****

# 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** | **1/22/2023** | **Robin Robinson** |  |

## 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

Robin Robinson

## Interpreting Client Needs

Secure communications would be important to a company like this because it is a financial institution. This means that it would handle people’s sensitive information like social security numbers and bank account numbers for example. It makes no indication that international transaction is involved with this company. It would be very important to protect the sensitive data of the customers. Some threats to consider would be hacker trying to obtain customer data or accessing the system in other ways. When considering the use of open source libraries it important to review the code to be used and to only use what is necessary when using those libraries. When considering new web application technologies, it would be important to consider any new vulnerabilities that they could introduce and to plan on how to eliminate or mitigate those vulnerabilities.

## Areas of Security

In the context of Artemis Financial the most important areas to consider when planning security would be input validation , secure API interactions, and cryptography. This is because first being sure that the user is who they say they are is very important to the wrong people don’t get access to other users information. It is also important to make sure that what the user inputs is what it is suppose to be, for example if entering a number then you need to make sure that it is in fact a number as well as making sure that it is a valid number and restricting or blacklisting numbers or inputs that can not be used. Because Artemis Financial uses a RESTful API it is important to consider the security risks that could come from that. Then there is also the consideration of protecting the user’s sensitive data, for example passwords and other security measures, so making sure that a level of cryptography or encryption is used in important to user security.

## Manual Review

There seemed to be a lack of restrictions on what is imputed into the system, as well as many instances where the user can input anything they wish with access to the system. Another issue I saw was there was no obscurity to the SQL database and could be vulnerable to SQL injection.

## Static Testing

CVE-2016-1000352 - In the Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.

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.

CVE-2022-42004 - In FasterXML jackson-databind before 2.13.4, resource exhaustion can occur because of a lack of a check in BeanDeserializer.\_deserializeFromArray to prevent use of deeply nested arrays. An application is vulnerable only with certain customized choices for deserialization.

CVE-2021-44832 - Apache Log4j2 versions 2.0-beta7 through 2.17.0 (excluding security fix releases 2.3.2 and 2.12.4) are vulnerable to a remote code execution (RCE) attack when a configuration uses a JDBC Appender with a JNDI LDAP data source URI when an attacker has control of the target LDAP server. This issue is fixed by limiting JNDI data source names to the java protocol in Log4j2 versions 2.17.1, 2.12.4, and 2.3.2.

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.

CVE-2022-41854 - Those using Snakeyaml to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS). If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stack overflow. This effect may support a denial of service attack.

CVE-2022-27772 - \*\* UNSUPPORTED WHEN ASSIGNED \*\* 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. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer.

CVE-2022-22971 - In spring framework versions prior to 5.3.20+ , 5.2.22+ and old unsupported versions, application with a STOMP over WebSocket endpoint is vulnerable to a denial of service attack by an authenticated user.

CVE-2022-42252 - If Apache Tomcat 8.5.0 to 8.5.82, 9.0.0-M1 to 9.0.67, 10.0.0-M1 to 10.0.26 or 10.1.0-M1 to 10.1.0 was configured to ignore invalid HTTP headers via setting rejectIllegalHeader to false (the default for 8.5.x only), Tomcat did not reject a request containing an invalid Content-Length header making a request smuggling attack possible if Tomcat was located behind a reverse proxy that also failed to reject the request with the invalid header.

## Mitigation Plan

First and foremost I would have the comments and explanations added to the code as I don’t really know what is going on in the code being review, making it very hard to work through and find vulnerabilities. I would also make sure to restrict the packages being used to only what is necessary to what is needed. Adding parameters to the data being used would reduce the number of vulnerabilities.