****

# 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]** | **Kelvin Pichardo Doughty** |  |

## Client



Developer

Kelvin Pichardo Doughty

## Interpreting Client Needs

Secure Communication:

Ensuring the security of communication is vital, especially for software handling personal information. Even for software that doesn't involve personal data, secure communication is necessary to protect against potential attacks. In the case of Artemis Financial, a company dealing with personalized financial plans and insurance, the need for secure communication becomes even more urgent. Gathering sensitive information such as salary and investment portfolios requires a high level of data protection to gain trust and keep the data away from unauthorized access.

International Transactions:

After reviewing the provided documents and existing code base from Artemis Financial, there is no explicit indication of international transactions. However, it is essential to treat this as a possibility and implement communication security measures, regardless of the absence of explicit mention. By following security principles, Artemis Financial should establish effective communication security protocols for both domestic and foreign transactions.

Governmental Restrictions:

Currently, in the United States, there are no specific regulations regarding the encryption or security of communications that require evaluation. However, it is crucial to plan ahead considering the regular introduction of bills related to this matter. One recent proposal, the Lawful Access to Encrypted Data Act, was presented to the US Senate on June 23, 2020. In the event of such a bill being passed, it would be necessary to design a system that easily incorporates a compliance module, if required. Additionally, if Artemis Financial engages in international business, it is important to acknowledge that communications may need to be accessible by law enforcement agencies in countries where it is mandated, as several countries including Australia, Canada, the UK, and France have such requirements.

External Threats:

In today's society, external threats are prevalent, and when dealing with sensitive financial and personal information, the risks of attacks are significantly heightened. While it is unlikely that government or high-level actors would target Artemis Financial, the company should remain vigilant against individual or small group attacks. These threats can arise from various sources, such as invalid inputs, SQL injection, data leakage in browser history, dependency vulnerabilities, and malicious bots attempting to gain unauthorized access. Additionally, authorized users accessing data beyond their privileges, potentially through privilege escalation, pose a significant risk. Implementing a robust API is crucial to defend against potential attacks.

Modernization Requirements:

During the process of modernization, it is important to consider the potential risks associated with open-source libraries and even actively maintained software. In the context of financial planning, integrating compiled stock/bond information from external sources may introduce additional threats to the system. The utilization of other software to enhance the program could also bring about additional risks if their API and underlying structures lack proper security measures. Furthermore, evolving web application technologies introduce new possibilities but also expose the application to security vulnerabilities when interacting with other applications. It is important to prioritize fail-safe measures in the face of evolving attacker strategies.

## Areas of Security

Primary Security Areas:

* Input Validation:

Effective input validation is essential whenever dealing with user input. In this program, there are instances where input is accepted, such as line 12 of CRUDController.java, which allows input in the form of a string. Proper validation of such input is crucial to prevent failures and mitigate the risks of SQL injection attacks. Input validation plays a critical role because it involves user interaction with the system, and any malicious intent could potentially cause significant damage or unauthorized access to sensitive data.

* APIs:

Since this application is intended to be used in an external environment, such as a web browser, a well-designed API is of utmost importance. The API defines the interactions between our program and third-party software, specifying acceptable methods and data. Additionally, if the project relies on third-party software, it is crucial to have a secure API to mitigate the risks associated with using external dependencies.

* Cryptography:

The inclusion of cryptography in this list arises from the potential for international transfers. Considering the regulatory aspects surrounding cryptography as an export item, secure and encrypted communication becomes important, particularly if Artemis Financial conducts business overseas. It is necessary to secure the data and ensure compliance with both US and destination country laws.

* Code Error Handling:

Code error handling is an important aspect that goes hand-in-hand with input validation. Proper handling of errors caused by input is crucial. Effective error handling is essential to prevent privilege escalation and address other vulnerabilities in the system.

* Code Quality:

Code quality is of utmost concern when APIs and input validation are involved. High-quality code ensures that no unintentional data exposure occurs and restricts access to unauthorized methods for end users. In short, by employing quality code, only authorized users gain access to appropriate data and methods aligned with their level of authorization.

