Date:

Exp No: 1

BASICS OF STUDY IN CYBER TOOLS

AIM:

To study the basics of cyber tools - Burp Suite and N-map.

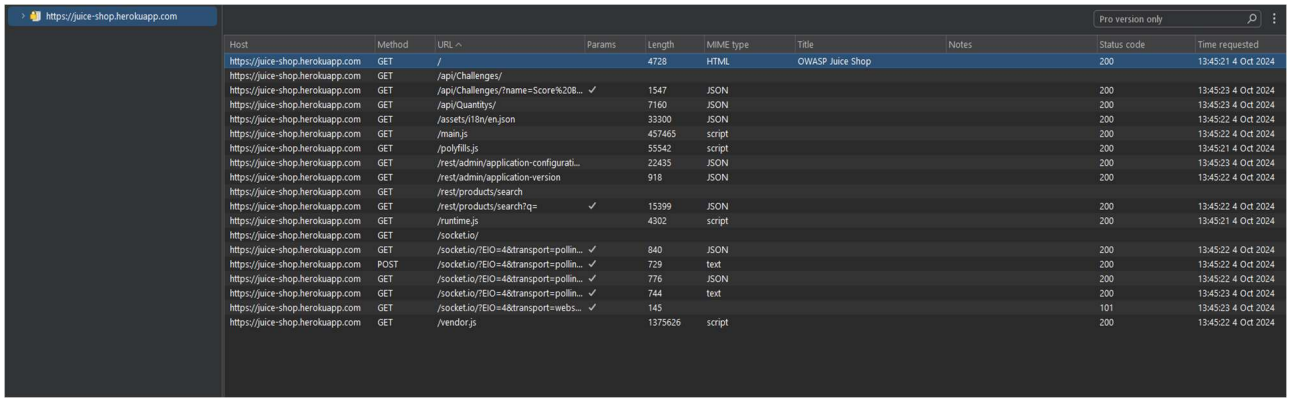
THEORY:

Burp Suite:

Burp Suite is a proprietary software tool for security assessment and penetration testing of web applications. Burp Suite gives you full control, letting you combine advanced manual techniques with state of the art automation, to make you work faster and more effectively. It features a collection of tools that can be used to assist in the process of web application pen testing. Some of the tools that Burp Suite offers are:

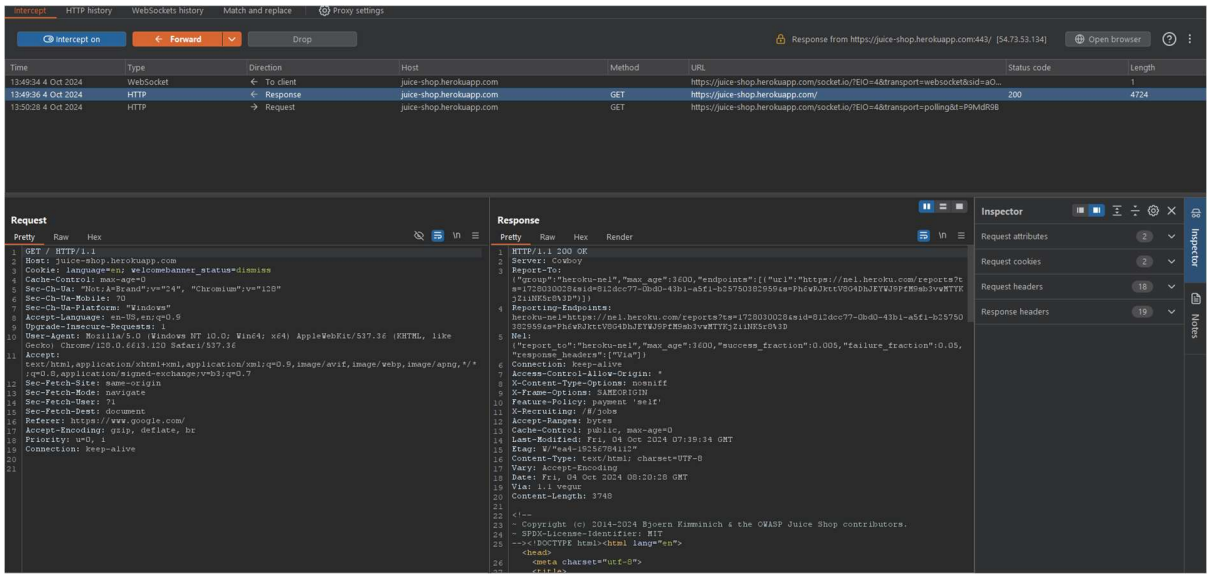
- **Spider**

It is a web spider/crawler that is used to map the target web application. The objective of the mapping is to get a list of endpoints so that their functionality can be observed and potential vulnerabilities can be found.



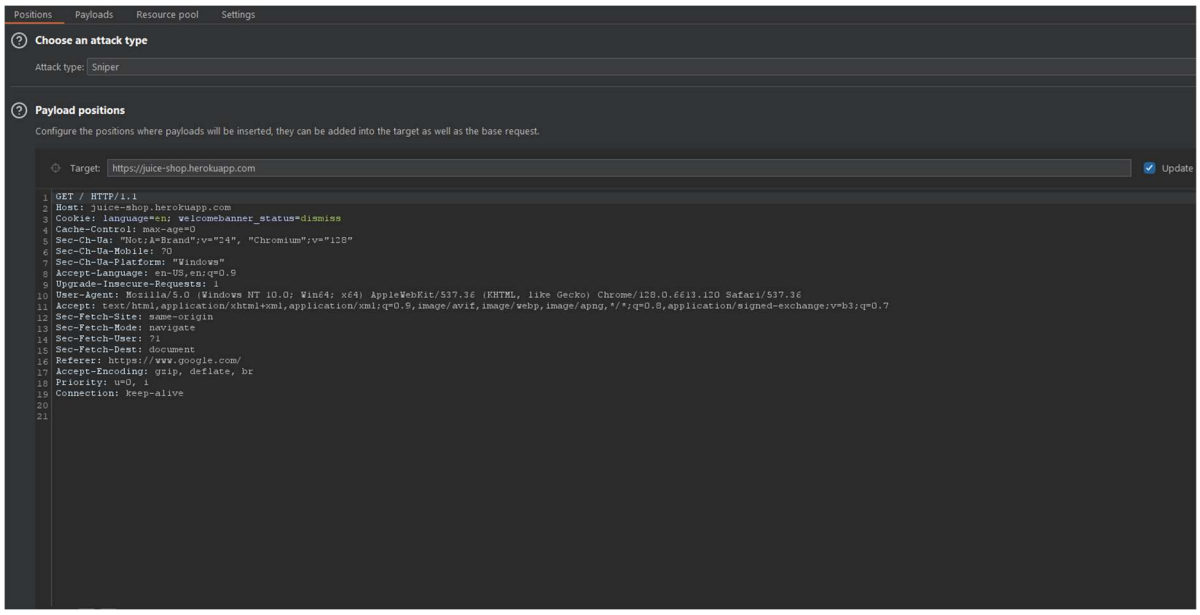
- **Proxy**

Burp Suite contains an intercepting proxy that lets the user see and modify contents of the request and response while they are in transit. The proxy server can be adjusted to run on a specific loop-back ip and a port.



