****

# 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** | **06/15/2023** | **Johnathan Wood** | **Initial creation** |

## 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

Johnathan Wood

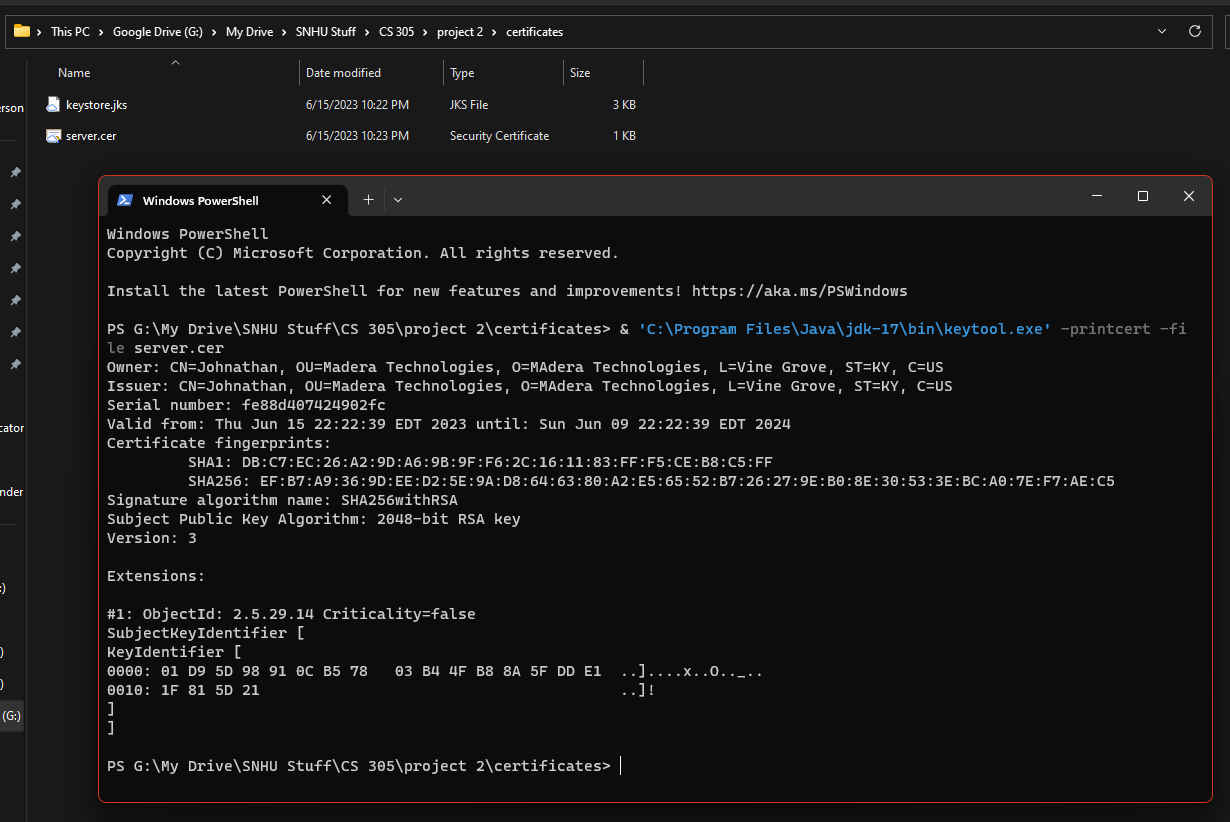
## Algorithm Cipher

Based on the provided scenario, it appears that artemis financial needs to employ two cryptographic algorithms. First, we need a set of certificates so that our web application can secure communication using HTTPS, and second, for verifying the integrity of uploaded files, we need a good hashing algorithm.

RSA is a widely adopted cryptographic algorithm that we learned about earlier this term. It is a good choice for HTTPS certificates for several reasons. RSA provides strong security by utilizing asymmetric encryption, where a public key is used for encryption and a private key is required for decryption. This ensures the confidentiality and integrity of data exchanged between a web server and a client. RSA's widespread implementation and long-standing reputation in the industry make it well-supported and compatible with most web browsers and servers.

