****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **4/17/2022** | **Tony Chiesa** |  |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

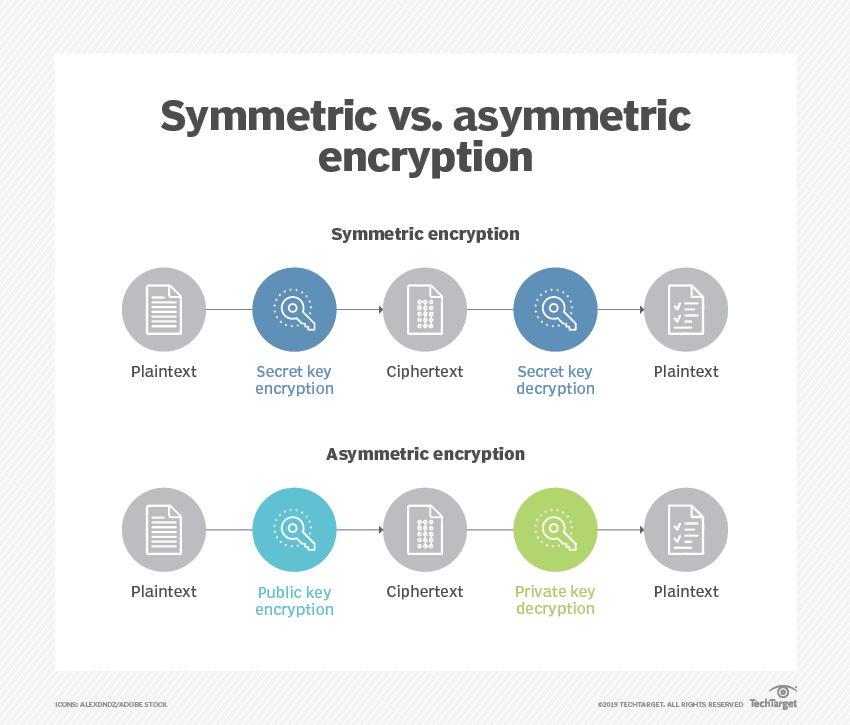
## Developer

Tony Chiesa

## 1. Algorithm Cipher

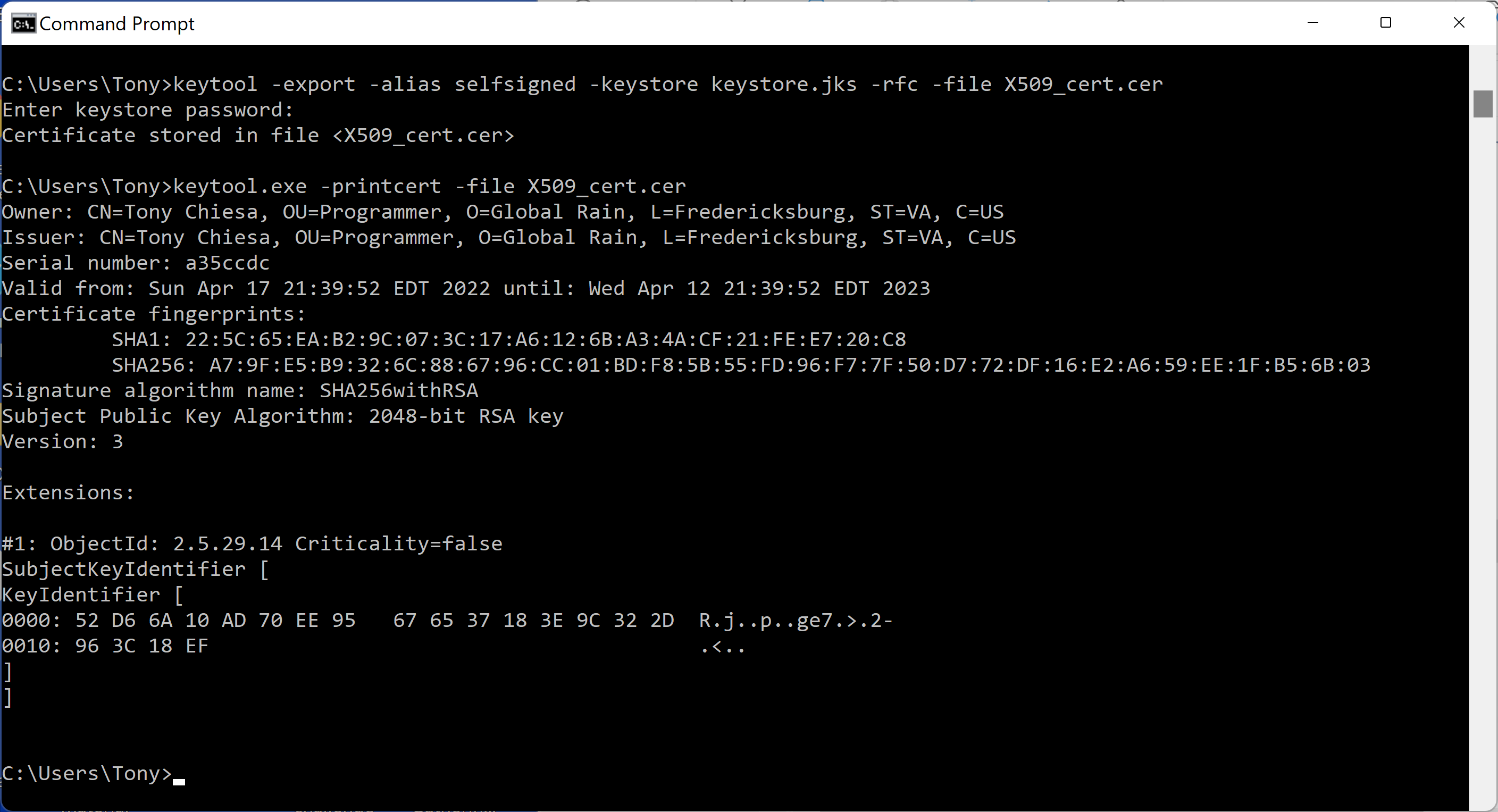
The chosen encryption algorithm cipher to deploy for Artemis Financial is the Secure Hash Algorithm (SHA)-256 encryption algorithm. In cryptography, a collision occurs when two different texts produce the same hash. Using the SHA-256 algorithm, the chance of a collision is 1 in 2^256. SHA-256 is secure enough to be trusted by the United States Federal Government as the Department of Defense (DoD) industry standard.

A secondary recommendation for an encryption is the RSA encryption algorithm. RSA encryption is a public-key encryption scheme, also known as asymmetric encryption. RSA encryption encrypts data via a code known as a public key. Public keys may be shared freely without exposing the encrypted data, because of the unique mathematical aspects of the RSA method, after a message has been encrypted with the public key, it can only be decoded using another key known as the private key. Each RSA user has a key pair that includes their public and private keys. The private key, as the name implies, must be kept confidential.

