****

# Practices for Secure Software Report

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 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/18/24** | **Jesse Moore** |  |

## 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

Jesse Moore

## Algorithm Cipher

SHA-256

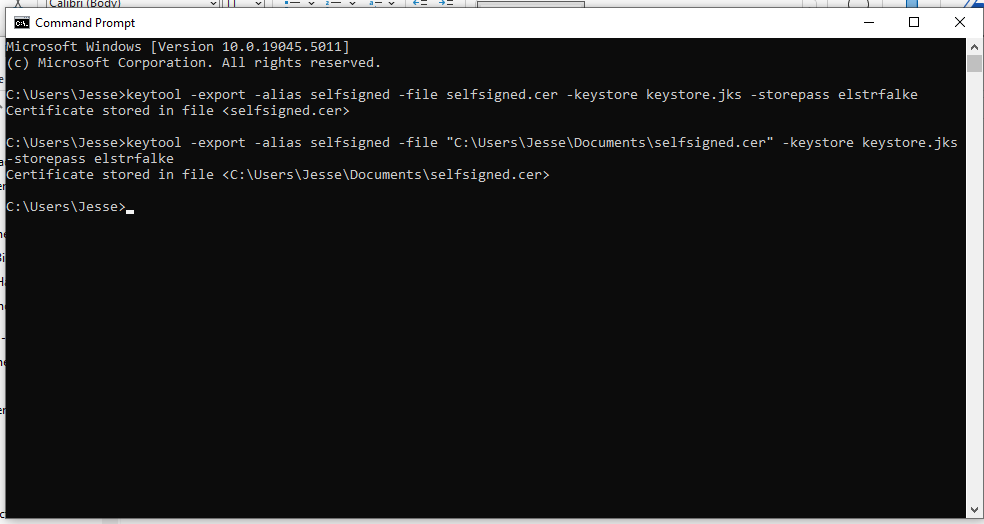
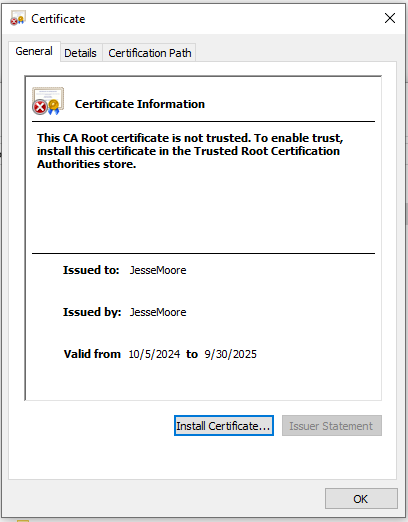
It’s a secure and trusted industry standard: SHA-256 is an industry standard that is trusted by leading public-sector agencies and used widely by technology leaders.

Collisions are incredibly unlikely: There are 2256 possible hash values when using SHA-256, which makes it nearly impossible for two different documents to coincidentally have the exact same hash value. (More on this in the following section).

The avalanche effect: Unlike some older hashing algorithms, even a very minor change to the original information completely changes the hash value—what is known as an avalanche effect.

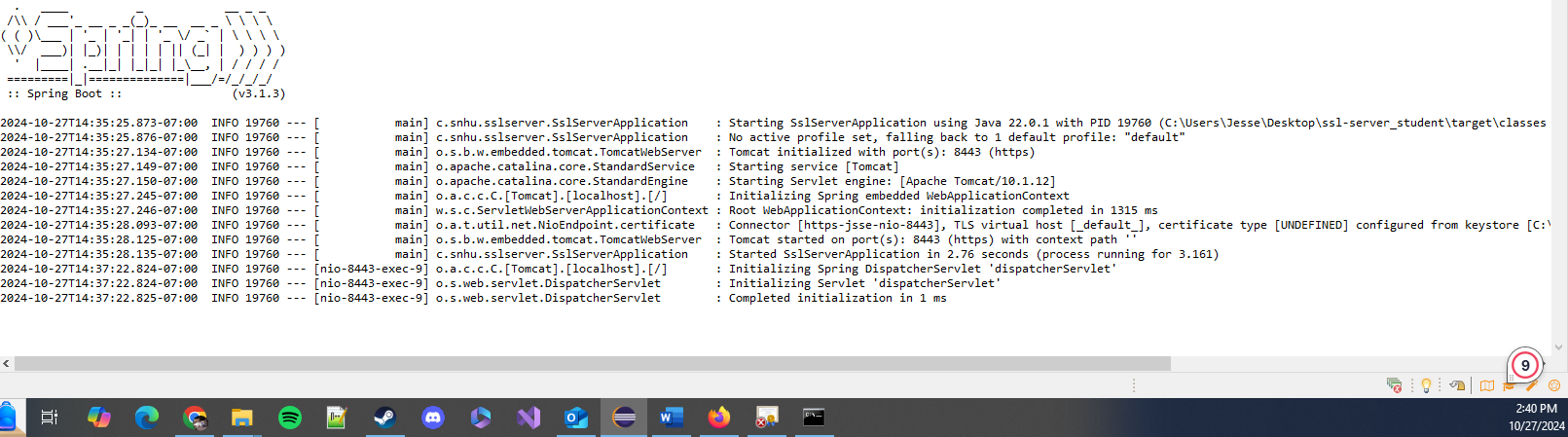
The main reason technology leaders use SHA-256 is that it doesn’t have any known vulnerabilities that make it insecure and it has not been “broken” unlike some other popular hashing algorithms.

