****

# 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** | **02/19/2023** | **John Costello** |  |

## 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

John Costello

## Algorithm Cipher

My recommendation for this application is to use the AES algorithm cipher, also known as the Rijndael algorithm. The algorithm manipulates a 4x4 matrix, where each element of the matrix is a byte. These blocks of data are always 128 bits, but the keys used to encrypt and decrypt them can be 128, 192, or 256 bits. The size of the key dictates how many rounds of transformation are performed on the plaintext or ciphertext. This algorithm uses symmetric keys, so the same key will be used for encryption and decryption.

## Certificate Generation

Insert a screenshot below of the CER file.

Text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Graphical user interface, text, application

Description automatically generated

## Secure Communications

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

Graphical user interface, text, application

Description automatically generated

## Secondary Testing

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

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

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

Graphical user interface, text, application

Description automatically generated

Any security vulnerabilities in the OWASP Dependency-Check Maven were not introduced from the code I wrote. These are all vulnerabilities seen in previous checks from external packages being used.

## Summary

Through this code refactoring, I have introduced a cipher algorithm to ensure the secure transmission of Artemis Financials’ documents and their clients’ documents. Through the use of symmetric keys, this algorithm ensures that only the key holders will be able to view sensitive information being passed between the two parties.

## Industry Standard Best Practices

It is important when developing software to design systems with security in mind, rather than trying to go back and shove systems into existing architecture. Doing so will minimize business losses related to security failures later in the development cycle and in production. Throughout my refactoring I ensured that my changes did not introduce any new vulnerabilities to the system, and I planned out changes with security in mind.

Attached to this report you will also find the code base which reflects the changes made while generating this report. These include the server code, the dependency report and the keystore.