****

# 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** | **January 25th, 2024** | **Corey Sampson** |  |

## 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

Corey Sampson

## Interpreting Client Needs

1. ***What is the value of secure communications to the company?***The value of secure communications to Artemis Financial is crucial since they deal with valuable and personal information of their clients internally and externally. It is evident that the protection of their client information and communications are a major focus regarding the value of security with the modernization of their operations. Providing protection from an attack by a hacker or some third-party application is crucial to the security structure of their software.
2. ***Does the company make any international transactions?***Though there is no note of whether Artemis Financial makes international transactions, we can safely assume they would want their software set up to be able to handle potential transactions that take place outside of the U.S.
3. ***Are there governmental restrictions about secure communications to consider?***Depending on the location of the company, when dealing with sensitive information (such as personal client information) it is required in at least 32 states to ensure secure holding of data. While this may be the only regulation/requirement on a governmental level, there are opportunities for increased regulations regarding security surrounding certain types of data for certain types of companies.
4. ***What external threats might be present now and in the immediate future?***The biggest threat that presents itself is the potential of data leaks including, but not limited to, client personal information being leaked and/or stolen. Masking information through heavy encryption can be one of multiple ways to help ensure external threats are dealt with properly as well as a consistent and regular update/maintenance to the code base.
5. ***What are the modernization requirements that you must consider? For example:***When modernizing software, you must consider libraries staying up to date within the application, especially if the libraries utilized are open source. Open-source libraries tend to find vulnerabilities and update them accordingly when an exploit is discovered. Another consideration for modernization to implement is authentication methods for client/user side logins regarding application technologies. By requiring something like a two-faced authenticator, you can ensure a much lesser chance of the wrong person getting into the wrong account.

## Areas of Security

When determining areas of security, it is important to determine what their software application makes use of. By reviewing the client needs, I can confidently determine the Areas of Security to be as follows:

1. ***Input Validation***Since the software will utilize user created accounts as well as potential forms for a user to fill out electronically, input validation is crucial in combatting against vulnerabilities and exploits.
2. ***APIs***As mentioned, Artemis Financial is using the RESTful API so it is important to implement secure API interactions. This is where the data is handled externally so preventing certain exploits and vulnerabilities that may be issued by a third-party software is crucial and done through proper API security measures.
3. ***Cryptography***With the potential of having international transactions it is important to utilize cryptography through encryption. When handling user data/personal information in transactions the data should be handled according to governmental regulations for both the company location and the transaction location. Encryption is the best way to prevent exploits in this scenario.
4. ***Code Error***Typically, code error goes hand in hand with input validation and secure API interactions. By ensuring secure error handling you can reinforce your security measures.
5. ***Code Quality***Similarly to code error handling, the quality of the code base is crucial. Using positive patterns/practices and avoiding anti-patterns with security can also help to reinforce the security for the software application.

## Manual Review

There are multiple concerns regarding security after reviewing the code base, they are as follows:

* The use of HTTP as opposed to the use of HTTPS.
* There is a lack of authentication for verifying users.
* Requests are not validated via POST and instead handled through the URL, leaving potential vulnerabilities.
* In the CRUD controller class, business names are sent as parameters, making them visible and presenting the potential for a data leak.
* There is a lack of error handling in the DocData.java class.
* There is no input validation.

These security concerns leave room for vulnerabilities and exploits such as data leaks, stolen data, and even SQL injection.