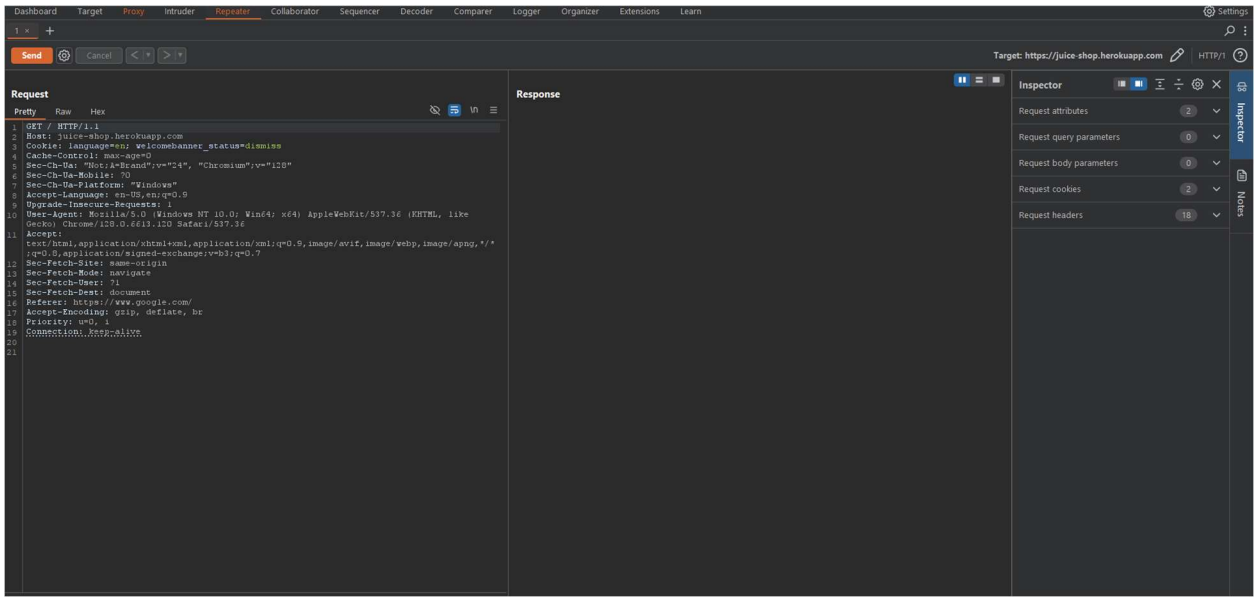
- **Intruder**

Intruder is a Burp Suite feature that can be used to brute-force attacks on password forms, pin forms, and other such forms. It can be also used to perform dictionary attacks on password forms, fields that are suspected of being vulnerable to XSS or SQL injection.



- **Repeater**

Repeater lets a user send requests repeatedly with manual modification. It can be used to verify whether the user-supplied values are being verified and what values is the server expecting in an input parameter/request header.



Nmap:

N-map is a network scanner tool that is used to discover hosts and services on a computer network by sending packets and analyzing the responses. It provides a number of features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection and other features. Some of the options that are available in N-map are:

```
sp@debian:~/Downloads/vpn$ nmap -h
Nmap 7.93 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports sequentially - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
```

- **-Pn option**

The “-Pn” option tells the Nmap to skip the host discovery phase(which includes ping probes) and proceeds directly to port scanning. This is particularly useful when dealing with hosts that may be configured not to respond to ICMP ping requests.

- **-sV option**

The “-sV” option helps in finding the version of the service that is currently running on the ports.

- **-sS option**

The “-sS” option initiates a TCP SYN scan, often referred to as a stealth scan, to determine which ports on a target system are open.

- **-T0 to -T5 option**

This option is used to perform searching levels from T0 to T5. T0 is extremely slow and designed to avoid detection. T3 is the default timing template, balances speed and reliability. T4 is faster than normal, increases the chance of detection and T5 is the fastest and most aggressive scan, likely to cause significant network disruption and may trigger IDS/IPS alerts.

RESULT:

Thus, the study of cyber tools (Burp Suite and N-map) has been completed and the output has been verified.

Date:

Exp No: 2

CREATE USER REGISTRATION AND LOGIN AUTHENTICATION USING PHP AND MYSQL

AIM:

To create a user registration and login authentication web applications using PHP and MySQL.

ALGORITHM:

Step 1: Start the XAMPP apache server and PhPmyadmin server

Step 2: Database Design: Create a users table to store user credentials.

Step 3: Configuration File: Create config.php for database connection.

Step 4: Registration Script: Create register.php to handle user registration.

Step 5: Login Script: Create login.php to handle user login.

Step 6: Welcome Page: Create welcome.php to display a welcome message after successful login.

Step 7: Logout Script: Create logout.php to handle user logout.

Source code:

1. Database Design - First, create a database and a table for storing user information.

```
CREATE DATABASE user_authentication;

USE user_authentication;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2. Configuration File- Create a configuration file to connect to the database.

config.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "user_authentication";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

3. User Registration- Create a script for handling user registration.

register.php

```
<?php
include 'config.php';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST['username']);
    $password = trim($_POST['password']);
    if (!$username) {
        echo "Please enter a username.";
    } elseif (!$password) {
        echo "Please enter a password.";
```

```
    } else {
        $stmt = $conn->prepare("SELECT id FROM users WHERE username = ?");
        $stmt->bind_param("s", $username);
        $stmt->execute();
        $stmt->store_result();
        if ($stmt->num_rows) {
            echo "This username is already taken.";
        } else {
            $stmt->close();
            $stmt = $conn->prepare("INSERT INTO users (username, password) VALUES (?, ?)");
            $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
            $stmt->bind_param("ss", $username, $hashedPassword);
            echo $stmt->execute() ? "Registration successful." : "Something went wrong. Please try
again later.";
        }
        $stmt->close();
    }
    $conn->close();
}
?>
<!DOCTYPE html>
<html>
<body>
    <h2>Register</h2>
    <form method="post">
        Username: <input type="text" name="username"><br>
        Password: <input type="password" name="password"><br>
        <input type="submit" value="Register">
```

```
    </form>
</body>
</html>
```

4. User Login

login.php

```
<?php
include 'config.php';
session_start();
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST['username']);
    $password = trim($_POST['password']);
    if (!$username) {
        echo "Please enter a username.";
    } elseif (!$password) {
        echo "Please enter a password.";
    } else {
        $stmt = $conn->prepare("SELECT id, username, password FROM users WHERE username
= ?");
        $stmt->bind_param("s", $username);
        $stmt->execute();
        $stmt->store_result();
        if ($stmt->num_rows == 1) {
            $stmt->bind_result($id, $username, $hashed_password);
            $stmt->fetch();
            if (password_verify($password, $hashed_password)) {
                $_SESSION['username'] = $username;
```

```
                $_SESSION['id'] = $id;
                header("location: welcome.php");
                exit;
            } else {
                echo "The password you entered was not valid.";
            }
        } else {
            echo "No account found with that username.";
        }
        $stmt->close();
    }
    $conn->close();
}
?>
<!DOCTYPE html>
<html>
<body>
    <h2>Login</h2>
    <form method="post">
        Username: <input type="text" name="username"><br>
        Password: <input type="password" name="password"><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

5. Welcome Page - Create a welcome page that users see after they log in.

welcome.php

```
<?php
session_start();
if(!isset($_SESSION['username'])){
   header("location: login.php");
   exit;
}
echo "Welcome " . $_SESSION['username'];
?>
<!DOCTYPE html>
<html>
<body>
   <h2>Welcome</h2>
   <p>Welcome, <?php echo htmlspecialchars($_SESSION['username']); ?>! You have successfully logged in.</p>
   <a href="logout.php">Logout</a>
</body>
</html>
```

