**Module Two Static Testing Assignment**

Jason Verrill

Southern New Hampshire University

CS 305: Software Security

Dr. Lyon Vivian

7-2 Project Two Practices for Secure Software Report

April 17, 2022



# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History4](#_Toc33111302)

[Client4](#_Toc33111303)

[Instructions4](#_Toc33111304)

[Developer5](#_Toc33111305)

[1. Algorithm Cipher5](#_Toc33111306)

[2. Certificate Generation5](#_Toc33111307)

[3. Deploy Cipher8](#_Toc33111308)

[4. Secure Communications8](#_Toc33111309)

[5. Secondary Testing8](#_Toc33111310)

[6. Functional Testing10](#_Toc33111311)

[7. Summary11](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **04/17/2022** | **Jason Verrill** |  |

## 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

Jason Verrill

## 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.

Artemis Financial needs to securely transmit sensitive files over the internet. For Artemis Financial's scenario, I recommend using the AES-256-CBC encryption algorithm cipher combined with RSA for asymmetric key use, and SHA-256 for hash values and checksum. AES is an industry standard and a highly secure encryption algorithm useful for encrypting large amounts of data. AES uses up to a 256 bit symmetric key than can be used to both encrypt and decrypt the data with the same key (Manico & Detlefsen, 2015). While RSA provides a higher level of encryption, it is not practical for the file transmission due to the size of the sensitive data (Manico & Detlefsen, 2015). Therefore, RSA can be used for its asymmetric keys for public use which is necessary for a secure key, while using AES for file transmission.

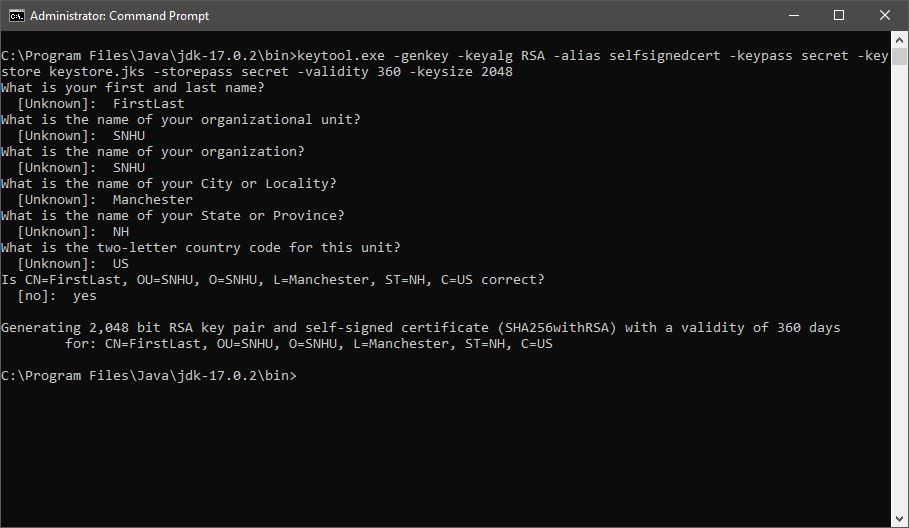
AES is widely used in modern day for cryptography. Other algorithms exist, some less applicable to Artemis's situation, while others have been cracked over time (Manico & Detlefsen, 2015). AES is a proven and up to date encryption method. In essence, AES (and other types of encryption) takes plain text and hashes it using an algorithm made from the plain text, a random seed number (usually something particularly difficult for computers to reverse such as a large prime number), and series of multiplications to make it unintelligible to the onlooker (Manico & Detlefsen, 2015). Through the use of AES keys, that data can be translated back to the original. Irreversible hashing does exist, but in Artemis Financial's case, the data that is encrypted needs to be viewable in the future, even if it is not frequently. To explain keys in more detail, a symmetric key uses a single key for both encrypting and decrypting data. Asymmetric keys offer a separate public key that can be given out that allows the user to encrypt data and send it to the original source. The decryption key is kept secret at the source so only it knows how to access the secret data, and eavesdroppers will not be able to understand any intercepted data. In this way, data can be securely transferred over the internet without worry.

