****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_30j0zll)

[**Client 3**](#_1fob9te)

[**Instructions 3**](#_3znysh7)

[**Developer 4**](#_2et92p0)

[**1. Algorithm Cipher 4**](#_tyjcwt)

[**2. Certificate Generation 4**](#_3dy6vkm)

[**3. Deploy Cipher 4**](#_1t3h5sf)

[**4. Secure Communications 4**](#_4d34og8)

[**5. Secondary Testing 4**](#_2s8eyo1)

[**6. Functional Testing 4**](#_17dp8vu)

[**7. Summary 4**](#_3rdcrjn)

[**8. Industry Standard Best Practices 4**](#_26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/19/2024** | **Jacob Virgilio** |  |

## 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

Jacob Virgilio

## Algorithm Cipher

Recommend an appropriate encryption algorithm cipher to deploy, given the security vulnerabilities, and justify your reasoning. Review the scenario and the supporting materials to support your recommendation. In your practices for secure software report, be certain to address the following actions:

Provide a brief, high-level overview of the encryption algorithm cipher.

Discuss the hash functions and bit levels of the cipher.

Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.

Describe the history and current state of encryption algorithms.

I would recommend the SHA-256 cipher algorithm. This algorithm has a very low chance of collisions, which is when an algorithm provides the same hash value for multiple different sets of data. This cipher has a total of 36^64 different combinations, which shows that the possibility of two different data sets being the same is very low. This cipher also has a 256 total length in bits, which dramatically helps protect against breaching attacks. Encryption algorithms have been around a very long time, just in different forms. One could even say that different flashes of lights or smoke signals from medieval times could be considered a form of encryption. In their current state, encryption algorithms are used frequently across technology and the internet, helping businesses, governments, and individuals keep their data secure by sharing a security key with the receiving end of the data.

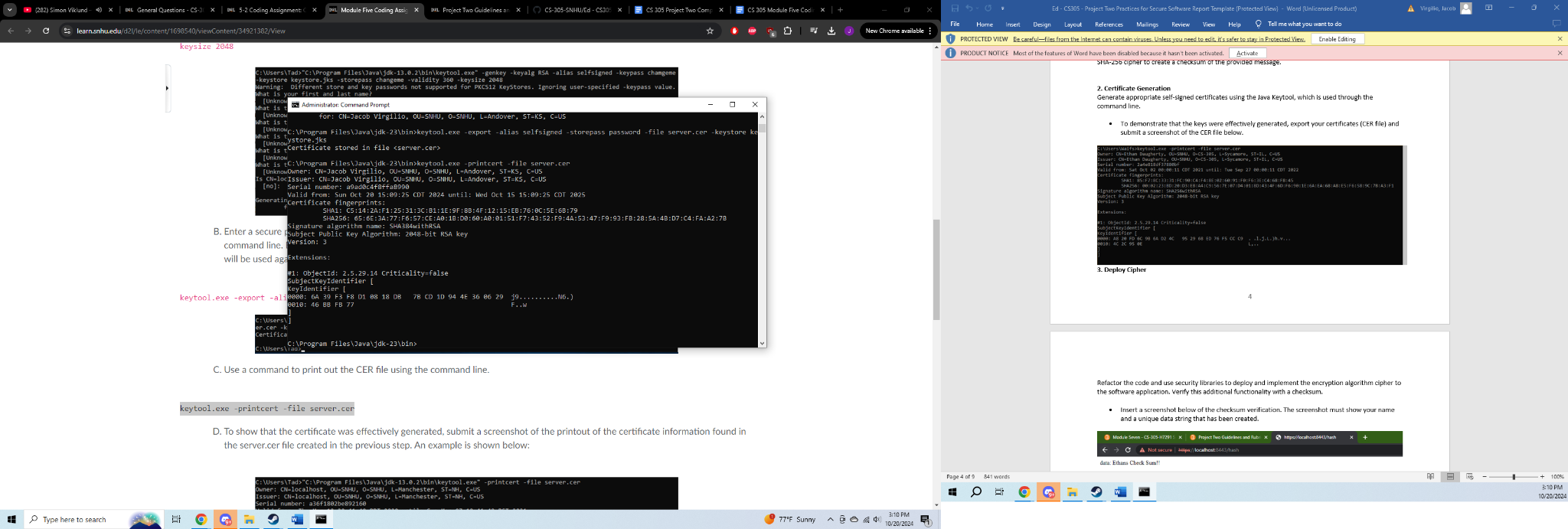
## Certificate Generation

Certificate Generation: Generate appropriate self-signed certificates using the Java Keytool in Eclipse.

