****

# 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** | **10/15/23** | **Brian DeMaris** | **Project Two** |

## 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

Brian DeMaris

## Algorithm Cipher

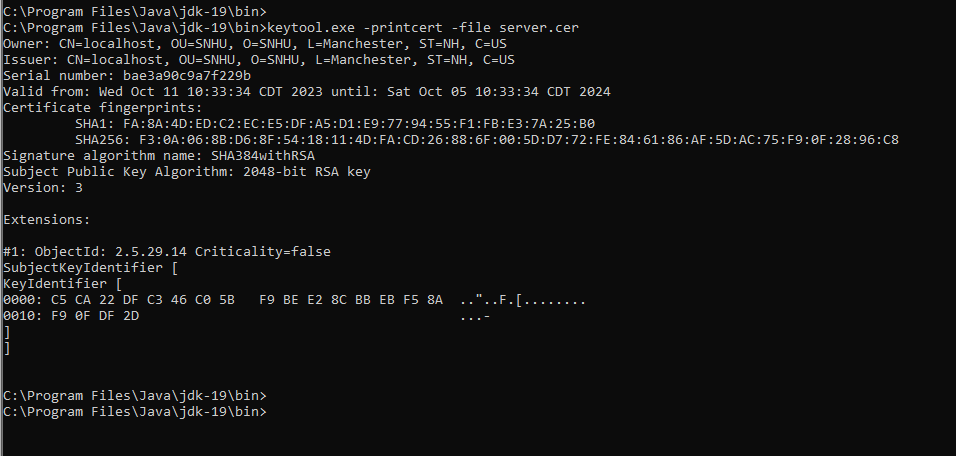
My recommendation for this project is the Advanced Encryption Standard with 256-bit keys in Galois/Counter Mode for symmetric encryption. AES is the current de facto standard. Many global legal requirements and governments recognize it as a secure solution that can meet the necessary legal requirements of these bodies. AES's widespread acceptance means more support in both hardware and software, which can mean more excellent compatibility and support.

AES supports multiple-bit levels, AES-128, AES-192, and AES-256. AES-256 is considered the most secure because its logger key length has many more possible key combinations. These random key combinations make brute-force attacks much more challenging because the longer the key, the more possible combinations. AES is a symmetric encryption algorithm that uses the same key for encryption and decryption. Non-symmetric keys use different keys for encryption and decryption. Symmetric key encryption

Modern encryption began in the 1970s with DES or Data Encryption Standard, but it was soon discovered that it was not secure. In 1977, the scientist Rivest-Shamir-Adleman introduced the first practical public-key cryptosystems, RSA, which is still used for data transmission today. Modern encryption leaped forward in the early 2000s with AES and ECC(Elliptic Curve Cryptography). AES resulted from a competition run by the National Institute of Standards and Technology to find a successor to DES.

