****

# 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** | **08/15/25** | **Juan Navarro** |  |

## 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

Juan Navarro

## Algorithm Cipher

For this project, I implemented SHA-256 as the recommended encryption algorithm because it is a secure, collision-resistant hash function that generates a 256-bit digest for data verification. Combined with HTTPS/TLS using a self-signed certificate, the application ensures both data integrity through checksum verification and secure communication in transit. SHA-256 avoids vulnerabilities found in older algorithms like MD5 and SHA-1, while TLS leverages asymmetric keys for authentication and symmetric keys for efficient encryption. Together, these updates align the application with current encryption best practices and mitigate the security risks identified in the initial assessment.

## 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

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

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

AI-generated content may be incorrect.

## Summary

The refactored code for Artemis Financial’s application was designed to comply with secure software testing protocols while addressing vulnerabilities identified in the vulnerability assessment process flow. I focused on securing data transfer and verification by implementing SHA-256 checksum validation and configuring the application to use HTTPS with a self-signed SSL certificate, ensuring both data integrity and secure communication. These updates specifically addressed areas of vulnerability related to data integrity, encrypted transmission, and certificate management. My process for adding layers of security involved first introducing input verification with hashing to protect against tampering, then enabling TLS/SSL through Spring Boot configuration, and finally validating the refactored code with dependency-check testing to confirm no new vulnerabilities were introduced. Together, these layers of protection strengthen the application, align it with industry standards for secure coding, and support Artemis Financial’s mission that “security is everyone’s responsibility.”

## Industry Standard Best Practices

I applied secure coding practices to strengthen the application without breaking anything that was already working. First, I made sure the program communicates over HTTPS, so all data sent between the server and users is protected. I also added a SHA-256 checksum feature, which makes sure that the data being sent or received has not been changed or tampered with. The new code was kept small and focused, so it doesn’t introduce unnecessary risks, and I tested it with different inputs to make sure it runs safely. I also used a security scan tool to confirm that adding this new feature did not create new problems or vulnerabilities.

Using these best practices helps maintain the security that was already in place while adding new layers of protection. This benefits the company by reducing the chances of data breaches, protecting customer trust, and showing that Artemis Financial takes security seriously. It also supports the company’s long-term well-being by lowering risks and ensuring that its software meets high standards for safety and reliability.