****

# Practices for Secure Software Report

**Table of Contents**

[**Practices for Secure Software Report 1**](#_heading=)

[**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 5**](#_heading=h.3dy6vkm)

[**3. Deploy Cipher 5**](#_heading=h.1t3h5sf)

[**4. Secure Communications 6**](#_heading=h.4d34og8)

[**5. Secondary Testing 7**](#_heading=h.2s8eyo1)

[**6. Functional Testing 8**](#_heading=h.17dp8vu)

[**7. Summary 9**](#_heading=h.3rdcrjn)

[**8. Industry Standard Best Practices 9**](#_heading=h.26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **18 June 2023** | **Allen Jhane Dela Cruz** |  |

## 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

Allen Jhane Dela Cruz

## Algorithm Cipher

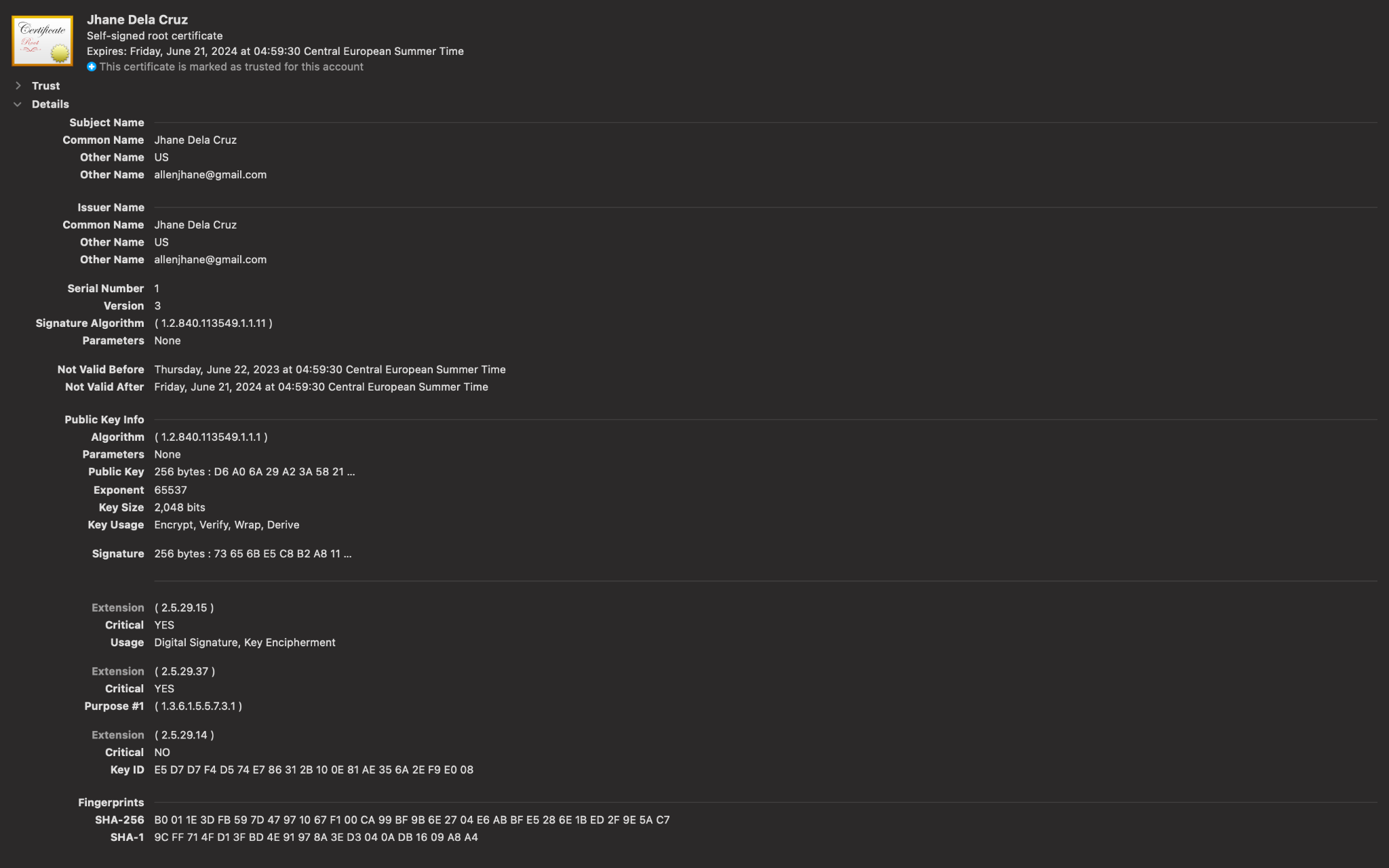
* 1. Provide a brief, high-level overview of the encryption algorithm cipher.
* The encryption algorithm I chose was the SHA-512 algorithm. This secure hash algorithm (SHA) belongs to the SHA-2 family of cryptographic hashes and produces a message digest. SHA-512 computes 64-bit words and outputs a hashed text of the given text.
  1. Discuss the hash functions and bit levels of the cipher.
* The message digest for this algorithm is longer than its SHA-256 counterpart but roughly 50% faster than it is on 64-bit machines due to its use of 64-bit words rather than 32-bit words (SHA-256).
  1. Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.
* Cryptographic algorithms, such as SHA-512, require keys. Random numbers are used in a key generation to create random strong keys that can also be made for cryptographic algorithms. Keys encode or decode cryptographic data with encryption. Encryption is the process of changing data to protect it from another person. Keys are categorized between symmetric and non-symmetric (asymmetric) keys. A symmetric key encrypts and decrypts data with the same key. Where an asymmetric key encrypts data with a pair of keys, public and private, to encrypt and decrypt a message, respectively. Since a symmetric key uses only one key to encrypt and decrypt a message it is less secure but much faster than an asymmetric key. An asymmetric key is considered to be more secure since it uses public and private key pairs but can be much slower due to the different key requirements.
  1. Describe the history and current state of encryption algorithms.
* Encryption has been recorded far back in history with its most famous early usage in the Old Testament of the Bible circa 500-600 B.C. Throughout ancient times, ciphers were used to encrypt ancient texts and protect them from unwanted eyes. This was done through transposition and substitution cipher methods. A transposition cipher is when the sender rearranges the letters in a word to make them appear garbled to anyone with a predefined system that only the sender and recipient know. A substitution cipher replaces characters with other characters based on predetermined rules. Later on in history, especially in war times, mechanical devices were used to encrypt messages. In modern times, encryption has evolved with technology, and computers created better encryption with mathematical equations and algorithms.

