****

# 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** | **[Date]** | **[Your name]** |  |

## Client



## Developer

ShiAnn Oliver

## Interpreting Client Needs

Artemis Financial is a consulting company which develops individualized financial plans for their customers, including savings, retirement, investments, and insurance. Their security needs should be at the center of their software so that their client's information, such as social security numbers and banking information, is secure. The documentation provided thus far has not identified a need for processing international transactions, but it should be assumed that they will need the ability to do so, although more information should be collected at the next meeting. The threats from bad actors on financial software are abundant and the software created for Artemis Financial should be of the highest security. This can be achieved by encryption at rest of all data as well as limiting access to the financial system by using role-based access control and encryption in all communications that the system makes.

## Areas of Security

* Input Validation: Artemis Financial will need to validate all input to their system to ensure information matches existing databases and is completely accurate. This will protect all existing and new clients.
* APIs: It is of utmost importance that all API interactions are secure between systems and locked down so that only the information intended will be available to outside systems.
* Cryptography: It is important to encrypt data at rest, so it is not accessible if a cyber-attack were to happen on the company. If any vulnerabilities were to exist in the software security, this would make loss of information less likely unless they were able to decrypt the data.
* Code Error: Proper code handling would help developers improve and patch the system to prevent issues and potential further vulnerabilities.
* Code Quality: Secure best practice coding techniques are the best way to prevent vulnerabilities that lead to successful cyber-attacks.

## Manual Review

Upon review of the application, the code has several errors and is of low quality, particularly in DocData.java. There is missing class and object declarations and no references to other files where needed. There is no input validation present, nor error handling in the entire application.

Upon review of the POM, the Spring Framework version referenced is outdated. The Java version being referenced is also not the current version. The Bouncy Castle crypto package being referenced is out of date.

## Static Testing

* bcprov-jdk15on-1.46.jar: Vulnerability in earlier versions of bouncy castle crypto package does not validate encoding of signature on validation. Recommend updating to most current version of 2.3 or higher
* hibernate-validator-6.0.18.Final.jar: A bug in the message interpolation processor enables invalid expressions to be evaluated as if they were valid, allowing attackers to bypass input sanitation controls.
* jackson-databind-2.10.2.jar: Vulnerability found in FasterXML Jackson Databind where it did not have entity expansion secured properly.
* log4j-api-2.12.1.jar: Improper validation of certificate could lead to SMTPS connections to be intercepted with a man-in-the-middle attack.
* logback-core-1.2.3.jar: In this version, an attacker with the required privileges to edit configurations could craft a malicious configuration
* snakeyaml-1.25.jar: Constructor class does not restrict types which can be instantiated during deserialization. Can lead to remote code execution.
* spring-boot-2.2.4.RELEASE.jar: Could be susceptible to a security bypass
* spring-core-5.2.3.RELEASE.jar: May be vulnerable to remote code execution
* spring-webmvc-5.2.3.RELEASE.jar: : Could be susceptible to a security bypass
* tomcat-embed-core-9.0.30.jar: Care must be taken when trusting incoming connections to apache tomcat as they may be exploited

## Mitigation Plan

The most important steps to remedy the vulnerabilities found is to update to the most current version of the systems. The updated version patches the known issues and it is important to continuously check for new updates in all systems you may use.

I would also include validation to boost the security of clients accessing their information and lock down any information that they should not have access to.

The application also needs error handling built into it so that the developers are continuously aware of issues and can improve the system over time