****

# 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** | **September 16, 2023** | **Christopher Sharp** |  |

## 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

Christopher Sharp

## Interpreting Client Needs

The client Artemis Financial is a consulting company that provides individualized financial plans for their customers. These plans include savings, retirement, investments, and insurance products. Now a company that handles sensitive customer information security is pushed to the forefront to protect that data from attacks that could cause damage to the customers and business itself. Therefore, secure communication is paramount to protect this sensitive information from unauthorized access.

With Artemis Financial being a financial consulting company, the likelihood of international transactions is a good bet. Therefore, consideration must be taken that all laws and regulations are followed, wherever those transactions may take place. As a financial institution, Artemis Financial must follow many government regulations, domestically and abroad. Some of those include:

**The Sarbanes-Oxley Act**: This act “establishes requirements for the secure storage and management of corporate-facing electronic financial records, including the monitoring, logging, and auditing of certain activities” (Evans, 2023).

**The Gramm-Leach-Bliley Act:** This act “regulates the collection, safekeeping, and use of private financial information. Additionally, GLBA requires covered companies and entities to be transparent with respect to information-sharing practices, which includes granting customers the right to opt out of the sharing of their data and information with third parties” (Evans, 2023).

**Payment Card Industry Data Security Standard:** This instructs the financial institutions to limit the access of the cardholder information to as few employees as possible and to implement controls and tracking of the account activity (Evans, 2023).

These are just some of the regulations listed, including US statutes, Congressional Committees, and Federal Regulatory Agencies. There are several regulatory restrictions internationally depending on where the company operates.

The type of data that Artemis Financial has access to makes it a target every second of the day for attacks from unscrupulous actors. These attacks can be phishing, malware, ransomware, DDoS Attacks, and direct attacks on the software itself. All these aim to try to gain unauthorized access to sensitive financial information that could cause irreparable harm to both business and customer.

When considering the modernization of the software, we must be on our toes. The software landscape is ever-changing. From the use of open-source libraries, we must ensure that it is not abandonware and is continuously updated so that any security vulnerabilities are patched promptly. Also, to consider, is that attacks on software are also ever-changing. That means what might have been relevant and worked in the past may not be suitable and as secure today.

## Areas of Security

While keeping the items in mind that were touched on in the previous section, we can identify several areas of the software that could be at risk of attack. They include the following:

**Input Validation:** This aims to validate all data that is input is within the acceptable range and format. This will help to identify and eliminate malicious attacks that could be executed by sending out of range data that would allow unmitigated access to protected data.

**Secure API:** Due to the use of a RESTful API, it is with utmost importance that the API is secure and that it functions correctly. Thus, preventing unauthorized access to functions of the API. This can be accomplished through encryption of the API, authentication of the users, and controlling the access to the many functions of the API.

**Cryptography:** With Artemis Financial being a financial institution many guidelines and laws must be followed to ensure that the customer data is always protected. ”According to the Federal Financial Institutions Examination Council, Financial institutions should employ encryption to mitigate the risk of disclosure or alteration of sensitive information in storage and transit” (Probasco).

**Code Error:** Error handling must be handled in accordance with specified guidelines to ensure that the accidental leak of sensitive information does not happen. An example is covered in the book, *Iron-Clad Java*, where the error production is rerouted so that sensitive information is not freely displayed to the user. They are instead taken to a cleaned version of the output. (Manico & Detlefsen, 2015)

**Code Quality:** Best practices must be followed to ensure the code is robust and secure. This includes using established design patterns and testing methods that analyze the code base and dependencies for known vulnerabilities. Many of these guidelines are present in guidelines presented and distributed by OWASP, which include “properly free memory after functions, restrict users from generating new code, and reviewing all third-party code to determine safe functionality (OWASP Foundation).

**Encapsulation:** By using encapsulation, we can obscure some of the data and ensure that only object members and their functions can access it. This builds on the Principle of least Privilege (PLOP) that Jerome Saltzer touches on.

## Manual Review

By going one by one through the files in the code base many vulnerabilities have been found.

**CRUD.java:**

* No input validation is present for the constructors
* The absence of access control for the getter methods

**CRUDController.java**

* No input validation for the method
* Possible exposure of sensitive data due to lack of encapsulation

**Customer.java**

* The account balance variable is exposed
* The showInfo() method does not have user authentication allowing access to all users
* No input validation thus injection attacks are possible

**DocData.java**

* Exposure of database username and password possible through error handling
* No input validation for the method
* Database closure is missing

**Greeting.java**

* Missing access control for the get methods
* No input validation for the constructor

**GreetingController.java**

* No input validation is present for the method

**myDateTime.java**

* No input validation for the method possibly allowing incorrect inputs
* No state set for the variables (public or private)

