****

# 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** | **December 15, 2024** | **Christopher Phillips** |  |

## 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 Phillips

## Algorithm Cipher

High-Level Overview: Recommend AES (Advanced Encryption Standard) with a 256-bit key.

Overview: AES is a symmetric encryption algorithm widely used for securing sensitive data.

Hash Functions and Bit Levels: AES-256 uses 256-bit keys and provides excellent resistance against brute-force attacks.

Symmetric vs. Non-Symmetric Keys: AES is symmetric, meaning the same key encrypts and decrypts, making it fast and efficient for extensive data.

Random Numbers: Use a cryptographically secure pseudorandom number generator (CSPRNG) to generate keys for encryption.

History and Current State:

AES was established in 2001 by NIST.

Today, it is the standard for data encryption, surpassing DES and 3DES.

## Certificate Generation

A white paper with black text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Secure Communications

A screenshot of a computer

Description automatically generated

## Secondary Testing

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

Successfully upgraded Artemis Financials’ application to ensure secure communication and implement checksum functionality. This involved cleaning up the existing code to boost security and thoroughly testing the application to make sure everything works properly and meets security standards.

Implemented a checksum algorithm, SHA-256, to ensure that data remains intact during communication. To enhance security, I generated a self-signed certificate using Java Keytool, which enabled us to establish HTTPS for secure communication. Updated the application’s properties to incorporate HTTPS support with the newly created keystore.

To verify the security of our application, I utilized OWASP Dependency-Check to ensure that the refactoring process did not introduce any new vulnerabilities. Additionally, thoroughly tested the checksum endpoint to confirm its proper functionality and the establishment of secure communication via HTTPS.

With these enhancements, the Artemis Financial application now meets modern security standards, giving us confidence in the integrity and confidentiality of data transmissions.

## Industry Standard Best Practices

The revised application incorporates key industry best practices for secure software development. Implemented SHA-256, a widely recognized cryptographic hash function, to generate checksums and ensure the integrity of our data. To enhance secure communication, I established HTTPS with a self-signed SSL certificate to encrypt data during transmission, significantly reducing the risk of eavesdropping and man-in-the-middle attacks. Additionally, sensitive configurations, such as keystore paths and passwords, are carefully managed within the application properties file to maintain security.

For dependency management, I utilized OWASP Dependency-Check to scan project dependencies for known vulnerabilities, helping address potential security issues at the library level. Testing protocols involved performing functional testing to confirm that the application behaves as expected, ensuring all features work correctly. Furthermore, additional testing was conducted to verify that no new security vulnerabilities emerged during the development process. To enhance clarity and maintainability, captured the checksum functionality within a dedicated service (ChecksumService) and adhered to proper RESTful design patterns.

By incorporating these best practices, the updated application ensures that Artemis Financial’s client data remains secure and reliable. This proactive approach to software security protects our clients and bolsters our reputation and ensures compliance with data protection standards.

Citations:

* National Institute of Standards and Technology. (2015). Secure hash standard (SHS) (FIPS PUB 180-4). U.S. Department of Commerce. https://doi.org/10.6028/NIST.FIPS.180-4
* Oracle Corporation. (2024). *Java Platform, Standard Edition Tools Reference*. Oracle. https://docs.oracle.com/en/java/
* OWASP Foundation. (2024). *OWASP Dependency-Check*. https://owasp.org/www-project-dependency-check/