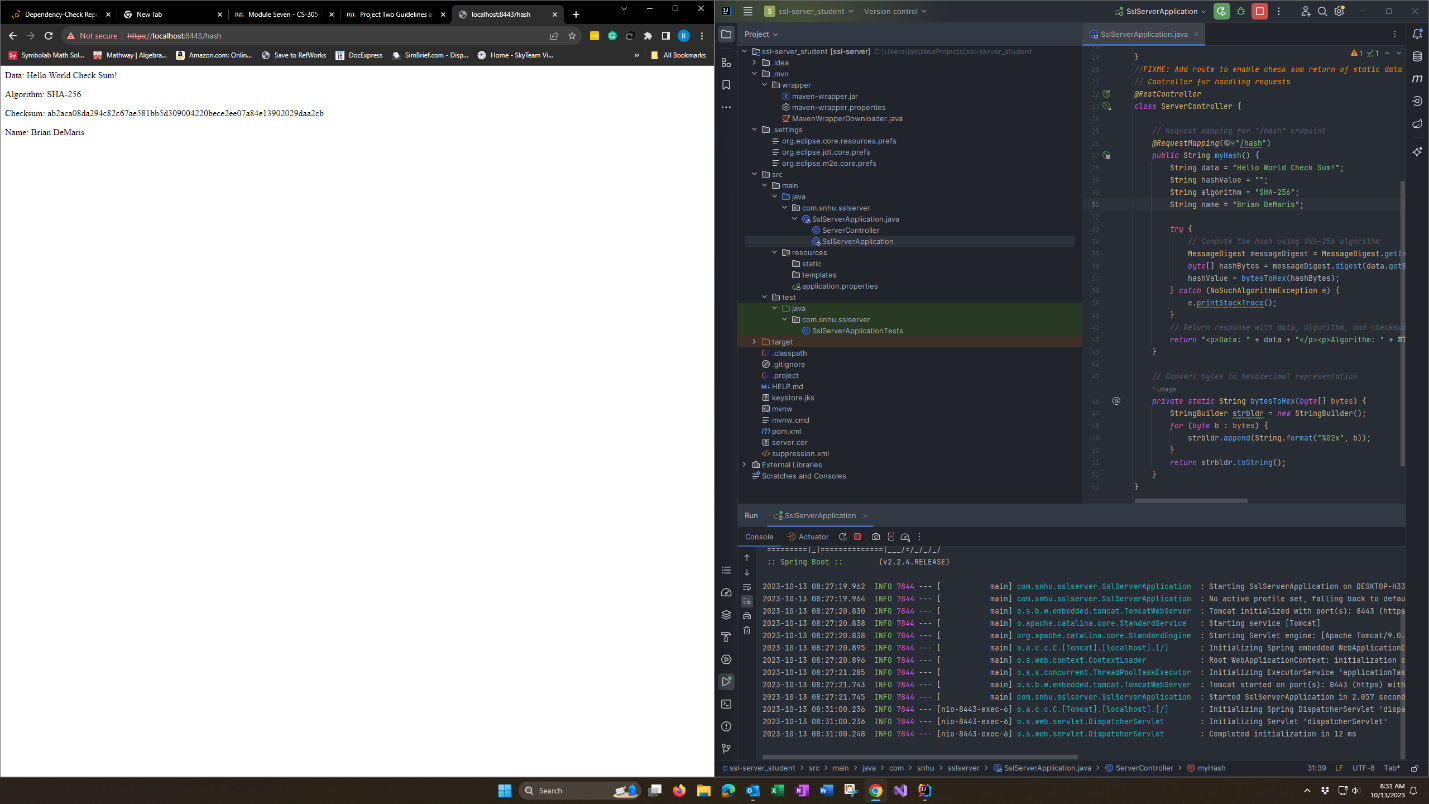
## Certificate Generation

Insert a screenshot below of the CER file.



