****

# 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** | **06/19/2021** | **Christopher Holmes** |  |

## 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

Christopher Holmes

## 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.

Artemis Financial requested a secure cipher for long term storage. Based on the need for long term storage and the absence of mention of transferring data, then the best option would be the SHA-256 Hash. SHA-256 Hash provides 256-bit security which is the best security on basic java. This combined with the random number makes sure that the documents are secure. The files would be safely encrypted and if anyone stole the files, they would be useless without the decryption key. Brute forcing the files would be nearly impossible because of the security of the SHA-256 Hash.

## 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.

Texto

Descripción generada automáticamente

## 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.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

## 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.

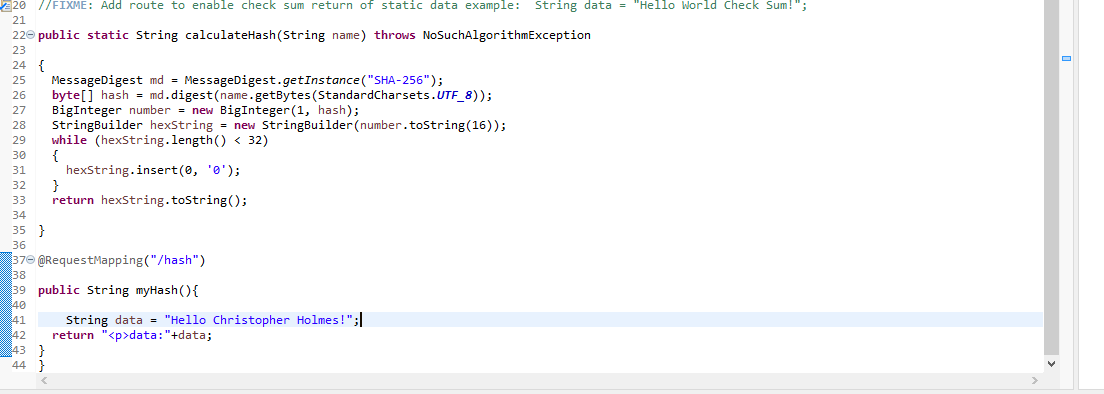
Interfaz de usuario gráfica, Aplicación, Word

