****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_c7mvuc68dgvm)

[**Client 3**](#_t33m429ga2xe)

[**Instructions 3**](#_ys93gdwtw5b)

[**Developer 4**](#_99xi5u79mfsr)

[**1. Algorithm Cipher 4**](#_z8e3v7mfuan2)

[**2. Certificate Generation 4**](#_3wfsl5p6r2pm)

[**3. Deploy Cipher 4**](#_6f0f2jhpx0v9)

[**4. Secure Communications 4**](#_ink6f7ilyhvt)

[**5. Secondary Testing 4**](#_qsenpsyzf57m)

[**6. Functional Testing 4**](#_kmxcbuogdr2r)

[**7. Summary 4**](#_jtw43y3sy2cp)

[**8. Industry Standard Best Practices 4**](#_j59r6j9yy8xu)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **6/22/2025** | **Thomas Comer** |  |

## 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

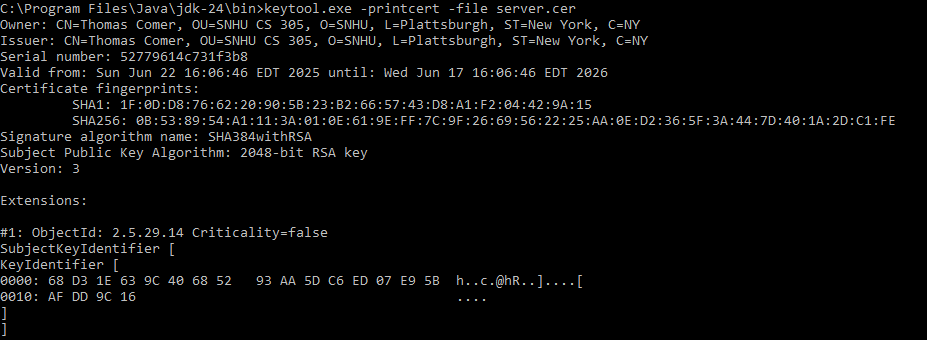
Thomas Comer

## Algorithm Cipher

Artemis Financial needs a strong encryption algorithm in order to handle financial data from clients all over the world. For this purpose, I recommend utilizing the Advanced Encryption Standard (AES) using the Secure Hash Algorithm (SHA) 256-bit encryption cipher. AES-256 works by using a pair of a public key, and a secured private key. There are 2 standards for the type of encryption key use, those being symmetric and asymmetric keys, with symmetric keys being used for both encryption and decryption, while asymmetric keys use a different set of keys for both steps. Asymmetric keys are more secure, but can take longer to compute, so it can sometimes not be worth utilizing. The best way to ensure that these keys are secure is to generate them using a truly random generation scheme, as to ensure that there is no pattern that can be discovered by an attacker. It then encrypts the input data into a 256-bit hash, which is then able to be freely used without fear of an unauthorized individual gaining access to the unencrypted data. AES-256 has been tested as one of the most robust encryption algorithms ever designed. Designed for use by the United States Government in 2001, it was created to ensure that private state data would be kept confidential in the incoming information age. The algorithm has a very large output size so it is very safe from collisions, as well being nigh impossible to crack, with experts saying the heat death of the universe would happen before a successful brute force attack on encrypted data. In general, AES is the best choice for Artemis Finanicial’s needs, and is the best choice for encrypting their client’s data.

## Certificate Generation

Insert a screenshot below of the CER file.



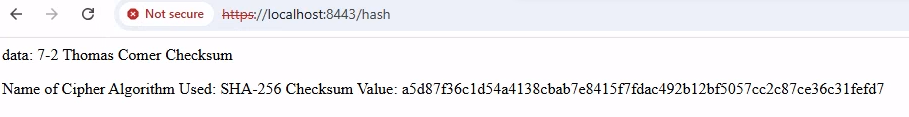
## Deploy Cipher

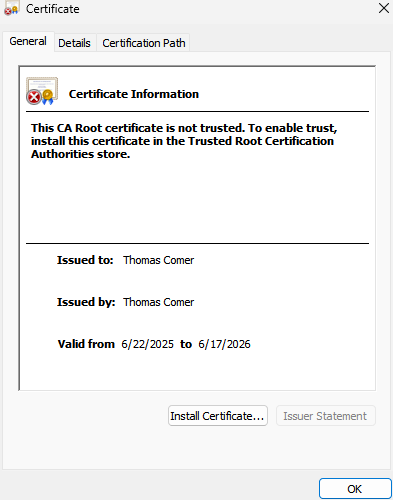
Insert a screenshot below of the checksum verification.