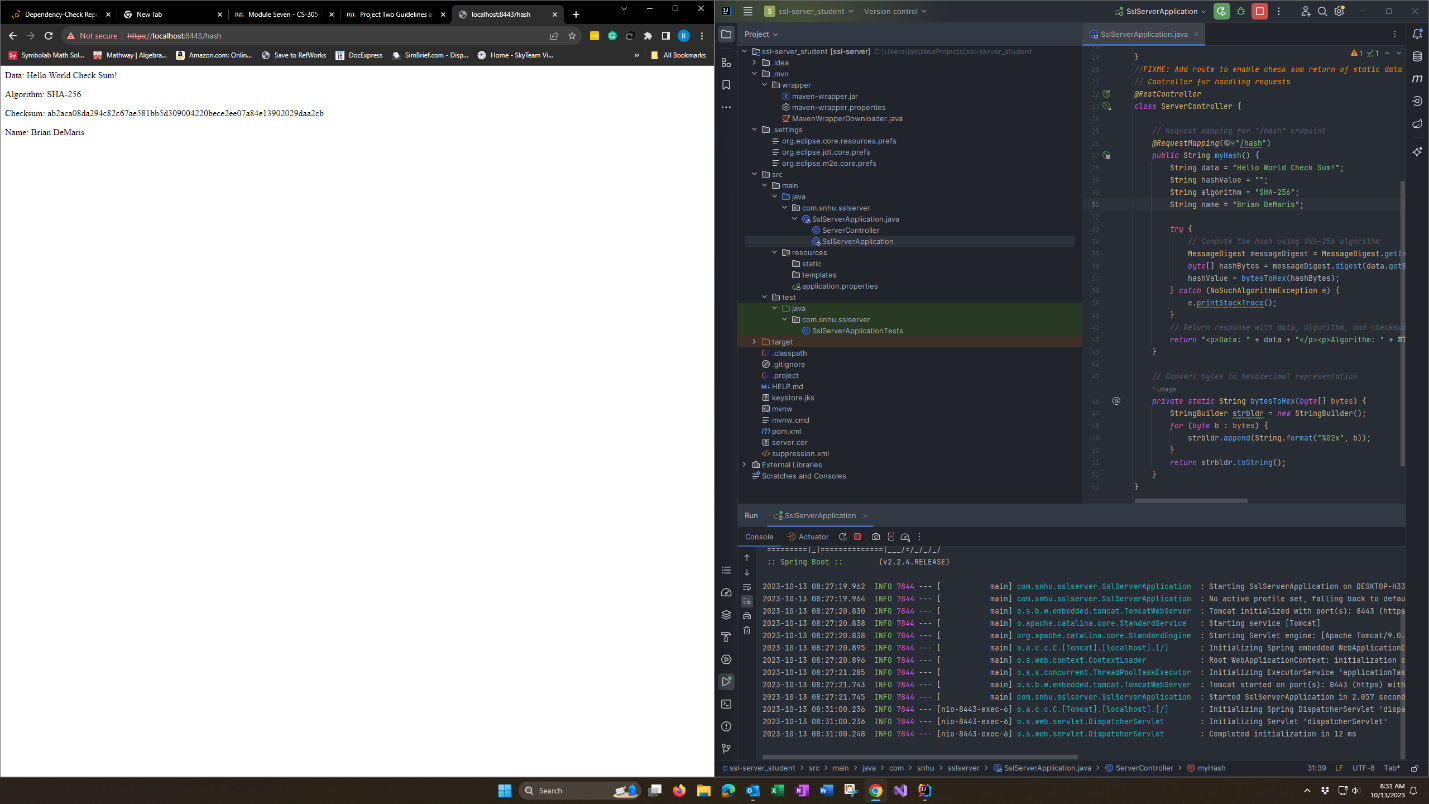
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

