****

# 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** | **10/22/2021** | **Tim Sudik** | **Completed security report for Artemis Financial** |

## 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 inyour own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Tim Sudik

## 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.

The algorithm that was chosen is AES-128, which is a 128 bit block cipher that supports keys of 128, 192, and 256 bits. AES-128 is currently the world standard encryption algorithm. It is currently used by all major governments, including the US government. Using the AES cipher with a 128 bit key is not the most secure cipher but comparing the 128 bit key to the 256 bit key, the 128 bit key is still very secure and is much more time and energy efficient. The 256 bit key costs a substantial amount more to implement and is also much slower due to decrypting the key. AES-128 contains little to no risks as long as the key remains secure. AES-128 u

The history behind encryption algorithms is that IBM started developing encryption ciphers to protect its customer’s data in the early 1970’s. This methodology became obsolete in 1997 and that led to the development of the AES algorithm in 2000. The AES algorithm is widely used today.

The purpose of these hash functions is for authentication and encryption. Authentication in this matter is to match the entered password to the password saved in the system whenever a login is executed. As for encryption, whenever data is sent, it’s encrypted using the key, which is determined by using a cryptographic random number generator. The encrypted message is sent to the desired destination and is decrypted using the same key. This practice is referred to as symmetrical keys. Using a 128 bit key would actually take around a billion years to decipher using a super computer. This really shows how secure this method is. The higher the bit level the more secure the cipher is but is also inversely proportional to how efficient the cipher is.

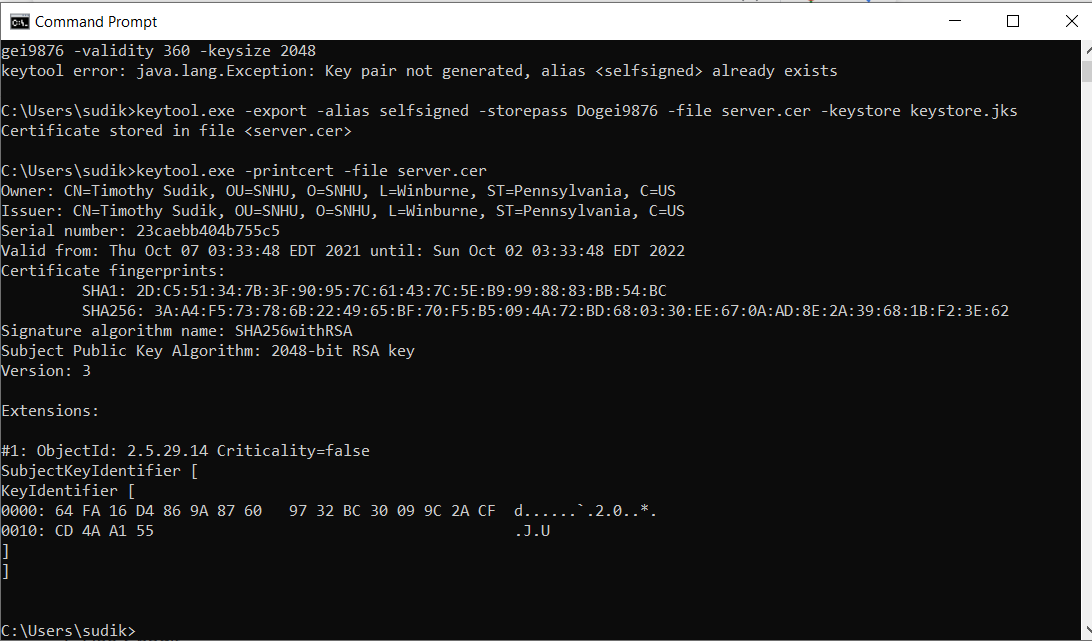
The purpose of these hash functions is for authentication and encryption. Authentication in this matter is to match the entered password to the password saved in the system whenever a login is executed. As for encryption, whenever data is sent, it’s encrypted using the key, which is determined by using a cryptographic random number generator. The encrypted message is sent to the desired destination and is decrypted using the same key. This practice is referred to as symmetrical keys. Using a 128 bit key would actually take around a billion years to decipher using a super computer. This really shows how secure this method is. The higher the bit level the more secure the cipher is but is also inversely proportional to how efficient the cipher is.

