****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_30j0zll)

[**Client 3**](#_1fob9te)

[**Instructions 3**](#_3znysh7)

[**Developer 4**](#_2et92p0)

[**1. Algorithm Cipher 4**](#_tyjcwt)

[**2. Certificate Generation 4**](#_3dy6vkm)

[**3. Deploy Cipher 4**](#_1t3h5sf)

[**4. Secure Communications 4**](#_4d34og8)

[**5. Secondary Testing 4**](#_2s8eyo1)

[**6. Functional Testing 4**](#_17dp8vu)

[**7. Summary 4**](#_3rdcrjn)

[**8. Industry Standard Best Practices 4**](#_26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **Oct 20th, 2024** | **Jisang Park** |  |

## 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

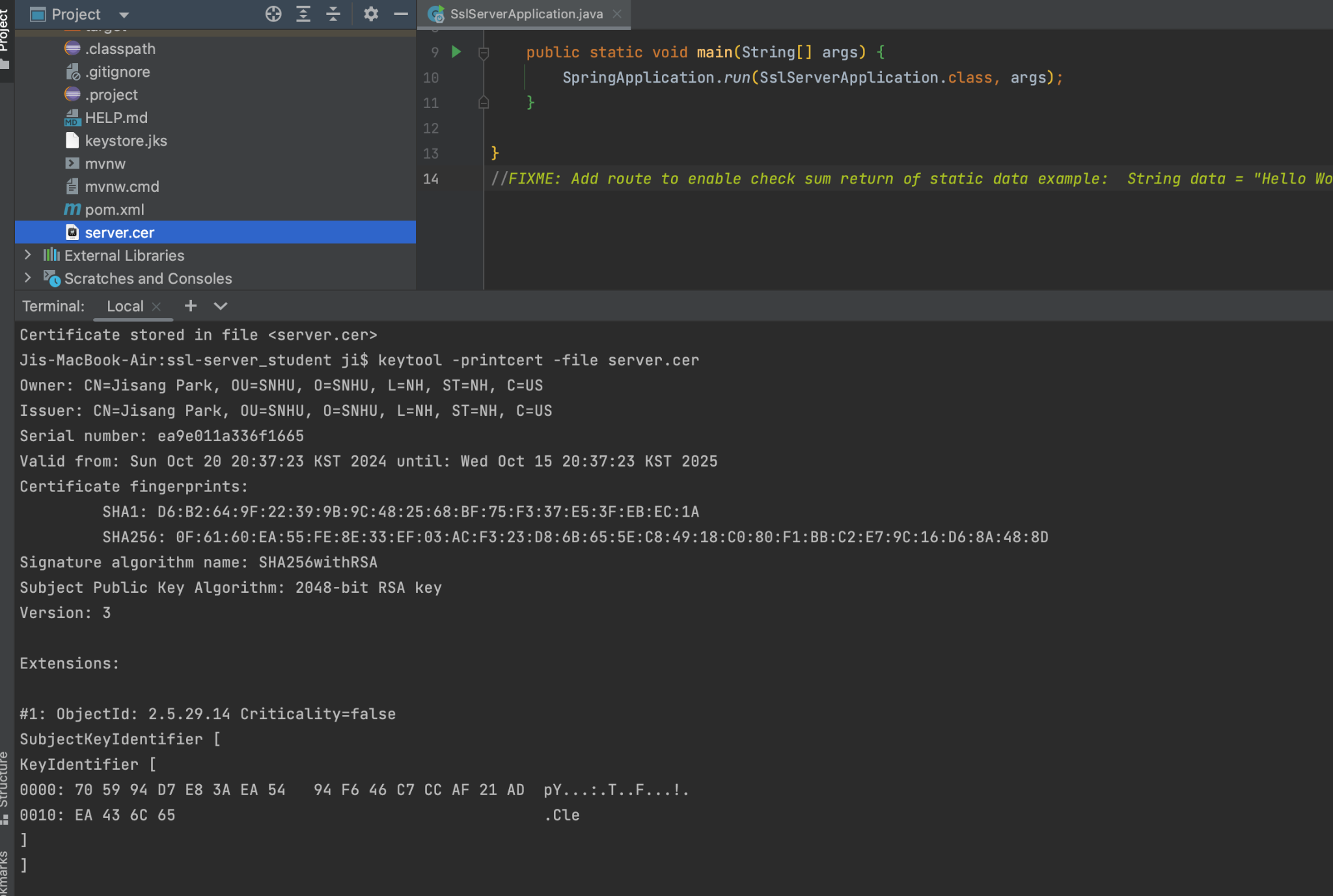
Jisang Park

## Algorithm Cipher

For Artemis Financial, I recommend using AES (Advanced Encryption Standard) 256-bit for file encryption. AES