****

# 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 5](#_Toc102040759)

[3. Deploy Cipher 5](#_Toc102040760)

[4. Secure Communications 5](#_Toc102040761)

[5. Secondary Testing 6](#_Toc102040762)

[6. Functional Testing 8](#_Toc102040763)

[7. Summary 8](#_Toc102040764)

[8. Industry Standard Best Practices 9](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[Date]** | **[Your Name]** |  |

## 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

Joshua Brown

## Algorithm Cipher: Recommend an appropriate encryption algorithm cipher to deploy, given the security vulnerabilities, and justify your reasoning. Review the scenario and the supporting materials to support your recommendation. In your practices for secure software report, be sure to address the following:

* 1. Provide a brief, high-level overview of the encryption algorithm cipher.

For the current software, I would recommend using a symmetric bulk cipher. Although an asymmetric cipher may be considered more secure, it would not be the best for this application considering the amount of data that would need to be encrypted. As for the type of bulk cipher, I’d recommend the most popular and more secure AES (Advanced Encryption Standard).

* 1. Discuss the hash functions and bit levels of the cipher.

When it comes to bulk ciphers, AES is government approved and contains a 128-bit security level which provides more security than a 64 bit-security level (such as Triple DES) and is less likely to have a ciphertext collision due to the amount of data. AES also has several key options at 128, 192, and 256 bits with the higher level equating to higher security.

* 1. Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.

Ciphers make use of random numbers to encrypt data; this is done by either using the generation of true random numbers (which can be harder to do) or by the generation of true random seeds (or starting points) which are used to generate pseudo-random numbers.

As I stated earlier, the use of asymmetric cryptography is considered more secure; this lies in the fact that both sender and recipient use 2 separate keys, a public key that is known and a private key that is never shared. When data is encrypted using the private key, the public key must be used to decrypt the message and vice versa. Though this method is more secure, asymmetric encryption tends to be a slower process and is not efficient for bulk data. Symmetric cryptography however uses only one key to encrypt and decrypt that is shared between sender and recipient. Because the key is shared, it is imperative that the key not be compromised or that will defeat the purpose of the encryption; there are methods such as key wraps to encrypt key data to prevent this from happening.

* 1. Describe the history and current state of encryption algorithms.

AES is currently considered the most popular and secure encryption algorithm; however, it is an old cipher created in 2001 and due to technological advances, it is only a matter of time before it is eventually compromised. This is why it is important to periodically check applications for vulnerabilities, and ensure that your application is running using the latest protections and artifacts.

## Certificate Generation: Generate appropriate self-signed certificates using the Java Keytool in Eclipse.

* 1. To demonstrate that the certificate was correctly generated:
     1. Export your certificates (CER file).
     2. Submit a screenshot of the CER file in your practices for secure software report.

A picture containing text, screenshot, document, font

Description automatically generated

## Deploy Cipher: Deploy and implement the cryptographic hash algorithm by refactoring code. Demonstrate functionality with a checksum verification.

1. Submit a screenshot of the checksum verification in your practices for secure software report. The screenshot must show your name and a unique data string that has been created.

A picture containing text, font, white, algebra

Description automatically generated

## Secure Communications: Verify secure communication. In the application.properties file, refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code. Then once the server is running, type https://localhost:8443/hash in a new browser to demonstrate that the secure communication works successfully.

1. Create a screenshot of the web browser that shows a secure webpage and include it in your practices for secure software report.

## 

A screenshot of a computer

Description automatically generated with medium confidence

## Secondary Testing: Run a secondary static testing of the refactored code using the OWASP Dependency-Check Maven (see Supporting Materials) to ensure code complies with software security enhancements. You need to focus on only the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities. Include the following in your practices for secure software report:

1. A screenshot of the refactored code executed without errors

A screenshot of a computer code

Description automatically generated with medium confidence

1. A screenshot of the report of the output from the dependency-check static tester



## Functional Testing: Identify the software application's syntactical, logical, and security vulnerabilities by manually reviewing code.

1. Complete this functional testing and include a screenshot of the refactored code, executed without errors, in your practices for secure software report.

A screenshot of a computer code

Description automatically generated with low confidence

## Summary: Discuss how the code has been refactored and how it complies with security testing protocols. In the summary of your practices for secure software report, be sure to address the following:

1. Refer to the Vulnerability Assessment Process Flow Diagram. Highlight the areas of security that you addressed by refactoring the code.

Areas that need to be addressed in the code are outdated artifacts. Spring Boot used in the code is 2.2.4 and should be updated to a newer version of spring boot; as for what version Spring Boot 2.7 is probably the minimum, but I’d recommend 3.0 or higher. With the upgrading of your use of Spring Boot, it is also necessary to upgrade the version of Java being used in the application. The application currently uses version 1.8 or Java 8. Although popular because it has been around for so long, it is not the best use for this application, especially since both Spring Boot 2.7 and 3.0 require Java 17 or higher.

1. Discuss your process for adding layers of security to the software application.

To add these layers of security, the framework should be changed over to 2.7 or higher; in the refactored code file that I created, I used Spring Boot version to 3.0.6. Once this is done, checks should be done to make sure that everything works fine also checking for new properties that may be required or no longer necessary. The same should be done to make changes from Java 8 to Java 17 or higher; I used Java 17 in the refactored version.

## Industry Standard Best Practices: Explain how you applied industry standard best practices for secure coding to mitigate against known security vulnerabilities. Be sure to address the following:

* 1. Explain how you used industry standard best practices to maintain the software application’s current security.

Vulnerability Flow Chart:

APIs: By using both the static testing and physically reviewing the pom.xml file, I was able to find that the version of Spring Boot and the version of Java were outdated. Upon realizing this, I recreated the application using an updated version of Spring Boot and Java.

Cryptography: By using a secure encryption algorithm (AES) along with a reliant hashing algorithm (SHA-256) I ensured that the message was properly encrypted, and unlikely to be compromised.

* 1. Explain the value of applying industry standard best practices for secure coding to the company’s overall wellbeing.

By ensuring proper coding techniques and checking for vulnerabilities such as input validation to make sure your application can’t be compromised due to user input, cryptography to ensure that your data is secure, and checking to make sure all artifacts and frameworks are up-to-date, will ultimately make your applications safer and more secure helping to prevent breaches due to cyber-attacks against the company.