****

# 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** | **March 17, 2023** | **Scott Dixon** | **Security Vulnerability Test** |

## 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

Scott Dixon

## Interpreting Client Needs

As a security expert and developer with Global Rain, I understand the importance of providing safe and effective software solutions to protect both the customers and the assets of Artemis Financial. By building programs with the latest security measures in mind, we as software developers, can ensure the online banking application is protected against any threats or attacks. In terms of client needs, Artemis Financial requires an elevated level of software security to protect the sensitive financial information of its customers. This will include protection against hacking attempts such as data compromises, and unauthorized access to their systems.

## Areas of Security

* Authentication and Authorization: This involves verifying the identity of users and ensuring that they have the appropriate access privileges to the application and its data.
* Input Validation: This involves ensuring that all user input is validated and sanitized to prevent injection attacks and other vulnerabilities.
* Encryption: This involves implementing strong encryption algorithms to protect sensitive data in transit and at rest.
* Error Handling: This involves implementing effective error-handling mechanisms to prevent information loss and other security issues.
* API Security: This involves securing the RESTful API used by Artemis Financial, including implementing proper access controls, rate limiting, and input validation.
* Logging and Monitoring: This involves implementing logging and monitoring mechanisms to detect and respond to security incidents and unauthorized access attempts.

## Manual Review

From the perspective of a manual review, there exist common security measures that should be implemented to protect an application like Artemis Financials software, for example:

Authentication and Authorization: Implement a robust authentication mechanism to verify user identity to ensure that users only have access to the resources they need. This can be achieved using techniques like two-factor authentication, password hashing and salting, and access controls.

Input Validation: Ensure that all user input is validated and sanitized to prevent injection attacks such as SQL injection, cross-site scripting, and command injection. Use frameworks and libraries that provide input validation and sanitization functions.

Encryption: Implement strong encryption algorithms. AES or RSA to protect sensitive data in transit and at rest. Encrypt data both in storage and while in transit using secure communication protocols such as HTTPS.

Error Handling: Implement effective error-handling mechanisms to prevent information loss and other security issues. Provide error messages that do not disclose sensitive information and ensure that errors are logged and reported to administrators.

API Security: Secure the RESTful API used by Artemis Financial by implementing proper access controls, rate limiting, and input validation. Use authentication and authorization mechanisms to ensure that only authorized users can access the API.

Logging and Monitoring: Implement logging and monitoring mechanisms to detect and respond to security incidents and unauthorized access attempts. Monitor logs and alerts for suspicious activities and take appropriate action in case of a security breach.

By implementing these security measures, Artemis Financials software can better protect against common security threats and minimize the risk of security breaches.

## Static Testing

The 14 files that were isolated thru the static testing process are listed below:

1. tomcat-embed-websocket-9.0.30.jar

This file is a tomcat-embed-websocket is a Java archive file (JAR) that contains the necessary classes and resources for implementing WebSocket functionality in Java web applications using the Apache Tomcat server.

2. tomcat-embed-core-9.0.30.jar

This vulnerable file is a Java archive file (JAR) that contains the core classes for embedding the Apache Tomcat server within a Java application.

3. spring-webmvc-5.2.3.RELEASE.jar

Java archive file (JAR) that contains the classes for building web applications and uses the Spring Framework's Model-View-Controller (MVC) architecture.

4. spring-web-5.2.3.RELEASE.jar

A Java archive file (JAR) that contains the necessary classes and resources for building web applications using the Spring Framework.

5. spring-core-5.2.3.RELEASE.jar

The JAR file that provides the classes for dependency injection.

6. snakeyaml-1.25.jar

A JAR file that allows working with YAML. YAML is the human-readable data serialization language that is often used for writing configuration files. (“What is YAML? - Red Hat”)

7. spring-boot-starter-web-2.2.4.RELEASE.jar

This file is a JAR file that allows for automatic configuration of Spring MVC and Tomcat servers.

8. spring-boot-2.2.4.RELEASE.jar

This dependency contains core classes for the Spring Framework.

