****

# Practices for Secure Software Report

**Lane Berrevoets**

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 5](#_Toc102040759)

[3. Deploy Cipher 6](#_Toc102040760)

[4. Secure Communications 7](#_Toc102040761)

[5. Secondary Testing 7](#_Toc102040762)

[6. Functional Testing 9](#_Toc102040763)

[7. Summary 10](#_Toc102040764)

[8. Industry Standard Best Practices 10](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/9/2023** | **Lane Berrevoets** |  |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Lane Berrevoets

## Algorithm Cipher

*Note to the Instructor: I utilized the Module 4 and Module 5 written assignments to assist in the writing of the following. I have re-cited the same sources as before. Just want to let you know that it may look similar in the Turnitin assessment (due to re-use of my prior work).*

Due to the requirements requested by Artemis Financial, Global Rain recommends the use of a checksum. A checksum utilizes the idea of hashing, or compiling user data into various other characters that are typically much shorter than the original piece of data, to verify the integrity of the data. The standard is SHA-256 or Secure Hashing Algorithm 256, which is 256 bits long meaning that it is “computationally infeasible to reverse engineer the original input from the hash value” (Griskenas, 2023). Each bit level will scramble the data a specific number of times, for instance, 256 -bit will rearrange the data 64 times, within each “round” it will shift data block segments around and will add constants to the data (Griskenas, 2023).

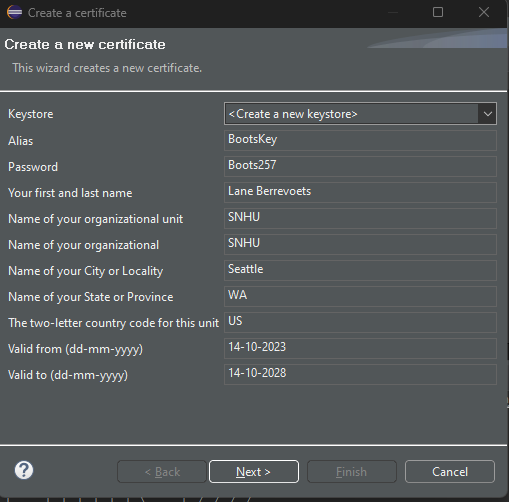
Hashing, as stated above, is typically utilized during data integrity verification. Hashes work due to the idea that each hash is specific to one set of data and only that set. That said, hashing collisions may exist, but are a rarity. Hashing collisions occur when separate inputs create identical hashes. If more than one hash exists for a piece of data, attackers can utilize that information to alter, mimic, or access the data.

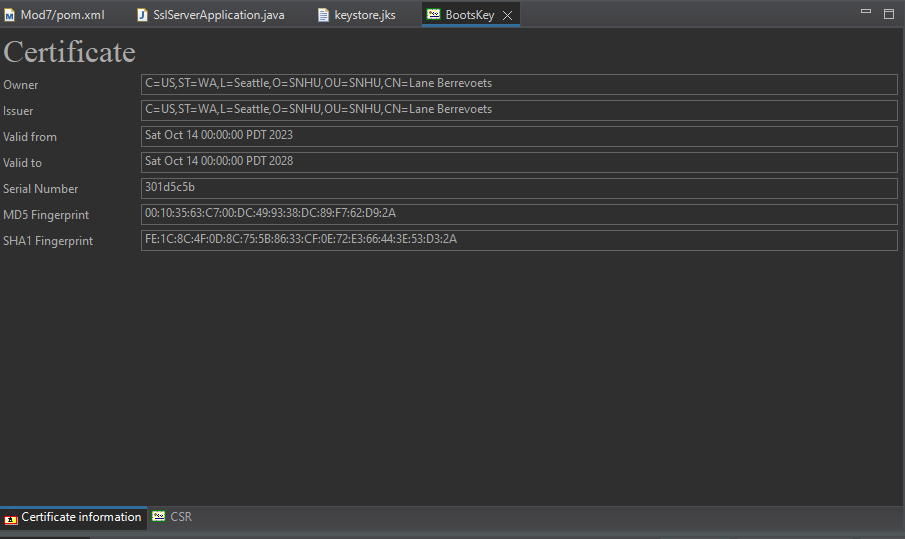
Asymmetric encryption uses two separate key values, one to encrypt the data and another key, which relates to the first, to decrypt that data. Symmetric encryption utilizes a single key to both encrypt and decrypt a set of data. Every byte is comprised of 8-bit. Using the 128-bit key as an example, there are 16 bytes within the key. Using a specific key schedule along with our “key”, we combine our data with the key, using binary, and shift the columns and rows within our “table”. This is referred to as a cycle or round (Rimkiene, 2022). The use of random number insertions typically occurs during each “round” and is specific to the cipher utilized. This insertion further protects the data due to it being specific to the cipher and being random due to the timing schedule of the cipher.

