****

# 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** | **2/15/2021** | **Kennedy Uzoho** |  |

## 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

Kennedy Uzoho

## 1. Algorithm Cipher

The appropriate encryption algorithm cipher I proposed to be used for the software security vulnerabilities mitigation is the AES-256 algorithm cipher.

AES 256 algorithm cipher is one of the distro of Secure Encryption Algorithm, specifically, it is called Advanced Encryption Standards (AES), and it was first introduced in 2001 by the United States Government and the cipher was adopted by many private business sectors. Before then, there was other encryption cipher functions like DES algorithm 56 bit key, AES-256 bits key was a successor to DES 56 bits. AES-256 is a very patented cryptographic encryption function that can encrypt a value that is 256 bits long. The most important feature of AES-256 is the resilience it has in protecting data from unintended access or hackers. For example, AES-256 has been used in some of the most popular authentication and encryption protocols, including bank transactions encryption, important data or classified data, online shopping, and social media apps data. The most important feature of AES is the resilience it has in protecting large data from unintended access or hackers and its symmetric cryptography function that allows using the same key to decrypt the encrypted data on the other end. Unlike, RSA which is asymmetric, RSA is asymmetric, it uses public and private keys, the speed for RSA is not that convenient for handling large data, however, RSA is mostly useful in handling SSL/TLS certificates, and browsers. Another type of software security system is SHA-256, SHA is nothing like AES because it is a hash function, it belongs to the family of Hash algorithms and is not an encryption algorithm. All of them turn data into random ciphertext that will be very hard to decrypt/read.

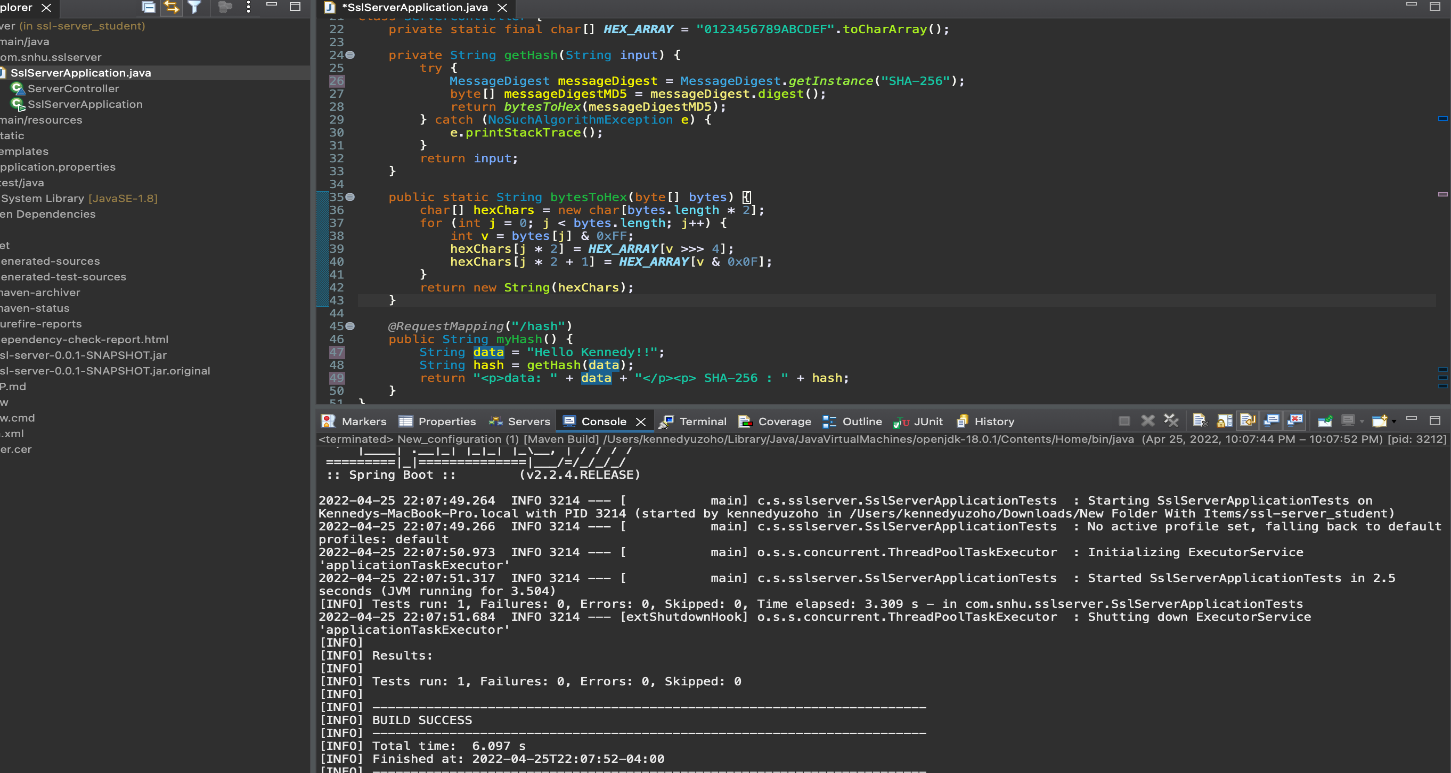
## 2. Certificate Generation

Generated appropriate self-signed certificates using the Java Keytool, which can be used through the command line. A screenshot of a computer

Description automatically generated with medium confidence

## 3. Deploy Cipher

I Refactored the code base and use security libraries to deploy and implement an encryption algorithm cipher to the software application in dev environment. I did verify the process through additional functionality with a checksum verification hash function.



