

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History3](#_Toc33111302)

[Client3](#_Toc33111303)

[Instructions3](#_Toc33111304)

[Developer4](#_Toc33111305)

[1. Algorithm Cipher4](#_Toc33111306)

[2. Certificate Generation4](#_Toc33111307)

[3. Deploy Cipher5](#_Toc33111308)

[4. Secure Communications5](#_Toc33111309)

[5. Secondary Testing6](#_Toc33111310)

[6. Functional Testing7](#_Toc33111311)

[7. Summary8](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **4/17/2022** | **Ido Tal** |  |

## 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

Ido Tal

## 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.

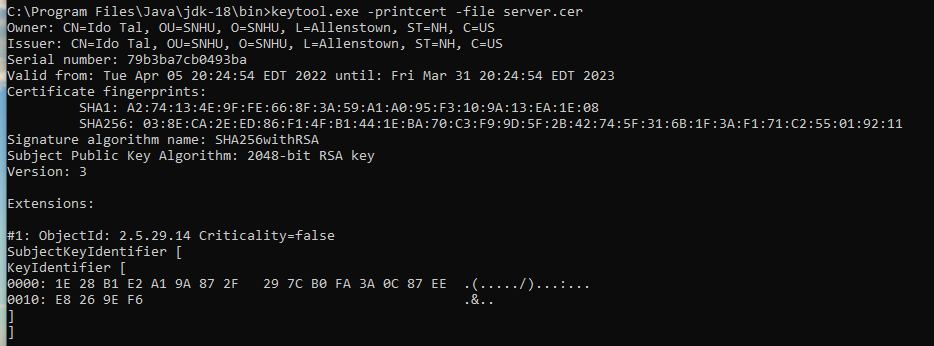
The encryption algorithm that I am recommending is the AES, specifically the 256 bit. The 256 bit is more secure then versions with less bits but it does run slower, but if security is a priority then it is preferable. Hashing functions work with encryption keys. Using a hashing function will result in a unique decryption that only works for a specific data. To decrypt the data a decryption key is also necessary. Next the unique hash is embedded into a certificate so that way it can be checked to make sure that no one tampered with the data. Ciphers are symmetric and asymmetric. Symmetric ciphers use only private keys for both encryption and decryption. Asymmetric ciphers use a private key for decryption and a public key for encryption. Random numbers help in improving encryption algorithm to make them harder to crack. Computers cannot choose random numbers so instead they use a method referred to as Cryptographic Pseudorandom Number Generator. This helps to generated what appears to be random numbers.

In the present day, many industries must comply with various regulations to make sure that their data is encrypted so that they provide security to their customers. The United States government also requires its various agencies to use SHA-256 to secure their data and communication. The financial industry also has to comply with regulation that requires data encryption algorithms to protect their customers and prevent data leaks from occurring. With online communication being the dominant and still growing mediums for doing business and carrying out financial transactions, it is important for Artemis Financial to keep up with the necessary security to protect their clients. To this end Artemis Financial needs to constantly work on their security to stay ahead of malicious users that want their data.

## 2. Certificate Generation

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

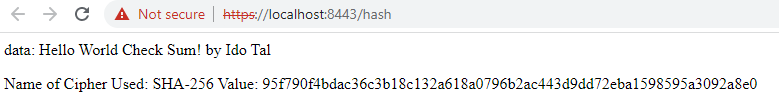
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

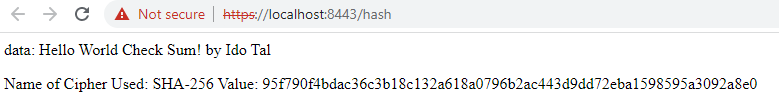


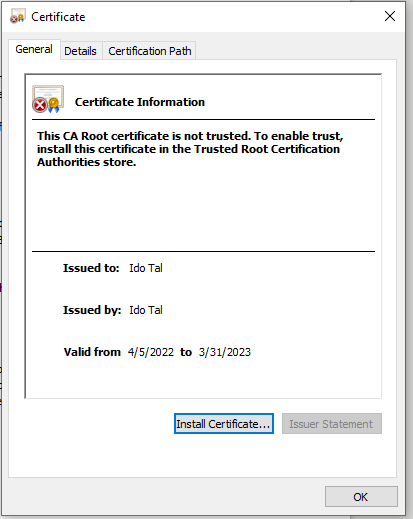
## 4. Secure Communications

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.

