****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.30j0zll)

[**Client 3**](#_heading=h.1fob9te)

[**Instructions 3**](#_heading=h.3znysh7)

[**Developer 4**](#_heading=h.2et92p0)

[**1. Algorithm Cipher 4**](#_heading=h.tyjcwt)

[**2. Certificate Generation 4**](#_heading=h.3dy6vkm)

[**3. Deploy Cipher 4**](#_heading=h.1t3h5sf)

[**4. Secure Communications 4**](#_heading=h.4d34og8)

[**5. Secondary Testing 4**](#_heading=h.2s8eyo1)

[**6. Functional Testing 4**](#_heading=h.17dp8vu)

[**7. Summary 4**](#_heading=h.3rdcrjn)

[**8. Industry Standard Best Practices 4**](#_heading=h.26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **04/12/2023** | **Andrew Thomas** |  |

## Client



## Instructions

Submit these completed practices for a 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

Andrew Thomas

## Algorithm Cipher

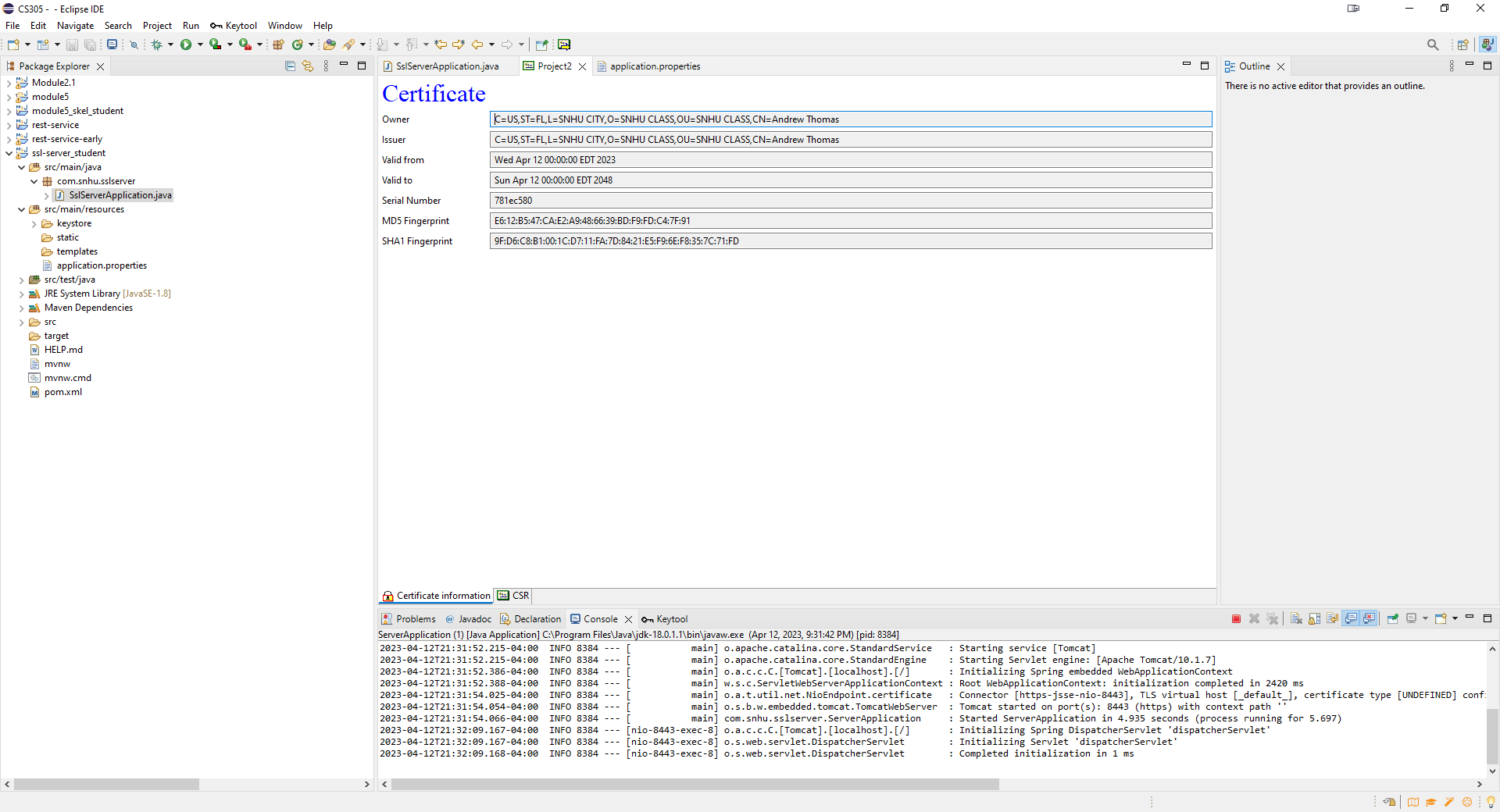
SHA-256 is the Algorithm cipher used and it creates a hash value that is virtually impossible to replicate unless passing through the same data again. In this way you can prevent collisions which in the long run creates more security as you can’t manipulate the code without giving a different checksum hash. Thus being able to ensure that the passed data is correct.