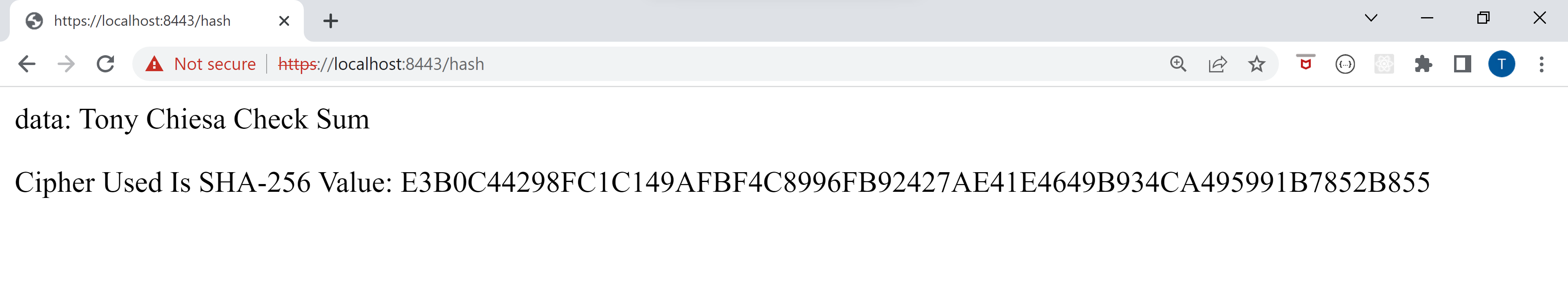


RSA keys are normally 1024 or 2048 bits long, however, experts feel that 1024-bit keys are no longer entirely safe against all threats. As a result, the government and several companies are requiring a minimum key length of 2048 bits, which is what is recommended for Artemis Financial being in the financial industry. The computational complexity of factoring big numbers drives RSA security. The current state and ability of quantum computing to factor larger and larger numbers currently limits the required length of keys. However, as computing power grows and more efficient factoring methods are developed larger keys will be required. The biggest advantage of RSA encryption is that it is stronger than any other symmetric key algorithm. The big disadvantage to RSA encryption is that it has a large amount of computation, which requires time. However, Artemis Financial is not worried about complex time requirements because that are more worried about clients personal and financial information being secure.

