****

# 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** | **11/7/23** | **Derek Kwasniewski** | **Start** |

## 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

Derek Kwasniewski

## Interpreting Client Needs

The client, Artemis Financial, uses custom software for their operations in which they develop individualized financial plans for its customers including savings, retirement, investments, and insurance. This means that the value of security to them is of the upmost importance as they will be dealing with lots of confidential customer information that will be transferred back and forth and most likely stored on a server for later access. When looking at the needs of the company, it is not known whether they operate internationally. Once thing to note about customer information security, is that if based in the USA, Artemis Financial must abide by the Gramm-Leach-Bliley Act which requires companies or businesses that deal with sensitive customer information to share information about third parties that they will be sharing the information with and give reasonable options for the customer to opt out of sharing their information. External attacks that should be noted are those that could intercept the transferred information. Since this information should be kept confidential, proper encryption should be used to ensure it is sent safely. Secure databases should be used with proper input validation from external entities to make sure that attackers are unable to gain access to a database via some form of injection. Any open-source libraries and web technologies should be updated to the latest version as soon as possible to ensure that you are not running on versions that may no longer receive service and could allow attackers to gain access after utilizing open vulnerabilities.

## Areas of Security

**Input Validation** – Proper input validation should be used to ensure that only expected input in received by not only the user, but also the RESTful web API to ensure that attempts through mean such as SQL Injection are prevented.

**APIs** – Since the RESTful web API will be used to allow customers and Artemis Financial to talk to one another and exchange information securely, these interactions should be assessed to make sure that the sensitive information transferred is safe.

**Code Error** – Secure code error handling is important as it will help ensure that in the event of a large error, that the system knows how to handle said error and prevent vulnerabilities from opening.

**Code Quality** – Using proper coding technique should be implemented to make certain that vulnerabilities are not being “built-in” and are prevented. Having a set coding standard set for the system makes it easier for future updates to be implemented.

## Manual Review

When looking through the files, I first started with the pom.xml file where the first issue is found. In here, I believe it would be beneficial to use the latest version of Spring boot that way the system is using the latest version with the most up to date bug fixes. Next, in the CRUDController.java class, I can see that no input validation is used when getting the requested “business\_name” parameter. Since this is just the name of the customer, I see no reason to use any encryption, however not using input validation for a URL request could be a site for vulnerabilities. This is also present in the GreetingController.java class when requesting the parameter “name”. As of right now proper input validation is the biggest vulnerability concern at the moment as there is no noted use of any API for the transfer of customer data that is sensitive and that will need to be encrypted.

## Static Testing

|  |  |  |
| --- | --- | --- |
| **Dependency Name** | **Dependency Description & Solution** | **Vulnerabilities** |
| 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. Update to latest version. | * CVE-2016-1000338 * CVE-2016-1000342 * CVE-2016-1000343 * CVE-2016-1000344 * CVE-2016-1000352 * CVE-2016-1000341 * CVE-2016-1000345 * CVE-2017-13098 * CVE-2020-15522 * CVE-2020-0187 (OSSINDEX) * CVE-2016-1000339 * CVE-2020-26939 (OSSINDEX) * CVE-2023-33201 (OSSINDEX) * CVE-2015-7940 * CVE-2018-5382 * CVE-2013-1624 * CVE-2016-1000346 * CVE-2015-6644 (OSSINDEX) |
| hibernate-validator-6.0.18.Final.jar | Hibernate's Bean Validation (JSR-380) reference implementation. Update to latest version. | * CVE-2020-10693 |
| jackson-databind-2.10.2.jar | General data-binding functionality for Jackson: works on core streaming API. Update to latest version, | * CVE-2020-25649 * CVE-2020-36518 * CVE-2021-46877 * CVE-2022-42003 * CVE-2022-42004 * CVE-2023-35116 |
| log4j-api-2.12.1.jar | The Apache Log4j API. Update to latest version. | * CVE-2020-9488 |
| logback-core-1.2.3.jar | logback-core module. Update to latest version. | * CVE-2021-42550 |
| snakeyaml-1.25.jar | YAML 1.1 parser and emitter for Java. Update to latest version. | * CVE-2022-1471 * CVE-2017-18640 * CVE-2022-25857 * CVE-2022-38749 * CVE-2022-38751 * CVE-2022-38752 * CVE-2022-41854 * CVE-2022-38750 |
| spring-boot-2.2.4.RELEASE.jar | Spring Boot. Update to latest version. | * CVE-2023-20873 * CVE-2022-27772 * CVE-2023-20883 |
| spring-boot-starter-web-2.2.4.RELEASE.jar | Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container. Update to latest version. | * CVE-2023-20873 * CVE-2022-27772 * CVE-2023-20883 |
| spring-core-5.2.3.RELEASE.jar | Spring Core. Update to latest version. | * 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 |
| spring-web-5.2.3.RELEASE.jar | Spring Web. Update to latest version. | * CVE-2016-1000027 * 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 |
| spring-webmvc-5.2.3.RELEASE.jar | Spring Web MVC. Update to latest version. | * 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 |
| tomcat-embed-core-9.0.30.jar | Core Tomcat implementation. Update to latest version. | * CVE-2020-1938 * CVE-2020-11996 * CVE-2020-13934 * CVE-2020-13935 * CVE-2020-17527 * CVE-2021-25122 * CVE-2021-41079 * CVE-2022-29885 * CVE-2022-42252 * CVE-2023-44487 * CVE-2020-9484 * CVE-2021-25329 * CVE-2021-30640 * CVE-2022-34305 * CVE-2023-41080 * CVE-2021-24122 * CVE-2021-33037 * CVE-2023-42795 * CVE-2023-45648 * CVE-2019-17569 * CVE-2020-1935 * CVE-2020-13943 * CVE-2023-28708 * CVE-2021-43980 |
| tomcat-embed-websocket-9.0.30.jar | Core Tomcat implementation. Update to latest version. | * CVE-2020-1938 * CVE-2020-8022 * CVE-2020-11996 * CVE-2020-13934 * CVE-2020-13935 * CVE-2020-17527 * CVE-2021-25122 * CVE-2021-41079 * CVE-2022-29885 * CVE-2022-42252 * CVE-2023-44487 * CVE-2020-9484 * CVE-2021-25329 * CVE-2021-30640 * CVE-2022-34305 * CVE-2023-41080 * CVE-2021-24122 * CVE-2021-33037 * CVE-2023-42795 * CVE-2023-45648 * CVE-2019-17569 * CVE-2020-1935 * CVE-2020-13943 * CVE-2023-28708 * CVE-2021-43980 |

## 

## Mitigation Plan

Based on the list of vulnerabilities listed, I believe it would be best to start by addressing the dependencies list and work through each one and update them to the most recent version. Doing this allows us to be sure that the dependencies are running a version with the most amount of bug fixes that would otherwise pop up in older versions. After the dependencies list has been addressed, the class files listed in the manual review should have their vulnerabilities addressed. The main issue for the manual review items being the issue with improper input validation in the CRUDController and GreetingController java classes.