****

# 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](#_Toc102040762) 5-7

[6. Functional Testing](#_Toc102040763) 8

[7. Summary](#_Toc102040764) 8

[8. Industry Standard Best Practices](#_Toc102040765) 9

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **April 21, 2024** | **David Faulkner** | **Final version.** |

## 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

David Faulkner

## Algorithm Cipher

The recommended cipher algorithm for the client’s data verification checksum is SHA-256. Unlike many other cipher algorithms, SHA-256 has never been broken, has no known collisions to date, and is recommended and outlined by NIST in FIPS PUB 180-4. As the name implies, SHA-256 has a total message digest size of 256 bits consisting of 8 32-bit words. SHA-256 hashing is an asymmetric function that cannot be reversed, unlike symmetric encryption algorithms used for file or disk encryption. SHA-256 hashing is used to verify that the contents of a file have not been changed. Note that while SHA-256 has not been broken using current technology, a further review of the algorithm or hash digest size may be necessary if significant strides are made with quantum computing.

## Certificate Generation

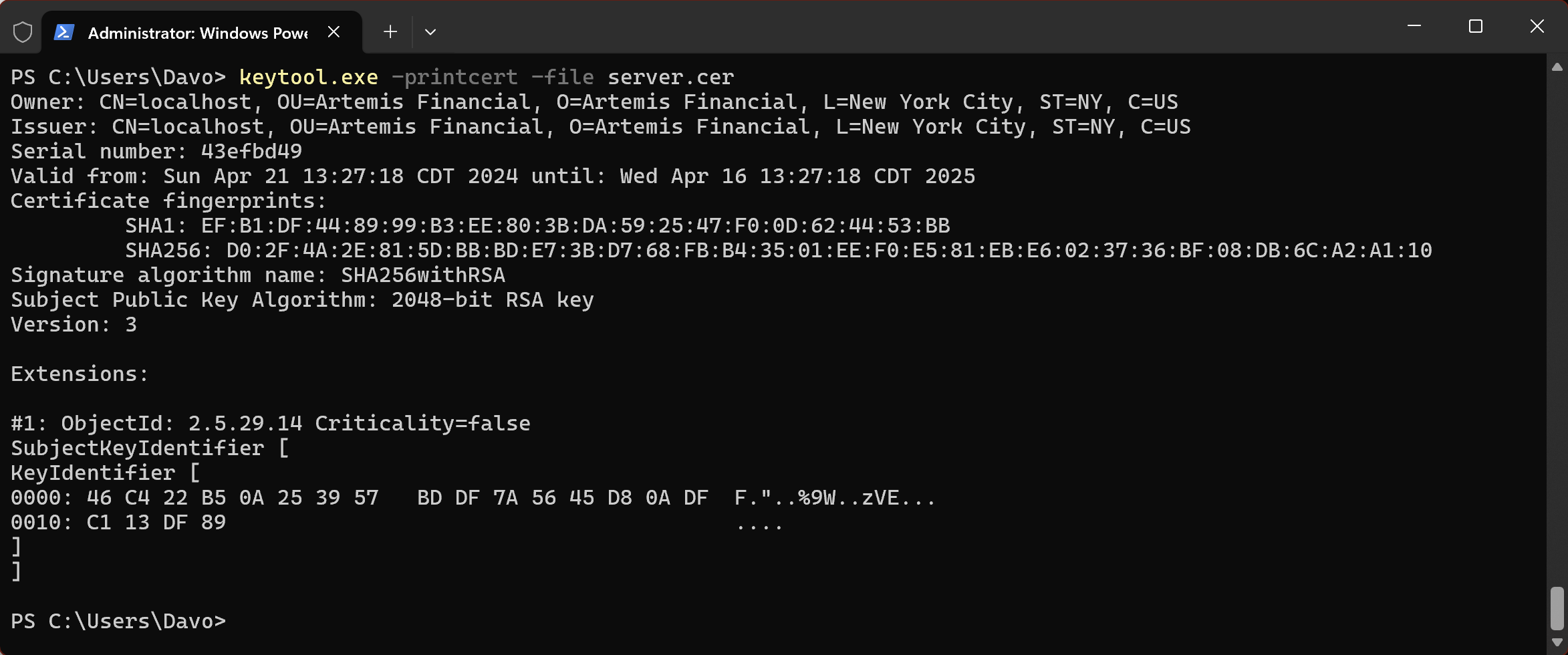


Figure 1 Self-signed certificate

