****

# CS 305 Project Two

**Practices for Secure Software Report**

**Table of Contents**

[Document Revision History 3](#_heading=h.1fob9te)

[Client 3](#_heading=h.3znysh7)

[Instructions 3](#_heading=h.2et92p0)

[Developer 4](#_heading=h.tyjcwt)

[1. Algorithm Cipher 4](#_heading=h.3dy6vkm)

[2. Certificate Generation 4](#_heading=h.1t3h5sf)

[3. Deploy Cipher 4](#_heading=h.4d34og8)

[4. Secure Communications 4](#_heading=h.2s8eyo1)

[5. Secondary Testing 4](#_heading=h.17dp8vu)

[6. Functional Testing 5](#_heading=h.3rdcrjn)

[7. Summary 5](#_heading=h.26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **2/26/2022** | **Renaldo Musto** |  |

## Client



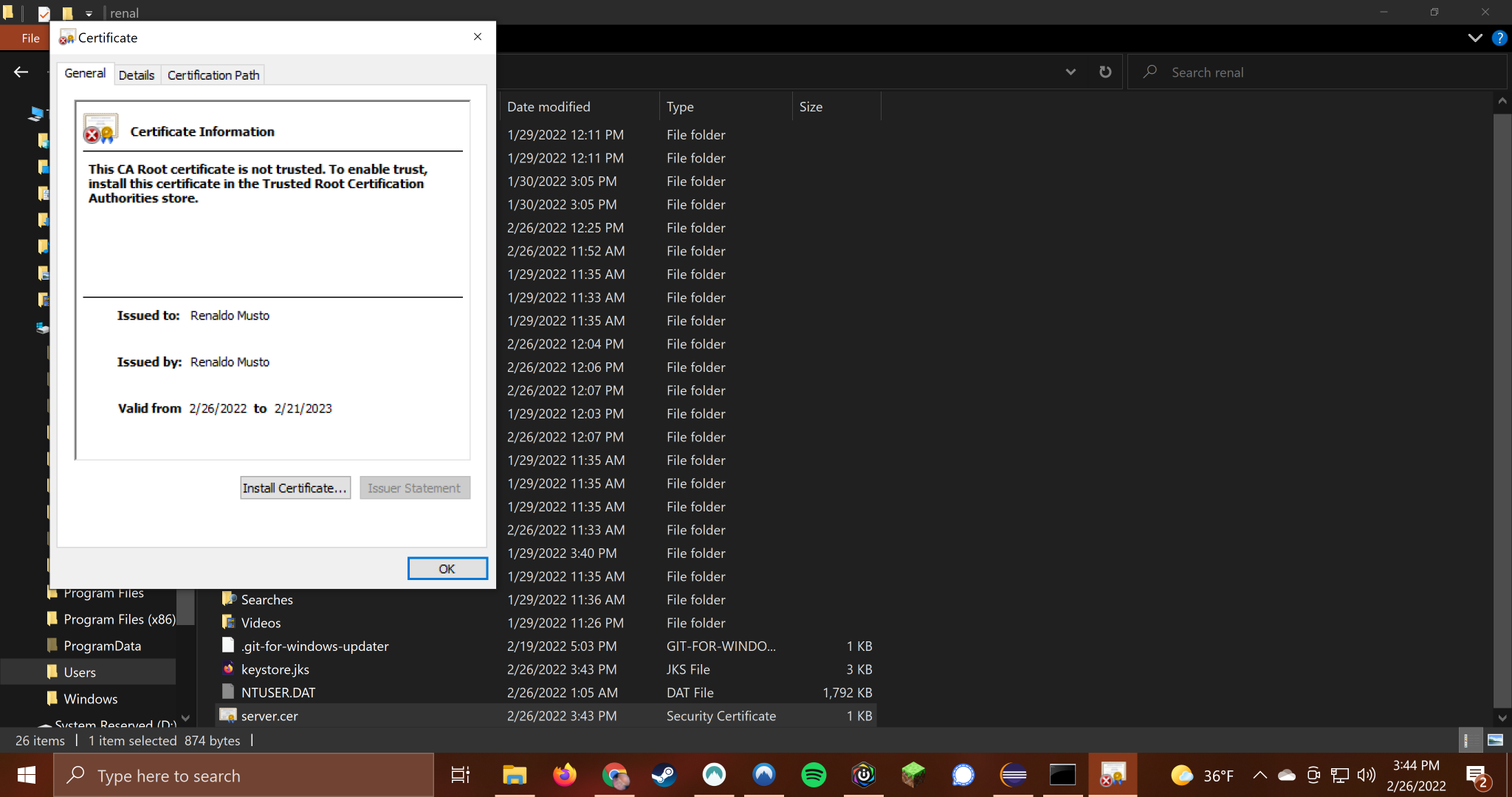
## Developer

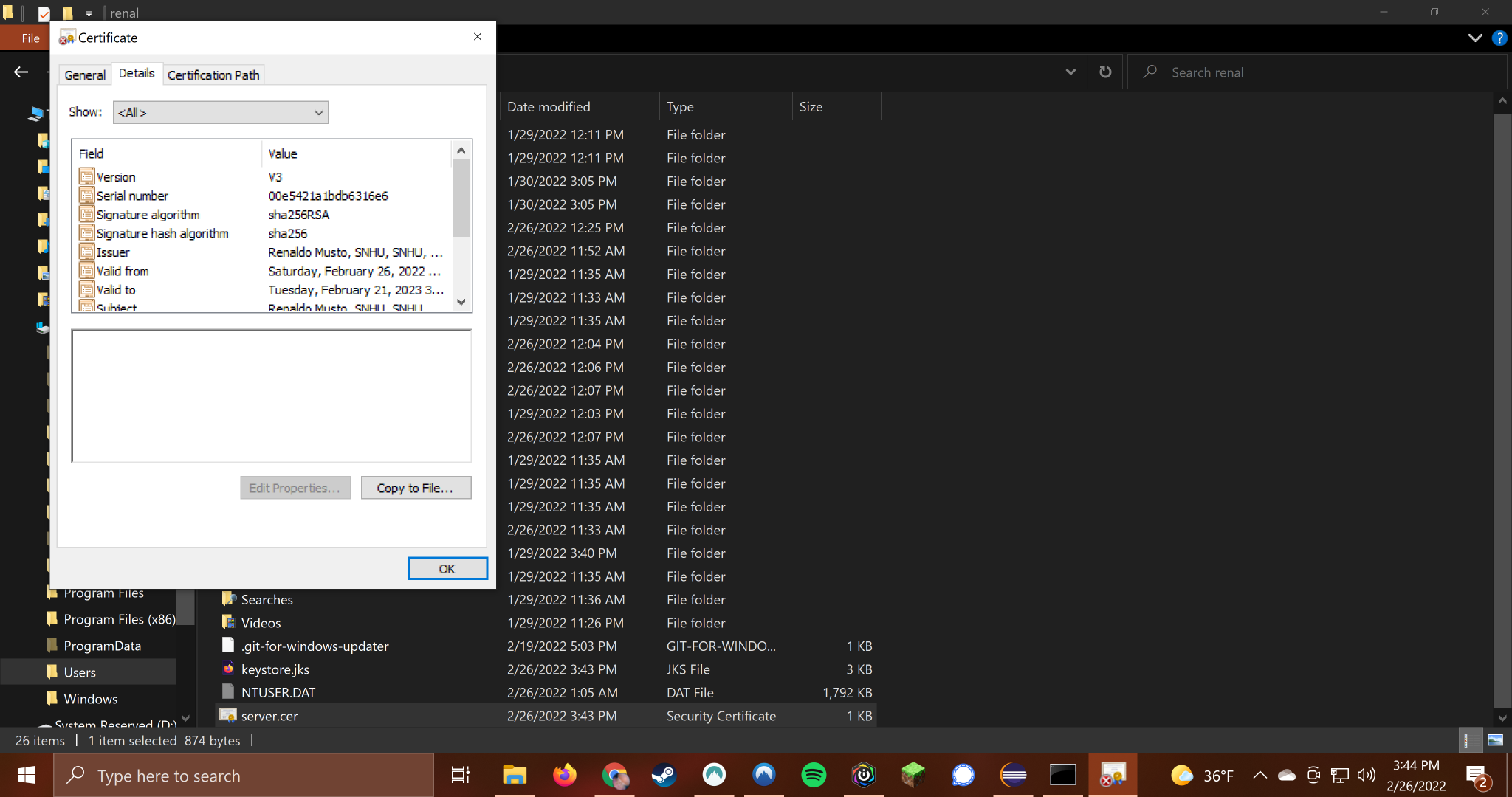
Renaldo Musto

## 1. Algorithm Cipher

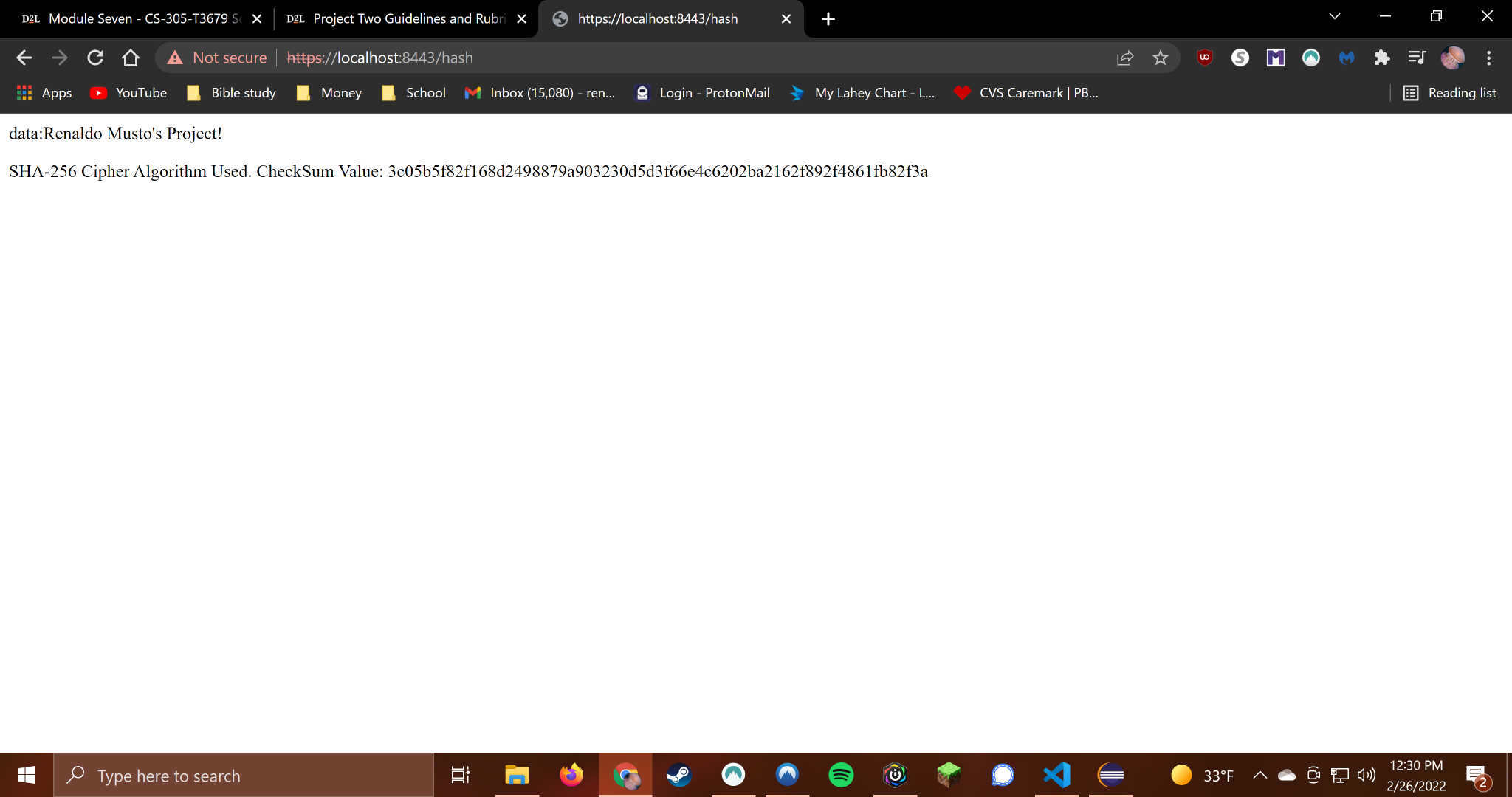
The encryption algorithm cipher that I chose to use for this assignment is the SHA-256 algorithm. In this case we are using what is called asymmetric cryptography. This means that the hashing algorithm works by using both a public and private key and that it is mainly used for signing rather than encrypting. In other words the original data string is not recoverable from the hash value. It is simply used as a means of verification that our data wasn’t corrupted. The bit level which is in the name of this cipher is 256 this means that the key we generate will be 256 bits long. This is currently the largest bit length we have for ciphers and therefore is the most secure. SHA stands for secure hash algorithm and the SHA-256 cipher is actually part of the second generation of this type of cipher. The SHA-1 family has been broken at this point so we use an algorithm part of the SHA-2 family which in this case I chose SHA-256.