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

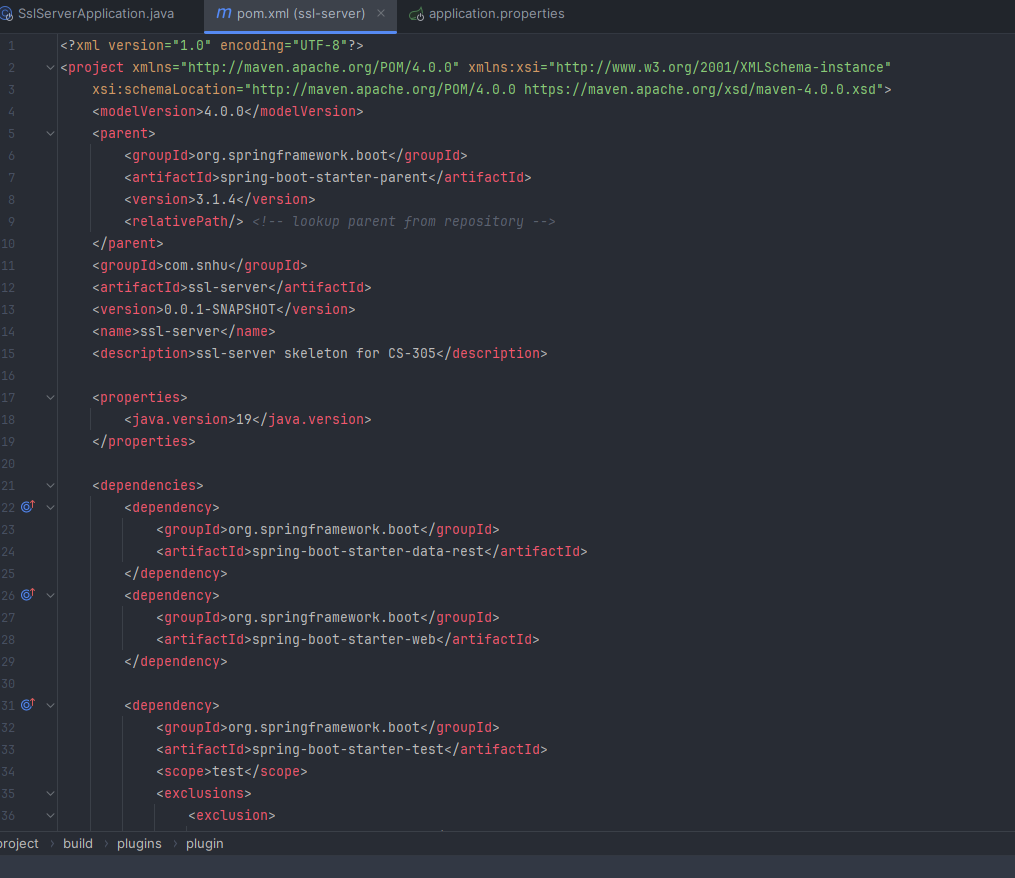


Neither of my browsers supported self-signed certificates, even with importing them into the certificate store.

