****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[\_Toc33111301](#_Toc33111301)

[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** | **06/14/2022** | **Briana Carlson** |  |

## 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

[insert name here]

## 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.

A secure method would be AES-256 and SHA-256. The Advanced Encryption Standard (AES) is a symmetric block cipher that is the strongest encryption standard around, it has become the industry standard for data security. It is used by the U.S. government to protect data. Given the total number of key combinations and the current state of computer processing it would take billions of years to crack. AES uses symmetric keys either 128, 192, or 256 bit, by using 256 we can ensure security for many years to come. Each bit level uses 128-bit blocks, the difference is in the length of the key. Hash functions take sets of inputs of any size and fits them into a data structure with fixed size elements. Using a hash function such as SHA-256 with 2^256 combinations making collisions or successful attacks unlikely.

With symmetric and asymmetric encryption plaintext is converted into ciphertext during encryption, the main difference is that asymmetric will use two unique key pairs to encrypt. Symmetric uses the same key to encrypt and decrypt information. Asymmetric uses two different keys, public and private keys, to encrypt and decrypt sender’s and receiver’s data. Asymmetric is considered more secure while symmetric is faster. Random numbers are secure due to their unpredictability, with a randomly generated number there are no clues or patterns to consider that could make open opportunities for attacks.

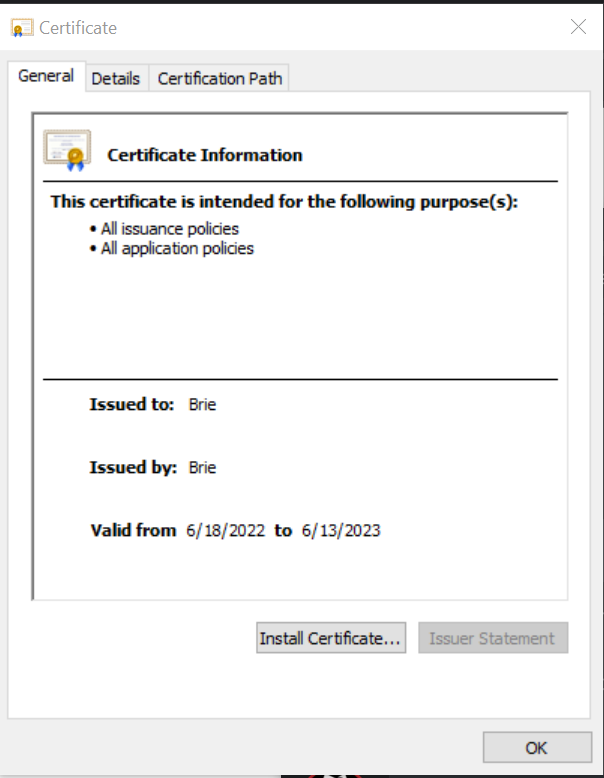
Until WWII cryptography was for military purposes until it gained commercial attention from businesses trying to secure data from competitors. In the 1970’s IBM customers started to demand some form of encryption, so a “crypto group” was formed where they designed a cipher called Lucifer which was accepted and called DES or Data Encryption Standard. DES was a small encryption key so as computing power advanced, brute force attacks became easy. In 2000 AES became the new standard which has evolved over time with different bit options for better security as computing power progressed further.

## 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.

Attachment for certificate generated below.

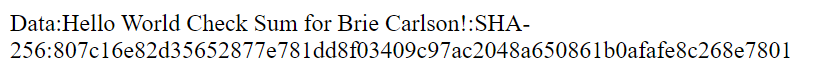


## 3. Deploy Cipher

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.

Attachment for checksum verification below.

