****

# 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** | **[12/12/24]** | **[Branden Boehnke]** | **Project Two Submission** |

## 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

Branden Boehnke

## Algorithm Cipher

SHA-256 is a cryptographic hash function that is widely used and designed for security and resistance to attacks like collisions. Regardless of the input size, SHA-256 provides a fixed-size output of 256 bits. SHA-256 ensures data integrity and provides checksum for verifying data alterations. With the output of 256 bits, the hash function guarantees that no two inputs could provoke the same output which would cause a collision. The bit level consists of 512-bit message blocks producing a 256-bit output. The function is used in combination with random values for hashing sensitive data in efforts to avoid collisions and other attacks. SHA-256 is not reversible like symmetric or asymmetric ciphers. It also does not include encryption keys but can be used in systems that have both symmetric or asymmetric algorithms. SHA-256 is a part of the SHA-2 family which was developed by the NSA. SHA-2 was the follow up of SHA-1 which became vulnerable to collisions. SHA-2’s larget bit output has held up over time from attacks. To this day, SHA-256 is still a widely used cryptographic hash function and is used in combination with SSL for secure transmissions.

## Certificate Generation

A screen shot of a computer program

Description automatically generated

A screenshot of a certificate

Description automatically generated

## Deploy Cipher

A screenshot of a chat

Description automatically generated

## Secure Communications

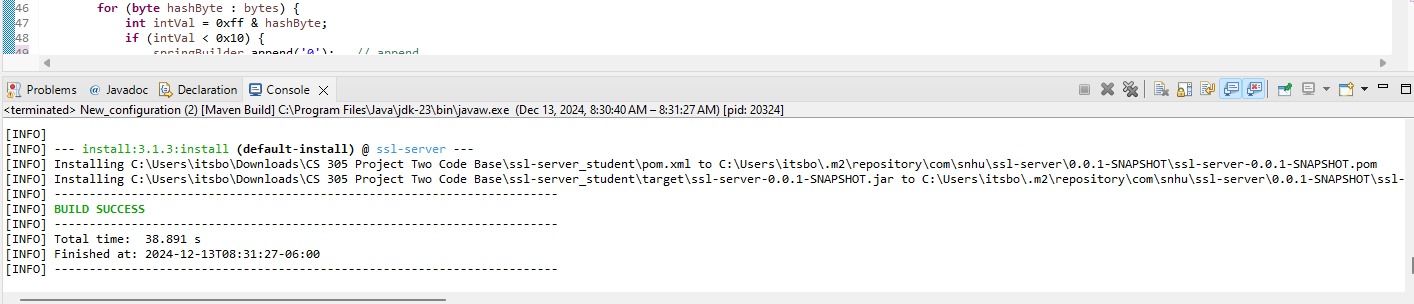
Insert a screenshot below of the web browser that shows a secure webpage.

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

A screenshot of a computer

Description automatically generated

## Summary

In this project, the code was refactored using SHA-256 for secure cryptography. It uses an SSL keystore configuration to ensure encrypted communication between the client and the server. Using SHA-256 ensures the checksum verifies the integrity of data. Within the application.properties file, keystore location, type and password have been implemented to enforce proper security.

With a self-signed SSL certificate, all transmissions from the client and server are encrypted. Hashing with SHA-256 confirms security against collisions. The OWASP Dependency Check plugin was implemented into the pom.xml file for vulnerability reports in the dependencies used within the project.

## Industry Standard Best Practices

With the industry’s best practices in mind, the implementation of a strong cryptography structure was crucial for the prevention of attacks. Ensuring that dependencies were up to date and did not present any vulnerabilities was also necessary. To reduce risks and ensure the integrity of the company stays intact, secure practices were taken to enforce reliability as well as minimize the likelihood of successful attacks. These secure coding practices also promote actions of scalability for future developments as well as compliance with security regulations.