****

# 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** | **03/24/23** | **Joel De Alba** |  |

## Client



## Instructions

The Artemis Financial Vulnerability Assessment Report aims to analyze and identify security vulnerabilities present in the web-based software application developed for Artemis Financial. This report serves as a comprehensive examination of the software's security posture, highlighting potential threats and providing recommendations for mitigation. Artemis Financial, a prominent consulting firm specializing in personalized financial planning, has embarked on a transformative journey to modernize its operations. As part of this initiative, Artemis Financial seeks to enhance the security of its web-based software application, which serves as a cornerstone in delivering tailored financial plans to its clientele. Recognizing the critical importance of safeguarding sensitive financial data and ensuring regulatory compliance, Artemis Financial has enlisted the expertise of Global Rain, a leading software engineering company, to conduct a comprehensive vulnerability assessment of its software ecosystem.

Financial information security and integrity are critical in today's ever-changing digital world. The choice made by Artemis Financial to strengthen its software program demonstrates its dedication to upholding the strictest guidelines for confidentiality and data protection. With a broad customer base that includes companies, government organizations, and entrepreneurs worldwide, Artemis Financial is vulnerable to several dangers that could jeopardize the privacy, accuracy, and accessibility of its financial planning services. Considering this, the vulnerability assessment fits with Artemis Financial's broader goal of providing safe, dependable, and cutting-edge financial solutions by acting as a proactive approach to detect and reduce security threats.

**Developer**

Joel De Alba

## Interpreting Client Needs

As we delve into the vulnerability assessment process, the first step entails interpreting Artemis Financial's specific security needs and requirements. When evaluating the usefulness of secure communications to Artemis Financial, it is critical to comprehend the subtleties of the company's business operations, clients, and regulatory environment. Our approach to detecting and resolving any security risks inside the software application will also be informed by factors like foreign transactions, legislative constraints, and changing modernization requirements. We hope to provide useful insights and suggestions that improve the overall security posture of the company's software ecosystem by closely coordinating our assessment with Artemis Financial's strategic goals and security priorities.

## Areas of Security

As we go into Artemis Financial's web-based software program, we focus important security areas that are critical for conducting a complete vulnerability assessment and developing effective risk mitigation plans. Streamlining our focus to the most vital aspects, we find the following essential security areas:

1.**Authentication and Authorization:** Authentication and Authorization: Insufficient validation of user credentials during authentication processes may lead to brute-force attacks or credential stuffing. Additionally, improper authorization checks could allow unauthorized access to sensitive financial data.

2**. Data Encryption:** Inadequate encryption algorithm implementation or poor encryption key management methods might leave sensitive data vulnerable to unwanted access. Furthermore, storing encryption keys alongside encrypted data carries significant danger.

3. **Input Validation and Sanitation:** Lack of input validation and sanitization in user inputs may leave the application vulnerable to SQL injection or cross-site scripting (XSS) attacks. Failure to sanitize user inputs before executing database queries or rendering dynamic content poses a significant risk.

4. **Secure Communication Protocols:** Failure to compel the usage of secure communication protocols such as HTTPS exposes data transported across the network to interception or eavesdropping. Furthermore, misconfiguration of SSL/TLS parameters might reduce encryption strength.

5. **Third-Party Library Security:** Failure to update or patch third-party libraries with known vulnerabilities exposes the program to exploitation by attackers. Furthermore, using obsolete or incompatible libraries may result in a lack of critical security features.

6. **Logging and Monitoring:** Insufficient logging of security events or failure to implement proper log management practices may hinder detection and response to security incidents. Lack of real-time monitoring capabilities leaves the application vulnerable to undetected attacks.

Artemis Financial may strengthen the resilience of its web-based software application against potential threats by addressing these coding weaknesses and implementing robust security measures. This proactive strategy not only protects the confidentiality, integrity, and availability of its financial planning services, but also emphasizes the importance of addressing all potential points of vulnerability to offer comprehensive protection.

## Manual Review

The manual review phase of the Vulnerability Assessment Process Flow Diagram involved a thorough evaluation of the code base to detect potential vulnerabilities. The POM.XML file and the Greeting Controller received special attention. The presence of an Apache Validator was verified in the XML file, but there was no input validation in the Greeting Controller, indicating a potential security gap. Although the code quality was rated adequate, the lack of error-handling capabilities was recognized as a problem. Several flaws were discovered when analyzing the API, including a vulnerability that exposed user input owing to poor implementation of POST methods. Furthermore, attempts to detect cryptography-related functions produced equivocal results.

