv1.2    85 Application Data
 384 4.308271     142.250.183.202    172.17.16.10       TLSv1.2    93 Application Data
 388 4.316754     142.250.183.202    172.17.16.10       TCP        60 443 → 51582 [ACK] Seq=994 Ack=578 Win=1065 Len=0
 389 4.352464     142.250.183.202    172.17.16.10       TCP        60 443 → 51591 [ACK] Seq=179 Ack=309 Win=538 Len=0
 430 4.539454     142.250.183.202    172.17.16.10       TLSv1.2   170 Application Data, Application Data
 402 4.448024     142.250.192.142    172.17.16.10       TCP        60 443 → 51569 [ACK] Seq=1 Ack=697 Win=17980 Len=0
 403 4.448277     142.250.192.142    172.17.16.10       TCP        60 443 → 51569 [ACK] Seq=1 Ack=2157 Win=18003 Len=0
 404 4.448277     142.250.192.142    172.17.16.10       TCP        60 443 → 51569 [ACK] Seq=1 Ack=3617 Win=18026 Len=0
 405 4.448502     142.250.192.142    172.17.16.10       TCP        60 443 → 51569 [ACK] Seq=1 Ack=5077 Win=18048 Len=0
 406 4.448502     142.250.192.142    172.17.16.10       TCP        60 443 → 51569 [ACK] Seq=1 Ack=6537 Win=18071 Len=0
 407 4.448502     142.250.192.142    172.17.16.10       TCP        60 443 → 51569 [ACK] Seq=1 Ack=6721 Win=18071 Len=0
 443 4.591951     142.250.192.142    172.17.16.10       TLSv1.2   784 Application Data, Application Data
 444 4.594307     142.250.192.142    172.17.16.10       TLSv1.2   258 Application Data
 445 4.594307     142.250.192.142    172.17.16.10       TLSv1.2    93 Application Data
 448 4.594932     142.250.192.142    172.17.16.10       TCP        60 443 → 51569 [ACK] Seq=974 Ack=6756 Win=18071 Len=0
 452 4.596997     142.250.192.142    172.17.16.10       TCP        60 443 → 51569 [ACK] Seq=974 Ack=6795 Win=18071 Len=0
  97 1.629476     142.250.192.37     172.17.16.10       TLSv1.2   127 Application Data
  99 1.630581     142.250.192.37     172.17.16.10       TCP        60 443 → 51522 [FIN, ACK] Seq=74 Ack=2 Win=1659 Len=0

> Frame 1148: 127 bytes on wire (1016 bits), 127 bytes captured (1016 bits) on interface \Device\NI
> Ethernet II, Src: Cisco_66:d1:41 (b0:aa:77:66:d1:41), Dst: Micro-St_0c:93:69 (d8:cb:8a:0c:93:69)
> Internet Protocol Version 4, Src: 142.250.182.205, Dst: 172.17.16.10
> Transmission Control Protocol, Src Port: 443, Dst Port: 51576, Seq: 1, Ack: 1, Len: 73

0000  d8 cb 8a 0c 93 69 b0 aa  77 66 d1 41 08 00 45 00   ·····i·· wf·A··E·
0010  00 71 a2 ca 40 00 3e 06  97 d9 8e fa b6 cd ac 11   ·q··@·>· ········
0020  10 0a 01 bb c9 78 dc d9  c9 da e0 11 80 81 50 18   ·····x·· ······P·
0030  01 0b ce d3 00 00 17 03  03 00 44 d5 8c e4 fd 77   ········ ··D····w
0040  81 ae 0b 0d 01 40 97 7a  10 a6 8a e6 70 a0 f4 d1   ·····@·z ····p···
```



Login successful!

auth.php

▾Form Data

username: admin

password: password

Outcome:

CO 4 *Understand Security issues related to Software, Web and Networks.*

Conclusion:

Learned and implemented a simple login page in php to understand authorisation and network sniffing using wireshark.

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