****

# CS 305 Project One

**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** | **03/24/2024** | **Jasmine Bell** |  |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Jasmine Bell

## 1. Interpreting Client Needs

.

Assess your client's requirements and potential threats related to application and software security needs. Evaluate how companies safeguard against external threats based on the given scenario details:

- What is the significance of secure communications for the company?

- Are international transactions conducted by the company?

- Are there any regulatory constraints concerning secure communications?

- What external threats may exist presently and in the near future?

- What are the modernization prerequisites, including the utilization of open-source libraries and the advancement of web application technologies?

Artemis Financial, a financial institution handling sensitive internal and external data for clients, deems secure communication vital. Given its nature, the firm likely engages in transactions domestically and internationally. Although no current state or federal restrictions exist concerning secure communications, Artemis Financial must maintain robust communication security protocols to avoid customer data breaches. Considering the handling of financial and confidential client data such as biometric information, social security numbers, and account details, Artemis Financial must encrypt information during storage and transmission, safeguarding any trade secrets. To modernize, Artemis Financial should keep its libraries updated within applications to incorporate the latest bug fixes and security enhancements.

## 2.Security Areas

## By referring to the Vulnerability Assessment Process Flow Diagram, pinpoint the security domains relevant to Artemis Financials' software application. Justify their importance in the context of the software application.

## Upon evaluating Artemis Financial's security landscape, the following vulnerabilities have been identified:

## - Input Validation: Validation of user input is critical to prevent potential failures or SQL injection attacks since the program allows user input. Proper input validation is essential to maintain data integrity and security.

## - APIs: Given the application's external and internal usage, a robust API is crucial for defining how end users interact with the program and ensuring secure data access methods, especially when integrating with third-party software.

## - Cryptography: With international transfers involving proprietary customer information, robust cryptography is essential to secure data in compliance with both North American and destination country regulations.

## - Code Error Handling: Effective error handling, in coordination with input validation and APIs, is vital to prevent unauthorized access or privilege escalation issues, ensuring system stability and security.

## - Code Quality: High code quality is imperative to mitigate unintentional data exposure risks when interacting with end users, safeguarding against unauthorized access to functionalities based on user permissions.

3. Manual Inspection

Proceed with the Vulnerability Assessment Process Flow Diagram by manually scrutinizing the codebase to identify all vulnerabilities.

Following the vulnerability assessment process, I began by examining input validation. Initially, I checked the POM.XML file for Apache validator settings. Moving on to the greeting controller, it was observed that input validation wasn't implemented. Without any visible output, verification of validation was challenging. Subsequently, while searching for an API, no operational API was discovered; however, data access was still possible without proper security measures. The program accessed data via URL instead of leveraging the POST method, potentially exposing data in browser histories for exploitation. Despite not displaying any output, accepting input via URL could still pose risks due to raw user input handling.

In the absence of an API, end users would face challenges understanding how to interact with the program unless reviewing the code directly. For a RESTful API, clear user interaction guidelines are crucial. Transitioning to cryptography, no data encryption mechanisms were identified. To align with international regulations, Artemis Financial should develop encryption protocols for secure data storage and international transactions. During the review for error handling, it was noted that the DocData.java class lacked comprehensive error handling, featuring only basic try and catch blocks. Finally, while the code quality was commendable, the absence of a functional API rendered the program less user-friendly. Furthermore, the lack of input validation and the handling of input via URL instead of the POST method could lead to data exposure risks and compromise data security.

4.Static Analysis

Conduct a dependency check on the Artemis Financial software application to uncover security weaknesses in the code. Document the results obtained from the dependency check report, detailing the following:

a. List the names or codes of the identified vulnerabilities.

b. Summarize the vulnerabilities and recommended solutions outlined in the dependency check report.

c. Provide any attributions indicating how these vulnerabilities were previously identified or documented, if available.

|  |  |  |  |
| --- | --- | --- | --- |
| Dependency | Vulnerability | Description | Solution |
| log4j-api-2.12.1.jar | |  |  | | --- | --- | |  | cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\* | | 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. |
| tomcat-embed-core-9.0.30.jar | 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:\*:\*:\*:\*:\*:\*:\* | 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. | - Upgrade to Apache Tomcat 10.0.6 or later |
| tomcat-embed-websocket-9.0.30.jar | 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:\*:\*:\*:\*:\*:\*:\* | 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. | - Upgrade to Apache Tomcat 10.0.6 or later |
| bcprov-jdk15on-1.46.jar | 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:\*:\*:\*:\*:\*:\*:\* | 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 update to 1.60 |
| jackson-databind-2.10.2.jar | |  |  | | --- | --- | |  | [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) cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\* | | 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. | Update to current version |
| spring-aop-5.2.3.RELEASE.jar | 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:\*:\*:\*:\*:\*:\* | 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. | Upgrade to current version |
| hibernate-validator-6.0.18.Final.jar | cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\* | 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. | Upgrade to hibernate-validator-6.0.20 |
| snakeyaml-1.25.jar | cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\* | 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 | 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) |

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application.

Many of these vulnerabilities can be mitigated by upgrading to current versions. Aside from that Changing the version of Snakeyaml and restricting aliases for collection can secure this vulnerability.