Southern New Hampshire University

Module 7: Project 2

Jacob McPherson

Vivian Lyon

CS-305-T6614 – Software Security

8/14/2022

****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **8/9/2022** | **Jacob McPherson** | **Cypher research** |
| **1.2** | **8/10/2022** | **Jacob McPherson** | **Cert gen/Checksum** |
| **1.6** | **8/11/2022** | **Jacob McPherson** | **Maven test/functional** |
| **2.0** | **8/12/2022** | **Jacob McPherson** | **Summary/Review** |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Jacob McPherson

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

* 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 vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

For the best results in security for our client I would recommend the AES-256 encryption cipher with the SHA-256 hash function. Over the years numerous ciphers have been tested and AES has shown the most success to date. Ruta Rimkiene remarked on her article in cybernews that no successful attacks have been recorded so far.

AES works by running a block of data through round after round of encryption, making it virtually impossible to revert back without the proper cipher key. The AES cipher works with symmetric keys, meaning both parties will need to have the cipher key in order to understand information sent between them. While getting the cipher key to the other party securely can be difficult, if the client helps or creates the account for the customer than giving them the cipher key can be done during that process.

The SHA-256 hash function serves as a good authenticator for tamper checking and password encryption when users need to login to the service. Hash functions encrypt 256-bit messages and data into unique keys with a consistent length of 256-bits. Simplilearn on their SHA-256 tutorial page talks about using the function for file verification and password hashing the most. SHA creates a unique key that is attached to a file when it is originally created and can be decrypted and used to verify no data was changed before the second viewing. Password hashing servers to keep stored passwords and information safe in the event a hacker managed to get a hold of that data from the server directly. When encrypted the hacker will have only gotten a jumbled mess instead of the actual data.

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

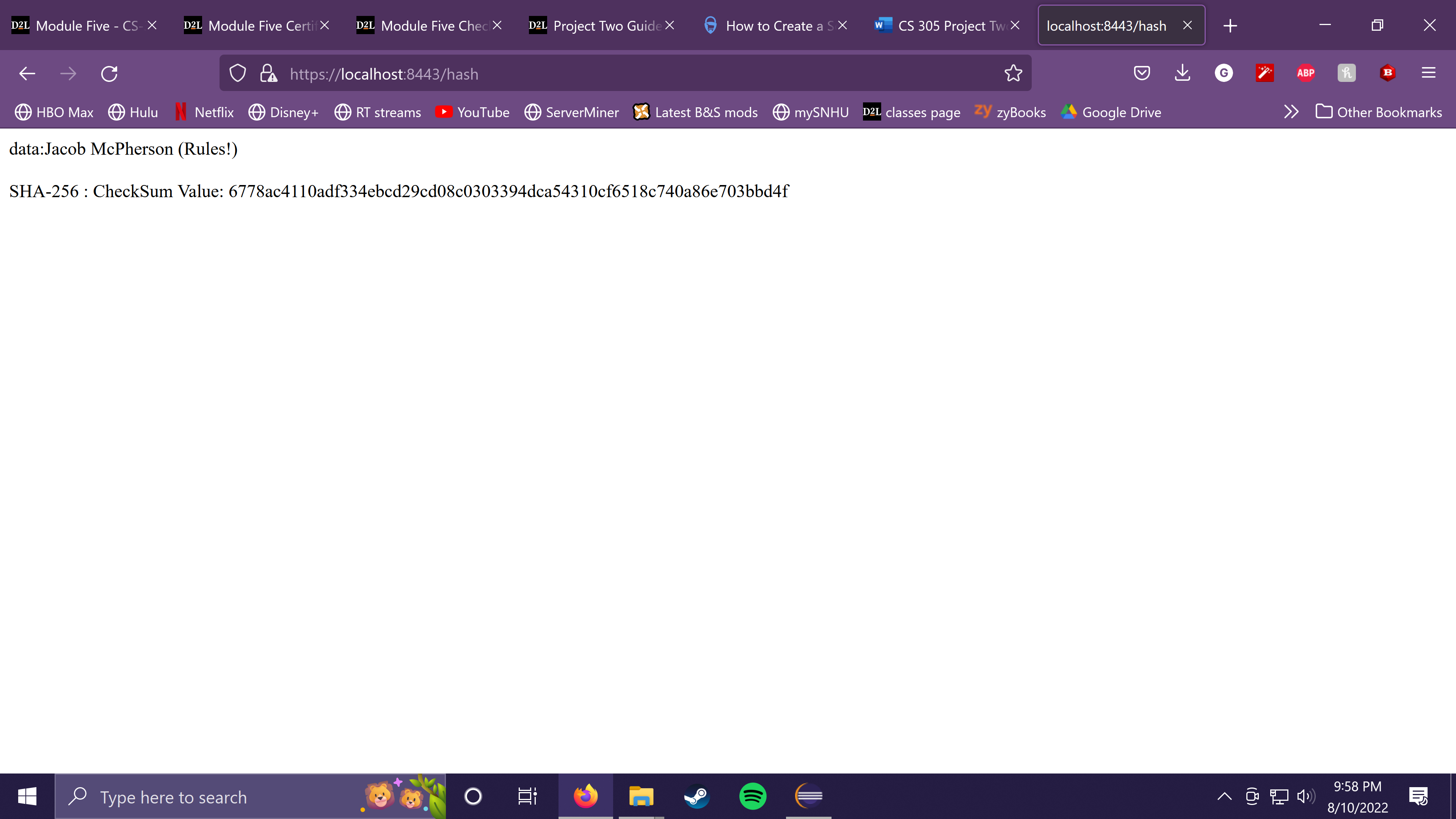
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

