****

# 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** | **August 18, 2024** | |  | | --- | | **Alexander Ouellet** |  |  | | --- | |  | | **Initial Submission** |

## Client



## Developer

Alexander Ouellet

## Algorithm Cipher

For securing the communications and data transfer within Artemis Financial's application, I recommend using the **Advanced Encryption Standard (AES) with a 256-bit key**. AES is widely recognized for its high level of security and efficiency in both software and hardware implementations.

**Overview:**

* **AES** is a symmetric key algorithm, meaning the same key is used for both encryption and decryption.
* It operates on a fixed block size of 128 bits, with key sizes of 128, 192, or 256 bits. For the highest security, a 256-bit key is recommended.
* AES is known for its speed and security, making it suitable for both small and large-scale applications.

**Hash Functions and Bit Levels:**

* **SHA-256** can be used for hashing within the system. SHA-256 generates a 256-bit hash, providing a robust mechanism for ensuring data integrity.

**Use of Random Numbers and Key Types:**

* AES uses symmetric keys, which means both parties share the same secret key for encryption and decryption.
* Secure random number generators (RNGs) are used to ensure the randomness of the keys, enhancing the security of the encrypted data.

**History and Current State:**

* AES was established as a standard by the National Institute of Standards and Technology (NIST) in 2001 and has since become the global standard for encryption. Its widespread adoption and rigorous testing have proven its reliability and security over the years.

## Certificate Generation

A screenshot of a computer

Description automatically generated

## Deploy Cipher

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

## Functional Testing

The terminal output confirms that all tests were executed successfully, and no errors were encountered. This verifies that the refactored code meets the functional requirements specified for the application.

A screenshot of a computer program

Description automatically generated

## Summary

In this project, I successfully refactored the code for Artemis Financial's application to enhance its security features by implementing a cryptographic hash algorithm for data verification, generating self-signed certificates, and enforcing secure HTTPS communication. The implemented SHA-256 hash function provided a secure checksum mechanism that ensures data integrity during transmission. I generated a self-signed certificate using Java Keytool, which was then utilized to enable HTTPS communication within the application. Furthermore, I verified that the secure communication was functioning correctly by accessing the application via the HTTPS protocol.

The secondary testing involved using the OWASP Dependency-Check tool to identify any new vulnerabilities introduced during the refactoring process. The results confirmed that no new vulnerabilities were introduced, and the application adheres to industry security standards. Finally, functional testing confirmed that the application operates as expected with no errors, and the new security measures have been properly integrated.

## Industry Standard Best Practices

In completing this project, I adhered to industry standard best practices for secure coding to mitigate potential security vulnerabilities. By implementing SHA-256, a widely recognized and robust cryptographic hash function, I ensured that the application's data verification process is secure against common attacks such as collision attacks. The use of SHA-256 is in line with best practices recommended by organizations such as NIST for ensuring data integrity.

Additionally, by generating a self-signed certificate and enforcing HTTPS communication, I aligned the application with best practices for securing web communications. HTTPS is the standard protocol for secure communication on the web, ensuring that data transmitted between clients and the server is encrypted and protected from eavesdropping and man-in-the-middle attacks.

Throughout the refactoring process, I also utilized the OWASP Dependency-Check tool, which is a best practice for identifying vulnerabilities in third-party dependencies. This tool allowed me to confirm that the application’s dependencies are secure and do not introduce any known vulnerabilities.

By following these industry standard practices, I ensured that Artemis Financial's application not only meets the immediate security requirements but also aligns with the broader best practices in software security, contributing to the overall resilience and trustworthiness of the software.