# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07/27/2025** | **Dakota Rogers** |  |

## 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

Dakota Rogers

**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 handles important financial data, so secure communication is a big deal. If info isn’t protected, it could lead to data leaks or fraud. The company may have international clients, so security for those transactions matters too.

There could be rules or laws (like GDPR or U.S. data laws) they need to follow. Some current threats might include hackers trying to steal data or break into the system through weak spots.

They also need to update the tech they use. That means being careful with open-source libraries and keeping up with modern web security tools like HTTPS or secure APIs.

**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.

Based on the vulnerability assessment process flow and the nature of Artemis Financial’s web-based software, the following areas of security are relevant:

* Input Validation - Since users will be entering sensitive financial data, it's critical to validate inputs to prevent injection attacks and data corruption.
* APIs - The application uses RESTful APIs to handle client interactions. Ensuring secure API communication (authentication, rate-limiting, data validation) is essential to prevent unauthorized access.
* Cryptography - Artemis deals with private financial information, so encryption must be correctly implemented for data at rest and in transit to prevent leaks or breaches.
* Client/Server - Financial applications require secure communication between clients (e.g., users’ browsers) and the server. This includes session management and data integrity.
* Code Error - Proper error handling is important to avoid exposing sensitive system information to users or attackers when something goes wrong.
* Code Quality - Financial software must follow best practices to ensure reliability, maintainability, and to reduce the risk of introducing security flaws through poor coding patterns.
* Encapsulation - Because the system handles structured financial data, secure encapsulation ensures data structures are well-protected and not exposed unnecessarily.

**3. Manual Review**

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

Here are a few issues I found while looking through the code:

* Hardcoded usernames/passwords in config files.
* No input checks in the login form.
* No HTTPS requirement in the settings.
* Error messages show too much info.
* jQuery version is really old.
* No two-factor authentication.
* Login lets you try as many times as you want.
* Some sensitive info is stored in plain text.

**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

I used the dependency-check tool to scan the code and found some issues in the project’s libraries:

* **CVE-2019-11358** (jQuery) – This version allows cross-site scripting (XSS) in certain functions.  
  **Fix:** Update to the latest version of jQuery.
* **CVE-2020-11023** (jQuery again) – Another XSS issue in the htmlPrefilter method.  
  **Fix:** Upgrade jQuery and check where this method is used.

These vulnerabilities are listed in the National Vulnerability Database and flagged in the report. The tool gave links to recommended updates and showed where these libraries were used.

**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.

Based on my manual and static checks, here are steps Artemis should take:

* Update dependencies, like jQuery, to safer versions.
* Add input validation to all user forms.
* Use HTTPS for all data transmission.
* Set up two-factor authentication for login.
* Hide detailed error messages from users.
* Hash and salt any stored passwords.
* Limit login attempts to prevent brute force attacks.
* Enable logging for failed logins and suspicious activity.

These changes would help close the gaps found in both the code and dependency scan and make the app more secure overall.