

# 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** | **6/3** | **Tyler Johnson** |  |

## 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

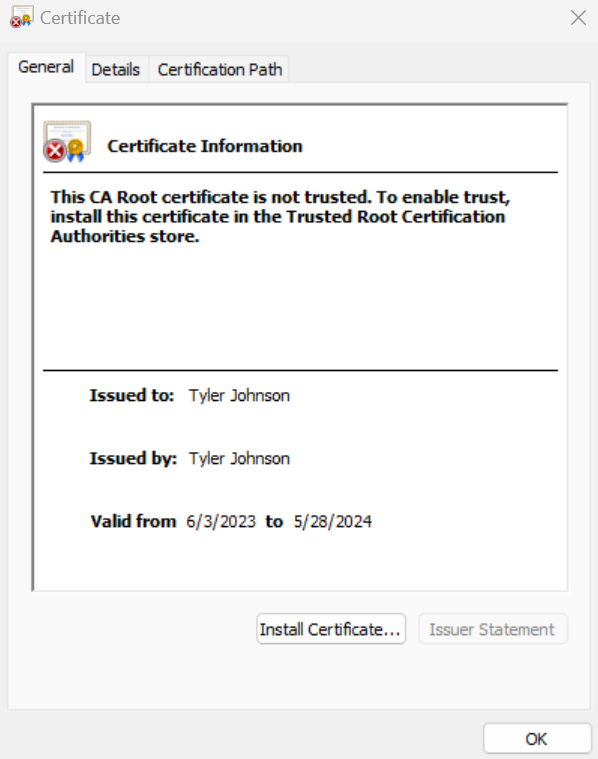
Tyler Johnson

## Algorithm Cipher

I would recommend SHA-265 for this project. also known as the Secure Hash Algorithm. SHA is a symmetric high-level one way hashing function that was created in 2001 by the NSA. SHA takes an arbitrary bit string as input and returns a fixed length string as output. There are four bit sizes that can be used for SHA, that includes: 224, 256, 384. 512 bits. Symmetrical means that it uses the same key for encryption and decryption. Random numbers are crucial in AES or SHA encryption, specifically for generating encryption keys, to create these keys, a random number generator is commonly employed. Encryption has been around for centuries dating back as far as ancient Greece and has evolved drastically with technology. There are obviously many ways to go about encryption, however I believe for this project the best choice is the current standard for high-level hashing.

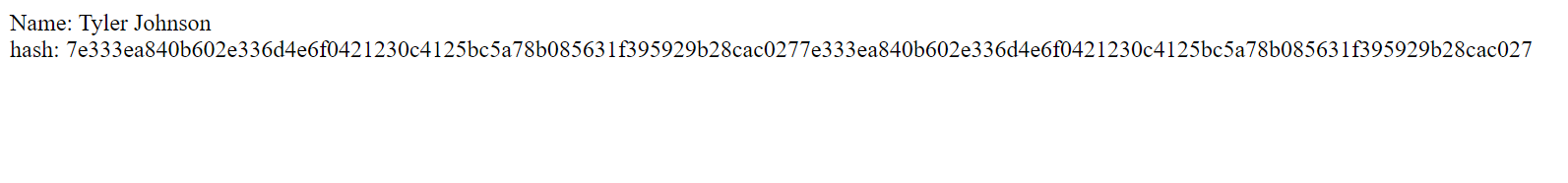
## Certificate Generation

Insert a screenshot below of the CER file.



