****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.30j0zll)

[**Client 3**](#_heading=h.1fob9te)

[**Instructions 3**](#_heading=h.3znysh7)

[**Developer 4**](#_heading=h.2et92p0)

[**1. Algorithm Cipher 4**](#_heading=h.tyjcwt)

[**2. Certificate Generation 4**](#_heading=h.3dy6vkm)

[**3. Deploy Cipher 4**](#_heading=h.1t3h5sf)

[**4. Secure Communications 4**](#_heading=h.4d34og8)

[**5. Secondary Testing 4**](#_heading=h.2s8eyo1)

[**6. Functional Testing 4**](#_heading=h.17dp8vu)

[**7. Summary 4**](#_heading=h.3rdcrjn)

[**8. Industry Standard Best Practices 4**](#_heading=h.26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/18/24** | **Shawn Henly** |  |

## 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

Shawn Henly

## Algorithm Cipher

For the SSL Server application, I recommend implementing the Advanced Encryption Standard (AES) as the encryption algorithm cipher. AES is a symmetric encryption standard that is broadly utilized due to its strong security and operational efficiency, making it ideal for scenarios demanding both high performance and rigorous data security. It is frequently employed to safeguard sensitive information both during transmission and while stored.

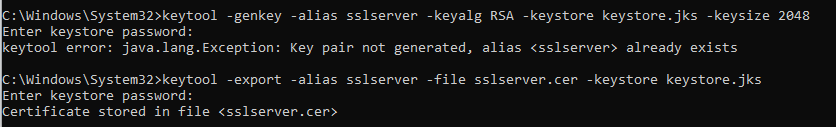
AES utilizes hash functions with key lengths of 128, 192, or 256 bits, offering different degrees of security. With increasing key length, the encryption strength enhances, thereby fortifying it against brute-force attacks. AES functions on symmetric keys, meaning the same key is applied for both encrypting and decrypting data, facilitating efficient and secure data transfers in trusted settings.

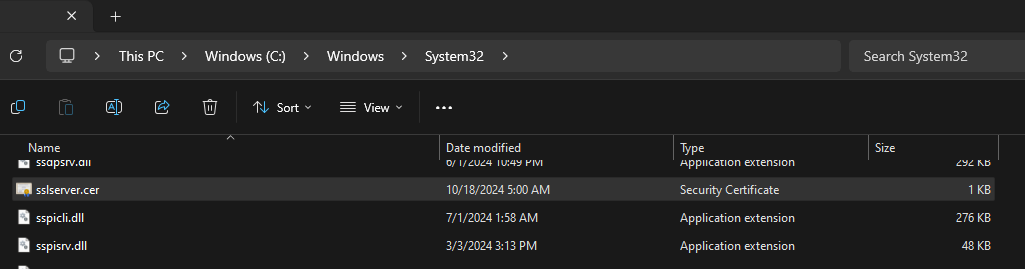
To increase security, AES includes random numbers in its initialization vector (IV) to prevent identical plaintext from yielding the same ciphertext, thereby boosting the security level of the encrypted data. Compared to asymmetric key algorithms like RSA, symmetric key encryption such as AES is generally quicker and more efficient.

Previously, encryption protocols like the Data Encryption Standard (DES) were standard, but advancements in computational capabilities rendered DES susceptible to brute-force assaults. AES superseded DES as the encryption standard endorsed by the U.S. government due to its superior security features. Today, AES continues to be the prevailing standard for encryption across various sectors and applications, attributed to its optimal mix of security, efficiency, and adaptability.

## Certificate Generation

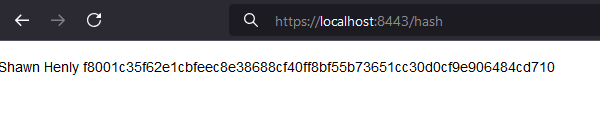
Insert a screenshot below of the CER file.





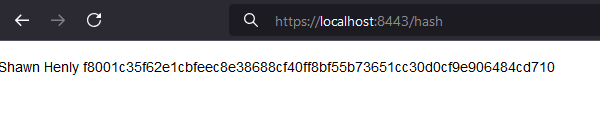
## Deploy Cipher

Insert a screenshot below of the checksum verification.



