****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.30j0zll)

[**Client 3**](#_heading=h.1fob9te)

[**Instructions 3**](#_heading=h.3znysh7)

[**Developer 4**](#_heading=h.2et92p0)

[**1. Algorithm Cipher 4**](#_heading=h.tyjcwt)

[**2. Certificate Generation 4**](#_heading=h.3dy6vkm)

[**3. Deploy Cipher 4**](#_heading=h.1t3h5sf)

[**4. Secure Communications 4**](#_heading=h.4d34og8)

[**5. Secondary Testing 4**](#_heading=h.2s8eyo1)

[**6. Functional Testing 4**](#_heading=h.17dp8vu)

[**7. Summary 4**](#_heading=h.3rdcrjn)

[**8. Industry Standard Best Practices 4**](#_heading=h.26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **2/21/24** | **Jason Restucci** |  |

## Client



## Instructions

Submit these completed practices for a 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

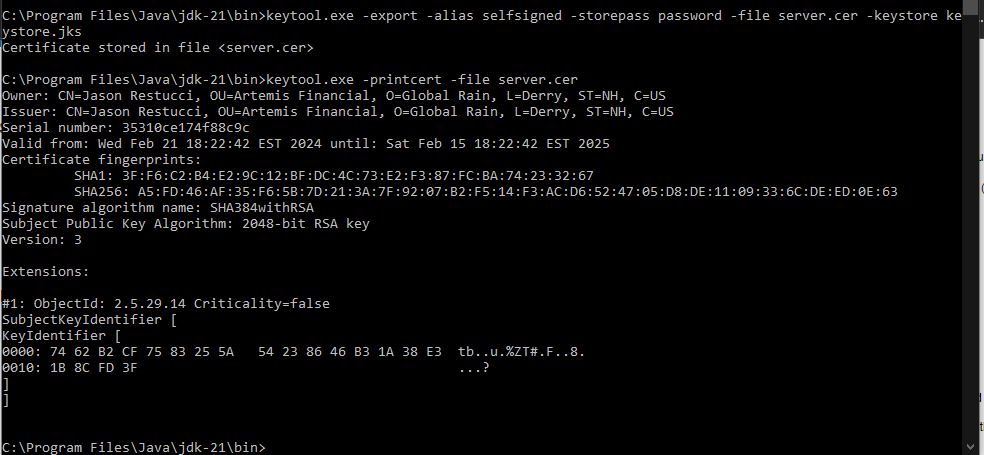
Jason Restucci

## Algorithm Cipher

Artemis Financial is requesting a recommendation for an encryption algorithm cipher to verify files in the form of a checksum when the web application is transferring data. I recommend SHA-256. It was developed by the NSA and NIST and is widely used around the globe. It produces a 256-bit hash value from input data of any length. In our circumstance, it will take the data to create a sequence of random numbers and letters that can be used to verify the data that is received is identical to the original. This is what is called a checksum, and what Artemis Financial wants us to implement. 256 bits is the length of the hash value, the larger the value, the more possible key combinations. This makes it harder to brute force your way through the encryption. Checksums do not use keys, as they are not meant to be decrypted, just to validate the data’s integrity. Regular encryption uses keys to decrypt messages or data after it has been sent. If a key is symmetrical it means that the same key is used to encrypt and decrypt. If a key is asymmetrical it means that there is a different key to encrypt and decrypt.

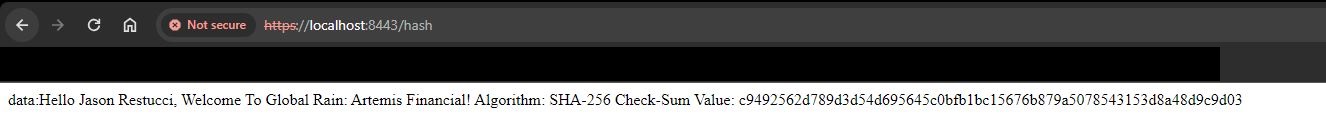
## Certificate Generation

Insert a screenshot below of the CER file.



