****

# 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** | **2-26-2023** | **Andrew Miles** |  |

## 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

Andrew Miles

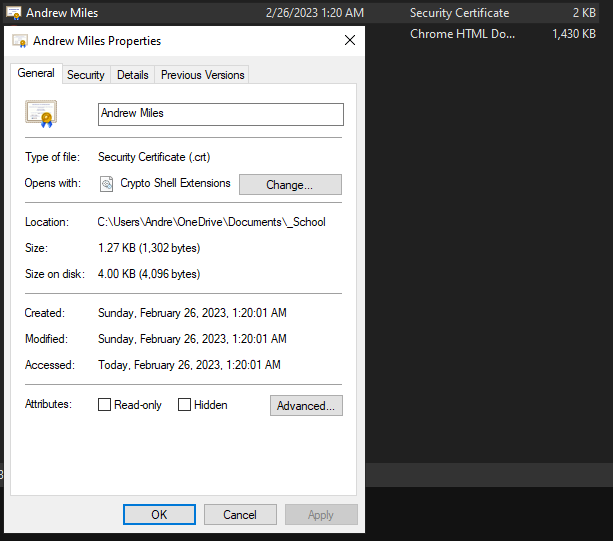
## Algorithm Cipher

I chose the RSA 2048-bit algorithm cipher as the algorithm for this project. RSA uses 3 numbers generated from 2 very large prime numbers that relate to each other through their factors. One of the numbers is kept as the private key, while 2 are sent as the public key. The public key is used to encrypt data sent to the server while the private key that was kept by the server is used to decrypt the data. The bit length is what decides the size of those two prime numbers we use to generate our key numbers, and the larger the bit size the harder it is to force-crack the encryption. Other encryption bit sizes used to be the security norm especially since smaller bit sizes meant faster encryption and decryption. But faster computers and smarter people proved the end of smaller bit sizes, forcing the IT security world to move to higher bit levels as the generally accepted secure bit sizes. It’s important to note that the random numbers that are generated should use a secure random number algorithm as well since a cipher is not secure if all it’s data that’s encrypted is using the same numbers to encrypt it, or easily guessed random numbers to encrypt it for that matter.

RSA is an A-symmetric algorithm cipher, meaning it has separate values for the encryption and decryption. This allows for applications to publish the public keys for locking data to anyone that wants one, then using a private key shared with no one to decrypt the data, therefore keeping the process of encryption safe from potential man in the middle attacks. Symmetric algorithm ciphers use the same value for encryption and decryption which is great for many circumstances but for web servers that need to pass that key out to a client it only creates problems with the privacy of that encryption/decryption key.

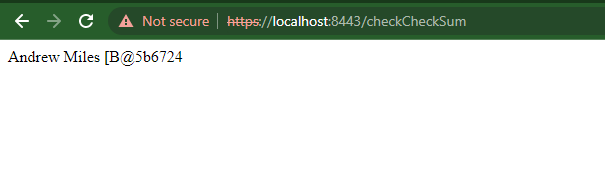
## Certificate Generation

Insert a screenshot below of the CER file.



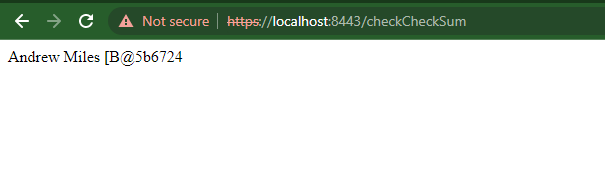
## Deploy Cipher

Insert a screenshot below of the checksum verification.



