****

# 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** | **Sept. 16, 2023** | **Morgan Getkin** |  |

## Client



## Developer

Morgan Getkin

## Interpreting Client Needs

* Secure communications are of utmost importance to protect sensitive financial data during transmission. This includes the use of encryption protocols like HTTPS and ensuring the integrity and confidentiality of data.
* Artemis Financial may engage in international transactions, so it’s crucial to comply with international data protection regulations and consider cross-border data transfer risks.
* Depending on the regions in which Artemis Financial operates, there may be governmental restrictions and compliance requirements related to secure communications and data protection.
* External threats may include cyberattacks, such as DDoS attacks, SQL injection, cross-site scripting (XSS), and credential stuffing. These threats can compromise the application’s security.
* The use of open-source libraries can accelerate development but may introduce security vulnerabilities. Careful selection and continuous monitoring of these libraries are essential.
* Additionally, keeping up with modern web application technologies is critical to ensure the application is resilient against emerging threats. This includes regular updates, patches, and adherence to best practices.

## Areas of Security

* Input validation is crucial to ensure data entered by users or received through the API is sanitized and safe to process. In a financial context, ensuring the integrity and accuracy of data is paramount to prevent manipulation or fraud.
* Artemis Financial relies on a RESTful API, making API security critical. Secure API interactions are necessary to protect data transmitted between clients and the server. This includes authentication, authorization, and data encryption to prevent unauthorized access or exposure.
* Given the sensitivity of financial data, encryption is essential to protect data at rest and in transit. Cryptography helps safeguard against data breaches and eavesdropping during data transmission.
* Secure code handling is crucial to prevent vulnerabilities such as buffer overflows, injection attacks (e.g., SQL injection), and other code-related security flaws. Detecting and handling errors securely ensures system reliability.
* Secure coding practices and patterns are essential to minimize the introduction of vulnerabilities during software development. Proper coding practices reduce the risk of common programming errors that could lead to security issues.
* Secure data structures are important for maintaining data integrity and preventing unauthorized access or modification of financial data. Proper encapsulation ensures data is accessed and modified only through authorized methods.

## Manual Review

* The “CRUDController” code doesn’t appear to have direct security vulnerabilities. It does, however, interact with the “DocData” class, which could have potential database-related security issues. The code doesn’t handle exceptions from database connections properly.
* The “customer” class doesn’t directly involve security issues, but since it represents a customer account, securing customer data and transactions should be a concern.
* The “DocData” code establishes a database connection without proper error handling or security measures. It uses hard-coded credentials (“root” as both username and password), which is insecure. It lacks proper error handling for database connection failures.
* The “GreetingController” doesn’t introduce security vulnerabilities directly. However, it may be exposed to potential attacks if not properly secured.
* The “myDateTime” class doesn’t seem to have direct security vulnerabilities, but since it is used for time-related operations, improper handling of dates and times might lead to security issues.
* The “RestServiceApplication” class is the entry point of the application. While it doesn’t introduce security vulnerabilities directly, it’s important to ensure proper security configurations and access controls for the entire application.

## Static Testing

The dependency check yielded the following results:

A white background with black dots

Description automatically generated

A screenshot of a computer

Description automatically generated

Vulnerable dependencies are as follows:

1. bcprov-jdk15on-1.46.jar – The Bouncy Castle Crypto package
   * Vulnerability ID: CVE-2016-1000338, CVE-2016-1000339, CVE-2016-1000341, CVE-2016-1000342, CVE-2016-1000343, CVE-2016-1000345, CVE-2016-1000346
   * Description: Improper Verification of Cryptographic Signature, does not fully validate ASN.1 encoding of signature on verification, signature generation is vulnerable to timing attack.
   * Recommended rectification: Upgrade Bouncy Castle packages.
2. hibernate-validator-6.0.18.Final.jar – Hibernate’s Bean Validation
   * Vulnerability ID: CVE-2020-10693
   * Description: Invalid EL expressions can be evaluated as if they were valid, bypassing input sanitation controls.
   * Recommended rectification: Update Spring starter to 3.1.3.
3. jackson-databind-2.10.2.jar – General data-binding functionality for Jackson
   * Vulnerability ID: CVE-2023-35116

Description: jackson before v2.15.2 was discovered to contain a stack overflow via the map parameter.

Recommended rectification: Add the depth variable to record the current parsing depth. If the parsing depth exceeds a certain threshold, an exception is thrown. (fcfc499) Change the recursive processing on deeply nested arrays or JSON objects to stack+iteration processing.

1. log4j-api-2.12.1.jar – The Apache Log4j API
   * Vulnerability ID: CVE-2020-9488
   * Description: Improper validation of certificate with host mismatch in Apache Log4j SMTP appender.
   * Recommended rectification: Upgrade to Apache Log4j 2.12.3 or 2.13.1.
2. logback-core-1.2.3.jar – logback-core module
   * Vulnerability ID: CVE-2021-42550
   * Description: In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.
   * Recommended rectification: Upgrade logback packages.
3. snakeyaml-1.25.jar – YAML 1.1 parser and emitter for Java
   * Vulnerability ID: CVE-2022-1471

Description: SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution.

Recommended rectification: Using SnakeYaml's SafeConstructor when parsing untrusted content to restrict deserialization and upgrading to version 2.0 and beyond.

