****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 4](#_Toc32574614)

[5. Mitigation Plan 4](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **3/19/2022** | **Tony Chiesa** |  |

## Client



## Developer

Tony Chiesa

## 1. Interpreting Client Needs

Artemis Financial wants to modernize its processes, and they want to build and apply the most current and effective software security as a critical component of the success of its proprietary software. Artemis Financial has a RESTful web application programming interface (API) and is looking for Global Rain's help in taking precautions against external attacks. Because Artemis Financial is a financial institution, I would anticipate both domestic and foreign transactions. There are no government constraints to consider; nonetheless, maintaining industry-specific security standards is essential since Artemis Financial will need to ensure that its communications, both external and internal, are safe in order to avoid exposing consumer information. Because Artemis Financial will handle financial information as well as private client information such as biometric information, social security numbers, and account information, it is necessary to mask information when storing and transmitting it. In terms of modernization, Artemis Financial must ensure that their libraries are kept up to date inside their application in order to apply the most recent bug patches and security concerns.

## 2. Areas of Security

* Input Validation – Because Artemis Financials' software requires input string validation to avoid potential errors or SQL injection, validation is necessary while gathering user input.
* APIs – A well-developed and documented API will be required because this program will operate not just internally but also externally, like on the end user's web browser. This API should decide which types of data access are permissible since it will define how an end user will interact with the software. Because this program may be used in conjunction with third-party software, a secure API will be required.
* Cryptography – Because international transfers with private customer information will be involved, encryption is required to keep data secure. The data should be protected in a way that complies with both North American regulations and any country-specific regulations, as well as any industry specific standards.
* Code Error – Ensuring the applications code is able to have secure error handling to prevent unauthorized access or privilege violations.
* Code Quality – Secure coding techniques and patterns will make it easier to spot unanticipated problems, reduce unintended data disclosure, and prevent end users from accessing methods that are not permitted by their user level.

## 3. Manual Review

There were a number of bugs detected after reviewing the code base. First, there appears to be no account security, which was discovered as a flaw. Except for the DocData.java class, where a password is required to access a database called test, there is no use of passwords throughout the software. Another fault is that the user and their credentials are not verified while utilizing the software's many functions. We also discovered redundant public CRUD methods in CRUD.java, with both methods writing data to the same variables. Lastly, t he application does not perform any input validation.

## 4. Static Testing

I am still not able to get a successful build to get the Maven dependency report to find the names or vulnerabilities within the code, however, a few of the main issues that I believe will be highlighted within the dependency check report will be unsecure API connections and out of date packages causing security risks.

## 5. Mitigation Plan

After interpreting the results from the manual review and static testing, there are steps to remedy the identified security vulnerabilities for Artemis Financials' software application. We should update all of the dependency libraries, and then run another Maven-dependency check to identify where the issues are and if any new issues have arisen as a result of these modifications. Additionally, establishing validations to keep our keys secure with some form of encryption will create a level of security that browsers will trust in addition to SSL certificates. Lastly, I would create a plan or strategy to handle any future errors that will occur. Having a plan or an update schedule in place will minimize the applications exposure to unnecessary risks.