****

# 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** | **06/16/2023** | **Afahri Kerr** | **Initial Review** |

## 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

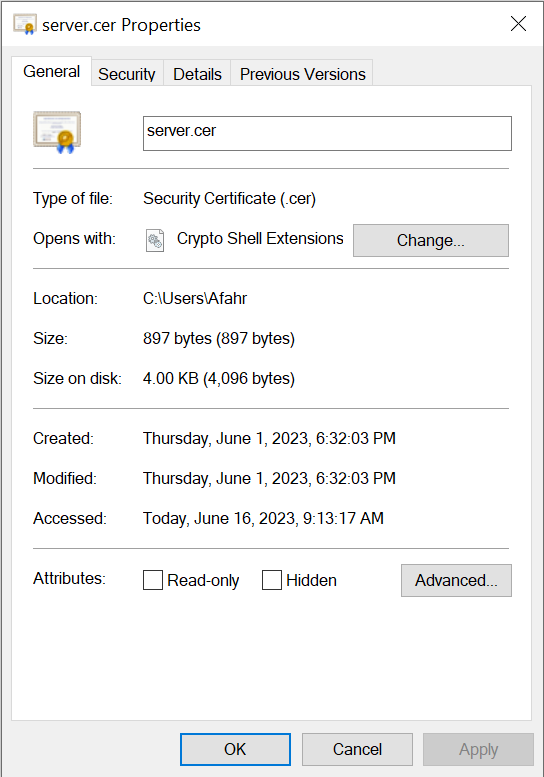
Afahri Kerr

## Algorithm Cipher

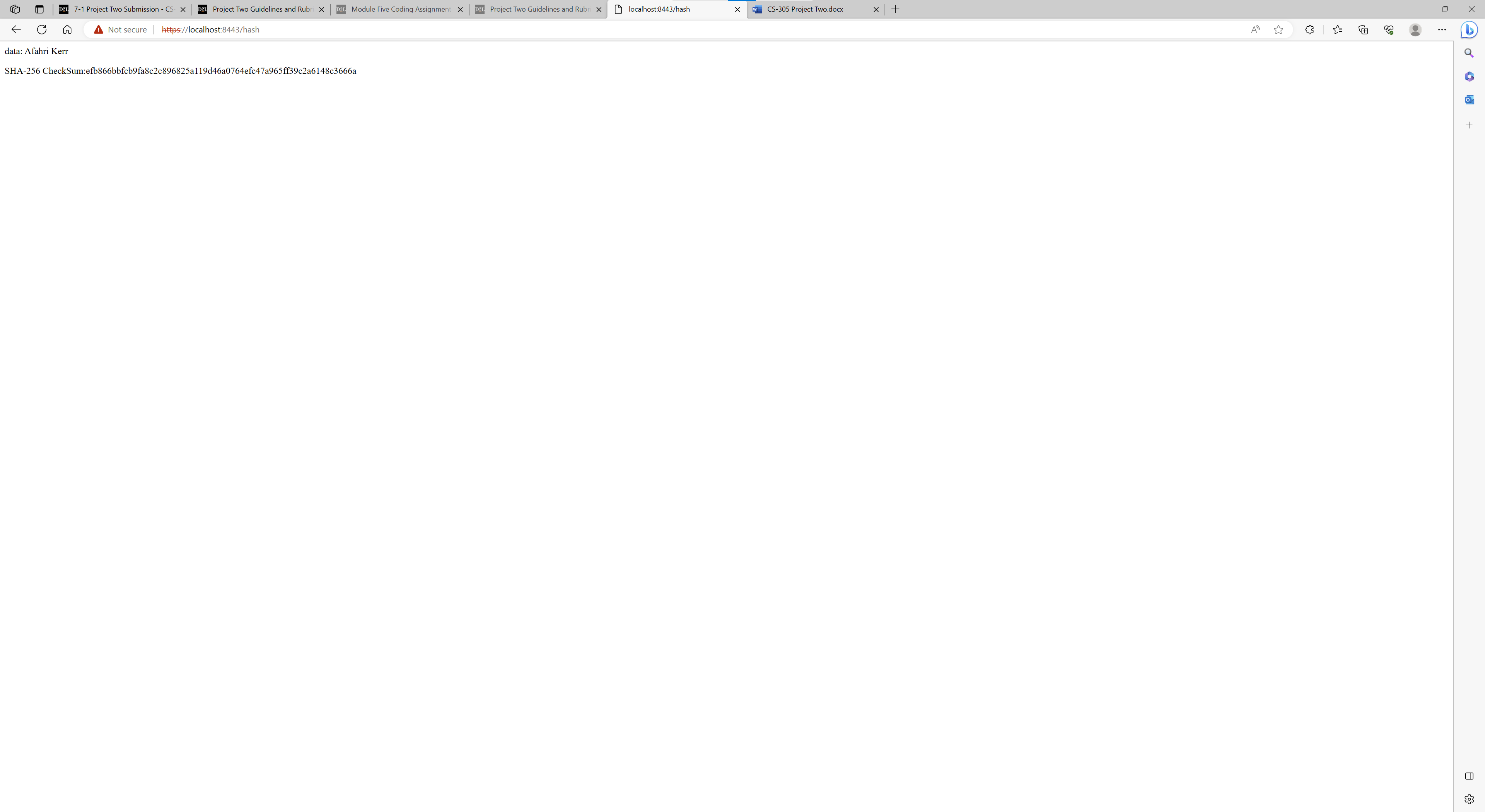
I recommend using the Advanced Encryption Standard (AES) with SHA-256 as the hashing function for this application. AES is an algorithm that allows you to encrypt sensitive information. It has been adopted by the U.S government and can be used with different key sizes such as 128, 192, 256, and 512 bits. I recommend the 256-bit key as it is harder to crack and I am unaware of anyone doing so. It is also currently the industry standard encryption protocol. This algorithm uses symmetric keys which means that it will use the same key for encryption and decryption. These keys generate random numbers in 256 bits. The hash function will ensure that the data blocks have not been tampered with.

