SQL Injection Vulnerability Demonstration

Project Title

Understanding SQL Injection in PHP Web Applications

Table of Contents

---

1. Introduction

This project demonstrates how a SQL Injection vulnerability can be exploited in a PHP-based web application. It helps in understanding how attackers manipulate SQL queries through user input to bypass authentication or retrieve sensitive data.

---

2. Objective

The key objectives of this project are:

- To demonstrate the working of a SQL Injection attack
- To understand how such vulnerabilities arise in PHP applications
- To learn methods to fix and prevent SQL Injection using secure coding practices

---

3. Technologies Used

- Programming Language: PHP
- Database: MySQL
- Server Environment: MAMP or XAMPP (for local development)
- Interface: HTML-based login form

---

4. Project Structure

The project typically consists of the following components:

- A database configuration file for connecting to MySQL
- An HTML form to accept login credentials
- A script that processes the login and contains the SQL vulnerability
- A page that displays upon successful login
- A SQL file to set up the required database and table with sample user data

---

5. How to Set Up the Project

1. Place the project folder in the `htdocs` directory (for XAMPP) or the equivalent directory in MAMP.

2. Launch Apache and MySQL servers through MAMP or XAMPP.

3. Open `phpMyAdmin` by visiting `http://localhost/phpMyAdmin`.

4. Create a new database, for example, `sql_injection_demo`.

5. Import the provided `.sql` file to create the users table and insert sample data.

6. Access the project in the browser at `http://localhost/your_project_folder`.

---

6. Understanding the Vulnerability

SQL Injection occurs when user inputs are directly embedded into SQL queries without any form of input validation or escaping. This allows an attacker to alter the structure of the SQL query, potentially bypassing login mechanisms or accessing unauthorized data.

---

7. Testing the SQL Injection

To test for SQL Injection:

- Navigate to the login form in the browser.

- Enter crafted input that manipulates the SQL query (such as a condition that always evaluates to true).

- If the application is vulnerable, it will authenticate the user without valid credentials, indicating successful exploitation.

This step demonstrates how attackers can bypass authentication in a vulnerable application.

---

8. How to Prevent It

SQL Injection vulnerabilities can be mitigated using the following practices:

- Use prepared statements and parameterized queries to separate data from SQL logic.
- Validate and sanitize all user inputs before using them in database queries.
- Avoid displaying raw database errors to users.
- Regularly update and audit your code for security flaws.

---

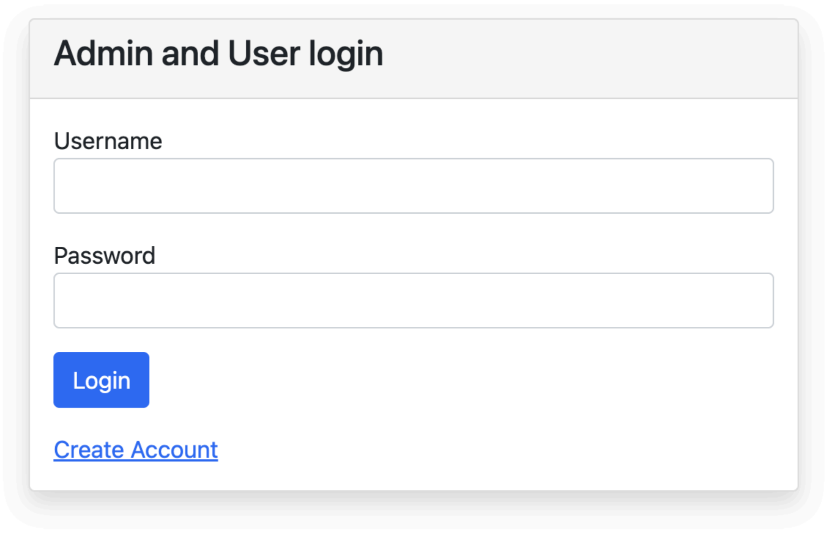
9. Conclusion

This project highlights how easily SQL Injection can be exploited if proper precautions are not taken. By simulating the attack and then securing the code, developers can better understand the importance of secure coding practices in PHP applications.

This hands-on experience strengthens the understanding of web application security and demonstrates the real-world impact of improper input handling.

Source code : https://drive.google.com/drive/folders/15gGZBny_FI_Ib0dTJ2f3SunJA3XJWhik?usp=sharing

