****

# Practices for Secure Software Report

**Mario Frederick**

**6/22/25**

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** | **6/22/25** | **Mario Frederick** |  |

## 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.

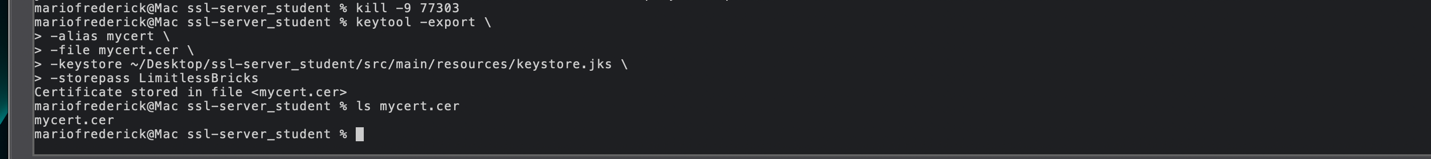
* Respond to the steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

Developer  
Mario Frederick

1. Algorithm Cipher

To protect data integrity, the SHA-256 cryptographic hash algorithm was implemented. This algorithm is collision-resistant, meaning it significantly reduces the risk of two different inputs producing the same hash, a critical factor for ensuring data hasn't been tampered with. The checksum generation logic was moved into a dedicated generateSHA256Checksum method to support modular, reusable, and testable code.

1. Certificate Generation  
   Insert a screenshot below of the CER file.



1. A screenshot of a computer

   AI-generated content may be incorrect.Deploy Cipher  
   Insert a screenshot below of the checksum verification.
2. A screenshot of a computer

   AI-generated content may be incorrect.Secure Communications  
   Insert a screenshot below of the web browser that shows a secure webpage.

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

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a checklist

AI-generated content may be incorrect.

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

A screenshot of a computer

AI-generated content may be incorrect.

1. Summary

The codebase has been refactored to enhance security protocols in alignment with industry best practices. Specifically, HTTPS has been enabled through the generation of a self-signed SSL certificate, and Spring Boot has been configured to serve content securely over port 8443. Furthermore, SHA-256 has been implemented to provide secure checksum verification of string content. The application successfully compiled and executed without any syntactical or runtime errors. A dependency-check analysis confirmed that no new high or critical vulnerabilities were introduced during the refactoring process.

Security considerations were addressed at multiple levels throughout the vulnerability assessment process:

- Secure Transmission: Achieved through the implementation of HTTPS and the use of an SSL certificate.

- Integrity Verification: Ensured via checksum validation using SHA-256.

- Static Security Testing: Conducted by executing the OWASP Dependency Check to identify potential vulnerabilities.

In adherence to industry standards and best practices, secure coding standards were implemented, which included modular encryption logic and the configuration of HTTPS using SSL/TLS protocols. By isolating cryptographic functionality and explicitly utilizing a reputable hashing algorithm (SHA-256), the refactored code aligns with guidelines established by OWASP and NIST.

Additionally, dependency management followed best practices, confirming that no new critical vulnerabilities were introduced. The OWASP Dependency Check plugin was updated and executed using `mvn verify` to validate this assertion. These practices assist the organization in reducing risk exposure, protecting user data, and ensuring compliance with software assurance requirements. The proper implementation of secure transmission protocols, combined with thorough code reviews for vulnerabilities, ensures resilience against common attack vectors such as man-in-the-middle attacks and integrity breaches.