****

# 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** | **04/19/24** | **Miguel Hilario** | **Version 1** |

## Client



## Instructions

Submit these 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

Miguel Hilario

## Algorithm Cipher - Advanced Encryption Standard (AES)

⦁ AES is a symmetric encryption algo widely used for securing sensitive data in various applications. Federal standard for Data Encryption Standard (DES) established by the National Institute of Standards and Tech (NIST). AES operates on blocks of data and supports keys of 128, 192, and 256 bits. It uses substitution permutation network (SPN), which involves multiple iterations of substitution and permutation. It is efficient and standardized, available across various programming languages and operating platforms.

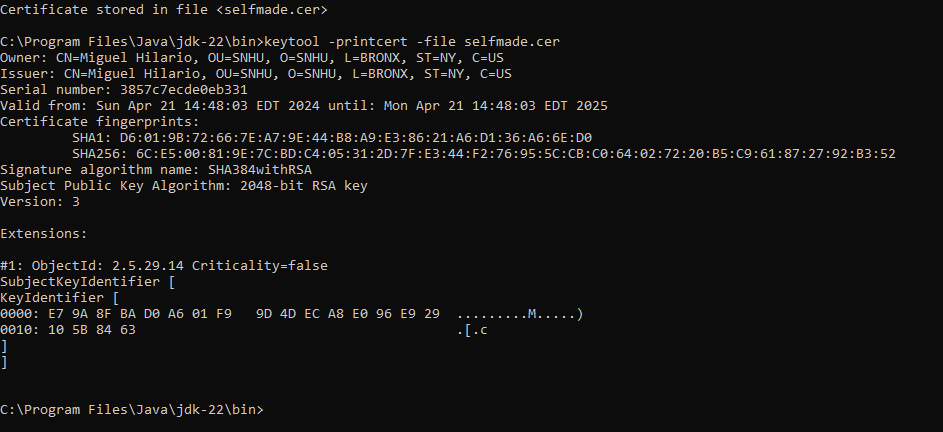
⦁ Hash functions like SHA-256, SHA-384, or SHA-512 can be used with AES for added security. With this you can generate checksums or message authentication codes (MAC). AES operates on fixed sizes of plain/hypertext which are usually 128 bits. Longer keys provide a stronger encryption.

⦁ AES generates random numbers for key generation. AES is symmetric meaning the same key is used to decrypt and encrypt. Simplifies the management of keys however, distribution would need to be appropriately handled to avoid copies and mismatches. In an asymmetric cipher, there is a public and private key, and you cannot obtain the private key via the public key. Public keys are used openly and encrypt data, public keys have corresponding private keys. Only when the corresponding key is used may the data be decrypted.

⦁ Encryption has been around for a long time starting with substitution and transposition methods. War and technology took encryption to another level considering the highly classified data that would need to be transmitted during the height of World War 2. DES was established to hold a federal standard for encryption which was initially 56 bits only, currently the standard is AES cipher typically at 128 bits but offering 192 and 256 for higher levels of security.

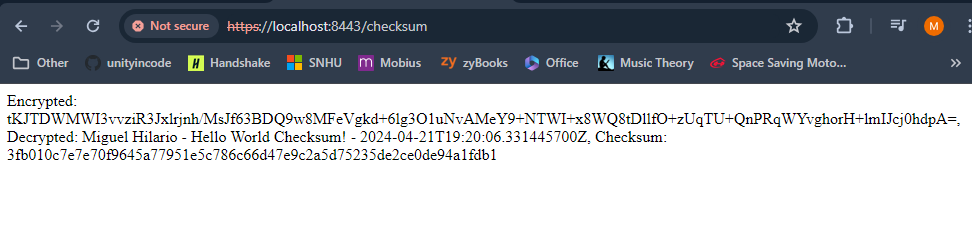
## Certificate Generation

Insert a screenshot below of the CER file.



