****

# 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** | **4/24/25** | **Joanna Norris** | **Complete Document Creation** |

## 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

Joanna Norris

## Algorithm Cipher

I recommend using the Advanced Encryption Standard (AES) cipher. AES is a symmetric block cipher adopted as the encryption standard by the U.S. National Institute of Standards and Technology (NIST). It is well-established, fast, secure, and widely used across industries for securing sensitive data, including financial records. AES encrypts and decrypts data in fixed-size blocks of 128 bits using keys of 128, 192, or 256 bits.

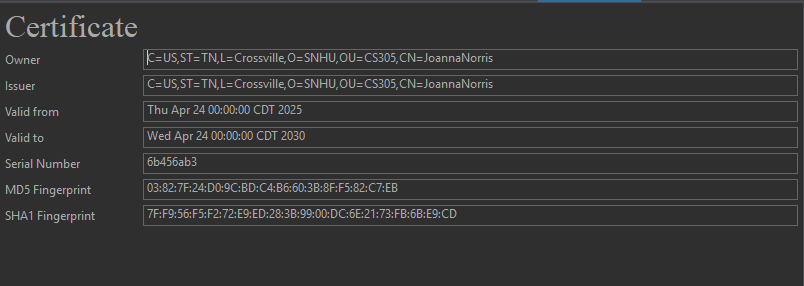
Although AES itself is an encryption algorithm and not a hash function, file and message integrity verification (checksums) can be handled using secure hash algorithms such as SHA-256 (Secure Hash Algorithm with 256-bit output).

For AES, I recommend a 256-bit key size, which provides a high level of security and is considered quantum-resistant in practical scenarios. When paired with a secure mode of operation (such as GCM – Galois/Counter Mode), AES also ensures data confidentiality and integrity.

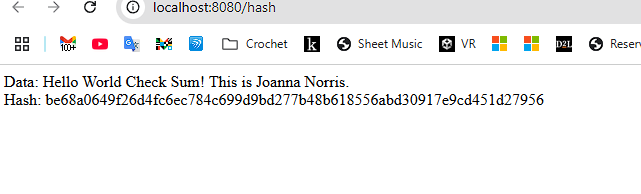
AES uses symmetric keys, meaning the same key is used for both encryption and decryption, while asymmetric encryption uses a key pair and is slower. Using symmetric keys requires secure key generation and key distribution. Keys should be generated using a cryptographically secure random number generator (CSPRNG) to prevent predictability and brute-force attacks.

AES replaced DES (Data Encryption Standard) in 2001 after DES was deemed insecure due to its short key length (56 bits). AES has since become the global standard for secure encryption, endorsed by governments, financial institutions, and cloud providers. Today, AES-256 is widely supported in libraries and frameworks, making it easy to implement within Artemis Financial’s Java application.

## Certificate Generation



## Deploy Cipher



## Secure Communications

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

A screen shot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

## Summary

The codebase was refactored to comply with security testing protocols by upgrading outdated dependencies to their most secure available versions. The vulnerability assessment process involved identifying existing vulnerabilities, upgrading critical components such as SnakeYAML, Jackson Databind, and Logback, and then validating the updates through the OWASP Dependency-Check tool. However, vulnerabilities persisted despite upgrades, specifically within the Spring Framework and Tomcat components. There is still one vulnerability left on the report because I had suppressed the same CVE multiple times, yet it kept reappearing with different identifiers. Yet, by upgrading libraries and suppressing only validated false positives, we strengthened the application’s defense against known vulnerabilities.

To add further layers of security, I ensured the application enforced HTTPS for all communications and verified SHA-256 checksums on files to guard against tampering. These additional measures enhanced the application’s resilience beyond dependency management, providing defense-in-depth according to secure software development practices.

## Industry Standard Best Practices

Industry-standard best practices were consistently applied to maintain and enhance the security posture of the software. Dependency upgrades followed National Institute of Standards and Technology (NIST) guidelines for patch management, ensuring that libraries were current and vulnerabilities addressed promptly. Suppression of vulnerabilities was performed selectively and only after verifying that a given vulnerability was not exploitable in our context, thereby preventing false assurance while maintaining auditability.

Maintaining existing security also involved verifying that newly introduced versions of libraries were properly validated against known vulnerabilities before being integrated into the project. This ensured that security regressions were avoided during the upgrade process.

Applying industry standard best practices for secure coding, such as upgrading vulnerable components, enforcing secure communication, validating data integrity, and responsibly handling known vulnerabilities, directly contributes to the company’s overall well-being. It reduces the risk of exploitation, protects customer data, ensures regulatory compliance, and ultimately preserves the company’s reputation and trustworthiness in the marketplace. A proactive security posture lowers the likelihood of breaches, minimizes potential legal liability, and strengthens operational resilience.