****

# 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 ……………………………………………………………………………………………………………………………………………… 8

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **05/21/2023** | **Micheal White** |  |

## 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

Micheal White

## Interpreting Client Needs

Artemis Financial offer individualized financial planning through services including savings, retirement, insurance, and investments. The company wants to modernize its software security to protect itself and its customers. Though not stated explicitly in the summary, the company offers numerous financial products which are likely to include interactions with international organizations. Therefore, the software must be compliant with several laws including SOX (financial recordkeeping) and GLBA (financial information management) in the United States and the GDPR (customer information privacy) in the European Union. The software also will need to be compliant with the PCI SSC to process credit cards securely and how to encrypt the information when being stored or transferred over a network.

As a financial services company, Artemis Financial would be targeted by threat actors who want money either stealing from customers or ransoming company data or personal identifying information including banking and payment information, SSNs, addresses, usernames, and passwords. We must take these considerations into account when developing secure access controls. We need to review the company’s API for vulnerabilities and to identify third party libraries for security that meet the needs of the company. We need to verify that all software and libraries used are up-to-date as well to modernize operations.

## Areas of Security

Input validation is a key concern when dealing with sensitive information. Only those users authorized should access only the information they are designated and prevent bad actors from gaining access to data. API interactions need to be secure, and code written to prevent attacks due to vulnerabilities in third party libraries. Thus, all requests must be made securely and consistently across all devices on the network. Cryptography should be implemented since Artemis Financial is handling sensitive information no one wants compromised. To support these considerations, we must develop quality code written using security best practices and effective patterns. We also must ensure errors are handled correctly to prevent unauthorized access.

## Manual Review

The first issue identified in the code is a lack of access control within the “GreetingController.” Our API has several opportunities. “CRUDController” and “DocData” classes make database calls but have not implemented parameterization to prevent SQL injection attacks. User inputs in the controller classes are vulnerable by not being routed through the POST method.

The overall code quality is sufficient; however, error handling and cryptography are not present. Checking the POM file, the dependencies being used are older and need to be updated to the latest, secure releases. Also, there was no evidence of validation plugins.

## Static Testing

Below is a summary of the static test:

|  |  |  |  |
| --- | --- | --- | --- |
| Dependency | Vulnerability | Description | Solution |
| bcprov-jdk15on-1.46.jar | 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 | 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. | Update to version 1.73 |
| spring-boot-2.2.4.RELEASE.jar | 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 | \*\* UNSUPPORTED WHEN ASSIGNED \*\* spring-boot versions prior to version v2.2.11.RELEASE was vulnerable to temporary directory hijacking. This vulnerability impacted the org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir method. NOTE: This vulnerability only affects products and/or versions that are no longer supported by the maintainer. | Update to latest Spring Boot 3.0.6 |
| logback-core-1.2.3.jar | 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 | In logback version 1.2.7 and prior versions, an attacker with the required privileges to edit configurations files could craft a malicious configuration allowing to execute arbitrary code loaded from LDAP servers. | Update to Logback 1.3 for Java EE or 1.4 for Jakarta EE |
| log4j-api-2.12.1.jar | 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 | Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This 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. Fixed in Apache Log4j 2.12.3 and 2.13.1 | Update to version 2.13.2 |
| snakeyaml-1.25.jar | 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 | The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564. | Upgrade to snakeyaml 1.26 |
| jackson-databind-2.10.2.jar | 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 | A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity. | Update to latest version with vulnerability patched |
| tomcat-embed-core-9.0.30.jar | 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 | The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely. | Upgrade to Tomcat 10.1.8 |
| hibernate-validator-6.0.18.Final.jar | 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 | 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. | Upgrade Hibernate to 6.0.20 |
| spring-web-5.2.3.RELEASE.jar | 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 | Pivotal Spring Framework through 5.3.16 suffers from a potential remote code execution (RCE) issue if used for Java deserialization of untrusted data. Depending on how the library is implemented within a product, this issue may or not occur, and authentication may be required. NOTE: the vendor's position is that untrusted data is not an intended use case. The product's behavior will not be changed because some users rely on deserialization of trusted data. | Upgrade to Spring Framework 6.0.2 |
| spring-beans-5.2.3.RELEASE.jar | 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 | A Spring MVC or Spring WebFlux application running on JDK 9+ may be vulnerable to remote code execution (RCE) via data binding. The specific 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, the nature of the vulnerability is more general, and there may be other ways to exploit it. | Upgrade to Spring Framework 6.0.2 |
| spring-webmvc-5.2.3.RELEASE.jar | 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 | In Spring Framework versions 5.3.0 - 5.3.13, 5.2.0 - 5.2.18, and older unsupported versions, it is possible for a user to provide malicious input to cause the insertion of additional log entries. This is a follow-up to CVE-2021-22096 that protects against additional types of input and in more places of the Spring Framework codebase. | Upgrade to Spring Framework 6.0.2 |
| spring-context-5.2.3.RELEASE.jar | 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 | In Spring Framework versions 5.3.0 - 5.3.18, 5.2.0 - 5.2.20, and older unsupported versions, the patterns for disallowedFields on a DataBinder are case sensitive which means a field is not effectively protected unless it is listed with both upper and lower case for the first character of the field, including upper and lower case for the first character of all nested fields within the property path. | Upgrade to Spring Framework 6.0.2 |
| spring-expression-5.2.3.RELEASE.jar | 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 | n Spring Framework versions 5.3.0 - 5.3.16 and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial of service condition. | Upgrade to Spring Framework 6.0.2 |

## Mitigation Plan

The first step in our mitigation plan is to update all our dependencies including Snakeyaml, Hibernator, Apache Tomcat, bouncycastle, and Spring to the latest versions. We then need to implement strong access controls and error handling measures using these libraries and frameworks to secure the code. For example, using Java’s built-in PreparedStatement class will prevent SQL injections.