

# 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** | **[Date]** | **[Your Name]** |  |

## 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

Nicholas Bequette

## Algorithm Cipher

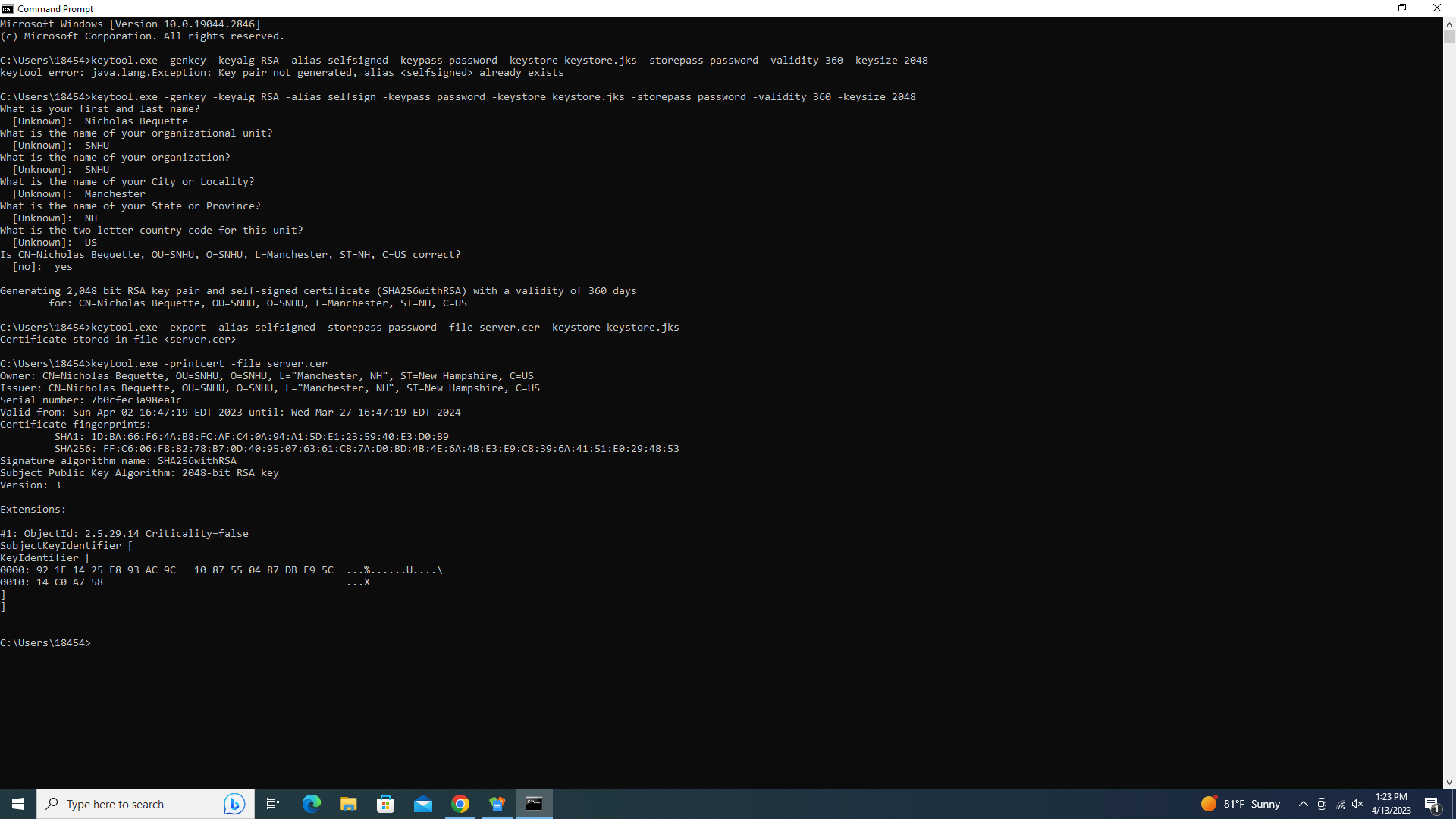
a. The algorithm I would choose to use is Advanced Encryption Standard (AES). I would choose this because it is highly secure and reliable, using a block cipher of size 128 bits as well as key sizes that range from 128, 192, and even 256 bits.

b. Hash functions are used to change data into a certain sized output, but for AES, a hash function isn’t directly used. Instead, a key expansion is used which takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes). This can basically be thought of like a hash function though it is not directly one. As for bit levels, data is translated into 128 bit blocks. During encryption, the size of the key is the number of bit levels (128, 192, or 256). These bits then determine how many rounds are used during encryption.

