****

# 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** | **5/15/2023** | **Kalee Li** |  |

## Client



## Developer

Kalee Li

## Interpreting Client Needs

1. What is the value of secure communications to the company?
2. Does the company make any international transactions?
3. Are there governmental restrictions about secure communications to consider?
4. What external threats might be present now and in the immediate future?
5. What are the modernization requirements that you must consider? For example:
   1. The role of open-source libraries
   2. Evolving web application technologies

Artemis Financial is a consulting firm that offers customized financial plans for individuals. Their plans comprise financial savings, retirement, investments, and insurance. Artemis Financial highly values the security of their software, supposed a financial consulting firm needs to obtain and store sensitive and personal data. The data may include personal monetary flows, loans, debt, wills, trusts, etc. Therefore, the key part of the success of their software is to protect the organization from external threats. The common external threats presenting currently for the financial industry are phishing and spoofing, whereas personal data such as identity, credentials, and accounts get stolen and impersonated. And in the immediate future, the threat to the financial industry that we should look out for is SQL injection. Attackers can bypass authentication, gain privileges to change the database structure and execute inconsistent behavior by exploiting SQL injection Vulnerabilities.

As a result, any financial industry must consider and adhere to the Financial Cybersecurity compliance law and regulations.

“Financial institutions depend on IT to deliver services. Disruption, degradation, or unauthorized alteration of information and systems can affect the financial condition, core processes, and risk profile of an institution. Further, because of the increasing volume and sophistication of cyber threats, it is imperative that financial institutions and their critical third-party service providers maintain diligence in identifying, assessing, and mitigating cybersecurity risks.” – *(FDIC | Banker Resource Center: Information Technology (IT) and Cybersecurity, n.d.)*

One of the most common cybersecurity legislation requirements is access control. We must implement authorization and authentication methods in the software to prevent the exploitation of vulnerabilities.

The evolving web application technologies, the open-source libraries should be considered in this project. Open-source software maintains a high level of security, stability, quality, and reliability. Moreover, it is free and customized. Open source code offers a wide range of capabilities, and users can choose whatever function meets their needs. Open source also allows developers and security professionals to access source code to scrutinize and repair bugs/errors. Therefore, it is important to consider using open-source code and framework.

## Areas of Security

After reviewing the scenario, the relevant areas of security I would suggest based on my general knowledge of the web application architecture following the VAPFD are:

* Input Validation: Secure input and representations - To prevent inconsistent input that might affect the program’s behaviors, adversaries add attack strings to legal input fields. Therefore, the web application needs secure input verification.
* APIs: Secure API interaction - We are using the Spring Framework (spring-data-rest-webmvc) to build the web application which supports APIs interaction. Therefore, secure API interaction is necessary.
* Cryptography: Encryption use and vulnerabilities – This project obtains and stores sensitive information. Therefore, cryptography is necessary to protect customer privacy and data confidentiality.
* Code Quality: Secure coding practices/patterns - Code Quality includes input validation and APIs; hence secure coding practices/patterns are vital to the areas of security.

## Manual Review

After carefully inspecting the code files and applying the Vulnerability assessment process, I noticed a few things that need to be addressed.

* Input Validation was not defined as what data is allowed and what to reject.
* Authentication was not implemented for user verification purposes
* Authorization /Access control – limitation of users to accessibility was not implemented
* The SQL connection URL included the username and password.
* The read\_document() function in DocData.java was partially done. The query statement was not present, and the catch block was left empty. Therefore, the system generated // TODO Auto-generated catch block e.printStackTrace();
* Encryption was not implemented.
* Unit Test was incompleted.
* Comments and descriptions were omitted

## Static Testing

**Dependency:** spring-core-5.2.3.RELEASE.jar , spring-web-5.2.3.RELEASE.jar, spring-webmvc-5.2.3.RELEASE.jar

**Vulnerability ID:** CVE-2023-20863, CVE-2023-20861, CVE-2022-22971,CVE-2022-22970, CVE-2022-22968, CVE-2022-22965, CVE-2022-22950, CVE-2021-22060, CVE-2021-22096, CVE-2021-22118, CVE-2020-5421, CVE-2016-1000027

**Description:** “ In spring framework versions prior to 5.2.24 release+ ,5.3.27+ and 6.0.8+ , it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition.”

**Dependency:** tomcat-embed-core-9.0.30.jar, tomcat-embed-websocket-9.0.30.jar

**Vulnerability ID:** CVE-2023-28708, CVE-2022-42252, CVE-2021-43980, CVE-2022-34305, CVE-2022-29885cCVE-2021-41079, CVE-2021-33037, CVE-2021-30640, CVE-2021-25329, CVE-2021-25122, CVE-2021-24122, CVE-2020-17527, CVE-2020-13943, CVE-2020-13935, CVE-2020-13934, CVE-2020-8022, CVE-2020-11996, CVE-2020-11996, CVE-2020-1938, CVE-2020-1935

**Description**: “ When using the RemoteIpFilter with requests received from a reverse proxy via HTTP that include the X-Forwarded-Proto header set to https, session cookies created by Apache Tomcat 11.0.0-M1 to 11.0.0.-M2, 10.1.0-M1 to 10.1.5, 9.0.0-M1 to 9.0.71 and 8.5.0 to 8.5.85 did not include the secure attribute. This could result in the user agent transmitting the session cookie over an insecure channel. “

## Mitigation Plan

Based on the manual review and dependency-check report, a few suggestions need to be implemented to fix each of the vulnerabilities found in this report.

* Whitelisting input validation – only accepts the appropriate data input and rejects everything else.
* Authentication (Verifies Users) and authorization users (Set Roles) by creating @confirguration class and use basic authentication to protect the REST endpoints.
* Remove the username and password in the database connection URL – create variables instead.

String username = “root”

String password = “root”

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test",username, password)

* Put this in the catch block- System.out.println(e);
* Implement Java Cryptography to protect data in transit and rest.
* Unit testing is an essential part of code quality. Each test case verifies the codes are executing as expected, and it tests all possible conditions to ensure the system behaves according to the requirements.
* Use Comments and descriptions to improve readability
* Use the most updated version of the Spring framework and Apache Tomcat
* review the dependency-check report to see if there is any false positive by evaluating the evidence and identifier.

Reference:

*FDIC | Banker Resource Center: Information Technology (IT) and Cybersecurity*. (n.d.). https://www.fdic.gov/resources/bankers/information-technology/

Manico, J., & Detlefsen, A. (2014). *Iron-Clad Java: Building Secure Web Applications*. McGraw Hill Professional.