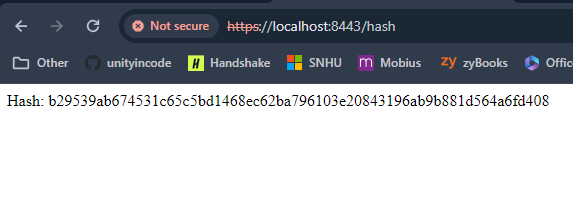
## Deploy Cipher

Insert a screenshot below of the checksum verification.



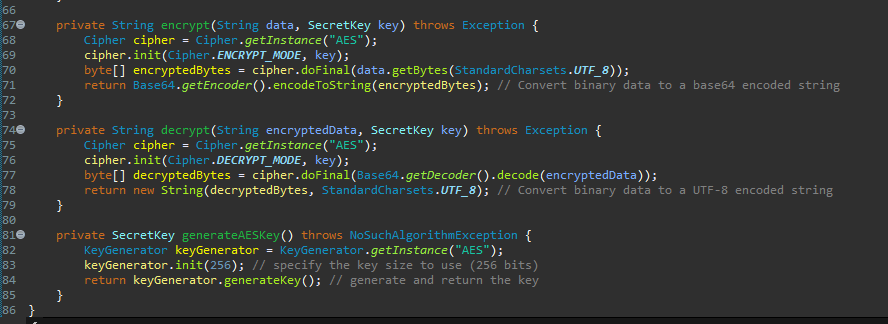
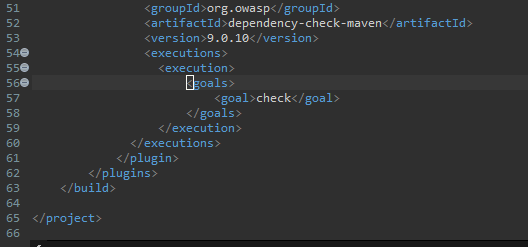
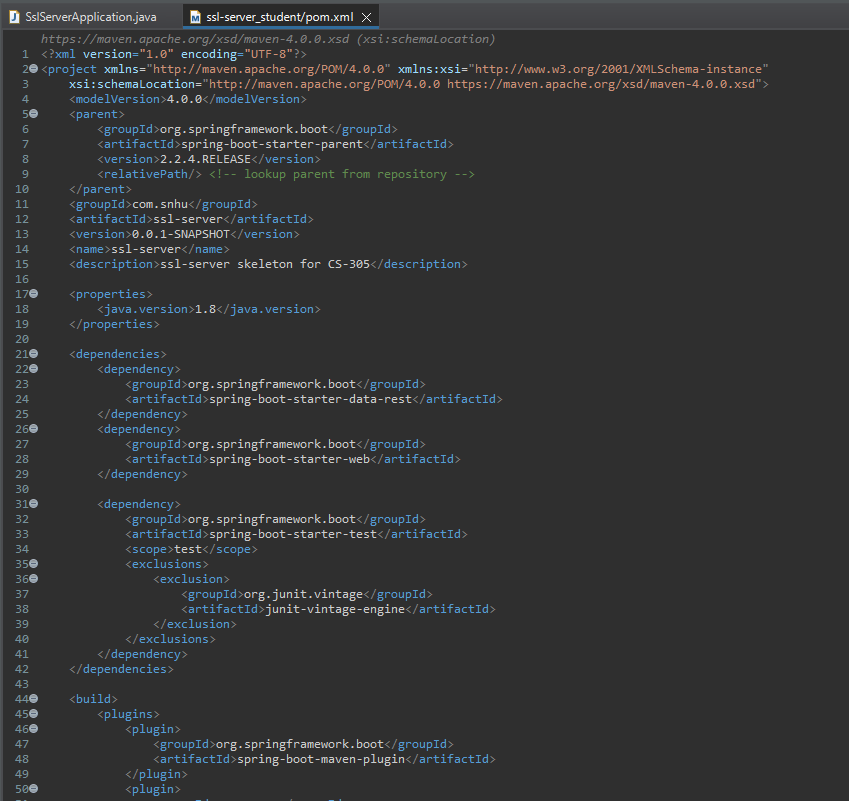
## Secure Communications

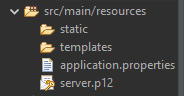
Insert a screenshot below of the web browser that shows a secure webpage.

