****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **02/26/2023** | **Lewis Quick** | **Updated All Sections** |

## Client



## 

## Developer

Lewis Quick

## Algorithm Cipher

Artemis Financial would like to utilize the most effective and current security software to protect client data and financial information on its public web application. Employing secure communication mechanisms is essential to mitigate security vulnerabilities and establish a secure web application. A checksum verification program will provide a layer of security for shared data files on the web application. The checksum verification program utilizes a hash value to authenticate and verify the integrity of shared files. An encryption cipher algorithm is required to encrypt the data and produce the necessary hash value for the checksum verification. Users can compare the generated hash value to a checksum value provided by the source of the data file to ensure the data is authentic and error-free. CRC, MD5, SHA-1, SHA-2, and SHA-3 are common algorithm ciphers that generate hash values for verification programs. However, the SHA-256 variant of the SHA-2 algorithm is the best cipher for the Artemis Financial web application.

Java Security Standard Algorithm Names by Oracle classifies the SHA-256 algorithm as a MessageDigest algorithm, a secure one-way hash function. The MessageDigest algorithm passes a string value through a cryptographic hash function that returns a hash value called Digest. The Digest acts like the data's fingerprint, a unique identifier. The Digest is irreversible as a cryptographic function can not reproduce the original value.

Cryptographic hash algorithms are vulnerable as it is possible to produce the same hash value from two different keys or inputs, known as a collision. The SHA-256 algorithm is collision-resistant, an essential requirement for the cipher algorithm for the verification program. A cipher algorithm must be collision-resistant to prevent unauthorized access to the data file. The SHA-256 algorithm is ideal because the length of the Digest will always be 256 bits regardless of the public key size. A 256-bit size Digest means there are 2256 possible combinations meaning that the probability of a collision or a successful brute-force attack is near impossible.

## Certificate Generation

Text

Description automatically generatedGraphical user interface, text, application

Description automatically generatedCRT File:

## Deploy Cipher

Graphical user interface, text, application

Description automatically generatedChecksum Verification:

## 

## Secure Communications

Secure Webpage:

Text

Description automatically generated

## Secondary Testing

Graphical user interface, text, application, email

Description automatically generatedOWASP Dependency-Check Maven:

Text

Description automatically generatedRefactored Code Build Success:

## Functional Testing

Text

Description automatically generatedA screenshot of a computer

Description automatically generated with low confidenceText

Description automatically generatedRefactored Code Executed Without Errors:

## Summary

Artemis Financial's public web interface implements modern and effective software security to protect its clients. After reviewing the architecture of Artemis Financial's public web interface and the Vulnerability Assessment Process Flow Diagram (VAPFD), I determined the most critical security areas are APIs, Cryptography, Code Error, Code Quality, and Input Validation. Adding layers of security to the web application involves identifying potential security threats, applying security testing protocols, implementing secure coding practices, and adopting industry standard best practices.

Identifying potential threats requires an in-depth analysis of the application's architecture to identify weaknesses and vulnerabilities. Identifying and preventing threats requires a proactive approach to security and anticipating various threats, such as unauthorized access, man-in-the-middle, injection, and denial of service attacks. I reviewed the file structure and identified the files that contain custom code using a web application framework and files that contain code locked away in libraries. I checked the custom code for syntax, logic, and functional errors and used the VAPFD to identify security vulnerabilities. I checked for coding errors that can lead to security vulnerabilities, such as buffer overflow errors, input validation errors, and SQL injection. I utilized static security testing techniques to identify vulnerabilities in the rest of the code locked away in libraries.

