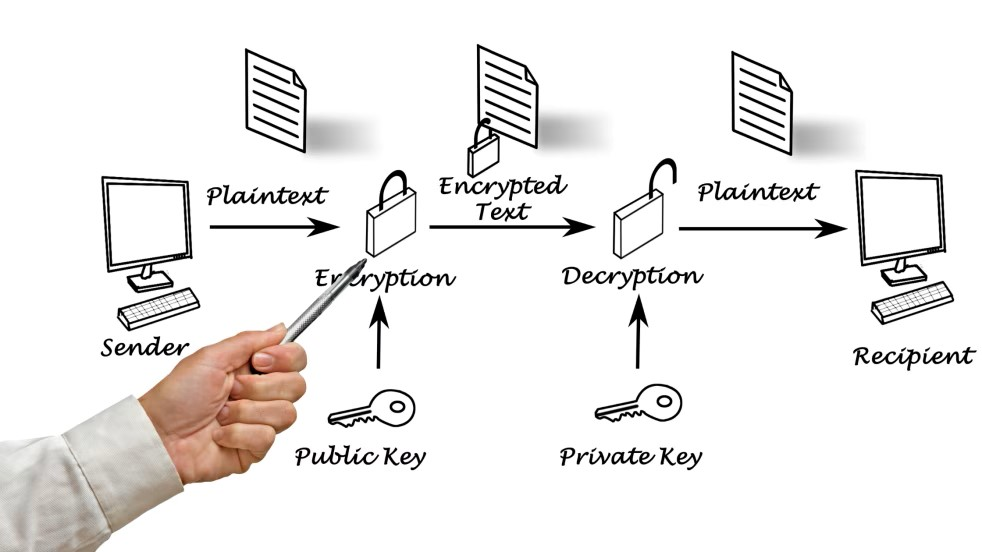
**PACKET SNIFFING MONITORING SYSTEM**

**What Is a Packet Sniffing Attack?**

**A packet sniffing attack (or simply a sniffing attack) is a network-created threat where a malicious entity captures network packets intending to intercept or steal data traffic that may have been left unencrypted.**

**Sniffing attacks are data thefts perpetrated by capturing network traffic with packet sniffers, which can illegally access and read unencrypted data. The data packets are collected when they pass through a computer network. The sniffing devices or media used to perform this sniffing attack and collect network data packets are known as packet sniffers.**

**Generally speaking, a packet sniffer refers to hardware or software that keeps track of network traffic by capturing packets. It is also known as a packet analyzer, protocol analyzer, or network analyzer. Sniffers analyse data packet streams that pass between computers on a network as well as between networked systems and the Internet. These packets are designed for specific machines, but utilizing a packet sniffer in “promiscuous mode,” IT professionals, end-users, or malevolent intruders can inspect any packet, regardless of destination.**

****

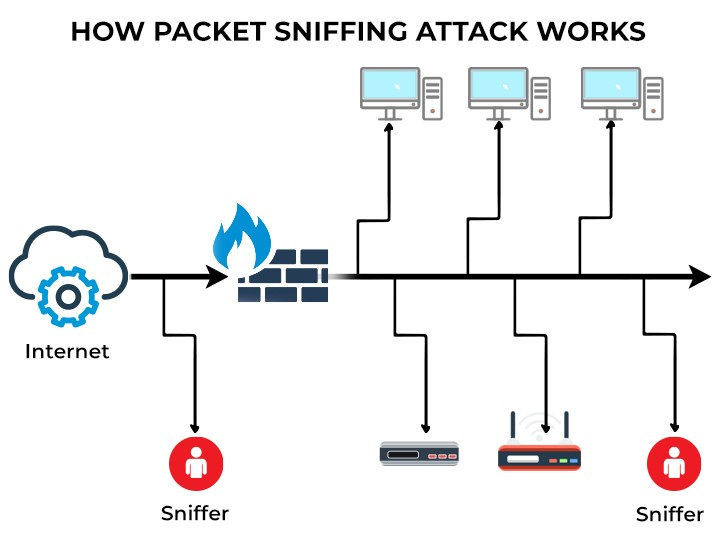
**How does packet sniffing work?**

### 

**A network interface card (NIC) is a hardware component that contains a circuit board in every computer network. By default, NICs ignore non-addressed traffic. Sniffing attacks require the NICs to be set to promiscuous mode, which allows the NICs to receive all network traffic.**