## 

## Certificate Generation

Insert a screenshot below of the CER file.

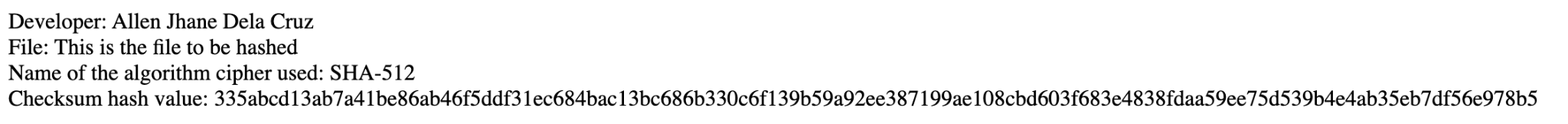






## Deploy Cipher

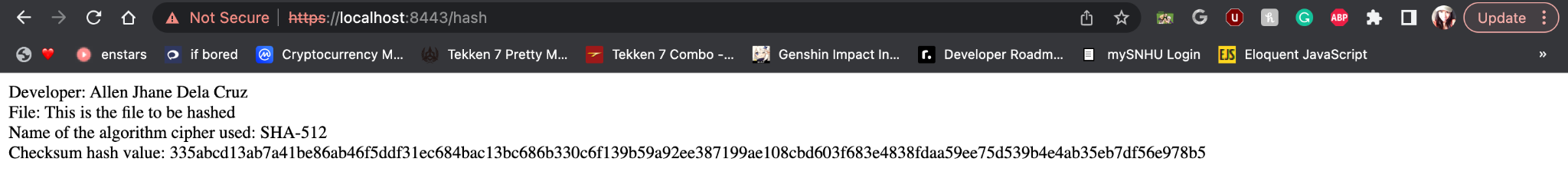
Insert a screenshot below of the checksum verification.

