****

# 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** | **02/16/2022** | **Hanah Deering** |  |

## 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

Hanah Deering­­

## 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.

Given the security vulnerabilities and the scale of what we are working with in this program, the appropriate encryption algorithm cipher to deploy is the Advanced Encryption Standard (AES) in conjunction with a SHA-256 hash function. AES is the strongest encryption standard around, as it is used by most banking institutions and even the United States Federal government. There are many applications we use in our daily lives that use this encryption algorithm cipher. AES uses symmetric keys (which can be 128, 192, or 256 bit) to encrypt and decrypt data. And since SHA-256 takes input data as plaintext and converts it to a ciphertext (Crane).

Using Secure Hashing Algorithms(SHA)SHA-256 as a hash function means that there are 2256 possible hash combinations (1.1579209e+77 possible combinations). As you can imagine with a number that large, using AES as the encryption algorithm cipher means that it is virtually impossible for any attacks to impede on sensitive data and very unlikely to create any collisions. This is not only the safest option for consumers, but also the most efficient option for developers.

Non-symmetric keys use two keys, one to encrypt data and another to decrypt data. One of these keys of generally public which allows for outside users (public) to send messages or inquiries to the system. The second key generally remains private so that responses are always protected because they tend to entail sensitive personal or company information. Symmetric keys use the same key for both the encryption and decryption of data. In order to use a symmetric key both the client and server must have private information.

AES is currently one of the most secure options available for data encryption. Data Encryption Standard (DES) used to be the standard encryption by the United States but replaced because of the multiple times being broken. It is said that this is due to the fact that it’s key is only 56 bits. This is drastically less than that of AES. With that being said, we can say that encryption algorithms have evolved over time.

## 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.

Text

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 with medium confidence

## 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

Refactored code executed without errors:

Text

Description automatically generated

Dependency check report:

Graphical user interface, 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.

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.

Through refactoring code, security is improved. The code was addressed to address cryptography. Refactoring was completed to include a hash function (SHA-256). Through creating a RESTful application, the API endures improved security, making it more secure. Adding a certificate improved the client/server security so that data could be transferred in a more secure fashion. A try and catch method was introduced to ensure best coding practices. Along with making sure that the spring-boot-parent and tomcat dependencies were updated to the latest versions increases the security of the application and protects against any know vulnerabilities.

This safer way of communication was made possible with a combination of a couple of different methods. By hashing the information, creating a RESTful application, updating all dependencies to the latest versions available, and using an SSL connection, the communications between Artemis Financial and their clients will be more secure.