****

# 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** | **5/28/2023** | **Zachary Derepentigny** |  |

## 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

Zachary Derepentigny

## Interpreting Client Needs

Artemis Financial requires security for their web-based application, so that it can be protected from external threats to their clients’ and their data. Secure communications are particularly valuable because of the sensitive financial data being shared between Artemis and its clients. This data is of high importance and vulnerability. In the field of business that Artemis is in, it is likely that they have a number of international clients, which will subject them to varying international privacy and security laws, as well as varying degrees of risk in the sharing of data with international entities. By extension, different international governments are likely to have varying laws that would also affect security requirements for Artemis. As discussed, external threats are the primary risk factor when dealing with clients’ financial information, so considerations should primarily target this area of vulnerability. As the application becomes more and more modernized, open-source libraries also become more utilized. These have the added risk that they are publicly available and may be more exploitable, however open access also means that vulnerabilities are also identified and resolved faster. In addition, the application must also keep up with modernized web applications, to ensure its security is always up to date.

## Areas of Security

Input validations are relevant to ensure that user inputs are limited only to those relevant and secure to the application, so that malicious inputs do not occur internally. APIs will often be important to ensure that inputs can be made from external sources securely. Cryptography is used to make sure that any data being exchanged in the application is properly encrypted and secured from unauthorized access. Client/Server ensures that any access to the application is only done as authorized by allowed entities.

## Manual Review

The use of usernames and passwords in the DocData.java class file makes the application vulnerable to straightforward attempts at access to the application by non-authorized users.

## Static Testing

The static testing process resulted in many identified vulnerable dependencies. These are as follows:

[bcprov-jdk15on-1.46.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7)

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.

[hibernate-validator-6.0.18.Final.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492)

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

[jackson-databind-2.10.2.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec)

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

[log4j-api-2.12.1.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb)

The Apache Log4j API

[logback-core-1.2.3.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l12_864344400c3d4d92dfeb0a305dc87d953677c03c)

logback-core module

[snakeyaml-1.25.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l14_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421)

YAML 1.1 parser and emitter for Java

[spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l15_225a4fd31156c254e3bb92adb42ee8c6de812714)

Spring Boot

[spring-boot-autoconfigure-2.2.4.RELEASE.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l16_36d2bb5d0eeb616eb84f754c4076b30e6b61614c)

Spring Boot AutoConfigure

[spring-boot-starter-web-2.2.4.RELEASE.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l17_ec75d01d212b5229c16d872fb127744c0ed46ed8)

Starter for building web, including RESTful, applications using Spring

MVC. Uses Tomcat as the default embedded container

[spring-core-5.2.3.RELEASE.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l18_3734223040040e8c3fecd5faa3ae8a1ed6da146b)

Spring Core

[spring-web-5.2.3.RELEASE.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l19_dd386a02e40b915ab400a3bf9f586d2dc4c0852c)

Spring Web

[spring-webmvc-5.2.3.RELEASE.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l20_745a62502023d2496b565b7fe102bb1ee229d6b7)

Spring Web MVC

[tomcat-embed-core-9.0.30.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l21_ad32909314fe2ba02cec036434c0addd19bcc580)

Core Tomcat implementation

[tomcat-embed-websocket-9.0.30.jar](file:///C:\Users\zachd\Documents\College\CS305\Project%20One\CS%20305%20Project%20One%20Code%20Base\rest-service\target\dependency-check-report.html#l23_33157f6bc5bfd03380ebb5ac476db0600a04168d)

Core Tomcat implementation

## Mitigation Plan

The mitigation plan would primarily revolve around access security, due to the current usage of username and password to access the application. Within this system, we can set strict requirements for user logins to reinforce them against potential malicious access. For consideration, there could potentially be advancements to two-factor authentication to add an additional layer of security to the application. In addition, dependencies such as bouncy castle are more vulnerable when using out of date versions, so the primary form of mitigation for these would be to ensure that the versions used in this application are always kept up to date.