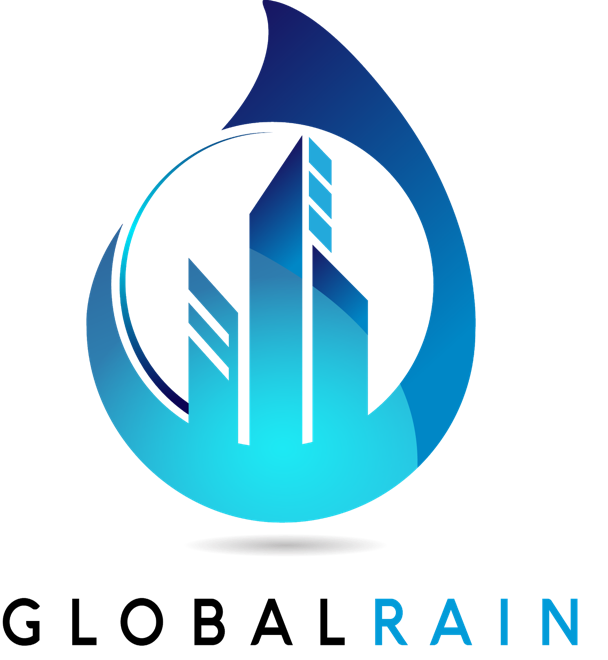
****

# CS 305 Project Two

**Practices for Secure Software Report**

**Table of Contents**

[Document Revision History 3](#_heading=h.1fob9te)

[Client 3](#_heading=h.3znysh7)

[Instructions 3](#_heading=h.2et92p0)

[Developer 4](#_heading=h.tyjcwt)

[1. Algorithm Cipher 4](#_heading=h.3dy6vkm)

[2. Certificate Generation 4](#_heading=h.1t3h5sf)

[3. Deploy Cipher 4](#_heading=h.4d34og8)

[4. Secure Communications 4](#_heading=h.2s8eyo1)

[5. Secondary Testing 4](#_heading=h.17dp8vu)

[6. Functional Testing 5](#_heading=h.3rdcrjn)

[7. Summary 5](#_heading=h.26in1rg)

## Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **1.0** | **2020-08-13** | **Camilo Hoyos** |  |

## 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

Camilo Hoyos

## 1. Algorithm Cipher

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

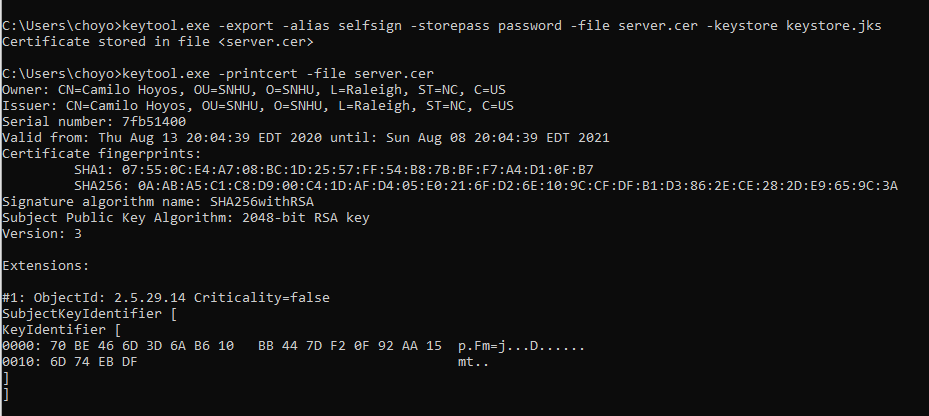
* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

The selected algorithm will be a SHA3-512 algorithm cipher. The hash function will take a message of any length and then scramble it using a hash that would decipher the original message. 512 is the bit level of the cipher and specifies that the message will be delivered in a 512 bit cipher. Since this is being used over the web, it needs to have a secondary level of security with a certificate for message signing. With this we will need non-symmetric keys. The public key will be the message signing key and the secret key will be responsible for the data decryption. One of the key attributes of this algorithm that is being used is that it belongs to an approved standard for federal information processing standards. Specifically it was released as part of FIPS PUB 202. This algorithm was also selected to be resistant to collision.

