****

# 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** | **10-15-2025** | **Ryan Blackburn** |  |

## 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

Ryan Blackburn

## Algorithm Cipher

For Artemis Financial’s web application, I recommend deploying the Advanced Encryption Standard, specifically the AES-256-Galois/Counter Mode, as the primary encryption algorithm. AES-256-GCM is a symmetric block cipher that provides both confidentiality and authentication, making it well-suited for protecting financial and personal data transmitted between clients and servers. It has a 256-bit key length which offers strong security against brute-force attacks, while Galois Counter Mode will immediately detect if any encrypted data has been altered without authorization.

The AES-256 algorithm uses the same secret key for both encryption and decryption, which increases performance efficiency when working with and securing large volumes of data. It relies on random initialization vectors generated through a cryptographically secure random number generator, which guarantees that each encryption session is unique. The AES-256 cipher operates within modern HTTPS protocols to provide end-to-end secure communication over the internet.

Although the encryption algorithm protects the data in transit, Artemis Financial’s system should also incorporate the SHA-256 hashing function to verify file integrity. This will ensure that transmitted data has not been altered and further strengthens the confidentiality provided by AES-256, and establishes a layered defense model that balances strength, efficiency, and compliance with current industry standards.

From a historical standpoint, AES replaced the older Data Encryption Standard (DES) as computing capabilities advanced and stronger security became necessary. Since its adoption by NIST in 2001, AES-256 has remained the benchmark for government and industry encryption standards worldwide. In its current state, AES-256-GCM continues to represent the modern best practice for securing sensitive information in client-server applications.

## 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 screenshot of a computer screen

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.

A screenshot of a computer screen

AI-generated content may be incorrect.

## Summary

I refactored my code to comply with software security testing protocols first by integrating a secure checksum verification endpoint and enabling HTTPS communication, including adding a secure ssl line. Doing this ensures that Artemis Financial’s application can both validate data integrity and protect client information in transit. The checksum feature uses Java’s built-in MessageDigest class with the SHA-256 algorithm, which generates a secure, verifiable hash of static data. HTTPS was implemented through a self assigned RSA-2048 certificate configured in a Java KeyStore, which forces encrypted client-server communication on port 8443.

According to the Vulnerability Assessment Process Flow diagram, my refactoring strengthened the areas of Cryptography, Secure API Interactions, and Secure Error Handling. The cryptography improvements provide secure encryption and hashing, and secure API interactions ensure data is transmitted safely through HTTPS. Secure error handling was confirmed through functional testing with no unhandled exceptions or data leaks.

Layers of security were added in a stepwise process. Checksum verification was introduced to detect file or data corruption, and HTTPS encryption was configured to protect data in transit. A static dependency check was also performed to verify that no new vulnerabilities were introduced during development. Altogether these updates align the code with secure-software lifecycle practices and meet modern testing and encryption standards.

## Industry Standard Best Practices

Industry standard best practices were applied throughout development to maintain and strengthen the application’s security. Throughout we used NIST-approved algorithms (RSA-2048 for encryption and SHA-256 for hashing), followed OWASP secure-coding principles, and enforced secure default configurations through Spring Boot’s HTTPS setup. No user input was accepted by the checksum endpoint, which prevented injection risks, and the dependency-check scan confirmed that existing libraries remained stable and secure. These actions demonstrate compliance with recognized software-security guidelines.

Applying these best practices preserves the software application’s existing security while also ensuring that any new features added later do not introduce weaknesses. Maintaining these standards adds value because they reduce the risk of data breaches, increase and maintain customer confidence, and support long-term regulatory compliance. Adhering to secure coding practices can also help the company protect sensitive financial data, creating a reputation accountability and trust in its software systems.