****

# 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** | **3-23-21** | **Alexis Scott** |  |

## 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

Alexis Scott

## 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?

Since personal finance is what Artemis Financial deals with it is important that all their client’s personal and financial information to be protected at all cost. If there were to be a breach in the company it would not only lose the trust of Artemis’ current clients, but it would also make it difficult for them to get many new clients in the future. A client’s trust is something hard to gain but easily lost.

It is possible for Artemis to have international transactions. With the possibility of having clients from other countries that will need to have the ability to manage and exchange foreign currency safely. To be able to due this they would need to take extra measures to ensure that all information is transferred safely and securely.

There are plenty governmental restrictions when it comes to communication over the internet. To be sure of the company’s current system it is important to know the governmental restrictions well and those restrictions may increase if you include international communication as well.

The external threats are the data that is stored in the systems and any communication being transferred over the internet. For the data that is being stored on the systems it would best to be worried about injection attacks where code can be injected into the user input boxes to extract data. It would also be good to make sure that data is properly limited behind credentials and have protocols to prevent any credentials from being wrongly accessed.

To modernize the company, I would update to the newest versions of libraries to get the security patches that are made available. I would also make sure that any of the third-party libraries that are being used are kept up to date.

## 2. Areas of Security

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

* **Input Validation:** I believe that Input validation would be one of the most important items to investigate since it is one of the simpler issues to identify and resolve. I must make sure to look for parameterized SQL.
* **Client/Server:** I will need to make sure that Artemis is properly protecting the data as it is being communicated from client to server. Also, I will need to ensure that permissions are properly enforced between client and server.
* **Code Error:** Checking to make sure that restricted data is not logged to the user in any error cases. These could be purposely trigger errors and if the error logs contain sensitive data it could possibly be a security breach.

## 3. Manual Review

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

* **Update to latest Spring-boot-starter-parent:** The parent version in pom.xml:8 should be updated from 2.2.4 to the current version. This ensures that we have the most recent dependency checking in infrastructure.
* **Dependency Versions:** Most of the dependency versions are not up to date. They were probably up to date when the application was first written but since then they are no longer. It would be good to update to latest dependency versions, this could improve the static testing results.

## 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

My testing has identified 7 vulnerable dependency with 34 vulnerabilities.

* **bcprov-jdk15on-1.46.jar** (Bouncy Castle Crypto package)
* **log4j-api-2.12.1.jar** : CVE-2020-9488 - Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.
* **snakeyaml-1.25.jar** : CVE-2017-18640- The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564.
* **jackson-databind-2.10.2.jar** : CVE-2020-25649 -A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.
* **tomcat-embed-core-9.0.30.jar** : CVE-2021-25329 -The fix for CVE-2020-9484 was incomplete. When using Apache Tomcat 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41, 8.5.0 to 8.5.61 or 7.0.0. to 7.0.107 with a configuration edge case that was highly unlikely to be used, the Tomcat instance was still vulnerable to CVE-2020-9494. Note that both the previously published prerequisites for CVE-2020-9484 and the previously published mitigations for CVE-2020-9484 also apply to this issue.
* **hibernate-validator-6.0.18.Final.jar :** CVE-2020-10693- A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enables invalid EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitation (escaping, stripping) controls that developers may have put in place when handling user-controlled data in error messages.
* **spring-core-5.2.3.RELEASE.jar** : CVE-2020-5421 -In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter.

## 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.

After all of my reviews and static testing, the first thing that I would do is update all of the library dependencies to the latest versions so that we are getting the most accurate information. I would also make sure that all of the account information is properly abstracted and restricted by permissions checks. Lastly, I would make sure that the database connections need to have certain authentication and are only made available to the people with certain permissions as well as make sure that any database username/passwords are abstracted and that they are not coded in as plain text.