## Secure Communications

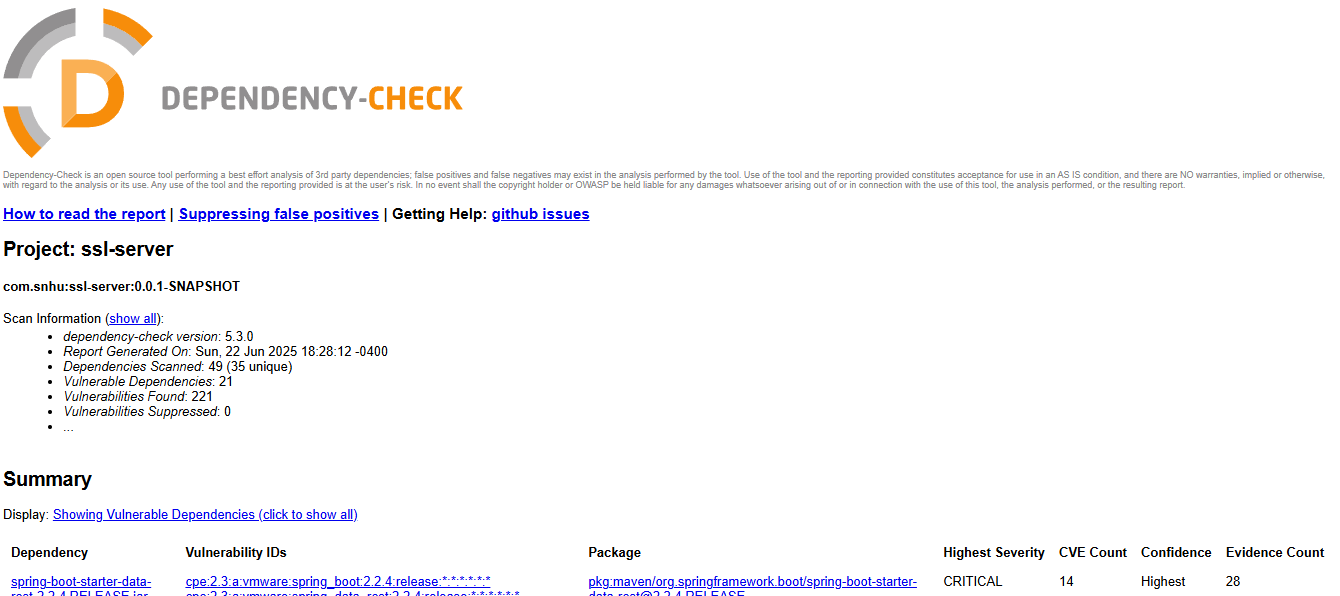
Insert a screenshot below of the web browser that shows a secure webpage.

