****

# 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** | **September 18, 2022** | **Jessica McAlum** |  |

## 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

Jessica McAlum

## Interpreting Client Needs

Artemis Financial is a consulting company that specializes in developing individualized financial plans for their clients. Secure communications are of vital importance to this company because their application will be dealing with sensitive customer information. Artemis Financial does not plan on making any international transactions at this time. However, since transactions will need to be made, we must ensure that our security and software complies with the PCI DSS (Payment Card Industry Data Security Standard). Because Artemis Financial is a type of financial institution, we must consider that United States federal law requires Artemis Financial to make their information-sharing practices public and take precautions to protect customer data. External threats that might be present now and in the immediate future would be untrusted data and any individual/hacker that attempts to infiltrate our databases to access sensitive data.

## Areas of Security

The areas of security that apply to Artemis Financial’s web application are Input Validation, APIs, Cryptography, Client/Server, and Code Quality. Secure verification of input and representations are needed for this web application to prevent any injection vulnerabilities. The company also utilizes APIs, so it’s essential to have secure API interactions to prevent any data breaches due to a hacked API. Cryptography is very also important to ensure secure communications, especially with Artemis Financial who will be working with very sensitive data. Secure communication between the client and server also apply to this application. Code quality involves all of the previously mentioned areas of security and not only does it directly impact the user experience, but it also directly impacts how secure the application will be.

## Manual Review

One vulnerability found while inspecting the code base was in the DocData class. The DocData class contains the database name, username and password. This sensitive information could potentially be accessed via SQL injection from the CRUDController class.

## Static Testing

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

**Description:** Core Tomcat implementation

**Severity**: Critical

**Vuln IDs:** CVE-2020-1938, CVE-2020-8022, CVE-2020-11996, CVE-2020-13934, CVE-2020-13935, CVE-2020-17527, CVE-2021-25122, CVE-2021-41079, CVE-2022-29885, CVE-2020-9484, CVE-2021-25329, CVE-2021-30640, CVE-2022-34305, CVE-2021-24122, CVE-2021-33037, CVE-2019-17569, CVE-2020-1935, CVE-2020-13943

**Recommended Solution:** Update to latest version as soon as the latest version is released.

* **Dependency**: spring-core-5.2.3.RELEASE.jar

**Description:** Spring Core

**Severity:** Critical

**Vuln IDs**: CVE-2022-22965, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096

**Recommended Solutions:** Upgrade to latest version that has been released where these issues have been fixed.

* **Dependency:** spring-web-5.2.3.RELEASE.jar

**Description:** Spring Web

**Severity:** Critical

**Vuln ID:** CVE-2016-1000027

**Recommended Solution:** Update to latest version where this issue has been fixed.

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

**Description:** Spring Boot

**Severity:** High

**Vuln ID:** CVE-2022-27772

**Recommended Solution:** Update to latest version where this is no longer a vulnerability.

* **Dependency:** snakeyaml-1.25.jar

**Description:** YAML 1.1 parser and emitter for Java

**Severity:** High

**Vuln IDs:** CVE-2017-18640, CVE-2022-25857, CVE-2022-38749, CVE-2022-38751, CVE-2022-38752 (OSSINDEX), CVE-2022-38750

**Recommended Solution:** Update to latest version where these issues have been resolved.

* **Dependency:** jackson-databind-2.10.2.jar

**Description:** General data-binding functionality for Jackson: works on core streaming API

**Severity:** High

**Vuln IDs:** CVE-2020-25649, CVE-2020-36518

**Recommended Solution:** Update to latest version**.**

* **Dependency:** bcprov-jdk15on-1.46.jar

**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.

**Severity:** High

**Vuln IDs:** CVE-2016-1000338, CVE-2016-1000342, CVE-2016-1000343, CVE-2016-1000344, CVE-2016-1000352, CVE-2016-1000341, CVE-2016-1000345, CVE-2017-13098, CVE-2020-15522, CVE-2020-0187 (OSSINDEX), CVE-2016-1000339, CVE-2020-26939 (OSSINDEX), CVE-2015-7940, CVE-2018-5382, CVE-2013-1624, CVE-2016-1000346, CVE-2015-6644 (OSSINDEX)

**Recommended Solution:** Back up existing installation then apply the update to the latest version.

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

**Description:** Hibernate's Bean Validation (JSR-380) reference implementation.

**Severity:** Medium

**Vuln IDs:** CVE-2020-10693

**Recommended Solution:** Upgrade to latest version.

* **Dependency:** log4j-api-2.12.1.jar

**Description:** The Apache Log4j API

**Severity:** Low

**Vuln ID:** CVE-2020-9488

**Recommended Solution:** Upgrade to latest version.

## Mitigation Plan

The static testing results indicated that there were many vulnerable dependencies found in our code base. These vulnerabilities were mainly due to outdated dependencies. We must ensure that we frequently check for any outdated dependencies because outdated dependencies could introduce vulnerabilities to our software. Keeping our software dependencies up to date will ensure that our software is up to date with any bug/security issue fixes. As for the vulnerability to SQL injections found in the manual review, we can help to mitigate this vulnerability by using techniques like query parameterization. Query parameterization can help us to avoid SQL injections.