Complete the following steps to demonstrate that the certificate was correctly generated:

Export your certificates as a CER file.

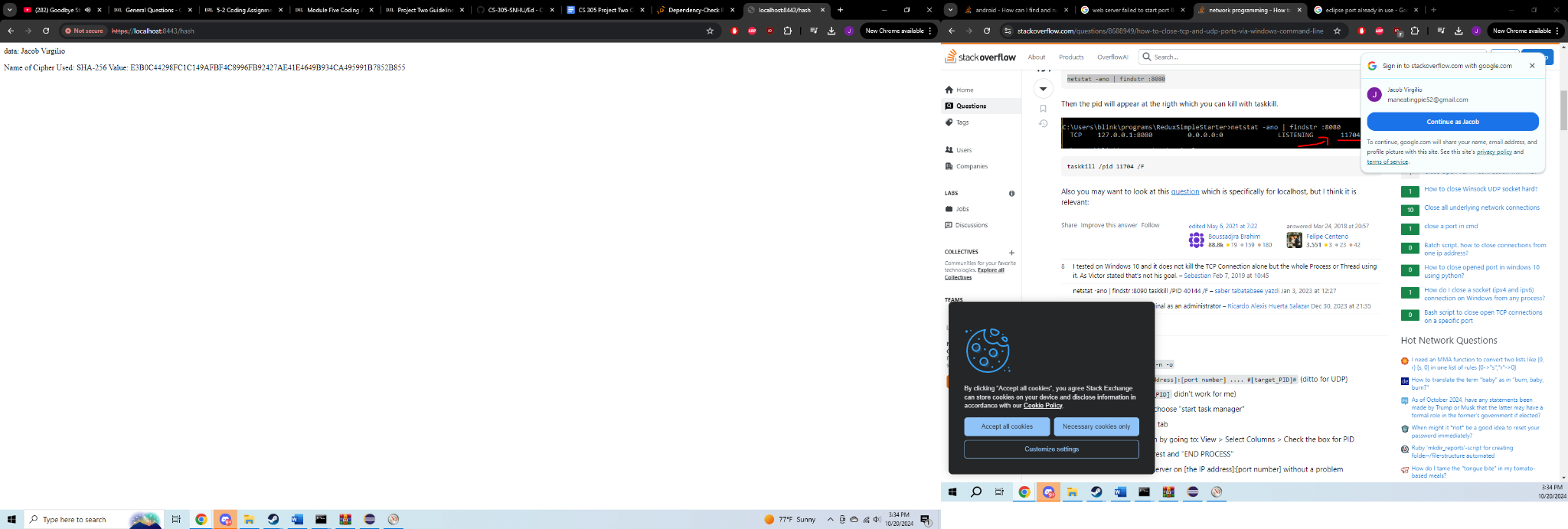
Submit a screenshot of the CER file in your practices for secure software report.



## Deploy Cipher

Deploy Cipher: Deploy and implement the cryptographic hash algorithm by refactoring code. Demonstrate functionality with a checksum verification.

