****

# 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** | **2/18/2023** | **Lars Dela Cruz** |  |

## 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

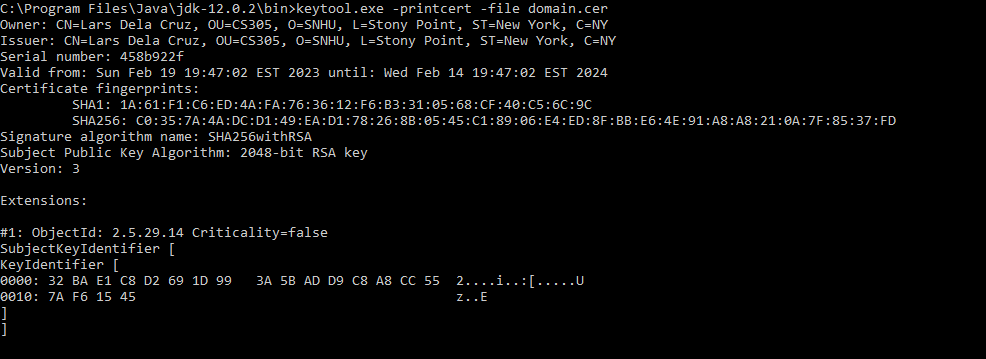
Lars Dela Cruz

## Algorithm Cipher

Artemis Financial is in need of security that will secure communications for their web application that holds sensitive banking information. With the use of SHA-256 as our encryption system, this cipher will be the most valuable in protecting the information from any users who are not allowed to. This asymmetrical communication in which the “key” to decrypt is private but public when encrypting is a powerful process that is bulletproof to brute force tactics. Hash functions are used to convert inputs to a compressed value and the length of that value is determined by the bit levels. In this case it's 256 bits. This particular security system was developed by the NSA themselves, its interesting history has garnered a reputation for the system to be almost impossible to breach.

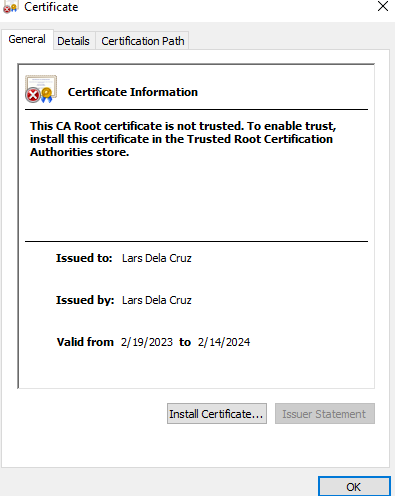
## Certificate Generation

Insert a screenshot below of the CER file.



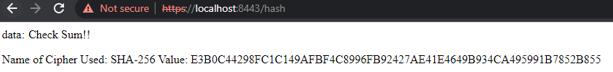
## Deploy Cipher

Insert a screenshot below of the checksum verification.



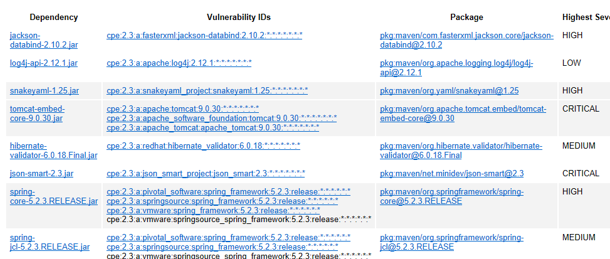
## Secure Communications

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



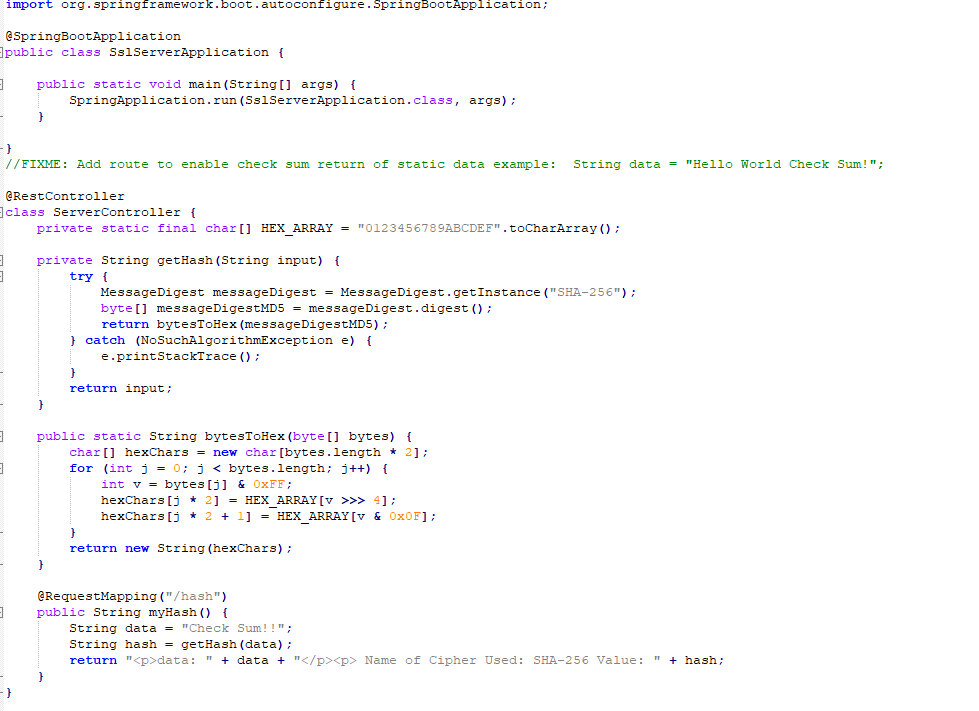
## Secondary Testing

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



## Functional Testing

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



## Summary

When refactoring the given code I first added a RestController in sslServiceApplication to act as RESTful stop. RequestMapping ensures the program runs when the hash is applied. SHA-256 seemed to be an obvious choice since its security and to minimize any change for collisions. SHA-256 may seem over-kill to some but for a banking firm that holds highly sensitive information, I do not think so. For this system to stand the test of time, dependency checks and any other updates to the server should be maintained in order for security to be consistent throughout Artemis Financial’s life span.

## Industry Standard Best Practices

As mentioned above, dependency checks and security updates need to be patched regularly in order for the initial insurance of security that's been applied, to live on. That being said, employee training should be a standard practice since it is up to them to maintain the updates but also any type of phishing attacks that require a real person to prevent.