****

# 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** | **4-20-2022** | **Meagan Holub** |  |

## 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

Meagan Holub

## 1. Algorithm Cipher

**Provide a brief, high-level overview of the encryption algorithm cipher.**  
AES 256-bit encryption is the strongest and most robust encryption standard that is commercially available today. While it is theoretically true that AES 256-bit encryption is harder to crack than AES 128-bit encryption, AES 128-bit encryption has never been cracked.

**Discuss the hash functions and bit levels of the cipher.**Values returned by a hash function are called message digest or hash values.  
Hash functions are extremely useful and appear in almost all information security applications. A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length. ADD MORE

**Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.**  
To ensure reliable hash keys a secure random number generator is needed. These are built of random bits of data that can be acquired through small deviations in systems. Because you need random combinations of numbers to form these hash keys and not be reused it becomes imperative to have a large bank of random numbers.  
Symmetric vs non-symmetric keys I have explained somewhat previously. But to be specific a symmetric key is a key that is used for both encrypting and decrypting data. To use a symmetric key both the client and the server must have private information. To achieve this the connection would need to be setup separately. A non-symmetric encryption would use separate keys for encrypting and decrypting.

**Describe the history and current state of encryption algorithms.**  
Historically ciphers have been used to send messages as secret communication. Commonly this was used as a means for militaries to communicate. These cyphers were initially simple they would use letter pairings to hide messages. However, these are very breakable ciphers since words follow patterns. To make these cyphers more complex cryptographic shifting started being used this is when the messages are encoded using a matrix which makes them virtually unreadable using standard pattern recognition. Today the current standard is RSA. And one of the most common practices is known as 2FA or 2 factor authentications. Which uses two separate methods to send private information.

## 2. Certificate Generation

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

Graphical user interface, text, application, email

Description automatically generated

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.

Graphical user interface, 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.

Text

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.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

## 6. Functional Testing

A screenshot of a computer

Description automatically generated

## 7. Summary

The first area of security that was addressed was cryptography. Refactoring to include a hash function encrypted the data. By creating a RESTful application, the API was made more secure. Client/server security was improved by adding a certificate so the data could be transferred in a more secure fashion. Including a try, catch ensured secure coding practices were followed. Finally, ensuring the latest versions of the spring-boot-parent and tomcat dependencies were in use increases the security of the application by protecting against known vulnerabilities. Creating a safe way for communication to occur by hashing the information, using an SSL connection, and creating a RESTful application means that the communications adds value, because communications between Artemis Financial’s systems and their clients will be more secure. In addition, by updating all versions of dependencies to the latest versions available, the application is free of known vulnerabilities that could lead to security weaknesses. To keep the application secure, dependencies used to build the application should be checked for new vulnerabilities using the Dependency-Check tool. The dependencies should be updated to newer versions when vulnerabilities are found. Running a maven check is the best way to do this and after I write any code I install a dependency check.