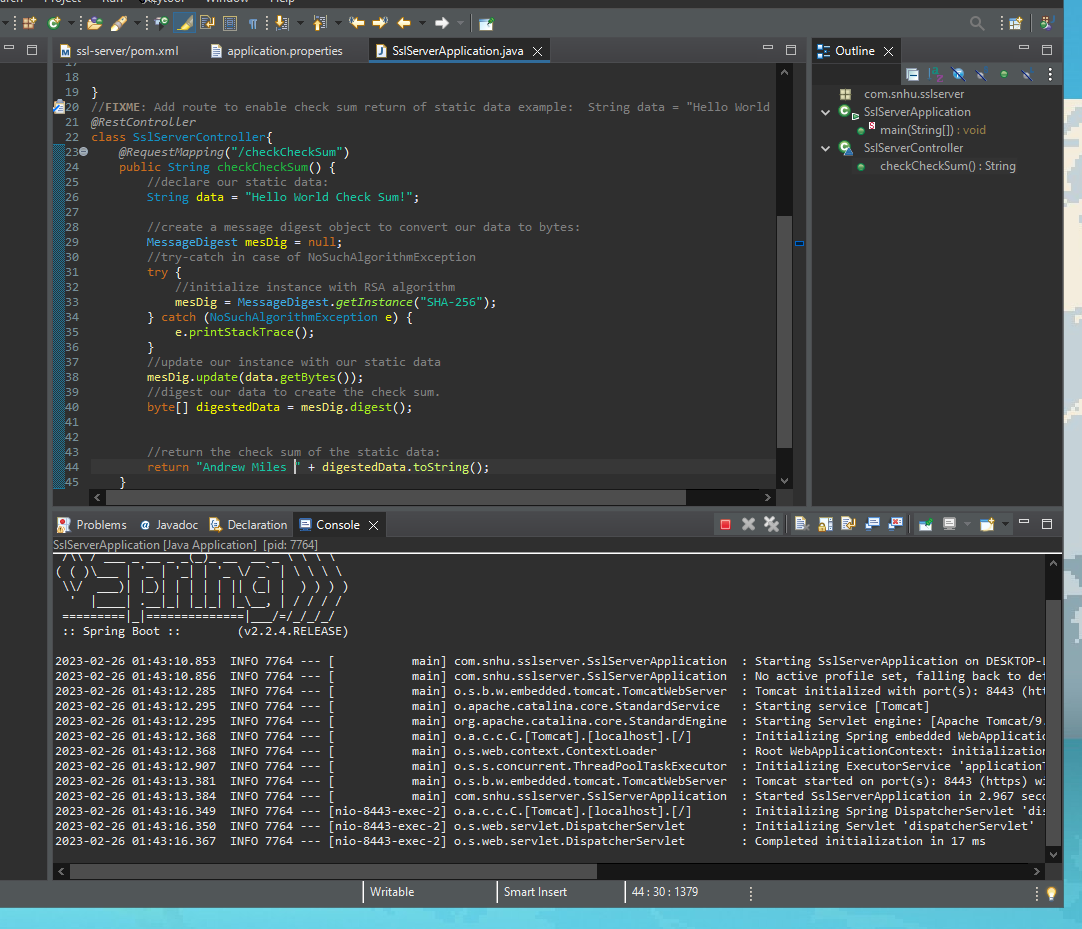
## Secure Communications

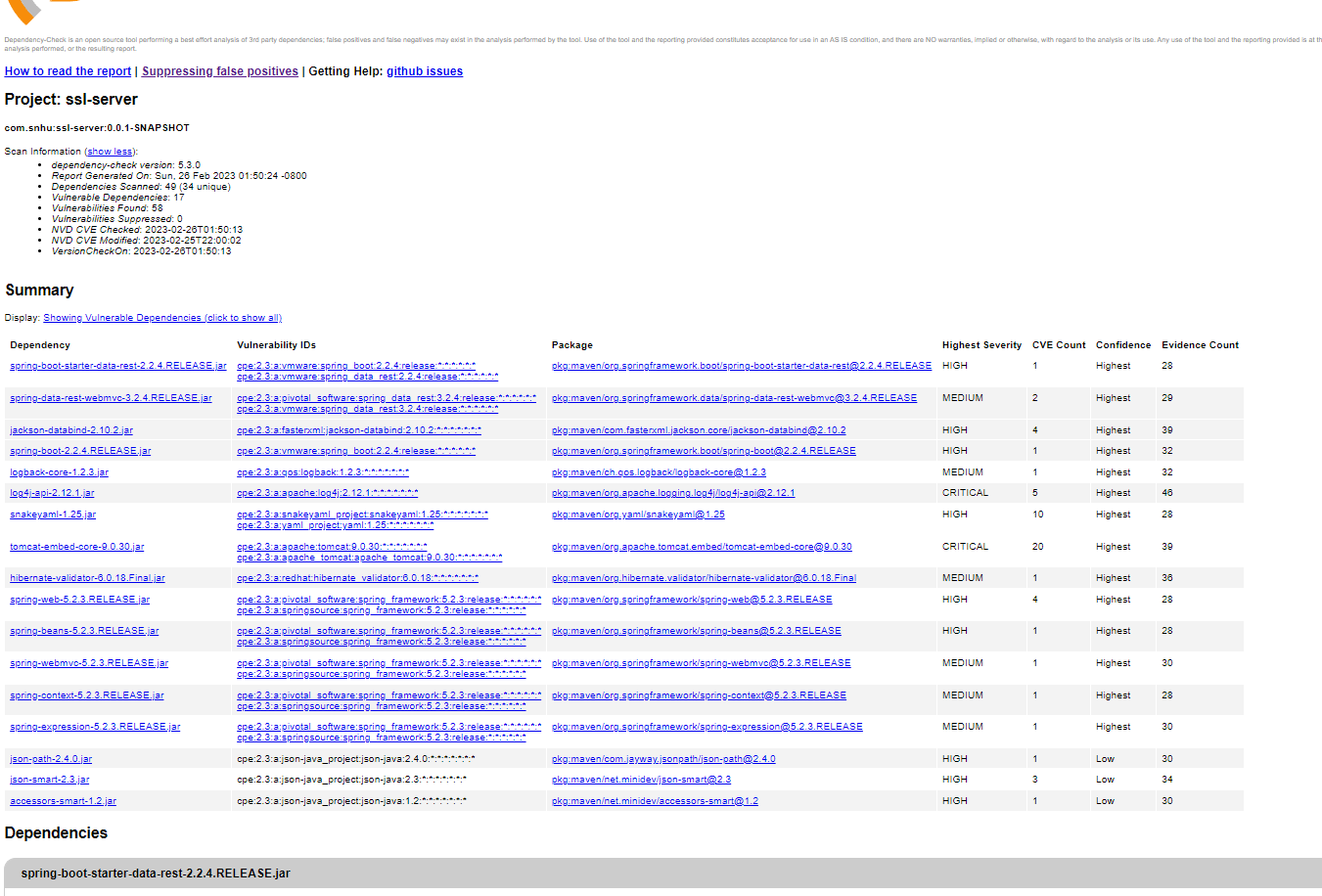
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

