****

# 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** | **2/20/2022** | **Matthew Trembley** |  |

## 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

Matthew Trembley

## 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 company Artemis Financial has contacted Global Rain in an attempt to add more security for their web application for more secure communication. Generally, the most common attack on a financial company would be a bad actor trying to access private information of clients of said company. With this being said, encryption of private data would be the best option. I recommend using the SHA-256 cipher algorithm with 256 bit keys for encryption. Hash functions will be used in conjunction with SHA-256 cipher to create a checksum of a message, that way to prove that it works. Also, this algorithm allows for near endless key combinations with its length of 256 bits. On top of that, the SHA-256 uses a random number generator to create random numbers which ensures that the encryption is secure by creating irreversible checksums.

## 2. Certificate Generation

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

* Text

  Description automatically generatedText

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

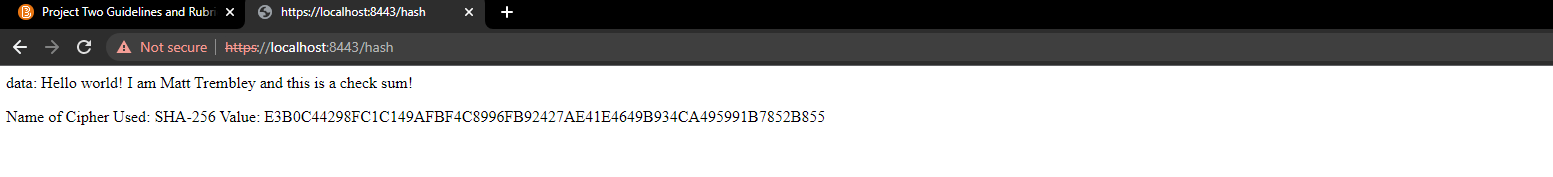
Graphical user interface, text, application

Description automatically generated

## 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.



## 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.

Because the certification is self-signed, the browser can not validate it’s authenticity and presents itself as Not secure.

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 generatedGraphical user interface, application, Teams

Description automatically generated

## 6. Functional Testing

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

Text

Description automatically generatedText

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.

In refactoring the code, it was important to add a secure @RestController, as to work as a secure controller for a RESTful stop for the program. The ServerController class enacts cryptography, which is the main goal for Artemis Financial in securing their communications. Alongside it, it also uses input validation and secure error handling to help security measures. The use of SHA-256 algorithm cipher it incredibly secure as its chance for collisions is very minute. Going forward, future dependency checks to help with updating databases and libraries will help keep the application more secure. I recommend monthly or every 2 month checks to keep up to date.