

# 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** | **12/12/2024** | **Matthew Carrillo** |  |

## 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.

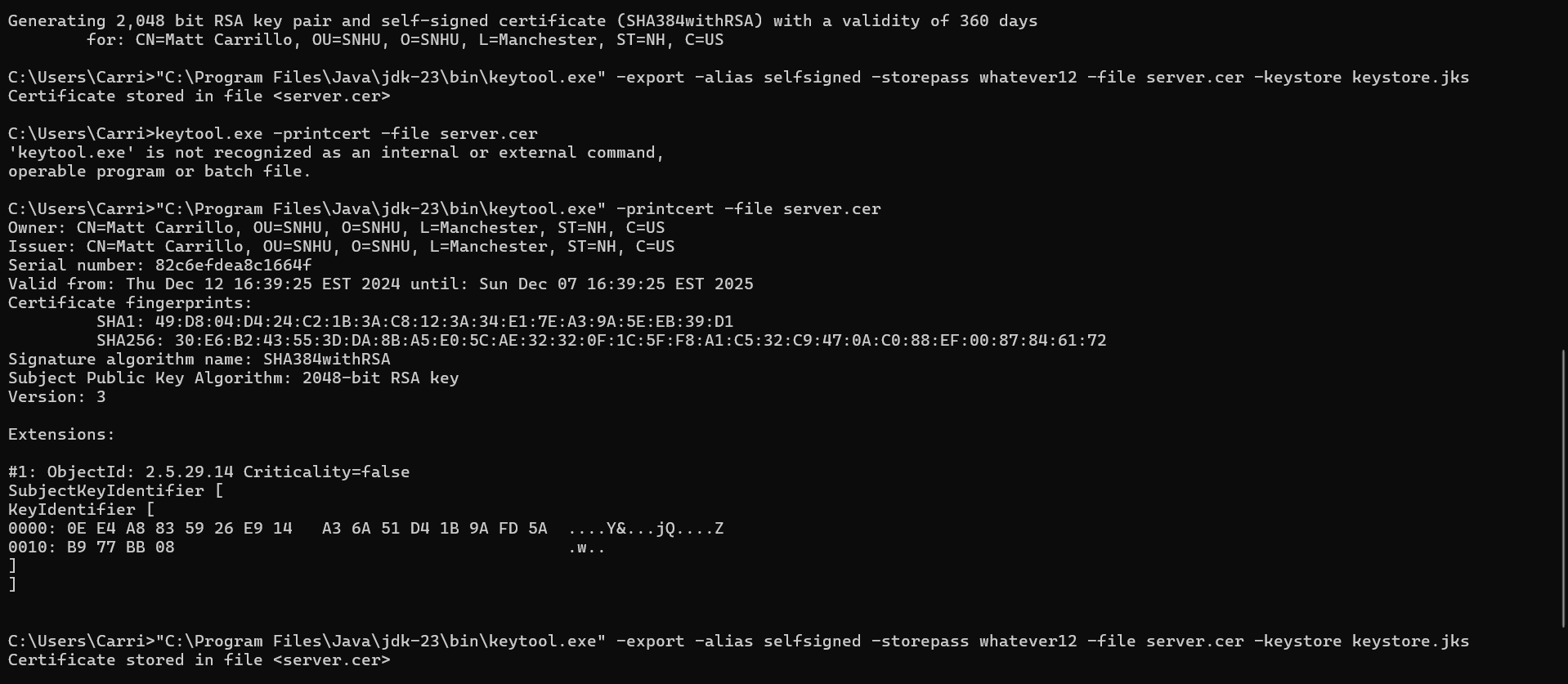
## Developer

Matthew Carrillo

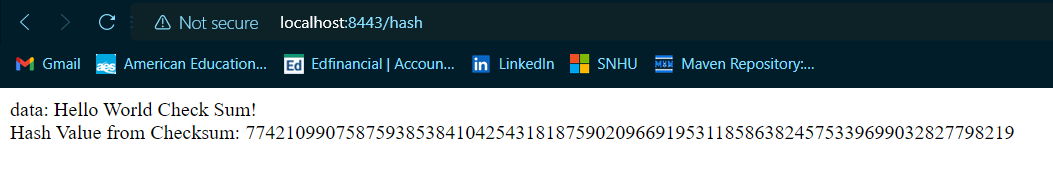
## Algorithm Cipher

Artemis Financial has requested a method of securing their web application and communication systems, so I will recommend some form of data encryption. Specifically I recommend the use of the SHA-256 algorithm being implemented because of its low chance of having a collision vulnerability and because it is generally regarded as the most secure hash function to utilize. For clarification, a collision in this case is when an algorithm takes a string to produce a hash value, but then the same hash value can be produced by a completely distinct string from the first one. SHA-256 outputs characters using lowercase letters and all numbers zero to nine, meaning it can use 36^64 different hash values and make it so two pieces of data having the same hash value very unlikely.

