****

# 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** | **4/17/2024** | **Chloe Britton** |  |

## 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

Chloe Britton

## Algorithm Cipher

## In selecting an appropriate encryption algorithm cipher, it's essential to assess the security vulnerabilities present in the deployment scenario. For instance, if the primary concern is securing data transmission over a network, the chosen cipher should provide robust protection against eavesdropping and tampering. In such cases, the Advanced Encryption Standard (AES) cipher is often recommended. AES is a symmetric encryption algorithm known for its efficiency and security. It operates on fixed-length blocks of data and supports key sizes of 128, 192, or 256 bits. AES has undergone extensive scrutiny by cryptographic experts and is widely adopted in various applications, including secure communication protocols like TLS/SSL.

## Additionally, it's crucial to consider factors such as the strength of hash functions and key management practices. Hash functions play a vital role in integrity verification and password hashing. They should exhibit collision resistance and pre-image resistance to ensure data integrity and prevent password attacks. As for key management, the choice between symmetric and asymmetric encryption depends on factors like performance requirements and the need for secure key exchange. Symmetric encryption, where the same key is used for both encryption and decryption, offers high performance and is suitable for securing data at rest. On the other hand, asymmetric encryption, which involves separate keys for encryption and decryption, is often used for key exchange and digital signatures.

## In assessing the history and current state of encryption algorithms, it's evident that the field has evolved significantly over time. Earlier algorithms like Data Encryption Standard (DES) have been replaced by more robust and efficient alternatives like AES. The cryptographic community continues to innovate, developing algorithms like elliptic curve cryptography (ECC) to address emerging security challenges and accommodate the growing complexity of modern computing environments. Overall, the selection of an encryption algorithm cipher should be guided by a comprehensive understanding of the security requirements and the strengths and weaknesses of available algorithms.

## Certificate Generation

A screenshot of a computer

Description automatically generated

## Deploy Cipher

A black text on 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

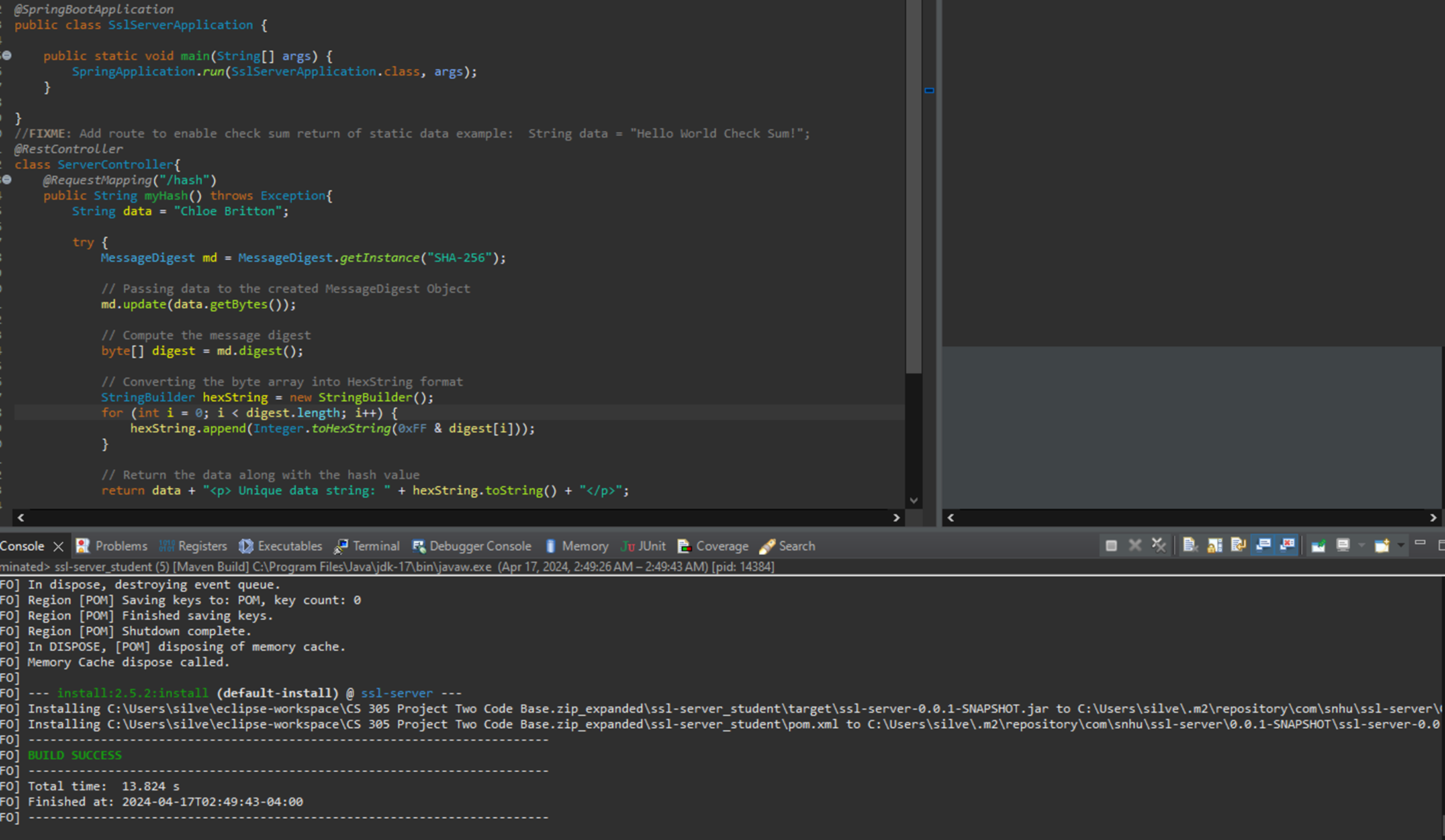
A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing



