****

# 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 5](#_Toc102040760)

[4. Secure Communications 5](#_Toc102040761)

[5. Secondary Testing 5](#_Toc102040762)

[6. Functional Testing 7](#_Toc102040763)

[7. Summary 7](#_Toc102040764)

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

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **[Date]** | **[Your Name]** |  |

## 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

Anthony Schissler

## Algorithm Cipher

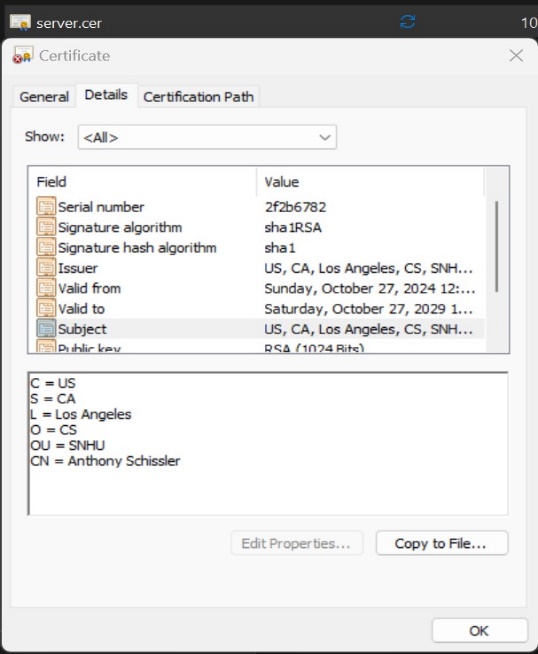
Artemis is looking for the addition of a security algorithm to act as a file verification for the transfer of data. Essentially, they need to ensure that the integrity of data hasn’t changed from its source to its destination through a checksum. This requirement is the perfect use for hashing, and specifically, this is a perfect use for SHA-256. Utilizing 64 rounds of hashing, and no known collisions up to the 64th round, any changes made to a file in transit will be detected, as only one unique output is known for each unique input. Hashing however does not provide encryption – it verifies that data has not changed, but doesn’t provide security to that end. To meet a more secure end, I would further suggest that the data be encrypted first utilizing the symmetric encryption algorithm AES at 256 bit strength, and then utilize SHA-256 on the encrypted data to generate a checksum. This provides two levels of security, and ensures that not only is file tampering detected, but also mitigated through encryption.

Using a symmetric algorithm aids in performance time while helping to ensure the protection of the data. It utilizes the same key for both encryption and decryption, which can be further protected by encrypting the encryption key if required. Random numbers can be used as what’s termed a salt in our hashing to strengthen the effects of our hashing algorithm, but for this use case I wouldn’t recommend doing so unless the company was concerned over the storage of sensitive items like passwords.

Encryption has a storied history from the 20th century, as it essentially formed the basis of what computer science is now through efforts in world war 2 to decrypt German data-in-transit. Overtime, algorithms have strengthened and the methodologies have changed, and we’ve reached a point where encryption algorithms like AES-256 provide a substantial benefit, and can be paired with hashing algorithms such as SHA-256 to ensure file integrity after transmission.

## Certificate Generation

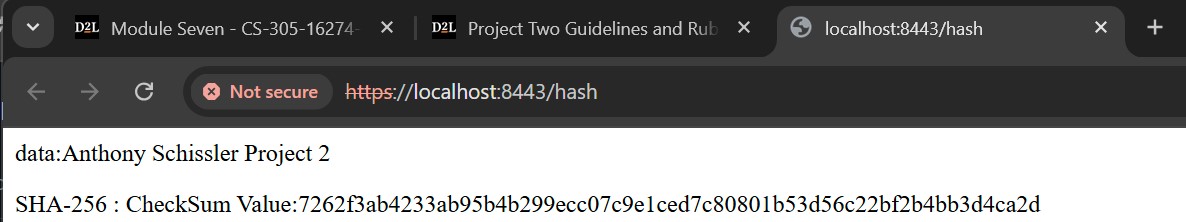
Insert a screenshot below of the CER file.