is a widely trusted symmetric encryption algorithm known for its speed and security. It operates on 128-bit data blocks and uses a 256-bit key, offering robust protection against brute-force attacks. AES relies on random numbers for key generation and ensures that encrypted data is highly secure and different each time it is encrypted. While it’s a symmetric key algorithm (the same key is used for both encryption and decryption), it is efficient for securing large amounts of data, which makes it ideal for financial applications. AES has been a global encryption standard, replacing the older DES algorithm, and is widely used in industries like finance and government.

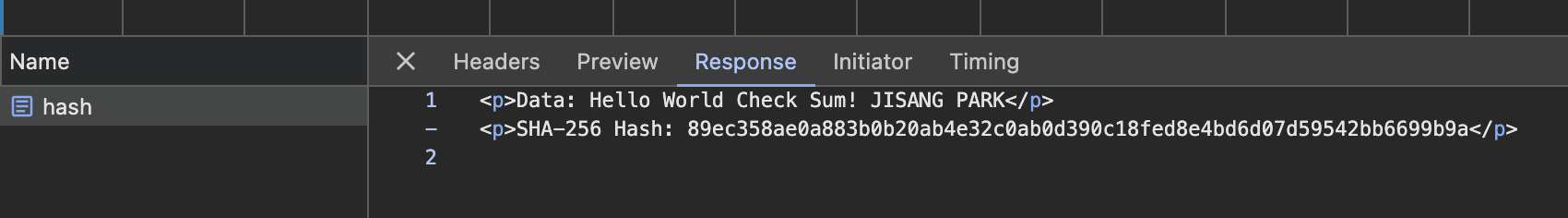
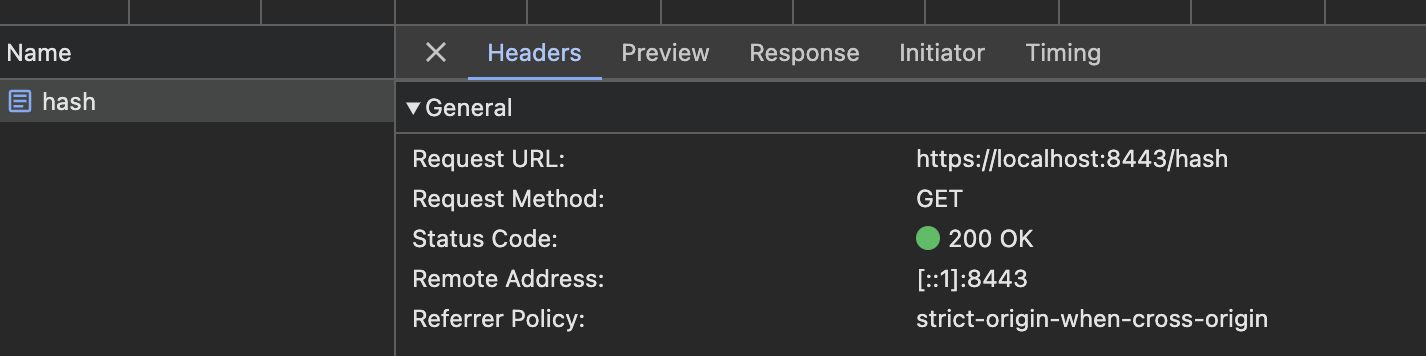
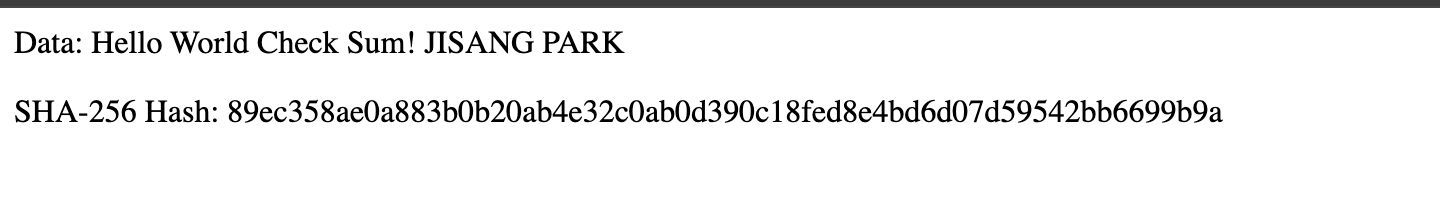
## Certificate Generation

