

**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** | **7/15/23** | **Phillip Hogan** |  |

## Client



## Developer

Phillip Hogan

## 1. Interpreting Client Needs

After concluding our meeting with Artemis Financial, we will have to make sure secure communications, authorization and authentication, database security, multitenancy role base security and access control will need to either be incorporated or checked to properly secure their web application. Secure communication of data is critical to protect their customers and the company. If not properly applied, they run the risk of allowing unapproved access to the application or untrusted data entering the system. In the event of international transactions, the company must make sure their security follows European Union so that they do not encounter any fines for not protecting personal data. When considering American transactions, they must comply with the Graham-Leach-Bliley Act to protect consumer data. External threat that are present now or in the immediate future are potential hackers looking to breach the application to either steal consumer information and exploit funds within accounts. As for modernization requirements that should be considered, we should look to have to latest trusted updates for viable libraries we intend to use for the application. We should also, stay up to date on current best practices when it comes to secure coding. Finally, hacker attacks evolve over time, we should also stay up to date on these attacks to stay aware of what type of security we need to implement for future application upgrades.

## 2. Areas of Security

Input Validation: This is a pivotal part of security that needs to be applied so that unauthorized user does not obtain access to the system.

APIs: Ensuring that we have secure communication between servers and the application is key. Also, communication between the database and the application is a key factor to eliminate any vulnerabilities regarding this communication.

Cryptography: This area is important due the fact that customer data is being transferred back and forth between the customer and the application. Encrypted PII or critical information is very important on this type of platform.

Client/Server: This is important because the telecommunication connection needs to be secure, HTTPS or other best practices, so data transfer or authentication stay secure.

Code Error: Error catching is a great defense to make sure that someone is not provided unauthorized access to the system, roles, or data.

Code Quality: Secure Coding best practice such as parameterization is pivotal to ensure that a SQL injection attack will not occur.

## 3. Manual Review

After reviewing the code, it looks like there application is transferring data over HTTP rather than HTTPS. This reduces the security drastically and posed the threat of the application being breached through untrusted input in the URL. There also looks to be no input validation as far as authentication or authorization for any roles. Which could also lead to leak in data or intrusion of the system. As for API interaction, it does not look like the application parameterized any query variables along with no input validation or error handling. This could lead to one of worst hacker attacks, SQL injection. Cryptography is not existent in the code which will lead easy access to customer or admin data. RBAC or ABAC is not defined in code, which gives the appearance that everyone has the same view depending on the input data from the customer. The code quality is basic, with no defined security measures anywhere regarding Input validation, parameterization of variables, firewalls, and transfer protocols. Finally, the dependencies need to be updated to reduce the risk of a security breach.

## 4. Static Testing

* ***bcprov-jdk15on-1.46.jar version***: The Bouncy Castle Crypto package is a java implementation of cryptographic algorithms. In the pom.xml file, the version is out of date. It is currently 1.46 version but could be upgraded to the 1.70 version to help eliminate vulnerabilities.
* ***jackson-databind-2.10.2.jar***: A general data-binding functionality for Jackson. Current version in dependencies is out of data. Any version before version 2.13.0 allows StackOverflow exeption and possibility for a DoS attack. Need to update version to update security measures to protect data.
* ***log4j-api-2.12.1.jar***: Current version improperly validates certificates with host mismatch. Issue was fixed in version 2.13.1. The course of action is to update version to eliminate vulnerabilities.
* ***logback-core-1.2.3.jar***: Current version allows attacker with required privileges to edit configurations for malicious purposes. Update to version after 1.2.7 to eliminate the issue.
* ***spring-boot-2.2.4.RELEASE.jar***: Versions prior to 2.2.11 Release was vulnerable to temporary directory hijacking. Suggested solution is to upgrade to the most recent version to eliminate vulnerabilities.
* ***spring-core-5.2.3.RELEASE.jar***: Spring MVC applications running on JDK 9+ may be vulnerable to remote code execution via data binding. Version extremely susceptible to untrusted data output from attacker that could provide unauthorized access to role base privileges in the system. Including some custom input validation code will help to eliminate issue.
* ***spring-web-5.2.3.RELEASE.jar***: Pivotal Spring Framework through 5.3.16 suffers from potential code execution issue is used for Java deserialization of untrusted data. Consider using HTTPS protocol to help eliminate the situation of being breached by untrusted data.
* ***spring-web-5.2.3.RELEASE.jar***: Current version susceptible to DoS attack. By using AJP, an attacker can exploit the system through different means. Upgrading the version to 9.0.31 will eliminate the issue.
* ***tomcat-embed-websocket-9.0.30.jar***: Current version susceptible to DoS attack. By using AJP, an attacker can exploit the system through different means. Upgrading the version to 9.0.31 will eliminate the issue.

## 5. Mitigation Plan

* Upgrade the dependencies: bcprov-jdk15on-1.46.jar, jackson-databind-2.10.2.jar, log4j-api-2.12.1.jar, logback-core-1.2.3.jar, spring-boot-2.2.4.RELEASE.jar, spring-web-5.2.3.RELEASE.jar, spring-web-5.2.3.RELEASE.jar, and tomcat-embed-websocket-9.0.30.jar
* Incorporate HTTPS data transfer protocol.
* Parameterize variables when incorporating database query in the code.
* Incorporate Input Validation and multitenancy instances capability within system to allow different view and role attributes.
* Incorporate encryption to data once received and authenticated.

# Works Cited

(2022). *Dependency-Check Report.* Dependency-Check.

Manico, J., Detlefsen, A., & Kenan, K. (2014). *Iron-Clad Java: Building Secure Web Applications.* Oracle Press.