****

# 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** | **4/24/25** | **Christopher Disla** | Throughout this project, we successfully set up and deployed a secure Spring Boot application configured with SSL encryption. We began by verifying and running the base project to ensure it compiled correctly without errors |

## 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

Christopher Disla

## Algorithm Cipher

For Artemis Financial’s software, I recommend using SHA-256 for file verification. SHA-256 creates a unique 256-bit hash that can quickly show if data has been tampered with, making it perfect for secure communications. It’s part of the SHA-2 family, which replaced older algorithms like SHA-1 due to security weaknesses. SHA-256 doesn’t require random numbers or keys, but if Artemis needs to encrypt sensitive information later, AES would be a strong choice because it uses symmetric keys and adds randomization with initialization vectors. Overall, SHA-256 is a modern, trusted option for keeping client data safe.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer screen

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A computer screen shot of a program

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 computer screen with many colorful lines

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

## Functional Testing

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

A screen shot of a computer

AI-generated content may be incorrect.

## Summary

Throughout this project, we successfully set up and deployed a secure Spring Boot application configured with SSL encryption. We began by verifying and running the base project to ensure it compiled correctly without errors. Then, we configured the server to use HTTPS on port 8443 with a secure keystore, allowing encrypted communication. Following that, we deployed a custom cipher service to demonstrate encryption and checksum verification. We also upgraded the dependency-check plugin to the latest version and generated a security report to ensure the project had no known vulnerabilities. Finally, we refactored the application, tested it thoroughly, and confirmed that the refactored code executed flawlessly. Each step was carefully documented and verified to meet the functional, security, and operational requirements of the assignment.

## Industry Standard Best Practices

To maintain the software application's existing security, I applied industry standard best practices such as configuring SSL/TLS encryption to secure communications and prevent data interception. I used a properly secured keystore with strong passwords and enabled HTTPS by default, ensuring that all network traffic was encrypted. Additionally, I incorporated the OWASP Dependency-Check plugin to scan for and identify known vulnerabilities in project dependencies, which helps prevent attackers from exploiting outdated libraries. Refactoring the code while preserving secure coding principles, such as validating input, managing sensitive information carefully, and minimizing exposure which further helped maintain a secure foundation.

Applying industry standard best practices for secure coding adds tremendous value to a company's overall well-being. It protects the company’s reputation by reducing the risk of data breaches, safeguards customer trust, and minimizes potential financial losses related to security incidents. Furthermore, proactive security practices lead to compliance with legal and regulatory requirements, such as GDPR or PCI-DSS, and create a culture of security awareness within the development team. This ultimately leads to stronger, more resilient software that supports the company’s growth and operational stability over the long term.