****

# 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** | **7/16/2023** | **Alexander Slowik** |  |

## 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

Alexander Slowik

## Interpreting Client Needs

Artemis Financial (AF) needs a RESTful Web API to facilitate their financial products. These include savings, retirement, investment, and insurance plans. They need a thorough review and static testing of their codebase.

1. AF values secure communications as a backbone of their companies’ offerings. As a financial company they deal in trust. They want to ensure that they are seen as a trusted institution and their software does not let down their clients. Financial data is especially sensitive and secure communications are a must for them. They must do all that they can to ensure that their clients data is not intercepted or manipulated.
2. International Transactions: AF has not been clear about their intentions to conduct international transactions. Modern financial planners could very easily conduct these transactions or offer them in the future. Without a clear statement from the company, it can be very hard to determine what specific actions are necessary. See C.
3. Governments have many specific restrictions on international financial data that can vary based on the country. For instance, if AF was to offer transactions to clients in the EU they would have to conform to Payment Services Directive (PSD2) standards. This means that they would need to implement Consumer Identity and Access Management solutions that have both multi factor and continuous authentication. They would need Know Your Customer and Identity proofing capabilities as well as robust consent management solutions. (OneWelcome 2022) Without further direction from AF we can not reasonably assess compliance.
4. External Threats now and in the immediate future: Hackers and other malicious actors see financial institutions as self-rewarding bug bounties. They attempt to exploit any vulnerability in software implementation for direct monetary compensation. The Harvard business Review has highlighted increased cybersecurity risks due to the escalation of the Russian/Ukrainian war. These attacks are not limited to government entities but are also targeting financial institutions amongst other targets.(Kolbe 2022)
5. Modernization: We must consider the risks of using outdated libraries in our application. Using outdated libraries opens our software to known vulnerabilities that will be easily actioned on by venerable hackers. To ensure that we have the most secure product possible we need to keep our libraries up to date and use secure frameworks to create and deploy our software. One of these possible risks is open-source software. Open-source software’s source code is in the public domain, this is a double-edged sword. One advantage of open-source software is having so many eyes on the software to discover vulnerabilities. Having a large diverse group of perspectives is a valuable tool when assessing any software. On the other hand, it can be a liability because there is equal opportunity for a large diverse group of bad actors to discover and take advantage of vulnerabilities. We must be deliberate in our selection of our open-source code.

## Areas of Security

1. Input Validation: Our software needs to execute input validation. We need to perform input validation to secure the application against SQL injection, XML External Entity attacks (XXE) and other malicious input attacks. In its current state our software is vulnerable to all three of these attacks, mentioned in detail below.
2. APIs: We need to ensure that we use secure API interactions in this application. We are using APIs for financial data and that must be secured lest it is intercepted or manipulated by bad actors. Artemis Financials reputation is a great risk if we fail in this area.
3. Authentication: This being a financial application we need to ensure that only users with the correct level of authorization can access relevant data. Financial data must be kept private to operate a well-functioning investment institution.
4. Cryptography: We will need to ensure that we implement strong cryptography. We need to ensure that data is always encrypted. This includes data sitting in our database or being moved via APIs. When transmitting information we should use encrypted protocols such as HTTPS.
5. Code Quality: We need to ensure that we implement secure coding practices and methodologies. No hardcoded passwords, no implementation of permission by default schemes.

## Manual Review

1. DocData class: Input validation required.
   1. The Doc data class contains a public method called read\_document takes both Key and Value strings but neither has any validation placed upon it. This needs to be corrected as it acts as a gateway for potential injections attacks, such as SQL injection. Additionally, this is a possible gateway for Spring Expression Language attacks, which can cause denial of service conditions.
2. DocData class: No secure connection to the database.
   1. DriverManager.getConnection lacks secure connection parameters. According to Microsoft when accessing a SQL database in a secure fashion one must implement the following: String connectionUrl =

"jdbc:sqlserver://localhost:1433;" +

"databaseName=AdventureWorks;integratedSecurity=true;" +

"encrypt=true; trustServerCertificate=false;" +

"trustStore=storeName;trustStorePassword=storePassword"; (David-Engel 2023)

We would have to change the name of the server and such, but we need to implement a secure connection.

