****

# 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** | **[Date]** | **[Your Name]** |  |

## 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

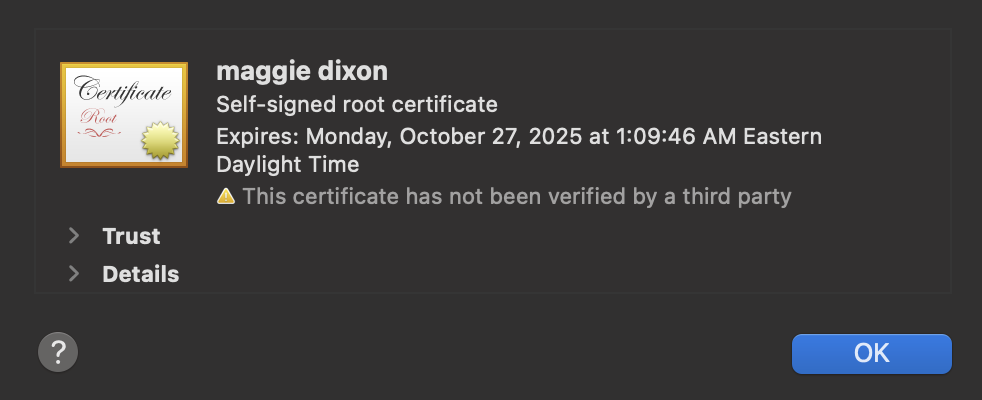
Maggie Dixon

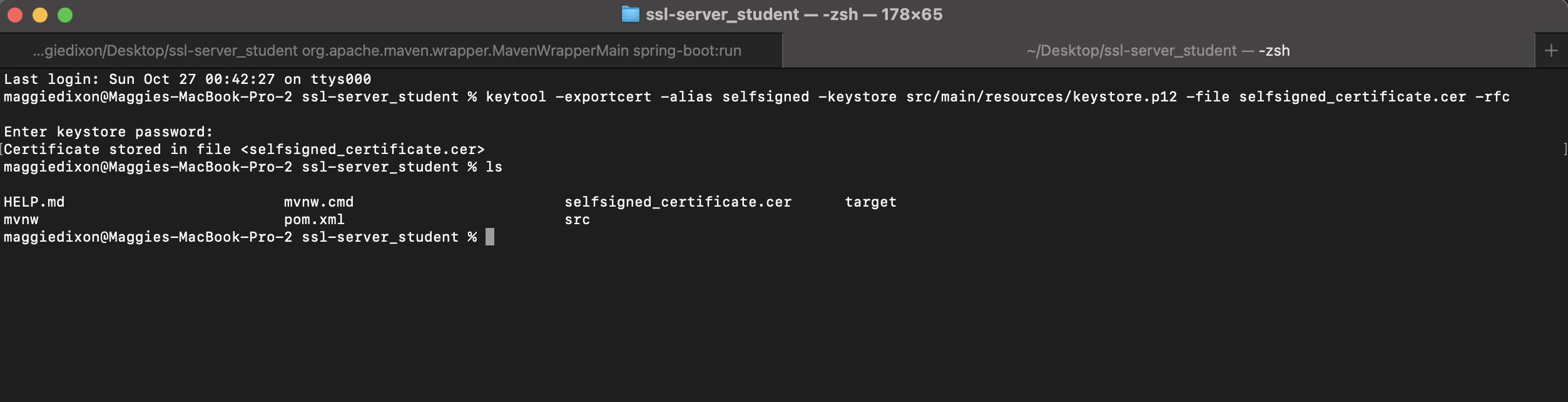
## Algorithm Cipher

Artemis Financial needs additional security for their web application to make sure communications are secure. Encryption would be the best solution to preventing unauthorized users that want to access data. This will make sure that the user can only access files with a pass key. I think that Asymmetric communication would work best as the key to encrypt is public and the decryption key is private. I would suggest using the SHA 256 cipher algorithm with 256 bit key to encrypt. This provides a high level encryption that can allow any password within 256 bits. Encryption is also very secure due to the random number generator that java uses. The hash function uses SHA 256 to make a checksum of the given message.

