****

# 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** | **September 14, 2023** | **Cory Doak** |  |

## 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

Cory Doak

## Interpreting Client Needs

Global Rain’s client, Artemis Financial, develops individualized financial plans for their customers including savings, retirement, investments, and insurance. Part of our mission here at Global Rain is that “Security is everyone’s responsibility” therefore ensuring Artemis Financials’ information remains secure is our upmost importance. As a financial institution, I would imagine international transactions are not amongst the uncommon services that are offered. In addition to this, the organization will be working with customer’s confidential information including social security numbers, addresses, account and routing numbers, as well as biometric scans, so their need for information security is a high necessity both for storing and transferring. The company is advancing and wants to remain a success by modernizing their operations and utilizing the most current and effective software security which will include open-source libraries to ensure software security, bug fixes and threat identifiers stay up to date.

## Areas of Security

After reviewing the Vulnerability Assessment Process Flow Diagram (VAPFD), some vulnerabilities identified within the areas of security include input validation, APIs, cryptography, code error and code quality.

**Input Validation**: User input within the system will require validation to avoid potential threats or failures.

**APIs**: Users will be using the service; therefore, Application Programming Interfaces (API) will be necessary as this is how the users will be able to gain access and interact with the system. Ensuring secure API is vital because the interface will have to give access to other software as transactions take place.

**Cryptography**: Ensuring only the user intended to see the information will be a requirement to ensure confidentiality of customer/client personal information. Since we will be handling bank accounts and transfers internationally, this information movement will need to be secure and comply with regulations of the countries or territories involved.

**Code** **Error**: Code error goes hand in hand with API and input validation; Error handling is necessary when dealing with input validation. This will help avoid privilege access violations and ensure there are no unauthorized accessibility incidents.

**Code Quality:** Code quality is a superset of API and Input Validation. End users will be protected as we ensure data exposure does not occur as well as keeping user roles in mind when accessing information.

## Manual Review

Reviewing the code involved reading through each file and understanding the role of the program and how they tied together with the others. Reviewing the POM.xml file for potential errors or Apache validators. Then moving into each program folder checking for validations for those with outputs. Checking for API, I noticed there is URL input, and this could potentially cause data leaks. Utilizing API will allow for the user to easily navigate and execute tasks safely. Checking for cryptography, I was not able to see any type of security here, so this will need to be implemented to keep data safe while being moved and stored. Finally, checking for code error and quality, besides the few missing portions to be implemented as mentioned, the code seemed fine to my novice eye.

## Static Testing

**log4j-api-2.12.1.jar**

Vulnerability: cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*

Description: Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.

Solution: Upgrade to 2.13.2 to support this feature. Previous versions can set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification for SMTPS connections

**tomcat-embed-core-9.0.30.jar**

Vulnerability: cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

Description: Apache Tomcat 10.0.0-M1 to 10.0.6, 9.0.0.M1 to 9.0.46 and 8.5.0 to 8.5.66 did not correctly parse the HTTP transfer-encoding request header in some circumstances leading to the possibility to request smuggling when used with a reverse proxy. Specifically: - Tomcat incorrectly ignored the transfer encoding header if the client declared it would only accept an HTTP/1.0 response; - Tomcat honoured the identify encoding; and - Tomcat did not ensure that, if present, the chunked encoding was the final encoding.

Solution: Upgrade to Apache Tomcat 10.0.6 or later.

**tomcat-embed-websocket-9.0.30.jar**

Vulnerability: cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:apache\_software\_foundation:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

Description: Apache Tomcat 10.0.0-M1 to 10.0.6, 9.0.0.M1 to 9.0.46 and 8.5.0 to 8.5.66 did not correctly parse the HTTP transfer-encoding request header in some circumstances leading to the possibility to request smuggling when used with a reverse proxy. Specifically: - Tomcat incorrectly ignored the transfer encoding header if the client declared it would only accept an HTTP/1.0 response; - Tomcat honored the identify encoding; and - Tomcat did not ensure that, if present, the chunked encoding was the final encoding.

Solution: Upgrade to Apache Tomcat 10.0.6 or later.

**bcprov-jdk15on-1.46.jar**

Vulnerability: cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*

Description: Legion of the Bouncy Castle Legion of the Bouncy Castle Java Cryptography APIs 1.58 up to but not including 1.60 contains a CWE-470: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') vulnerability in XMSS/XMSS^MT private key deserialization that can result in Deserializing an XMSS/XMSS^MT private key can result in the execution of unexpected code. This attack appear to be exploitable via A handcrafted private key can include references to unexpected classes which will be picked up from the class path for the executing application. This vulnerability appears to have been fixed in 1.60 and later.

Solution: update bouncycastle to Version 1.60.

**jackson-databind-2.10.2.jar**

Vulnerability: cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*

cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\*

Description: A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.

Solution: Update to current version.

**spring-aop-5.2.3.RELEASE.jar**

Vulnerability: cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*

cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*

cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  
cpe:2.3:a:vmware:springsource\_spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*

Description: In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter.

Solution: Upgrade to current version.

**hibernate-validator-6.0.18.Final.jar**

Vulnerability: cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*

Description: A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.

Solution: Upgrade to hibernate-validator-6.0.20.

**snakeyaml-1.25.jar**

Vulnerability: cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*

Description: The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

Published: December 11, 2019; 10:15:10 PM -0500

Solution: Migrate to SnakeYAML Engine. It has a configuration option to restrict aliases for collections (the aliases for scalars cannot grow and they are not restricted).

## Mitigation Plan

Updates and changes to versions will remedy the identified security vulnerabilities for Artemis Financials’ software application.