****

# 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** | **2/25/2024** | **Matthew Cruz** | **-Added sections 1-8** |

## 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

Matthew Cruz

## Algorithm Cipher

Given the scenario's requirement for secure communication and data verification within Artemis Financial's web application, the most suitable encryption algorithm cipher to deploy is the Advanced Encryption Standard (AES). AES is a symmetric encryption algorithm widely recognized for its robust security and efficiency. It was established as the standard encryption algorithm by the U.S. National Institute of Standards and Technology (NIST) in 2001, replacing the older Data Encryption Standard (DES).

AES operates on fixed block sizes of 128 bits and supports key sizes of 128, 192, or 256 bits, providing flexibility in security levels based on the sensitivity of the data being encrypted. While AES itself is not a hash function, it incorporates hash functions such as the Rijndael key schedule internally. The algorithm transforms plaintext into ciphertext through a series of substitution and permutation steps, ensuring the confidentiality and integrity of the data.

In terms of key management, AES relies on symmetric encryption, meaning the same key is used for both encryption and decryption. Secure random number generation is crucial for generating strong encryption keys, and AES supports the generation of random keys to maintain the security of the encrypted data.

Over the years, AES has undergone extensive analysis and review by cryptographers worldwide, establishing its reputation as a highly secure encryption algorithm. It has shown flexibility against known cryptographic attacks when implemented correctly. This track record of security, combined with its widespread adoption in various industries and applications, makes AES the ideal choice for securing communication and implementing data verification mechanisms in Artemis Financial's web application.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen with white text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

Description automatically generated

## Secure Communications

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

[Insert screenshots here.]

## Secondary Testing

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

A screenshot of a computer program

Description automatically generated

A screenshot of a computer error

Description automatically generated

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

Through the process of refactoring the code and aligning with security testing protocols, several crucial security enhancements were implemented to fortify the software application. Firstly, the introduction of a hashing algorithm, specifically MD5, allowed for the calculation of checksums to ensure data integrity verification. By incorporating the myHash() method and creating a route (/hash), static data undergoes MD5 hashing, establishing a robust layer of security to authenticate and verify the integrity of transmitted information. This step directly addresses concerns outlined in the Vulnerability Assessment Process Flow Diagram, particularly emphasizing data integrity verification, which is paramount in safeguarding against unauthorized alterations or tampering of sensitive data during transmission or storage.

Moreover, a significant security enhancement was achieved by transitioning the application from HTTP to HTTPS protocol. By configuring the application.properties file to support HTTPS, with SSL certificate information specified, the software application now encrypts data transmitted between clients and servers. This transition bolsters network security, mitigating the risks associated with eavesdropping and man-in-the-middle attacks. Furthermore, the adoption of HTTPS aligns with best practices for secure communication, providing authentication and encryption mechanisms to safeguard sensitive information exchange. By adhering to secure coding practices and integrating HTTPS for secure communication, alongside implementing a cryptographic hash algorithm for data integrity verification, the refactored code demonstrates a proactive approach towards fortifying the software application against potential security threats and vulnerabilities.

## Industry Standard Best Practices

In revamping the software application, I ensured to follow to widely accepted rules and practices for writing secure code. One significant change I made was transitioning from regular HTTP to HTTPS for communication between the application and users' web browsers. HTTPS encrypts the information sent back and forth, significantly enhancing security by making it much harder for hackers to steal or tamper with sensitive data like passwords or financial details. This step is akin to adding a secure lock to the door of the application, effectively safeguarding the information inside from unauthorized access.

Additionally, I implemented a special method to verify if the data being sent or received by the application has been tampered with. I utilized a technique called hashing, which generates a unique fingerprint for the data. This fingerprint acts as a seal, ensuring the integrity of the data and detecting any unauthorized modifications. By incorporating this measure, I guarantee that the information handled by the application remains reliable and trustworthy, particularly crucial for sensitive operations like financial transactions or handling personal information.

Following these best practices not only enhances the security of the application but also benefits the company. By proactively protecting our software and the data it handles, I mitigate the risk of costly security breaches or damage to our reputation. Moreover, adhering to industry standards advances trust with customers and partners, demonstrating our commitment to safeguarding their security and privacy. Furthermore, staying beside industry standards equips the company to effectively tackle emerging security challenges, ensuring our resilience and competitiveness in the ever evolving digital landscape.