****

# 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** | **[Date]** | **[Your name]** |  |

## 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

Tonya Sanders

## Interpreting Client Needs

Artemis Financial, as a financial institution handling both internal and external priority client information, places great importance on secure communication, whether it's internal or external. Given the nature of its financial services, transactions are expected to originate from both domestic and international sources. While there are currently no specific state or federal mandates regarding secure communications, Artemis Financial is committed to upholding stringent security standards.

The company recognizes the need to secure both external and internal communication channels to prevent any unauthorized disclosure of customer information. Handling financial and proprietary client data, such as biometrics, social security numbers, and account details, necessitates the implementation of data masking for both storage and transmission. Additionally, safeguarding any trade secrets within Artemis Financial is imperative. In line with modernization efforts, Artemis Financial is dedicated to ensuring that their application libraries remain up to date, thereby enabling the prompt implementation of fixes for bugs and security threats.

## Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, we need to identify the relevant security aspects for Artemis Financials' software application and provide justification for their importance in the context of the application.

After examining the security requirements of Artemis Financials, these potential vulnerabilities have been identified as cause for concern:

1. Input Validation:

Justification: In the process of collecting user input, robust input validation is crucial. Given that the software allows user input, implementing input string validation is essential to prevent potential failures or SQL injection attacks.

1. APIs (Application Programming Interfaces):

Justification: Artemis Financials' application is not limited to internal use; it will also be accessible externally, including through end users' web browsers. A well-designed API is essential for defining how end users interact with the program. It should specify which methods for accessing data are permissible. Moreover, as this software may integrate with third-party applications, a secure API is critical to maintain data integrity and safeguard against unauthorized access.

1. Code Error Handling:

Justification: Effective error handling is essential, particularly when dealing with input validation, to prevent unauthorized access or privilege escalation. Properly managing errors is vital for maintaining the software's security and reliability, especially in situations where users may attempt to exploit vulnerabilities.

1. Code Quality:

Justification: High-quality code is imperative when interacting with end users, handling input, and managing APIs. It ensures there is no unintended data exposure and enforces access controls based on user roles. Maintaining code quality helps prevent unauthorized access to methods and data by end users who lack the necessary permissions.

1. Cryptography:

Justification: Given that international transfers with proprietary customer information are part of the application's functionality, cryptography is indispensable. It secures data in a manner that complies with both North American regulations and any regulations applicable in the destination country. Protecting sensitive customer information during international transfers is essential to ensure compliance and data security.

In summary, these identified areas of concern align with Artemis Financials' software application, considering the need to secure user input, manage APIs, handle errors, maintain code quality, and implement cryptography to protect sensitive data during international transfers.

## Manual Review

Following the vulnerability assessment process, I first looked into input validation. I started by checking the pom.xml file to see if there was any Apache validator integration. After that, I checked the greeting controller and found that it didn't seem to use any validation for the input. Unfortunately, I couldn't confirm this lack of validation conclusively as there was no output data to verify it.

Next, I checked for the presence of an Application Programming Interface (API), but I couldn't find one that worked. Surprisingly, even without a secure API, the program could access data without proper security measures. This is concerning because the program accessed data through the URL instead of the more secure POST method, which could potentially lead to data leakage in the browser history and exploitation. Even though the program doesn't visibly show information, it still accepts input through the URL, which can be a potential vulnerability since it allows raw user input. Additionally, the absence of a functional API makes it difficult for end users to understand how to interact with the application unless they look into the source code. A well-designed API, especially one following a RESTful model, should offer a clear way for users to interact with it.

After completing the evaluation of input validation and API, I turned my attention to examining cryptography practices, but unfortunately, I couldn't find any evidence of data encryption protocols. To meet security and regulatory standards, it's advisable for Artemis Financial to implement data encryption measures for both data storage and international transactions, ensuring compliance with international regulations.

Subsequently, I delved into the code to assess error handling mechanisms. I found that the DocData.java class lacked comprehensive error handling, with only try and catch blocks in place. I didn't evaluate other aspects of error handling.

In the final evaluation of the application, I noted that the code quality was good. However, the absence of a functional API made the program less user-friendly and limited its functionality. Moreover, the lack of input validation and the handling of input through the URL, instead of the more secure POST method, posed potential risks of data leakage into the browser history, which requires further attention to enhance security measures.