## Deploy Cipher

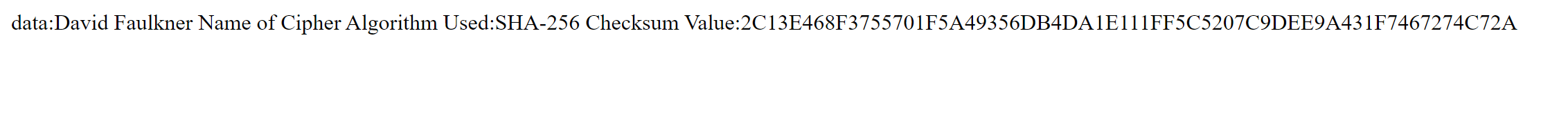


Figure 2 Checksum verification using SHA-256

## Secure Communications

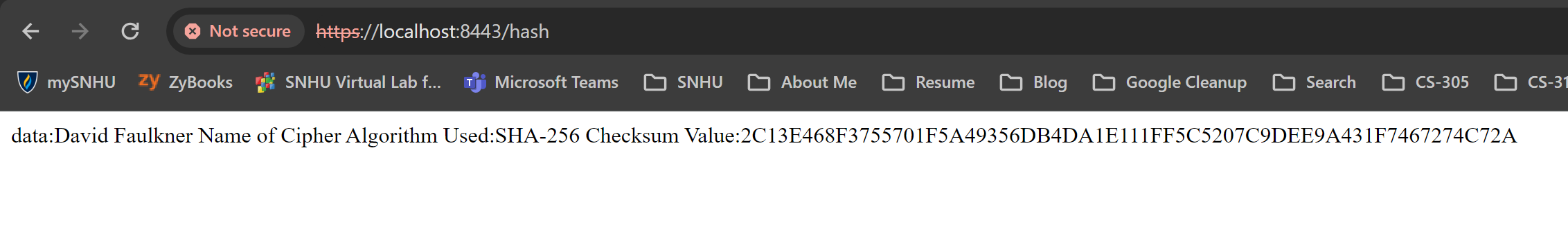


Figure 3 HTTPS working successfully

## Secondary Testing

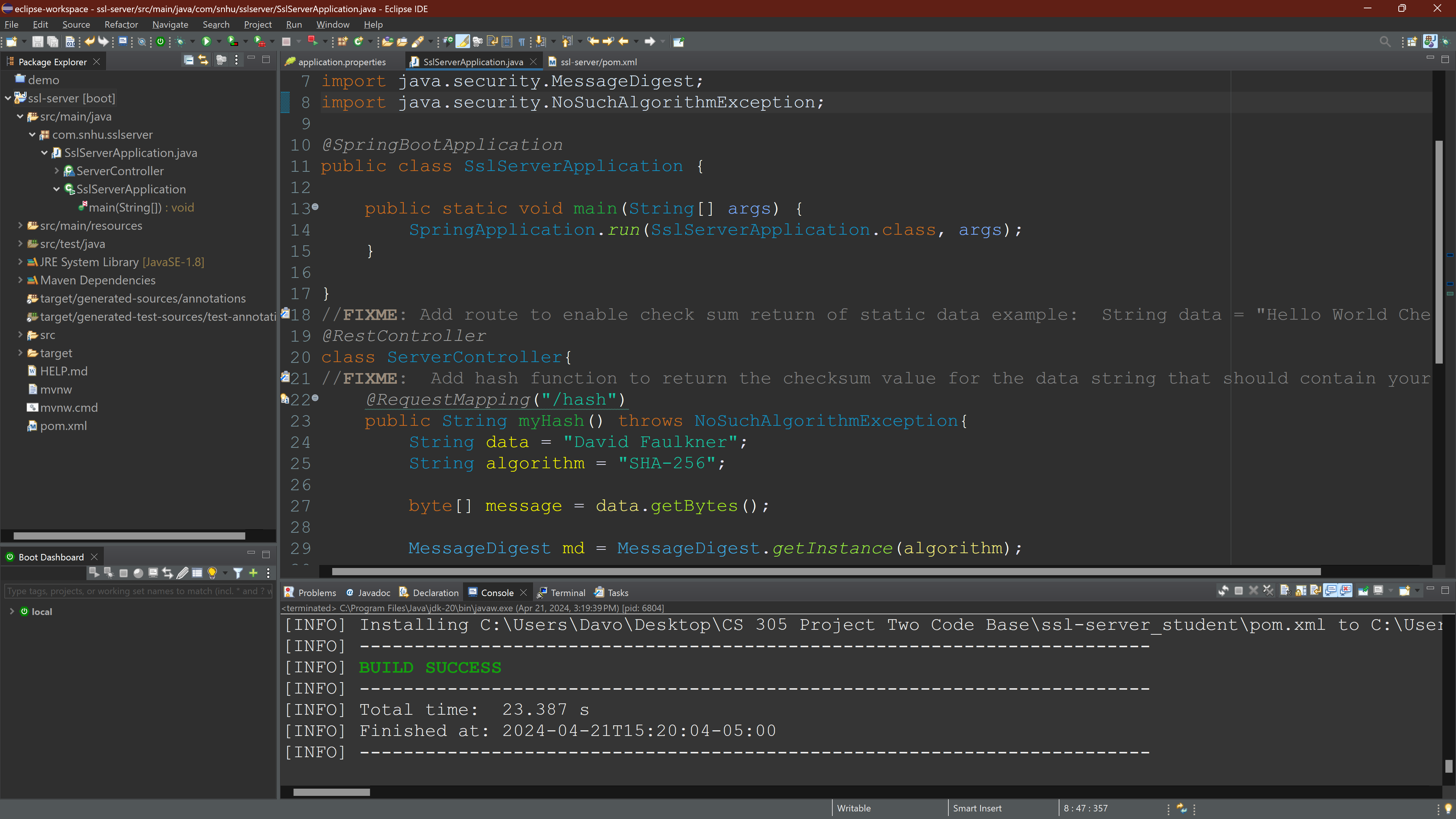


Figure 4 Refactored code executed without errors

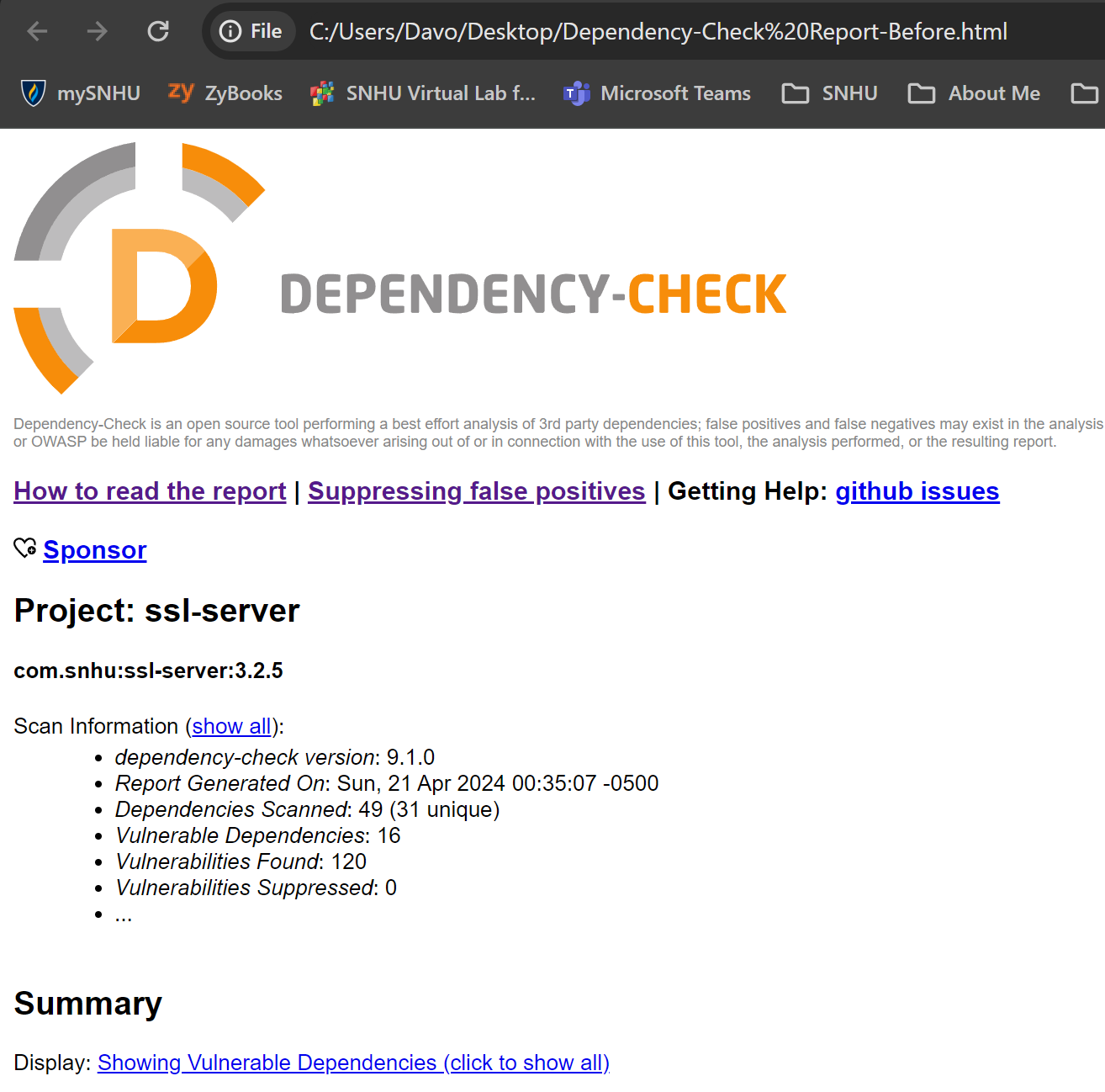


Figure 5 Dependency-check report before refactoring with 120 vulnerabilities found

