****

# 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** | **10/14/2024** | **Jacob Perry** |  |

## Client



## Instructions

Submit these 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

Jacob Perry

## Algorithm Cipher

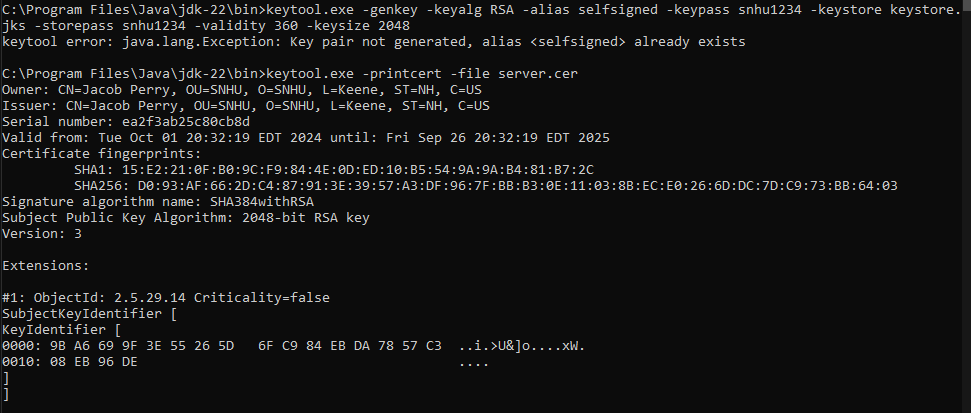
I have decided to use SHA-256, or Secure Hash Algorithm 256-bit, for this project. This algorithm produces a unique, fixed size 256-bit hash no matter what is used as input data. The smallest change in input data can result in an entirely different hash, which further proves the integrity of this algorithm for security.

Logically, I had thought to use AES encryption, which makes use of random numbers and uses symmetric keys, but SHA-256 made more sense to me when it came to generating a checksum like we did in Week 5. SHA-256 does not specifically make use of random numbers or keys, but it is a fully secure hash function, meaning this algorithm generates a unique 256-bit hash form any input.

SHA-256 was introduced by the National Insititute of Standards and Technology, or NIST, in 2001. It is commonly used nowadays in blockchain tech and software integrity verification because of how secure and reliable it has proven to be.

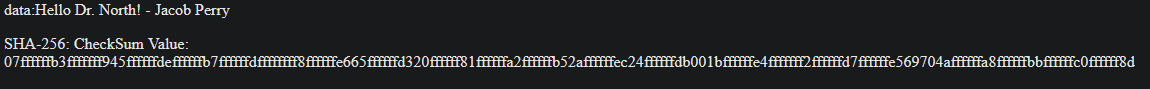
## Certificate Generation

Insert a screenshot below of the CER file.



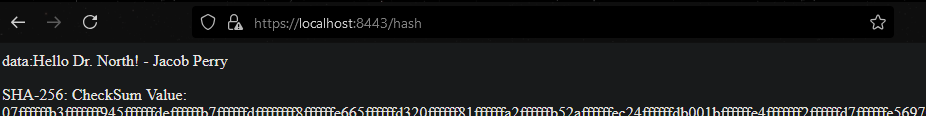
## Deploy Cipher

Insert a screenshot below of the checksum verification.