**The act of sniffing can be classified into two types: active and passive.**

* **Active sniffing: It is the technique of inserting address resolution protocols (ARPs) into a network to overload the switch content address memory (CAM) table. As a result, legitimate traffic is redirected to other ports, allowing the attacker to sniff traffic from the switch. Active sniffing methodologies are used for spoofing attacks, dynamic host configuration protocol (DHCP) attacks, and domain name system (DNS) poisonings.**

****

**Passive sniffing: It consists solely of listening and is typically used in networks connected by hubs. The traffic is visible to all hosts in this form of network. To discreetly monitor a company’s network, hackers will often employ one of two approaches to passive sniffing.**

* **In the case of organisations that use hubs to connect several devices on a single network, hackers can use a sniffer to passively “spy” on all the traffic flowing through the system. This sort of passive sniffing is incredibly difficult to detect.**

**What is packet sniffing monitoring system**

**packet sniffing monitoring system has many uses, including TROUBLESHOOTING NETWORKS that have performance issues. Cybersecurity professionals often use Wireshark to trace connections, view the contents of suspect network transactions and identify bursts of network traffic.**

**In this project the packet sniffing monitoring system that we have used is WIRESHARK.**

**WIRESHARK**

**Wireshark is a network protocol analyzer, or an application that captures packets from a network connection, such as from your computer to your home office or the internet. Packet is the name given to a discrete unit of data in a typical Ethernet network.**

**Wireshark is the most often-used packet sniffer in the world. Like any other packet sniffer, Wireshark does three things:**

1. **Packet Capture: Wireshark listens to a network connection in real time and then grabs entire streams of traffic – quite possibly tens of thousands of packets at a time.**
2. **Filtering: Wireshark is capable of slicing and dicing all of this random live data using filters. By applying a filter, you can obtain just the information you need to see.**
3. **Visualization: Wireshark, like any good packet sniffer, allows you to dive right into the very middle of a network packet. It also allows you to visualize entire conversations and network streams.**

**FEATURES:-**

**Wireshark is a data capturing program that "understands" the structure (**[**encapsulation**](https://en.wikipedia.org/wiki/Encapsulation_(networking))**) of different networking protocols. It can parse and display the fields, along with their meanings as specified by different networking protocols. Wireshark uses** [**pcap**](https://en.wikipedia.org/wiki/Pcap) **to capture packets, so it can only capture packets on the types of networks that pcap supports.**

* **Data can be captured "from the wire" from a live network connection or read from a file of already-captured packets.**
* **Live data can be read from different types of networks, including** [**Ethernet**](https://en.wikipedia.org/wiki/Ethernet)**,** [**IEEE 802.11**](https://en.wikipedia.org/wiki/IEEE_802.11)**,** [**PPP**](https://en.wikipedia.org/wiki/Point-to-Point_Protocol)**, and** [**loopback**](https://en.wikipedia.org/wiki/Loopback)**.**
* **Captured network data can be browsed via a** [**GUI**](https://en.wikipedia.org/wiki/Graphical_user_interface)**, or via the terminal (**[**command line**](https://en.wikipedia.org/wiki/Command_line_interface)**) version of the utility, TShark.**
* **Captured files can be programmatically edited or converted via command-line switches to the "editcap" program.**
* **Data display can be refined using a display filter.**
* [**Plug-ins**](https://en.wikipedia.org/wiki/Plug-in_(computing)) **can be created for dissecting new protocols.**[**[23]**](https://en.wikipedia.org/wiki/Wireshark#cite_note-23)
* [**VoIP**](https://en.wikipedia.org/wiki/Voice_over_IP) **calls in the captured traffic can be detected. If encoded in a compatible encoding, the media flow can even be played.**
* **Raw** [**USB**](https://en.wikipedia.org/wiki/Universal_Serial_Bus) **traffic can be captured.**[**[24]**](https://en.wikipedia.org/wiki/Wireshark#cite_note-24)
* **Wireless connections can also be filtered as long as they traverse the monitored Ethernet.**
* **Various settings, timers, and filters can be set to provide the facility of filtering the output of the captured traffic.**

