****

# 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** | **08/13/2024** | **Merrik Wright** |  |

## 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

Merrik Wright

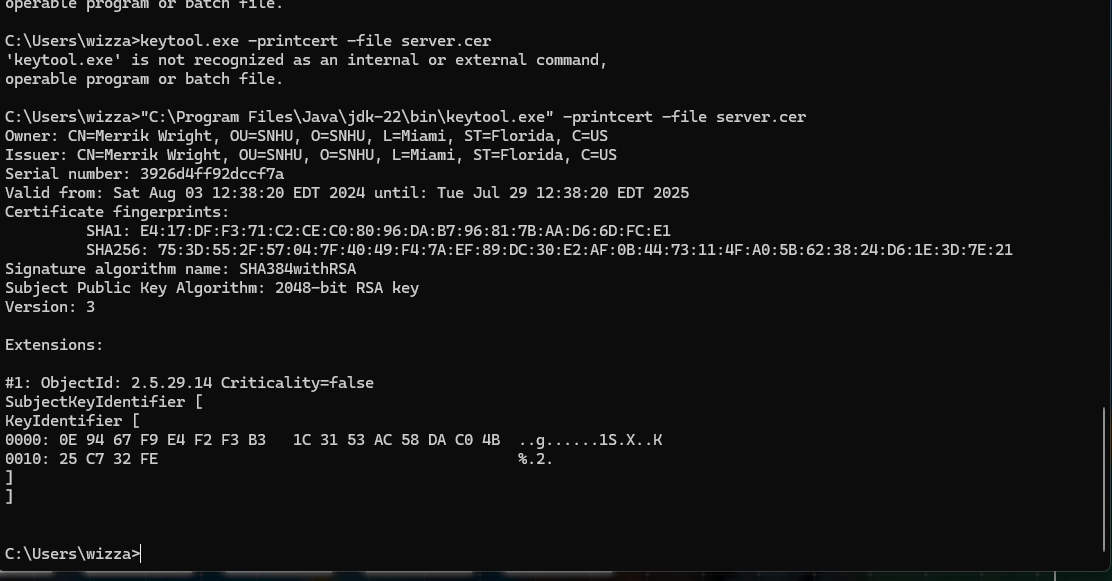
## Algorithm Cipher

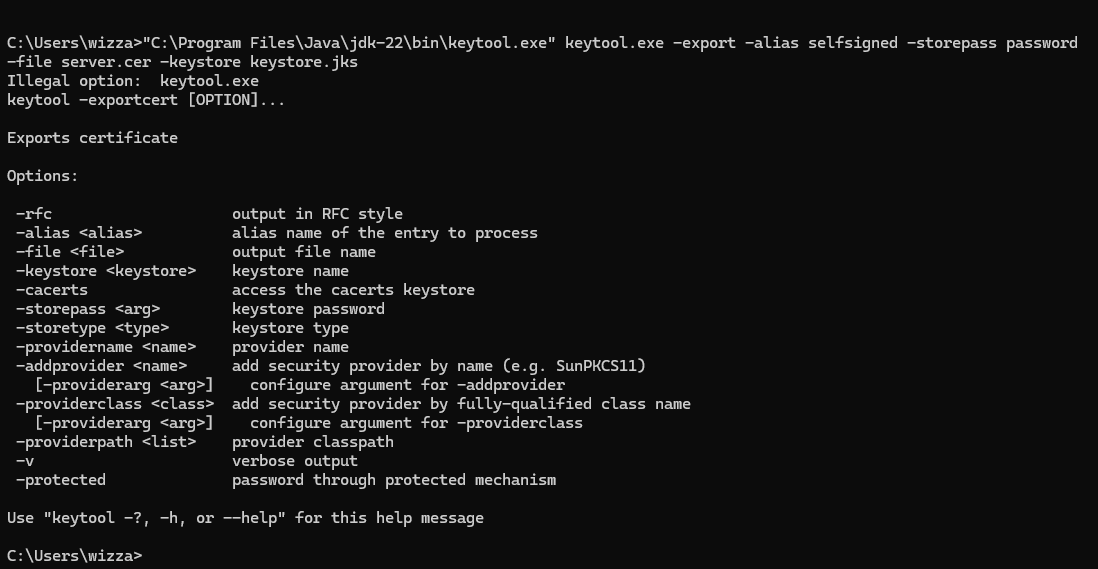
Artemis Financial is seeking to enhance the security of their web application to ensure secure communications. Given that the most likely attack vector for a financial institution involves bad actors attempting to gain unauthorized access to sensitive information for financial gain, encryption is the best recommendation. Encryption will render the data useless to any potential attacker without the corresponding decryption key. Since the firm is focused on securing communication, I recommend implementing an asymmetric encryption method. In asymmetric encryption, the key used for encryption is public, while the key used for decryption is private, providing an additional layer of security.

