****

# 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** | **15 AUG 2023** | **Urika Pye** |  |

## 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

Urika Pye

## Algorithm Cipher

Given the scenario of a Spring Boot application, I recommend using the Advanced Encryption Standard (AES) cipher for encryption. AES is a widely recognized and accepted encryption algorithm that has been adopted by the U.S. government and is used globally. It is a symmetric key encryption algorithm, which means it uses the same key for both encryption and decryption. AES operates on blocks of data and supports key sizes of 128, 192, or 256 bits. The choice of key size depends on the level of security required and the computational resources available. Generally, a larger key size provides stronger security.

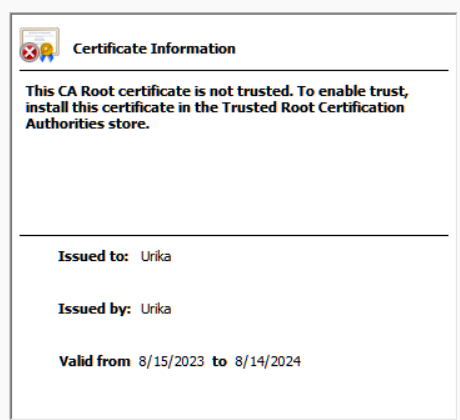
AES uses a series of transformations on the data blocks, including substitution, permutation, and mixing operations. These transformations are repeated multiple times, known as rounds, to increase the security of the encryption. The number of rounds depends on the key size, with 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

Hash functions are used in encryption to create a fixed-size output (hash) from variable-size input data. This hash can be used to verify the integrity of the data. In the context of AES, hash functions are not directly used in the encryption process, but they can be used in conjunction with AES for data integrity and authentication purposes.

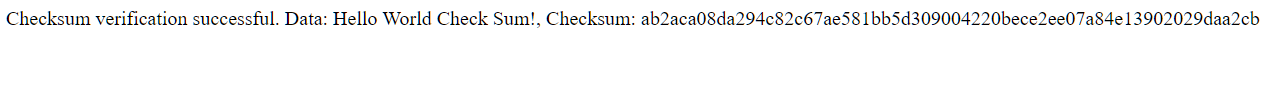
Random numbers play a crucial role in encryption, especially in the generation of encryption keys. In AES, the initial key is expanded using a key schedule that involves the use of random numbers. This ensures that the keys used in each round of encryption are different, adding to the security of the algorithm.

The history of encryption algorithms dates back to ancient times, with simple substitution ciphers used by the Greeks and Romans. Modern encryption algorithms have evolved significantly, with DES (Data Encryption Standard) being a widely used symmetric key algorithm in the late 20th century. However, DES was found to be vulnerable to brute-force attacks, leading to the development of AES as a more secure alternative. AES was established as a standard by the National Institute of Standards and Technology (NIST) in 2001 after a public competition to select a suitable algorithm. Since then, AES has become the de facto standard for encryption and is widely used in various applications, including secure communications, data storage, and financial transactions.

## Certificate Generation



## Deploy Cipher



## Secure Communications

A black and white text

Description automatically generated

## Secondary Testing

A screenshot of a computer error

Description automatically generated

A white screen with black text

Description automatically generated

## Functional Testing

A white screen with black text

Description automatically generated

## Summary

In the process of refactoring the code, several key areas of security were addressed to ensure compliance with security testing protocols and to enhance the overall security posture of the software application. The refactoring process was guided by the Vulnerability Assessment Process Flow Diagram, which provided a structured approach to identifying and mitigating potential security vulnerabilities.

One of the primary areas of security addressed during the refactoring process was the implementation of secure communication protocols. The code was updated to support HTTPS, ensuring that data transmitted between the client and server is encrypted and secure. This was achieved by configuring a keystore with a self-signed certificate and enabling SSL/TLS in the Spring Boot application. While self-signed certificates are suitable for testing and development purposes, it is recommended to use certificates issued by trusted Certificate Authorities for production environments.

Another area of security addressed was the secure handling of data. The code was refactored to include a cryptographic hash algorithm, which was used to generate checksums for data verification. This ensures the integrity of the data and helps prevent tampering. The hash algorithm used was SHA-256, which provides a high level of security and is widely accepted as a secure hashing algorithm.

The process of adding layers of security to the software application involved a multi-faceted approach. In addition to implementing secure communication protocols and secure data handling, the code was reviewed for potential vulnerabilities such as SQL injection and cross-site scripting. Static analysis tools were used to automatically scan the code for vulnerabilities, and the results were reviewed to identify and address any security issues. Additionally, the application's dependencies were checked for known vulnerabilities using the OWASP Dependency-Check plugin, and any vulnerable dependencies were updated or replaced.

In conclusion, the refactoring process involved a comprehensive approach to enhancing the security of the software application. By addressing key areas of security, such as secure communication and data handling, and by following the Vulnerability Assessment Process Flow Diagram, the code was refactored to comply with security testing protocols and to provide a more secure and robust software application.

## Industry Standard Best Practices

In the process of refactoring the code and enhancing the security of the software application, industry standard best practices for secure coding were applied to mitigate against known security vulnerabilities. These best practices are widely recognized and accepted within the software development industry and serve as a foundation for building secure and reliable software applications.

To maintain the software application's current security, several industry standard best practices were employed. First, secure communication protocols were implemented by enabling HTTPS and configuring a keystore with a certificate. This ensures that data transmitted between the client and server is encrypted and secure. Second, secure data handling practices were applied by implementing a cryptographic hash algorithm for data verification. This helps prevent data tampering and ensures the integrity of the data. Third, the code was reviewed for potential vulnerabilities such as SQL injection and cross-site scripting, and static analysis tools were used to automatically scan the code for vulnerabilities. Finally, the application's dependencies were checked for known vulnerabilities, and any vulnerable dependencies were updated or replaced.

Applying industry standard best practices for secure coding to the software application has significant value for the company's overall wellbeing. First, it helps protect the company's data and assets from unauthorized access and tampering, reducing the risk of data breaches and financial losses. Second, it enhances the company's reputation and trustworthiness, as customers and partners are more likely to do business with a company that takes security seriously. Third, it helps ensure compliance with industry regulations and standards, reducing the risk of legal and regulatory penalties. Finally, it reduces the cost and effort of addressing security issues in the future, as secure coding practices help prevent vulnerabilities from being introduced in the first place.

In conclusion, applying industry standard best practices for secure coding is essential for mitigating against known security vulnerabilities and maintaining the security of the software application. It provides significant value for the company's overall wellbeing by protecting data and assets, enhancing reputation and trustworthiness, ensuring compliance with industry regulations and standards, and reducing the cost and effort of addressing security issues in the future.