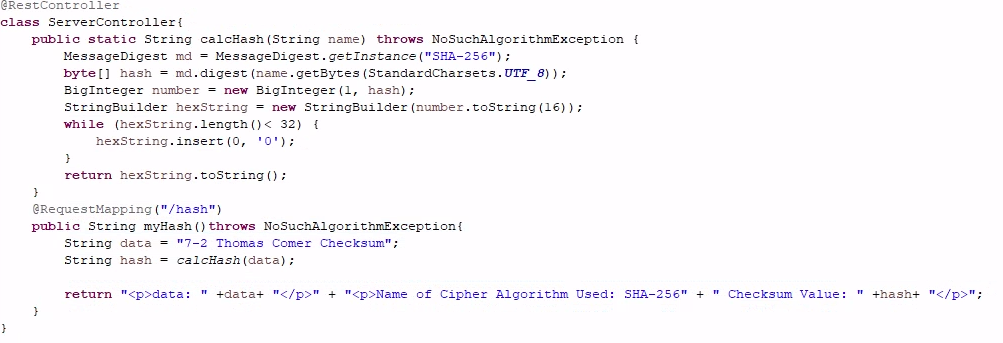


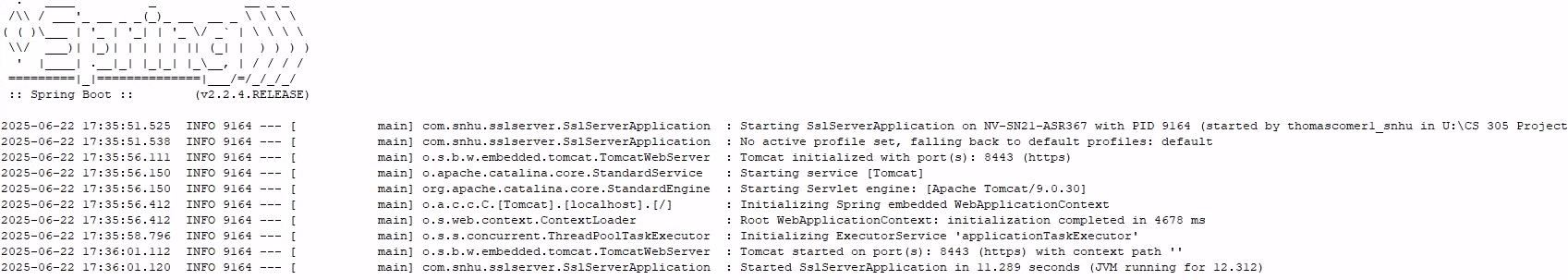
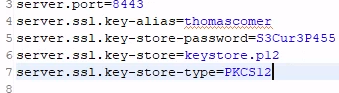


## Secondary Testing

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

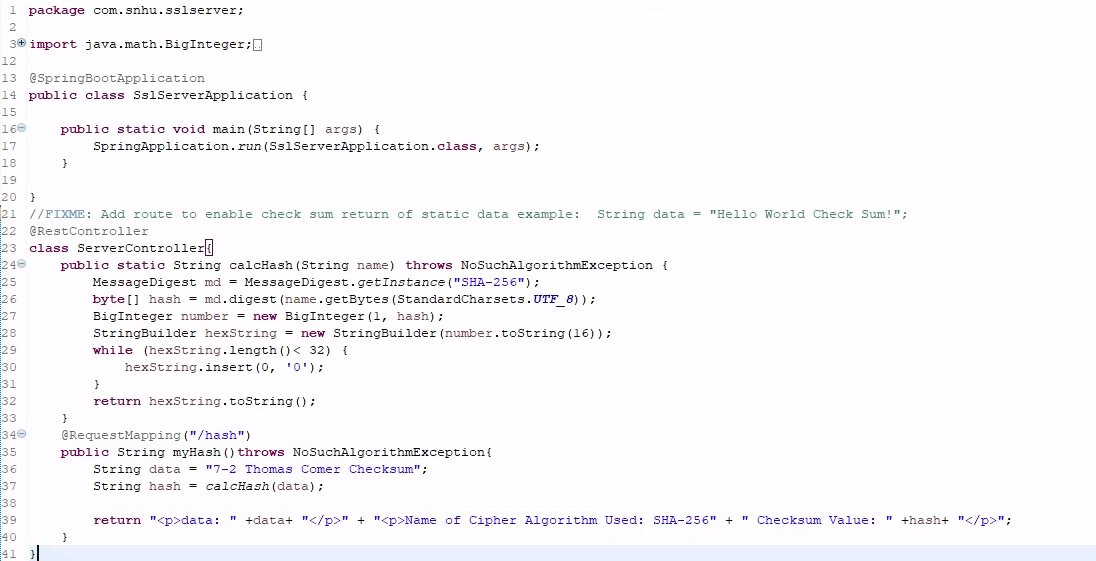




## Functional Testing

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



## Summary and Industry Standard Best Practices

Prior to refactoring the code, the main issue with the program was that it did not utilize an encryption algorithm to ensure that data sent through the program can be safely stored and shared without fear of an unauthorized party handling it. In order to address this, I created a class utilizing the RESTful API RestController called ServerController. In its current implementation it receives a string “data”, and encrypts it, and then sends out the data to a local network. For visibility purposes, it also sends the unencrypted string along with the encrypted string, so currently whoever is reading the data can see the unencrypted information. This can lead to a security risk if an unauthorized party were to be able to connect to the local network in the program’s current state, so this feature should most likely be removed in the final version of the program in order to preserve the security of the data. The network connection also utilizes HTTPS with a secure connection, which ensures a safe connection to the network. I also ran a dependency check and encountered no new vulnerabilities with my additions to the program, meaning that the program at the dependency level is safe from attacks, although regular dependency checks should be performed in order to ensure safety for the foreseeable future. There are a few other industry standard best practices that could have been utilized, such as input validation, and Principle of least privilege that were not considered, due to the program currently not accepting any form of user input, and only sending out one string of information, making these practices unnecessary in the current version. If later versions were to add those as features, then I recommend ensuring that Industry Standard Best Practices are kept in mind to ensure security. In conclusion, the program is far more secure now than it was before, and follows the industry standard for best practices to ensure that Artemis Financial’s client data is safe and secure.