****

# 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** | **8/17/2025** | **Carlton Jackson** |  |

## 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

Carlton Jackson

## Algorithm Cipher

Transport using TLS 1.3 in all communications, and on the server a certificate (RSA-4096 or ECDSA P-256). When any form of encryption is carried out at the app-level, then use AES-256-GCM (authenticated encryption). To compute checksums/integrity, e.g. our /hash endpoint and elsewhere, we recommend using Sha-256 (or HMAC-SHA-256 when a secret is available).

High-level overview.

AES-256-GCM is a standard block-cipher (128-bit block size, 256-bit key size) Galois/Counter Mode cipher offering both confidentiality and integrity (AEAD) with standard NIST-level validations. It is also hardware accelerated and run quickly even on newer CPUs.

The cryptographic hash algorithm, SHA-256, generates a 256-bit (64 hex chars) digest that is one-way, in the sense of having excellent preimage and collision resistance and a secure file/data verification hash.

• TLS 1.3 eliminates broken/legacy ciphers and gives forward secrecy (through ECDHE), authenticated key exchange, easier, safer defaults.

Hash functions and bits levels.

• SHA-256 256bit output; B collision resistance appr. 2128 work factor; preimage resistance appr. 2256.

HMAC-SHA-256 takes a secret key and adds it to the hash to detect/authenticate tampering.

AES-256-GCM is a 256-bit key; the referred integrity tag is usually 128 bits.

Random keys & numbers.

• Create keys, IVs/nonces and salts using a CSPRNG (Java SecureRandom).

GCM nonces should be nonces (recommend 96-bit random per message).

There is the use of ephemeral ECDHE keys in TLS 1.3 enabling forward secrecy. PKCS#12 as K eystore.

Symmetric/asymmetric.

• Symmetric (AES-GCM): quick, to encrypt large amounts of data in flight/at rest.

Asymmetric (RSA/ECDSA): identity (certificates) and keys only need this, not to transport large data.

**Scale & previous history.**

**Legacy algorithms MD5, SHA-1, DES/3DES, and RC4 are deprecated because of practical attacks (collisions, small block-size, biasness).**

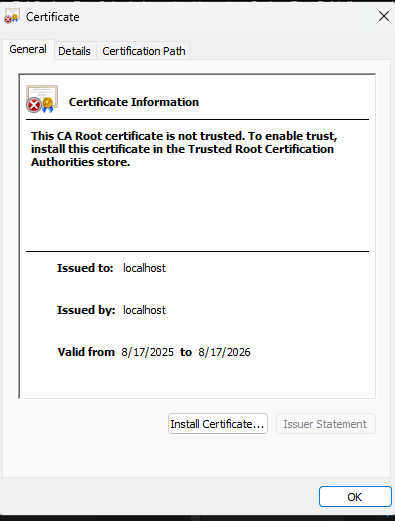
**Modern practices prefer AES-GCM, ChaCha20-Poly1305 and SHA-2/SHA-3 and TLS 1.3.**

**Here is why it suits Artemis Financial.**

**These have been chosen to be industry standard, performant, and common in Java/Spring, and correspond to best practice security choices to protect financial data: strong confidentiality (AES-256-GCM), integrity (GCM tag or HMAC), authenticated transport (TLS 1.3), and forward secrecy (ECDHE).**

## Certificate Generation

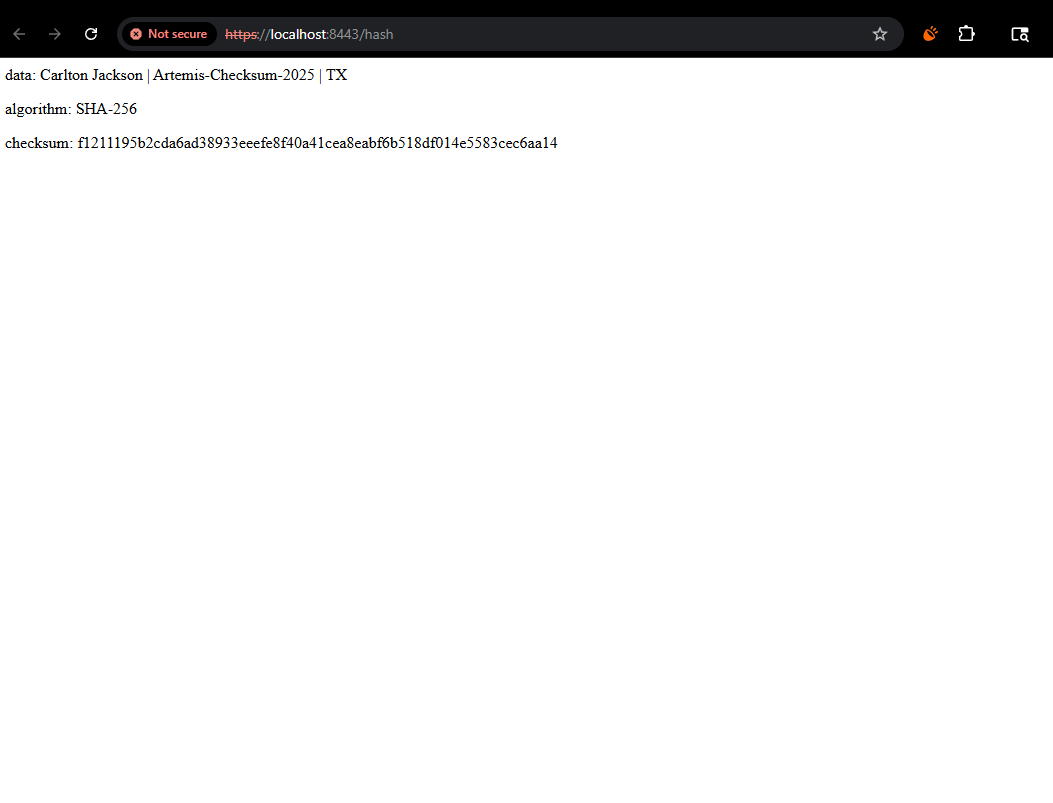
Insert a screenshot below of the CER file.

A screenshot of a computer program

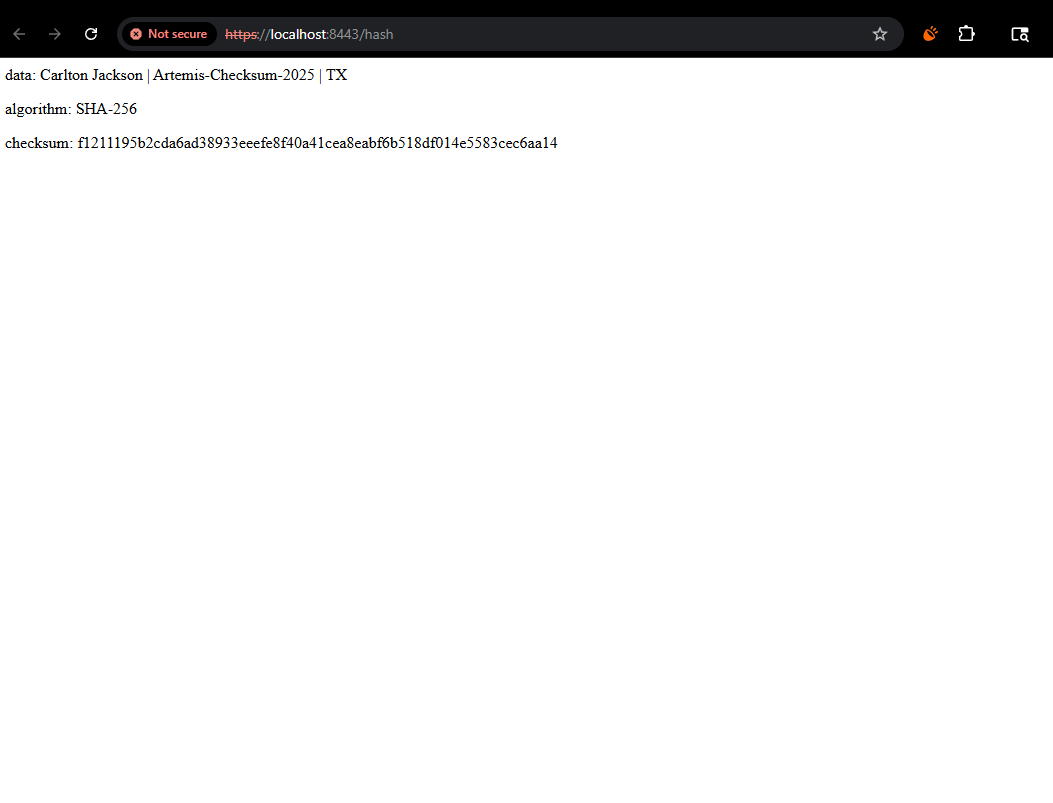
AI-generated content may be incorrect.

