****

# 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** | **6/19/2022** | **Kathryn McNeil** |  |

## 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

Kathryn McNeil

## 1. Algorithm Cipher

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

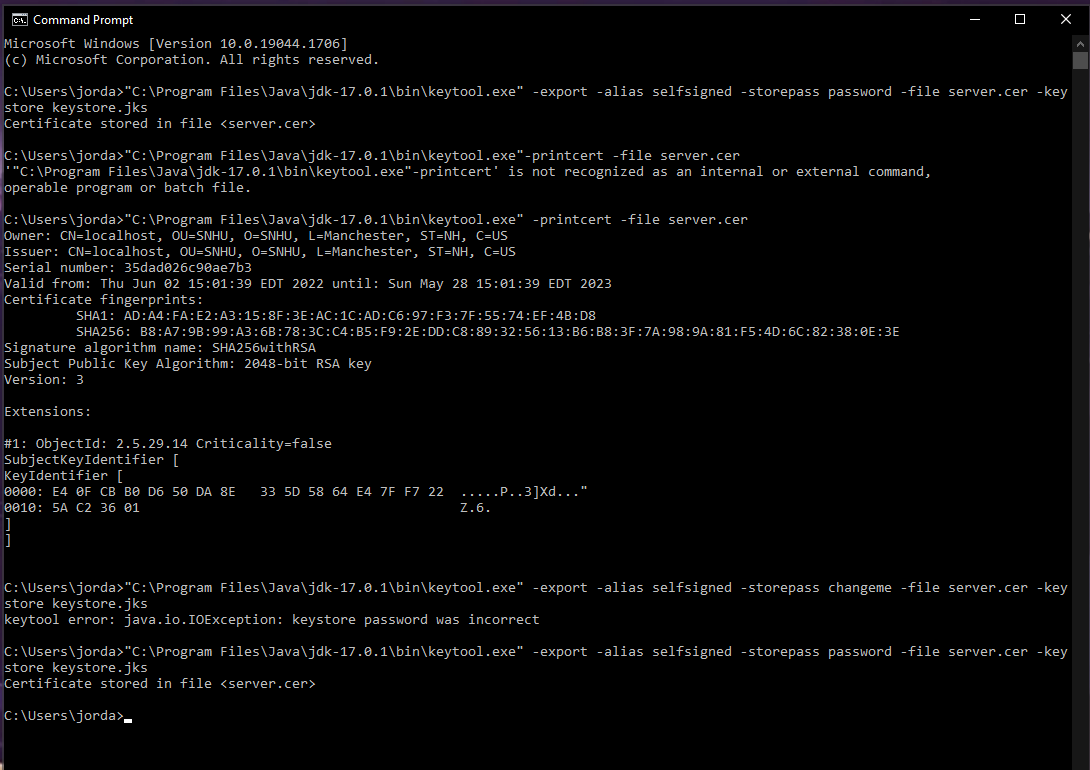
* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

Given the security vulnerabilities an appropriate encryption algorithm to deploy is the SHA-256 (Secure Hash Algorithm). The SHA-256 algorithm is a novel hash function which uses eight 32-bit words to produce a hash value that is always 256-bits. This hash value is irreversible meaning that when it is passed back through the function, the hash should not produce the original value. This also means that it is not a symmetric key algorithm as the same key cannot be used to encrypt and decrypt the information. The algorithm is part of the SHA-2 family which was first published in 2001. It uses Merkle-Damgård construction making it collision-resistant. The algorithm also uses the Davies-Meyer structure which turns “any normal block cipher into a one-way compression function” (“One-way compression function”, 2022). Both the Merkle-Damgård construction and Davies-Meyer structure make one-way compression functions, hence why SHA-256 is irreversible.

