# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/24/25** | **Jordan Landry** |  |

## 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

Jordan Landry

**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 helps individuals with financial planning, including savings, retirement, investments, and insurance. Because the company deals with sensitive information like Social Security numbers and tax documents, secure communication is extremely important.

While it's not confirmed that Artemis only operates in the U.S., it's likely the company handles international transactions. This means they need to follow rules about protecting data, especially trade secrets and private client information.

The biggest external threat is someone trying to steal client data. To prevent this, Artemis must use strong encryption to protect information from outsiders.

It’s also important for Artemis to regularly update its systems. Fixing bugs, patching security issues, and keeping software up to date, especially when using open-source tools, its essential to staying secure as technology evolves.

**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**: Input validation is needed to make sure users are entering correct and safe information. It helps confirm user identity and protects the system from harmful data, like in hacking attempts.
* **Code Quality and Access Control**: Good code allows the system to control what each user can see or do. For example, users should only access their own data, no other users' information or the server.
* **APIs**: Since the application works both inside and outside the company, secure APIs are important. They help control what data can be shared and who can access it.
* **Error Handling**: Handling errors properly helps the company find and fix problems in the system without showing sensitive information to users or attackers.
* **Cryptography**: Encryption is important to protect user data, especially when working with international currencies. It keeps the data safe from hackers, no matter where they are.

**3. Manual Review**

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

As part of the vulnerability assessment, I reviewed both the pom.xml file and the GreetingController class. In the pom.xml, I checked for the presence of the Apache Validator library but did not find it, which could be useful for input validation. In the GreetingController, I observed a lack of input validation, which is a concern and should be addressed in future updates.

The overall code quality was acceptable; however, there was no error handling present, which could lead to issues during failures or unexpected inputs.

When reviewing the API, I noticed several weaknesses. Most importantly, user input was not being handled through a POST method, which can increase the risk of exposing sensitive information. Using a GET method for handling input is unsafe and can lead to data leaks, so this should be corrected to improve security.

Lastly, I checked for signs of cryptography in the code but was unable to find any evidence of its implementation.

**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
* **Bouncy Castle version 1.46 has several vulnerabilities. Recommendation: update to at least 1.60**
* CVE-2013-1624
* CVE-2015-6644
* CVE-2015-7940
* CVE-2016-1000338
* CVE-2016-1000339
* CVE-2016-1000341
* CVE-2016-1000342
* CVE-2016-1000343
* CVE-2016-1000344
* CVE-2016-1000345
* CVE-2016-1000346
* CVE-2016-1000352
* CVE-2017-13098
* CVE-2018-1000613
* CVE-2018-5382
* **FasterXML Jackson Databind 2.10.2 has one vulnerability. Recommendation: update to at least 2.10.5.1**
* CVE-2020-25649
* **Apache Log4j API 2.12.1 has one vulnerability. Recommendation: update to at least 2.13.2**
* CVE-2020-9488
* **SnakeYAML 1.25 has one vulnerability. Recommendation: update to at least 1.26**
* CVE-2017-18640
* **Spring Core 5.2.3 has one vulnerability. Recommendation: update to at least 5.2.8**
* CVE-2020-5421
* **Apache Tomcat 9.0.30 has several vulnerabilities. Recommendation: update to at least 9.0.40**
* CVE-2019-17569
* CVE-2020-11996
* CVE-2020-13934
* CVE-2020-13935
* CVE-2020-13943
* CVE-2020-17527
* CVE-2020-1935
* CVE-2020-1938
* CVE-2020-8022
* CVE-2020-9484
* CVE-2021-24122

**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.

* **Enforce HTTPS**: Ensure all communications between clients and the server use HTTPS to protect data in transit from interception or tampering.
* **Secure Parameter Handling**: Move sensitive request parameters from the URL (URI) to the request headers or body to prevent exposure through browser history, logs, or referrer headers.
* **Remove Hard-Coded Credentials**: Eliminate any hard-coded database connection credentials from the codebase. Instead, use secure environment variables or a secrets management tool.
* **Implement Secure Authentication**: Introduce a secure and modern authentication mechanism (e.g., OAuth 2.0, JWT) to manage user access safely and prevent unauthorized access.
* **Update Dependencies**: Upgrade all outdated or vulnerable libraries and frameworks, including SnakeYAML, Hibernate Validator, Apache Tomcat, and Bouncy Castle, to their latest secure versions.