****

# 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** | **June 21, 2025** | **Riley Eichenour** | **Initial Creation of Document** |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the 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 Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Riley Eichenour

## Algorithm Cipher

Artemis Financial needs to employ encryption of the data they store in relation to their standard business operations. The data needs to be encrypted both at rest and in transit to reduce the impact of a data breaches/leaks, man in the middle attacks, and brute force decryption attempts. Their encryption choice must also comply with the regulations set in place for the type of data they store and work with. The recommended cipher, SHA-256, will aim to protect the archived data.

In the current state of the codebase, there are no utilizations of insecure or vulnerable code. While the data is archived, the business case may require that the data be retrievable after encryption. The SHA-256 algorithm should be utilized in encryption and checksum verification of the archived information for the typical business operations at Artemis Financial. This algorithm will meet the FIPS 140-3 standard, which meets international standard ISO/IEC 19790:2012(E). Additional this will meet industry standards set by PCI-DSS, the FTC Safeguards Rule/Gramm-Leach-Bliley Act, and SOX.

While SHA-256 may not be as strong as other encryption algorithms, such as RSA-2048, it meets the use case for the problem domain to protect the information and provides an algorithm that will take a significant amount of time for consumer grade computers to crack while keeping the operating cost of operations for Artemis Financial reasonable. With the consideration for computational time and resources, it is not appropriate to utilize a stronger encryption algorithm with the time frame the archived documents need to be stored.

The SHA-256 algorithm keeps a low probability of collisions and smaller runtime footprint to account for potentially low performance clients. While SHA-256 is more computationally expensive than other algorithms, such as MD5 or SHA-1, is less susceptible to collisions. The algorithm also has widespread use for cipher implementations. As stated in the FIPS PUB180-4, “it is computationally infeasible 1) to find a message that corresponds to a given message digest, or 2) to find two different messages that produce the same message digest” (2015).

The SHA-256 algorithm functions by taking an input message of an arbitrary length and processing it into a deterministic hash. First the input message is padded to be 448 bits leaving a 64bit block of padding, if needed, to make the message an even multiple of 512 bit blocks. Each block goes through 64 rounds of processing. Eight buffers then need to be created to be utilized through each of the rounds. Each block feeds into the input for the next block. Once all the hash values are aggregated, they compose the final hash utilized to verify the input matches the value of the hash.

For symmetric based ciphers the random numbers are utilized to generate secure keys and different initialization vectors, which adds more possibilities to the cipher. The symmetric algorithms utilize the same key for encrypting and decrypting the massage/data. In comparison, the asymmetric encryption ciphers utilize two different keys, one that is a public key and another that is the private key. The public key is utilized to encrypt the information, while the private key is used to decrypt the information. In both types of encryptions, randomness serves to create entropy in the information encryption process, which in turn reduces the predictability of the method to reverse the encryption without the appropriate tools.

Cipher algorithms start as transposition/substitution of letters, where the alphabet or ordering of the alphabet differ from the original word. Once frequency analysis was introduced, which lead to the introduction of a cipher disk. Later the well-known cipher machine utilized by the Germans in World War II, Enigma, was introduced as a difficult to decipher key. Modern cryptography relies of the utilization of computers to perform complex mathematical operations to create difficulty to decrypt algorithms and introduced the principles of asymmetric and symmetric keys.

## Certificate Generation

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

A screenshot of a computer

AI-generated content may be incorrect.

## Secure Communications

A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

## Functional Testing

A screenshot of a computer program

AI-generated content may be incorrect.

## Summary

My initial review of the application started with reviewing the architecture. I noticed the application properties were using hard coded strings for the values as part of the architecture review. The application did not accept user input, expose external API endpoints, utilize cryptography, present errors, utilize bad code quality, or data structures outside of the dependencies. After a precursory code review, I updated the Maven Dependency check tool to the latest version, and then ran a dependency check report. Each of the dependencies in the report did not show any utilization with vulnerabilities, though there were vulnerabilities present in the dependencies.

The refactoring of the code I worked on added a secure API endpoint that would relay information to the Artemis Financial customers for validation of file content. This validation provided a SHA256 checksum on request. Additionally, the hard coded strings relating to the keystore values were migrated to utilize environment variables to prevent including passwords in plain text after refactoring of the application. The API was updated to utilize SSL for connections from clients over HTTPS for secure communication while data is in transit.

## Industry Standard Best Practices

I utilized industry standard best practices by conducting multiple levels of code reviews on the application. The review started by reviewing the structure and architecture available for the application. I then progressed into the next stage of review by reviewing sources of user input, API communications, cryptography utilization, client/server relationships, error handling, quality of code, and data structure utilization. After these review steps I ran a Maven Dependency check for a report of the dependencies in the project. Based on the output of the Maven Dependency report I reviewed the code base to locate any prospective utilization of vulnerable code. Once the initial review was conducted, I added the requested secure code. Based on the changes, I again ran a dependency report and reviewed the output to confirm no additional vulnerabilities were added and that the new code was not in a vulnerable state. The code implemented created appropriate separation of concerns through data encapsulation, error handling, and appropriate documentation comments.

Appropriate application of secure coding practices is essential for the well being of Artemis Financial to reduce the impact and/or risk of security breaches, establish a trustworthy reputation among clients, proper adherence to the financial regulations/standards, and reduces development cost with higher quality and more maintainable code.