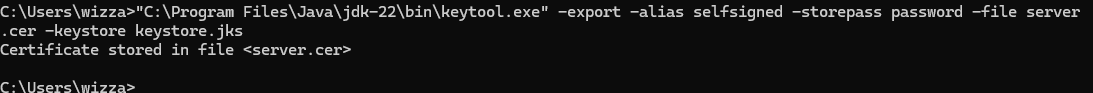
To achieve the highest level of security, particularly because the information could be transmitted externally, I suggest using the SHA-256 algorithm with 256-bit keys for encryption. SHA-256 offers robust encryption with a high degree of security, as it generates an extensive range of possible key combinations with a 256-bit key length. Furthermore, the SHA-256 algorithm utilizes Java’s random number generator, which ensures the security of the encryption by creating a non-reversible checksum that verifies the file's integrity. The hash function will employ the SHA-256 algorithm to generate a checksum for the provided message, ensuring that the data remains secure and valid during transmission.

## Certificate Generation

Insert a screenshot below of the CER file.

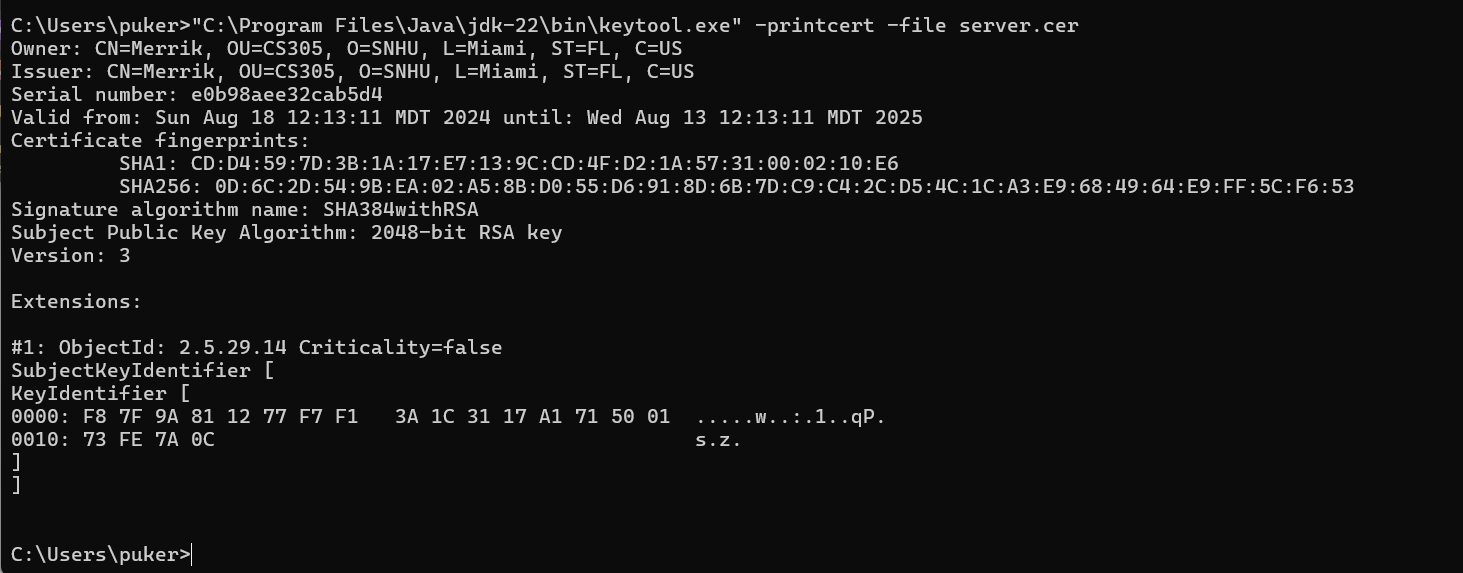






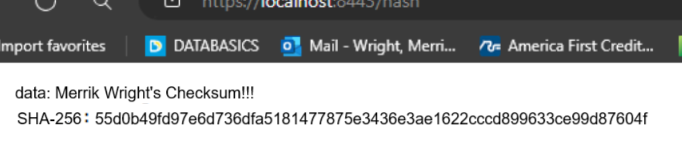
I finished the assignment at home. But I had to travel back home because of a family emergency.