Encryption algorithms are only as good as the strength of the computer, meaning that as computing technology advances, so too will the encryption standards. As soon as quantum computing becomes commonplace, those machines will have the power to brute force break most of today’s algorithms.

## Certificate Generation

Insert a screenshot below of the CER file.

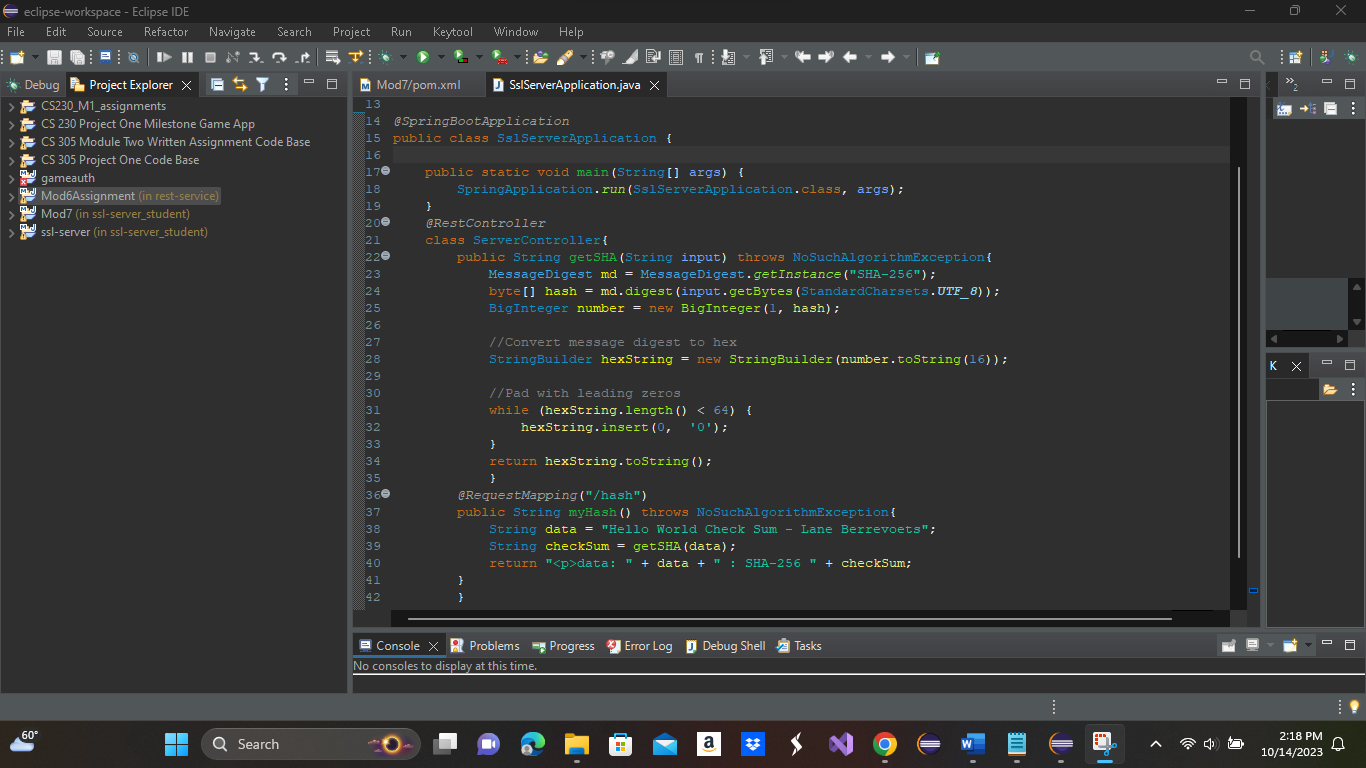




Note: I downloaded and installed Java Keytool prior to this. I ran into multiple issues prior to finally getting this to work. The “hints” definitely helped. I could not figure out how to save the created key within the program, yet all I had to do was create a new file.

## Deploy Cipher

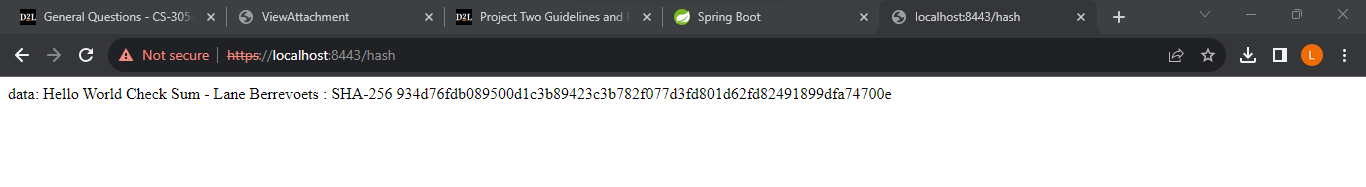
Insert a screenshot below of the checksum verification.



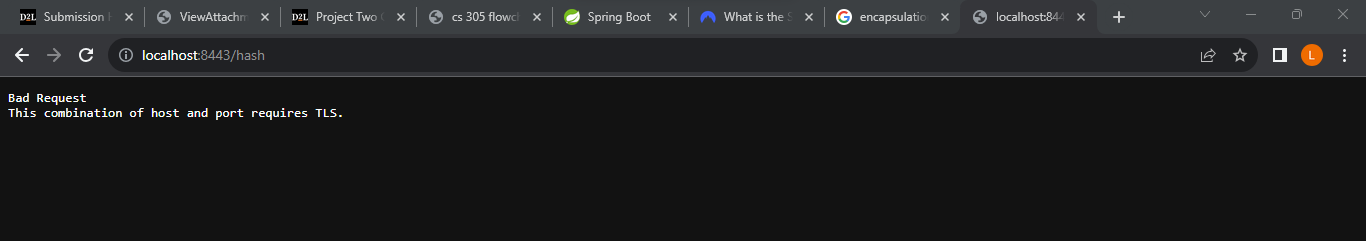
Note: This is the same checksum used during one of the previous modules.

