****

# 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** | **11/08** | **Hunter Richards** | **Adjusted the ‘Interpreting Client Needs’ section** |
| **1.1** | **11/09** | **Hunter Richards** | **Finished interpreting client needs and started Areas of Security** |
| **1.2** | **11/10** | **Hunter Richards** | **Finished Areas of Security section of the document** |
| **1.3** | **11/10-11/11** | **Hunter Richards** | **Finished the Manual Review section of the document** |
| **1.4** | **11/11** | **Hunter Richards** | **Ran the static report and embedded the document within the report.** |
| **1.5** | **11/12** | **Hunter Richards** | **Finished the Mitigation Plan section of the document** |
| **1.6** | **11/13** | **Hunter Richards** | **Final touches and adjustments prior to submission.** |

## 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

Hunter Richards

## Interpreting Client Needs

1. What is the value of secure communications to the company?
   1. As this company focuses on finance, secure communication is integral to the operation of the system. If data were to be mishandled, then sensitive financial information could be leaked, or the software itself could be manipulated through unauthorize access.
2. Does the company make any international transactions?
   1. This is unclear and will need to be verified by the company
3. Are there governmental restrictions about secure communications to consider?
   1. The scenario isn’t clear where the client is located. However, if the client is in the US, there will be government restrictions they will need to abide by. There are federal and state laws which have direct rules on privacy of data related to personal identifiable information. There must be adequate risk assessments of any software with this type of information to prevent leaks and malicious activity/entities.
4. What external threats might be present now and in the immediate future?
   1. Any entity which seeks to gain personal identifiable information about Artemis Financials’ clients would be a potential present or immediate future threat. Any software rework is going to invite outside sources to inspect/exploit any points of weakness, especially those which involve sensitive information. Foreign entities may also probe for vulnerabilities depending on the clients of Artemis Financials (if noteworthy). These vulnerabilities could lead to illicit activity such as identity theft, ransomware, or malware injections.
5. What are the modernization requirements that you must consider? For example:
   1. The role of open-source libraries
      1. Open-source software and libraries are open about their potential vulnerabilities. This makes it far easier to track and account for security risks when using open-source applications. Open source also tends to have a larger community of support which leads to greater chances of vulnerability detection and alert (if a new detection occurs).
   2. Evolving web application technologies
      1. New versions of libraries and applications often will patch bugs and vulnerabilities within the codebase. Therefore, it’s important to keep versions up-to-date, or, at the very least, account for the vulnerabilities of older patch versions. Likewise, infrastructure and protocol changes/upgrades can help alleviate and/or resolve many types of security concerns. Overall, web applications need to adapt to new vulnerabilities and bugs, which could occur anytime, throughout an application’s lifecycle.

## Areas of Security

The four primary areas of security that would be important to Artemis Financials’ RESTful web application include: Cryptography, APIs, Input Validation, and Code Quality.

Cryptography

* Cryptography is fundamental to secure connections between two endpoints. Any data that could by hijacked, whether through MITM (man-in-the-middle) or some other form of attack, needs to be encrypted and secured so that no PII (personal identifiable information) data falls into the wrong hands. There are many state laws which heavily fine those who negligently handle sensitive data in regard to data breaches (i.e. lack of encryption). This, of course, doesn’t account for the major reputational harm it would cause for any organization; especially so for a financial institution such as Artemis Financial.

APIs

* As Artemis Financial is requesting Global Rain to assist in their RESTful web application, which in of itself is an API, this is an implicit area of security that must be well looked after. Poorly written AI’s can lead to vulnerabilities such as weak authentication, excessive data exposure, data injections, and more. All these examples would devastate a financial institution which desires top-level security for its clients. Thus, it is essential that the API be well written and account for the aforementioned vulnerabilities.

Input Validation

* Input validation is crucial when it comes to the creation of any RESTful framework. Maliciously crafted queries could cause API frameworks to run unexpectedly. Two examples of attacks which could be encountered include integer overflow and directory traversal. If input is validated, preferably more than once and early on, then these types of attacks can be drastically mitigated.

Code Quality

* Code quality is a superset of Input Validation and APIs. Thus, Code Quality must be maintained throughout the codebase to ensure software security. This means writing code which focuses on minimizing potential exploit paths (i.e., understanding various execution paths which may have otherwise gone unnoticed), and focusing on data security, especially regarding sensitive client data.

## Manual Review

As the code is currently quite fragmented due to the lack of implementation, it’s difficult to identify key vulnerabilities. However, there are three instances of vulnerabilities I immediately see as potential issues. These issues are present within the DocData class file. The read\_document method has a try/catch error handling, with both portions harboring exploitable code. The connection uses a username/password for the connection /test database connection. This shouldn’t be used with plain text, but rather should be encrypted with some sort of authentication token. Furthermore, the catch statement prints the stack trace which shouldn’t exist when the code is publicly available. Printing the stack trace should be used only for debugging the program as allowing it to be public could cause issues with data leaking. Lastly, one other issue with this program is the customer class. The customer information should be encrypted on a database and then pulled from said database. The class shouldn’t be sitting, unencrypted in memory, without some form of authentication.

## Static Testing

Graphical user interface, text, application, email

Description automatically generated

### Full Report



**The report has far too many listed vulnerability codes, names, descriptions, and attributions to feasibly list in this document. Please review the embedded static report for full details.**

To summarize, many of the detected vulnerabilities and related to outdated libraries, libraries with improper input validation/sanitization, and improper certificate validation (which should be resolved prior to deployment).

## Mitigation Plan

* DocData.java class file –
  + The ‘try’ block issue:
    - The issue within the try block within the read\_document method in the DocData class can be mitigated by modifying the connection to the MySQL database to have an encrypted authentication method versus a public facing request for username and password to connect to the database.
  + The ‘catch’ block issue:
    - The issue within the catch block within the read\_document method in the DocData class can be mitigated by modifying the block so that printing the stack trace (which could be used to leak sensitive information) is stored privately versus publicly (or removed entirely).
* Customer.java class file –
  + The data within this file needs to be stored in an encrypted database and pulled via some form of authentication to the program, not running unencrypted within memory. This class contains sensitive information which could cause potential data breach issues if left as is.
* Static Report file –
  + There were many vulnerabilities reported, with the primary focus on the bcprov-jdk15on-1.46.jar library, the tomcat embedded code for Apache, and the Spring Framework. These libraries list many various vulnerabilities, some critical and some with many unwanted minor points of exploitation (upwards of twenty). I recommend that all these libraries be updated and then scanned for further issues.
  + If we include proper data validation/sanitization when managing data between endpoints, we should be able to mitigate many of the reported exploits.
  + We need to include proper authentication when accessing data between endpoints. This can mitigate multiple vulnerabilities detected within the report by ensuring that only authenticated requests are granted access to the application.