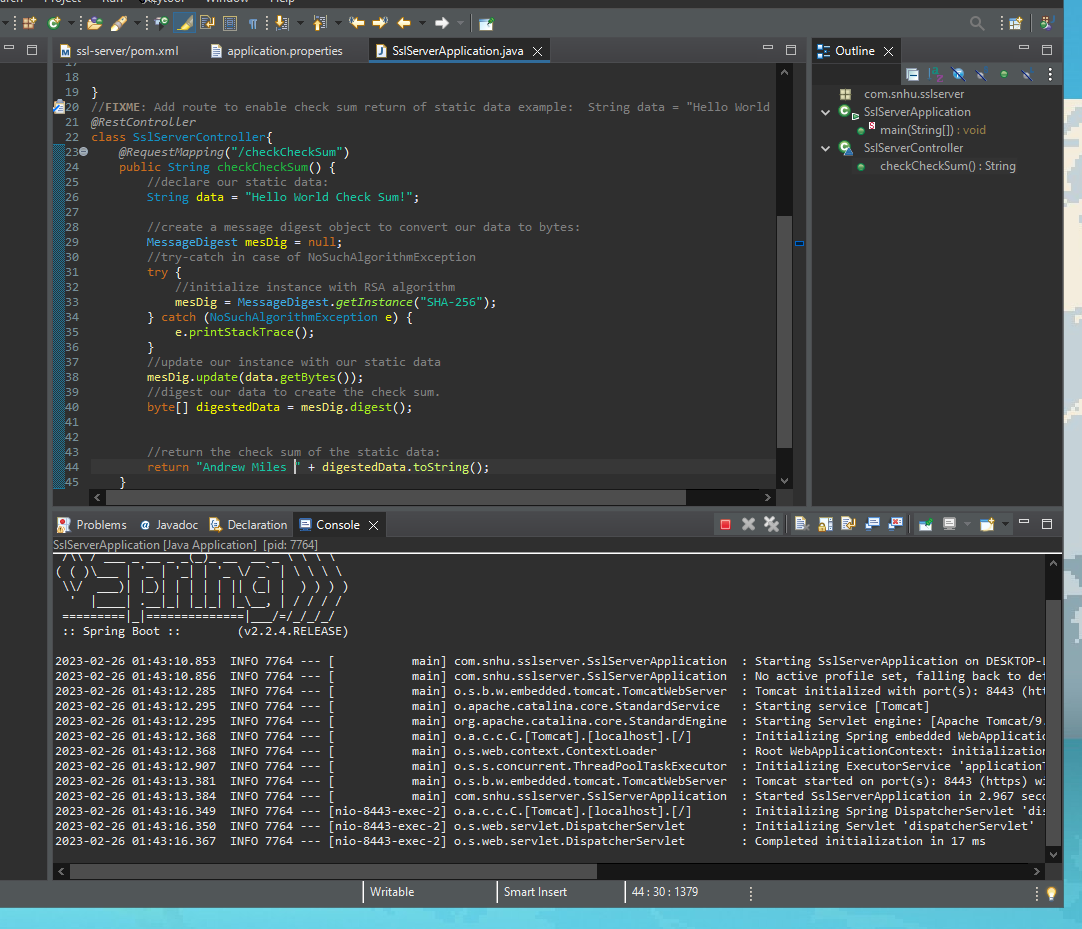
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

In my code refactoring, I added an endpoint in the Spring API that accepted requests at “/checkCheckSum” and returned a random hashed hex string from a static string of data along with my name. I went through the Vulnerability Assessment Process Flow diagram after completing my refactoring and found that code Error, code Quality and API security sections all applied to my code. In my programming I made sure to avoid those by not printing out error information to the website rendered back to the client. I also made sure to use try and catch statements to avoid any potential code failures that would crash the web server. I made sure to check my API end point I added to ensure it didn’t send back data in any kind of unsecure way.

## Industry Standard Best Practices

By using OWASP’s decency check software, I used a Industry Standard software tool to statically test my code. I also made sure to use industry standard best practices in code hygiene by using comments as well as checking approved project security diagrams to verify that applicable areas of security were covered by my code. It’s very important to use industry’s best practices because cyber security is always changing and security vulnerabilities are always a probable risk that could jeopardize the company and its assets, but also jeopardize the lives of employees that work there. Data that is breached could expose employees or customers or even leave company finances open for public access. That is why it’s important to follow up to date industry standard best practices.