c. Random numbers are what is used to generate unique keys. PRNGs are what generates these random numbers, allowing for most keys to be unpredictable by brute force. For symmetric and non-symmetric keys, symmetric means that during encryption and decryption both the same key is used. The sender and receiver both have the same key which does pose a slight security risk, but it is extremely fast which allows for large data amounts to be encrypted very efficiently. Non-symmetric means that two unique keys are used, one for encryption and one for decryption. This is extremely secure, but way less efficient. AES uses symmetric keys.   
d. Encryption originated thousands of years ago with examples such as the scytale that allowed secret communication between military commanders. In today’s world however, it has become much more advanced and widely used. Some examples of encryption that are used today are AES, DES, and Blowfish. Today we use these algorithms to protect sensitive data from being accessed by unauthorized users. While we have come a long way, there is still a need for more security when it comes to these algorithms

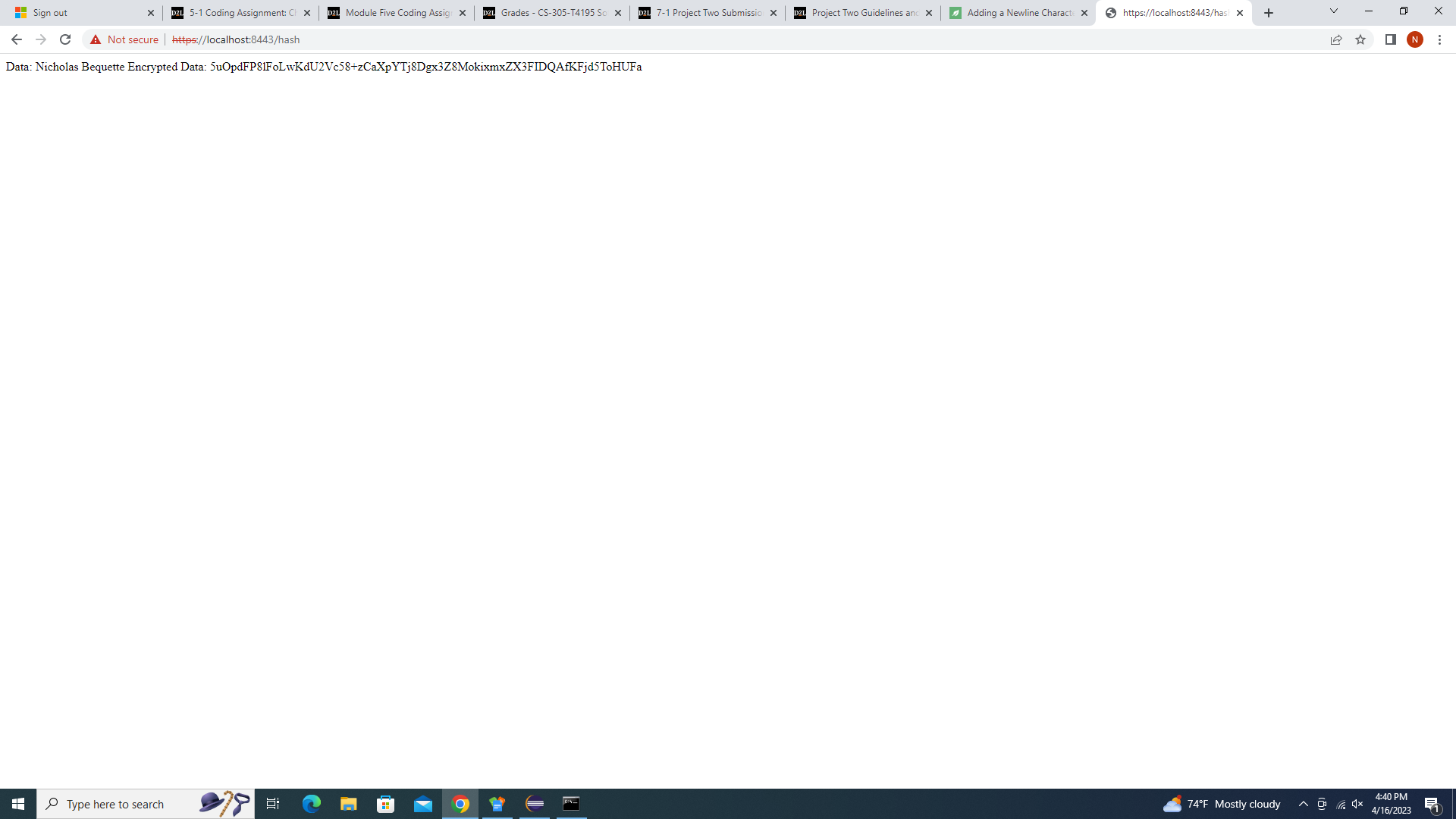
## Certificate Generation

Insert a screenshot below of the CER file.