* **Vulnerability Assessment:**

The **OWASP Dependency-Check tool** identified several dependencies with known vulnerabilities. Notable dependencies include:

1. **CVE-2013-1624:** This vulnerability in *bcprov-jdk15on-1.46.jar* **could lead to a man-in-the-middle (MITM) attack, compromising data confidentiality and integrity.**
2. **CVE-2022-27772:** Vulnerability in *spring-boot-2.2.4.RELEASE.jar* **may allow attackers to bypass access controls, leading to unauthorized access to sensitive financial data.**
3. **CVE-2021-42550:** Vulnerability in *logback-core-1.2.3.jar* **could result in remote code execution (RCE) attacks, compromising the entire application infrastructure.**
4. **CVE-2022-3064:** Vulnerability in *snakeyaml-1.25.jar* **may lead to insecure deserialization, enabling remote code execution and system compromise.**
5. **CVE-2020-25649:** Vulnerability in *jackson-databind-2.10.2.jar* may lead to **deserialization vulnerabilities, allowing attackers to execute arbitrary code on the server.**
6. **CVE-2020-36518:** Vulnerability exposing the application to *XML External Entity (XXE)* **attacks, potentially leading to data theft or system compromise.**
7. **CVE-2021-46877:** Vulnerability in the *Jackson* library may lead to **injection attacks or data breaches, posing a risk of unauthorized access and data leakage.**
8. **CVE-2017-18640:** Vulnerability in *snakeyaml-1.25.jar* **exposes the application to YAML parsing vulnerabilities, risking remote code execution and unauthorized access.**
9. **CVE-2023-6378:** Vulnerability in *logback-core-1.2.3.jar* may result in **information disclosure, potentially allowing access to sensitive information processed by the logging framework.**
10. **CVE-2022-38749:** Vulnerability in *snakeyaml-1.25.jar* **exposes the application to heap-based buffer overflow attacks, leading to potential remote code execution and system compromise.**

Urgent attention is imperative to mitigate the identified vulnerabilities within the application. Immediate steps should be taken to update the affected dependencies to patched versions or explore alternative solutions. Continuous monitoring of dependency vulnerabilities and prompt updates are essential to uphold the application's security posture. Additionally, it's advisable to consider upgrading Dependency-Check to the latest version (9.1.0) to leverage potential improvements and enhancements in vulnerability detection capabilities.

## Static Testing

The following table provides an overview of critical vulnerabilities identified within the dependencies of the project's software application. Each dependency is listed along with a brief description of the vulnerability it poses and the recommended solution to mitigate the risk.

|  |  |  |
| --- | --- | --- |
| **Dependency** | Vulnerability Description | Solution |
| log4j-api-2.12.1.jar | Improper validation of certificate in Apache Log4j SMTP appender. | Upgrade to version 2.13.2. |
| tomcat-embed-core-9.0.30.jar | HTTP transfer-encoding parsing issue in Apache Tomcat. | Upgrade to Apache Tomcat 10.0.6 or later. |
| tomcat-embed-websocket-9.0.30.jar | HTTP transfer-encoding parsing issue in Apache Tomcat. | Upgrade to Apache Tomcat 10.0.6 or later. |
| bcprov-jdk15on-1.46.jar | Unsafe reflection vulnerability in Bouncy Castle Java Cryptography APIs. | Update to version 1.60. |
| spring-boot-2.2.4.RELEASE.jar | Various vulnerabilities in Spring Boot. | Upgrade to the latest version of Spring Boot (2.5.0 or later). |
| snakeyaml-1.25.jar | Several vulnerabilities in SnakeYAML. | Upgrade to the latest version of SnakeYAML (1.27 or later). |
| jackson-databind-2.10.2.jar | Multiple vulnerabilities in Jackson Databind. | Upgrade to the latest version of Jackson Databind (2.13.0 or later). |
| hibernate-validator-6.0.18.Final.jar | Vulnerability in Hibernate Validator. | Upgrade to the latest version of Hibernate Validator (6.1.0 or later). |
| spring-web-5.2.3.RELEASE.jar | Various vulnerabilities in Spring Web. | Upgrade to the latest version of Spring Web (5.3.0 or later). |
| spring-beans-5.2.3.RELEASE.jar | Vulnerability in Spring Beans. | Upgrade to the latest version of Spring Beans (5.3.0 or later). |
| spring-webmvc-5.2.3.RELEASE.jar | Vulnerability in Spring Web MVC. | Upgrade to the latest version of Spring Web MVC (5.3.0 or later). |
| spring-context-5.2.3.RELEASE.jar | Vulnerability in Spring Context. | Upgrade to the latest version of Spring Context (5.3.0 or later). |
| spring-expression-5.2.3.RELEASE.jar | Vulnerability in Spring Expression. | Upgrade to the latest version of Spring Expression (5.3.0 or later). |

## Mitigation Plan

To address the vulnerabilities identified during the assessment process, a comprehensive mitigation strategy is essential. The plan will consist of several key steps aimed at bolstering the security posture of the application. Firstly, immediate action will be taken to prioritize and remediate critical vulnerabilities. This will involve updating affected dependencies to patched versions or exploring alternative solutions where necessary. Additionally, regular monitoring of dependency vulnerabilities will be instituted, with a focus on timely updates to mitigate emerging threats. Proactive measures will be implemented to stay abreast of security advisories and patch releases, ensuring the ongoing resilience of the application against evolving threats.