****

# 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** | **25 Sep 2022** | **Jesse Carter** | **Updated all 5 sections of this report** |

## 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

Jesse Carter

## Interpreting Client Needs

Secure communications are especially important given the scope of Artemis Financials work with sensitive financial customer information (e.g. savings, retirement, investments, and insurance). Compliance standards must be met to include Gramm-Leach-Bliley law. International transactions would be expected considering the breadth of scope investments and other financial matters would entail. The European Union is expecting to pass law that further protects sensitive information in this area. Best to build meeting the highest of standards.

## Areas of Security

The areas of security that require the greatest scrutiny:

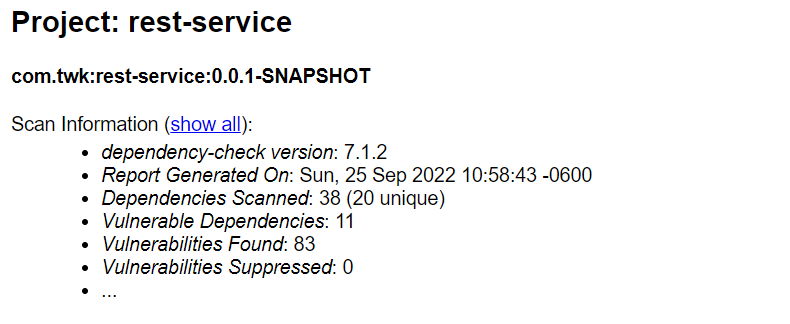
1. Input Validation
2. Secure API Interactions
3. Code Error
4. Encapsulation

## Manual Review

1. Consider using blacklisting and whitelisting for data validation throughout files requiring input
2. Older API frameworks are being utilized. Use current versions.
3. Only DocData.java utilizes error handling. Consider using throughout other files.
4. In the DocData.java, comment declares the root format in a public method. May compromise method if attacker can review and create an instance of this method.

## Static Testing

The summary produced from the dependency test:



Dependencies:

* spring-boot-2.2.4.RELEASE.jar
* spring-core-5.2.3:RELEASE.jar
* spring-web-5.2.3:RELEASE.jar
* tomcat-embed-core-9.0.30.jar
* tomcat-embed-websocket-9.0.30.jar

CRITICAL:

* CVE-2022-22965 under specific conditions, may be vulnerable to RCE via data binding
* CVE-2016-1000027 potential RCE if used for Java deserialization of untrusted data
* CVE-2020-1938 under certain conditions, an attacker my exploit in surprising ways

HIGH:

* CVE-2022-27772 spring-boot prior to v2.2.11 are vulnerable to temporary directory hijacking
* CVE-2021-22118 vulnerable to privilege escalation on specific versions before 5.3.7.
* CVE-2022-29885 potential for DoS risks
* CVE-2021-41079 could be used to trigger a DoS
* CVE-2021-25329 Tomcat version still vulnerable to CVE-2020-9494
* CVE-2021-25122 could duplicate request headers
* CVE-2020-17527 information could leak between requests
* CVE-2020-13935 multiple requests could lead to DoS
* CVE-2020-13934 OutOfMemoryException potential leading to DoS
* CVE-2020-8022 incorrect default permissions vulnerability in packaging of many servers
* CVE-2020-11996 enough requests and server could become unresponsive
* CVE-2020-9484 under specific conditions, an attack could succeed

## Mitigation Plan

The dependencies point to older versions of Spring Framework and Apache Tomcat. Make the following upgrades: Spring Boot 2.2.4 to most current version, 2.7.4; Apache Tomcat 9.0.30 to most current version, 10.0.23. Re-accomplish static testing to determine dependencies after current stable frameworks are utilized. Analyze these dependencies and determine false positives to suppress, as applicable. This will improve review of remaining vulnerabilities to determine most appropriate solutions.