****

# 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** | **Oct 20th, 2024** | **Devante Crenshaw** |  |

## 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

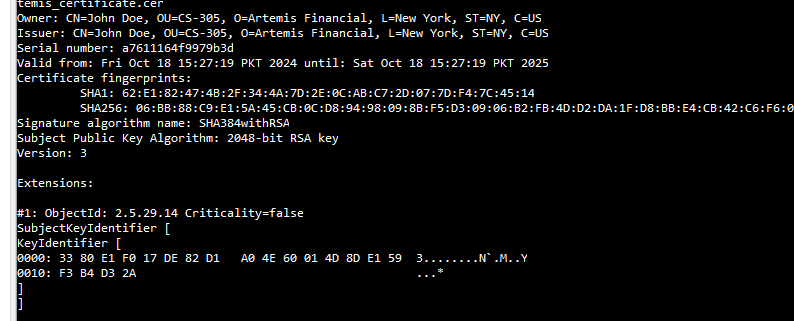
Jhon Doe

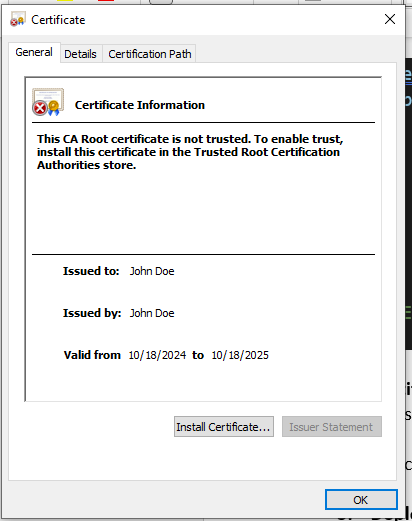
## Algorithm Cipher

Artemis Financial is requesting stronger security for their web application to ensure secure communication. Since a likely threat to a financial institution is an attacker trying to access sensitive data for financial gain, encryption would be the best solution. This would make any stolen files unreadable without the correct key. For secure communication, I recommend Asymmetric encryption is what I suggest for secure communication, here the key to encrypt the data is public but the key to decrypt is private. Because the presented information is rather confidential, it is quite reasonable to apply the SHA-256 cipher algorithm resulting from 256-bit keys. This helps in affording high encryption with many possible combinations of keys. SHA-256 also integrates java’s random number generator to generate a nonce unaltered signature, which genuinely confirms the file authenticity. The hash function would take a SHA-256 cipher to ensure the messages exchanged are as secure as possible.

## Certificate Generation

Insert a screenshot below of the CER file.

