****

# 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** | **10/17/2021** | **Antonio Ocasio** | **Made security modifications** |

## 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

Antonio Ocasio

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

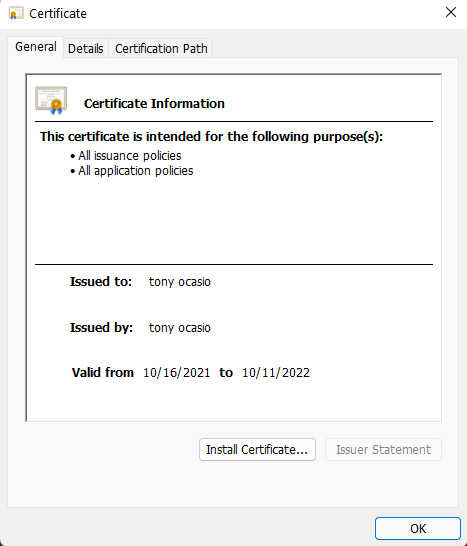
* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

The encryption algorithm cypher that I chose to use is Advanced Encryption Standard 256. This encryption algorithm has been used since 2001 and is almost impenetrable to brute-force attacks. Secure hashing algorithms are functions that can be used with AES to increase security. Secure hashing algorithms take plaintext and converts it to a ciphered text which is only converted back by using a special key. Symmetric keys are keys that are used to encrypt and decrypt data, this key is the same for both encrypting and decrypting. Non-symmetric keys are two separate keys, one used to encrypt data and the other key is used to decrypt data. Random numbers are used by using a specific seed which will produce the same random sequence every time. Encryption has been used in computers since the early 1970’s. The data encryption standard was used from 1973 until 1997 until it was cracked for the first time, then we switched to AES-256, which is currently what we use and is almost impossible to crack.

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

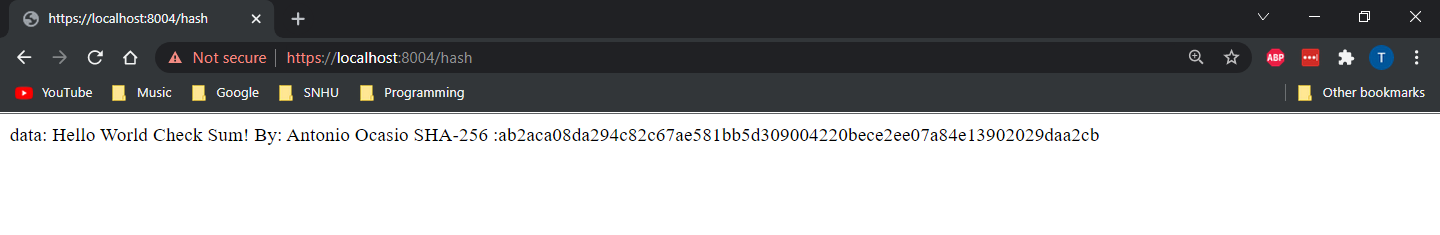
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

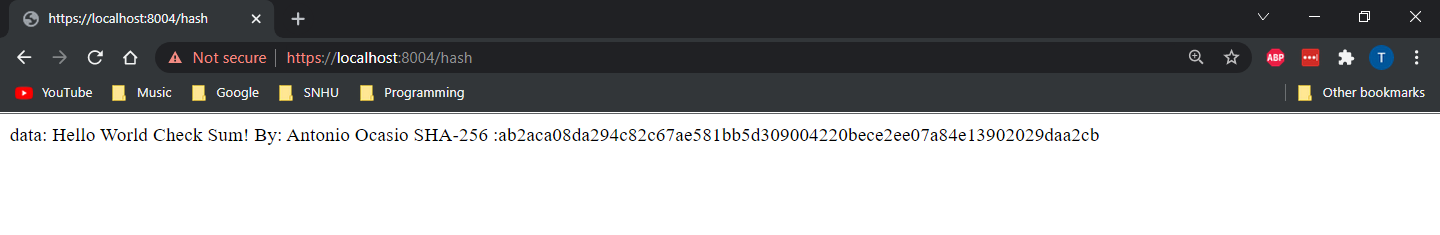
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