## 2. Certificate Generation

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

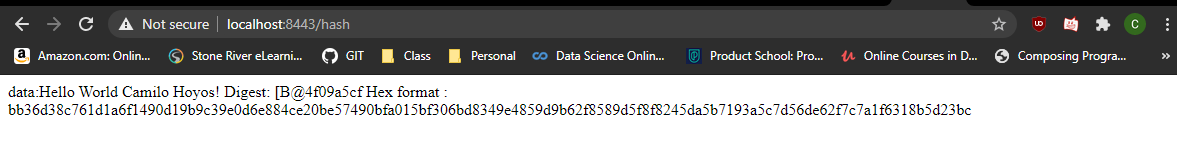
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

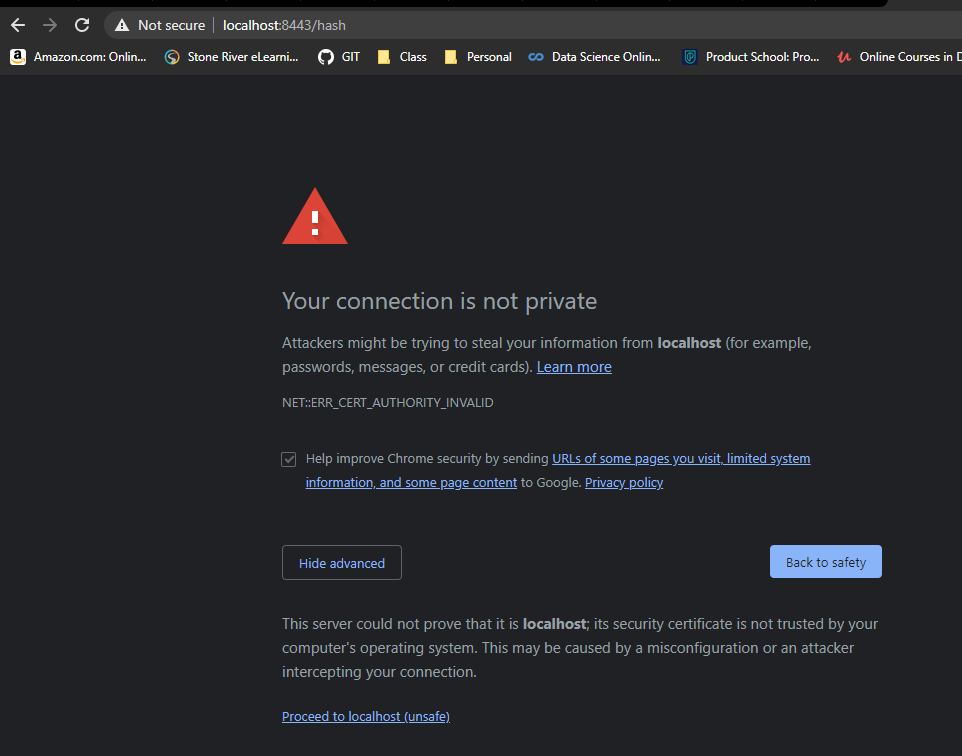
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

