# CS 305 Project One

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **08/20/2024** | **Megan Zurawski** |  |

## 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

Megan Zurawski

**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?

The company, Artemis Financial, deals with sensitive financial data, making secure communication critical to protect client information and maintain trust. If Artemis Financial engages in international transactions, they must comply with global data protection regulations like GDPR, which mandates strict communication security. Additionally, financial institutions often face strict regulations, such as those from the SEC or FINRA, which require secure security measures. Current and future external threats could include phishing, SQL injection, DDoS attacks, and data breaches. As part of their modernization efforts, Artemis Financial needs to integrate the latest security practices, including the use of secure open-source libraries and evolving web application technologies.

**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.

These areas may include Authentication and Authorization, ensuring that only authorized users have access to sensitive information. Data Encryption, to protect data in transit and at rest, and Input Validation, to prevent SQL injections and other forms of attacks. Dependency Management is crucial for ensuring all third-party libraries are up-to-date and free from vulnerabilities. Additionally, focusing on Error Handling to avoid revealing sensitive information to attackers, Logging and Monitoring to detect and respond to threats in real-time, and Configuration Management to secure configurations and prevent unauthorized changes or exposures.

**3. Manual Review**

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

1. Lack of Input Validation: The input in the GreetingController class is not validated, leaving the application vulnerable to injection attacks.
2. No Apache Validator: The POM.xml file does not include any Apache validator, showing a lack of standard input validation.
3. Absence of a Functional API: There is no working API, making the program difficult for end users to interact with in a secure way.
4. Unsafe Data Access via URL: The program accesses data through the URL rather than using the POST method, risking exposure in the browser history.
5. No Data Encryption: The application lacks any form of data encryption, risking the security of stored and transmitted information.
6. Poor Error Handling in DocData.java: The error handling in this class is poor, with only basic try and catch blocks implemented.
7. Unfriendly User Interface: The lack of a functional API and proper input handling makes the program difficult and insecure for users.

**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

**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.

Most of the vulnerabilities that were identified can be mitigated simply by upgrading to the current versions of these programs. This is the main option, other than changing the versions, we could also restrict any aliases to secure any vulnerability.