****

# 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** | **6-23-2024** | **Charles Morris, Jr.** | **Initial version** |

## 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

**Charles Morris, Jr.**

## Algorithm Cipher

The SHA-256 algorithm is the best choice for generating checksums. This cipher takes an input of varying length and returns an output of fixed length (in this case, 256 bits).

SHA-256 is an asymmetric algorithm, meaning that two keys are used: a public key for encryption and a private key for decryption. This differs from a symmetric algorithm, which uses the same key for encryption and decryption. An example of a symmetric algorithm is AES.

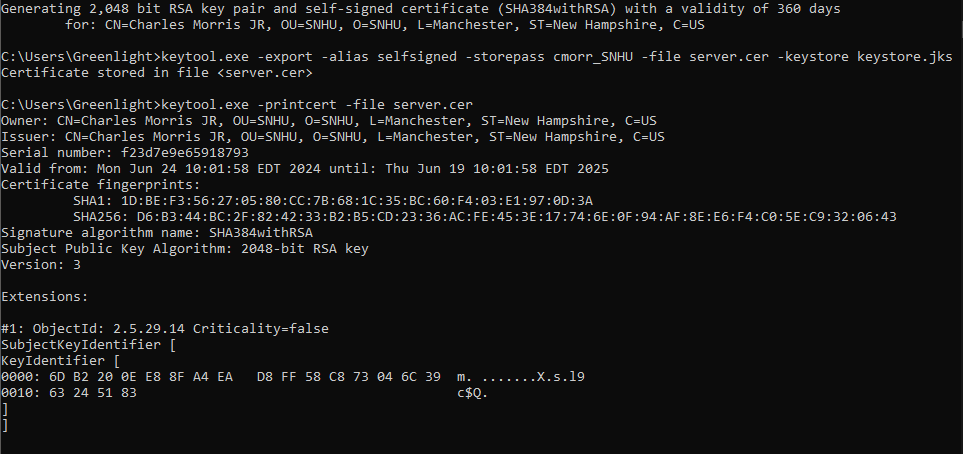
The bit level of SHA-256 is 256, meaning a 256-bit output is generated for a given input. The output is called a hash value. This makes the algorithm resistant to reverse engineering or collisions.

Random numbers are used in ciphers to create entropy, which is the degree of uncertainty in a system. This makes ciphers more unpredictable and challenging to crack.

Ciphers were created before computers existed. An example of an ancient cipher was the Caesar cipher, which encrypts messages by shifting characters in a message by three characters in the alphabet. Over time, ciphers became more complex. IBM created the DES algorithm in the 70s, eventually leading to ciphers like AES and SHA today.

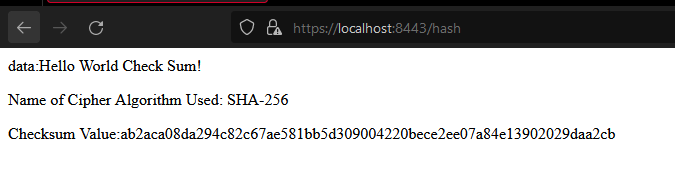
## Certificate Generation

Insert a screenshot below of the CER file.



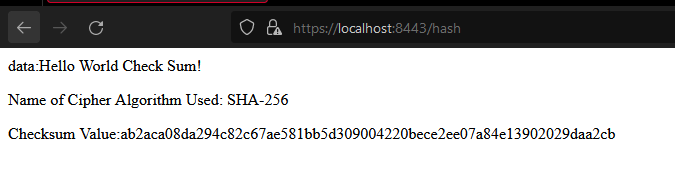
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

