****

# 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** | **9/17/2023** | **Bradly Van Hoorebeke** |  |

## 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

Bradly Van Hoorebeke

## Interpreting Client Needs

Artemis Financial specializes in crafting personalized financial plans encompassing savings, retirements, investments, and insurance for clients. Given the nature of the sensitive data involved, such as Social Security Numbers (SSNs) and tax information, the company must prioritize robust application and software security measures.

**1. Secure Communications:** Secure communication is paramount for protecting client data, ensuring trust, and safeguarding the company's reputation.

**2. International Transactions:** Given the potential for international transactions, adherence to international data protection standards is crucial for cross-border data security.

**3. Governmental Restrictions:** Artemis Financial must comply with relevant regulations, especially guarding against trade secret exposure.

**4. External Threats:** Protecting against external threats includes guarding against data breaches, phishing attacks, ransomware, and zero-day vulnerabilities through robust security measures.

**5. Modernization Requirements:** To stay competitive and secure, Artemis Financial should focus on careful open-source library use, adapting to evolving web technologies, patch management, and regular penetration testing.

## Areas of Security

**Input Validation:**

* **Applicability:** Relevant
* **Justification:** Input validation is necessary for Artemis Financial to ensure data accuracy and prevent potential data corruption or unauthorized access. By validating user input, especially when handling sensitive client information, the application can protect user data from unintended manipulation or misuse.

**Code Quality:**

* **Applicability:** Relevant
* **Justification:** Code quality is essential for maintaining a secure and efficient application. It enables the application to control user access effectively, ensuring that users can only access their respective information. This safeguards sensitive data and prevents unauthorized access, enhancing overall security.

**APIs (Application Programming Interfaces):**

* **Applicability:** Relevant
* **Justification:** Artemis Financial's software operates both internally and externally, necessitating the use of APIs. APIs allow controlled data access, ensuring that only authorized entities can interact with the application. This control over data access is vital for maintaining data security and enforcing access restrictions.

**Code Error Handling:**

* **Applicability:** Relevant
* **Justification:** Implementing effective error handling is crucial for identifying and addressing potential vulnerabilities within the application. Proper error handling enhances overall security by promptly addressing issues and reducing the risk of exposing user data or the system to security threats.

**Cryptography:**

* **Applicability:** Highly Relevant
* **Justification:** Cryptography plays a crucial role in safeguarding user information, particularly in an international context where various currencies are involved. Implementing cryptographic techniques ensures the confidentiality and integrity of user data, preventing unauthorized access and ensuring secure handling of sensitive financial information.

In summary, these security areas—input validation, code quality, APIs, code error handling, and cryptography—are all relevant and important for Artemis Financial's software application. They collectively contribute to data protection, access control, error prevention, and international data security, which are vital for the company's operations and client trust.

## Manual Review

As part of the Vulnerability Assessment Process Flow Diagram, a manual inspection of the code base was conducted, focusing on the POM.XML file and the Greeting Controller. The objective was to identify vulnerabilities within the code base.

**POM.XML:**

* **Apache Validator:** The assessment aimed to confirm the presence of Apache Validator within the POM.XML file. The analysis indicated that Apache Validator was not explicitly mentioned in the dependencies.

**Greeting Controller:**

* **Input Validation:** Upon review of the Greeting Controller, it was observed that there was a lack of input validation. This absence of input validation is a noteworthy concern that should be addressed to ensure the security and integrity of user-provided data.
* **Code Quality:** The assessment found that the code quality in the Greeting Controller was generally acceptable. However, the absence of input validation could be considered an area for improvement.
* **Error Handling:** It was noted that there was an issue with error handling within the Greeting Controller. Proper error handling is a critical aspect of ensuring the application's robustness and security. The absence of effective error handling could potentially lead to vulnerabilities.

**API:**

* **Breach Risk:** During the examination of the API, it became evident that there was a potential security risk. Specifically, the API lacked appropriate security measures as user input was not properly filtered through a POST method. This oversight could expose user data to security breaches.

**Cryptography:**

* **Verification:** An attempt was made to identify signs of cryptography within the codebase. However, no explicit evidence of cryptographic techniques or libraries was found during the assessment.

In conclusion, the Vulnerability Assessment Process identified several areas of concern within the code base, including the absence of input validation and error handling in the Greeting Controller, potential security risks in the API, and the absence of Apache Validator and explicit cryptography usage. These findings suggest areas for improvement in terms of code security and robustness.

