****

# 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** | **08/18/2024** | **Jacob Englund** | **Initial Document Creation** |

## Client



## Developer

Jacob Englund

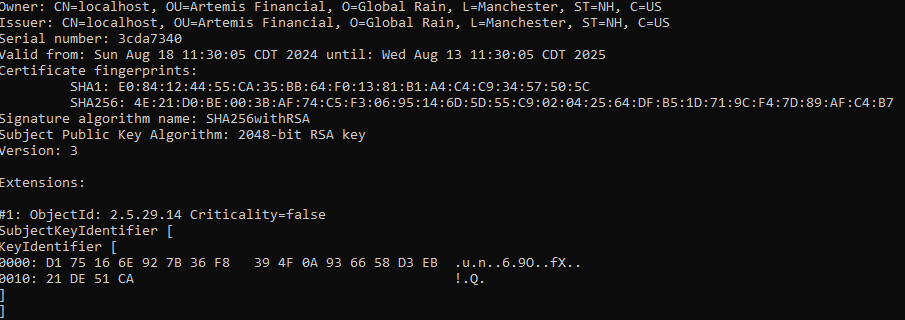
## Algorithm Cipher

To keep Artemis Financial's data safe, I suggest using the SHA-256 hashing algorithm. SHA-256 is a well-known method that helps ensure data has not been changed and is often used in digital signatures and certificates. Instead of encrypting data like other algorithms, SHA-256 turns data into a fixed 256-bit hash value, making it perfect for checking if the data has been tampered with. When used alongside encryption methods, SHA-256 creates a unique checksum for each file, ensuring that it stays secure during transfers. The strength of SHA-256 comes from its ability to prevent different inputs from creating the same hash value, making it impossible to break. Even though hashing methods have improved over time, SHA-256 is still one of the most trusted and secure options available. By using SHA-256, Artemis Financial can make sure its client data is protected and that their software security is up to date.

## Certificate Generation

Insert a screenshot below of the CER file.

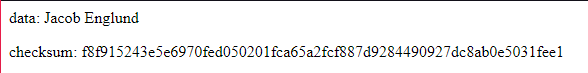
CER File:



## Deploy Cipher

Insert a screenshot below of the checksum verification.

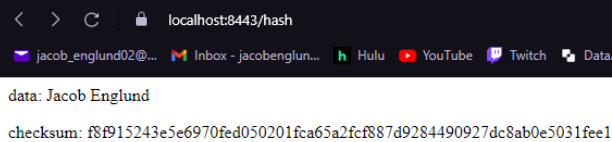
Checksum Verification:



## Secure Communications

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

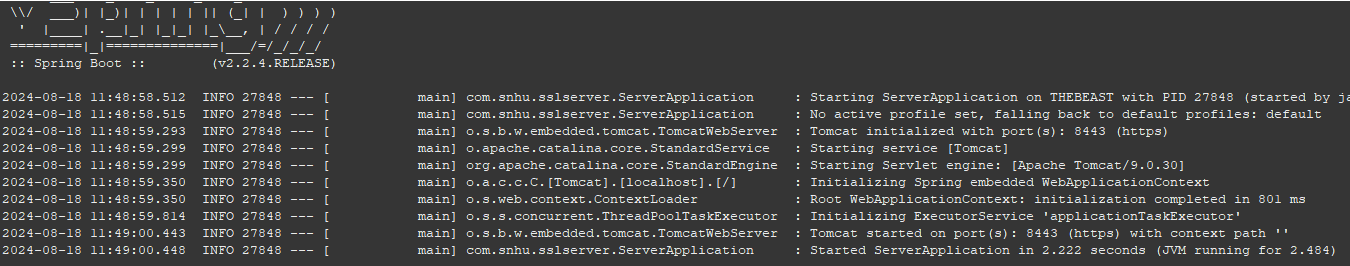
Secure Webpage:



## Secondary Testing

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

Refactored Code:



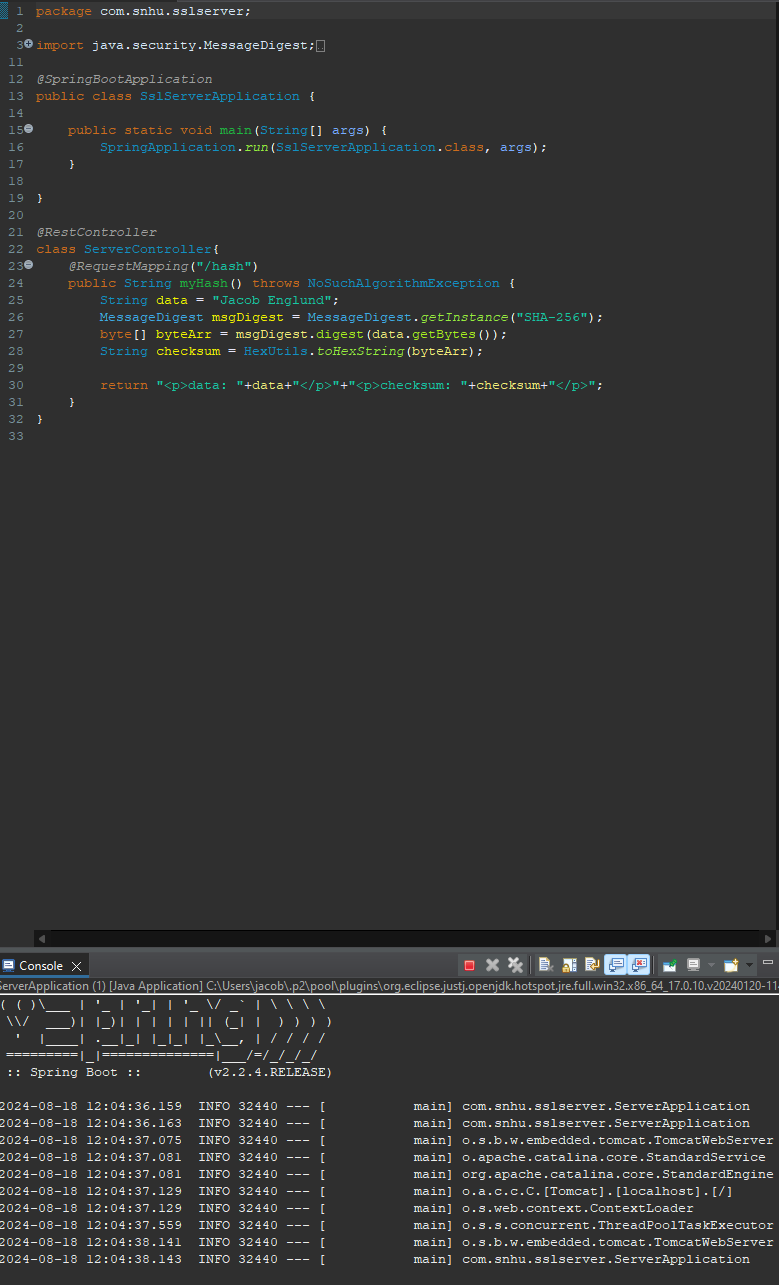
Dependency Check:



## Functional Testing

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

Refactored Code:



## Summary

To enhance the website's security, I added SHA-256 encryption. SHA-256 is a secure hashing algorithm that generates a 256-bit hash to keep data safe from unauthorized access. Even though the website does not store data, I used SHA-256 to ensure that any sensitive information processed or transmitted is protected. By converting this information into a secure hash, it becomes impossible for attackers to reverse-engineer and access the original data. This extra layer of security helps prevent potential tampering or interception during data exchanges. I also keep everything updated to address any new security issues, making sure the website remains protected and reliable.

## Industry Standard Best Practices

Even though the website does not store any data, I used SHA-256 encryption to keep any data that is processed or transmitted secure. SHA-256 creates a strong hash that is hard to reverse, so it helps protect against tampering or interception. I also followed best practices for security by using HTTPS to encrypt data sent over the internet and being careful with error handling to avoid exposing any sensitive information. Even though there is no data storage or user inputs, I made sure to keep everything updated with the latest security fixes and used tools to check for any potential issues in the code. These steps help the company stay secure, follow industry standards, keep a good reputation, and avoid expensive security problems.