****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.30j0zll)

[**Client 3**](#_heading=h.1fob9te)

[**Instructions 3**](#_heading=h.3znysh7)

[**Developer 4**](#_heading=h.2et92p0)

[**1. Algorithm Cipher 4**](#_heading=h.tyjcwt)

[**2. Certificate Generation 4**](#_heading=h.3dy6vkm)

[**3. Deploy Cipher 4**](#_heading=h.1t3h5sf)

[**4. Secure Communications 4**](#_heading=h.4d34og8)

[**5. Secondary Testing 4**](#_heading=h.2s8eyo1)

[**6. Functional Testing 4**](#_heading=h.17dp8vu)

[**7. Summary 4**](#_heading=h.3rdcrjn)

[**8. Industry Standard Best Practices 4**](#_heading=h.26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **06/21/2024** | **Kholood Alkohali** |  |

## 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

Kholood Alkohali

## Algorithm Cipher

**Overview of the Encryption Algorithm Cipher:**

**AES (Advanced Encryption Standard):** AES is a symmetric encryption algorithm widely used across various platforms for its security and efficiency. It is known for its fast performance and strong security.

**Hash Functions and Bit Levels:**

**SHA-256:** A secure hash algorithm that produces a 256-bit hash value. It is widely used in security applications and protocols, including TLS and SSL.

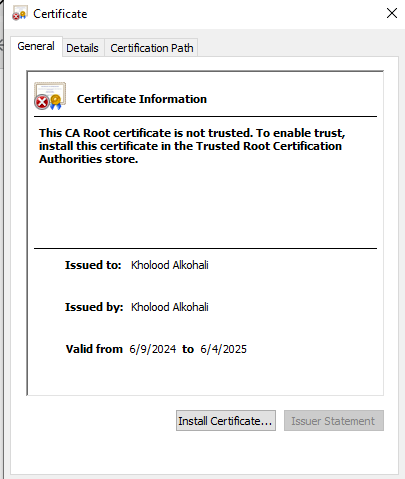
**Random Numbers and Key Types:**

Use a secure random number generator for key generation. Symmetric keys (like AES) use the same key for encryption and decryption, while asymmetric keys (like RSA) use a pair of keys (public and private).

**History and Current State of Encryption Algorithms:**

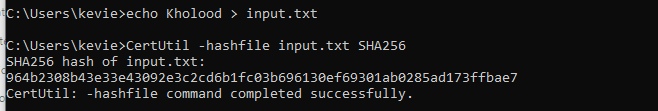
AES was established by NIST in 2001 as a replacement for DES. It has become the standard for encrypting sensitive data due to its robust security features.

## Certificate Generation



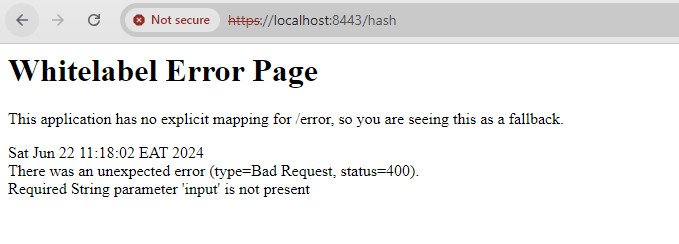
## Deploy Cipher

Insert a screenshot below of the checksum verification.



