****

# 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** | **6/18/24** | **Stephon Banks** |  |

## 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

Stephon Banks

## Algorithm Cipher

SHA-256.

SHA-256 is the industry standard for cryptographic hashing, renowned for its security and unbroken status as of June 18th, 2024. It processes data in 512-bit blocks, each padded to fit, with the final block including the message length. The algorithm uses a compression function that transforms each block with logical functions and bitwise operations, ensuring high diffusion and avalanche effects. Using 64 K-values from prime numbers, SHA-256 creates a 256-bit hash output that's computationally infeasible to reverse. Collisions are extremely unlikely, with a probability of 1 in 2^128, maintaining the hash function's integrity.

## Certificate Generation

CER file.

A computer screen with white text

Description automatically generated

## Deploy Cipher

Checksum verification.

A black text on a white background

Description automatically generated

## Secure Communications

Secure page checked with security.

A screenshot of a computer screen

Description automatically generated

## Secondary Testing

A screenshot of a computer

Description automatically generated

## Functional Testing

Refactored code executing without errors.

A screenshot of a computer

Description automatically generated

## Summary

Most of the trouble with this came due to a misunderstanding with the springboot framework. The Application.properties were a bit harder to understand. Generating the Keystore and certificate was simple as well.

## Industry Standard Best Practices

The software application's security was implemented by using a CER file and SHA-256 encryption on the server for secure communications. A dependency check was preformed to ensure all libraries were up-to-date and free from known vulnerabilities. This unfortunately was not the chase but most of the issues were fixed with updates that can be installed, I’ll leave that up to the team. These measures adhered to industry standard best practices, effectively mitigating known security threats.