## Certificate Generation

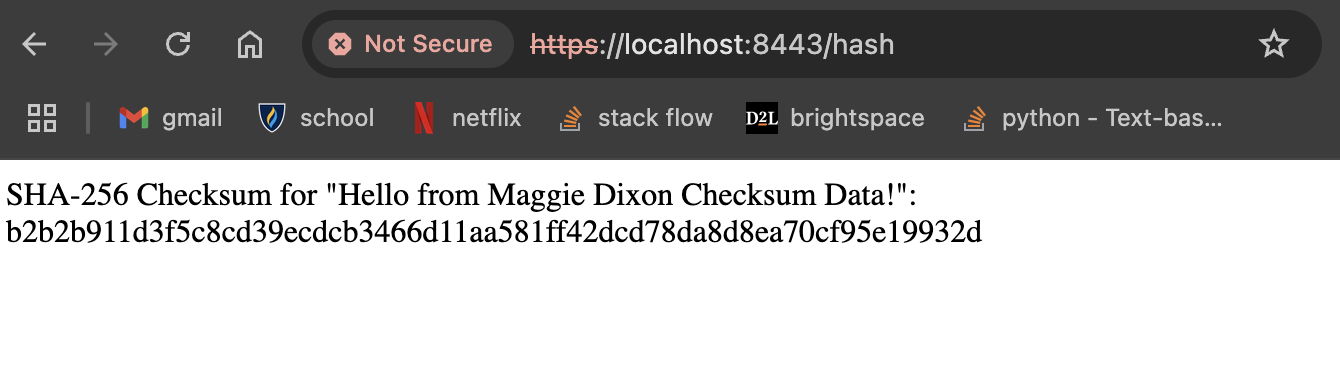
Insert a screenshot below of the CER file.





## Deploy Cipher

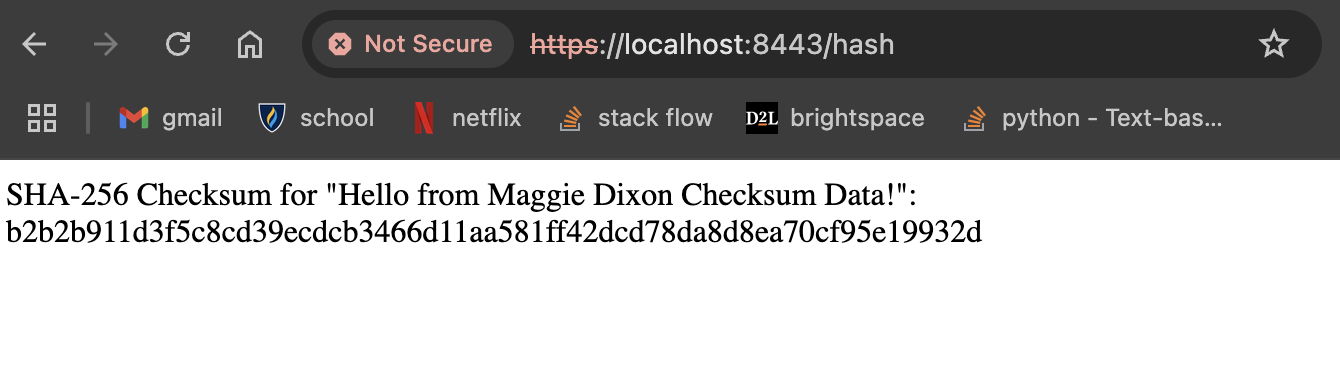
Insert a screenshot below of the checksum verification.



