****

# 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]** | **[Your Name]** |  |

## 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

Zinedine De Leon

## Algorithm Cipher

Encryption plays a crucial role in securing data by ensuring data integrity and confidentiality. The algorithm of choice for this application is SHA-256. It produces a fixed 256-bit output. Its common use cases include verification and digital signatures. It is a one-way function that cannot be reversed. Symmetric key encryption is when a single key is used for both encryption and decryption. An industry standardized algorithm that falls in this category is AES-256. It is also a 256-bit encryption algorithm. Encryption originally involved using ciphers, but current day practices have evolved into cryptographic algorithms. Standards have since then evolved. AES-256 is suggested for encryption, but SHA-256 for hashing and salting

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen with white text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A close-up of a logo

Description automatically generated

## Secure Communications

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

A close-up of a logo

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

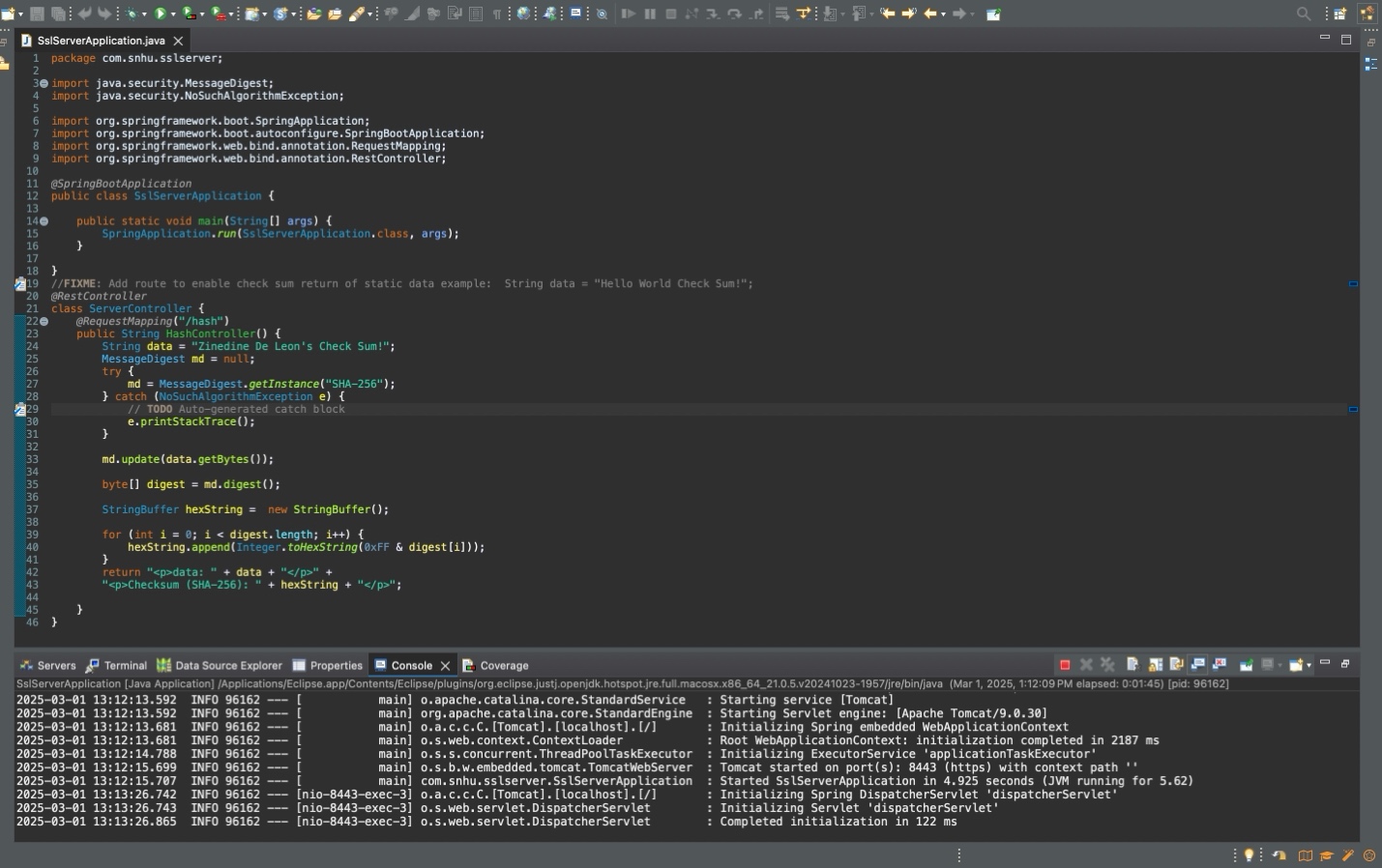
Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

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



## Summary

I used secure API interactions with the Spring Framework annotations and MessageDigest API clients. To build on this, I utilized cryptography with the SHA-256 algorithm. This ensured that the input data was hashed properly. The client/server relationship was secured using HTTPS. I handled any potential errors in the hashing algorithm implementation with the try/catch block that threw an exception if the algorithm didn’t exist.

## Industry Standard Best Practices

I used industry best practices by first having a proof of concept that worked. After that, I made sure to handle errors using exceptions and the try/catch block. This ensured that my code only executed if it ran without errors. The use of the SHA-256 algorithm also followed industry best practices, as it encoded data effectively when sending it to the client.