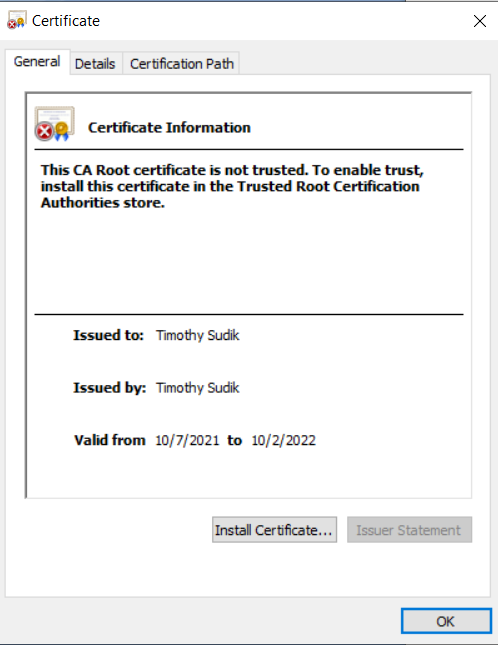
## 2. Certificate Generation

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.

[Insert screenshot(s) here.]

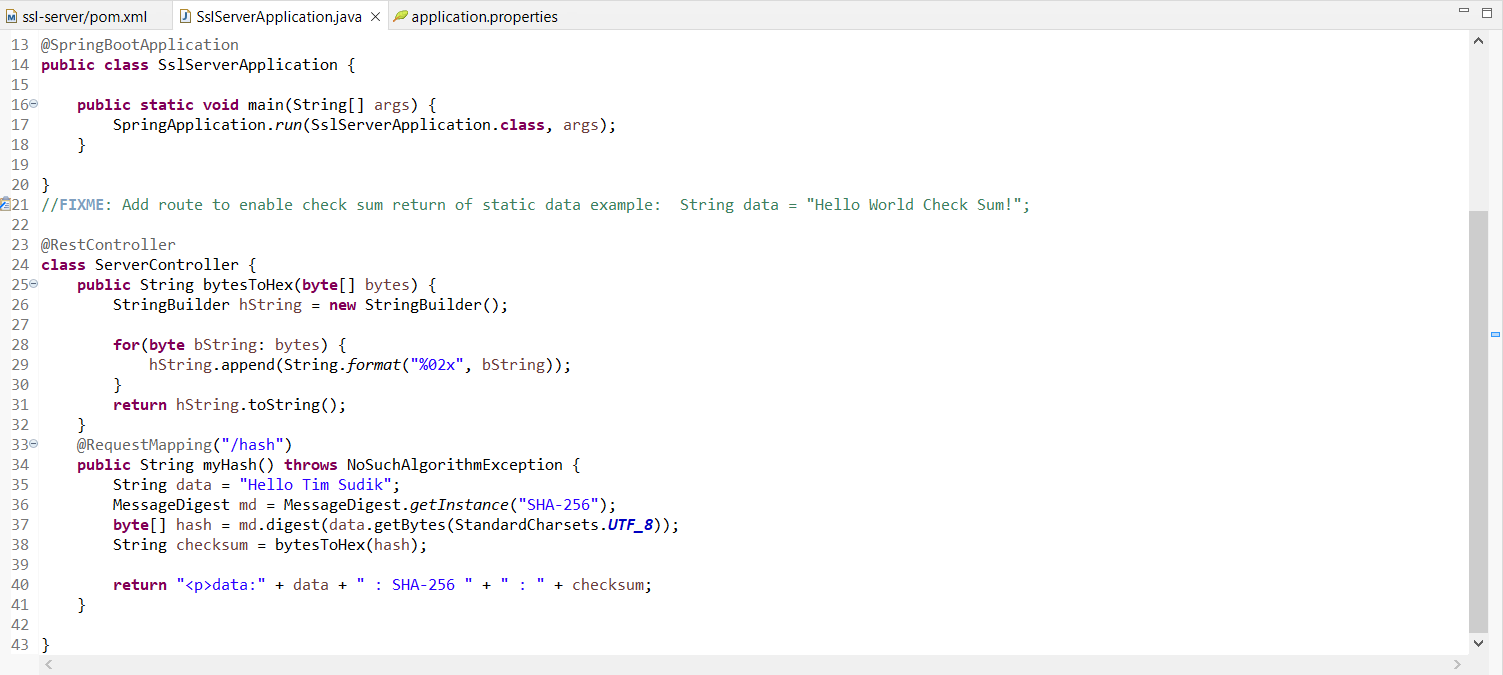




## 3. Deploy Cipher

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.



