****

# 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** | **10/14/23** | **Emilio Hernandez** |  |

## 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

Emilio Hernandez

## Algorithm Cipher

After looking through the many different algorithm ciphers, I have found that the best form for this type of program is to use AES with SHA-256. This is due to the fact that AES has a good reputation for protection against cyber-attacks. Also, with a SHA-256 it will create a more secure password authentication system that will use 256 bits. Using this algorithm, it will take an input from the user and convert it into an output called a message digest. The digest is 256 bits that is checks for verification with the hash algorithm that is stored on file to then verify the user trying to access the network.

## Certificate Generation

A screenshot of a certificate

Description automatically generatedA screenshot of a computer program

Description automatically generatedInsert a screenshot below of the CER file.

## 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

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 program

Description automatically generated

A computer screen with text on it

Description automatically generated

## Functional Testing

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

A screenshot of a computer program

Description automatically generated

## Summary

After reading through the code and going through testing I was met with minimal issues that needed to be addressed. The biggest result of the dependency check was to keep an up-to-date version of spring-boot. This can create security issues based on the old version not being kept up with all recent changes to possibly prevent old problems. The other potential risk is issue of myself being the certified user of the server which creates issues of the server not being recognized as secure.

## Industry Standard Best Practices

* User secure servers
* Keep up to date with software