****

# 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** | **10/17/2025** | **Timothy Jayson** |  |

## 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

Timothy Jayson

## Algorithm Cipher

The Algorithm Cipher I would recommend that Artimis Financial use is SHA-256.

A: SHA-256 (Also known as Secure Hash Algorithm 256-bit) is a part of the SHA-2 family that was developed by the National Security Agency (NSA). SHA-256 is published by the National Institute of Standards and Technology (NIST). This is a cryptographic hash function that transforms data into a fixed 256-bit hash value. This helps to ensure that the integrity of the data can be verified without exposing the original input. SHA-256 is widely used today due to its high collision resistance and strong security against reverse engineering attacks.

B: The SHA-256 algorithm takes input data, processes it into 512-bit blocks, and outputs a 256-bit hash. This fixed length output helps to ensure consistency and maintains a very low probability of collisions. This means that it is extremely unlikely that two different inputs will generate the same hash value under normal conditions. The SHA-256 algorithm works in the following steps:

* Input preparation: The first step is preparing the input data for processing. This involves padding the data to ensure that it fits into fixed-size chunks.
* Initial Setup: The SHA-256 algorithm starts with initial hash values. These values are constraints that are predefined by the algorithm and derived from the fractional parts of the square root of the first eight prime numbers.
* Processing the data into blocks: Next the prepared data is split into blocks that are 512 bits each that are processed in 64 rounds of operations.
* The compression function: For each of the 64 rounds, the algorithm uses a combination of bitwise operations, modular addition, and bit shifts. This mixes up the data in a complex but predictable way that ensures that even a small change in the input will drastically change the output hash.
* Producing the hash: After all of the blocks are processed, the final hash values are combined to produce the 256-bit hash.

C: SHA-256 does not use encryption keys in the same fashion that symmetric or asymmetric do. Instead it relies on mathematically derived constraints rather than random numbers.

D: The original SHA-1 (Secure Hash Algorithm 1) was introduced in 1993 by the National Security Agency (NSA) and was one of the first widely adopted cryptographic hash functions. For a while this was an adequate level of protection but as computational power increased it was discovered that collision attacks were becoming a concern. In response to the concern of collision attacks, NIST released SHA-2 in 2001 as well as SHA-224, SHA-256, SHA-384, and SHA-512. These are currently considered secure and are widely adopted today. SHA-256 has become the industry standard for secure hashing of digital certificates, TLS/SSL protocols, and blockchain technologies.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a certificate

AI-generated content may be incorrect.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.

A screenshot of a computer program

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 program

AI-generated content may be incorrect.

## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

A computer screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Based on the list of vulnerabilities the dependency-check returned, the new code did not add any new vulnerabilities to the software.

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

Below is the code I added to the software. I also edited the pom file to use 12.1.0 for the dependency check.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A black background with text

AI-generated content may be incorrect.

## Summary

Upon reviewing the vulnerability assessment process flow diagram, I enhanced the following areas of security.

* APIs: I made sure that all communication between the client and the server is secure by making HTTP endpoints into HTTPS.
* Cryptography: I implemented SHA-256 hashing for collision-resistant checksum verification. As well are configuring SSL/TLS encryption through keystore.p12
* Client/Server: Verified HTTPS configuration on port 8443 and verified proper SSL handshake as well as a valid certificate. I also ensured that HTTP did not connect.
* Code Error: I tested the code to ensure that it runs without errors as well as not introducing new vulnerabilities on the OWASP dependency check.
* Code Quality: I modified the code/pom file to use the most recent version 12.1.0 for the dependency check to ensure that all known vulnerabilities are found.

After reviewing the code packet provided for this project, I implemented the following steps to make it more secure.

* I updated the pom file to 12.1.0 for the OWASP dependency check to provide better security and vulnerability detection.
* I implemented SHA-256 hashing to ensure that any data is verifiable and resistant to tampering.
* I ensured that HTTPS was enforced for report 8443 and disabled HTTP communication.
* I performed an OWASP dependency check to look for known vulnerabilities and to ensure that new code did not add any additional vulnerabilities.

## Industry Standard Best Practices

How I used industry standard best practices: I updated the pom file to use the most recent OWASP dependency check to look for all known vulnerabilities. I also help to ensure that the software uses secure communication by enforcing transport layer security (TLS) through HTTPS on port 8443. I also made sure that port 8443 will not function with HTTP. I also implemented SHA-256 which is a collision resistant algorithm to help with data integrity.

The value of applying industry standard best practices: It is extremely important to use best practices with any software program that is being developed. This will not only help to produce a better overall product but it will also be more secure from malicious attack. Using secure coding practices help to reduce the likelihood of data breaches, service downtime, and possible issues with government compliance. These practices not only help to protect the client’s/company’s data but also help to ensure that the company’s reputation stays strong.

Sources:

Christian, A.-A. (2024, September 20). SHA-256 algorithm: What is it and how it works?. SSL2BUY. https://www.ssl2buy.com/wiki/sha-256-algorithm

GeeksforGeeks. (2025a, July 12). *Difference between symmetric and asymmetric key encryption*. GeeksforGeeks. https://www.geeksforgeeks.org/computer-networks/difference-between-symmetric-and-asymmetric-key-encryption/

Gitlan, D. (n.d.-b). *What is the SHA-256 algorithm & how it works - SSL dragon*. SSL Dragon. https://www.ssldragon.com/blog/sha-256-algorithm/

*Home*. CSR Creation | Java Keytool SSL Certificates. (n.d.). https://knowledge.digicert.com/tutorials/csr-creation-java-keytool-ssl-certificates

Mehta, M. (2020, December 4). *Hash function in cryptography: How does it work?*. InfoSec Insights. https://sectigostore.com/blog/hash-function-in-cryptography-how-does-it-work/

*Owasp dependency-check*. OWASP Dependency-Check | OWASP Foundation. (n.d.). https://owasp.org/www-project-dependency-check/

Tharaka, S. (2024, October 2). *The evolution of secure hashing: Sha Family*. Medium. https://tharakasachin98.medium.com/the-evolution-of-secure-hashing-sha-family-6d6ab8e7fefc