Insert a screenshot below of the CER file.



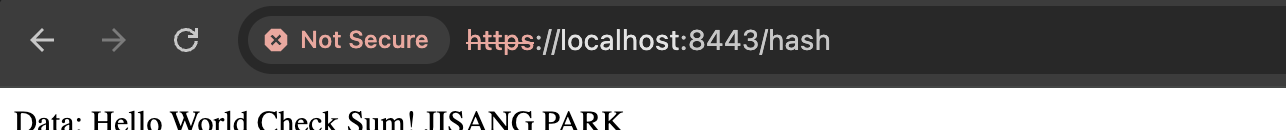
## Deploy Cipher

Insert a screenshot below of the checksum verification.



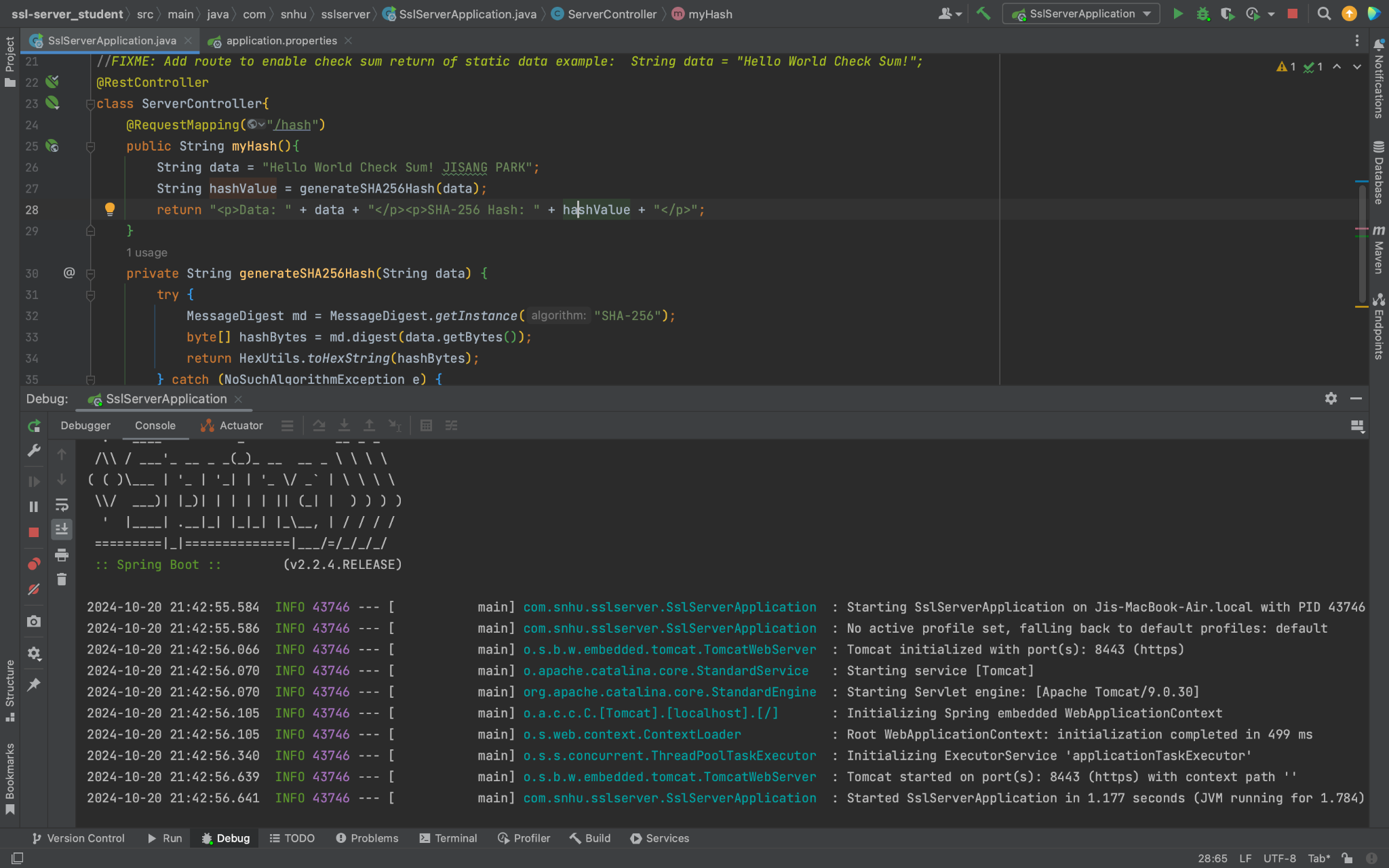
## Secure Communications

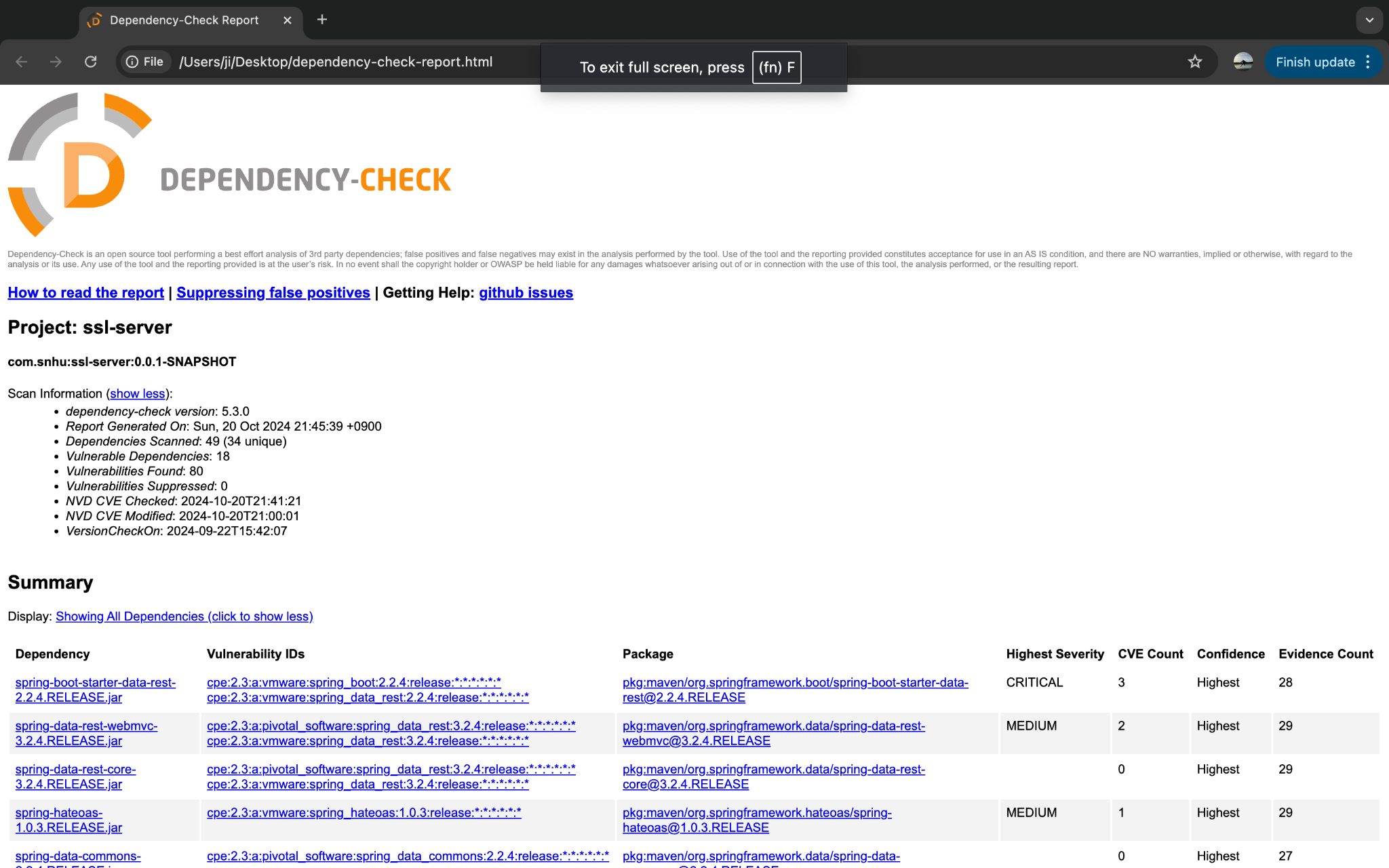
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