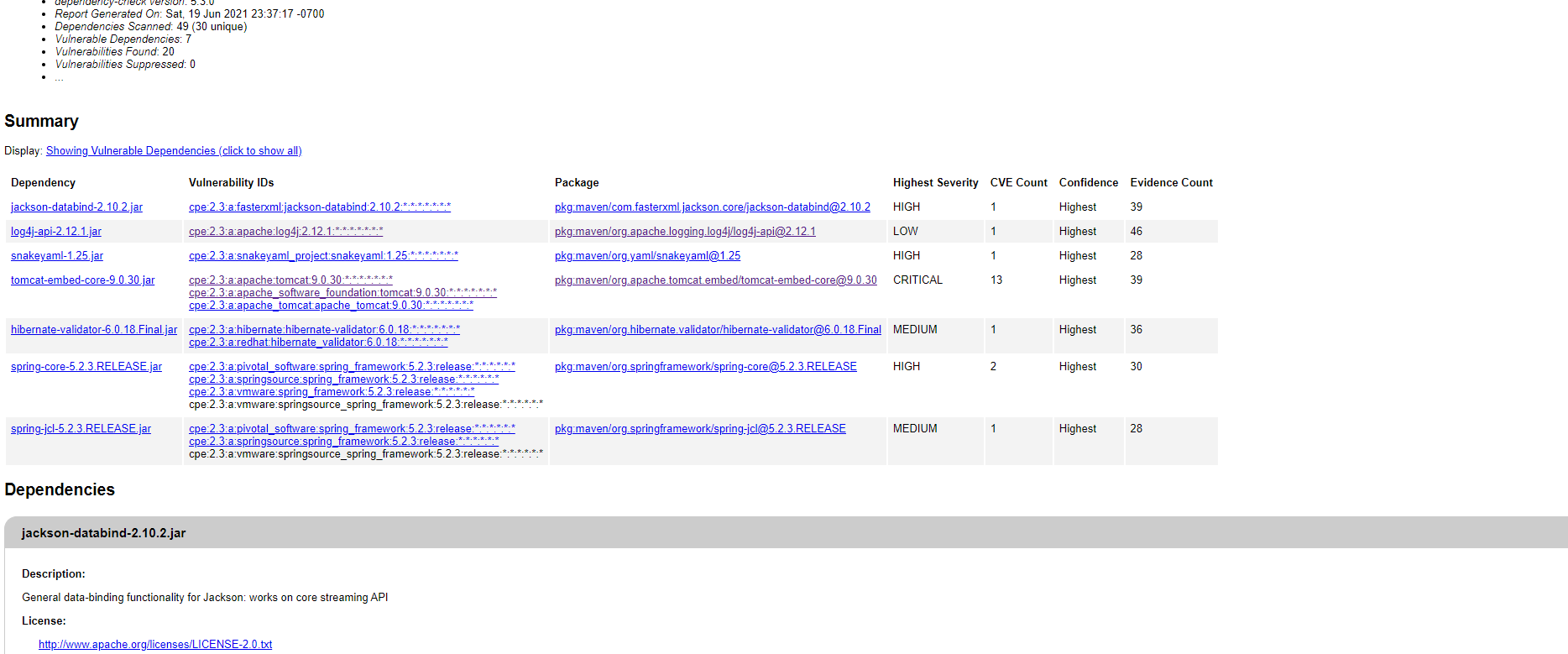
Descripción generada automáticamente

## 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

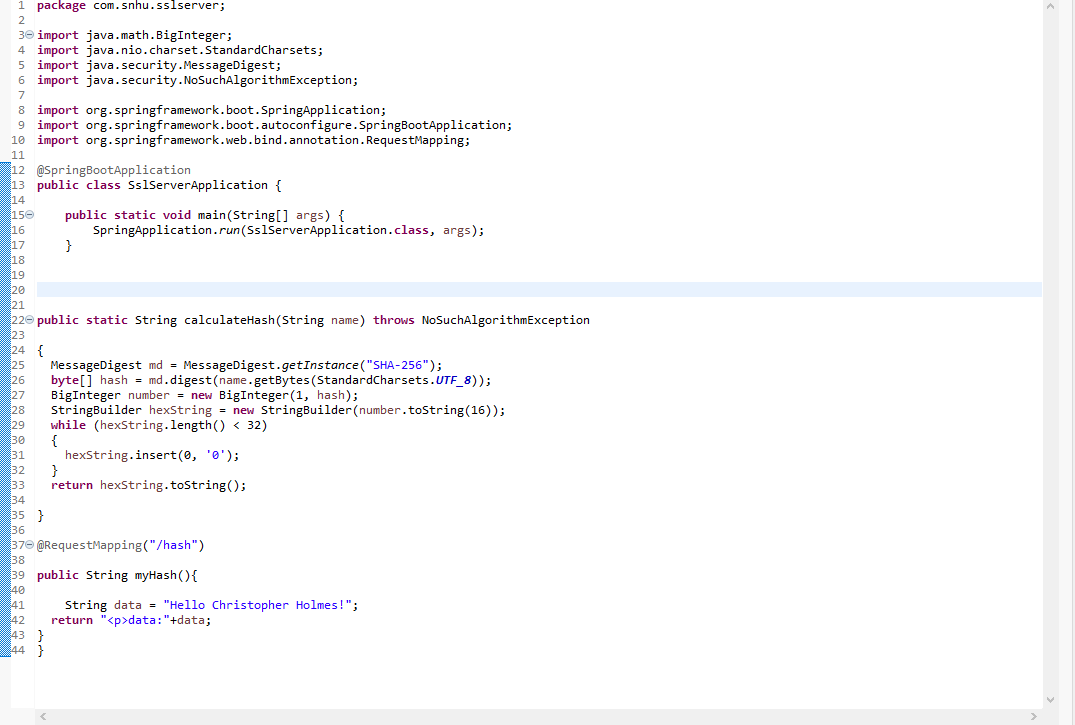




## 6. Functional Testing

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.

Refactoring the code, I solved the problems in the vulnerability assessment diagram. The hash function was added to provide cryptography to encrypt the data being sent and received. This combined with the certificate lets the user know that they are on the correct site and that their data is safe. This with the hash provides the secure client server connection. I made sure that the code was error free and kept the code as simple as possible. This makes for high quality, error free code. The part of the flow diagram that were covered were: Cryptography, Client/server, Code Error, Code Quality.

Adding different layers of security give protections against attacks. It takes multiple levels of security to cover every type of attack. When adding levels of security, I think of the risk of every type of attack and the most sensitive data based on the business needs. If a business is more likely to be attacked by DDOS then I should focus extra attention on making the denial-of-service attack impossible. This can be changed based on the business and what I consider as the most likely attack. I also run test for security to patch up any error of the code.

I have updated the maven test dependency to the latest version, 6.2.2. I have also added the hash cipher for the RESTful. The cipher I chose was SHA-256 which provides great security that cannot brute forced. To maintain security, I recommend that the maven dependency test to be run 2 times a month, at the beginning and middle of the month to make sure that the dependency security is up to date. The pom.xml file needs to be updated also when new updates are available.