# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **03/20/2025** | **Eddy Kwon** | **Initial Draft** |

## 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

Eddy Kwon

**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 requires robust security measures to protect sensitive financial data, including client savings, retirement, investments, and insurance plans. Secure communication is essential to prevent data breaches and unauthorized access, ensuring compliance with financial regulations such as GDPR and PCI DSS. If the company engages in international transactions, it must adhere to cross-border data protection laws. External threats such as SQL injection, cross-site scripting (XSS), and API vulnerabilities pose risks to data integrity and confidentiality. Additionally, as Artemis Financial modernizes its operations, it must manage the security risks associated with open-source libraries and evolving web application technologies, ensuring all dependencies remain up-to-date and vulnerabilities are mitigated.

**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.

Based on the Vulnerability Assessment Process Flow, the following security areas are relevant to Artemis Financials’ software application:

1. Input Validation – Ensuring all user inputs are validated prevents common attacks such as SQL Injection and Cross-Site Scripting (XSS), which can compromise financial data integrity.
2. APIs – As a RESTful web API, secure API interactions are crucial to prevent unauthorized access, data leaks, and API abuse. Implementing authentication and authorization measures is necessary.
3. Cryptography – Financial transactions and sensitive customer data require strong encryption to protect data at rest and in transit from potential breaches. Weak encryption could expose confidential financial details.
4. Client/Server Security – The system must maintain secure distributed composing, ensuring authentication and data integrity between client and server to prevent man-in-the-middle attacks and session hijacking.
5. Code Quality – Secure coding practices should be followed to eliminate vulnerabilities introduced through poor development practices, reducing the risk of buffer overflows, insecure dependencies, and logic flaws.
6. Encapsulation – Proper data encapsulation is necessary to prevent unauthorized access to sensitive information, ensuring financial records are only accessible to authorized users.

These areas are critical in securing Artemis Financials’ web-based application against external and internal security threats.

**3. Manual Review**

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

Upon manually inspecting the code base for Artemis Financials web-based application, the following vulnerabilities were identified:

1. Use of Outdated Libraries – Several dependencies in pom.xml, such as spring-core-5.2.3. RELEASE and logback-classic-1.2.3, are outdated and known to contain security vulnerabilities, making the application susceptible to attacks like Remote Code Execution (RCE).
2. Hardcoded Credentials – Some configuration files contain plaintext credentials, which pose a high risk of unauthorized access if exposed in a security breach.
3. Lack of Input Validation – Certain API endpoints accept user input without proper sanitization, making the system vulnerable to SQL Injection and Cross-Site Scripting (XSS) attacks.
4. Exposed Sensitive Information in Logs – The application logs detailed error messages, including stack traces and database queries, which can be exploited by attackers to gain insight into the system architecture.
5. Insecure API Endpoints – Several API routes lack authentication and authorization mechanisms, allowing unauthorized users to access sensitive financial data.
6. Missing HTTPS Enforcement – The application allows HTTP connections, increasing the risk of Man-in-the-Middle (MITM) attacks where an attacker can intercept and manipulate transmitted data.
7. Improper Session Management – User sessions remain active beyond the intended logout period, making the system vulnerable to session hijacking.
8. Weak Encryption for Stored Data – Customer financial data is stored without strong encryption algorithms, making it easier for attackers to extract sensitive information.
9. Lack of Rate Limiting on API Calls – The system does not implement rate-limiting measures, making it vulnerable to brute-force attacks where attackers can repeatedly attempt unauthorized access.
10. Unrestricted File Uploads – The application lacks file type validation, allowing malicious files to be uploaded, which could lead to server compromise and malware execution.

These vulnerabilities present significant risks to Artemis Financials application security and require immediate attention to mitigate potential cyber threats.

**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

A Maven dependency check was conducted on Artemis Financial’s software application to identify security vulnerabilities within the codebase. The following key vulnerabilities were detected:

**Top Vulnerabilities Identified:**

| **Dependency** | **Vulnerability Code** | **Severity** | **Description** | **Recommended Solution** |
| --- | --- | --- | --- | --- |
| **spring-core-5.2.3.RELEASE.jar** | **CVE-2022-22965** | **Critical** | **Remote code execution vulnerability in Spring framework** | **Upgrade to a patched version, such as Spring 5.3.18+** |
| **logback-classic-1.2.3.jar** | **CVE-2021-42550** | **High** | **Logback deserialization vulnerability leading to RCE** | **Update Logback to version 1.2.9+** |
| **json-smart-2.3.jar** | **CVE-2023-1370** | **High** | **JSON Smart library vulnerable to remote code execution** | **Upgrade to version 2.4.8+** |
| **hibernate-validator-6.0.18.Final.jar** | **CVE-2020-10693** | **Medium** | **Insecure input validation vulnerability** | **Upgrade Hibernate Validator to version 6.1.5+** |
| **jackson-databind-2.10.2.jar** | **CVE-2020-25649** | **High** | **Allows deserialization of untrusted data, leading to security risks** | **Update to Jackson 2.12.6+** |
| **spring-data-rest-webmvc-2.6.5.RELEASE.jar** | **CVE-2017-8046** | **Critical** | **Vulnerability in REST controllers exposing unauthorized access** | **Use Spring Data REST 3.2.5+** |
| **snakeyaml-1.25.jar** | **CVE-2022-1471** | **Critical** | **Allows remote attackers to execute arbitrary code** | **Upgrade SnakeYAML to version 2.0+** |
| **tomcat-embed-core-9.0.30.jar** | **CVE-2020-1938** | **Critical** | **Apache Tomcat Ghostcat vulnerability enables RCE** | **Update to Tomcat 9.0.40+** |

**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.

To mitigate the identified security vulnerabilities, Artemis Financial should update all outdated dependencies to their latest secure versions and apply regular security patches. Strengthening API security through authentication, authorization, and rate limiting will help prevent unauthorized access. Enforcing HTTPS for secure communication and improving session management will reduce risks of data breaches. Additionally, removing hardcoded credentials, sanitizing user inputs, and restricting alias usage in SnakeYAML will further enhance security and prevent exploitation.