****

# 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** | **4/25/24** | **Jordon Burris** |  |

## 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

Jordon Burris

## Interpreting Client Needs

Artemis Financial is a financial institution that has a web API that would like to make sure it is secure. Artemis deals with monetary transactions so they will need to make sure that nobody can intercept the information being transmitted back and forth. Some of the ways to do this are to make sure everything is up to date and to make sure the web API has some sort of SSL or encryption. I would also like to see the implementation of some sort of authentication.

## Areas of Security

The first one in the flow chart would be, secure API interactions, we will need to make sure the API communicate securely and nobody can access that information except for the intended people. Next, would be input validation, this being because there will be monetary transaction and they will need to be validated. Last one I would like to point out would be secure coding, this will make sure all areas of the code is secure and free of vulnerabilities.

## Manual Review

From what I am seeing there is the root username and password hard coded into the code and I also am not seeing any form of authentication.

## Static Testing

* Bouncy Castle:
  + CVE-2016-1000352
  + CVE-2016-1000346
  + CVE-2016-1000345
  + CVE-2016-1000344
  + CVE-2016-1000343
  + CVE-2016-1000342
  + CVE-2016-1000341
  + CVE-2016-1000339
  + CVE-2016-1000338
  + CVE-2018-5382
  + CVE-2017-13098
  + CVE-2013-1624
    - Recommendation: Update to the newest version.
* Hibernate Validator:
  + CVE-2020-10693
    - Recommendation: Update to the newest version.
* Faster XML:
  + CVE-2023-35116
  + CVE-2021-46877
  + CVE-2022-42004
  + CVE-2022-42003
  + CVE-2020-36518
  + CVE-2020-25649
    - Recommendation: Update to the newest version.
* Apache Log4:
  + CVE-2021-44832
  + CVE-2021-45105
  + CVE-2021-45046
  + CVE-2021-44228
  + CVE-2020-9488
    - These are vulnerable to remote execution code and recursion attacks.
    - Recommendation: Update to newest version.
* Logback:
  + CVE-2023-6378
  + CVE-2021-42550
    - Does not protect against arbitrary code or denial of service attack.
    - Recommendation: Update to newest version.
* Spring Boot:
  + CVE-2023-20883
  + CVE-2023-20873
  + CVE-2022-27772
    - Does not protect against denial of service attack and vulnerable to temporary directory hijacking.
    - Recommendation: Update to newest version.
* Spring Framework:
  + Multiple vulnerabilities detected between spring boot, spring web, and spring webmvc.
    - Recommendation: Update entire spring framework.

## Mitigation Plan

* Update the outdate frameworks and dependencies for vulnerabilities.
* Get rid of the hard coded root log ins.
* Get an authentication.
* Use HTTPS.