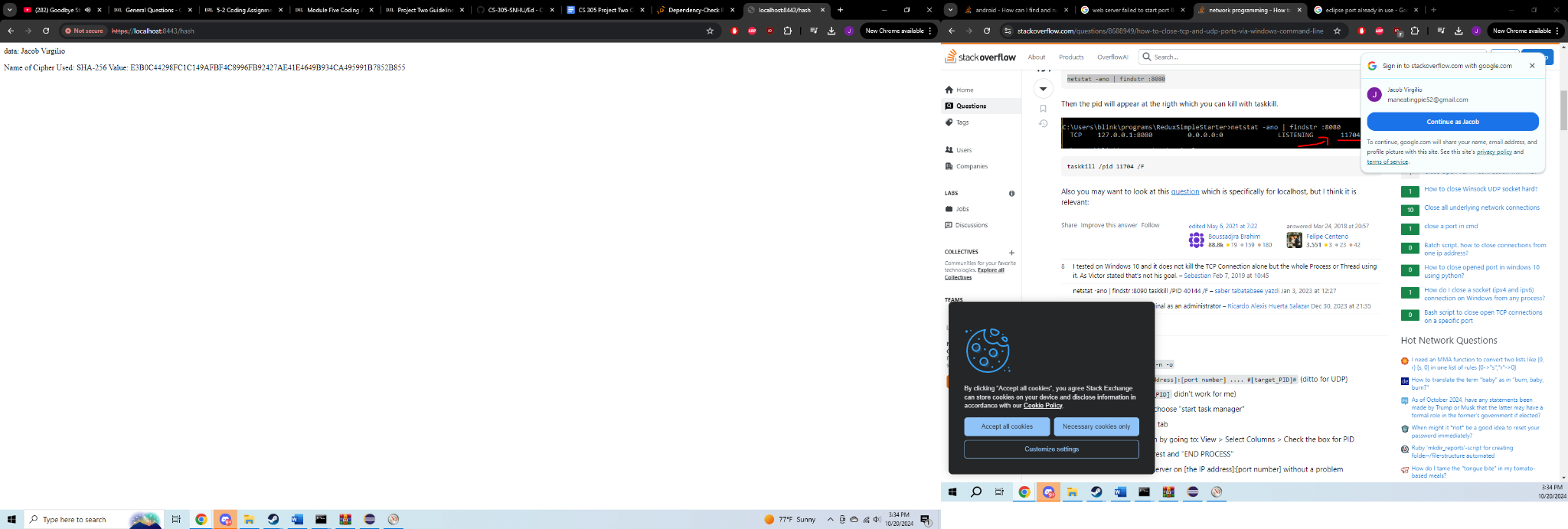
Submit a screenshot of the checksum verification in your practices for secure software report. The screenshot must show your name and a unique data string that has been created.



## Secure Communications

Secure Communications: Verify secure communication. In the application properties file, refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code. Once the server is running, type “https://localhost:8443/hash” in a new browser to demonstrate that the secure communication works.

Create a screenshot of the web browser that shows a secure webpage and include it in your practices for secure software report.