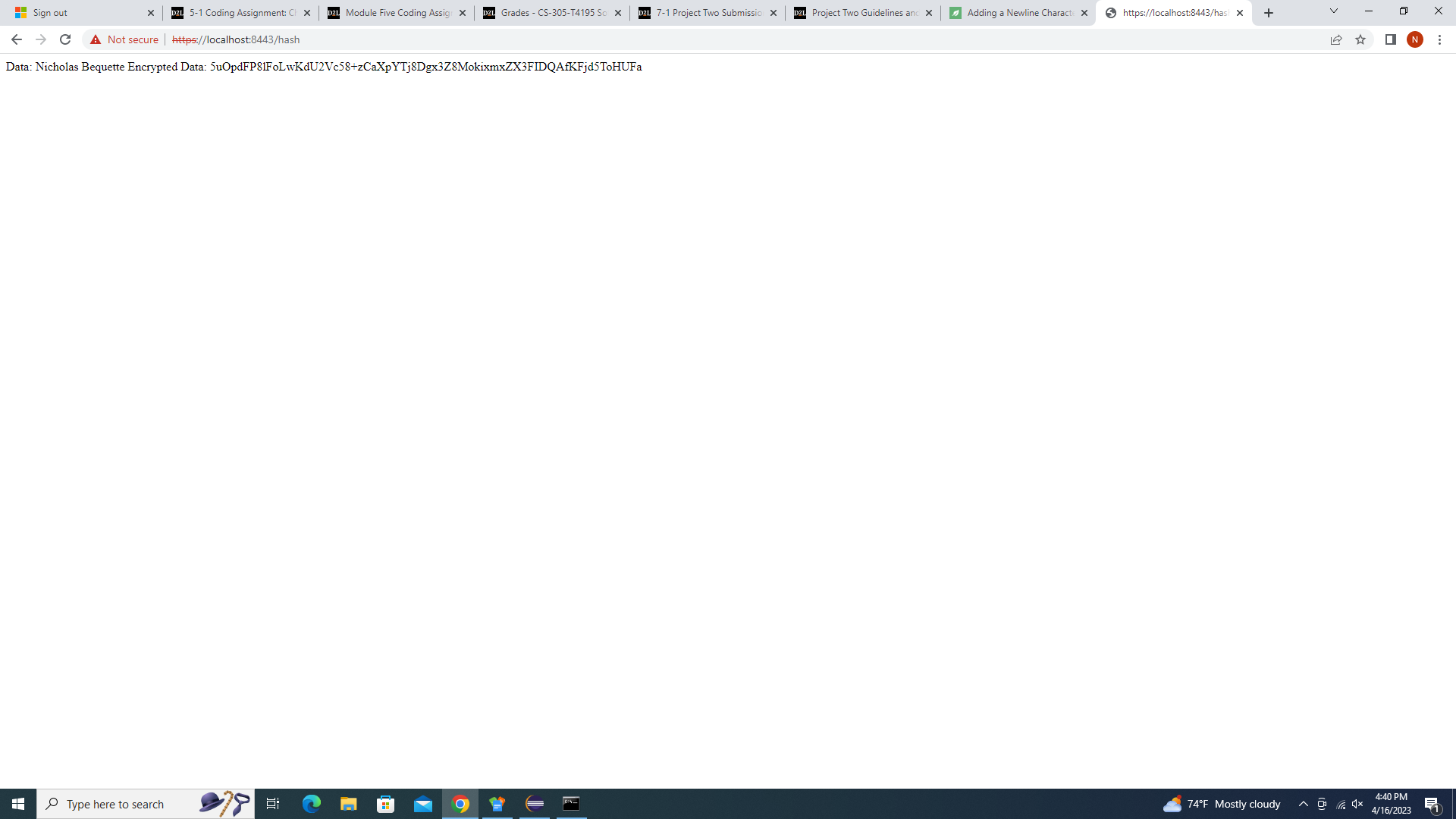
## Deploy Cipher

Insert a screenshot below of the checksum verification.