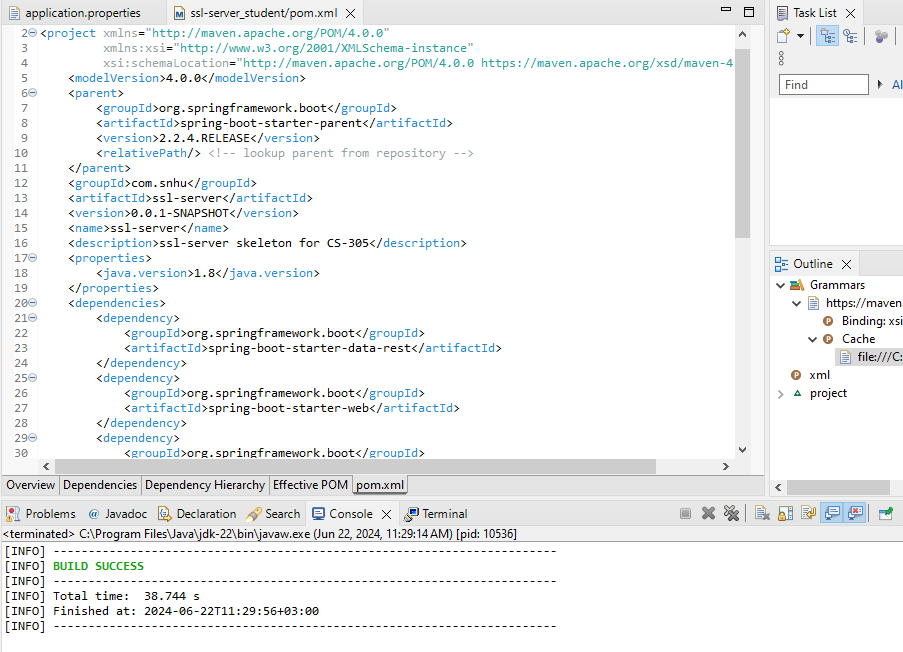
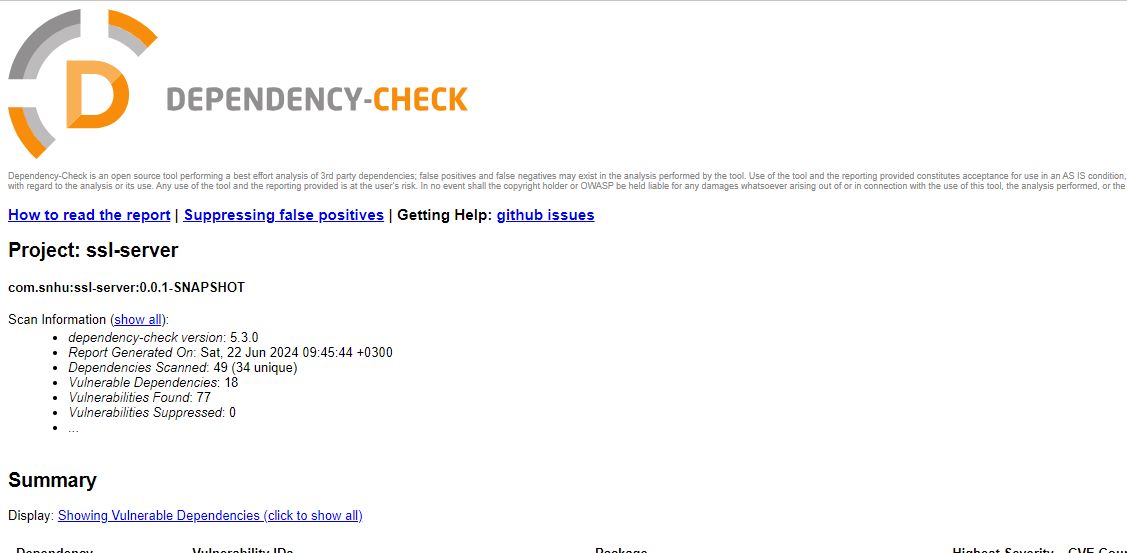
## Secure Communications

