****

# 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** | **06/14/2021** | **Joshua Major** | **Initial version** |

## 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

Joshua Major

## 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.

**Provide a brief, high-level overview of the encryption algorithm cipher:** The cipher used in this program is the SHA-3, Secure Hash Algorithm 3, standard that was released by NIST in 2015. SHA-3 is the most secure cipher of the SHA cryptographic family and is especially suited to avoid collisions.

**Discuss the hash functions and bit levels of the cipher:** For this program we will be using the 256 bit version of SHA-3. This will allow communication at the rate of 136 bytes / 1104 cycles at 306 MHz. command. The cipher uses a hash function to take in data of any length and map that data to a fixed size data set that is unreadable.

**Explain the use of random numbers, symmetric vs non-symmetric keys, and so on:** Because hash functions divide the input data by a numbering system in order to output the hashes, random numbers must be used in order to prevent non-authorized people from deciphering the data. These random numbers are then used to make the cipher key that is used to decrypt the data. These keys can be ether symmetric or asymmetric. The symmetric key used the same private key to encrypt and decrypt the data. Asymmetric keys have a private and a public key that are different so that once key can’t be used to access all of the data.

**Describe the history and current state of encryption algorithms:** Ciphers have been in use well before the time of computers. With the advancement of technology it has become easier to decrypt ciphers because you can use a computer to try many different combinations until you stumble across the right key. This is called a brute force attack. Because of this ciphers have needed to become more complex and use mathematical functions, such as hash functions, in order to become harder to break.

## 2. Certificate Generation

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

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

Graphical user interface, text, application

Description automatically generated

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.

* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.

A screenshot of a computer

Description automatically generated with medium confidence

## 4. Secure Communications

Refactor 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.

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

Graphical user interface, text, application

Description automatically generated

## 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.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.

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.

**Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code:** In the code refactoring I addressed input validation, cryptography, code error, code quality, and encapsulation.

**Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing:** The first layer of security that was added was data encryption. The code was refactored to use a SHA-256 cipher to create a checksum verification. The second layer of security was to create a self-signed certificate to authenticate. The last bit of security comes from having error free quality code that is encapsulated. This prevents faults within the code from being taken advantage of by hackers.

**Point out best practices for maintaining the current security of the software application to your customer:** The best practice for maintaining the security of the code is to continually check for updated in the code base that may address newly identified vulnerabilities. This includes running an occasional dependency vulnerability check.