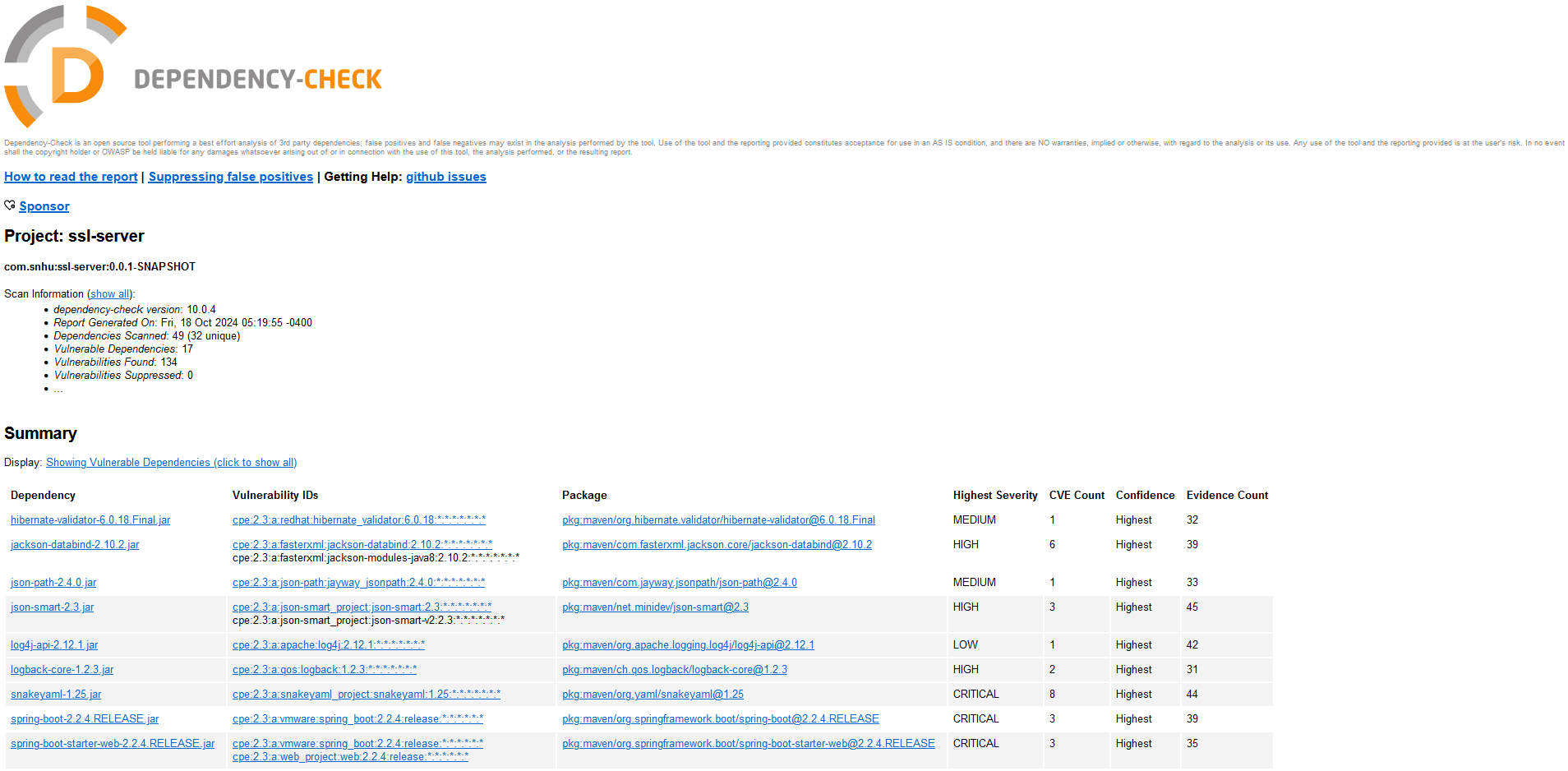
## Secure Communications

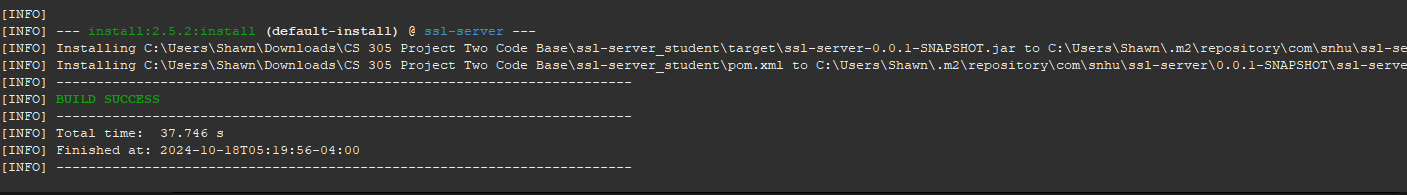
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