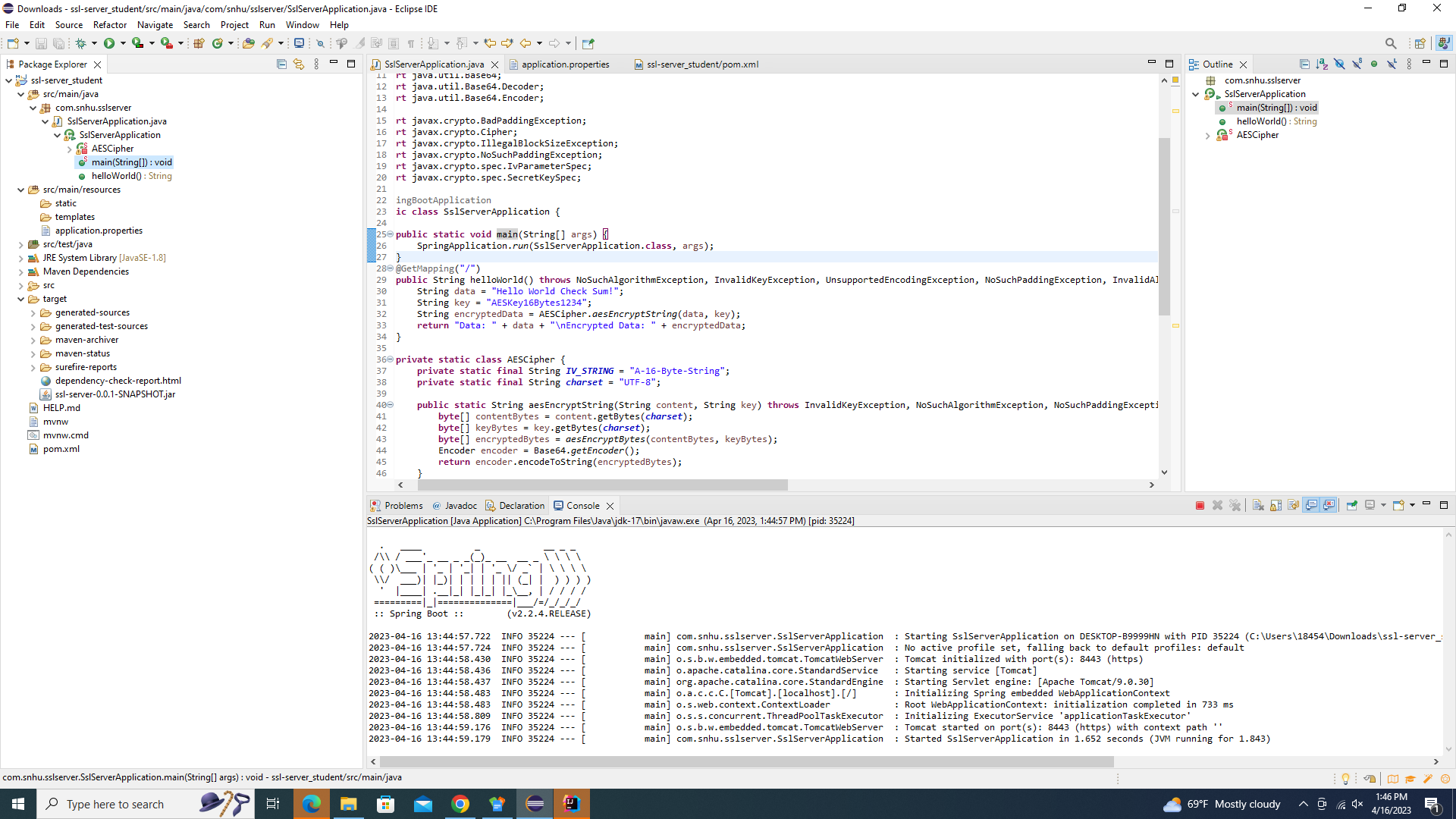
## Secure Communications

