****

# 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** | **6/21/2024** | **Chantel Hoesman** |  |

## 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

Chantel Hoesman

## Algorithm Cipher

We have chosen SHA-256 as it is widely used and well known to be a good fit for financial communications. It also follows the guidelines for FIPS Federal Information Processing Standard. SHA-256 insures all facets of the CIA triad. It gives integrity by making sure two different inputs cannot have the same hash. This also ensures authenticity and confidentiality. SHA which stands for Secure Hash Algorithm takes input and creates a unique hash output. It has a 256 bit hash length offering a high level of security. Historically, we utilizes shorter or lesser bits in encryption such as SHA-1 but with the invention of AI and other items we had to increase to 256 bits to make sure the encryption is not able to be bypassed.

## Certificate Generation

Insert a screenshot below of the CER file.

A black screen with white text

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 computer screen shot of a 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 screenshot of a computer

Description automatically generated

## Summary

We implemented a self-signed certificate and a checksum this allowed the code to be refactored as a https secure site. Spring opened on port 8443 which is the secure port. This means we successfully refactored the code. In the checksum we implemented SHA-256 which helps with cryptography section of the vulnerability assessment process workflow.

## Industry Standard Best Practices

Industry standard best practices is to utilize FIPS which calls for the use of SHA-256 to secure data. Which is what I implemented in this case. It allowed for secure message to be used on the website. It is best to utilize time tested options when creating software. We know SHA-256 works for now and it works for our type of application.