****

# 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** | **19 February, 2023** | **Alan Palmer** | **Initial** |

## 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

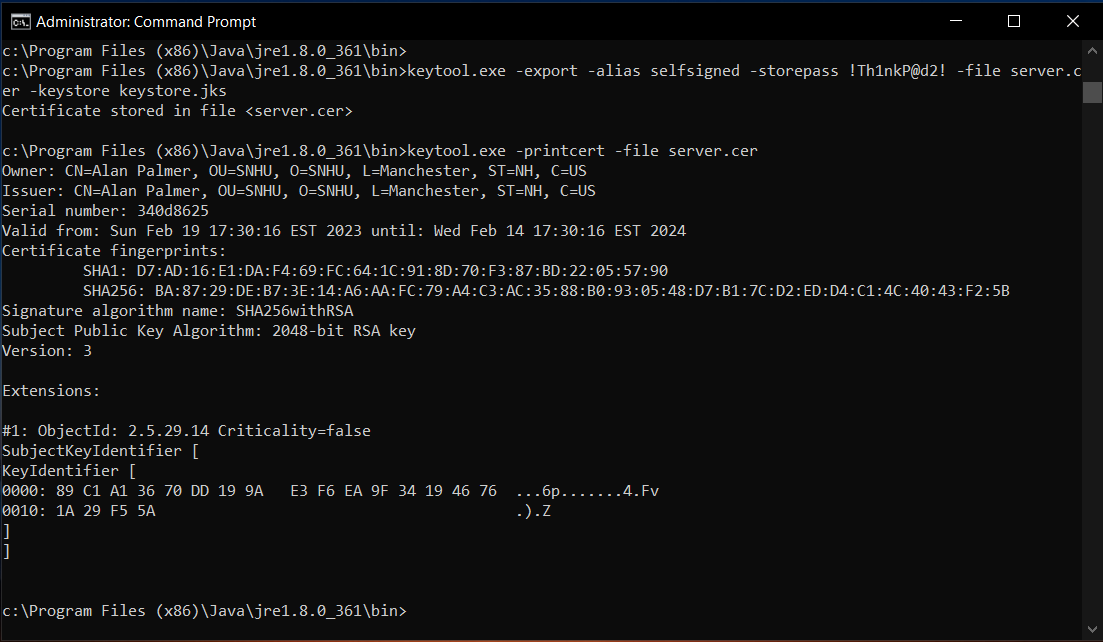
Alan Palmer

## Algorithm Cipher

RSA 256-bit is the best cipher to use in this scenario. It is symmetric, uses a secret key, has never been broken, and is the strongest available. The cipher's hash and bit levels mathematically alter the data through a complex process. The time and processing power necessary to decrypt the data would prove too much to make the effort worthwhile. The more complex the hash and the higher the bit level adds decades of time it would take to access the data through brute force methods. The process of encrypting the data could take hours or days depending on the size of the data, hash, and bit sized used. A true random number that is used as the baseline for mathematical computations makes it almost impossible for an attacker to guess the number used. Symmetric uses a secret key. The same key or passphrase is used to encrypt and decrypt the data. This makes key protection and management paramount. This method is commonly used for data at rest encryption. Non-symmetric or asymmetric uses a private key to encrypt and a public key, which is shared, to decrypt. This method is more useful in digitally signing data for originator verification purposes. Encryption was originally developed in the ancient world and used by the Hebrews, Greeks and Spartans, to name a few, to hide information in plain sight. The Medieval times saw advances in breaking the old ancient ciphers and creating more advanced ciphers for use mostly by the Italians. During the two World Wars, advancements in ciphers, mostly to allow for coded communications during heated times of war while using machines due to the complexity. Modern cryptography, in use today, is so complex that the use of computers, vice mechanical machines, is necessary. Advancements in complexity today is unprecedented and new bit levels and methods are constantly being developed, broken, and redeveloped for the main purpose of securing data and communications.