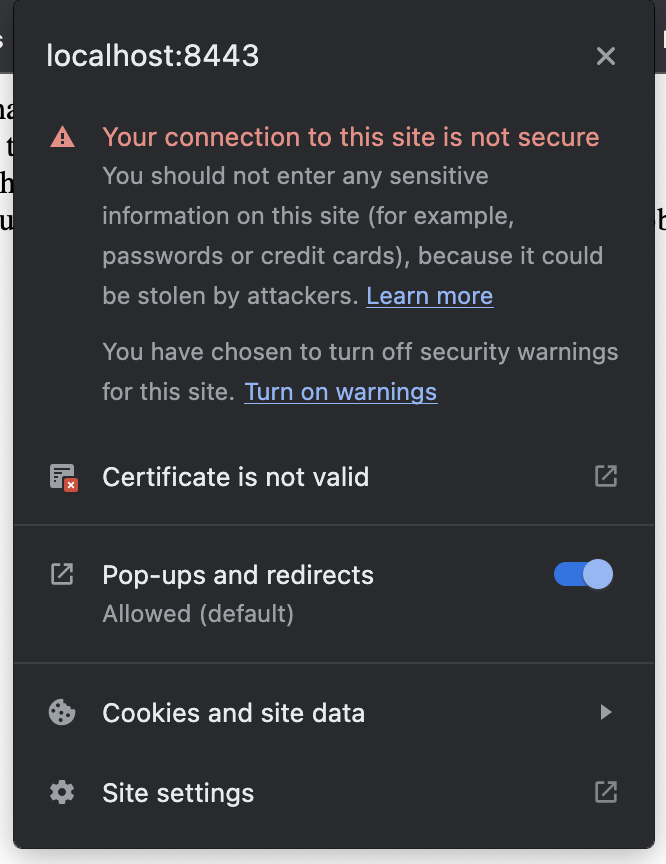


## 

## Secure Communications

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

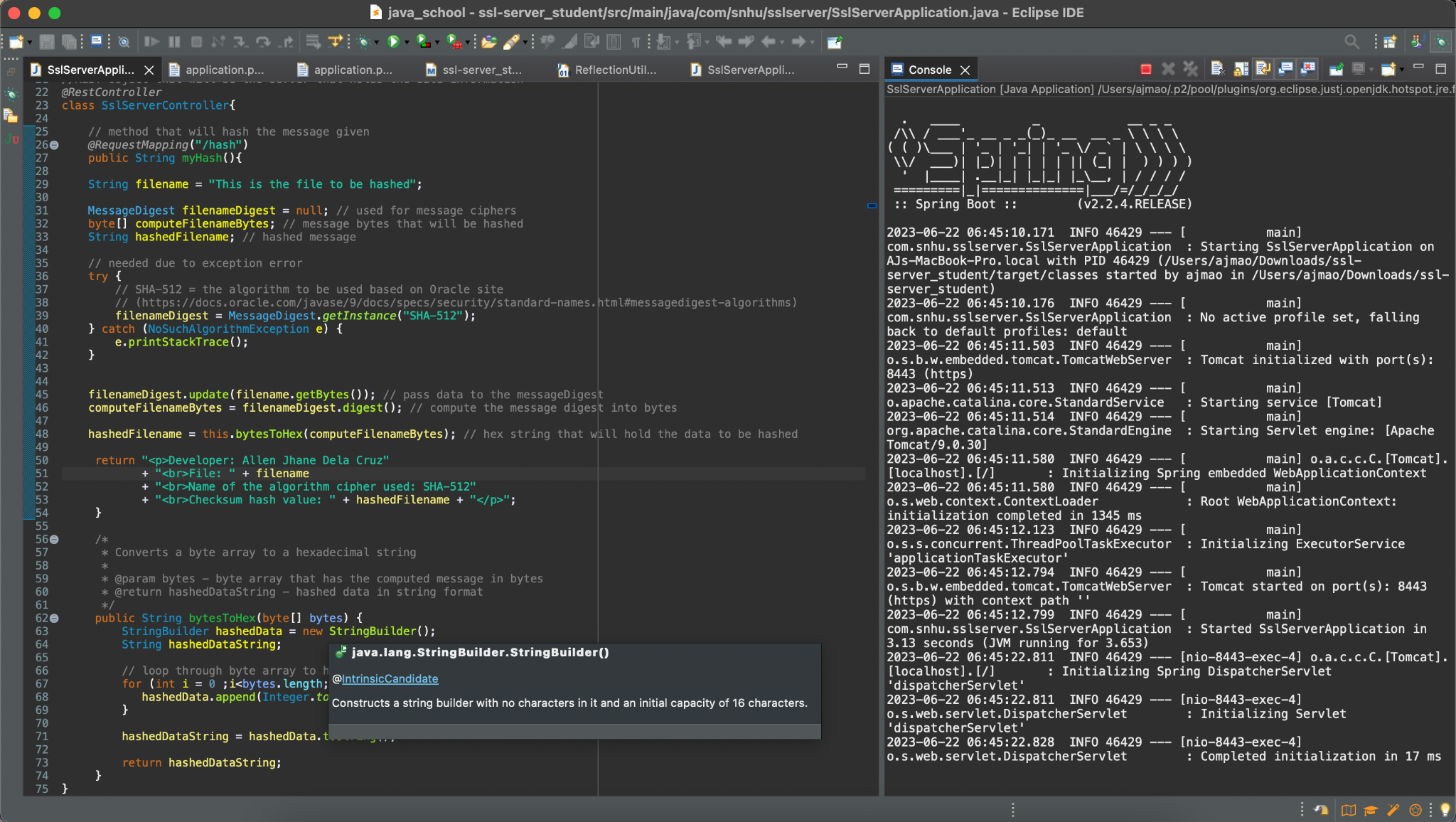


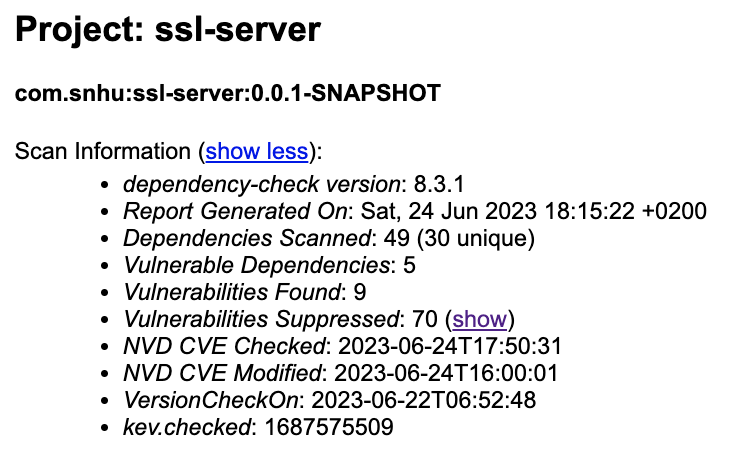


After further research into why the localhost cannot have my certificate verified, I have confirmed that there are no CA issues to SSL certificates for localhost since no one can directly own it (*SSL certificate for localhost - is it possible?* 2020). I have also read multiple articles that say the same thing as the one I quoted. Though I cannot verify the certificate, I was able to generate and use this along with the localhost to generate a secure website with the use of HTTPS.

