****

# 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** | **02/19/24** | **Joseph J Les** |  |

## Client



Developer

Joseph J. Les

## Algorithm Cipher

In the realm of encryption algorithms, SHA-256 stands as a robust choice, particularly when addressing security vulnerabilities. This cryptographic hash function, part of the SHA-2 family, is engineered to enhance data integrity and security. It achieves this through its design that prevents collision attacks—situations where two different inputs result in the same output hash. Such a feature is indispensable in applications requiring unique hash values for digital signatures and secure transmissions.

The utilization of random numbers in encryption enhances security by adding unpredictability to cryptographic keys, making them harder to guess or replicate. In the context of symmetric and asymmetric (non-symmetric) key cryptography, SHA-256 can be used alongside both types to ensure the integrity and authenticity of the data. Symmetric key algorithms use the same key for encryption and decryption, offering speed and efficiency in environments where secure key exchange has occurred. Asymmetric key algorithms, on the other hand, use a pair of keys (public and private) for encryption and decryption, respectively, facilitating secure data exchange even in the absence of a secure channel for key exchange.

The history of encryption algorithms reveals a continuous evolution aimed at enhancing security and efficiency. From the early symmetric ciphers, which relied on simple substitution and transposition methods, to the development of asymmetric key cryptography, which introduced the concept of public key infrastructure (PKI), this field has seen significant advancements. The introduction of hash functions like SHA-256 marks a development in ensuring data integrity and authentication without the need for encryption of the data itself.

## Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool in Eclipse.

A screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

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 screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated A screenshot of a computer error

Description automatically generated

## Functional Testing

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

A screen shot of a computer

Description automatically generated

## Summary

The refactoring of the code, aligned with the Vulnerability Assessment Process Flow Diagram, specifically targeted areas such as secure input handling, secure API interactions, encryption use, secure error handling, and secure coding practices. This process involved analyzing the application architecture for potential vulnerabilities, ensuring all inputs were validated to prevent injection attacks, and securing API interactions against unauthorized access. Cryptography practices were refined to address encryption vulnerabilities, enhancing data security during transmission. Error handling mechanisms were improved to prevent information leakage. By incorporating these layers of security, the application now adheres to industry-standard security testing protocols, significantly enhancing its resilience against common security threats.

## Industry Standard Best Practices

The SslServerApplication.java file incorporates secure coding practices by using SHA-256 hashing to ensure data integrity, a technique aligned with industry standards for mitigating known security vulnerabilities. This approach ensures that any data handled by the application maintains its originality and is protected against tampering. Applying such industry-standard best practices is important for maintaining the software application’s security. It helps in safeguarding sensitive information, thereby enhancing the trustworthiness and reliability of the system. For the company, adopting these practices not only helps in protecting against data breaches and potential security threats but also contributes to the organization's overall wellbeing. It ensures compliance with security regulations, avoids legal and financial repercussions associated with data breaches, and preserves the company's reputation by fostering trust among clients and stakeholders.

References

Hojjati, A. (n.d.). Benefits of Partnering with a Certificate Authority. DigiCert.com. <https://www.digicert.com/blog/benefits-of-partnering-with-a-certificate-authority>

IBM documentation. (n.d.). <https://www.ibm.com/docs/en/b2b-integrator/6.0.3?topic=certficates-benefits-self-signed-ca-signed-digital-certificates>

Java Security Standard Algorithm Names. (n.d.). Oracle. <https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names>

Perlner, R. A., Kelsey, J., & Cooper, D. A. (2022). Breaking Category Five SPHINCS+with SHA-256 <https://doi.org/10.1007/978-3-031-17234-2_23>

Nikolic, I., & Biryukov, A. (2008). Collisions for Step-Reduced SHA-256. University of Luxembourg. <https://www.iacr.org/archive/fse2008/50860001/50860001.pdf>