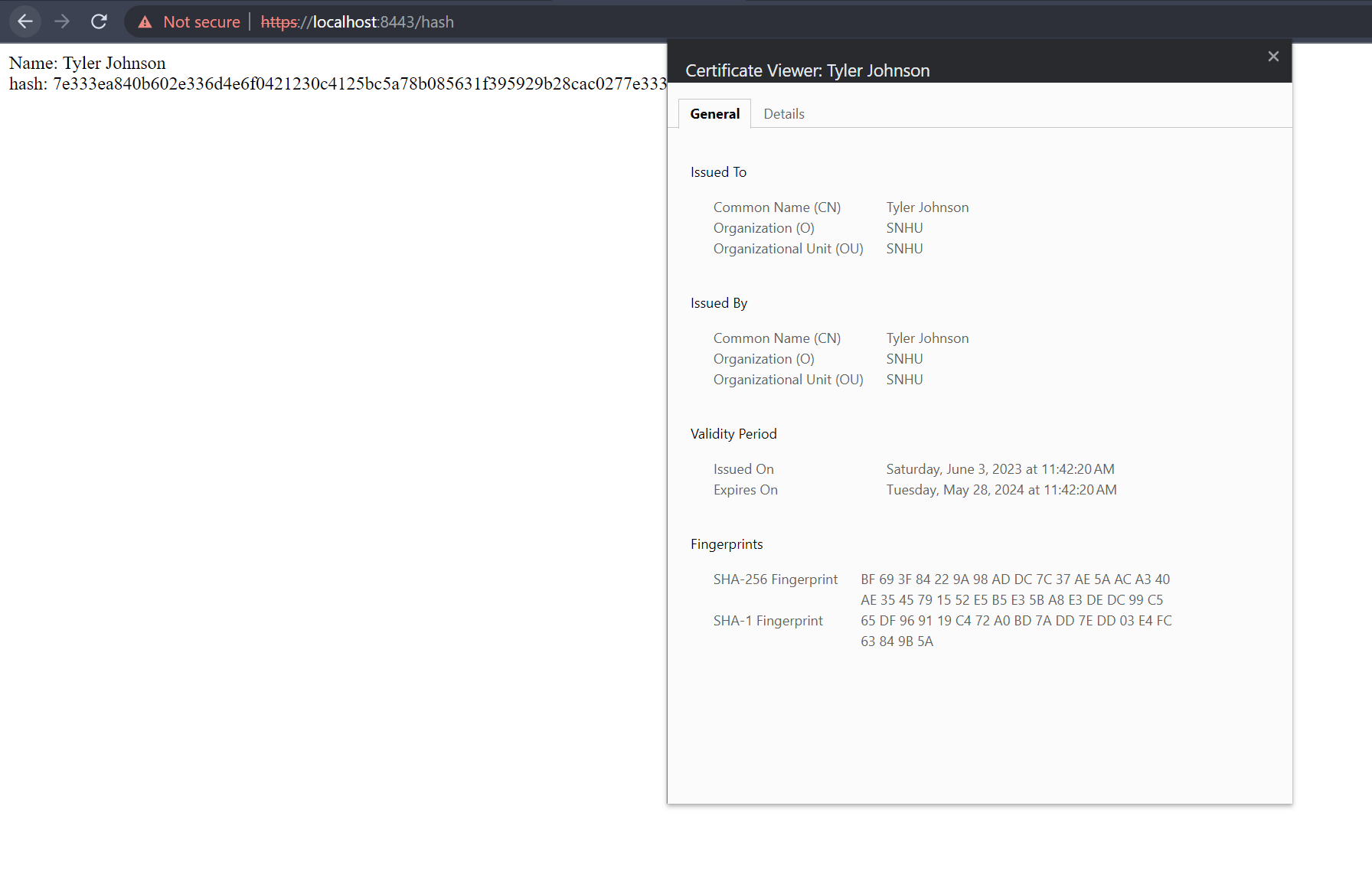
## Deploy Cipher

Insert a screenshot below of the checksum verification.



