WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Amit Gaddi

Email: gaddiat@mail.uc.edu

Short-bio: Amit has keen interests in IT.

Repository Information

Repository's URL: https://github.com/gaddiat-uc/waph.git

This is a private repository for Amit Gaddi to store all code from the course. The organization of this repository is as follows.

Individual Project 2: Full-stack Web Application Development

Individual Project 2 Link

Project Overview

This individual project entails creating a secure user management system with a variety of features. Users may register by entering their username, password, name, and email address, which is validated both on the client and server sides to verify data correctness. The login system employs secure authentication and session management. Once logged in, users may safely access, edit, and update their profiles. The program prioritizes security by delivering via HTTPS, hashing passwords before storage, and employing prepared statements to avoid SQL injection. To avoid XSS attacks, comprehensive input validation has been applied. The project entails database design, front-end programming in HTML, CSS (with optional frameworks), and JavaScript for a responsive interface. Secure session management and CSRF protection are also used to prevent session hijacking and CSRF attacks during database changes.

Functional Requirements

- **User Registration:** Firstly, I created a database before creating any of the forms. For the user registration form, I have created two PHP files. One of them is to take input from the user, validate the input, and store it in the database. Once the user fills out the details, the data is sent to the next page where it is again validated and stored in the database.
- **Login:** To implement a login page, I created a login form that asks for the username and password to authenticate the user. When the details match those in the database, the user is allowed to proceed to the next page where their name and email are displayed, along with options to change the profile details, update

the password, and logout. When the user visits the index.php page, session management starts and the user's details are stored in the session data, which is then accessible across other pages.

- **Profile Management:** Here, the user is allowed to visit the "updateprofile" page only when logged in. On this page, the user can change basic details such as name and email.
- **Password Update:** Here too, the user is allowed to access the page only if authenticated. On this page, the user enters their username, current password, and new password.

Security and Non-technical Requirements

- **Security:**The web application is HTTPS secured with a proper certificate provided for local IP and domain addresses. The passwords sent to the database are hashed, as seen in the `database.php` file. This file contains only functions related to the database, and non-root user details have been used to access the database. Every SQL query has a prepared statement to enhance page security as much as possible.
- **Input Validation:** Input validation has been implemented on both the server and client sides, as well as in the database. This includes the registration page, where data is sanitized using built-in functions like `trim()`, `htmlentities()`, and special functions to prevent XSS attacks.
- **Database Design:** First, I created a database account where I created a database and then I created a user for it, as seen in the file in the appendix. Furthermore, I created a table named `users` which stores the details of the users. To secure the database, I used prepared statements for every query execution, and I maintained a separate `database.php` file that contains only database methods.
- **Front-end Development:** For the front-end, I used HTML and CSS for visual purposes, along with JavaScript for client-side validations. Additionally, I utilized PHP for server-side functionality. I created my own `styles.css` file and linked it in every page to maintain a consistent theme across all my pages.
- **Session Management:** For session management, I used `session_start()` on all pages except the login and registration ones. Additionally, I implemented session authentication to ensure that users must log in before accessing other pages. I also added a session hijacking prevention mechanism where the application checks the session browser and HTTP user-agent to verify the user's identity and grants access only if they match, thereby preventing session hijacking by attackers.
- **CSRF Protection:** I have included CSRF protection on two pages where users can update their information: `changepasswordform.php` and `updateprofile.php`. First, I assign a random value to a new session variable. Then, I add a hidden element in my form that sends the CSRF token generated from this session. When the form is submitted, the page checks whether the CSRF token from the session matches the one passed in the form. If they match, the form is submitted; otherwise, it is rejected.

Demonstration Video:

Individual Project 2 source code Link

Appendix

Individual Project 2 source zip code Link

Included `registration.php`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Registration page</title>
  <link rel="stylesheet" href="style.css">
  <script type="text/javascript">
    function displayTime() {
      document.getElementById('digit-clock').innerHTML = "Current time: " + new Date().toLocaleString();
    }
    setInterval(displayTime, 500);

    function validateForm() {
      var fullname = document.getElementById('fullname').value.trim();
      var email = document.getElementById('email').value.trim();
      var username = document.getElementById('username').value.trim();
      var password = document.getElementById('password').value;
      var confirmPassword = document.getElementById('confirmPassword').value;
      var errorElement = document.getElementById('error-message');

      // Client-side validation for password match
      if (password !== confirmPassword) {
        errorElement.textContent = "Passwords do not match";
        return false;
      }

