****

# Practices for Secure Software Report

Tammy Hartline

SNHU CS305

Final Project

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** | **08/10/2023** | **Tammy Hartline** | 1. **Algorithm Cypher Recommendation** 2. **Proof of Certificate** 3. **Implemented and Deployed Cipher** 4. **Proof of Secure Communication** 5. **Secondary Testing** 6. **Functional Testing** 7. **Summary** 8. **Conclusion** |

## Client



## Instructions

Submit these 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

Tammy Hartline

## Algorithm Cipher

*Recommendation: AES (Advanced Encryption Standard)*

* *Brief Overview:*

AES is a widely used symmetric encryption algorithm that has been established as a standard for secure data encryption. It is considered secure, efficient, and suitable for various applications, including web communication. AES operates on blocks of data and uses keys of 128, 192, or 256 bits.

* *Hash Functions and Bit Levels:*

AES itself is not a hash function; it's a symmetric encryption algorithm. However, for hashing purposes (like generating checksums), you can use cryptographic hash functions like MD5 or SHA-256. MD5 is not recommended for security-critical applications due to its vulnerabilities, while SHA-256 is more secure.

* *Use of Random Numbers:*

Random numbers are crucial in cryptographic operations. AES uses random initialization vectors (IVs) to add randomness to the encryption process, enhancing security. Cryptographically secure random number generators (CSPRNGs) are used to generate these IVs.

* *Symmetric vs. Non-Symmetric Keys:*

AES is a symmetric encryption algorithm, meaning the same key is used for both encryption and decryption. It's efficient for bulk data encryption. Asymmetric encryption, on the other hand, uses different keys for encryption and decryption. While RSA is a popular asymmetric algorithm, it's computationally intensive and not recommended for bulk data encryption.

* *History:*

The need for a new encryption standard arose due to the weaknesses of the Data Encryption Standard (DES), which was becoming vulnerable to more advanced attacks as computing power increased. In response, the National Institute of Standards and Technology (NIST) initiated a competition in 1997 to select a successor. This competition attracted cryptographers from around the world, each submitting their encryption algorithms for evaluation.

After several years of rigorous analysis and testing, the Rijndael algorithm, proposed by Belgian cryptographers Vincent Rijmen and Joan Daemen, emerged as the winner. In 2001, NIST announced Rijndael as the Advanced Encryption Standard (AES), making it the official encryption standard for the United States government, and subsequently adopted worldwide.

* *Current State:*

Today, AES remains one of the most widely used symmetric encryption algorithms. Its robust security and efficiency have led to its incorporation into various applications across industries, including financial institutions, healthcare, e-commerce, and government agencies. The algorithm has stood the test of time, with no practical attacks against its core cryptographic properties.

AES supports key lengths of 128, 192, and 256 bits, providing different levels of security. The larger the key, the stronger the encryption, but also the higher computational overhead. The algorithm operates on blocks of data, typically 128 bits at a time, using a series of transformations including substitution, permutation, and mixing.

* *Strengths and Security:*

AES's strength lies in its resistance to all known practical attacks, including brute-force attacks, which involve trying all possible keys. The security of AES is rooted in its well-designed and thoroughly analyzed structure, making it resistant to differential and linear cryptanalysis, two powerful attack techniques.

* *Cryptanalysis and Updates:*

Since its adoption, AES has been subjected to extensive cryptanalysis, involving both academic research and industry scrutiny. While no successful attacks against the full AES algorithm have been discovered, researchers continue to explore related avenues, such as side-channel attacks and implementation vulnerabilities.

In response to emerging threats and technological advancements, NIST periodically reviews and updates its cryptographic standards. AES itself has remained largely unchanged, as its underlying principles and security characteristics have proven sound. However, NIST does recommend using the latest key lengths and cryptographic practices to ensure the highest level of security.

In conclusion, AES is a testament to the success of international collaboration in cryptographic research. Its historical significance, robust security, and continued relevance make it an excellent choice for securing Artemis Financials’ web application and data communication. However, staying informed about evolving cryptographic standards and practices is essential to maintaining the highest level of security in the ever-changing landscape of cybersecurity.

## Certificate Generation

A white background with colorful dots

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

## Deploy Cipher

A screenshot of a computer

Description automatically generated

## Secure Communications

A screenshot of a computer

Description automatically generated

## Secondary Testing

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

A screenshot of a computer

Description automatically generated

## Summary

The code was refactored to enhance its security posture by incorporating various security measures. To ensure secure software development, the following areas were addressed:

* **HTTPS with SSL/TLS:**

The code was modified to enable HTTPS with a valid SSL certificate. This involved configuring the application to use an SSL keystore for encrypted communication, enhancing data confidentiality and integrity during transmission.

* **Input Validation:**

The code was reviewed to ensure proper input validation. User inputs, such as context paths and keystore entries, were sanitized and validated to prevent potential injection attacks and unauthorized access.

* **Deprecation and Version Updates:**

Deprecated and vulnerable dependencies were updated to their latest versions to mitigate known vulnerabilities.

* **Dependency Scanning:**

The OWASP Dependency-Check plugin was integrated into the build process. It scans the project's dependencies for known vulnerabilities and generates a report, aiding in identifying and mitigating security risks introduced by third-party libraries.

**Added Layers of Security:**

* **Transport Security:**

HTTPS was enforced using a valid SSL certificate, adding an essential layer of encryption to protect data during transit.

* **Input Sanitization:**

User inputs were properly validated and sanitized to prevent common attacks like SQL injection and path traversal.

* **Dependency Management:**

Regular dependency scans were integrated into the build process, allowing the identification and prompt mitigation of vulnerabilities in third-party libraries.

## Industry Standard Best Practices

* **Secure Coding Guidelines:**

Industry standard best practices for secure coding were followed. This included input validation, secure configuration management, and the principle of least privilege.

* **Regular Auditing:**

The vulnerability assessment process, including static code analysis, was conducted as part of the development cycle to catch vulnerabilities early.

* **Continuous Improvement:**

By using the latest libraries and tools, security vulnerabilities were addressed proactively. The OWASP Dependency-Check plugin was particularly valuable for identifying and fixing dependencies with known vulnerabilities.

**Value of Industry Standard Best Practices:**

* **Maintaining Current Security:**

Industry standard best practices ensure that the software application remains up-to-date against evolving security threats. This mitigates the risk of emerging vulnerabilities and reduces the likelihood of breaches.

* **Company Wellbeing:**

Implementing secure coding practices safeguards sensitive data, maintains customer trust, and prevents costly security incidents. By adhering to best practices, the company minimizes the potential for reputational damage, legal liabilities, and financial losses associated with security breaches.

In conclusion, the code refactoring process addressed various aspects of security through HTTPS enforcement, input validation, dependency management, and adherence to industry standard best practices. By doing so, the software application's security posture was enhanced, contributing to the company's overall wellbeing and maintaining a strong defense against security threats.