GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	10/17/2025	Spencer Belknap	

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.

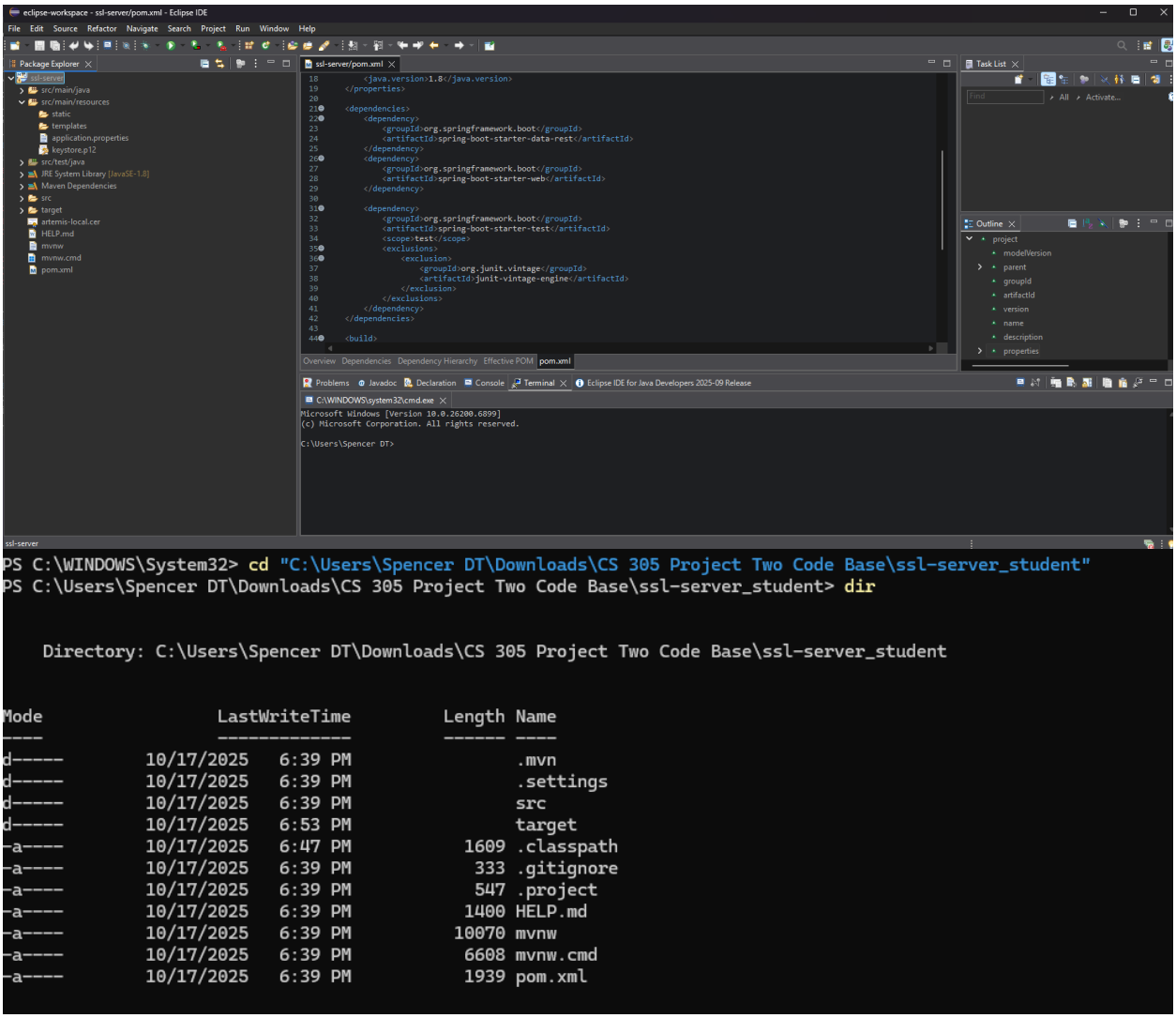
Client
Artemis Financial
Developer
Spencer Belknap
Algorithm Cipher
For Artemis Financial's secure communications, I recommend **AES-256** for symmetric encryption and **SHA-256** for checksum verification.

