****

# 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** | **9/13/2023** | **Rajya Kolluri** | **Finalized Vulnerability Assessment Report** |

## Client



## Developer

Rajya Kolluri

## Interpreting Client Needs

To modernize its business processes, Artemis Financial is looking for the best and most recent software security techniques. They are especially searching for knowledge on how to defend the company from attacks from outside sources.

Artemis Financial values secure communications greatly since they handle consumers' private financial information. Secure communication guarantees the privacy and accuracy of financial planning, which is essential for their company's reputation and client confidence.

The Scenario includes that the software is for entrepreneurs, business, and government agencies “around the world” which does not explicitly mention international transactions but there is a great chance for the company to have international transactions.

Governmental rules or industry-specific compliance specifications pertaining to secure communications and data security may need to be considered by Artemis Financial. The scenario mentions that their software is also for government agencies around the world.

External threats to Artemis Financials’ web-based software application could include data breaches where there is unauthorized access to sensitive customer financial data, phishing attacks, and many others.

Strategies to modernize Artemis Financials’ web application include the use of open-source libraries for potential vulnerabilities and to ensure that they are up to date.

## Areas of Security

I have determined the following security concerns that relate to the web-based software application used by Artemis Financial based on the information offered and a review of the program. These topics are pertinent to the software program because they deal with critical security elements that are necessary for protecting private financial data and guaranteeing the availability and integrity of the application:

1. APIs: In the scenario the client mentioned using the third-party gateways which is why the API section would apply. It is described that Artemis Financial has a RESTful web application programming interface (API). The application interacts with external services or APIs, as secure communication, authentication, and authorization are crucial for data protection.
2. Cryptography: The goal is to use the most current and effective software security. The application handles sensitive data that should be encrypted, such as financial information. Proper encryption is essential for data protection.
3. Code Quality: The main objective is to examine web-based software applications, modernizing their operations and ensuring software security. This relies on secure coding practices and patterns that are essential for writing a robust and secure code.
4. Encapsulation: Artemis Financial are seeking Global Rain’s expertise in how to protect the organization from external threats. Secure data structures ensure that sensitive information is properly protected and accessed only by authorized parties.

## Manual Review

The codebase consists of several Java classes that make up Artemis Financials’ web-based software application. Starting off with the first security area in the vulnerability assessment diagram, “Input Validation”, there is no explicit input validation to be implemented in the code nor is mentioned in the scenario.

The code doesn’t have any third parties included but the scenario mentioned that the company has a RESTful web application programming interface (API). If the application interacts with external services, it's essential to ensure that “Secure API” authentication and communication practices are followed.

Several significant security and coding quality concerns have been observed during the code review of the software application used by Artemis Financial. The program is vulnerable to unauthorized access due to the first obvious lack of authentication methods.

“Cryptography” operations are not evident in the provided code. Depending on the application's requirements, encryption and secure key management should be considered when handling sensitive data. Furthermore, sensitive data processing lacks encryption, which raises questions regarding the security of private data.

The code samples use almost no error management techniques. It is advisable to design thorough “error handling” to give consumers helpful error messages without disclosing technical information.

Variable naming conventions are poor, with ambiguous names that make it difficult to interpret code. The code lacks functionality in addition to having several declared but unfinished functions. Additionally, it is unsafe to hardcode credentials straight into the code, such as "root," and secure credential management procedures should be used instead. Maintainability of the code depends on methods for writing high-quality code, such as uniform formatting, comments, and documentation. “Code quality” may be improved by ensuring standards compliance.

Finally, using specialized database user accounts with restricted capabilities is advised rather than connecting to the database using root access credentials. To improve the application's security, maintainability, and general quality, these problems must be fixed. Secure distributed computing involves ensuring that communication between “client and server” components is secure.

The usernames and passwords are mentioned freely in the code moreover, both usernames and passwords are same which makes the system weak. While encapsulation is a fundamental principle, it's not explicitly demonstrated in the code snippets. “Encapsulation” is critical for data protection, especially for handling sensitive financial data.