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



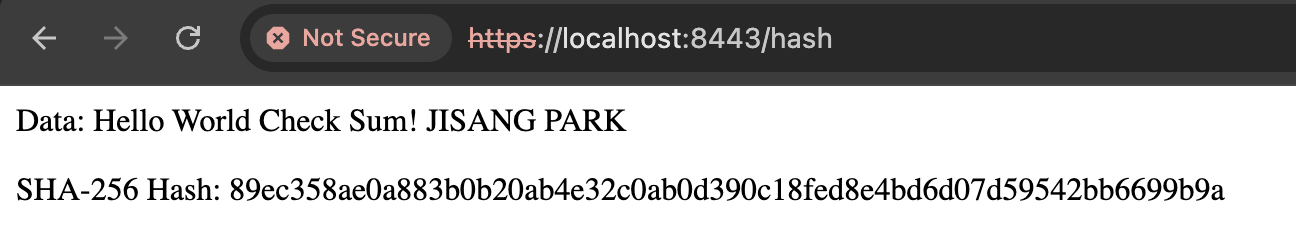
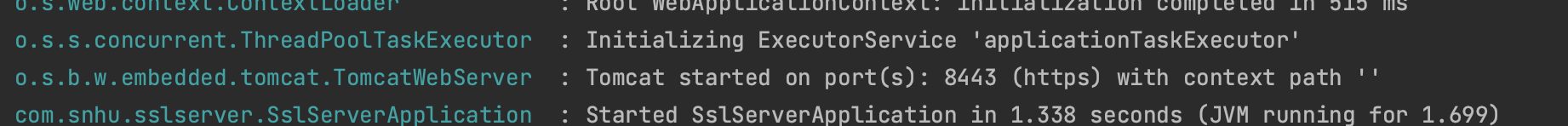


## Functional Testing

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

For the functional test, I ran the refactored code in my local development with my IDE. I accessed the “/hash” endpoint via a web browser at the URL “<https://localhost:8443/hash>”. The application successfully returned a response containing the static data string, "Hello World Check Sum! JISANG PARK," and the associated SHA-256 hash value. I verified that the SHA-256 checksum was generated correctly. Additionally, the server was configured to use HTTPS, ensuring that communication was secured during the test.

No errors occurred during the execution, and the application functioned as expected, meeting both syntactical and logical requirements. A screenshot of the successful output is included for verification.



## Summary

The refactoring process improved the application's security by implementing encryption and securing communication. I added a checksum verification using the SHA-256 hash algorithm, which ensures data integrity during transfers. Additionally, HTTPS was enabled to secure the communication channel, preventing unauthorized access and protecting sensitive information. I conducted a static analysis using the OWASP dependency-check tool to verify no new vulnerabilities were introduced. These changes address key security areas such as input validation, cryptographic handling, and secure communication, ensuring compliance with security protocols.

## Industry Standard Best Practices

In this project, I applied industry standard best practices for secure coding, such as using the SHA-256 algorithm for encryption, which is widely trusted for data integrity. I also enabled HTTPS to ensure secure communication between the server and client. Following secure coding practices like input validation and regular static analysis helped maintain the application’s existing security while minimizing the risk of introducing new vulnerabilities. By implementing these practices, the software not only protects sensitive data but also enhances the company’s overall security posture, contributing to better client trust and long term reliability.