6. Logout Script

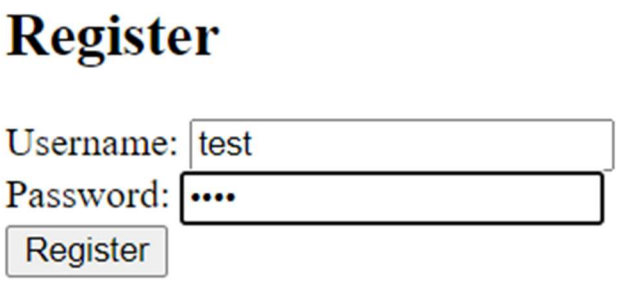
logout.php

```
<?php
session_start();
$_SESSION = array();
session_destroy();
header("location: login.php");
```

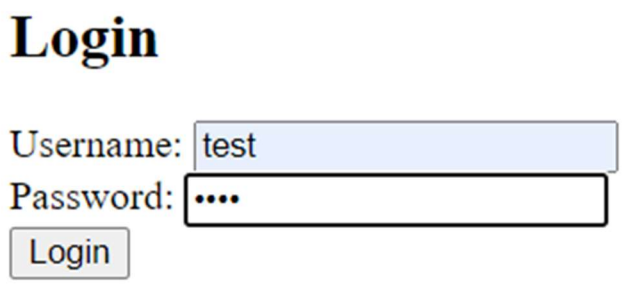
```
exit;

?>
```

OUTPUT

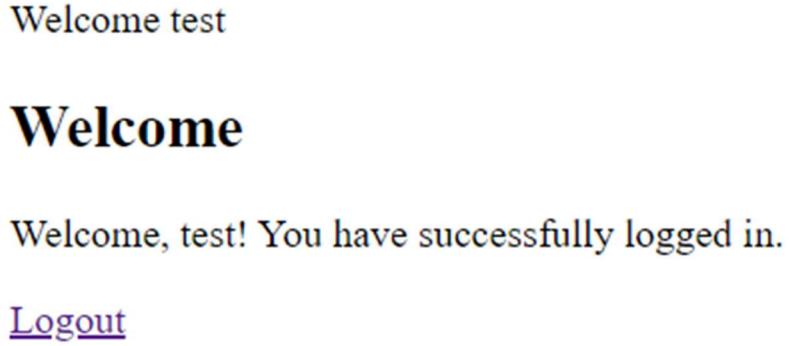
register.php



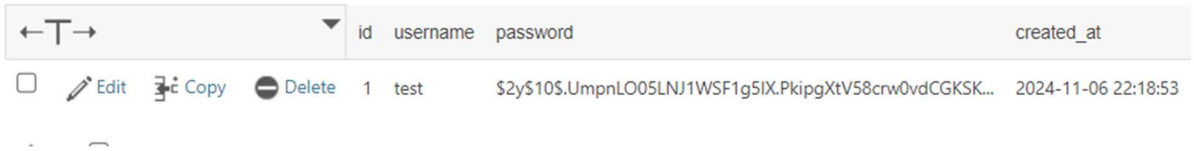
login.php



welcome.php



SQL Database:



←T→	id	username	password	created_at
Edit Copy Delete	1	test	$2y$10$.UmpnLO05LNJ1WSF1g5IX.PkipgXtV58crw0vdCGKSK...	2024-11-06 22:18:53

Result:

Thus, the user registration and login authentication web application using PhP and MySQL was created successfully.

Date:

Exp No: 3

TO BUILD AN APPLICATION TO TEST SQL INJECTION PREVENTION TECHNIQUE

AIM:

To test SQL injection prevention and set up web applications using PHP & MySQL and Burp Suite.

Algorithm:

Step 1: Setup Your Web Application

Step 2: Install and configure Burp Suite to intercept and modify HTTP requests.

Step 3: Use Burp Suite to inject SQL payloads into the web application and analyze the responses.

Step 4: Ensure your PHP code uses prepared statements and parameterized queries to prevent SQL injection.

Step 5: Confirm that SQL injection attempts fail and your application handles inputs securely.

Testing with Burp Suite

Intercepting Requests

1. Open your web application in the configured browser.
2. Go to the registration page and fill out the form with valid data, then submit it.
3. Burp Suite will intercept the request. You can now examine and modify the request.

Testing for SQL Injection

1. In Burp Suite, go to the "Proxy" tab and intercept a request by toggling the "Intercept is on" button.
2. Submit the login form with a payload that might exploit SQL injection. For example:
    - **Username:** admin' or '1'='1--
    - **Password:** password

3. Analyze the request in Burp Suite's "Intercept" tab. Modify the intercepted request to include SQL injection payloads in the parameters.
4. Forward the modified request to the server and observe the response in the "HTTP history" tab.

Example of SQL Injection Payloads

- **Username:** ' OR '1'='1
- **Password:** anything

Source Code:

Database Setup

```
create database sql_injection_test;

use sql_injection_test;

create table users(

    id int auto_increment primary key,

    username varchar(50) not null unique,

    password varchar(50) not null

);

insert into users (username, password) values ('admin','password');
```

PHP Configuration File- db.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "sql_injection_test";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

Vulnerable Login Script (vulnerable_login.php)

```
<?php
session_start();
include 'db.php';
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $query = "SELECT id FROM users WHERE username = '$username' AND password =
'$password'";
    $result = $conn->query($query);
    if ($result->num_rows > 0) {
        $_SESSION['username'] = $username;
        echo "Login successful!";
```

```
    } else {
        echo "Invalid credentials!";
    }
}
$conn->close();
?>
```

Secure Login Script (secure_login.php)

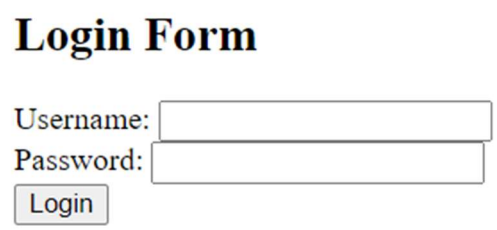
```
<?php
session_start();
include 'db.php';
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $stmt = $conn->prepare("SELECT id FROM users WHERE username = ? AND password =
?");
    $stmt->bind_param("ss", $username, $password);
    $stmt->execute();
    $stmt->store_result();

    if ($stmt->num_rows > 0) {
        $_SESSION['username'] = $username;
        echo "Login successful!";
    } else {
```

```
        echo "Invalid credentials!";
    }
    $stmt->close();
}
$conn->close();
?>
```

Login Page (login.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
</head>
<body>
    <h2>Login Form</h2>
    <form action="vulnerable_login.php" method="POST">
        Username: <input type="text" name="username"><br>
        Password: <input type="password" name="password"><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

OUTPUT



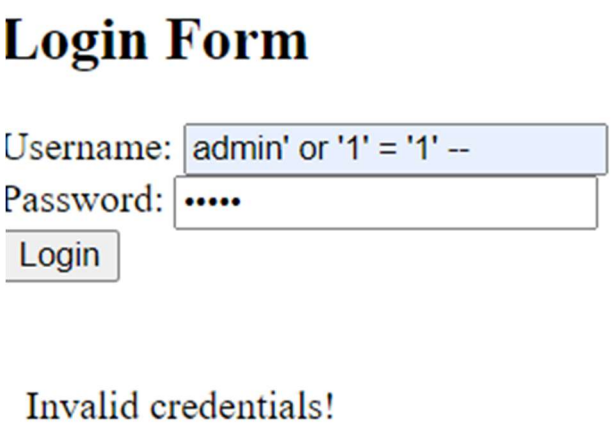
The code can be made vulnerable to SQLi by updating this line - >