1. spring-boot-2.2.4.RELEASE.jar – Spring Boot
   * Vulnerability ID: CVE-2023-20873, CVE-2022-27772, CVE-2023-20883
   * Description: 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, spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking, In Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older unsupported versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache.
   * Recommended rectification: 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+.
2. spring-boot-starter-web-2.2.4.RELEASE.jar – Starter for building web applications using Spring MVC, including RESTful
   * Vulnerability ID: CVE-2023-20873, CVE-2022-27772, CVE-2023-20883
   * Description: 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, spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking, In Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older unsupported versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache.
   * Recommended rectification: 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+.
3. spring-core-5.2.3.RELEASE.jar – Spring Core
   * Vulnerability ID: CVE-2022-22965, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096
   * Description: A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding, In Spring Framework, versions 5.2.x prior to 5.2.15 and versions 5.3.x prior to 5.3.7, a WebFlux application is vulnerable to a privilege escalation, In Spring Framework versions 6.0.0 - 6.0.6, 5.3.0 - 5.3.25, 5.2.0.RELEASE - 5.2.22.RELEASE, and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition, In Spring Framework versions 5.3.0 - 5.3.10, 5.2.0 - 5.2.17, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries.
   * Recommended rectification: Upgrade Spring framework packages
4. spring-web-5.2.3.RELEASE.jar – Spring Web
   * Vulnerability ID: CVE-2022-22965, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096
   * Description: A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding, In Spring Framework, versions 5.2.x prior to 5.2.15 and versions 5.3.x prior to 5.3.7, a WebFlux application is vulnerable to a privilege escalation, In Spring Framework versions 6.0.0 - 6.0.6, 5.3.0 - 5.3.25, 5.2.0.RELEASE - 5.2.22.RELEASE, and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition, In Spring Framework versions 5.3.0 - 5.3.10, 5.2.0 - 5.2.17, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries.
   * Recommended rectification: Upgrade Spring framework packages
5. spring-webmvc-5.2.3.RELEASE.jar – Spring Web MVC
   * Vulnerability ID: CVE-2022-22965, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2023-20861, CVE-2023-20863, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096
   * Description: A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding, In Spring Framework, versions 5.2.x prior to 5.2.15 and versions 5.3.x prior to 5.3.7, a WebFlux application is vulnerable to a privilege escalation, In Spring Framework versions 6.0.0 - 6.0.6, 5.3.0 - 5.3.25, 5.2.0.RELEASE - 5.2.22.RELEASE, and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition, In Spring Framework versions 5.3.0 - 5.3.10, 5.2.0 - 5.2.17, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries.
   * Recommended rectification: Upgrade Spring framework packages
6. tomcat-embed-core-9.0.30.jar – Core Tomcat implementation
   * Vulnerability ID: CVE-2023-41080
   * Description: URL Redirection to Untrusted Site ('Open Redirect')
   * Recommended rectification: Upgrade to one of the following or later – Apache Tomcat 11.0.0-M11, 10.1.13, 9.0.80, 8.5.93
7. tomcat-embed-websocket-9.0.30.jar – Core Tomcat implementation
   * Vulnerability ID: CVE-2020-1938
   * Description: When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising.
   * Recommended rectification: Upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later.

## Mitigation Plan

* There is potential vulnerability in the “CRUDController” and “DocData” files. The “CRUDController” class establishes a database connection without proper exception handling and resource management. Mitigation steps include:
  + Implementing proper exception handling and resource management using try-catch-finally blocks to ensure that database connections are properly closed.
  + Utilizing connection pooling to improve resource utilization and connection management.
* The “customer” class lacks proper access control modifiers, potentially exposing sensitive data. Mitigation steps include:
  + Use of private access modifiers for class members to encapsulate data and prevent unauthorized access.
  + Implementation of proper getters and setters with access control to ensure data integrity.
* The “myDateTime” class exposes fields directly without encapsulation. Mitigation steps include:
  + Encapsulating class fields by making them private and providing public getter and setter methods.
  + Ensuring that setter methods validate and sanitize input data to prevent potential security issues.
* The “GreetingController” class uses user-provided input without proper input validation or sanitation. Mitigation steps include:
  + Implementing input validation to sanitize user-provided data to prevent potential injection attacks.
  + Use of parameterized queries or prepared statements when interacting with databases to prevent SQL injection.
* The “CRUD” class lacks validation of input data, potentially leading to security vulnerabilities. Mitigation steps include:
  + Implementing input validation and sanitizing user-provided data to prevent potential security issues.
  + Validating and sanitizing data before processing it in any way to mitigate vulnerabilities.
* The “DocData” class establishes a database connection without proper exception handling and resource management. Mitigation steps include:
  + Implementing proper exception handling and resource management using try-catch-finally blocks to ensure that database connections are properly closed.
  + Utilizing connection pooling to improve resource utilization and connection management.
* Despite this long list of vulnerabilities, many of the dependency vulnerabilities can be mitigated by using an updated version of the spring-boot-starter-parent artifact. The version is explicitly stated in the POM file as 2.2.4.RELEASE. This version of the starter has three known vulnerabilities itself, and when updated automatically uses updated versions of its own dependencies. This would eliminate 97 of the 118 vulnerabilities found in 9 of the 13 vulnerable dependencies, leaving only 4 remaining vulnerable dependencies and 21 known vulnerabilities:
  + bcprov-jdk15on-1.46.jar
  + jackson-databind-2.15.2.jar
  + snakeyaml-1.33.jar
  + tomcat-embed-core-10.1.12.jar
* Additionally, SnakeYaml can be upgraded to version 2.0 explicitly in the program’s POM file.
* Bouncy Castle and Apache Tomcat can be upgraded to appropriate versions as stated above.
* Jackson-databind has some specific use-case rectifications as stated above, preventing a stack overflow and therefore DOS.
* It is essential to thoroughly review and test these changes to ensure the security of the software. Additionally, applying secure coding practices and conducting regular code reviews help prevent future vulnerabilities.