## Secure Communications

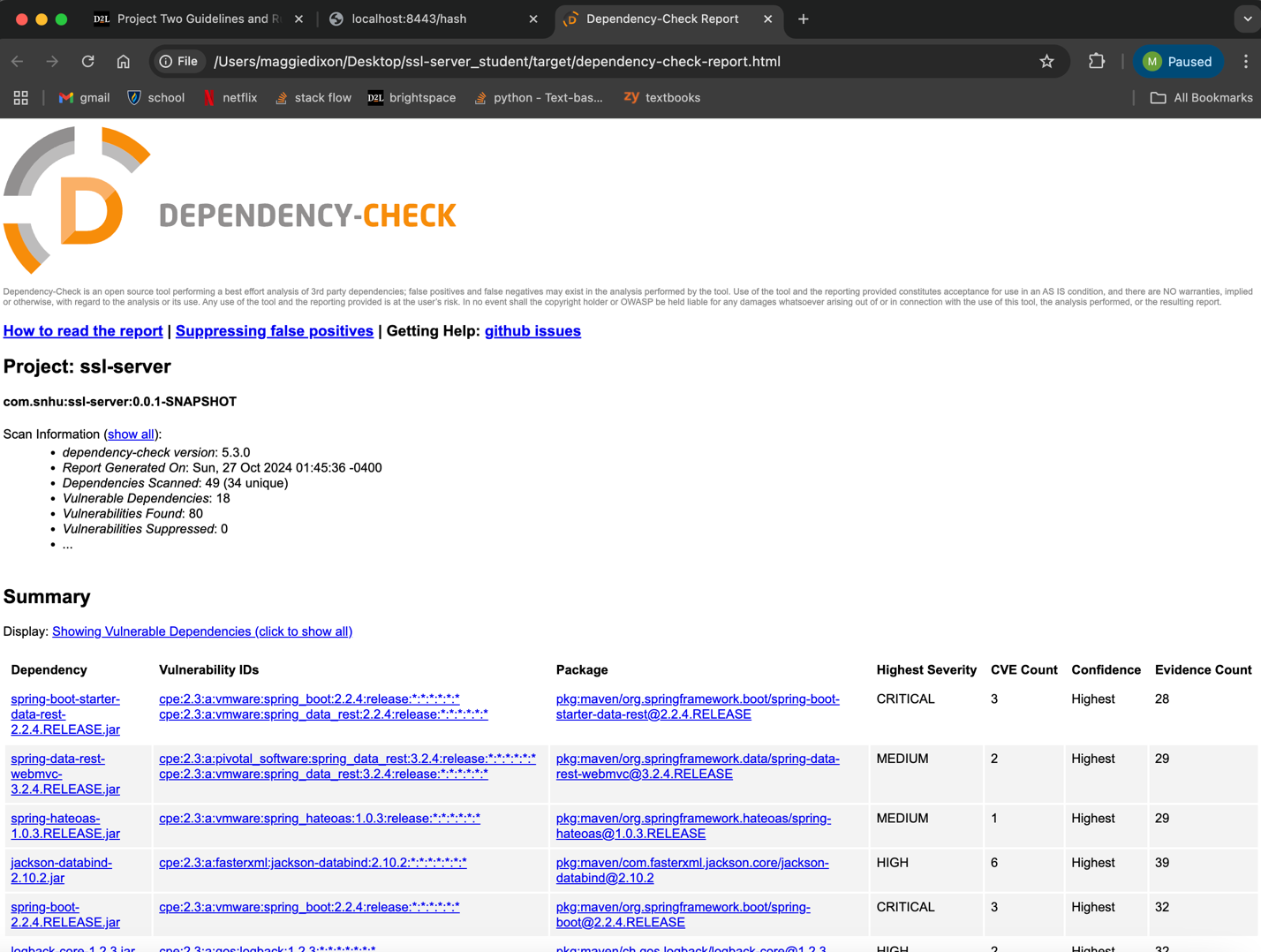
Insert a screenshot below of the web browser that shows a secure webpage.

Not a secure webpage since my certificate is self signed.



