****

# CS 305 Project One

**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** | **5/28/2022** | **Josh Millan** |  |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

[Insert your name here]

## 1. Interpreting Client Needs

Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Consider the following regarding how companies protect against external threats based on the scenario information:

* What is the value of secure communications to the company?
* Are there any international transactions that the company produces?
* Are there governmental restrictions about secure communications to consider?
* What external threats might be present now and in the immediate future?
* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

Customers can take use of the company's financial services. This might indicate that the data they're handling and sending contains sensitive information about the customers. This information might be the target of a hostile entity attempting to obtain it. The organization appears to have the ability to conduct worldwide transactions and deliver communications. When considering the possible hazards and safeguards that must be taken, this must be taken into account.

The firm will very certainly be required to obey the established government laws in the country in which it is situated, as well as make adaptations for overseas clients to comply with any special local ordinances that may be in place. Individuals and groups with a malicious intent to obtain access to a client's personal information are the most probable and urgent hazards to the business. HTTPS and two-factor authentication are two current criteria that can help safeguard patron information and thwart attackers.

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

It is critical to provide users with a secure environment. The API must be secure so that when users use the system, the commands they submit over the API are not stopped or changed, potentially resulting in a security breach. This portion will heavily rely on the quality of the code. If the computer code is not kept up to date and adheres to the most recent standard for safe coding, vulnerabilities can be exploited, perhaps leading to a hazardous occurrence.

The client / server will need to be protected as well, because the firm and software will be utilized to connect with and deliver a service to a consumer. Encrypting data transmitted over the network and testing it after it arrives can help avoid and identify any possible tampering. If you can use an encryption method that adds a secure bit to it and then verifies that bit when it's received, you may be confident that the data's integrity will be preserved.

## 3. Manual Review

Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

Within the program's coding, there does not appear to be a validation technique. This could lead to security threats infiltrating the system. The requests are not validated when they are sent out through the system's software. The potential usage of HTTPS as a secure communication protocol is the final piece that is missing.

## 4. Static Testing

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from dependency check report. Include the following:

1. The names or vulnerability codes of the known vulnerabilities
2. A brief description and recommended solutions provided by the dependency check report
3. Attribution (if any) that documents how this vulnerability has been identified or documented previously

* slf4j-api-1.7.30.jar
  + The slf4j API
* [bcprov-jdk15on-1.46.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l2_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.
* [spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l3_225a4fd31156c254e3bb92adb42ee8c6de812714)
  + Spring Boot
* [logback-core-1.2.3.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l4_864344400c3d4d92dfeb0a305dc87d953677c03c)
  + logback-core module
* [log4j-api-2.12.1.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l5_a55e6d987f50a515c9260b0451b4fa217dc539cb)
  + The Apache Log4j API
* [jul-to-slf4j-1.7.30.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l6_d58bebff8cbf70ff52b59208586095f467656c30)
  + JUL to SLF4J bridge
* [jakarta.annotation-api-1.3.5.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l7_59eb84ee0d616332ff44aba065f3888cf002cd2d)
  + Jakarta Annotations API
* [snakeyaml-1.25.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l8_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421)
  + YAML 1.1 parser and emitter for Java
* [jackson-databind-2.10.2.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l9_0528de95f198afafbcfb0c09d2e43b6e0ea663ec)
  + General data-binding functionality for Jackson: works on core streaming API
* [jackson-annotations-2.10.2.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l10_3a13b6105946541b8d4181a0506355b5fae63260)
  + Core annotations used for value types, used by Jackson data binding package.
* [jackson-core-2.10.2.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l11_73d4322a6bda684f676a2b5fe918361c4e5c7cca)
  + Core Jackson processing abstractions (aka Streaming API), implementation for JSON
* [jackson-datatype-jdk8-2.10.2.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l12_dca8c8ab85eaabefe021e2f1ac777f3a6b16a3cb)
  + Add-on module for Jackson (http://jackson.codehaus.org) to support
  + JDK 8 data types.
* [tomcat-embed-core-9.0.30.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l13_ad32909314fe2ba02cec036434c0addd19bcc580)
  + Core Tomcat implementation
* [tomcat-embed-el-9.0.30.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l14_406e0c7cc45204c0f81853d73e8cfedfa4e00945)
  + Core Tomcat implementation
* [jakarta.validation-api-2.0.2.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l15_5eacc6522521f7eacb081f95cee1e231648461e7)
  + Jakarta Bean Validation API
* [hibernate-validator-6.0.18.Final.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l16_7fd00bcd87e14b6ba66279282ef15efa30dd2492)
  + Hibernate's Bean Validation (JSR-380) reference implementation
* [jboss-logging-3.4.1.Final.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l17_40fd4d696c55793e996d1ff3c475833f836c2498)
  + The JBoss Logging Framework
* [classmate-1.5.1.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l18_3fe0bed568c62df5e89f4f174c101eab25345b6c)
  + Library for introspecting types with full generic information including resolving of field and method types.
* [spring-core-5.2.3.RELEASE.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l19_3734223040040e8c3fecd5faa3ae8a1ed6da146b)
  + Spring Core
* [spring-jcl-5.2.3.RELEASE.jar](file:///C:\Users\jrica\Desktop\rest-service\target\dependency-check-report.html#l20_fcba3ae00724ee18f37aa49ac887a5fd4ad77502)
  + Spring Commons Logging Bridge

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application.

It's critical to update the files that were detected during the dependency test. By failing to do so, the program will be losing out on important security upgrades as well as potential weaknesses in previous versions. The use of RESTful APIs with HTTPS as a communication protocol can assist safeguard data while it is being sent between customers and the firm. It's also necessary to develop and deploy a better two-factor/validation technique for users.