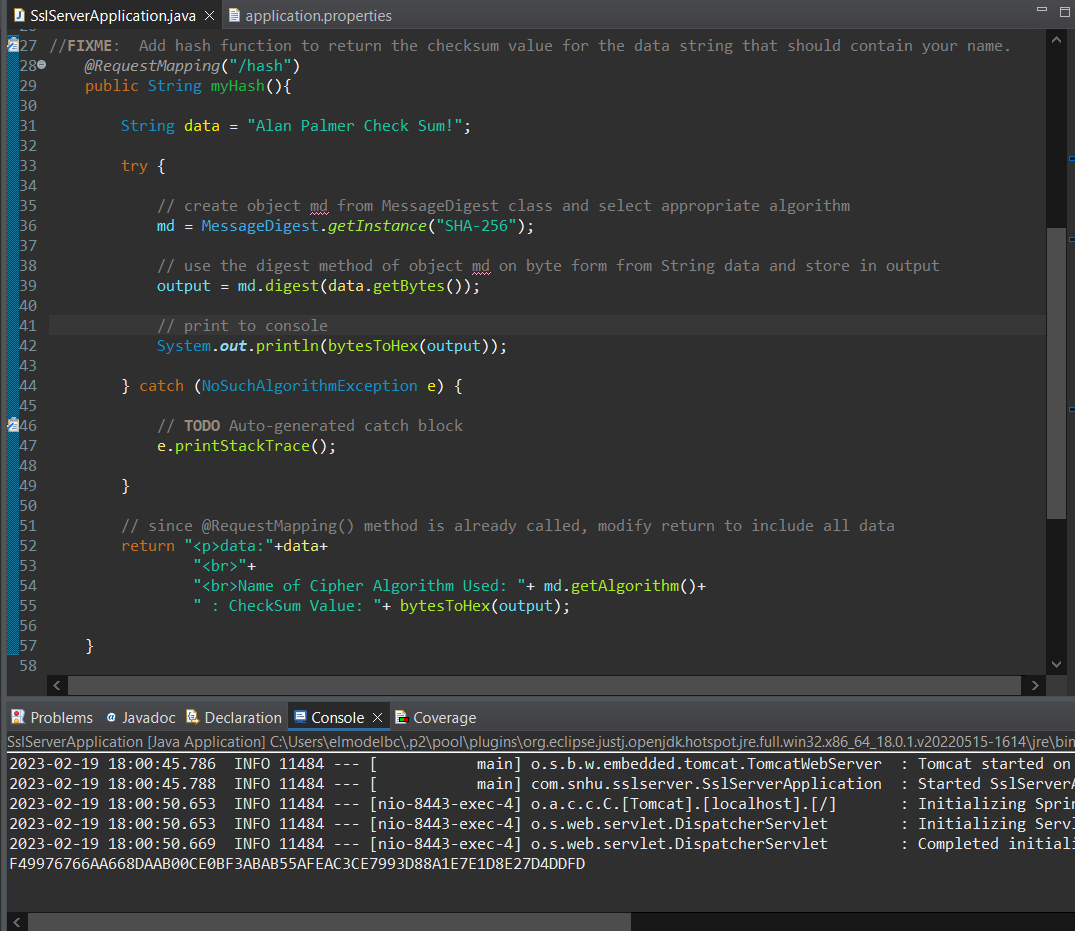
## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