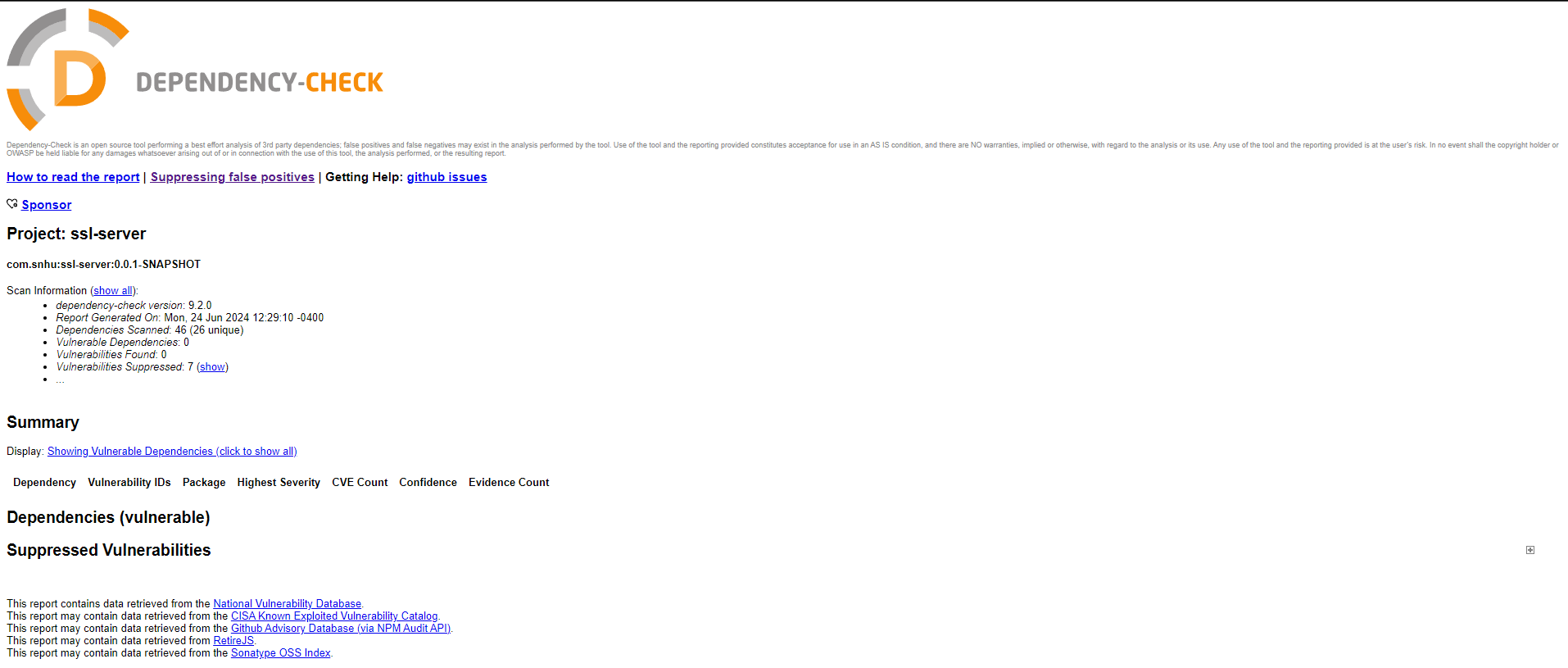
Insert a screenshot below of the web browser that shows a secure webpage.

