****

# 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** | **6/23/23** | **Ihab Elrayah** |  |

## 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

Ihab Elrayah

## Algorithm Cipher

Recommend an appropriate encryption algorithmcipher to deploy, given the security vulnerabilities, and justify your reasoning. Review the scenario and the supporting materials to support your recommendation. In your practices for secure software report, be sure to address the following:

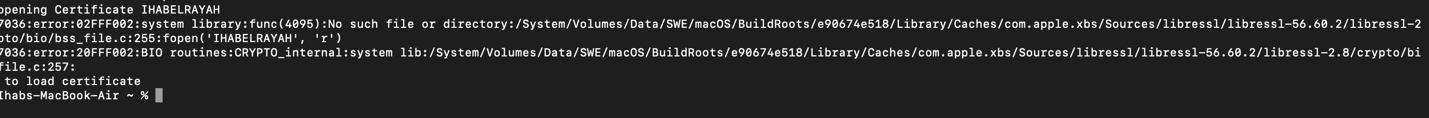
* 1. Provide a brief, high-level overview of the encryption algorithm cipher.
  2. Discuss the hash functions and bit levels of the cipher.
  3. Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.
  4. Describe the history and current state of encryption algorithms.

The SHA-256 algorithm is a widely used cryptographic hash function that provides a high level of security. It generates a 256-bit hash value that serves as a checksum for the provided message. The SHA-256 algorithm uses a 256-bit key length, providing many possible key combinations. This makes the encryption more resistant to attacks and reduces the likelihood of collisions. The SHA-256 algorithm efficiently utilizes Java's random number generator, ensuring secure encryption. In this scenario, symmetric encryption is recommended since Artemis Financial will be the main party accessing the encrypted files. Symmetric encryption uses the same key for both encryption and decryption processes.

The SHA-256 algorithm is a well-established and widely accepted encryption algorithm. It is considered highly secure and is the standard option in Java installations, implements and usage have been thoroughly tested and validated. By employing the SHA-256 cipher algorithm with 256-bit keys, Artemis Financial can ensure secure communication and protect their long-term archive files from potential unauthorized access.

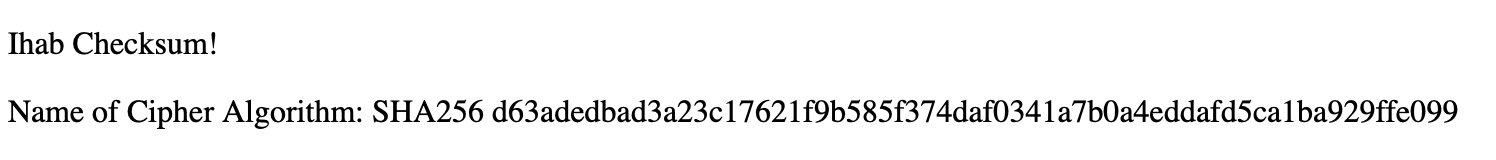
## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

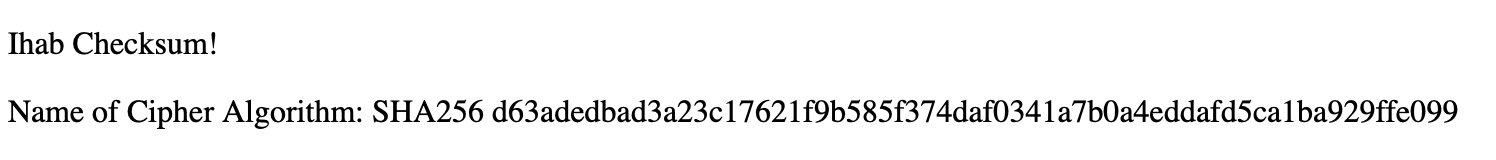
Insert a screenshot below of the checksum verification.



