****

# 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** | **04-20-2025** | **Crailin Mogged** |  |

## 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

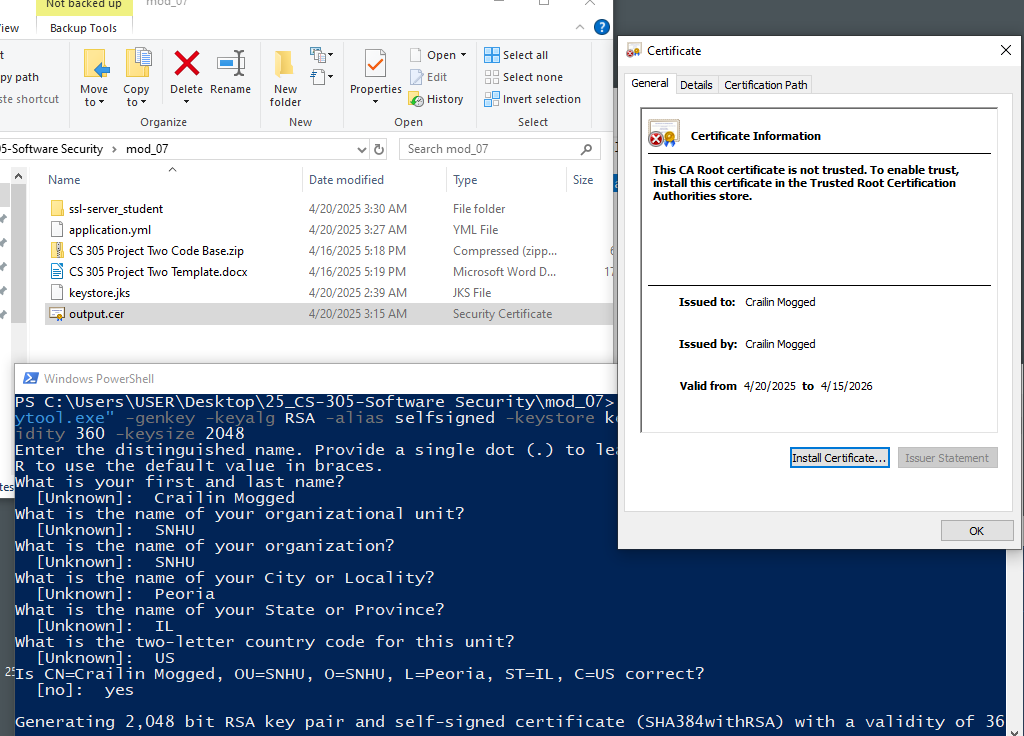
Crailin Mogged

## Algorithm Cipher

SHA-256

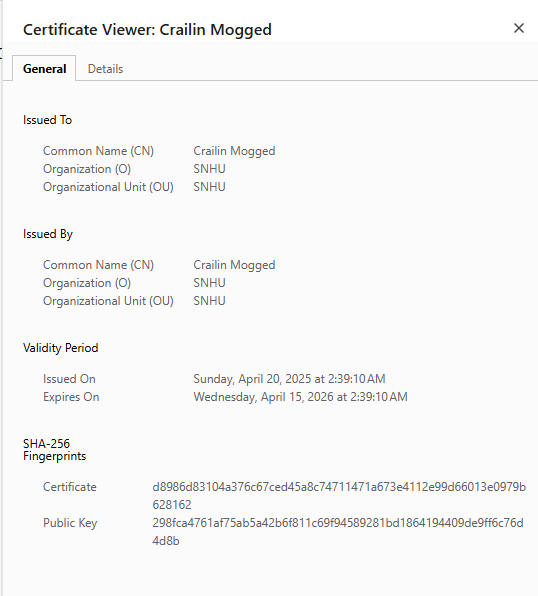
## Certificate Generation

Insert a screenshot below of the CER file.



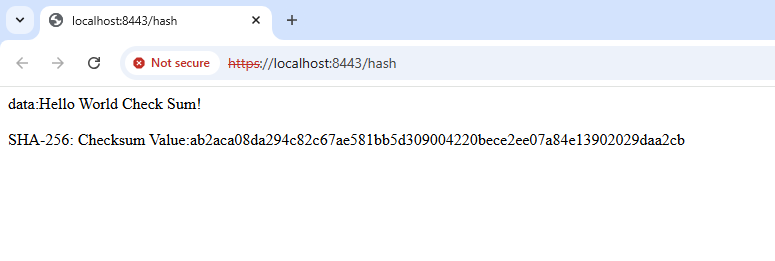
## Deploy Cipher

Insert a screenshot below of the checksum verification.