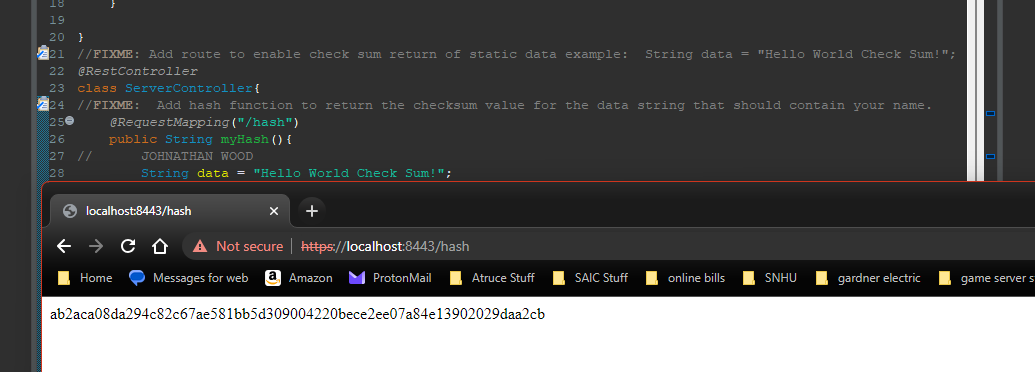
SHA-256 (Secure Hash Algorithm 256-bit) is widely used in various security protocols and applications. It generates a fixed-size, unique hash value of 256 bits, making it computationally infeasible to reverse-engineer or find collisions. This algorithm has stood the test of time and is considered secure against known cryptographic attacks. Its reliability and efficiency make it ideal for tasks such as password storage, digital signatures, and data integrity verification. When it comes to ensuring data security, SHA-256 is a solid choice.

## Certificate Generation

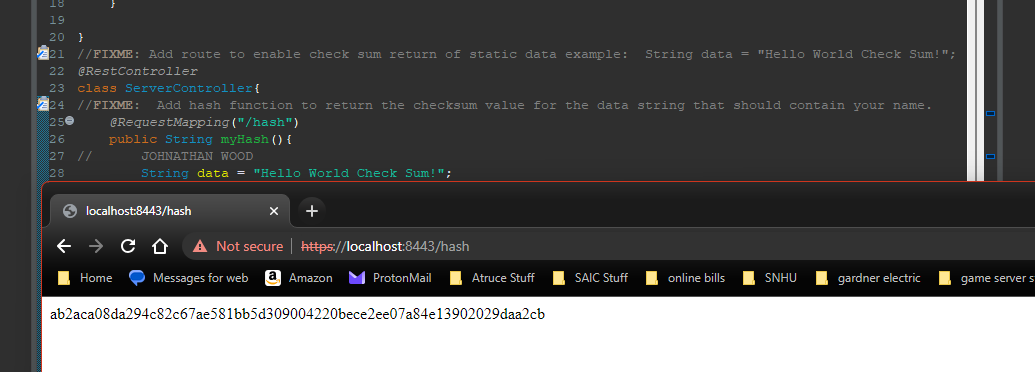


## Deploy Cipher

Insert a screenshot below of the checksum verification.

