****

# 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]** | **[Your name]** |  |

## 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

Mr. Nii Amatey Tagoe.

## Interpreting Client Needs

Primarily, one of the objectives of Global Rain is that “Security is everyone’s responsibility”. They believe securing PII or SPII is essential, and every individual must perform their best not to fall for malicious acts. The value of secure communication in this scenario is extremely relevant simply because Artemis Financial is handling loads of Sensitive Personally identifiable Information belonging to clients. Examples of SPIIs that Artemis Financials wants to keep secure may include Social Security numbers, financial account numbers, and Driver’s license numbers. The company faces potential threats such as phishing and DDos.

Secondly, in the case where Artemis Financial chooses to make international transactions, it will have to follow and meet government export regulations. To ensure that the company is less vulnerable while performing international transactions Artemis Financial needs to have a web-based application that hides or codes data to prevent unauthorized access, maintain privacy, and guarantee secrecy. This will stop malicious attacks such as ransomware.

Furthermore, there will be governmental restrictions on secure communication that Artemis Financial will need to follow if they decide to work with or intermingle with clients from other countries that have governments imposing strict data sovereignty laws or have restrictions on encryption technologies. These restrictions will affect the design of Artemis Financials’ web-based application.

The external threats present now may include cybersecurity vulnerabilities such as malware, ransomware, phishing attacks, and DDoS attacks. Some of the present threats might occur in the immediate future but Artemis Financial may face AI-powered attacks because of the rapid growth of AI innovations.

Finally, some modernization requirements that could be considered are the use of open-source libraries and evolving web application technologies. The use of evolving web application technologies such as frameworks and open-source libraries can be cost-effective but will leave the web application exposed to multiple cybersecurity vulnerabilities if updates are not done religiously.

## Areas of Security

After interpreting the needs of Artemis Financial and referring to the vulnerability assessment process flow, I believe the following areas of security should be a priority.

1. Input Validation

Input validation is an important area to focus on because Artemis Financial’s web-based application will need secure input and representation. The architecture of the software must have the latest structure set in place to ensure that all users go through authentication processes and are authorized to access certain data.

1. APIs, Clients, and Sever

Artemis Financial is currently using a RESTful web application programming interface to secure API interactions. The secure exchange of information allowed by the RESTful API is an important area of security to consider because Artemis Financial has the SPII of its customers in their home country and possibly other countries.

1. Code Quality

The use of secure coding methods, practices, and patterns is an important aspect to consider because it will ensure code readability, clarity, modularity, and most importantly in this case security.

1. Cryptography

Cryptography is an important feature to consider simply because it safeguards against vulnerabilities and ensures the use of encryption. Implementing cryptography into the code structure will enhance privacy, confidentially, and integrity of data.

## Manual Review

Artemis Financial’s web-based application is currently exposed to numerous vulnerabilities after a manual review was performed on the code. For instance, the code does not comprise any form of authentication or authorization for the API endpoints. This misshape can be exploited by attackers and could lead to a data breach. In the DocData class, linking to the database using default credentials (**root**/**root**) could pose security vulnerabilities, and the default usernames and passwords are recognized and could lead to easy exploitation. Also, there is inadequate error handling in this same class.

There were numerous vulnerable dependencies after a Dependency-check report was generated on the code. Most of the Spring frameworks need to be updated to ensure all vulnerabilities are resolved.

## Static Testing

After integrating a dependency-check plug-in into Maven, it generated an HTML file that exhibits all the potential vulnerabilities the code possesses. The PDF below shows the results of the dependency check.



The dependency-check report shows a maximum of 13 vulnerable dependencies with 131 vulnerabilities found.

Primarily, the dependency [tomcat-embed-websocket-9.0.30.jar](#l22_33157f6bc5bfd03380ebb5ac476db0600a0) **which is a Core Tomcat implementation has the highest count of Common vulnerabilities and exposure. There are 27 CVEs which 30 are evident. The confidence rating of this dependency is “highest” and the severity of at critical. The CVE ID is CVE-2020-1938 and suggests that when using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising.**

**The second dependency with a severity of “critical” is** [tomcat-embed-core-9.0.30.jar](#l20_ad32909314fe2ba02cec036434c0addd19b) with a CVE ID of [**CVE-2020-1938**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938)**. This CVE is described by the National Vulnerability Database as Apache Tomcat treats Apache JServ Protocol (AJP) connections as having higher trust than, for example, a similar HTTP connection. Attackers can cause serious damage if the mentioned connections are available to them. Other dependencies with a severity rating of “critical” are** [spring-webmvc-5.2.3.RELEASE.jar](#l19_745a62502023d2496b565b7fe102bb1ee22), [spring-web-5.2.3.RELEASE.jar](#l18_dd386a02e40b915ab400a3bf9f586d2dc4c), and [spring-core-5.2.3.RELEASE.jar](#l17_3734223040040e8c3fecd5faa3ae8a1ed6d). These dependencies have CVE counts of 11, 13, and 11 respectively. The CVE ID [**CVE-2022-22965**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965) **is described as a Spring MVC or Spring WebFlux application running on JDK 9+ and may be vulnerable to remote code execution (RCE) via data binding while the CVE ID of** [**CVE-2016-1000027**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000027) **is clarified as Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data by the NVD(National Vulnerability Database). Finally, the CVE ID of** [**CVE-2022-22965**](https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2022-22965) **is explained as Spring Framework, versions 5.2.x before 5.2.15 and versions 5.3.x before 5.3.7, a WebFlux application is vulnerable to a privilege escalation: by (re)creating the temporary storage directory, a locally authenticated malicious user can read or modify files that have been uploaded to the WebFlux application, or overwrite arbitrary files with multipart request data by the National Vulnerability Database.**

## Mitigation Plan

All the vulnerabilities must be resolved and the best way to do so is to formulate a mitigation plan. The most obvious mitigation plan is to constantly and regularly update dependencies and frameworks to their latest versions in order to stop malicious acts such as DDos and ransomware. Another plan of action is to ensure the use of Cryptography, a RESTful API that ensures authentication or authorization on its endpoints, the use of strong passwords and usernames is also effective.

Work Cited

[Common Vulnerabilities and Exposures](https://cve.mitre.org/cve/search_cve_list.html)

[National Vulnerability Database](https://nvd.nist.gov/vuln/search)