****

**Capturing raw network traffic from an interface requires elevated privileges on some platforms. For this reason, older versions of Wireshark and TShark often ran with** [**superuser**](https://en.wikipedia.org/wiki/Superuser) **privileges. Considering the huge number of protocol dissectors that are called when traffic is captured and recognizing the possibility of a bug in a dissector, a serious security risk can be posed. Due to the rather large number of vulnerabilities in the past (of which many have allowed remote code execution) and developers' doubts for better future development,** [**OpenBSD**](https://en.wikipedia.org/wiki/OpenBSD) **removed Ethereal from its ports tree prior to OpenBSD .**

**Elevated privileges are not needed for all operations. For example, an alternative is to run tcpdump or the *dumpcap* utility that comes with Wireshark with superuser privileges to capture packets into a file, and later analyze the packets by running Wireshark with restricted privileges. To emulate near realtime analysis, each captured file may be merged by *mergecap* into a growing file processed by Wireshark. On wireless networks, it is possible to use the** [**Aircrack**](https://en.wikipedia.org/wiki/Aircrack) **wireless security tools to capture . frames and read the resulting dump files with Wireshark.**

**FRONTEND CODE (USING CSS AND HTML)**

Login.html

<!DOCTYPE html>

<html>

<head>

    <title>Login</title>

    <link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

    <div id="frm">

        <h1>Login</h1>

        <form name="f1" action="authentication.php" onsubmit="return validation()" method="POST">

            <p>

                <label> UserName: </label>

                <input type="text" id="user" name="user" />

            </p>

            <p>

                <label> Password: </label>

                <input type="password" id="pass" name="pass" />

            </p>

            <p>

                <input type="submit" id="btn" value="Log" />

            </p>

        </form>

    </div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

    <title>Products</title>

    <link rel="stylesheet" type="text.css" href="style.css">

</head>

<body>

    <h1>Products</h1>

    <!-- List of products -->

    <div class="product">

        <h2>Product 1</h2>

        <p>Product description goes here.</p>

    </div>

    <!-- Add more product listings as needed -->

    <?php

    include('authentication.php');

    if ($authentication\_successful) {

        // Display the product page content

    } else {

        // Display the login form if not authenticated

        echo '

        <form name="f1" action="authentication.php" onsubmit="return validation()" method="POST">

            <p>

                <label> UserName: </label>

                <input type="text" id="user" name="user" />

            </p>

            <p>

                <label> Password: </label>

                <input type="password" id="pass" name="pass" />

            </p>

            <p>

                <input type="submit" id="btn" value="Log" />

            </p>

        </form>';

    }

    ?>

</body>

</html>

**BACKEND CODE (USING PHP )**

Authentication.php

<?php

    include('connection.php');

    $username = $\_POST['user'];

    $password = $\_POST['pass'];

        //to prevent from mysqli injection

        $username = stripcslashes($username);

        $password = stripcslashes($password);

        $username = mysqli\_real\_escape\_string($con, $username);

        $password = mysqli\_real\_escape\_string($con, $password);

        $sql = "select \*from log where username = '$username' and password = '$password'";

        $result = mysqli\_query($con, $sql);

        if ($result === false) {

            // handle the error

            die('Query failed: ' . mysqli\_error($con));

        }

        $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

    ?>

Connection.php

<?php

    $host = "localhost";

    $user = "root";

    $password = '';

    $db\_name = "php";

    $con = mysqli\_connect($host, $user, $password, $db\_name);

    if(mysqli\_connect\_errno()) {

        die("Failed to connect with MySQL: ". mysqli\_connect\_error());