## Secure Communications

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

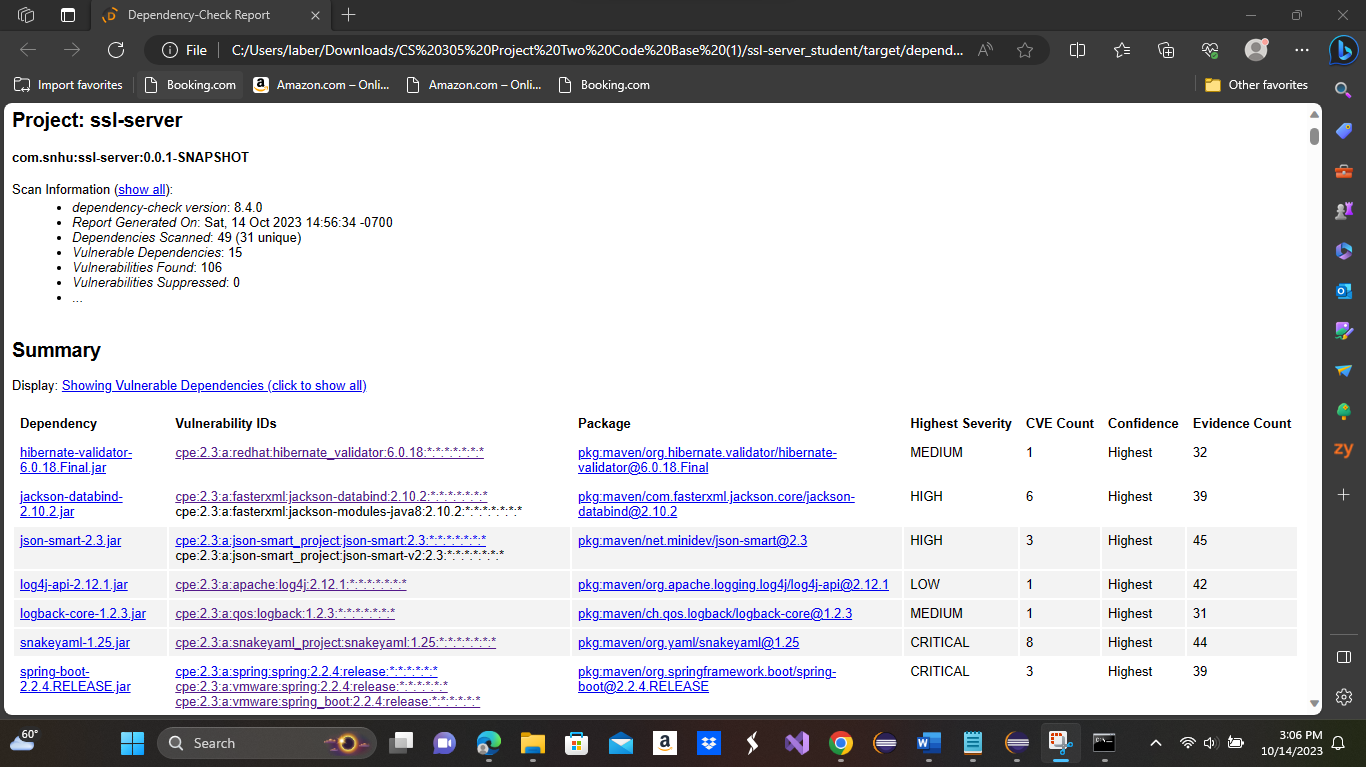


If, instead, the user utilizes an HTTP request, they are given the following:



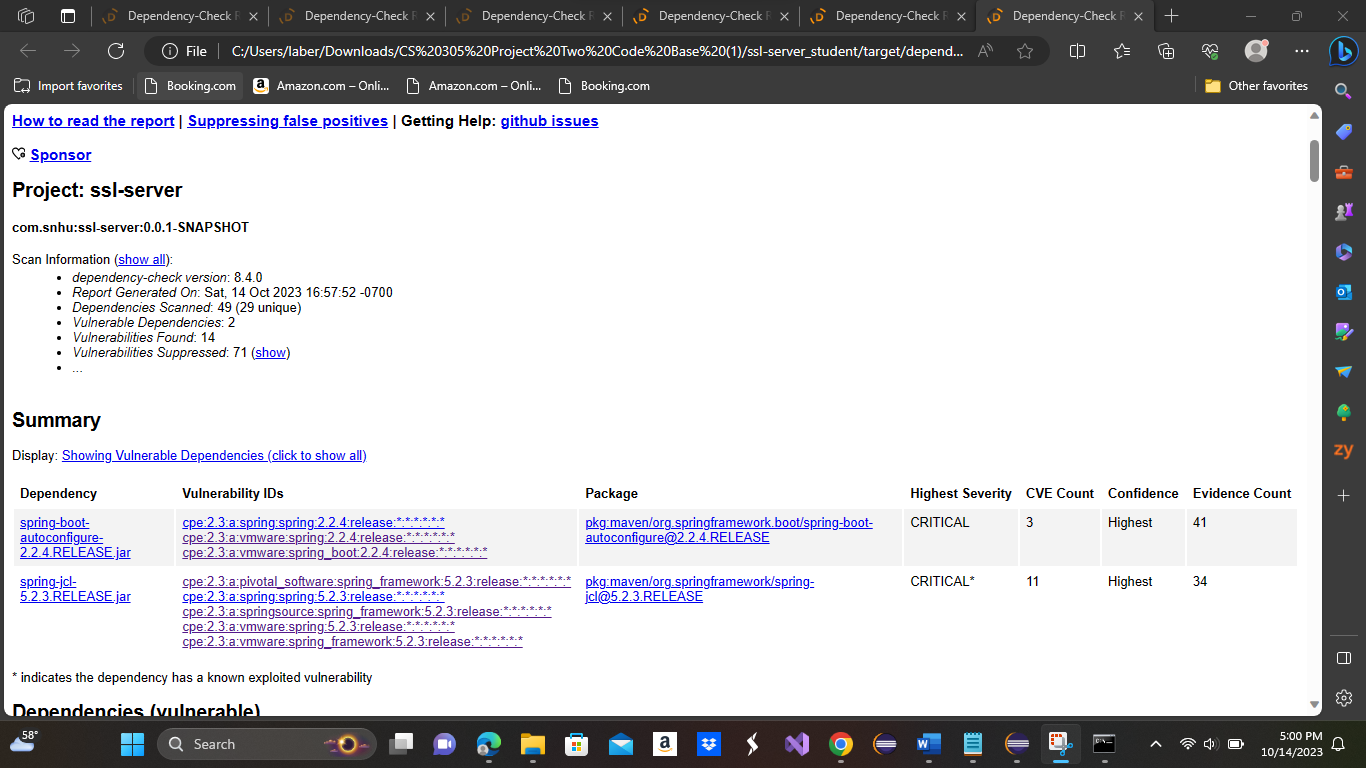
## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

