****

# 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** | **17 September 2023** | **Jessie Smith** |  |

## 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

Jessie Smith

## Interpreting Client Needs

* Secure communications are highly value to the company as it ensures protection of sensitive data to the customers. If communications were unsecure, there would be no reason for customers to want to use Artemis Financials’ application. Secure communication is a way for Artemis Financial to pull more customers by guaranteeing the protection of their data.
* If Artemis Financial wishes to engage in international transactions, we would need to ensure that the application can be protected based on policies and regulations in other areas across the borders of the US. Data encryption will help ensure that data is protected while communicating internationally.
* There are governmental restrictions and regulations that need to be considered not only if communicating data internationally, but also across other states of the US. Complying with governmental restrictions will help prevent any legal penalties and can help ensure that the application is secure.
* There are many external threats that can affect the application now and in the immediate future, including unauthorized access to users with malicious intentions, DOS attacks, injections of unsecure code, data breaches, etc. which all need to be considered when designing the security of the application. Prevention of these attacks will help secure the data of our users, including ourselves and customers.
* It is extremely important to consider modernization requirements as we would not want our application to use outdated libraries or software. Outdated libraries could leave room for loopholes and breaches of data. Updating the software and libraries used in the application will help avoid these risks.

## Areas of Security

* API: Artemis Financial currently uses an API. API security violations include DOS attacks, unauthorized access to sensitive data, etc. To prevent these attacks a firewall could be used to block unauthorized access, authentication and authorization could help prevent attackers, logging data related to the API could help catch and prevent any unwanted access, and restricting who has access to data could help further secure the API. API security is crucial to Artemis Financial as they are currently using an API.
* Cryptography: Cryptography needs to be considered for this application since there will be a lot of private data stored in the system. Private data could be stored encrypted so that if an unauthorized user gains access to the system, the data they have gained access to it is still encrypted and protected. I would also recommend encrypting user passwords so that if an attacker gains access to the stored passwords, they cannot read them and gain access to user data.
* Client/Server: A client/server approach could further ensure that data is protected by limiting the access of data, and protection of data sent over between the client and the server.
* Code quality: Code quality ensures secure code and that the code is following the best practices to ensure security. By building the application to initially follow secure coding policies we can further protect data from unwanted attacks. This also includes secure error handling, information and data should not be shared in an error code.

## Manual Review

* DocData.java: In the DocData.java file, “root” is used as the username and password in the method used to gain access to data through the data location, which is not recommended because it is easily guessable and could potentially risk data to be accessed by unwanted users. The same method also does not use any secure connection method while making connections to the data.

## Static Testing

Here is a screenshot of the dependency report:

A screenshot of a computer

Description automatically generated

* The first dependency in the report is The Bouncy Castle Crypto Package, which is labeled with high severity and 17 CVE counts. The dependencies related to this can allow invalid signature validation, decryption fail due to padding, invalid key use, etc. Solutions found in the dependency report include updating software and avoiding use of older software that is affected by these dependencies or disabling the affected software.
* The next dependency is Spring Boot, which is of critical severity and has a 3 CVE count. Older versions of Spring Boot are at risk of DoS attacks, security bypass, and temporary directory hijacking. The recommended solution found in the dependency report is to update Spring Boot from versions including: 3.0.0-3.0.6, 2.7.0-2.7.11, 2.6.0-2.6.14, and 2.5.0-2.5.14, all of which are outdated and unsupported.
* Next is the logback-core-module, which is of medium severity and has a CVE count of 1. The logback version 1.2.7 and previous versions, attackers can execute malicious code. Recommendation for a solution is to upgrade the software used and not user 1.2.7 and previous versions of the logback module.
* The next dependency is the Apache Log4j API, of critical severity and CVE count of 6, in which versions of the API allows attackers to gain control over certain properties of the API, improper certificate validation, man-in-the-middle attacks, etc. This dependency has already been corrected in newer versions of the API, so the API simply needs to be updated.
* Next is SnakeYAML which is of severity critical and has 10 CVE counts. SnakeYAML’s contrcutor() method does not limit types that can be deserialized, which can lead to remote code execution. Using SnakeYAML’s safeconstructor is recommended to prevent this in the dependency report. YAML parser 1.1 is also a dependency that can result in DoS attacks, the dependency report identifies that a security update has been made to avoid said attacks.
* The Jackson-databind-2.10.2 has also been identified as a threat in the dependency report and has a severity of high with 6 CVE counts. The dependency report shows that Jackson-databind software before 2.14.0 is at risk of DoS attacks, resource exhaustion, etc. Fixes to these issues are available in newer versions of the Jackson-databind, recommendation is to update the software.
* The core Tomcat implementation version 9.0.30 is affected as well which can allow smuggling attacks, interception of data, DoS attacks, etc. The Tomcat implementation simply needs to be updated to fix these issues.
* Hibernate’s Bean Validation version 6.0.18 has a severity of medium with 1 CVE count, which allows attackers to bypass input validation. This threat has been patched and the validator needs to be updated to avoid this risk.
* The rest of the dependencies in the report, including Spring Web, Spring Beans, Spring Web MVC, Spring Context, and Spring Expression Language, were false positives.

## Mitigation Plan

* The Bouncy Castle Crypto Package versions 1.55 and earlier has lost support from its provider as it has been proved to be secure and safe. Patches have been released in newer versions; we simply need to update the software to avoid potential threats.
* Sprint Boot 2.2.4 has proven to be unsafe and needs to be updated to 3.0.6+ or 2.7.11+ to prevent possible attacks.
* The logback-core-module version 1.2.3 allows attackers to execute malicious code. In order to prevent this, we need to update to versions 1.3.11 or 1.4.11.
* The Apache Log4J API 2.12.1 which gives access to attackers, improper certificate validation, etc., needs to be updated to one of the following versions: 2.17.1, 2.12.4, and 2.3.2, all of which have patched this issue.
* SnakeYAML 1.25 constructor class has flaws and can be avoided by using SnakeYAML’s safeconstructor or updating to 2.0+.
* Jackson-databind-2.10.2 is at risk of DoS attacks, resource exhaustion and more. FasterXML has released a patch for this on github which needs to be implemented in the project.
* Core Tomcat implementation version 9.0.30 is affected by dependencies and Apache has recommended upgrading to version 9.0.80 or higher to avoid potential threats. This will need to be completed to move forward with the project.
* Hibernate’s Bean Validator version 6.0.18 has bug that allows attackers to bypass input validation. This bug was patched in version 6.0.20, which will need to be implemented to avoid attacks.