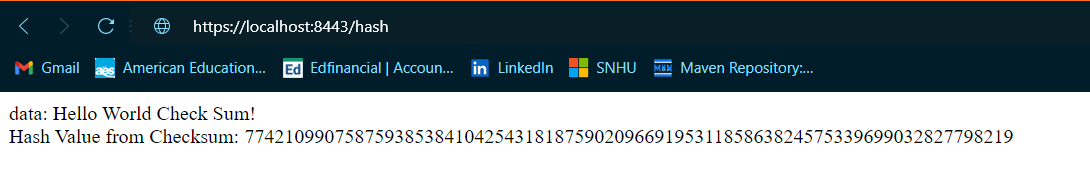
## Certificate Generation



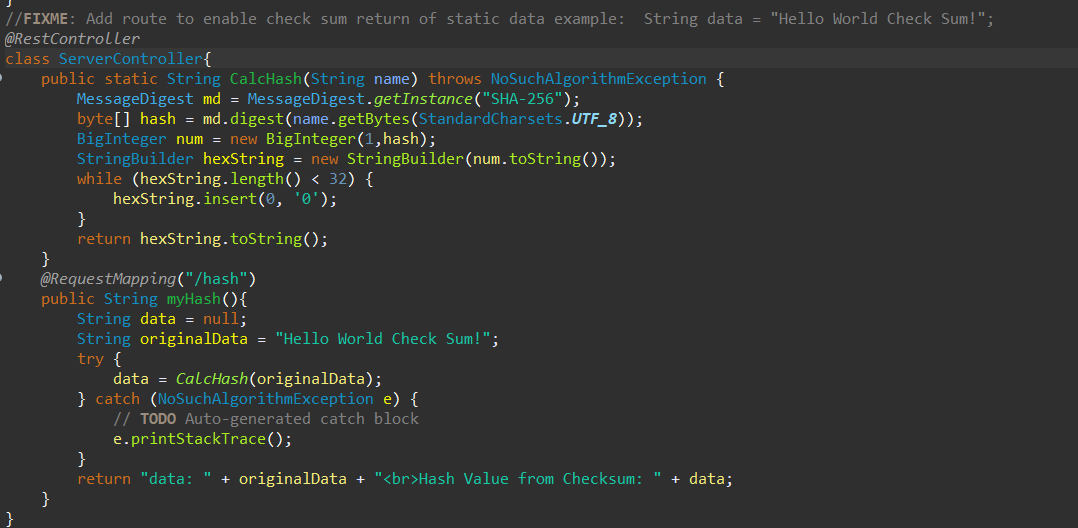
## Deploy Cipher

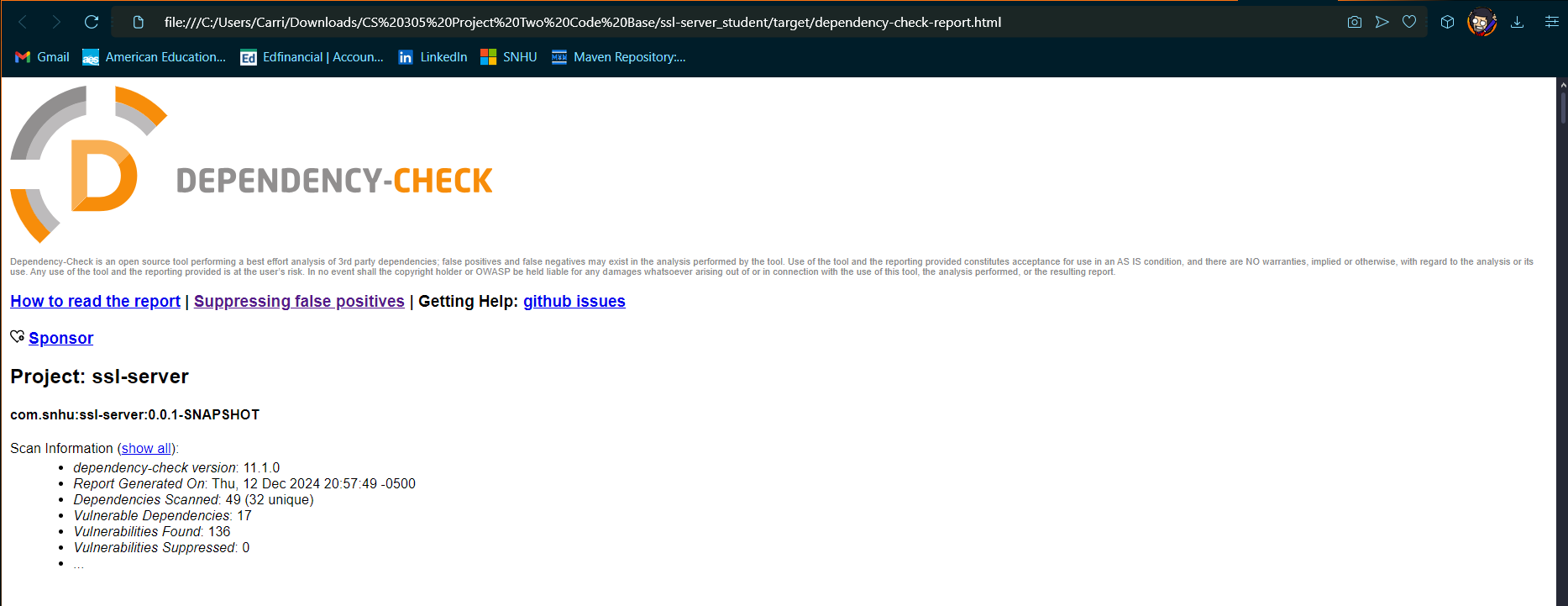


## Secure Communications

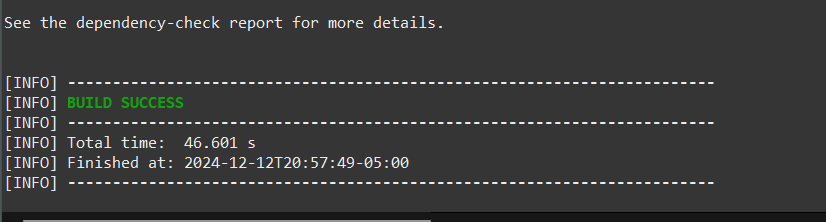


## Secondary Testing





## Functional Testing



## Summary

The application has self-signed certificates which enable the usage of HTTPS. It was refactored in the pom.xml file in order to ensure that all vulnerabilities found during the dependency check were addressed. The first step in the work process was making sure the certificates were created appropriately, and making sure that the website is secure so that users can be sure they are at the correct URL. These security measures also help the program adhere to industry best practices. The next step was ensuring that the hashing function operated correctly with the checksum. Once it is verified that the data of the user is effectively scrambled and difficult to recover, then external attacks become less likely to occur. The last step involved is to make sure that as many vulnerabilities as possible were fixed. By doing so, the software and systems are further kept current and aligned with best practices. By doing this, outdated systems are protected from even more protected potential attacks.