<form action="vulnerable_login.php" method="POST">

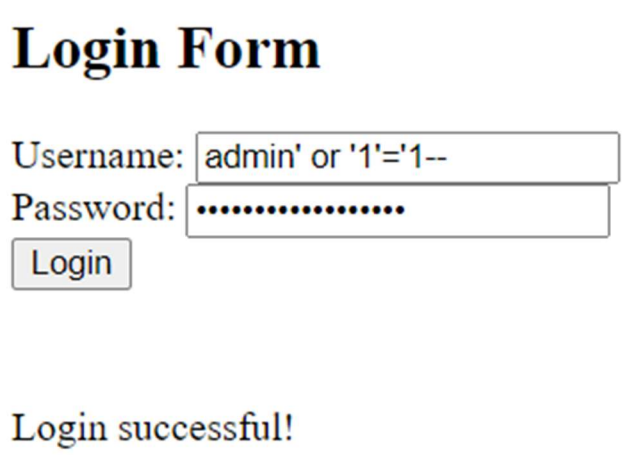
If the form action is set as vulnerable_login.php, the application is prone to SQLi and when changed to **secure_login.php,** the application will not be prone to SQLi.

The SQLi script to be given in the username is "**admin' or '1' = '1' –**"

SECURE_LOGIN OUTPUT:



VULNERABLE_LOGIN OUTPUT:



As we can see different outputs for the codes, we can confirm which login is vulnerable to SQLi.

Result:

Thus, the SQL injection prevention and set up web applications using PHP & MySQL and Burp Suite were successfully tested.

Date:

Exp No: 4

To test XSS prevention techniques like output encoding, input validation, and Content Security Policy (CSP) using BurpSuite

Aim:

To test XSS (Cross-Site Scripting) prevention techniques using PHP, MySQL, and Burp Suite (output encoding, input validation, and Content Security Policy (CSP)).

Algorithm:

Step 1: Setup Database Environment

Step 2: PHP Configuration for Database Connection

Step3: Implement XSS Prevention Techniques (output encoding, input validation, and CSP)

Step 4: Display Comments Script with Output Encoding
Step 5: Input Validation: Implement input validation to ensure that user input does not contain malicious scripts.

Step 6: Output Encoding: Use html special characters or similar functions to encode user input before displaying it.

Step 7: Content Security Policy (CSP): Use CSP headers to restrict the sources from which scripts can be loaded and executed.

Step 8: Testing: Use Burp Suite to test the application by injecting XSS payloads and verifying that they do not execute.

Step 9: Stop the applications

Database Setup

Create a database and a table for storing user posts or comments.

```
CREATE DATABASE test_xss;
USE test_xss;

CREATE TABLE comments (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    comment TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Create a PHP file for the database connection:

config.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "test_xss";

$conn = new mysqli($servername, $username, $password, $dbname);

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

Comment Submission Script with Input Validation and Output Encoding

submit_comment.php

```
<?php

include 'db_connect.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {

    $username = (trim($_POST['username']));

    $comment = (trim($_POST['comment']));

    if (empty($username) || empty($comment)) {

        echo "Username and comment are required.";

    } else {

        $stmt = $conn->prepare("INSERT INTO comments (username, comment) VALUES (?,
?)");

        $stmt->bind_param("ss", $username, $comment);
```

```
        if ($stmt->execute()) {
            echo "Comment submitted successfully.";
        } else {
            echo "Error: " . $stmt->error;
        }
        $stmt->close();
    }
}
$conn->close();
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self';">
    <title>Submit Comment</title>
</head>
<body>
    <h2>Submit Comment</h2>
    <form method="post">
        Username: <input type="text" name="username"><br>
        Comment: <textarea name="comment"></textarea><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Display Comments Script with Output Encoding

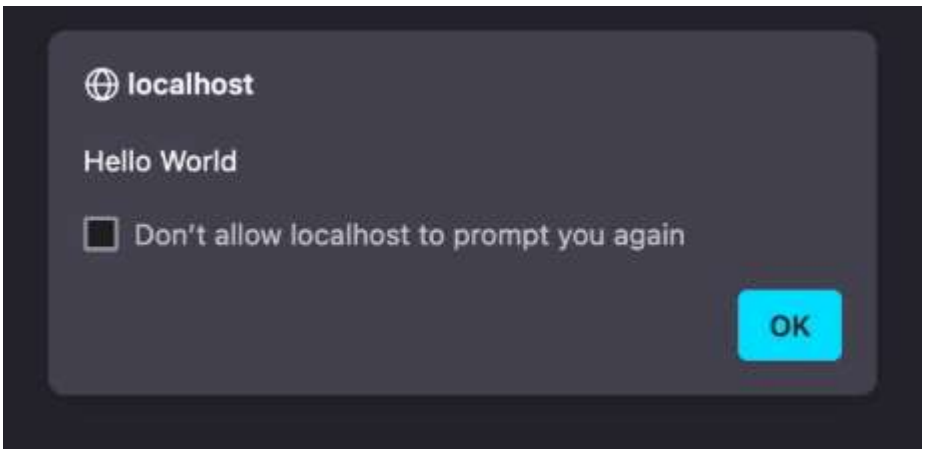
view_comments.php

```
<?php

include 'db_connect.php';

$sql = "SELECT username, comment, created_at FROM comments ORDER BY created_at
DESC";

$result = $conn->query($sql);

?>

<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self';">

   <title>View Comments</title>

</head>

<body>

   <h2>Comments</h2>

   <?php

   if ($result->num_rows > 0) {

      while($row = $result->fetch_assoc()) {

         echo "<p><strong>" . $row['username'] . ":</strong> " . $row['comment'] . " <em>at " .
$row['created_at'] . "</em></p>";

//to make the code secure, replace the above line with “echo "<p><strong>" .
htmlspecialchars($row['username']) . ":</strong> " . htmlspecialchars($row['comment']) . " <em>at " .
$row['created_at'] . "</em></p>";"

      }

   } else {

      echo "No comments yet.";
```

```
    }
    $conn->close();
    ?>
</body>
</html>
```

1. **Testing for XSS Vulnerabilities:**
    - In Burp Suite, go to the "Proxy" tab and intercept the request by toggling the "Intercept is on" button.
    - Submit the form with a potential XSS payload. For example:
        - **Username:** <script>alert('XSS');</script>
        - **Comment:** <script>alert('XSS');</script>
    - Analyze the request in Burp Suite's "Intercept" tab. Modify the intercepted request to include XSS payloads in the parameters.
    - Forward the modified request to the server and observe the response in the "HTTP history" tab.

OUTPUT:



Result:

Thus, the XSS (Cross-Site Scripting) prevention technique (output encoding, input validation, and Content Security Policy (CSP)) and set up web applications using PHP & MySQL and Burp Suit was successfully tested.

Date:

Exp No: 5

To build an application to test password hashing and multifactor Authentication

Aim:

To test password hashing and multifactor authentication (MFA) implementation using PHP, MySQL, and Burp Suite.

Algorithm:

1. Setup Database Environment
2. Configure PHP for Database Connection
3. Implement Password Hashing
    - Hash passwords upon user registration.
    - Verify passwords during login.
4. Implement Multifactor Authentication (MFA)
    - Generate and send a One-Time Password (OTP) upon login.
    - Verify OTP as the second authentication step.
5. Testing: Use Burp Suite
    - Test password hashing to ensure plain-text passwords are not stored.
    - Test OTP flow for vulnerabilities in the MFA process.