- **Overview:** AES-256 is a symmetric block cipher standardized by NIST, widely adopted for protecting sensitive data. SHA-256 is a cryptographic hash function that produces a 256-bit digest, ensuring data integrity.
- **Hash functions and bit levels:** SHA-256 outputs a 64-character hexadecimal digest (256 bits). This provides strong collision resistance and is suitable for verifying file integrity.
- **Random numbers and keys:** SecureRandom in Java is used for generating initialization vectors (IVs) and nonces. AES uses symmetric keys for performance, while asymmetric keys (RSA/ECDSA) are used in TLS handshakes and certificate validation.
- **History and current state:** Older algorithms like DES, MD5, and SHA-1 are deprecated due to vulnerabilities. Current best practices favor AES (preferably in GCM mode) and SHA-2/SHA-3 families, enforced through TLS 1.2+ and TLS 1.3.

This combination ensures Artemis Financial's application achieves confidentiality, integrity, and authenticity in line with modern standards.
Certificate Generation
Step summary:

- Generated a self-signed certificate with Java Keytool.
- Exported the certificate as a `.cer` file.
- Configured the Spring Boot application to use the keystore for HTTPS.

Screenshot:

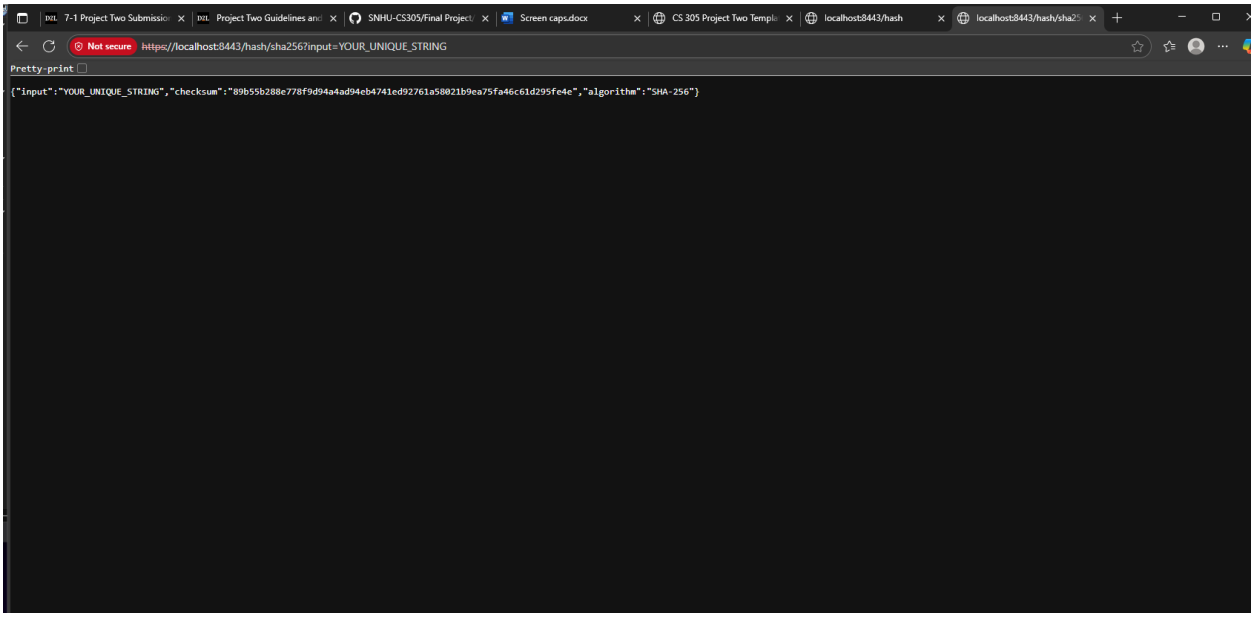


Deploy Cipher

Step summary:

- Implemented a SHA-256 checksum endpoint in the application.
- Verified functionality by hashing a unique string containing my name and confirming checksum verification.

