****

# Artemis Financial Vulnerability Assessment Report

**Table of Contents**

[Document Revision History 3](#_30j0zll)

[Client 3](#_1fob9te)

[Instructions 3](#_3znysh7)

[Developer 4](#_2et92p0)

[1. Interpreting Client Needs 4](#_tyjcwt)

[2. Areas of Security 4](#_3dy6vkm)

[3. Manual Review 4](#_1t3h5sf)

[4. Static Testing 4](#_4d34og8)

[5. Mitigation Plan 4](#_2s8eyo1)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **March 31, 2024** | **Celeste Wally** | **1st,2nd,3rd ,4th, 5th part done** |

## Client



## Developer

Celeste Wally

## Interpreting Client Needs

Artemis Financial requires a secure modernization of their web-based software application to protect against external threats. The following points address their needs:

* Secure Communications: For a financial consulting company, secure communications are paramount to protect sensitive client data during transit. The value here is twofold: maintaining client trust and meeting regulatory compliance standards.
* International Transactions: If Artemis Financial engages in international transactions, it must adhere to global standards like GDPR and possibly deal with varying encryption standards and restrictions by country.
* Governmental Restrictions: As financial data is often subject to stringent regulations, the company must ensure all communications meet local and international laws, such as encryption standards and data sovereignty.
* External Threats: Current threats include phishing attacks, malware, ransomware, and advanced persistent threats (APTs). Future threats could evolve with technology, requiring constant vigilance and updates to security measures.
* Modernization Requirements: Incorporating the latest security practices is crucial. This involves considering the use of open-source libraries and their vulnerabilities, updating web application technologies, and applying secure coding practices.

## Areas of Security

Relevant areas of security for Artemis Financial’s web application are:

* Data Protection and Encryption: To ensure confidentiality and integrity of financial data, especially for data at rest and in transit.
* Authentication and Authorization: To control access to resources, ensuring only legitimate users can perform actions on the application.
* Input Validation: To prevent common web vulnerabilities like SQL injection, which could lead to data breaches.
* Error Handling: To avoid leakage of sensitive information through error messages and logs.
* Configuration Management: To manage application configurations securely, preventing unauthorized access or manipulation.
* Dependency Management: Ensuring that all third-party components are up to date and free from known vulnerabilities.

## Manual Review

**Vulnerabilities Found:**

* Authentication/Authorization: There's no evidence of an authentication or authorization framework within the application's code, presenting a risk of unauthorized access.
* Hardcoded Credentials: DocData.java contains hardcoded database credentials, a high risk for database security.
* Input Validation: The CRUDController.java lacks input validation, making it vulnerable to injection attacks.
* Error Handling: Use of e.printStackTrace() in DocData.java for error handling can expose system details to an attacker.
* Encryption: There's no implementation of encryption for data in transit or at rest, risking data exposure.

## Static Testing

**Vulnerabilities Identified:**

The dependency check reveals several critical and high-severity vulnerabilities in the third-party libraries used within Artemis Financial's application. The following summarizes the key points from the static analysis:

Bouncy Castle Crypto (bcprov-jdk15on-1.46.jar): Multiple CVEs with HIGH severity were found. These include vulnerabilities related to cryptographic issues, information exposure, and improper verification of a cryptographic signature.

**Key CVEs: CVE-2013-1624, CVE-2015-6644, CVE-2015-7940, CVE-2016-1000338, CVE-2016-1000339.**

Spring Boot (spring-boot-2.2.4.RELEASE.jar): CRITICAL vulnerabilities detected, which can lead to security issues like remote code execution.

**Key Vulnerabilities: CVEs specific to Spring Boot 2.2.4.RELEASE.**

Logging Libraries (logback-core-1.2.3.jar, log4j-api-2.12.1.jar): Identified vulnerabilities that can lead to information leakage and unauthorized access.

Data Binding (jackson-databind-2.10.2.jar): HIGH severity issues found that could allow an attacker to perform unauthorized actions on the system.

**Key Vulnerabilities: CVEs associated with jackson-databind version 2.10.2.**

Embedded Tomcat (tomcat-embed-core-9.0.30.jar): Multiple CRITICAL vulnerabilities that could potentially be exploited to compromise the web server.

**Key Vulnerabilities: CVEs linked to Apache Tomcat 9.0.30.**

Validation (hibernate-validator-6.0.18.Final.jar): MEDIUM severity vulnerabilities that could impact the integrity of data validation processes.

**Key Vulnerabilities: CVEs relevant to hibernate-validator version 6.0.18.Final.**

Spring Framework Jars (spring-web, spring-beans, spring-webmvc, spring-context, spring-expression): Various HIGH and MEDIUM vulnerabilities, which can impact different aspects of the application, from MVC components to the core context.

## Mitigation Plan

Upgrade to non-vulnerable software versions as recommended by the vulnerability details.

Apply available patches from the software vendors.

Implement workarounds where upgrades or patches are not feasible, adhering to vendor advice.

**Static Testing and Hardening**

Conduct static analysis of the codebase to identify potential security weaknesses.

Implement hardening measures such as removing unused features, restricting access permissions, and input validation to prevent exploitation.

**Continuous Monitoring and Review**

Employ real-time monitoring to detect abnormal behavior indicative of a breach.

Regularly review systems and software for new vulnerabilities.

**Documentation and Knowledge Sharing**

Document the mitigation process and decisions for future reference.

Share knowledge about vulnerabilities and their mitigation within the team.

**Incident Response Planning**

Develop or update incident response plans to include these vulnerabilities and associated indicators of compromise (IoCs).

Now let’s apply this strategy

**spring-boot CVE-2022-27772, CVE-2023-20873, CVE-2023-20883:**

Upgrade Spring Boot to the latest, non-vulnerable version.

Verify temporary directories have appropriate permissions to prevent hijacking.

Use a supported version of Spring Boot and avoid deploying to vulnerable Cloud Foundry environments.

**logback CVE-2021-42550, CVE-2023-6378:**

Upgrade to a non-vulnerable version of logback.

Remove or restrict access to the logback configuration file.

**log4j CVE-2020-9488, CVE-2021-44228, CVE-2021-44832, CVE-2021-45046, CVE-2021-45105:**

Upgrade to log4j version 2.16.0 or higher where the JNDI features that enable this vulnerability are disabled by default.

Ensure that no user-controlled data can be used in log message format or parameters.

**snakeyaml CVE-2017-18640, CVE-2021-4235, CVE-2022-1471, CVE-2022-25857, CVE-2022-3064, CVE-2022-38749, CVE-2022-38750, CVE-2022-38751, CVE-2022-38752, CVE-2022-41854:**

Upgrade to version 1.26 or newer, which includes a fix for the reported vulnerability.

Avoid parsing untrusted YAML content or use SafeConstructor for parsing.

**jackson-databind CVE-2020-25649, CVE-2020-36518, CVE-2021-46877, CVE-2022-42003, CVE-2022-42004, CVE-2023-35116:**

Upgrade to a non-vulnerable version of jackson-databind.

Avoid deserializing data from untrusted sources.

**Apache Tomcat CVE-2019-17569, CVE-2020-11996, CVE-2020-13934, CVE-2020-13935, CVE-2020-13943, CVE-2020-17527, CVE-2020-1935, CVE-2020-1938, CVE-2020-8022, CVE-2020-9484:**

Update to the latest version of Tomcat that has remediated these vulnerabilities.

Ensure AJP connectors are disabled if not in use and secure any AJP usage with secret attributes.