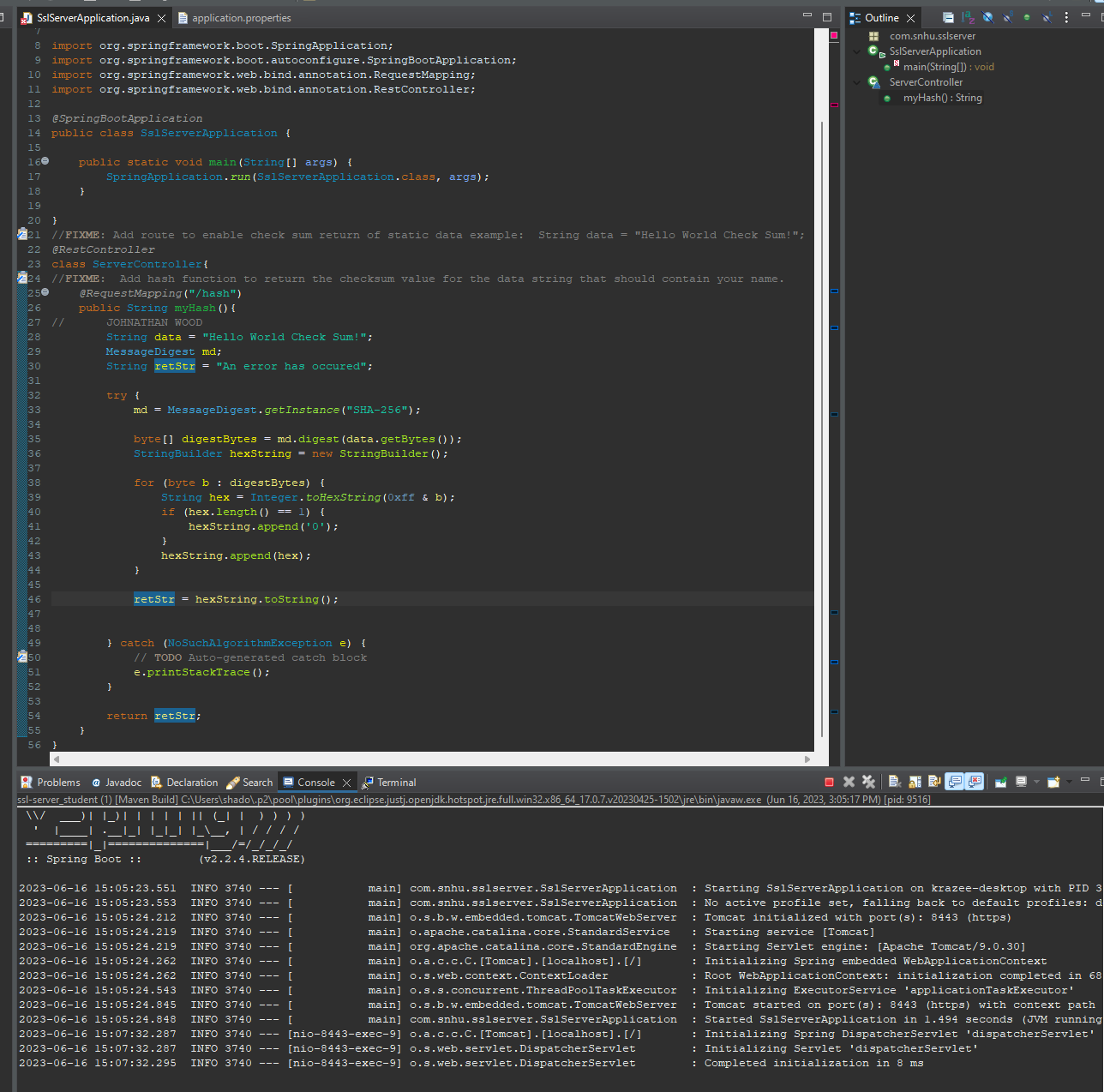


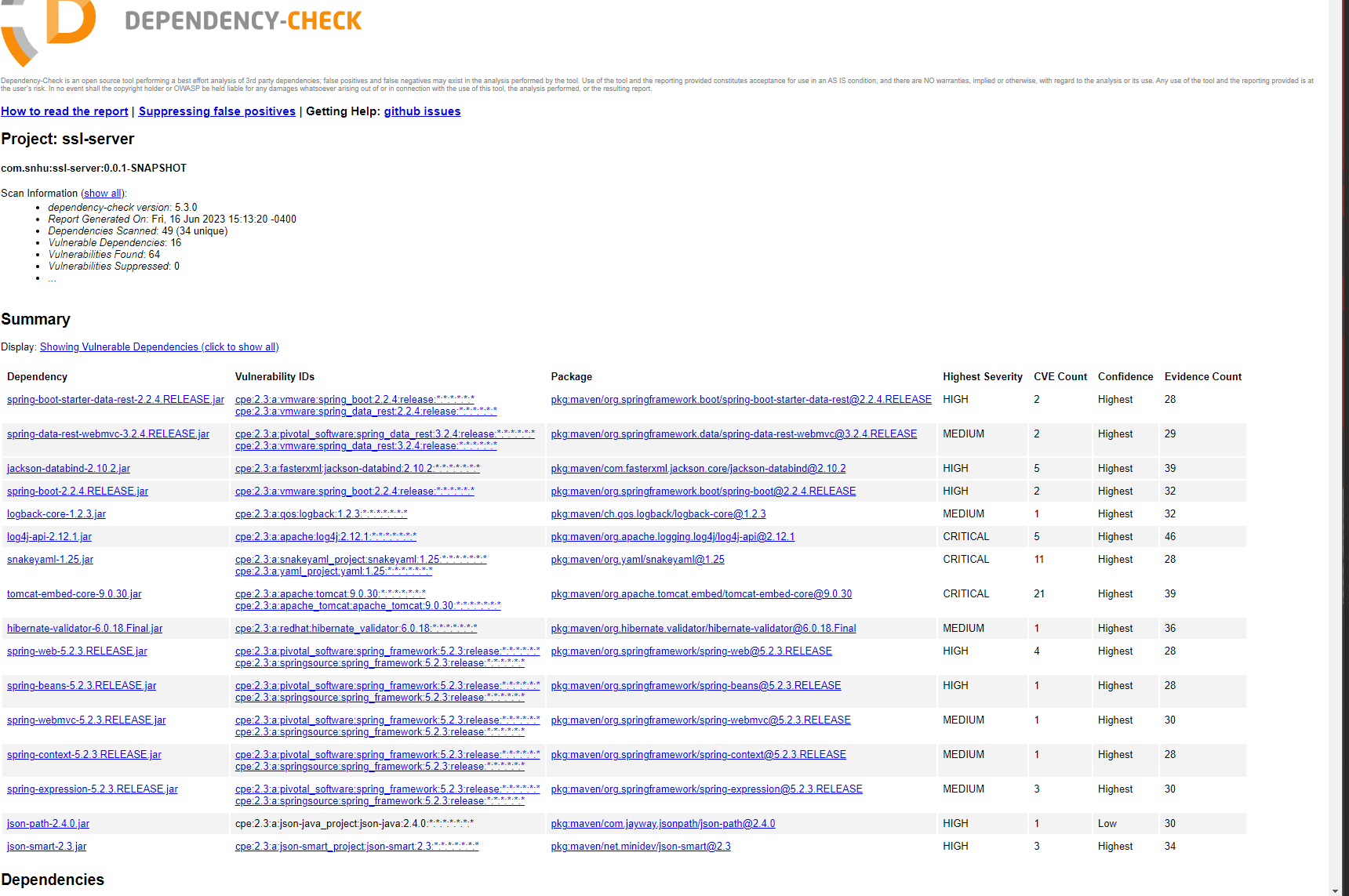
## Secure Communications



## Secondary Testing

