Alexandria National University
Faculty of Computer and Information
Lecturer: Dr. Mohamed Hassan

Course: Secure Software Development

Final project:

Secure Software Development Final Project Instructions

Objective

This project aims to design and develop a secure software application of your choice, integrating the security measures studied during the semester. This project aims to help you apply theoretical knowledge to practical scenarios, ensuring the software is functional and secure.

Project Requirements

1. **Software Application**:
    - You have two options for this part:
        - **Develop a software application from scratch**, choosing the type and features of your application.
        - **Enhance an existing software application** with security features.
    - Teams should consist of **2 to 4 members**.
2. **Security Criteria**:
    - Implement **at least 5 security features** from the following list:
        - **Using Java Security APIs and Libraries**:
            - Parsing and Logging Libraries.
            - Validator API for input validation.
            - Encoder Library for output encoding.
        - **Store Password Hashes**:
            - Use a secure hashing algorithm (e.g., SHA-256, bcrypt) for storing passwords securely.
        - **Attack Mitigation Examples**:
            - Implement protections against SQL Injection.
            - Include code that demonstrates mitigation of Buffer Overflow vulnerabilities.
        - **Multi-Factor Authentication (MFA)**:
            - Add MFA to secure critical parts of the application.

        - **Monitoring Application Security**:
            - Use JDK tools and the Security Manager for:
                - Access Control.
                - File Permission handling.

3. **Software Development Lifecycle (SDLC)**:
    - Ensure your project follows the **5 SDLC phases**:
        - **Planning**: Define the purpose, features, and scope of your software.
        - **Design**: Include security measures in your design and prepare diagrams (e.g., architecture diagrams).
        - **Implementation**: Code your application with a focus on security.
        - **Testing**: Test your application thoroughly for vulnerabilities (e.g., SQL Injection, Buffer Overflow).
        - **Maintenance**: Prepare a plan for future updates and handling potential security issues.

Deliverables

1. **Software Application**:
    - Submit the complete source code with proper comments.
2. **Presentation**:
    - Prepare a **10-minute presentation** where:
        - **Part 1**: Highlight the purpose of your software and its features.
        - **Part 2**: Describe the SDLC phases and the security measures applied in each phase.
        - **Part 3**: Demonstrate how your implemented security features enhance the application.
    - **Team Member Participation**:
        - Each member must present part of the presentation.
        - Each team member will be asked questions about the code to ensure individual contribution and understanding.
3. **Diagrams**:
    - Create **Use, Misuse, and Abuse Case Diagrams** using any free diagramming tool (e.g., Lucidchart, Draw.io, or StarUML).
    - Include these diagrams as part of the presentation.

4. **Report**:
    - Submit a **detailed report** (PDF format) covering:
        - Project description.
        - SDLC phase explanations.
        - Code explanations for each implemented security feature.
        - Steps to replicate your implementation.
        - Justifications for each security feature used.

Timeline and Discussion

- **Discussion and Presentation**:
    - Discussions will take place in **Week 14**.
    - Each team will have an allocated time slot to present their work.
    - Teams must be prepared to answer questions during the discussion session.

Free Tools for Case Diagrams

1. **Lucidchart** (https://www.lucidchart.com/)
2. **Draw.io** (https://app.diagrams.net/)
3. **StarUML** (https://staruml.io/)