****

# 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** | **06/22/2024** | **Andrea C. Sherry** |  |

## Client



## Instructions

Submit these 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

Andrea Carmen Sherry

## Algorithm Cipher

My recommended encryption algorithm cipher is the Advanced Encryption Standard or AES. This encryption algorithm cipher is a highly regarded encryption algorithm established by the U.S. National Institute of Standards and Technology in 2001. It is a symmetric key encryption standard used around the world to secure data. Advanced Encryption Standard operates on fixed block sizes of 128 bits and supports key sizes of 128, 192, and 256 bits. It transforms plaintext into ciphertext through a series of substitution, permutation, and mixing operations, providing robust security for data. Although AES is mainly an encryption algorithm, it also works alongside hash functions like SHA-256 and SHA-3 to ensure data integrity and authenticate data origins. AES supports bit levels of 128, 192, and 256, corresponding to 10, 12, and 14 rounds of processing, respectively. Higher bit levels offer more security but also needs more computational power. Random numbers are essential in cryptographic operations for creating secure keys, initialization vectors, and nonces. This method of encryption relies on high-quality random numbers to ensure the unpredictability and security of keys and IVs. Inadequate random number generation can weaken encryption and make it vulnerable to attacks. Advanced Encryption Standard uses symmetric key encryption, meaning the same key is used for encryption and decryption. This method is faster and more efficient, making it appropriate for encrypting large data volumes. Yet, it requires secure key distribution. In contrast, asymmetric encryption uses a pair of for encryption and decryption, perfect for secure communications without pre-shared keys. Historically, encryption started with classical ciphers like the Caesar cipher. The Data Encryption Standard appeared in the 1970s but was later thought to be unsafe due to its short key length. Triple DES or 3DES was an improvement but had performance limitations. AES, selected as the standard in 2001, replaced DES and 3DES and has become the foundation of modern encryption. Present improvements include Elliptic Curve Cryptography and post-quantum cryptography to counter quantum computing threats. Finally, AES is recommended for its recognized security, efficiency, and versatility. It is fit for encrypting large data volumes with minimal performance impact and is supported by many platforms. AES works well with other cryptographic protocols, ensuring interoperability, and offers scalable security through different significant sizes. Its general acceptance and strong performance make it the ideal choice for securing sensitive data.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer program

Description automatically generated

A screenshot of a certificate

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer program

Description automatically generated

## Secure Communications

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

A screen shot of a computer

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

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

Description automatically generated  
  
A screenshot of a computer

Description automatically generated

## Summary

During the functional testing and refactoring of the “SslServerApplication.java” code, I addressed different important areas of security vulnerabilities as outlined in the vulnerability assessment process flow diagram. I ensured proper input validation, secure error handling, and the use of strong cryptographic practices. Specifically, I applied SHA-256 for secure hashing, enforced HTTPS to secure data in transit, and added error handling to manage exceptions without exposing sensitive information.

The refactored code adheres to secure coding standards and practices. I updated the “application.properties” file to enable HTTPS, ensuring encrypted communication. The functional testing verified that the application executes without errors, demonstrating compliance with secure software practices and effectively mitigating identified vulnerabilities.

## Industry Standard Best Practices

I applied industry standard best practices for secure coding to mitigate known security vulnerabilities in the “SslServerApplication.java” code. By implementing SHA-256 for hashing, I ensured the use of a strong cryptographic algorithm to keep data integrity. I enforced HTTPS in the “application.properties” file to secure data in transit, ensuring encrypted communication between the server and clients. Strong error handling was added to manage exceptions securely, preventing the exposure of sensitive information. These practices align with industry standards, like those recommended by OWASP, to ensure the application is secure against common threats.

Applying industry standard best practices for secure coding is important for the company's overall well-being as it protects the software from potential security breaches that could lead to data loss and financial and reputational damage. These practices help create a secure and reliable software foundation, fostering customer trust and compliance with regulatory requirements. By actively addressing security vulnerabilities, the company can lower risks and increase the overall quality and security of its software products.