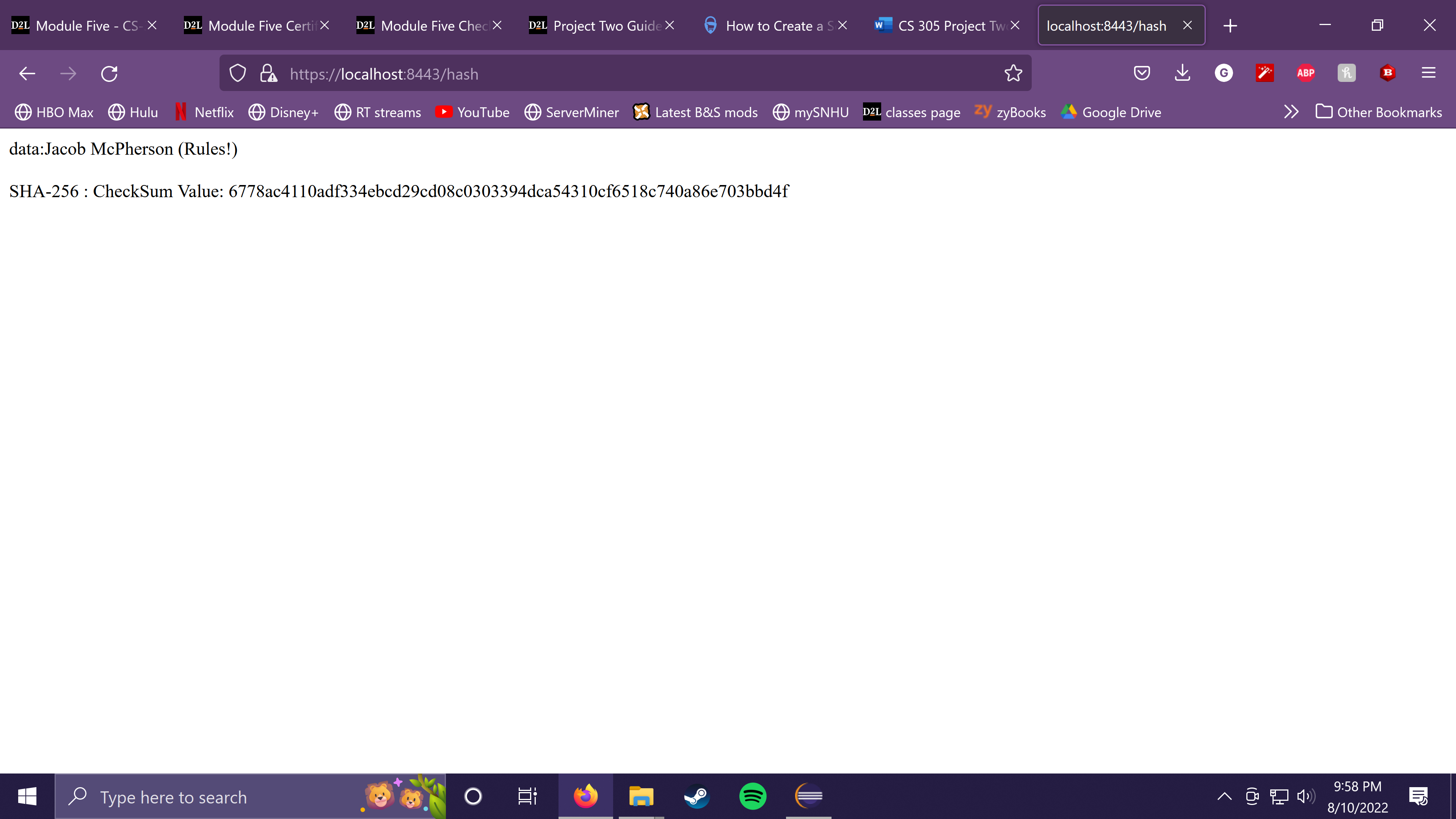
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

