# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/18/24** | **Dylan Schwering** |  |

## 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

Dylan Schwering

**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?

Secure communications are vital to the client because of the nature of their business. As a consulting company that deals with financial plans, the protection of this data is critical.

The company specifies that they deal with business, entrepreneurs, and even government agencies around the world, this would imply that there are international transactions.

There would be various governmental restrictions to consider, when working with financial data there would be restrictions, and considering the company works with government agencies the restrictions would be even more strict.

There would be numerous external threats to plan for, for instance there could be phishing attacks, sql injection, denial of service and more to consider.

Modernization requirements to consider would include making sure the use of open source libraries is secure, using encryption to keep the data safe, and using authentication and authorization to ensure only the correct people have access to the data.

**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.

Input Validation: As the company manages a piece of software that is client focused, there needs to be input validation to prevent attacks such as SQL injection.

Cryptography: Cryptography is vital for keeping sensitive financial data secure.

Secure Error Handling: Handling errors properly is essential in avoiding exposing sensitive information being leaked through error messages.

Secure Coding Practices: Creating secure code is vital to developing a secure application.

**3. Manual Review**

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

In DocData.java, the read document method does not use prepared statements which exposes it to SQL injection.

Again in DocData.java the database credentials are hardcoded in, which is a security risk if the code is exposed.

In CRUDController.java the read endpoint doesn’t perform any validation on business\_name, making it vulnerable to injection.

In customer.java account\_balance isn’t given an appropriate access modifier exposing it to misuse.

In GreetignsController.java greeting does not validate name which could lead to injection attacks

In myDateTime.java the variables aren’t given access modifiers

In CRUDController.java the controller directly exposes implementation details through API response.

**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

Bcprov-jdk15on-1.46.jar: A java implementation of cryptographic algorithms; the software does not properly ensure that communications come from the right source

Jackson-databind-2.15.0.jar: General data-binding functionality; allows attackers to cause a denial of service

Logback-core-1.4.7.jar: logback-core module; allows attackers to mount denial of service attacks

Snakeyaml-1.33.jar: YAML 1.1 parser and emitter for Java; does not restrict types, can lead to remote code execution; recommended using SafeConstructor to restrict possibility, recommend updating

Spring-aop-6.0.9.jar: Spring AOP; possibility of denial of service attacks

Spring-boot-3.1.0.jar: Spring Boot; possibility of denial of service attacks

Spring-boot-starter-web-3.1.0.jar: Starter for building web applications; possibility of denial of service attacks

Spring-core-6.0.9.jar: Spring Core; possibility of denial of service attacks

Spring-web-6.0.9.jar: Spring Web; possibility of an open redirect attack or a SSRF attack

Spring-webmvc-6.0.9.jar: Spring Web MVC; possibility of denial of service attacks

Tomcat-embed-core-10.1.8.jar: Core Tomcat implementation; possibility of an information leak

**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.

My mitigation plan includes the following; use prepared statements for database queries to prevent SQL injection, remove any hardcoded credentials and use variables instead, apply appropriate access modifiers, update all vulnerable dependencies to their latest versions.