

# 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** | **[Date]** | **Brandon Walters** |  |

## 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

Brandon Walters

## Interpreting Client Needs

The value of secure communications to Artemis Financial is the need to protect sensitive financial data for their users. The company does not state that it makes any international transactions, but it may be safe to assume due to their goal to modernize their operations. There are most definitely governmental restrictions to cover due to it being a business that handles important finances such as retirement plans for their customers. Some external threats that may be present now and in the immediate future are SQL injections to steal users' data such as financial records or even bank accounts and more.

Some modernization requirements: Open-source libraries should be utilized due to their secure nature. Evolving web application technologies should be used carefully to avoid any issues yet to be identified within those technologies, such as security risks.

## Areas of Security

One area of security that applies to Artemis financials web application is encryption use and vulnerabilities. My reasoning for this is due to the handling of client's sensitive financial data, which is likely to be attacked if the vulnerabilities and encryptions are not handled correctly. Furthermore, another area of security that applies to Artimes Finacial's web application is secure API interactions. My reasoning for secure API interactions being relevant to this software application to protect data transferred while handling sensitive financial data. Also, secure API interactions will also ensure that endpoints are only accessible by the applications users that have proper permissions. Finally, input validation also applies to the Artemis financial web application. My reasoning for why input validation is relevant to this software application is due to vulnerabilities such as SQL Injection. If the input is stored in or communicating with a database, the application will be exposed to many common SQL exploits that input validation may help negate

## Manual Review

One vulnerability I identified within the code base is within the “CRUDController” class. It is lacking input validation, specifically when handling the “business\_name” parameter which could leave this open to injection attacks. Furthermore, another vulnerability I identified was within the “DocData” class which creates a connection to a database, but the data is not encrypted which leaves the users sensitive data at risk to attackers.

## Static Testing

There was a total of thirteen vulnerable dependencies identified ranging from a severity of medium to critical. Many of these vulnerable dependencies have a severity of critical or high, which indicates that this application has a lot of potential attack vectors that may be exploited by malicious parties. Furthermore, the confidence of these tests is all at their highest, showing that many of the vulnerabilities are detrimental to the application and must not be ignored. ‘

* The “bcprov-jdk15on-1.46.jar” dependency has an issue detailed as: “does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks.”
* The “spring-boot-2.2.4.RELEASE.jar” dependency has an issue detailed as: “vulnerable to temporary directory hijacking.”
* The “logback-core-1.2.3.jar” dependency has an issue detailed as: “an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers..”
* The “log4j-api-2.12.1.jar” dependency has an issue detailed as: “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.”
* The “snakeyaml-1.25.jar” Dependency has an issue detailed as: “a maliciously crafted YAML file can cause the system to consume significant system resources. If parsing user input, this may be used as a denial of service vector..”
* The “jackson-databind-2.10.2.jar” dependency has a issue detailed as: “allows vulnerability to XML external entity (XXE) attacks.”
* The “tomcat-embed-core-9.0.30.jar” dependency has an issue detailed as: “a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner.”
* The “hibernate-validator-6.0.18.Final.jar” dependency has an issue detailed as: “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.”
* The “spring-web-5.2.3.RELEASE.jar” dependency has an issue detailed as: “suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data.”
* The “spring-beans-5.2.3.RELEASE.jar” dependency has an issue detailed as: “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.”
* The “spring-webmvc-5.2.3.RELEASE.jar” dependency has an issue detailed as: “it is possible for a user to provide malicious input to cause the insertion of additional log entries.”
* The “spring-context-5.2.3.RELEASE.jar” dependency has an issue detailed as: “the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless it is listed with both upper and lower case for the first character of the field.”
* The “spring-expression-5.2.3.RELEASE.jar” Dependency has an issue detailed as: “it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service condition.”

## Mitigation Plan

The first course of action I would take to mitigate the vulnerabilities within this code base is to add proper input validation when handling the business name within the “CRUDController” class. Next, I would ensure that connections to the data base an encrypted specifically in the “DocData” class in this case. There are many different routes that may be taken to best address the dependencies within this code base. Many of the spring related dependencies share a lot of the same vulnerabilities so let us first address those. The most obvious way to address these is documented in the dependency report, which is done by upgrading most or all the spring dependency versions. After analyzing the dependency check, the main action I would take is to upgrade the code base to more recent and secure versions that match the recommendations from the check. Lastly, I would also make sure that false positives are filtered from the dependency check tool because it may cause problems by identifying “issues” that need not to be fixed.