****

# 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** | **6/16/2025** | **Elric Wilder** |  |

## 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

Elric Wilder

## Algorithm Cipher

SHA-256 is an ideal encryption algorithm cypher to use to implement a checksum. It is collision resistant and offers a good tradeoff between security and performance. It works by performing a one way encryption on an input value, and using the resulting hash value as a comparison to ensure the integrity of the data compared to it. 256 refers to the length in bits of the hash value generated. While there are other algorithm ciphers that are very effective, such as AES, for generating a checksum we are looking to use a hash function as we do not need to decrypt our data. AES, for example, is a symmetric encryption algorithm that is also used to decrypt the data. This is useful for encryption data that is being transmitted, as it is encrypted, sent, and then decrypted by the receiver. Our checksum does not need to be decrypted, and is instead used for data integrity and authentication, so SHA-256 is the better choice.

## Certificate Generation

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

A number on a white background

AI-generated content may be incorrect.

## Secure Communications

A screen shot of a computer

AI-generated content may be incorrect.

## Secondary Testing

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

A screen shot of a computer code

AI-generated content may be incorrect.

A close-up of a computer screen

AI-generated content may be incorrect.

## Functional Testing

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

A screen shot of a computer code

AI-generated content may be incorrect.

## Summary

In getting my project working, I encountered a bug related to the Spring framework that I solved by updating my Spring version to the latest version, 3.5. I noticed that running my dependency check after this change resulted in much fewer vulnerabilities than in recent projects. This exemplifies the need to use up to date dependencies, which is easily managed using tools like Maven. The dependency checker still found one vulnerability, but in researching the conditions in which the application is vulnerable, I can safely say that none of them apply to my project. This is an example of where suppressing false positives is necessary.

In reviewing the vulnerability assessment flowchart, the relevant areas to focus on in this project are secure API interactions, accomplished by using a REST API, cryptography, accomplished through the implementation of SHA-256 for the checksum, and code quality, by implementing secure code practices.

## Industry Standard Best Practices

Industry standard best practices for security include validating/sanitizing user inputs, enforcing secure API interactions, implementing secure encryption techniques, using secure data structures, implementing error handling, and implementing access control policies. Secure coding practices are vital to protecting not just company data, but sensitive personal and customer data, and should always be taken seriously. In this project, secure coding practices were applied where relevant.