9. json-smart-2.3.jar

The JSON JAR file allows reading and writing of JSON files in Java.

10. jackson-databind-2.10.2.jar

The Jackson library provides a set of Java classes and utilities for converting Java objects to and from JSON format.

11. logback-core-1.2.3.jar

The "logback-core" module provides the core functionality of the Logback framework, including the logging API, configuration classes, and utility classes.

12. spring-data-rest-webmvc-3.2.4.RELEASE.jar

This JAR can be used in any Spring-based application that requires RESTful web services functionality.

13. hibernate-validator-6.0.18.Final.jar

The "hibernate-validator" module provides the core functionality of the Hibernate Validator framework.

14. log4j-api-2.12.1.jar

This module provides the API for the Log4j framework, including the logging API, configuration classes, and utility classes.

There were some false-positive results because of the testing.

* For the tomcat-embed-websocket-9.0.30.jar there were no known vulnerabilities.
* The tomcat-embed-core-9.0.30.jar also had no known vulnerabilities.
* The spring-webmvc is listed as having improper output neutralization for logs. I would suggest using an updated version of this file.
* Spring-web-5.2.3.RELEASE.jar The vulnerability relies on using the file incorrectly. "The vendor's position is that untrusted data is not an intended use case." (“[CVE-2016-1000027] CWE-502: Deserialization of Untrusted Data”) I would suggest using the file only for the intended purpose.
* Spring-core-5.2.3.RELEASE.jar has no known vulnerability and can be used without a problem.
* Snakeyaml-1.25.jar This file has critical vulnerabilities that can lead to deserialization of untrusted data. I recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization. (“NVD - CVE-2022-1471”)
* Spring-boot-starter-web-2.2.4.RELEASE.jar has no known vulnerabilities listed.
* The spring-boot-2.2.4.RELEASE.jar has a problem leading to exposure of resources to wrong sphere. I would suggest caution with this JAR file since it was vulnerable to directory hacking.
* The Json-smart JAR file has vulnerabilities and I suggest using the most up to date version available. This vulnerability affected versions 1.3 and 2.4.
* Jackson-databind-2.10.2.jar I would be incredibly careful with this JAR file since vulnerabilities create uncontrolled resource consumption and suggest using an updated file version.
* The logback-core-1.2.3.jar is vulnerable to outside hacking influence since it processes an XML document that can lead the code to embed incorrect documents to its output. I suggest using a different application such as log4j-api.jar, which is a more recent JAR file by Apache.
* For the spring-data-rest-webmvc-3.2.4.RELEASE.jar and hibernate-validator-6.0.18.Final.jar JAR files, I would suggest using the updated versions of both and alternative JAR files.
* log4j-api-2.12.1.jar. This JAR file has no known issues.

## Mitigation Plan

Based on the vulnerabilities identified through the static testing a software threat mitigation plan for Artemis Financials software will include the following:

1. Implement a robust authentication and authorization system to ensure that only authorized users have access to the application and its data.
2. Implement input validation and sanitization techniques to prevent injection attacks such as SQL injection, cross-site scripting, and command injection.
3. Encrypt sensitive data both in storage and while in transit using strong encryption algorithms and secure communication protocols such as HTTPS.
4. Implement effective error-handling mechanisms to prevent information loss and other security issues. Provide error messages that do not disclose sensitive information and ensure that errors are logged and reported to administrators.
5. Secure the RESTful API used by Artemis Financial by implementing proper access controls, rate limiting, and input validation.
6. Implement logging and monitoring mechanisms to detect and respond to security incidents and unauthorized access attempts. Monitor logs and alerts for suspicious activities and take appropriate action in case of a security breach.
7. Regularly update and patch the software and its dependencies to address known security vulnerabilities.
8. Conduct regular security audits and penetration testing to identify and address any potential security issues.
9. Train and educate all employees on the importance of security awareness and best practices, including password hygiene, phishing awareness, and data protection.
10. Have a documented incident response plan that includes steps for containing, analyzing, and mitigating security incidents and restoring normal operations.