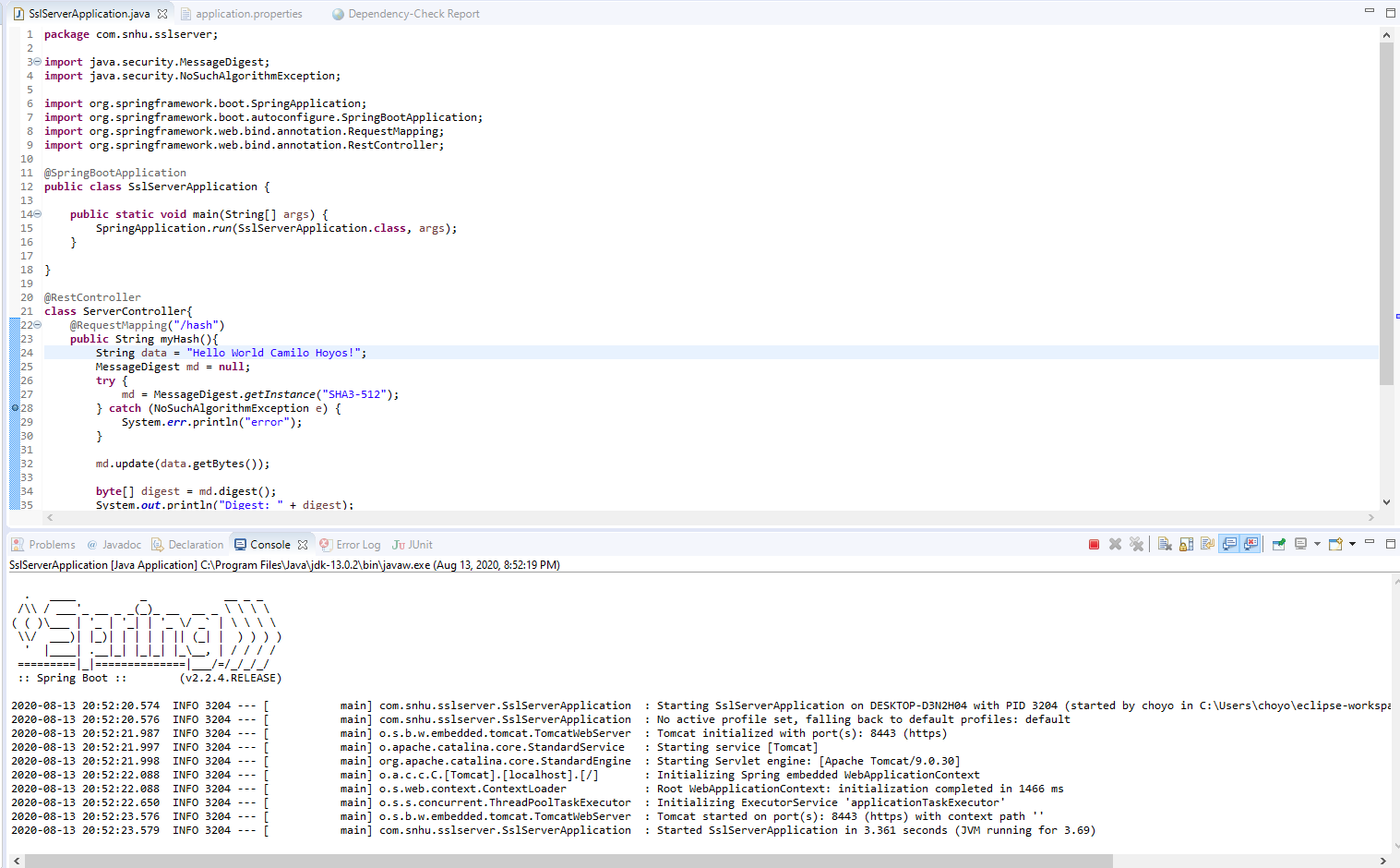


## 5. Secondary Testing

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.

