****

# 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** | **[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

Ali ARSLAN

## Algorithm Cipher

Encryption is a technique developed to ensure that data can only be accessed by specific individuals. Historically, one of the earliest methods was the Caesar cipher used by Julius Caesar, where each letter in a message was shifted by a certain number of places in the alphabet. Today, encryption has evolved into complex mathematical processes involving different types of algorithms such as symmetric encryption, asymmetric encryption, and hash algorithms. Symmetric encryption uses a single key for both encryption and decryption, while asymmetric encryption involves a pair of keys: a public key for encryption and a private key for decryption. Hash algorithms, like SHA-256, are used to generate a fixed-length hash value from data and are irreversible, ensuring data integrity without the need for decryption The bit level of the algorithm determines this fixed length, specifying the total number of bits produced, which is the length of the hash. The longer the bit key, the harder it is to produce the same hash, but it also increases the computational power and complexity required to calculate it. .

In modern encryption practices, the security of data is further enhanced through the use of large prime numbers in asymmetric encryption, which makes it difficult to predict the keys. Symmetric encryption, considered the most secure for certain applications, is used in scenarios where key distribution can be physically controlled. Hash functions play a crucial role in verifying data integrity, particularly in systems like CADES, which is used for securely storing and verifying archived files. The AES algorithm is widely trusted for encrypting data due to its reliability and symmetric nature, while RSA is commonly used in digital signatures for its asymmetric properties. These advancements have made encryption a cornerstone of data security in today's digital world.

In light of all this information, at least one of these encryption algorithms is definitely present in any application. We have to choose some of them in the project, the reasons for choosing the encryption methods we will choose are stated below.

* Asymmetric Encryption for secure connection
* Hash for storing user passwords
* We will use symmetric encryption and a hash algorithm for encrypting and securely storing files
* The algorithms chosen for this and the reasons are presented below.

**HASH:** We will use SHA-256 for hash, this method has a higher bit level (256 bits) and the probability of creating the same hash value (Collusion) is lower than MD5 or SHA-1 and the complexity level is at a level that modern systems can easily handle, so it is used as the default hash algorithm in many encryption systems.

**Asymmetric Encryption:** RSA has proven its reliability and is the standard algorithm used in most security certificates, so it is considered appropriate to use it for symmetric encryption.

**Symmetric Encryption:** AES (Formerly known Rjindel) This algorithm was chosen because it is the most used algorithm in the industry, since a single key is used, the higher the key and salt bit length is, the higher the security it provides.

## Certificate Generation

Insert a screenshot below of the CER file.

Since the application will be running on localhost at the moment, CN (Common name must be created as localhost) so that browsers do not give a warning

First of all, we need to create a keystore based on this information

keytool -genkeypair -alias localhost -keyalg RSA -keysize 2048 -validity 365 -keystore "C:\Users\Ali\Desktop\SNHU\CS-305\week 7\certificate\keystore.jks" -dname "CN=localhost, OU=CS305\_Ali\_ARSLAN, O=SNHU, L=Watertown, ST=Massachussts, C=US"

The command will create the keystore for us, then we need to produce a CER certificate file from this key store

keytool -exportcert -alias localhost -file "C:\Users\Ali\Desktop\SNHU\CS-305\week 7\certificate\localhost.cer" -keystore "C:\Users\Ali\Desktop\SNHU\CS-305\week 7\certificate\keystore.jks" -rfc"

We can create the certificate this way but this is not enough now since this certificate does not come from a CA there is no CRL list to verify it so we need to transfer this certificate to our own operating system's trusted root certificates directory but after this process the certificate is ready to be usedyou can see the screenshots below

A computer screen shot of a black screen

Description automatically generated

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a certificate

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 screen

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

A screenshot of a computer

Description automatically generated

## Functional Testing

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

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

First, I added a controller to the SslserverApplication.java file to handle and display the hash operation. While writing this code, I used the @RequestMapping annotation to redirect this controller to the hash directory. Then, I created a keystore with CN and SAN set to localhost and the IP address 127.0.0.1. From this keystore, I generated two certificates and installed the CER one into the Windows Trusted Root Certification Authorities directory so that browsers would not show errors. I placed the P12 file under the resources directory. After that, I configured the settings in the application.properties file, ensuring that the certificate would work as expected and that no errors or warnings would be displayed.

When I ran a static test, it gave many warnings, but the only reason was an old version of Spring, so I upgraded it to version 3.3.2. However, after this, I continued to receive warnings for spring-web and snakeyaml 2.2, which were the default versions used by this version. Therefore, I forced spring-web to use version 6.1.12 by overriding it. Since the latest version of snakeyaml is 2.2, which has a security vulnerability and no patched versions have been released yet, I excluded it from all dependencies using it, thus preventing its usage. This resolved all issues. Finally, I manually reviewed the code to check for any errors, conducted a debug session by setting breakpoints, and verified the data. After ensuring that the code ran without errors and the data was handled correctly, I ran the code and checked the console output for errors, confirming that it worked flawlessly. Thus, I ensured that the program met all requirements and was secure.

## Industry Standard Best Practices

Following industry standards is crucial for ensuring the quality and security of an application. This is applied extensively, from selecting encryption algorithms to writing code.

When writing code, I chose the Allman notation, which is one of the standards, to ensure high readability. Secure coding standards were followed during the code development. Although there is no input validation, secure coding standards were adhered to by releasing the resources used and applying exception handling.

OWASP dependency testing was performed, and all warning modules were upgraded to issue-free versions. Modules that could not be updated were excluded from the project to prevent their use. Error handling was also implemented to prevent critical errors, thus maintaining security standards.

Following these standards provides significant benefits to the company. Additionally, in Europe, many projects require these standards to be applied; otherwise, the company may face legal responsibilities in case of issues. Thus, adhering to these standards can be considered a primary step in protecting the company from legal obligations. Furthermore, even when not legally required, it ensures user or customer trust, positively impacting the company’s reputation, which in turn benefits as a customer. It also reduces insurance premiums for the company (especially if certifications for these standards are obtained by the company and its employees) and has a positive effect on costs.