****

# 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** | **10/16/2024** | **Jose Lara Hernandez** |  |

## 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

Jose Lara Hernandez

## Algorithm Cipher

For the encryption algorithm cipher, I recommend using the Advanced Encryption Standard (AES). AES is widely regarded as the standard for secure encryption due to its speed and high level of security. It supports key sizes of 128, 192, and 256 bits, with AES-256 being the most secure. The algorithm employs symmetric encryption, which means the same key is used for both encryption and decryption, making it efficient and secure for most use cases. AES also utilizes secure hash functions like SHA-256 to ensure data integrity and employs random numbers to generate cryptographic keys, enhancing the overall encryption strength. AES’s history dates back to its selection by the U.S. National Institute of Standards and Technology (NIST) in 2001 as the replacement for the older Data Encryption Standard (DES). Today, it remains the global encryption standard, trusted for both government and commercial applications due to its proven effectiveness in preventing unauthorized access to sensitive data.

This recommendation satisfies the need for high-level security and meets modern encryption standards by providing protection against attacks, ensuring data integrity, and supporting strong key management.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

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

Insert screenshots below of the refactored code executed without errors and the dependency-check report.



A screenshot of a computer

Description automatically generated

## Functional Testing

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

A computer screen shot of green text

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

In this project, I refactored the code to address identified security vulnerabilities, focusing on secure communication, encryption, and integrity verification. By implementing a checksum mechanism and using HTTPS protocol, I ensured that secure communication was enforced across the web application. Additionally, I generated self-signed certificates and integrated them into the application to secure the transport layer. Referring to the vulnerability assessment process flow diagram, the primary areas of security addressed included ensuring data integrity through checksum verification, encrypting data using SSL/TLS, and preventing unauthorized access through HTTPS. I employed the OWASP dependency-check tool to identify security vulnerabilities in third-party libraries, and although several were mitigated, some critical vulnerabilities remain, especially in external dependencies. These will require further attention and remediation. Throughout this process, I layered security enhancements by introducing secure communication protocols (HTTPS and self-signed certificates) and integrating cryptographic checksum algorithms for data validation, thus strengthening the overall security posture of the application.

## Industry Standard Best Practices

To maintain the software application's security, I applied industry-standard best practices such as utilizing HTTPS for secure communication, employing SHA-256 for checksum generation to ensure data integrity, and using the OWASP dependency-check tool to identify vulnerabilities in third-party libraries. Additionally, SSL certificates were managed securely to protect encryption keys. These practices help mitigate known security vulnerabilities and ensure the application is resilient against attacks. Applying these best practices is crucial for safeguarding the company's data, maintaining client trust, reducing the risk of security breaches, and ensuring the long-term security and scalability of the application.