Project: Building a Secure Web Application - Detection and Mitigation of Security Vulnerabilities

Objective:

In this project, you will develop a simple web application and secure it against common vulnerabilities. This will help you apply key concepts from the course, such as encryption, access control, malware detection, SQL injection, web security, and network security. By the end of this project, you should be able to demonstrate secure coding practices and effectively mitigate security threats in web applications.

Instructions:

1. Web Application Setup (4 points):

- **Framework**: Choose any framework you are comfortable with (e.g., Flask for Python; Node.js with Express for JavaScript; etc.).
- **Functionality**: Develop a simple user management system with the following features:
    - **User Registration**: Users should be able to create an account by providing a username and password.
    - **Login**: Users can log in with their credentials.
    - **Dashboard**: After logging in, users should be able to view a dashboard displaying basic information (e.g., their username, profile details).

Points Breakdown:

- Basic functionality (register, login, dashboard) – **3 points**
- Code structure and organization – **1 point**

2. Detect and Mitigate Vulnerabilities (7 points):

Your web application should be vulnerable to at least five common security issues. Then, you will mitigate these vulnerabilities using the following techniques:

- **SQL Injection (1 point)**: The login and registration pages should initially be vulnerable to SQL injection. Then, implement parameterized queries to fix the vulnerability.
- **Weak Password Storage (1 point)**: Initially, passwords should be stored using an insecure method (e.g., MD5 hash). You should then implement a stronger hashing mechanism (e.g., bcrypt).
- **Cross-Site Scripting (XSS) (1 point)**: Allow user-generated content (e.g., comments) to be rendered without proper sanitation. Then, implement input sanitization to protect against XSS attacks.
- **Access Control (1 point)**: Implement a role-based access control (RBAC) system. Initially, users should be able to access pages they should not (e.g., an admin page). Then, implement proper access control based on user roles.

- **Encryption (2 points)**: Implement encryption to protect sensitive data (e.g., passwords, session tokens). Ensure that sensitive data is stored securely and transmitted over HTTPS (use TLS/SSL).

3. Report and Documentation (4 points):

- **Security Report (2 points)**: Write a report that describes:
  - The vulnerabilities you identified in your application.
  - The steps you took to mitigate each vulnerability.
  - Any challenges you encountered and how you solved them.
- **Code Explanation (2 points)**: Provide clear, concise comments and documentation within your code. Explain your decisions for key security implementations (e.g., why you chose *bcrypt* for password storage, how you ensured secure communication).

Submission Requirements:

1. Submit a link to your project repository (e.g., GitHub)
2. Include a **README.md** file with the following:
   - Overview of your project.
   - Steps to run the application.
   - Any instructions needed to test the security features (e.g., how to test SQL injection, XSS prevention, etc.).

Evaluation Rubric:

Criteria	Max Points	Description
Web Application Functionality	**4**	Basic functionality for user registration, login, and dashboard. All features must be working correctly and intuitively.
SQL Injection Mitigation	**1**	Proper use of parameterized queries to prevent SQL injection vulnerabilities.
Weak Password Storage Fix	**1**	Secure password hashing using bcrypt or other strong algorithms.
XSS Prevention	**1**	Proper input sanitization to prevent Cross-Site Scripting vulnerabilities.
Access Control Implementation	**2**	Role-based access control system ensuring users can access only appropriate resources (e.g., admin pages).
Encryption of Sensitive Data	**2**	Implementing encryption for sensitive data (e.g., session cookies, passwords). Ensure HTTPS for secure communication.
Security Report	**2**	A well-written report that discusses vulnerabilities, fixes, and challenges faced.

Criteria	Max Points	Description
Code Explanation and Documentation	**2**	Clear and detailed comments and explanations in the code.
Total	**15**	

Additional Notes:

- You are allowed to use online resources for learning, but you must not directly copy and paste solutions. Please ensure that all code is written by you, and credit any external resources/ libraries you reference.
- Ensure all features work as expected and thoroughly test your web application for vulnerabilities.
- Follow best practices for secure coding and document all decisions in your report.

Due Date: Sunday May 4, 2025