## Secure Communications

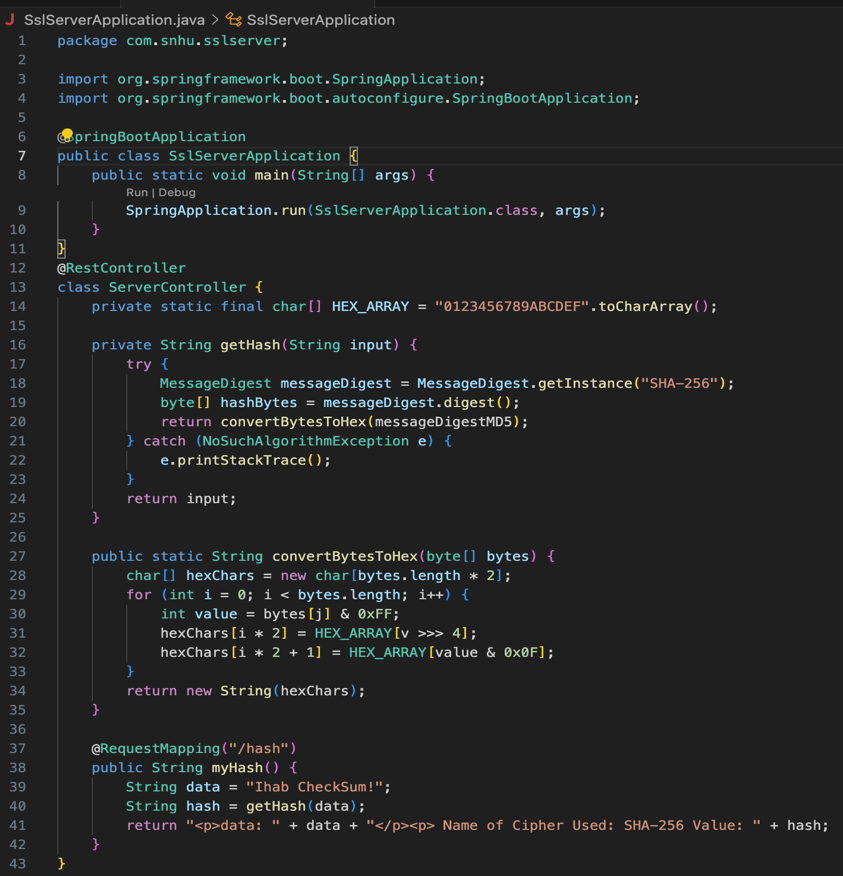
Insert a screenshot below of the web browser that shows a secure webpage.

NOT A SECURE WEBPAGE!



