# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[5/31]** | **[Allen Falcon]** |  |

## 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

[Allen Falcon]

**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 needs safe software to protect customer information. They need secure communication to stop hackers and keep trust. If they work with people in other countries, they must follow global rules . They also need to follow U.S. laws about data and encryption.

Hackers might try to steal data, trick users, or break into the system. Artemis must keep everything updated to stop these threats.

They use free tools to build their app, like Spring and Jackson. These tools are helpful but can be risky if not updated(everything can be unsafe if not upadated). ] so they should write safe code, check for problems often, and fix them fast these are based on what I learn but could be missing some.]

**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 most important security areas for Artemis’s app include checking user input to block harmful data, making sure only the right people can access parts of the app, and using strong, updated tools to protect information. Some parts of the code don’t check access properly, and others use old tools for encryption that need to be updated. The code also doesn't always show or record errors, which makes problems harder to find and fix. Lastly, the code uses old software parts that are known to have issues updating is know by everyone helps patch anything bad so these need to be updated.]

**3. Manual Review**

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

[When I looked at the code, I found some problems that could make the app unsafe. In the file DocData.java, the username and password for the database are written right in the code I know they are placeholder, but it can be used before they are changed meaning it can be hacked bc the username ans password is there. So it is risky if someone sees the code. In CRUDController.java, the app takes user input but doesn’t check it ☹, which could let bad data in. There’s also a method called read\_document in DocData.java that isn’t finished, which could cause errors unfinished code = error = uknown error and vulnerability . The same file catches errors but doesn’t show or log them, so we wouldn’t know if something goes wrong. In customer.java, the account\_balance is not protected and might be changed by mistake. In GreetingController.java, users can type anything for their name, and the app doesn’t check it. Lastly, the app has a part called /read that gives data without checking who is asking. These things should be fixed to help keep the app safe.]

**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

[So the scan found 14 unsafe libraries and 159 total problems. One of the biggest issues was with log4j-api-2.12.1, which has dangerous bugs like Log4Shell (CVE-2021-44228). The app also uses old versions of spring-boot-2.2.4.RELEASE and spring-web, which have many problems, including CVE-2022-22965 and CVE-2024-38828. The tool also found issues in jackson-databind-2.10.2, which can lead to hackers running bad code. Other unsafe libraries include bcprov-jdk15on-1.46 (17 CVEs), snakeyaml-1.25 (10 CVEs), and logback-classic-1.2.3. The report says these libraries should be updated to newer versions to help protect the app from known attacks.]

**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 fix the problems, the app should start by updating all the old tools it uses. For example, log4j should be changed to a safer version to stop known bugs. And other tools like spring-boot, jackson-databind, snakeyaml, and logback also need updates. And the code should not have the username and password written inside it. Instead, they should be stored in a safe place where others can’t see them. The app should also check anything a user types in to make sure it’s safe before using it. If there’s a problem, the app should show a helpful error message and keep a record of it. Also, parts of the app that show private data should only be used by people who are allowed to see it. Which I believe this steps will help keep the app and its users safe.