## 2. Certificate Generation





## 3. Deploy Cipher



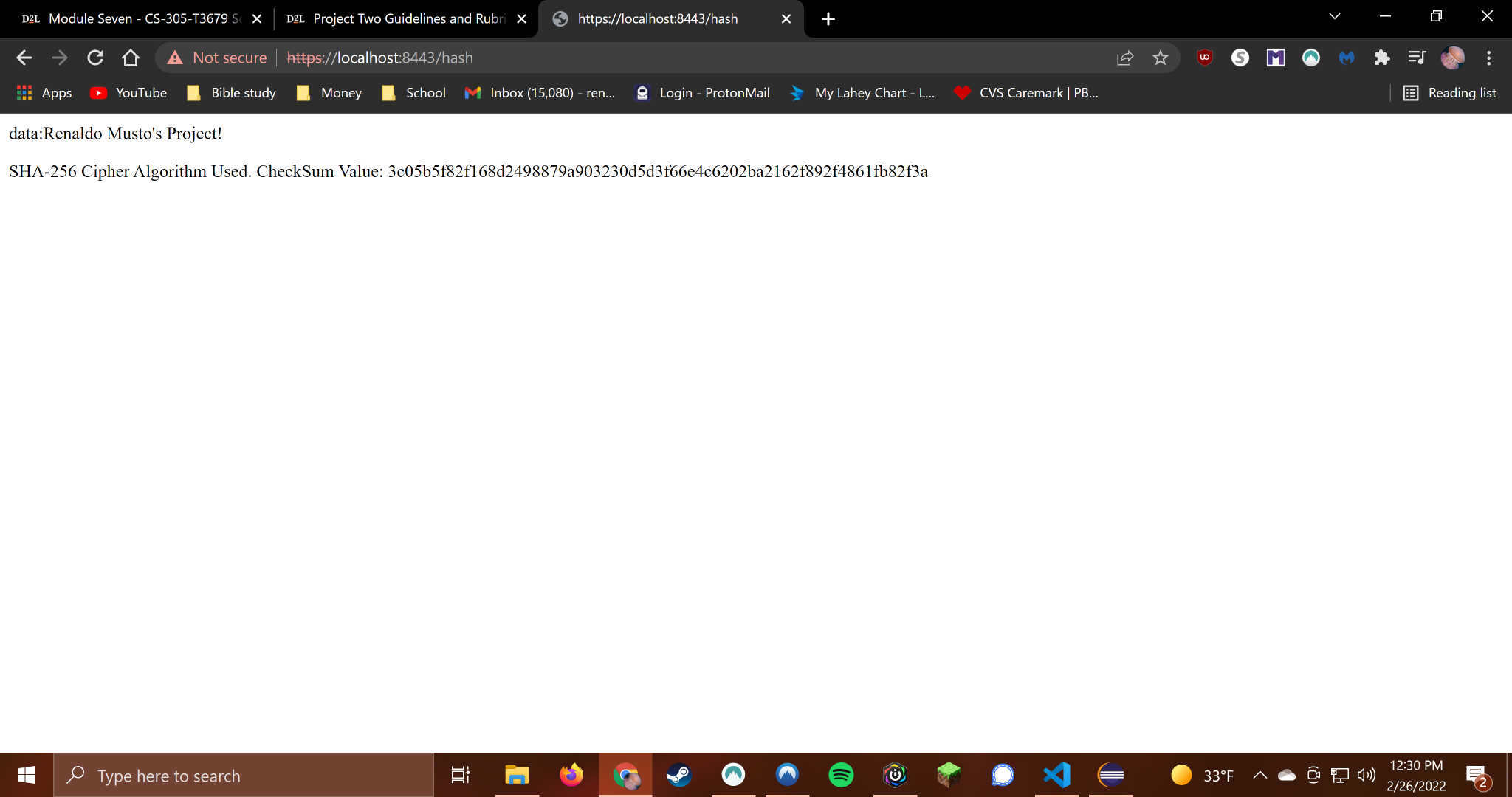
