****

# 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 5](#_Toc32574612)

[3. Manual Review 5](#_Toc32574613)

[4. Static Testing 6](#_Toc32574614)

[5. Mitigation Plan 13](#_Toc32574615)

6. References…………….……………………………………………………………………………………………………………………..14

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **9-17-2023** | **Justin Starr** |  |

## 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

Justin Starr

## Interpreting Client Needs

*The Value of Secure Communications to the Company*

Secure communications are of extreme value. Artemis Financial handles sensitive information for its clients. This information includes individualized financial plans for their customers, such as savings, retirement, investments, and insurance plans. Because of handling such sensitive information, Artemis Financial must treat security and its associated risks with high regard. Implementing secure communication measures, such as the use of a RESTful API, is paramount for ensuring secure and reliable client/server connections and transfers of information.

*International Transactions and Governmental Restrictions to Consider*

Because of the type of company that Artemus Financial is, it is likely that they do make international transactions. Various government restrictions do exist, that must be followed, to help ensure secure communication and the handling of consumers’ sensitive data. Such restrictions include restrictions from the Federal Trade Commission about what information can be shared with other companies and what actions companies must take to protect or safeguard consumers’ information (Federal Trade Commission, 2021). Such actions include providing consumers with details about how their information is shared with other companies and even how much and to what extent their information is shared (Federal Trade Commission, 2021). Updates to the Safeguards Rule which was first mandated by Congress under the 1999 Gramm-Leach-Bliley Act, “include more specific criteria for the safeguards that financial institutions must implement as part of their information security program, such as limiting who can access consumer data and using encryption to secure the data” (Federal Trade Commission, 2021).

*External Threats that Might Be Present Now and in the Immediate Future*

Because of the type of company Artemis Financial is, one that develops individualized financial plans for their customers, external threats are present now and will be in the immediate future. Such threats include ransomware, phishing, web application and vulnerability exploitation attacks, denial of service (DoS) attacks, and even insider threats. These attacks are made possible when malicious attackers exploit weaknesses and vulnerabilities that exist within a particular system. One common method that attackers utilize is making use of intercepting web proxies, which attackers can use to exploit a web application. Additional ways attackers attempt to gain access is when software developers have a

lack of parameterized SQL, which can lead to SQL injection, making use of weak or incorrect ciphers, and lack of input validation (Manico & Detlefsen, 2014, Chapter 1). It is imperative that proper access control methods be utilized such as the Principle of Least Privilege (POLP), which essentially is only giving enough privileges to any program or user of a system, enough privilege to accomplish exactly what they need to accomplish and nothing more (Manico & Detlefsen, 2014, Chapter 1). This helps to limit the data that is made available to users. Any person whose aim is to gain access to sensitive personal and financial information will always be a significant threat.

Modernization Requirements

The use of open-source libraries has become increasingly beneficial for software developers. Utilizing open-source libraries cuts down on the time it takes to create applications by utilizing code bases that have been made available for public use. However, as beneficial as they can be, it is important that security concerns remain at the forefront of the developer's mind and that the libraries they use may expose their applications to additional vulnerabilities or threats. It is essential that when outside libraries are used, secure APIs are implemented to ensure the secure handling and transmission of data between web applications. Also, while evolving web application technologies continue to add convenience and often make the ability to accomplish tasks faster, they also pose additional threats and areas of vulnerability to systems. We must remain vigilant in our efforts to thwart attackers by being aware of the potential risks associated with these evolving technologies, especially as many of these growing technologies are new and can potentially expose sensitive data/information.

## Areas of Security

The areas of security relevant to the software application according to the Vulnerability Assessment Process Flow Diagram (VAPFD) are as follows:

* Input Validation: Secure input and representations
* APIs: Secure API interactions
* Cryptography: Encryption Use and Vulnerabilities
* Code Quality: Secure coding practices/patterns

Input validation was chosen because the application needs to be able to securely verify all input. Additionally, we want to ensure the application does not accept untrusted input or illegal input. APIs were chosen because the application makes use of the Spring framework and implements a REST controller. Both support secure API interactions where secure client/server communication and transmission of data is required. Cryptography was selected because the application is expected to handle sensitive information and must be protected. Lastly, Code Quality was chosen because it is a superset of input validation and secure API interactions. It is essential that the code base be reviewed to identify coding practices/patterns that do not comply with secure coding best practices. Design patterns should be implemented to help ensure the overall security of the application.