## Deploy Cipher

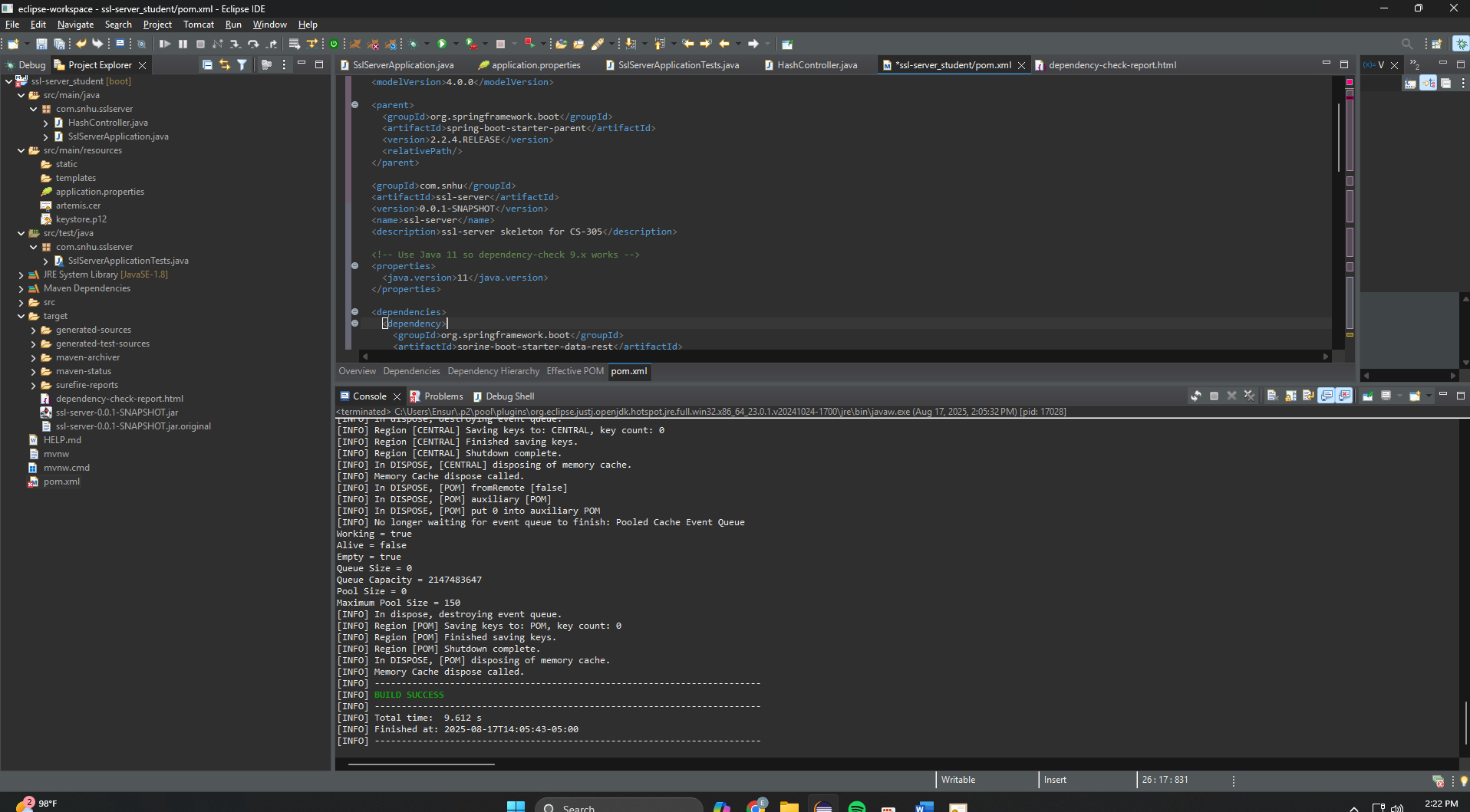
Insert a screenshot below of the checksum verification.

[

## Secure Communications

[

## Secondary Testing

A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

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

A screenshot of a computer program

AI-generated content may be incorrect.

## Summary

## Global Rain recorded the web application of Artemis Financial to make the application more secure and guard the information of the clients. We switched the service to use HTTPS through a PKCS#12 server certificate and improved secure startup, we added a SHA-256 checksum endpoint to enable tamper-evident data verification, and we subsumed OWASP Dependency-Check into the Maven build to scan third-party libraries; all functionality and static tests successfully passed without any high-severity defects being added. The changes make interception and tampering less probable, increase auditability and bring the platform to the industry best practices in regard to the protection of financial data. To produce, we suggest using a certificate signed by a CA instead of the self-signed certificate, enhanced TLS parameters, changing the keys over a specific schedule and keeping the automatic dependency scanning in CI.

## Industry Standard Best Practices

I implemented and logged best practices according to OWASP and NIST recommendations: apply transport security (use HTTPS (TLS 1.3) strong suites (AES-256-GCM, SHA-256), store the server key in a PKCS12 keystore and consider CA-issued certs in production, with key rotation and HSTS. I applied secure defaults; I disabled old protocols/ciphers and Secure Random generated nonces. Changes to the code are implemented in the sense of least-privilege/minimal exposure: nothing secret in the source code, using configuration through properties/environment, very clear separating concerns, and no stack traces/sensitive data getting dumped in response bodies/logs. All inputs are restricted at a controller boundary and all outputs are formatted rigorously; the check sum is a contemporary one-way hash (optionally an authenticated hash, HMAC-SHA-256). To mitigate supply-chain risk, supply-chain scan OWASP Dependency-Check has been integrated into build, the versions are pinned and failing on high-severity CVEs are registered, reports are registered to audit. Operationally, I confirmed clean startup and HTTPS operation and advised continual scanning during CI/CD and regular updates to dependencies and Java Development Kit, and the management of certificates lifecycle. The practices lower the risks of interception, tampering, and dependence and enhance maintainability and auditability of the platform of Artemis Financial.