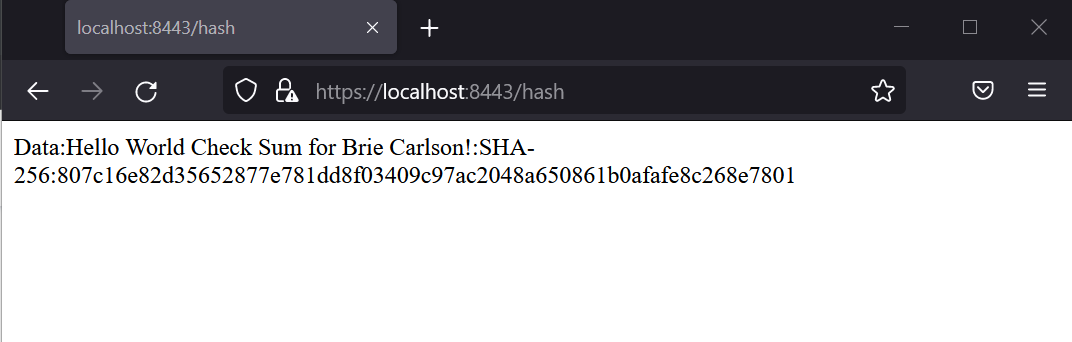


## 4. Secure Communications

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.

Attachment for https below.

