****

# 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** | **[28-01-2024]** | **[Ugochuku Enyi Ebere]** |  |

## 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

[Ugochuku Enyi Ebere]

## Interpreting Client Needs

[Seeing as Artemis Financial deals with individualized financial plans for their customers, they will have access to certain sensitive info, meaning that any security breaches would be devastating. Hence, security is quite important to them. Due to the industry they operate in, they will most definitely have international transactions. Due to this, there will be government regulations on the transactions and communication. As for threats, Artemis uses RESTFUL API, and these are vulnerable to a number of things, such as injection attacks. The system would interact with a database that would hold information of the clients, and can be vulnerable if input parameters are not properly validated. One way to counter such a problem would be the use of open source libraries. Just like we did with the code base, we can continue to scan and monitor them(open source libraries) for any known vulnerabilities by using tools like the OWASP maven dependency check. We can update them as well.]

## Areas of Security

[input validation: the API makes use of user input, having it properly validated will protect against things like SQL injection.

APIs: will allow certain parts of the application to communicate securely with each other

Code quality: high-quality code ensures that there are little to no mistakes, especially exploitable ones.

## Manual Review

[The try-catch section has minimal exception handling in DocData.java. Outside of that, there isn’t much else I can say is wrong with the code. The testing will reveal more.]

## Static Testing

[Dependencies,

* 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. There are several vulnerabilities tied to the bouncy castle

* [**CVE-2016-1000338**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338)
* CVE-2016-1000342
* CVE-2016-1000343
* CVE-2016-1000344
* CVE-2016-1000352
* CVE-2016-1000341
* hibernate-validator-6.0.18.Final.jar: Hibernate’s Bean validation (JSR-380) reference implementation. It contains one published vulnerability. The vulnerability allows attackers to simply walk by input validation/sanitation, which could potentially allow them to carry out injection attacks.
  + CVE-2020-10693
* Jackson-databind-2.10.2.jar: General data-binding functionality for Jackson: works on core streaming API. Has several vulnerabilities. They cause certain problems like DOS attacks, exceptions, resource overuse, and so on. A good fix for some of these might be resource limting, or throttling, improving code quality, and enforcing better input validation. There are several published vulnerabilities.
  + CVE-2020-25649
  + CVE-2020-36518
  + CVE-2021-46877
  + CVE-2022-42003
  + CVE-2022-42004
  + CVE-2023-35116
* log4j-api-2.12.1.jar. This has one vulnerability susceptible to man-in-the-middle attack.
  + CVE-2020-9488
* logback-core-1.2.3.jar. This dependency has 2 published vulnerabilities, that could allow an attacker abuse their privileges.
  + CVE-2023-6378
  + CVE-2021-42550
* There are several other dependencies with numerous vulnerabilities, such as
* tomcat-embed-websocket-9.0.30.jar, which could lead to a DOS, amongst other things
* tomcat-embed-core-9.0.30.jar. A core tomcat implementation, with one known, exploited vulnerability, in which Apache Tomcat treats Apache JServ Protocol (AJP) connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited. To curb such a problem, updates could be applied.
* spring-webmvc-5.2.3.RELEASE.jar. With several vulnerabilities.
  + CVE-2021-22060
  + CVE-2022-22965
* spring-core-5.2.3.RELEASE.jar. One CISA known exploited vulnerability
  + CVE-2022-22965
* 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. The problems can be mitigated with upgrades to at least a 3.0.6+ version if current version is 3.0x
  + CVE-2023-20873
  + CVE-2022-27772
  + CVE-2023-20883.
* snakeyaml-1.25.jar. Deserializing yaml content provided by an attacker can lead to remote code execution
  + CVE-2022-1471
  + CVE-2017-18640.

]

## Mitigation Plan

* [Update the listed dependencies
* Consider removing hard-coded database connection credentials
* Use of proper input validation
* Input sanitization
* Code quality
* Error handling
* Use of open source libraries
* Rate limiting to prevent resource overuse.]