****

# 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** | **1-22-2022** | **Matt Sandoval** |  |

## Client



## Developer

Matt Sandoval

## Interpreting Client Needs

Artemis Financial is a consulting company that develops individual financial plans that include saving, retirement, investment, and insurance. Secure communication is critical to a financial company; customers expect security, and governments demand security. The US government has created numerous regulations to enforce security practices. Some rules that Artemis may be required to follow are PCI-DSS for credit card management, HIPPA if Artemis gathers any health information to purchase insurance and Gramm-Leach-Bliley for financial information management. If Artemis operates in the European Union, Artemis will have many more privacy laws to follow. It is invaluable for Artemis to follow these regulations because failure to do so could lead to massive fines, such as in the EU, a violation can led to a 250,000 EUR fine. Even more important, a loss of customer trust could lead to a loss of business.

Artemis faces many external and internal threats. Thief of assets and identity thief are common in the financial sector. Artemis must protect against internal threats such as an employee stealing funds or installing malicious software. Artemis must also guard against external threats, such as hackers gaining unauthorized access to Artemis’s system. Hackers may attack Artemis directly or via a supply chain attack where hackers gain access through a compromised API that Artemis uses.

Security can be significantly improved through modernization. By utilizing well-supported APIs, programming languages, and software services, Artemis can reduce the likelihood of security vulnerabilities. Keeping software up to date by using the latest stable releases is crucial, as vendors often release fixes for vulnerabilities in their software updates.

## Areas of Security

1. Input Validation
2. APIs
3. Cryptography
4. Client/Server
5. Code Quality

## Manual Review

Pom.xml

Java 1.8 is not the current release

Bouncy Castle Provider 1.46 is not the current release

It does not appear that this API is used.

Spring Boot Starter Web 2.2.4 is not the current release

Spring Boot Starter Test 2.2.4 is not the current release

Jayway Json Path 1.46 is not the current release

CRUD.java

Class name “CRUD” is used twice with similar variable names. Comments should be added what the intention of the overloaded class function is.

The second “CRUD” class does not appear to be used.

CRUDController.java

It appears the business name is being parametrized then past to DocData for entry into the SQL database, this is good coding practice.

customer.java

“account\_balance” and “a” should not be integers, more than likely, a fraction deposit will need to be made and displayed.

DocData.java

The username and password should be more complex than “root”

Password should not be in the comments

Password should be encrypted

The “to do” comment should be completed

Greeting.java

“id” variable type does not match GreetingController.java variable type

GreetingController.java

“counter” variable type does not match Greeting.java variable type

myDataTime.java

This class never gets used.

RestServiceApplication.java

## Static Testing

Static testing performed with Maven Dependency Check 5.3.0.

|  |  |  |
| --- | --- | --- |
| Dependency | Description | Solutions |
| 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 Bounce Castle Provider 1.70 |
| spring-boot-2.2.4.RELEASE.jar | Spring Boot | Update to Spring Boot Start Web 3.0.2 |
| logback-core-1.2.3.jar | logback-core module | Update to Logback Core Module 1.4.5 |
| log4j-api-2.12.1.jar | The Apache Log4j API | Update Apache Log4j API to 2.19.0 |
| snakeyaml-1.25.jar | YAML 1.1 parser and emitter for Java | Update to SnakeYamle 1.33 |
| jackson-databind-2.10.2.jar | General data-binding functionality for Jackson: works on core streaming API | Update to Jackson Databind 2.14.1 |
| tomcat-embed-core-9.0.30.jar | Core Tomcat implementation | Update Tomcat Embed Core 11.0.0-M1 |
| hibernate-validator-6.0.18.Final.jar | Hibernate's Bean Validation (JSR-380) reference implementation. | Update Hibernate Validator Engine 8.0.0 Final |
| spring-web-5.2.3.RELEASE.jar | Spring Web | Update Spring Web 6.0.4 |
| spring-beans-5.2.3.RELEASE.jar | Spring Beans | Update Spring Beans 6.0.4 |
| spring-webmvc-5.2.3.RELEASE.jar | Spring Web MVC | Update Spring Web MVC 6.0.4 |
| spring-context-5.2.3.RELEASE.jar | Spring Context | Update Spring Web 6.0.4 |
| spring-expression-5.2.3.RELEASE.jar | Spring Expression Language (SpEL) | Update Spring Web 6.0.4 |

## Mitigation Plan

1. Edit the Pom.XML file to include each API's appropriate Maven repository information, simplifying updating dependencies.
2. Update all APIs via Maven
3. Rerun the static test to ensure the vulnerabilities have been fixed.
4. Meet with the development team and discuss solutions to the following issues.
   1. CRUD.java appears to have an unused class, and needs comments
   2. Customer.java should allow for decimal deposits and account balances
   3. DataDoc.java needs to handle the login information in a more secure way
   4. Greetings.java and GreetingController.java need matching variable types for ID and Counter
   5. myDataTime.java does not appear to be used and should be removed