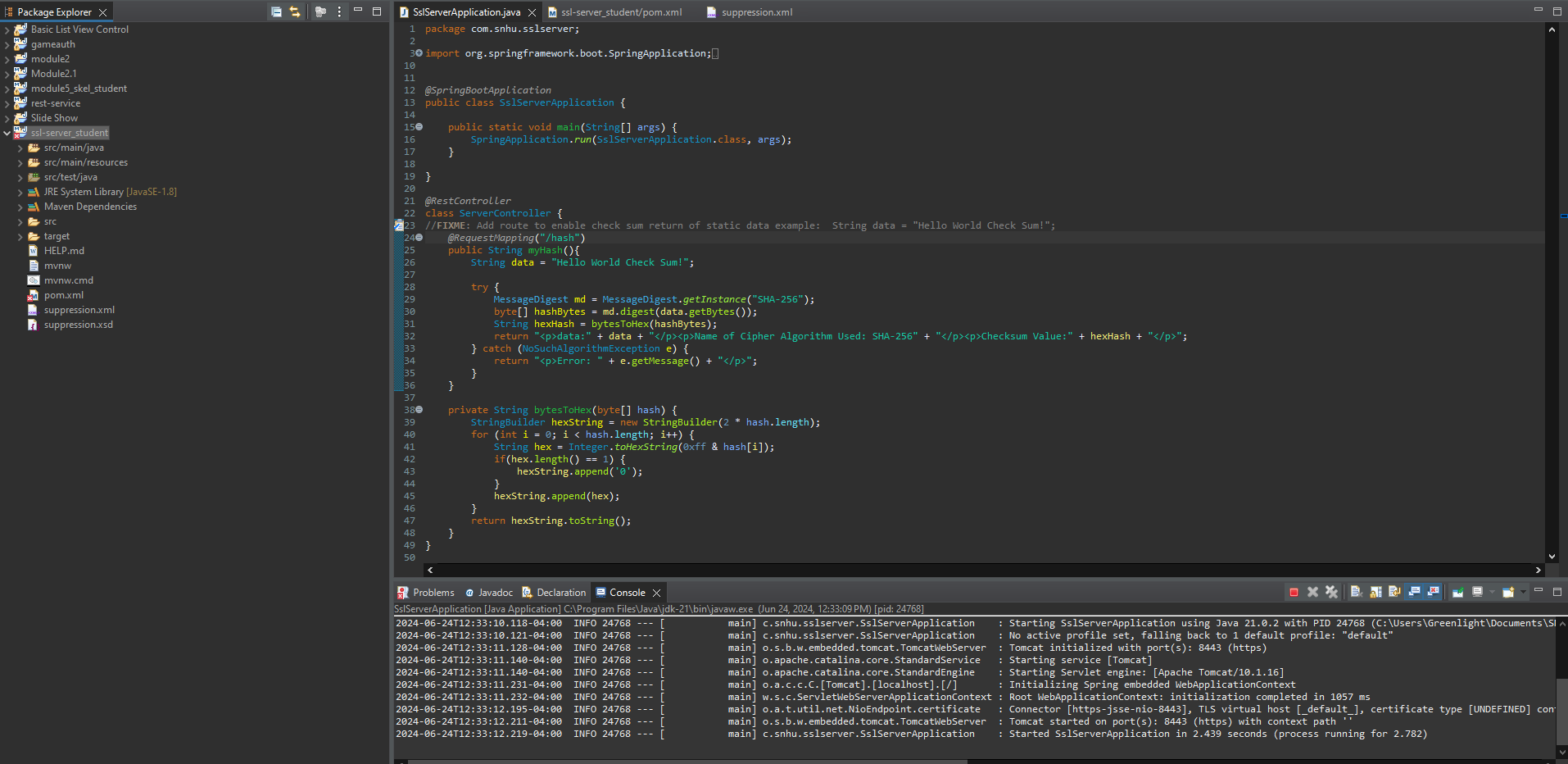


The connection is not secure because the browser does not recognize a self-signed certificate as being secure. I added an exception for the certificate so that it does not get flagged by the browser. However, the connection would be secure if the certificate were not self-signed.