**RestServiceApplication.java**

* Due to the use of the args argument, sanitation must occur

## Static Testing

A screenshot of a computer

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Dependency | Severity | CVE Count | Evidence Count |
| [bcprov-jdk15on-1.46.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l2_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7) | High | 17 | 37 |
| [spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l3_225a4fd31156c254e3bb92adb42ee8c6de812714) | Critical | 3 | 32 |
| [logback-core-1.2.3.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l4_864344400c3d4d92dfeb0a305dc87d953677c03c) | Medium | 1 | 32 |
| [log4j-api-2.12.1.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l5_a55e6d987f50a515c9260b0451b4fa217dc539cb) | Critical | 5 | 46 |
| [snakeyaml-1.25.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l8_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421) | Critical | 10 | 28 |
| [jackson-databind-2.10.2.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l9_0528de95f198afafbcfb0c09d2e43b6e0ea663ec) | High | 6 | 39 |
| [tomcat-embed-core-9.0.30.jar](file:///C:\\Users\\chris\\eclipse-workspace\\rest-service\\target\\dependency-check-report.html" \l "l13_ad32909314fe2ba02cec036434c0addd19bcc580) | Critical | 22 | 39 |
| [hibernate-validator-6.0.18.Final.jar](file:///C:\\Users\\chris\\eclipse-workspace\\rest-service\\target\\dependency-check-report.html" \l "l16_7fd00bcd87e14b6ba66279282ef15efa30dd2492) | Medium | 1 | 36 |
| [spring-web-5.2.3.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l19_dd386a02e40b915ab400a3bf9f586d2dc4c0852c) | High | 4 | 28 |
| [spring-beans-5.2.3.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l20_0250c8c641433dc06b1b44e4563fa08a2fbf8954) | High | 1 | 28 |
| [spring-webmvc-5.2.3.RELEASE.jar](file:///C:\\Users\\chris\\eclipse-workspace\\rest-service\\target\\dependency-check-report.html" \l "l21_745a62502023d2496b565b7fe102bb1ee229d6b7) | Medium | 1 | 30 |
| [spring-context-5.2.3.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l22_7750c95c96c7a1885c8b1b503ba915bc33ca579a) | Medium | 1 | 28 |
| [spring-expression-5.2.3.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l23_d0c6bb10758805b2153c589686b8045554bfac2d) | Medium | 3 | 30 |

By looking at the outputs from the dependency check, we see that there are many vulnerabilities that are present in the software that must be addressed so that the software is as secure as possible.

## Mitigation Plan

As we can see from the code review and the static testing performed, several issues must be addressed to ensure the code base is secured from malicious attacks. First, we look at the results from the code review. The present issues are easily addressable by correcting the mistakes and following the guidelines set forth to ensure that the code is written securely. Next, we look at the results from the static testing. This shows us that several vulnerabilities are present. To correct these issues, we will look at ways to mitigate them. The first step would be to look for any updates to the dependencies to see if those vulnerabilities have been corrected.

* [bcprov-jdk15on-1.46.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l2_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7) – update to version 1.70
* [spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l3_225a4fd31156c254e3bb92adb42ee8c6de812714) – update to version 6.0.12
* [logback-core-1.2.3.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l4_864344400c3d4d92dfeb0a305dc87d953677c03c) – update to version 1.4.11
* [log4j-api-2.12.1.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l5_a55e6d987f50a515c9260b0451b4fa217dc539cb) - update to version 2.20.0
* [snakeyaml-1.25.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l8_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421) - update to version 2.2
* [jackson-databind-2.10.2.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l9_0528de95f198afafbcfb0c09d2e43b6e0ea663ec) - update to version 2.15.2
* [tomcat-embed-core-9.0.30.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l13_ad32909314fe2ba02cec036434c0addd19bcc580) - update to version 11.0.0-M11
* [hibernate-validator-6.0.18.Final.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l16_7fd00bcd87e14b6ba66279282ef15efa30dd2492) – update to version 8.0.1.Final
* [spring-web-5.2.3.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l19_dd386a02e40b915ab400a3bf9f586d2dc4c0852c) – update to version 6.0.12
* [spring-beans-5.2.3.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l20_0250c8c641433dc06b1b44e4563fa08a2fbf8954) - update to version 6.0.12
* [spring-webmvc-5.2.3.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l21_745a62502023d2496b565b7fe102bb1ee229d6b7) – update to version 6.0.12
* [spring-context-5.2.3.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l22_7750c95c96c7a1885c8b1b503ba915bc33ca579a) – update to version 6.0.12
* [spring-expression-5.2.3.RELEASE.jar](file:///C:\Users\chris\eclipse-workspace\rest-service\target\dependency-check-report.html#l23_d0c6bb10758805b2153c589686b8045554bfac2d) – update to version 6.0.12

\*All updated versions of the dependencies were located at https://mvnrepository.com/

As you can see, many of the dependencies have had updates published to them and the vulnerabilities have been patched in many of the updates. Thus, the software as a whole will become much more secure by updating the code base using the best secure and safe practices and updating all the dependencies that the code base requires.

**References**

Evans, L. (2023, August 7). *Financial Institutions Regulatory Checklist*. Arctic Wolf. https://arcticwolf.com/resources/blog/a-simplified-regulatory-checklist-for-financial-institutions/

Manico, J., & Detlefsen, A. (2015). *Iron-clad Java: Building secure web applications*. McGraw-Hill Education.

Probasco, L. (n.d.). Encryption requirements for Banks &amp; Financial Services. Encryption Requirements for Banks &amp; Financial Services. https://info.townsendsecurity.com/encryption-requirements-for-banks-financial-services

OWASP SCP quick reference guide V1 - OWASP Foundation. (n.d.). https://owasp.org/www-pdf-archive/OWASP\_SCP\_Quick\_Reference\_Guide\_v1.pdf