# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **5/23/2025** | **Brycen McEuen** |  |

## 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

Brycen McEuen

**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?

Considering that Artemis Financial deals with people’s personal finances, security is of the utmost importance in order to keep people's money and investments safe. Secure communication is important to them because they deal with sensitive information such as bank account numbers, social security numbers, transaction details, etc. By securing their communication lines, they prevent theft/fraud, comply with regulations, and ensure trust in their customers. Another thing to consider is that the company has customers from all around the world, which means we’ll have to make sure the application complies with government regulations from many different countries. Speaking of government regulations, there are some governmental restrictions that we must think about as well. For example, many countries have strict encryption policies, data protection/privacy laws, and communication surveillance laws. The application must follow these rules in order for the company to operate internationally.

External threats to think about with a company like Artemis Financial include hackers, insider threats, non-compliance fines/penalties, and fraud/scams. Some future/emerging threats to consider would include things like AI attacks and cloud security risks.

To keep the application as modern and secure as possible, it’s important to consider newer resources such as open-source libraries and evolving web application technologies. Open-source libraries can be very useful because they provide access to a wide range of ready-made frameworks and tools that can help simplify and speed up the development process. Not to mention, they are cost-efficient and have huge communities that offer support. Some newer web application technologies to consider might include progressive web apps, WebAssembly, and AI-driven interfaces. These can help Artemis Financial stay on top of the latest tech trends and help incorporate more engaging services into their products.

**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.

According to the vulnerability assessment process flow diagram, I’ve identified three main areas that I think are very important for this project. The first important area to think about would have to be API interactions. Since this application is a web-based API program, it absolutely must have secure API transmissions in order to fend off hackers and other cyber threats. Poor API security can lead to data breaches, account takeovers, injections, and much more, which can all lead to financial loss and trust issues with customers. Another area to consider is input validation. Poor input validation can lead to injection attacks, buffer overflows, and many other security exploits. Once again, this can lead to financial loss as well as loss of trust. Lastly, I think it's also important to consider the subject of cryptography, which is the practice of securing communication through coding algorithms and encryption. Poor cryptography can lead to sensitive data being exposed and exploited by hackers, once again leading to financial and reputational consequences.

**3. Manual Review**

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

After reviewing the codebase manually, I noticed a few vulnerabilities that should be addressed as soon as possible to ensure secure communications for Artemis Financial.

* No input validation for greeting controller file.
* Not enough error handling
* Poor cryptography
* Not enough inline comments to explain the code's logic
* Code quality was decent
* Lack of authorization
* Lack of access control and permissions

**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

AI-generated content may be incorrect.

**-** Issues with Bouncy Castle cryptography API, recommended to upgrade to the latest version with improved security features.

A screenshot of a computer

AI-generated content may be incorrect.

- This version of Hibernate Validator has issues with expression language injection. Recommended to upgrade to the latest version.

A screenshot of a computer

AI-generated content may be incorrect.

* This version of Jackson Databind has issues with remote code execution. Recommended to upgrade to the latest version.

**A screenshot of a computer

AI-generated content may be incorrect.**

* Issues with remote code execution through JNDI injection. Recommended to upgrade to the latest versions.

**A screenshot of a computer

AI-generated content may be incorrect.**

* Issues with improper deserialization. Recommended to upgrade to the latest version.

**A screenshot of a computer

AI-generated content may be incorrect.**

* Similar to logback-classic vulnerability, it deals with improper deserialization. Recommended to upgrade both logback core and logback classic to latest versions.

**A screenshot of a computer

AI-generated content may be incorrect.**

* **A screenshot of a computer

  AI-generated content may be incorrect.**Issues with code execution and DoS. Recommended to upgrade to the latest version.
* Issues with transitive dependencies. Recommended to upgrade Spring Boot to the latest version.

**A screenshot of a computer

AI-generated content may be incorrect.**

* Starter pulls in outdated libraries and dependencies, which have vulnerability risks. Recommended to upgrade to a more recent version of Spring Boot.

**A screenshot of a computer

AI-generated content may be incorrect.**

* This version has issues with vulnerabilities like injection and insecure deserialization. Recommended to upgrade Spring framework.

**A screenshot of a computer

AI-generated content may be incorrect.**

* Older version of the core module, which can potentially cause issues with injection and deserialization. Recommended to upgrade Spring Framework Core.

**A screenshot of a computer

AI-generated content may be incorrect.**

* Vulnerable to SpEL injection attacks with improper input validation/sanitization. Reccomended to upgrade Spring Expression module.

**A screenshot of a computer

AI-generated content may be incorrect.**

* Outdated version of Web module. Recommended to upgrade to a newer version.

**A screenshot of a computer

AI-generated content may be incorrect.**

**-** Vulnerabilities related to the Web MVC module. Recommended to upgrade the model to a later version.

A screenshot of a computer

AI-generated content may be incorrect.

* Tomcat embedded core is outdated and presents multiple vulnerabilities. Recommended to upgrade to a later version.

**A screenshot of a computer

AI-generated content may be incorrect.**

* Similar to the last one, this version is outdated and presents multiple vulnerabilities. Recommended to upgrade to a later 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.

To fix these vulnerabilities, I’ve created a list of solutions that can help make the program more secure.

* Upgrade to the latest versions for all libraries, frameworks, modules, and dependencies.
* Add input validation and sanitization.
* Add error handling.
* Implement better cryptography methods
* Implement better access control and permissions
* Include more inline comments to explain code logic