## Certificate Generation



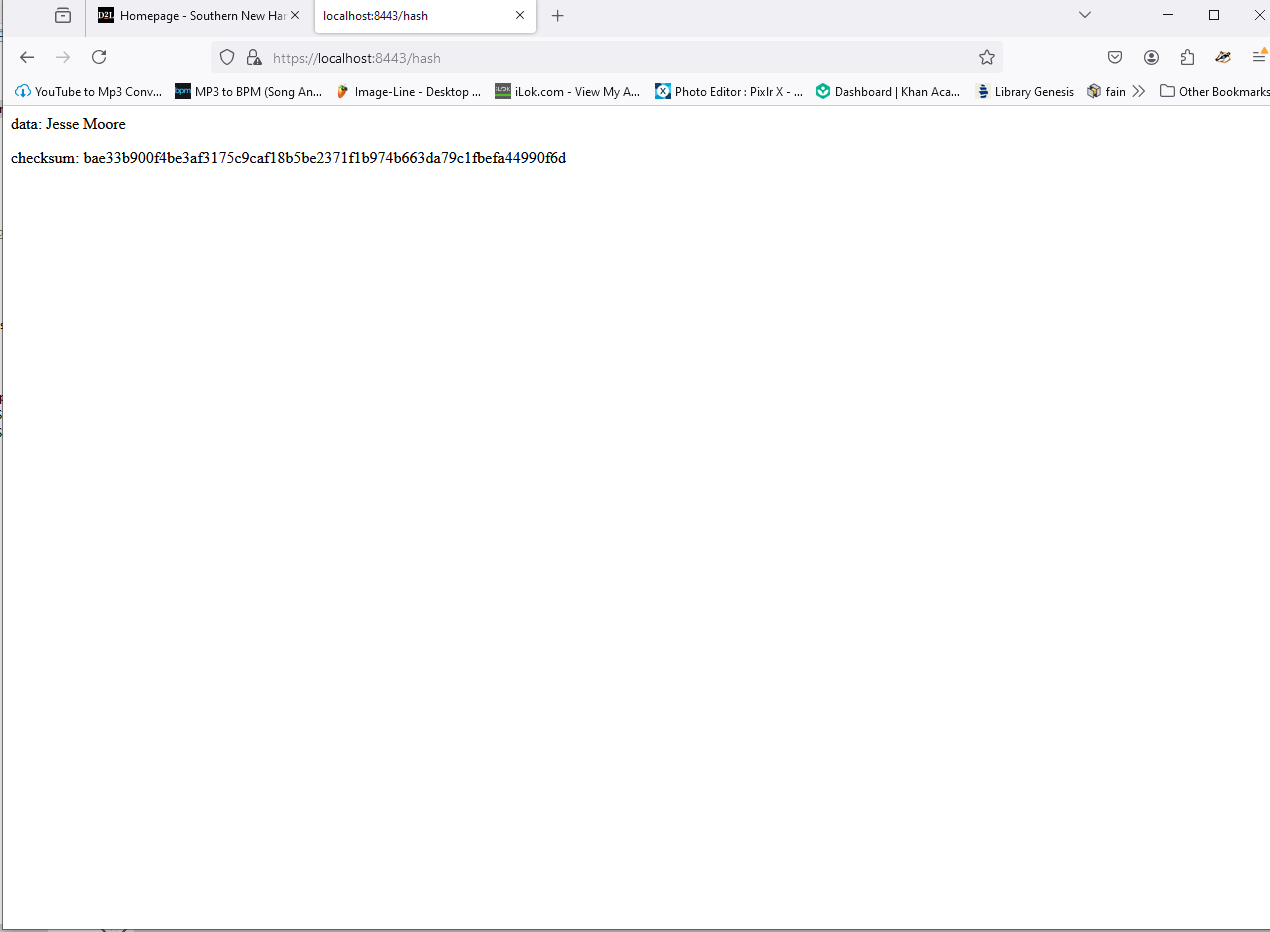
## Deploy Cipher

Insert a screenshot below of the checksum verification.