Secure communication did not work because it is not trusted, likely due to being self-signed.



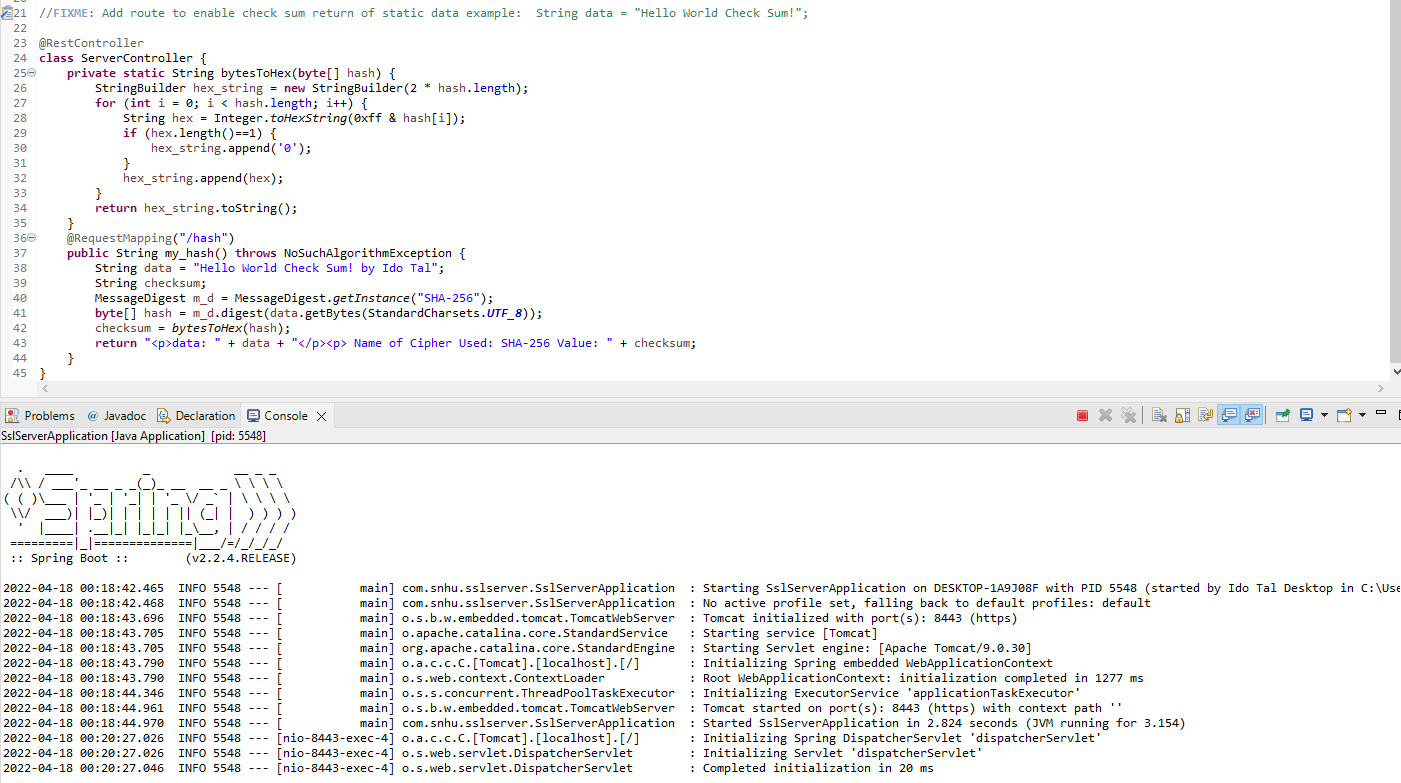


## 5. Secondary Testing

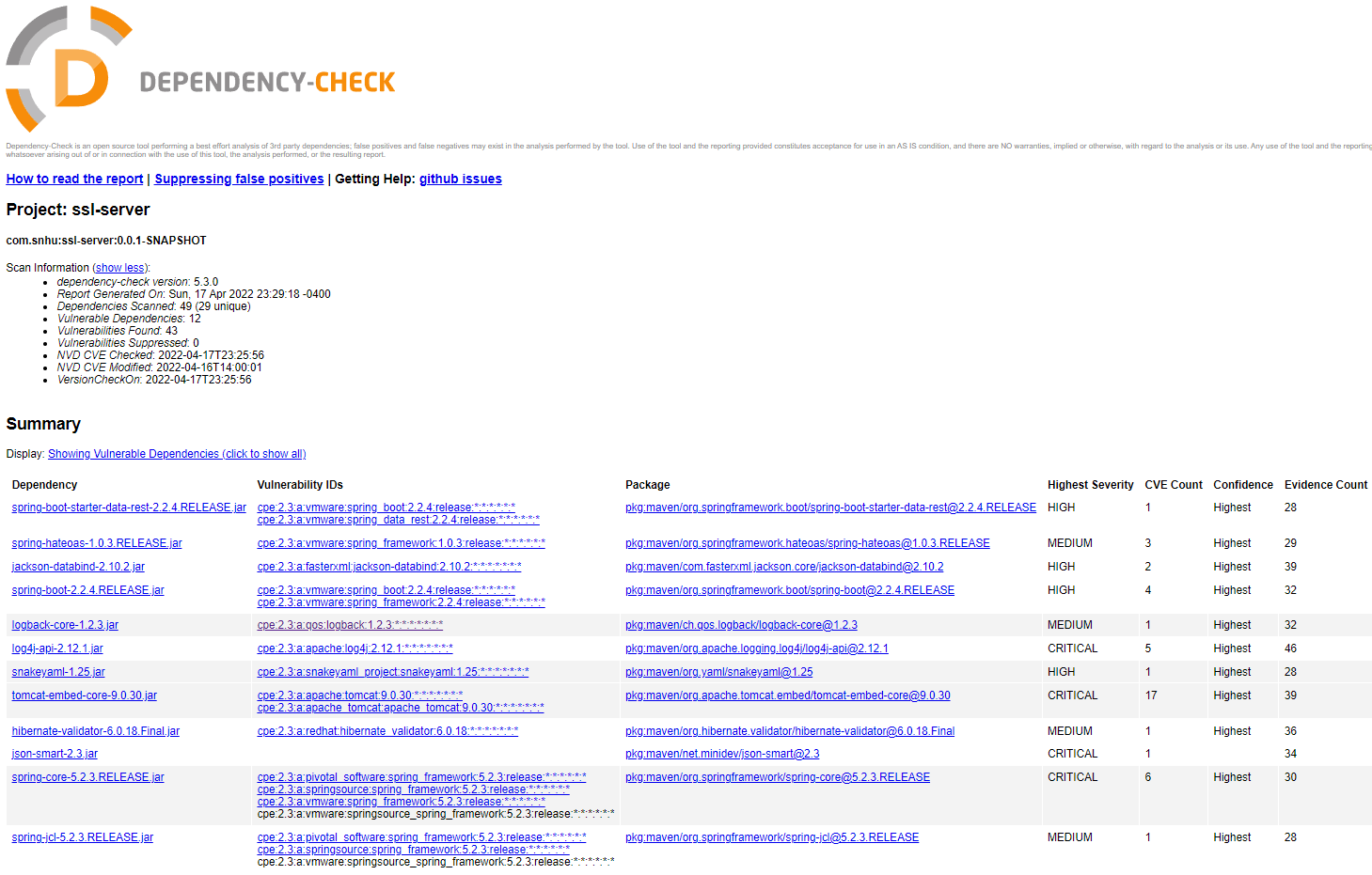
Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not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

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

Refactored Code:



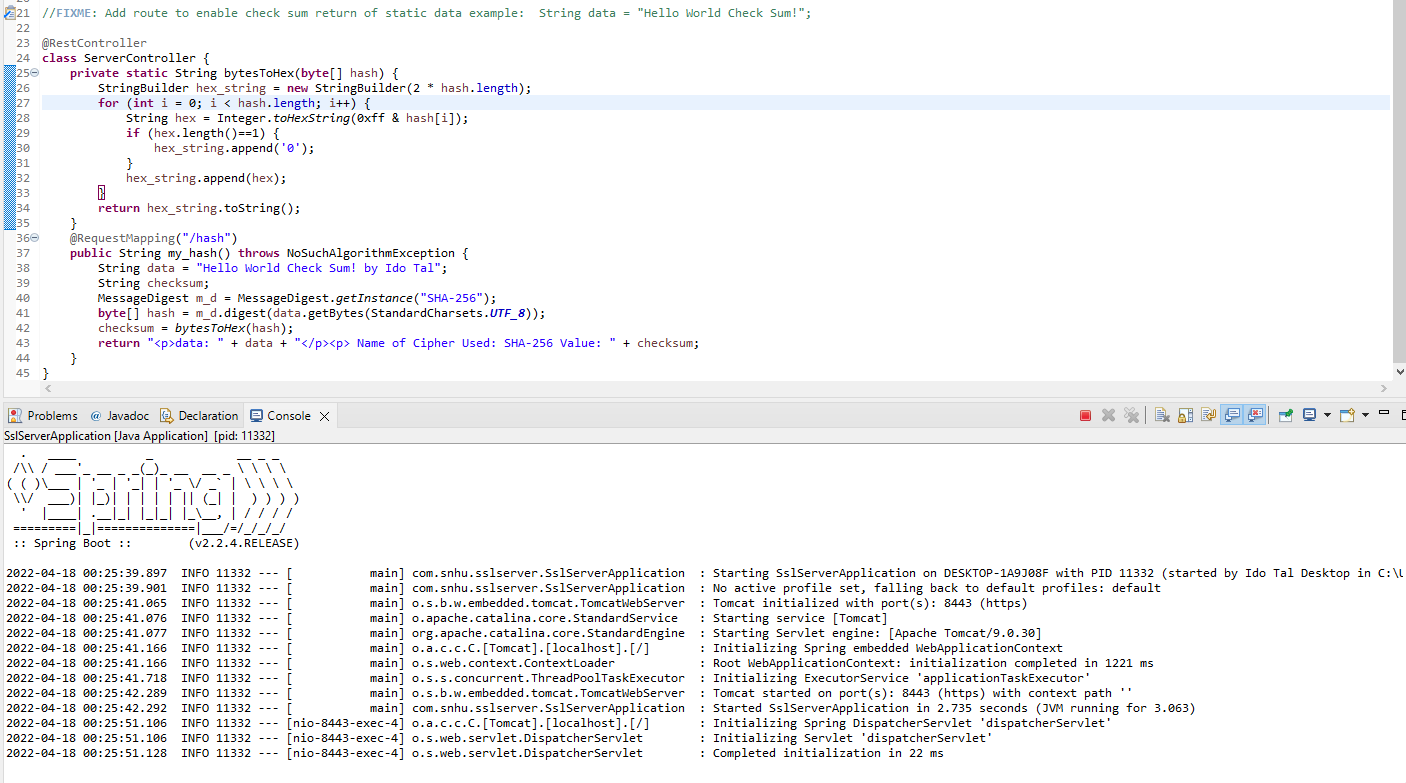
Dependency check report:



## 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.

Using the Vulnerability Assessment Process Flow Diagram there are sections that I covered in this project including cryptography, APIs, code error, Code Quality. Cryptography was covered with the use of an encryption that secures data from a string value. This only has an encryption, for this to be more useful there would also have to be a decryption that would be developed for another machine to understand what the data is. APIs was addressed through the use of the Spring boot framework. This can help in providing security from potential external threats from user requests. Code error was handled by implementing error handling such as “throw NoSuchAlgorithmException.” This helps prevent technical problems from arising or from being exploited. For code quality the best practices was used, and the code was written efficiently and in a simple way that makes it easy to understand how it works.

The layers of security that I added include the use of HTTPS and a decryption for sensitive data. This meant using the SHA-256 algorithm to encrypt the data, which among the best algorithm for decryption and is also used by the United States government for keeping its data secure.

For Artemis Financial, it is critical that they check their code and applications against vulnerability databases to catch potential problems before they arise. In addition it is important for them to keep updating their software to the latest versions that are available. Lastly, Artemis Financial should also work on suppressing false positives so as to prevent things that are non-issues from causing potential problems.