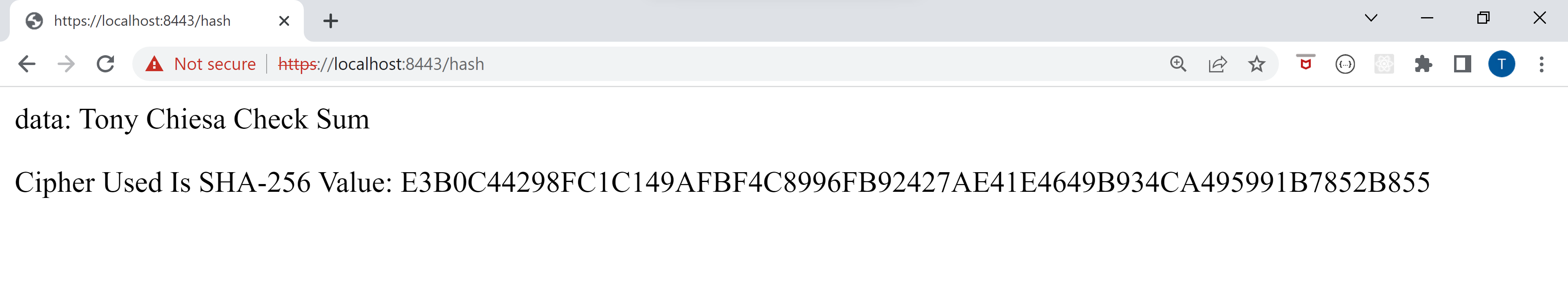
## 2. Certificate Generation



## 3. Deploy Cipher

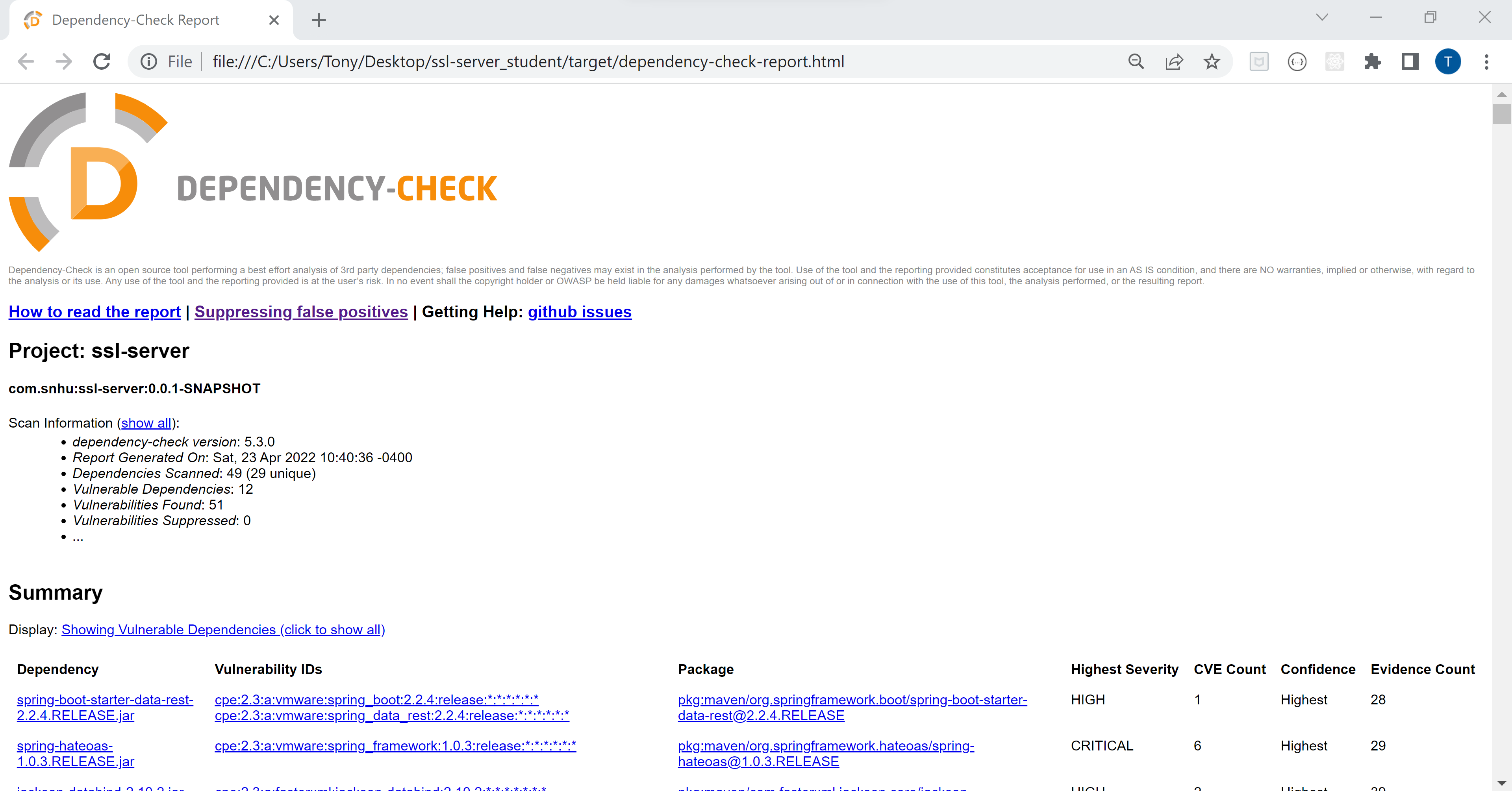


## 4. Secure Communications

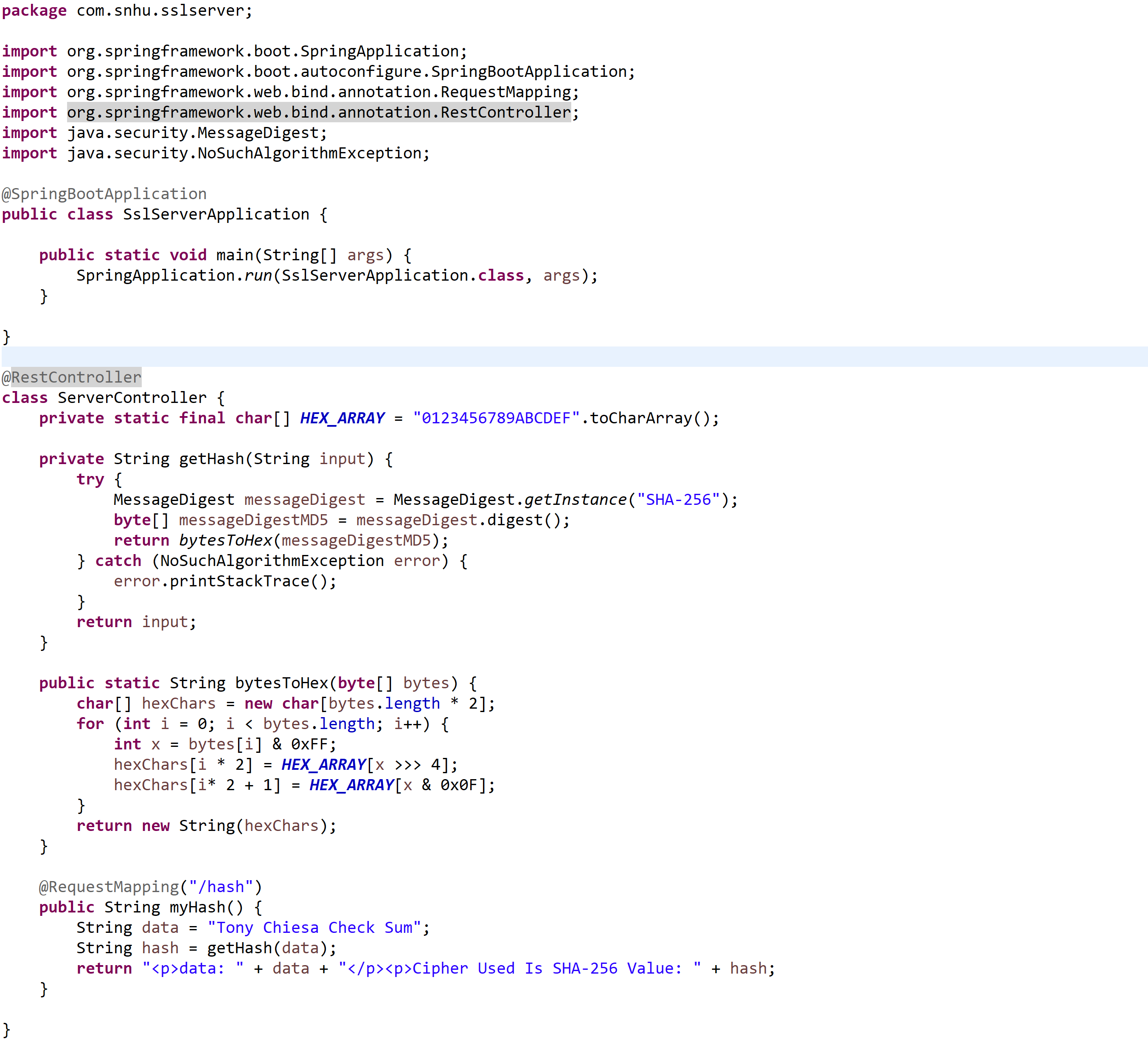


## 5. Secondary Testing





## 6. Functional Testing





## 7. Summary

During this project I implemented a protected RestController to act as the secure controller for my program's hash RESTful stop. The ServerController class is responsible for matching the issues depicted in the vulnerability assessment diagram. I also selected to use the SHA-256 hashing cipher since it is incredibly safe and has a very low probability of collisions. Using the SHA-256 algorithm, the chance of a collision is 1 in 2^256. This is secure enough to be trusted by the United States Federal Government as the Department of Defense (DoD) industry standard. To maintain the application's existing security, I would recommend scheduled dependency checks to stay up to speed on any vulnerabilities as well as an appropriate plan utilizing a continuous integration and continuous delivery tool like Jenkins, which would allow for security to be continuously checked at all levels of the lifecycle.