Essentially it will take the data and convert it into binary code. From this point it will then divide that binary data into 512 bit blocks, expanding the blocks as necessary by padding the data to fit 512 bit criteria. Then it will divide those blocks into smaller blocks of 32 bits each and then perform compression functions to rotate the has values in specific patterns. Using this new pattern, it will create new hash values and then produce one final 256 bit hash value which is the end product. This process will create the same hash value at the end for the same exact data that is sent into it and therefore will be used to verify that the data is correct.

Symmetric and non-symmetric keys are used for security of application. However some are more secure than others and for different instances. Symmetric keys are a key that is used to both encrypt and decrypt the cipher, whereas asymmetric keys will use one cipher to encrypt data but then rely on another cipher to decrypt the data, giving a bit more security as if a malicious party is able to decipher the original key, it doesn’t mean they’ll be able to figure out what the data is, as they would need the second key to do so.

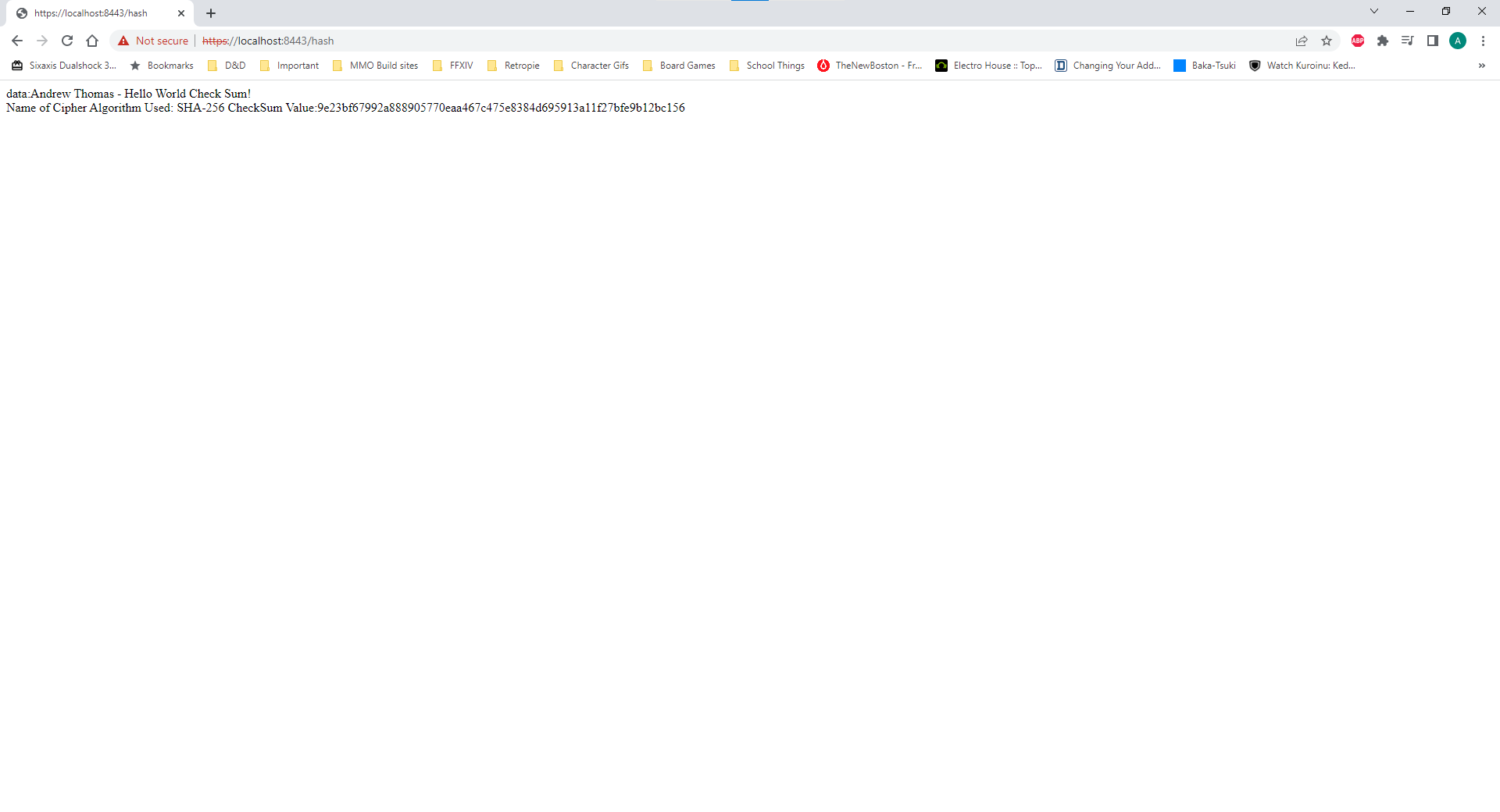
## Certificate Generation

Insert a screenshot below of the CER file.