    }

?>

**OUTPUT**

**STEPS FOLLOWED FOR OUTPUT:-**

**1. Login the web page.**

**2. authenticating the the password and user name.**

**3. connection between database and frontend.**

**4. Now coming to wireshark.**

**5. adding filter name.**

**6. after filtering the packets are there.**

**7. 1st packet tell about the network and connection.**

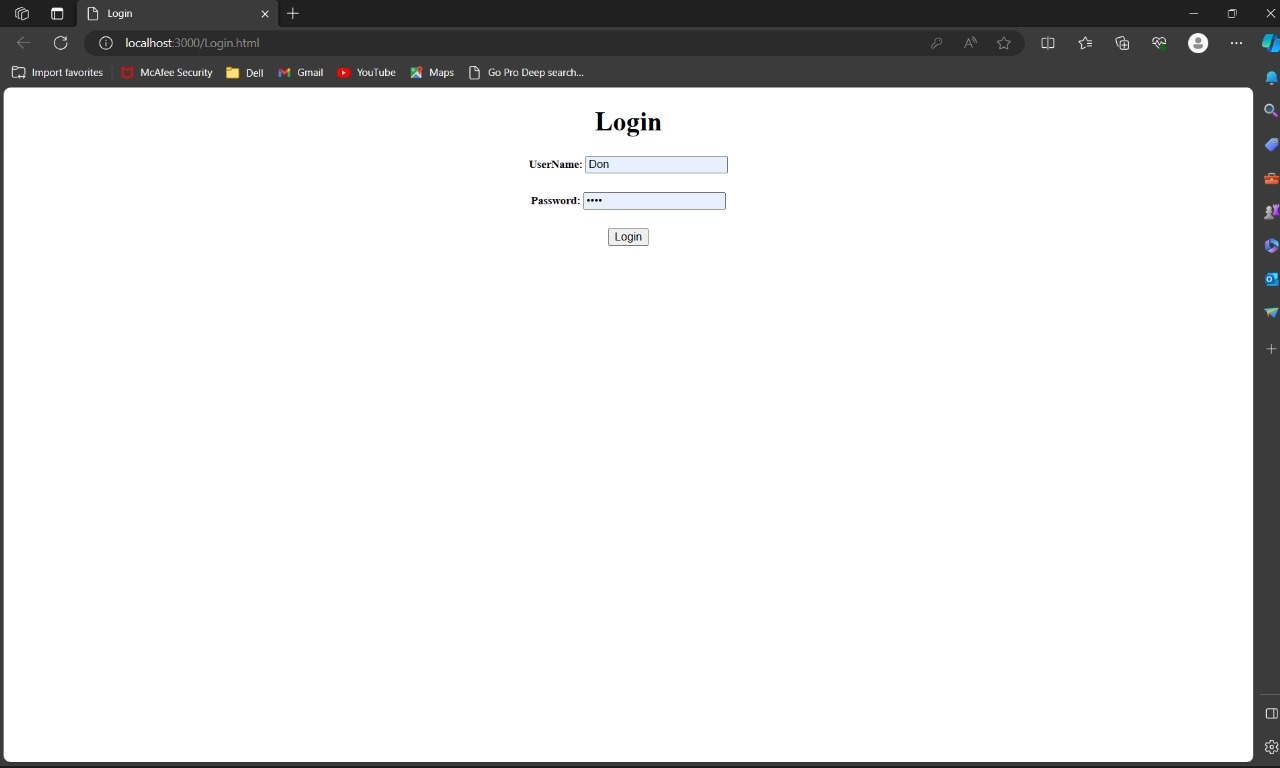
**8. 2nd packets tell about database.**

**9.3rd packet send request and quary.**

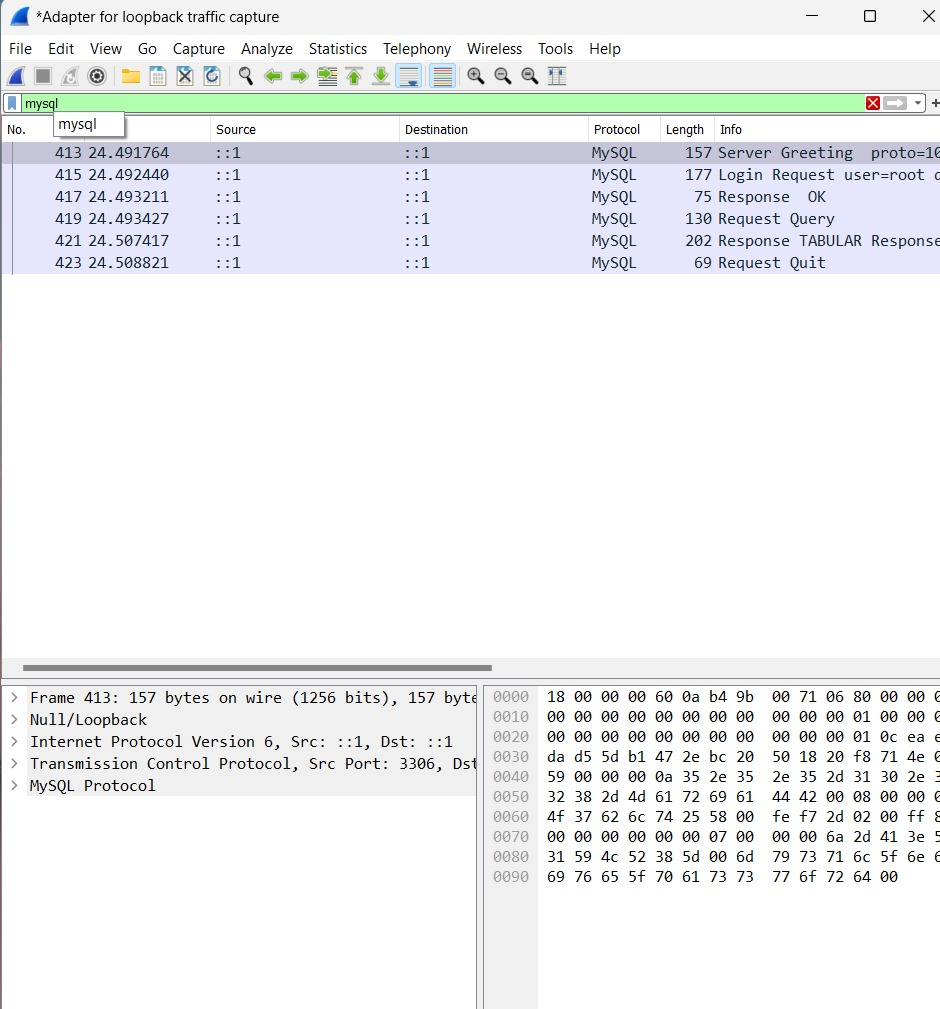
**10. 4th packet recives the data the username and password.**

