****

# 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** | **12/11/21** | **Gayle Sakiewicz** | **See attached CS 305 Project 2.zip** |

## 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

Gayle Sakiewicz

## 1. Algorithm Cipher

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

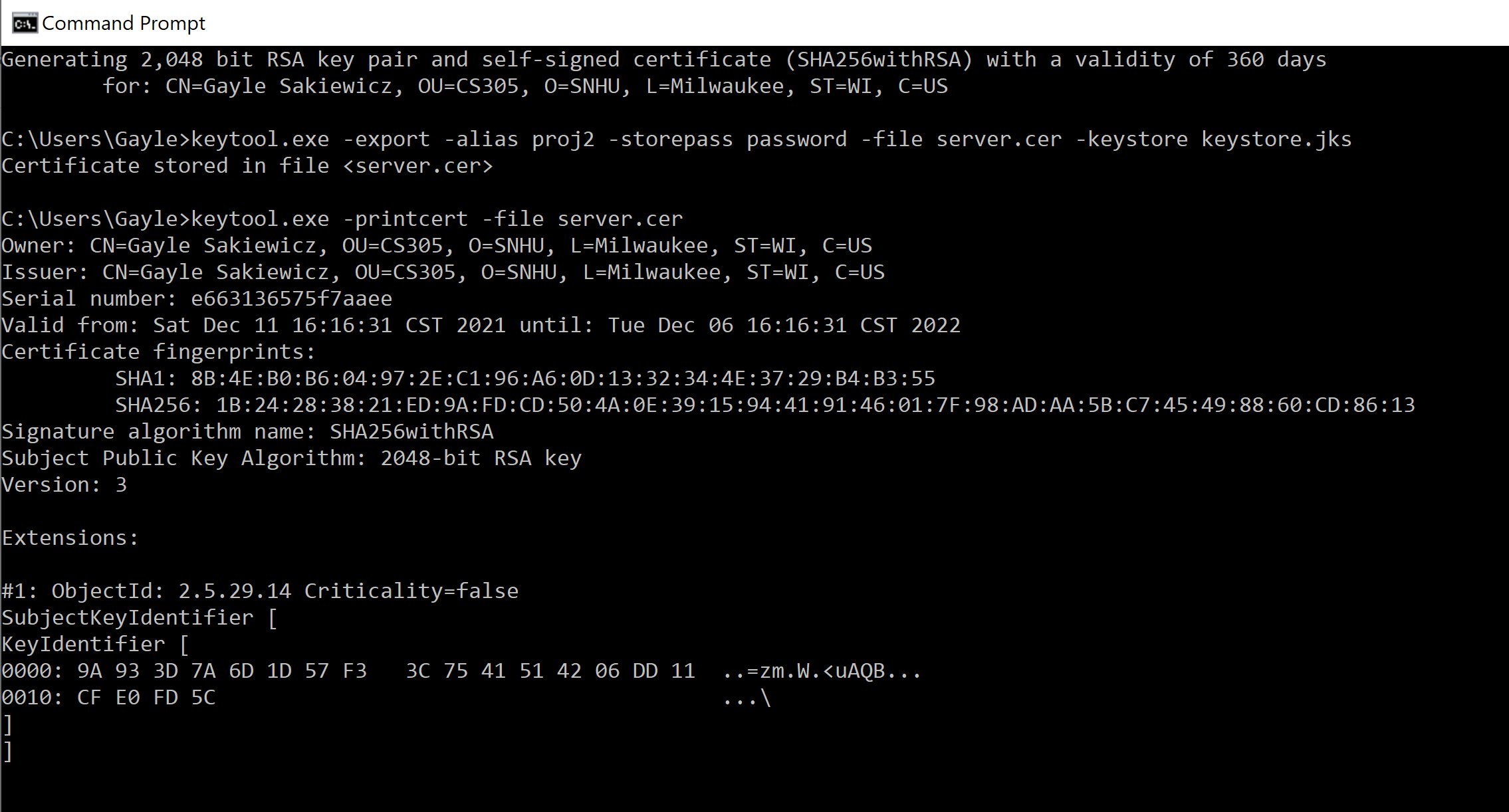
* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