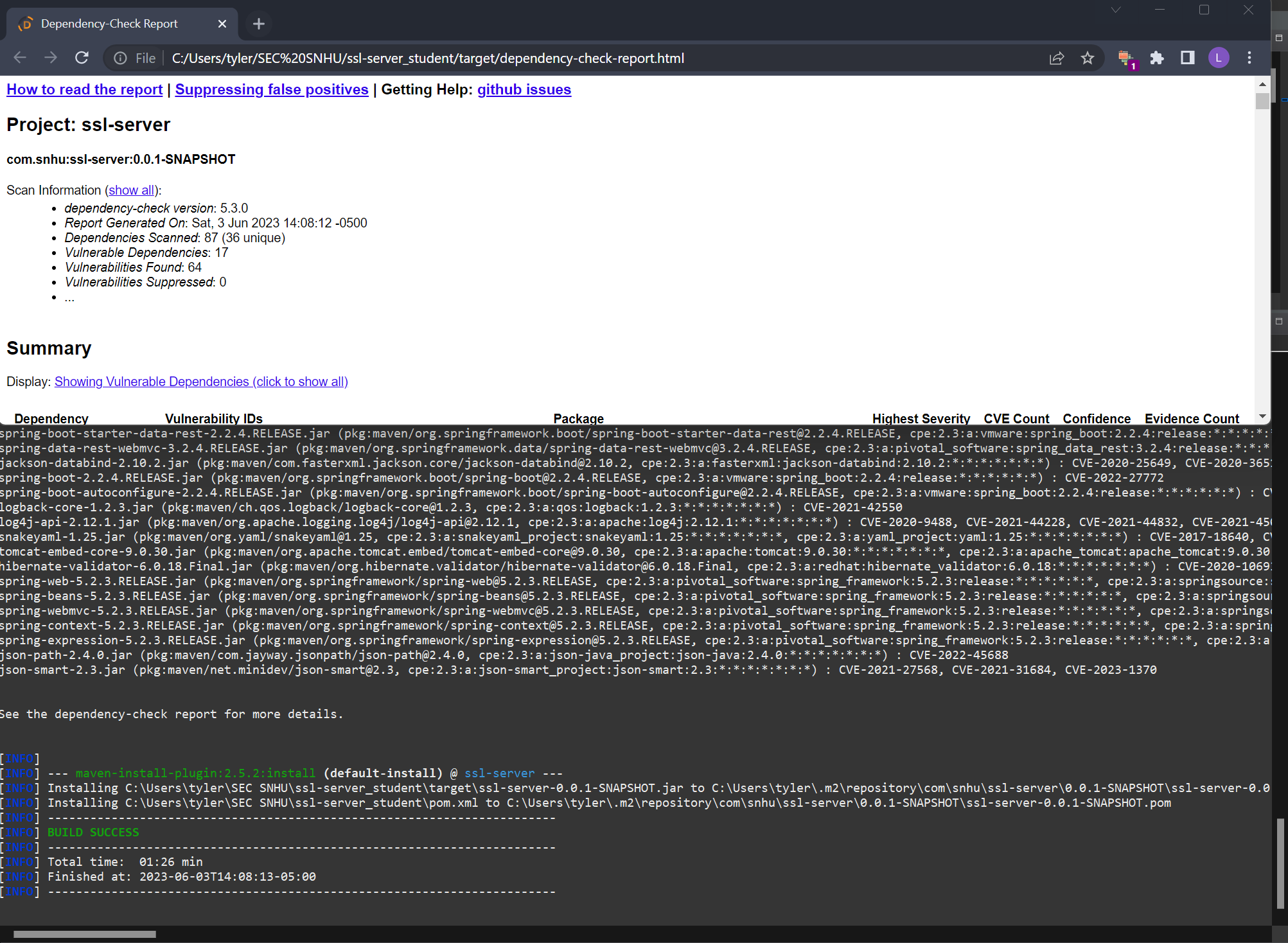
