****

# 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** | **[Date]** | **Joseph Valle** |  |

## 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

Joseph Valle

## Algorithm Cipher

My recommendation for an encryption algorithm cipher to deploy would be using Advanced Encryption Standard. AES is a symmetric-key encryption, which simply mean that it can encrypt and decrypt. AES operates using 128-, 192-, and 256-bits key sizes. The different bit sizes operate different from one another, essentially 128 bits runs 128 bits of data at a time. So, 192 and 256 do the same based of their size. These bit sizes of the keys are used for both the encryption and the decryption. AES is a very highly used and strong encryption. Overall, the use of encryption algorithms has only increased throughout time and has highly become advanced and rightfully so with all the new technology being introduced into the world. Not only has the encryption algorithms advanced, so have the hackers. So in order to stay ahead these encryption algorithms must always be advancing as well.

Certificate Generation

Insert a screenshot below of the CER file.

Text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Graphical user interface, text, website

Description automatically generated

## Secure Communications

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

Graphical user interface, application

Description automatically generated

## Secondary Testing

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

Graphical user interface, text, application, email

Description automatically generatedText

Description automatically generated

## Functional Testing

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

## Summary

Input validation was one way I refactored the code for it to compile and meet security testing protocols. getChecksum is a way that the users input is taken and then converted into a SHA-256 checksum using the MessageDigest class. This allows for the data that was inputted to be remain secure and tamper proof during the exchange of data or if stored. The additional layers of security added were using SSL/TLS encryption. This provided a secure connection between the client and server.. This was done by using the Spring boot application and having it deploy an HTTPS url with a trusted certificate that I created.

## Industry Standard Best Practices

The way I sued industry standard best practices was to consistently run the code after each refactoring I did. Mainly when it came to examining the dependency report. I had many vulnerabilities and needed to update certain version for the vulnerabilities to be low threat. This meant updating the POM file. In doing so I would update one at a time in order to run the report again and make sure that the vulnerabilities were fixed. Otherwise even updating the current version may not always fix these issues. By being able to follow these best practices, I was able to catch myself when updating certain areas of the POM file or my code, new vulnerabilities would appear. If I did not ruin the report consistently, in the end I may have made the application much weaker allowing for way more vulnerabilities.