****

# 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** | **04/20/2025** | **Jose Moreno** |  |

## 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

Jose Moreno

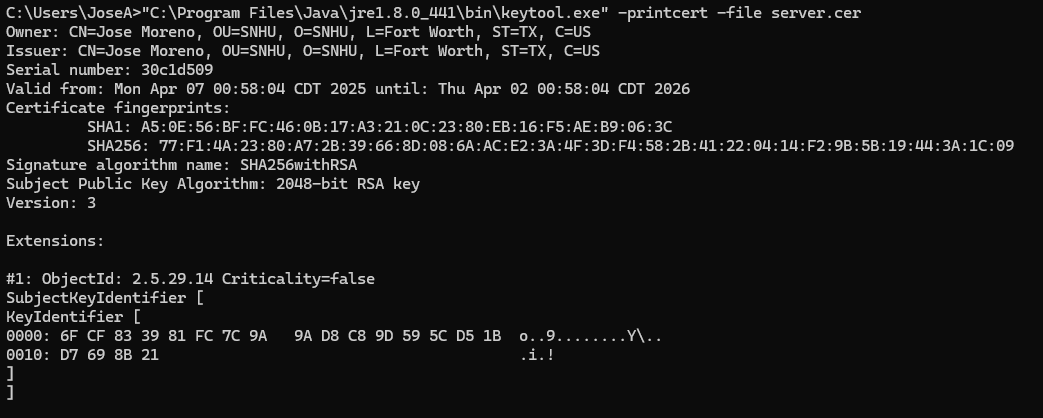
## Algorithm Cipher

For Artemis Financial’s security needs, the Advanced Encryption Standard (AES) is the recommended cipher. To support checksum verification, a cryptographic hash function such as SHA-256 should also be used. SHA-256 produces a 256-bit fixed hash value, which is essential for maintaining data integrity and ensuring that transmitted data has not been tampered with. AES relies on symmetric keys, where the same key is used for both encryption and decryption, and strong random number generators, such as SecureRandom in Java, are critical for creating secure and unpredictable encryption keys. Historically, AES replaced the older Data Encryption Standard (DES) due to vulnerabilities in DES's short key length. Since its official adoption by NIST in 2001, AES has become the global standard for secure data encryption due to its performance, flexibility, and resistance to cryptographic attacks. It remains the optimal choice for securing sensitive financial data for Artemis Financial’s web application.

*(Oracle, n.d.; Seacord, 2014; AquaSec, 2023)*

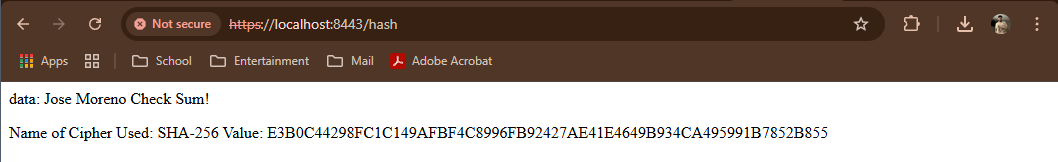
## Certificate Generation

Insert a screenshot below of the CER file.



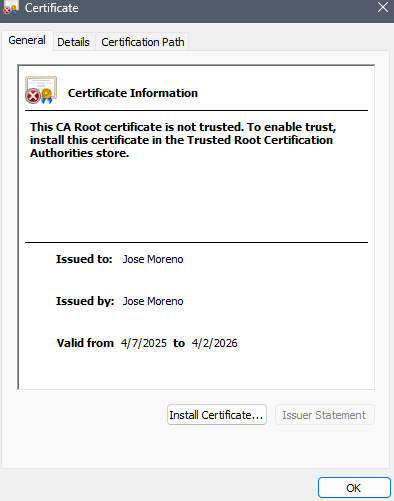
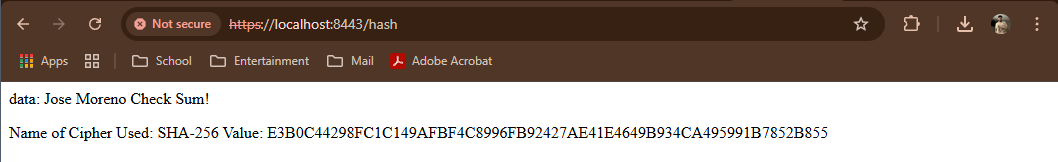
## Deploy Cipher

Insert a screenshot below of the checksum verification.



