****

# 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** | **2/25/22** | **Jayd Cheshire** |  |

## 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

Jayd Cheshire

## 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.

The encryption algorithm cipher that I would recommend deploying to the given Artemis Financial’s needs is TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA. This would fall in line with security protection best practices in order to defend against various types of security attacks because many of the others contain vulnerabilities. The hash algorithm SHA is extremely secure because of how large the number is there is no reliable way to brute force it. Bit levels of the cipher determine how big the cipher is. Random numbers are used to seed algorithms. These random numbers are gathered typically via hardware events like interrupts or other internal signals. The amount of random numbers then is limited by the amount of events completed on the hardware level. If you use all these numbers/events, then you must wait for more events to happen to be able to grab another random number. Symmetric cryptography is when the same key is used for both encryption and decryption. This however means that the key needs to be shielded from possible attackers, so you will want to typically encrypt the key as well. Asymmetric cryptography is when you have two separate keys. One key will need to be kept private and will only be known to the sender. The other key can be made public and is used to validate the signature. The history of encryption is a long one starting with small bit levels, and by today’s standards many vulnerabilities. As time went on the bit level has increased, and many known vulnerabilities have been remedied. Also support for many functionalities has increased but is not complete.

## 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.

Graphical user interface, text, application

Description automatically generated

## 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

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

## 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.

Text

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.

Things like input validation and encapsulation were not really implemented in this refractoring, however things like, cryptography, and code quality were implemented. Cryptography was used in the form of SHA256 bit encryption which was used to has the string containing my name. The process of adding layers of security to the software application must begin with the start of the project and should not be delayed until later. Security needs to be built into the design of the code ensuring that the vulnerability assessment process is implemented at every step of the way. It is better to put in the time and money upfront to add security rather than having an incident, getting fined, and then having to implement the security the right way anyway. This can be very costly for a company. Best practices for maintaining the current security of the software application to my customer is to ensure that everything is updated to the current version or a version that mitigates the vulnerability, and that the dependencies have no known vulnerabilities.