Database setup:

```
CREATE DATABASE secure_auth;

USE secure_auth;

CREATE TABLE users (

   id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    username VARCHAR(50) NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    email VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE otp_codes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    otp_code VARCHAR(6) NOT NULL,
    expires_at TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Create PHP File for Database Connection

config.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "secure_auth";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

Password Hashing for User Registration - register.php

```
<?php
include 'config.php';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST['username']);
    $email = trim($_POST['email']);
    $password = trim($_POST['password']);
    if (empty($username) || empty($email) || empty($password)) {
        echo "All fields are required.";
    } else {
        $password_hash = password_hash($password, PASSWORD_BCRYPT);
        $stmt = $conn->prepare("INSERT INTO users (username, password_hash, email)
VALUES (?, ?, ?)");
        $stmt->bind_param("sss", $username, $password_hash, $email);
        if ($stmt->execute()) {
            echo "Registration successful.";
        } else {
            echo "Error: " . $stmt->error;
        }
        $stmt->close();
    }
}
$conn->close();
```

?>

Login Script with Password Verification and MFA OTP Generation – login.php

```
<?php
include 'config.php';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST['username']);
    $password = trim($_POST['password']);
    $stmt = $conn->prepare("SELECT id, password_hash FROM users WHERE username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $stmt->store_result();
    if ($stmt->num_rows > 0) {
        $stmt->bind_result($user_id, $password_hash);
        $stmt->fetch();
        if (password_verify($password, $password_hash)) {
            $otp_code = strval(rand(100000, 999999));
            $expires_at = date("Y-m-d H:i:s", strtotime("+10 minutes"));
            $otp_stmt = $conn->prepare("INSERT INTO otp_codes (user_id, otp_code, expires_at)
VALUES (?, ?, ?)");
            $otp_stmt->bind_param("iss", $user_id, $otp_code, $expires_at);
            if ($otp_stmt->execute()) {
                echo "OTP generated. Check your email for the code.";
            } else {
                echo "Error generating OTP.";
```

```
            }
            $otp_stmt->close();
        } else {
            echo "Incorrect password.";
        }
    } else {
        echo "Username not found.";
    }
    $stmt->close();
}
$conn->close();
?>
```

OTP Verification Script - verify_otp.php

```
<?php
include 'config.php';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = trim($_POST['username']);
    $otp_code = trim($_POST['otp_code']);
    $stmt = $conn->prepare("SELECT id FROM users WHERE username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $stmt->store_result();
    if ($stmt->num_rows > 0) {
        $stmt->bind_result($user_id);
```

```
        $stmt->fetch();

        $otp_stmt = $conn->prepare("SELECT otp_code, expires_at FROM otp_codes WHERE user_id = ? ORDER BY expires_at DESC LIMIT 1");

        $otp_stmt->bind_param("i", $user_id);

        $otp_stmt->execute();

        $otp_stmt->bind_result($stored_otp_code, $expires_at);

        $otp_stmt->fetch();

        if ($otp_code === $stored_otp_code && strtotime($expires_at) > time()) {

            echo "OTP verified. Login successful.";

        } else {

            echo "Invalid or expired OTP.";

        }

        $otp_stmt->close();

    } else {

        echo "Username not found.";

    }

    $stmt->close();

}

$conn->close();

?>
```

HTML Forms

Registration Form (register.html)

```
<form action="register.php" method="POST">

    Username: <input type="text" name="username"><br>
```

```
    Email: <input type="email" name="email"><br>
    Password: <input type="password" name="password"><br>
    <input type="submit" value="Register">
</form>
```

Login Form (login.html)

```
<form action="login.php" method="POST">
    Username: <input type="text" name="username"><br>
    Password: <input type="password" name="password"><br>
    <input type="submit" value="Login">
</form>
```

OTP Verification Form (verify_otp.html)

```
<form action="verify_otp.php" method="POST">
    Username: <input type="text" name="username"><br>
    OTP Code: <input type="text" name="otp_code"><br>
    <input type="submit" value="Verify OTP">
</form>
```

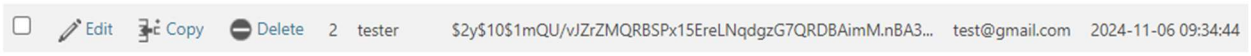
Procedure

1. **Password Hashing Test**
    - Register a user in the registration page.
    - Check the database to confirm that passwords are stored as hashed strings (not plain text).
2. **OTP Testing**
    - Login the user and check the database for the OTP.
    - Check for OTP generation and expiry.
    - Inject payloads to test for vulnerabilities in OTP verification.

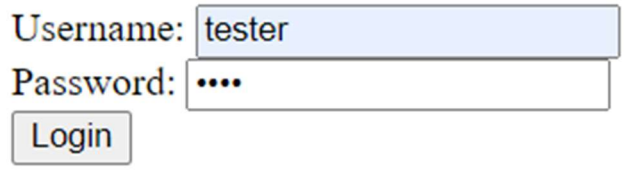
OUTPUT:

register.html:

The password is stored in a hashed format in the database..

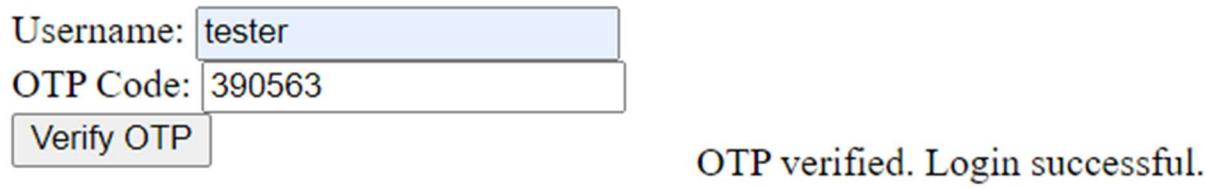


login.html



After logging in, the OTP is created and stored in the database, through which the user can verify the otp in the **verify_otp.html** page

verfiy_otp.html



RESULT:

Thus, an application to test password hashing and multifactor authentication has been executed and the output has been verified.

Date:

Exp No: 6

Implement techniques like session token rotation, session expiration, and secure cookie attributes using Burp Suite

Aim:

To implement techniques like session token rotation, session expiration, and secure cookie attributes using Burp Suite

Algorithm:

Step 1: Session token rotation helps mitigate session hijacking by frequently changing the session ID.

Step 2: Session expiration ensures that sessions are terminated after a certain period of inactivity.

Step 3: Setting secure cookie attributes like HttpOnly, Secure, and SameSite helps protect cookies from being accessed or transmitted insecurely.

Step 4: Burp Suite can be used to verify that the session management and cookie attributes are correctly implemented.

Source Code:

Session Token Rotation

```
function secure_cookie_settings() {
    $cookieParams = session_get_cookie_params();
    session_set_cookie_params([
        'lifetime' => $cookieParams['lifetime'],
        'path' => $cookieParams['path'],
        'domain' => $cookieParams['domain'],
        'secure' => isset($_SERVER['HTTPS']),
        'httponly' => true,
```

