****

# 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** | **09/17/2023** | **Brian Smith** |  |

## Client



## Developer

Brian Smith

## Interpreting Client Needs

Artemis Financial is a company that develops and provides financial plans to their customers. They are seeking to modernize their operations and part of this is updating their RESTful web API. Some critical items to focus on with this are making sure that any communication that happens with the web API are secure so that customer information is not able to be intercepted. As customers of financial services often have to supply personal identifiable information (PII) to use services there are numerous governmental restrictions regarding the secure transmission of this data. In addition to the government regulations for this kind of data it is also important to protect customer data so that the company can maintain a good reputation and the trust of its customers so that the company can maintain current customers and attract new ones.

## Areas of Security

The areas of security vulnerability that are important to this project are input validation, apis, cryptography, code error, and code quality. Input validation is important to this project because controlling what is put into and processed by the api is critical to the correct running of the api software. API security is critical because it is a RESTful API that we are focusing on. Cryptography is necessary because we need to be encrypting the transmission of customer data due to government regulation and protecting this information from malicious entities. Code error and code quality are important because we need to control what comes in and out of the software.

## Manual Review

Things that I found during the manual review of the code were as follows:

* No authentication is present to protect customer information currently
* No encryption is currently present in the application
* No input validation is currently present in either crudcontroller or greetingcontroller, both of which take input from the user

## Static Testing

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

Vulnerabilities: CVE-2013-1624, CVE-2015-6644, CVE-2015-7940, CVE-2016-1000338, CVE-2016-1000339, CVE-2016-1000341, CVE-2016-1000342, CVE-2016-1000343, CVE-2016-1000344, CVE-2016-1000345, CVE-2016-1000346, CVE-2016-1000352, CVE-2017-13098, CVE-2018-5382, CVE-2020-0187, CVE-2020-26939, CVE-2023-33201

Spring-boot-2.2.4.RELEASE.jar

Spring Boot

Vulnerabilities: CVE-2022-27772, CVE-2023-20873, CVE-2023-20883

Logback-core-1.2.3.jar

Logback-core module

Vulnerabilities: CVE-2021-42550

Log4j-api-2.12.1.jar

Apache Log4j API

Vulnerabilities: CVE-2020-9488, CVE-2021-44228, CVE-2021-44832, CVE-2021-45046, CVE-2021-45105

Snakeyaml-1.25.jar

YAML 1.1 parser and emitter for Java

Vulnerabilities: CVE-2017-18640, CVE-2021-4235, CVE-2022-1471, CVE-2022-25857, CVE-2022-3064, CVE-2022-38749, CVE-2022-38750, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854

Jackson-databind-2.10.2.jar

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

Vulnerabilities: CVE-2020-25649, CVE-2020-36518, CVE-2021-46877, CVE-2022-42003, CVE-2022-42004, CVE-2023-35116

Tomcat-embed-core-9.0.30.jar

Core Tomcat implementation

Vulnerabilities: CVE-2019-17569, CVE-2020-11996, CVE-2020-13934, CVE-2020-13935, CVE-2020-13943, CVE-2020-17527, CVE-2020-1935, CVE-2020-1938, CVE-2020-8022, CVE-2020-9484, CVE-2021-24122, CVE-2021-25122, CVE-2021-25329, CVE-2021-30640, CVE-2021-33037, CVE-2021-41079, CVE-2021-43980, CVE-2022-29885, CVE-2022-34305, CVE-2022-42252, CVE-2023-28708, CVE-2023-41080

Hibernate-validator-6.0.18.Final.jar

Hibernate’s Bean Validation (JSR-380) reference implementation

Vulnerabilities: CVE-2020-10693

Spring-web-5.2.3.RELEASE.jar

Spring Web

Vulnerabilities: CVE-2016-1000027, CVE-2020-5421, CVE-2021-22096, CVE-2021-22118

Spring-beans-5.2.3.RELEASE.jar

Spring Beans

Vulnerabilities: CVE-2022-22965

Spring-webmvc-5.2.3.RELEASE.jar

Spring Web MVC

Vulnerabilities: CVE-2021-22060

Spring-context-5.2.3.RELEASE.jar

Spring Context

Vulnerabilities: CVE-2022-22968

Spring-expression-5.2.3.RELEASE.jar

Spring Expression Language (SpEL)

Vulnerabilities: CVE-2022-22950, CVE-2023-20861, CVE-2023-20863

## Mitigation Plan

For the vulnerabiliites found during the manual review I recommend implementing HTTPS for api communication so that data transmission can be encrypted and secured. In addition there needs to be user authentication added so that it can be verified who is accessing the system and limit their data access only to information that user is entitled to. There also needs to be input validation implemented so that data being processed by the application is controlled and limited only to valid input to prevent errors and possible damage to the application services. For the static testing vulnerabilities, getting the dependencies updated to the current versions of the dependencies will resolve the security errors and harden the API against malicious use.