Security testing protocols are essential to identify vulnerabilities in the web application. The web application depends on various libraries and APIs and accounts for most of the code. Many libraries are complex and extensive, and manual review of the code in the libraries would be very challenging. Fortunately, tools are available to help identify vulnerabilities associated with library and API dependencies. After manually reviewing and refactoring the application's code, I used a static testing technique called a dependency check. Specifically, I used the OWASP Maven Dependency-Check Plug-In. I ran the dependency check and refactored the code several times before I was satisfied with the outcome of the dependency check report. The first run of the dependency check results are forty-nine dependencies scanned, seventeen vulnerable dependencies, and ninety-two vulnerabilities found. Unsatisfied with the results, I did some CIS work, tweaked some things, and ran it again. The final results are forty-six dependencies scanned, one vulnerable dependency, and one vulnerability found. I am satisfied with the final results because the vulnerability occurs when parsing and deserializing yaml content. The web application does not require a yaml parser because it utilizes a Project Object Model (POM) as a build tool and dependency manager instead of a yaml file for the project configuration. The modifications required to obtain the final dependency check results are changed the Spring Boot framework from 2.2.4.RELEASE to the latest version of 3.0.3, update the application.properties file to enable secure communications and deploy a checksum cipher algorithm to verify shared data.

Implementing secure coding practices will help prevent malicious attacks and threats. Writing secure code, properly handling user input, and avoiding buffer overflow errors will ensure the application is less prone to attacks. I utilized the OWASP Secure Coding Practices Quick Reference Guide to ensure secure coding practices and mitigate common software vulnerabilities. The best practices for secure coding include input validation, error handling, access control, communication security, and cryptography.

It's important to follow industry standard best practices to mitigate against known security vulnerabilities and enhance the company's overall well-being. Industry standard best practices include cryptography for sensitive data, secure communication protocols, and robust password policies. Artemis Financial's public web interface utilizes the @RequestMapping method to create a RESTful route that deploys and returns a checksum verification program called myHash(). The checksum verification program demonstrates industry standard best practices that protect Artemis Financial's client's data and personal information. The checksum verification program demonstrates the use of Transport Layer Security (TLS) as a layer of communication security. The checksum program also indicates that error handling is essential and adds value to the security of the public web interface. Cryptographic practices, output encoding, and input validation are other industry standard best practices displayed by the checksum verification program. Finally, implementing industry best practices save the company from disaster and losses. Implementing industry best practices is cost-effective, saving the company from headaches and other resources like time and money.

As a last-minute thought and no time to figure out where to add these comments, Spring Boot is a fantastic framework to help mitigate security vulnerabilities for a web application. The Spring Boot framework has built-in security features. The built-in features are enabled by default, including CSFR protection, XSS protection, and HTTP Strict Transport Security (HSTS). These features provide authentication, authorization, and session management mechanisms.

References

And Friends, O. (n.d.). *keytool-Key and Certificate Management Tool*. Docs.oracle.com. Retrieved February 27, 2023, from https://docs.oracle.com/javase/1.5.0/docs/tooldocs/solaris/keytool.html

*How to get HTTPS working on your local development environment in 5 minutes*. (2018, January 19). FreeCodeCamp.org. https://www.freecodecamp.org/news/how-to-get-https-working-on-your-local-development-environment-in-5-minutes-7af615770eec/

*How to use HTTPS for local development*. (n.d.). Web.dev. https://web.dev/how-to-use-local-https/

Manico, J., & Detlefsen, A. (2015). *Iron-clad Java building secure web applications*. New York, Ny Mcgraw-Hill Education.

*OWASP Secure Coding Practices-Quick Reference Guide*. (2022, December 1). Owasp.org. https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/

*Secure Coding Guidelines for Java SE*. (2019). Oracle.com. https://www.oracle.com/java/technologies/javase/seccodeguide.html

*Secure Development | Software Engineering Institute*. (n.d.). Www.sei.cmu.edu. Retrieved February 27, 2023, from https://www.sei.cmu.edu/our-work/secure-development/index.cfm

*The Most Common Java Keytool Keystore Commands*. (n.d.). Www.sslshopper.com. https://www.sslshopper.com/article-most-common-java-keytool-keystore-commands.html