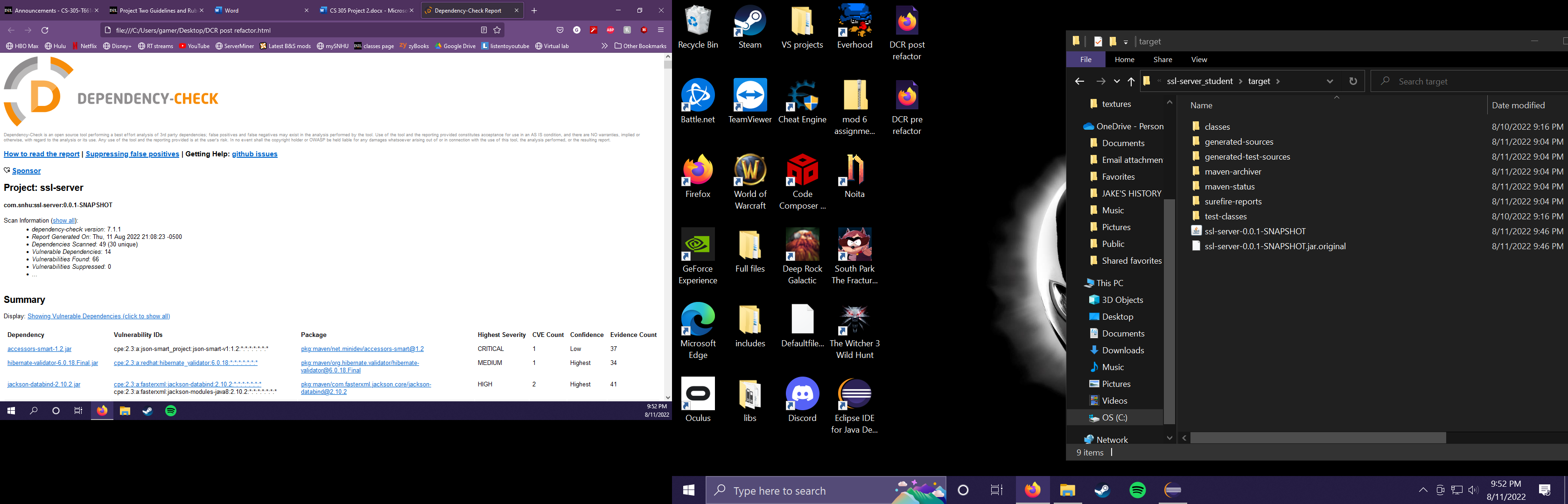
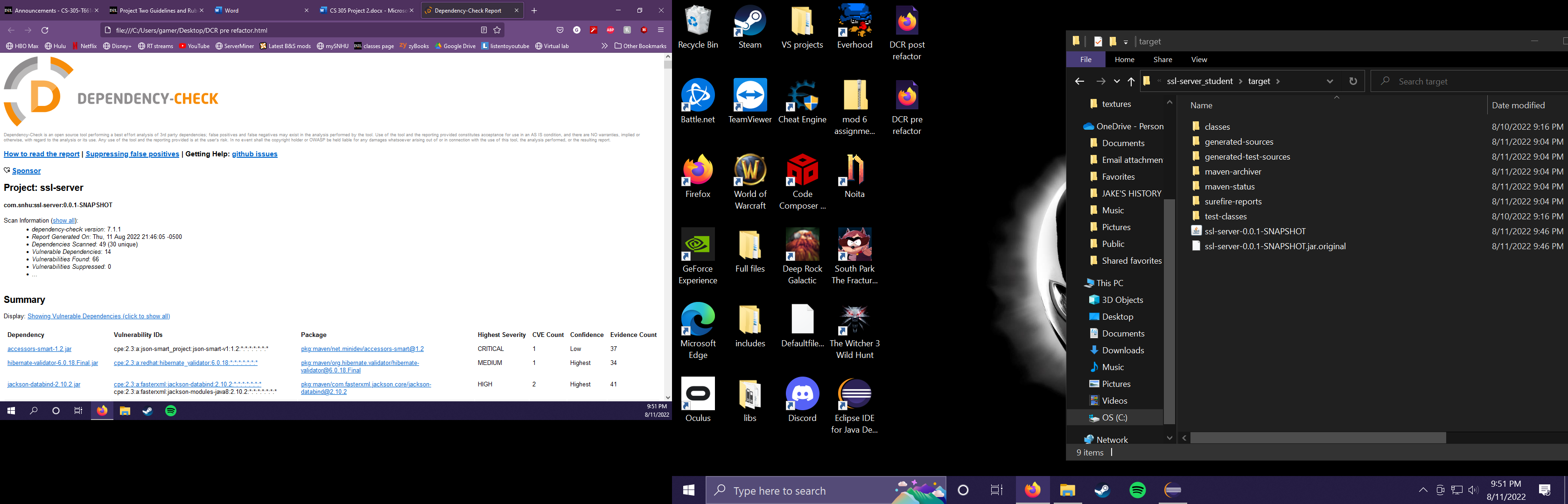
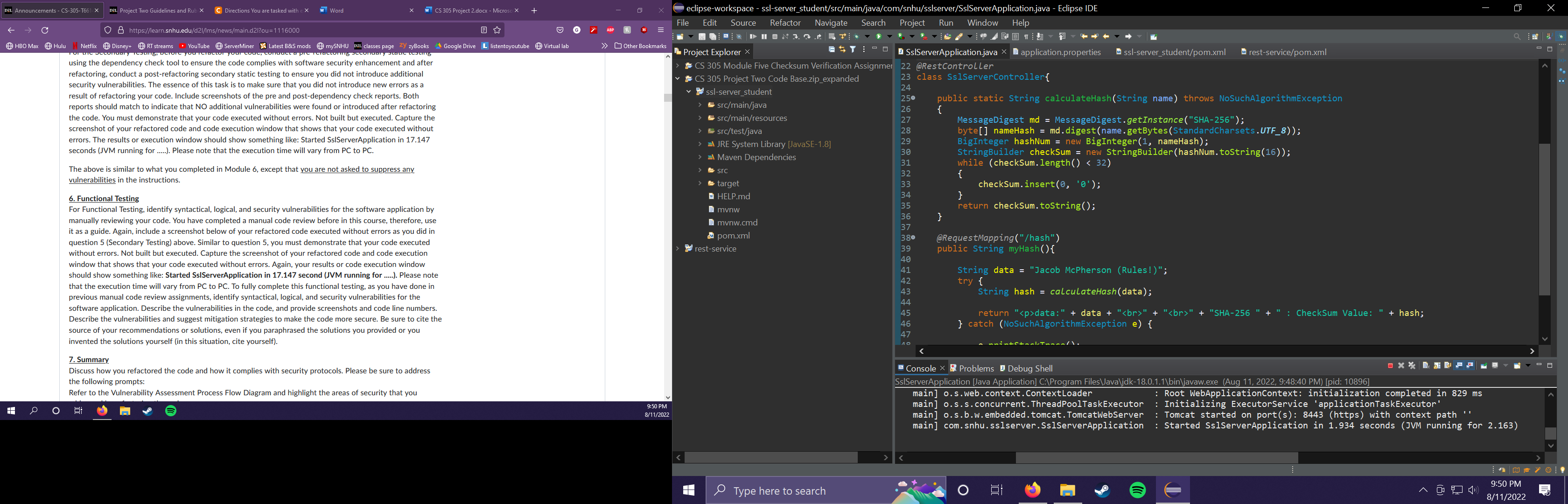
* Insert a screenshot below of the web browser that shows a secure webpage.



## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report



## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.

 Looking over the refactored code I couldn’t help but notice I had created public methods and string that could have been altered by other classes in the software. While right now that isn’t an issue since there are no other files, however if this were integrated elsewhere, then it would have been vulnerable to such modification.

## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

Looking at the VAP diagram I have found that I focused mainly on the cryptography, client/server, code errors/quality, and encapsulation. Cryptography is focused on when creating the checksum and sending out the data string to a local host URL. When creating the checksum I used the SHA-256 hash function to encrypt the message I was sending out.

The creation of a security certificate and passing it through the application to create an HTTPS connection helps to establish a way for future programmers to import other more reputable certificates and allow for more connections or information to reach the server. These methods had to be looked over multiple times to ensure no vulnerabilities were left such as, open variables, public functions, and overlooked error catches.

When determining how I could add security to the application I reviewed my past works and investigated the basics of AES and SHA-256 given their current reputation. Once I gathered the information I needed, I recreated the checksum from my previous work in module 5 and modified the functions to be private. The encryption for information I put in place should allow the company to alter the code slightly to encrypt any messages or information they need to send out to other websites. The security certificate check should also keep the application from sending anything to websites that don’t have a verified certificate.

The client should be able to maintain the security put in place by keeping the libraries and hash function up to date. Many of the vulnerabilities stem from the outdated libraries and functions being used still in the software. An improvement would be to obtain or create a non-selfsigned certificate to verify the connection being established by the software. With this the client should be well on their way to a much more secure program.

**References**

Simplilearn. (2022, July 11). *What is SHA-256 algorithm: How it works and applications [2022 edition]: Simplilearn*. Simplilearn.com. Retrieved August 9, 2022, from <https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm>

Rimkiene, R. (2022, April 21). *What is AES encryption and how does it work?* Cybernews. Retrieved August 9, 2022, from <https://cybernews.com/resources/what-is-aes-encryption/>

NIST. (2001, November 26). Advanced Encryption Standard. Virginia; NIST.