****

# 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** | **5/21/2023** | **Jennifer Wells** |  |

## 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

Jennifer Wells

## Interpreting Client Needs

Secure communications are highly valuable to financial consulting companies. They protect sensitive client information, ensure regulatory compliance, safeguard intellectual property, mitigate cyber threats, and enhance client trust. Secure channels facilitate both internal and external communication, enabling the company to maintain confidentiality and professionalism in handling financial data.

There are governmental restrictions that need to be considered regarding secure communications, though they may vary depending on the country or region in which the company operates. These include data protection and privacy laws, financial regulations, export control regulations, and national security laws. Compliance with these regulations is essential to ensure the secure handling of client data, encryption practices, and cooperation with law enforcement agencies when required. Staying informed and working with legal and compliance teams is crucial for adherence to these governmental restrictions.

Companies face various external threats in secure communications. These include cyberattacks, data breaches, ransomware, social engineering, Insider threats, advanced persistent threats (APTs), IoT vulnerabilities, and regulatory changes.

Modernization requirements for secure communications include versioning and backward compatibility, allowing existing clients to continue using the API while new clients can benefit from the latest features, implementing proper error handling mechanisms with meaningful error messages and appropriate HTTP status codes. Also, incorporate logging functionality to capture errors, exceptions, and relevant information for troubleshooting and analysis, rate limiting and throttling to control usage and prevent abuse of the API, performance monitoring and analytics, and automated testing and continuous integration.

## Areas of Security

## Manual Review

The Spring Boot REST controller named "CRUDController" does not perform any validation or sanitization on the "business\_name" query parameter. It does not handle potential errors or exceptions that may occur during the execution of the "DocData" object or the creation of the "CRUD" instance. It also does not include any authentication or authorization mechanisms. Without proper authentication, any can access the "/read" endpoint and retrieve data, which may pose a security risk if the endpoint provides sensitive or confidential information.

In the Java class named "customer", there is a lack of encapsulation and access modifiers. The class variables "account\_number" and "account\_balance" are declared without access modifiers, making them accessible to any class within the same package. This can lead to unauthorized access or modification of sensitive customer data.

The "deposit" method adds the amount directly to the "account\_balance" variable without any validation or authorization checks. It does not perform any input validation on the "deposit" method's parameter "a." Failing to validate user input can expose the application to security risks such as integer overflows, invalid input, or abuse of the system.

In the Java class named "DocData", it does not demonstrate any input validation or the use of prepared statements when interacting with the database. Failing to validate and sanitize user input or not using prepared statements can make it susceptible to SQL injection attacks, where malicious users can manipulate the input to execute unauthorized or harmful SQL queries. There is hard-coded database credentials directly in the connection string. Storing credentials in plain text within the source code is insecure and exposes them to potential unauthorized access. There is also a catch block that prints the stack trace of any caught exception but does not provide meaningful error handling or logging. Proper error handling is crucial to maintain the confidentiality, integrity, and availability of the system and to prevent sensitive information leakage.

## Static Testing

Version 1.46 of Bouncy Castle's Legion of the Bouncy Castle Java Cryptography API has several vulnerabilities, including issues with ECB mode, weak key generation, padding oracle attack vulnerability, and other security weaknesses. These vulnerabilities were identified in June 2018. Update Bouncy Castle library to a newer, more secure version. By upgrading to a version beyond 1.46, bug fixes, security patches, and improvements will address the identified vulnerabilities.

CVE-2022-27772 is a vulnerability in older versions of Spring Boot (prior to v2.2.11.RELEASE) that allows temporary directory hijacking. This vulnerability only affects unsupported versions of Spring Boot. Upgrading to a supported version is recommended to mitigate this issue. The vulnerability was published on March 30, 2022.

CVE-2022-27772 is a vulnerability in unsupported versions of Spring Boot (prior to v2.2.11.RELEASE) that allows temporary directory hijacking. It affects the **org.springframework.boot.web.server.AbstractConfigurableWebServerFactory.createTempDir`** method. Upgrading to a supported version is necessary to fix this issue. Published on March 30, 2022.

CVE-2021-42550 is a vulnerability found in logback versions 1.2.7 and earlier. It allows an attacker with sufficient privileges to modify configuration files to create a malicious configuration. This malicious configuration can be used to execute arbitrary code that is loaded from LDAP servers. It is recommended to update logback to a version beyond 1.2.7 to mitigate this vulnerability.

Apache Log4j has multiple vulnerabilities, including remote code execution (RCE), denial-of-service, information leakage, and certificate validation issues. These vulnerabilities affect various versions of Log4j and can potentially allow attackers to execute arbitrary code, cause service disruptions, leak sensitive information, or intercept log messages. It is recommended to update to the fixed versions to mitigate these vulnerabilities.

Multiple vulnerabilities have been identified in the Jackson-databind library by FasterXML. These vulnerabilities can lead to denial of service attacks and resource exhaustion. It is recommended to update to the latest versions to mitigate these risks.

Apache Tomcat has several vulnerabilities, including issues with session cookies, request smuggling, concurrency bugs, cross-site scripting, documentation flaws, TLS packet validation, HTTP transfer-encoding parsing, JNDI Realm protection bypass, incomplete fixes, JSP source code disclosure, information leakage, resource response issues, WebSocket frame payload length validation, memory exhaustion, denial of service, and privilege escalation.To enhance the security of Apache Tomcat,

keep Tomcat updated, enable secure session cookies, follow secure coding practices, Secure the server environment, Use TLS/SSL encryption, Disable unnecessary features, regularly audit and monitor the server and follow general security best practices.

A vulnerability has been found in Hibernate Validator version 6.1.2.Final, used in Red Hat products, allowing attackers to bypass input sanitation controls in error messages.To address the vulnerability in Hibernate Validator version 6.1.2.Final, the recommended solution is to update to a newer version of Hibernate Validator that includes a fix for the issue.

## Mitigation Plan

To mitigate vulnerabilities related to sanitization, authentication and authorization, user input should be validated and sanitized, implementing the use of parameterized queries, apply output encoding, enforce secure password policies, use RBAC or ABAC for access control and practice least privilege, secure session management and utilize secure communication protocols, conducting regular security testing and coding reviews and keeping software dependencies up to date to address known vulnerabilities.

To mitigate vulnerabilities related to a lack of encapsulation and access modifiers, implementing proper encapsulation by defining private fields and providing public getter and setter methods for accessing and modifying the internal state of an object, using appropriate access modifiers such as private, protected, and public to restrict access to class members based on their intended usage, and using access modifiers consistently throughout the codebase.

To mitigate input validation security vulnerabilities, implement input validation on both client and server sides, define input patterns and rejecting what doesn't conform, sanitize input by removing or encoding potentially malicious characters or sequences, rely on server-side input validation as the primary defense, while client-side validation can improve user experience, applying context-specific validation rules based on the intended use of the input, and using secure coding practices, frameworks and keep software dependencies updated and patched to benefit from security fixes.

A comprehensive mitigation plan to ensure overall security includes regular updates and Patch Management, Vulnerability Management assessment tools, Implementing network security measures, following secure configuration practices, enforcing strong access controls, regular security audits and having an incident response plan.