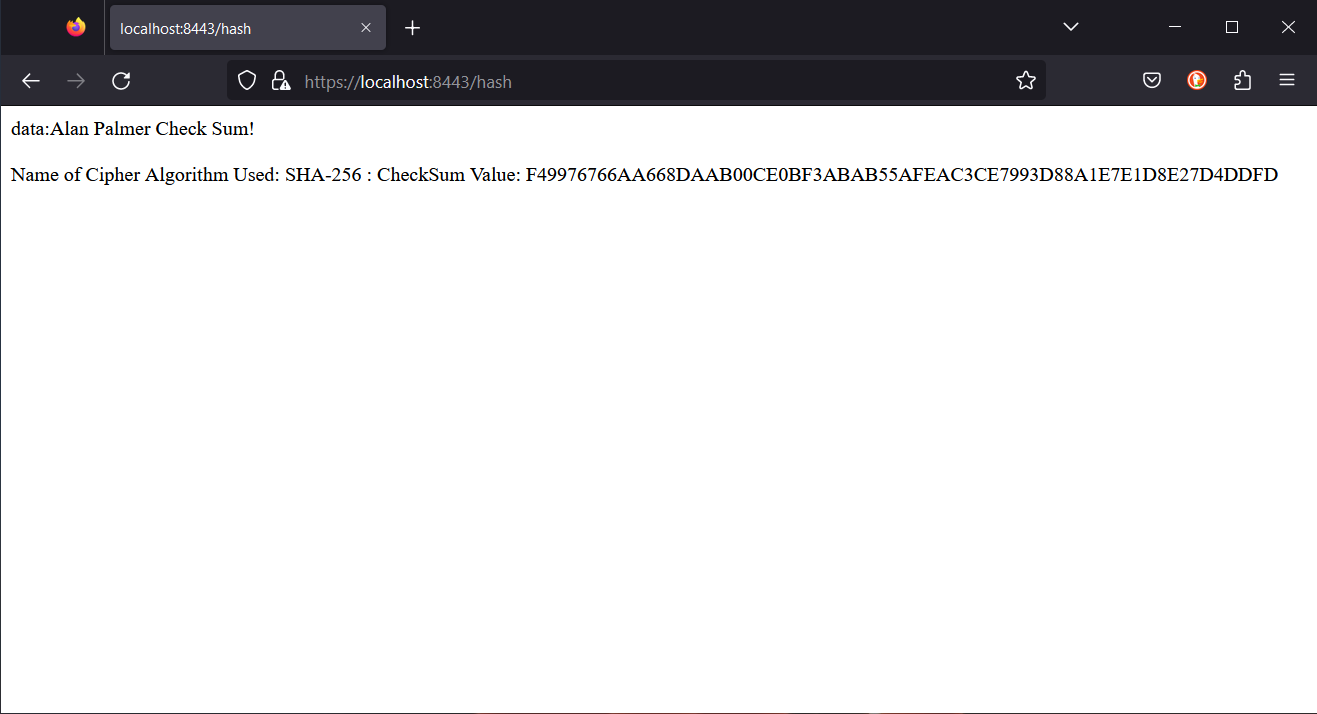
Insert a screenshot below of the checksum verification.