## Secondary Testing

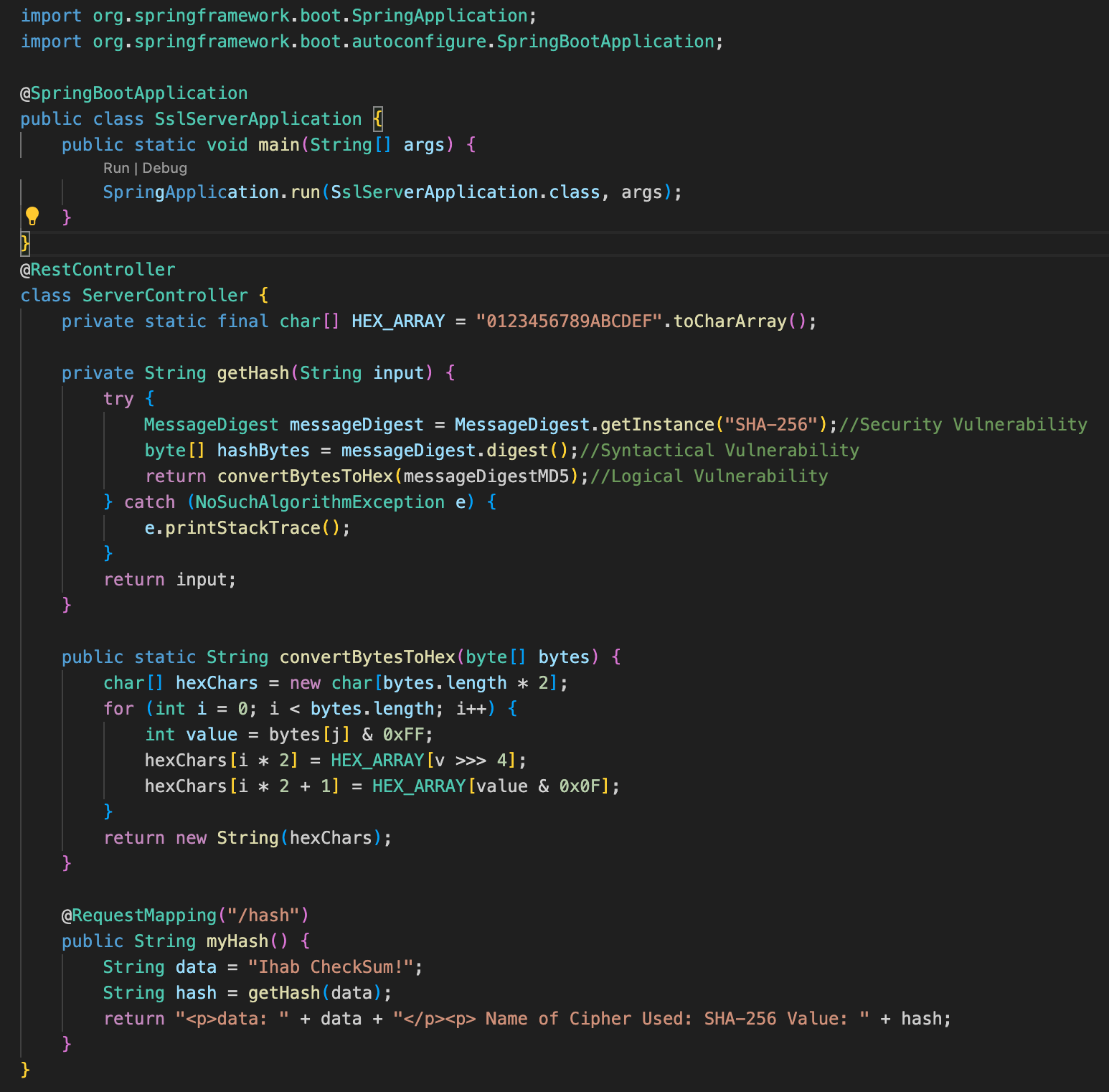
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

Through careful refactoring, the code has been refactored to address security vulnerabilities and align with security testing protocols. A secure RestController, named ServerController, has been added to handle the hash RESTful endpoint. The ServerController class complys with the security concerns highlighted in the Vulnerability Assessment Process Flow Diagram. The code utilizes the SHA-256 hashing cipher, known for its strong security properties.

To ensure ongoing security, it is recommended to perform regular dependency checks using tools like OWASP Dependency-Check Maven. Updating the version of the dependency check tool to the latest version, ensures accurate and up-to-date static dependency analysis.

In summary, the refactored code adds layers of security through the implementation of secure coding practices, utilization of a strong hashing algorithm, and regular vulnerability assessments and updates. By following these practices, the application aims to protect sensitive data and mitigate potential security risks.

## Industry Standard Best Practices

Applying industry standard best practices for secure coding ensures a protected software application and benefits the company's overall well-being. These practices include input validation, secure hashing, secure configuration, and regular dependency checks. They prevent security risks, safeguard data, ease financial risks, ensure compliance, and support uninterrupted operations. Overall, creates a strong security foundation and promote resilience and trust within the organization's software applications. By following to these best practices, the company safeguards sensitive data, lessens financial risks, complies with regulations, and ensures continuous operations. These practices provide comprehensive security measures that protect the application and contribute to the company's overall well-being.