## Manual Review

We begin reviewing the code base to look for vulnerabilities in each of the identified areas of security. For input validation, we want to identify all areas of each class file that take in input and determine whether those areas properly validate input. Next, we need to look at any code that has implemented an API and ensure that these APIs are up-to-date and implemented correctly. Because cryptography is chosen as an area of security that could potentially have vulnerabilities, we want to look at the application to see if it is making use of cryptography. Lastly, we want to review the code base again to ensure that best practices are being implemented in all aspects of the design. We also want to ensure the application incorporates secure design patterns.

In CRUDcontroller.java and in GreetingController.java, we see methods that accept input but do not validate the input they take, for example, dangerous character input. The application should ensure it handles expected and unexpected input appropriately. In CRUDController.java, business names are sent as request parameters. This increases the likelihood that sensitive information can become exposed. In both CRUDController.java andGreetingController.java, request parameters are not validated.

We also notice that the application makes use of the Spring framework, which is an API. The RestServiceApplication.java should incorporate methods for authentication and user access control. This will ultimately help to ensure the secure transfer and handling of sensitive data. Also, the API does not make use of HTTPS (Hypertext Transport Protocol Secure). For it to be a RESTful API, it should use HTTPS not HTTP to ensure the secure transmission of data.

Upon looking through the code base, it appears that no secure form of encryption is being used. Because of the type of information that Artemis Financial stores about its customers, this data should be encrypted when it is being accessed. Associated files where we would need to see encryption occurring are when customer.java, DocData.java, and myDateTime.java are accessed.

## Static Testing

The following is a screenshot of the dependency-check report that was conducted for Artimus Financial’s web-based software application:

A screenshot of a computer

Description automatically generated

After running the dependency check on the Artemis Financial software application, 13 vulnerable dependencies were identified with a total of 118 vulnerabilities found. The following is a list of the dependencies, their associated codes, solutions, and relevant information as to how the vulnerability was identified or how it was documented in the past.

* [bcprov-jdk15on-1.46.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l1_991c96a4e31e6c19e2b9136c8955bd423f2dc4c7) – HIGH Severity – CVE Count: 18

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.

Vulnerabilities include the possibility for injection attacks, timing attacks, and other known cryptographic-related attacks. The following is a list of different CVEs with various weakness types:

* CVE-2016-10000338 – Weakness type (CWE-347) - Improper Verification of Cryptographic Signature

Suggested Fix: Upgrade to version org.bouncycastle:bcprov-debug-jdk15on:1.55,org.bouncycastle:bcprov-debug-jdk14:1.55,org.bouncycastle:bcprov-ext-jdk14:1.55,org.bouncycastle:bcprov-ext-jdk15on:1.55,org.bouncycastle:bcprov-jdk14:1.55,org.bouncycastle:bcprov-jdk15on:1.55,org.bouncycastle:bcprov-ext-debug-jdk15on:1.55

* CVE-2016-1000343 – Weakness type (CWE-310) Cryptography Issues.

Suggested Fix: Upgrade to version org.bouncycastle:bcprov-debug-jdk14:1.56,org.bouncycastle:bcprov-ext-jdk15on:1.56,org.bouncycastle:bcprov-jdk14:1.56,org.bouncycastle:bcprov-jdk15on:1.56,org.bouncycastle:bcprov-ext-debug-jdk15on:1.56

* CVE-2016-1000341 – Weakness Type (CWE-361) Time and State

Suggested Fix: Upgrade to version org.bouncycastle:bcprov-debug-jdk15on:1.56,org.bouncycastle:bcprov-debug-jdk14:1.56,org.bouncycastle:bcprov-ext-jdk15on:1.56,org.bouncycastle:bcprov-jdk14:1.56,org.bouncycastle:bcprov-jdk15on:1.56,org.bouncycastle:bcprov-ext-debug-jdk15on:1.56

* CVE-2017-13098 – Weakness Type (CWE-203) Observable Discrepancy and (CWE-310) Cryptographic Issues

Suggested Fix: Upgrade to version 1.0.3

* CVE-2020-15522 – Weakness Type (CWE-362) Concurrent Execution using Shared Resource with Improper Synchronization (‘Race Condition’)