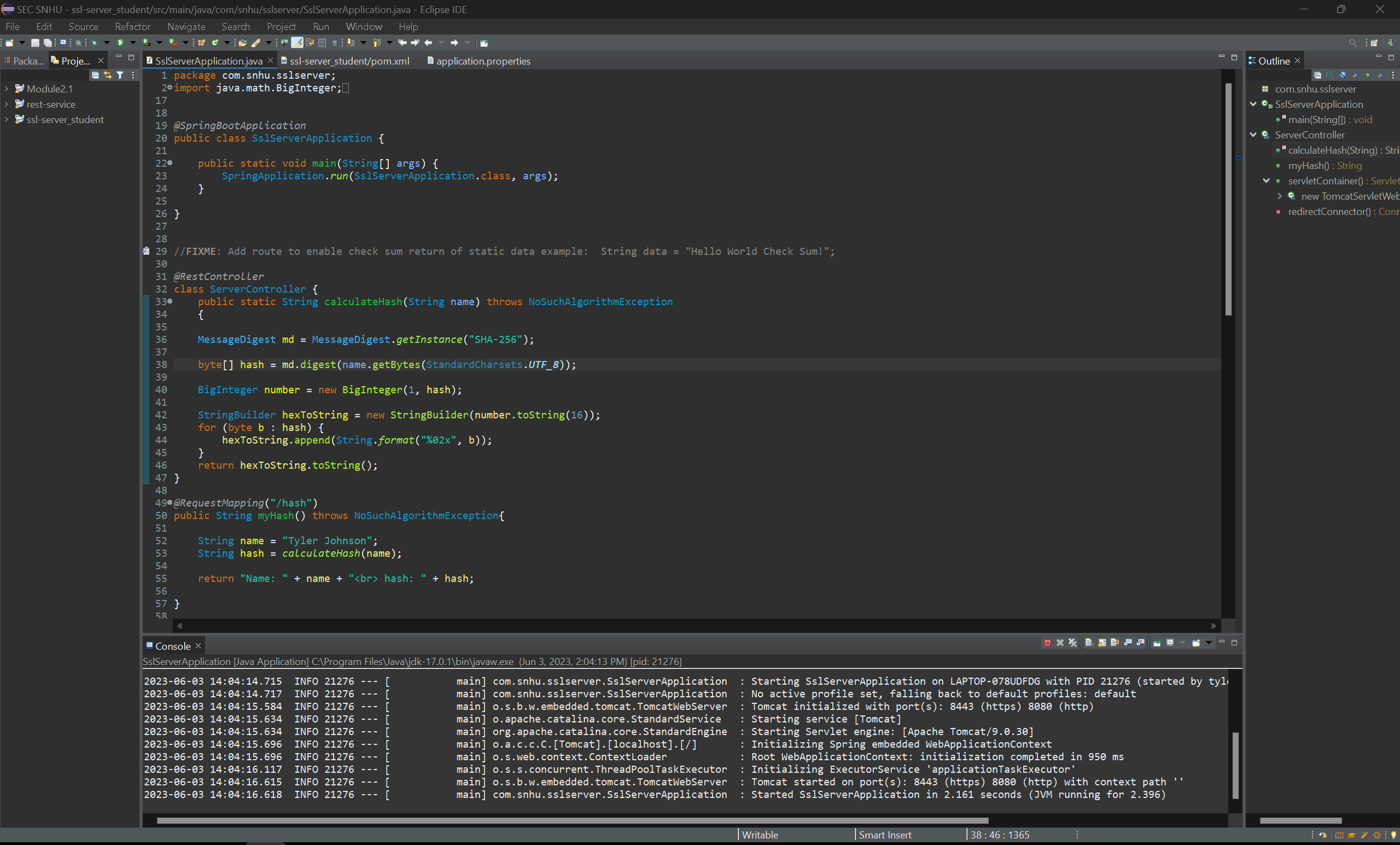
## Secure Communications