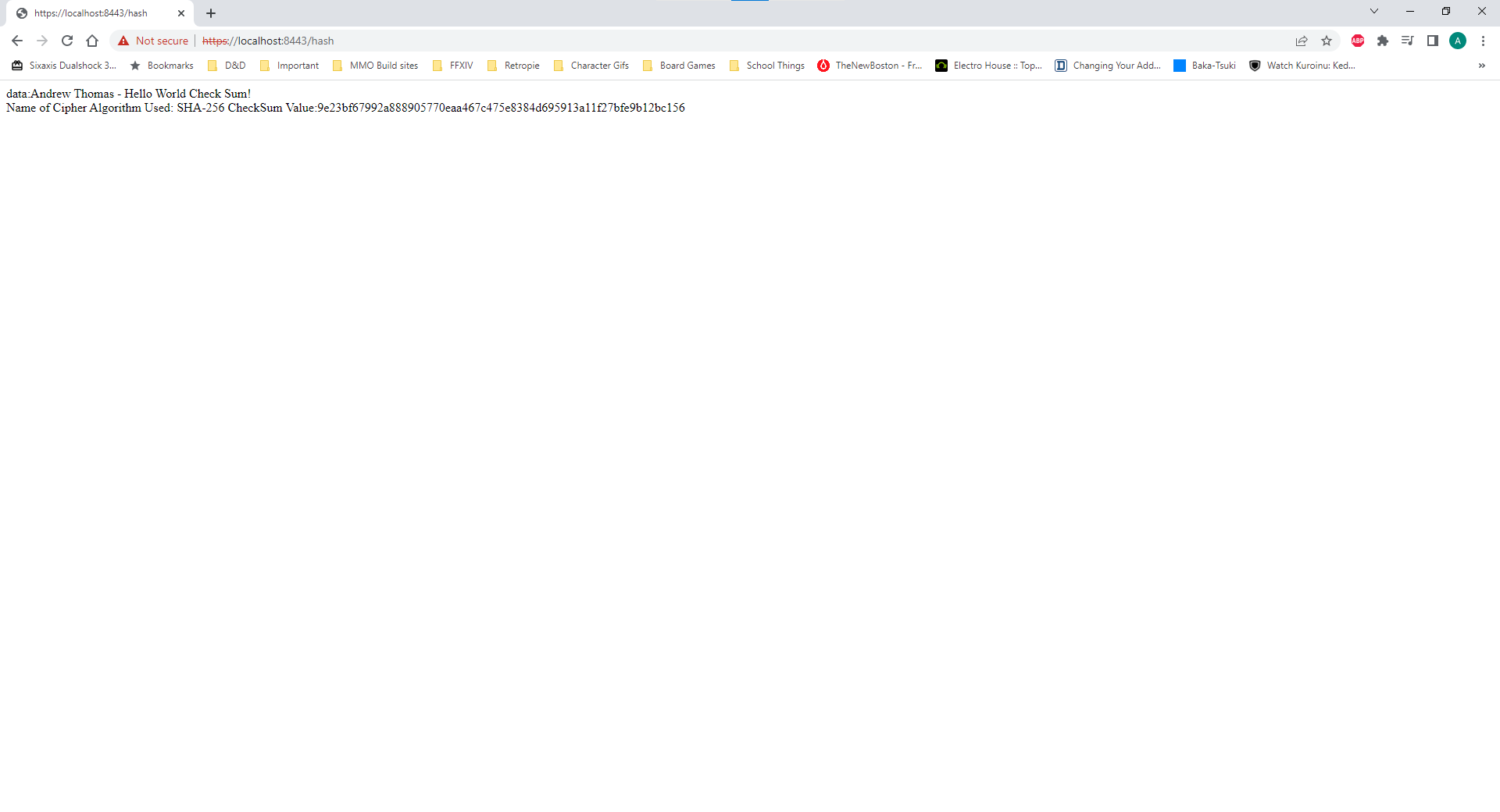
## Deploy Cipher

Insert a screenshot below of the checksum verification.