**\*\* Console output displays same checksum shown in the next screen shot**

## Secure Communications

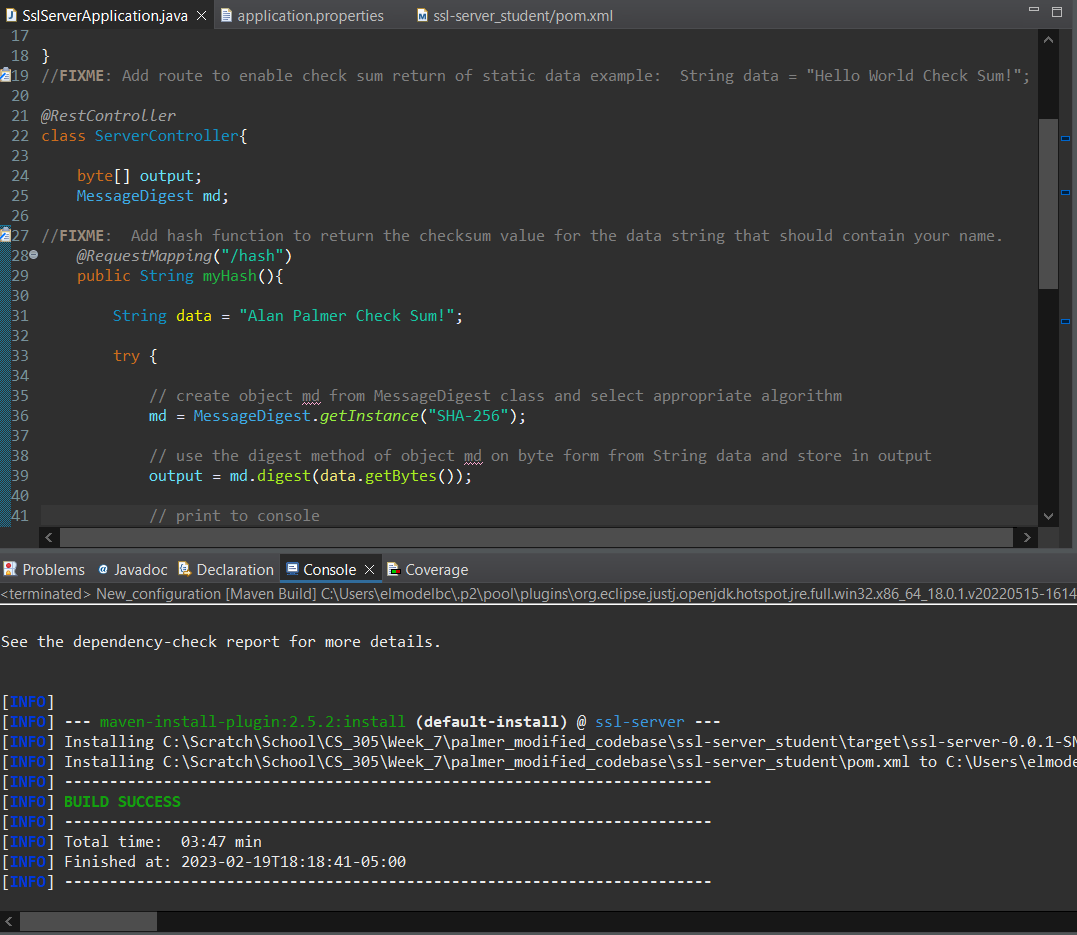
Insert a screenshot below of the web browser that shows a secure webpage.

