****

# 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** |
| --- | --- | --- | --- |
| **0.9** | **2/22/2025** | **Fnu Samim** | **Discussed the best Algorithm Cipher to be implemented.** |
| **1.0** | **2/22/2025** | **Fnu Samim** | **Added a self-signed certificate to enable HTTPS communication and secure data transmission.** |
| **1.1** | **2/22/2025** | **Fnu Samim** | **Deployed SHA-256 cryptographic cipher to enhance data encryption and ensure secure communication.** |
| **1.2** | **2/22/2025** | **Fnu Samim** | **Implemented secure SSL/TLS communication to protect data in transit and prevent eavesdropping.** |
| **1.3** | **2/22/2025** | **Fnu Samim** | **Enhanced the cipher algorithm by adding a dynamically generated salt for hashing and encoded the salt using Base64 to ensure proper handling and representation of binary data.** |
| **1.4** | **2/22/2025** | **Fnu Samim** | **Wrote the summary and made sure industry best practices.** |

## 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

Fnu Samim

## Algorithm Cipher

Given the security vulnerabilities in the scenario, I recommend deploying the AES-256 (Advanced Encryption Standard) encryption algorithm cipher. AES-256 is a symmetric key encryption algorithm, meaning it uses the same key for both encryption and decryption. It operates on 128-bit blocks of data and supports key sizes of 256 bits, providing a massive keyspace of 2^256, which makes it highly resistant to brute-force attacks. AES-256 is widely regarded as one of the most secure encryption standards and is trusted by governments, financial institutions, and industries globally. Its efficiency and strong security make it ideal for protecting sensitive data, such as financial records, while ensuring compliance with regulations like GDPR, HIPAA, and FIPS 140-2.

AES-256 can be used in modes like CBC (Cipher Block Chaining) or GCM (Galois/Counter Mode). GCM is particularly recommended because it provides both confidentiality and authenticity through built-in authentication capabilities. To enhance security, encryption keys should be generated using secure random number generators and stored in a key management system (KMS) or hardware security module (HSM). Additionally, hash functions like SHA-256 can be used alongside AES-256 to ensure data integrity. SHA-256 generates a unique 256-bit hash value for input data, making it nearly impossible for attackers to alter data without detection. The combination of AES-256 for encryption and SHA-256 for hashing creates a robust defense against data breaches and unauthorized access.

The history of encryption algorithms highlights the importance of adopting modern standards like AES-256. Early ciphers, such as the Caesar cipher, were simple and easily broken, while DES (Data Encryption Standard) became obsolete due to its small 56-bit key size. AES, introduced in 2001, addressed these limitations and has since become the gold standard for symmetric encryption. Its resistance to cryptographic attacks and widespread adoption make it the best choice for securing sensitive data. While AES-256 may introduce some performance overhead, its superior security justifies its use, especially for long-term archival of sensitive financial data. By implementing AES-256, Artemis Financial can ensure data confidentiality, integrity, and compliance with industry regulations, safeguarding the company’s reputation and operational well-being.

## Certificate Generation

Insert a screenshot below of the CER file.

A screen shot of a computer

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 computer screen with numbers and text

AI-generated content may be incorrect.

## Secondary Testing

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

A screen shot 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 computer screen shot of a program

AI-generated content may be incorrect.

## Summary

The refactored code improves security by addressing critical vulnerabilities identified during the assessment process. Originally, the application used a fixed data string for hashing without any protective measures, leaving it vulnerable to precomputed attacks such as rainbow table attacks. To address this, a secure salt generation mechanism was implemented using SecureRandom, ensuring that each hash is unique and resistant to such attacks. Additionally, the salt is now encoded in Base64, ensuring proper handling and representation of binary data. The refactoring process followed a structured approach: identifying security gaps, applying cryptographic best practices, and testing the application to confirm its resilience against common vulnerabilities. By integrating these security enhancements, the application now adheres to secure software development principles, significantly reducing risks associated with predictable hashing and unauthorized data exposure.

(I understand that salting isn't necessary in this scenario, as it's primarily used for password security and should be stored for verification. However, I wanted to experiment with it as a learning experience and thought it was a cool implementation.)

## Industry Standard Best Practices

Industry-standard best practices were applied throughout the development and refactoring processes to maintain the software application’s existing security. First, secure coding principles were implemented to address vulnerabilities such as predictable hashing and static data exposure. For instance, a secure salt generation mechanism utilizing SecureRandom was introduced to ensure unique hashes and mitigate rainbow table attacks. Base64 encoding of the salt was also incorporated to ensure proper data handling and representation. These measures align with cryptographic best practices and demonstrate a proactive approach to securing the application against common threats.

Applying industry-standard best practices for secure coding is essential for the company’s well-being. By adhering to these standards, the company reduces the risk of data breaches, financial losses, and reputational damage. Secure coding practices protect sensitive information and build trust with users and stakeholders. Furthermore, they ensure compliance with regulatory requirements, minimizing legal and operational risks. Ultimately, investing in secure coding practices promotes a culture of security awareness and positions the company as a reliable and responsible entity in the industry.