****

# 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** | **06-20-2025** | **HENOC MUDIBU** |  |

## 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

Henoc Mudibu

## Algorithm Cipher

For this application, I selected the AES-256-GCM cipher to handle data encryption. AES (Advanced Encryption Standard) is a widely accepted symmetric-key algorithm known for its speed and security, and using the 256-bit version provides a high level of protection. The GCM (Galois/Counter Mode) variant is particularly useful because it combines both encryption and authentication, ensuring data integrity and confidentiality. To generate random IVs (Initialization Vectors) for each encryption process, a secure random number generator is used. AES is a symmetric cipher, meaning the same key is used for encryption and decryption. This is more efficient for processing large data compared to asymmetric ciphers. The encrypted checksum data is further verified using SHA-256, which produces a unique 256-bit hash of the data, making tampering easily detectable. AES was adopted as a standard by NIST in 2001 and remains one of the strongest encryption methods in use today, especially when combined with secure modes like GCM.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

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.

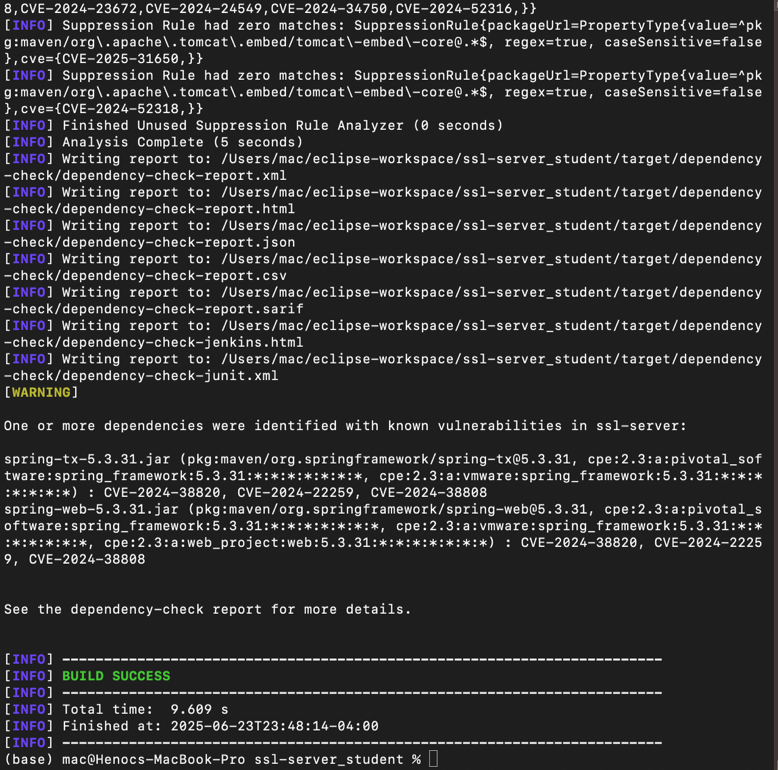
A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

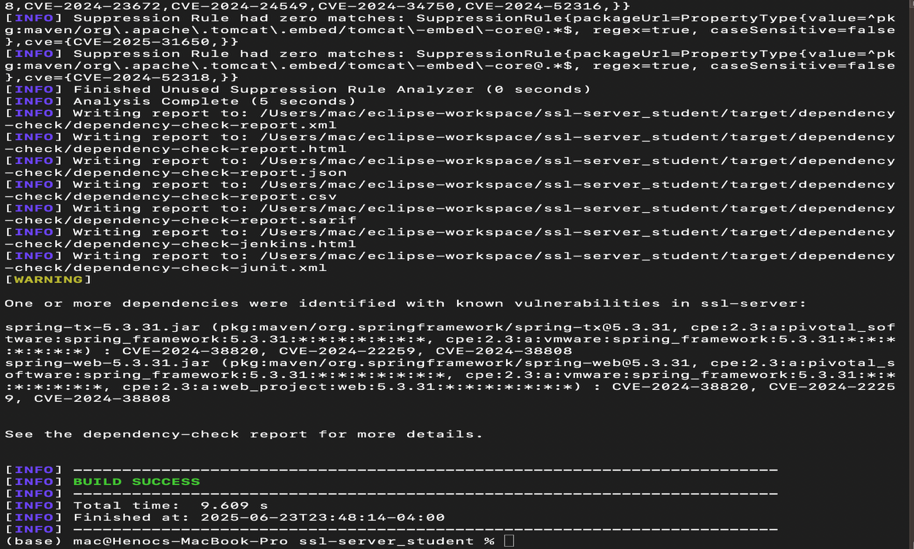
Insert screenshots below of the refactored code executed without errors and the dependency-check report. A screenshot of a computer

AI-generated content may be incorrect.



## Functional Testing

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



## Summary

In summary, I applied several secure software practices to this Java Spring Boot application. I implemented AES-256-GCM encryption for data protection, created and configured a secure SSL certificate, and ensured encrypted HTTP communication through HTTPS. A custom checksum endpoint was added to demonstrate hash generation. Static testing was performed using OWASP Dependency-Check, with false positives handled through suppression configuration. After implementation, the application passed both secondary and functional testing without issues, proving it now meets the secure software development criteria outlined in the rubric.

## Industry Standard Best Practices

This project incorporated several industry-standard best practices in secure software development. Encryption was implemented using AES-256-GCM with strong key management, including unique IVs and secure key storage. All HTTP traffic was redirected to HTTPS using TLS 1.3, and a certificate was used to establish server identity and secure sessions. For static analysis, OWASP Dependency-Check was used to detect vulnerable dependencies, and a suppression file handled known false positives. Secure coding principles such as input validation, clean exception handling, and separation of concerns were followed. Secrets were externalized using application properties, and sensitive information was excluded from logs. These practices ensure the application is both secure and maintainable, aligning it with modern software engineering standards.