****

# 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** | **2024/08/12** | **Dalton Rose** |  |

## 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

Dalton Rose

## Algorithm Cipher

To protect Artemis Financial’s data, the SHA-256 hashing algorithm should be used. Hashing algorithms are essentially a method by which to transform data in a consistent manner such than an input will always result in the same hashed value. SHA-256 is an irreversible hashing algorithm, meaning that the original value cannot be recovered. Instead, an input is hashed and compared to a known hashed value to verify they match. For example, when used for authentication, passwords would not be stored but instead the hashed value of the password and when logging in, the entered password would be hashed and compared against the stored hashed password value.

SHA-256 is one of the most secure hashing algorithms available today and is widely used in many industries. SHA-256 comes recommended and approved by the National Institute of Standards and Technology (NIST). SHA-256 works by taking an input, applying the hashing algorithm, and returning an output of 64 hexadecimal characters (valid characters being 0-9 and A-F).

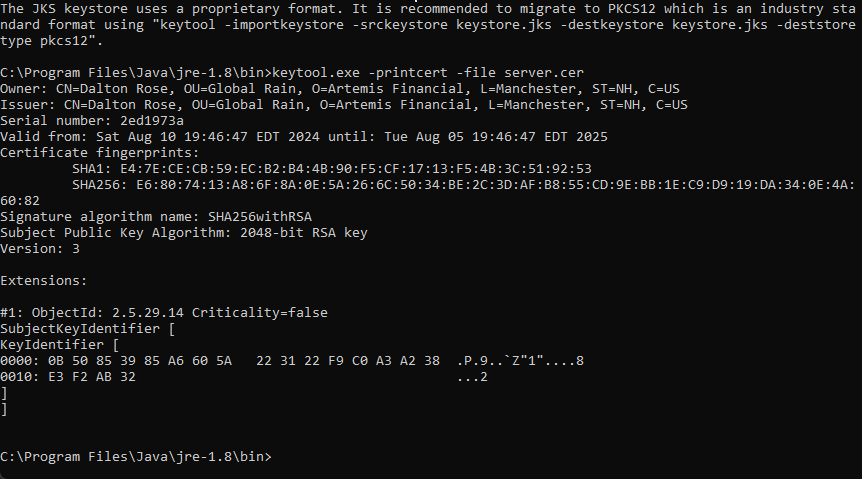
Encryption should also be used to secure backup data and communication between the server and client. There are two main types of encryption: symmetric and asymmetric. Symmetric encryption refers to encryption methods where the same key is used both to encrypt and decrypt the data. Asymmetric encryption refers to encryption methods where there are two keys: public and private. The public key is used to encrypt the data, and the private key is used to decrypt the data.

For archival data, the AES (Advanced Encryption Standard) algorithm should be used. AES supports several different configurations of key-length (128-bit, 192-bit, and 256-bit) with larger key sizes offering increased security at the cost of performance. As performance is less significant for archival purposes, using the 256-bit variation of AES would be preferred. AES-256 is one of the most secure encryption algorithms used today and has been extensively used by banks, government agencies, and militaries to secure critical data.

To ensure communication between the server and the client, TLS/SSL should be utilized. TLS can be configured to use several encryption algorithms, but AES-256 is the preferred encryption algorithm. TLS performs several functions: encrypting the data being transmitted, authentication to prevent data from being intercepted, and verifying the integrity of data. In addition to using symmetric encryption algorithms like AES, TLS also utilizes asymmetric encryption. TLS therefore utilizes both public and private keys. Public keys are often managed by digital certificate authorities. Private keys are utilized in the form of session keys, which are generated to be used to encrypt/decrypt data until the session expires.