Suggested Fix: Upgrade to version org.bouncycastle:bc-fips:1.0.2.1;org.bouncycastle:bcprov-ext-jdk14:1.66;org.bouncycastle:bcprov-ext-jdk15on:1.66;org.bouncycastle:bcprov-jdk14:1.66;org.bouncycastle:bcprov-jdk15on:1.66;BouncyCastle - 1.8.9;Portable.BouncyCastle - 1.8.8

* CVE-2020-26939 – Weakness Type (CWE-295) Improper Certificate Validation

Suggested Fix: Upgrade to version org.bouncycastle:bc-fips:1.0.2.1;org.bouncycastle:bcprov-ext-jdk14:1.66;org.bouncycastle:bcprov-ext-jdk15on:1.66;org.bouncycastle:bcprov-jdk14:1.66;org.bouncycastle:bcprov-jdk15on:1.66;BouncyCastle - 1.8.9;Portable.BouncyCastle - 1.8.8

This list continues with other CVEs with common weakness types. All suggest updating to the latest version.

* [hibernate-validator-6.0.18.Final.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l3_7fd00bcd87e14b6ba66279282ef15efa30dd2492) – MEDIUM Severity – CVE Count: 1

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

CVE-2020-10693 – Its Weakness type (CWE) is Input validation. Fix: Upgrade to version org.hibernate-validator:6.0.20.Final,6.1.5.Final

* [jackson-databind-2.10.2.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l5_0528de95f198afafbcfb0c09d2e43b6e0ea663ec) – HIGH Severity – CVE Count: 6

Description: General data-binding functionality for Jackson: works on core streaming API - The following is a list of associated CVEs with different weakness types and their associated fixes.

* + CVE-2023-35166 - Its weakness type is (CWE-611) Allocation of resources without limits or throttling.

Suggested Fix: Upgrade to version com.fasterxml.jackson.core:Jackson-databind:2.12.6,2.13.1

* + CVE-2021-46877 – Its weakness type (CWE-770) is allocation of resources without limits or throttling.

Suggested Fix: Upgrade to version com.fasterxml.jackson.core:jackson-databind:2.12.6,2.13.1

* + CVE-2022-42004 - Its weakness type (CWE-502) is deserialization of untrusted data.

Suggested Fix: Upgrade to version com.fasterxml.jackson.core:jackson-databind:2.13.4

* + CVE-2022-42003 - Its weakness type is (CWE-502) Deserialization of Untrusted Data.

Suggested Fix: Upgrade to version com.fasterxml.jackson.core:jackson-databind:2.12.7.1,2.13.4.1

* + CVE-2020-36518 - Its weakness type (CWE-787) is out-of-bounds write.

Suggested Fix: Upgrade to version com.fasterxml.jackson.core:jackson-databind:2.12.6.1,2.13.2.1

* + CVE-2020-25649 - Its weakness type (CWE-611) is improper restriction of XML external entity reference.

Suggested Fix: Upgrade to version com.fasterxml.jackson.core:jackson-databind:2.6.7.4,2.9.10.7,2.10.5.1,2.11.0.rc1.

* [log4j-api-2.12.1.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l10_a55e6d987f50a515c9260b0451b4fa217dc539cb) – LOW Severity – CVE Count: 1

Description: The Apache Log4j API

* + CVE-2020-9488 - Its weakness type (CWE-295) is improper certificate validation.

Suggested Fix: Upgrade to version ch.qos.reload4j:reload4j:1.2.18.3

* [logback-core-1.2.3.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l12_864344400c3d4d92dfeb0a305dc87d953677c03c) – MEDIUM Severity – CVE Count: 1

Description: logback-core module

* + CVE-2021-42550 - Its weakness type (CWE-502) is deserialization of untrusted data.

Suggested Fix: Upgrade to version ch.qos.logback:logback-classic:1.2.9;ch.qos.logback:logback-core:1.2.9

* [snakeyaml-1.25.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l14_8b6e01ef661d8378ae6dd7b511a7f2a33fae1421) – CRITICAL Severity – CVE Count: 8

Description: YAML 1.1 parser and emitter for Java - The following is a list of associated CVEs with different weakness types and their associated fixes.

* + CVE-2022-1471 – Weakness type (CWE-502) Deserialization of Untrusted Data.

Suggested Fix: Upgrade to version org.yaml:snakeyaml:2.0.