(Screenshot of the certificate opened up, showing the fields that we input into the eclipse keytool program)

## Deploy Cipher

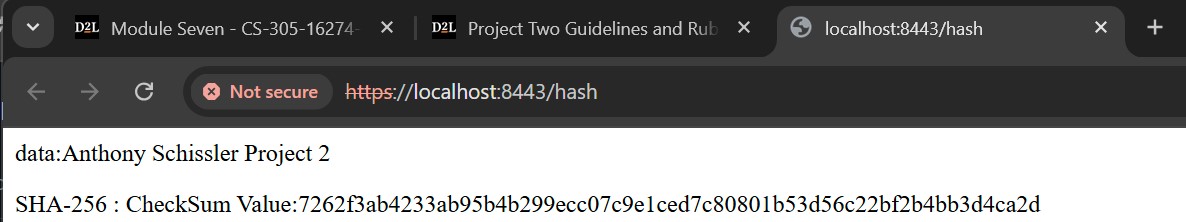
Insert a screenshot below of the checksum verification.



(Snapshot of the cipher using the string in the “data” section to generate the checksum at the bottom)

## Secure Communications

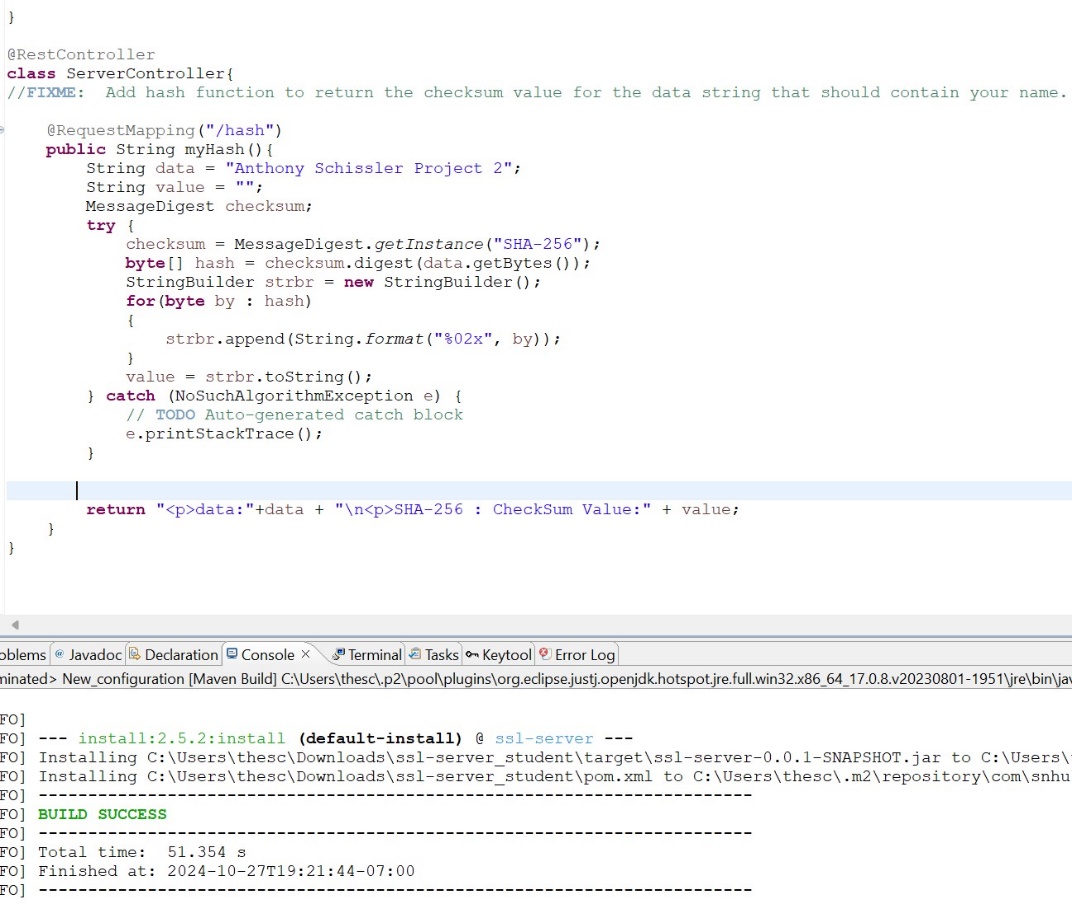
Insert a screenshot below of the web browser that shows a secure webpage.



