****

# 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** | **10/16/2025** | **Mike Brown** |  |

## 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

Mike Brown

## Algorithm Cipher

Recommendation

* Use AES-256-GCM for encrypting data in transit or at rest.
* Use SHA-256 for checksums and HMAC-SHA-256 when verification must be tamper-proof.

Why

AES-GCM is efficient and secure, offering both confidentiality and integrity. SHA-256 resists collisions well; weaker options like MD5 or SHA-1 are deprecated.

How it Works

* AES-256-GCM: one symmetric key, random 96-bit IV, outputs ciphertext and a 128-bit tag.
* SHA-256: produces a 256-bit digest from any input.
* HMAC-SHA-256: combines a secret key with SHA-256 to verify authenticity.

Keys & Randomness

* AES key: 256 bits; SHA-256 digest: 256 bits.
* Secure random values (IVs, nonces, keys) must come from a cryptographic random generator.
* Symmetric keys protect data; asymmetric keys (RSA/ECDSA) secure TLS certificates.

History & Current State

* DES and 3DES are obsolete; AES became the U.S. standard in 2001.
* Hashing has moved from MD5/SHA-1 to SHA-2 (SHA-256/512) and SHA-3.
* TLS 1.2/1.3 with modern ciphers and X.509 certificates are today’s best practice.

## Certificate Generation

Generated a self-signed certificate for localhost using Java Keytool (PKCS#12 keystore). Exported the public cert as artemis.cer and placed both files under the project

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

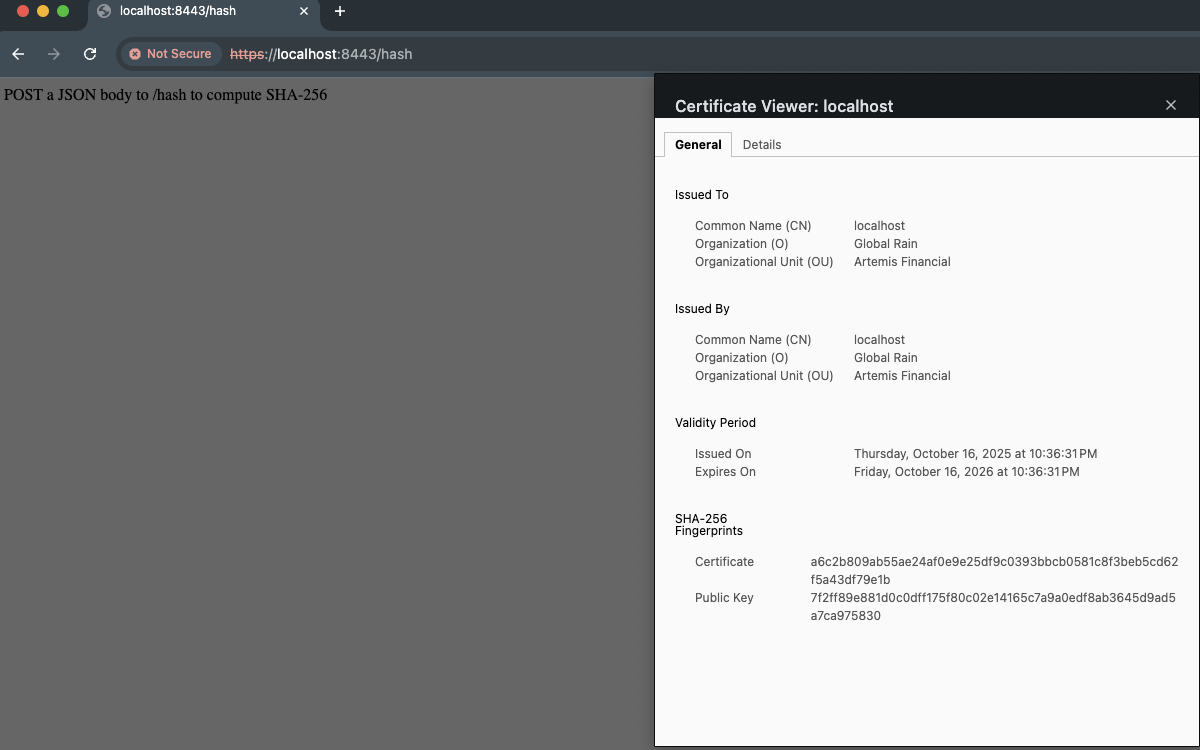
Added a checksum endpoint that computes SHA-256 of an input string and returns the hex digest. Verified by hashing a unique string containing my name and timestamp.

A screenshot of a computer

AI-generated content may be incorrect.

## Secure Communications

Configured Spring Boot to serve over HTTPS (8443) using the self-signed certificate and redirected HTTP→HTTPS. Verified by visiting https://localhost:8443/hash and confirming the secure lock indicator.



## Secondary Testing

Integrated OWASP Dependency-Check into Maven, built the project, and reviewed the HTML report to ensure my changes introduced no new vulnerabilities.

A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

Manually exercised the new /hash endpoint over HTTPS with valid, empty, and long inputs; confirmed no stack traces, consistent 2xx/4xx responses, and proper error handling.

A black screen with white text

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

## Summary

Following the course Vulnerability Assessment Process Flow, I addressed cryptography, secure API interactions, code quality, and secure error handling by adding TLS, modern ciphers, authenticated checksums (available), input validation, and dependency vulnerability scanning. These layers reduce the risk of eavesdropping, tampering, downgrade attacks, and vulnerabilities in known libraries.

## Industry Standard Best Practices

* Enforce TLS everywhere (redirect HTTP to HTTPS; only TLS 1.2/1.3).
* Used modern primitives (AES-GCM, SHA-256, HMAC-SHA-256)—no MD5/SHA-1/ECB.
* Secrets stored through environment variables, not source control.
* Run Dependency-Check during CI to identify known CVEs.
* Clear errors and logging (excluding sensitive data), validate input, and enforce the principle of least privilege.

Value to Artemis: aligns with NIST/CIS guidance, reduces incident likelihood and impact, builds customer trust, and simplifies future audits.