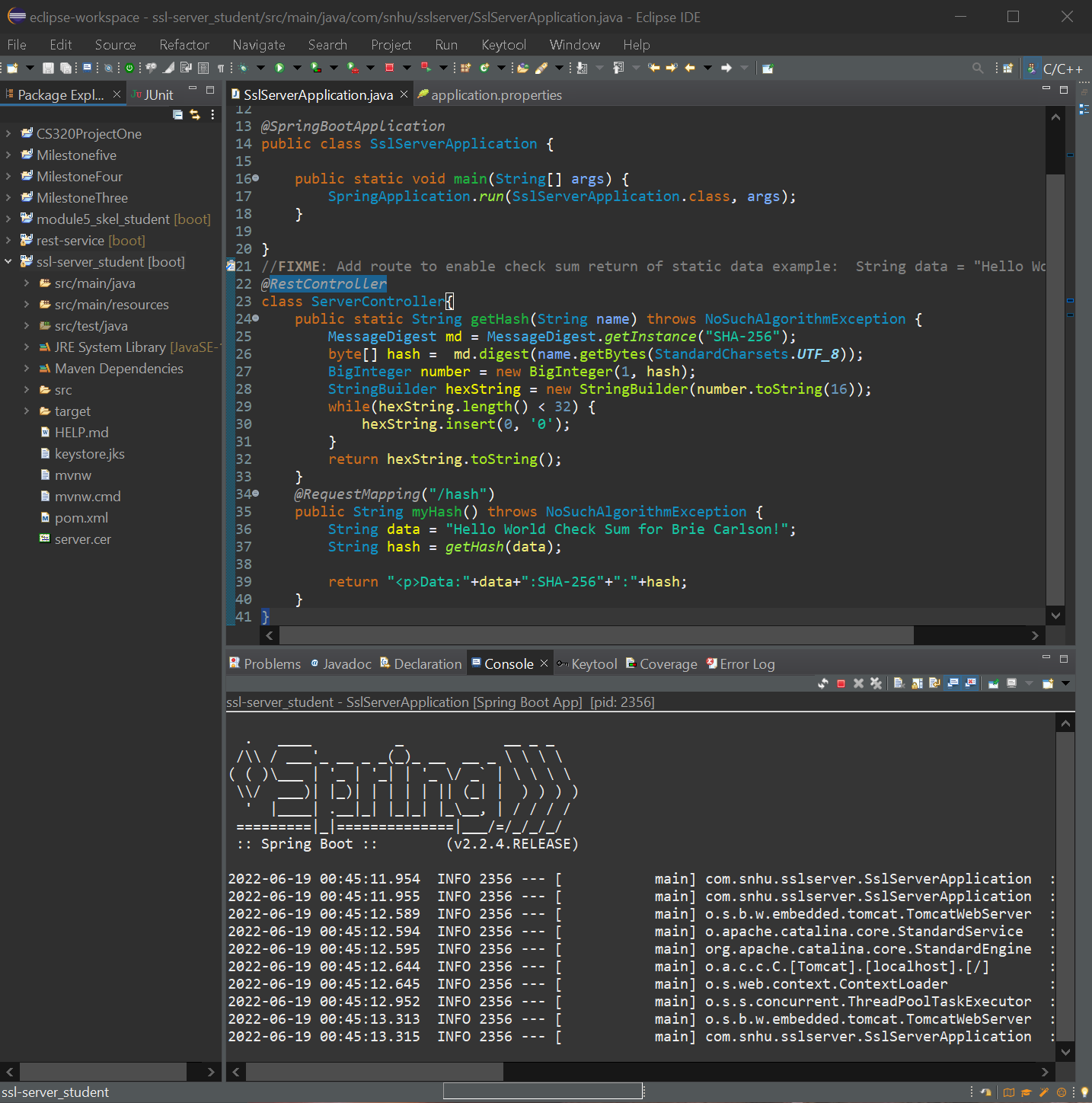


## 5. Secondary Testing

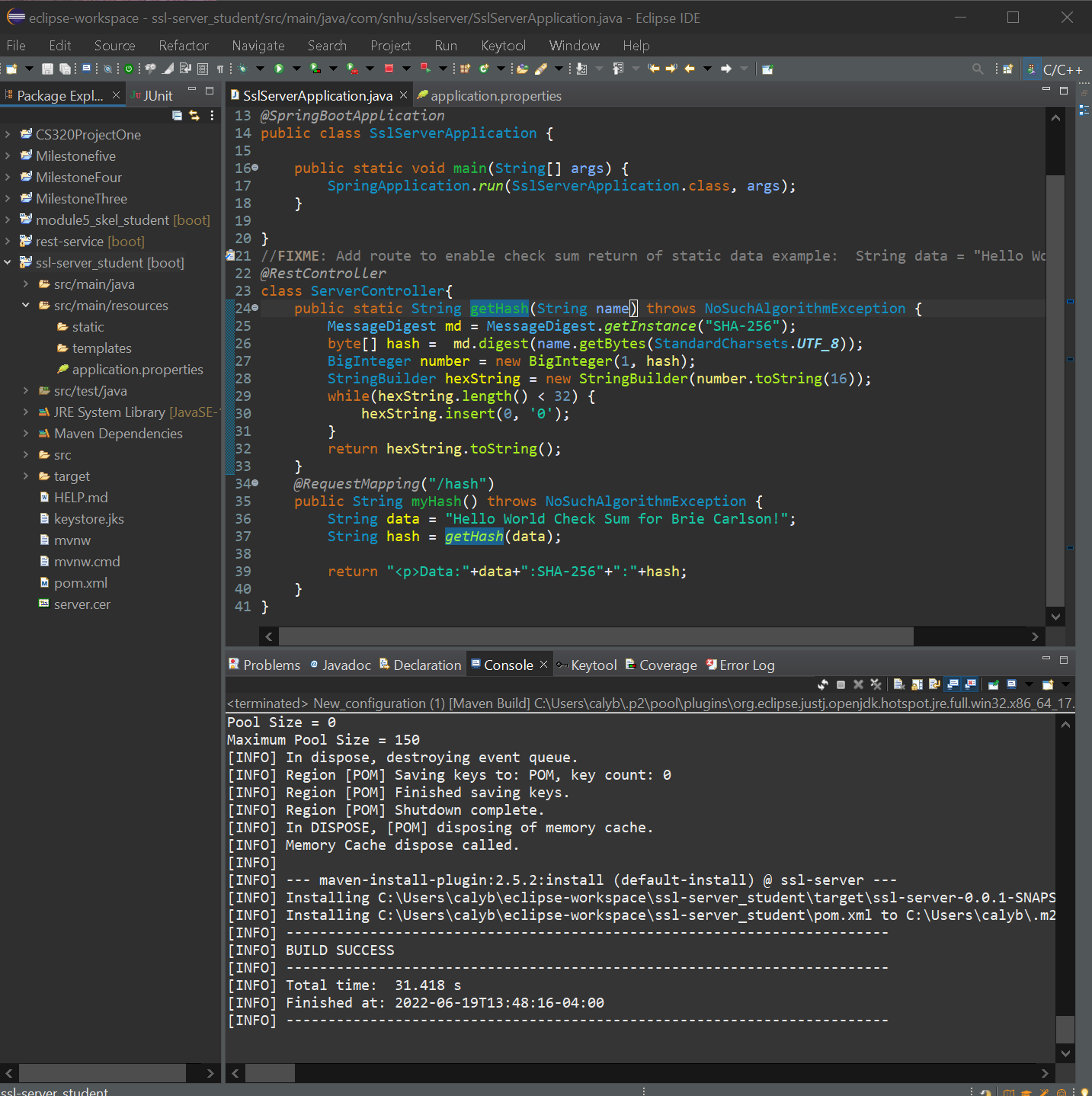
Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

