****

# 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** | **07/10/2023** | **William Cocomise** | **Client Needs** |
| **1.1** | **07/11/2023** | **William Cocomise** | **Areas of Security** |
| **1.2** | **07/12/2023** | **William Cocomise** | **Manual Review, Static Testing, Mitigation Plan** |

## 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

William Cocomise

## Interpreting Client Needs

Artemis Financial is totally dependent on secure communications. Whether domestic or abroad, there are government regulations that make secure applications a must. PCI compliance for example is a contractual obligation with many payment processing companies. Loss or theft of confidential customer information brings additional risk of customer litigation and loss of consumer confidence. Artemis financial is a target for individual and state actors alike. The resources dedicated to these attacks will only increase as the business expands. Modernization of the company web API and proprietary software is the main focus of this report. Analyzing the dependencies of these applications and addressing their vulnerabilities is our first consideration. Next we will ensure these applications utilize the most secure versions of existing technologies. Finally, we will recommend upgrading to new technologies where the opportunity exists.

## Areas of Security

Our Vulnerability Process Flow outlines seven areas of security to assess.

* Input Validation: This vector applies because of incoming customer data. Failure to validate leads to attacks such as SQL injection.
* APIs: This vector applies because Artemis uses a RESTful Web API. Securing this API can mitigate data leaks or breaches.
* Cryptography: This vector applies due to regulatory requirements for web traffic.
* Client / Server: This vector applies because of the distributed architecture of business applications.
* Code Error: This vector may apply based on existing error handling practices. The code assessment will determine whether more secure error handling is needed.
* Code Quality: This vector may apply. The code assessment will determine the level of adherence to secure coding principles.
* Encapsulation: This vector may apply. The code assessment will grade existing data structures on their level of security.

## Manual Review

Our Vulnerability Process Flow outlines seven aspects of manual code review.

* Views: This application does not contain views.
* Models: CRUD.java contains two public accessors. Customer.java contains a non-private field and a public accessor. Greeting.java contains two public accessors. MyDateTime.java contains non-private fields and an accessor which should be removed if possible.
* Controllers: CRUDController.java contains one mapping which does specify request type and it ingests an unused request parameter. GreetingController.java contains private fields and other service logic.
* Data Access: DocData.java contains a public default constructor, a public accessor, and hardcoded database connection credentials.
* Services: This application does not contain any service classes.
* Plug-Ins: The pom.xml contains four dependencies (one with an exclusion) and two plug-ins. These all need to be checked for vulnerabilities / updates.
* APIs: The application does not call any external APIs. It does contain two publicly accessible endpoints (noted in ‘Controllers’ above).

## Static Testing

Our static testing utilizes OWASP technology and results from the National Vulnerability Database.

log4j-api-2.12.1.jar: CVE-2020-9488 - Possible man-in-the-middle attacks and data leaks. Issues fixed in later version.

spring-beans-5.2.3.RELEASE.jar - CVE-2022-22965 – Possible remote code execution attack. Application should be run as a spring boot jar file rather than a war file.

spring-web-5.2.3.RELEASE.jar - CVE-2016-1000027 - Possible remote code execution attack. Not to be used for Java deserialization of untrusted data.

tomcat-embed-core-9.0.30.jar - CVE-2019-17569 – Possible HTTP request smuggling. Not to be used behind a reverse proxy that could incorrectly handle invalid transfer-encoding headers.

snakeyaml-1.25.jar - CVE-2017-18640- Possible XML entity expansion. Should employ proper restrictions of recursive entity references.

spring-boot-2.2.4.RELEASE.jar - CVE-2022-27772 – Possible directory hijacking. Issues fixed in later version.

jackson-databind-2.10.2.jar - CVE-2020-25649 – Possible XML external entity attack. Must use proper restrictions of XML external entity references.

bcprov-jdk15on-1.46.jar - CVE-2013-1624 – Possible side-channel and plain-text recovery attacks. Issue fixed in later version.

logback-core-1.2.3.jar - CVE-2021-42550 – Possible remote code execution. Issue fixed in later version.

spring-expression-5.2.3.RELEASE.jar - CVE-2022-22950 – Possible denial of service attack. Issue fixed in later versions.

spring-context-5.2.3.RELEASE.jar - CVE-2022-22968 – Possible unprotected fields related to case sensitivity. Issue fixed in later version.

hibernate-validator-6.0.18.Final.jar - CVE-2020-10693 – Possible input validation failure. Issue fixed in later version.

spring-webmvc-5.2.3.RELEASE.jar - CVE-2021-22060 – Possible output neutralization failure in logs. Issue fixed in later version.

## Mitigation Plan

The final analysis in this report involves laying out action items for the vulnerabilities listed above.

Manual Review

* Makw all object fields private
* Remove object accessors where possible
* Specify request types (Get, Post, Delete, etc)
* Remove unused request parameters
* Remove unused constructors
* Remove hardcoded credentials – create abstract env variables
* Create service classes and move all service level logic to them

Static Testing

* Update all dependencies to secure versions where possible
* Run application as Spring Boot Executable jar rather than war file
* Do not deserialize untrusted data
* Implement code to handle invalid transfer-encoding headers if necessary
* Restrict recursive entity references if necessary
* Restrict xml external entity references if necessary