```
        'samesite' => 'Lax'
    ]);
}
secure_cookie_settings();
session_start();
function rotate_session() {
    if (!isset($_SESSION['last_rotation'])) {
        $_SESSION['last_rotation'] = time();
    }
    if (time() - $_SESSION['last_rotation'] > 5) { // 5 seconds
        session_regenerate_id(true);
        $_SESSION['last_rotation'] = time();
    }
}
function check_session_timeout() {
    $session_timeout = 60; // 60 seconds
    if (isset($_SESSION['last_activity']) && (time() - $_SESSION['last_activity'] >
$session_timeout)) {
        session_unset();
        session_destroy();
        session_start();
    }
    $_SESSION['last_activity'] = time();
}
rotate_session();
```

check_session_timeout();

Testing with Burp Suite

Burp Suite can be used to verify that the session management and cookie attributes are correctly implemented.

1. Session Token Rotation Testing:

- **Intercept Requests:** Use Burp Suite's Proxy to intercept HTTP requests.
- **Observe Session ID Changes:** Log in to the web application and perform actions that should trigger session rotation. Verify that the session ID changes appropriately (e.g., every 5 minutes).

2. **Session Expiration Testing:**

- **Intercept Requests:** Use Burp Suite to observe the behavior of the session over time.
- **Simulate Inactivity:** Leave the session idle for longer than the timeout period and check if the session is invalidated upon subsequent requests.

3. **Secure Cookie Attributes Testing:**

- **Intercept Requests:** Use Burp Suite to capture HTTP requests and inspect cookies.
- **Check Cookie Flags:** Verify that session cookies have the HttpOnly, Secure, and SameSite attributes set correctly.

OUTPUT:

When we first open the website, we will be getting the session token as follow:

The PHPSESSIDcontains the value “vqmngijt29mtv64v2cijd0tmk7”.

When we refresh the site after 5 seconds, we will be able to notice the change in the session id as follow:

The new session id will be “0ot7ul66orrh2uu295intm5k93”

Result:

Thus, a program to Implement techniques like session token rotation, session expiration, and secure cookie attributes using Burp Suite has been executed and output has been verified.

Date:

Exp No: 7

Implement techniques like file type validation, content-type checking, and secure file storage

Aim:

To implement techniques like file type validation, content type checking and secure file storage web applications using PhP and MySQL.

Algorithm:

Step-1 Ensure that the "uploads" directory is located outside of the web root to prevent direct access to uploaded files.

Step-2 Implement access control mechanisms to restrict access to uploaded files based on user permissions.

Step-3 regularly audits your code base and server configurations for security vulnerabilities.

Source code

Configuration file:

config.php

```
<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "secure_uploads";

$conn = new mysqli($servername, $username, $password, $dbname);
```

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

File Upload Form :

index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Secure File Upload</title>
</head>
<body>
    <h2>Upload File</h2>
    <form action="upload.php" method="post" enctype="multipart/form-data">
        <input type="file" name="fileToUpload" required>
        <input type="submit" value="Upload File">
    </form>
</body>
</html>
```

File Upload PHP Script

upload.php

```
<?php
```

```
include 'config.php';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $allowedTypes = ["image/jpeg", "image/png", "application/pdf"];
    if (isset($_FILES["fileToUpload"]) && $_FILES["fileToUpload"]["error"] == 0) {
        $fileName = basename($_FILES["fileToUpload"]["name"]);
        $fileType = $_FILES["fileToUpload"]["type"];
        $fileTempPath = $_FILES["fileToUpload"]["tmp_name"];
        if (!in_array($fileType, $allowedTypes)) {
            die("Error: Unsupported file type.");
        }
        $finfo = finfo_open(FILEINFO_MIME_TYPE);
        $actualType = finfo_file($finfo, $fileTempPath);
        finfo_close($finfo);
        if ($actualType !== $fileType) {
            die("Error: File content does not match declared type.");
        }
        $storagePath = __DIR__ . '/secure_uploads/';
        if (!is_dir($storagePath)) {
            mkdir($storagePath, 0700, true);
        }
        $targetFilePath = $storagePath . $fileName;
        if (move_uploaded_file($fileTempPath, $targetFilePath)) {
            $stmt = $conn->prepare("INSERT INTO files (filename, mime_type) VALUES (?, ?)");
            $stmt->bind_param("ss", $fileName, $fileType);
            if ($stmt->execute()) {
```

```
            echo "File uploaded and saved securely.";
        } else {
            echo "Failed to save file information.";
        }
        $stmt->close();
    } else {
        echo "Error: File upload failed.";
    }
} else {
    echo "No file uploaded or file too large.";
}
}
$conn->close();
?>
```

MySQL Database Schema:

```
CREATE DATABASE secure_uploads;
use secure_uploads;
CREATE TABLE files (
    id INT AUTO_INCREMENT PRIMARY KEY,
    filename VARCHAR(255) NOT NULL,
    mime_type VARCHAR(50) NOT NULL,
    upload_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Burp Suite Testing:

1. **Proxy Configuration:**

    - Configure your browser to use Burp Suite as a proxy.
    - Intercept requests when submitting files through the file upload form.

2. **Testing:**

    - Upload files with different extensions and MIME types.
    - Tamper with the Content-Type header in the request to test content-type checking.
    - Review the responses to ensure files are stored securely and filenames are not easily guessable.

OUTPUT:

File upload window

Uploading a valid file type:

Uploading invalid file type:

Upload File

Choose File Weirdness.ovpn Upload File

Error: Unsupported file type.

The uploaded files are stored in **/secure_uploads/** directory

Index of /secure/file_validation/secure_uploads

Name	Last modified	Size	Description
Parent Directory		-	
d35e2f39e4008a0ce1bf..>	2024-11-06 22:08	16K	

Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.2.12 Server at localhost Port 80

Result:

Thus, file type validation, content type checking and secure file storage using PhP and MySQL were successfully tested.

Date:

Exp No: 8

Prevent Cross-Site Request Forgery (CSRF) attacks in a web application

Aim:

To demonstrate and prevent Cross-Site Request Forgery (CSRF) attacks in a web application.

Algorithm:

Step 1 : Create a web app with a form that allows users to perform actions (like submitting data).

Step 2 : Try a CSRF attack by sending fake requests using the user's session without their knowledge.

Step 3 : Add CSRF protection by creating a unique token for each user session.

Step 4 : Make sure the token is included in every form or request sent by the user.

Step 5 : On the server, check if the token is correct; if it's missing or wrong, reject the request.

Step 6 : Test to ensure that the CSRF protection works by trying the attack again.

Source Code :

1.Database Design :

```
CREATE DATABASE csrf_database;

USE csrf_database;

CREATE TABLE  users (

        id INT AUTO_INCREMENT PRIMARY KEY,

        email VARCHAR(255) NOT NULL,

        username VARCHAR(50) NOT NULL,
```

```
    password VARCHAR(255) NOT NULL
);

INSERT INTO users (email, username, password)
VALUES ('test@example.com', 'testuser', 'password123');
```

2. Configuration File :

config.php

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "csrf_database";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

3. Vulnerable code CSRF

vulnerable.php

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>CSRF Vulnerable Form</title>
</head>
<body>
    <h2>Change Email (Vulnerable)</h2>
    <form id="vulnerableForm" action="vulnerable/change_email.php" method="POST">
        <input type="email" id="emailInput" name="new_email" placeholder="Enter new email" required>
        <input type="submit" value="Change Email">
    </form>
    <script>
        document.getElementById("vulnerableForm").addEventListener("submit", function(event) {
            // Modify the email value before the form is submitted
            document.getElementById("emailInput").value = "hacked@example.com";
        });
    </script>
</body>
</html>
```

vulnerable/change_email.php

```
<?php
session_start();
$_SESSION['user_id'] = 1;
```

```
include '../config.php';

if (!isset($_SESSION['user_id'])) {
    die("Please log in first.");
}
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    $new_email = $_POST['new_email'];
    $stmt = $conn->prepare("UPDATE users SET email = ? WHERE id = ?");
    $stmt->bind_param("si", $new_email, $_SESSION['user_id']);
    $stmt->execute();
    echo "Email address updated successfully (vulnerable).";
}
?>
```