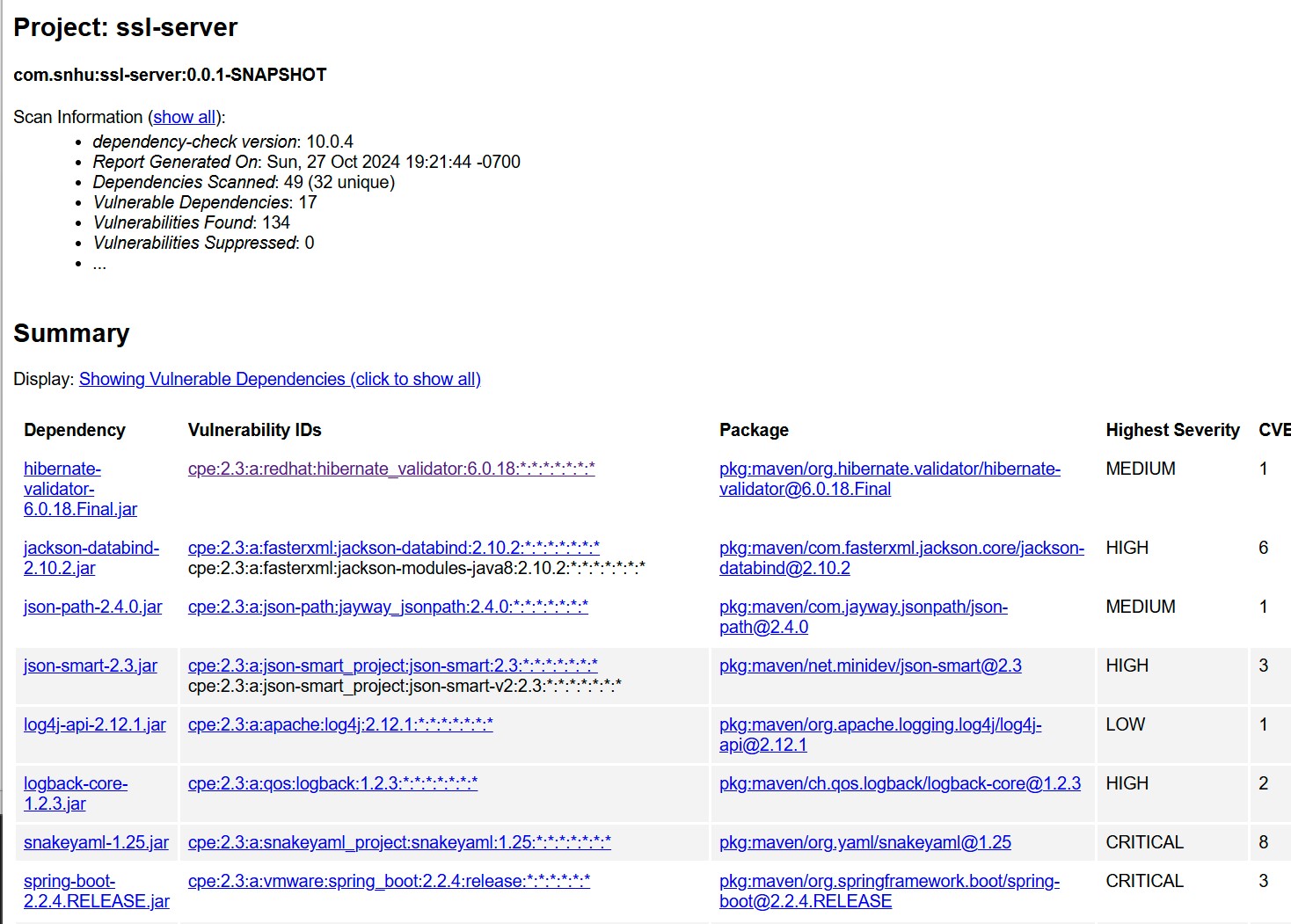
(Same snapshot as before, demonstrating that the webpage is using https. The site lists “not secure” because this browser does not trust the root certificate authority of the certificate that we generated)

