****

# 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** | **02/17/2025** | **Amanda Nelson** |  |

## 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

Amanda Nelson

## Algorithm Cipher

Due to the clients’ need to add a verification step to the web application it is my recommendation that the appropriate encryption algorithm used is SHA-256. This encryption is a widely recognized standard for generating unique, tamper-proof hashes that can be used to verify that uploaded files have not been tampered with. SHA-256 is easy to implement, given that most platforms and programming languages have built in support for this algorithm. This algorithm also has great security against collisions. This means that there is a very high percentage that no two files will have the same hash values.

The purpose of the cipher's hash function is to keep data safe and protected. To do this the cipher generates a unique, fixed-size string from the data. This makes it so the data can be verified to check that it was not altered in some way. The bit levels are used for the amount of security needed. For example, the SHA-256, that was recommended, is a higher bit level than SHA1. The bit levels mean that there is a larger range of potential hash values. Making it harder to decipher (Detlefsen et al., 2015).

The use of random numbers is to add unpredictability to the encryption. This makes the key stronger and more unique. The difference between symmetric versus non-symmetric keys is that symmetric keys use the same key to both encrypt and decrypt. While non-symmetric keys, also known as asymmetric, are where one is made private and kept secret, this key is for decryption. While the other is made public and used for encryption.

The history of encryption algorithms can be dated back to 1900 BC by hieroglyphs in the Old Kingdom of Egypt (Schneider, 2024). The form of Morse code is the most known encryption used in WWI and WWII. Morse code was invented to encrypt telegrams for long distance trips. The current state of encryption algorithms is all computer based now and is far more advanced. The most used today is the AES-256. This algorithm is also known as the best there is thus far.

## Certificate Generation

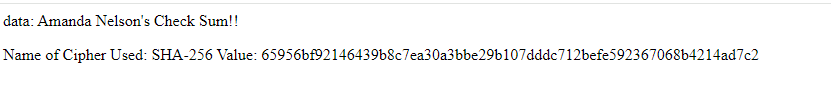
Insert a screenshot below of the CER file.

A screenshot of a certificate

AI-generated content may be incorrect.

## 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

AI-generated content may be incorrect.

My browser is showing HTTPS crossed off, likely due to the use of a self-signed certificate. Browsers do not trust self-signed certificates by default, because they are not issued by a trusted Certificate Authority (CA).

## Secondary Testing

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

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Summary

When refactoring the code, I added a RestController to the SslServerApplication.java file. This will handle incoming HTTP requests and provide responses. It will also take the resuested endpoint of “/hash” and output the message as well as the encryption of the message. Within this RestController I added extra security in the form of an authentication and encryption algorithm, known as SHA-256. I used MessageDigest to get the instance of SHA-256, then calculated the digest hash of the input. BigInterger is used to exceed the limits of fixed sizes. Therefore, it is great to use when encryption is necessary. HexString is then used to convert the byte array of the digest into a hexadecimal string and returns that hash. After creating a self-signed certificate, I then had to update the application.preperties file. This included using the same alias, password, store and store type as I did with the creation of the certificate. This is meant to turn HTTP into HTTPS for even more added security.

## Industry Standard Best Practices

The value of using industry standard best practices for a company’s overall well-being is immense. By using best practices as a developer you are minimizing risks of cyberattacks, protecting clients’ sensitive data, safeguarding a company’s reputation, and maintaining user trust for that company. For secure practices, things like input validation, error handling, authentication and authorization, data protection, code reviews, and dependency management should be considered and used going forward. Input validation is used to filter out things like SQL injection, which can destroy databases. Error handling is used to avoid leaking sensitive data from companies and clients. Authentication and authorization is great for verifying users are who they say they are, and that restricted users are only getting the information they are allowed. When using data protection, it is important to make sure the data is being protected when in transit and at rest. Code reviews should be done regularly to identify potential vulnerabilities, and the dependencies checks need to be done by a trusted third-party to make sure there are no vulnerabilities within the dependencies being used.

**Resources**

Detlefsen, A., Manico, J., & Kenan, K. (2015). Iron-Clad Java: Building Secure Web Applications (M. Smith, Ed.) (1st ed., p. 263). McGraw-Hill Education.

Schneider, J. (2024, November 25). *The history of Cryptography*. IBM. https://www.ibm.com/think/topics/cryptography-