## Static Testing

**Dependency Vulnerabilities:**

* Vulnerabilities:
  + CVE-2013-1624: Timing side-channel attacks
  + CVE-2016-1000341: Timing side-channel attacks
  + CVE-2015-6644 (OSSINDEX): Local data manipulation
  + CVE-2015-7940 (OSSINDEX): Cryptography issues
  + CVE-2016-1000338: Input validation issues
  + CVE-2016-1000342: Input validation issues
  + CVE-2016-1000343: Cryptography issues
  + CVE-2016-1000344: Cryptography issues
  + CVE-2016-1000352: Cryptography issues
* Description: These vulnerabilities in the Bouncy Castle Crypto package pose various security risks, including timing attacks, unauthorized data access, input validation problems, and cryptographic weaknesses.
* Recommended Solutions: Upgrade Bouncy Castle to version 1.5 or higher for Android. Adjust parameter settings and consider more secure cipher modes.

Dependency: log4j-api-2.12.1.jar

* Vulnerability:
  + CVE-2020-9488: Low-severity due to a newer available version.
* Description: This vulnerability in Apache Log4j API relates to the availability of a new version.
* Recommended Solution: Consider upgrading to log4j2.

Dependency: snakeyaml-1.25.jar

* Vulnerability:
  + CVE-2017-18640: High-severity denial of service vulnerability.
* Description: This high-severity vulnerability in SnakeYAML occurs during untrusted data parsing, resulting in a denial of service.
* Recommended Solution: Implement filtering and follow OWASP recommendations.

Dependency: tomcat-embed-core-9.0.30.jar

* Vulnerabilities:
  + CVE-2019-17569: Medium-severity regression issue affecting specific Tomcat versions.
  + CVE-2020-11996: High-severity, potentially causing server overload.
  + CVE-2020-13934: High-severity, potential denial of service from multiple requests.
  + CVE-2020-13935: High-severity, potential denial of service from multiple requests.
  + CVE-2020-1938: Severe, affecting client-server integrity.
  + CVE-2020-8022: Medium-severity due to incorrect default permissions.
* Description: These vulnerabilities affect Apache Tomcat, ranging from medium to severe severity, with impacts like server overload, denial of service, and default permission issues.
* Recommended Solutions: Upgrade Tomcat to versions 9.0.31 or higher, apply relevant patches or hotfixes, and configure connectors securely.

Addressing these vulnerabilities promptly is essential to enhance the security of Artemis Financial's software application. Following the recommended solutions and staying vigilant about future security updates is critical for maintaining a secure system.

## Mitigation Plan

Based on the results of the manual review and static testing, the following steps should be taken to remedy the identified security vulnerabilities for Artemis Financial's software application:

1. **Update Outdated Dependencies:**
   * Most of the vulnerabilities are attributed to outdated plugin versions. It is critical to update these dependencies to their latest secure versions.
   * **Action:** Review and update all outdated dependencies to their latest versions.
2. **Regular Dependency Checks:**
   * Implement a regular process for checking and updating dependencies to ensure that the software remains secure.
   * **Action:** Establish a schedule for periodic dependency checks and updates.
3. **Spring Framework Update:**
   * During the manual review, an outdated version of the Spring Framework was identified. This poses a potential security risk.
   * **Action:** Update the Spring Framework to the latest secure version.
4. **Tomcat Server Update:**
   * The Tomcat server was also identified as needing an update, which was confirmed by static testing, as it exposed some vulnerabilities in the XML file.
   * **Action:** Upgrade the Tomcat server to the latest secure version.
5. **Static Testing Findings:**
   * Utilize the findings from static testing to understand the specific vulnerabilities in the code.
   * **Action:** Review and address each vulnerability as per the recommendations provided.
6. **Security Awareness and Training:**
   * Ensure that developers are aware of security best practices and are trained to identify and address security vulnerabilities during development.
   * **Action:** Conduct security awareness and training sessions for the development team.
7. **Security Audits and Code Reviews:**
   * Incorporate regular security audits and code reviews into the development process to proactively identify and remediate vulnerabilities.
   * **Action:** Establish a process for conducting security audits and code reviews at defined intervals.
8. **Monitoring and Maintenance:**
   * Set up continuous monitoring for security updates and patches for all dependencies.
   * **Action:** Implement a system for ongoing monitoring and maintenance of the software's security posture.
9. **Documentation and Compliance:**
   * Document the actions taken to address vulnerabilities and ensure compliance with security standards.
   * **Action:** Maintain records of security updates and compliance efforts.
10. **Quality Assurance Testing:**
    * Integrate security testing and quality assurance into the development pipeline to catch vulnerabilities early in the development process.
    * **Action:** Implement automated security testing and quality assurance checks.

By following these steps, Artemis Financial can effectively address the identified security vulnerabilities, enhance the security of its software application, and establish a proactive approach to maintaining software security in the future. It's crucial to prioritize security as an integral part of the development process to mitigate risks associated with outdated dependencies and code vulnerabilities.