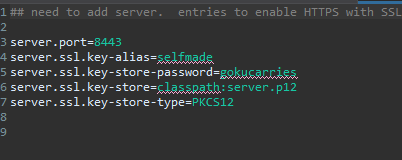


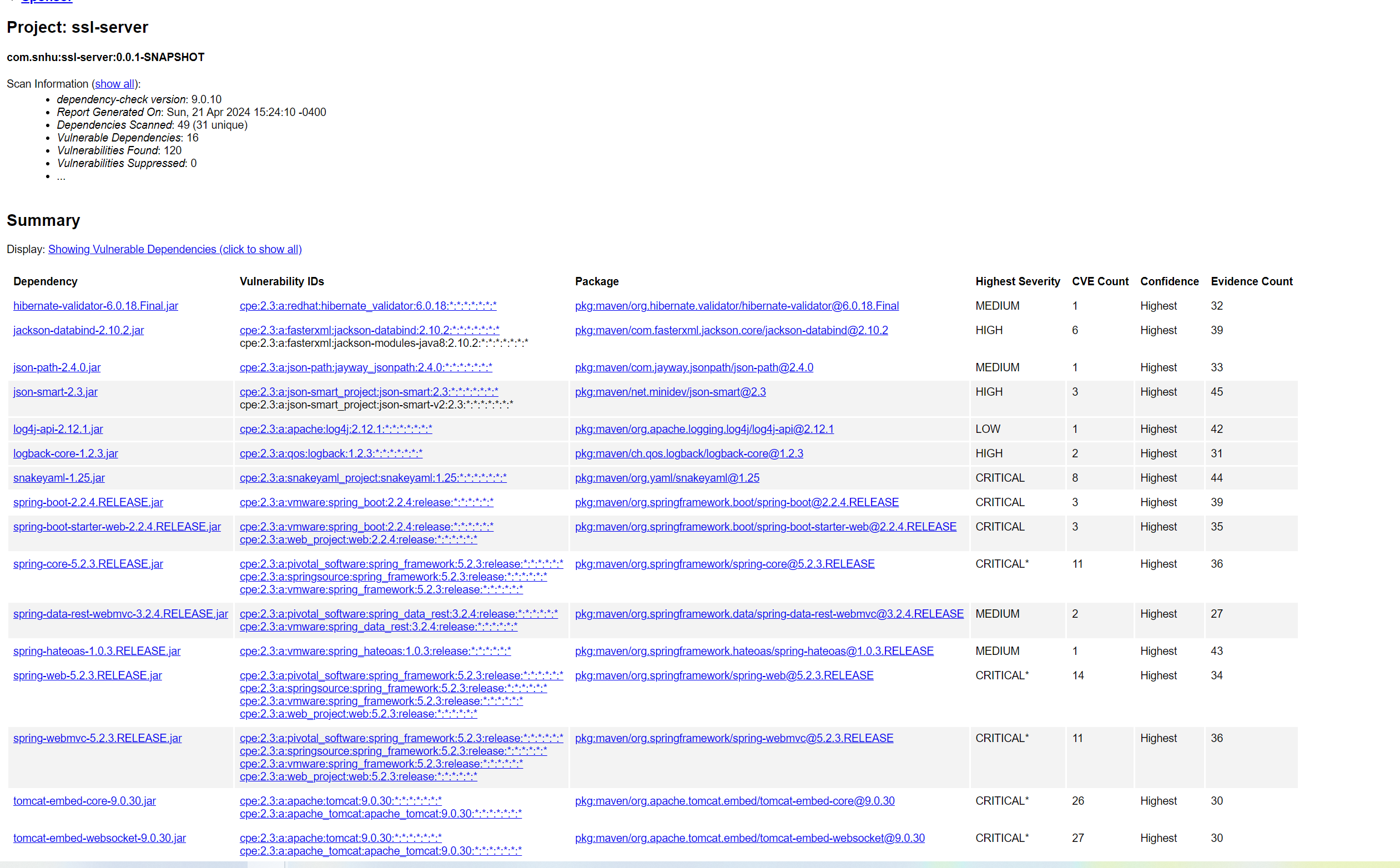