I did not have enough time to dedicate all of my energy, and know this is going to be worse. But see below for my new process as I did it all again. I am on my brother’s computer, and that is why the computer name has changed.



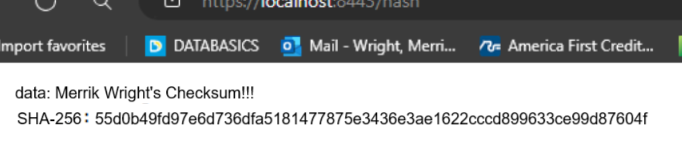
## Deploy Cipher

Insert a screenshot below of the checksum verification.



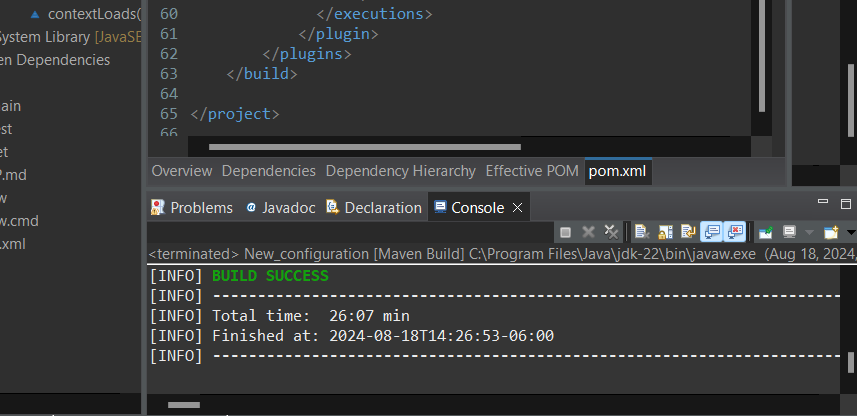
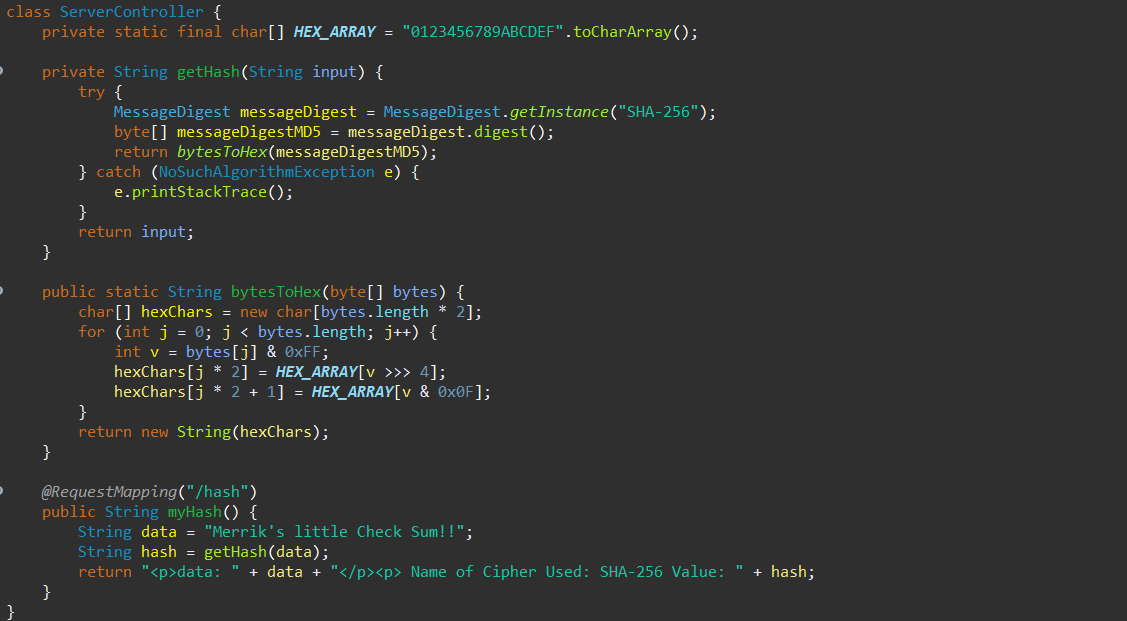
## Secure Communications