* 1. The DocData class has leftover plaintext testing passwords and connections in the code. Line 27 contains a test connection with a hardcoded password/username combo of root/root. We need to avoid this as it could possibly lead to an ingenious user finding a way to manipulate the authentication of the test server and apply it to something outside of its scope in some sort of Exposure of Resource to Wrong Sphere. Meaning that if we authenticate a user as root on the test server a hacker could use this authenticated state to access data through our unsecured SQL connection. See below for CWE’s that mention this vulnerability.

## Static Testing

### bcprov-jdk15on-1.46.jar

**Description:**

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.

### spring-boot-2.2.4.RELEASE.jar

**Description:**

Spring Boot

### logback-core-1.2.3.jar

**Description:**

logback-core module

### log4j-api-2.12.1.jar

**Description:**

The Apache Log4j API

### snakeyaml-1.25.jar

**Description:**

YAML 1.1 parser and emitter for Java

### jackson-databind-2.10.2.jar

**Description:**

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

### tomcat-embed-core-9.0.30.jar

**Description:**

Core Tomcat implementation

### hibernate-validator-6.0.18.Final.jar

**Description:**

Hibernate's Bean Validation (JSR-380) reference implementation.

### spring-web-5.2.3.RELEASE.jar

**Description:**

Spring Web

### spring-beans-5.2.3.RELEASE.jar

**Description:**

Spring Beans

### spring-webmvc-5.2.3.RELEASE.jar

**Description:**

Spring Web MVC

### spring-context-5.2.3.RELEASE.jar

**Description:**

Spring Context

### spring-expression-5.2.3.RELEASE.jar

**Description:**

Spring Expression Language (SpEL)

## Mitigation Plan

Bouncy Castle has weaknesses in versions before 1.55 and needs to be upgraded to at least that point or to a newer version. These vulnerabilities can allow attackers to break the encryption through invalid curve attacks and timing attacks (CVE-2015-7940, CVE,2016-1000341 respectively). We can mitigate these errors by patching to the current version 1.75. Alternatively, we could increase security by implementing closed source encryption.

Spring boot 2.2.4 has vulnerabilities that expose resources to the wrong sphere. This means that functionalities and resources can be exposed to unintended users (CVE-2022-27772). Additionally, the whole of Spring is pulling along with it many outdated packages and they need to be upgraded. We should upgrade to version 3.11 to mitigate these issues.

Log4j has a vulnerability that allows for man in the middle attacks that could leak log information (CVE-2020-9488). This was fixed in even one version newer. We should update to version 2.20.

While these are just some of the relevant CVEs that are created by out-of-date packages there are a total of 74. We need to update Spring to 3.11, and Bouncy Castle to 1.75 or replace it with a different encryption tool as our highest priority items. Additionally, we need to review the DocData class. We need to implement a secure database connection and eliminate hard coded testing passwords. We also need to perform data validation on the read\_document method.

Works Cited:

Kolbe, P. (2022, February 24). *The cybersecurity risks of an escalating Russia-Ukraine conflict*. Harvard Business Review. https://hbr.org/2022/02/the-cybersecurity-risks-of-an-escalating-russia-ukraine-conflict

*What do you need to be PSD2 compliant?: Blog*. OneWelcome. (n.d.). <https://www.onewelcome.com/news/what-do-you-need-to-be-psd2-compliant>

David-Engel. (n.d.). *Connecting with encryption - JDBC driver for SQL server*. JDBC Driver for SQL Server | Microsoft Learn. https://learn.microsoft.com/en-us/sql/connect/jdbc/connecting-with-ssl-encryption?view=sql-server-ver16