The encryption algorithm cipher used in the attached zip file (CS 305 Project 2) is SHA-256. SHA-256 is recommended as it is infeasible for it to have any collisions. It also exceeds the recommended least amount of bits of 180 with 256 bits. SHA-256 is also recommended and approved by the National Institute of Standards and Technology and Secretary of Commerce. The algorithm cipher scrambles information for anyone without the key, then with the correct key, the data can be decrypted. With symmetric cryptography, “a secret key is used to “scramble” information in a way that is virtually impossible to unscramble without the secret key…this type of cryptography is called symmetric because the same key is used for both encrypting and decrypting the data. That key must remain secret for the system to remain secure” (Jeganathan, 2019). Asymmetric cryptography “is essential to signing…uses two different keys. One key, the private key, is kept secret and is known only to the sender… used to sign the data. The other key can be made public and can be known by anyone—hence the name, public key cryptography. This public key is used to validate the signature” (Jeganathan, 2019). Encryption algorithms are updated just like anything else when vulnerabilities and weaknesses are exposed. For instance, SSL was replaced with TLS and Google developers came up with Tink to replace Keyczar. Encryption algorithms evolve over time when new vulnerabilities are introduced, just as any other part of technology.

## 2. Certificate Generation

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

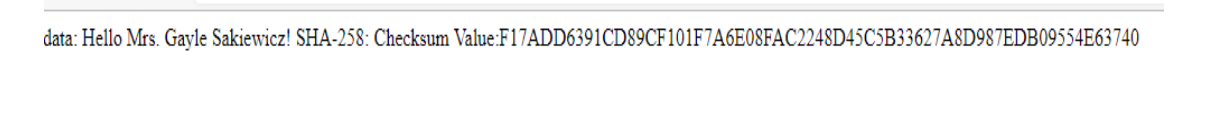
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

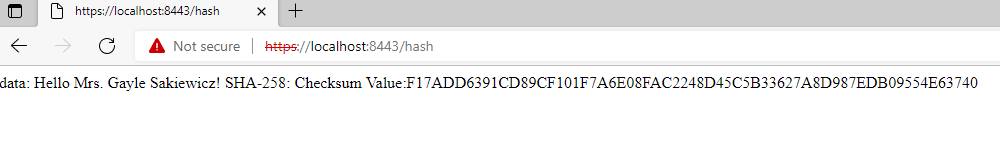
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

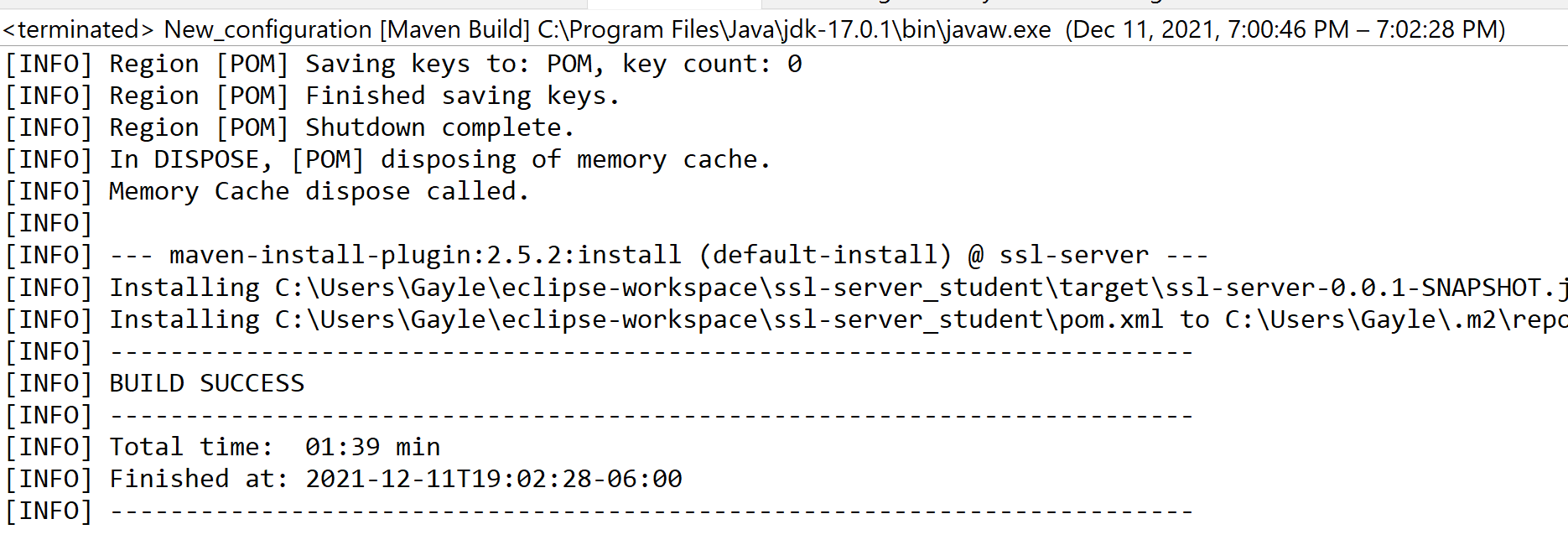
* Insert a screenshot below of the web browser that shows a secure webpage.