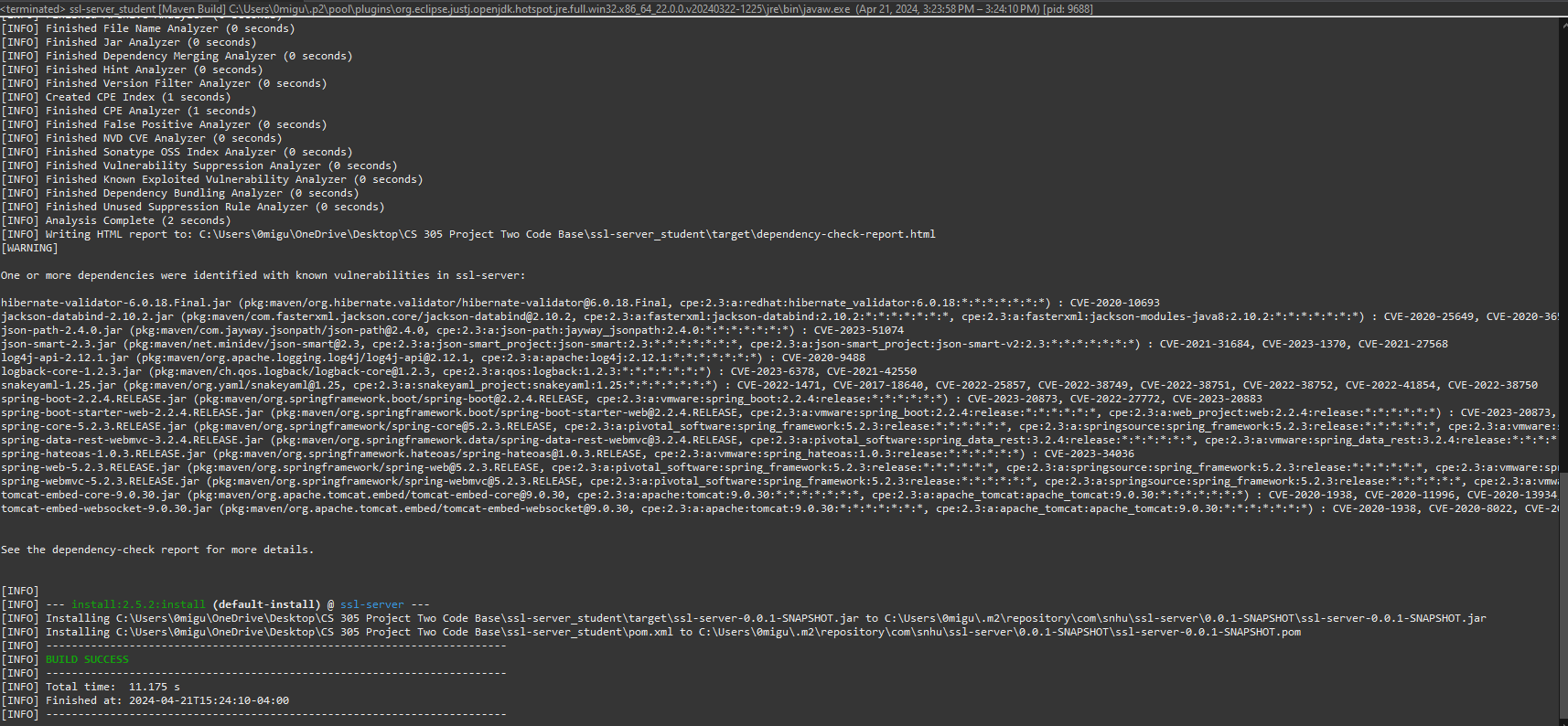
## Secondary Testing