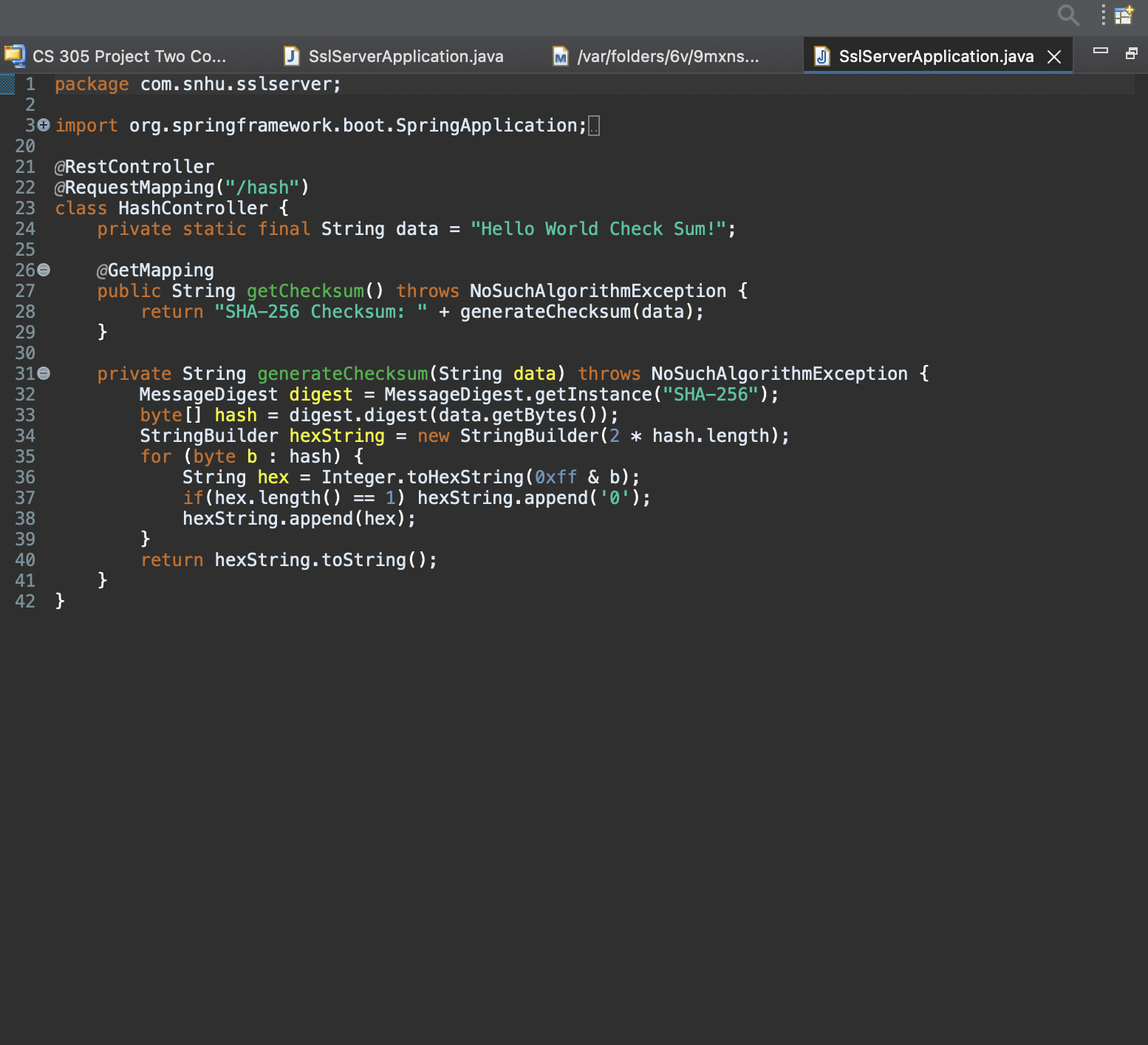
## Secondary Testing

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

**Refactoring in my code I added rest controller, so that it works for the restful stop. The server controller class matches problems that are shown by the vulnerability diagram. I used SHA 256 because it is very secure and has a small chance of collisions. To keep up with industry standards and security, there should be monthly dependency checks on the application to keep the most up do date on possible vulnerabilities. This will help protect sensitive data. Keeping the plugins in the pom.xml file will also help keep the latest plugins up to data with the most up to date security.**