****

# 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** | **08/17/2025** | **Eleanor Shamble** |  |

## 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

Eleanor Shamble

## Algorithm Cipher

I will be using SHA-256.

The requirements are as follows: there will need to be checksums for file verification when transferring data. I will be using MessageDigest.

MessageDigest narrows down the choices to MD5, SHA-1, and SHA-256. Compared to SHA-1, MD5 is considered insecure (“Difference between MD5 and SHA1”, 2025). MD5 less secure because it is more vulnerable to collisions, it’s easier to crack the code (“Difference between MD5 and SHA1”, 2025). The only advantages MD5 has over SHA-1 are speed, output length, and simplicity (“Difference between MD5 and SHA1”, 2025), and none of those are the biggest factor here.

SHA-1 has its own weaknesses. SHA-1 is less vulnerable to collisions than MD5, but SHA-256 is even less vulnerable than SHA-1 (“Difference between SHA1 and SHA256”, 2025). SHA-256 has a larger hash size than SHA-1, and is considered the standard while SHA-1 is obsolete (“Difference between SHA1 and SHA256”, 2025).

So: considering all three options given by the MessageDigest requirement, SHA-256 is the best choice, SHA-1 is second best choice, and MD5 is the worst choice. Even with the specific considerations for this assignment, there is only one choice that is not obsolete.

SHA-256 prepares the input by padding it to make sure it is the correct length, or splitting it into chunks that are the correct length (Gitlan, 2025): 512 bits, with the last 64 bits containing the length of the original input (RedBlockBlue, 2022).The 512 bit block is split into 16 chunks of 32 bit words (Gitlan, 2025). The chunks are expanded into 64 chunks, before the blocks are compressed in 64 rounds of bitwise operations (Gitlan, 2025). The final hash value should be unique (Gitlan, 2025).

Considering vulnerability to collisions is important, because collisions can result in collision attacks, where hackers find another value that produces the same hash (Lake, 2023). An example of spoofing the same hash would be an employee trying to submit the wrong document but changing it enough that it has the same hash value as the legitimate document (Lake, 2023). That would be a classical collision (Lake, 2023). Chosen-prefix collision attacks are more deliberate- they involve finding the exact value that will cause the same hash, rather than brute forcing it (Lake, 2023). Collision attacks can result in hackers inputting malicious data and getting into places they aren’t supposed to get into.

## Certificate Generation

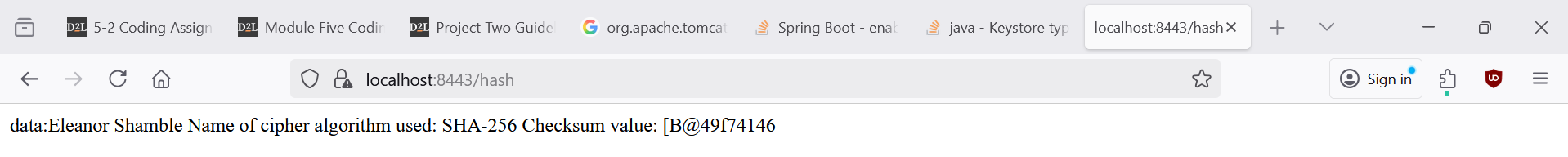
Insert a screenshot below of the CER file.

A computer screen with text and numbers

AI-generated content may be incorrect.

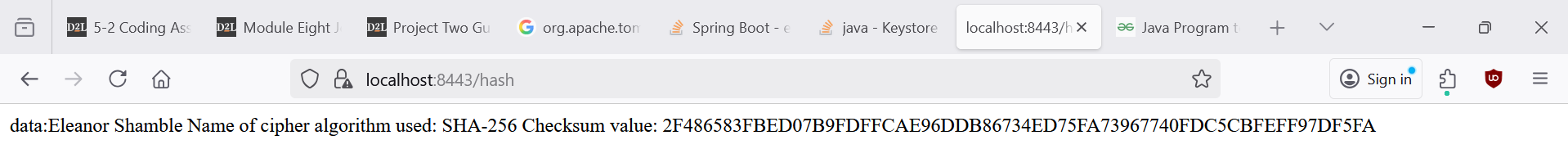
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