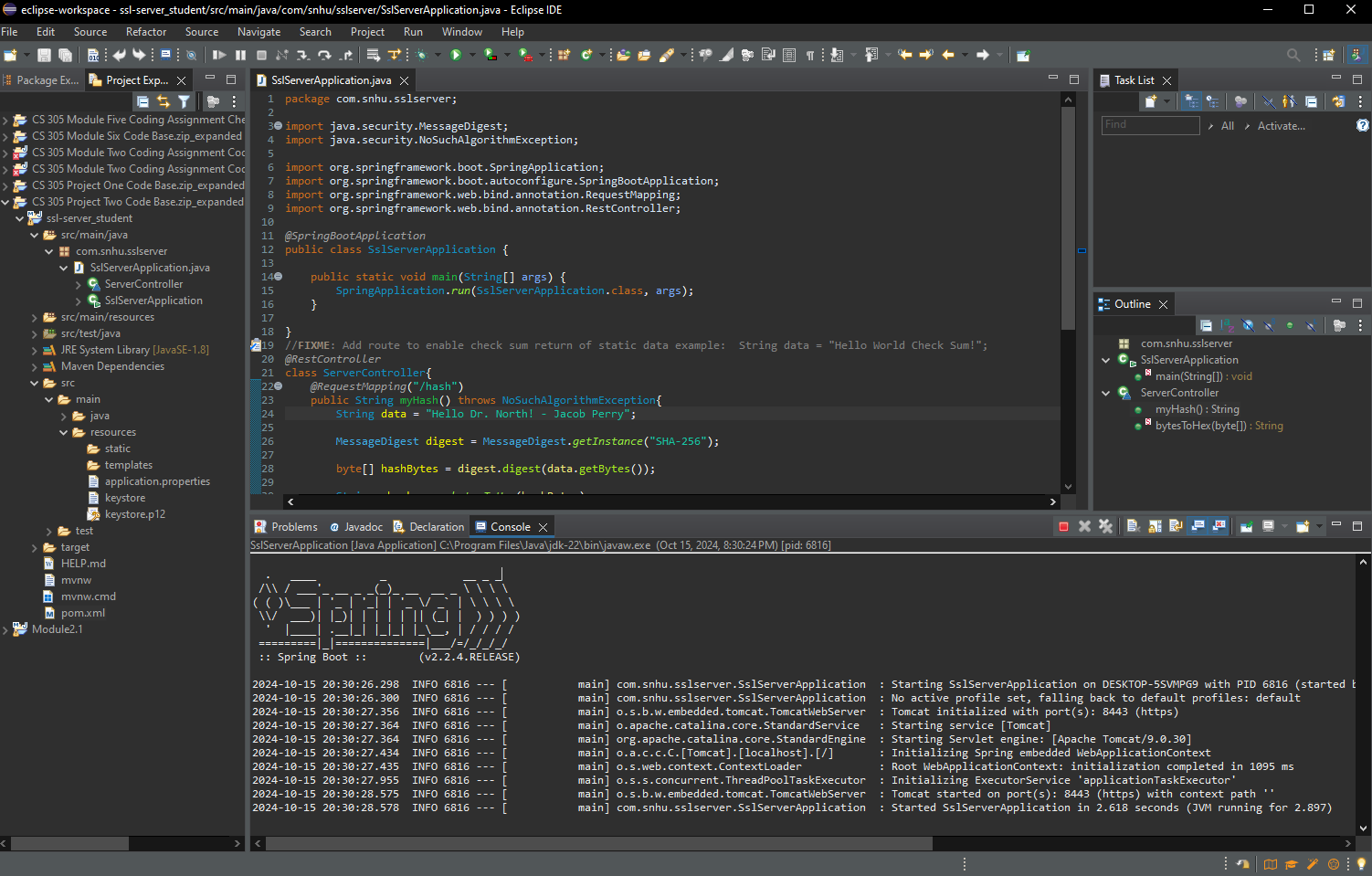
## Secure Communications