## Static Testing

|  |  |  |  |
| --- | --- | --- | --- |
| [bcprov-jdk15on-1.46.jar](#RANGE!l2_991c96a4e31e6c19e2b9136c8955bd) | [cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Abouncycastle&cpe_product=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api&cpe_version=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api%3A1.46) | 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. | Update BouncyCastle to version 1.60 |
| [spring-boot-2.2.4.RELEASE.jar](#RANGE!l3_225a4fd31156c254e3bb92adb42ee8) | [cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_boot&cpe_version=cpe%3A%2F%3Avmware%3Aspring_boot%3A2.2.4) | In the Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider. | Users of affected versions should apply the following mitigation: 3.0.x users should upgrade to 3.0.6+. 2.7.x users should upgrade to 2.7.11+. Users of older, unsupported versions should upgrade to 3.0.6+ or 2.7.11+. |
| [logback-core-1.2.3.jar](#RANGE!l4_864344400c3d4d92dfeb0a305dc87d) | [cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aqos&cpe_product=cpe%3A%2F%3Aqos%3Alogback&cpe_version=cpe%3A%2F%3Aqos%3Alogback%3A1.2.3) | In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass. | Update to current version. |
| [log4j-api-2.12.1.jar](#RANGE!l5_a55e6d987f50a515c9260b0451b4fa) | [cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Alog4j&cpe_version=cpe%3A%2F%3Aapache%3Alog4j%3A2.12.1) | 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. | Upgrade to 2.13.2 which supports this feature. Previous versions can set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification for SMTPS connections. |
| [snakeyaml-1.25.jar](#RANGE!l8_8b6e01ef661d8378ae6dd7b511a7f2) | [cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\* , cpe:2.3:a:yaml\_project:yaml:1.25:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Asnakeyaml_project&cpe_product=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml&cpe_version=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml%3A1.25) | The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564. | 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) |
| [jackson-databind-2.10.2.jar](#RANGE!l9_0528de95f198afafbcfb0c09d2e43b) | [cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Afasterxml&cpe_product=cpe%3A%2F%3Afasterxml%3Ajackson-databind&cpe_version=cpe%3A%2F%3Afasterxml%3Ajackson-databind%3A2.10.2) | Published: December 11, 2019; 10:15:10 PM -0500 | Update to current version. |
| [tomcat-embed-core-9.0.30.jar](#RANGE!l13_ad32909314fe2ba02cec036434c0a) | cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*  cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\* | 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. | Update to Apache Tomcat 10.0.6 or later |
| [hibernate-validator-6.0.18.Final.jar](#RANGE!l16_7fd00bcd87e14b6ba66279282ef15) | [cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aredhat&cpe_product=cpe%3A%2F%3Aredhat%3Ahibernate_validator&cpe_version=cpe%3A%2F%3Aredhat%3Ahibernate_validator%3A6.0.18) | 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. | Update to hibernate-validator-6.0.20 |
| [spring-web-5.2.3.RELEASE.jar](#RANGE!l19_dd386a02e40b915ab400a3bf9f586) | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | No CVE Reported | - |
| [spring-beans-5.2.3.RELEASE.jar](#RANGE!l20_0250c8c641433dc06b1b44e4563fa) | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | No CVE Reported | - |
| [spring-webmvc-5.2.3.RELEASE.jar](#RANGE!l21_745a62502023d2496b565b7fe102b) | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | No CVE Reported | - |
| [spring-context-5.2.3.RELEASE.jar](#RANGE!l22_7750c95c96c7a1885c8b1b503ba91) | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | No CVE Reported | - |
| [spring-expression-5.2.3.RELEASE.jar](#RANGE!l23_d0c6bb10758805b2153c589686b80) | cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*  cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\* | No CVE Reported | - |

## Mitigation Plan

A specific measure that can enhance security involves modifying the version of snakeyaml and implementing restrictions on aliases for collections. By making these changes, the system can reinforce its defenses against potential security threats. Restricting aliases for collections is particularly important as it reduces the risk of malicious entities exploiting these aliases to carry out harmful actions, thereby enhancing the overall security posture of the system.

Aside from that, a significant portion of these vulnerabilities can be effectively addressed through the adoption of up-to-date software versions. This includes not only benefiting from the latest features and improvements but also crucially, receiving essential security patches and fixes to counteract known vulnerabilities.