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

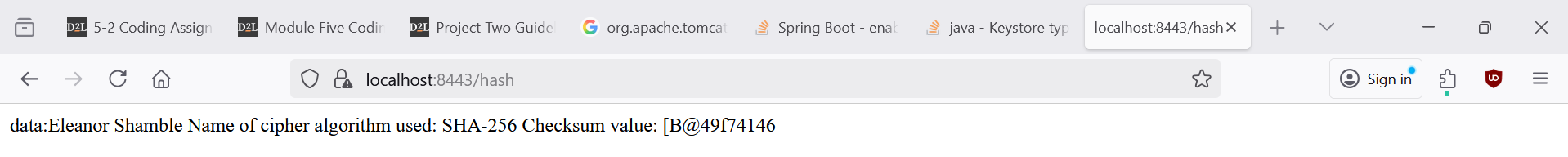


## Secondary Testing

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

A close-up of a text

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

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

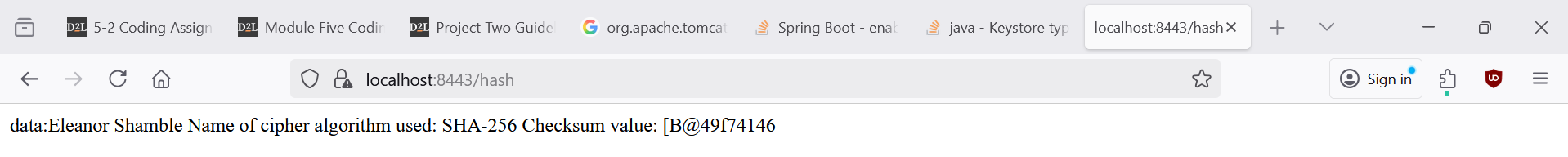
A screenshot of a computer program

AI-generated content may be incorrect.A close-up of a computer code

AI-generated content may be incorrect.

A close-up of a text

AI-generated content may be incorrect.



## Summary

The code overall has been refactored to include a dependency check, a keystore, a hash function, and checksum verification.

Vulnerability assessment:

* Input validation
  + At this stage in the code, input validation is not as relevant, as the data entered is static, and it is entered by the developer, not the user. When the code is at a more complete state and is ready to verify files and transfer data, input validation will need to be added to ensure that no malicious files are uploaded and stored. I would recommend that the file types would be checked at the very least, to make sure that users send in the appropriate type of file.
    - Input validation is important, because without it, the program could be vulnerable to SQL injection (with text input), trojan horses (when uploading files), unwanted file contents (like profanity), et cetera.
  + For now, I added code to check whether the message is null or an empty string, and to throw an exception if it is. This is in lines 33-39 of SslServerApplication.java.
* APIs
  + APIs should be authenticated so users don’t upload any malicious data.
  + There is a keystore implemented for security, the details are in application.properties, and keystore.jks is in the resources folder with it.
* Cryptography
  + SHA-256 was utilized to encrypt static messages. This is in lines 27-53 of SslServerApplication.java.
* Client/Server
  + HTTPS protocol is enforced.
* Code Error
  + There is error handling for the /hash mapping such that if the message fails to hash, the user will be greeted with an error page showing the error message. In SslServerApplication.java, there is a try-catch loop for this.
  + Currently, there are no errors when the application is launched.
* Code Quality
  + The code is simple, but I added comments to make it easier to understand.
  + False positives were suppressed in the dependency check, which makes room for new vulnerabilities to be noticed. The dependency check is in ssl-server\_student/target/dependency-check-report.html
* Encapsulation
  + There are no relevant data structures in this stage of development.

## Industry Standard Best Practices

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

* I included input validation appropriate to the program to minimize the risk of unwanted errors.
* I added error handling, so an error doesn’t crash the application.
* I added comments to ensure that future developers understood my code.
* I ensured that HTTPS protocol was enforced to help with security.
* I implemented the industry standard SHA-256 algorithm for generating checksums so it would be harder to crack.

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

Secure coding is important for many reasons:

* Input validation ensures that all user input meets the company’s standards. This can include security standards (preventing SQL injection, viruses, and insecure passwords), content standards (profanity filters), and helps prevent bugs (preventing duplicate user ids). Programs storing faulty input creates technical debt, as someone will need to find some way to delete or change it through code, or take the time to do so manually. SQL injections and viruses can result in security breaches, or other crises.
* Insecure APIs can also result in security breaches.
* Without cryptography, sensitive information such as passwords would be very vulnerable if there was a security issue.
* Secure connections should be enforced to prevent malicious actors from exploiting insecure connections.
* Code errors can negatively impact the user experience, but could also be catastrophic, depending on the error.
* Maintaining code quality through modularization, encapsulation, et cetera makes code easier to test and easier to maintain, which is incredibly important for any program long-term.

References

GeeksforGeeks. (2025, July 11). *Difference between MD5 and SHA1*. https://www.geeksforgeeks.org/computer-networks/difference-between-md5-and-sha1/

GeeksforGeeks. (2025, July 23). *Difference between SHA1 and SHA256*. https://www.geeksforgeeks.org/computer-networks/difference-between-sha1-and-sha256/

Gitlan, D. (2025, May 13). *What is the SHA-256 algorithm & how it works - SSL dragon*. SSL Dragon. <https://www.ssldragon.com/blog/sha-256-algorithm/>

Lake, J. (2023, September 13). *What is a collision attack?*. Comparitech. https://www.comparitech.com/blog/information-security/what-is-a-collision-attack/

RedBlockBlue. (2022, March 17). *SHA-256 | COMPLETE Step-By-Step Explanation (W/ Example)*. YouTube. https://www.youtube.com/watch?v=orIgy2MjqrA