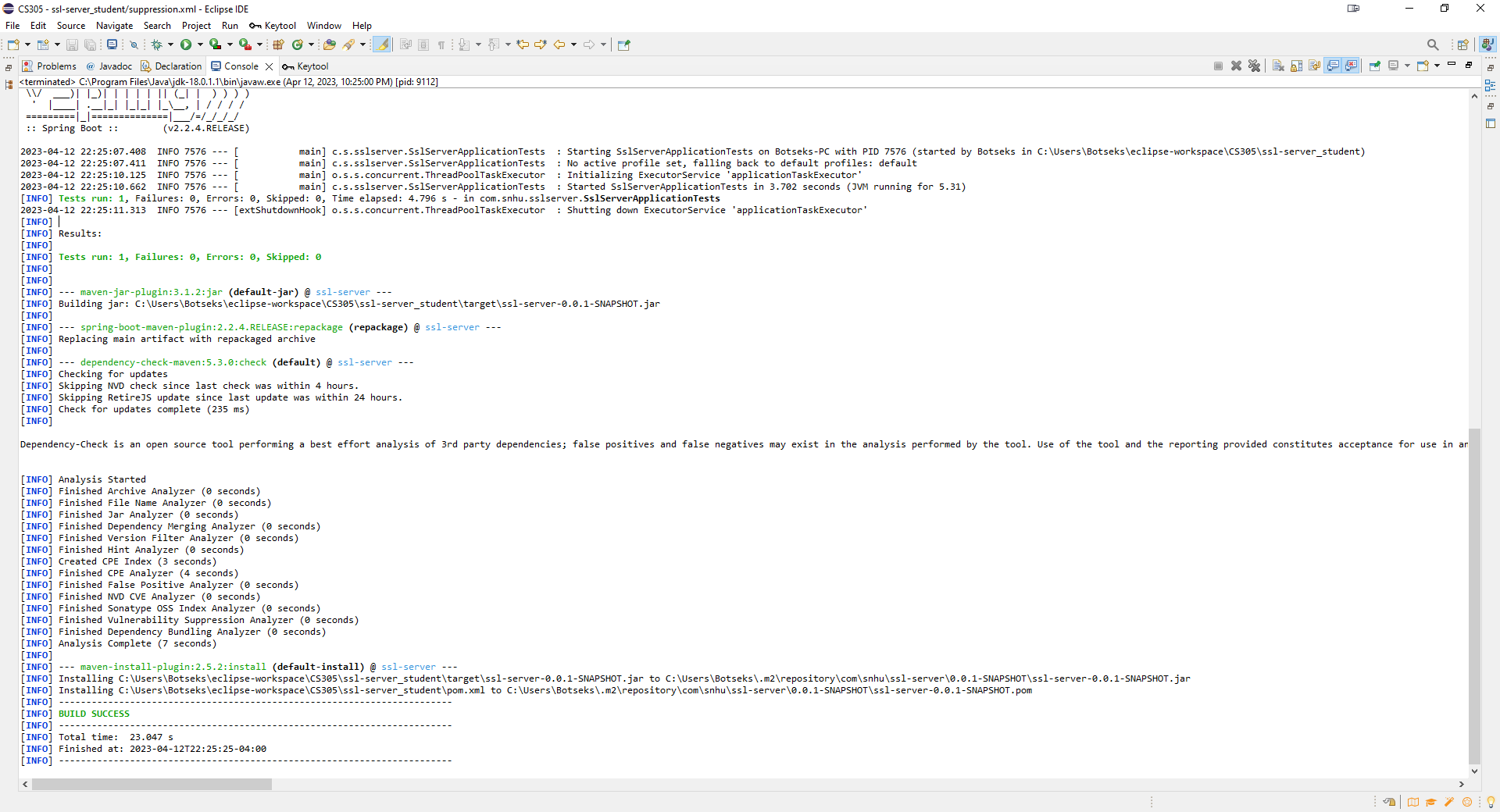
## Secure Communications

