****

# Artemis Financial Vulnerability Assessment Report

Matt Smith

3/19/2023

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** | **3/19/2023** | **Matt Smith** | **Filled out document** |

## 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

Matt Smith

## Interpreting Client Needs

Secure communication is of high value to the company since they deal with sensitive information like savings, retirement, investments, and insurance. Without secure communication, information could be intercepted and misused, resulting in harm to the company’s reputation and financial losses to their customers.

From the description provided about Artemis Financial, it doesn’t clearly state if they deal with international transactions or not. Either way, the company still needs to take measures to ensure secure communication.

There weren’t specific government restrictions mentioned but the company should make sure to comply with security regulations and guidelines.

External threats that might be present now or in the future include the following:

* Brute-force attacks.
* Malware attacks.
* SQL injection attacks.
* Cross-site scripting attacks.
* Distributed Denial of Service attacks.
* Man-in-the-middle attacks.

The role of open-source libraries is to reduce development time and cost. The downside to them is that they can introduce security vulnerabilities if they aren’t up-to-date or are not properly tested.

As web application technologies evolve, new security vulnerabilities and threats are introduced. Software needs to be designed to keep up with these changes and address them effectively.

## Areas of Security

Based on the software application and potential threats I previously identified, I have identified the following areas of security that are applicable:

1. Input Validation: Input Validation is important to protect again SQL Injection and Cross-site scripting attacks. Any data input by the user should be validated to ensure it’s in the expected format and range.

1. APIs: Since the company has a RESTful API, it’s important to ensure that the API is secure against attacks. Authentication and authorization methods should be implemented to control access to the API and ensure that only authorized users can access information.
2. Cryptography: Cryptography is necessary to ensure secure communication between the client and server. The communication needs to be encrypted using strong encryption algorithms.
3. Client/Server: The Client/Server communication needs to be secured using encryption and authentication to protect against man-in-the-middle attacks. The server also needs to be configured to deny unauthorized access and prevent data loss or corruption.
4. Code Quality: Code quality is necessary to ensure that there are no security vulnerabilities introduced by coding errors or unhandled exceptions. The code needs to be tested using tools like static code analysis to detect and fix any security vulnerabilities.
5. Encapsulation: Encapsulation is necessary to ensure that sensitive information is not accessible to unauthorized users. All sensitive information should be encrypted and stored securely in a database. The database should also be secured using access controls and encryption methods.

## Manual Review

After manually reviewing the code, I found the following issues:

1. Input Validation:
   1. The GreetingController class doesn’t implement input validation, allowing a user to input any value for the name parameter.
   2. The CRUDController class doesn’t implement input validation for the “business\_name” parameter, which could allow for SQL injection attacks.
2. APIs:
   1. The CRUDController class looks like it exposes a read API that could return sensitive data without any authentication or authorization.
3. Cryptography:
   1. The program has a dependency for cryptography but nothing in the code implements it to protect data.
4. Client/Server:
   1. There’s nothing in the code that implements secure communication protocols between the client and server which could leave data in transit vulnerable.
5. Code Error:
   1. The myDateTime class does not properly implement the setter and getter methods.
6. Code Quality:
   1. The DocData class has a hardcoded database connection string, username, and password which is a bad practice.
   2. The customer class has no access modifiers for its variables, allowing them to be accessed and modified from anywhere in the code.
7. Encapsulation:
   1. The myDateTime class has public variables, which can be accessed and modified from anywhere in the code.
   2. The customer class doesn’t set “account\_balance” as private, which means it can be accessed or modified from anywhere in the code.

## Static Testing

1. **Dependency**: bcprov-jdk15on-1.46.jar  
   **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.

**Codes**: CVE-2016-1000338, CVE-2016-1000342, CVE-2016-1000343, CVE-2016-1000344, CVE-2016-1000352, 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.

**Recommended Solution**: Upgrade to the latest version to address to issues.  
**Attribution**: [NVD Bouncy Castle Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Abouncycastle&cpe_product=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api&cpe_version=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api%3A1.46)

1. **Dependency**: hibernate-validator-6.0.18.Final.jar

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

**Codes**: CVE-2020-10693.

**Recommended Solution**: Upgrade to the latest version to address to issue.  
**Attribution**: [NVD Hibernate-Validator Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aredhat&cpe_product=cpe%3A%2F%3Aredhat%3Ahibernate_validator&cpe_version=cpe%3A%2F%3Aredhat%3Ahibernate_validator%3A6.0.18)

1. **Dependency**: jackson-databind-2.10.2.jar

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

**Codes**: CVE-2020-25649, CVE-2020-36518, CVE-2022-42003, CVE-2022-42004

**Recommended Solution**: Upgrade to the latest version to address to issues.  
**Attribution**: [NVD Jackson-Bind Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Afasterxml&cpe_product=cpe%3A%2F%3Afasterxml%3Ajackson-databind&cpe_version=cpe%3A%2F%3Afasterxml%3Ajackson-databind%3A2.10.2)

1. **Dependency**: log4j-api-2.12.1.jar

**Description**: The Apache Log4j API

**Codes**: CVE-2020-9488

**Recommended Solution**: Upgrade to the latest version to address to issue.  
**Attribution**: [NVD Apache Log4j API Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Alog4j&cpe_version=cpe%3A%2F%3Aapache%3Alog4j%3A2.12.1)

1. **Dependency**: logback-core-1.2.3.jar

**Description**: logback-core module

**Codes**: CVE-2021-42550

**Recommended Solution**: Upgrade to the latest version to address to issue.  
**Attribution**: [NVD Logback-Core Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aqos&cpe_product=cpe%3A%2F%3Aqos%3Alogback&cpe_version=cpe%3A%2F%3Aqos%3Alogback%3A1.2.3)

1. **Dependency**: 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-2022-38751, CVE-2022-38752, CVE-2022-41854, CVE-2022-38750.