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



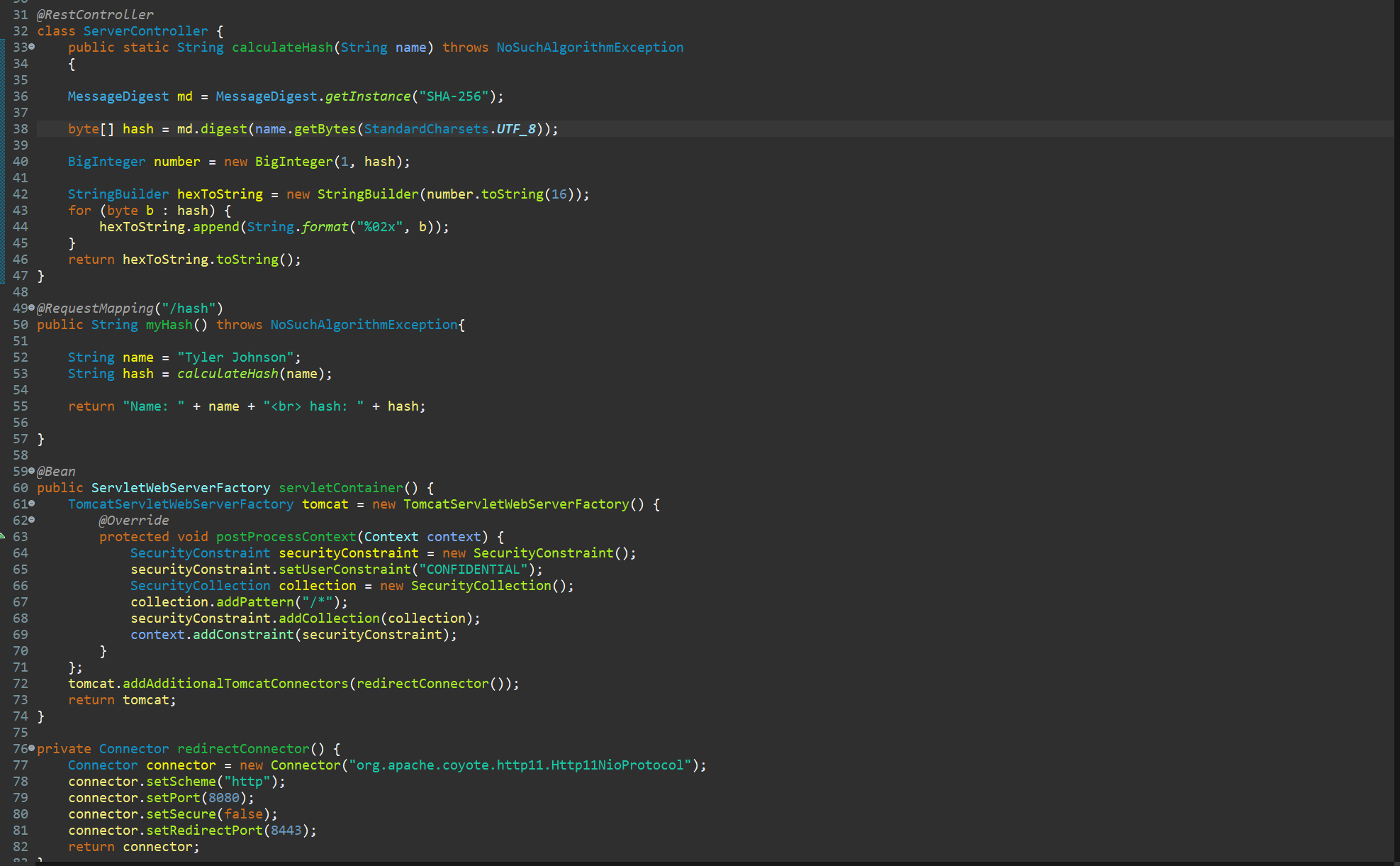
## Secondary Testing

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

Code error as well as Cryptography are the main areas that were focused on in this project. I added the hash function to be able to take the static test variable and encrypt with the one way hash function. I also added some error handling and configured the application to reroute http traffic to https. As well as created and implemented a self signed certificate.

## Industry Standard Best Practices

Industry standards are important to follow when adding security to an application. With this project I added a hash function as well as generated a self signed certificate and set the project up to reroute http traffic to https. Industry best practices for hashing is SHA-256, which is what I decided on for this project.