****

# 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** |
| --- | --- | --- | --- |
| **2.0** | **2/23/2023** | **Matthew Dunfee** |  |

## 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

Matthew Dunfee

## Interpreting Client Needs

Artemis Financial handles Savings, retirement, investment, and insurance data. This is all considered IP of the highest level. On top of that, they use APIs which further depredates the defense of their data. However, Global Rain is the only one listed in the scenario as working with agencies around the world, and since Artemis Financial doesn't list any target market listed the answer is no. In practice this should be investigated with the customer, management, and the software contract agreement and the same logic applies here as it does in the earlier-mentioned question. Only Global Rains' target market is listed. Any concern about this should likewise follow the same investigation.

The external threats that might be present now and in the immediate future that are of most concern are data breaches for PI and DDoS by competitors. Other threats can come from within some of the modernizations we see today are the use of open-source libraries. They are free and a great way to launch the project forward. With this type of open contribution model, careful monitoring needs to be in place in case new bad code is added to the libraries on updates. Often, we see companies take open-source software and rework and brand it for their only application and use. The evolving web application technologies are the driving force behind the need to update the software to keep things working, however, change brings risk.

## Areas of Security

Some areas of security concern are Client/server and Code Quality which both have to do with input sanitation which is the first line of defense against active hacking. If hackers were about to exploit a defect, Cryptography for the database would be paramount. These are often the big-ticket items that companies get sued over and thus lose the trust of their customers. API reinforcement is also a concern since DDoS can impact performance and impact the customer's assurance of the company's ability to safeguard their assets.

## Manual Review

Manual Review brought to like some vulnerabilities in the Project One code base. The main area of concern is that there is no sanitation, which was apparent from the review of CRUNController.java. This type of vulnerability could be paired with the lack of privilege restrictions on the database to inject and retrieve all kinds of customer PI. This would lead to a serious lawsuit. This oversight could have easily been avoided by using best coding practices. Lastly, the myDateTime.java file was missing encapsulation which while isn’t an immediate threat goes to show the careless coding.

## Static Testing

Static Testing brought a much larger swab of vulnerabilities. The dependency check found 131 known vulnerabilities. However, after parsing the logs twice, I only found 128. Then once added to a set those 128 vulnerabilities boiled down to 86 unique CVEs (the list will be at the end to make reading easier).

I will give a brief description and recommended solutions for the critical issue. Since there are over 80 issues, I am just looking at Major issues. The following 5 CVEs are a combination of retrieving data from niv.nict.gov and composing it with chat.openai.com throughout the extended dialog.

CVE-2020-1938:

Summary:

* Return arbitrary files from anywhere in the web application. Process any file in the web application as a JavaServer Pages (JSP), potentially leading to remote code execution.

Response:

* Disable the AJP Connector if not required.
* Upgrade Apache Tomcat to version 9.0.31, 8.5.51, or 7.0.100 or later to apply mitigation changes to the default AJP Connector configuration.
* Review and update configurations as necessary after the upgrade.

CVE-2023-20873:

Summary:

* Potentially allows unauthorized access to sensitive resources or functionality within the application.

Response:

* Upgrade Spring Boot to version 3.0.6+ if using 3.0.x or version 2.7.11+ if using 2.7.x.
* If using an older, unsupported version, upgrade to version 3.0.6+ or 2.7.11+.
* Apply the recommended mitigations provided by the Spring Boot team.

CVE-2022-22965:

Summary:

* Spring MVC or Spring WebFlux Remote Code Execution Vulnerability. The vulnerability described pertains to potential remote code execution (RCE) risk in Spring MVC or Spring WebFlux applications running on JDK 9 or later.

Response:

* Ensure that the application runs on Tomcat as a WAR deployment.
* If the application is deployed as a Spring Boot executable JAR, it is not vulnerable to the exploit.
* Review the deployment configuration to protect the application against the specific exploit.

