****

# 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** | **6/17/2023** | **Allan Lee** |  |

## 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

Allan Lee

## Algorithm Cipher

The most famous and prevalently used encryption method that I would like to recommend to bolster the security of Artemis Financial’s application is Advanced Encryption Standard (AES) encryption method. AES follows a symmetric encryption logic, meaning it utilizes one key to encrypt and decrypt. Although this might seem to be less secure, in reality, this key is used as a supplement towards the existing public and private key that each console/client already possesses. Therefore, even though AES only provides one single key per transaction, it is strong enough to prevent hacking attempts and efficient enough to retrieve necessary data. The encryption method itself undergoes multiple rounds of data transformation and then returns a unique key for each transaction.

AES encryption method by default does not involve hash functions; however, if a string is too long (requires truncation) or too short (requires fulfillment), then hash functions will be involved to generate random alphanumeric characters to ensure the final key generated is unique and has a fixed length. One example of hash function that AES uses in this case is SHA-256.

Similar to the idea of using hash functions to truncation or fulfill a message string to generate keys, random numbers and symmetric keys are considered to be major part in AES encryption method. Random numbers and operations will make the keys generated less interpretable by hackers and therefore ensure their security. Symmetric key technique ensures that the data is interpretable by those authorized and indecipherable by those who are not authorized.

The history of AES started with DES (Data Encryption Standard). When data first became prevalent, DES was good enough to cipher majority of the message. Later on as the digitalization era comes, DES was deemed inadequate because of collisions and key repetitions mainly due to its smaller key size. Therefore, AES became popular with a larger key size and enhanced security. Currently, AES remains to be the most prevalently used encryption method and has shown its efficiency in data ciphering and hack prevention.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a certificate

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer code

Description automatically generated with medium confidence

## Secure Communications

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

A screenshot of a computer

Description automatically generated with medium confidence

## Secondary Testing

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

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

## Functional Testing

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

A screenshot of a computer program

Description automatically generated with medium confidence

## Summary

Security is always the most important matter when it comes to coding design. When designing a preliminary code, multiple security measures could have been omitted or forgotten, which should later be picked up by refactoring and fixing the code. For my code, the areas of security I have addressed are cryptography and code quality. After several attempts of modification, my code is much clearer and self-explanatory than the first draft. With regards to cryptography, I have used SHA-256 to represent AES encryption with 256 bits of length.

As a future reference for my project, I could add additional security measures to address vulnerabilities like user input. For this project specifically, no user input is required and therefore this would not be a major focus. However, in the future when I get asked to establish an API that requires user interactions such as logging in or changing password, adding security features to address secured user input would be necessary and critical for the integrity of the application.

## Industry Standard Best Practices

One major part of the industry standard best practices guideline for coding states that no developer can come up with a perfect code model at the first attempt. All codes should be undergoing careful consideration, scrutinization, and inspection to gradually improve the quality of the code. However, although developers are not expected to develop a perfect code at the first try, basic features for the application should be offset or having a basic infrastructure before the application goes into production environment. One of these basic features is security. For the past seven weeks I have learned a variety of techniques to enhance the security of a specific application and to discover potential vulnerabilities of the application. Ensuring the preliminary security features are efficient for defending against hacking attempts would be very critical to secure coding to company’s overall wellbeing.