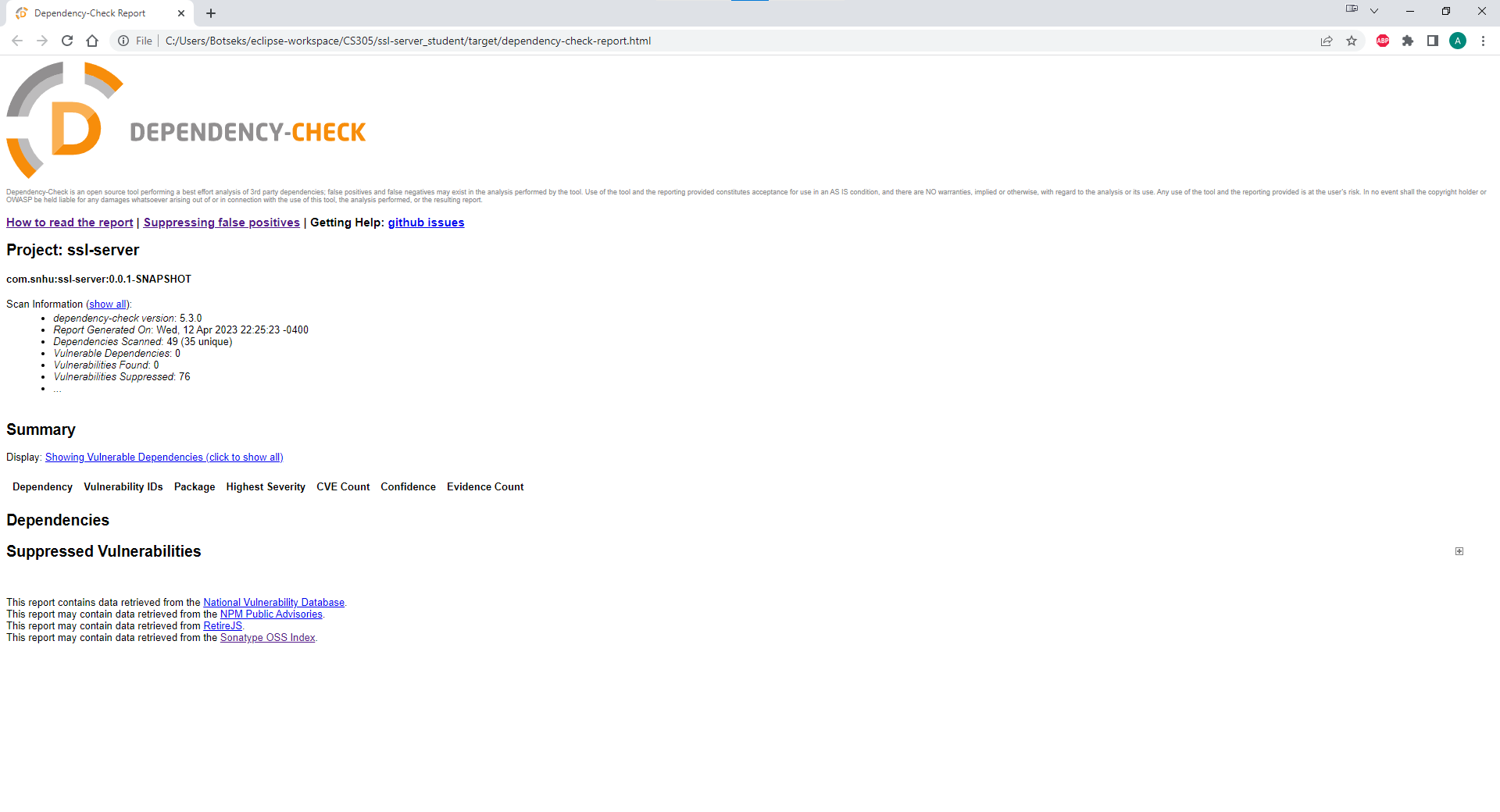
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