      // Client-side validation for preventing XSS (Cross-Site Scripting)
      var htmlPattern = /<\/?[a-z][\s\S]*>/i; // Regex to detect HTML tags
      if (htmlPattern.test(fullname) || htmlPattern.test(email) || htmlPattern.test(username)) {
        errorElement.textContent = "Invalid characters detected";
        return false;
      }

      // Client-side validation for email format
      var emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!emailPattern.test(email)) {
        errorElement.textContent = "Invalid email format";
        return false;
      }-

      // Clear error message if all validations pass
      errorElement.textContent = "";
      return true;
    }
  </script>
</head>
<body>
  <div class="container">
    <h1>Registration Form</h1>
    <div id="digit-clock"></div>
    <form action="addnewuser.php" method="POST" class="form login" onsubmit="return validateForm()">
      <input type="text" class="text_field" id="fullname" name="fullname" placeholder="Full Name" required
      <input type="text" class="text_field" id="email" name="email" placeholder="Email" required><br>
      <input type="text" class="text_field" id="username" name="username" placeholder="Username" required>
      <input type="password" class="text_field" id="password" name="password" placeholder="Password" requi
      <input type="password" class="text_field" id="confirmPassword" placeholder="Confirm Password" requir
      <span class="error-message" id="error-message"></span>
      <button class="button" type="submit">Register</button>
    </form>
  </div>
  <div class="container">
        <br>
        <a href="login.php" class="login-link">Already a user? Login here</a>
  </div>
</body>
</html>
```

Included addnewuser.php

```php
<?php
require "database.php";

// Initialize variables to null
$username = $password = $fullname = $primaryemail = "";

// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Basic validation
    $username = test_input($_POST["username"]);
    $password = test_input($_POST["password"]);
    $fullname = test_input($_POST["fullname"] ?? ''); // Using null coalescing operator
    $primaryemail = test_input($_POST["email"] ?? '');

    // Validate username and password
    if (empty($username) || empty($password)) {
        echo "No username/password provided";
    } elseif (strlen($password) < 8) {
        echo "Password must be at least 8 characters long";
    } elseif (!filter_var($primaryemail, FILTER_VALIDATE_EMAIL)) {
        echo "Invalid email format";
    } else {
        // Attempt to register the user
        $success = addNewUser($username, $password, $fullname, $primaryemail);
        echo $success ? "Registration Succeed" : "Registration Failed";
    }
} else {
    // Form not submitted
    echo "Please submit the form";
}

// Function to sanitize input data
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>User Registration</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <br>
        <a href="login.php" class="login-link">Login here</a>
    </div>
</body>
```

```
</html>
```

Included `login.php`

```
<?php
session_start();
session_destroy();
?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Login page</title>
  <link rel="stylesheet" href="style.css">
  <script type="text/javascript">
      function displayTime() {
        document.getElementById('digit-clock').innerHTML = "Current time:" + new Date();
      }
      setInterval(displayTime, 500);

      function validateForm() {

      var username = document.getElementById('username').value.trim();
      var password = document.getElementById('password').value;
      var errorElement = document.getElementById('error-message');

      // Check if username is empty
        if (username === "") {
            errorElement.textContent = "Please enter your username.";
            return false; // Prevent form from submitting
        }

        // Check if password is empty
        if (password === "") {
            errorElement.textContent += (errorElement.textContent.length > 0 ? "\n" : "") + "Please enter 
            return false; // Prevent form from submitting
        }

          // Clear error message if all validations pass
          errorElement.textContent = "";
          return true;
        }
  </script>
</head>
<body>
  <div class="container">
    <h1>login Page</h1>
    <div id="digit-clock"></div>
    <?php
      // PHP code to display visited time
      echo "Visited time: " . date("Y-m-d h:i:sa");
    ?>
    <form action="index.php" method="POST" class="form login" onsubmit="return validateForm()"><br>
      Username: <input type="text" class="text_field" name="username" /> <br>
      Password: <input type="password" class="text_field" name="password" /> <br>
      <span class="error-message" id="error-message"></span>
      <button class="button" type="submit">Login</button>
    </form>
  </div>
</body>
</html>
```

Included `index.php`

```
<?php

        session_set_cookie_params([
        'lifetime' => 15*60,
        'path' => '/',
        'domain' => 'localhost',
        'secure' => TRUE,
        'httponly' => TRUE
    ]);

    session_start();

    require "database.php";

    if (isset($_POST["username"]) and isset($_POST["password"])){
        $username = htmlspecialchars($_POST["username"]); // Sanitize input
        $password = htmlspecialchars($_POST["password"]); // Sanitize input

        if (checklogin($username,$password)) {
            $_SESSION['authenticated'] = TRUE;
            $_SESSION['username'] = $_POST["username"];
            $_SESSION['browser'] = $_SERVER["HTTP_USER_AGENT"];
        }else{
            session_destroy();
            echo "<script>alert('Invalid password/username');window.location='login.php';</script>";
            die();
        }
    }
    if (!isset($_SESSION['authenticated']) or $_SESSION['authenticated'] != TRUE) {
        session_destroy();
        echo "<script>alert('Nice try!! You have to login first!')</script>";
        header("Refresh: 0; url=login.php");
        die();
    }

    if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
    session_destroy();
    echo "<script>alert('Alert! Alert ! Session hijacking is detected')</script>";
    header("Refresh: 0; url=login.php");
    die();
}

$userDetails = getUserDetails($username);
// Check if user details were found
if ($userDetails) {
    $name = htmlentities($userDetails['fullname']);
    $email = htmlentities($userDetails['primaryemail']);
} else {
    $name = "Unknown";
    $email = "Unknown";
}

?>

 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h2>Welcome <?php echo htmlentities($_SESSION['username']); ?>!</h2>
        <p>Your Email: <?php echo $email; ?></p>
        <h2>Your Name <?php echo $name; ?>!</h2>
        <a class="btn" href="changepasswordform.php">Change Password</a>
        <a class="btn" href="updateprofile.php">Edit Profile</a>
        <a class="btn" href="logout.php">Logout</a>
    </div>
</body>
</html>
```

Included `changepasswordform.php`

```
<?php
session_start();

if (!isset($_SESSION['authenticated']) or $_SESSION['authenticated'] != TRUE) {
        session_destroy();
        echo "<script>alert('Nice try!! You have to login first!')</script>";
        header("Refresh: 0; url=login.php");
        die();
    }

    if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
    session_destroy();
    echo "<script>alert('Alert! Alert ! Session hijacking is detected')</script>";
    header("Refresh: 0; url=login.php");
    die();
}


// Generate and store CSRF token in the session if it doesn't exist
if (!isset($_SESSION['nocsrftoken'])) {
    $_SESSION['nocsrftoken'] = bin2hex(openssl_random_pseudo_bytes(32)); // Generate a random token
}

// Validate CSRF token on form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Validate CSRF token from form submission
    $tokenFromForm = $_POST['nocsrftoken'] ?? '';

    if (!hash_equals($_SESSION['nocsrftoken'], $tokenFromForm)) {
        // Invalid CSRF token: Handle the error (e.g., display an error message)
        die("CSRF Token Validation Failed.");
    }

    // Proceed with form processing (e.g., change password logic)
    require "database.php";

    if (isset($_POST['username'], $_POST['current_password'], $_POST['new_password'])) {
        $username = $_POST['username'];
        $current_password = $_POST['current_password'];
        $new_password = $_POST['new_password'];

        $stmt = $mysqli->prepare("SELECT password FROM users WHERE username = ?");
        $stmt->bind_param("s", $username);
        $stmt->execute();
        $result = $stmt->get_result();

        if ($result->num_rows == 1) {
            $row = $result->fetch_assoc();
            $current_password_from_database = $row['password'];

            if (md5($current_password) == $current_password_from_database) {
                // Update password securely using prepared statement
                $new_password_hash = md5($new_password); // This should be improved for better security (u
                $update_stmt = $mysqli->prepare("UPDATE users SET password = ? WHERE username = ?");
                $update_stmt->bind_param("ss", $new_password_hash, $username);

                if ($update_stmt->execute()) {
                    echo "Password updated successfully.";
                } else {
                    echo "Failed to update password.";
                }
            } else {
                echo "Current password is incorrect.";
            }
        } else {
            echo "User not found.";
        }

        $stmt->close();
    } else {
        echo "All fields are required.";
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Change Password</title>
    <link rel="stylesheet" href="style.css">
</head>
```

```
<body>
    <h2>Change Password</h2>
    <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
        <label>Username:</label><br>
        <input type="text" name="username" required><br>
        <label>Current Password:</label><br>
        <input type="password" name="current_password" required><br>
        <label>New Password:</label><br>
        <input type="password" name="new_password" required><br><br>
        <input type="hidden" name="nocsrftoken" value="<?php echo htmlspecialchars($_SESSION['nocsrftoken']
        <input type="submit" value="Change Password">
    </form>
</body>
</html>
```

Included `updateprofile.php`