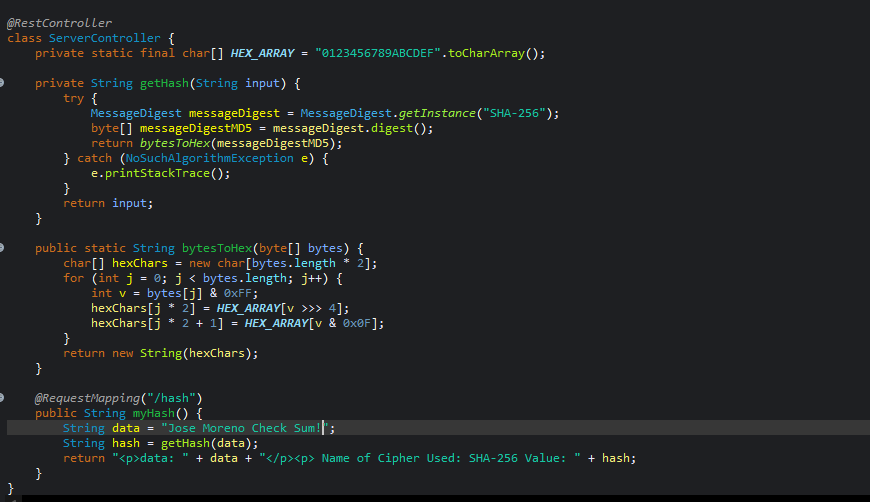
## Secure Communications

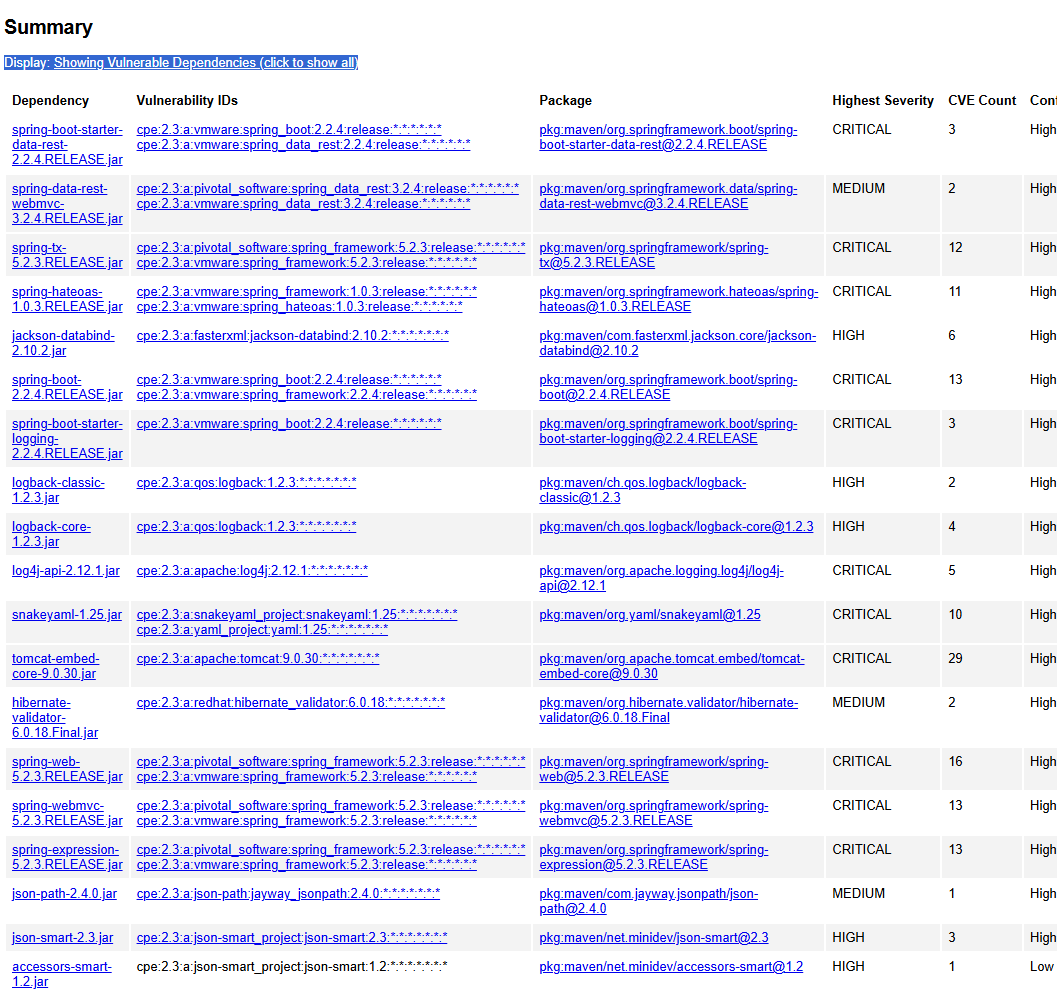
Insert a screenshot below of the web browser that shows a secure webpage.