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



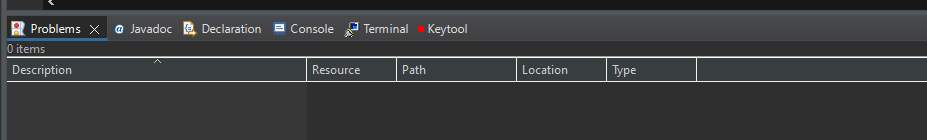
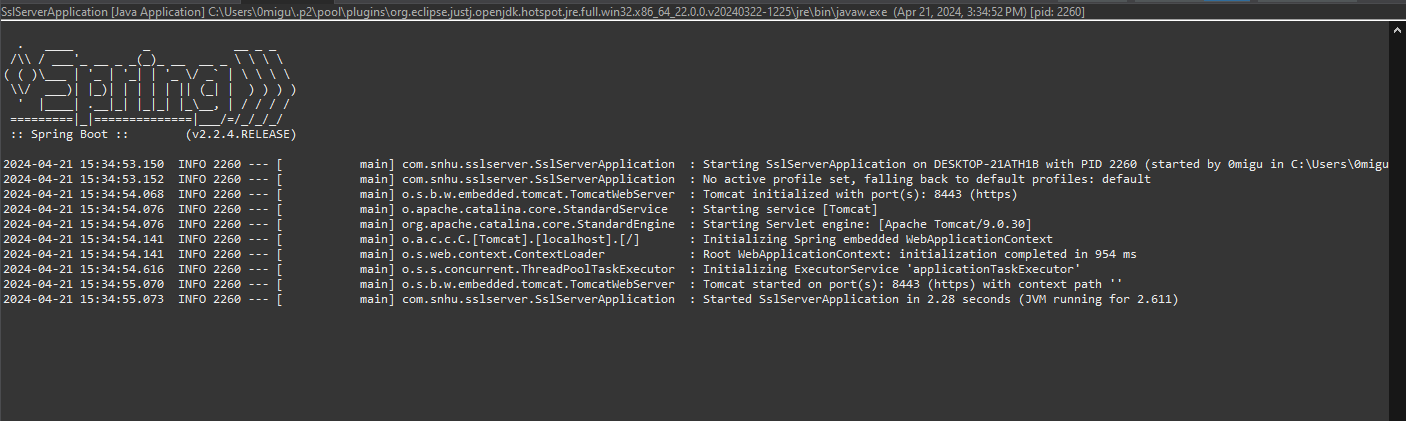
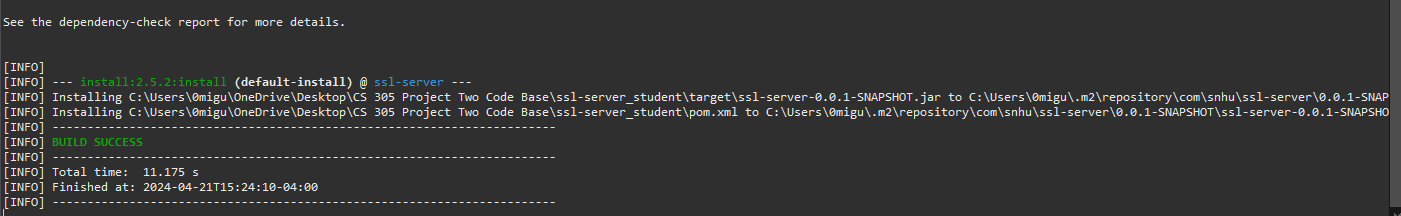






