****

# 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** | **[6/21/2024]** | **[Lauren-Ann Javier]** |  |

## 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

Lauren-Ann Javier

## Algorithm Cipher

SHA-256 itself does not involve random numbers directly in its hashing process. However, secure applications utilizing SHA-256 may incorporate random numbers for generating keys in symmetric or asymmetric encryption methods. Symmetric keys (e.g., AES-256) are simpler and use the same key for both encryption and decryption, while asymmetric keys (e.g., RSA) use a pair of keys (public and private) for encryption and decryption respectively, offering better security for internet communications.

Encryption techniques have played an important role throughout history, evolving from basic methods to sophisticated algorithms used in modern cryptography. One notable example is the Enigma machine, employed by the German military during World War II. The Enigma machine used complex rotor mechanisms to encrypt messages, which were considered unbreakable at the time. However, British mathematician Alan Turing and his team at Bletchley Park successfully cracked the Enigma code, a breakthrough that significantly influenced the outcome of the war.

## Certificate Generation

A screenshot of a computer

Description automatically generated

## Deploy Cipher

A close-up of a number

Description automatically generated

## Secure Communications

A screenshot of a computer

Description automatically generated

## Secondary Testing

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screen shot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

A screenshot of a computer

Description automatically generated

## Summary

I refactored the code by incorporating a secure RestController into the SSLServerApplication.java file. This ServerController class serves as the secure controller for the hash RESTful endpoint, addressing the secure coding concerns highlighted in the Vulnerability Assessment Diagram and meeting all specified requirements. For hashing, I chose to implement SHA-256, a strong hashing algorithm known for its strong security properties. The code has been streamlined to minimize the potential attack surface, emphasizing simplicity and security.

## Industry Standard Best Practices

In refactoring the code to improve security, I combined industry-standard best practices expected to modify known vulnerabilities and maintain the software application's current security posture. Key measures included accurate input validation to stop SQL injection, cross-site scripting (XSS), and command injection attacks. Additionally, I prioritized secure authentication and password management by enforcing strong password policies, employing strong hashing algorithms like SHA-256 for password storage, and considering multi-factor authentication (MFA) to boost user access control. Following the principle of least privilege, I restricted user and application permissions to essential levels, thereby reducing the risk of unauthorized access and actions.

Secure data storage and transmission were guaranteed through encryption of sensitive data at rest and the use of secure communication protocols such as HTTPS, protecting information both in transit and within the application. Regular updates and patching of the software and its dependencies were maintained to promptly address security vulnerabilities and minimize the potential for exploitation. Furthermore, strong error-handling mechanisms were implemented to prevent the exposure of sensitive information during runtime.

By applying these practices, the software not only protects sensitive data but also ensures agreement with regulatory requirements, reduces costs associated with security breaches and enhances trust among customers and partners. This active approach contributes significantly to the company's overall well-being by safeguarding its reputation, maintaining operational continuity, and encouraging a positive brand image in the competitive market landscape.