****

# 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** | **9/17/2023** | **Brian DeMaris** | **3-3 Project One** |

## 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 following 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, insert them in all the relevant locations in the document.
* Refer to the Project One Guidelines and Rubric for detailed instructions about each template section.

## Developer

Brian DeMaris

## Interpreting Client Needs

Security is considered a foundational expectation for a financial company. Customers and business partners expect that the financial companies they do business with will protect their private data and that all communications with the company will use industry-standard security to protect web-based interactions. Any financial company that fails in these areas will quickly realize the severity of the mistake and could have grave consequences for the company's future.

Artemis Financial has many international customers, so it is even more critical for us to ensure the security of the APIs used for their customer software. Financial companies are prime targets for threat actors looking to compromise companies' security and gain access to customer data. Many laws and regulations cover financial companies and their security.

(GLBA)Gramm-Leach-Bliley Act requires financial institutions to explain how they share and protect their customer's private information. (SOX) The Sarbanes-Oxley Act addresses the integrity of the firm's financial reporting processes and audits. (PCI DSS) Payment Card Industry Data Security Standard is a set of security standards to ensure that all companies that accept, process, store, or transmit credit card information maintain a secure environment. The EU also has laws concerning financial companies and how personal data is stored and processed. (GDPR) General Data Protection Regulation significantly affects how companies store and process customers' data.

As we will see as we progress, we must ensure that all components, software, and libraries are updated regularly to combat newly discovered vulnerabilities. These vulnerabilities can lead to attacks such as SQL Injection, which can expose company databases to theft and loss of data. Insecure data transmissions are also a significant concern. A lack of or poorly implemented encryption leaves companies vulnerable to eavesdropping and man-in-the-middle attacks. Poorly implemented authentication can allow unauthorized users to access areas they should not have.

We must always consider these technologies' dangers when considering modernization and the rise of open-source libraries and new technologies to stay competitive. Open-source libraries can accelerate development and reduce costs but can introduce vulnerabilities if not regularly updated. Care must be taken when choosing to use open-source software. Choosing software that is well-maintained and updated frequently can reduce security risks.

Using new technologies offers many benefits to customers and companies but brings many new challenges. Because the trend has become using software as a service or serverless architectures such as AWS and Azure, it brings new challenges and vulnerabilities that developers may not fully understand. New front-end technologies such as React, Vue, and Angular help developers build web applications that are self-contained, reusable, and easily integrated, but they also introduce new vulnerabilities. Over-reliance on client-side logic can be risky if the service code is hacked.

## Areas of Security

API security is critical to securing web applications for Artemis Financial. The proper use and implementation of the REST API is needed to ensure security. Insecure endpoints can expose data to attackers, and by implementing strong authentication and authorization checks for each endpoint, we can mitigate these risks. Using standard authentication, we limit unauthorized access to API endpoints and avoid sending credentials in the URL. We must validate and sanitize input data and parameterized queries for database access. The use of rate-limiting can help prevent DDoS attacks.

Cryptography is the cornerstone of secure web communications and authentications. It protects data both at rest and in transit. Its proper implementation is critical.

Secure coding starts with knowledgeable developers who understand the risks and how to mitigate them while coding the application. Secure coding is involved with every aspect of software development. Knowing the risks involved and how to develop secure code that reduces these risks requires developers who constantly watch the latest vulnerabilities and techniques to deal with them.

Building a web-based application client/server design and implementation is something that must be taken into consideration. This category builds upon all the previously mentioned concerns, such as APIs, cryptography, and secure coding practices.

## Manual Review

The DocData.java file contains a security risk. This method passes the username and password in the connection request URL, leaving access credentials unsecured. The CRUDController.java can expose the underlying code. The method parameter uses the "business name" to reference and access the underlying object. This parameter leaves the code vulnerable to SQL Injection attacks.

## Static Testing

***Bcprov-jdk15on-1.46.jar: Bouncy Castle Java Library***

Description: Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms.

