****

# 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** | **08/21/22** | **Kalin Mason** |  |

## 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

Kalin Mason

## 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.

A top rated encryption algorithm cipher is the AES cipher because of it’s support for a large amount of key sizes and one of the best standards in the industry. Offering a 128-bit and 256-bit key for the application is preferred while the 256-bit is harder to crack. The symmetric keys created allow for the application to encrypt data when requested and returned keys to the client.

The use of symmetric and non-symmetric keys depends on the purpose of the application. Symmetric keys are shared between the client and server while non-symmetric keys include public and private keys with private keys only been shared to the client. The purpose of these keys is to encrypt and decrypt data when the correct key is used. One downside to this is that if a key is lost, the data is also lost and can’t be decrypted. Currently, AES has the capability to encrypt using up to a 256-bit which is highly resistant to being cracked.

## 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.

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.

[Insert screenshot(s) here.]

## 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.

[Insert screenshot(s) here.]

## 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

Text

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.

With the refactored code we have addressed the following areas of security; APIs, cryptography, and client/server. During this module I’ve created my own self signed certificate and have generated keys for the application as well. With this encryption only the intended recipients can view and access the data and information from the server. When working with and creating secure communications, it’s important to maintain the encryption algorithms in the application to prevent unsecure communication leaks and attacks on the application. Security breaches can cause financial losses, lost of trust between clients and the application owner, and increase in expenses for the company to repair the vulnerabilities. One of the best methods to maintain security in an application is check the code for vulnerability, even more so after implementing new features or changing existing functions. If a vulnerability doesn’t have a current resolution, then the development team will need to decide which solution would work best to address the vulnerability. Now if the development team finds that a vulnerability doesn’t affect the code base or it’s elements, they can choose to suppress it in the report.