# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **1/29/2025** | **Francis Lapointe** |  |

## 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

Francis Lapointe

**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?

In this project we are working with Artemis Financial which is consulting company that develops individualized financial plans for its customers from savings, retirement, investments and insurance. With them being a global operation they likely will have international transactions and have to follow government rules and restrictions on secure communications and transactions. Anyone who might want to steal information or money would be a potential threat now and in the immediate future unless you make the application very secure, using open source libraries would potentially leave you open to attack as they are available to anyone and thus someone could find an exploit on their own and use it for their own gain.

**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.

For this project we likely want to focus on multiple areas of the vulnerability assessment process flow diagram, such as Input validation to make sure that the user’s inputs and representations are secure and truthful, APIs and their security as they could change over time, there’s also the client/server side of things which needs to be secure as well as code error/code quality and encapsulation as bad code will leave you even more vulnerable to attack.

**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. Hardcoded database name in plain text in DocData.java could lead to unauthorized database access

2. Account Balance is a public variable meaning it can be directly accessed

3. In Crud.java there’s data that’s stored twice for seemingly no reason in different variables this.content = content, this.content2 = content.

4. There’s no secure communication configured in the RestServiceApplication.java

5. In Customer.java there’s no checks for financial operation meaning anyone could make modifications.

6. In CRUDController.Java there’s no validation on the on the business name parameter

7. Incomplete DateTime Implementation could lead to undefined behavior in myDateTime.java

**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 screenshot of a computer

Description automatically generated

bcprov-jdk15on-1.46.jar: In the Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.

Solution: Update to the latest version of Bouncy Castle JCE

spring-boot-2.2.4.RELEASE.jar: In Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older unsupported versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache.  
Solution: Update to latest version of Springboot

logback-classic-1.2.3.jar: A serialization vulnerability in logback receiver component part of logback version 1.4.11 allows an attacker to mount a Denial-Of-Service attack by sending poisoned data  
  
logback-core-1.2.3.jar: In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.

Solution: Update Logback Core

log4j-api-2.12.1.jar: Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1  
Solution: Update Log4j

snakeyaml-1.25.jar: SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution. We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization

Solution: Upgrading to version 2.0 and beyond.

jackson-databind-2.10.2.jar: has a multitude of flaws which include data integrity, DoS attack openings and resource exhaustion.

Solution: upgrade to the latest version.

**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.

Update Dependencies to their latest versions, add more secure code and implement better database security by addressing the issues stated in the manual review. Finish implementation of various features.