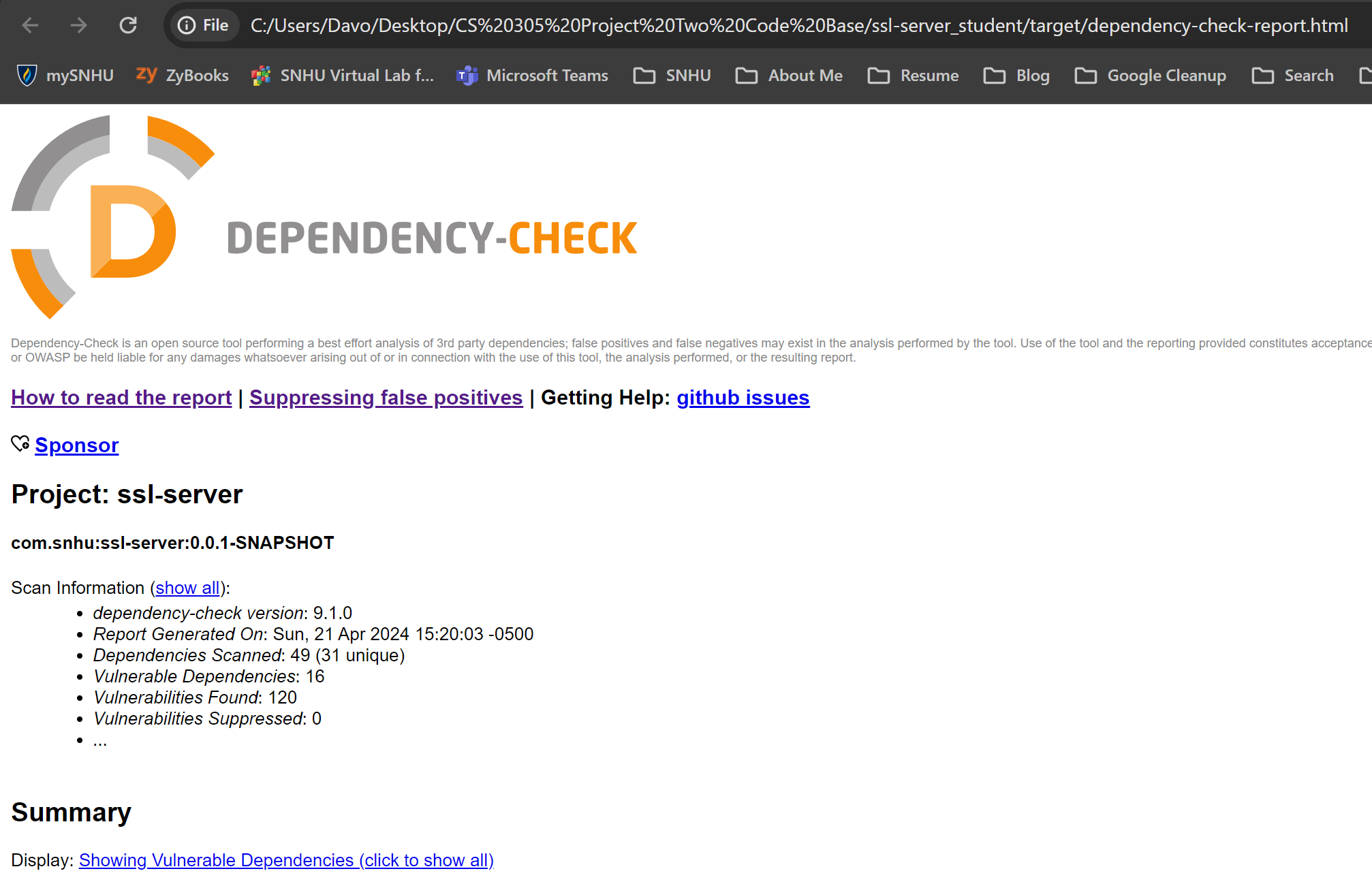


Figure 6 Dependency-check report after refactoring, still with 120 vulnerabilities found (no new vulnerabilities due to refactoring)

