****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/20/2022** | **Christian Rojas** |  |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Christian Rojas

## Algorithm Cipher

I would recommend using an SHA-256 encryption algorithm. Artemis Financial retains their client’s personal information, and since SHA can be used to encrypt plaintext files it is a great fit for Artemis Financials’ needs. SHA is an industry standard for hash functions. SHA can be used with little computing power; the function takes plaintext and transforms it into non-readable data. Once data is hashed, it provides a key. The provided key is the only way to decrypt the data.

Hash functions appear in most security applications. Hash functions are mathematical functions that take numerical input and convert it to a compressed numerical value. Hash functions always have a fixed length output regardless of the input.

Hash keys use random number generators to ensure reliability, since human input is much easier to predict. Random number generators are built by data that is acquired through small system deviations. A large database of numbers is important to reduce the possibility of duplicate, or reused keys.

Symmetric keys use the same key for both users, typically the user sends the key with the encrypted file. Non-symmetric keys are used when both users have separate keys, one for encrypting and the other for decrypting.

Ciphers have been historically used by militaries to send messages to each other without the contents of the message being intercepted by the opposing militaries. Early ciphers were typically basic, comprised of letter parings and following patterns. Ciphers became more complex when computers got involved. Today, we use ciphers like AES, and Two-factor authentication.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Graphical user interface, text, application

Description automatically generated

## Secure Communications

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

Graphical user interface, text, application

Description automatically generated  
HTTPS is working.

Graphical user interface, text, application, email

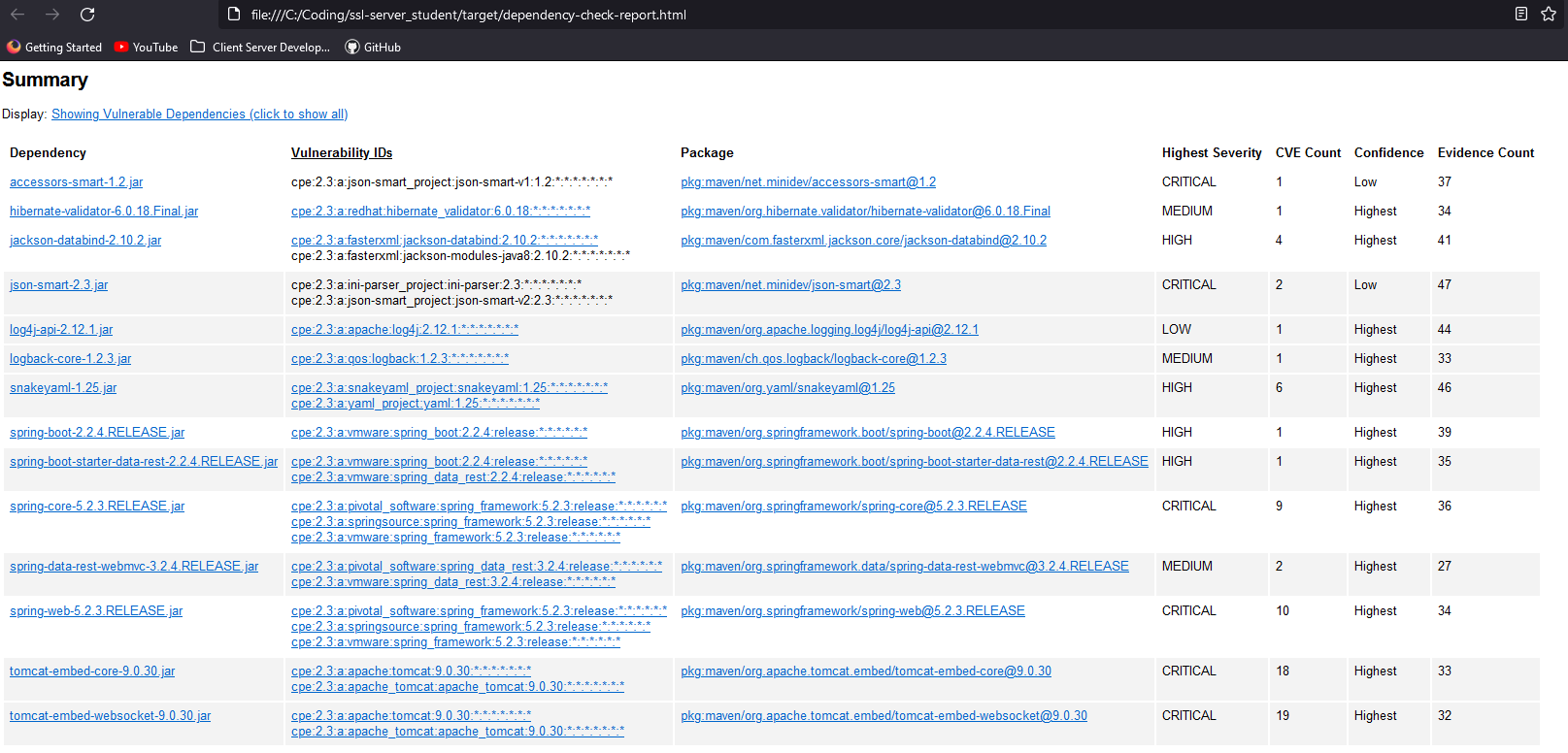
Description automatically generated

## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

Text

Description automatically generated



## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

Text

Description automatically generated

## Summary

The code was refactored in the following ways:

* SslServerApplication.java: This file was refactored by adding the SHA-256 hash function and a secure RestController. The ServerController class functions to work out the issues presented in the vulnerability check. The hashResult function returns the hash function results in the web browser.
* Pom.xml: I added the tomcat dependency and updated the maven version to 7.3. I would recommend checking on this file every 3 months to ensure software versions are still up to date.
* Application.properties: I altered the port number, added the keystore location, password, and alias.

I initially ran into some issues getting the system to compile. I checked all my systems and verified the code looked correct. After some research, I found that SpringBoot is very sensitive to directory names and my code directory had a space in it. After fixing this and rebuilding the application, I was able to get it running.

## Industry Standard Best Practices

I followed industry standards by observing secure coding practices by using private strings to reduce the possibility of data leakage. By implementing an industry standard cipher and running tests against the applications vulnerabilities, I was able to ensure the application is secure.

It is imperative that secure industry standards be observed due to the cost of losing sensitive data. With the value of data and the rising risks of losing that data, the company must be able to feel confident that their applications will not lie vulnerable to attacks.