## 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.

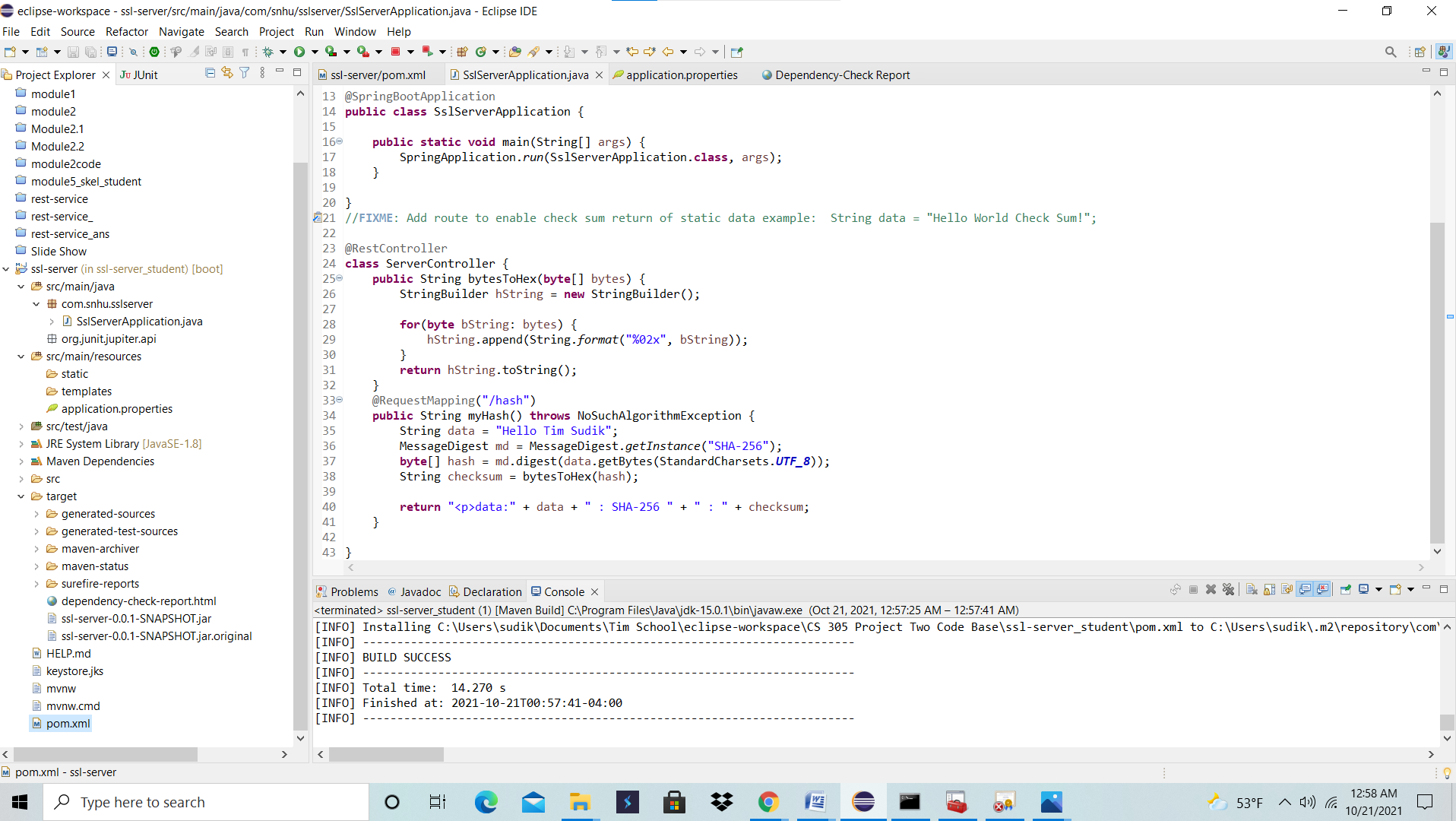
* Insert a screenshot below of the web browser that shows a secure webpage.