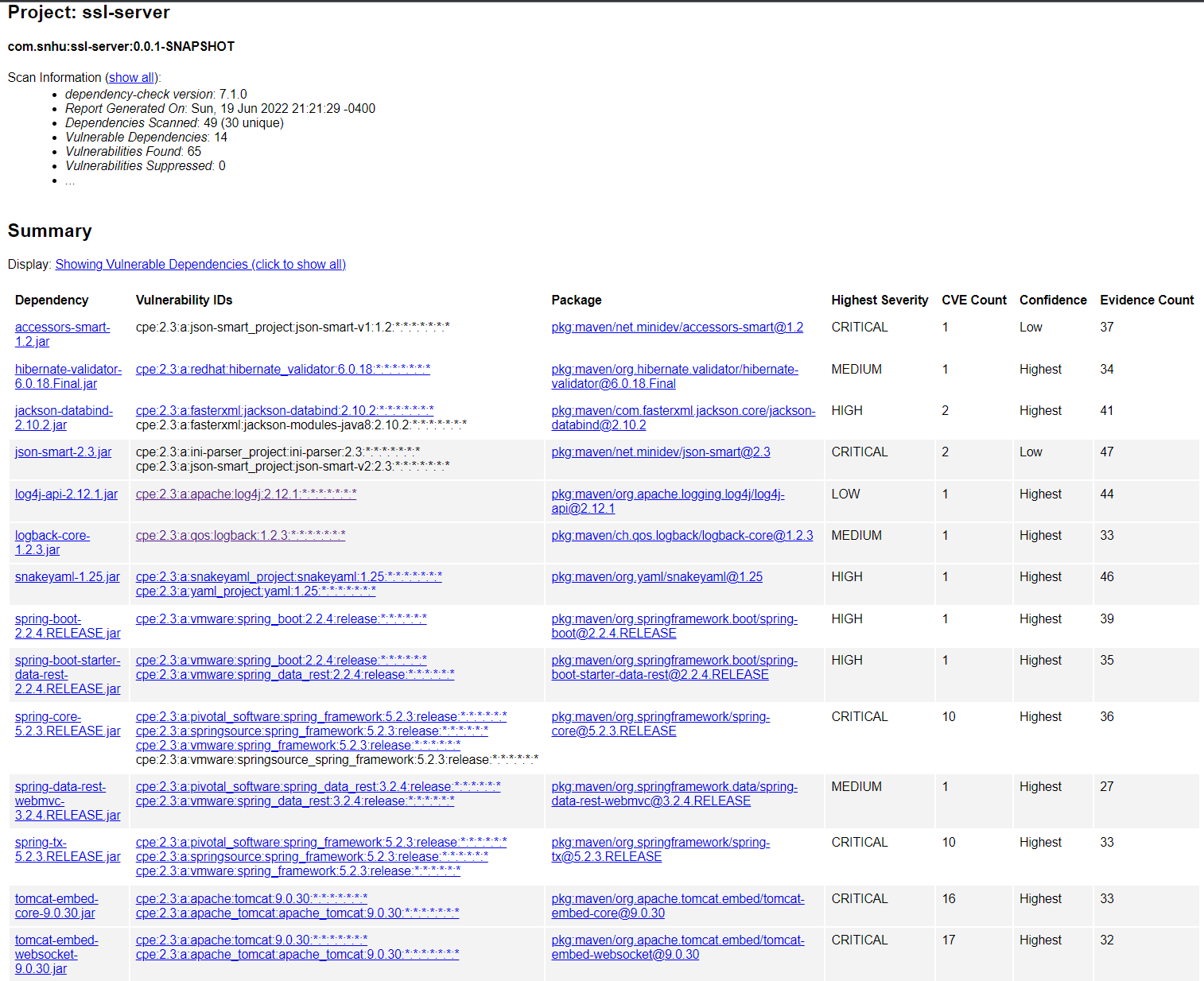
Attachment for code running spring boot below.



