****

# 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** | **11/19/24** | **Iesha Sahin** | **1.0** |

## 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 Rubric for more detailed instructions about each section of the template.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

## Developer

Iesha Sahin

## Interpreting Client Needs

The client, Artemis Financial, is a consulting company that develops individual financial plans for their customers including savings, retirements, insurance, and investments. Because of the type of information that is being delt with, secure communication should be the top priority of the company. Naturally, with the type of clientele I would assume Artemis Financial has or would hope to have, international transactions seem like a given. Always being alert and identifying external threats is of extreme importance for Artemis Financial. One data breach can put all clients at risk. Because this is a financial company, the most private of information will be stored and it needs to be kept secured. When it comes to “modernization” requirements for Artemis Financial, staying up to date on all libraries being utilized and keeping the program error free should not only be considered, but implemented.

## Areas of Security

* Input validation— This program collects user input making validation crucial. This will prevent potential access from anyone without validated information and ensure others can’t access information that is not theirs.
* APIs— Because Artemis Financial will be working with customers from all over, an API is needed. With an API both Artemis Financial and their customers can feel safe and at ease that their correspondence will be secure. Both can also feel protected with a good API as it will control what data is accessible to who.
* Code error— Like any team or recipe, everything works together for the best outcome. With secure error handling, input validation and APIs can run smoothly. The last thing a company or their customers would want is unauthorized access. Breaches can be prevented with secure error handling.
* Cryptography— Because of the sensitivity of the information being transferred, cryptography is an absolute must. This will ensure that all data will be unreadable and secure.

## Manual Review

Within the GreetingController.java file, there currently is no use of input validation. There also seems to be no use of an API within the entire program yet. The same can be said for the lack of cryptography within the program. While there are only a few lines of code overall, I do think the quality is good, just missing a few important security measures.

## Static Testing

**bcprov-jdk15on-1.46.jar**

**ID**: cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*

**Description**: 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.

**Solution**: Update to version 1.7.

**spring-boot-2.2.4.RELEASE.jar**

**ID:** cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*

**Description**: spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method.

**Solution:** Update to version 3.1.5.

**logback-core-1.2.3.jar**

**ID**: cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*

**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.

**Solution**: Update to version 1.4.11.

**log4j-api-2.12.1.jar**

**ID**: cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*

**Description**: Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender. Fixed in Apache Log4j 2.12.3 and 2.13.1

**Solution**: Upgrade to version 2.21.1.

**snakeyaml-1.25.jar**

**Description**: The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

**Solution**: Upgrade to version 2.2.

**jackson-databind-2.10.2.jar**

**ID**: cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*

**Description**: A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.

**Solution**: Upgrade to version 2.15.

**tomcat-embed-core-9.0.30.jar**

**ID**: cpe:2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

**ID**: cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

**Description**: The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.

**Solution**: Upgrade to version 11.0.0.

**hibernate-validator-6.0.18.Final.jar**

**Description**: A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.

**Solution**: Update to version 6.0.23.

## Mitigation Plan

First and foremost, the easiest thing to do would be upgrading all systems to their current versions.

Next, add validation to GreetingController.java to ensure customer data safety. Finally, with the addition of cryptography, the transfers between Artemis Financial and their customers will be secured.