****

# 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** | **03/19/2023** | **Beth Campbell** | **Updated Interpreting Client Needs, Areas of Security, Manual Review, Static Testing, and Mitigation Plan** |

## 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

Beth Campbell

## Interpreting Client Needs

Secure communications within Artemis Financial will protect sensitive information, such as financial data and customer information, from unauthorized access. This will help build trusts with clients and investors while also preventing data breaches.

In the event Artemis Financial has international transactions, having secure communications becomes even more vital. International transactions may entail conformity with data protections laws in different countries. Having secure communications would safeguard against security risks that could be caused by from cross-border transmission.

With the consideration that Artemis Financial does offer international transactions various government restrictions must be followed. The most common are:

* The European General Data Protection Regulation which is designed to protect citizens from personal data compromise. This regulation requests website form submissions, collecting cookie data, storing IP addresses, and shredding documents containing personal information.
* Sarbanes-Oxley act of 2002 was passed by the United States Congress to protect investors. It ensures financial institutions address common security risks such as phishing attacks.
* Gramm-Leach-Bliley Act requires financial institutions to protect customer data and disclose all data-sharing practices with customers.

External threats that may be presented are:

* Ransomware
* Malware
* DDoS attacks
* Phishing
* Tyupkin

Modernization requirements will depend on the type of technology used. Open-source libraries can help developers create secure software efficiently and effectively. Evolving web-based applications also offer new opportunities for secure communications. To stay ahead of threats, regular updates and upgrades to security systems should be essential.

## Areas of Security

* Thorough input validation must be performed to prevent vulnerabilities. As users will input sensitive information such as usernames, passwords, and other data linked to their accounts, it is critical to validate this data to protect against potential threats.
* APIs will be a crucial component of the application since it is a RESTful web application. APIs can help provide secure communication between different components of the application.
* Cryptography will be required as encryption is necessary to protect the financial data stored by financial institutions.
* To minimize security risks, it is crucial to identify and address any code errors that could be exploited by malicious actors.
* As the code contains data structures, encapsulation should be used to ensure that sensitive data is kept secure and is not exposed to potential threats.

## Manual Review

Input Validation: The parameter requested at /greeting is being added to the "template" string without being validated, leaving it vulnerable to potentially harmful content. Additionally, improper formatting of the string due to lack of validation could lead to errors in the program.

APIs: A more secure method, unlike GET, is needed. The parameter is also included in the URL. This method could be leaked into the browser history and become a vulnerability.

Cryptography: There is currently no noticeable data encryption set.

## Static Testing

* bcprov-jdk15on-1.46.jar – The Bouncy Castle Crypto package is for cryptographic algorithms. Vulnerable to Hash Collision
* spring-boot-2.2.4.RELEASE.jar- Bundles different classes written to accomplish the task the application is designed for. Vulnerability includes ClassLoader access.
* logback-core-1.2.3.jar – Core implementation of Logback. Dependable, common, quick, and accommodating logging framework.
* log4j-api-2.12.1.jar- Extremely configurable monitoring tool that focuses on performing and minimal garbage production.
* snakeyaml-1.25.jar- Parser and emitter for Java
* jackson-databind-2.10.2.jar- Converts JSON to and from POJO
* tomcat-embed-core-9.0.30.jar- single Java web app with a Tomcat server
* hibernate-validator-6.0.18.Final.jar- Bean Validation 2.0 reference implementation
* spring-web-5.2.3.RELEASE.jar- Provides assimilation such as multipart file upload functionality
* spring-beans-5.2.3.RELEASE.jar- Provides configuration framework and functions to instantiate, configure, and assemble java objects
* spring-webmvc-5.2.3.RELEASE.jar- Contains Spring’s MVC and REST services
* spring-context-5.2.3.RELEASE.jar- Provides access to configured objects like a registry
* spring-expression-5.2.3.RELEASE.jar- Powerful Expression Language for interrogating and manipulating an object graph at runtime

## Mitigation Plan

Upgrading the majority of these vulnerabilities to the current versions would increase security.

Set up validation for user input at /greeting.

Use the POST method for APIs instead of GET.