

# **Practices for Secure Software Report**

**Table of Contents**

Document Revision History. 3

Client. 3

Instructions. 3

Developer. 4

1. Algorithm Cipher. 4

2. Certificate Generation. 4

3. Deploy Cipher. 4

4. Secure Communications. 4

5. Secondary Testing. 4

6. Functional Testing. 4

7. Summary. 4

8. Industry Standard Best Practices 4

## **Document Revision History**

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **12/14/2024** | **Liam Breslin** | **N/A** |

## **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**

Liam Breslin

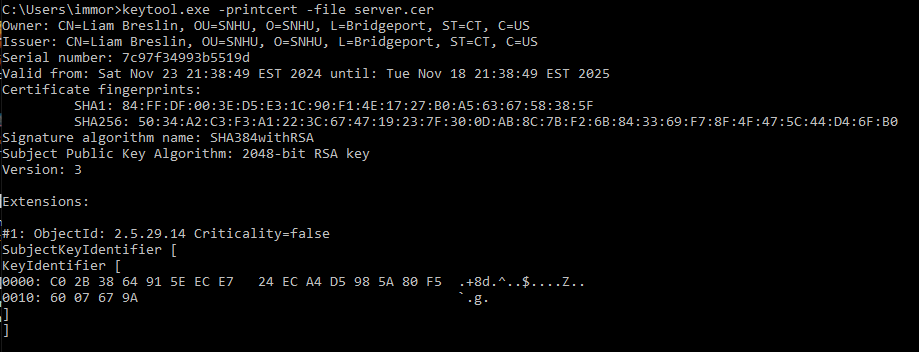
## **1.** **Algorithm Cipher**

SHA-256 is a cryptographic hash function designed by the National Security Agency (NSA). It transforms data into a 256 bit or 64 character hash value. It’s considered to be very reliable and secure with minimal risk of collision and brute-force attacks. The input itself is padded and divided for the processing. Each chunk of data undergoes multiple rounds of mathematical operations to produce the hash values. After all of the process the final 256 bit hash is produced, representing a perfectly unique fingerprint of the data. Since SHA-256 is a hash function and is not considered an encryption cipher. The difference between the two are that hash functions are a one-way function making it irreversible, while encryption ciphers are two-way and are reversible. For encryption symmetric keys are used for encryption and decryption, as well as for verifying integrity via hashing the message before and after encryption. Asymmetric keys are used often in public/private key pairs, and in digital signatures to hash data before.

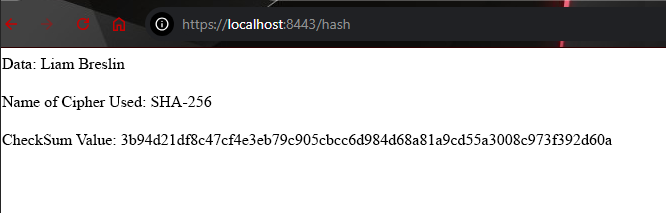
Encryption algorithms historically have come a long way being able to now address many different security challenges. Early methods such as Caesar and Substitution ciphers were simple and vulnerable to basic attacks. In the 20th century mechanical and mathematical ciphers such as Enigma and DES were introduced into the field. These ciphers were much better for general security but faded out of use when computing power in systems grew. Today some modern algorithms that are used would be AES or RSA. These two in particular offer symmetric and non-symmetric encryption. These modern encryption algorithms could stay at the top for the next 10-15 years, but could be replaced in a post-quantum cryptographic world. Quantum cryptography leverages the principles of quantum mechanics to secure communication and ensure data confidentiality. Though access to this technology is limited, it is viewed as a potential future for encryption.

SHA-256 is important in order to support the data integrity of the overall system. Hash functions are used frequently to verify data integrity. A collision would allow an attacker to input or extract malicious data, compromising security; SHA-256 is strong at combating exactly this. This can be applied to authentication for users, or even password storage. SHA-256’s strength against collisions makes it ideal for secure environments and verifying overall file integrity without a large strain on system performance.