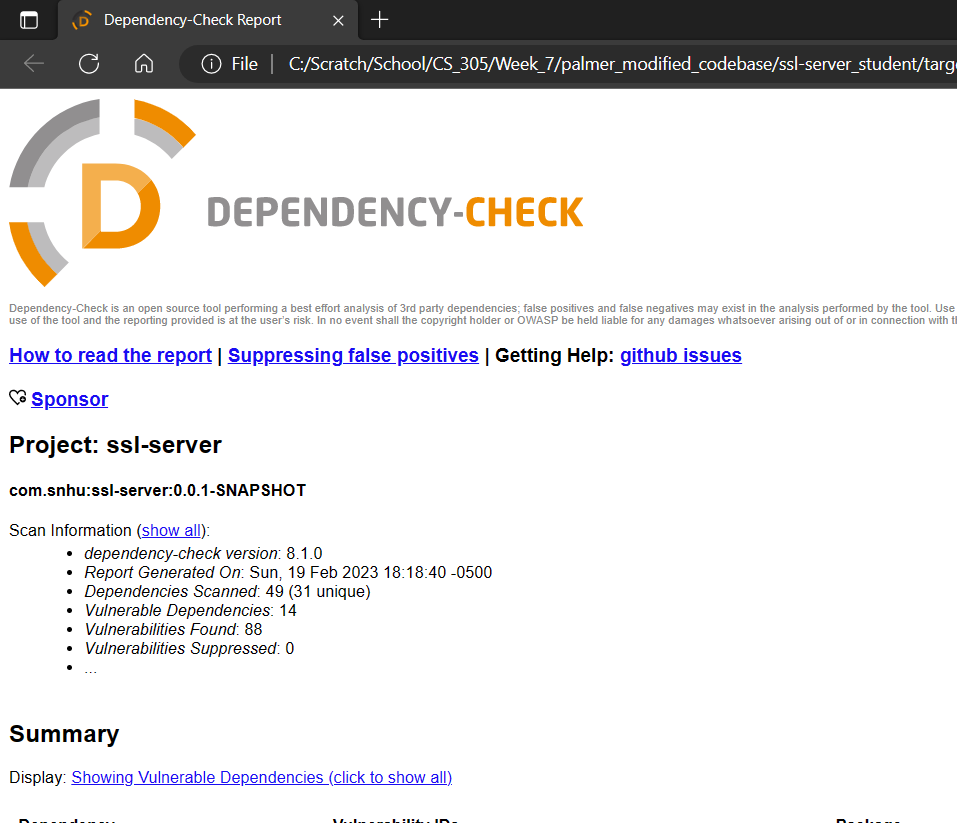


**\*\* Secure website output displays same checksum shown in the previous console screen shot**

## Secondary Testing

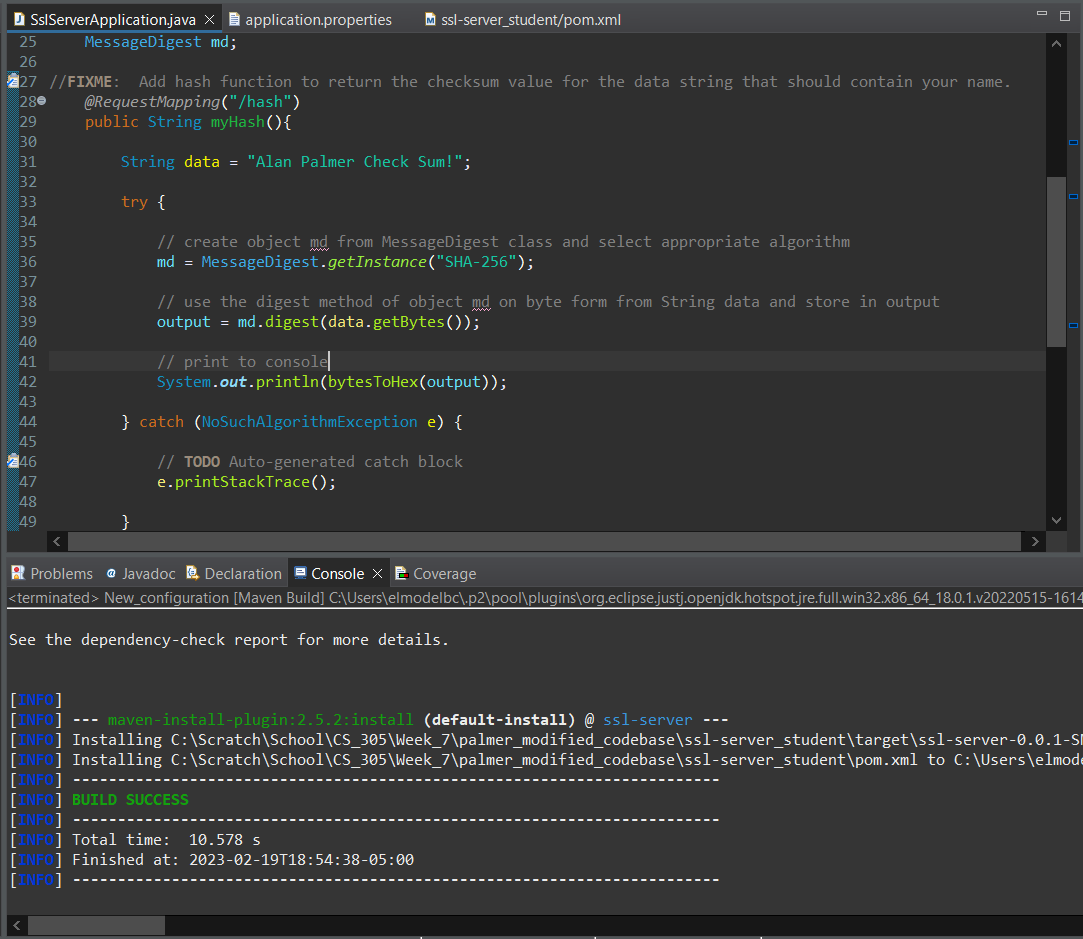
Insert screenshots below of the refactored code executed without errors and the dependency-check report.