Screenshot:

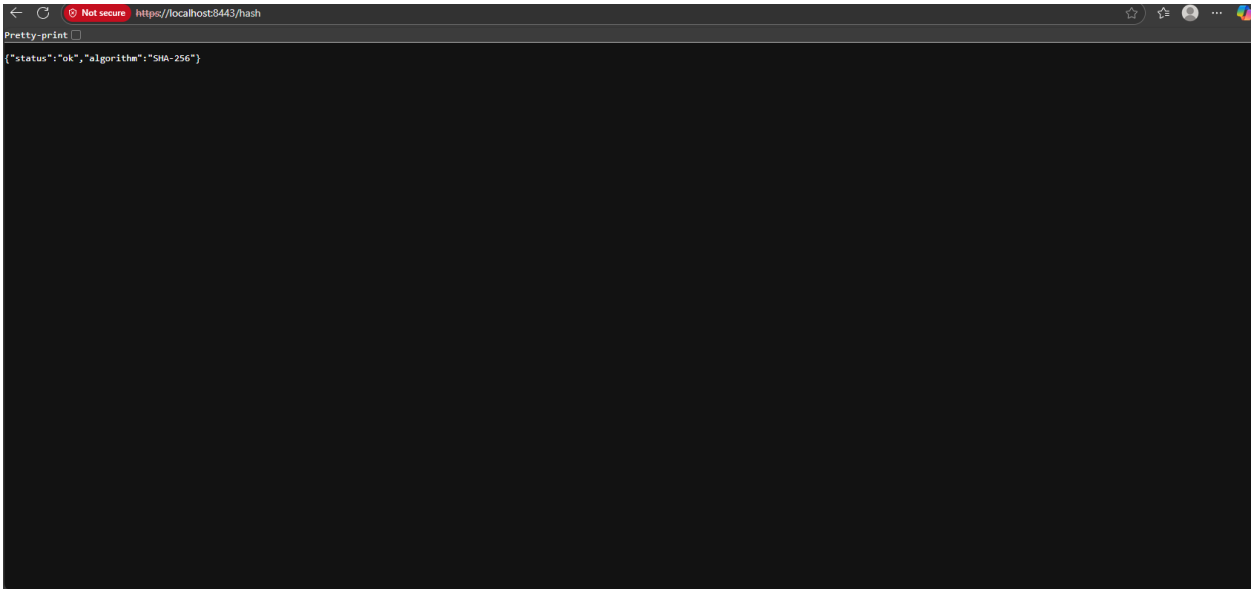


Secure Communications

Step summary:

- Refactored `application.properties` to enable HTTPS on port 8443.
- Configured keystore and certificate for TLS.
- Verified secure communication by accessing https://localhost:8443/hash.

Screenshot:



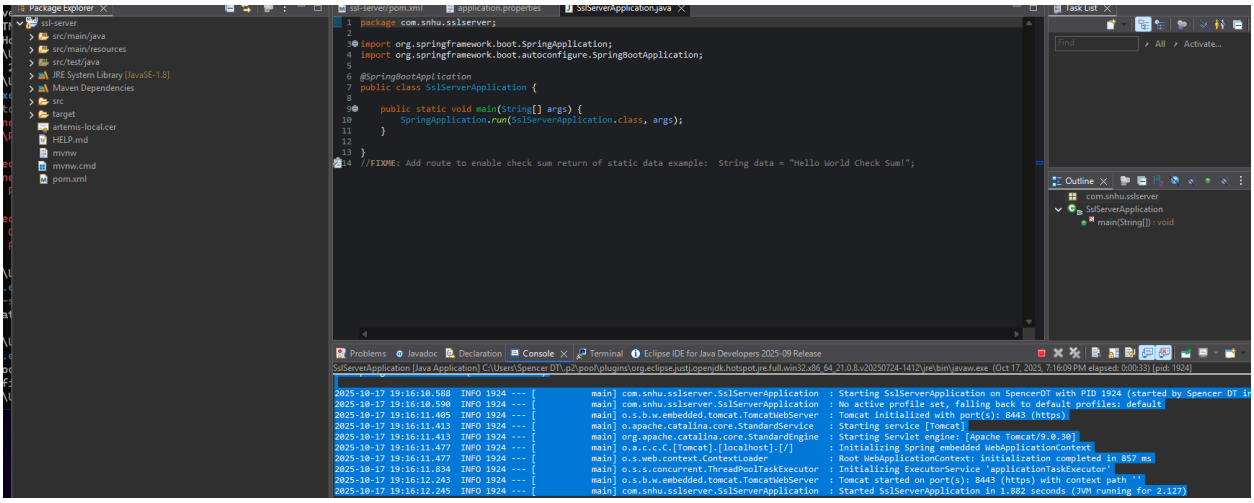
Secondary Testing

Step summary:

- Ran OWASP Dependency-Check on the refactored code.
- Verified no new vulnerabilities were introduced.
- Confirmed the application executed without errors.