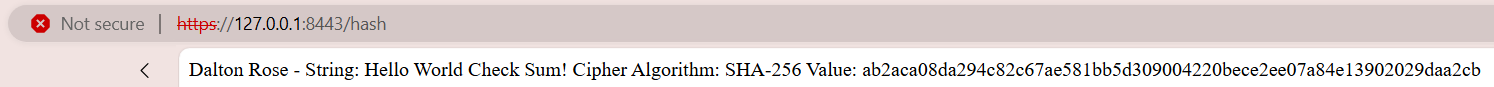
## Certificate Generation

Insert a screenshot below of the CER file.



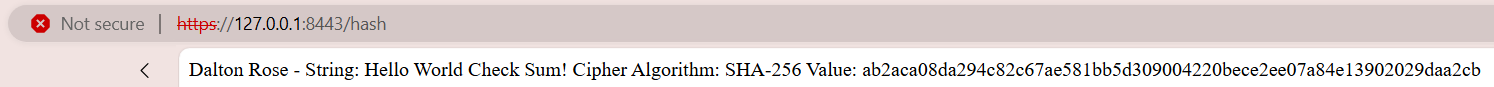
## Deploy Cipher

Insert a screenshot below of the checksum verification.

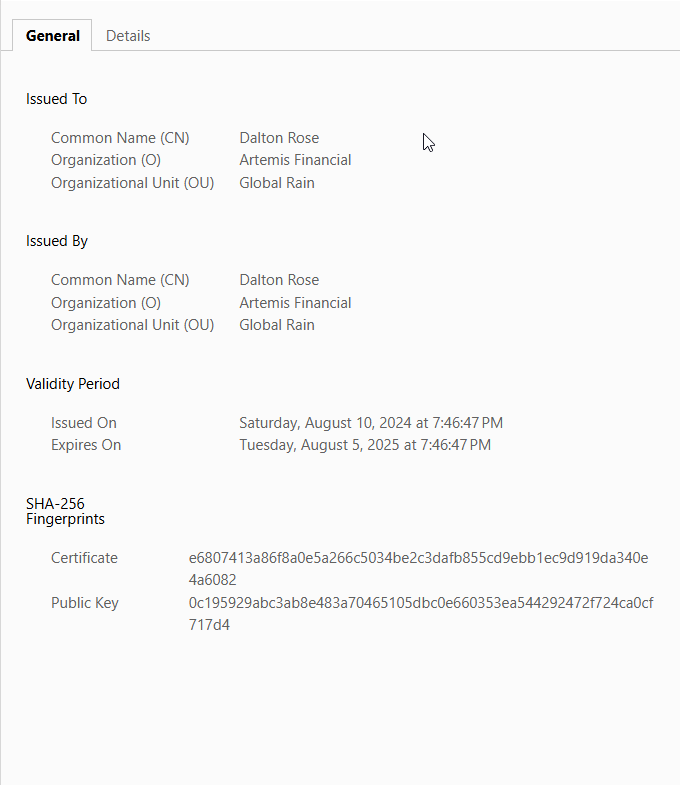


## Secure Communications

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

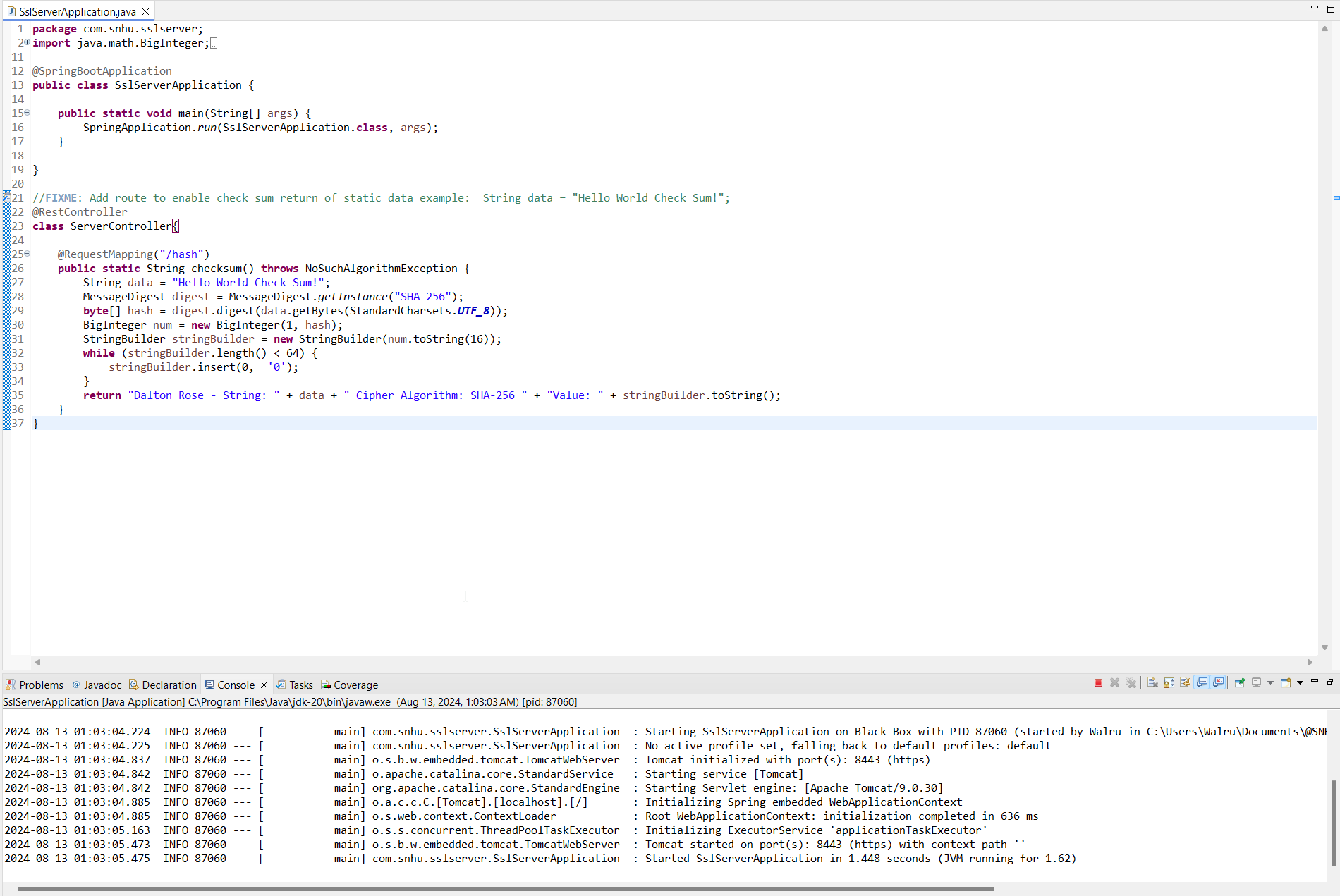


Note: While it shows the not secure icon because it is a self-signed certificate, it is correctly using the certificate and HTTPS.

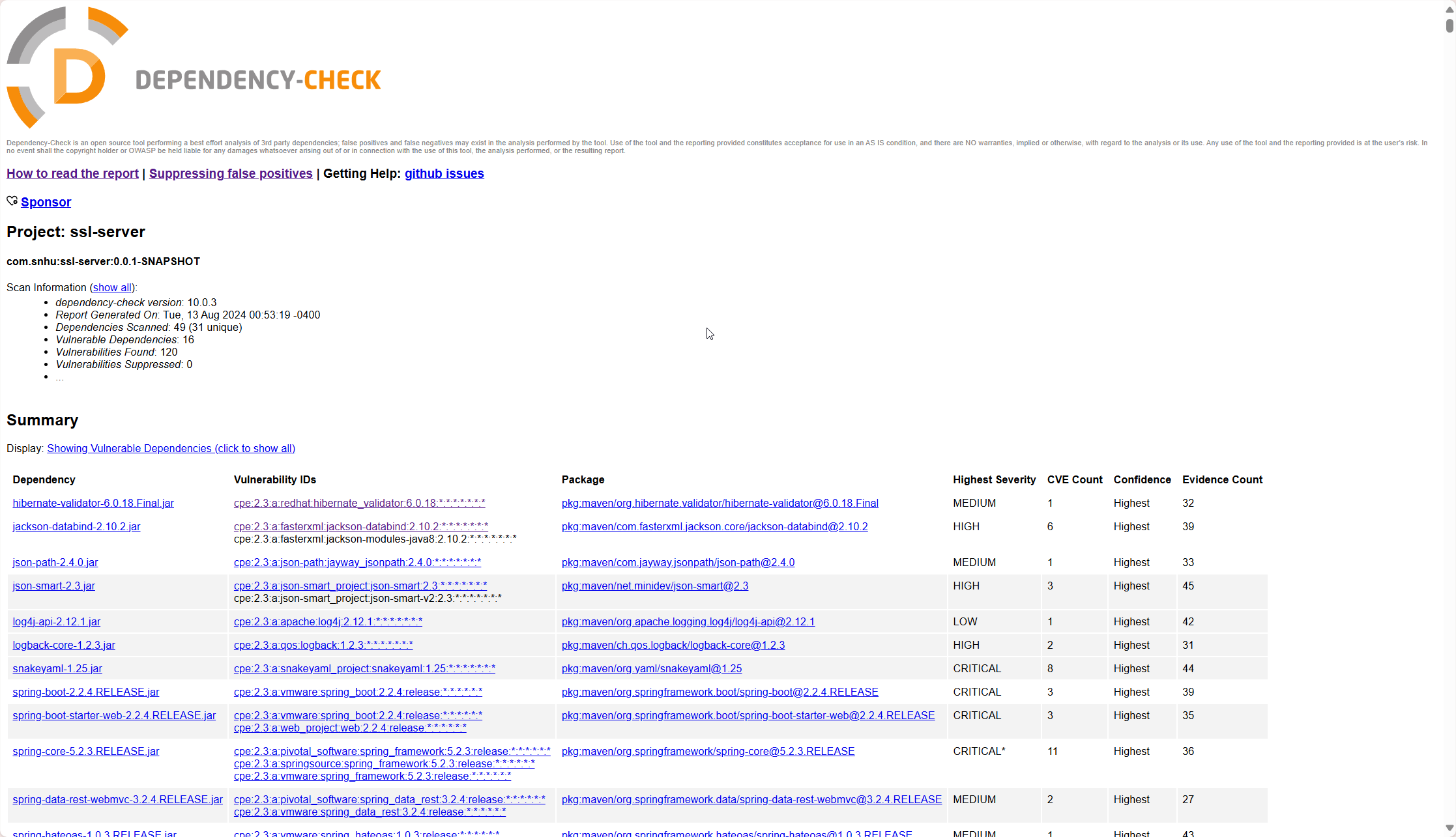


## Secondary Testing

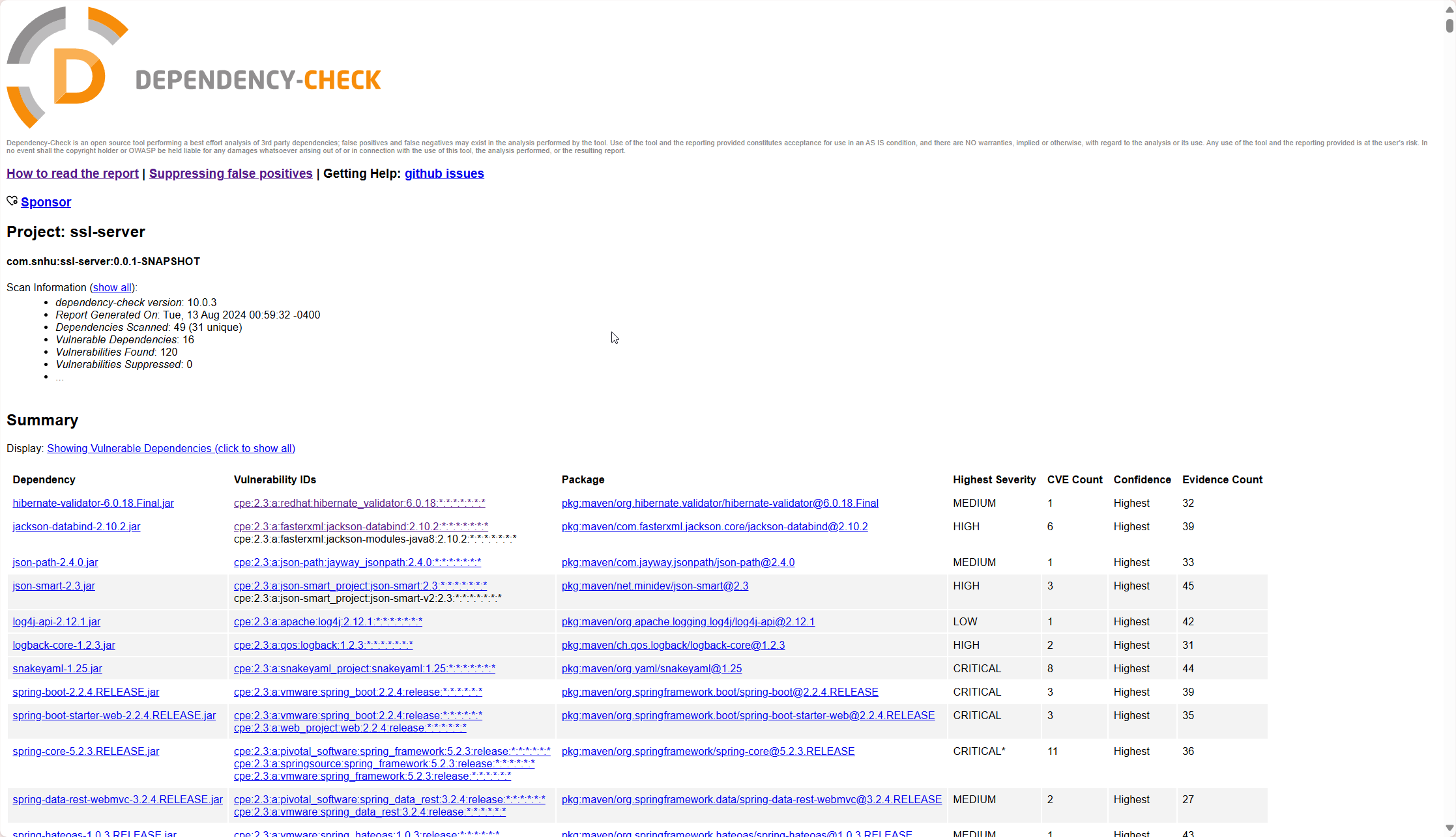
Insert screenshots below of the refactored code executed without errors and the dependency-check report.