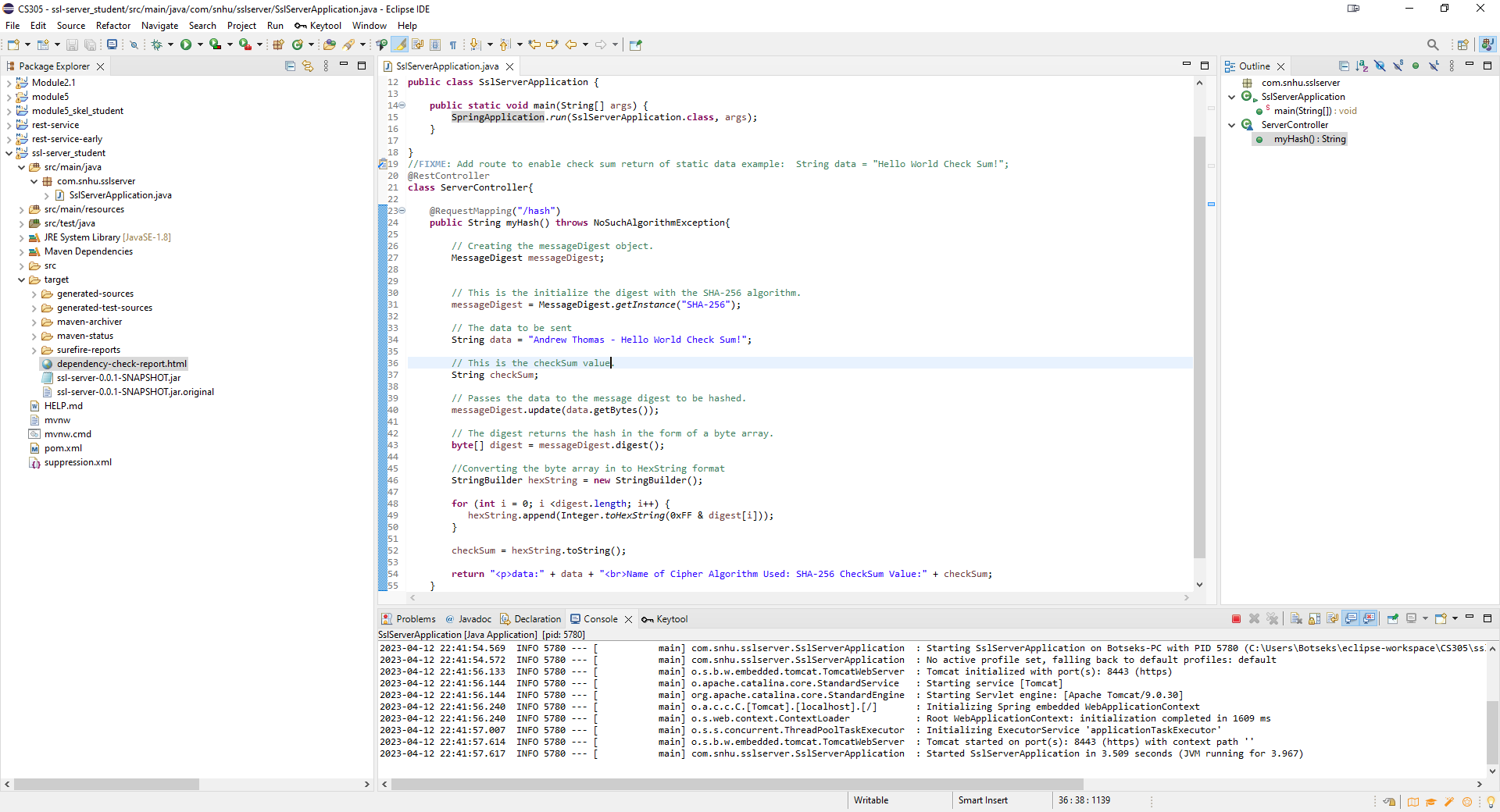
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

Insert a screenshot below of the refactored code executed without errors.



## Summary

The two areas of the vulnerability assessment process flow diagram that were addressed were cryptography and Client/server security. Cryptography was expressed through the encryption of the data string and outputting the checksum verification code while dealing with Client/server security through creation of a keytool certificate in order to verify secure communication through accessing the server via https rather than http. We managed this by passing the string data through a hash algorithm of SHA-256 and showing the checksum verification at the end of the hashed data as well as creating a keytool certificate and using it to access the server.

## Industry Standard Best Practices

Some of the industry best practice standards was proper documentation around the different processes of the coding as well as keeping out ‘phantom data’ which is data that is simply hardcoded into the program. Such as instead of using a variable, using a hardcoded string in the software. It also avoids plugging in user input directly into the processes by assigning the variables first and then inserting the data within them to avoid potential injection of code.

By using industry standard best practices from the beginning you can lower the cost of the development of the software by avoiding having to go back and reimplement and re-work code in order to follow the security protocols of your software in its end stages. You also better protect the assets of your company and clients by keeping your applications secure from at least the known threats that could potentially threaten these assets.