## 4. Secure Communications

Refactored 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, chat or text message

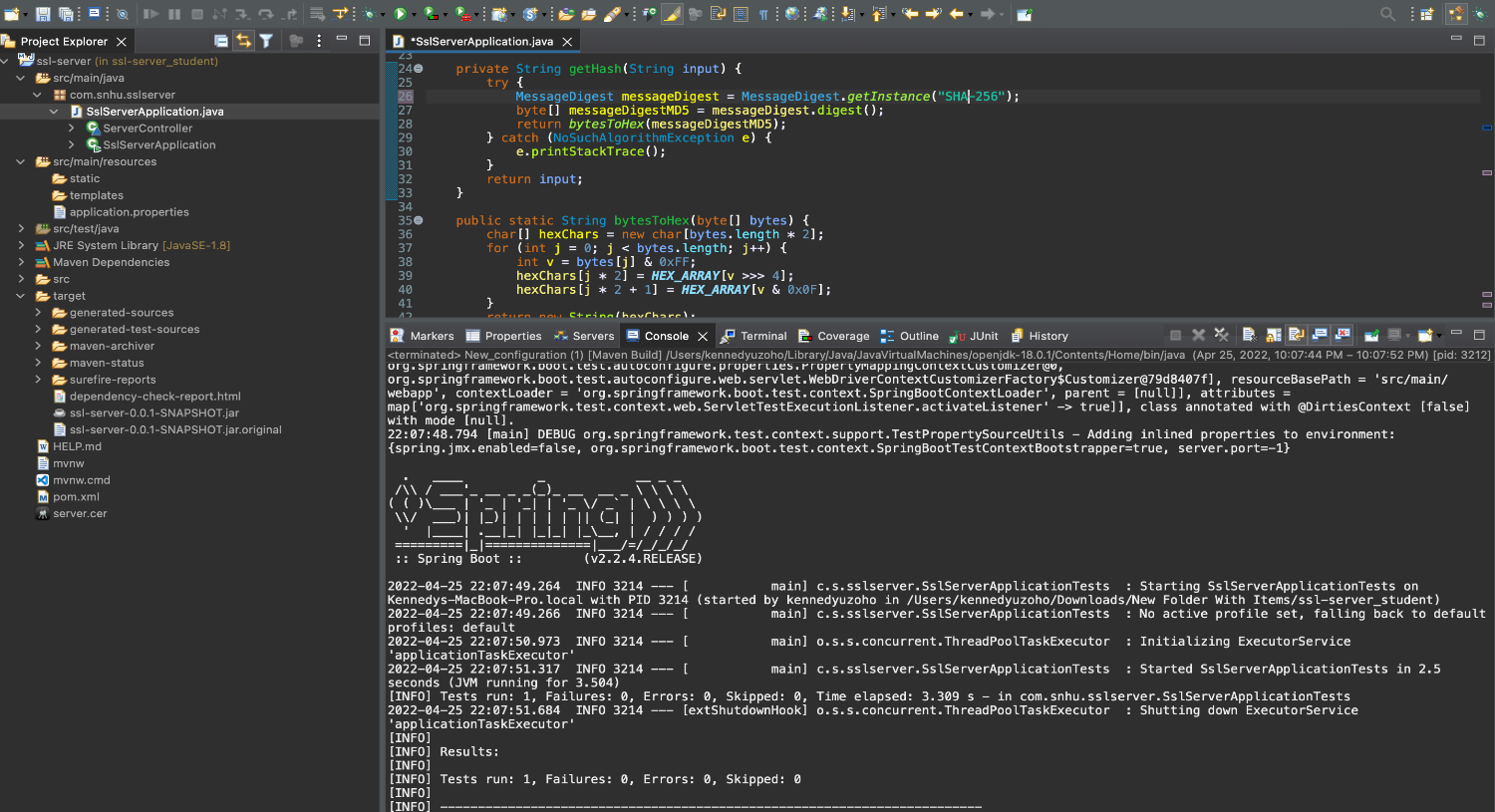
Description automatically generated

For some reason, chrome was the browser used and it seems to not trust the certificate generated in the previous step.

## 5. Secondary Testing

Completed a secondary static testing of the refactored code using the dependency check tool to ensure the code complies with software security enhancements. Here is the part of the code I refactored. Completed a dependency check and review the output to ensure I did not introduce additional security vulnerabilities. Below are details.

