****

# 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** | **9/16/2023** | **Daniel Escobedo** | **Dependency check & updated client needs** |

## 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

Daniel Escobedo

## Interpreting Client Needs

Artemis Financial is a consulting company that develops individualized financial plans for their customers. The company is seeking Global Rain's expertise in how to protect the organization from external threats.

Secure communications are essential for Artemis Financial to protect the confidentiality and integrity of their customers' financial information. If a hacker were to intercept or tamper with this information, it could have serious consequences for Artemis Financials’ customers.

Artemis Financial does not make any international transactions. However, their customers may be in different countries. Therefore, it is important for Artemis Financial to implement security measures that will protect their customers' information regardless of where they are located.

There may be governmental restrictions on secure communications in the countries where Artemis Financials’ customers are located. Artemis Financial should consult with an attorney to ensure that they are complying with all applicable laws and regulations.

Artemis Financial is at risk of a variety of external threats such as cyberattacks, data breaches, and compliance violations.

The company is seeking to modernize their operations. This includes using the most current and effective software security and its requirements such open-source libraries since they can be a great way to save time and money when developing software. Evolving web application technologies can also be used since web application technologies are constantly evolving. Artemis Financial needs to make sure that their web application is using the latest and most secure technologies.

## Areas of Security

* Input validation: Artemis Financial needs to validate all user input to prevent malicious attacks.
* APIs: Artemis Financial needs to secure their APIs to prevent unauthorized access and use.
* Cryptography: Artemis Financial needs to use strong cryptography to protect sensitive data.
* Client / Server: Artemis Financial needs to implement security measures to protect communication between the client and server.
* Code quality: Artemis Financial needs to write high-quality code that is free of security vulnerabilities.
* Encapsulation: Artemis Financial should encapsulate their code to prevent unauthorized access and modification.

***Based on my findings it is encouraged to focus more time and effort on adequately improving Input validation and cryptography. It is essential we help users secure data transmission over networks and ensure that only individuals with designated keys can access encrypted data. All data should be handled with the utmost security in mind.***

## Manual Review

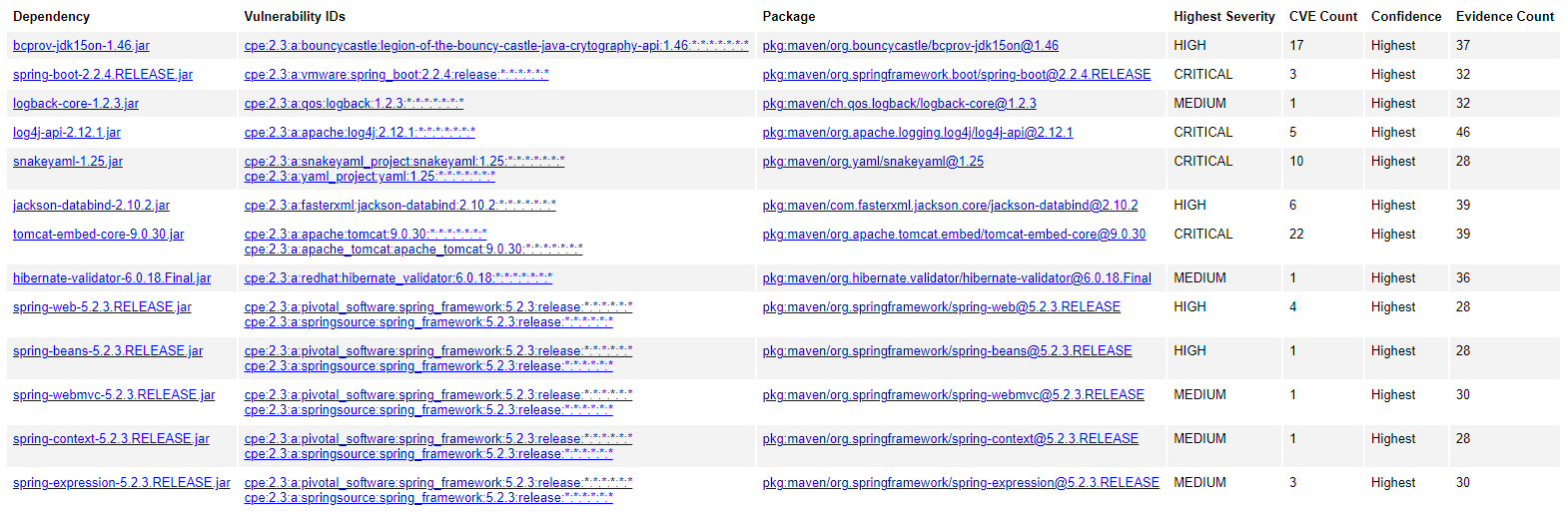
After successfully importing the files to eclipse and reviewing the code I noticed that in the DocData.java file, line 26 had a variable that was unused. To test, the test uses the database name, and root is username and password. This leaves the system vulnerable since the root is using default logins, such as username and password. It also states that “the value of the local variable con is not used”.

## Static Testing

*Maven dependency check:*

A screenshot of a web page

Description automatically generated



*bcprov-jdk15on-1.46.jar:*

The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7.

The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169.

*spring-boot-2.2.4.RELEASE.jar:*

\*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.

*logback-core-1.2.3.jar:*

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.

*log4j-api-2.12.1.jar:*

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. Fixed in Apache Log4j 2.12.3 and 2.13.1

snakeyaml-1.25.jar:

The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564. CWE-776 Improper Restriction of Recursive Entity References in DTDs ('XML Entity Expansion')

*jackson-databind-2.10.2.jar:*

General data-binding functionality for Jackson: works on core streaming API

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

The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.

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

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.

*spring-web-5.2.3.RELEASE.jar:*

Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data.

*spring-beans-5.2.3.RELEASE.jar:*

A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of vulnerability is more general, and there may be other ways to exploit it.

*spring-webmvc-5.2.3.RELEASE.jar:*

In Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries. This is a follow-up to CVE-2021-22096 that protects against additional types of input and in more places of the Spring Framework codebase.

*spring-context-5.2.3.RELEASE.jar:*

In Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older unsupported versions, the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless it is listed with both upper and lower case for the first character of the field, including upper and lower case for the first character of all nested fields within the property path.

*spring-expression-5.2.3.RELEASE.jar:*

n Spring Framework versions 5.3.0 - 5.3.16 and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service condition.

## Mitigation Plan

To prevent brute force attacks Artemis Financial should implement stronger password parameters with two-factor authentication for accounts containing highly valuable information. They must improve the code within their programs to improve code quality and implement appropriate authentication and error detection protocols to ensure that all vulnerabilities are resolved. All software and servers used on the system must be updated to the latest software versions, including patches for any vulnerabilities found. It is also essential to use HTTPS to ensure the safety of all business and customer data. It is also recommended to move request parameters to headers. It is highly recommended that any code validates certificates on both the application and the web server.

Other recommendations:

* SQL injection vulnerability: Artemis Financial should validate all user input before using it in database queries. This can be done using a variety of methods, such as prepared statements and parameterized queries.
* XSS vulnerability: Artemis Financial should encode all output before displaying it to the user. This can be done using a variety of methods, such as HTML encoding.