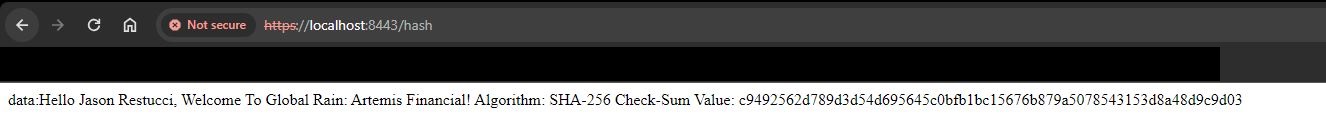
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

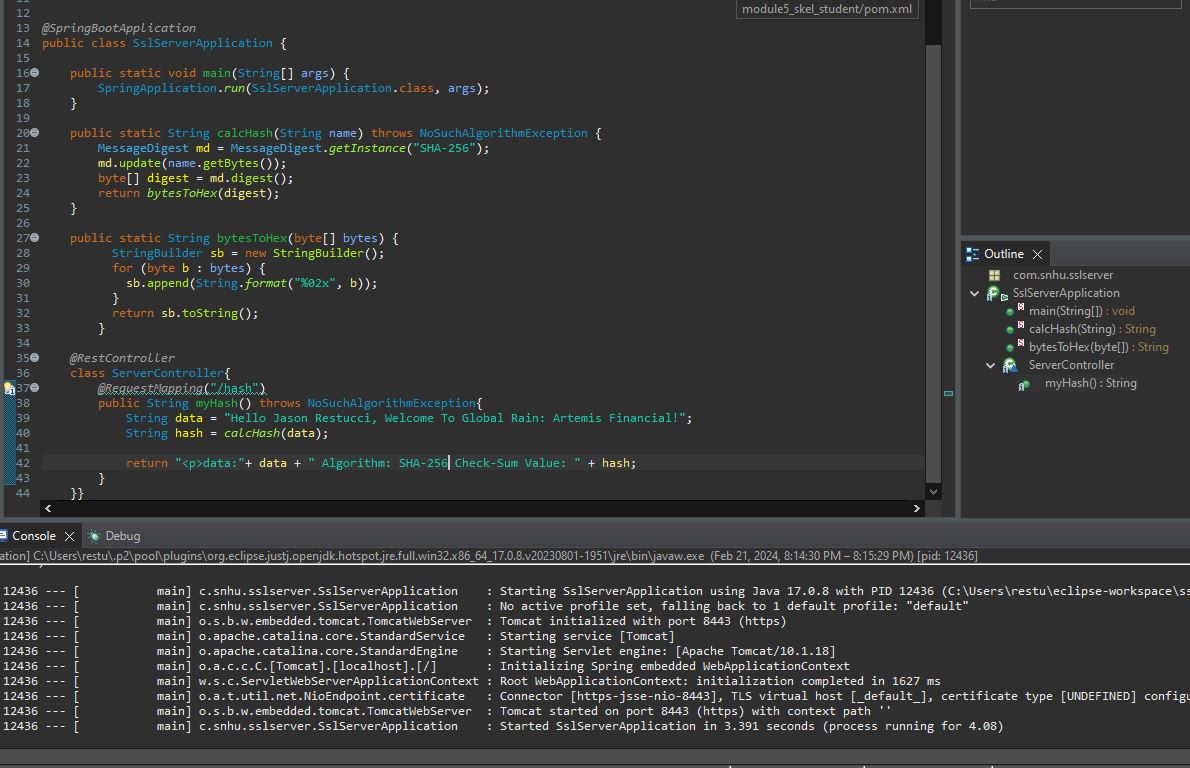
Insert a screenshot below of the web browser that shows a secure webpage.



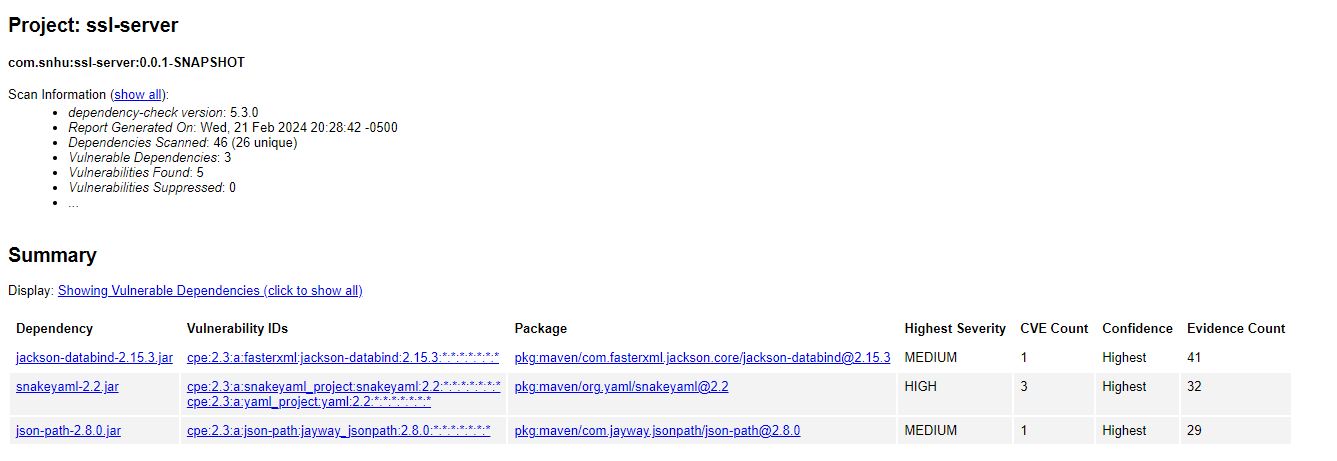
The web browser above says not secure because it is a self-signed certificate but as you can see by the URL it is using HTTPS.