## Static Testing

A screenshot of a computer

Description automatically generated

*In order from lowest severity to highest severity\**

* ***log4j-api-2.12.1.jar(LOW)*  
  Published Vulnerabilities(1):** *CVE-2020-9488*  
  **Description:** *CVE-2020-9488:* *“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:** Update to 2.13.2 or most recent version.
* ***Hibernate-validator-6.0.18.Final.jar(MEDIUM)*Published Vulnerabilities(1):** *CVE-2020-10693***Description:** *CVE-2020-10693:**“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 6.0.20.Final or most recent version.
* ***logback-core-1.2.3.jar(HIGH)*Published Vulnerabilities(2):** *CVE-2023-6378, CVE-2021-42550***Description:** *CVE-2023-6378: “A serialization vulnerability in logback receiver component part of logback version 1.4.11 allows an attacker to mount a Denial-Of-Service attack by sending poisoned data.”  
  CVE-2021-42550: “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 most recent version release.
* ***bcprov-jdk15on-1.46.jar(HIGH)*Published Vulnerabilities(19):** *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, CVE-2023-33202, CVE-2020-26939, CVE-2023-33201, CVE-2016-1000339, CVE-2015-7940, CVE-2018-5382, CVE-2013-1624, CVE-2016-1000346, CVE-2015-6644***Description:** *“Improper verification of cryptographic signature(CWE-347), cryptographic issues(CWE-310) and time and state(CWE-361) in version 1.55 and earlier. Prior to version 1.0.3, a discrepancy(CWE-203) allows “visible” private keys during TLS cipher suite using RSA key exchange. Before 1.66, timing issue within the EC math library can expose information about the private key(CWE-362). Before 1.73, a potential DoS issue occurs due to uncontrolled resource consumption(CWE-400).”***Solution:** Update to most recent version release.
* ***jackson-databind-2.10.2.jar(HIGH)*Published Vulnerabilities(6):** *CVE-2020-25649, CVE-2020-36518, CVE-2021-46877, CVE-2022-42003, CVE-2022-42004, CVE-2023-35116***Description:** *“A flaw found in FasterXML Jackson Databind allowed vulnerability to XML external entity (XXE) attacks due to not having entity expansion properly secured. Version 2.15.2 allows DoS attacks via crafted object that uses cyclic dependencies.”***Solution:** Update to the most recent version release.
* ***tomcat-embed-core-9.0.30.jar(CRITICAL\*), tomcat-embed-websocket-9.0.30.jar(CRITICAL\*)*Published Vulnerabilities(25 & 26):** *CVE-2020-1938 (CISA Known Exploited Vulnerability), CVE-2023-44487 (CISA Known Exploited Vulnerability), CVE-2020-8022 (tomcat-embed-websocket-9.0.30.jar)***Description:** *CVE-2020-1938: “Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an Apache JServ Protocol (AJP) Connector enabled by default that listened on all configured IP addresses. This vulnerability allowed returning arbitrary files from anywhere in the web application – processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application, then this along with the ability to process a file as a JSP, made remote code execution possible. “**CVE-2023-44487: “The HTTP/2 protocol allows a denial of service (server resource consumption) because request cancellation can reset many streams quickly.”  
  CVE-2020-8022: “An Incorrect Default Permissions vulnerability in the packaging of tomcat on multiple servers.”***Solution:** For CVE-2020-1938, apply oracle Critical Patch Updates. For CVE-2023-44487, Disable the HTTP/2 protocol on your web server by using the Registry Editor. Lastly, update to the most recent version release.
* ***spring-web-5.2.3.RELEASE.jar(CRITICAL\*), spring-core-5.2.3.RELEASE.jar(CRITICAL\*), spring-webmvc-5.2.3.RELEASE.jar(CRITICAL\*)*Published Vulnerabilities(12, 11, & 11):** *CVE-2022-22965 (CISA Known Exploited Vulnerability)***Description:** *“A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e. the default, it is not vulnerable to the exploit. However, the nature of the vulnerability is more general, and there may be other ways to exploit it.”***Solution:** Block both incoming and outgoing connections between the system and internet. Update to the most recent version release.
* ***spring-boot-starter-web-2.2.4.RELEASE.jar(CRITICAL), spring-boot-2.2.4.RELEASE.jar(CRITICAL)*Published Vulnerabilities(3):** *CVE-2023-20873, CVE-2022-27772, CVE-2023-20883***Description:** *CVE-2023-20873: “In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application that is deployed to Cloud Foundry could be susceptible to a security bypass.****”****CVE-2022-27772: “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.****”****CVE-2023-20883: “In Spring Boot versions 3.0.0 - 3.0.6, 2.7.0 - 2.7.11, 2.6.0 - 2.6.14, 2.5.0 - 2.5.14 and older unsupported versions, there is potential for a denial-of-service (DoS) attack if Spring MVC is used together with a reverse proxy cache.****”*Solution:** Update to most recent version release.
* ***snakeyaml.1.25.jar(CRITICAL)*Published Vulnerabilities(8):** *CVE-2022-1471, CVE-2017-18640, CVE-2022-25857, CVE-2022-38749, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854, CVE-2022-38750***Description:** *CVE-2022-38749, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854, CVE-2022-38750: “Using snakeYAML to parse untrusted YAML files may be vulnerable to Dos attacks.”  
  CVE-2022-25857: “The package org.yaml:snakeyaml from 0 to 1.31 are vulnerable to DoS due to missing nested depth limitation for collections***.”***CVE-2022-18640: “The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation.”  
  CVE-2022-1471: SnakeYaml’s Constructor() class does not restrict types which can be instantiated during deserialization. Deserializing yaml content provided by an attacker can lead to remote code execution.”***Solution:** For CVE-2022-18640, restricting aliases for collection can prevent vulnerability. Upgrade to most recent version release.

## Mitigation Plan

Nearly all the above vulnerabilities can be mitigated via updates. By keeping up to date on everything you can ensure any previous vulnerabilities that existed have been patched. Some exceptions to strictly updating are for Snakeyaml and restricting aliases for collection to avoid the CVE-2022-18640 vulnerability, blocking both incoming and outgoing connections between the system and internet for SpringBoot, and for Tomcat, disabling the HTTP/2 protocol on your web server via the Registry Editor, you can avoid the vulnerability CVE-2023-44487.