## Secondary Testing

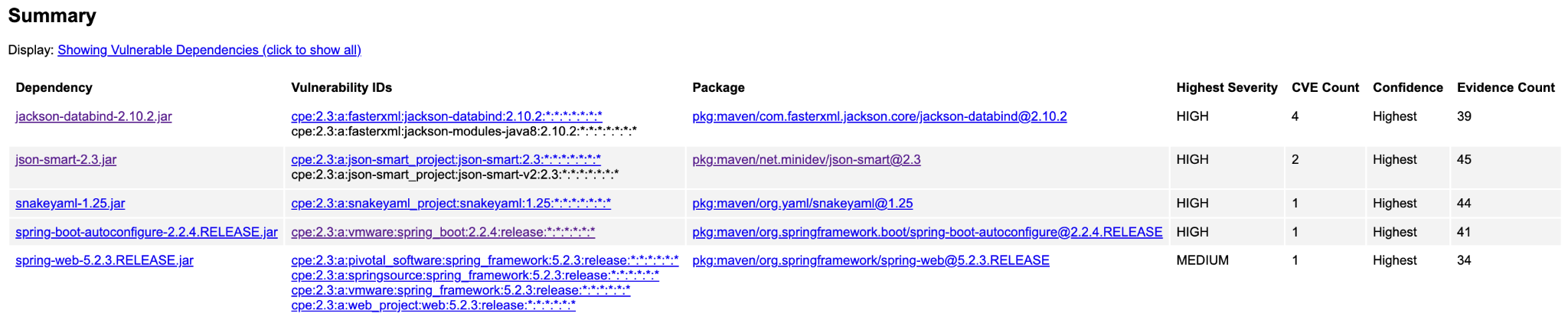
Insert screenshots below of the refactored code executed without errors and the dependency-check report.



****

A dependency check was done with the previous module’s suppression file. The file was able to suppress 84 vulnerabilities, however, there were 45 vulnerabilities found, (without checking if these dependencies were false positives). After checking for false positives, 5 vulnerable dependencies were found, leaving 9 vulnerabilities present in the code base. These vulnerabilities were also found in the previous model after inspection of the two dependency reports. The vulnerabilities are shown below.