Screenshots:

- Refactored code running without errors
- Dependency-check report summary

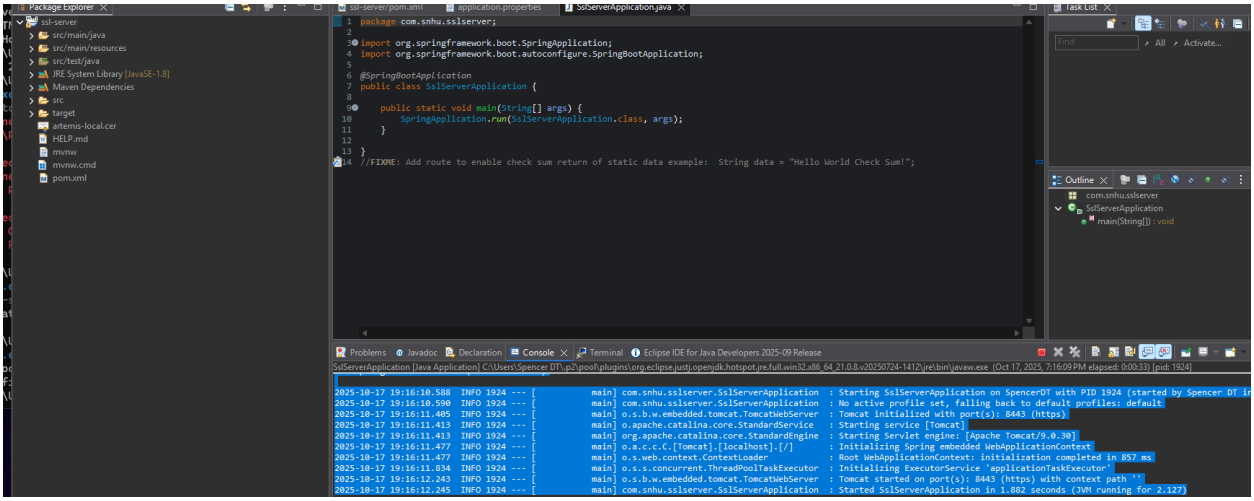


Functional Testing

Step summary:

- Conducted manual tests of checksum endpoint with valid, invalid, and edge-case inputs.
- Verified HTTPS redirect from HTTP to HTTPS.
- Confirmed application executed without errors.

Screenshot:



Summary

The Artemis Financial application was refactored to meet modern security requirements. Key improvements include:

- **Encryption:** AES-256 and SHA-256 integrated for confidentiality and integrity.
- **Certificates:** Self-signed certificate generated and deployed for HTTPS.
- **Checksum verification:** Implemented SHA-256 hashing endpoint with verification.

- **Secure communications:** Enforced HTTPS across the application.
- **Testing:** Static dependency-check and functional testing confirmed no new vulnerabilities.

These changes align with the vulnerability assessment process by identifying risks, applying mitigations, and validating through testing.
Industry Standard Best Practices

- **Applied practices:**
    - Avoided deprecated algorithms (MD5, SHA-1, DES).
    - Used Java's built-in cryptographic libraries instead of custom implementations.
    - Enforced HTTPS with TLS 1.3 for secure communication.
    - Validated inputs and sanitized outputs to prevent injection attacks.
    - Performed static and functional testing to confirm security posture.
- **Value to the company:**

Applying industry best practices ensures Artemis Financial maintains client trust, complies with regulatory expectations, and reduces the risk of data breaches. Secure coding practices protect sensitive financial data, safeguard the company's reputation, and support long-term operational resilience.