****

# 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** | **11/14/2021** | **Oliver Milani** |  |

## 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

Oliver Milani

## 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?

The value of secure communications to the company is very high as secure communications are key to overall success. It helps sustain customer confidence in the company and assure their confidence that their privacy is secure.

There are international transactions that the company produces. Those internal transactions are stocks that are being bought and sold by users anywhere the web service can be accessed. The diversity in locations and governments that the system will allow access can lead to a lot of different issues.

There are a lot of governmental restrictions about secure communications. This application deals with a lot of stocks and money in general which has certain money transfer features that are at risk and tax features that are at risk. The system will have to gain access to each user’s location to make sure the rules of where they live are abided by. The United States government as well as the EU require companies like this one to have very strong security when it comes to user information as having this information compromised, such as financial account information and social security numbers, could ruin people’s lives.

Those restrictions help highlight the most possible external threats. The biggest external threats are any hackers that may be looking to steal that sensitive information. There are many ways for them to do this and our system will need to be able to handle any of those attacks. We will also need to be able to provide a service that helps notice if any users are being taken advantage of. This can be whether they are being taken advantage of via a keylogger or another type of virus that can see into their system.

Being web-based, this application will always need to evolve with all kinds of modernization requirements. Open-source libraries will increase the transfer of information which will increase how many people will be able to expose vulnerabilities. It is up to us to be able to constantly be ready to update our system and API when needed, and pay attention to any other groups that figure out ways to protect their systems.

## 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.

API

API is a given as the application is already implemented using REST API. The API will is vital as it is a big component to making sure we stay on top of any evolving technologies and possible attacks in the future. Failing to consistently update the API will allow for the team get behind and be vulnerable for a long time and possibly for the future.

Input Validation

Input Validation is a basic but still key feature to ensuring overall security. Every time there is input, we need to make sure it is secure as it can be one of the easiest things for a hacker to look at. It is important to have enough moments of validation to ensure that the correct people have access to what is being inputted. Making sure each user is authenticated this way is not only required by law but it any extra ways of authentication will help secure the user’s data and their faith in the company.

Client/Server

Alongside Input Validation, that client and server communication is a key area of security to check. In general, this is an area where there is some of the most vulnerability as it is the gate at which any potential hackers can get access. There are a lot of ways for the application to be vulnerable of the client/server framework is not strong such as hijacking, overloading servers with DDoS attacks, and other methods.

Cryptography

Within those client/server interactions, as a next level of security, it is important to make sure each packet of information is protected. The information being transferred is very sensitive and it is important that anyone who is not supposed to see that information is kept away. Through encryption, any and all information needs to be protected.

## 3. Manual Review

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

There were a few areas that I found that had some vulnerabilities. Within the customer class, in the DocData class, and the CRUDController file.

The customer class has a few parts where they are giving too much access to other classes. The accont\_number variable is private but the account\_balance variable is left public. This means that variable can be accessed elsewhere. Any variables that can store valuable information need to be private. The showInfo() method is also set to public. This raises a similar issue as it can be accessed to retrieve sensitive information which not only is bad for the user but does not follow the laws.

In the DocData file, there were many similarities to the customer class where there are methods that are public that should probably be private. The DocData class itself is public which provides no security. There is a String “id” that is set to private, but can easily be found by using the getID() method that is incorrectly set to public. This shows a general inability to consistently protect variable data as a similar issue can be found in the CRUDController file. The objects within that file can also be accessed easily through hijacking.

## 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 (Bouncy Castle Crypto Package)**

This vulnerability is one that can allow for hackers to get into the application easily. It can give them access to important information and have them not following regulations set by the government.

**hibernate-validator-6.0.18.Final.jar (Hibernate’s Bean Validation)**

Hibernate validator is referring to invalid expressions to return as valid in the interpolation processor. The recommendation for this fix is an updated version.

**jackson-databind-2.10.2.jar (Jackson Data-bind)**

Jackson Data-bind has to do with the FasterXML Jackson Databind. There is a vulnerability to external attacks and leaves important information at risk.

**snakeyaml-1.25.jar (YAML Parser)**

SnakeYAML, like jackson data-bind, also has to do with entity expansion and will allow a user to cause a denial-of-service attack. Recommendation for this is to update to a newest version and the issue should be fixed.

**log4j-api-2.12.1.jar (Apache Log4j API)**

This vulnerability is referring to an improper certificate validation. It also refers to a host mismatch in the SMTP appender. The recommendation here is also to update the API and it should be fixed.

**spring-aop-5.2.3.RELEASE.jar (Spring Framework)**

This vulnerability refers to the Spring Framework Reflected File Download vulnerability. This means that there are ways to bypass into the application. Update will solve this issue.

**tomcat-embed-core-9.0.30.jar (Tomcat Implementation)**

This vulnerability has some high issue vulnerabilities. There is an ability for hackers to change files. It also allows for outside programs to access classes within this one and use their methods. An update should fix the issues.

## 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.

There are quite a few vulnerabilities in the Artemis software right now. A lot of them stem from inconsistent code that allows access to sensitive information, and a lot of the other ones require updated API and other code to make sure they are fixed. The issues themselves all revolve around giving potential hackers access to objects and methods within the classes. While each issue needs to be taken care of one by one, the fact that they all stem from similar issues should help make the mitigating process a lot easier. A lot of the code needs to be combed over and double checked that objects and methods are protected, and the APIs that are used need to be updated.