## 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.

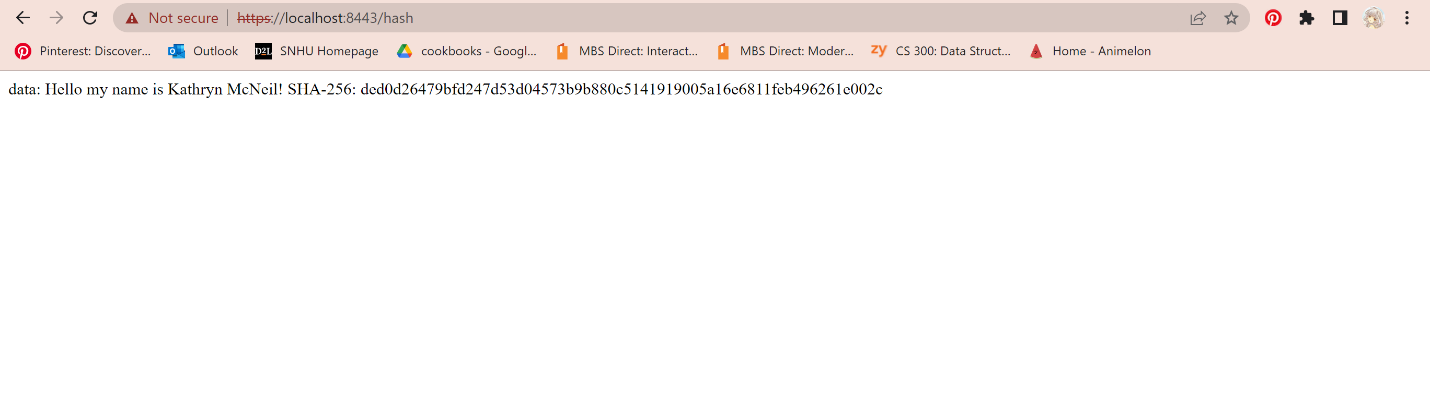
The certificate file after it has been exported to notepad.

The certificate file when printed in the command prompt.

## 3. Deploy Cipher

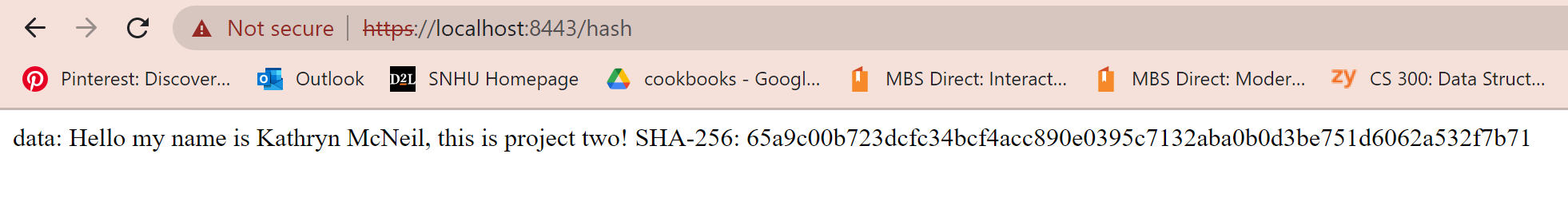
Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

* Text, application

  Description automatically generated with medium confidenceInsert 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.