**11. 5th packet tells us about whether the packet Is accepted or rejected.**

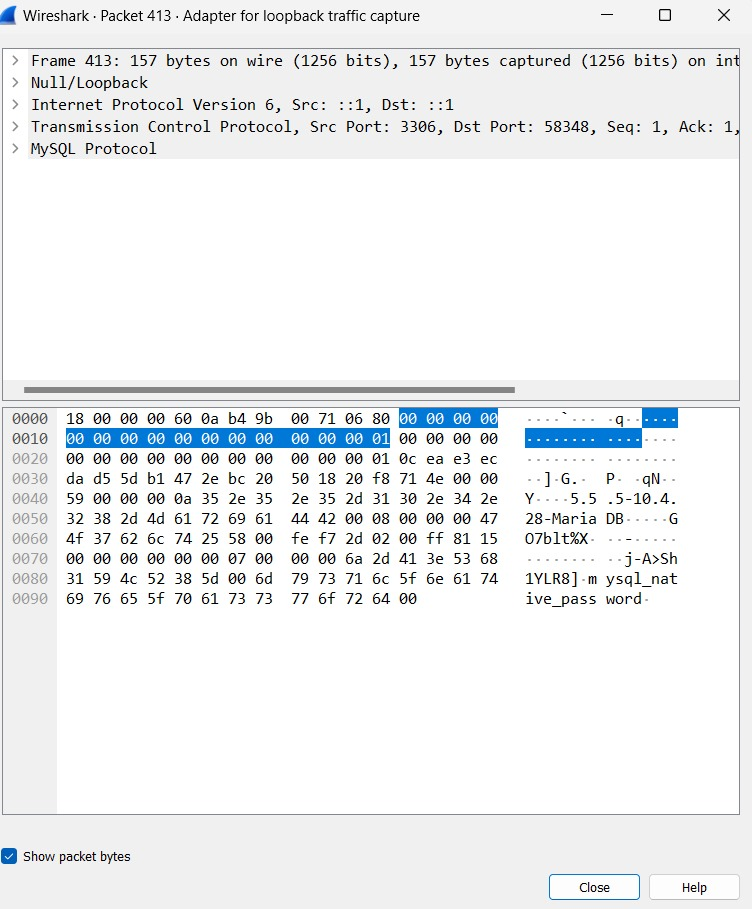