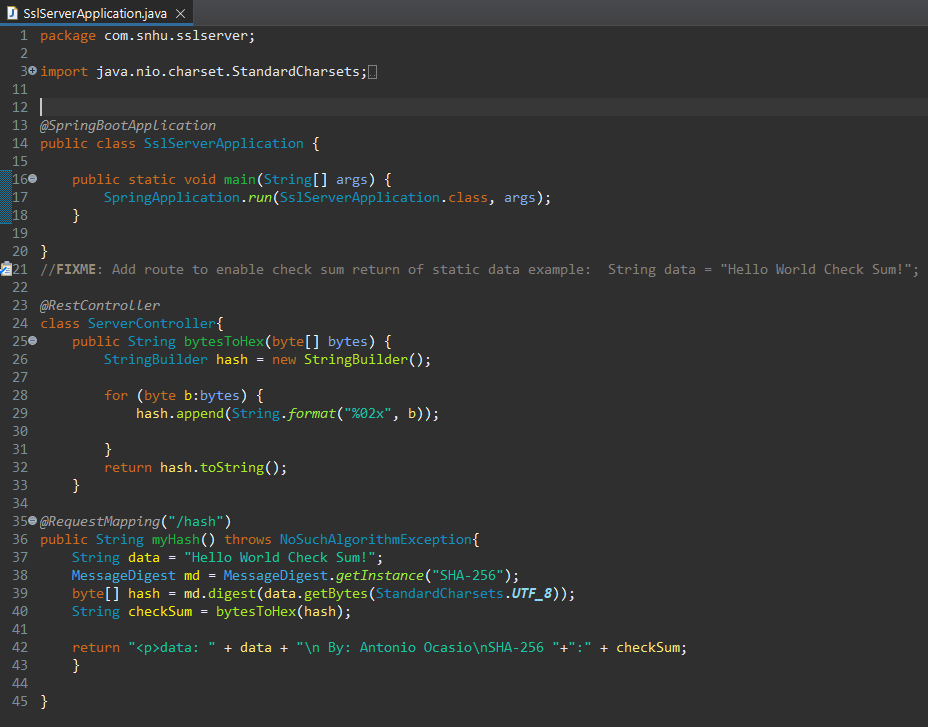
* Insert a screenshot below of the web browser that shows a secure webpage.