## 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.

**Certificate Prompts**



**Certificate Printout**

## 

## Certificate Export

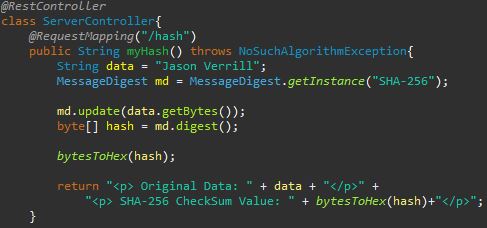
## 

## 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.

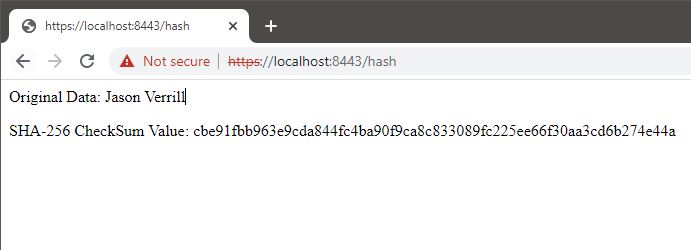
**Checksum Code**



## 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.



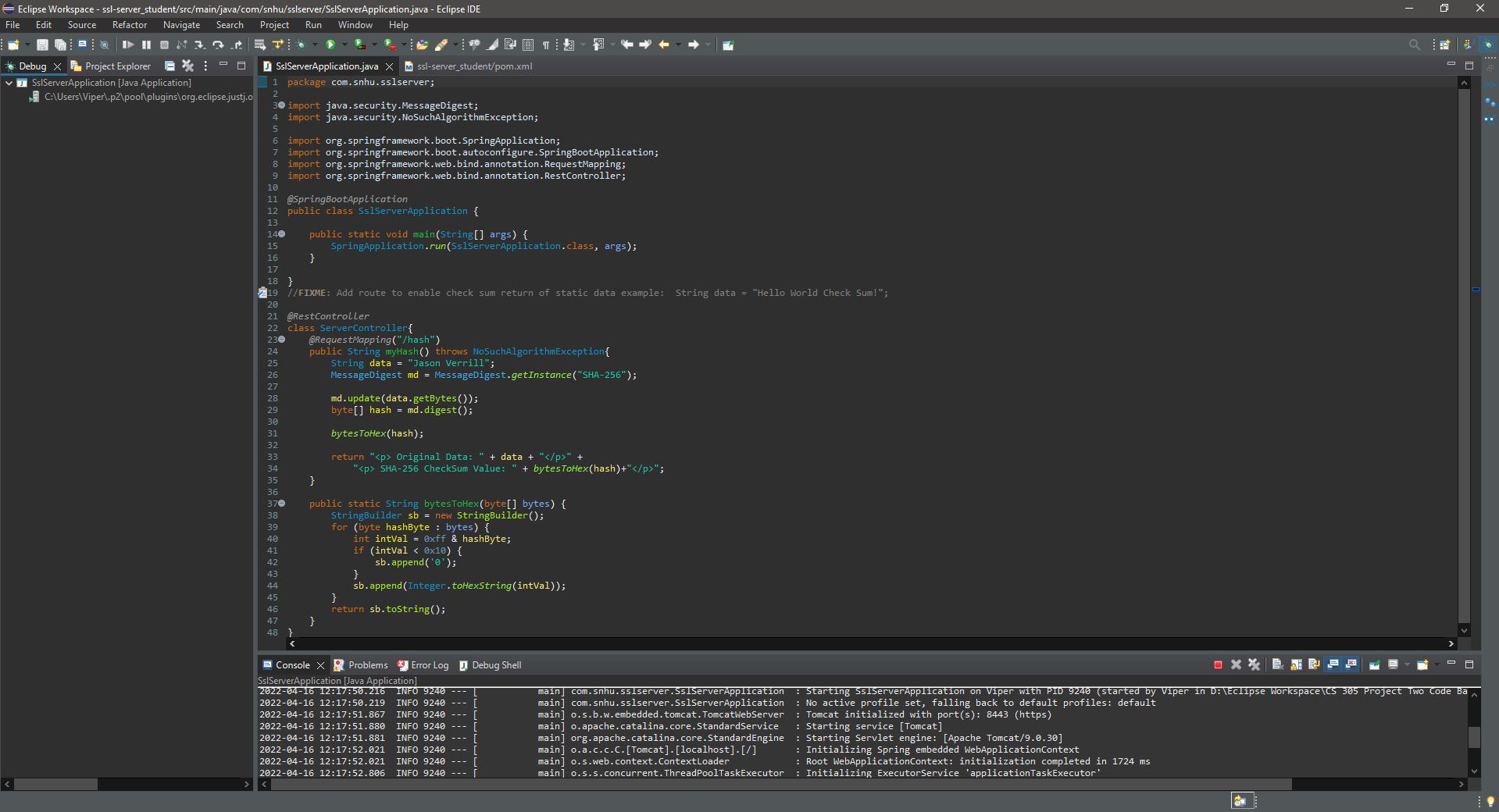
**Checksum Webpage**

## 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**

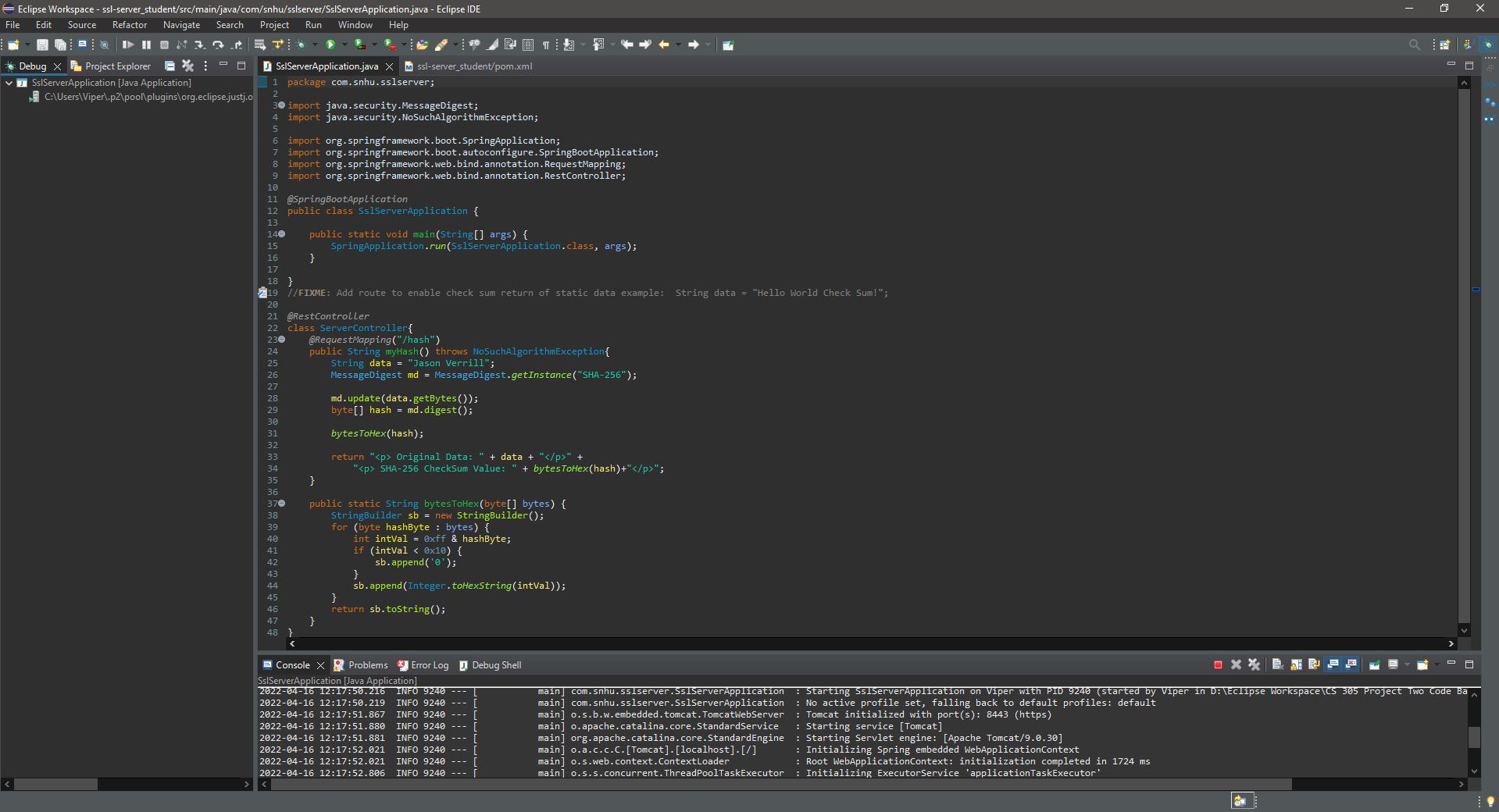


## Dependency Report

## 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.



**Refactored Code**

Incoming input from untrusted sources should be verified to ensure it complies with expectations. For example, future input will need to be compared against the checksum to make sure the files were not tampered with. In addition, file access should be restricted to least permissions. If the file does not need to be accessed by everyone, any source that attempts to download the file should be authenticated and have valid authorization beforehand.

## 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 that were addressed by the refactored code listed on the Vulnerability Assessment Process Flow are APIs, input validation, cryptography, code error, and code quality. The web page offers an API in the form of /hash, which when called, returns a checksum value. The checksum can be used for input verification or for file verification. The hash makes use of cryptography to keep the customer's data secure and prevent hackers from causing damage to the company's client's computers, client base, and business reputation (Manico & Detlefsen, 2015). The use of a checksum also prevents code error if the data was corrupted in transit. Code quality is a superset of the other areas of security.

**References**

Manico, J., & Detlefsen, A. (2015). *Iron-clad Java*. O'Reilly Online Learning. Retrieved March 27, 2022, from https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/ ch06.html