****

# Practices for Secure Software Report

Jack Coster 2/25/24

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** | **2/25/24** | **Jack Coster** |  |

## 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

Jack Coster

## Algorithm Cipher

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

Artemis Financial needs additional security to support their web application to ensure a high level of confidence in their communications and handling of customer data. Assuming that the most common threat facing Artemis would be a bad actor trying to access customer financial accounts and personal data, we need to include the highest level of encryption available to the client. I would suggest using the SHA-256 algorithm cipher because it provides incredibly high levels of encryption without any known vulnerabilities at this current time. SHA-256 generates a unique output for every input and has a hash value of 256 (32 bytes) making it extremely hard to reverse engineer. This also means that using SHA-256 has an almost impossible chance at collision when use since the keys are so unique to each encryption. This algorithm is widely used across a variety of applications and is approved by NIST meaning it has gone through extension testing and certification by government agencies.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer 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.

My certificate is self signed so it is not valid but HTTPS is working.

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 screenshot of a computer

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 screenshot of a computer

Description automatically generated

## Summary

When refactoring my code I added a new secured RestController to help as the secure controller for the applications hash RESTful stop. This controller helps address common problems found in the vulnerability assessment diagram. I used the above mentioned SHA-256 cipher to secure the data. I would recommend scheduled dependency checks to see if any new vulnerabilities have appeared that would then have to be addressed. Also checking to see if any packages have new versions to update too.

## Industry Standard Best Practices

I used an industry standard cipher with SHA-256 and checked the dependency check for any new potential issues or outdated packages that needed addressing. I also used industry standard coding principles and formatting in my code to make it easy to read and review. These practices ensure that other developers can see my code and security principles and easily understand what level my application is operating at.