****

# 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** | **September 18, 2023** | **Jarrid Kamphenkel** | **Initial Draft** |

## 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

Jarrid Kamphenkel

## Interpreting Client Needs

Artemis Financial is a company focused on individualized financial plans, so it's essential to provide robust security measures. The confidential nature of financial data makes it crucial to maintain high standards of confidentiality and integrity. Breaches can lead to significant financial loss, reputational damage, and potential legal repercussions. Clients entrust Artemis with their sensitive financial data, expecting it to be safeguarded against potential threats and breaches. One important step is reliable and safe means of communication. Secure communications are a commitment to clients, who expect transparency in how their data is used, stored, and protected.

Reviewing the Maven dependency check and the customer class, you can see the customer class exposes account numbers, which is a data privacy concern. Proper encapsulation and hiding of such data are necessary.

Artemis Financial's software stack relies heavily on open-source libraries. It's essential to ensure these libraries are updated, and deprecated or insecure libraries are replaced.

With the rapid advancements in web technologies, it's important for Artemis Financial to keep its technology updated. Newer versions of frameworks often come with security patches, performance enhancements, and new features. Not only that, but incorporating robust security measures, such as adding authentication and authorization layers using Spring Security, will significantly enhance the application's security posture.

## Areas of Security

Authentication: The codebase lacks authentication mechanisms, a significant oversight, especially when handling financial data.

Authorization: There are no permission checks, posing risks of unauthorized data access or modifications.

Data Security: Critical attributes like account\_number and account\_balance appear to be in plain text, indicating potential risks to data confidentiality.

Input Validation: The code lacks effective validation of user inputs, particularly visible in the CRUDController class.

Data Integrity: The absence of mechanisms ensuring the accuracy and consistency of stored data over its lifecycle is a concern.

## Manual Review

The Customer class attributes are currently public, making it open to other parts of the project. It would probably be wise to keep these attributes private and write public methods that can query or write data from or to the class, such as getters and setters.

Also, account\_number and account\_balance appear in plain text.

The AtomicLong in GreetingController might expose system activity insights or be manipulated maliciously.

The business\_name input is directly used without validation or sanitation in CRUDController.

Methods retrieveDateTime and setMyDateTime are placeholders, lacking actual functionality in myDateTime. Also, it’s naming convention is seems to be wrong, unless there’s a reason to use camelCase for classes in Java that I’m unaware of.

## Static Testing

1. bcprov-jdk15on-1.46.jar

Vulnerability IDs: cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46

Description: This library from BouncyCastle, commonly used for cryptography, contains known vulnerabilities that can compromise data integrity and confidentiality.

Recommended Solutions: Upgrade to the latest version and restrict the library's use to specific modules or services that are isolated from main operations or sensitive data.

Attribution: Vulnerabilities documented in the BouncyCastle official advisories and CVE database.

2. spring-boot-2.2.4.RELEASE.jar

Vulnerability IDs: cpe:2.3:a:vmware:spring\_boot:2.2.4:release

Description: The Spring Boot version in use is outdated and has critical vulnerabilities that could lead to unauthorized data access or system compromises.

Recommended Solutions: Upgrade to the most recent version of Spring Boot. Test the application thoroughly after upgrading.

Attribution: Vulnerabilities listed in Spring's official advisories.

3. log4j-api-2.12.1.jar

Vulnerability IDs: cpe:2.3:a:apache:log4j:2.12.1

Description: Log4j, a popular logging library, has severe vulnerabilities in the mentioned version, potentially leading to information disclosure or remote code execution.

Recommended Solutions: Reduce the verbosity of logs in production environments and mask sensitive information.

Attribution: Apache's official advisories and CVE database.

4. snakeyaml-1.25.jar

Vulnerability IDs: cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25

Description: SnakeYAML, used for parsing YAML files, has vulnerabilities in the version in use, potentially leading to denial of service or data breaches.

Recommended Solutions: Upgrade to the latest version and Validate YAML input against a known schema before parsing.

Attribution: Official advisories from the SnakeYAML project.

5. tomcat-embed-core-9.0.30.jar

Vulnerability IDs: cpe:2.3:a:apache:tomcat:9.0.30

Description: Apache Tomcat's version in use has critical vulnerabilities that could allow attackers to execute arbitrary code or compromise the server.

Recommended Solutions: Disable any unnecessary features and ensure that the server is running with minimum required permissions.

6. Various Spring Framework JARs

Vulnerability IDs: Multiple IDs related to cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release

Description: Several components of the Spring Framework in the application are outdated and contain vulnerabilities that could compromise the application's security.

Recommended Solutions: Upgrade the entire Spring ecosystem in the application to the latest stable version.

## Mitigation Plan

Incorporate authentication to verify user identity and authorization to make sure that user is allowed to do what they’re trying to, possibly using Spring Security, to restrict and regulate access.

Deploy encryption mechanisms such as MFA or Data Tokens for sensitive data, ensuring security both when transmitted and stored.

Input validation and sanitation are fundamental to ensuring security in applications. Implement comprehensive input validation and sanitation to mitigate risks like SQL injections or execution of malicious payloads. Examples of this would be Whitelists, length checks, malicious string checks, etc.

Review and refine the use of access modifiers throughout the codebase, ensuring appropriate data encapsulation.

Integrate robust logging and monitoring systems to detect and respond to malicious activities or anomalies promptly. Use a reliable logging framework such a N4j or Nlog to log start up and shut down sequences, errors and more.