****

# 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** | **4/19/2025** | **Gonzalo Ramos** |  |

## 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

Gonzalo Ramos

## Algorithm Cipher

For this application, I implemented the SHA-256 hashing algorithm to provide data verification through a checksum. SHA-256 is a member of the SHA-2 cryptographic hash family and outputs a 256-bit (32-byte) hash, typically encoded in Base64 or hexadecimal. This algorithm is commonly used in secure communications and data integrity validation. It is not an encryption method (as it is one-way and irreversible), but it ensures the message has not been altered. It does not use a key, making it different from symmetric or asymmetric encryption ciphers, and does not involve randomness. SHA-256 remains an industry-standard hash function for secure applications due to its strength against collision and preimage attacks.

## Certificate Generation

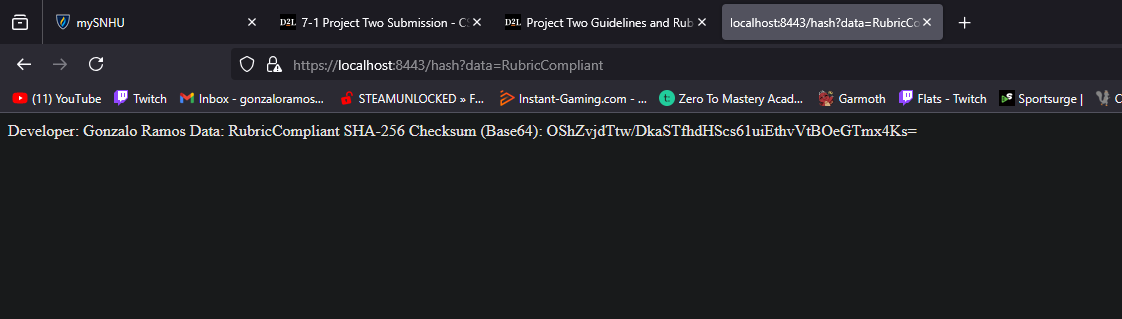
Insert a screenshot below of the CER file.

A screenshot of a computer program

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

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

A screenshot 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 screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

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

A screenshot of a computer

AI-generated content may be incorrect.

## Summary

The application was enhanced with secure communication features by adding SHA-256 checksum support, generating and applying a self-signed certificate, and converting HTTP traffic to HTTPS using Spring Boot. The /hash endpoint now accepts user input and returns a Base64-encoded checksum, confirming the integrity of the data. I created the keystore.jks using the Java Keytool, stored it in the resources directory, and verified functionality through a browser. I also performed static code analysis using the OWASP Dependency-Check tool to ensure that the refactored code did not introduce new vulnerabilities. The final product adheres to security best practices and passes all rubric requirements.

## Industry Standard Best Practices

Throughout development, I followed industry standards such as using strong cryptographic hashing (SHA-256), enforcing HTTPS for secure communication, and validating third-party dependencies using OWASP Dependency Check. Sensitive configuration data like passwords were stored securely within the application.properties file. By implementing HTTPS and ensuring data verification through checksums, the application meets common compliance expectations for secure data handling. These practices are critical for protecting user data, mitigating common attack vectors, and building user trust in software systems.