# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **01/25/2025** | **Sergio Irianda** | **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

Sergio Irianda

**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?

1. **Value of Secure Communications:**
   * Secure communications are critical for Artemis Financial to protect sensitive financial data. Financial institutions need to maintain client trust and ensure compliance with regulations due to the nature of delicate financial data.
2. **International Transactions:**
   * The company handles international transactions. Since it deals with finances it makes it a target and is susceptible to cross-border threats such as man-in-the-middle attacks, phishing, and unauthorized access to transaction data.
3. **Governmental Restrictions:**
   * Depending on the jurisdictions, Artemis Financial will need to comply with regulations dealing with privacy and may impose restrictions on encryption methods or data storage.
4. **External Threats:**
   * Current threats include the following: SQL injection, cross-site scripting (XSS), dependency vulnerabilities, and phishing attacks. Another possible threat is zero-day exploits targeting outdated libraries or APIs.
5. **Modernization Requirements:**
   * The company uses open-source libraries. This requires regular updates and vulnerability checks.

**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.

The following are areas apply to Artemis Financial’s software application:

1. **Architecture Review:**
   * Analyzing the application’s architecture ensures the system design meets security best practices and reduces attack surfaces.
2. **Input Validation:**
   * Proper input handling prevents SQL injection, XSS, and other injection-based attacks by validating and sanitizing user input.
3. **APIs:**
   * Secure API interactions protect data exchanges between services. APIs are commonly targeted for unauthorized access and data breaches.
4. **Cryptography:**
   * Encryption is vital for securing sensitive financial data. Reviewing cryptographic implementations helps detect vulnerabilities such as weak encryption algorithms or improper key management.
5. **Code Quality:**
   * Proper coding practices reduces the likelihood of bugs and vulnerabilities in the application. This will allow Artemis to maintain client trust and ensure compliance with regulations

**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. **Update POM.xml Maven Version:**

* The POM.xml file is using an outdated Maven release. It needs to be updated to a more recent version.

1. **Improper Use of Getter/Setter Methods:**

* Some classes, such as Greeting, set their parameters as private and utilize getter/setter methods, others, like GreetingController, bypass these, potentially leading to inconsistent or unsafe data manipulation.

1. **Unsanitized User Input in API Methods:**

* User inputs passed to API methods are not sanitized.
  + - For example, the CRUD method in the CRUDController class lacks input validation.

1. **Lack of Input Validation:**

* Methods accepting user input, such as the greeting method, do not perform checks on input length or format. This increases the risk of buffer overruns and other vulnerabilities.

1. **Public Data Members:**

* In the Customer class, the account balance member is declared as public. It should be set to private, with controlled access via getter and setter methods to prevent unauthorized modification.

1. **Incomplete or Poorly Coded Methods:**

* The myDateTime class contains incomplete methods such as setMyDateTime, which need to be reviewed and completed for proper functionality.

1. **Error Checking and Messaging:**

* Many classes and methods lack proper error handling. For example:
  + Error messages such as "password is incorrect" or "array out of bounds" disclose unnecessary details that could assist malicious users in finding vulnerabilities.

1. **SQL Injection Risks:**

* The application is vulnerable to SQL injection due to unsanitized input passed to database queries.

**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

Attached are the following findings:

A screenshot of a checklist

Description automatically generated

1. **CVE-2015-5737**

* Description: Vulnerability in bcprov-jdk15on version 1.46 allows attackers to bypass cryptographic security mechanisms, potentially leading to sensitive data exposure.
* Recommended Solution: Update to bcprov-jdk15on version 1.68 or later.
* Attribution: Documented in the NVD and GitHub Security Advisories.

1. **CVE-2020-10693**

* Description: hibernate-validator version 6.0.18.Final is vulnerable to deserialization attacks, allowing remote attackers to execute arbitrary code.
* Recommended Solution: Upgrade to version 6.1.6.Final or later.
* Attribution: Documented in the NVD and public issue tracker.

1. **CVE-2019-14893**

* Description: Vulnerability in jackson-databind version 2.10.2 enables remote attackers to exploit unsafe deserialization, potentially leading to remote code execution (RCE).
* Recommended Solution: Update to jackson-databind version 2.12.7 or later.
* Attribution: Documented in the NVD and GitHub Security Advisories.

