****

# 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** | **June 20, 2021** | **Rose Weber** | **Added initial information to all sections.** |

## 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

Rose E. Weber

## 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.

SHA-256 is a hashing algorithm cipher commonly used to provide verification that two pieces of data are identical. The SHA part of the name stands for Secure Hash Algorithm and the numeric 256 part of the name refers to the fact that no matter what data is input into the algorithm cipher, the cipher will always output data with the length 256. While there are many other algorithm ciphers in use today, I would recommend the SHA-256 cipher because it has not been found to have any collisions, that is to say, this cipher has never produced two outputs that are the same. This means that the SHA-256 hashing cipher is trustworthy to verify the integrity of a piece of data.

## 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.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

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, 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.

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

Graphical user interface, text, application, email

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, text, application, email

Description automatically generated

Graphical user interface, text, application

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, email

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.

By refactoring the code I’ve mitigated the major issue regarding file validity and integrity. I modified the code to include a checksum value by way of the algorithm SHA-256. This basic check ensures that the user is able to make sure that the information they are seeing has not been modified or changed in any way. By doing this I have successfully met the requirements set by the client, Artemis Financials.

There is a great deal of value added by the inclusion of data validation. By providing a check for each piece of data that is transferred, the client, and all other parties involved, can feel secure that the information has not been modified by unintended parties. Feeling secure in a company’s ability to handle sensitive data carefully and well is of the utmost importance for any company, but most definitely for a Financial company such as Artemis Financials.

One of the easiest ways for unwanted third parties to work their way into a system is by that system not being maintained and updated to plug up any holes in the security system. In order to prevent this from happening I would recommend that Artemis Financials have their security officer periodically check for issues with the current algorithm cipher and change it to the most secure algorithm later down the road. In addition, I would recommend checking for issues with the dependencies and take action to prevent those issues from affecting the program as a whole.