****

# CS 305 Project One

**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** | **7/14/22** | **Dalton Soto** |  |

## Client



## Instructions

Deliver this completed vulnerability assessment report, identifying your findings of security vulnerabilities and articulating recommendations for next steps to remedy the issues you have found.

Respond to the five steps outlined below and include your findings. Replace the bracketed text on all pages with your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Dalton Soto

## 1. Interpreting Client Needs

Determine your client’s needs and potential threats and attacks associated with their application and software security requirements. Consider the following regarding how companies protect against external threats based on the scenario information:

* What is the value of secure communications to the company?
* Are there any international transactions that the company produces?
* Are there governmental restrictions about secure communications to consider?
* What external threats might be present now and in the immediate future?
* What are the “modernization” requirements that must be considered, such as the role of open source libraries and evolving web application technologies?

Artemis Financial will be holding very sensitive information since they handle funds and create plans, which could be something outsiders might want to get at. There should be high secure communication going on to the company and between the customers. They must follow any government regulations that have to do with transactions, communications, so that that they won’t be a security risk for the customer and the company. Seeking personal information of the client and the company would be the current and potential threats. If the API’s are not secure than it will be a threat, which can cause information leaks depending on the structure of the security. To prevent any logins that are not wanted there should be two-way authentication along with all communication being through HTTPS.

## 2. Areas of Security

Referring to the Vulnerability Assessment Process Flow Diagram, identify which areas of security are applicable to Artemis Financial’s software application. Justify your reasoning for why each area is relevant to the software application.

To create a level of security from any outside attacks there must be secure coding. Code errors like multiple login attempts need to be handled securely by the company’s end. The RESTful API will need to have a level of secure communication. The RESTful API will take user input so input validation is relevant and must be secured.

## 3. Manual Review

Continue working through the Vulnerability Assessment Process Flow Diagram. Identify all vulnerabilities in the code base by manually inspecting the code.

* System is vulnerable from outside attackers because requests are not validated
* No HTTPS, this should be used when sensitive info is shared
* No authentication for verifying

## 4. Static Testing

Run a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. Record the output from dependency check report. Include the following:

1. The names or vulnerability codes of the known vulnerabilities
2. A brief description and recommended solutions provided by the dependency check report
3. Attribution (if any) that documents how this vulnerability has been identified or documented previously

* bcprov-jdk15on-1.46.jar
  + CVE-2016-1000338
  + CVE-2016-1000342
  + CVE-2016-1000343
  + CVE-2016-1000344
  + CVE-2016-1000352
  + CVE-2016-1000341
  + CVE-2016-1000345
  + CVE-2017-13098
  + CVE-2020-15522
  + CVE-2020-0187
  + CVE-2016-1000339
  + CVE-2020-26939
  + CVE-2015-7940
  + CVE-2018-5382
  + CVE-2013-1624
  + CVE-2016-1000346
  + CVE-2015-6644
* classmate-1.5.1.jar
* hibernate-validator-6.0.18.Final.jar
  + CVE-2020-10693
* jackson-core-2.10.2.jar
* jackson-databind-2.10.2.jar
  + CVE-2020-25649
  + CVE-2020-36518
* jakarta.annotation-api-1.3.5.jar
  + CVE-2022-31569
* jakarta.validation-api-2.0.2.jar
* jboss-logging-3.4.1.Final.jar
* jul-to-slf4j-1.7.30.jar
* log4j-api-2.12.1.jar
  + CVE-2020-9488
* log4j-to-slf4j-2.12.1.jar
* logback-core-1.2.3.jar
  + CVE-2021-42550
* slf4j-api-1.7.30.jar
* snakeyaml-1.25.jar
  + CVE-2017-18640
* spring-boot-2.2.4.RELEASE.jar
  + CVE-2022-27772
* spring-core-5.2.3.RELEASE.jar
  + CVE-2022-22965
  + CVE-2021-22118
  + CVE-2020-5421
  + CVE-2022-22950
  + CVE-2022-22971
  + CVE-2022-22968
  + CVE-2022-22970
  + CVE-2021-22060
  + CVE-2021-22096
* spring-web-5.2.3.RELEASE.jar
  + CVE-2016-1000027
  + CVE-2022-22965
  + CVE-2021-22118
  + CVE-2020-5421
  + CVE-2022-22950
  + CVE-2022-22971
  + CVE-2022-22968
  + CVE-2022-22970
  + CVE-2021-22060
  + CVE-2021-22096
* tomcat-embed-core-9.0.30.jar
  + CVE-2020-1938
  + CVE-2020-11996
  + CVE-2020-13934
  + CVE-2020-13935
  + CVE-2020-17527
  + CVE-2021-25122
  + CVE-2021-41079
  + CVE-2022-29885
  + CVE-2020-9484
  + CVE-2021-25329
  + CVE-2021-30640
  + CVE-2022-34305
  + CVE-2021-24122
  + CVE-2021-33037
  + CVE-2019-17569
  + CVE-2020-1935
  + CVE-2020-13943
* tomcat-embed-el-9.0.30.jar
* tomcat-embed-websocket-9.0.30.jar
  + CVE-2020-1938
  + CVE-2020-8022
  + CVE-2020-11996
  + CVE-2020-13934
  + CVE-2020-13935
  + CVE-2020-17527
  + CVE-2021-25122
  + CVE-2021-41079
  + CVE-2022-29885
  + CVE-2020-9484
  + CVE-2021-25329
  + CVE-2021-30640
  + CVE-2022-34305
  + CVE-2021-24122
  + CVE-2021-33037
  + CVE-2019-17569
  + CVE-2020-1935
  + CVE-2020-13943

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application.

Switching to HTTPs for all direct communication is something that should be done to protect the information of clients and the company from outsiders. Moving request parameters to headers is something to be done. Within the hard coding database credentials any business names should be removed. There should be two-factor authentication to protect the user’s information and identity. Update all the dependencies from the dependency check above.