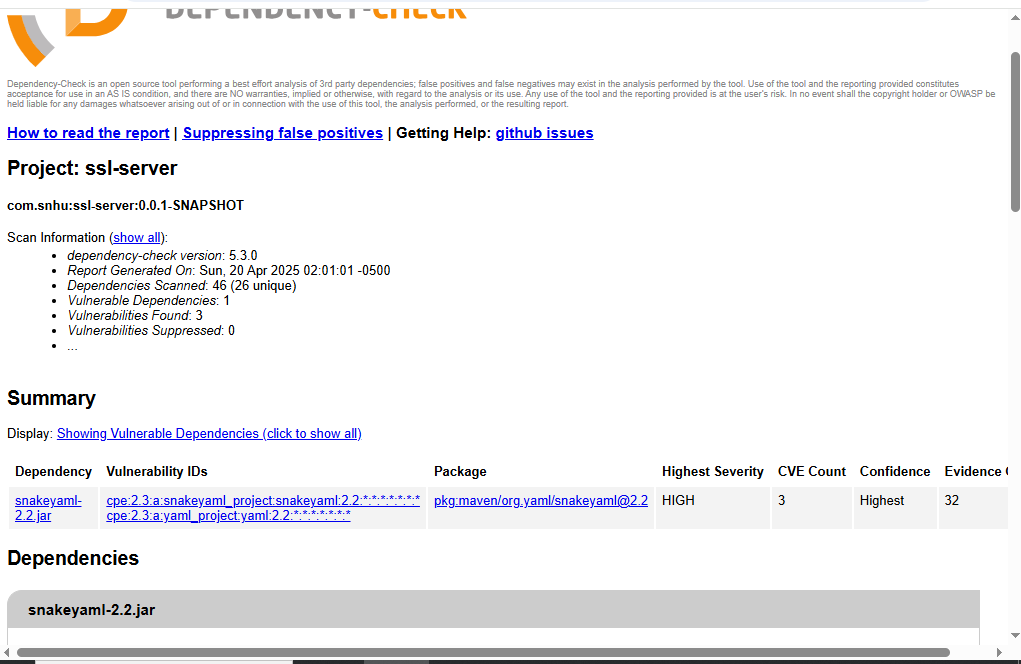
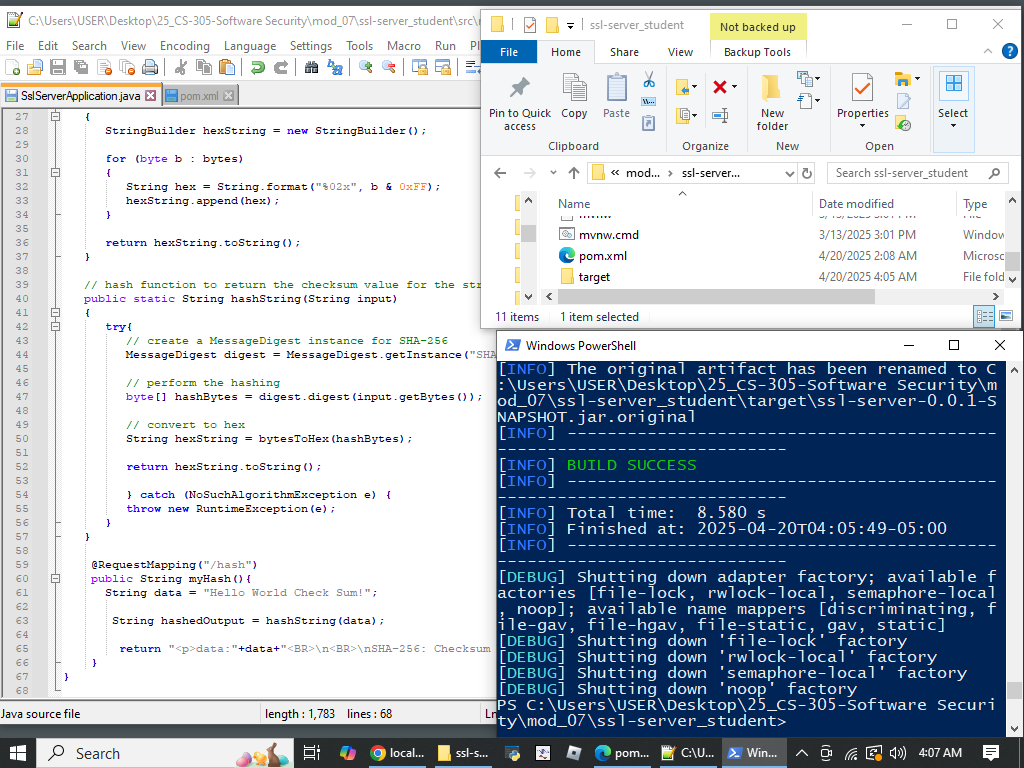
## Secure Communications

