****

# 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** | **10/15/23** | **Robert Lulashi** |  |

## 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

Robert Lulashi

## Algorithm Cipher

* The algorithm of SHA-256 is utilized for its function of making it impossible to track the original input hash even with the output present. This version also addresses collisions which is a concern for this code.
* Hash functions enter data in 64-bit segments in order to group inputs. These inputs end up being scrambled through use of pseudorandom number generation. Numbers are generated in a seed, and that seed is randomly generated allowing for impossible chances of being able to guess the correct seed that the functions are generated on. The seeds are also constantly changing, and each seed has different sequences.
* Symmetric keys are utilized by having one key to encrypt and decrypt data, while asymmetric keys have one public key and one private key to encrypt and decrypt data.
* Ciphers were used to keep secret information through encryption in times of war. This later evolved into keeping secure information exchanged between parties for private information and sensitive data.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

[//\*\*\* Could not get the verification to appear. Not sure how to go about the solution]

## Secure Communications

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

[//\*\*\* Could not display the secure webpage]

## Secondary Testing

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

A screenshot of a computer screen

Description automatically generated



## Functional Testing

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

The system is written with standard practices, but there are some vulnerabilities that need to be addressed. Spring Boot is currently running on an outdated version and needs to update to either 2.7.11+ or 3.0.6+. I had difficulty in trying to update to the latest version, but this was creating other errors at the current time. This can affect API’s which is second in priority just after input according to the Vulnerability Assessment Flow Chart. Cryptography is also at risk with an outdated version of snakeyaml-1.26.jar. Updating this should address the problem. Code error and code quality are also vulnerabilities that need to be addressed when reviewing the code.

A screenshot of a computer screen

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

## Summary

Overall, the system is written with standard practices, but there are some vulnerabilities that need to be addressed. There are outdated files that are open to exposure if discovered such as Spring Boot and snakeyaml-1.26.jar. Updating these should address the problem. I could not update with the skills I currently have. Code error and code quality are also vulnerabilities that need to be addressed when reviewing the code. It is important to note that code error and quality can be problematic if misspellings occurred in the pom.xml file or simply using an outdated version of the maven dependency check could lead to additional errors.

## Industry Standard Best Practices

I’ve employed a secure cipher algorithm and used best practices to ensure that the code is as secure as possible. Notifying that updates to Spring Boot are necessary for best practices is currently the best that can be done for the time being. Code is secured using one of the best algorithms around by using SHA-256.