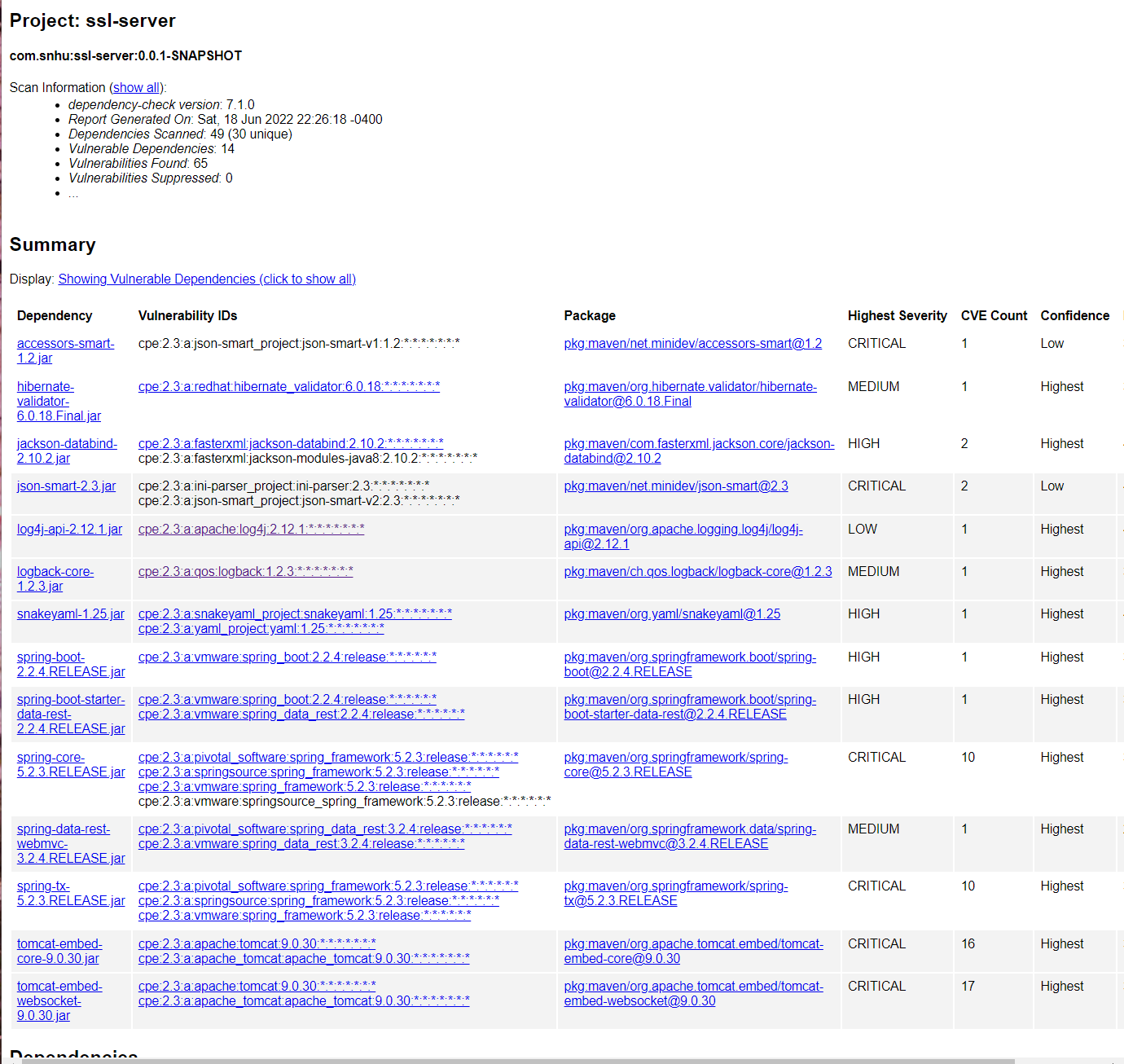
Attachment for code running in maven build below.



Attachment for dependency check before refactoring below.



Attachment for dependency check after refactoring below.

