****

# 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** | **12/14/2024** | **Stephen Bailey** |  |

## 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

Stephen Bailey

## Algorithm Cipher

Because of its strong security characteristics, general acceptance, and fit for verifying data integrity in financial systems, SHA-256 Secure Hash Algorithm 256-bit was selected for this use. Computedly infeasible to execute brute force attacks or uncover collisions when two separate inputs yield the same hash, SHA-256 offers a high degree of protection with a 256-bit hash output as part of the SHA-2 family. Artemis Financial must have this degree of security if it is to safeguard private client information and confirm the integrity of moved files.  
  
SHA-256 is a perfect candidate for checksum validation since it can create fixed-length, unique representations of input data, therefore offering one of the main benefits. This guarantees that data stays unmodified either on storage or during transfer. Widespread acceptance and use of SHA-256 in sectors such government, healthcare, and banking highlights even further its dependability and confidence. Many cryptographic systems, like SSL/TLS certificates and blockchain technology which satisfy the greatest security requirements have this fundamental component.  
  
Furthermore resistant to cryptographic flaws including preimage attacks, second preimage attacks, and collision attacks is SHA-256, therefore guaranteeing the checksum process is safe against contemporary threats. Using the MessageDigest class, Java's solution is simple and has little overhead to provide effective speed even for big inputs. Moreover, complying with industry best practices for forward-looking software security, SHA-256 is advised by major standards groups like NIST and predicted to remain safe for the foreseeable future.  
  
Leveraging SHA-256, the program offers a safe, dependable, and quick checksum verifying system that satisfies Artemis Financial's needs for data integrity and secure communication, therefore safeguarding the private data of its customers.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer program

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A close-up of a white background

Description automatically generated

## Secure Communications

Insert a screenshot below of the web browser that shows a secure webpage.

A screenshot of a computer

Description automatically generated

## Secondary Testing

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

A screenshot of a computer error

Description automatically generated

A computer screen shot of a computer

Description automatically generated

A close-up of a text

Description automatically generated

## Functional Testing

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

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

## Summary

The refactored code and project changes have been intended to solve important security flaws and improve the general software application security. Starting with the OWASP Dependency-Check tool, the vulnerability assessment process started with spotting obsolete or susceptible dependencies. Updated to their most recent stable versions, vulnerable libraries include Hibernate Validator, Jackson Databind, SnakeYAML, and others reduced hazards including deseralization vulnerabilities and antiquated cryptographic implementations. Using supported, safe, and industry-recommended libraries guarantees compliance with safe software development techniques.  
  
Using SHA-256 in the /checksum endpoint added cryptographic hash to improve security and data integrity. A self-signed certificate was also set up to allow HTTPS, encrypting all data in route and shielding against man-in- the-middle attacks and eavesdropping. By addressing any weaknesses in data management and transmission, these developments give the program levels of security. The program was examined closely to make sure the updates followed security guidelines, so guaranteeing that no fresh vulnerabilities were brought about. This rework shows a strong attitude to safe software development in line with contemporary industry criteria.

## Industry Standard Best Practices

Carefully observed industry-standard best practices helped to improve and preserve the security of the software application while fixing recognized flaws. We identified vulnerable and outdated libraries using tools such the OWASP Dependency-Check plugin, therefore applying safe dependability management techniques. To reduce risks including injection attacks and deseralization vulnerabilities, all highlighted dependencies including Hibernate Validator, Jackson Databind, and SnakeYAML were updated to their most recent secure versions. Using the SHA-256 hash algorithm which is generally known for its resilience and collision attack resistance ensuring data integrity was the method of implementing the cryptographic capability. In line with best standards for safe data transfer, HTTPS was also enabled encrypting all connections using a self-signed SSL certificate.

The use of these guidelines ensures that the application is resistant to threats and satisfies present security standards. These developments help safeguard private client information as well as help the business maintain a good reputation by lowering the possibility of data leaks and other security issues. Based on how the application is utilized, using secure coding standards can help to decrease technical debt, simplify maintenance and expansion of software, and enable compliance with laws such GDPR and HIPAA. These kinds of behaviors are required to retain consumers trusting the organization and to maintain the business healthy generally in a market growing more security conscious.

References:

OWASP Foundation. (2024). Dependency-Check. <https://owasp.org/www-project-dependency-check/>

NIST. (2020). Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180-4. <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>