## 

## 

## 

## 

## 4. Secure Communications



## 

## 

## 

## 

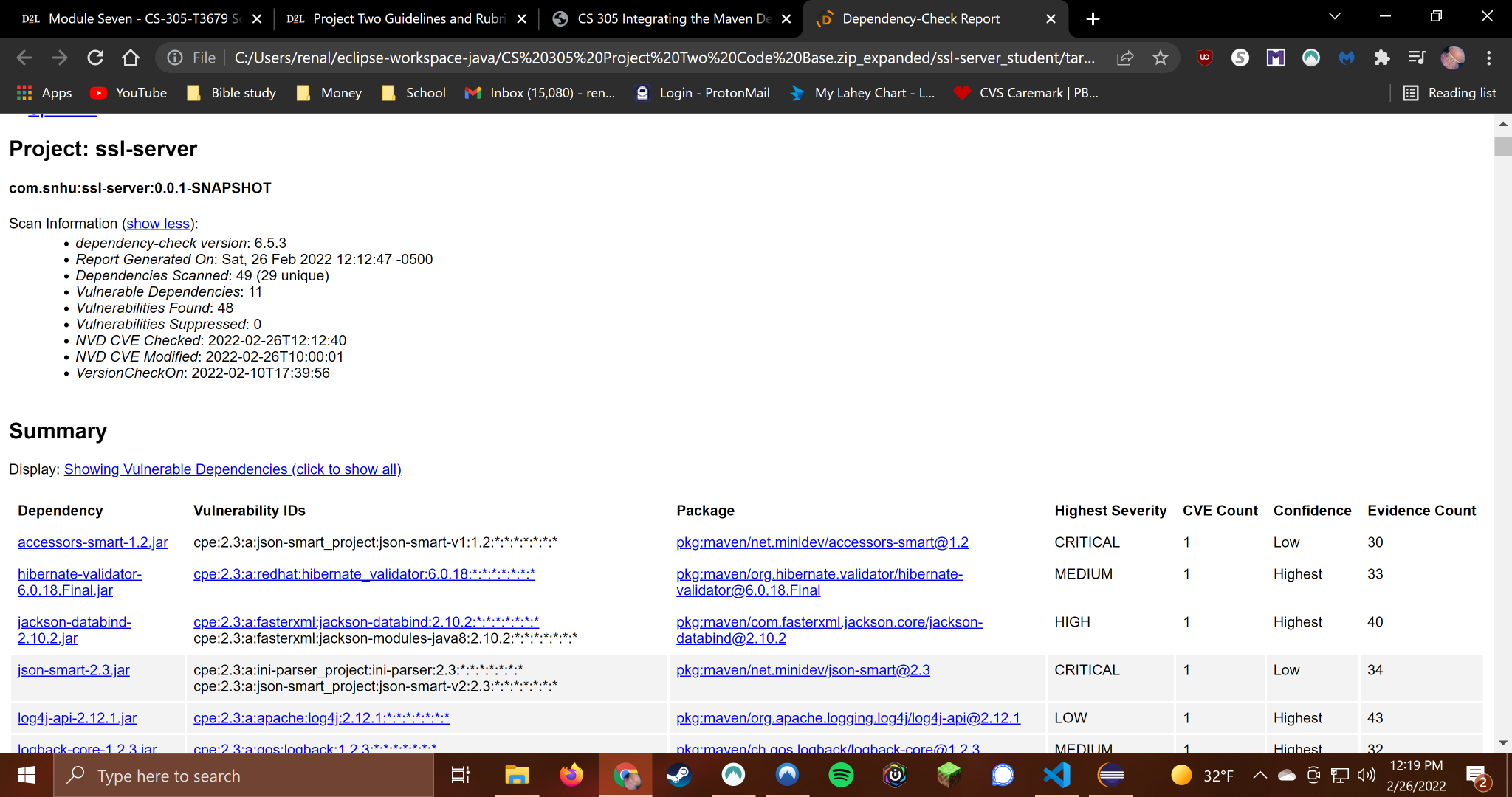
