****

# 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** | **12/10/2023** | **Ederson Marcellus** |  |

## 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

Ederson Marcellus

## Algorithm Cipher

**Advanced Encryption Standard (AES):** AES is a symmetric encryption algorithm widely adopted for its security and efficiency. It operates on fixed-size blocks of data (128 bits) and supports key lengths of 128, 192, or 256 bits. AES has become the de facto standard for encrypting sensitive information.

**Hash Functions:** The code utilizes SHA-256, but it's essential to distinguish between hashing and encryption. Hash functions, like SHA-256, generate a fixed-size hash value from input data and are primarily used for data integrity verification. In contrast, encryption algorithms, like AES, are designed to protect confidentiality. AES supports key lengths of 128, 192, or 256 bits. In the provided code, a 128-bit key is derived from the SHA-256 hash of a secret key, aligning with the AES 128-bit key length.

**Random Numbers:** Random numbers play a crucial role in cryptographic operations. In this code, the key pass (**KEY\_PASS**) should be generated securely using strong random numbers. The **generateKey** method derives the encryption key from the SHA-256 hash of the secret key.

AES is a symmetric encryption algorithm, meaning the same key is used for both encryption and decryption. Symmetric encryption is efficient and well-suited for scenarios where a secure channel for key exchange can be established. The code employs a symmetric key approach, enhancing performance without compromising security.

AES was established by the U.S. National Institute of Standards and Technology (NIST) in 2001 to replace the aging Data Encryption Standard (DES). AES has since become a global standard for securing sensitive information, offering a high level of security and performance.

AES remains widely adopted and is considered secure against known cryptographic attacks when used with appropriate key lengths. It is supported by various programming languages and cryptographic libraries, making it a reliable choice for securing data in transit or at rest.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated

## Deploy Cipher

A screenshot of a computer code

Description automatically generated

[Insert screenshots here.]

## Secure Communications

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

A computer code with black text

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 program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

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

A screenshot of a computer program

Description automatically generated

## Summary

The provided code implements encryption using the Advanced Encryption Standard (AES) algorithm with a key length of 128 bits. It encrypts a given string using a secret key derived from the SHA-256 hash of a predefined secret key. The resulting encrypted data is then Base64-encoded for ease of representation. The code exemplifies a symmetric encryption approach, leveraging AES for its efficiency and security. The use of SHA-256 ensures the derivation of a secure key.

## Industry Standard Best Practices

1. Key Management:

Best Practice: Secret keys must be generated securely using strong random numbers and stored in a secure manner.

Implementation: The code uses a predefined secret key. In practice, keys should be generated securely and managed following industry standards, considering key rotation and protection mechanisms.

2. Algorithm Selection:

Best Practice: Choose well-established and widely adopted encryption algorithms.

Implementation: The code utilizes AES, a NIST-approved symmetric encryption standard widely accepted for its security and efficiency.

3. Key Length Consideration:

Best Practice: Choose appropriate key lengths to balance security and performance.

Implementation: The code uses a key length of 128 bits, providing a reasonable level of security. Adjust key lengths based on specific security requirements.

4. Random Number Generation:

Best Practice: Ensure secure random number generation for key creation.

Implementation: The code uses SHA-256 for key derivation, which relies on the randomness of the input secret key.

5. Error Handling:

Best Practice: Implement robust error handling to manage exceptions.

Implementation: The code catches NoSuchAlgorithmException and prints the stack trace. In a production environment, handle exceptions more gracefully, considering logging and appropriate error responses.

6. Secure Communication:

Best Practice: Implement secure communication protocols (e.g., HTTPS).

Implementation: The code does not explicitly address network security. In a complete system, secure communication practices such as HTTPS should be implemented to protect data in transit.

7. Documentation:

Best Practice: Maintain comprehensive documentation for code understanding and future development.

Implementation: The code lacks comments and documentation. In practice, thorough comments, README files, and inline documentation should be provided.