****

# 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)-5

[4. Static Testing 5](#_Toc32574614)-6

[5. Mitigation Plan](#_Toc32574615) 6

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **7/13/2023** | **Robert Lowrey** |  |

## 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

Robert Lowrey

## Interpreting Client Needs

The client is a consulting company that develops individualized financial plans for their customers. Therefore, the security level is going to be very high. This is because the company has access to individual customers finances and accounts in order to plan the best financial strategy. The plans they have are for savings, insurance, investments, and retirement. Artemis Financial deals with multiple people from around the world. Therefore, there will be a lot of potential security threats to the system.

There might be potential external threats right now such as an attacker wanting to get savings information in order to commit identity theft. There also might be future threats of this happening. Another external threat may be that an attacker may want to get a financial information of one of the customers in order to get ahold of investments. This could allow the attacker to make decisions that would hurt the customer that was involved.

Somethings to consider while looking over this project are going to be the new advanced technology that comes out daily that can cause a potential vulnerability in the system. Another thing to consider is the role of authorization when it comes to individuals. If we are going to want authorization for select individuals or allow everyone to access everything in the system. Lastly, I would say to consider the use of authentication. Whether it be two factor or multifactor in order to ensure a better layer of security.

## Areas of Security

An area of security that is needed is Input validation, this is relevant to the application because if we are using authentication or two factor authentication we need to ensure an attacker cannot bypass the system via injection.

Another are of security is the APIs. Since the system uses a RESTful web application programming interface, we need to ensure the secure interaction between the customer and the company along with the tools needed to develop a more in depth financial plan.

Also, cryptography is an area that is needed by the software application in order to allow encryption of the information that the system accesses and stores for the customers. This data can either be the financial information or the physical information of the customer.

Code Quality is another area that needs to be applied because this is a vital part of the security. This area can either allow or deny an attacker access to the system. With secure coding practices and principles we can make the system overall more secure and less vulnerable to a potential security risk in the present and future. Code quality will also allow for future maintenance of the application in order to stay up-to-date with the current known vulnerabilities of software applications.

The Last area of security I believe that needs to be added is the encapsulation area. This is relevant to the system because the database that holds the individual customers information needs to be secure from attacks. Either by encapsulating the database in multiple layers to ensure the security of the vulnerable information or encapsulating it in a very well defined secured structure.

## Manual Review

Upon manually inspecting the code for vulnerabilities, I have found there is an input validation vulnerability in the code base. This takes place in the GreetingController class. The code base does not show input validation when getting the string input. Another thing that I have found was that the cryptography needs improvement. There is no specification of the HTTPS file in the application.properties class. There is also no certificate location that was located in this file either. This is crucial for secure data transfers of important information. Code Quality is also another vulnerability that I found. There is an improper naming convention in the customer.java file. The deposit class uses a parameter of a; however, the parameter is never initialized. This will lead to a Code Error vulnerability if the system does not handle that exception properly,

## Static Testing

The names, description, and possible attributions for the current system vulnerabilities are as follows:

* **Dependency:** bcprov-jdk15on-1.46.jar
  + **Description:** The system uses a Bouncy Castle type of encryption system as well as a lightweight API for this cryptography
    - **Attributions:** Developed by the Legion of the Bouncy Castle
* **Dependency:** hibernate-validator-6.0.18.Final.jar
  + **Description:** A bug in the interpolation processor enables invalid expressions to be evaluated as if they were valid
    - **Attributions:** Developed by Red Hat
* **Dependency:** jackson-databind-2.10.2.jar
  + **Description:** The FasterXML Databind did not have entity expansion secured properly

* **Dependency:** log4j-api-2.12.1.jar
  + **Description:** Improper validation of certificate with host mismatch in Apache Log4j SMTP appender
* **Dependency:** logback-core-1.2.3.jar
  + **Description:** In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers.
    - **Attributions:** Logback website went live in 2006
* **Dependency:** snakeyaml-1.25.jar
  + **Description:** SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization
* **Dependency:** spring-boot-2.2.4.RELEASE.jar
  + **Description:** In some Spring Boot versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache.
    - **Attributions:** In a prior spring-boot version, the system was vulnerable to temporary directory hijacking
* **Dependency:** spring-boot-starter-web-2.2.4.RELEASE.jar
  + **Description:** A starter for building web applications and uses Tomcat as the default embedded container
    - **Attributions:** Spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. Also, in older supported versions there is a potential for a DoS attack.
* **Dependency:** spring-core-5.2.3.RELEASE.jar
  + **Description:** Spring Core
    - **Attributions:** If the application running on JDK 9+ may be susceptible to remote code execution via data binding
* **Dependency:** spring-web-5.2.3.RELEASE.jar
  + **Description:** Spring Web
* **Dependency:** spring-webmvc-5.2.3.RELEASE.jar
  + **Description:** Spring Web MVC
    - **Attributions:** May be vulnerable to remote code execution via data binding
* **Dependency:** tomcat-embed-core-9.0.30.jar
  + **Description:** Core Tomcat implementation
    - **Attributions:** Tomcat treats AJP connections as having higher trust than HTTP connections
* **Dependency:** tomcat-embed-websocket-9.0.30.jar
  + **Description:** Core Tomcat implementation
    - **Attributions:** Tomcat treats AJP connections as having higher trust than HTTP connections

## Mitigation Plan

A mitigation plan that I would recommend is to have the application.properties file needs to be corrected immediately in order to better secure the transfer of the sensitive data linked to each individual customer around the world. There also needs to be an implementation of the HTTPS protocol. This will help the cryptography security and secure the vulnerable data that the customers of Artemis Financial disclose to the company.

I would also recommend to assess the area of code quality in the security aspect. Meaning to use proper coding principles such as input validations. In the GreetingController class lines 15-18, there was no input validation. Therefore an attacker can bypass the authorization via injection.

Also, in the customer class, there was a parameter of a, but the class within that file was never accessed. I would do more digging into your Data Access area of security because this can throw an exception that an attacker can potentially get ahold of and know how to bypass.

The last step of this plan is to include a parser in the system. This will enable a safe parser of untrusted JSON data. The data must be either sanitized or parsed within web applications in order to prevent JSON injection. This is the mitigation plan I have came up with based on the knowledge of your system and its vulnerabilities. Following my plan will better help protect your customers at Artemis Financial and secure the customers trust with the company.