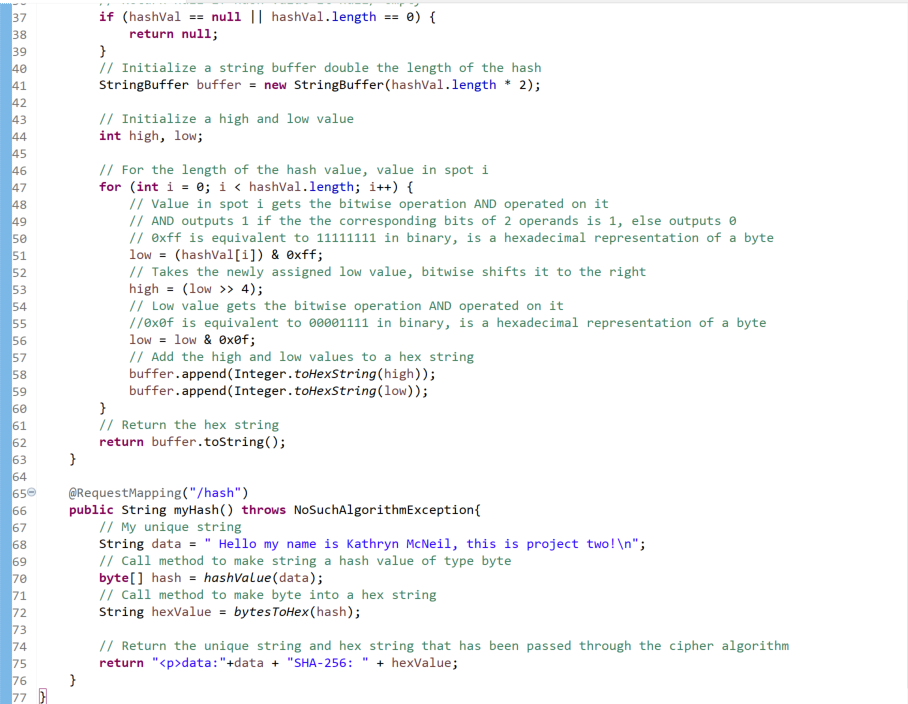
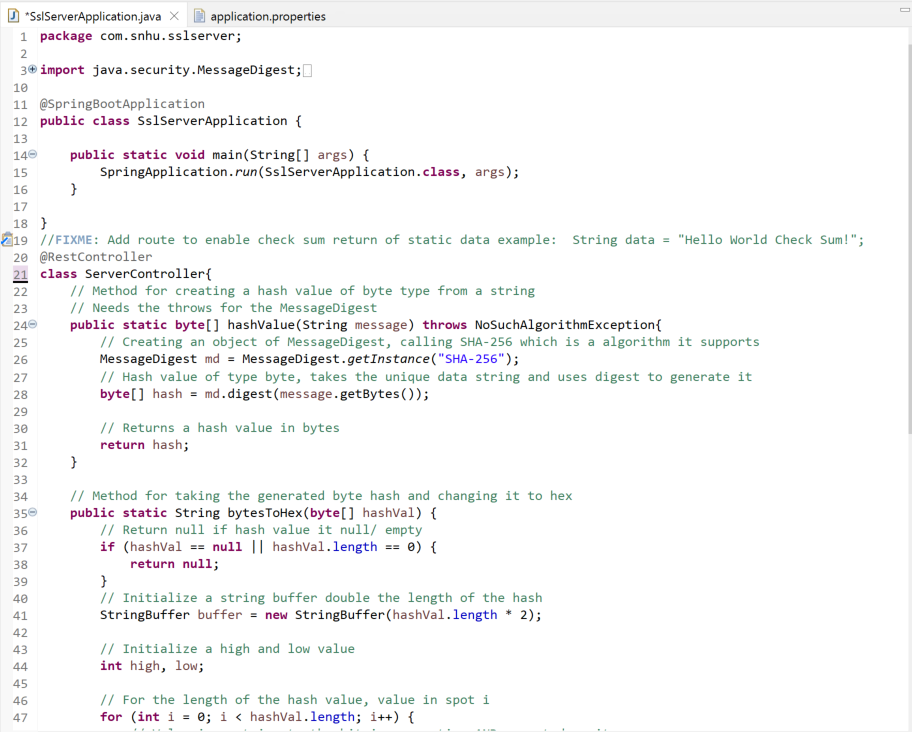
## 5. Secondary Testing