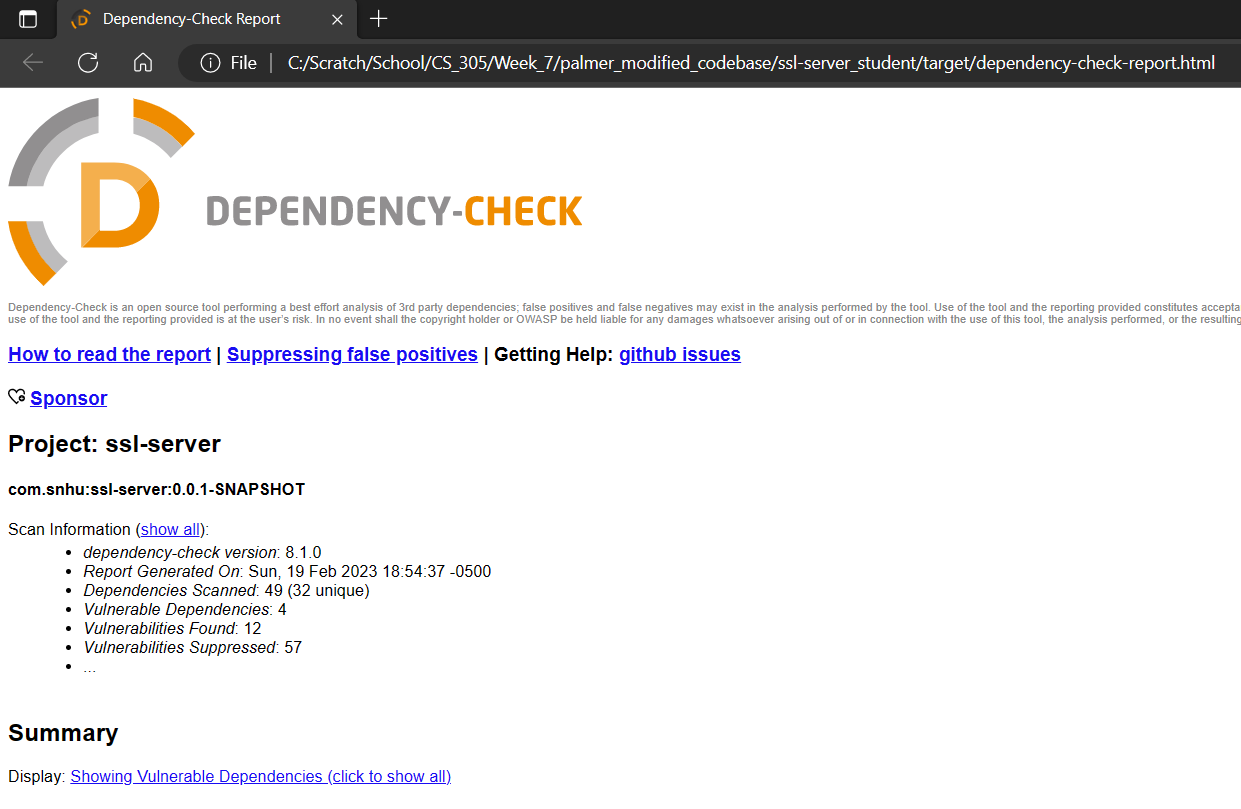




## Functional Testing

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





## Summary

With refactoring the code, we were able to address secure input and representations, secure API interactions, encryption use and vulnerabilities, secure distributed composing, secure error handling, secure coding practices and patterns, and secure data structures. Now, we can move on to adding depth and layers of security through views, models, controllers, data access, services, plug-ins, and APIs with rigorous testing and extensive code review.

## Industry Standard Best Practices

Industry best practices were used by coding with standard coding practices, the use of standardized APIs and libraries, the use of static testing with known industry standard tools using documented practices and procedures. The use of industry standard certificate creation and implementation was used as well. Using industry standard best practices is paramount in the development of an application that is compatible with current ecosystems. This mentality will only increase the speed at which vulnerabilities can be tracked and mitigated. When industry standards are used, the developer and the customer are on the same playing field. The product is built on industry standard technologies which are captured in an RFC that the customer can read and challenge the developer to build a better product.