## Summary

## We have a Spring Boot controller responsible for computing SHA-256 hashes of input data strings. Through refactoring, several security considerations have been addressed. The code employs the secure SHA-256 hashing algorithm to generate hash values, ensuring the integrity and confidentiality of sensitive data. Error handling mechanisms have been implemented to gracefully manage scenarios where the hashing algorithm is unavailable, enhancing the application's resilience. Furthermore, the inclusion of error logging facilitates the detection and diagnosis of potential security incidents, supporting proactive security maintenance. While the code demonstrates fundamental security practices, additional layers of security, such as input validation, output encoding, access control, and data encryption, can further fortify the application against a broader spectrum of security threats. These measures collectively contribute to bolstering the application's security posture and aligning with comprehensive security testing protocols.

## Industry Standard Best Practices

In maintaining the security of the software application, industry-standard best practices for secure coding were diligently applied to mitigate known security vulnerabilities. This involved a multifaceted approach aimed at bolstering the application's resilience against potential threats. Adherence to secure coding standards played a pivotal role, with the codebase structured to minimize common vulnerabilities like injection attacks, cross-site scripting (XSS), and insecure direct object references. By following established guidelines outlined by organizations such as OWASP (Open Web Application Security Project) or CERT (Computer Emergency Response Team), developers ensured that the software's foundation was built upon secure coding principles.

Implementation of strong authentication and authorization mechanisms further fortified the application's defenses. Robust authentication protocols, including multi-factor authentication (MFA) and sophisticated session management techniques, were integrated to ensure that only authorized users could access sensitive functionalities and data. Authorization controls were meticulously enforced, restricting privileges based on user roles and permissions, thereby mitigating risks associated with unauthorized access and privilege escalation.

Secure data handling was prioritized throughout the development process. Industry-standard encryption algorithms like AES (Advanced Encryption Standard) were employed for data encryption, while protocols such as TLS (Transport Layer Security) were utilized to secure communication channels. Input validation and output encoding techniques were rigorously applied to prevent injection attacks and safeguard against data tampering and leakage, bolstering the application's overall resilience against data breaches and malicious exploitation.

Continuous security testing and monitoring were paramount to maintaining the application's security posture. Regular security assessments, including penetration testing, code reviews, and vulnerability scanning, were conducted to identify and remediate potential security weaknesses proactively. Robust logging and monitoring mechanisms were implemented to detect and respond to security incidents in real-time, ensuring that any anomalies or suspicious activities were promptly addressed, thereby minimizing the impact of security breaches, and enhancing the application's overall security posture.

The value of applying industry-standard best practices for secure coding to the company's overall well-being cannot be overstated. By prioritizing security throughout the software development lifecycle, organizations can mitigate the risk of costly data breaches, regulatory non-compliance, reputational damage, and financial losses. Moreover, adherence to industry standards fosters customer trust and confidence, enhancing the company's reputation and competitiveness in the marketplace. Ultimately, investing in robust security measures not only safeguards the integrity and confidentiality of critical assets but also reinforces the organization's commitment to prioritizing security as a fundamental aspect of its operations.

1. References

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

How to create a self signed certificate using Java Keytool. SSL Certificate Comparison and Reviews. (n.d.). https://www.sslshopper.com/article-how-to-create-a-self-signed-certificate-using-java-keytool.html

Keytool - key and certificate management tool. keytool-Key and Certificate Management Tool. (n.d.). <https://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html>