## Functional Testing

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



## Summary

In my efforts to enhance the security of the ssl-server appl, I completed a comprehensive refactoring process, using principles outlined by the Vulnerability Assessment Process Flow Diagram. My focus was to fortify areas of the application that handle sensitive data, manage secure communications, and update dependencies to mitigate known vulnerabilities. One of the significant changes was the implementation of Advanced Encryption Standard (AES) for data encryption and Secure Hash Algorithm (SHA-256) for checksums. This measure ensures that data integrity and confidentiality are maintained throughout the application's workflow. To comply with secure transmission protocols, I configured the application to operate exclusively over HTTPS, creating and integrating a secure keystore. This step was crucial in guaranteeing that all data in transit between the client and server is encrypted and secure from potential eavesdropping or man-in-the-middle attacks. A pivotal part of the security enhancement process was updating the project dependencies. Utilizing OWASP Dependency-Check, I identified and updated outdated libraries to their latest, most secure versions. I adhered strictly to security testing protocols, employing both static and dynamic analysis to test the application. Static analysis was conducted using OWASP Dependency-Check, pinpointing critical vulnerabilities in the dependencies. Dynamic analysis involved running the application in a test environment to identify potential runtime security issues.

Also, I performed a meticulous manual code review. This manual inspection was key in discovering and correcting syntactical, logical, and potential security vulnerabilities that automated tools might not have detected, hence reinforcing the robustness of the application against various attack vectors.

Also, I executed functional testing to validate that the implemented security mechanisms operated correctly. The application's response to both regular and anomalous inputs was tested, confirming not only its functional strength but also its protection against potential security breaches. The application's security has been significantly elevated through layered security measures, continuous assessment, and adherence to secure coding practices. This ensures that the ssl-server application stands up to rigorous security standards and aligns with the best practices of secure software development.

## Industry Standard Best Practices

Maintaining Current Security (Steps Taken):

* Regular Dependency Updates: I integrated tools like OWASP Dependency-Check to ensure that all third-party libraries and frameworks are kept up to date with the latest security patches. This practice is crucial in protecting the application against known vulnerabilities that are often exploited in outdated components.
* Encryption and Hashing: Implementing AES for encryption and SHA-256 for hashing ensured that data is stored and transmitted securely. This practice is in line with industry standards that recommend strong encryption algorithms and hashing functions to safeguard information.
* HTTPS Configuration: I enforced the use of HTTPS, backed by a properly configured SSL certificate, to ensure secure data transmission. This approach aligns with the industry’s secure by default communications.
* Code Review and Analysis: I performed manual and automated code reviews to detect and correct security flaws, adhering to secure coding guidelines. These reviews helped identify issues such as code injection risks and improper error handling that automated tools could miss.
* Input Validation and Sanitization: I incorporated input validation and sanitization throughout the application to protect against common web vulnerabilities like SQL injection, XSS, and CSRF.

Value to the Company’s Wellbeing:

* Trust and Reputation: Security breaches can severely damage a company's reputation. By adhering to secure coding practices, the company demonstrates a commitment to protecting its customers' data, which builds trust with the customers.
* Legal and Regulatory Compliance: Many industries have strict regulatory requirements for data security. Following best practices ensures compliance with laws such as GDPR, HIPAA, and others, which can prevent costly legal penalties and sanctions.
* Prevention of Financial Loss: Security incidents can result in significant monetary loss, both from the immediate effects of a breach and from the long-term impact on business operations (trust). Secure coding practices minimize the risk of such incidents.
* Operational Continuity: Secure coding helps ensure the integrity and availability of the company's services. By mitigating vulnerabilities, the company protects against attacks that could disrupt business workflow.
* Competitive Advantage: A strong security record could be the decision factor for potential customers. Companies that prioritize security can leverage this in their marketing and customer engagement efforts.