## 

## 

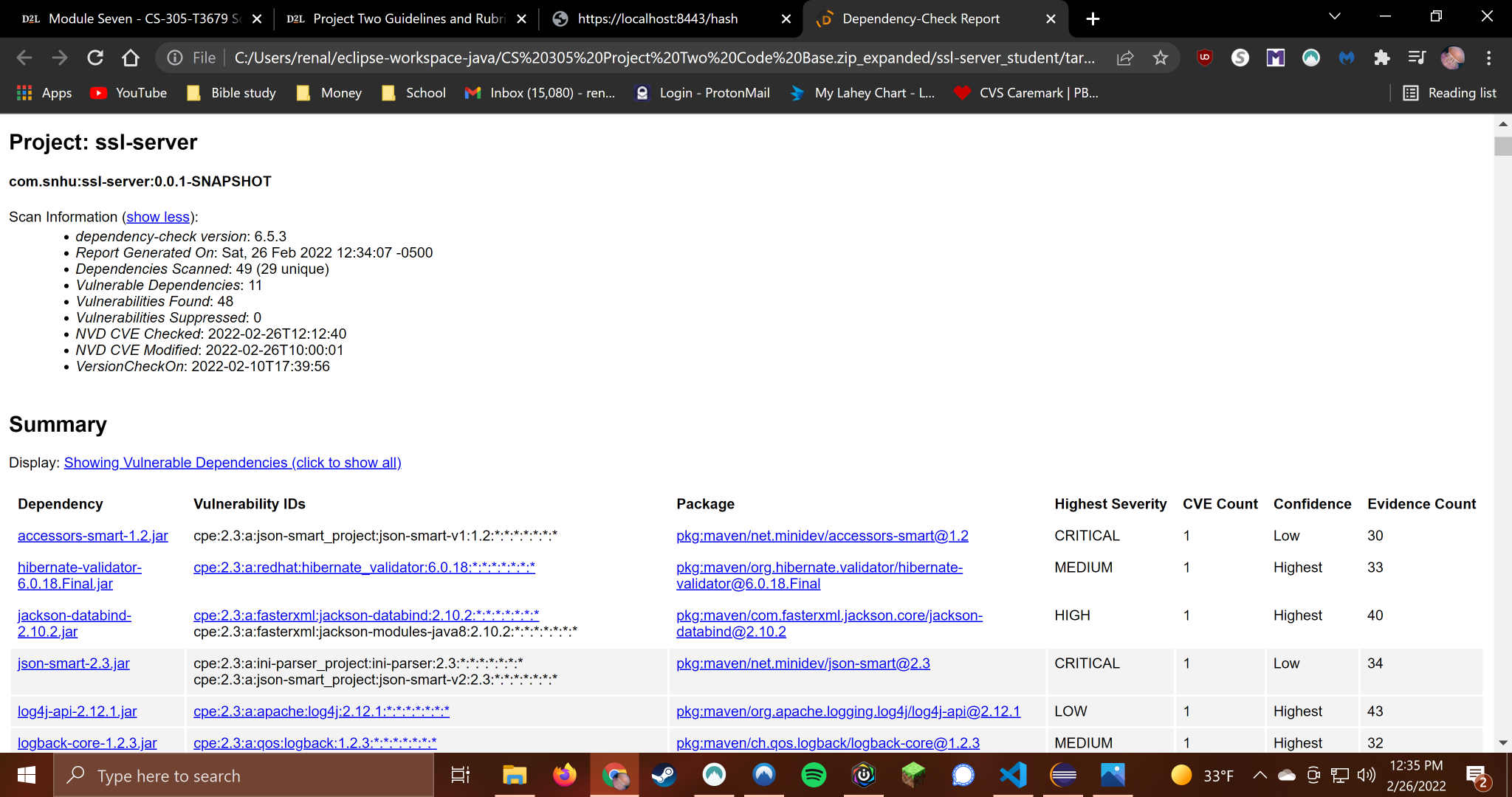
## 

## 5. Secondary Testing