**1.)**

****

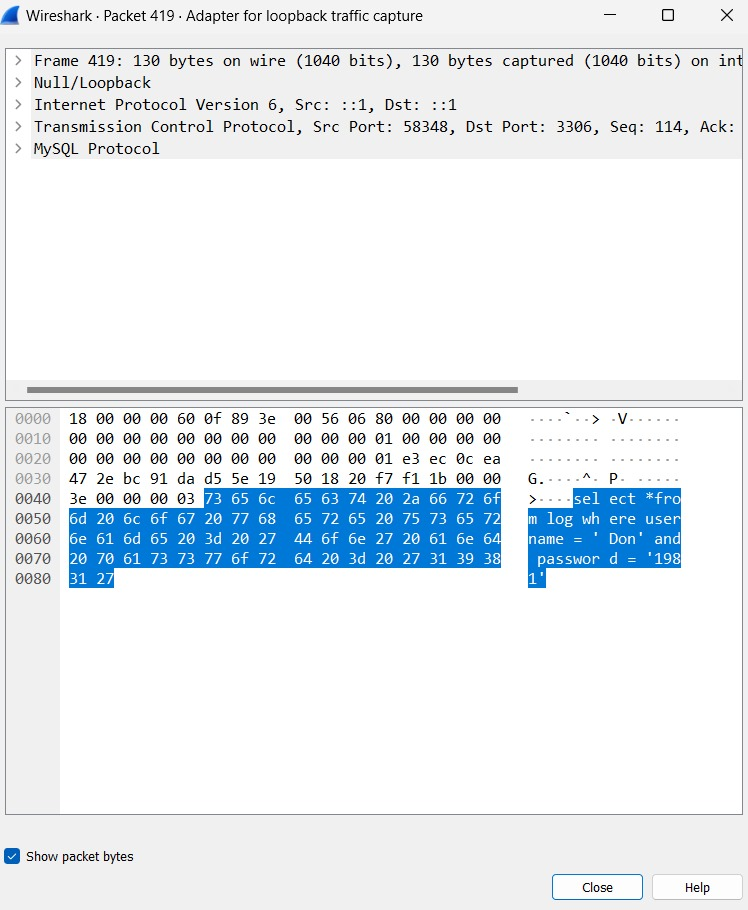
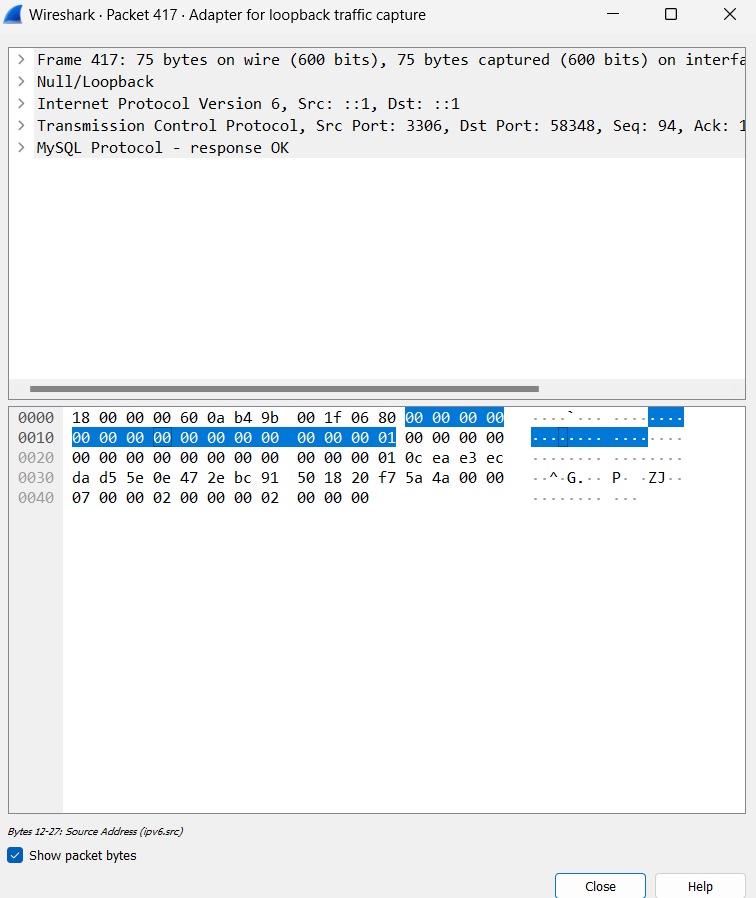
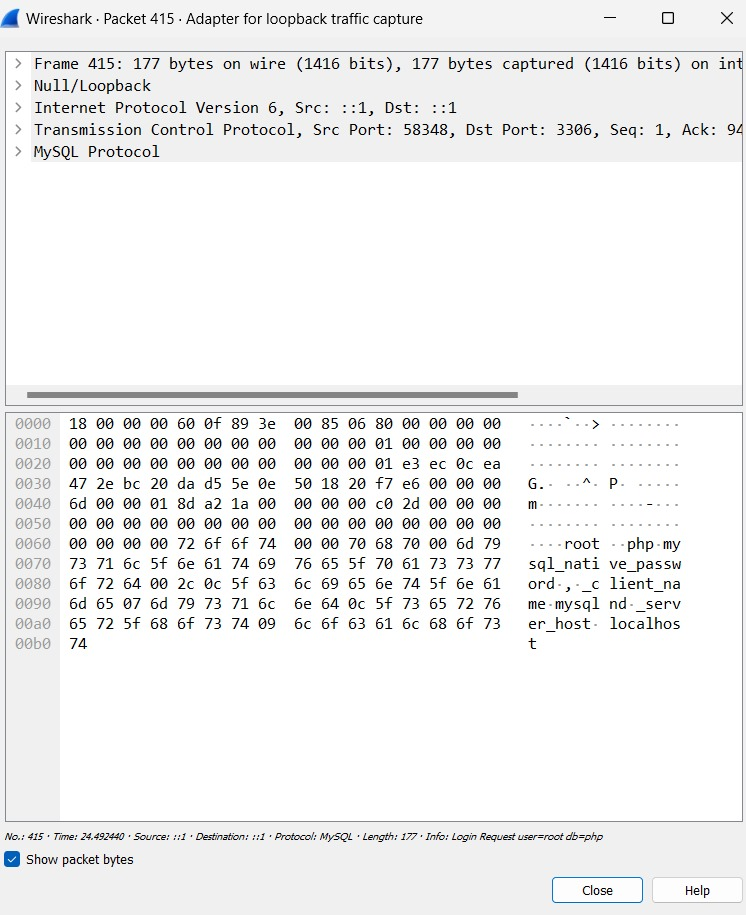
**2.)**