## Secondary Testing

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



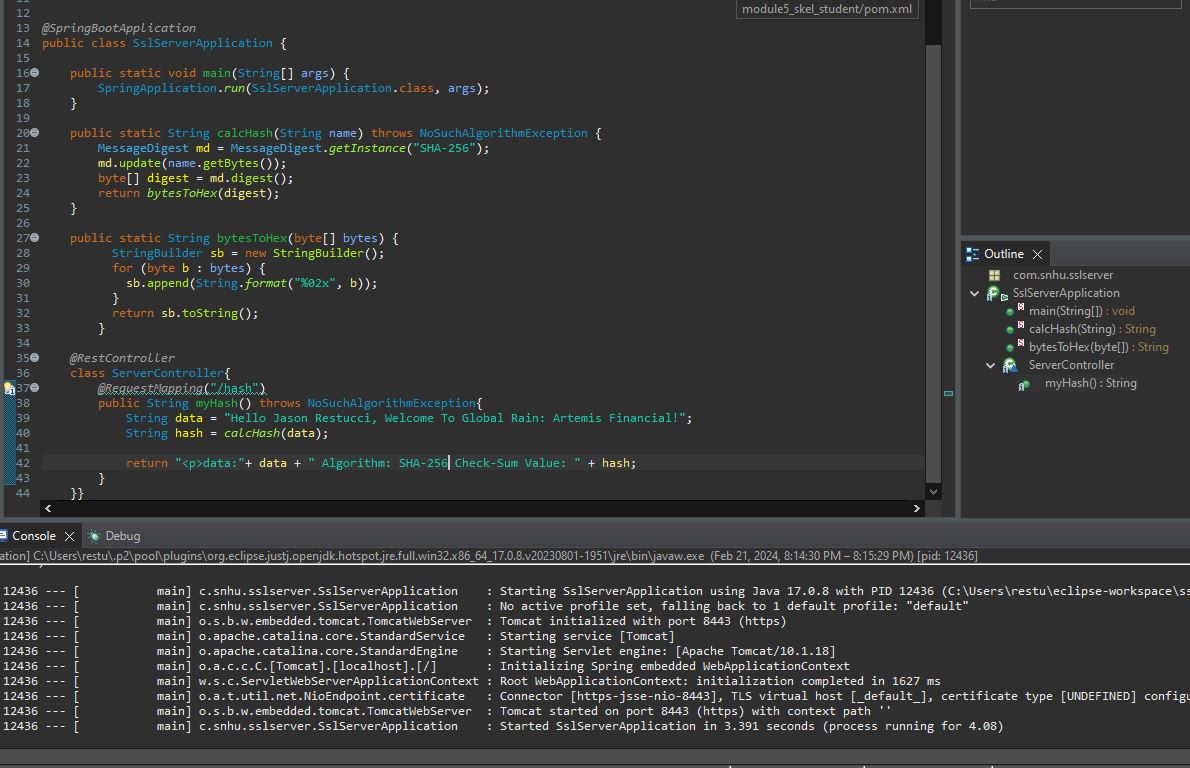
Above is the refactored code and the console showing it ran successfully with no errors.



Above is the Dependency Check

## Functional Testing

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



## Summary

During code refactoring, I added the Server Controller to act as the controller to the RESTful endpoint. I also added a function to calculate the hash and one that converts bytes to hexadecimal. The hash is using SHA-256 to ensure maximum security. All functions and classes adhere to and follow the Vulnerability Assessment Diagram.

## Industry Standard Best Practices

I have followed the industry standard best practices by manually reviewing my code and all vulnerable areas listed in the Vulnerability Assessment Diagram and keeping code quality in mind while I am coding. I have also run a dependency check to ensure my dependencies are secure as well. It is so vital to follow industry standard best practices and catch a possible security risk before it happens. This is for the well-being of the user and the company both. If the user suffers an attack because they used your application, the company can be held liable. It’s in everyone's best interest to stop these security risks early, before release.