Standards Lab 2 Secure Coding

This project demonstrates how to build a secure web application by applying fundamental secure coding practices.

Apply the following to make the code secure, also check the TODO's added to the code to help you but not everything is left as a TODO it's just a quick guide make sure you read the following well and apply them.

Secure the code:

1. **Input Validation**

   Check all user inputs before processing.

   For example, verify that text fields don't contain malicious characters like < or >.

2. **Password Hashing**

   Always store passwords securely using a hashing algorithm like **bcrypt**.

   Never store passwords as plain text.

3. **Encryption**

   Sensitive data (like transaction details) should be encrypted before being saved in files or databases.

4. **Logging Vulnerabilities**

   Monitor and log suspicious activity, such as repeated failed logins or unusual inputs.

   The log should trigger alerts when common vulnerabilities (e.g., SQL Injection, XSS attempts) are detected.

5. **CSRF Protection**

   Use CSRF tokens in all sensitive forms to prevent unauthorized actions by attackers.

6. **Prevent SQL Injection**

Use **parameterized queries** instead of directly inserting user input into SQL commands.

7. **Secure Registration Page**

   Include checks for duplicate users and validate all fields (e.g., valid email and password).

   Ensure passwords are hashed before saving them.

8. **Starter Page**

   Create a landing page where users can either **register** or **log in**.

9. **Error Handling**

   Make sure errors are logged but not shown directly to users.

   Prevent leaking system details through pup-up error messages.

10. **Prevent XSS (Cross-Site Scripting)**

    Sanitize user inputs by removing or escaping special characters.

11. **Add Rate Limiting**

Bonus:
1-Add a simple security dashboard where administrators can view security logs, the number of failed login attempts, and potential threats such as CSRF or XSS attacks. Display it visually with graphs and tables to make it easy to track trends and issues.

2-The code Provided and Visuals are messy refactor the code and add visuals making the site and the code cleaner.

Test Vulnerabilities

1. Attempt SQL Injection on login and see if it works.
2. Add unsanitized comments and test for XSS.
3. Create a CSRF attack and verify it on the transfer page.

Submit a PDF file containing screen shots of your attempts and how these vulnerabilities appear (e.g: your input which triggers the vulnerability)

You can trigger these vulnerabilities in the login page, comments page and transfer page.

Submit **a secure version of the project with explanations of fixes in a video with your screen shared.**

Submission format .zip folder STUDENTNAME_CODE

Maximum Number of team members is 3.