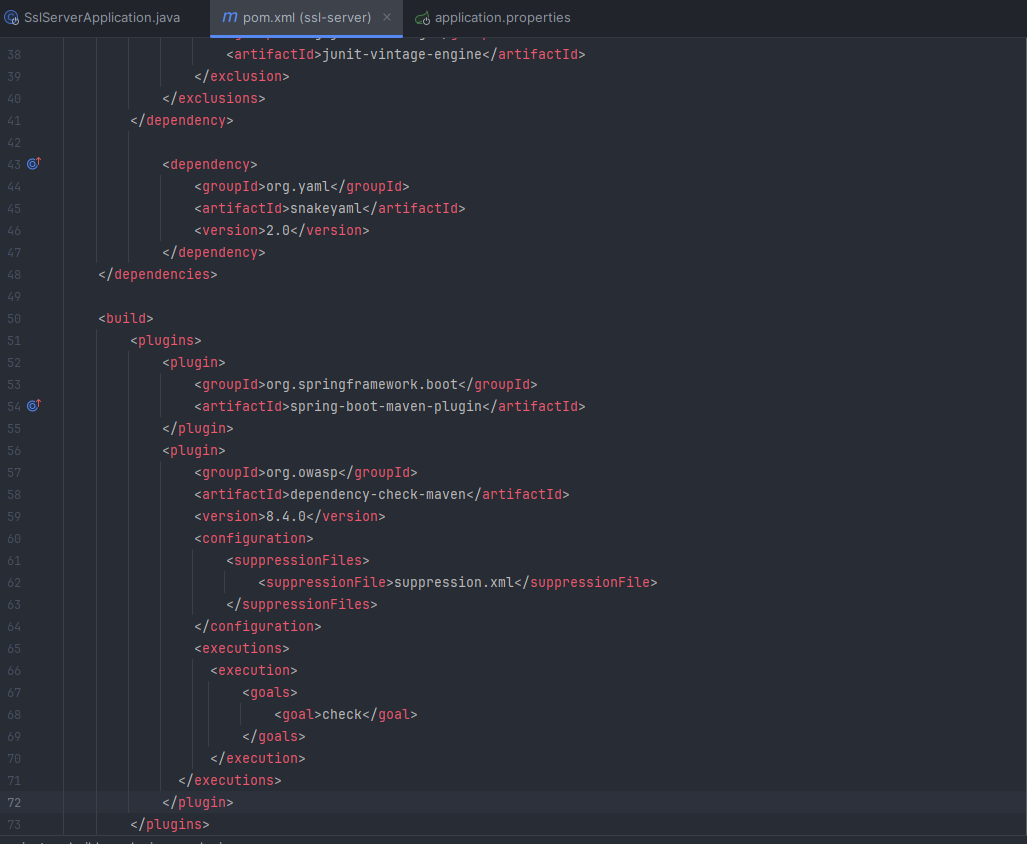
## Secondary Testing

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

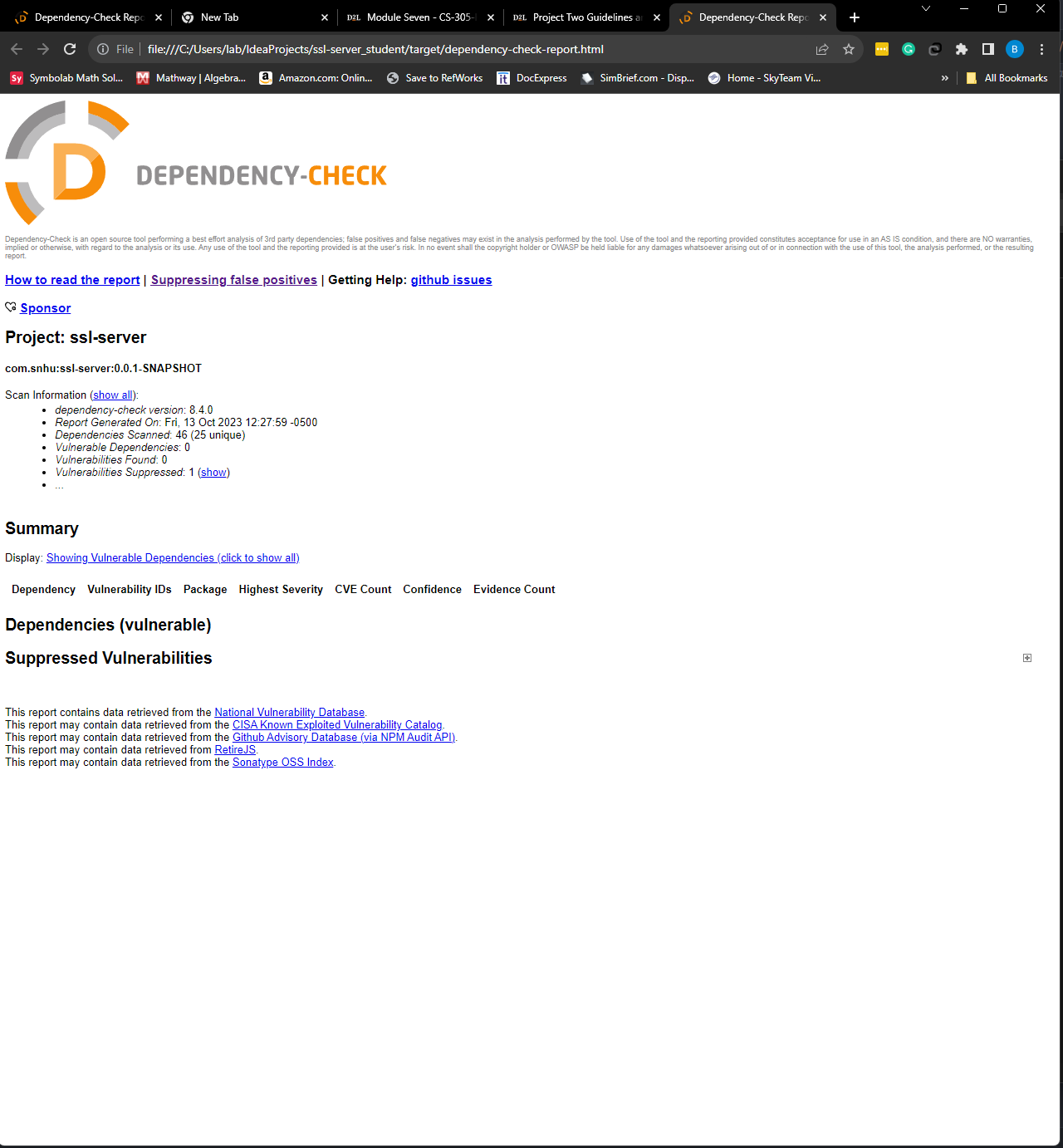
Spring-boot version updated to 3.1.4, and the Java version updated to 19.



The dependency check version was updated to 8.4.0, and the configuration for the suppression.xml file was added. Snakeyaml updated to 2.0.