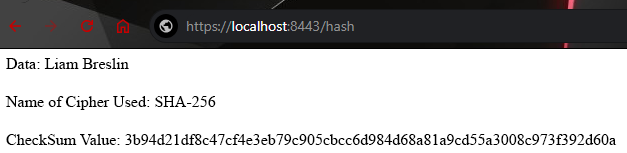
## **2.** **Certificate Generation**



## **3.** **Deploy Cipher**

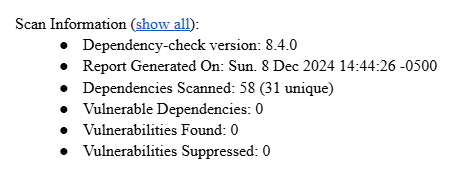


## **4.** **Secure Communications**

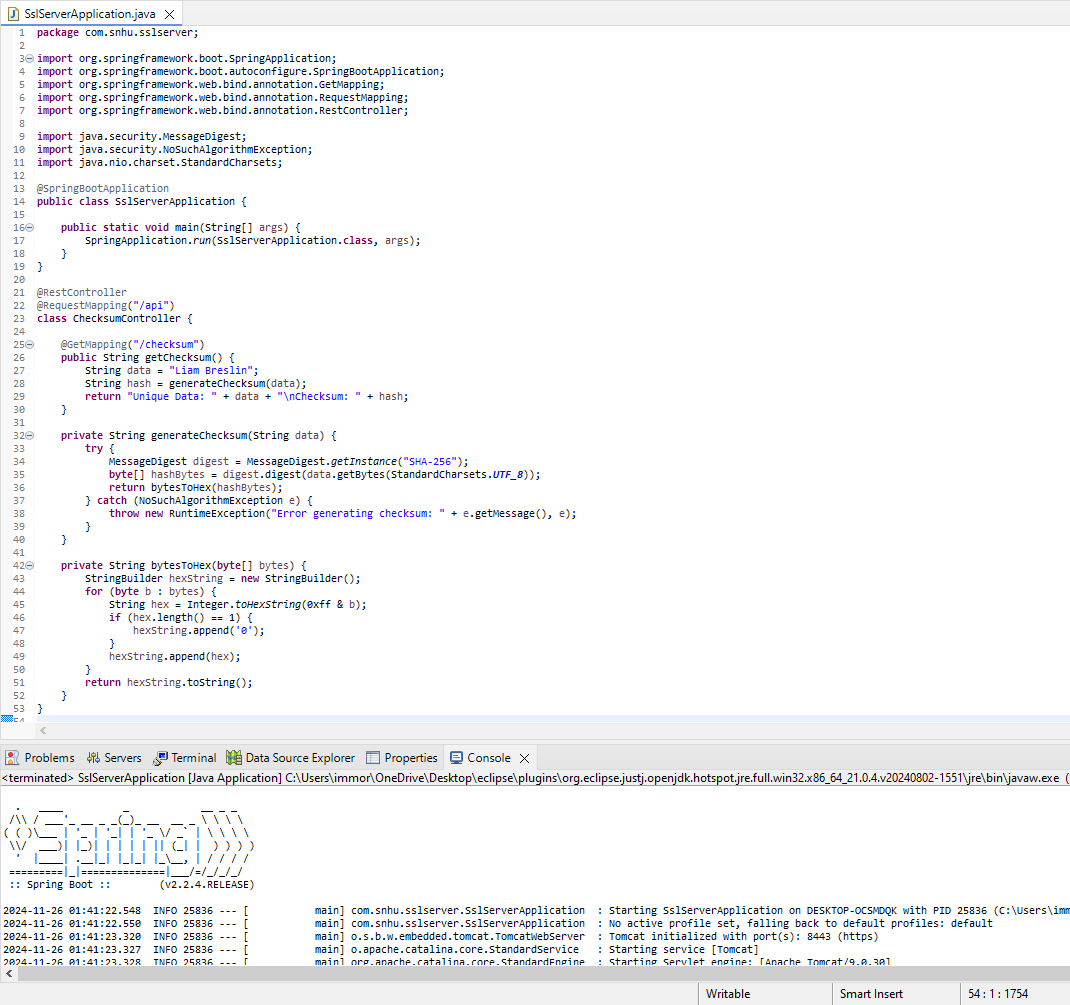


## 

## **5.** **Secondary Testing**



## **6.** **Functional Testing**



In the refactored code there are no syntax errors, as it compiles and runs properly. There are some parts of the code that I believe can lead to potential runtime errors. The API does not validate the user input for checksum generation. In terms of security the hash is returned in plain text which isn’t properly secure for professional settings. Anyone can access the /checksum endpoint which can lead to unauthorized access. Finally the application doesn’t force you to use HTTPS, which if missed can lead to MITM attacks.

## **7.** **Summary**

In the vulnerability assessment process flow diagram the 4 main areas that were addressed were Cryptography, APIs, Code Quality, and Code Errors. For cryptography we enforced hash generation by utilizing SHA-256. We further secured our API by encapsulating checksum logic. We went through the entire code base and ensured that we followed proper coding practices that deal with error and exception handling as well. Finally we were thorough with avoiding situations that could lead to potential runtime errors during hash generation.

## **8.** **Industry Standard Best Practices**

We maintain good security practices by utilizing HTTPS which will help secure data transmission between differing parties. Leveraging SHA-256 as our hash function to build resilience in our system versus both brute-force attacks and collision was very important for strong system integrity. With it we will be able to prevent many potential security breaches. Furthermore we followed standard best practices when it came to general code quality and proper sanitization and validation of our code. It’s crucial to value the importance of secure coding practices. At the most basic level following industry standards will aid in the scalability of our application far into the future. Beyond the code itself without a secure system we would lose trust from our consumers. Without consumers' confidence in our ability to keep their data safe, the system is doomed from the start. Generally following proper standards should also reduce costs, due to avoidance in both legal problems and mitigating recovery or damage expenses from security breaches.