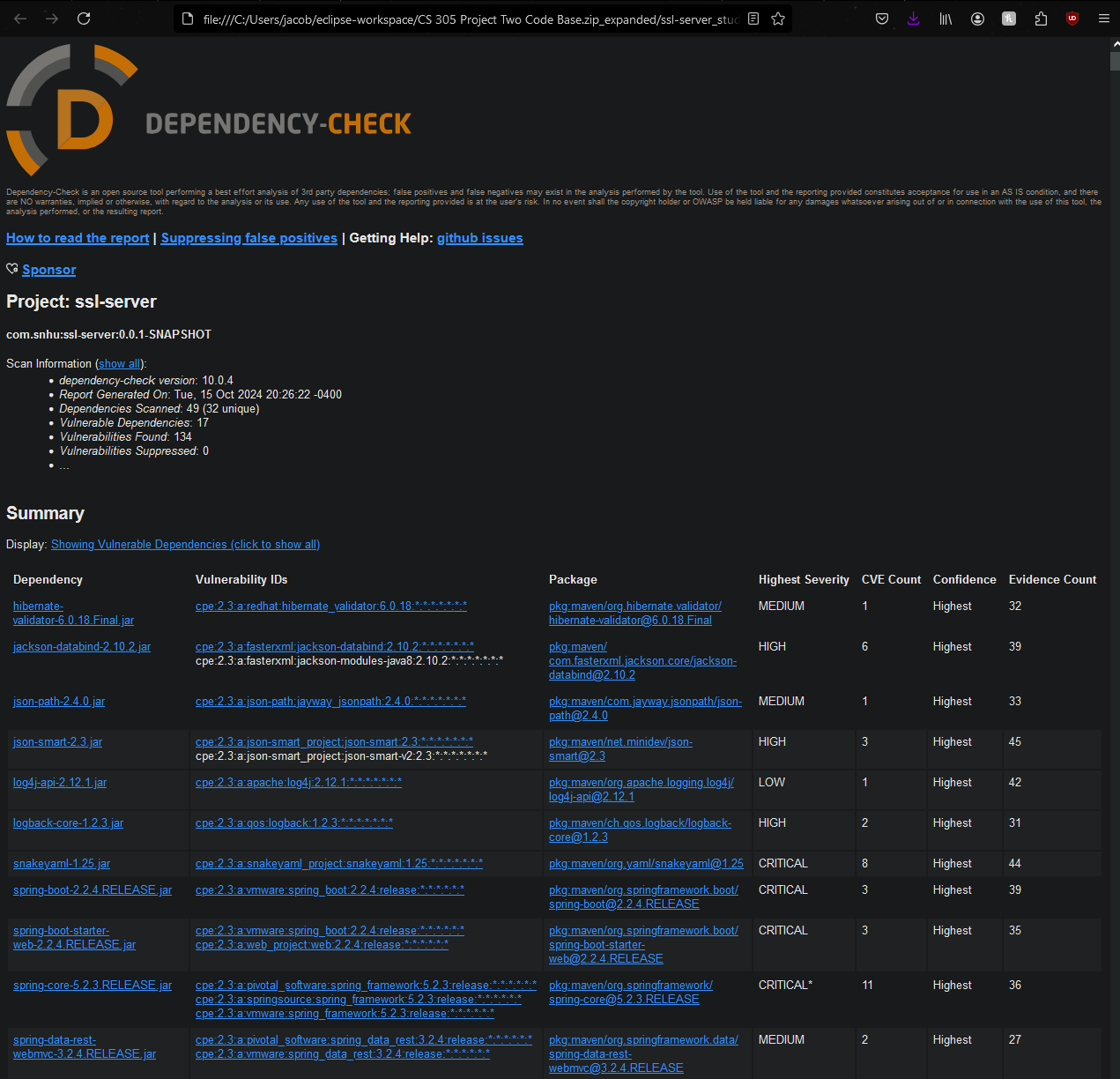
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