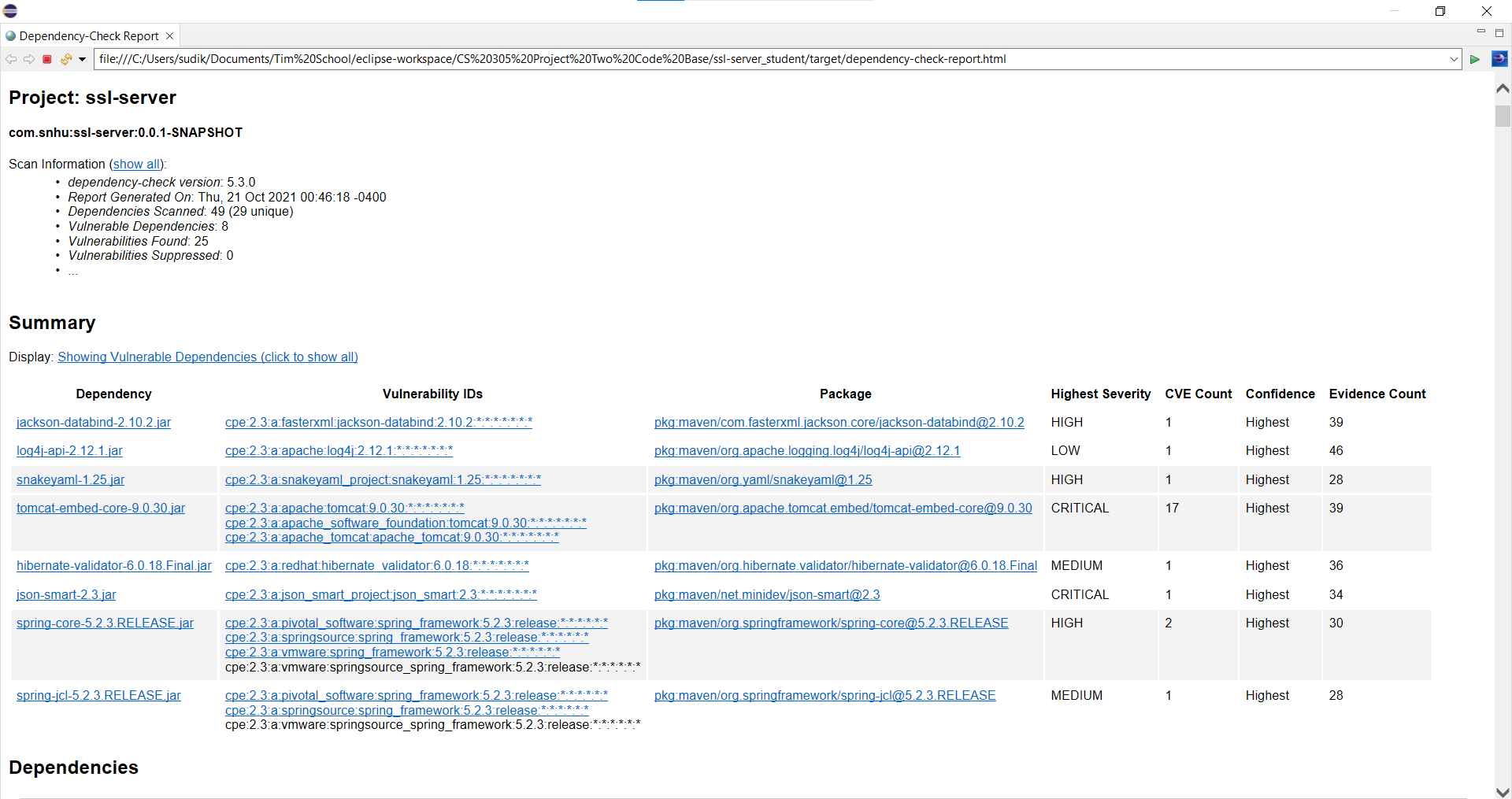
[Insert screenshot(s) here.]

