****

# 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** | **08/07/2023** | **Zachary Nicholas** | **Filling in required information** |

## 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

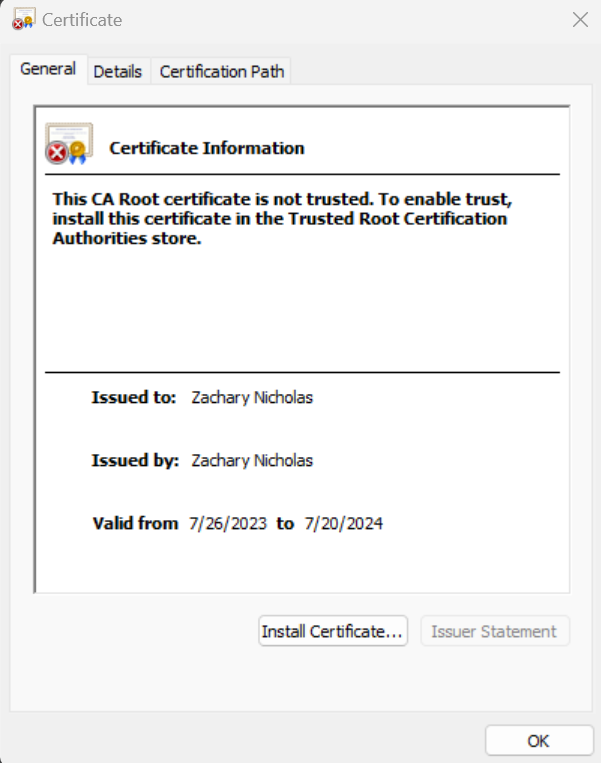
Zachary Nicholas

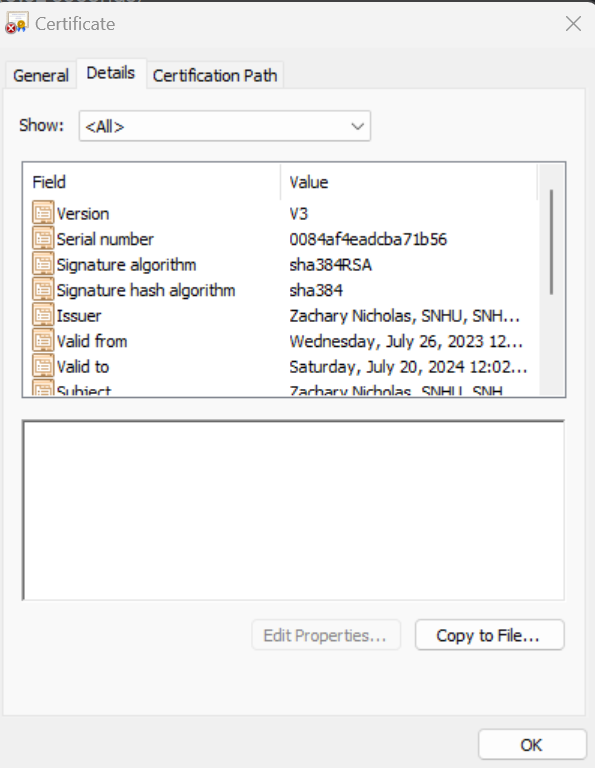
## Algorithm Cipher

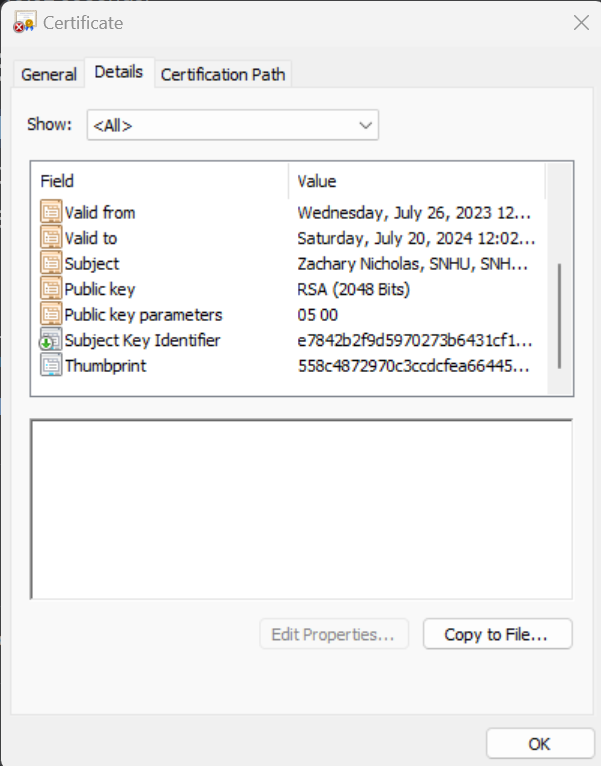
I recommend for our programs to use SHA-256 and if we need more security in the future we may update to a new version like SHA-512, With both of these encryption types we are given a lot of protections that things like SHA-1 and others don’t where SHA-2 has never been decrypted before a lot of the other types of encryptions have. SHA-2 also has a lot brute force protection built in as the number generally are harder to brute force.