## Static Testing

I performed a dependency check on Artemis Financial’s software application to identify all security vulnerabilities in the code. It scanned the code base for dependencies and produced an HTML report. Here is a screenshot of the resulting HTML report that identifies all vulnerabilities in the code base by analyzing results from running the code through a static test:

A screenshot of a computer

Description automatically generated

1. bcprov-jdk15on-1.46.jar : 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.
2. hibernate-validator-6.0.18.Final.jar: Hibernate's Bean Validation (JSR-380) reference implementation.
3. jackson-databind-2.10.2.jar: General data-binding functionality for Jackson: works on core streaming API
4. log4j-api-2.12.1.jar : The Apache Log4j API
5. logback-core-1.2.3.jar: logback-core module
6. snakeyaml-1.25.jar: YAML 1.1 parser and emitter for Java
7. spring-boot-2.2.4.RELEASE.jar: Spring Boot
8. spring-boot-starter-web-2.2.4.RELEASE.jar: Starter for building web, including RESTful, applications using Spring MVC. Uses Tomcat as the default embedded container
9. spring-core-5.2.3.RELEASE.jar: Spring Core
10. spring-web-5.2.3.RELEASE.jar: Spring Web
11. spring-webmvc-5.2.3.RELEASE.jar: Spring Web MVC
12. tomcat-embed-core-9.0.30.jar: Core Tomcat implementation
13. tomcat-embed-websocket-9.0.30.jar: Core Tomcat implementation

Several significant vulnerabilities in the project's dependencies have been discovered after a thorough study of the Dependency-Check report. The project's source contains a number of libraries and other components that are vulnerable. These vulnerabilities have moderate to high severity ratings, which indicate a possible danger to the application's stability and security. A diverse strategy is advised to address these vulnerabilities, which includes upgrading out-of-date dependencies, applying available patches, and, if required, taking into account alternative libraries. To further guarantee that the project is robust to new security threats and that the codebase is safe and current, ongoing monitoring and preventative measures will be implemented.

Analyzing the results of the dependency check is crucial in order to identify the best choices for resolving dependencies in the codebase. It is critical to assess the severity of any vulnerabilities associated with each highlighted dependency, as well as any possible impacts on the project's security and the availability of patches or updates. Prioritization should be done based on the severity of the vulnerabilities, with the most important issues receiving top consideration. When vulnerabilities are present in unsupported or outdated versions, it is encouraged to upgrade to more recent, secure versions. Examining industry-standard resources like the Common flaws and Exposures (CVE) database and the National Vulnerability Database is also advised to gain a deeper knowledge of the flaws and possible fixes.

## Mitigation Plan

Several major security flaws and areas that need quick repair were discovered during the evaluation of Artemis Financial's software application. Unauthorized access and security breaches might potentially result from the application's lack of effective user authentication measures, which is a big problem. Establishing user roles and access limits as well as adopting strong user authentication methods, such OAuth or JWT, are crucial steps in reducing this risk.

The first step is to update all vulnerable dependencies to their latest secure versions. This includes dependencies like bcprov-jdk15on-1.46.jar, snakeyaml-1.25.jar, Spring dependencies, and Tomcat-related dependencies. This will ensure that known security issues are resolved.

Lack of encryption for sensitive information like usernames and passwords exposes this data to exploitation, which is a serious concern. Strong encryption techniques for data storage and transmission and safe encryption key management should be included in the application to handle this.

Additionally, the code demonstrates poor coding standards, such as ambiguous variable names and unfinished procedures, which might inhibit code readability and functionality. To increase code quality, incomplete methods must be finished, and variable names must be refactored to be more informative.

Finally, the usage of root database access and hardcoded credentials provide significant security issues. The application should be set up to utilize a specialized database user with restricted rights rather than root access for database connections. Remove any direct dependencies on the root user and adhere to the principle of least privilege. This reduces the risk of unauthorized access and potential system vulnerabilities.

Perform thorough testing, including penetration testing and vulnerability scanning, to validate the effectiveness of security measures and identify any overlooked vulnerabilities.

By following this action list, the identified security vulnerabilities can be effectively mitigated, enhancing the overall security posture of the application and reducing the risk of potential threats and attacks.

In conclusion, it is essential to solve these weaknesses in order to improve the security and dependability of Artemis Financial's software application. This may be done by effective authentication, encryption, code quality improvements, and safe credential management. These steps will lower the possibility of security lapses greatly and enhance overall application performance.