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



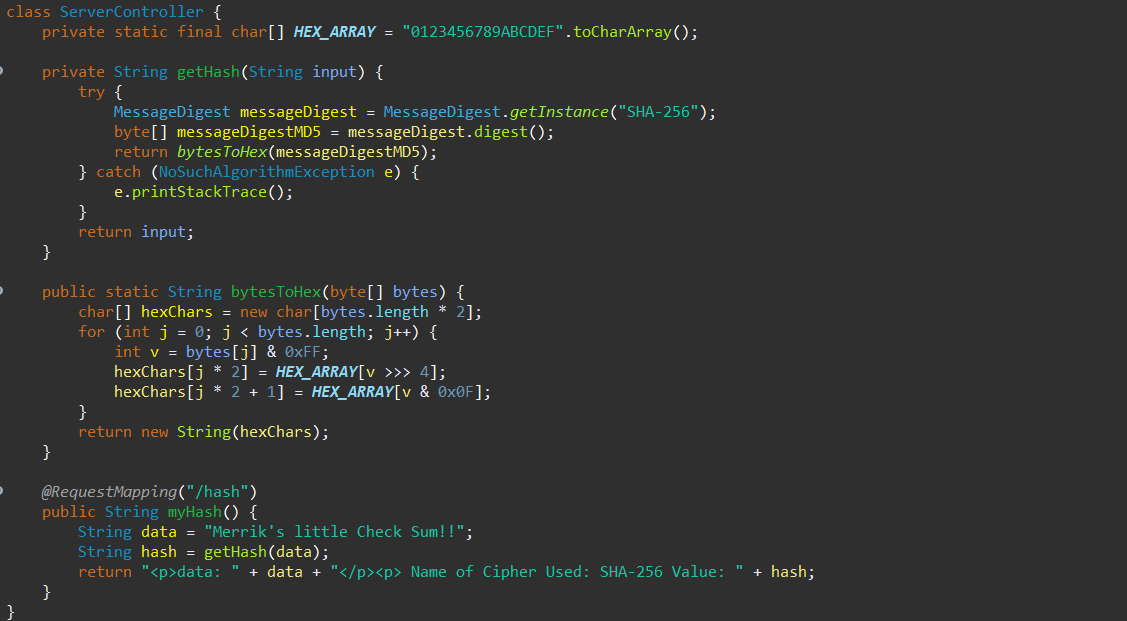
## Secondary Testing

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

In refactoring my code, I added a secured RestController to serve as the secure interface for the program's hashing RESTful endpoint. The ServerController class is designed to address the issues identified in the vulnerability assessment diagram. I chose to implement the SHA-256 hashing algorithm because it offers a high level of security with minimal risk of collisions. To maintain the application's security, I recommend conducting dependency checks once or twice a month to stay updated on potential vulnerabilities. This practice will help protect the company and its sensitive data. Additionally, keeping the dependencies and plugins in the pom.xml file up to date will ensure that the application is running the latest versions, which is crucial for maintaining optimal security.

## Industry Standard Best Practices

In developing the software application, I adhered to industry standard best practices for secure coding to mitigate known security vulnerabilities and maintain the existing security of the system. I implemented strong encryption algorithms, such as SHA-256, to ensure data integrity and confidentiality. Additionally, I enforced secure communication protocols by using HTTPS for data transmission and ensured that all sensitive data, such as user credentials and financial information, were stored securely and encrypted. Regular dependency checks were also scheduled to promptly address any vulnerabilities in third-party libraries, and all plugins were kept up to date to protect against new security threats.

Applying these industry standard best practices for secure coding is crucial to the company's overall well-being, as it significantly reduces the risk of data breaches and unauthorized access to sensitive information. By implementing these practices, the company not only protects its clients' data but also strengthens its reputation as a secure and reliable financial institution. This proactive approach to security helps to build trust with clients, ensures compliance with industry regulations, and ultimately safeguards the company's long-term success.