# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/25/2024** | **Kholood Alkohali** | **The best there is.** |

## 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

Kholood Alkohali

**1. Interpreting Client Needs**

Determine your client’s needs and potential threats and attacks associated with the company’s application and software security requirements. Consider the following questions regarding how companies protect against external threats based on the scenario information:

* What is the value of secure communications to the company?
* Are there any international transactions that the company produces?
* Are there governmental restrictions on secure communications to consider?
* What external threats might be present now and in the immediate future?
* What modernization requirements must be considered, such as the role of open-source libraries and evolving web application technologies?

Artemis Financial, a consulting company specializing in individualized financial plans, requires secure and modernized operations for their RESTful web API. To address their needs and protect against external threats, consider the following points:

* Secure communications are crucial for protecting sensitive financial data, maintaining customer trust, and ensuring compliance with regulations.
* If Artemis Financial handles international transactions, they must comply with various international data protection regulations such as GDPR, and secure communications are essential to protect data during these transactions.
* Compliance with financial and data protection laws such as GDPR, CCPA, and other local regulations is mandatory. Encryption and secure data handling practices are necessary to meet these requirements.
* Potential threats include cyber-attacks such as SQL injection, Cross-Site Scripting (XSS), man-in-the-middle attacks, and data breaches. Advanced persistent threats (APTs) and zero-day vulnerabilities also pose significant risks.
* Modernization Requirements:
  + Open-Source Libraries: While beneficial, open-source libraries can introduce vulnerabilities if not properly managed and updated.
  + Evolving Web Technologies: Adoption of the latest web technologies (e.g., frameworks, languages) to improve performance and security, but requires continuous learning and adaptation.

**2. Areas of Security**

Refer to the vulnerability assessment process flow diagram. Identify which areas of security apply to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

**Architecture Review:** Essential to understand the overall design and identify potential weaknesses in the system architecture.

**Input Validation:** Crucial for preventing injection attacks by ensuring that all user inputs are validated and sanitized.

**APIs:** Secure API interactions to prevent unauthorized access and data breaches through the API endpoints.

**Cryptography:** Use of strong encryption protocols to protect sensitive data both at rest and in transit.

**Client/Server Security:** Ensuring secure communication between client and server to protect data integrity and confidentiality.

**Code Quality:** Adherence to secure coding practices to minimize vulnerabilities such as buffer overflows and improper error handling.

**Encapsulation:** Protecting data structures to prevent unauthorized access and manipulation.

**3. Manual Review**

Continue working through the vulnerability assessment process flow diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

1. **SQL Injection:** Lack of prepared statements in FinancialDataController.java.
2. **Cross-Site Scripting (XSS):** Unsanitized user input in UserProfile.jsp.
3. **Insecure API Endpoints:** No authentication on certain API endpoints in ApiController.java.
4. **Hardcoded Credentials:** Hardcoded database credentials in DatabaseConfig.java.
5. **Sensitive Data Exposure:** Logging sensitive information in plain text in TransactionService.java.
6. **Broken Authentication:** Weak password policy in UserService.java.
7. **Outdated Libraries:** Use of an outdated version of the Spring Framework in pom.xml.
8. **Improper Error Handling:** Displaying stack traces to users in ErrorHandler.java.
9. **Insecure Direct Object References (IDOR):** Insufficient access control in AccountController.java.
10. **CSRF Vulnerability:** Lack of CSRF tokens in forms within AccountSettings.jsp.

**4. Static Testing**

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from the dependency-check report. Include the following items:

* The names or vulnerability codes of the known vulnerabilities
* A brief description and recommended solutions provided by the dependency-check report
* Any attribution that documents how this vulnerability has been identified or documented previously

1. CVE-2021-44228: Log4j Remote Code Execution Vulnerability
   * Description: A critical RCE vulnerability in Apache Log4j 2.x, allowing attackers to execute arbitrary code.
   * Recommended Solution: Update to Log4j 2.17.1 or later.
   * Attribution: Documented by Apache Software Foundation and widely recognized in security advisories.
2. CVE-2020-5398: Spring Framework Data Binding Vulnerability
   * Description: Vulnerability in Spring Framework allows attackers to manipulate data binding resulting in potential data breaches.
   * Recommended Solution: Upgrade to Spring Framework 5.2.2 or later.
   * Attribution: Identified and documented by Pivotal Software in their security advisories.
3. CVE-2019-17571: Logback Logging Framework Vulnerability
   * Description: Deserialization vulnerability in Logback leading to arbitrary code execution.
   * Recommended Solution: Update Logback to 1.2.3 or later.
   * Attribution: Documented in multiple security advisories including NVD.

**5. Mitigation Plan**

Interpret the results from the manual review and static testing report. Then identify the steps to mitigate the identified security vulnerabilities for Artemis Financial’s software application.

1. SQL Injection: Implement prepared statements and parameterized queries in FinancialDataController.java.
2. Cross-Site Scripting (XSS): Sanitize user inputs using libraries like OWASP Java Encoder in UserProfile.jsp.
3. Insecure API Endpoints: Implement proper authentication and authorization mechanisms in ApiController.java.
4. Hardcoded Credentials: Use environment variables or a secure vault for credentials in DatabaseConfig.java.
5. Sensitive Data Exposure: Mask or encrypt sensitive information in logs within TransactionService.java.
6. Broken Authentication: Enforce a strong password policy and use multi-factor authentication in UserService.java.
7. Outdated Libraries: Regularly update libraries and dependencies, and monitor for security patches.
8. Improper Error Handling: Configure a generic error page and avoid exposing stack traces in ErrorHandler.java.
9. Insecure Direct Object References (IDOR): Implement access control checks in AccountController.java.
10. CSRF Vulnerability: Add CSRF protection tokens to forms in AccountSettings.jsp.