Arab Academy for Science Technology and Maritime Transport College
of Computing and Information Technology

Course	**Software Security**
Course Code	**CCY3101**
Lecturer	**Dr. Nada Hany Sherief**
TA	**Salma Elkady**

12th Project – Milestone 2

Part 1 - E-Learning System with Secure Logging and Secure Factory Pattern.

This project aims to Implement a **secure logger** and secure **factory pattern** to enhance the security and maintainability of the E-Learning System available at

https://github.com/SuwaidAslam/E-Learning-System-Java-GUI-Application,

1. Secure Logger (Review Section 9)

Develop a custom logger class that logs system events and errors while safeguarding sensitive information.

Key Requirements:

- **Sensitive Data Logging:** Securely log sensitive data (e.g., passwords, usernames).
- **Log Destination:** Configure the logger to read and write log files securely.
- **Classes to Implement:**
  - `SecureLoggerFactory()`
  - `SecureLogger()`
  - `EncryptLogger()`

2. Secure Factory Pattern (Review Section 8)

Implement the factory pattern to create different user types with varying accessibility levels, promoting code reusability and loose coupling.

Key Requirements:

- Create user types such as **Student**, **Instructor**, and **Administrator**.
- Manage varying security levels for database connections.
- **Classes to Implement:**
  - DatabaseConnection()
  - LessSensitiveDBConnection()
  - SensitiveDBConnection()
  - AbstractDBFactory()
  - LessSensitiveDBFactory()
  - SensitiveDBFactory()
  - SecurityCredentials()

Part 2 - Penetration Testing Project

Conduct a comprehensive penetration testing exercise on **Mutillidae**, a vulnerable web application running on Metasploitable 2. This exercise focuses on identifying and exploiting vulnerabilities related to spidering, target scoping, SQL injection, Cross-Site Scripting (XSS), and brute force attacks.

Scope of Testing:

- **Spidering and Target Scoping:**
  - Utilize automated tools (Burp Suite) to spider the application and identify all accessible web pages and parameters.
  - Analyze the identified pages and parameters to prioritize targets for further testing based on their potential vulnerability (as forms, login pages, user input fields).
- **SQL Injection:**
  - Test input fields (e.g., search boxes, login forms) for vulnerabilities using Burp Suite.
  - Identify points where malicious SQL code can be injected to manipulate database queries.
- **Cross-Site Scripting (XSS):**
  - Test for both reflected and stored XSS vulnerabilities across different input points.
  - Inject malicious JavaScript code into various input fields and observe the impact on the application and potential impact on users.
- **Brute Force Attacks:**
  - Attempt to crack user passwords using brute force techniques against the login functionality.
  - Utilize Burp Suite Intruder to automate the brute force process.

Methodology:

- **Information Gathering:**
  - **Spidering:** Utilize automated tools (Burp Suite) to spider the Mutillidae application and map its entire structure, including all web pages, parameters, and forms.
  - **Target Scoping:** Analyze the spidering results to identify high-value targets for further testing, such as:
    - User input fields in forms (login, registration, search, and others).
    - Dynamic parameters in URLs
    - Cookies and other session-related data.
- **Vulnerability Assessment:**
  - **SQL Injection:**
    - Utilizing tools like Burp Suite Intruder to automate the testing process and identify potential injection points.
  - **XSS:**
    - **Reflected XSS:**
      - Inject malicious JavaScript code (as <script>alert('XSS');</script>) into various input fields and observe if the code is reflected back to the user.
    - **Stored XSS:**

        - Inject malicious JavaScript code into input fields where the data is stored (as comments, forums, profiles).
        - Check if the injected code is stored and executed when other users access the stored data.
    - **Brute Force Attacks:**
        - **Login Page:** Attempt to crack user passwords by:
            - Using a wordlist of common passwords or custom wordlists.
            - Varying username combinations (if applicable).
            - Implementing rate limiting to avoid triggering intrusion detection systems.

- **Exploitation and Impact Assessment:**
    - **Brute Force Attacks (Review Section12)**
        - **Successful Attacks:** If passwords are successfully cracked, assess the potential impact:
            - Account takeover
            - Unauthorized access to sensitive data
            - Lateral movement within the system (if applicable)
    - **SQL Injection (Review Section 13)**
        - If successful SQL injection is found, attempt to:
            - Extract sensitive data from the database (e.g., usernames, passwords, credit card information).
            - Modify or delete existing data.
            - Gain remote code execution on the server.
    - **XSS (Review Section 13)**
        - If successful XSS is found, assess the potential impact:
            - **Data Theft:** Steal user cookies, session IDs, or other sensitive information.
            - **Session Hijacking:** Take over user sessions.
            - **Phishing:** Trick users into revealing sensitive information.
            - **Defacement:** Modify the website's appearance.

- **Reporting**

    - Document all findings in a comprehensive report, including:
        - Detailed methodology used
        - List of identified vulnerabilities (SQL injection, XSS, and brute force)
        - Proof-of-concept exploits for each vulnerability
        - Potential impact of each vulnerability
        - Remediation recommendation.

Submission Notes

1. **Design Pattern Implementation:**
   - Implement the secure logger and factory pattern in the GitHub repository provided.
2. **Penetration Testing Report:**
   - Submit a professional report covering all project requirements, including detailed explanations, methodology, and findings.
   - Include screenshots to document penetration testing results.
3. **Source Code Submission:**
   - Attach the source files for the project within the report or provide a link to a hosted version (e.g., Google Drive).
4. **Group Work:**
   - This is a group project for teams of **3–4 students**.
5. **Original Work:**
   - All submitted work must be original. Cases of plagiarism or collusion will result in disciplinary action.