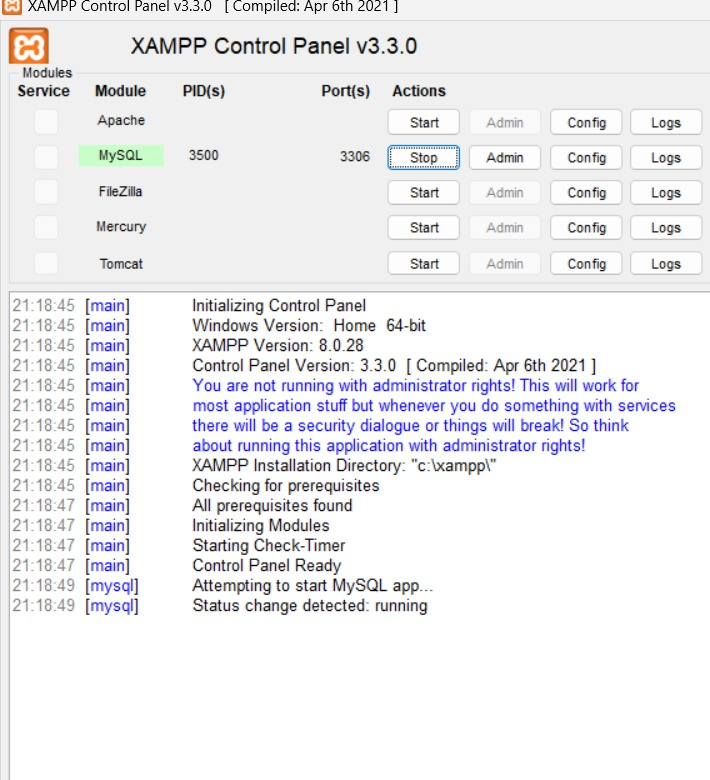
****

**3.)**

****

**4.)**

****

****