****

# 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** | **Kenneth Gollaher** | **Revised: Client Needs Revised: Areas of Security**  **Revised: Manual Review**  **Revised: Static Testing**  **Revised: Mitigation Plan** |

## 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

Kenneth Gollaher

## 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?

A company that develops financial plans for their customers, security is one of the most important aspects of their system. Artemis Financial sees great value for secure communications and their highest priority is to safeguard their data as well as their customer from outside threats. When we use the term “secure communication,” I am insinuating that the company is using a technique to send and receive data between the company’s server and the client. Having secure communications will allow for the data to be transferred without being tampered with. Our client provides global transactions, so the data will need to be accessed globally without affecting secure communications. Securing data is not only a concern for the civilian sector, but also a concern and high priority for government agencies. There shouldn’t be any governmental restrictions for data security, but it is good to review the government policies in relation to secure communications. There will always be an external threat when it comes to software, but with available security techniques in place, it reduces the risk significantly. A common form of unauthorized access is from misconfigured authentication mechanisms, such as weak passwords on the client side. SQL injections are a concern now and will be in the future. A modernized requirement, such as open-source libraries, should be considered due to their flexibility, fewer bugs, and a way to fix them quickly. Open source helps detect bugs early and be more secure. Also, they usually have a lower cost.

## 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.

Looking over the provided Vulnerability Assessment Process Flow Diagram (VAPFD), the areas of security that would be useful to analyze and implement are Input Validation, API’s, Cryptography, and Encapsulation. For input Validation, unauthorized access can be attempted through HTTP requests, cookies, or injections. APIs will be another level to implement security due to its sending and receiving data through an HTTP request. Our client is currently using a RESTful API so there is a need to secure these HTTP requests from unauthorized access. Cryptography is another level applicable level of security for our client. Since they collect the customers sensitive information, it is critical to protect the customers data from being read while being sent and received across networks that may have vulnerabilities. Lastly, encapsulation is another critical level of security for our client. Ensuring that the data structures have no vulnerabilities is avoiding extensive damage that includes modified, destroyed, or stolen data.

## 3. Manual Review

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

I notice an issue that will allow unauthorized access if a user bypasses the input validation. Access control can be compromised, and data can be stolen or destroyed through the URL. Requests sent aren’t verified which allows vulnerabilities. In the CRUD controller class, I notice the business name /string is sent and can create a vulnerability. Lastly, user authentication. I don’t see a location where it verifies the users credentials.

## 4. Static Testing

Run a dependency check on Artemis Financials 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

Text

Description automatically generated

In my dependency check report, it resulted in 9 dependencies having vulnerabilities with 60 total vulnerabilities. Below are the 9 listed dependencies on the report.

\*\*Each dependency below is minimized for better readability\*\*

## 1. bcprov-jdk15on-1.46.jar (Severity: Unknown)

**Recommendation:** Update to the latest version

**Description:** The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7.

**Published Vulnerabilities:**

[**CVE-2013-1624**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624)

**CVE-2015-6644**

[**CVE-2015-7940**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7940)

[**CVE-2016-1000338**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338)

[**CVE-2016-1000339**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000339)

[**CVE-2016-1000341**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000341)

[**CVE-2016-1000342**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342)

[**CVE-2016-1000343**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000343)

[**CVE-2016-1000344**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000344)

[**CVE-2016-1000345**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000345)

[**CVE-2016-1000346**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000346)

[**CVE-2016-1000352**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000352)

[**CVE-2017-13098**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-13098)

[**CVE-2018-1000613**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-1000613)

[**CVE-2018-5382**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-5382)

[**CVE-2020-15522**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-15522)

**CVE-2020-26939**

## 2. hibernate-validator-6.0.18.Final.jar (Severity: Medium)

**Recommendation:** Update to the latest version

**Description:** Hibernate's Bean Validation (JSR-380) reference implementation.

**Published Vulnerabilities:**

[**CVE-2020-10693**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=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.**

## 3. jackson-databind-2.10.2.jar (Severity: High)

**Recommendation:** Update to the latest version

**Description:** General data-binding functionality for Jackson: works on core streaming API

**Published Vulnerabilities:**

[**CVE-2020-25649**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649)

## 4. log4j-api-2.12.1.jar (Severity: Low)

**Recommendation:** Update to the latest version

**Description:** The Apache Log4j API

**Published Vulnerabilities:**

[**CVE-2020-9488**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488)

## 5. snakeyaml-1.25.jar (Severity: High)

**Recommendation:** Update to the latest version

**Description:** YAML 1.1 parser and emitter for Java

**Published Vulnerabilities:**

[**CVE-2017-18640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640)

## 6. spring-aop-5.2.3.RELEASE.jar (Severity: High)

**Recommendation:** Update to the latest version

**Description:** Spring AOP

**Published Vulnerabilities:**

[**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421)

[**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096)

[**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118)

## 7. spring-core-5.2.3.RELEASE.jar (Severity: High)

**Recommendation:** Update to the latest version

**Description:** Spring Core

**Published Vulnerabilities:**

[**CVE-2020-5421**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421)

[**CVE-2021-22096**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22096)

[**CVE-2021-22118**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118)

## 8. tomcat-embed-core-9.0.30.jar (Severity: Critical)

**Recommendation:** Update to the latest tomcat version

**Description:** Core Tomcat implementation

**Published Vulnerabilities:**

[**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569)

[**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996)

[**CVE-2020-13934**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934)

[**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935)

[**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943)

[**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527)

[**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935)

[**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938)

[**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484)

[**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122)

[**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122)

[**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329)

[**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640)

[**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037)

[**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079)

[**CVE-2021-42340**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42340)

## 9. tomcat-embed-websocket-9.0.30.jar (Severity: Critical)

**Recommendation:** Update to the latest tomcat version

**Description:** Core Tomcat implementation

**Published Vulnerabilities:**

[**CVE-2019-17569**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569)

[**CVE-2020-11996**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996)

[**CVE-2020-13934**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934)

[**CVE-2020-13935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935)

[**CVE-2020-13943**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943)

[**CVE-2020-17527**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527)

[**CVE-2020-1935**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935)

[**CVE-2020-1938**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938)

[**CVE-2020-8022**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-8022)

[**CVE-2020-9484**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484)

[**CVE-2021-24122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122)

[**CVE-2021-25122**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122)

[**CVE-2021-25329**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329)

[**CVE-2021-30640**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640)

[**CVE-2021-33037**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037)

[**CVE-2021-41079**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-41079)

[**CVE-2021-42340**](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-42340)

## 

## 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 Financials software application.

First, I would recommend removing any strings that are linked to business names within the code as it can create a vulnerability or door for unauthorized access. Second, I would recommend integrating an authentication system, possibly a two-step verification, i.e., username and password or secret question etc., which is the basic form of security to protect user’s account. Third, possibly implement a permission-based authorization system vs a role-based authorization system. Lastly, I would recommend updating the dependencies to ensure that all previous vulnerabilities are fixed with the latest updates.