****

# 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** | **4/26/2025** | **Anteneh Denbel** |  |

## 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

Anteneh Denbel

## Algorithm Cipher

For Artemis Financial’s application, I recommend using SHA-256 for checksum hashing and AES-256 for encryption if additional data protection is required. SHA-256, a cryptographic hash function from the SHA-2 family, produces a 256-bit (32-byte) hash value, making it ideal for verifying data integrity through checksums. AES-256, a symmetric encryption algorithm with a 256-bit key, offers robust security for protecting sensitive data. SHA-256 does not require keys since it’s a hash function, but for AES-256, keys should be generated using a secure random number generator like Java’s SecureRandom; AES-256 uses the same key for encryption and decryption, unlike asymmetric algorithms such as RSA. Historically, SHA-256 was developed by the NSA in 2001 as part of the SHA-2 family, replacing the less secure SHA-1, and is widely used in applications like SSL/TLS and blockchain. AES, standardized by NIST in 2001, remains a global standard for secure encryption, commonly implemented in protocols like HTTPS. Together, SHA-256 ensures secure data verification, while AES-256 provides strong encryption for enhanced data protection.

## Certificate Generation

Insert a screenshot below of the CER file.

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 screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

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

* Syntactical: The refactored code is syntactically correct (proper Java syntax, no compilation errors).
* Logical: The SHA-256 checksum logic is correct, and the HTTPS configuration ensures secure communication.
* Security: No sensitive data is exposed; the checksum endpoint doesn’t introduce vulnerabilities like injection (since it uses static data).

A screen shot of a computer

AI-generated content may be incorrect.

## Summary

The refactoring and security enhancements for Artemis Financial’s application involved adding SHA-256 checksum functionality to verify data integrity, directly addressing the "Cryptography" and "Code Quality" steps outlined in the vulnerability assessment process flow diagram, while also enabling HTTPS to secure communications, aligning with the "Client/Server" and "Encapsulation" steps. Secure coding practices were employed, such as avoiding hardcoded secrets and implementing proper error handling in the checksum endpoint, to ensure robustness. The process followed the diagram’s flow by reviewing the application’s architecture, securing APIs, implementing cryptography, and maintaining code quality across layers, ultimately adding layers of security through the integration of HTTPS and checksum verification for a more resilient application.

## Industry Standard Best Practices

Industry standard best practices were applied by utilizing SHA-256, a widely accepted hashing standard recommended by OWASP, to ensure secure data verification, while HTTPS was enabled using a self-signed certificate to follow secure communication protocols, and dependency checks were conducted using the OWASP dependency-check tool to avoid introducing vulnerabilities. The existing security of the original Spring Boot application was maintained by preserving its core functionality while carefully adding security layers, ensuring that changes were limited to the checksum and HTTPS features to avoid introducing new vulnerabilities. These measures protect Artemis Financial’s client data and financial information, aligning with their need for secure operations, building trust with customers through compliance with modern security standards, and reducing the risk of data breaches, thereby safeguarding the company’s reputation and financial well-being.