* + CVE-2022-25857 - Weakness type (CWE-776) Improper restriction of recursive entity references in DTDs (‘XML Entity Expansion’).

Suggested Fix: Upgrade to version org.yaml:snakeyaml:1.31

* + CVE-2022-38749 – Weakness type (CWE-787) Out of Bounds.

Suggested Fix: Upgrade to version org.yaml:snakeyaml:1.31

There were five additional CVEs that had the same weakness types and their fixes were also similar.

* [spring-boot-2.2.4.RELEASE.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l15_225a4fd31156c254e3bb92adb42ee8c6de812714) – CRITICAL Severity – CVE Count: 3

Description: Spring Boot - The following is a list of associated CVEs with different weakness types and their associated fixes.

* + CVE-2023-20873 -no information regarding the weakness(NVD-CWE)

Suggested Fix: Upgrade to version org.springframework.boot:spring-boot-actuator-autoconfigure:2.7.11,3.0.6

* + CVE-2022-27772 – Its weakness type (CWE-668) Exposure of resources to wrong sphere.

Suggested Fix: Upgrade to version org.springframework.boot:spring-boot-actuator-autoconfigure:2.7.11,3.0.6

* + CVE-2023-20883 – Its weakness type (CWE-400) Uncontrolled Resource Consumption.

Suggested Fix: Upgrade to version org.springframework.boot:spring-boot-actuator-autoconfigure:2.7.11,3.0.6

* [spring-boot-starter-web-2.2.4.RELEASE.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l16_ec75d01d212b5229c16d872fb127744c0ed46ed8) – CRITICAL Severity – CVE Count: 3

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

The CVEs, their weakness types, and their fixes are all the same as the previously mentioned dependency.

* [spring-core-5.2.3.RELEASE.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l17_3734223040040e8c3fecd5faa3ae8a1ed6da146b) – CRITICAL Severity – CVE Count: 11

Description: Spring Core - The following is a list of associated CVEs with different weakness types and their associated fixes.

* + CVE-2022-22965 – Its weakness type (CWE-94) is improper control of generation of code (‘Code Injection’).

Suggested Fix: update to Spring framework 5.3.18 and 5.2.20 or greater.

* + CVE-2021-22118 – Its weakness type (CWE-668) is exposure of resource to wrong sphere.

Suggested Fix: Upgrade to version org.springframework:spring-web:5.2.15,5.3.7

* + CVE-2020-5421 – no info regarding weakness type.

Suggested Fix: Upgrade to version org.springframework:spring-web:4.3.29,5.0.19,5.1.18,5.2.9

* + CVE-2022-22950 – Weakness type (CWE-770) allocation of resources without limits or throttling.

Suggested Fix: Upgrade to version org.springframework:spring-expression:5.2.20,5.3.17

* + CVE-2023-20863 – Weakness type (CWE-917) is Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection').

Suggested Fix: Upgrade to version org.springframework:spring-expression - 5.2.24.RELEASE,5.3.27,6.0.8

* + CVE-2022-22968 – Weakness type (CWE-178) is improper handling of case sensitivity.

Suggested Fix: Upgrade to version org.springframework:spring-expression - 5.2.24.RELEASE,5.3.27,6.0.8

There were five additional CVEs that had the same weakness types and their fixes were also similar.

* [spring-web-5.2.3.RELEASE.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l18_dd386a02e40b915ab400a3bf9f586d2dc4c0852c) – CRITICAL Severity – CVE Count: 12

Description: Spring Web - The following is a list of associated CVEs with different weakness types and their associated fixes.

* + CVE-2016-1000027 – Weakness type (CWE-502) is deserialization of untrusted data.

Suggested Fix: Upgrade to version org.springframework:spring-web:6.0.0

* + CVE-2022-22965 - Its weakness type (CWE-94) is improper control of generation of code (‘Code Injection’).

Suggested Fix: update to Spring framework 5.3.18 and 5.2.20 or greater.

* + CVE-2021-22118 – Its weakness type (CWE-668) is exposure of resource to wrong sphere.

Suggested Fix: Upgrade to version org.springframework:spring-web:5.2.15,5.3.7

* + CVE – 2022-22950 – Its weakness type (CWE-770) is allocation of resources without limits or throttling.

Suggested Fix: Upgrade to version org.springframework:spring-expression:5.2.20,5.3.17

* + CVE-2023-20863 – Weakness type (CWE-917) is Improper Neutralization of Special Elements used in an Expression Language Statement ('Expression Language Injection').

Suggested Fix: Upgrade to version org.springframework:spring-expression - 5.2.24.RELEASE,5.3.27,6.0.8

This list continues on with many of the same types of weaknesses and associated CVEs from previously discussed dependencies. Again, there were a total of 12 CVE associated with this dependency, all of their fixes were similar requiring an update.

* [spring-webmvc-5.2.3.RELEASE.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l19_745a62502023d2496b565b7fe102bb1ee229d6b7) – CRITICAL Severity – CVE Count: 11

Description: Spring Web MVC

This dependency was also quite similar to previously mentioned spring related dependencies. Their central solution to all identified CVEs was to update the corresponding version or framework.

* [tomcat-embed-core-9.0.30.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l20_ad32909314fe2ba02cec036434c0addd19bcc580) – CRITICAL Severity – CVE Count: 21

Description: Core Tomcat implementation - The following is a list of associated CVEs with different weakness types and their associated fixes.

* CVE-2020-1938 – Weakness type (CWE-20 & CWE-269) is input validation and improper privilege management.

Suggested Fix: Upgrade to version 7.0.100,8.5.51,9.0.31

* CVE-2020-13934 – Weakness type (CWE-401 & CWE-476) Missing release of memory after effective lifetime and NULL pointer dereference.

Suggested Fix: Upgrade to version org.apache.tomcat:tomcat-coyote:8.5.57,9.0.37,10.0.0-M7

* CVE-2020-13935 – Weakness type (CWE-835) is loop with unreachable exit condition (‘Infinite Loop’).

Suggested Fix: Upgrade to version org.apache.tomcat:tomcat-websocket:7.0.105,8.5.57,9.0.37,10.0.0-M7;org.apache.tomcat.embed:tomcat-embed-websocket:7.0.105,8.5.57,9.0.37,10.0.0-M7

* CVE-2020-17527 – Weakness type (CWE-200) is exposure of sensitive information to an unauthorized actor.

Suggested Fix: Upgrade to version org.apache.tomcat:tomcat-websocket:7.0.105,8.5.57,9.0.37,10.0.0-M7;org.apache.tomcat.embed:tomcat-embed-websocket:7.0.105,8.5.57,9.0.37,10.0.0-M7

This list continues with similar vulnerabilities and various weakness types. All show common fixes as performing updates.

* [tomcat-embed-websocket-9.0.30.jar](file:///C:\Users\justi\eclipse-workspace\rest-service\target\dependency-check-report.html#l22_33157f6bc5bfd03380ebb5ac476db0600a04168d) – CRITICAL Severity – CVE Count: 22

Description: Core Tomcat implementation

This dependency was similar to the prior dependency. CVEs, their weakness types were matching all providing the same similar solutions: upgrade version.

## Mitigation Plan

* Implement Input Validation design patterns to ensure secure input validation.
* Implement changes to request parameters by moving them to the <body> which will help to ensure the security of sensitive data.
* Implement Query Parameterization where possible.
* Implement secure APIs with RESTful states to ensure secure connections and transmission of sensitive data.
* Implement authentication methods in the API.
* Utilize the HTTPS protocol instead of HTTP for secure connections and transmission of sensitive data.
* Update framework and dependencies to their latest version.

**References**

Federal Trade Commission. (2021, October 27). *FTC Strengthens Security Safeguards for Consumer*

*Financial Information Following Widespread Data Breaches*. <https://www.ftc.gov/news-events/news/press-releases/2021/10/ftc-strengthens-security-safeguards-consumer-financial-information-following-widespread-data>

Manico, J., & Detlefsen, A. (2014). *Iron-Clad Java: Building Secure Web Applications*. McGraw Hill

Computing. <https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/?sso_link=yes&sso_link_from=SNHU>

NIST - National Institute of Standards and technology. (n.d.). National Vulnerability Database.

<https://nvd.nist.gov/>

*Open Source Vulnerability Database: MEND*. Mend.io. (2022, May 25).

https://www.mend.io/vulnerability-database/

OWASP. (2023). How To Read the Reports. dependency-check – How To Read The Reports.

https://jeremylong.github.io/DependencyCheck/general/thereport.html