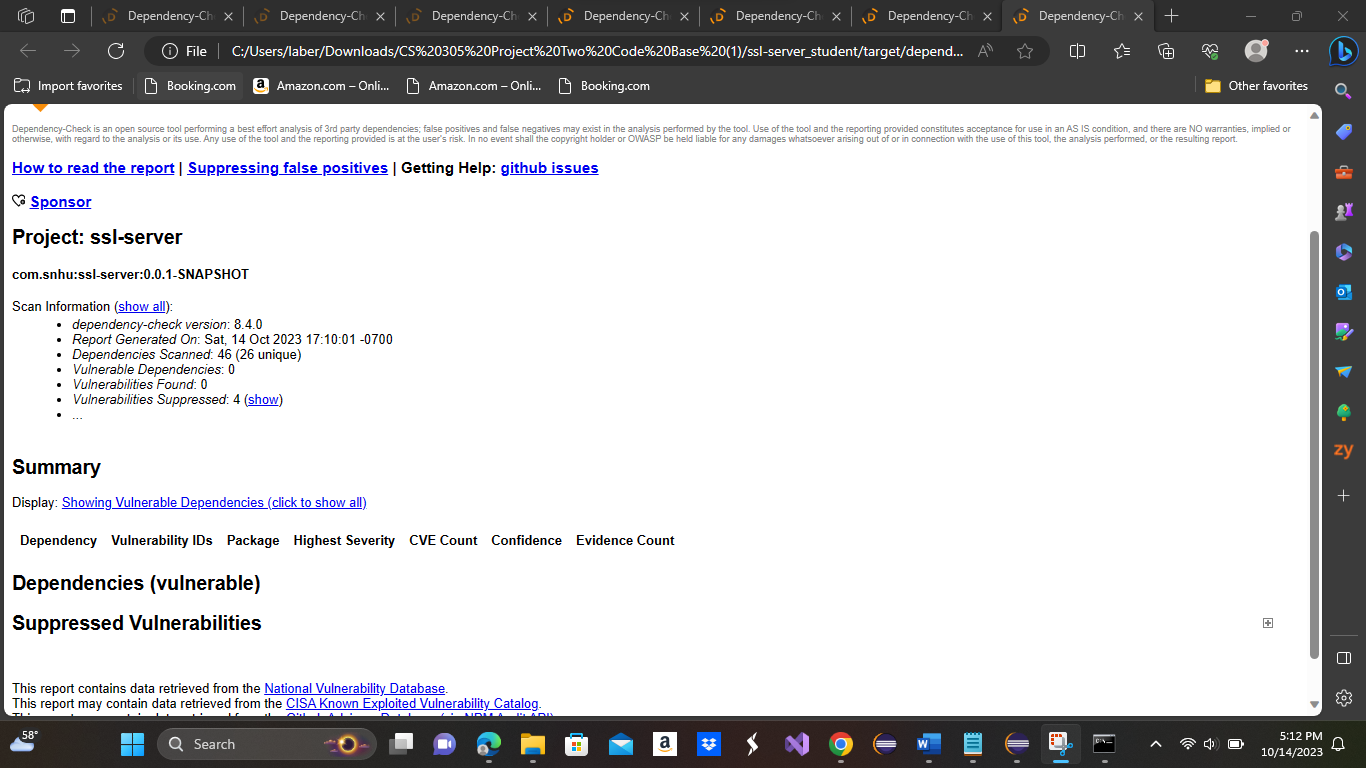


Above is the original dependency check performed following the creation of the checksum.

Following multiple attempts at suppression, I was left with:



