****

# 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** | **3/23/2024** | **Ally Miller** |  |

## Client



## Instructions

Submit this completed vulnerability assessment report. Replace the bracketed text with the relevant information. In the report, identify your findings of security vulnerabilities and provide recommendations for 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 choose to include images or supporting materials. If you include them, make certain to insert them in all 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

Ally Miller

## Interpreting Client Needs

Artemis Financial has expressed a desire to revitalize their application through both modernization and the integration of the latest, most robust software security measures. Given the company's involvement in handling sensitive financial information, including savings, retirement plans, and insurance, identifying and addressing vulnerabilities in their web-based application is vital. First and foremost, modernizing security protocols can ensure the safeguarding of data such as personal information, financial plans, and investments. This is important for maintaining confidentiality and integrity of client communications, which is a cornerstone for building and retaining consumer trust. For a financial institution like Artemis Financial, consumer trust is critical to success. Moreover, elevating security measures is synonymous with modernization, and will allow Artemis Financial to safely implement more modern practices such as cloud computing or mobile services. Lastly, for companies like Artemis Financial that engage in international transactions, secure communication is vital for protecting cross-border data transfers as well as maintaining compliance with international data protection laws.

Artemis Financial faces several external threats both presently and in the foreseeable future that can be addressed now with more robust security measures. Financial institutions are prime targets for hackers seeking monetary gain, with potential attacks ranging from the theft of customer data to unauthorized fund transfers. The rise of ransomware in recent years poses a significant concern, where malicious software encrypts institutional files, demanding a ransom for restoring services. For a financial planning institution like Artemis Financial, losing access to critical data could severely disrupt services and erode client trust.

In the process of modernizing Artemis Financials’ web-based application, careful attention must be paid to the use of open-source libraries. It’s important to evaluate the security of any open-source library that is used and be sure to regularly update them. Tools such as Maven can aid in detecting and addressing dependency vulnerabilities. Additionally, leveraging web applications such as cloud computing can be a significant benefit at a company like Artemis Financial, but security must be at the forefront with any evolving web technology. Adhering to cloud security best practices such as proper access controls and encrypting sensitive data in transit can mitigate risk. By carefully considering these aspects, Artemis Financial can ensure its web-based application remains secure and adaptable to future needs.

## Areas of Security

* 1. Input validation – Artemis Financial should ensure all user input is validated to prevent malicious input from compromising the application’s security. This helps with injection attacks, where malicious code is injected into input fields to exploit vulnerabilities in the application.
  2. API – Artemis Financial should secure application programming interfaces to prevent unauthorized access or attacks such as API injection or man in the middle attacks.
  3. Cryptography – Artemis Financial should implement secure encryption to protect highly sensitive financial data from unauthorized access or tampering.
  4. Code Error – Artemis Financial should identify and fix code errors that could lead to security vulnerabilities.
  5. Code Quality – Artemis Financial should ensure that quality code practices are followed such as proper code documentation and adherence to secure coding standards.

## Manual Review

* 1. The customer class handles sensitive information such as account numbers and balances but does not use any form of encryption for secure data handling. Recommendation is to implement encryption for sensitive data.
  2. In the DocData class the method read\_document includes code that is commented out. This commented out code snippet hints at direct database access, which if not carefully implemented could lead to SQL injection vulnerabilities. Recommendation is to ensure that all database access follows secure coding practices. Also in this file, is the use of hardcoding credentials on line 27. Recommendation is to store this in a environment variable to avoid unintended access.
  3. The CRUD method in CrudController takes a business name parameter without validating the input. Recommendation is to implement input validation to ensure that only proper input is processed.
  4. Overall, none of the provided code snippets mention the use of HTTPS for securing data in transit. Recommendation is to ensure the application enforces HTTPS to encrypt data in transit.

## Static Testing

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| Dependency | Description of dependency |
|  |  |
| bcprov-jdk15on-1.46.jar | The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7.  Update Version |
| hibernate-validator-6.0.18.Final.jar | Hibernate's Bean Validation (JSR-380) reference implementation.  Update version. |
| jackson-databind-2.10.2.jar | General data-binding functionality for Jackson: works on core streaming API  Update version. |
| log4j-api-2.12.1.jar | The Apache Log4j API  Update version. |
| logback-core-1.2.3.jar | logback-core module  Update version. |
| snakeyaml-1.25.jar | YAML 1.1 parser and emitter for Java  Update Version. |
| spring-boot-2.2.4.RELEASE.jar | Spring Boot  Update version. |
| spring-boot-starter-web-2.2.4.RELEASE.jar | Starter for building web, including RESTful, applications using Spring  MVC. Uses Tomcat as the default embedded container  Update version. |
| spring-core-5.2.3.RELEASE.jar | Spring Core  Update version. |
| spring-web-5.2.3.RELEASE.jar | Spring Web  Update version. |
| spring-webmvc-5.2.3.RELEASE.jar | Spring Web MVC  Update version. |
| tomcat-embed-core-9.0.30.jar | Core Tomcat implementation  Update version. |
| tomcat-embed-websocket-9.0.30.jar | Core Tomcat implementation  Update version. |

## Mitigation Plan

Based on the manual review of the code provided and the static testing report from Maven, there are a few vulnerabilities that I’ve identified for Artemis Financial. Firstly, there is some insecure data handling within the code. Sensitive data like account numbers and balances are handled without encryption. Implementing encryption for sensitive data is recommended. There is also a lack of input validation in some areas, and it is recommended to implement robust input validation for all user inputs through the code. Furthermore, there are hardcoded database credentials in the code provided and this poses a significant security risk. Recommendation is to remove the hardcoded credentials from the code and use environment variables to store sensitive information. Additionally, the Maven report noted several dependency vulnerabilities in the form of outdated libraries. Recommended to update all outdated libraries to their latest and most secure versions and to regularly use dependency check tools to identify and update these security risks.

Action list:

1. Implement secure handing of data.
2. Encrypt all data in transit.
3. Validate all inputs.
4. Ensure safe database access practices.
5. Keep all dependencies up to date.