****

# 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** | **April 15, 2025** | **Kaylea Carpenter** |  |

## 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

Kaylea Carpenter

## Algorithm Cipher

For this project, I used the SHA-256 algorithm to generate a checksum for a static string. SHA-256 is a secure hashing algorithm from the SHA-2 family that produces a 256-bit hash value, commonly used to verify data integrity. This algorithm does not use encryption keys but is useful for detecting changes in data. It was chosen because it is widely trusted, collision-resistant, and ideal for verifying that the data transmitted through Artemis Financial’s application remains unchanged.

## Certificate Generation

Screenshot showing the successful generation and export of the self-signed SSL certificate using keytool.

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

Screenshot of the browser output displaying the original string and its SHA-256 checksum, confirming the hash was implemented successfully.

A screenshot of a computer

AI-generated content may be incorrect.

## Secure Communications

## Screenshot of the secure /hash route accessed via HTTPS, verifying that SSL is enabled and the application uses secure communication.

A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

## Screenshot of the OWASP Dependency-Check report summary, confirming that no new vulnerabilities were introduced during code refactoring.

A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

## Screenshot of the application running successfully in the Eclipse console with no runtime errors, confirming the /hash route is functioning correctly.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Summary

The code for Artemis Financial was successfully refactored to include secure communication protocols using SHA-256 checksum verification and HTTPS encryption. Functional testing confirmed that the /hash route operates correctly and returns a valid Base64-encoded checksum. The code was also scanned using OWASP Dependency-Check, and no new vulnerabilities were introduced.

Following the vulnerability assessment process, I addressed encryption, file verification, and secure communication vulnerabilities. The layers of security added include a checksum to verify data integrity, a self-signed SSL certificate to enable HTTPS, and Maven plugin testing to ensure dependency-level security. These enhancements contribute to the secure transfer of client data within the Artemis Financial web application.

## Industry Standard Best Practices

## I applied industry standard best practices for secure software development, including the use of SHA-256 for cryptographic hashing and HTTPS for encrypted communication. These protocols ensure data integrity and confidentiality during file transfer operations.

## I also used OWASP Dependency-Check to verify that no known vulnerabilities were introduced during the refactoring process. By scanning the project dependencies and updating the Maven pom.xml appropriately, I ensured the software complies with modern security testing protocols.

## Following these best practices strengthens the security posture of the software and helps build trust with clients. Maintaining secure coding practices protects against data breaches, supports regulatory compliance, and upholds the company’s mission that “Security is everyone’s responsibility.”