## Manual Review

* After conducting a manual review of the code base for the Artemis Financial project, I have made several observations. Firstly, I examined the presence of input validation. My initial focus was on the pom.xml file, as it could potentially contain Apache validators. Subsequently, I inspected areas where user input is received. One such area was the greeting controller, where input is assumed to be a string without any validation. While I was able to input various values into the program, the absence of output (addressed under API) made it difficult to verify their processing. Additionally, another section that requests information also lacks built-in validators and simply expects a string input.
* Then, I proceeded to detect the presence of an API and evaluate its functionality. Surprisingly, I found no evidence of an API whatsoever. Despite the absence of a visual or functional API, the program can accept data. However, it does so in an entirely insecure manner which presents future and current breaches of the data. Data is received via the URL instead of utilizing the more secure POST method, which raises concerns about potential exploits and browser history leakage. Although entering text into the URL did not display anything and resulted in errors on every page, the program remains vulnerable due to its handling of raw user input. The program lacks a discernible means of interaction unless one examines the source code. If this were to be published, there would need to be clear documentation or a portal explaining how to effectively interact with it. To be considered a RESTful API, it is essential for an API to have an easily discernible method of interaction for users.
* In consideration of potential international communications, I also assessed the presence of any form of cryptography. Upon a quick review, I found no implementation of encryption whatsoever. If Artemis Financial requires international communications, they would need to develop an encryption system that adheres to both United States and international regulations and laws.
* Furthermore, I examined code error handling, specifically within the DocData.java class. Although this class includes a method for reading a document with a specified key and value, it lacks comprehensive error handling. There is a try/catch block present, but no actual error handling logic. No other classes demonstrated notable error handling practices.
* Lastly, I evaluated the code quality of the existing implementation. While the code appears to be adequate, it is not entirely functional and requires substantial expansion to achieve full readiness. As mentioned earlier, the API does not provide any guidance to end users regarding expected interactions. Implementing input validation is crucial, and data must be securely transferred from the URL to the POST method to prevent leakage into browser history. Code quality encompasses both the API and input validation. The challenge lies in coding securely within these subsets and ensuring the program's overall reliability.

## Static Testing

The dependencies and vulnerability ID's listed below are sourced from the provided code base:

* bcprov-jdk15on-1.46.jar: The Bouncy Castle Crypto package is used for cryptographic algorithms, but this version (1.46) has a vulnerability that could lead to the use of invalid keys, exposing sensitive data and compromising secure communications.
* jackson-databind-2.10.2.jar: This dependency provides data-binding functionality for Jackson, but it has a vulnerability that allows an attacker to submit XML files outside the intended scope, potentially exposing sensitive data through error messages1.
* log4j-api-2.12.1.jar: The Apache Log4j API is used for logging, but this version (2.12.1) has a vulnerability that can impact SSL security by improperly validating certificates, potentially allowing connections to malicious hosts.
* snakeyaml-1.25.jar: SnakeYAML is a YAML parser for Java, but this version (1.25) has a vulnerability that can lead to denial of service (DoS) attacks due to uncontrolled recursive definitions, causing excessive resource consumption.
* spring-core-5.2.3.RELEASE.jar: Spring Core is a fundamental component of the Spring Framework, but versions up to 5.2.8 have a vulnerability that allows bypassing protections against reflected file download (RFD) attacks using the jssessionid path.
* tomcat-embed-core-9.0.30.jar: This is the core implementation of Apache Tomcat, and it has known vulnerabilities, including the potential for unauthorized access to sensitive information if the code inserts or manages resources containing sensitive data.
* tomcat-embed-websocket-9.0.30.jar: Similar to the previous dependency, this is also a core implementation of Apache Tomcat with the same vulnerabilities.

## Mitigation Plan

After analyzing the outcomes of the manual review and static testing, it is essential to implement the following steps to address the identified security vulnerabilities in Artemis Financial's software application.

To begin with, the overall mitigation plan revolves around transforming the existing DevOps pipeline into a DevSecOps pipeline. By instilling a security mindset within the development team early on, we can ensure that security is prioritized throughout the project. This proactive approach will enable us to address concerns like input validation, which has been a recurring issue flagged by NIST, before the application becomes operational.

In addition to the general mitigation plan, specific measures must be taken to address the identified threats. The following list outlines the required actions:

bcprov-jdk15on-1.46.jar: Mitigating this vulnerability involves upgrading to version 1.56 or later, as these versions include key parameter validation during agreement calculation. Properly calculating and validating keys is crucial for effective cryptography and ensuring the secure exchange of data.

jackson-databind-2.10.2.jar: To mitigate this vulnerability, the XML parser and validator should be configured to disable external entity expansion. By doing so, attackers will no longer be able to exploit altered DTD files within the XML documents, as the parser will detect and ignore any malicious code.

log4j-api-2.12.1.jar: Mitigating this vulnerability requires refraining from using certificate pinning or ensuring that all certificate properties, especially the hostname, are thoroughly validated before pinning. It is vital to enforce SSL certificate validation correctly to maintain the trust and security provided by the SSL protocol.

snakeyaml-1.25.jar: To address this vulnerability, it is necessary to disallow the use of DTDs or employ a parser that restricts the expansion of recursive definitions. By limiting recursive calls, the risk of DoS attacks resulting from excessive recursion can be significantly reduced.

spring-core-5.2.3.RELEASE.jar: While no specific mitigation measures are mentioned for this vulnerability, upgrading to the latest version, 5.3.1, can be considered as a mitigating measure. Keeping software up to date ensures that the latest patches for known vulnerabilities are applied, although it is important to balance this with the maturity and real-world exposure of the updated version.

tomcat-embed-core-9.0.30.jar & tomcat-embed-websocket-9.0.30.jar: Mitigating these vulnerabilities involves implementing the principle of least privilege and establishing secure zones within the system. By confining sensitive information within secure structures with proper authorization mechanisms, the risk of data leakage is significantly minimized.

By following these specific mitigation steps and adopting a DevSecOps approach, Artemis Financial can effectively address the identified security vulnerabilities and enhance the overall security posture of their software application.