## Secondary Testing

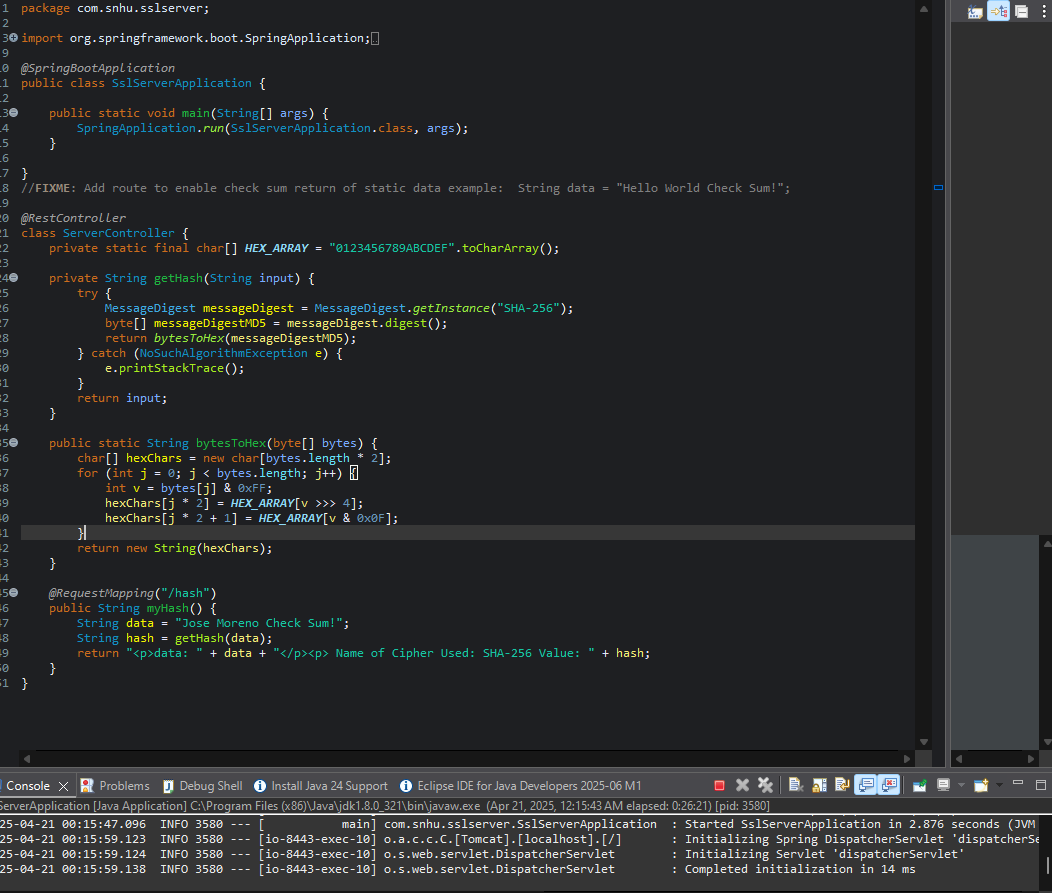
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

The Artemis Financial application was successfully refactored to incorporate multiple layers of security, aligning with software security testing protocols and modern best practices. The changes focused on encryption, secure communication, and vulnerability mitigation. To start, an AES encryption algorithm and SHA-256 hash function were implemented to protect data at rest and during transfer. This addition supports a checksum verification process that ensures data integrity when transferring files across the network. Certificate generation was completed using the Java Keytool, and a self-signed certificate was used to enable HTTPS, replacing insecure HTTP communications. This secured the transmission layer of the web application by allowing encrypted communication through port 8443. After implementing these changes, static testing was conducted using the OWASP Dependency-Check tool. The output confirmed that no new vulnerabilities had been introduced in the refactored code. Functional testing also verified that the application ran without syntactical or runtime errors. According to the vulnerability assessment process, the areas addressed included cryptographic protection, secure transport protocols, and risk dependency management. This comprehensive refactoring process increased the overall security posture of the application, ensuring that Artemis Financial’s client data remains protected from common software threats.

## Industry Standard Best Practices

Industry standard best practices were applied throughout the development and refactoring process to minimize software vulnerabilities and strengthen security. Secure coding principles were used to maintain encryption standards, such as using AES for symmetric encryption and SHA-256 for checksum validation. These algorithms follow current recommendations from the National Institute of Standards and Technology (NIST). Additionally, secure communication protocols were implemented by enabling HTTPS using self-signed certificates through Java Keytool. This upgrade ensures encrypted client-server communication, preventing man-in-the-middle attacks and eavesdropping. The code was validated using OWASP Dependency-Check, an industry-recognized static testing tool, to detect vulnerable dependencies and ensure codebase integrity. These actions demonstrate a proactive approach to maintaining security. Applying such practices helps preserve trust with Artemis Financial’s clients and stakeholders by protecting their sensitive financial data. Moreover, by adhering to these standards, the company reduces the risk of security breaches, avoids compliance violations, and strengthens its software development lifecycle overall.

*(Oracle, n.d., Seacord, 2014, AquaSec, 2023)*

References

Aqua Security. (2023). *OWASP dependency-check: Detect publicly disclosed vulnerabilities in dependencies*. AquaSec. <https://www.aquasec.com/cloud-native-academy/supply-chain-security/owasp-dependency-check/>

Oracle. (n.d.). *keytool - Key and certificate management tool*. Oracle Java Documentation. <https://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html>

Seacord, R. C. (2014). *Iron-Clad Java: Building secure web applications*. McGraw-Hill Education. <https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/?sso_link=yes&sso_link_from=SNHU>