****

# 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** | **2/20/24** | **Corey Sampson** | **Algorithm Cipher determined and explained, certificate generated, and screenshot inserted.** |
| **1.1** | **2/21/24** | **Corey Sampson** | **Cipher deployed; screenshots of checksum verification/secure communications added. Screenshots added for refactored code and dependency check.** |
| **1.2** | **2/22/24** | **Corey Sampson** | **Summary added.** |

## 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

Corey Sampson

## Algorithm Cipher

Considering the services Artemis Financial offers to users worldwide, I recommend using SHA-256 as the algorithm cipher. This algorithm cipher is considered the golden standard for encryption when dealing with sensitive information and will prevent attacks such as data leaks and breaches. This algorithm also avoids collisions. A collision is when more than a single piece of data shares the same hash with another piece of data. Since the SHA-256 hash and bit functions are derived from Java’s random number generator. The use of a random number generator ensures secure encryption by disallowing a reversible checksum verification for validity of a file.

Regarding symmetric versus asymmetric keys there is one major difference to each. With symmetric keys, only one key is used to both encrypt and decrypt data whereas with an asymmetric key there is a different key for both encryption and decryption. Symmetric encryption may be faster and easier to use, but asymmetric encryption is more secure. AES-256 is an example of symmetric encryption. Even though symmetric encryption is viewed as less secure, the AES-256 algorithm is considered unbreakable by brute force attacks and still secure. The use of symmetrical or asymmetrical encryption is entirely dependent on the application and specific security requirements.

The history of encryption dates all the way back to 600 BC with the ancient Spartans using a scytale device to send secret messages during battle. Fast forward to today and encryption is used for many different security measures including prevention of unauthorized access and securing sensitive information. With the evolution of technology increasing day by day, encryption will become more advanced and more secure.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen shot of a black screen

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

Description automatically generated

## Secure Communications

Insert a screenshot below of the web browser that shows a secure webpage.

A screenshot of a computer

Description automatically generated

## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

Referring to the Vulnerability Assessment Process Flow Diagram I made a focus on three aspects of security. I addressed concerns in cryptography, client/server, and code quality. To deal with cryptography, I utilized one of the most secure algorithms for cryptographic purposes, SHA-256, to ensure that sensitive information is kept well hidden for potential attackers. This implementation can ensure that any data that Artemis Financial has stored such as estimates, client information, or corporate documentation, is secured by encryption before being stored, preventing an attacker from being able to leak sensitive data. Secondly, I addressed client/server security. This was done by creating and integrating a certificate to ensure the host is run through HTTPS as opposed to HTTP which is less secure and even vulnerable. There is also a forced TLS connection to the web application to strengthen client/server security. Lastly, all code was reviewed manually, and through static testing to ensure dependencies and code quality is up to industry best standards.

The first added layer of security is the implementation of an SSL certificate. This ensures a secure site, and lets the client know that whatever communications occur on the site is encrypted. This allows Artemis Financial to hold trust with its clients as well as their data being secured. Another layer of security that has been added is the encryption algorithm used. I used the SHA-256 algorithm. When dealing with TLS it is typically not enough to encrypt a communication by session and thus added encryption via data encryption will help strengthen potential threats from happening. If the data stored is encrypted as well as the communications through the site, an attacker would need to have access to the keys to successfully perform an attack on the data. Lastly, HTTPS was enforced to ensure the session is not performed over HTTP. This means that the session must be encrypted to help protect both client and user from older browsers having less than secure connections helping evade certain attacks such as man-in-the-middle attacks.

## Industry Standard Best Practices

To maintain the application’s current security requires keeping up to date with known vulnerabilities and implementing proper updates. By running frequent dependency checks the application can be kept up to industry standard best practices. One step I have taken already is making sure that the Maven dependency check version is up to date in the pom.xml file. Originally, the version being used was 5.3.0 whereas the current available version is 9.0.9. More simple approaches can be used to maintaining the current security as well such as keeping up to date on all plugins in the pom.xml configuration files. Doing these things can help keep the security up to date with known vulnerabilities and allow for potential implementations for added security if needed.