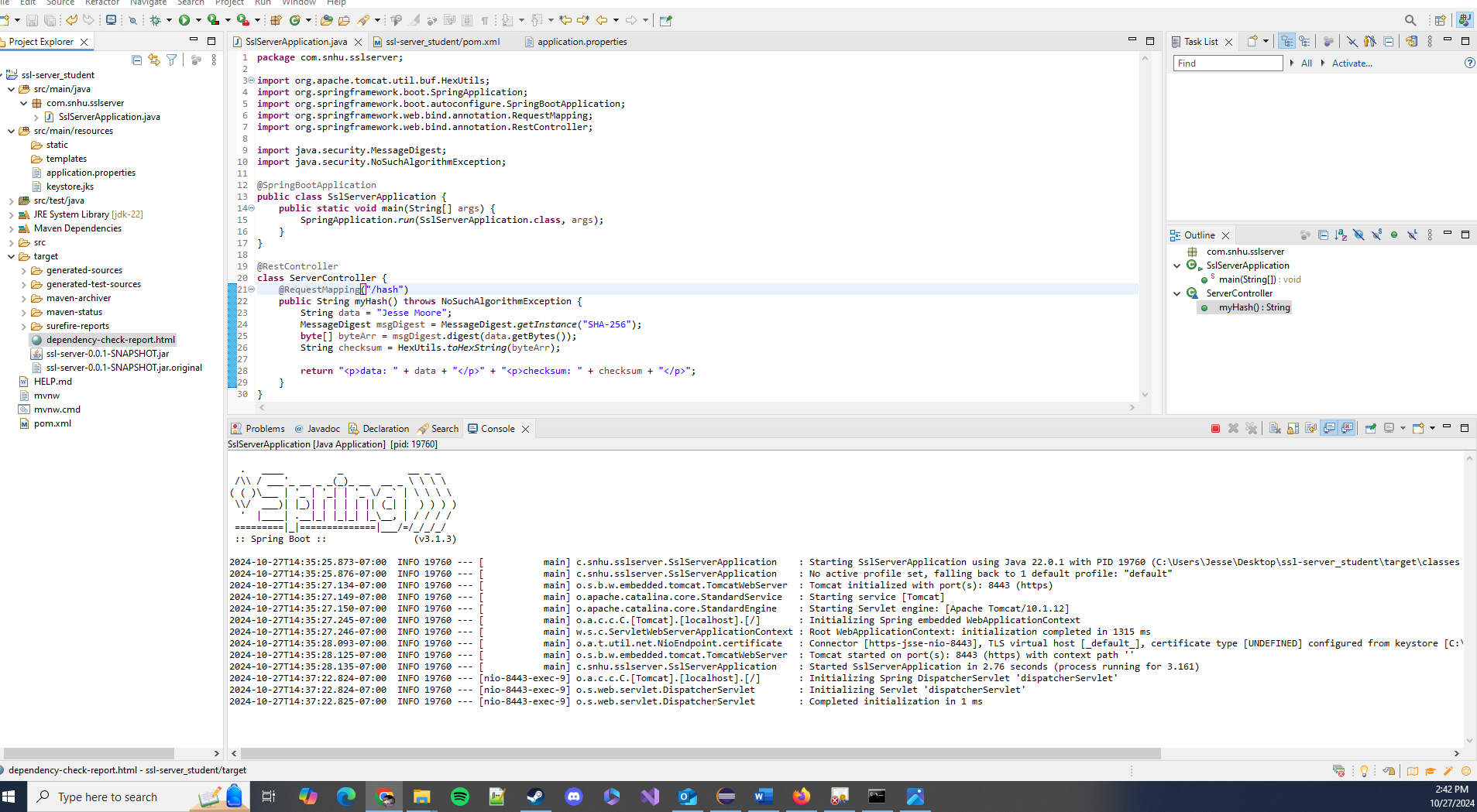
## Secure Communications

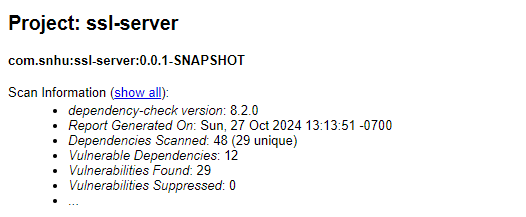
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

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





## Functional Testing

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

[Insert screenshots here.]

## Summary

In this project I refactored the SSL Server Application’s code in order to enhance security measures. This was done by deploying a cipher to ensure safe communications through HTML, ensuring data encryption. The checksum verification using SHA-256 specifically guaranteed data integrity. All of this was done by following industry standard best practices.

## Industry Standard Best Practices

In developing this SSL Server Application I made sure to apply industry-standard best practices in order to provide a safe and secure product. This was done by mitigating vulnerabilities, eliminating false positives wherever possible, and deploying a strong and trusted algorithm cipher, SHA-256. By using SHA-256 I secured sensitive data and enabled safe data transmission through HTML. By making sure the used libraries were up to date I made sure that the product delivered was as protected from vulnerabilities as possible. Also, by implementing a contingency for errors by adding throws exceptions I was able to handle errors without opening the program up for security breaches in information.