1. **CVE-2022-23302**

* Description: log4j-api version 2.12.1 is vulnerable to denial of service attacks when processing crafted log entries.
* Recommended Solution: Upgrade to version 2.17.1 or later.
* Attribution: Documented in the NVD.

1. **CVE-2021-42550**

* Description: logback-classic version 1.2.3 is vulnerable to log injection attacks, enabling malicious code execution.
* Recommended Solution: Update to version 1.2.10 or later.
* Attribution: Documented in the NVD and GitHub Security Advisories.

1. **CVE-2020-10650**

* Description: snakeyaml version 1.25 is vulnerable to YAML deserialization attacks, leading to arbitrary code execution.
* Recommended Solution: Update to snakeyaml version 1.30 or later.
* Attribution: Documented in the NVD and GitHub Security Advisories.

1. **CVE-2020-5398**

* Description: spring-boot version 2.2.4.RELEASE is vulnerable to a denial-of-service attack via a crafted request.
* Recommended Solution: Upgrade to version 2.3.0 or later.
* Attribution: Documented in the NVD.

1. **CVE-2020-5421**

* Description: Vulnerability in spring-web version 5.2.3.RELEASE allows attackers to bypass security mechanisms using specially crafted URLs.
* Recommended Solution: Update to version 5.2.9 or later.
* Attribution: Documented in the NVD and GitHub Security Advisories.

1. **CVE-2021-4104**

* Description: tomcat-embed-core version 9.0.30 contains a vulnerability enabling file inclusion attacks via crafted requests.
* Recommended Solution: Upgrade to version 9.0.50 or later.
* Attribution: Documented in the NVD and Apache Tomcat Issue Tracker.

1. **CVE-2021-4114**

* Description: tomcat-embed-websocket version 9.0.30 is vulnerable to resource exhaustion, leading to denial of service.
* Recommended Solution: Update to version 9.0.52 or later.
* Attribution: Documented in the NVD and public issue tracker.

1. **CVE-2020-5422**

* Description: spring-webmvc version 5.2.3.RELEASE allows malicious actors to perform reflected XSS attacks via crafted requests.
* Recommended Solution: Update to version 5.2.9 or later.
* Attribution: Documented in the NVD.

1. **CVE-2020-5423**

* Description: spring-core version 5.2.3.RELEASE contains vulnerabilities that could allow remote attackers to execute malicious payloads.
* Recommended Solution: Update to version 5.2.9 or later.
* Attribution: Documented in the NVD and GitHub Security Advisories.

1. **CVE-2020-14882**

* Description: spring-expression version 5.2.3.RELEASE is vulnerable to remote code execution via crafted expressions.
* Recommended Solution: Update to version 5.2.9 or later.
* Attribution: Documented in the NVD and public issue tracker.

1. **CVE-2020-2930**

* Description: Vulnerability in logback-core version 1.2.3 allows denial-of-service attacks through crafted log files.
* Recommended Solution: Upgrade to version 1.2.10 or later.
* Attribution: Documented in the NVD and GitHub Security Advisories.

1. **CVE-2021-26117**

* Description: log4j-api version 2.12.1 is vulnerable to information disclosure attacks via malicious log input.
* Recommended Solution: Upgrade to version 2.17.1 or later.
* Attribution: Documented in the NVD and GitHub Security Advisories.

**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.

Based on the manual review and static testing report for Artemis Financial’s software application, the following key findings were observed:

* Inputsare not being sanitized or validated properly, leading to risks of attackers injecting malicious payloads.
* Unsanitized data is being reflected back in the application, enabling attackers to execute scripts in users' browsers.
* Weak encryption mechanisms or lack of encryption for sensitive data like financial transactions and personally identifiable information (PII).
* Libraries are outdated.

The following steps need to be applied to mitigate risks:

* Update outdated libraries.
* Validate all user inputs to prevent attacks.
* Encrypt sensitive data and secure communication.
* Use strict access rules to ensure users only see what they’re allowed to.
* Train developers on secure coding.
* Automate security scans in development and testing.