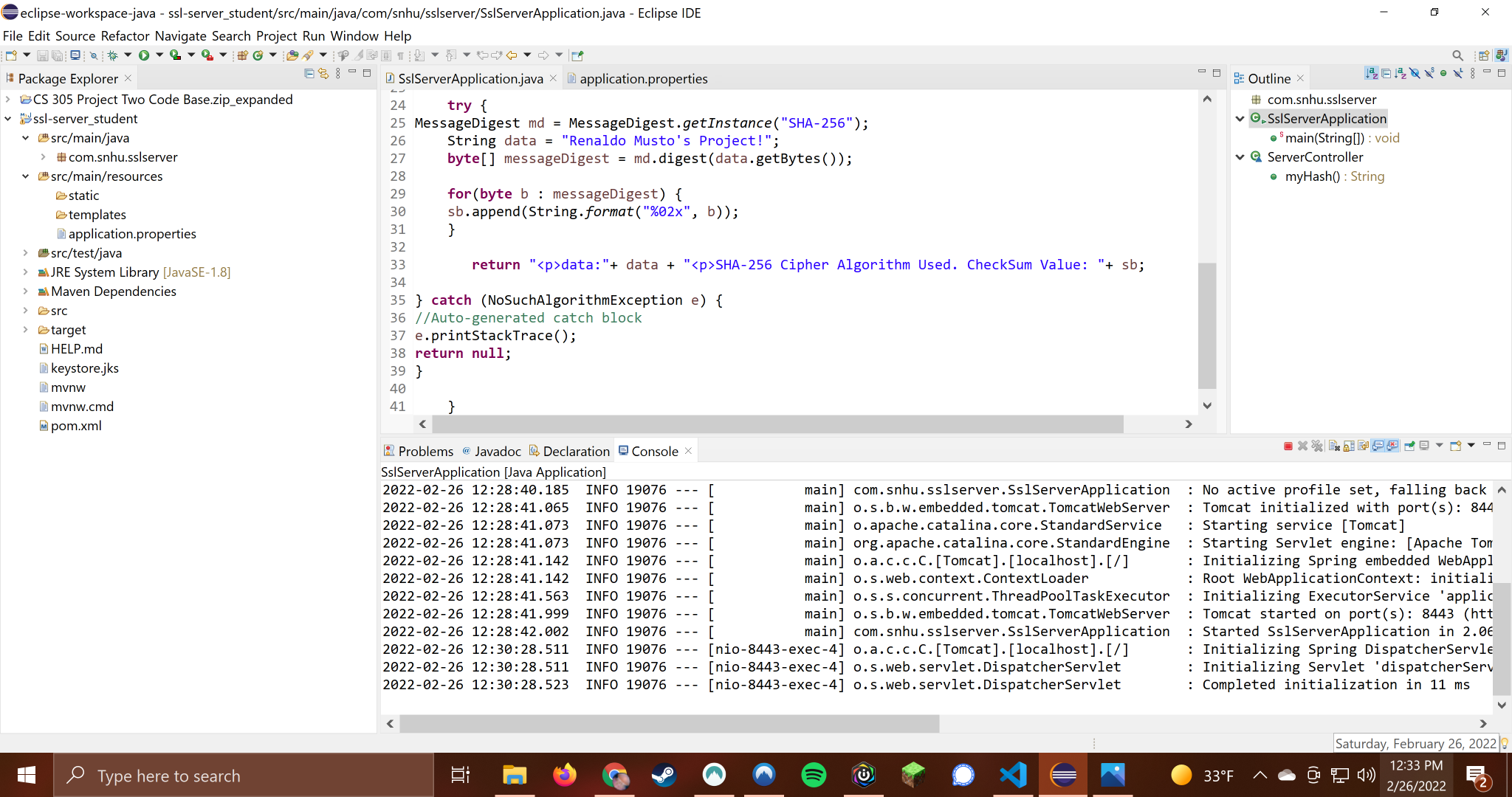
**First dependency check before code refactoring**