## Certificate Generation

Insert a screenshot below of the CER file.

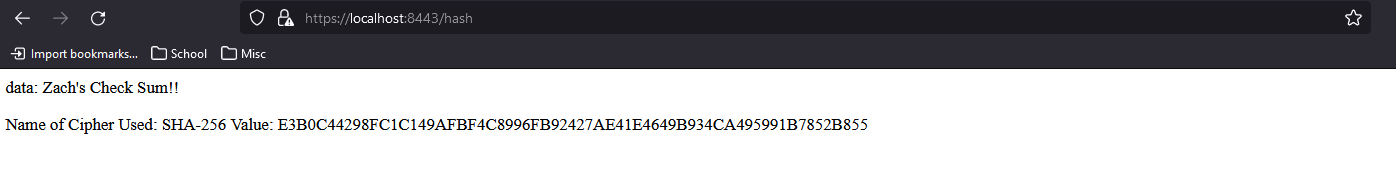






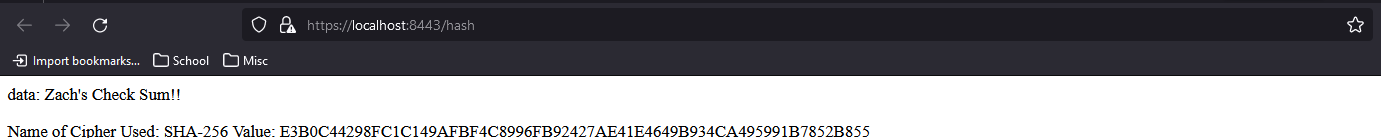
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

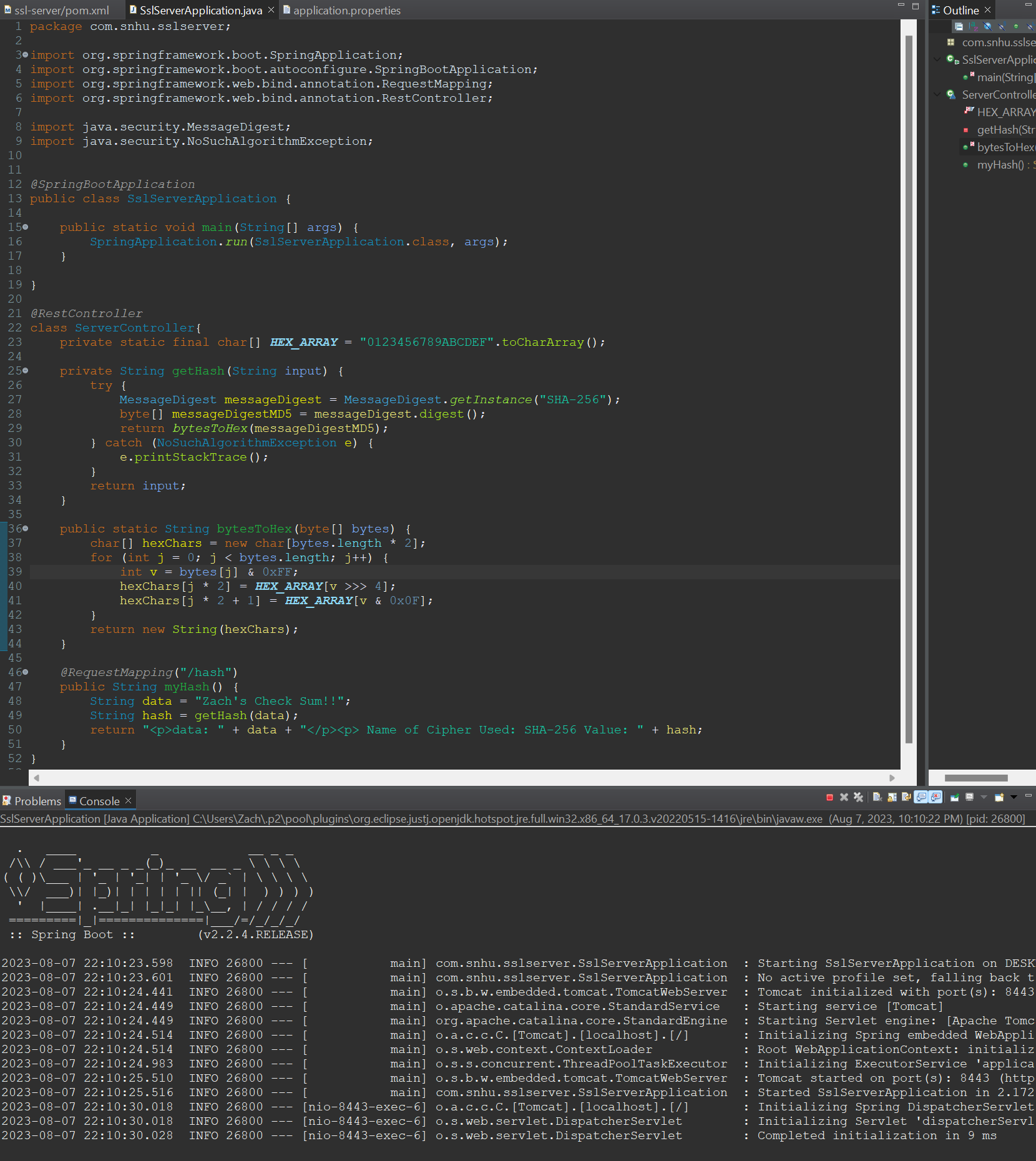
Insert a screenshot below of the web browser that shows a secure webpage.

