****

# 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** | **11/11/2023** | **Nicholas Huffines** | **First Draft** |

## 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

Nicholas Huffines

## Interpreting Client Needs

First and foremost, the client needs to comply with the Gramm-Leach-Bliley Act. This means we need to keep the following things in mind. The financial privacy rule, the safeguards rule, and pretexting protections. Of these, our domain is the safeguards rule and pretexting protections. Our client needs us to modernize the software’s security features, improve data security, and keep client data private. As a financial institution, our client will also need maximum uptime to keep their clients happy. Our job is to ensure the security and stability of their software to prevent data leakage or hostile takeover.

## Areas of Security

The areas of security we should focus on are Input Validation, API Security, Cryptography, and Encapsulation. First is Input Validation. We need to make sure that attacks, such as SQL Injection, are unable to affect our client by making sure we sanitize our input. Second, our client has a RESTful API, which means we need to make sure the API dependencies are up to date, and any functions are secure. Third is Cryptography, which is an important part of keeping client account numbers, passwords, and other identifiables from being stolen. Finally, Encapsulation will make sure we have secured any data structures being used in the software. The software will eventually use some structure to keep every account and customer’s data in.

## Manual Review

After inspection, there are a few things that can be done. First is adding some form of access control. Whether a dependency is used or we provide our own, we should implement role based access control. Second, is input validation. The greeting controller class gets a parameter without any kind of validation. We should ensure some validation happens for any login that may be dependency driven. Third is encapsulation. In the customer class, the account balance is not a private variable, which would leave it open to manipulation from anyone that gained unauthorized access. Fourth, there is no obvious use of encryption to protect each customer’s account number.

## Static Testing

Below is an image of the static dependency check.

A screenshot of a computer

Description automatically generated

The first dependency on the list deals with bouncy castle and is primarily of cryptographic vulnerabilities. The next few have to deal with improper input validation and DoS attacks. After that, most deal with improper input validation or broken access control which lead to things like remote code execution, DoS, or code injection, which could lead to hijacking or exposure of sensitive information.

## Mitigation Plan

In order to mitigate the potential for a successful attack against the client, we need to focus on four things. First, we need to ensure that every dependency used is updated to its most recent version. Many of the CVEs associated with the dependency report are listed for out of date versions. Updating the project dependencies will cover a lot of ground. Second, we need to implement some form of role based access control. Either we can code our own, or we can use what is provided in Spring. This will help ensure that if for some reason the updates do not work, that we are still protected from the listed vulnerabilities. Third, is making sure we validate every piece of input in the software. If there is any portion of code that gets a parameter or requires input, we need to make sure we are getting that input in a secure fashion. Finally, we need to ensure the software is using some form of encryption. Whether it is from the dependencies or we use encryption directly, we need to ensure the client can protect client information.