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

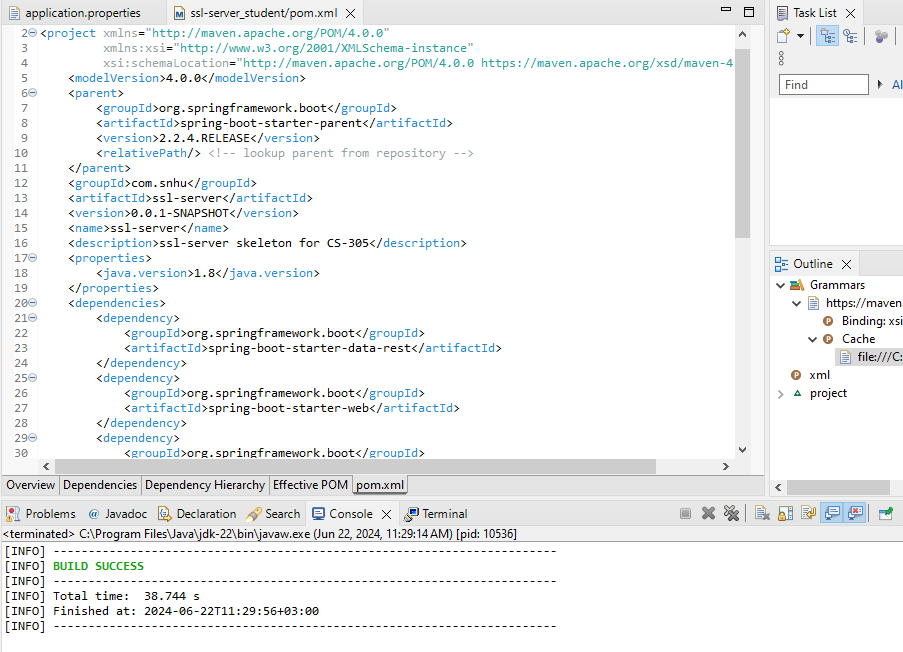


## Secondary Testing

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

## Functional Testing



## Summary

### **Refactoring and Security Compliance in Software Development**

In the context of refactoring a Spring Boot application to enhance security, several key areas were addressed to ensure compliance with security testing protocols and industry best practices. Here’s a detailed discussion based on the points provided:

#### **Areas Addressed by Refactoring:**

1. **HTTPS Implementation**:The code was refactored to enforce HTTPS protocol instead of HTTP. This ensures that data transmitted between the client and server is encrypted, protecting against eavesdropping and tampering during communication.
2. **Secure Storage of Credentials**:Secure management of sensitive information, such as passwords and certificates, was implemented. This includes storing credentials in a secure keystore (keystore.jks), which is accessed securely using proper configuration (server.ssl.\* properties).
3. **Input Validation and Sanitization**:Input parameters, such as the input parameter in the /hash endpoint, were validated to prevent injection attacks and ensure that only expected data types and formats are processed by the application.
4. **Dependency Vulnerability Management**:Dependency-Check tool was employed to scan the application’s libraries and frameworks for known security vulnerabilities. This proactive measure helps identify and mitigate risks associated with third-party dependencies.

#### **Process for Adding Layers of Security:**

1. **Requirement Analysis**:Initially, the requirements were assessed to determine necessary security enhancements, such as transitioning from HTTP to HTTPS, securing credentials, and implementing input validation.
2. **Implementation of HTTPS**:HTTPS configuration was added to the application properties (application.properties) to enforce encrypted communication. This involved generating a self-signed certificate (server.cer) and configuring the server to use it (server.ssl.\* properties).
3. **Secure Credential Management**:Credentials were securely stored in a Java keystore (keystore.jks) using industry-standard practices. This includes specifying the keystore location, password, type (PKCS12), and alias (selfsigned) in the application properties.
4. **Input Validation**:Input validation mechanisms were implemented within the application logic to sanitize user inputs and prevent malicious data from being processed. This ensures data integrity and guards against common vulnerabilities like injection attacks.

## Industry Standard Best Practices

**Maintaining Existing Security**: By adhering to industry best practices, such as HTTPS enforcement, secure credential storage, and input validation, the application maintains robust security standards. This approach ensures compliance with security frameworks and regulations.

**Value of Secure Coding Practices**:Applying industry standard best practices for secure coding is crucial for enhancing the company’s overall security posture. By mitigating known vulnerabilities and adhering to established security protocols, the application reduces the risk of data breaches, protects sensitive information, and fosters trust with users and stakeholders.

**Compliance and Risk Mitigation**:Proactively addressing security concerns through refactoring and adherence to industry standards not only mitigates risks but also aligns the organization with regulatory requirements and industry benchmarks. This proactive approach strengthens the software’s resilience against evolving cyber threats.