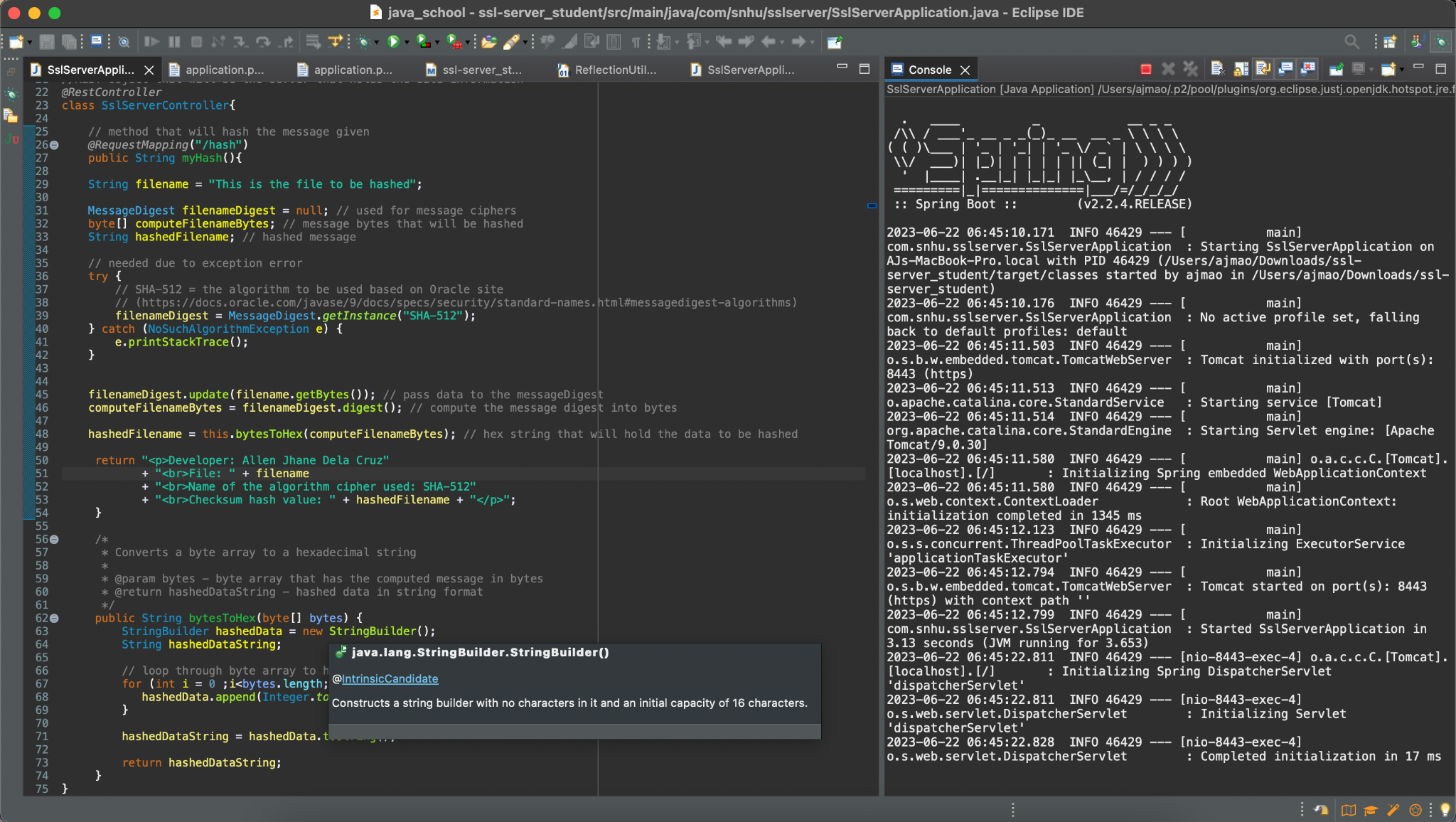
## 



Out of the remaining vulnerable dependencies, there are no known exploitations.

## Functional Testing

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



The explanations for the vulnerabilities below are based on the packages in the dependency check report.