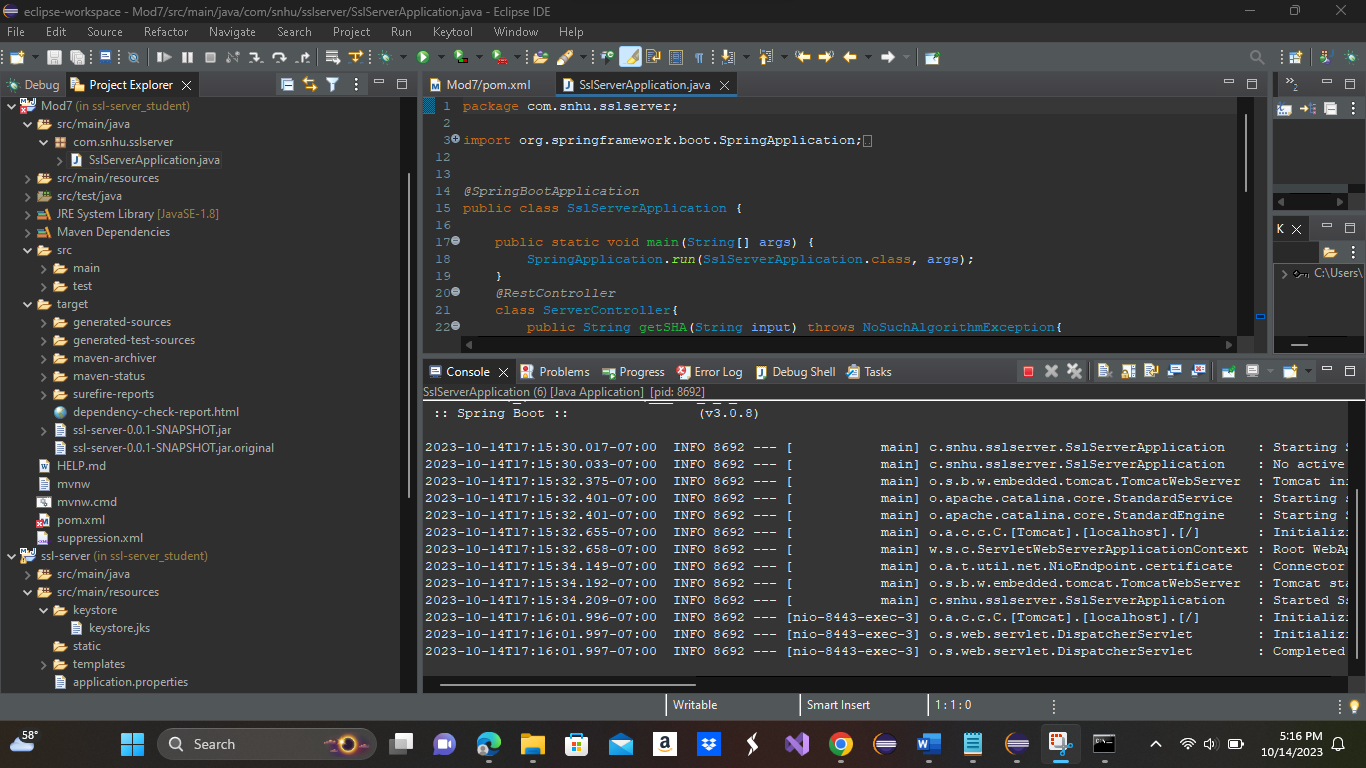
I believe this is referring to the same issue that another student was having given in the “hints”. Next, I will attempt to update Spring Boot as said within the “hints”.