A screenshot of the refactored code executed





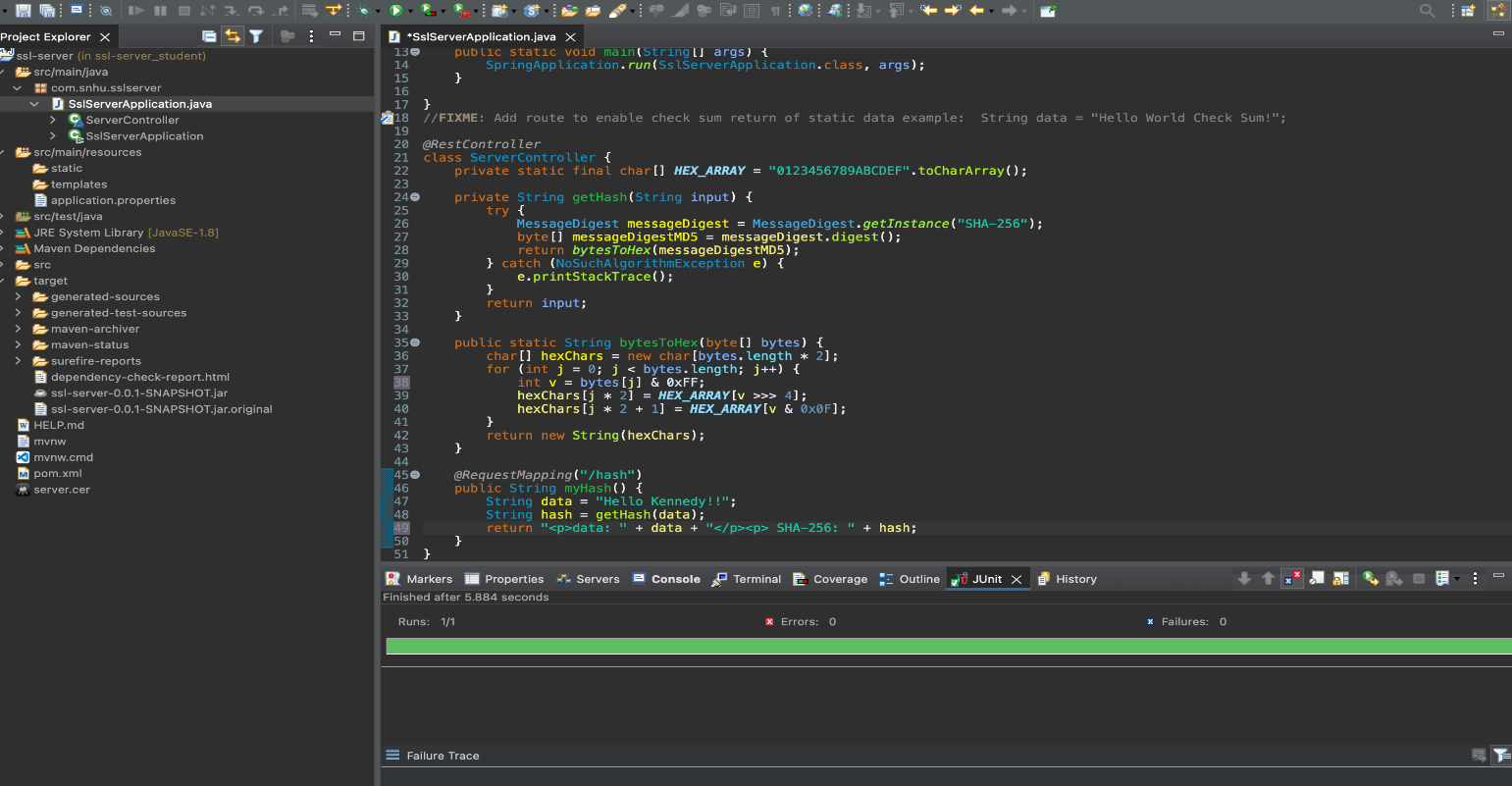
A screenshot of the dependency check report

Graphical user interface, text, application, email

Description automatically generated

## 6. Functional Testing

Completed a JUnit functional testing and here is the screenshot of the refactored code executed without errors. The dependency check report has other vulnerable links information, I believe that some links are very old and needs reviewing, so I will keep working on it and make sure we are clear, but for the main time the software can be used and managed as usual.



## 7. Summary

## Here is a summary of the how the code base was refactored.

## The Spring web application initializer was used to generate a code base and I implemented a unit, functional and dependency vulnerabilities check. The refactored code has a function that will take an alphanumerical string and encode it using cipher text hash values, this is not exactly encryption, there was a key pair generated in the code directories, that key pair used the RSA function, which is a typical encryption function. I particularly did well in verifying the function with a checksum verification. Checksum is what most of the software developers use to perform a secure file transfer and downloads.