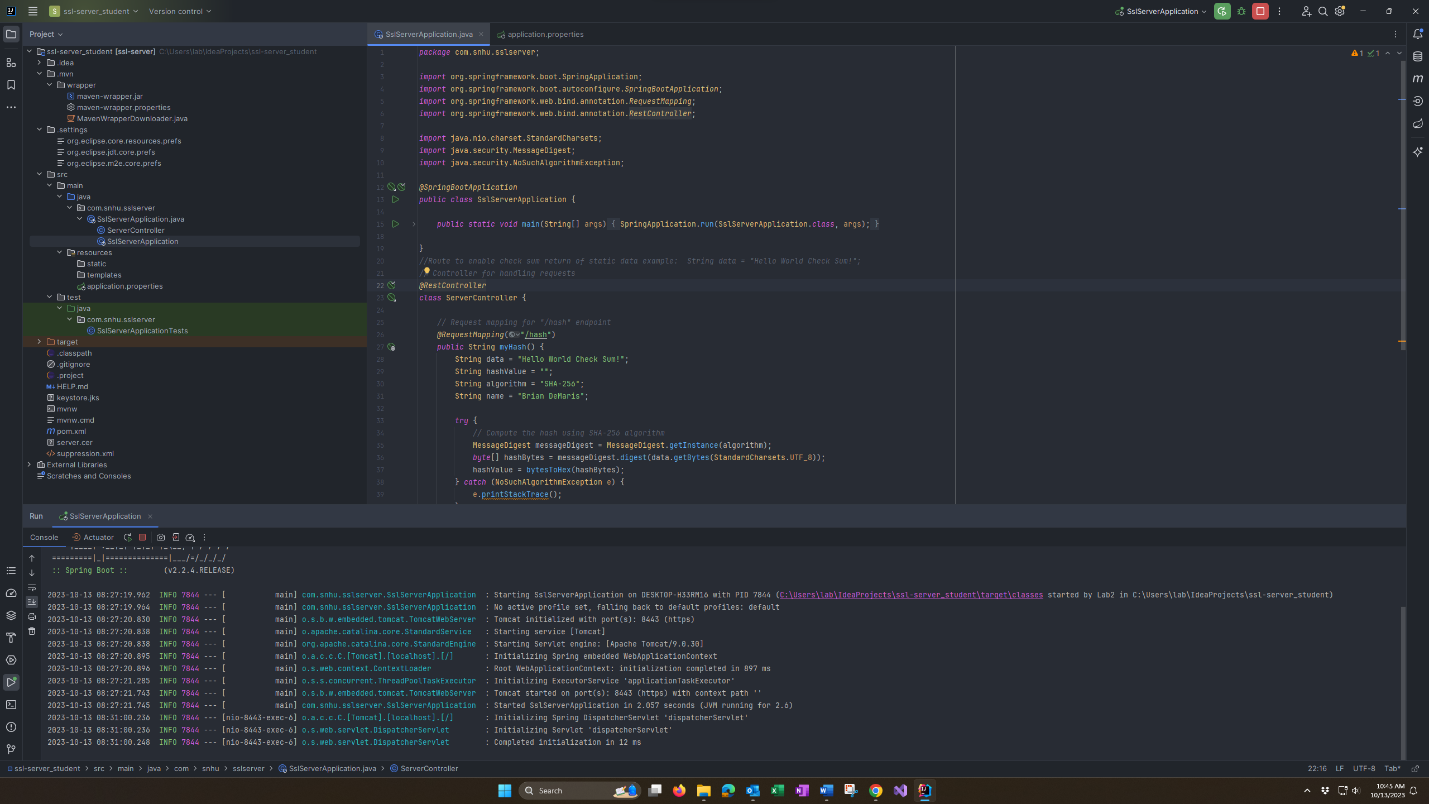


Dependency checks with all vulnerabilities corrected and 1 suppressed as the developer has disputed CVE-2023-35116 jackson-databind-2.15.2 as invalid and can not be achieved by an external attacker.



## Functional Testing

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



## Summary

The code has been refactored to correct out-of-date dependencies in Spring Boot, Java, and OWASP Dependency Check, and a suppression file was added to address a false positive. Code was also added to enable SSL by refactoring the application properties with the necessary SSL info. The SslServerApplication was refactored with a RestController that used SHA-256 to compute a hash converted to hex, then printed "Hello World Check Sum!", the hash value, algorithm type, and name.

These vulnerabilities affected many Vulnerability Assessment Process Flow diagram areas, such as APIs, Cryptography, Client/Server, and encapsulation. As with many vulnerabilities, the solution is to keep our dependencies and software up to date.

## Industry Standard Best Practices

Software security encompasses critical aspects such as vulnerability identification and mitigation, secure coding practices, and robust cryptographic measures. One crucial aspect is conducting regular vulnerability assessments using tools like OWASP Dependency-Check to scan software dependencies for known vulnerabilities. This proactive approach helps identify and address potential security flaws early in the development lifecycle, reducing the risk of exploitation. Another vital component is employing robust hash algorithms for data integrity and secure storage. Robust hash algorithms, like SHA-256, ensure data remains unchanged and secure against malicious tampering.

Cryptography is pivotal in software security by ensuring data confidentiality, integrity, and authentication. Implementing industry-standard encryption algorithms such as AES (Advanced Encryption Standard) for sensitive data helps protect information from unauthorized access. It's imperative to design systems with a comprehensive understanding of potential vulnerabilities, employ secure coding practices, and apply secure development methodologies like Secure SDLC (Software Development Life Cycle) to embed security from the ground up. This approach significantly contributes to creating resilient software that can withstand evolving cyber threats and maintain users' trust.