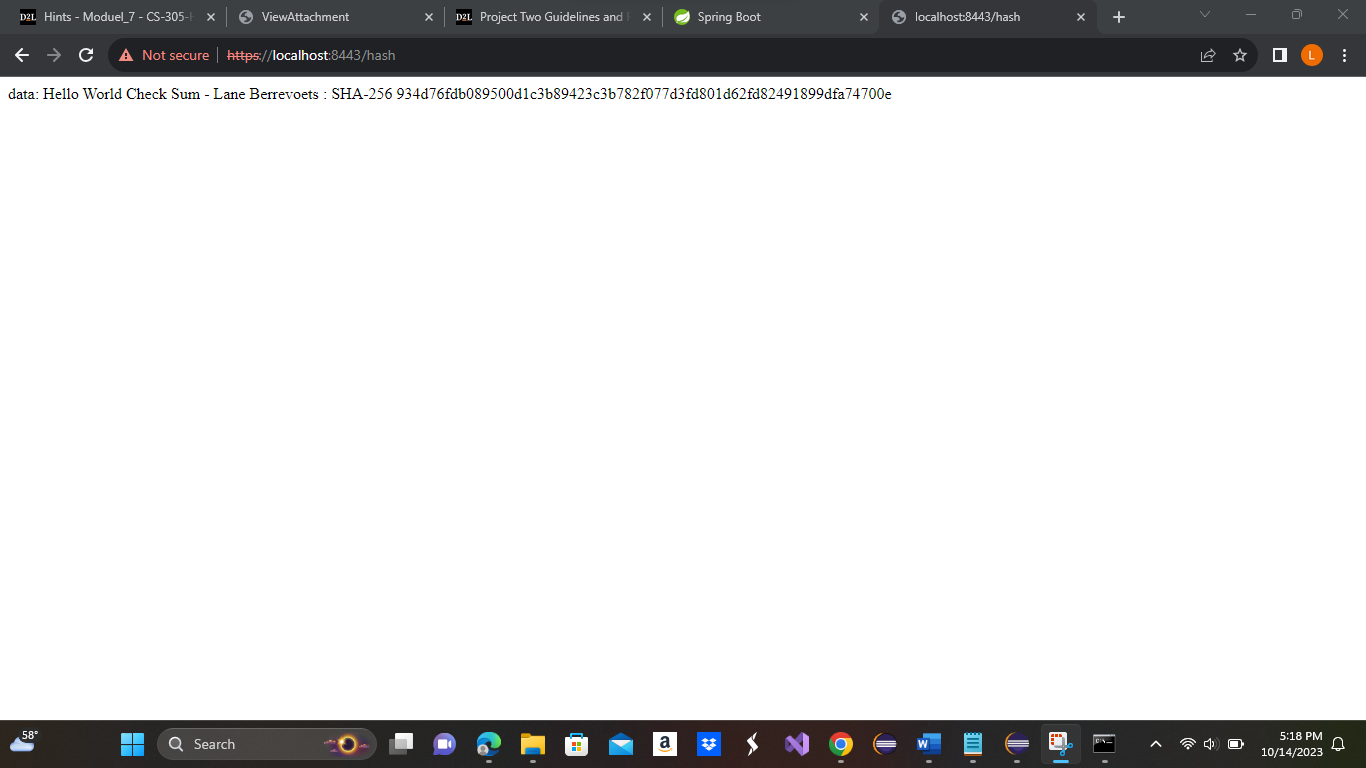


Note: Following the “hints” I updated Spring Framework to version 3.0.8. This removed all additional listed dependencies that I was not able to remove otherwise.

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.





Above is the result following running the code after refactoring both the pom file and the code itself.

## Summary

The program has been improved upon in the following areas of security: Cryptography and Code Error/Quality. First, we added a certificate to prevent attackers from gaining immediate access to our code. That said, the certificate is a self-signed certificate and should be updated prior to dissemination of the program as an added security feature without the ability to be self-signed. Next, the code was refactored to include the checksum, improving the area of Cryptography. This was achieved by implementing a MessageDigest class that hashed the data using SHA-256 as the cipher algorithm. Following, we converted the MessageDigest to a hexadecimal string, which allows us to view the hash as a string and allows the performance of the validity check to occur. By updating the application.properties, we ensured that the program would only run on an HTTPS connection, preventing use of the unsecured HTTP. By implementing a secondary test through use of Maven dependency check, we had insight into any known errors and were able to update needed programs to remove dependencies. The only program that needed update was the Spring Framework, which was updated to version 3.0.8 to remove its associated dependencies.

By performing a manual review of code, and checking the code console during code runs, we checked the code for both quality and errors. Errors were removed from the code to ensure proper operation of the code. The quality of the code was also checked to ensure that the program is easily readable, and updatable if future updates need to take place.

## Industry Standard Best Practices

Industry standard best practices were utilized throughout the program creation. From the moment of planning through completion best practices were employed to ensure that the program created is secure. Utilizing a hashing algorithm, like the checksum utilized in our program, verifies the integrity of the data. We use a hashing algorithm to provide an assurance to the user that the data or files they are expecting are what they are getting. Performing additional testing and performing dependency checks within the program itself, we are able to view and update the program’s dependent plugins to a version in which known errors do not exist. Refactoring the code utilizing Object Oriented Programming principles ensures that the code is refactored using best practices.

Any business that provides a service or a product to its customer base should do all they can in verifying that the product or service is of the highest quality. In order to maintain or grow its customer base Artemis Financial must provide a secure platform for storing and transferring data so that important customer information is not at risk. Utilizing standard best practices such as Object Oriented Programming principles or secure coding will ensure that Artemis Financials’ customer base is both maintained and able to grow as popularity grows.

**Source Citation**

Rimkiene, R. (2022, August 29). *What is AES encryption and how does it work?*. cybernews. https://cybernews.com/resources/what-is-aes-encryption/%0a/

Griskenas, S. (2023, September 1). *What is the SHA-256 algorithm?*. NordVPN. https://nordvpn.com/blog/sha-256/