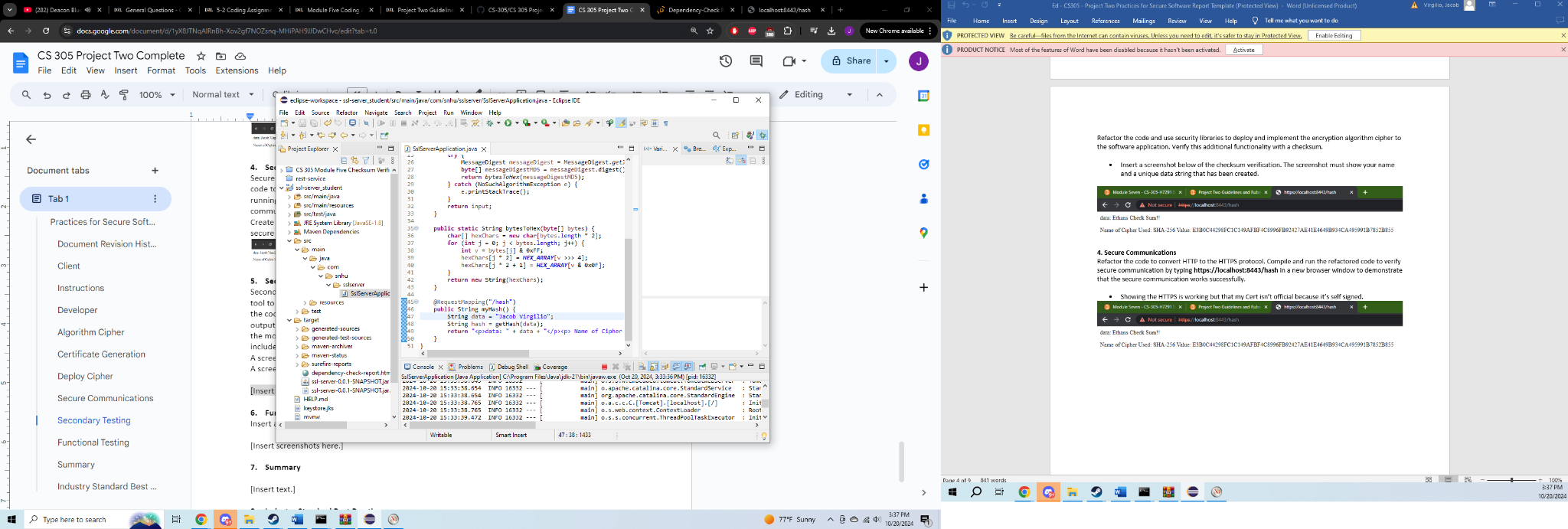


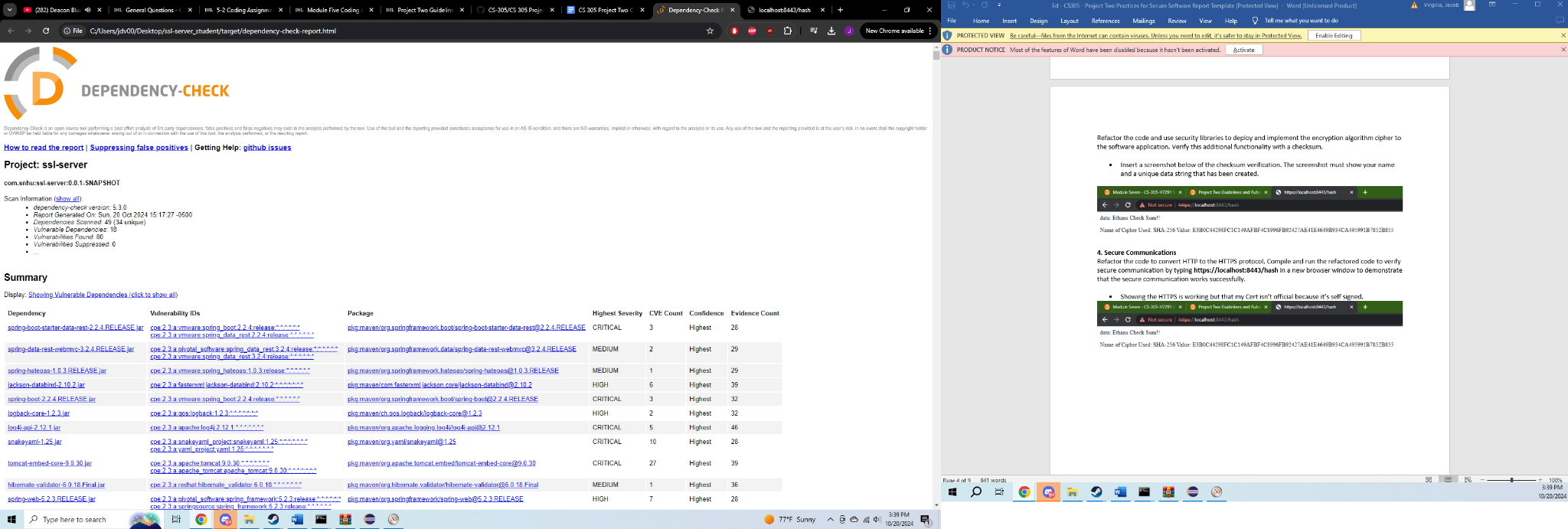
## Secondary Testing

Secondary Testing: Run a secondary static testing of the refactored code using the dependency-check tool to make certain the code complies with software security enhancements. You need to focus only on the code you have added as part of the refactoring. Complete the dependency check and review the output to make certain you did not introduce additional security vulnerabilities. Refer to the resources in the module’s Resources section for help on this action. In your practices for secure software report, include the following items:

A screenshot of the refactored code executed without errors

A screenshot of the report of the output from the dependency-check static tester

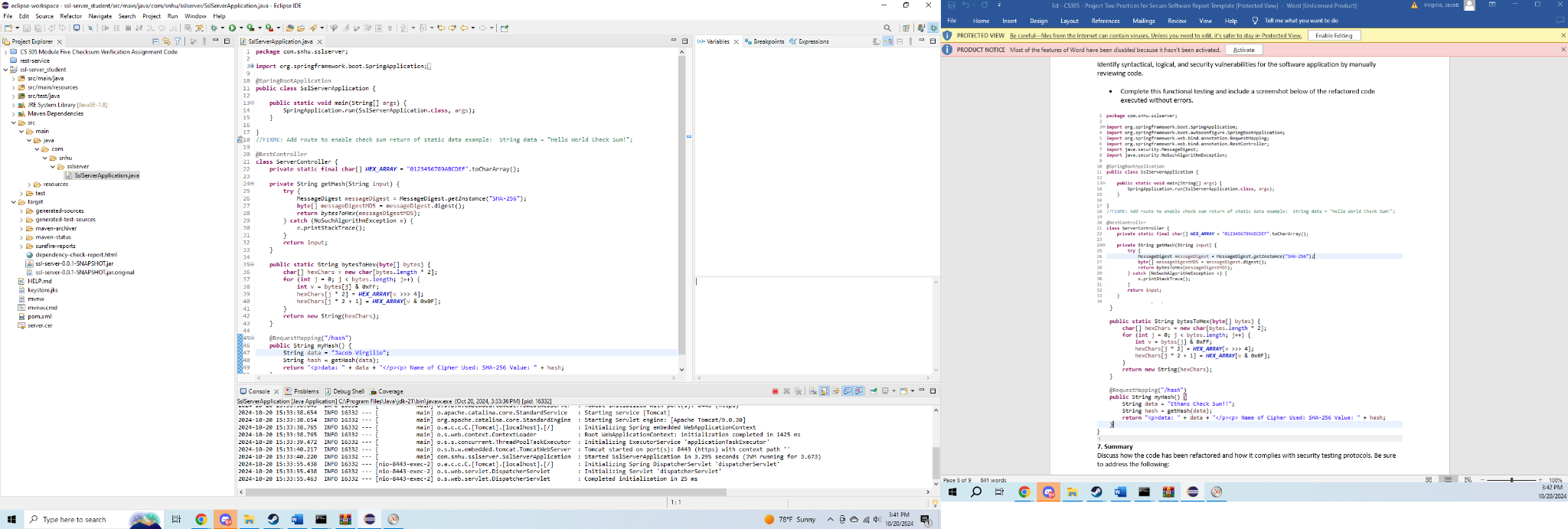




## Functional Testing

Functional Testing: Identify the software application's syntactical, logical, and security vulnerabilities by manually reviewing the code.

Complete this functional testing and include a screenshot of the refactored code, executed without errors, in your practices for secure software report.



## Summary

Summary: Discuss how the code has been refactored and complies with security testing protocols. In the summary of your practices for secure software report, be certain to address the following items:

Refer to the vulnerability assessment process flow diagram in the Supporting Materials section. Highlight the areas of security that you addressed by refactoring the code.

Discuss your process for adding layers of security to the software application.

In refactoring this code, RestController has been secured with a shared hash. Additionally, ServerController matches possible problems outlined in the assessment diagram. The SHA-256 cipher was chosen for its overall security, and very low chance of two combinations being the same. Additional added layers of security can be introduced by doing a monthly dependency check. This helps remove errors regarding out of date software.

## Industry Standard Best Practices

Industry Standard Best Practices: Explain how you applied industry standard best practices for secure coding to mitigate known security vulnerabilities. Be sure to address the following items:

Explain how you used industry standard best practices to maintain the software application’s existing security.

Explain the value of applying industry standard best practices for secure coding to the company’s overall well-being.

Ciphers were included in data, to keep data private and only in the places it needs to be sent and received. Additionally, any software security already present in the data was not tampered with, only additional security measures were added. The value of applying industry standard best practices for secure coding is very important for a company's overall well being. Keeping data secure helps avoid breaches not only for the company, but also for any customers who have worked with or alongside them. This can help drastically reduce the cost of maintaining systems and preventing cybercrime.