## Secondary Testing

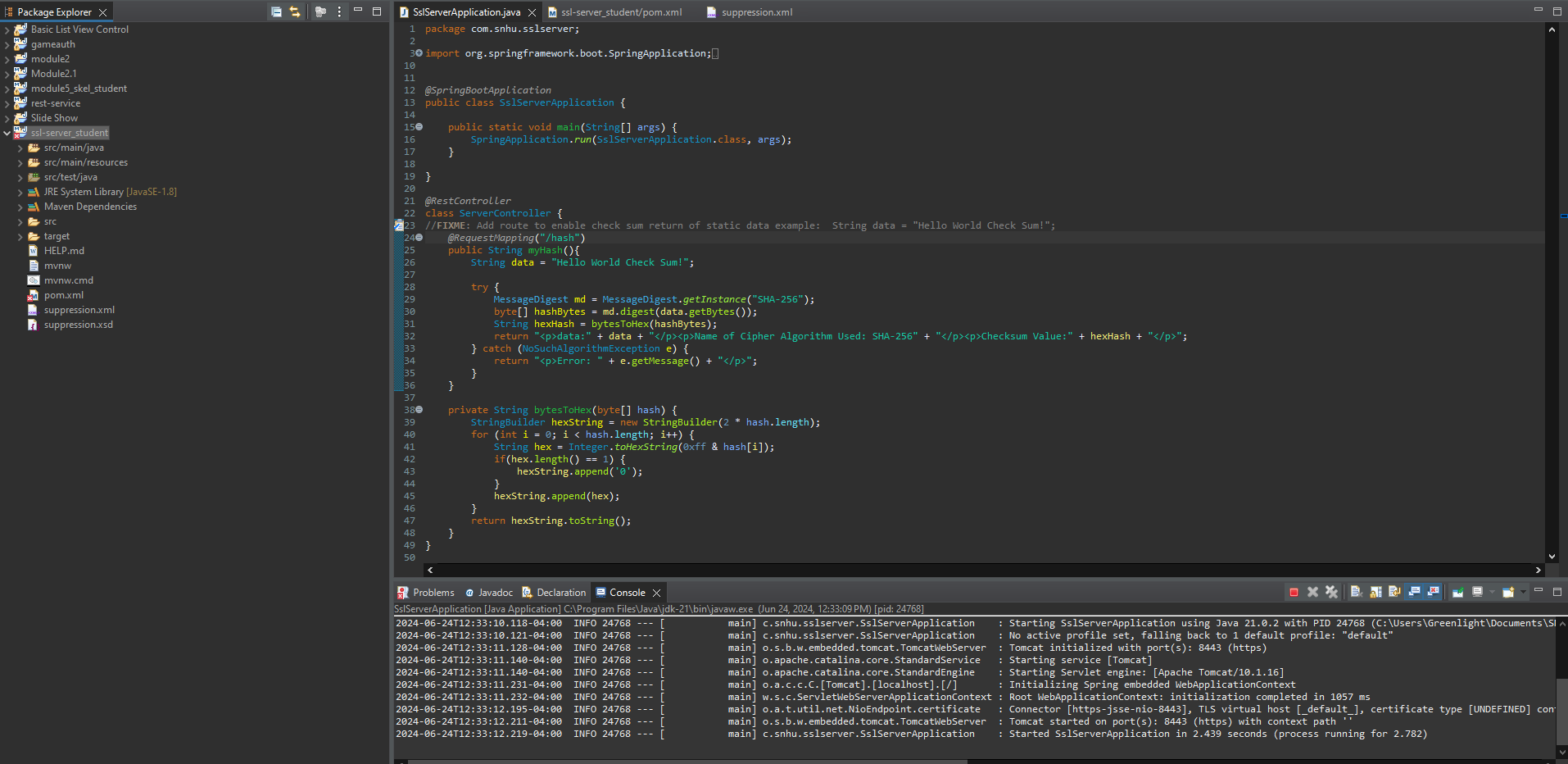
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## 

## Summary

When refactoring the code, I added a REST controller that generates a hash value and outputs it to a webpage for the client to view. This satisfies the cryptography and client/server processes of the flow chart. I placed that code in a try-catch clause, which handles the code error process. In addition, I updated the versions of the Spring Framework and Dependency Check dependencies in the POM files to the latest versions that would still allow the code to function correctly.

## Industry Standard Best Practices

I used industry-standard best practices to maintain this software application’s existing security. For the code, I used SHA-256 hashing to encrypt the data string. I also used exception handling to prevent the application from crashing if an error occurred when using the SHA-256 algorithm.

In addition to the code, I created a security certificate to allow the user to connect to the server using HTTPS, ensuring that client/server communication is secure. Then, I ran a dependency check to test for vulnerabilities while suppressing false positives that could confuse future developers.

Following best practices for secure coding is essential to saving time and ensuring projects progress smoothly. Best practices ensure minimal vulnerabilities are introduced when developing a company’s software.