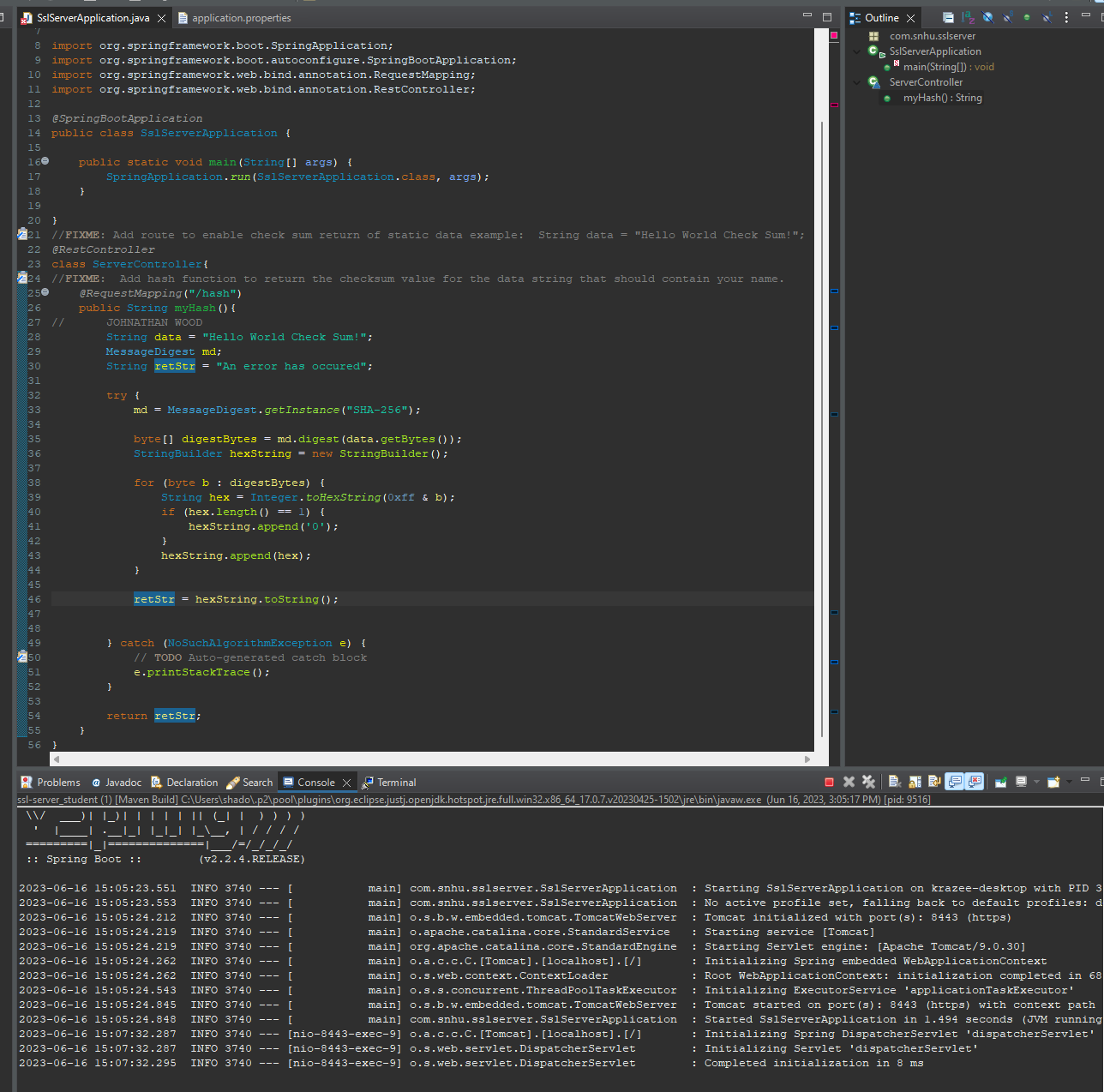
Insert screenshots below of the refactored code executed without errors and the dependency-check report.