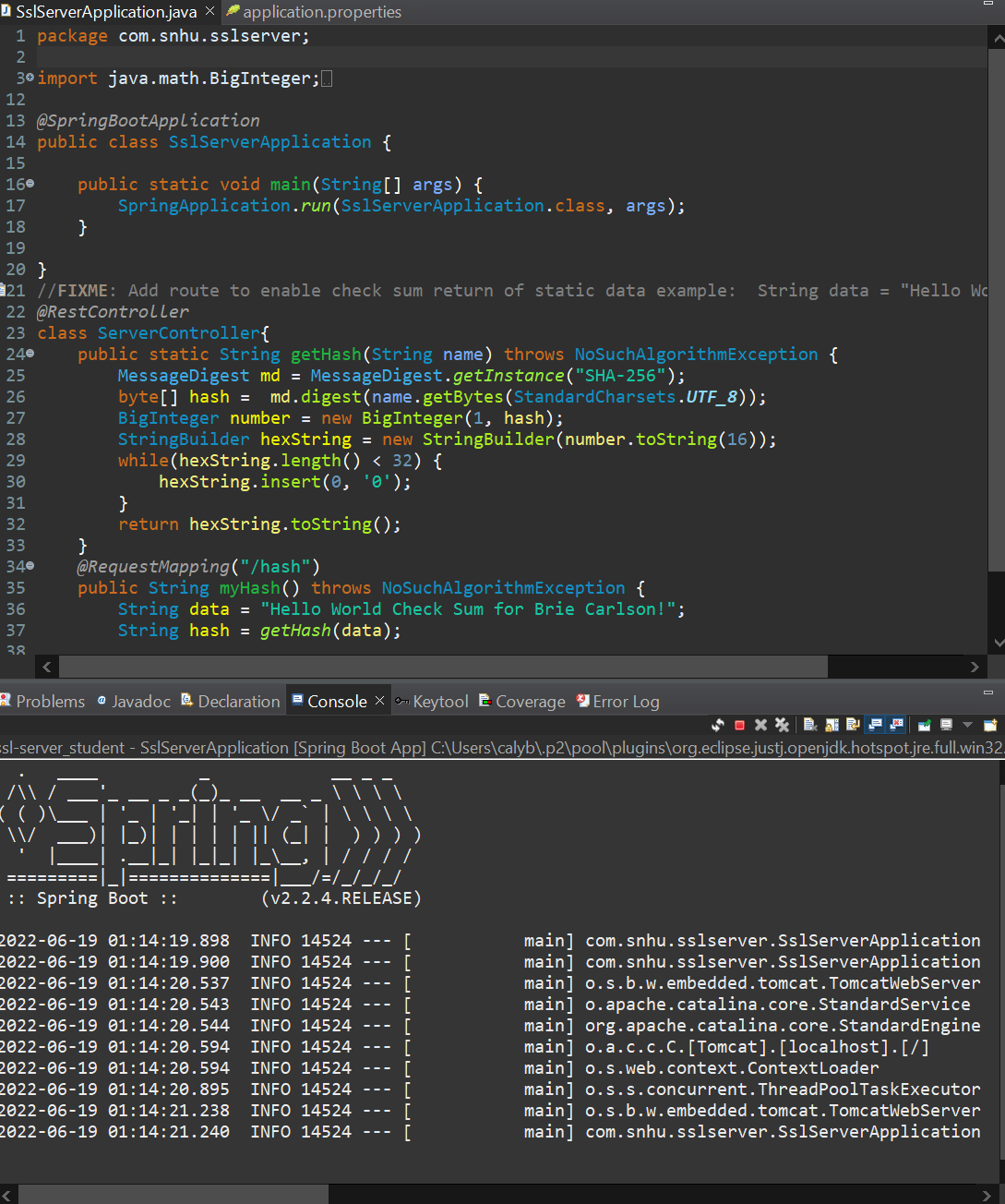


## 6. Functional Testing

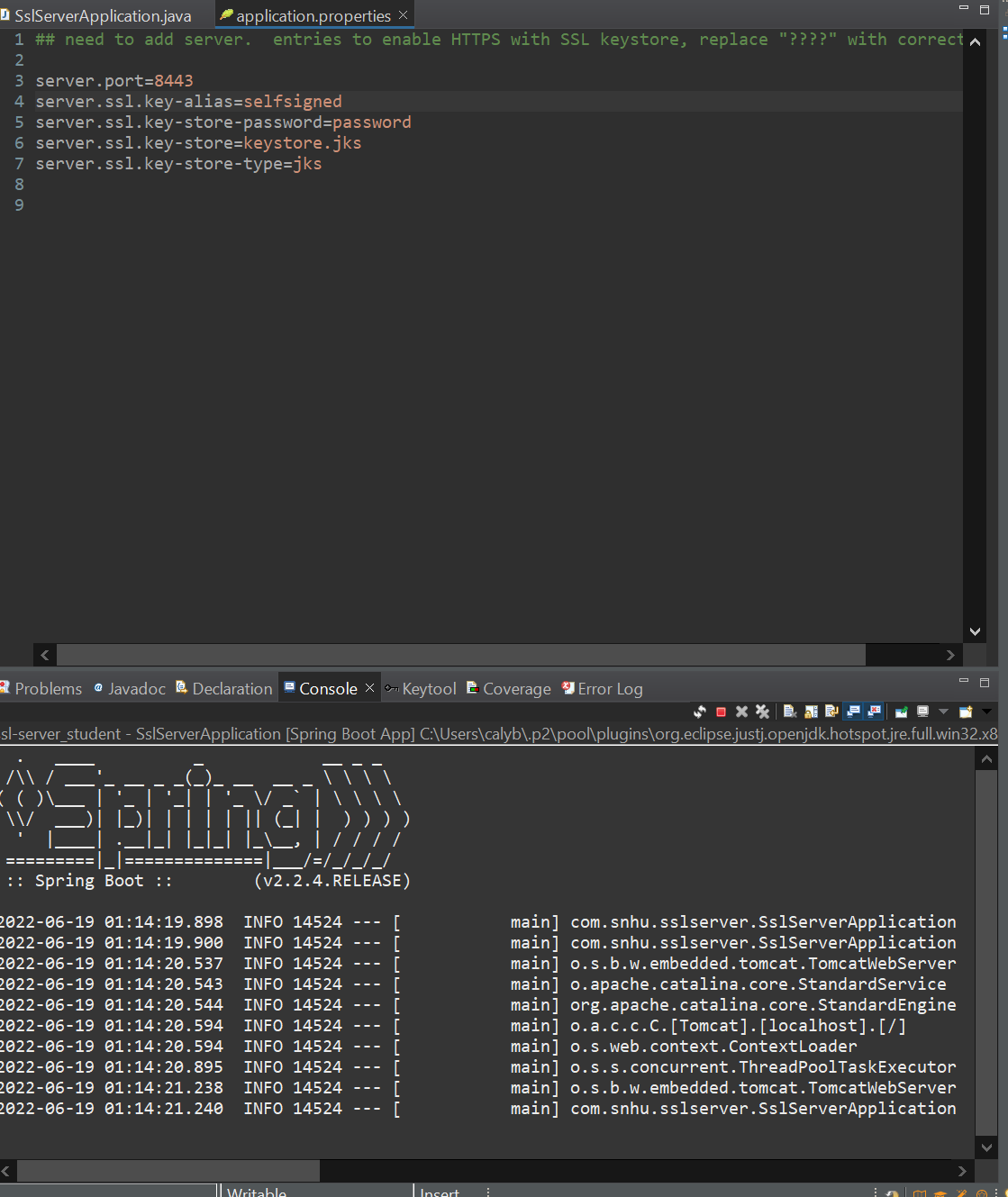
Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.

Attachment for code running without errors below.



Additional attachment to show application properties while running without errors below.



## 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.

In my refactoring I have created a restful application including a hash function SHA-256 to ensure better security. By adding a certificate, client/server security is improved and can transfer information in a more secure way. I used NoSuchAlgorithmException to be thrown in the event that a cryptographic algorithm is not available. To maintain better security, we should use the most recent versions such as sprint boot or tomcat. We should be running routine dependency checks and keeping all dependencies up to date. Using an SSL connection should keep the communications between clients and Artemis Financial more secure. To prevent future issues from attackers, measures could be implemented to restrict the number of requests from one IP by slowing repeated requests. Continued maintenance on our code will ensure the best coding practices remain in place for years to come to give us an extra layer of security.