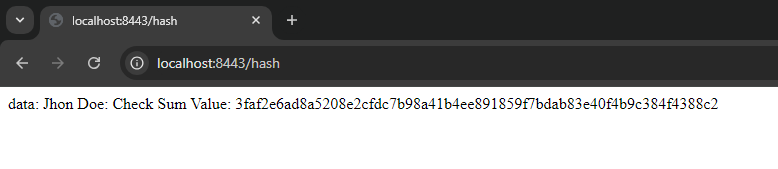


 A screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

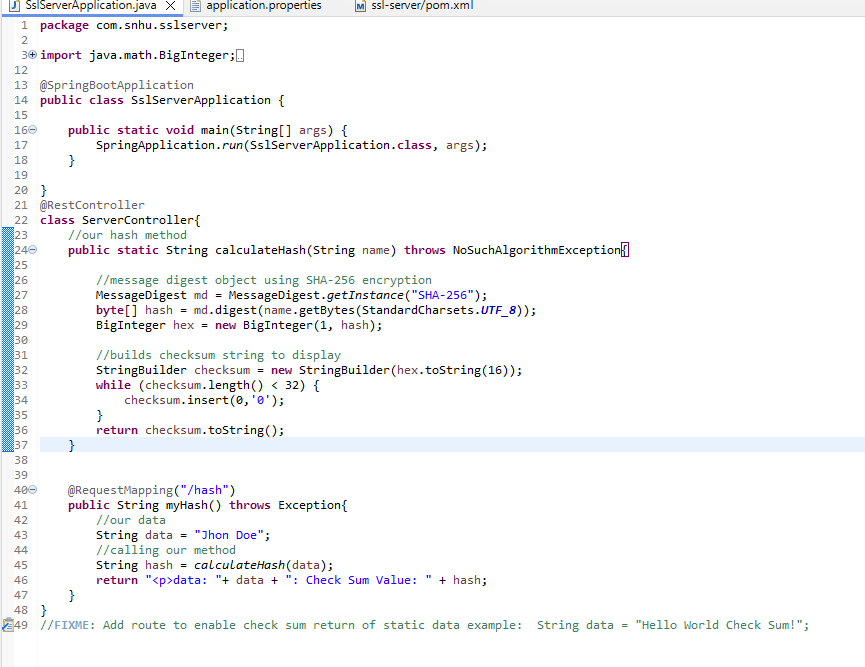
Insert a screenshot below of the web browser that shows a secure webpage.

A screenshot of a computer

Description automatically generated

## Secondary Testing

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

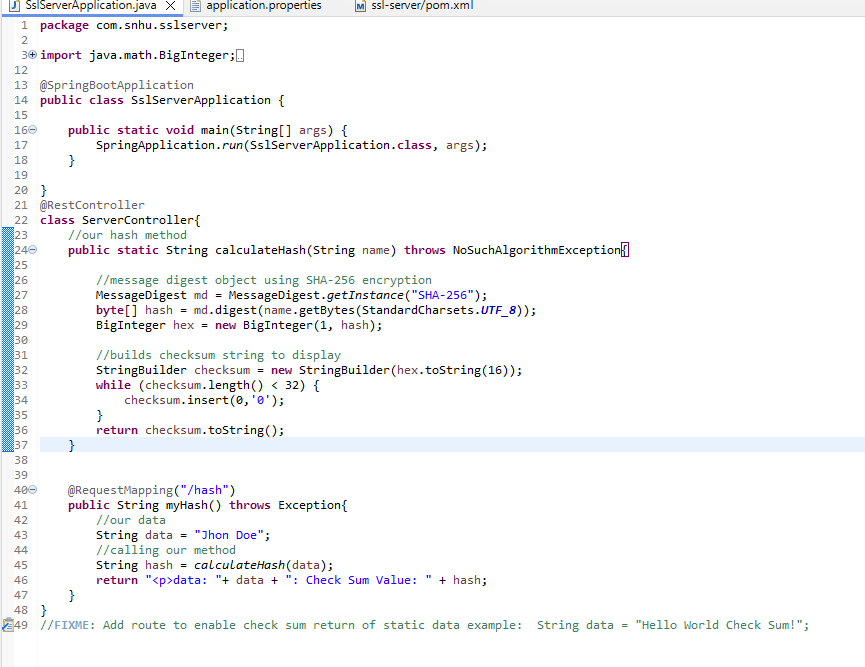


A screenshot of a computer

Description automatically generated

## Functional Testing

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



## Summary

To address the issues from the vulnerability assessment, I refactored the code by adding a secured Rest Controller which handles the program’s hash RESTful endpoint. It used the SHA-256 hashing cipher because it is strong security and provides low collision risk.

To keep things secure I advise you to run dependency checks monthly to ensure that you are up to date with vulnerabilities and updates. Updating the plugins in the pom.xml will keep the application using the most recent security tools and protect the company and its sensitive data.

## Industry Standard Best Practices

To ensure secure coding and protect sensitive data, I applied industry-standard best practices throughout the development process. This includes the use of SHA-256 for secure hashing, which provides a high level of security with minimal risk of collisions.

I also implemented a secured RestController to protect RESTful endpoints and applied regular dependency checks to identify vulnerabilities in third-party libraries. Additionally, keeping the pom.xml plugins up to date ensures the project benefits from the latest security patches and improvements.

These practices help maintain the overall security of the application and protect Artemis Financial’s sensitive data while aligning with the best practices recommended by OWASP and other security frameworks.