****

# 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** | **08/21/2022** | **Drew Pepin** |  |

## 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

## 1. Algorithm Cipher

The best and most appropriate encryption algorithm for this specific assignment from Artemis Financial would-be encryption of user and private information. The likely attack on Artemis would be an outside attack from someone looking to access user information and financial information, and the encryption and securing the data will help prevent those attacks. Asymmetric communication would be what I recommend, using SHA-256 would encrypt the information using 256-bit keys. 256 bits allows for several combinations for the hashes used making it nearly impossible for info to be cracked and accessed illegally.

## 2. Certificate Generation

Graphical user interface, text, letter, email

Description automatically generated

## 3. Deploy Cipher

## Graphical user interface, text, application, email 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.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## 5. Secondary Testing

Graphical user interface, text, application, email

Description automatically generated

## 6. Functional Testing

Text

Description automatically generated

## 7. Summary

The RestController works as a secured controller for the hash, and the ServerController is a class that matches problems that the vulnerability diagram shows. Using SHA-256 secures the user data and has a small chance of collision when used. To maintain the security for the program, keeping the depencies up to date and following up with any concerns presented by customers / developers would be my first steps.