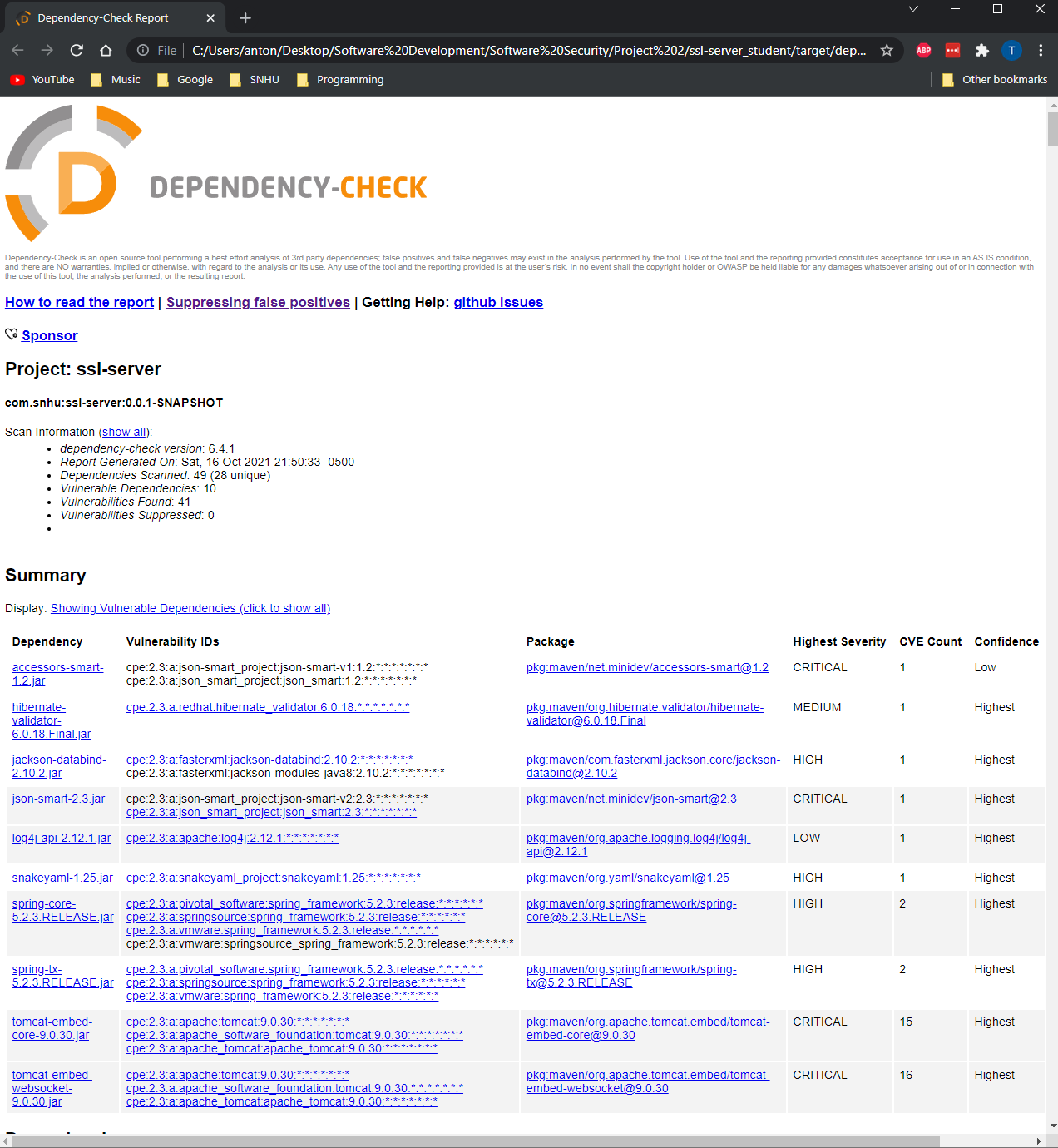


## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

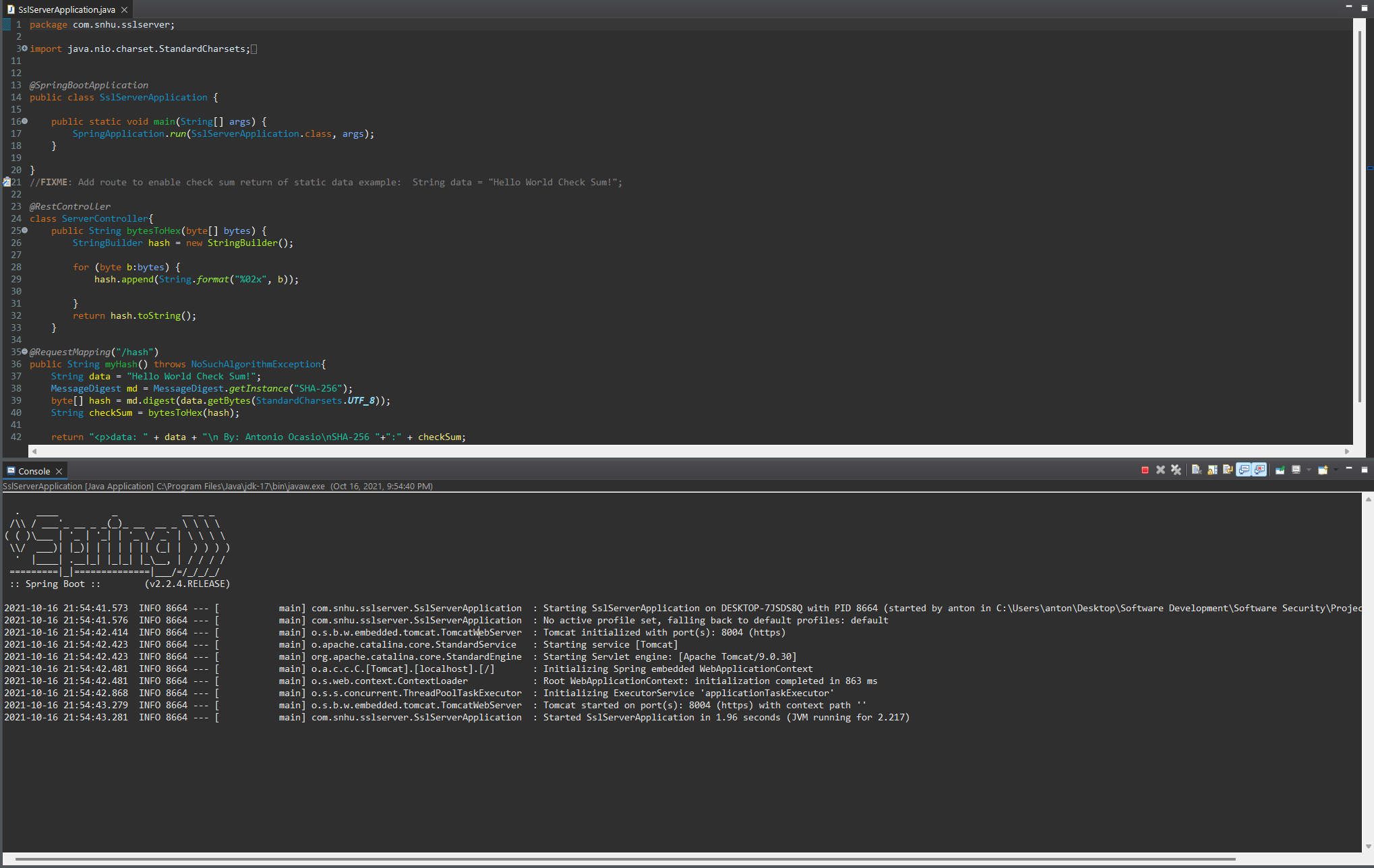


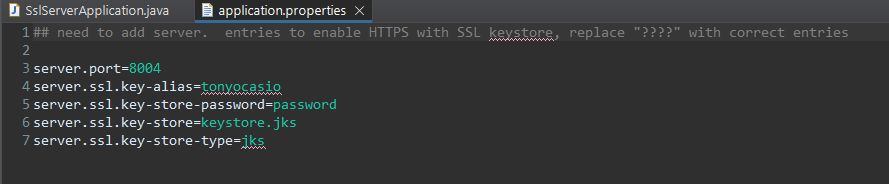


## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.





The only vulnerability that is found in the code is in the file “application.properties”, where the password is visible which is not secure. A fix for this would be to have the password encrypted by another file or program so the password cannot be viewed.

## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

The code has been refactored by adding self-signed certificates which allows us to use https instead of http, this makes the connection secure. The next thing that we did was use the maven dependency check on the “pom.xml” file to check for any vulnerabilities. We can see in the dependency check html file that there are still vulnerabilities. The way that we added layers of security is by adding certificates that are made specifically for the company so the webpage is secure which makes sure that we know who we are dealing with. After creating the certificate, the next step was to create the hash function. This uses SHA-256 encryption which is almost impenetrable. After creating the hash function, it was time to test and make sure that the data is encrypted correctly. You can see in my screenshot above that “Hello World Check Sum!” is properly converted by using the hash function.