**Recommended Solution**: Upgrade to the latest version to address to issues.  
**Attribution**: [NVD Snakeyaml Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Asnakeyaml_project&cpe_product=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml&cpe_version=cpe%3A%2F%3Asnakeyaml_project%3Asnakeyaml%3A1.25)

1. **Dependency**: spring-boot-2.2.4.RELEASE.jar

**Description**: Spring Boot

**Codes**: CVE-2022-27772

**Recommended Solution**: Upgrade to the latest version to address to issue.  
**Attribution**: [NVD Spring Boot Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_boot&cpe_version=cpe%3A%2F%3Avmware%3Aspring_boot%3A2.2.4https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_boot&cpe_version=cpe%3A%2F%3Avmware%3Aspring_boot%3A2.2.4)

1. **Dependency**: spring-boot-starter-web-2.2.4.RELEASE.jar

**Description**: Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container.

**Codes**: CVE-2022-27772

**Recommended Solution**: Upgrade to the latest version to address to issue.  
**Attribution**: [NVD Sprint Boot Starter Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_boot&cpe_version=cpe%3A%2F%3Avmware%3Aspring_boot%3A2.2.4)

1. **Dependency**: spring-core-5.2.3.RELEASE.jar

**Description**: Spring Core

**Codes**: CVE-2022-22965, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096

**Recommended Solution**: Upgrade to the latest version to address to issues.  
**Attribution**: [NVD Spring Core Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Apivotal_software&cpe_product=cpe%3A%2F%3Apivotal_software%3Aspring_framework&cpe_version=cpe%3A%2F%3Apivotal_software%3Aspring_framework%3A5.2.3)

1. **Dependency**: spring-web-5.2.3.RELEASE.jar

**Description**: Spring Web

**Codes**: CVE-2016-1000027, CVE-2022-22965, CVE-2021-22118, CVE-2020-5421, CVE-2022-22950, CVE-2022-22971, CVE-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096.

**Recommended Solution**: Upgrade to the latest version to address to issues.  
**Attribution**: [NVD Spring Web Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3)

1. **Dependency**: 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-2022-22968, CVE-2022-22970, CVE-2021-22060, CVE-2021-22096

**Recommended Solution**: Upgrade to the latest version to address to issues.  
**Attribution**: [NVD Spring Web MVC Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Avmware&cpe_product=cpe%3A%2F%3Avmware%3Aspring_framework&cpe_version=cpe%3A%2F%3Avmware%3Aspring_framework%3A5.2.3)

1. **Dependency**: 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-2021-43980.

**Recommended Solution**: Upgrade to the latest version to address to issues.  
**Attribution**: [NVD Tomcat Embed Core Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.30)

1. **Dependency**: tomcat-embed-websocket-9.0.30.jar

**Description**: Core Tomcat implementation.

**Codes**: CVE-2020-1938, CVE-2020-8022, 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-2021-43980.

**Recommended Solution**: Upgrade to the latest version to address to issues.  
**Attribution**: [NVD Tomcat Embed WebSocket Vulnerabilities](https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Aapache&cpe_product=cpe%3A%2F%3Aapache%3Atomcat&cpe_version=cpe%3A%2F%3Aapache%3Atomcat%3A9.0.30)

## Mitigation Plan

To fix each vulnerability I found in the manual review of the code, I recommend the following:

1. Input Validation:
   1. Implement input validation in the GreetingController class to prevent users from inputting any value for the name parameter.
   2. Implement input validation for the “business\_name” parameter in the CRUDController class to prevent SQL injection attacks.
2. APIs:
   1. Implement authentication and authorization for the read API in the CRUDController class to prevent unauthorized access to sensitive data.
3. Cryptography:
   1. Implement cryptography to protect sensitive data by using the cryptography dependency in the code.
4. Client/Server:
   1. Implement secure communication protocols between the client and server to prevent data in transit from being vulnerable.
5. Code Error:
   1. Properly implement the setter and getter methods in the myDateTime class.
6. Code Quality:
   1. Remove hardcoded database connection string, username, and password from the DocData class and use configuration files to store them securely.
   2. Set access modifiers for variables in the customer class to prevent them from being accessed or modified from anywhere in the code.
7. Encapsulation:
   1. Set the public variables in the myDateTime class to private to prevent them from being accessed or modified from anywhere in the code.
   2. Set the “account\_balance” variable in the customer class to private to prevent it from being accessed or modified from anywhere in the code.

To fix all the dependencies from the static testing report, I recommend upgrading them all to their latest versions to address the security issues. Once all of these changes have been implemented, I recommend conducting a new assessment to make sure that all of the vulnerabilities have been addressed and that the code is secure.