```
<?php


session_start();


if (!isset($_SESSION['authenticated']) or $_SESSION['authenticated'] != TRUE) {
        session_destroy();
        echo "<script>alert('Nice try!! You have to login first!')</script>";
        header("Refresh: 0; url=login.php");
        die();
    }

    if ($_SESSION['browser'] != $_SERVER["HTTP_USER_AGENT"]) {
    session_destroy();
    echo "<script>alert('Alert! Alert ! Session hijacking is detected')</script>";
    header("Refresh: 0; url=login.php");
    die();
}

// Retrieve user's current profile data
$username = ""; // Initialize variables to store current user's data
$fullname = "";
$primaryemail = "";


// Generate and store CSRF token in the session if it doesn't exist
if (!isset($_SESSION['nocsrftoken'])) {
    $_SESSION['nocsrftoken'] = bin2hex(openssl_random_pseudo_bytes(32)); // Generate a random token
}

// Validate CSRF token on form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Validate CSRF token from form submission
    $tokenFromForm = $_POST['nocsrftoken'] ?? '';

    if (!hash_equals($_SESSION['nocsrftoken'], $tokenFromForm)) {
        // Invalid CSRF token: Handle the error (e.g., display an error message)
        die("CSRF Token Validation Failed.");
    }



require "database.php";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST['username']) && isset($_POST['fullname']) && isset($_POST['primaryemail'])) {
        $username = $_POST['username'];
        $fullname = $_POST['fullname'];
        $primaryemail = $_POST['primaryemail'];

        // Update user's profile
        if (updateUserProfile($username, $fullname, $primaryemail)) {
            echo "Profile updated successfully.";
        } else {
            echo "Failed to update profile.";
        }
    } else {
        echo "All fields are required.";
    }
}
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Update Profile</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h2>Edit Profile</h2>
    <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
        <label>Username:</label><br>
        <input type="text" name="username" value="<?php echo $username; ?>" ><br>
        <label>Full Name:</label><br>
        <input type="text" name="fullname" value="<?php echo $fullname; ?>"><br>
        <label>Primary Email:</label><br>
        <input type="text" name="primaryemail" value="<?php echo $primaryemail; ?>"><br>
        <input type="hidden" name="nocsrftoken" value="<?php echo htmlspecialchars($_SESSION['nocsrftoken'
        <input type="submit" value="Update Profile">
    </form>
</body>
</html>
```

Included `logout.php`

```
<?php
session_start();
session_destroy();
?>

<!DOCTYPE html>
```

```
<html>
<head>
    <title>User Registration</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <br>
        <p> You are logout! </p>
        <a href="login.php">Login again</a>
    </div>
</body>
</html>
```

Included database.php

```
<?php
$mysqli = new mysqli('localhost', 'a', '1234', 'ip2');

if($mysqli->connect_errno) {
    printf("Database connection failed: %s\n", $mysqli->connect_error);
    return FALSE;
}


function addNewUser($username, $password, $fullname, $primaryEmail) {
    global $mysqli;

    // Basic validation checks
    if (empty($username) || empty($password) || empty($fullname) || empty($primaryEmail)) {
        return false; // Ensures that no field is empty
    }

    if (!filter_var($primaryEmail, FILTER_VALIDATE_EMAIL)) {
        return false; // Ensures the email is in a valid format
    }

    // Here you can add additional validation as needed, e.g., minimum lengths
    if (strlen($password) < 8) {
        return false; // Password must be at least 8 characters
    }

    // Hash the password using a more secure method
    $hashedPassword = md5($password);

    $preparedSql = "INSERT INTO users (username, password, fullname, primaryEmail) VALUES (?, ?, ?, ?)";
    $stmt = $mysqli->prepare($preparedSql);
    if (!$stmt) {
        return false; // Could not prepare the statement
    }

    $stmt->bind_param("ssss", $username, $hashedPassword, $fullname, $primaryEmail); // Bind the variables

    if ($stmt->execute()) {
        return true;
    } else {
        return false;
    }
}


function checklogin($username, $password) {
        global $mysqli;
        if($mysqli->connect_errno){
            printf("Database connection failed: %s\n", $mysqli->connect_errno);
            exit();
        }

        if (empty($username) || empty($password)) {
        return false; // Return false if username or password is empty
        }