## Secondary Testing

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



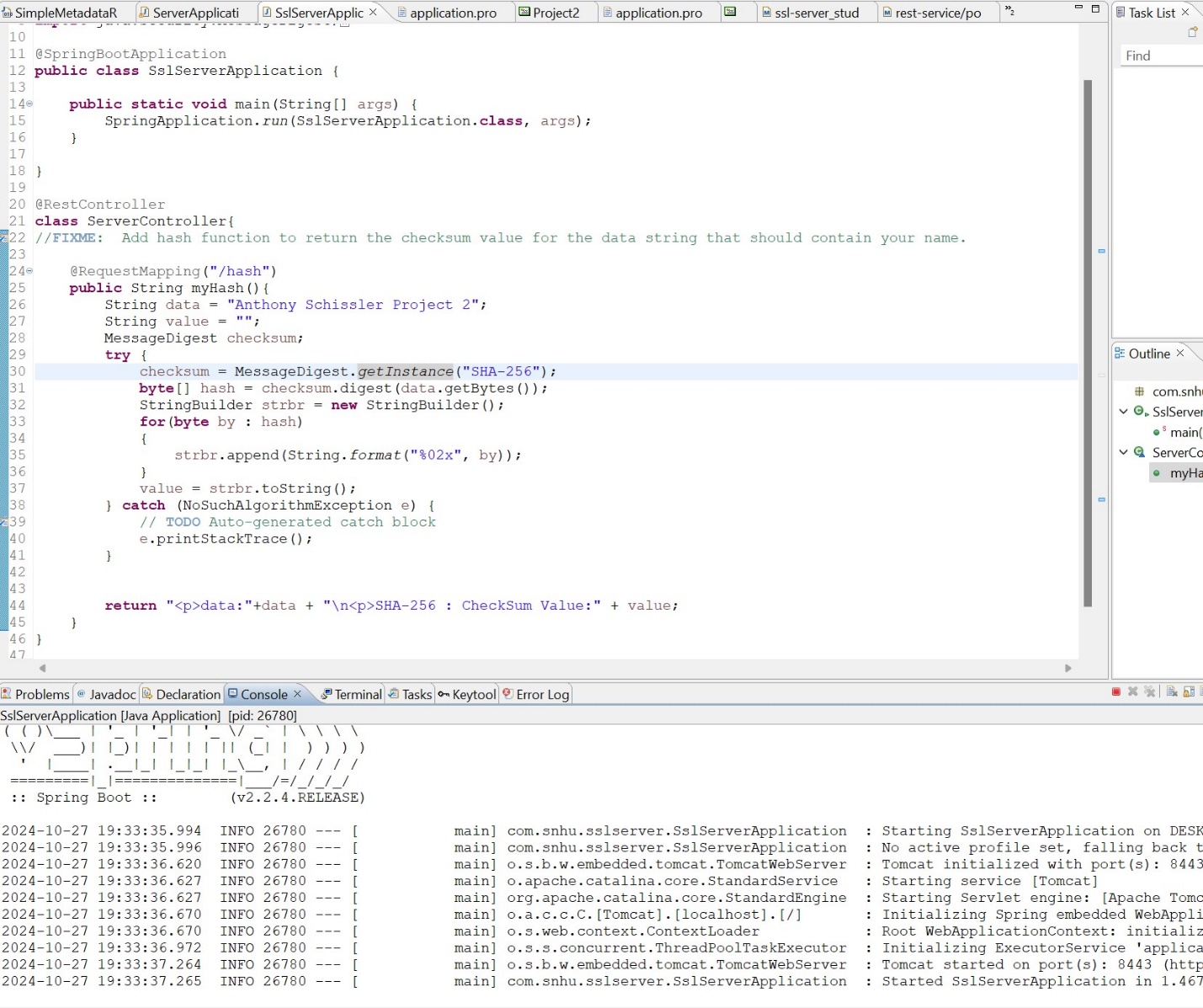
(SslServerApplication running the dependency checker without issue)



(Vulnerability report from project 2)

## Functional Testing

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



(SslServerApplication.java demonstrating running without errors)

## Summary

Using the knowledge from Module 5, I added in a Rest Controller into the SslServerApplication file. In there, I used request mapping to map the /hash directory to the hashing method that I created. By utilizing Rest functionality and request mapping, this allows us to address a number of items in the vulnerability assessment process flow. Specifically, we addressed using secure API interactions, we utilized cryptography to hash data to verify its validity and error handling was utilized in case issues arose with running the functionality on the server. In addition, we configured the web server to utilize HTTPS using a certificate that we generated, ensuring that all traffic to the server has been encrypted with TLS.

## Industry Standard Best Practices

I applied best practices by identifying areas that required enhanced compliance, and mitigating threats to those areas. When maintaining projects, it’s imperative that we not only look for vulnerabilities and document them, but try to write our programs with a mindset towards writing secure code. Designing a web server to utilize things like cryptography and HTTPS are the minimum of what we need to do. Then, by running reports on the vulnerabilities found, we can identify dependencies we rely on that are unsecure, and come together as a team and find ways to mitigate those issues.

Applying these practices brings immense value to companies utilizing technology in the 21st century. Threat actors appear against almost every facet of industry in the technological world, for a number of reasons. To combat this, its imperative that companies enforce and apply standards and best practices for securely coding and maintaining technology. In many countries such as the U.S, it’s not only financial wellbeing that is at stake, but there often is legality in protecting certain characteristics of user data, and companies may face serious legal challenges if they don’t practice industry standards for security. Protecting their resources ensures that users and the companies that serve them can continue to conduct business securely and efficiently.