****

# 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** | **8/24/2024** | **John Munguia** |  |

## 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

John Munguia

## Algorithm Cipher

In this project, I implemented a hashing utility using Java's built-in MessageDigest class to generate SHA-256 hashes. This approach ensures that the application remains lightweight and avoids external cryptographic libraries, minimizing potential vulnerabilities.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a certificate

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.



## 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 computer screen shot of a 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 screen shot of a computer program

Description automatically generated

## Summary

The project successfully demonstrated the implementation of secure communication using SSL certificates and SHA-256 hashing for data integrity. The transition from Bouncy Castle to Java's built-in cryptography reduced potential vulnerabilities, ensuring a more secure application environment. All testing phases, including unit testing, secondary testing, and functional testing, were completed successfully with no errors.

## Industry Standard Best Practices

Throughout the project, I followed industry-standard best practices:

* Using Java's built-in cryptography for secure hashing.
* Generating a self-signed SSL certificate to enable HTTPS for secure communication.
* Regularly updating dependencies to their latest versions to minimize security vulnerabilities.
* Conducting comprehensive testing to ensure code quality and security compliance.