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



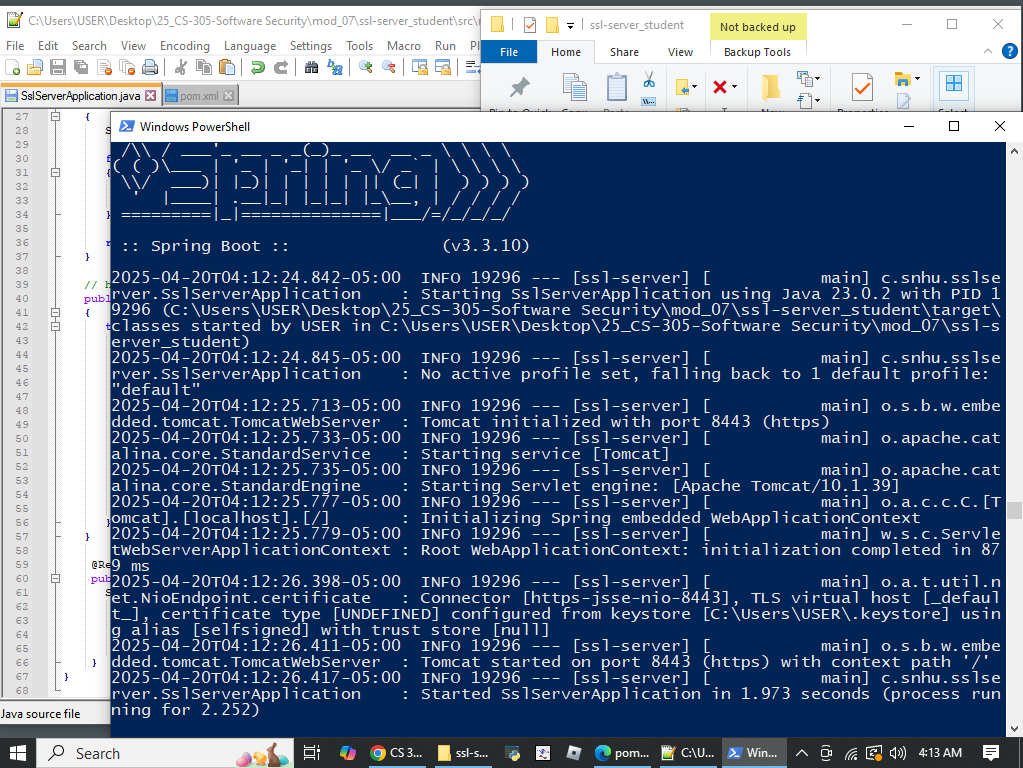
## Secondary Testing

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

My experience using the dependency-check tool was as follows: When first compiling the code, there were many vulnerabilities pointed out by the dependency-check tool. However, this was mitigated by updating the "spring boot framework". Using the dependency-check once more revealed still one vulnerability left. It is recommended to implement restrictions on the size of YAML files that can be processed to mitigate this.

## Industry Standard Best Practices

In dealing with large amounts of code it seems quite difficult to maintain safe practices and mitigate vulnerabilities. Keeping up to date as vulnerabilities are found seems like an important part of present-day software development. Large libraries are quickly used and continuously processed for bugs and insecurities. When security is not as important as the usability of such large libraries in the business world, it seems this is the dominant practice. Tools are now being developed, such as the dependency-check tool for java, to help developers keep track and mitigate vulnerabilities in web applications.