****

# 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** | **Winnie Kwong** | **Project 1 CS 305** |

## 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

Winnie Kwong

## Interpreting Client Needs

The client, Artemis Financial, is a consulting company that develops individualized financial plans for their customers, such as savings, retirement, investments, and insurance plans. The client wishes to modernize their operations using the most current and adequate software security. Currently, the client has a RESTful web application programming interface (API) but would like to pursue more ways to protect their organization from external threats. Secure communications with the company are essential because companies must keep sensitive information confidential to maximize profits. Although there is no record of the company's international transactions, a customer may be involved in international investments. Although the United States does not have federal law regulating cybersecurity and privacy, several states have cybersecurity and data breach notification laws. Many forms of external threats are occurring in the present and immediate future. As technology progresses, attackers are also growing in numbers and finding new and clever ways to breach the system. As Artemis Financial works with sensitive information, the client will be more attractive to data breaches. Ways to modernize requirements are to focus on maintaining the open-source libraries and evolving their web applications.

## Areas of Security

After reviewing the functionality of the software application, areas of security that would apply to the client are input validations, APIs, cryptography, code error, and code quality.

* Input validations can help to prevent a wide range of threats and attacks that can lead to unauthorized access or even prevent websites from functioning. However, by implementing input validations properly, input validations can also improve user experiences for the client and help reduce failures and SQL injections on the server side.
* APIs allow applications to communicate with one another and share data across organizations. By securing API vulnerabilities, it can help to prevent attacks from gaining access to services and information.
* Cryptography helps regulate exported items by using algorithms only known to the sender and recipient. If Artemis Financial is involved in international transactions, cryptology is an additional layer of protection that can help comply with regulations worldwide.
* Code Errors work simultaneously with input validations and APIs to notify developers of improper input handling. When code errors are resolved, it can prevent privilege escalations and mitigate other vulnerabilities the system may encounter.
* Code quality ensures no data is accidentally exposed when working with API and input validations. Code quality helps to distribute the correct access rights to end user to verify methods and data match their level of authorization.

## Manual Review

Upon reviewing the code base, I identified a few red flags that must be addressed. The first significant issue needed to be locating layers of input validation. Under the pom.xml file, the Apache validator was missing. The Apache validator helps to validate things such as email, website, credit card data, and other personal information. Also, the inputs are submitted as strings when looking at the greeting controller, but there is no verification if the inputs are correct. Within the code base, I was unable to locate any APIs. The system does accept data; however, it needs to be better secured. Since the program was not written using the POST method, it is vulnerable to being exposed to a data breach because the data could leak into the browsing history. POST is intended to identify resources that are submitted in HTML form. The POST method is beneficial because it can help to send more information and change the server state from being exposed. As we advance with code errors, most of the classes within the code base do not have any form of error handling. The only attempt is a catch block within the DocData file that is used to string the kay and value for reading documents such as the username and password. The overall quality of the code is adequate in its current form; nevertheless, it needs to revise its input validations and APIs.

## Static Testing

* bcprov-jdk15on-1.46.jar  
    
  Vulnerabilities:
  + cpe:2.3:a:bouncycastle:bouncy-castle-crypto-package:1.46:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:bouncycastle:bouncy\_castle\_crypto\_package:1.46:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:bouncycastle:legion-of-the-bouncy-castle-java-crytography-api:1.46:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:bouncycastle:the\_bouncy\_castle\_crypto\_package\_for\_java:1.46:\*:\*:\*:\*:\*:\*:\*

Description & Recommended Solutions:

The Bouncy Castle Crypto package is a Java implementation of cryptographic algorithms. This jar contains the JCE provider and lightweight API for the Bouncy Castle Cryptography APIs for JDK 1.5 to JDK 1.7. The dependency has issues with cryptography, with versions 1.55 and older potentially leak formation because of the approach used in the algorithms. A way to resolve this issue is by updating to a newer version.

Attribution:

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-2023-33201, CVE-2015-7940, CVE-2018-5382, CVE-2013-1624, CVE-2016-1000346, CVE-2015-6644

* hibernate-validator-6.0.18.Final.jar

Vulnerabilities:

* + cpe:2.3:a:redhat:hibernate\_validator:6.0.18:\*:\*:\*:\*:\*:\*:\*

Description & Recommended Solutions:

Hibernate's Bean Validation (JSR-380) reference implementation. 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 implemented when handling user-controlled data in error messages. The defects can be resolved by upgrading to hibernate-validator 7.0.0.Alpha2, hibernate-validator 6.1.5.Final, or hibernate-validator 6.0.20.Final

Attribution: CVE-2020-10693

* jackson-databind-2.10.2.jar

Vulnerabilities:

* + cpe:2.3:a:fasterxml:jackson-databind:2.10.2:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:\*:\*:\*:\*:\*:\*:\*

Description & Recommended Solutions:

General data-binding functionality for Jackson works on core streaming API. A flaw was found in FasterXML Jackson Databind, where it needed to have entity expansion appropriately secured and allow vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity, but it can be resolved by upgrading to jackson-databind-2.11.0 or jackson-databind-2.10.5.1.