4. Protected code CSRF

protected.php

```
<?php
session_start();
// Generate a CSRF token and store it in the session if it doesn't exist
if (!isset($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CSRF Protected Form</title>
</head>
<body>
    <h2>Change Email (Protected)</h2>
    <form id="protectedForm" action="protected/change_email_protected.php" method="POST">
        <!-- CSRF token hidden input -->
        <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token']; ?>">
        <input type="email" id="emailInput" name="new_email" placeholder="Enter new email" required>
        <input type="submit" value="Change Email">
    </form>
    <script>
        document.getElementById("protectedForm").addEventListener("submit", function(event) {
            // Here we will NOT modify the email to simulate an attack, so it will send what the user typed
            console.log("Form submitted with CSRF token for protection.");
        });
    </script>
</body>
</html>
```

protected/change_email_protected.php

```
<?php
session_start();
include '../config.php';

if (!isset($_SESSION['user_id'])) {
    die("Please log in first.");
}

if ($_SERVER["REQUEST_METHOD"] === "POST") {
    // Verify the CSRF token
    if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token']) {
        die("CSRF validation failed. Request denied.");
    }

    // Proceed to update email if CSRF validation passes
    $new_email = $_POST['new_email'];
    $stmt = $conn->prepare("UPDATE users SET email = ? WHERE id = ?");
    $stmt->bind_param("si", $new_email, $_SESSION['user_id']);
    $stmt->execute();
    $stmt->close();
    echo "Email address updated successfully (protected).";
}
```

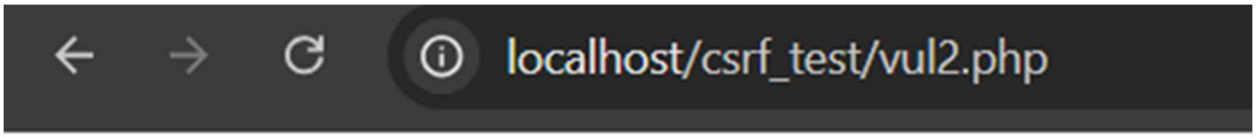
?>

OUTPUT :

Initial Database:

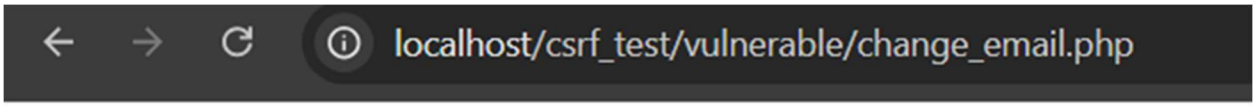
←T→	id	email	username	password
Edit Copy Delete	1	test@example.com	testuser	password123

Vulnerable Code:



Change Email (Vulnerable)

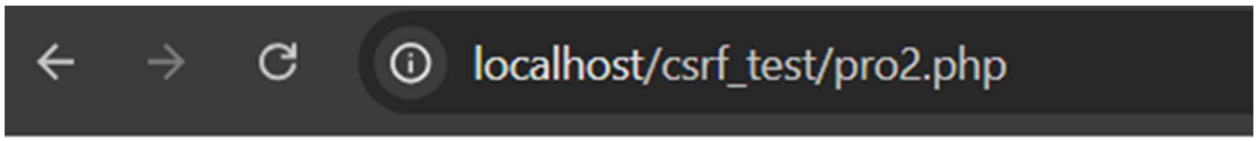
bot@gmail.com Change Email



Email address updated successfully (vulnerable).

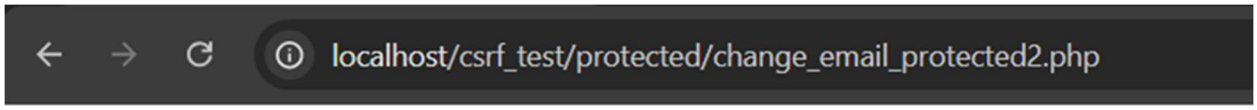
←T→	id	email	username	password
Edit Copy Delete	1	hacked@example.com	testuser	password123

Protected Code:

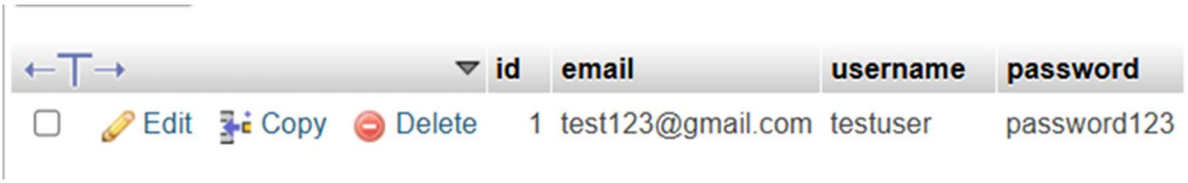


Change Email (Protected)

test123@gmail.com Change Email



Email address updated successfully (protected).



	id	email	username	password
Edit Copy Delete	1	test123@gmail.com	testuser	password123

Result:

Thus, the program to demonstrate and prevent Cross-Site Request Forgery (CSRF) is executed successfully.

Date:

Exp No: 9

Write a python program to encrypt and decrypt data using pycryptodome

Aim:

To implement data encryption and decryption using AES (Advanced Encryption Standard) with the pycryptodome library in Python.

Algorithm:

1. Start.
2. Input the sensitive data to be encrypted and a secure password for key generation.
3. Generate Key:
    - Derive a key from the password and a randomly generated salt using PBKDF2 (Password-Based Key Derivation Function 2).
4. Encrypt Data:
    - Generate a random Initialization Vector (IV).
    - Pad the plaintext to match AES's block size.
    - Encrypt the padded data with AES in CBC mode.
    - Concatenate the salt, IV, and ciphertext, then encode in Base64 for storage.
5. Decrypt Data:
    - Decode the stored Base64 data.
    - Extract the salt, IV, and ciphertext.
    - Regenerate the key using the password and salt.
    - Decrypt the ciphertext and remove padding to recover the original message.
6. Output the decrypted data to verify successful encryption and decryption.
7. End.

Procedure:

1. **Install the pycryptodome library**:
    - Open a terminal or command prompt.
    - Run the command **pip install pycryptodome** to install the necessary cryptographic library.
2. **Write the Python Code**:

Create a Python file (e.g., encryption_example.py).