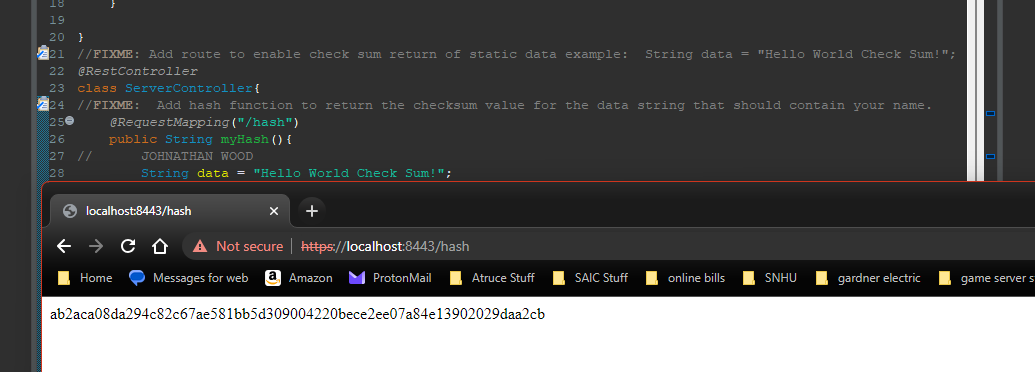




## Functional Testing

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





## Summary

In order to secure our communications, the codebase has been refactored to utilize HTTPS, which means traffic between clients and our application is encrypted using SSL. This was done by generating and adding a keystore to the project and configuring spring to use SSL.  
  
We have also implemented a demonstration showing how the sha-256 hashing algorithm can be used to ensure authenticity and integrity of communications. This was done by adding an endpoint to our secure server /hash, that returns the hash value of a preset message.

## Industry Standard Best Practices

Using industry standard best practices like HTTPS (Hypertext Transfer Protocol Secure) and hashing algorithms ensure data security and integrity. HTTPS encrypts the communication between a user's browser and a website, preventing unauthorized access, data interception, and tampering. It establishes a secure connection, protecting sensitive information such as passwords, credit card details, and personal data from being intercepted by malicious actors. Hashing algorithms, on the other hand, are cryptographic functions that convert data into a fixed-length string of characters called a hash. They are commonly used for password storage, data verification, and digital signatures. Hashing ensures that even if the hashed data is obtained, it is computationally infeasible to reverse-engineer the original input. It adds an additional layer of security, making it challenging for attackers to retrieve sensitive information or modify data without detection. By adopting these best practices, organizations can safeguard their users' data and maintain a higher level of trust and security in their online services.