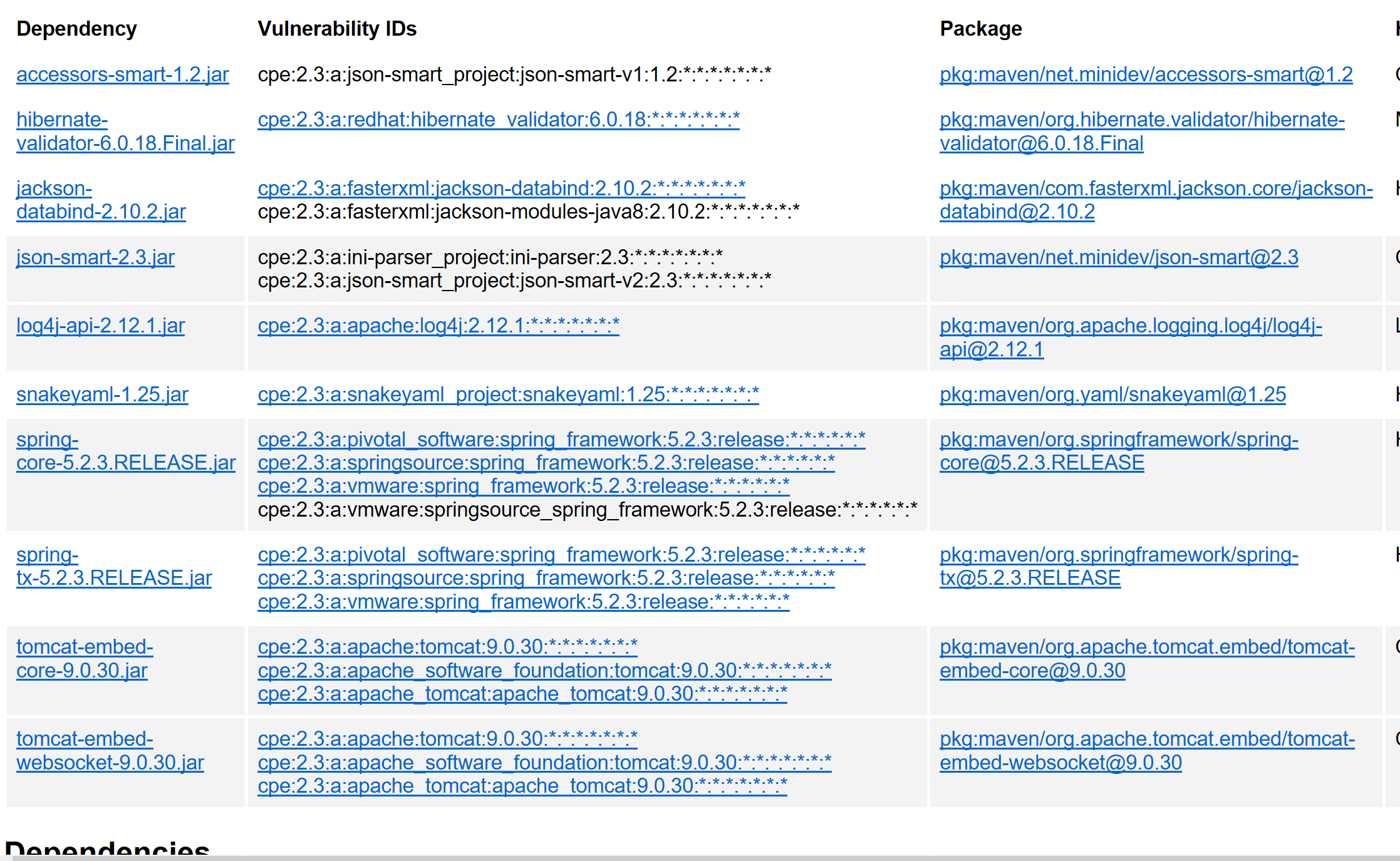


## 5. Secondary Testing

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

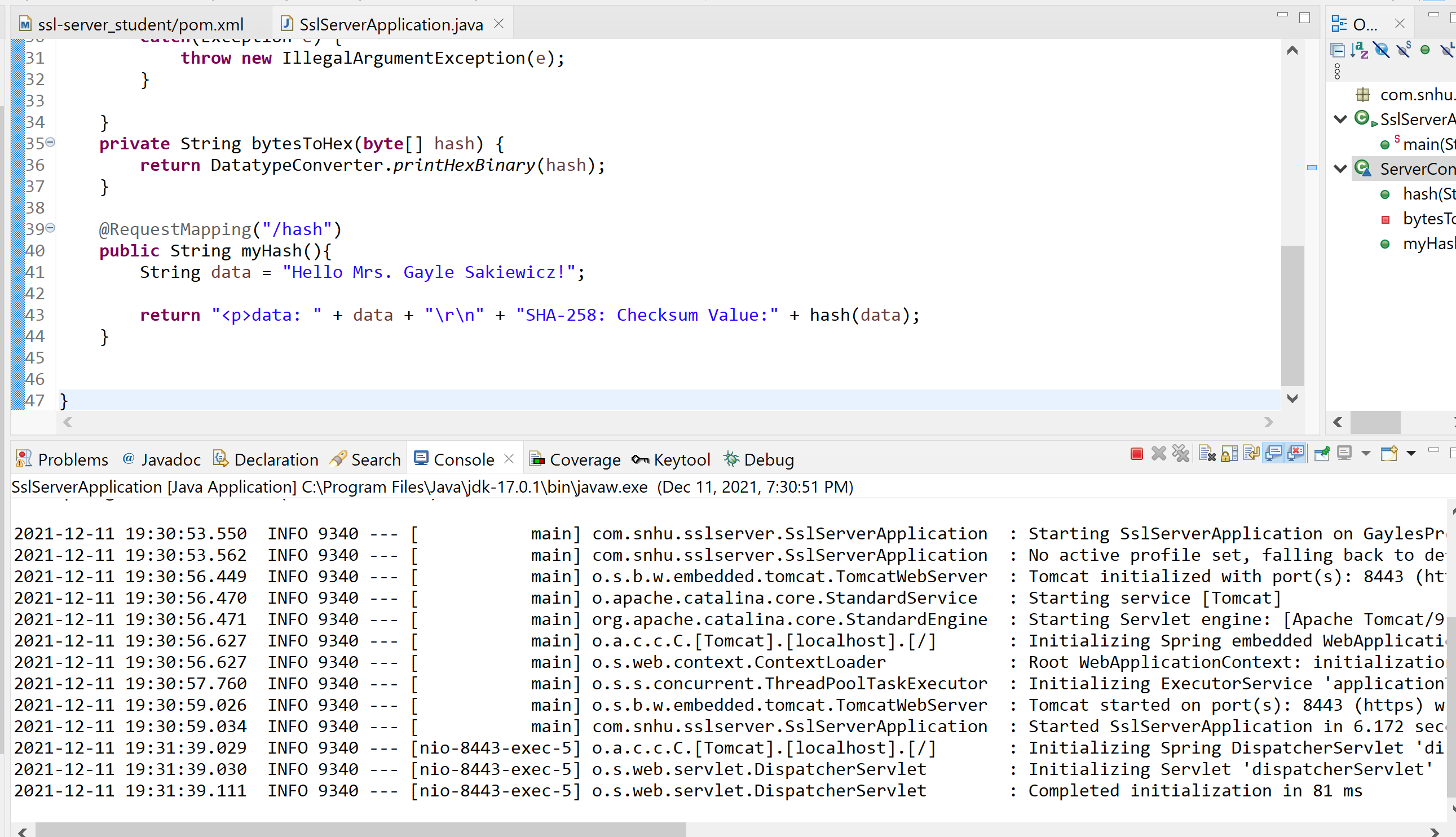




## 6. Functional Testing

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.



## 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.

Cryptography and secure API interactions were addressed (please see the attached zip file: CS 305 Project 2). The API already has a security measure in place from being RESTful with Spring, but I also used the Java security API when implementing the hash function, which brings in cryptography. The MessageDigest created the cryptographic hash function, SHA256, to encrypt the message in the application. Right now, a self-signed certificate is used, the company may want to move to a certificate authority in the future. The key for keeping the application secure is proper key management as well as keeping everything up to do with patches or updates.