Code Running Successfully



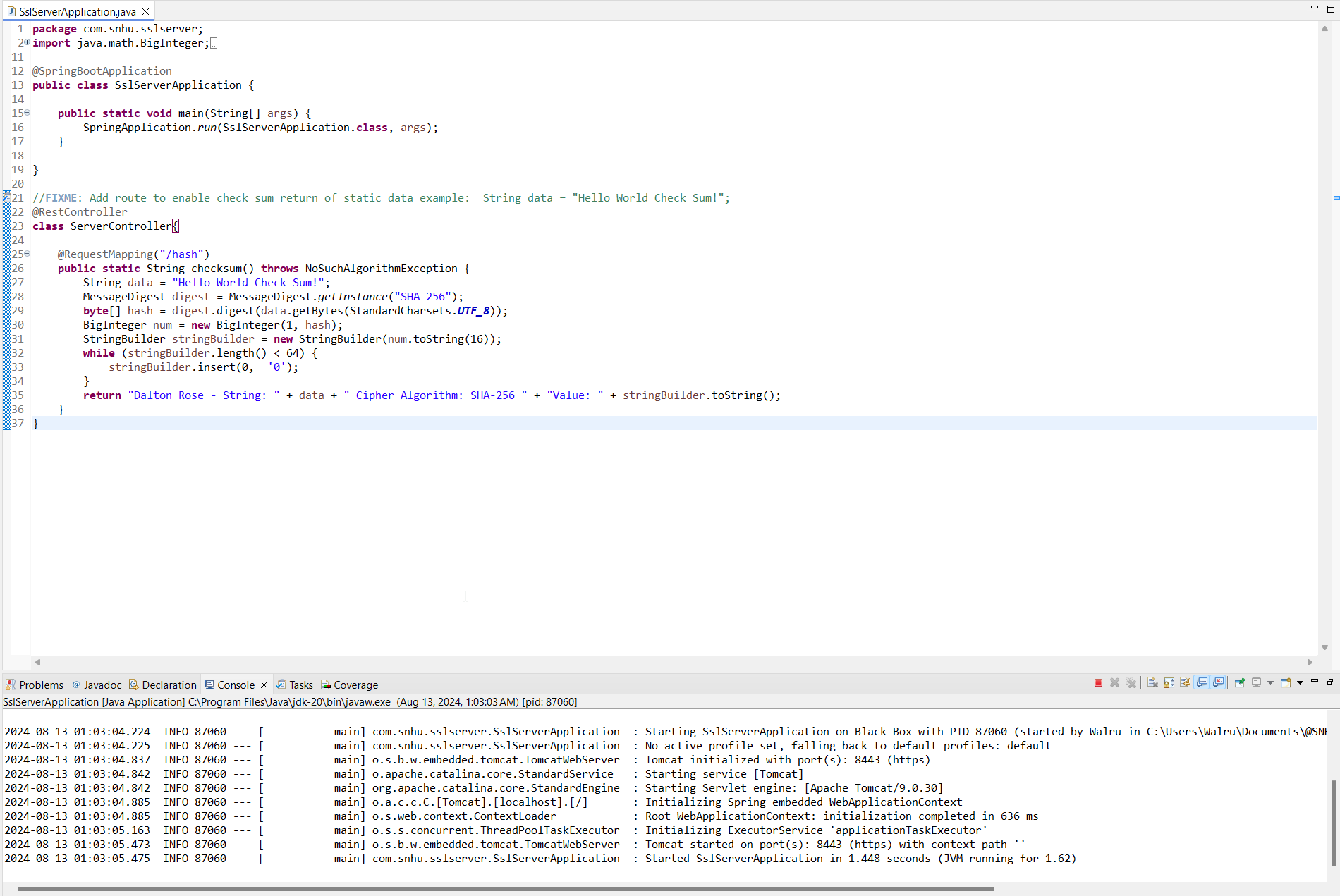
Dependency Check Before



Dependency Check After

## Functional Testing

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



## Summary

This project addresses several areas of security. Client/server communication was made more secure with cryptography in the form of TLS. Cryptography was also utilized for the SHA-256 hashing algorithm. Our software also uses a RESTful API in a secure manner. A dependency check tool was used to ensure that the changes to the software did not introduce new vulnerabilities. Overall, my focus was on implementing the specified security features in ways that limited the risk of introducing additional vulnerabilities.

## Industry Standard Best Practices

My selection of encryption algorithms relied heavily on industry standard best practices. SHA-256 and AES-256 are commonly used in many industries including the financial industry. These algorithms being widely used by banks, militaries, and governments grant us an increased level of confidence as to their security. Considering the industry standard best practices is valuable because we can learn from the past mistakes of companies and examine how they are protecting their companies in the present.