I was never able to get my HTTPs to connect correctly and the screenshot is exactly like the first one which still required me to accept the risk that the certificate couldn’t be trusted

This was most likely due to my name of my certificate being named server.cer and not a more unique name which could’ve cause a lot of my issues.

## Secondary Testing

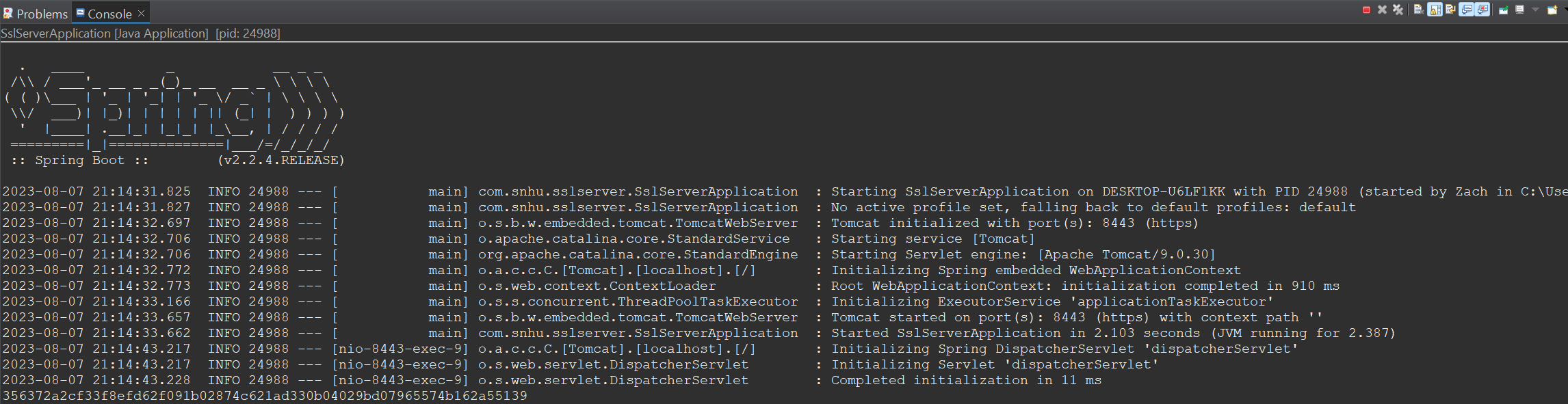
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

The things I did in this to ensure proper functioning were:

* APIs – I used APIs when it came to making the actual HTTPS connection as well as a place for the users to land after having the correct certificate
* Cryptography – For cryptography I used encryption algorithm ciphers, hash functions and verified it using the checksum.
* Client / Server – With the client server communications, even though the machine were the same. I could still see when my webpage would request information and the program would provide the requested information while still keeping it encrypted.
* Code error – For the code errors the first thing that comes to mind is the fact that I used the NoSuchAlgorithm exception inside of my hash method in order to resolve errors that could’ve been thrown.
* Code Quality – With the code quality I reviewed the code I was given and added and modified the things I needed in order to change the functionality, after this I went through and made sure there wasn’t any extra bits of code left over such as instructions.

## Industry Standard Best Practices

For all of the things that I changed I made sure that they were the correct thing to do in that case, such as adding my self signed certificate into my program so that it can use HTTPS. As well as hashing the data it sends in order for it to not be visible to those without access. Which was ensured was done correctly using the checksum.

References used:

*Java https://localhost (SSL) - possible without installing certs on client?*. Stack Overflow. (1964, June 1). https://stackoverflow.com/questions/47783200/java-https-localhost-ssl-possible-without-installing-certs-on-client

Bailly, A. (2005, November 11). *Guide to remote repository access through authenticated https*. Maven. https://maven.apache.org/guides/mini/guide-repository-ssl.html

Kalnichevski, O., & Sutton, A. (n.d.). HttpClient - httpclient SSL Guide. https://hc.apache.org/httpclient-legacy/sslguide.html

Mkyong. (2020, February 19). *Spring Boot SSL (HTTPS) examples*. Mkyong.com. https://mkyong.com/spring-boot/spring-boot-ssl-https-examples/