Vulnerabilities:

* Syntactical: There is improper case handling/sensitivity, improper input validation, improper check for unusual or exceptional conditions, and improper output neutralization for logs. These syntactical vulnerabilities can then cause the software to either crash, expose sensitive information, or cause additional insertion into the software.
* Logical: The software does not restrict the size of consumption of data which can then have it consume more data than needed. This can also be affected by a large depth of nested objects that may allow a StackOverflow exception and denial of service thus crashing the software. This is also possible through an out-of-bound write vulnerability in the software.
* Security: Remote code execution can be possible for attackers through the deserialization of untrusted data. Exposure of resource to wrong sphere can also make software vulnerable to temporary directory hijacking.

## Summary

1. Refer to the Vulnerability Assessment Process Flow Diagram. Highlight the areas of security that you addressed by refactoring the code.

* The areas of security that I address by refactoring the code are API, Cryptography, Code Error, and Code Quality. From the old code, I added additional lines to the application.properties file in the resources folder of the project in order to create an HTTP Secure localhost webpage. I also updated the code by rearranging the “@RequestMapping” function for the REST API into a method that will only hold the logic to print to the webpage (printHash()). By refactoring some of the logic in the myHash() original function, I was also able to consolidate algorithm logic to myHash() and separate all other logic to their own methods. While refactoring the code base, I also changed code formats and variables for the functions which address the code error and code quality aspects of the code base.

1. Discuss your process for adding layers of security to the software application.

* In order to add layers of security to the software application, I referenced the dependency-check report and suppressed all the false positive vulnerabilities in the application in order to know what are the actual potential vulnerabilities. I then looked into the consolidated list of vulnerabilities and suppressed all the vulnerabilities that did not apply to my program. Having a large depth of nested objects causes some vulnerabilities however this was taken care of through refactorization of the code base. User input is not a part of the program so this will not be a vulnerability either. To combat the other factors of vulnerabilities I ensured that the Eclipse IDE was up to date as well as the Java that was installed onto my computer. I also ensured that the certificate and key that I created were a part of the resources folder of the application and included in the application.properties file. With these security components, it ensures the web browser for the localhost is secure (HTTPS).

## Industry Standard Best Practices

1. Explain how you used industry standard best practices to maintain the software application’s current security.

* I used industry standard best practices by conducting thorough research of the vulnerabilities, commenting code to explain my program, and structuring the program for easy maintainability. First, I did thorough research on the vulnerabilities in order to know the false positives of the program and to know how to combat the remaining security flaws in the system. Additionally, Commenting code helped me understand my thought process throughout development and help understand the code base system as a whole so I would not forget as I added more logic. Finally, I looked over my code and refactored the code to ensure that its structure was maintainable and considered the vulnerabilities that are present in the program.

1. Explain the value of applying industry standard best practices for secure coding to the company’s overall well-being.

* Applying industry standard best practices for secure coding can ensure a company’s overall well-being. This is because this can be the difference between protecting a company’s data and being susceptible to malicious actors. Without ensuring the vulnerabilities in a system, you can not know how your company is susceptible to attacks and how to mitigate these vulnerabilities. Also having a more maintainable code base can help other developers update and maintain the code. Knowing what is best in the industry and understanding the code to apply the best standard practices can help ensure a company’s code base is secure and knowledgeable of all vulnerabilities for mitigation.

**References:**

*SSL certificate for localhost - is it possible?*. Comodo SSL Resources. (2020, May 25). https://comodosslstore.com/resources/ssl-certificate-for-localhost/

Murithi, L. (2021, April 26). *How to get SSL HTTPS for localhost*. Section.io. https://www.section.io/engineering-education/how-to-get-ssl-https-for-localhost/

*SHA-512, SHA-512/224, SHA-512/256¶*. SHA-512, SHA-512/224, SHA-512/256 - PyCryptodome 3.18.0 documentation. (n.d.). https://pycryptodome.readthedocs.io/en/latest/src/hash/sha512.html

Scholten, U., & Smirnoff, P. (2019, April 23). *The use of random number generators in cryptography*. Cryptomathic. https://www.cryptomathic.com/news-events/blog/the-role-of-random-number-generators-in-cryptography

GeeksforGeeks. (2023, March 20). *What is asymmetric encryption?*. GeeksforGeeks. https://www.geeksforgeeks.org/what-is-asymmetric-encryption/

GeeksforGeeks. (2023, May 22). *Difference between symmetric and asymmetric key encryption*. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-symmetric-and-asymmetric-key-encryption/

*Encryption: The past, present, and future*. AXELorg. (2021, May 28). https://www.axel.org/2021/05/28/history-of-encryption/