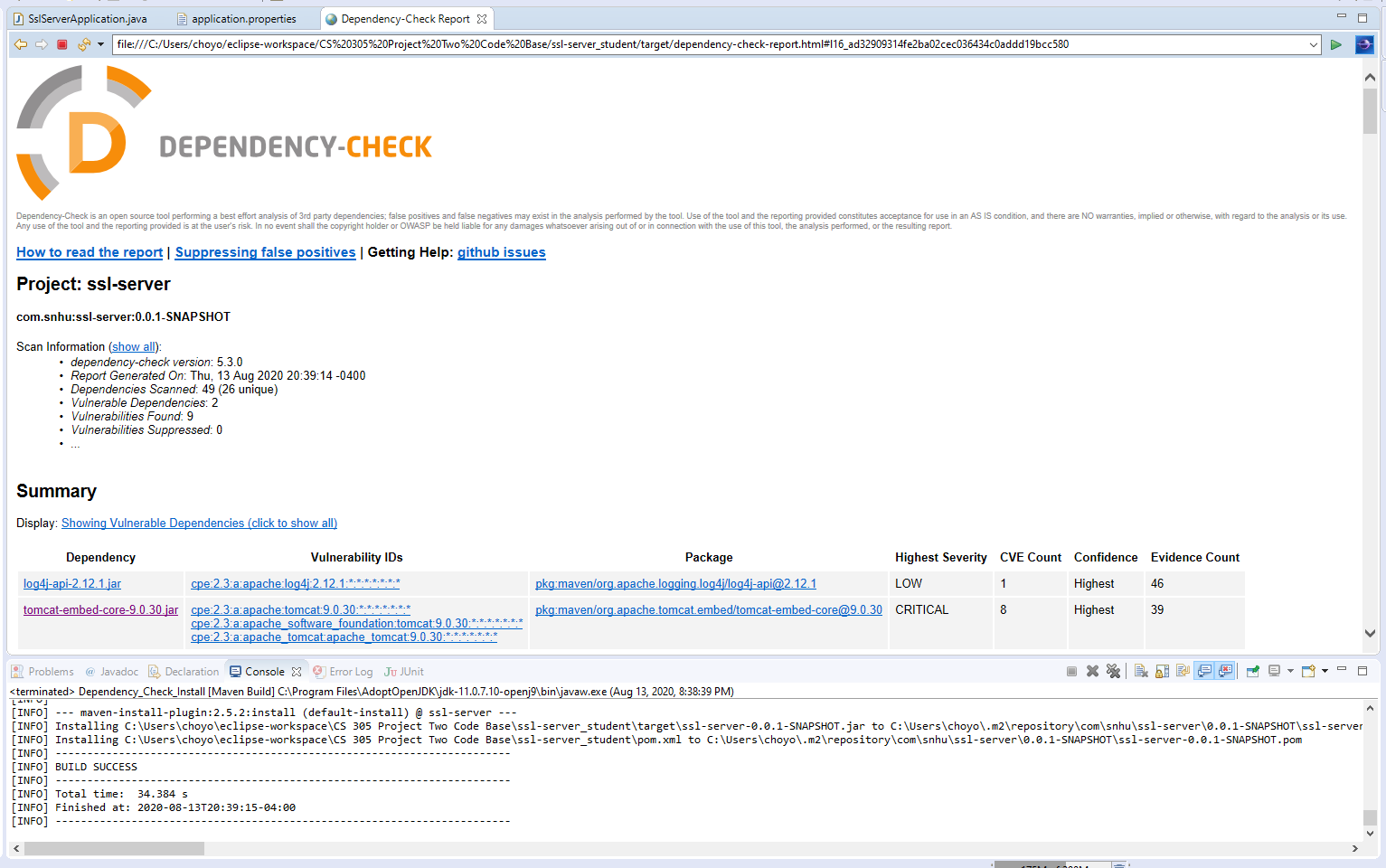
* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

Refactored code execution:



(Shows successful server start)

Dependency check report:



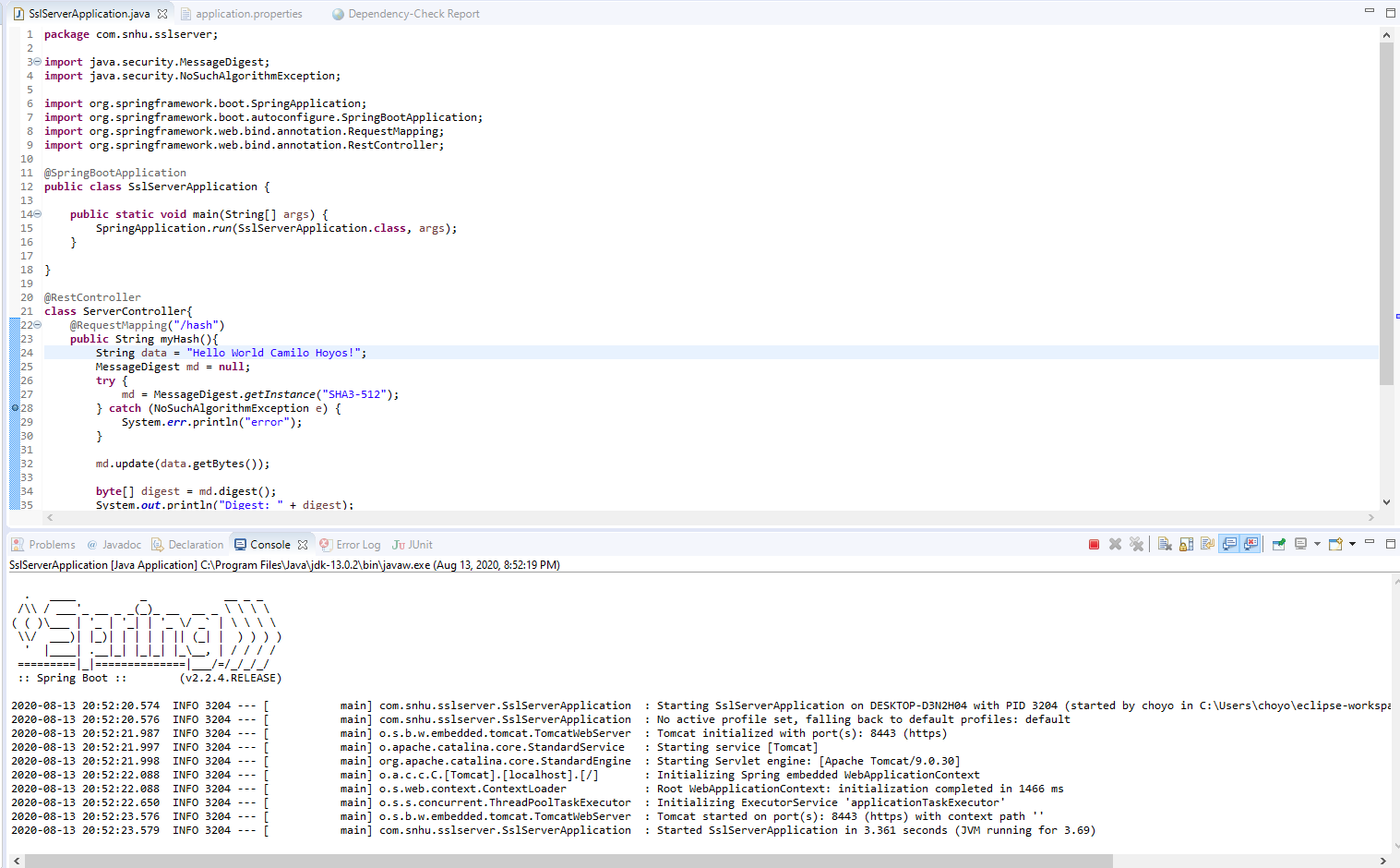
(Shows no changes in vulnerabilities)

## 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.

Refactored code execution:



(Shows successful server start)

## 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.

Areas addressed in vulnerability assessment diagram:

API: Since we’re using RESTFUL APIs these APIs ensure secure interactions. The only other items used are the message digest algorithms. the new URL /hash has no client interactions.

Cryptography: Refactored code utilizes SHA3-512 algorithm for message encryption.

Client/Server: Client and server interaction is secure and does not allow more access to the user than would be needed. There is also a certificate authority for signing to enable HTTPS.

Code Quality: Software is secure and validated using a dependency check.

The two major layers of security added is certificate authority using a self-signed certificate for website signing, and we included a hash algorithm to sign data as it enters the server.

Best practices include ensuring that cryptography meets government regulations as well as keeping these up to date with new findings. What is also a contributing factor for security is continuously using OWASP dependency checks to keep up to date with security vulnerabilities that need to be compensated for, or suppressed if they are false-positives.