Attribution: CVE-2020-25649, CVE-2020-36518, CVE-2021-46877, CVE-2022-42003, CVE-2022-42004, CVE-2023-35116

* log4j-api-2.12.1.jar

Vulnerabilities:

* + cpe:2.3:a:apache:log4j:2.12.1:\*:\*:\*:\*:\*:\*:\*

Description & Recommended Solutions:

The Apache Log4j API has improper certificate validation with a host mismatch in the Apache Log4j SMTP appender. It could allow an SMTPS connection to be intercepted by a man-in-the-middle attack, which could leak any log messages sent through that appender. The issue can be resolved by upgrading to Apache Log4j 2.12.3 and 2.13.1.

Attribution: CVE-2020-9488

* logback-core-1.2.3.jar

Vulnerabilities:

* + cpe:2.3:a:qos:logback:1.2.3:\*:\*:\*:\*:\*:\*:\*

Description & Recommended Solutions:

A Logback-core module. In log back version 1.2.7 and prior versions, an attacker with the required privileges to edit configuration files could craft a malicious configuration, allowing the execution of arbitrary code loaded from LDAP servers. This issue can be resolved by upgrading to versions 1.3.9 and 1.4.9.

Attribution: CVE-2021-42550

* snakeyaml-1.25.jar

Vulnerabilities:

* + cpe:2.3:a:snakeyaml\_project:snakeyaml:1.25:\*:\*:\*:\*:\*:\*:\*

Description & Recommended Solutions:

Snakeyaml is a YAML 1.1 parser and emitter for Java. SnakeYaml's Constructor() class does not restrict types that can be instantiated during deserialization. Deserializing YAML content provided by an attacker can lead to remote code execution. The recommended option is using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization. We recommend upgrading to version 2.0 and beyond. This issue can be resolved using constructors such as the new YAML(). version 2.0 and beyond. This issue can be resolved by using constructors such as new YAML().

Attribution: CVE-2022-1471, CVE-2017-18640, CVE-2022-25857, CVE-2022-38749, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854, CVE-2022-38750

* spring-boot-2.2.4.RELEASE.jar & spring-boot-starter-web-2.2.4.RELEASE.jar

Vulnerabilities:

* + cpe:2.3:a:vmware:spring:2.2.4:release:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:vmware:spring\_boot:2.2.4:release:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:web\_project:web:2.2.4:release:\*:\*:\*:\*:\*:\*

Description & Recommended Solutions:

In Spring Boot versions 3.0.0 - 3.0.5, 2.7.0 - 2.7.10, and older unsupported versions, an application deployed to Cloud Foundry could be susceptible to a security bypass. Users of affected versions should apply the following mitigation: 3.0.x users should upgrade to 3.0.6+. 2.7.x users should upgrade to 2.7.11+. Users of older, unsupported versions should upgrade to 3.0.6+ or 2.7.11+. Software fixes will be available through the NetApp Support website in the Software Download section.

Attribution: CVE-2023-20873, CVE-2022-27772, CVE-2023-20883

* spring-web-5.2.3.RELEASE.jar & spring-webmvc-5.2.3.RELEASE.jar

Vulnerabilities:

* + cpe:2.3:a:pivotal\_software:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:springsource:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:vmware:spring:5.2.3:release:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:vmware:spring\_framework:5.2.3:release:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:web\_project:web:5.2.3:release:\*:\*:\*:\*:\*:\*

Description & Recommended Solutions:

Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The exploit requires the application to run on Tomcat as a WAR deployment. If the application is deployed as a Spring Boot executable jar, i.e., the default, it is not vulnerable to the exploit. However, vulnerability is more general, and other ways to exploit it may exist. Issues can be resolved by upgrading to the most current version.

Attribution: CVE-2016-1000027, 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 & tomcat-embed-websocket-9.0.30.jar

Vulnerabilities:

* + cpe:2.3 :2.3:a:apache:tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*
  + cpe:2.3:a:apache\_tomcat:apache\_tomcat:9.0.30:\*:\*:\*:\*:\*:\*:\*

Description & Recommended Solutions:

Tomcat treats AJP connections as similar to HTTP connections, and if the links are available to an attacker, there is a high chance of the system being exploited. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50, and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. Also, if the web application allowed file upload and stored those files within it, it would be possible to remote code to process a file. Another issue was that mitigation was only required if an AJP port was accessible to untrusted users. The issue can be resolved by upgrading to Apache Tomcat 10.0.6 or later.

Attribution: 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-2023-41080, CVE-2021-24122, CVE-2021-33037, CVE-2019-17569, CVE-2020-1935, CVE-2020-13943, CVE-2023-28708, CVE-2021-43980

## Mitigation Plan

To mitigate the identified vulnerabilities, update software dependencies, use secure communication protocols, and conduct regular security assessments. Updating the software dependencies such as Bouncy Castle, Hibernate's Bean Validation, Apache Log4j, Spring, and Apache Tomcat can help to improve security for Artemis Financial because it will prevent newly disclosed vulnerabilities and save developers and security teams a lot of time and work. Using secure communication protocols, such as improper constructor uses like SnakeYaml, can help communication platforms reduce the risk of data breaches, cyber-attacks, and other security incidents. Lastly, regular security assessments can ensure sensitive data is secured in the local environment. Routine security assessment can protect databases, servers, and cloud storage from potential threats and attacks.