**Second dependency check after refactoring (no vulnerabilities introduced)**



**Refactored code executed without errors**



## 

## 

## 

## 

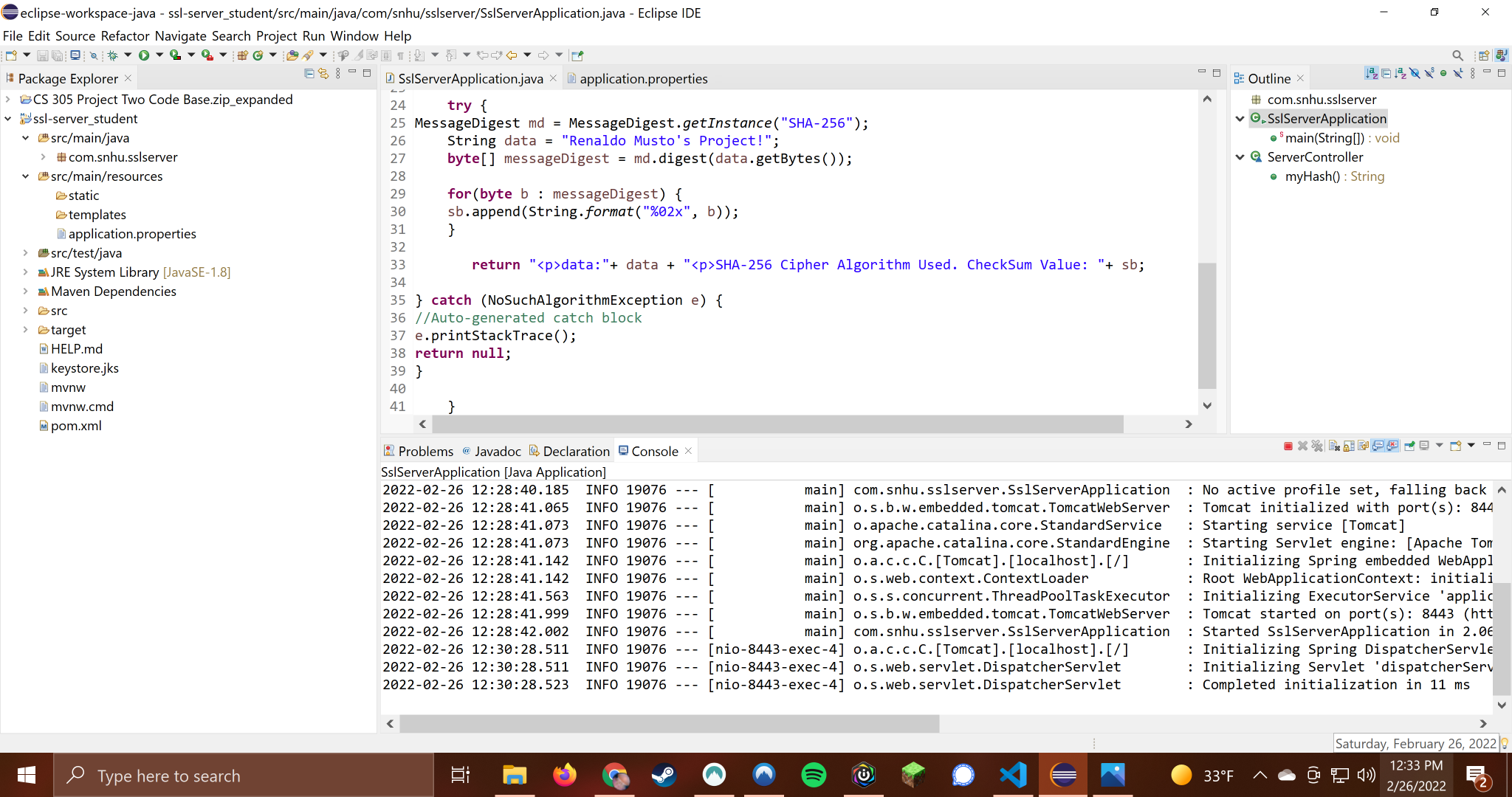
## 

## 

## 

## 

## 6. Functional Testing



After manually reviewing my code I did not notice further security vulnerabilities or errors and it was still able to run without errors or warnings.

## 

## 

## 7. Summary

When looking over the vulnerability assessment diagram I noticed that I was able to touch on several of the aspects displayed and in my opinion I covered everything necessary for this particular application. Firstly I would like to go over the secure API interactions aspect. I used a well known and secure framework of Spring to work in. I then made sure to use RESTful design practices to create a route for the data to make it to the web browser. Next I covered cryptography. I deployed the sha-256 algorithm cipher to encode my data string which also in turn assures us that the data was not corrupted on it’s way from the server to the client. I also made sure to ensure secure client/server interactions by utilizing an https connection as well as generating a certificate for the site so I can know that I am visiting the site I created. The code runs without errors and we have a NoSuchAlgorithm exception to catch any potential errors with bad ciphers if the code were to be refactored in the future. I used the minimal amount of functions and kept everything as simple as possible while maintaining conventional variable naming in line with industry best practices. My process for improving the security of a program is to first give it a manual review and see where I think any improvements need to be made. In this case I noticed that there was no certificate, https connection or checksum setup so I saw that as a major issue and sought to fix that all immediately. The company will greatly benefit from the security improvements because it will help prevent data breaches that are often very costly in terms of fines, settlements and a company's reputation among consumers. I think that one of the best practices for maintaining the security of this code would be to run regular dependency checks to ensure that no new vulnerabilities arise. The libraries that are used in this code as well as java itself should be regularly updated as well to ensure no we always have the best and most up to date defenses against attackers.

References

Manico, J., & Detlefsen, A. (2014, September). *6 Protecting Sensitive Data*. O'Reilly Online Learning. Retrieved January 21, 2022, from https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/ch06.html#ch06lev2sec7