# CS 305 Project One Template

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/21/24** | **Britian Holcomb** |  |

## 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

Britian Holcomb

**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 client information both within the company and with external partners, so secure communication across all channels is a key focus. Since they deal with financial transactions from both local and international sources, keeping data safe is crucial to prevent leaks of sensitive customer details. Even though there are no current laws forcing stricter security, Artemis Financial prioritizes strong measures to protect things like financial records, client data and internal business secrets. To stay up to date, the company must regularly update their software to fix bugs and apply the latest security patches.

**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: It's crucial to check user input carefully to avoid errors or attacks like SQL injection, especially when the program takes in text from users.

APIs: Since the app will be used both internally and by end users on web browsers, a reliable API is critical. This API should set clear rules for accessing data and make sure third-party software interacts securely.

Cryptography: Strong encryption is needed to protect customer data during international transfers. Artemis Financial must follow all rules of the country receiving the data.

Error Handling: Good error management is necessary to work alongside input validation and APIs. Handling code errors properly helps stop unauthorized access or misuse of privileges.

Code Quality: High-quality code is key to preventing data leaks and ensuring users only have access to what they're allowed. Well-written code supports secure user interactions, especially when dealing with input and APIs.

**3. Manual Review**

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

While following the vulnerability assessment process, I started by examining input validation. I checked the POM.XML file for any Apache validator and found none. Then, I looked at the greeting controller, which handles user input without validation or any output confirmation.

Next, I reviewed the API setup but found no functional API, even though the Artemis program accesses databases. Data is being passed through the URL instead of using the POST method, which could lead to sensitive information being saved in browser history and possibly exposed. The program accepts raw input through the URL, making it vulnerable to exploitation. Without a clear API, users have no easy way to interact with the program unless they directly access the code. A well-structured RESTful API would allow for safer user interaction.

After input validation and API checks, I reviewed cryptography. Currently, no data encryption is in place. Artemis Financial should add encryption for both storage and international transfers to meet regulatory standards.

For error handling, I noted that while the DocData.java class has try and catch blocks, proper error handling is missing. Finally, even though the overall code quality is strong, the absence of an API makes the program less user-friendly. Also, not validating input and using URL handling instead of POST requests increases the risk of data leaking into browser history, which could lead to security issues.

**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

1. log4j-api-2.12.1.jar: This version doesn’t properly validate certificates, which can make the system vulnerable to man-in-the-middle attacks. Upgrading to the latest version is recommended.

2. tomcat-embed-websocket-9.0.30.jar: An issue with parsing the HTTP request header in this version can allow request smuggling when using a reverse proxy. It’s advised to update to the latest Apache Tomcat version.

3. jackson-databind-2.10.2.jar: The unsecured entity expansion process here can lead to XXE (XML External Entity) attacks, risking data integrity. Updating to the latest version is recommended.

4. hibernate-validator-6.0.18.Final.jar: A bug in the MI processor can let attackers bypass input sanitation by stripping or escaping characters. Updating to the latest version of hibernate-validator is advised.

**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.

All the security risks mentioned can be resolved by simply updating each to the latest version. Additionally, updating the snakeyaml version would help reduce some serious security concerns.