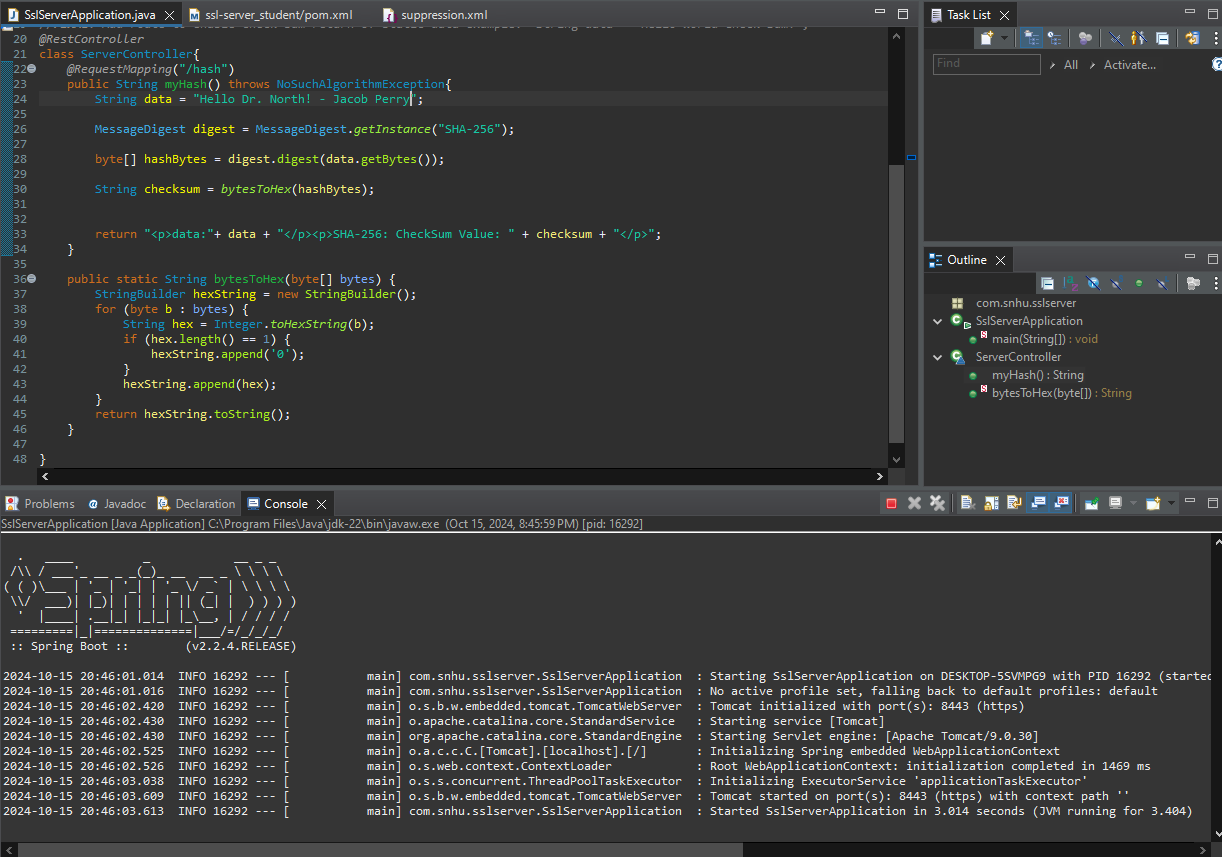
Insert screenshots below of the refactored code executed without errors and the dependency-check report.