        $prepared_sql = "SELECT * FROM users WHERE username=? AND password = md5(?)";
        $stmt = $mysqli->prepare($prepared_sql);
        $stmt->bind_param("ss", $username, $password);
        $stmt->execute();
        $result = $stmt->get_result();
        if($result->num_rows ==1)
            return TRUE;
        return FALSE;
    }




function updatePassword($username, $newPassword) {
    global $mysqli;

    if ($mysqli->connect_errno) {
        printf("Connect failed: %s\n", $mysqli->connect_error);
        exit();
    }
    $hashed_password = md5($newPassword);

    $prepared_sql = "UPDATE users SET password = ? WHERE username = ?";
    $stmt = $mysqli->prepare($prepared_sql);
    $stmt->bind_param("ss", $hashed_password, $username);

    if ($stmt->execute()) {
        $stmt->close();
        $mysqli->close();
        return true;
    } else {
        $stmt->close();
        $mysqli->close();
        return false;
    }
}

function updateUserProfile($username, $fullname, $primaryemail) {
    global $mysqli;
    if ($mysqli->connect_errno) {
        printf("Connect failed: %s\n", $mysqli->connect_error);
        exit();
    }
    $prepared_sql = "UPDATE users SET fullname = ?, primaryemail = ? WHERE username = ?";
    $stmt = $mysqli->prepare($prepared_sql);
    $stmt->bind_param("sss", $fullname, $primaryemail, $username);

    if ($stmt->execute()) {
        return true;
    } else {
        return false;
    }
}

// Function to fetch user details by username
function getUserDetails($username) {
    global $mysqli;
    $stmt = $mysqli->prepare("SELECT fullname, primaryemail FROM users WHERE username = ?");
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $result = $stmt->get_result();
    $userDetails = $result->fetch_assoc();
    $stmt->close();
    return $userDetails;
}

?>
```

Included `style.php`

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f8f8f8;
    text-align: center;
    padding-top: 50px;
    margin: 0;
}

.container {
    max-width: 400px;
    margin: 50px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    display: flex;
    flex-direction: column;
    align-items: center;
}

h1 {
    text-align: center;
```

```css
    color: #ff1493;
    margin-bottom: 20px; /* Added margin below heading */
}

#digit-clock {
    text-align: center;
    margin-bottom: 20px;
    color: #ff6347;
}

.form {
    text-align: center;
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 80%; /* Set width of form */
}

.text_field,
.button,
input[type="text"],
input[type="password"],
input[type="submit"] {
    padding: 10px;
    margin: 10px;
    width: 100%; /* Adjusted width to fit container */
    border: 2px solid #87cefa;
    border-radius: 50px;
    outline: none;
    box-sizing: border-box;
}

.button,
input[type="submit"] {
    background-color: #ff1493;
    color: #fff;
    font-weight: bold;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.button:hover,
input[type="submit"]:hover {
    background-color: #ff6347;
}

.error-message {
    color: red;
    margin-top: 5px;
}

.login-link {
    display: inline-block;
    margin-top: 20px;
    padding: 10px 20px;
    font-size: 16px;
    text-decoration: none;
    color: #fff;
    background-color: salmon;
    border-radius: 5px;
    transition: background-color 0.3s ease;
}

.login-link:hover {
    background-color: #ff8080; /* Lighter shade of salmon on hover */
}


a {
    display: inline-block;
    margin-top: 20px;
    padding: 10px 20px;
    font-size: 16px;
    text-decoration: none;
    color: #fff;
    background-color: salmon;
    border-radius: 5px;
    transition: background-color 0.3s ease;
}

a:hover {
    background-color: #ff8080; /* Lighter shade of salmon on hover */
}
```

Included `database-account.sql`

```
create database ip2;
create user 'a'@'localhost' IDENTIFIED BY '1234';
grant ALL on ip2.* TO 'a'@'localhost';
```

Included `database-data.sql`

```
-- if the table exists, delete it
DROP TABLE IF EXISTS users;

-- create a new table
CREATE TABLE users(
    username VARCHAR(100) PRIMARY KEY,
    password VARCHAR(100) NOT NULL,
    fullname VARCHAR(100),
    primaryemail VARCHAR(100)
);

-- Insert data into the users table
INSERT INTO users (username, password, fullname, primaryemail) VALUES ('amit', MD5('1234'), 'Amit Gaddi',
INSERT INTO users (username, password, fullname, primaryemail) VALUES ('admin', MD5('1234'), 'Admin', 'adm
```