Codes:

CVE-2016-1000338

CVE-2016-1000342

CVE-2016-1000343

CVE-2016-1000344

CVE-2016-000352

CVE-2016-1000341

CVE-2016-1000345

CVE-2017-13098

CVE-2020-15522

CVE-2020-0187

CVE-2016-1000339

CVE-2020-26939

CVE-2015-7940

CVE-2018-5382

CVE-2013-1624

CVE-2016-1000346

CVE-2015-6644

**Hibernate-validator-6.0.18.Final.jar**

Description: Hibernates Bean Validator

Codes:

CVE-2020-10693

**Jackson-databind-2.10.2.jar**

Description: General data-binding functionality for Jackson.

Codes:

CVE-2020-25649

CVE-2020-36518

CVE-2021-46877

CVE-2022-42003

CVE-2022-42004

**Logback-core-1.2.3.jar**

Description: logback-core module

Codes:

CVE-2021-42550

**Snakeyaml-1.25.jar**

Description: YAML 1.1 parser and emitter for Java

Codes:

CVE-2022-1471

CVE-2017-18640

CVE-2022-25857

CVE-2022-38749

CVE-20220-38751

CVE-2022-38752

CVE-2022-41854

CVE-2022-38750

**Spring-boot-2.2.4.RELEASE.jar**

Description: Spring Boot

Codes:

CVE-2022-27772

**Spring-boot-autoconfigure-2.2.4.RELEASE.jar**

Description: Spring Boot AutoConfigure

Codes:

CVE-2023-20883

**Spring-boot-starter-web-2.2.4.RELEASE.jar**

Description: Starter for building web, including RESTful, applications using spring MVC.

Codes:

CVE-2022-27772

CVE-2021-22118

CVE-2020-5421

CVE-2022-22950

CVE-2022-22971

CVE-2023-20861

CVE-2023-20863

CVE-2022-22968

CVE-2022-22970

CVE-2021-22060

CVE-2021-22096

**Spring-web-5.2.3.RELEASE.jar**

Description: Spring Web

Codes:

CVE-2016-100027

CVE-2022-22965

CVE-2021-22118

CVE-2020-5421

CVE-2022-22950

CVE-2022-22971

CVE-2023-20861

CVE-2023-20863

CVE-2022-22968

CVE-2022-22970

CVE-2021-22060

CVE-2021-22096

**Spring-webmvc-5.2.3.RELEASE.jar**

Description: Spring Web MVC

Codes:

CVE-2022-22965

CVE-2021-22118

CVE-2020-5421

CVE-2022-22950

CVE-2022-22971

CVE-2023-20861

CVE-2023-20863

CVE-2022-22968

CVE-2022-22970

CVE-2021-22060

CVE-2021-22096

**Tomcat-embed-core-9.0.30.jar**

Description: Core Tomcat implementation

Codes:

CVE-2020-1938

CVE-2020-11996

CVE-2020-13934

CVE-2020-13935

CVE-2020-17527

CVE-2021-25122

CVE-2021-41079

CVE-2022-29885

CVE-2022-42252

CVE-2020-9484

CVE-2021-25329

CVE-2021-30640

CVE-2022-34305

CVE-2021-24122

CVE-2021-33037

CVE-2019-17569

CVE-2020-1935

CVE-2020-13943

CVE-2023-28708

CVE-2021-43980

**Tomcat-embed-websocket-9.0.30.jar**

Description: Core Tomcat Implementation

Codes:

CVE-2020-1938

CVE-2020-11996

CVE-2020-13934

CVE-2020-13935

CVE-2020-17527

CVE-2021-25122

CVE-2021-41079

CVE-2022-29885

CVE-2022-42252

CVE-2020-9484

CVE-2021-25329

CVE-2021-30640

CVE-2022-34305

CVE-2021-24122

CVE-2021-33037

CVE-2019-17569

CVE-2020-1935

CVE-2020-13943

CVE-2023-28708

CVE-2021-43980

## Mitigation Plan

The first step is implementing a robust monitoring and logging system to help detect and address suspicious activities. Dealing with the listed vulnerabilities involves updating the libraries to the latest versions, which have been patched against these vulnerabilities. The rest of the mitigation plan includes using secure coding practices, regularly reviewing our code for new vulnerabilities and ongoing testing for new threats. Security is an ongoing process that needs updating and refinement as new threats arise.