****

# 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

[Insert your name here.]

Duc Huy Cao

## Algorithm Cipher

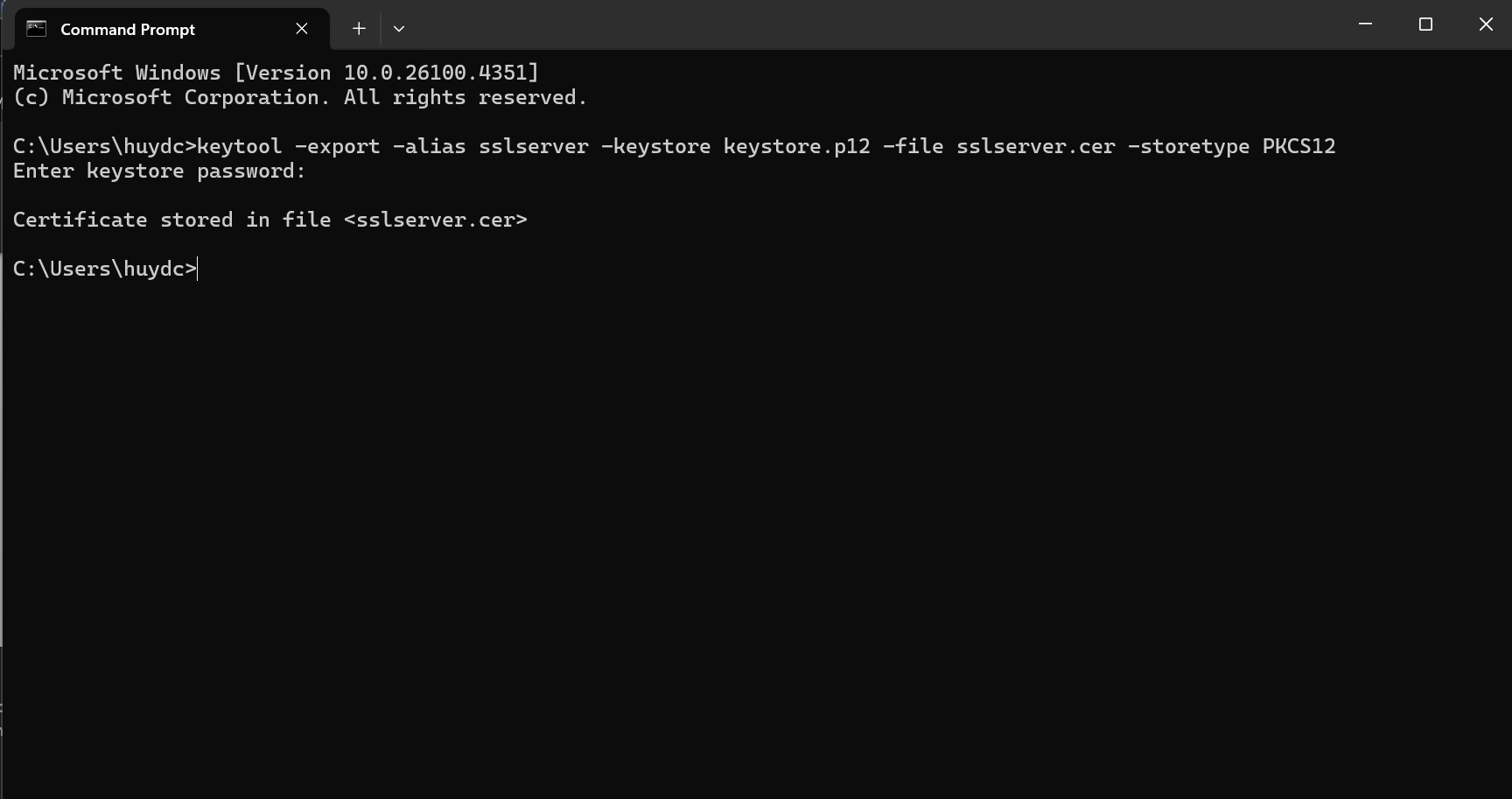
[Insert text.]