9. Screenshots

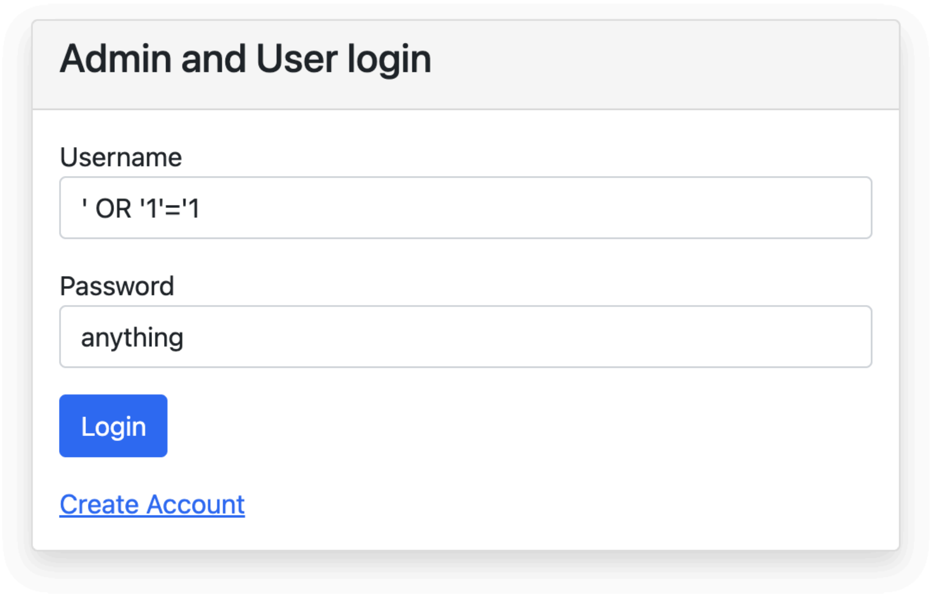


Admin and User login

Username

Password

Create Account



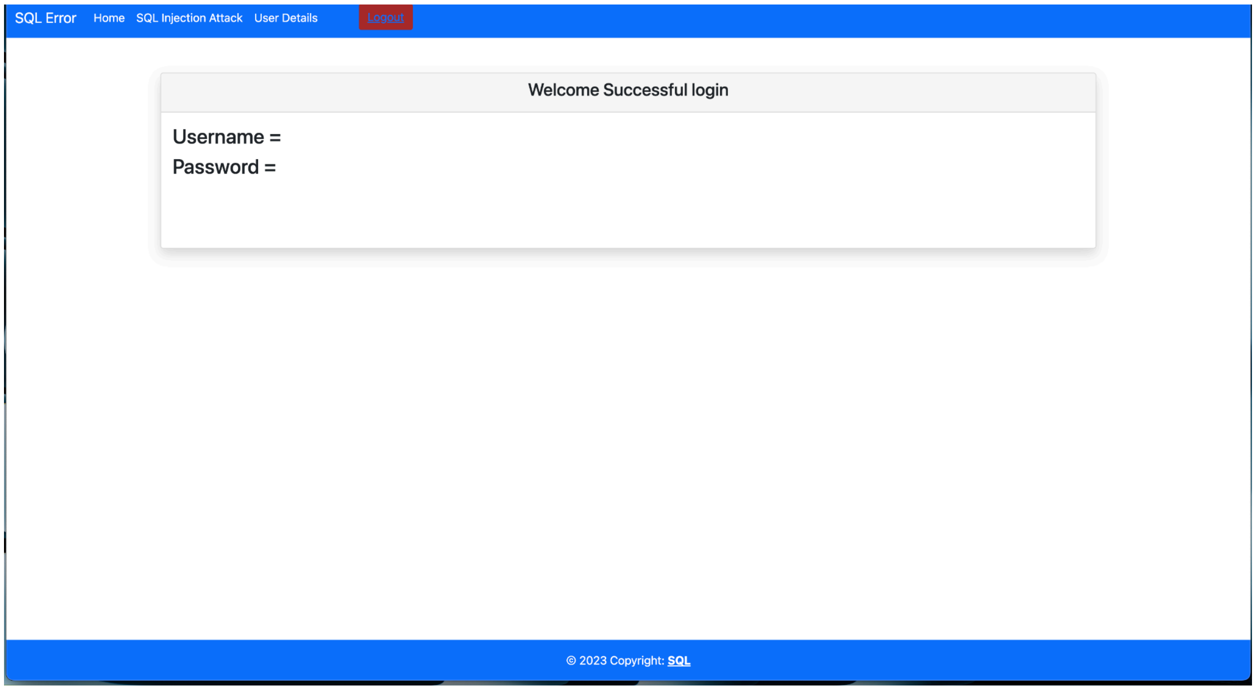
Admin and User login

Username

' OR '1'='1

Password

anything

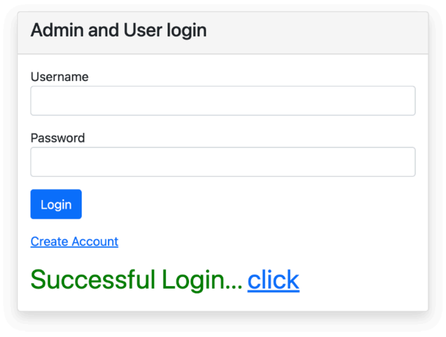


Welcome Successful login

Username =

Password =

© 2023 Copyright: SQL



Admin and User login

Username

Password

Successful Login... click

SELECT * FROM user WHERE username = '' OR '1'='1' AND password = 'anything'