## Functional Testing

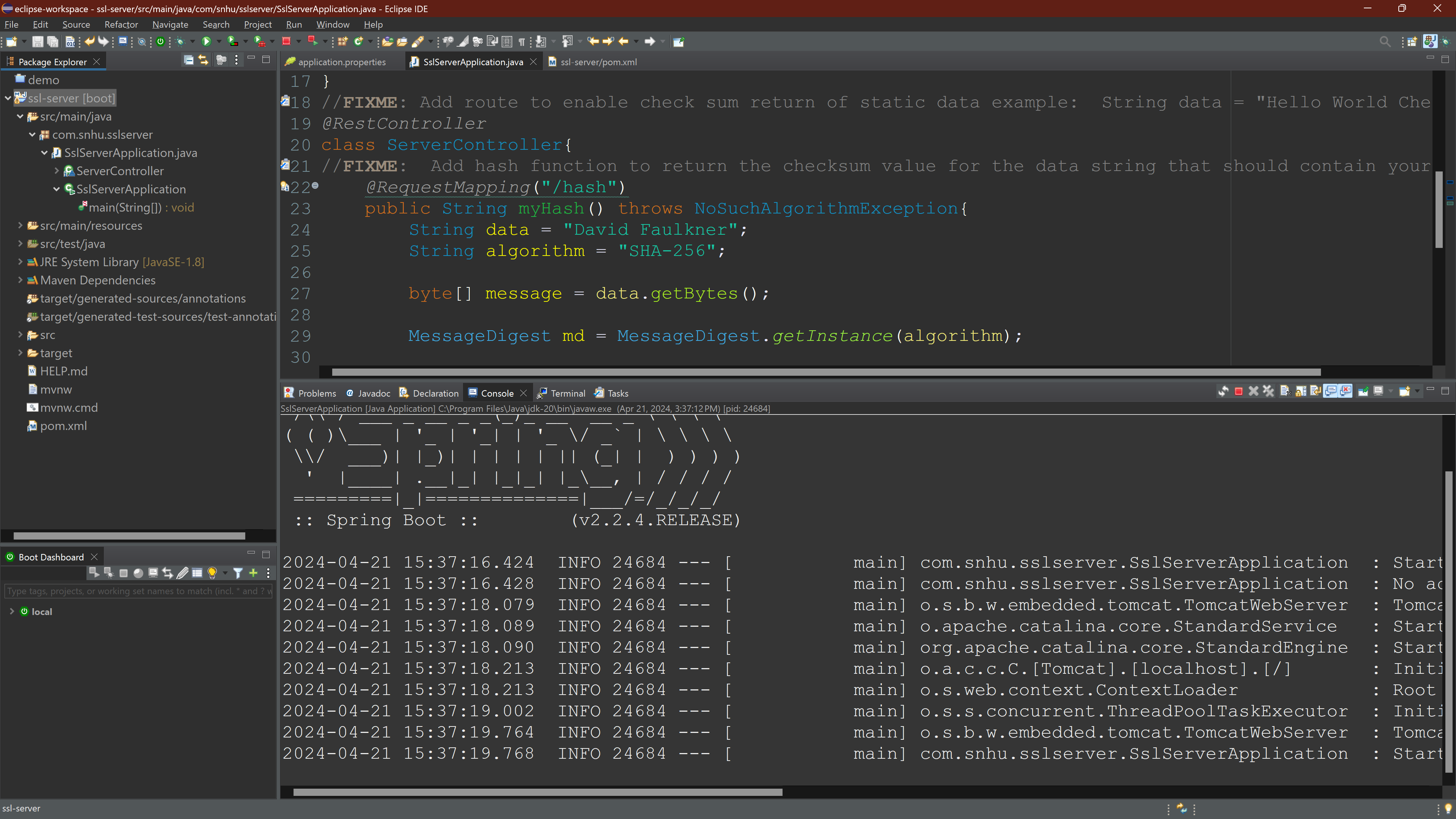


Figure 7 Refactored code (server) running without errors

## Summary

The code has been refactored to provide a SHA-256 checksum for file verification and to provide TLS over a secure HTTPS connection rather than an unsecure HTTP connection. This refactoring was accomplished without adding any additional vulnerabilities through added dependencies and without any errors.

The areas of security that I addressed by refactoring the code are APIs, Cryptography, and Code Quality. This code is a RESTful API, so securing it, in general, is an over-arching concern that affects the entire application. Cryptography was applied to both the data which needed to be validated through a checksum, and through the use of TLS to enable secure viewing of the site using HTTPS rather than HTTP. Code Quality was addressed by using HTTPS rather than HTTP and by providing a checksum for data verification by the end user.

The process for adding checksum verification began with choosing a cipher algorithm. SHA-256 was used since it has no known collisions to date, has never been broken, and is recommended by NIST. The data was then converted from text to a byte array, and then to hex values for the final hash.

The process for adding TLS to the API started with generating a self-signed key. That key was then imported to the project, and the application.properties file was changed to reflect the credentials for the key such as alias name and keystore password. After a successful build, the localhost could be viewed in the browser using HTTPS on port 8443 rather than using HTTP.

## Industry Standard Best Practices

Industry standard best practices were followed by selecting a cipher algorithm for the checksum verification that had no known collisions to date and had never been broken. SHA-256, as recommended by NIST, fit those criteria. Best practices were also observed by securing the RESTful API with TLS allowing the user to view the site over HTTPS rather than HTTP. Finally, best practices were observed through the monitoring and verification that no dependencies with known security vulnerabilities were added to the original code when these changes were made through refactoring.