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from Crypto.Protocol.KDF import PBKDF2
import base64
def generate_key(password: str, salt: bytes) -> bytes:
    return PBKDF2(password, salt, dkLen=32)
def encrypt(data: str, password: str) -> str:
    salt = get_random_bytes(16)
    key = generate_key(password, salt)
    iv = get_random_bytes(16)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    pad_len = AES.block_size - len(data) % AES.block_size
    padded_data = data + chr(pad_len) * pad_len
    encrypted_data = cipher.encrypt(padded_data.encode('utf-8'))
    return base64.b64encode(salt + iv +
encrypted_data).decode('utf-8')
def decrypt(encrypted_data: str, password: str) -> str:
    decoded_data = base64.b64decode(encrypted_data)
    salt = decoded_data[:16]
    iv = decoded_data[16:32]
    ciphertext = decoded_data[32:]
    key = generate_key(password, salt)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_padded_data = cipher.decrypt(ciphertext)
```

```
    pad_len = decrypted_padded_data[-1]
    return decrypted_padded_data[:-pad_len].decode('utf-8')
password = "securepassword123"
data = "Sensitive information to encrypt"
encrypted_text = encrypt(data, password)
print("Encrypted:", encrypted_text)
decrypted_text = decrypt(encrypted_text, password)
print("Decrypted:", decrypted_text)
```

3. **Run the Code**:
    - In a terminal or command prompt, navigate to the directory where the Python file is saved.
    - Run the file using python encryption_example.py.
4. **Observe Output**:
    - The script will display the encrypted text and the decrypted text, showing that the original data can be accurately recovered after encryption and decryption.

OUTPUT:

```
Encrypted: Hdo0G8Pp/yAxJ1DDh/x5aWy0B7xQiHrFz3cF0SVhdKwxwPg7cFqOaJtyUbXo8B84HMkt9
UG93qhMWdWSVlXgNvozx8oGBDVqwrE4t1peO1c=
Decrypted: Sensitive information to encrypt
```

RESULT:

Thus, a program to implement AES encryption and decryption using pycryptodome has been executed and the output has been verified.

Date:

Exp No: 10

Role based access content control in web application

Aim:

To implement Role-Based Access Control (RBAC) in a web application, where different roles (e.g., Admin, Editor, Viewer) are given different access rights to control the visibility and accessibility of content based on the user's role.

Algorithm:

1. Start.
2. Define roles and permissions for various types of users (e.g., Admin, Editor, Viewer).
3. Set up a database with tables for users, roles, and permissions.
4. Authenticate users at login and retrieve their role.
5. Check permissions based on the user's role before granting access to content.
6. Display content based on access rights, restricting actions (like edit or delete) for unauthorized roles.
7. End.

Procedure:

1. **Database Setup:**
    - Define tables in MySQL for users, roles, and permissions.
    - Set up relationships between users and roles, and roles and permissions.
2. **User Interface:**
    - Create HTML forms for user login and protected pages.
    - Display different content/actions based on the user's role.
3. **Role Assignment:**
    - Assign roles to users and set specific permissions for each role.
    - Ensure roles have necessary permissions in the database.
4. **Authentication and Authorization:**
    - Authenticate users based on username and password.
    - Retrieve the user's role after login.

- Check permissions for each page or action request based on the user's role.

5. **Implementation:**
   - Code the login and access control mechanisms using PHP.
   - Use session management to track logged-in users and their roles.
   - Apply conditional rendering in HTML/PHP to display content based on roles.

SOURCE CODE:

Database Schema (MySQL)

```
CREATE DATABASE rbac_demo;

USE rbac_demo;

CREATE TABLE roles (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50) NOT NULL
);

CREATE TABLE users (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50) NOT NULL,
   password VARCHAR(255) NOT NULL,
   role_id INT,
   FOREIGN KEY (role_id) REFERENCES roles(id)
);


CREATE TABLE permissions (
   id INT AUTO_INCREMENT PRIMARY KEY,
   role_id INT,
```

```
    page VARCHAR(50) NOT NULL,
    has_access BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (role_id) REFERENCES roles(id)
);
INSERT INTO roles (name) VALUES ('admin'), ('editor');
INSERT INTO users (username, password, role_id) VALUES ('admin', '1234', 1);
INSERT INTO users (username, password, role_id) VALUES ('editor', '1234', 1);
INSERT INTO permissions (role_id, page, has_access) VALUES
(1, 'admin_page.php', true),
(2, 'admin_page.php', false);
```

PHP Script for User Authentication (login.php)

```
<?php
include 'auth.php';
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_POST['username'];
    $password = $_POST['password'];
    if (login($username, $password)) {
        header("Location: dashboard.php");
        exit();
    } else {
        echo "Invalid credentials";
    }
}
```

```
?>
<form method="post">
    <label>Username:</label>
    <input type="text" name="username" required>
    <br>
    <label>Password:</label>
    <input type="password" name="password" required>
    <br>
    <button type="submit">Login</button>
</form>
```

Dashboard with Role-Based Access Control (dashboard.php)

```
<?php
include 'auth.php';
if (!is_logged_in()) {
    header("Location: login.php");
    exit();
}
?>
<h1>Dashboard</h1>
<p>Welcome, <?php echo $_SESSION['user_id']; ?>!</p>
<ul>
    <li><a href="admin_page.php">Admin Page</a></li>
```

```
</ul>
<form method="post" action="logout.php">
    <button type="submit">Logout</button>
</form>
```

Database Connection File (db.php)

```
<?php
$host = 'localhost';
$db = 'rbac_demo';
$user = 'root';
$pass = “”;
try {
    $pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Could not connect to the database: " . $e->getMessage());
}
?>
```

USER AUTHENTICATION PHP SCRIPT (auth.php)

```
<?php
session_start();
include 'db.php';
```

```
function login($username, $password) {
    global $pdo;
    $stmt = $pdo->prepare("SELECT * FROM users WHERE username = ?");
    $stmt->execute([$username]);
    $user = $stmt->fetch();
    if ($user && $user['password'] === $password) {
        $_SESSION['user_id'] = $user['id'];
        $_SESSION['role_id'] = $user['role_id'];
        return true;
    }
    return false;
}
function is_logged_in() {
    return isset($_SESSION['user_id']);
}
function logout() {
    session_destroy();
}
?>
```

ADMIN PAGE (admin.php)

```
<?php
session_start();
include 'access_control.php';
```

```
if (!has_access('admin_page.php')) {
    die("Access denied: You do not have permission to access this page.");
}
echo "Welcome to the admin page!";
?>
```

ACCESS CONTROL SCRIPT (access_control.php)

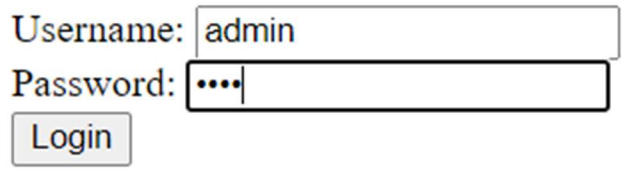
```
<?php
include 'db.php';
function has_access($page) {
    if (!isset($_SESSION['role_id'])) {
        return false;
    }
    $role_id = $_SESSION['role_id'];
    global $pdo;
    $stmt = $pdo->prepare("SELECT has_access FROM permissions WHERE role_id = ? AND page = ?");
    $stmt->execute([$role_id, $page]);
    $permission = $stmt->fetchColumn();
    return $permission == 1;
}
?>
```

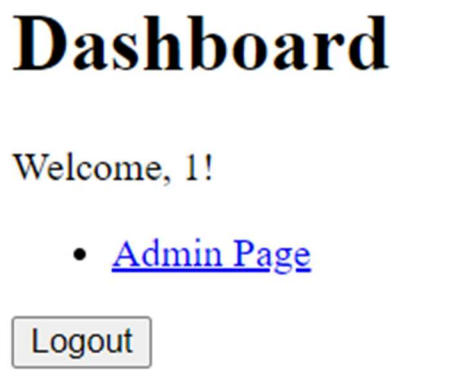
LOGOUT SCRIPT (logout.php)

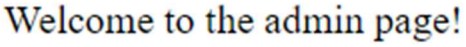
```
<?php
include 'auth.php';
logout();
header("Location: login.php");
exit();
?>
```

OUTPUT:

For admin user:







For non-admin users:

RESULT:

Thus, an application to implement role-based access content control has been implemented and the output has been verified.