# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **11/13/2024** | **Jesse J. Aranda** | **3-1 Project One Submission** |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In this report, identify your security vulnerability findings and recommend the next steps to remedy the issues you have found.

* Respond to the five steps outlined below and include your findings.
* Respond using your own words. You may also include images or supporting materials. If you include them, make certain to insert them in the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Jesse J. Aranda

**1. Interpreting Client Needs**

Artemis Financial, a consulting company specializing in individualized financial plans, needs to ensure that its web-based software application is secure to protect sensitive customer data, such as savings, retirement, investments, and insurance information. The value of secure communications to Artemis Financial is paramount, as it maintains customer trust and ensures compliance with various regulatory requirements.

* **Value of Secure Communications:** Secure communications are crucial to protect sensitive financial data from interception and unauthorized access. Ensuring data integrity and confidentiality is essential to maintain customer trust and comply with legal obligations.
* **International Transactions:** If Artemis Financial engages in international transactions, it must adhere to international data protection regulations such as GDPR for EU customers. This would involve ensuring secure data transmission across borders and complying with specific regional security standards.
* **Governmental Restrictions:** Compliance with regulations like the Gramm-Leach-Bliley Act (GLBA) and the Payment Card Industry Data Security Standard (PCI-DSS) is necessary. These regulations impose strict guidelines on how financial data should be handled and protected.
* **External Threats:** Potential threats include phishing attacks, man-in-the-middle attacks, SQL injection, cross-site scripting (XSS), and distributed denial-of-service (DDoS) attacks. Future threats may involve advanced persistent threats (APTs) and evolving malware targeting financial systems.
* **Modernization Requirements:** The application must integrate secure open-source libraries, follow best practices for secure coding, and utilize evolving web application technologies such as secure RESTful APIs and encryption standards. Ensuring the application is updated with the latest security patches and using frameworks that support secure development is also crucial.

**2. Areas of Security**

Based on the vulnerability assessment process flow diagram, the following areas of security are relevant to Artemis Financial’s software application:

* **Authentication and Authorization:** Ensuring that only authorized users can access sensitive data and perform certain actions is crucial. Implementing multi-factor authentication (MFA) and role-based access control (RBAC) can enhance security.
* **Data Protection:** Encrypting sensitive data both at rest and in transit is necessary to protect against data breaches and ensure data integrity and confidentiality.
* **Input Validation:** Properly validating and sanitizing user inputs can prevent common attacks such as SQL injection and XSS.
* **Session Management:** Securely managing user sessions, including implementing timeouts and protecting against session hijacking, is essential to maintain user security.
* **Error Handling and Logging:** Properly handling errors and securely logging events can help in identifying and responding to security incidents promptly.
* **Configuration Management:** Ensuring that the application’s configuration settings are secure and regularly updated helps to prevent security misconfigurations.
* **Dependency Management:** Regularly updating and managing third-party libraries and dependencies to address known vulnerabilities is critical.

**3. Manual Review**

The manual review of Artemis Financial’s code base revealed the following vulnerabilities:

1. **SQL Injection:** The login method in AuthenticationService.java accepts user inputs without proper validation, making it susceptible to SQL injection attacks.
2. **Cross-Site Scripting (XSS):** The userProfile.jsp page displays user input without proper encoding, which can lead to XSS attacks.
3. **Insecure Password Storage:** Passwords are stored using a weak hashing algorithm (MD5) in User.java, which can be easily cracked.
4. **Hardcoded Credentials:** Hardcoded database credentials are found in DatabaseConfig.java, posing a security risk if exposed.
5. **Missing Input Validation:** User inputs in RegistrationController.java are not properly validated, increasing the risk of malicious inputs.
6. **Sensitive Data Exposure:** Sensitive information is transmitted over HTTP instead of HTTPS in PaymentController.java.
7. **Unrestricted File Uploads:** The file upload functionality in DocumentService.java does not check file types, allowing potential malicious file uploads.
8. **Insufficient Logging:** The application lacks comprehensive logging for security-related events in AuditService.java.
9. **Insecure Direct Object References:** Direct object references without access control checks are found in AccountController.java.
10. **Cross-Site Request Forgery (CSRF):** The application does not implement CSRF protection tokens in forms, making it vulnerable to CSRF attacks.

**4. Static Testing**

Running the dependency-check tool revealed the following known vulnerabilities in the code base:

* **CVE-2020-1938:** A vulnerability in Apache Tomcat affects the AJP protocol. Recommended solution: Update to the latest version of Apache Tomcat.
* **CVE-2019-17571:** A vulnerability in Log4j affects the SocketServer class. Recommended solution: Update to Log4j 2.13.3 or later.
* **CVE-2021-44228:** A critical vulnerability in Log4j known as Log4Shell. Recommended solution: Update to Log4j 2.15.0 or later.
* **CVE-2020-9488:** A vulnerability in Apache Tomcat affects the handling of the Transfer-Encoding header. Recommended solution: Update to the latest version of Apache Tomcat.

**5. Mitigation Plan**

To address the identified security vulnerabilities, the following steps should be taken:

1. **SQL Injection:** Implement prepared statements and parameterized queries in AuthenticationService.java.
2. **Cross-Site Scripting (XSS):** Use proper encoding for user inputs in userProfile.jsp and implement a Content Security Policy (CSP).
3. **Insecure Password Storage:** Use a strong hashing algorithm such as bcrypt or Argon2 for password storage in User.java.
4. **Hardcoded Credentials:** Move database credentials to a secure environment variable or configuration management tool in DatabaseConfig.java.
5. **Missing Input Validation:** Implement comprehensive input validation using a validation framework in RegistrationController.java.
6. **Sensitive Data Exposure:** Ensure all data transmissions use HTTPS by configuring PaymentController.java to enforce HTTPS.
7. **Unrestricted File Uploads:** Implement file type validation and scanning for file uploads in DocumentService.java.
8. **Insufficient Logging:** Enhance logging for security events in AuditService.java and ensure logs are securely stored.
9. **Insecure Direct Object References:** Implement access control checks for object references in AccountController.java.
10. **Cross-Site Request Forgery (CSRF):** Implement CSRF protection tokens in forms across the application.