The cipher algorithm implemented in this project is SHA-256, a cryptographic hash function from the SHA-2 family. It takes an input and produces a fixed 256-bit (32-byte) hash. SHA-256 is a one-way function—meaning it cannot be reversed—and is widely used for password hashing, digital signatures, and integrity checks. Its resistance to collision and pre-image attacks makes it a modern industry standard in secure applications. In this project, SHA-256 is applied to generate a checksum for a fixed message in the /hash REST endpoint.

## Certificate Generation

Insert a screenshot below of the CER file.

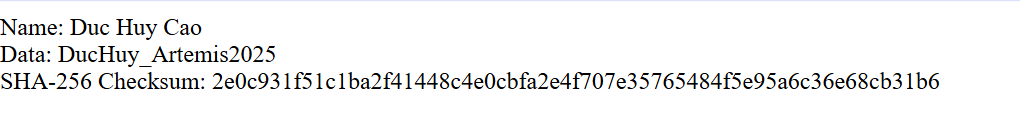
[Insert screenshots here.]



## Deploy Cipher

Insert a screenshot below of the checksum verification.

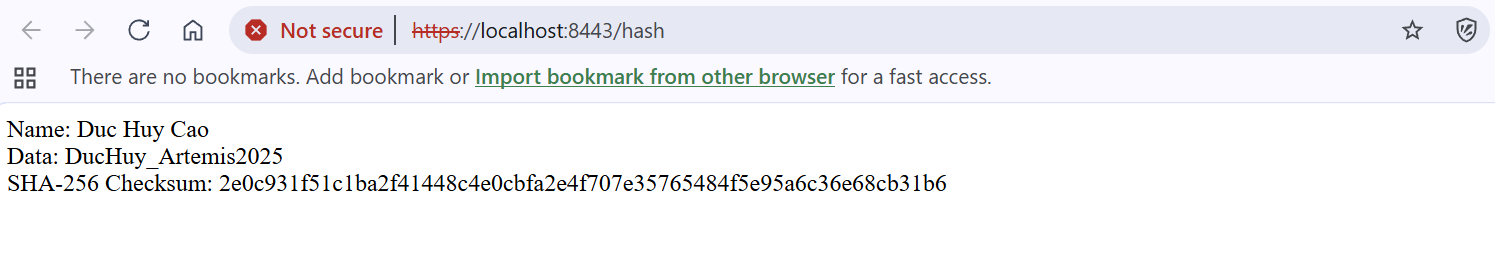
[Insert screenshots here.]



