****

# 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** | **Bryce Burroughs** |  |

## 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

Bryce Burroughs

## Interpreting Client Needs

1. Secure communication is paramount to Artemis Financial. Given the nature of financial planning, ensuring confidentiality and integrity of data during transmission is essential to protect customer data from eavesdropping or tampering.
2. Artemis Financial does engage in international transactions and are therefore subject to international regulations in regards to secure communications and data protection.
3. Governmental regulations may apply to secure communications, data storage, and handling of financial data. Compliance with these regulations is mandatory for Artemis Financial.
4. Potential threats that Artemis Financial must consider include:
   1. Malicious actors attempting to steal sensitive customer financial information
   2. DDoS attacks
   3. Exploitation of software vulnerabilities
   4. Insider threats, such as disgruntled employees
5. Modernization requirements include using secure and up-to-date open source libraries to minimize vulnerabilities and staying current with evolving web application technology to benefit from security enhancements.

## Areas of Security

Input validation: Input validation is critical to prevent common web vulnerabilities such as SQL injection and XSS attacks. Given the web based nature of the application, ensuring that user inputs are validated and that outputs are properly encoded is necessary to prevent malicious manipulation of data, which will in turn prevent external threats.

APIs: Since the web-based applications uses RESTful APIs, it is important to secure the interactions that take place between these services.

Cryptography: Encryption of customer data is crucial to ensure that customer’s financial information stays confidential and retains the integrity of this sensitive data.

Error handling: Secure error handling is important in order to minimize information leakage. This ensures that error messages and stack traces do not reveal sensitive information in regards to the application’s internal workings.

Encapsulation: Encapsulation will help with data protection, as it allows us to hide sensitive data within objects and control access to that data with well-defined interfaces, such as setters and getters.

## Manual Review

1. The DocData class interacts with a database, but there is not a method for input validation or sanitization, which could lead to SQL injection vulnerabilities.
2. The DocData class also hardcodes connection credentials into the code, rather than being saved in a secure config file.
3. There is no code in these classes that relates to authentication or authorization.
4. There is a lack of error handling and HTTPS

## Static Testing

## bcprov-jdk15on-1.46.jar: ECDSA doesn’t fully validate ASN.1 encoding of signature on verification, making it possible to inject extra elements in the sequence and have it validate which may allow the introduction of ‘invisible’ data. This dependency alone makes up many of the vulnerabilities in this application, which includes many cryptography-related vulnerabilities. Recommended action: update the dependency to latest version

1. hibernate-validator-6.0.18.Final.jar: A bug in the message interpolation processor enables invalid EL expressions to be evaluated incorrectly. This allows attacks to bypass input sanitation. Recommended action: update the dependency to latest version
2. jackson-databind-2.10.2.jar: Many vulnerabilities that allow XXE attacks, as well as a Java StackOverflow exception and denial of service. Biggest risk is to data integrity. Recommended action: update the dependency to the latest version
3. log4j-api-2.12.1.jar: Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. Could allow connection to by intercepted. Update dependency to resolve vulnerability
4. logback-core-1.2.3.jar: An attacker with the necessary privileges to edit configurations could craft a malicious file allowing execution of arbitrary code. Update dependency to resolve vulnerability
5. snakeyaml-1.25.jar: Constructor() class does not restrict types which can be instantiated. Upgrade to version 2.0 or higher to resolve vulnerability
6. spring-boot-2.2.4.RELEASE.jar: In this Spring Boot version and older versions, an application deployed to Cloud Foundry could be susceptible to security bypass. Upgrade dependency to resolve vulnerability.
7. spring-core-5.2.3.RELEASE.jar: A Spring MVC or WebFlux application running on JDK 9+ may be vulnerable to remote code execution. Update dependency to resolve vulnerability.
8. spring-web-5.2.3.RELEASE.jar: Similar to Spring-core-5.2.3. Suffers from potential remote code execution if used for Java deserialization. Update dependency to resolve.
9. tomcat-embed-core-9.0.30.jar: Tomcat treats AJP connections as having higher trust than a similar HTTP connection, which could allow an attacker to exploit this. Recommended to disable this connector if not required.

## Mitigation Plan

After reviewing all of the vulnerabilities discovered by the static check, it is advised that we update all dependencies to their most recent version to remove any vulnerabilities that exist in older versions of these dependencies. Per the manual review, we should also address the SQL Injection Vulnerability in the DocData class, remove the hardcoded connection credentials, and implement authentication and authorization mechanisms in order to control access to sensitive data. The final step we should take is determine whether the tomcat core is essential to the application, and disable the AJP connector if it is not.