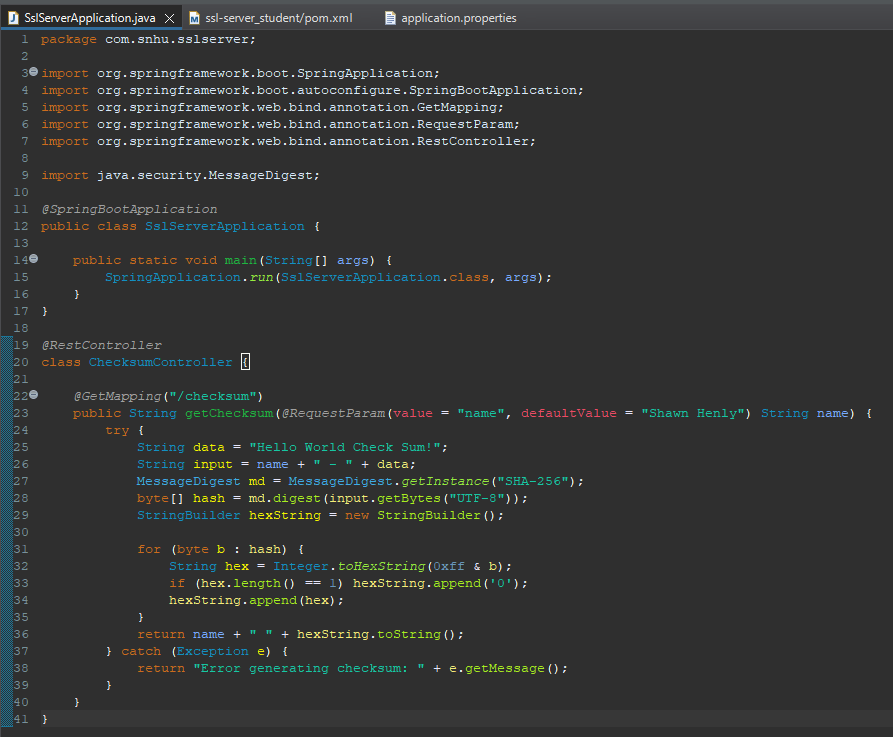
Insert screenshots below of the refactored code executed without errors and the dependency-check report.

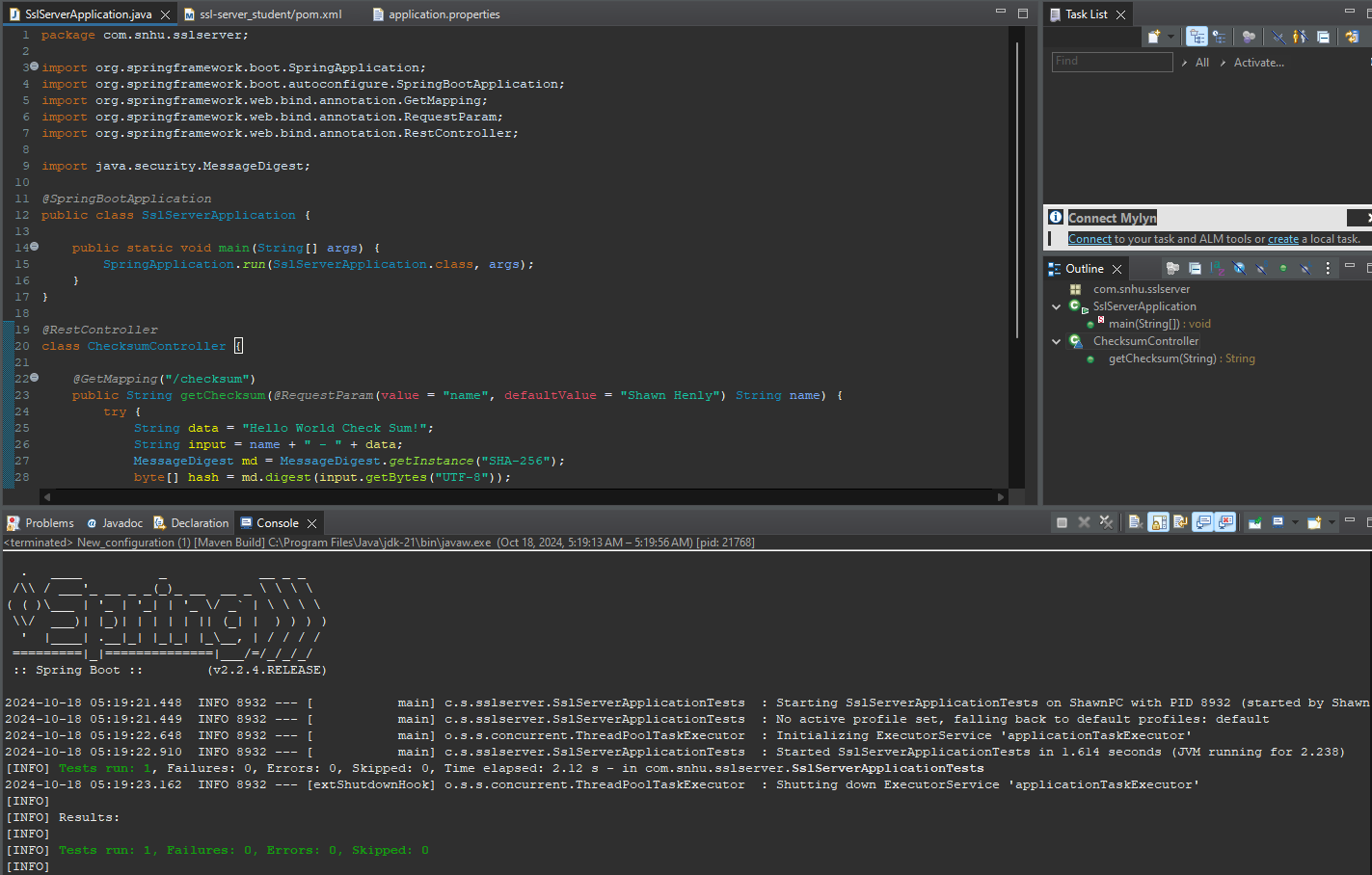




## Functional Testing

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





## Summary

During the revamp of the SSL Server application, my emphasis was on boosting security through improvements in data integrity and secure transmission. I incorporated a cryptographic hash algorithm, specifically SHA-256, to enable checksum verification, which ensures the data remains unaltered during transfer. Furthermore, I transitioned the application to HTTPS to secure communications and prevent data breaches such as interceptions and man-in-the-middle attacks. Static analysis using the OWASP Dependency-Check plugin was conducted to detect and address potential security issues in third-party dependencies. These enhancements collectively strengthen the application's security framework, minimizing susceptibility to vulnerabilities.

## Industry Standard Best Practices

In the refactoring of the SSL Server application, I incorporated a number of best practices from the industry to bolster security and address known vulnerabilities. A crucial step was the adoption of HTTPS to secure communication lines, which encrypts data and reduces the risk of man-in-the-middle attacks. I also implemented SHA-256 hashing to verify data integrity, enhancing the application’s defense against alterations. To further secure the application, I conducted static analysis using the OWASP Dependency-Check to pinpoint and mitigate vulnerabilities in third-party dependencies, safeguarding against supply chain threats.