Encryption has been used for many years. There is a form called Atbash that was used by the Hebrews in the bible around 500-600 B.C. Encryption has even been used by the Spartans and Julius Caesar. World War II is when cryptography became a well-known issue as the Nazis were using an encryption tool called the Enigma. It used an electromechanical rotor system to scramble letter inputs. DES (Data Encryption Standard) was developed in 1975 and used a 56-bit key but is obsolete today since computing power has increased. These keys can be cracked in hours using a brute force method. AES builds off DES and uses keys with greater bits.

## Certificate Generation

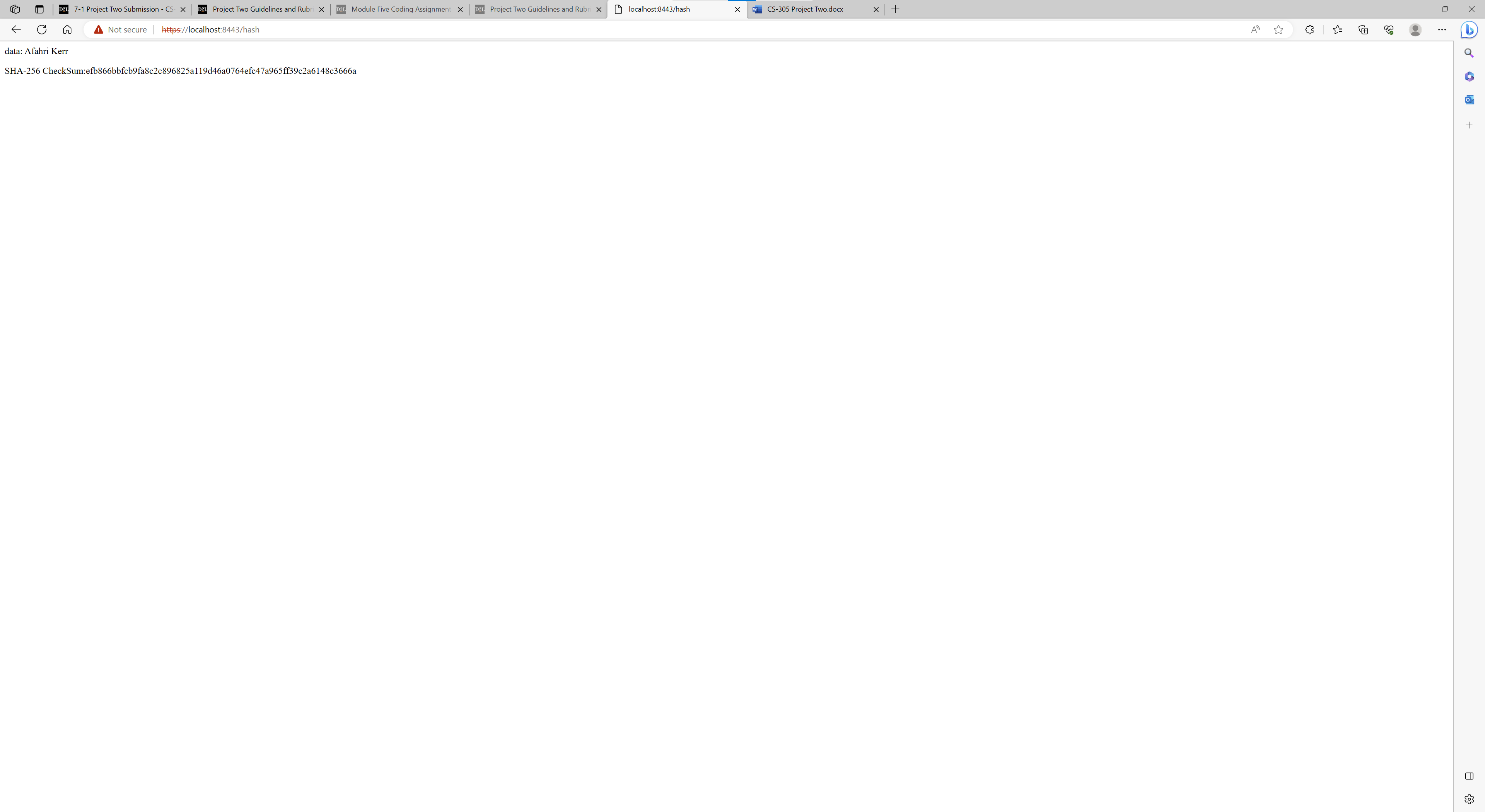


## Deploy Cipher

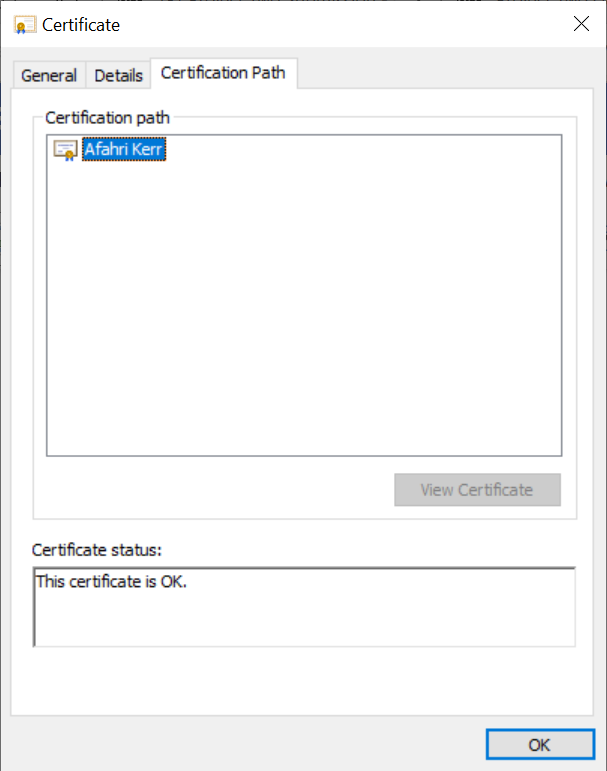


## Secure Communications

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

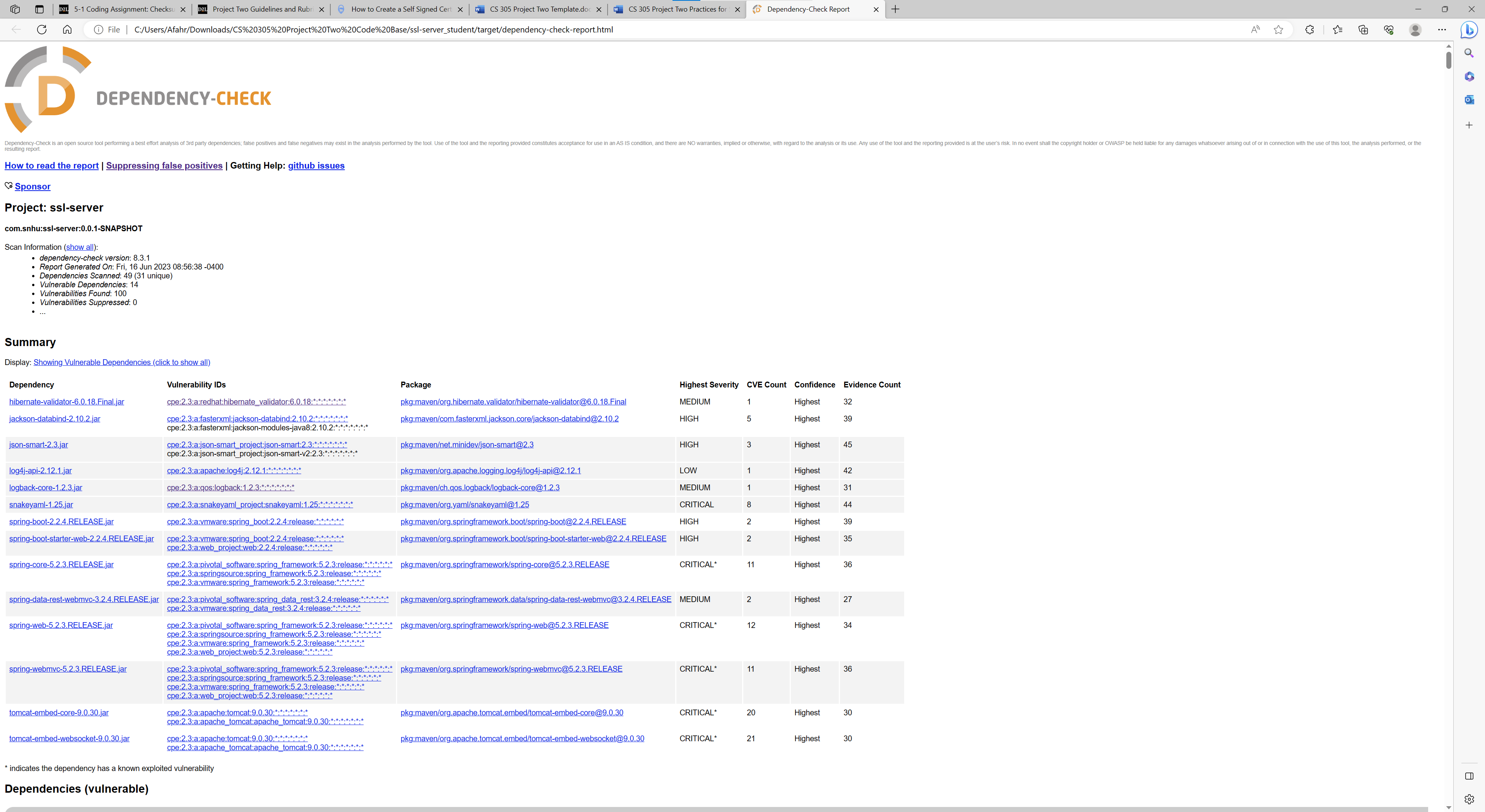


It kept saying not secure even after I added the certificate to my main certificate folder. Edge says it will do this for all self signed certificates but the site will not run unless it is under HTTPS.

