****

# 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** | **8/16/2025** | **Daniel Smith** |  |

## 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

Daniel Smith

## Algorithm Cipher

For Artemis Financial’s secure communications and file verification needs, I recommend implementing the Advanced Encryption Standard (AES) in conjunction with the SHA‑256 cryptographic hash function. AES is a symmetric block cipher standardized by the U.S. National Institute of Standards and Technology (NIST) in 2001, replacing the older and less secure Data Encryption Standard (DES). It is widely regarded as the industry standard for encrypting sensitive data due to its balance of performance, security, and broad platform support (National Institute of Standards and Technology [NIST], 2001). SHA‑256, part of the SHA‑2 family, is a one‑way cryptographic hash function that produces a fixed‑length 256‑bit hash value, making it ideal for checksum verification and data integrity validation (Eastlake & Hansen, 2011).

**Hash Functions and Bit Levels**

* **AES Key Sizes:** 128, 192, or 256 bits. For financial data, AES‑256 is recommended for maximum security.
* **SHA‑256 Output:** Alwaysproduces a 256‑bit (32‑byte) hash, regardless of input size.
* **Security Strength:** AES‑256 combined with SHA‑256 provides strong resistance against brute‑force and collision attacks (Menezes, van Oorschot, & Vanstone, 1996).

**Symmetric vs. Asymmetric Keys**

* **AES (Symmetric):** Uses the same secret key for encryption and decryption. This makes it computationally efficient and well‑suited for encrypting large volumes of data once a secure channel is established.
* **RSA (Asymmetric):** Uses a public/private key pair. In this solution, RSA can be used for secure key exchange during TLS/HTTPS handshakes, while AES handles the actual data encryption.
* **Integration:** In Artemis Financial’s application, HTTPS/TLS will use RSA for certificate‑based authentication, then negotiate an AES session key for secure communication (Rescorla, 2018).

**Use of Random Numbers**

* **Key Generation:** AES keys should be generated using Java’s SecureRandom class to ensure cryptographic‑grade entropy.
* **Initialization Vectors (IVs):** For modes like AES‑CBC or AES‑GCM, a unique, unpredictable IV must be generated for each encryption operation to prevent pattern analysis (Ferguson, Schneier, & Kohno, 2010).

**History and Current State of Encryption Algorithms**

* **Historical Context:** DES, introduced in the 1970s, became vulnerable to brute‑force attacks as computing power increased. Triple DES (3DES) extended its life but was eventually deprecated (NIST, 2017).
* **AES Adoption:** Selected through an open competition in the late 1990s, AES (based on the Rijndael algorithm) became the U.S. federal standard in 2001 and is now used globally in banking, government, and cloud services.
* **Current Status:** AES remains unbroken when implemented correctly, and SHA‑256 is still considered secure against practical attacks. Both are recommended by NIST and widely supported in modern cryptographic libraries.

**Justification for Artemis Financial**

* **Regulatory Alignment:** Meets NIST and OWASP recommendations for financial data protection.
* **Performance:** AES is optimized in modern CPUs, ensuring minimal performance impact.
* Security: AES‑256 + SHA‑256 provides confidentiality, integrity, and resistance to known cryptographic attacks.
* **Compatibility:** Fully supported in Java’s javax.crypto and java.security packages, making integration straightforward in the existing Spring Boot application.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen with white text

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

AI-generated content may be incorrect.

## Secure Communications

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

A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

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

A close up of text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

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 white background with black text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Summary

The refactoring of the SslServerApplication and its associated test class was guided by the **Vulnerability Assessment Process Flow** outlined in the Supporting Materials. Following the process, I:

* **Identified potential vulnerabilities** during the initial code review, including the absence of a secure hashing implementation and the risk of using outdated or insecure algorithms.
* **Analyzed the impact** of these vulnerabilities on data integrity and confidentiality.
* **Applied mitigations** by replacing insecure or missing functionality with a SHA‑256–based checksum endpoint, ensuring cryptographic operations follow modern standards.
* **Validated the fix** through functional testing (/checksum endpoint output) and automated unit tests (SslServerApplicationTests), confirming that the application runs securely without breaking existing functionality.

This approach addressed key areas in the process flow: *Identify Vulnerabilities → Assess Risk → Implement Remediation → Verify Security Controls*.

**Process for Adding Layers of Security**

To strengthen the application’s security posture, I implemented multiple layers of defense:

1. **Secure Hashing** – Introduced SHA‑256 with Base64 encoding to ensure data integrity and prevent tampering.
2. **Unique Data Generation** – Incorporated UUID and timestamp to prevent predictable outputs, mitigating replay and collision risks.
3. **HTTPS Enforcement** – Maintained Spring Boot’s SSL configuration to ensure encrypted communication between client and server.
4. **Minimal Exposure** – Limited endpoint output to non-sensitive, non-identifiable data, reducing the attack surface.
5. **Automated Testing** – Retained and validated the contextLoads() test to ensure the application’s secure configuration loads without errors.

## Industry Standard Best Practices

**Maintaining Existing Security**

* **Cryptographic Standards** – Used SHA‑256, a NIST‑approved hashing algorithm, instead of weaker alternatives like MD5 or SHA‑1.
* **Secure Defaults** – Preserved HTTPS configuration and avoided exposing internal system details in responses.
* **Dependency Management** – Ensured Maven dependencies are scanned with OWASP Dependency‑Check, suppressing only verified false positives.

**Mitigating Known Vulnerabilities**

* **Input/Output Sanitization** – Controlled endpoint output to prevent injection or leakage of sensitive data.
* **Principle of Least Privilege** – Limited the scope of the new endpoint to a single, well-defined function.
* **Defense in Depth** – Combined cryptographic integrity checks, secure transport, and controlled output to create multiple barriers against exploitation.

**Value to the Company’s Well‑Being**

Applying industry standard best practices for secure coding directly supports the company’s operational and reputational health by:

* **Reducing Risk** – Minimizing the likelihood of data breaches, service disruptions, and compliance violations.
* **Ensuring Compliance** – Aligning with regulatory frameworks (e.g., NIST, OWASP Top 10) that clients and auditors expect.
* **Building Trust** – Demonstrating to customers and stakeholders that security is embedded in the development lifecycle, not treated as an afterthought.
* **Lowering Long‑Term Costs** – Preventing vulnerabilities early in development reduces the expense of post‑deployment fixes and incident response.

**References**

Eastlake, D., & Hansen, T. (2011). *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)* (RFC 6234). Internet Engineering Task Force. https://doi.org/10.17487/RFC6234

Ferguson, N., Schneier, B., & Kohno, T. (2010). *Cryptography Engineering: Design Principles and Practical Applications*. Wiley.

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. CRC Press.

National Institute of Standards and Technology. (2001). *Announcing the Advanced Encryption Standard (AES)* (FIPS PUB 197). https://doi.org/10.6028/NIST.FIPS.197

National Institute of Standards and Technology. (2017). *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher* (NIST SP 800‑67 Rev. 2). https://doi.org/10.6028/NIST.SP.800-67r2

Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3* (RFC 8446). Internet Engineering Task Force. https://doi.org/10.17487/RFC8446