CVE-2016-1000027:

           Summary:

* involves a potential remote code execution (RCE) risk when the framework is used for Java deserialization of untrusted data. Pivotal Spring Framework Remote Code Execution Vulnerability (CVE-2020-1935):

           Response:

* Review the implementation of the Pivotal Spring Framework within the product.
* If the product uses Java deserialization of untrusted data, assess the risk of remote code execution.
* Consider whether authentication is required for the vulnerable functionality.
* Follow the vendor's recommendations regarding the use of untrusted data and potential mitigations.

CVE-2022-1471:

           Summary:

* The vulnerability described pertains to SnakeYaml's Constructor() class, which does not impose restrictions on the types that can be instantiated during deserialization.

Response:

* Upgrade SnakeYaml to version 2.0 or later to address the vulnerability.
* When parsing untrusted content, use SnakeYaml's SafeConstructor to restrict deserialization and prevent remote code execution.
* Review and update the codebase to ensure that the recommended practices are followed for handling YAML content securely.

The HTML document provides supporting evidence for impacted libraries in the references section of each CVE. Outside of that nvd.nist.gov is an amazing hub for hyperlinks to advisories, solutions, and tools.

## Mitigation Plan

In summary, the mitigation plan should start with clarifying software contract agreements about government and overseas specifications. Secondly, look at critical and high-level dependency vulnerabilities from static testing. This is of high priority since the ramifications could cause code to be changed or libraries to be swapped out. Third the code should be updated to meet the bare minimum requirements for security with initiations, least privileges, and encryption of PI. Lastly, less pressing issues like input validation and encapsulation should be hardened to ensure nothing slips through.

**Citations:**

National Institute for Vocational Education and Training. (n.d.). Retrieved from https://niv.nict.gov

OpenAI. (n.d.). ChatGPT [Online]. Available: https://www.openai.com/chatgpt

**Notes**:

'CVE-2013-1624', 'CVE-2015-6644', 'CVE-2015-7940', 'CVE-2016-1000027', 'CVE-2016-1000338', 'CVE-2016-1000339', 'CVE-2016-1000341', 'CVE-2016-1000342', 'CVE-2016-1000343', 'CVE-2016-1000344', 'CVE-2016-1000345', 'CVE-2016-1000346', 'CVE-2017-13098', 'CVE-2017-18640', 'CVE-2018-5382',  'CVE-2019-17569', 'CVE-2020-0187', 'CVE-2020-10693', 'CVE-2020-11996', 'CVE-2020-13934', 'CVE-2020-13935', 'CVE-2020-13943', 'CVE-2020-15522', 'CVE-2020-17527', 'CVE-2020-1935', 'CVE-2020-1938', 'CVE-2020-25649', 'CVE-2020-26939', 'CVE-2020-36518', 'CVE-2020-5421', 'CVE-2020-8022', 'CVE-2020-9484', 'CVE-2021-22118', 'CVE-2021-24122', 'CVE-2021-25122', 'CVE-2021-25329', 'CVE-2021-30640', 'CVE-2021-33037', 'CVE-2021-41079', 'CVE-2021-42550', 'CVE-2021-43980', 'CVE-2022-1471', 'CVE-2022-22950', 'CVE-2022-22965', 'CVE-2022-22968', 'CVE-2022-22970', 'CVE-2022-22971', 'CVE-2022-25857', 'CVE-2022-27772', 'CVE-2022-29885', 'CVE-2022-34305', 'CVE-2022-35116', 'CVE-2022-38749', 'CVE-2022-38750', 'CVE-2022-38751', 'CVE-2022-38752', 'CVE-2022-41854', 'CVE-2022-42003', 'CVE-2022-42004', 'CVE-2022-42252', 'CVE-2022-44487', 'CVE-2022-46589', 'CVE-2023-20861', 'CVE-2023-20863', 'CVE-2023-20873', 'CVE-2023-20883', 'CVE-2023-28708', 'CVE-2023-33201', 'CVE-2023-33202', 'CVE-2023-35116', 'CVE-2023-41080', 'CVE-2023-42795', 'CVE-2023-44487', 'CVE-2023-45648', 'CVE-2023-46589', 'CVE-2023-6378', 'CVE-2024-21733', 'CVE-2024-22243', 'CVE-2016-1000346'