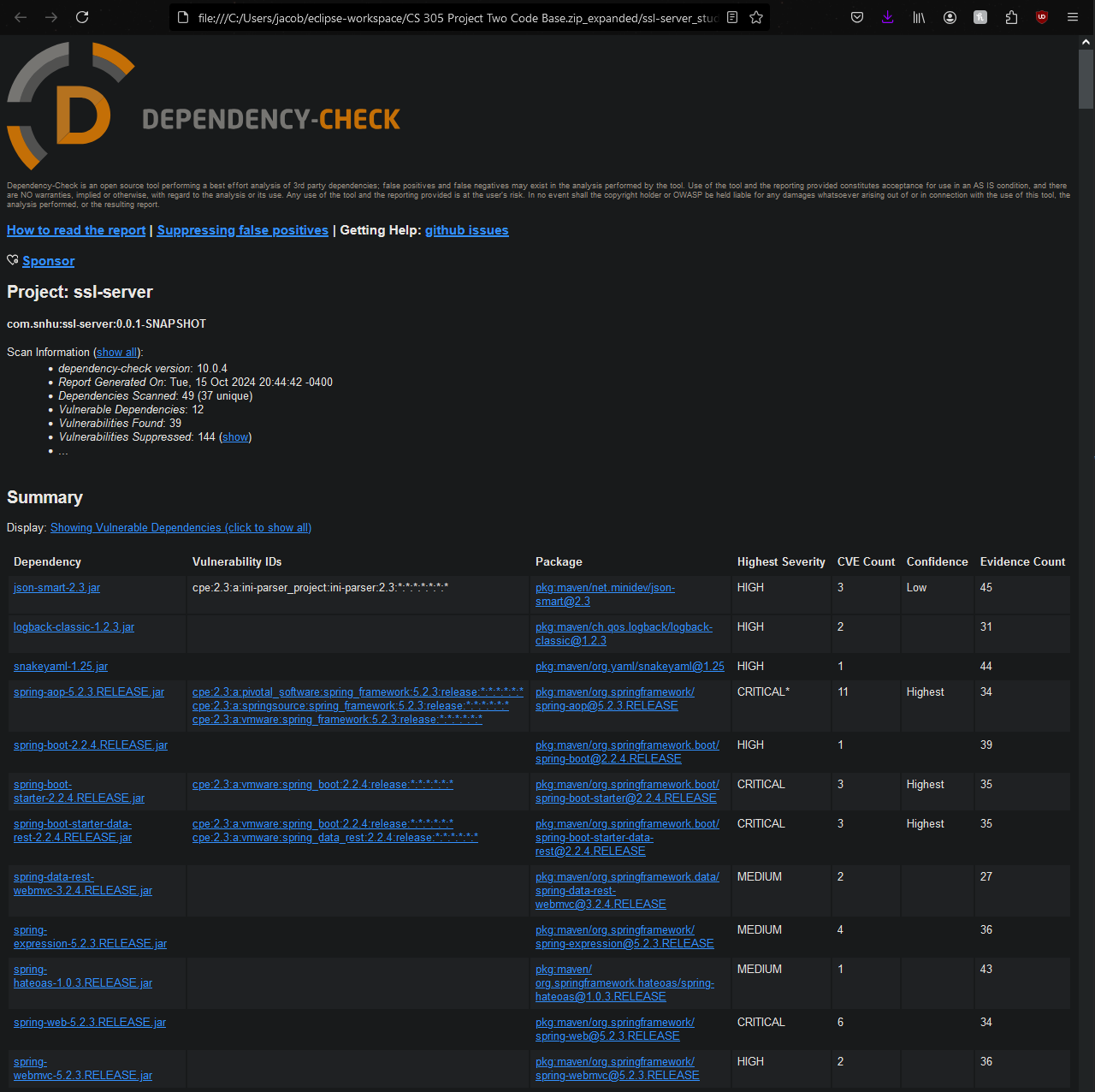




## Functional Testing

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





## Summary

One of my main focuses throughout this project was to follow each of the seven steps from the flow chart we were provided in our resources. For input validation, we validated all input prior to hashing with our algorithm to lower our risk for injection attacks. We refactored the communication in our API to use HTTPS and to use SHA-256, where both are beneficial to the security and integrity of our app. We utilized the SHA-256 hashing function for the cryptography within our app for high levels of reliability and security. Our final product resulted in error-free code as seen in the screenshots throughout this document. I properly encapsulated all sensitive data within the project by making sure there was not raw input data being passed to our functions, but rather storing things in variables like “data” and passing that variable into the method that was created.

## Industry Standard Best Practices

Given that the code base we were supplied was using some outdated versions of APIs, one example being the version of Springboot, I still mitigated security risks to the best of my ability while still allowing the code to run error free. When going over the dependency check report after each batch of changes I made, I made sure to suppress any vulnerabilities that I did not believe to either be directly harmful to the integrity of the product or be directly relevant in this product.

By integrating SHA-256 into the product, I was directly following guidelines laid out by NIST, or the National Institute of Standards and Technology. I also verified that steps were being taken in preventing unauthorized access to the data being stored by making use of HTTPS.

The idea of companies following industry standard best practices is integral to the overall health and success of any company. Following these standards allows customers to build trust with the companies they choose to work with. They are able to trust that companies will store and transmit their data without the worry of it being accessed by those who shouldn’t and that the data remains untampered with. All of these processes blanketed together will help companies build better reputations and relationships in the long run.