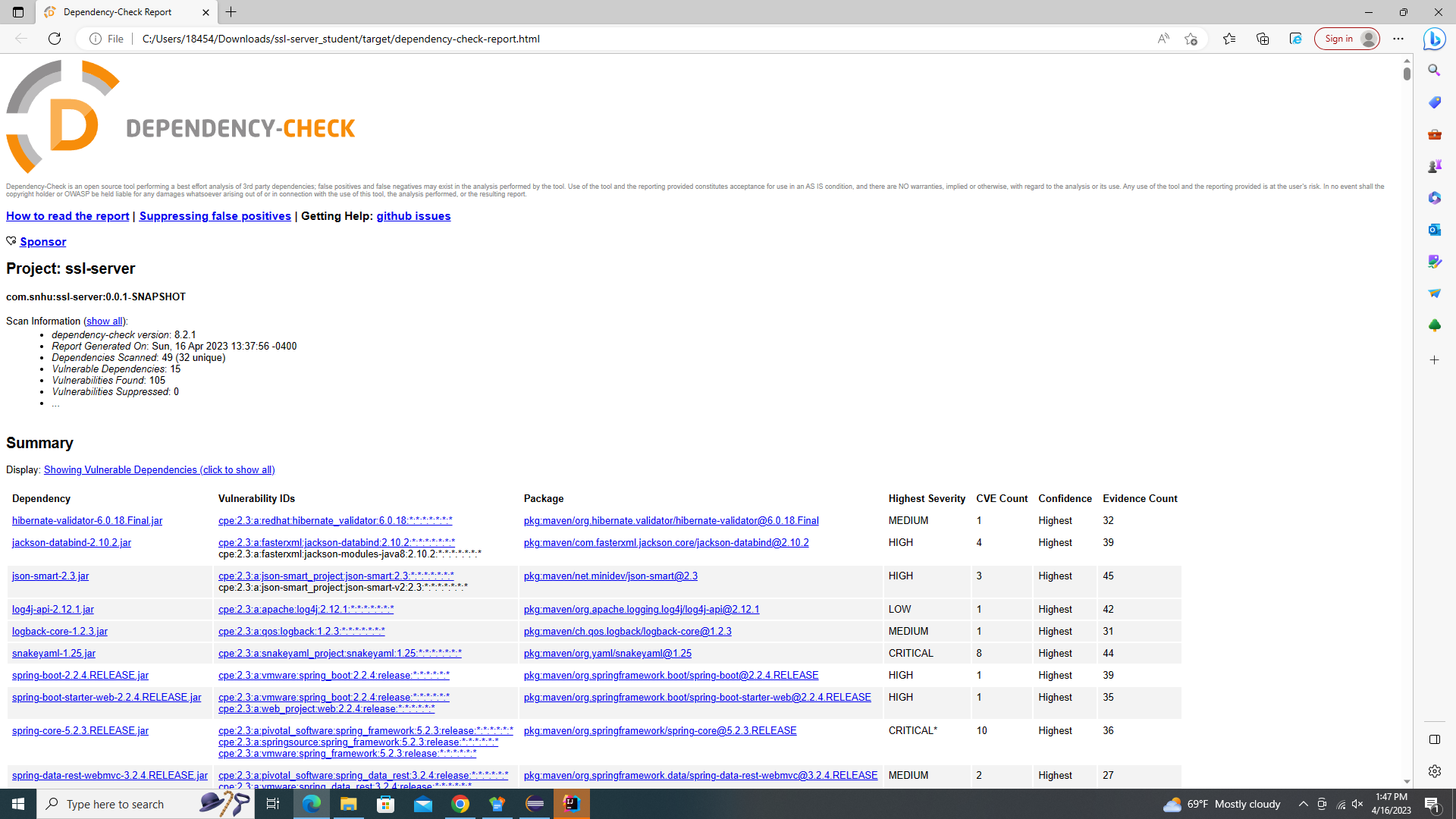
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

