****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **08/14/2022** | **Zavalla Huggins** |  |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Zavalla Huggins

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

SHA stands for secure hash algorithm, in 256-bit. All hash functions are irreversible and all of the hashes created take 2^256 attempts to generate the same data. There is a one in 2^256 chance of the same hash being created.

Algorithms are step-by-step instructions that must be completed to achieve a goal. As algorithms become more complicated, they also become more vulnerable. Algorithms were first used in mathematics which are represented by widely accepted formulas that we still use today. The first known encryption algorithms was the Caeser cipher, used by the Romans in 60 BC. This cipher simply shifted letters by three spaces, which a Caeser cipher was used to unencrypt the message. Encryption in the present uses the same principle of changing information to hide it, and then using a key to translate the message once received back. Casual web-surfing usually utilize a public key infrastructure, also known as an asymmetric algorithm. These keys can be created with random number generators. RNA’s are used since it is not likely for someone to guess a key, especially if it is 256 digits, but does not mean someone will not attempt to. More private data connections typically use an RSA algorithm of some sort. RSA’s use a combination of both private, and public keys to encrypt data.

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.

A computer screen capture

Description automatically generated with medium confidence

Text

Description automatically generated

## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

Graphical user interface, text, application, email

Description automatically generated

## 4. Secure Communications

Graphical user interface, text, application

Description automatically generatedRefactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on 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.

Graphical user interface, text

Description automatically generated with medium confidence

Graphical user interface, application, Teams

Description automatically generated

## 6. Functional Testing

First, a message digest object is created with an instance in SHA-256 format and stored in the variable, md. The md message digest variable is then converted to a hash and stored in an array of bytes. The hash variable is then converted to a hex string and stored in the variable hexString. There is a 0 inserted into the hex string and stored back into the hexString variable. The next function takes the string generated and prints it on the dashboard.

Graphical user interface, text, application

Description automatically generated

## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

The changes added are heavily hinged on cryptography. The information is stored into 256 bit string, which is then translated into a hash, and then translated back using the public key generated. This facilitates a more secure API transaction than if there was no translation. The translation is encapsulated into one function that can be executed in many different situations.

These security measures help protect the data of a plethora of people. The information of employees are protected. Also, the information of customers and vendors is more protected with the SHA-256 algorithm.

As a user, the number of digits used in the cipher should not be shared, since this helps a hacker know what length of key they are looking for. This weeds out the possibility of there being 128 bytes, 64 bytes, etc. Common practice is that any login information should not be shared either. These gives hackers the easiest way in as possible, and makes them harder to track, since the account is not seen as a foreign entity. User permissions should only be given as needed. Having an overprivileged user opens the opportunity for that user or an outside entity to use the account for malicious things, such as granting other users permission, or deleting/copying sensitive data.

References

Thales Group. (2021, October 1). *A brief history of encryption (and cryptography)*. https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption