****

# 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** | **12/1/2023** | **Robert Murphy** |  |

## 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

Robert Murphy

## Algorithm Cipher

Cryptographic Hash Functions protect against the possibility of forgery (the creation of input data with the same hash as the expected data, a collision) by potentially malicious participants. RSA encryption is an asymmetric encryption algorithm named after its founders (Rivest, Shamir & Adleman) that uses block cipher methodology to encrypt data. Asymmetric encryption, or “public-key cryptography,” pairs two keys together to encrypt and decrypt messages to ensure it is kept secure during a transfer. This method is often considered a better option than Symmetric encryption for larger businesses. The key length and value are determined by a series of transform functions that are fed “seeds” from random number generators. These seeds are used as coefficients, or parameters, of the algorithms’ transform functions. These transform functions are modular within the default JSSE (Java Secure Socket Extension). Several modules are chosen and combined to form “cipher suites”. These suites can be easily implemented and are highly configurable using the modules as parameters.

The nature of the client’s business requires an asymmetrical security approach. As such, the firm has opted to deploy the Java Native RSA keypair generator with SHA-256 (Secure Hash Algorithm utilizing 256-bit key lengths) message digest algorithm. Additionally, SSL (Secure Socket Layer) context algorithm TSL (Transport Security Layer) v1.3 will be used for both intranet and internet transmissions.

The SHA algorithm will be used for the signing of security certificates that authenticate their origin. The sender can sign a certificate using his own private key, and the receiver verifies the certificate using the sender’s public key. Since both keys are mathematically linked, it is impossible to duplicate the public and private key.

The most popular semi-modern use of cryptography was the Enigma machine used by the Germans during World War II. The encrypting mechanism used an electromechanical rotor system to scramble letter input by an attached keyboard. Today, instead of relying on complex mechanical devices, computers use mathematical equations and algorithms to create better encryption.

Modern encryption is proficient at protecting data against common breach methods, but it isn’t future proof. Analysts expect quantum computing to become powerful enough that modern algorithms could easily be defeated. Software developers will need to update their products accordingly before quantum computing becomes commonplace and readily available. However, there are already encryption algorithms such as CRYSTALS-Kyber that NIST has selected as one of the quantum proof algorithms.

## Certificate Generation

Insert a screenshot below of the CER file.

[A screenshot of a computer

Description automatically generated 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

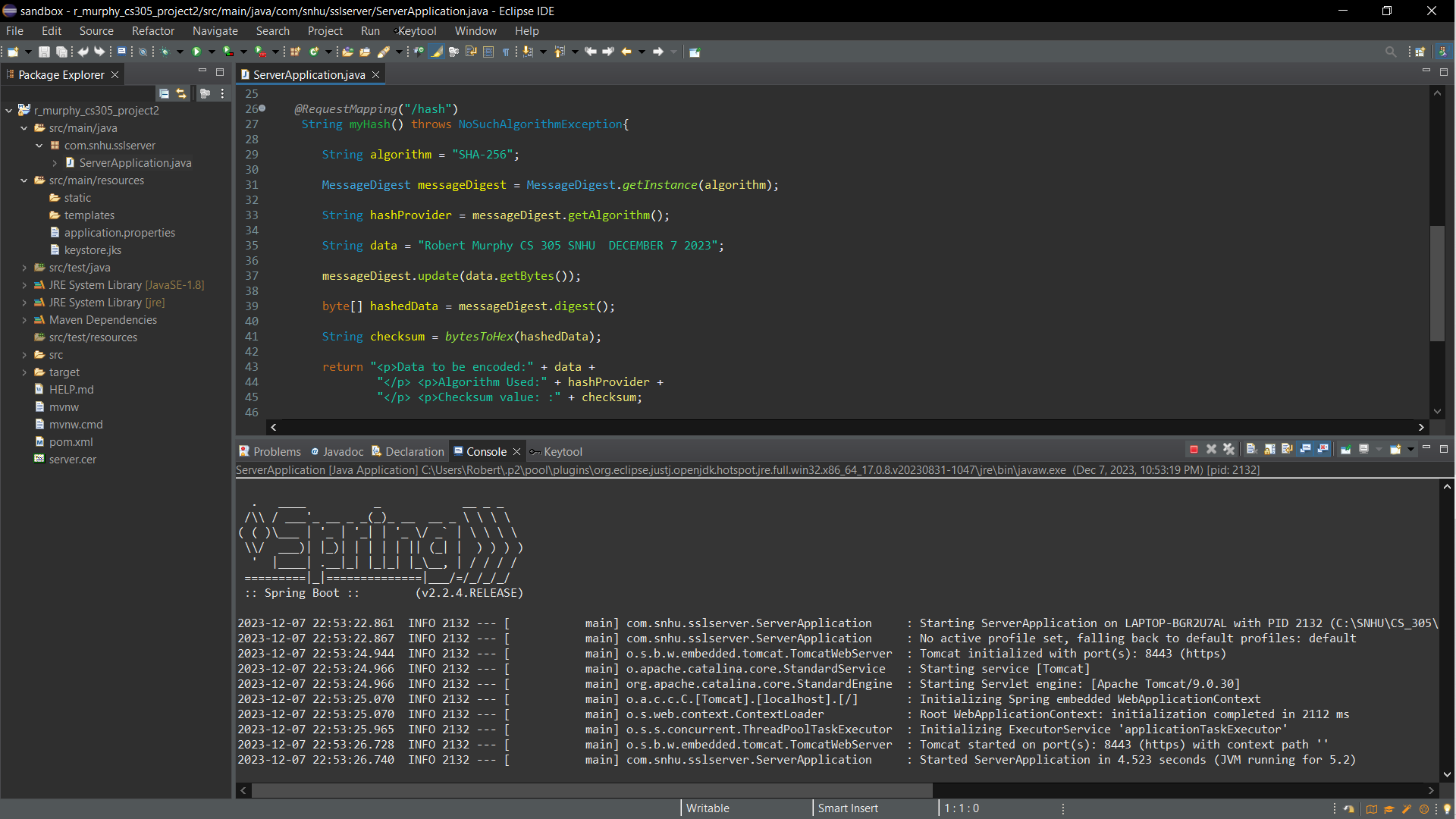
Insert a screenshot below of the web browser that shows a secure webpage.

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 screenshot of a computer program

Description automatically generated

## Summary

The code within the package submitted to the firm did not contain any security layers of any kind. This means that all communications and transactions are critically vulnerable. First and foremost, the data in transit must take priority. Utilizing data encryption and security certificates will shield the data when it leaves the control of an API.

Using an asymmetrical security approach, the firm has opted to deploy the Java Native RSA keypair generator with SHA-256 message digest algorithm. SSL context algorithm TSL v1.3 will be used for all internet transmissions. Any future mobile applications will comply will these implementations. The SHA algorithm deployed for security certificates will authenticate origin and maintain data fidelity by producing a checksum.

Finally, a thorough analysis of the proposed codebase incorporates the OWASP dependency check, giving full insight of potential risks within the design that may not be easily detected otherwise.

## Industry Standard Best Practices

The client’s main program will continue to utilize the Spring framework through Java. With the Spring framework, a common RESTful API was developed and tested using the JUnit test modules. Within the proprietary API, the data to be encrypted has been properly encapsulated by the created server class. Using parameter abstraction, the cipher algorithm is captured and protected. The design of the Server class follows the RESTful API standard set by the professional DevSec community and is easily maintained using occasional dependency checks through the OWASP dependency check utility.