## Secure Communications

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

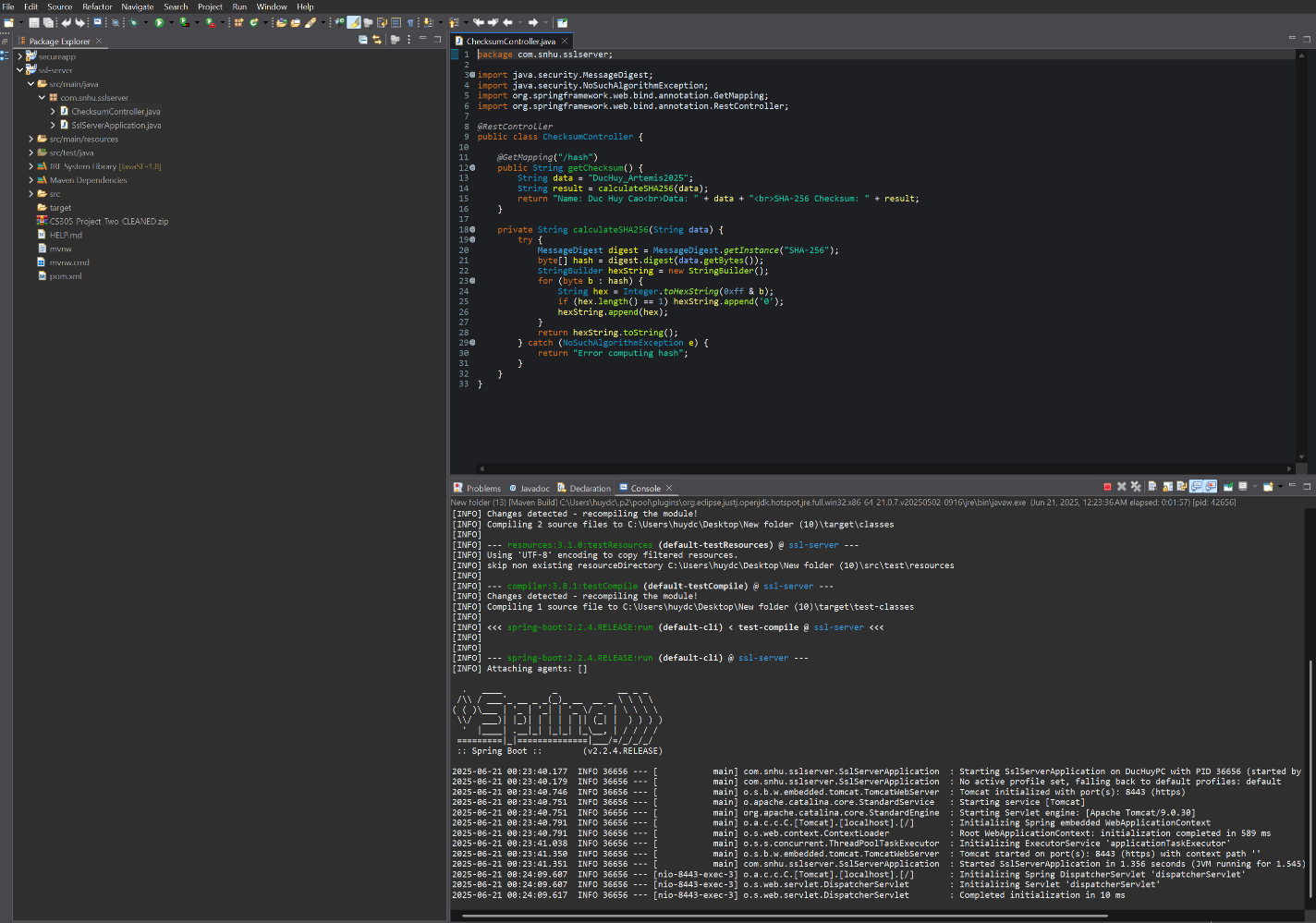
[Insert screenshots here.]



## Secondary Testing

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

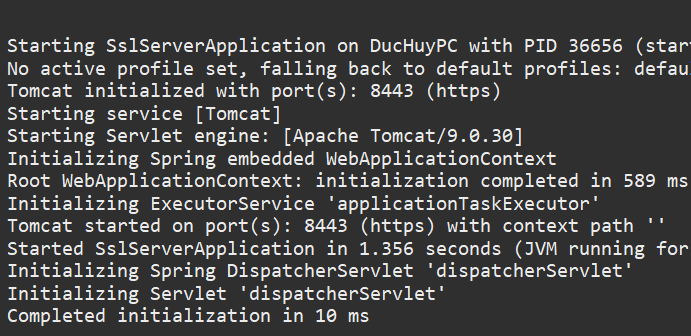
[Insert screenshots here.]



## Functional Testing

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

[Insert screenshots here.]



## Summary

[Insert text.]

This secure application demonstrates key security principles, including cryptographic hashing with SHA-256 and HTTPS-based communication using SSL/TLS. The Spring Boot REST API exposes a secure /hash endpoint that returns a SHA-256 hash, served over HTTPS. Security checks using OWASP tools confirm the use of secure dependencies, and functional tests verify that the system runs correctly on a secure channel. Overall, the project illustrates the importance of encrypting communications and validating data integrity.

## Industry Standard Best Practices

[Insert text.]

The project follows industry best practices from OWASP and NIST guidelines. These include:

* Using SHA-256 for secure, one-way hashing
* Configuring HTTPS with self-signed certificates to protect data in transit
* Running static analysis via OWASP Dependency-Check to avoid known vulnerabilities
* Following Spring Boot conventions to securely bind to secure ports  
  This ensures a secure development lifecycle aligned with real-world enterprise security requirements.