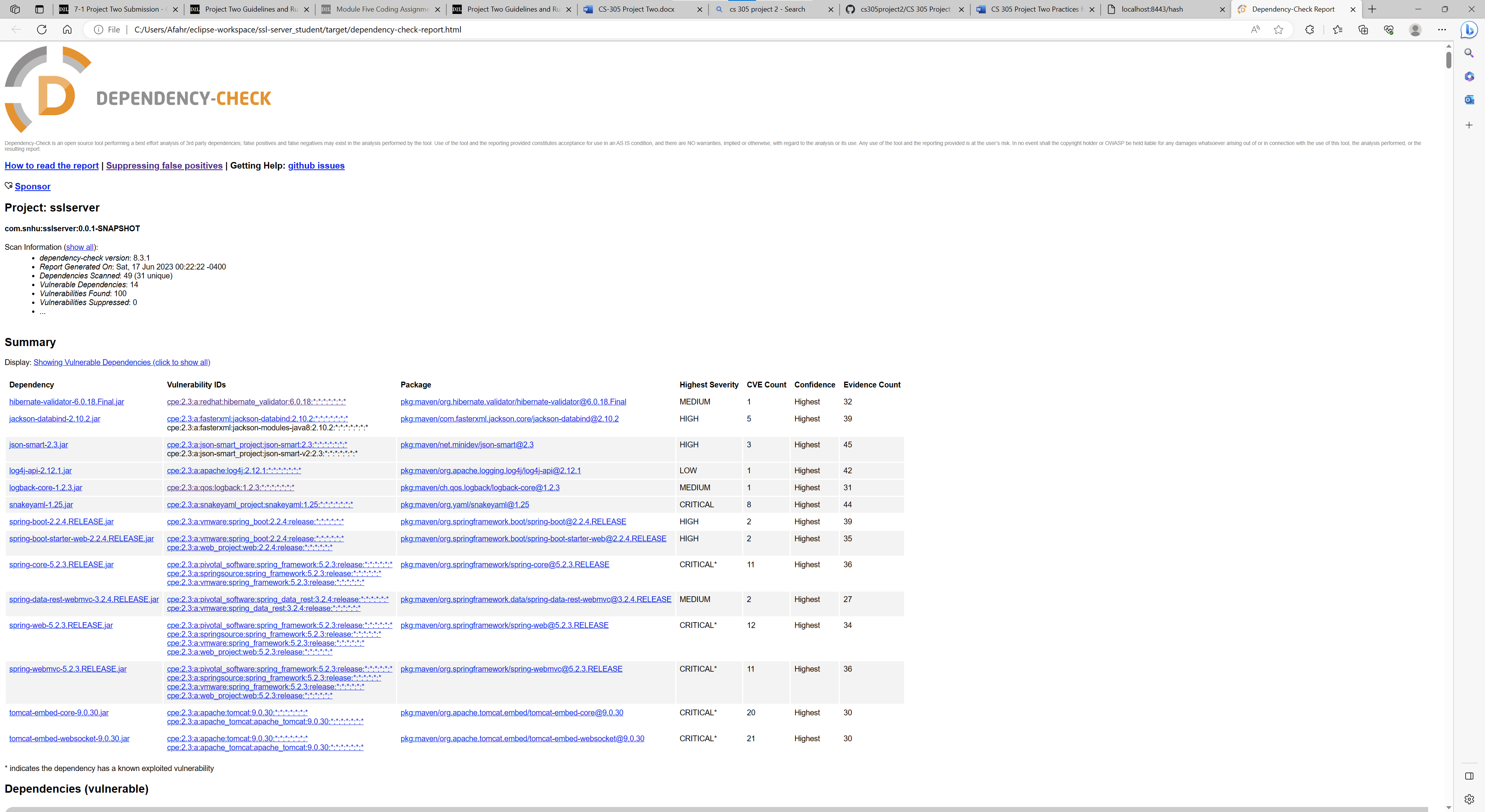


## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

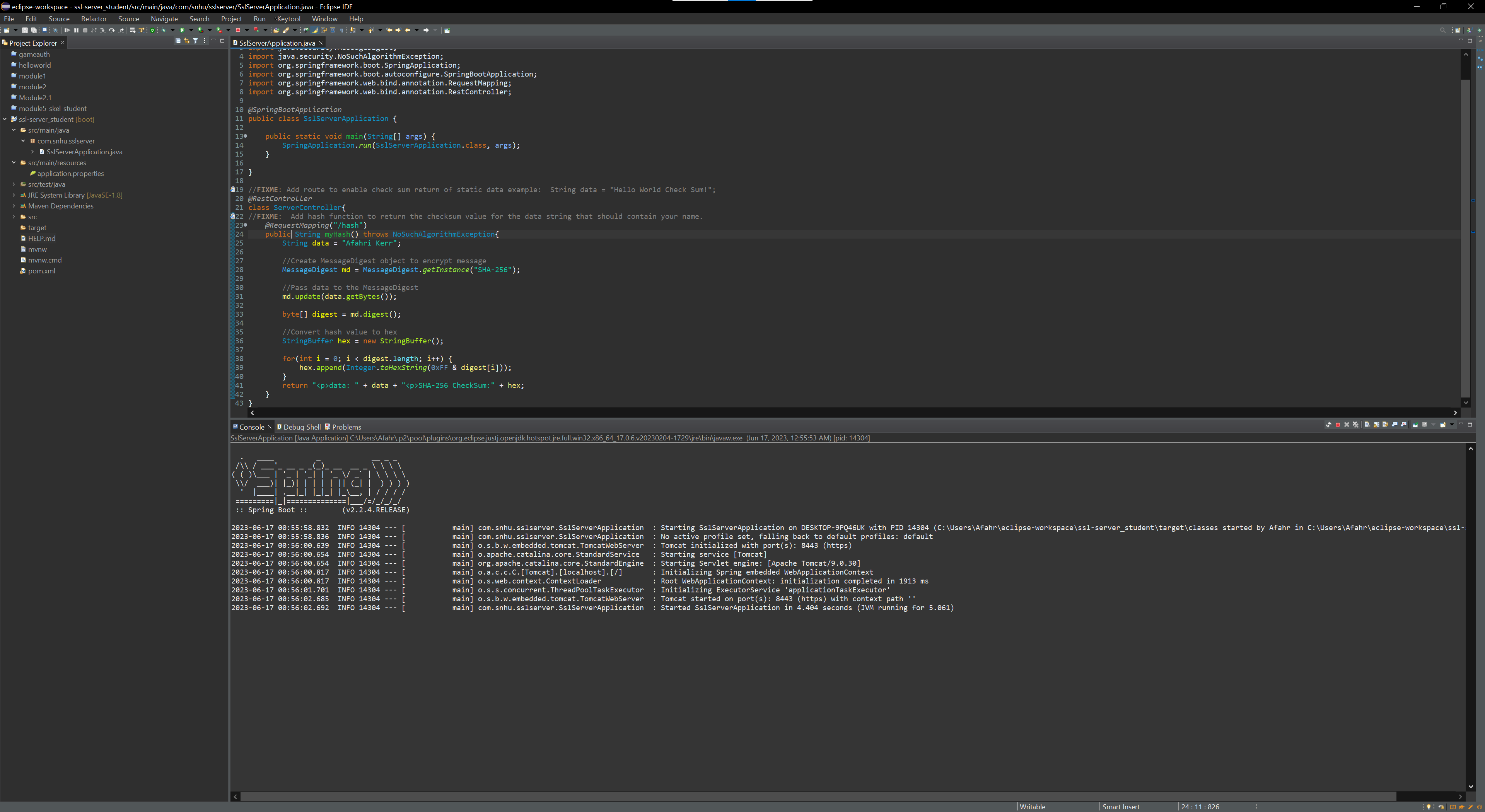


This was my original dependency report



This is my second scan showing that no extra vulnerabilities were detected.

## Functional Testing



## Summary

This code just demonstrates how to encrypt some data which in this case is just my name but it can be used to encrypt all types of data that may be sent to the server. The code was also signed with a certificate and encrypted so the site is now HTTPS compatible. This will ensure that the connection is secure if you trust the publisher. Finally, it was passed through the static analyzer to see all the vulnerabilities and make sure that no new ones were added.

## Industry Standard Best Practices

It should be further upgraded with a character limit to prevent DoS attacks and incorporate blacklisting to prevent SQL injection for when a database is added to collect information. Also, while it does have a certificate and encryption allowing it to work under HTTPS, the certificate should be issued by a CA instead of a self-signed one. It is encrypted by AES 256 which as of now has not been cracked so potential data should be safe even if the encrypted data is somehow stolen. This app has a long way to go to be ready for commercial use, but the encryption of data will help it get by current financial laws.