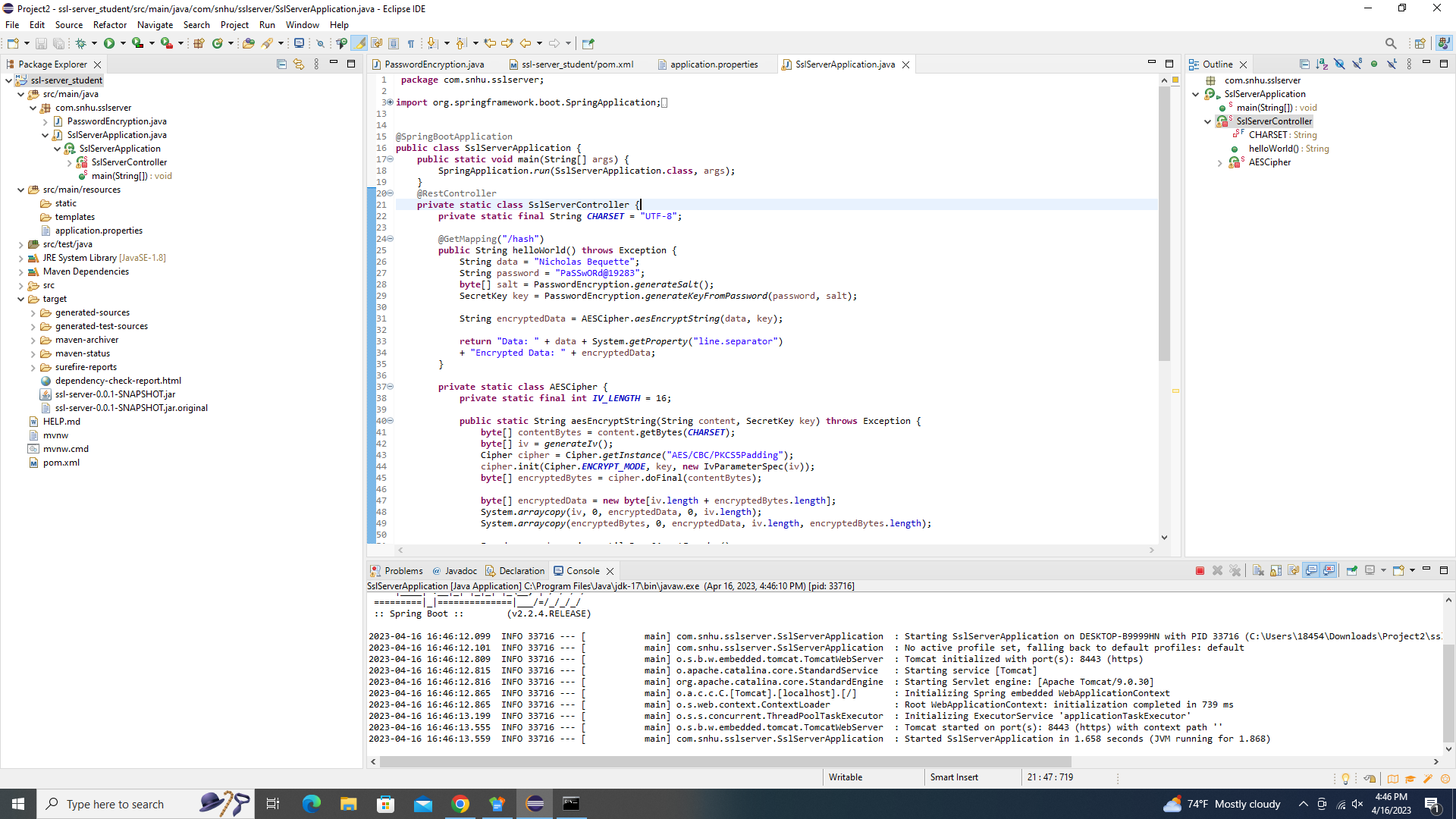
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

Initially, the code I had wrote had a few security vulnerabilities that could not be overlooked at all. At first, I had a hard-coded encryption key, no input validation, and even returned data in regular text instead of encrypted text. So, I implemented some input validation to prevent a buffering overload. I used cryptography to generate a unique key rather than hard-coding one. I also made methods to encrypt and decrypt data. My code was reviewed multiple times to ensure that there were no errors, and that it was of high quality. And last but not least, I used tried to use the best algorithm possible to ensure that the software would be as secure as it possibly could be.

## Industry Standard Best Practices

To keep security high, I used the industry’s best practices to maintain secure software. The practices I used include using the best, secure encryption algorithms I possibly could for this specific software. I also used cryptography, generating unique keys that can be used for encryption and decryption. I also used input validation to prevent buffering, as well as HTTPS to make sure communication was secure. Finally, I made sure there were no vulnerabilities in code error and that my code used best industry standard practices. By doing all of the above, I ensured that everything done with Artemis is secure, which not only ensures we’ll rarely have attacks, but also builds trust with customers so that we can keep growing and avoid legal liability.

REFERENCES

GitHub. (n.d.). Build software better, together. Retrieved April 12 2023, from <https://github.com/topics/aes-algorithm?l=java>

BrainKart. (n.d.). AES Key Expansion. Retrieved April 12, 2023, from <https://www.brainkart.com/article/AES-Key-Expansion_8410/>