## 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

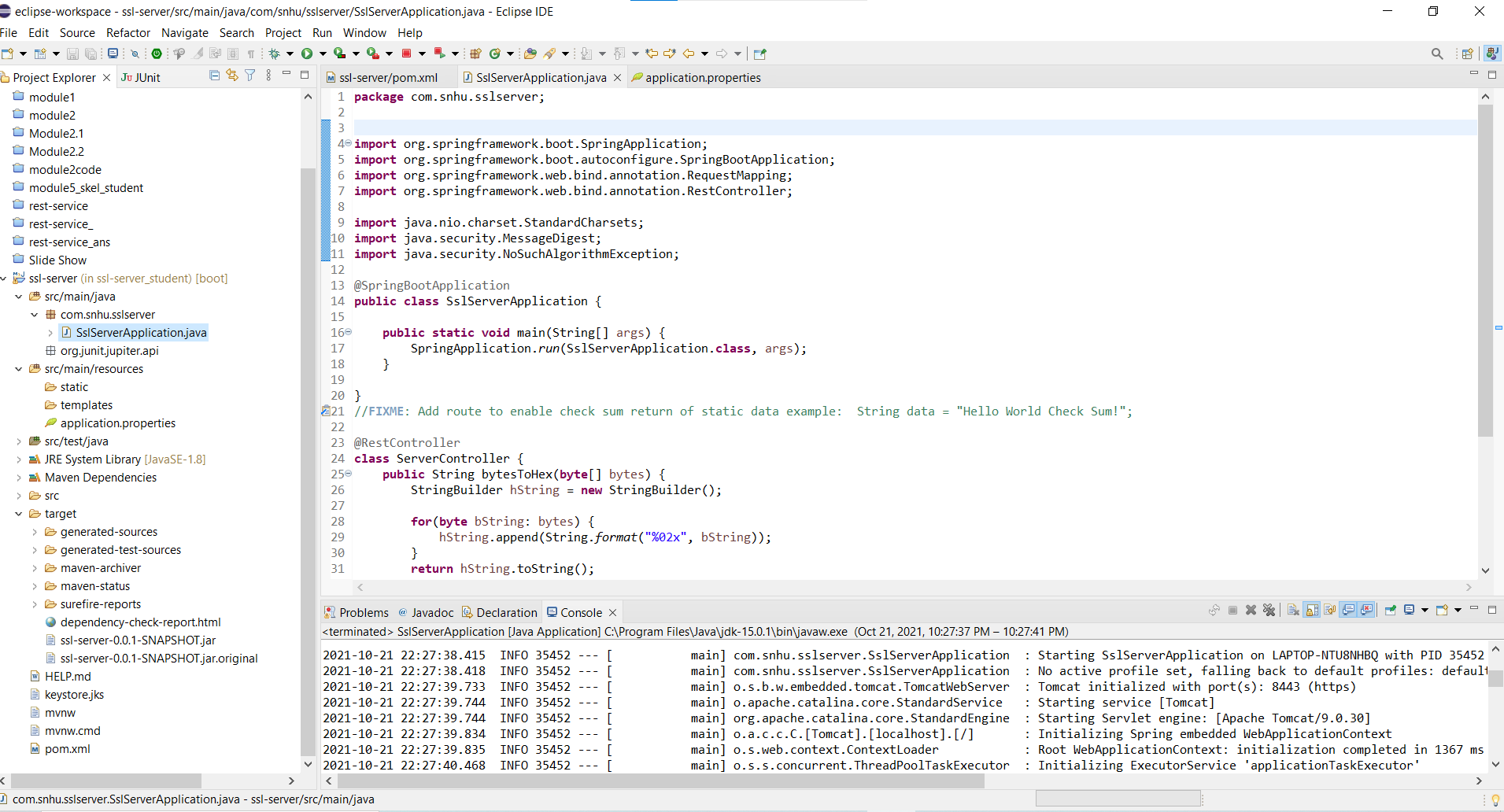




## 6. Functional Testing

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.



## 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.

The areas of security that were addressed were API’s, cryptography, client/server, and code quality. The API area of security dealt with interactions between the user and the system. In order for these transactions to work and be secure , the code needed to be reviewed.

Cryptography and client/server work together in this application. The focus on the client/server area was a secure connection. To ensure a secure connection and reduce cost, a self generating certificate authority was implemented. On the cryptography side, the cipher algorithm AES-128 was used to encrypt messages and data communicated back and forth between the user and the server.

Code quality was emphasized in a manual code review looking for vulnerabilities. Inspecting the code for any vulnerabilities or possibilities for errors was completed and no major vulnerabilities were found.