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

Graphical user interface, text, application

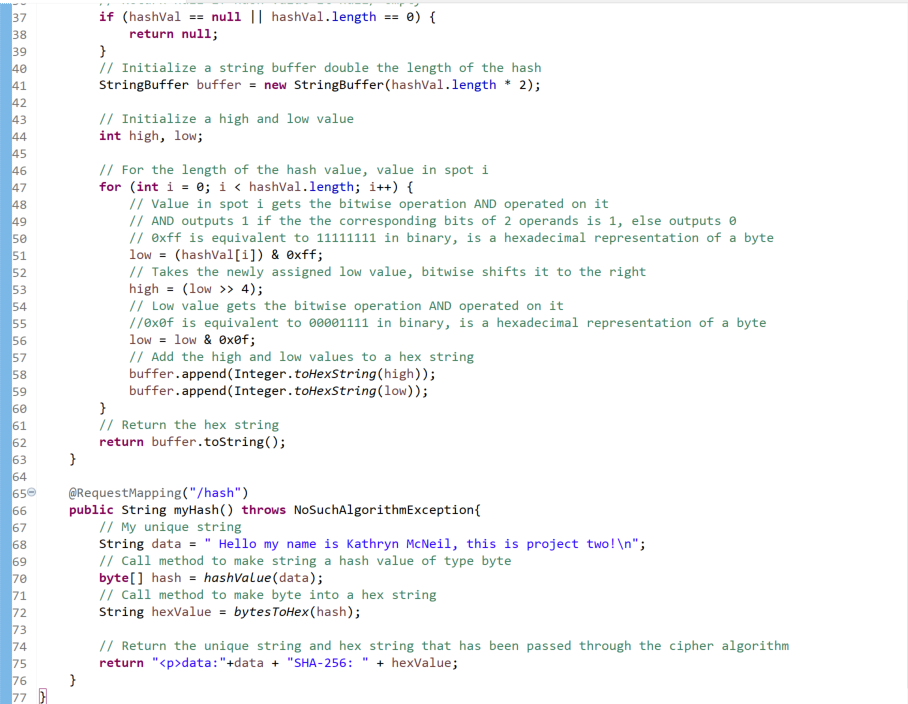
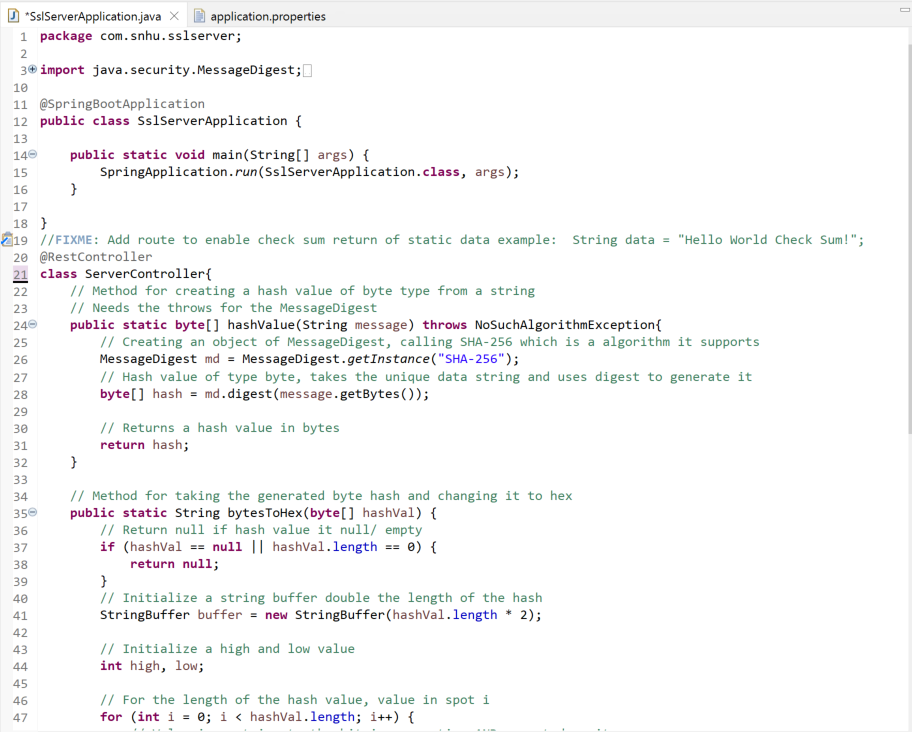
Description automatically generated



## 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.



## 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.

I have written code that takes a string literal, digests it using SHA-256 to create a hash value in bytes, and then converts that to a hex value. By using SHA-256 the string data can be encrypted in such a way that avoids collisions. SHA-256 was created by the United States National Security Agency and is secure enough to be used by the U.S. Government.

When refactoring the code, I addressed a few areas of security as seen on the Vulnerability Assessment Process Flow Diagram. For instance, I practiced architecture review. I looked at the requirements for the code and reviewed how it should be architected. A self-signed certificate was used to have secure API interactions, and encryption was used to secure data. I also made sure to follow good coding practices to ensure my code was well-made and secure. Using a certificate in the API added a layer of security as it ensured that the connection was secure. Encryption also added a layer of security as it encrypts data that may be sensitive. Both of these can help protect users and the companies as it makes the application more secure. The customer can maintain current security of the software application by making sure to update all plug ins that are used. For instance, spring boot and tomcat should both be kept up to date in order to resolve any security problems that are found. A current encryption algorithm should also be used, as if the one being used becomes out of date then it will not be as secure as it is now.

Resources:

Merkle–Damgård construction. (2022, April 9). In *Wikipedia*. <https://en.wikipedia.org/w/index.php?title=Merkle%E2%80%93Damg%C3%A5rd_construction&oldid=1081786995>

One-way compression function. (2022, June 13). In *Wikipedia*. <https://en.wikipedia.org/w/index.php?title=One-way_compression_function&oldid=1092854203>

SHA-2. (2022, June 17). In *Wikipedia*. <https://en.wikipedia.org/w/index.php?title=SHA-2&oldid=1093520556>

Simplilearn. (2022, May 25). *A definitive guide to learn the SHA-256 (Secure Hash Algorithms)*. https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm