****

# Practices for Secure Software Report

Contents

[Practices for Secure Software Report 1](#_Toc164616843)

[Document Revision History 3](#_Toc164616844)

[Client 3](#_Toc164616845)

[Instructions 3](#_Toc164616846)

[Developer 4](#_Toc164616847)

[1. Algorithm Cipher 4](#_Toc164616848)

[2. Certificate Generation 5](#_Toc164616849)

[3. Deploy Cipher 5](#_Toc164616850)

[4. Secure Communications 5](#_Toc164616851)

[5. Secondary Testing 6](#_Toc164616852)

[6. Functional Testing 7](#_Toc164616853)

[7. Summary 8](#_Toc164616854)

[8. Industry Standard Best Practices 9](#_Toc164616855)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **4/21/2024** | **Braydon Woodward** | **Refactored code for this version has been submitted through D2L to complete the report.** |

## 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

Braydon Woodward

## Algorithm Cipher

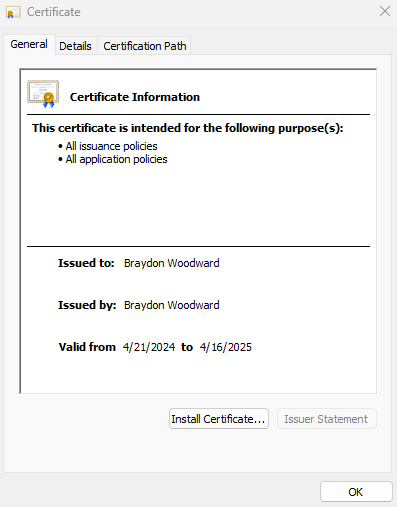
When advising Artemis Financial on encrypting their archive files, the selection of an encryption algorithm cipher is critical for safeguarding the security and integrity of their financial records. Our recommendation is AES-256, as this symmetric encryption algorithm cipher is recommended by the NIST and OWASP, and it aligns with the company's security needs and objectives. AES-256 is widely known as a secure encryption standard, it was developed by the NIST to provide the highest level of security for sensitive data (FIPS 197 – NIST, 2023).

In AES-256, the "256" denotes the key length, indicating that a 256-bit key is used for encryption and decryption. Symmetric encryption means that the same key is used for both encryption and decryption processes. This key is randomly generated by a Cryptographically Secure Pseudo-Random Number Generator to ensure the randomness (OWASP – Key Management Cheat Sheet, n.d.). In implementing AES-256 for storing archived files, a unique 256-bit key is generated, which serves as the secret code for encrypting the data. This key is used in the AES encryption algorithm to transform plaintext into cipher text. During decryption, the same 256-bit key is applied to the cipher text to revert it back to plaintext. AES-256 complies with current government regulations, addressing concerns regarding regulatory compliance.

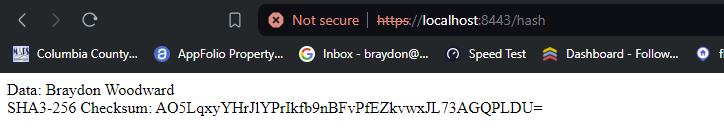
However, it is essential to acknowledge potential risks associated with AES-256, including the vulnerability of storing key and potential performance impacts. While AES-256 offers strong encryption, the security of the system heavily relies on the secrecy and integrity of the encryption key. If the key is compromised or inadequately protected, it could lead to unauthorized access to the encrypted data. Additionally, AES-256 encryption may impose a slight overhead on computational resources, particularly when encrypting or decrypting large volumes of data. To mitigate the risk in storing keys we will implement OWASP’s key management ensuring secure key management practices by encrypting the keys, storing the keys in a cryptographic vault in memory separate from the application, and following all other guidelines (OWASP – Key Management Cheat Sheet, n.d.). Since these are archived files, they will not be decrypted often, and performance issues can be mitigated by encrypting files during system downtime.

In conclusion, our recommendation, AES-256, is endorsed by reputable entities such as NIST and OWASP, offering a solution to the company's security needs. AES-256, with its 256-bit key length, provides strong encryption, ensuring data confidentiality and integrity. Developed by NIST to offer the highest security standards, AES remains a cornerstone in modern encryption algorithms.

## Certificate Generation



## Deploy Cipher



## Secure Communications

A screenshot of a computer

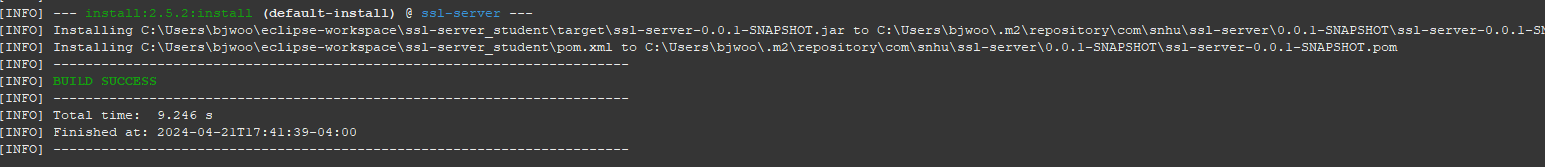
Description automatically generated

I could not get the browser to confirm the sight as secure after configuring the application.properties file. I also imported it into the Trusted Root Certification Authorities (vmware Docs. 2019).

A screenshot of a computer

Description automatically generated

## Secondary Testing



A screenshot of a computer

Description automatically generated

## Functional Testing

A screen shot of a computer screen

Description automatically generatedScreenshot showing the server has been deployed

A screen shot of a computer program

Description automatically generatedScreenshot of refactored code

## Summary

The code has been refactored to address several security concerns and comply with security testing protocols. Here is the process for adding layers of security according to the Vulnerability Assessment Process Flow:

**Input Validation:**

Addressed Vulnerabilities: Injection attacks such as SQL injection and Cross-Site Scripting (XSS) could occur if input data is not properly validated.

Refactoring: Added input validation to the /hash endpoint to ensure that the data parameter is present and not empty. This prevents injection attacks by ensuring that only valid input data is processed.

**Error Handling:**

Addressed Vulnerabilities: Inadequate error handling can lead to information leakage or unexpected behavior, potentially exposing vulnerabilities.

Refactoring: Improved error handling by providing informative error messages for scenarios such as empty input data and exceptions during checksum calculation. This helps in diagnosing and resolving issues while minimizing the risk of exposing sensitive information.

**HTTPS Configuration:**

Addressed Vulnerabilities: Communication over insecure channels can lead to eavesdropping, data tampering, and man-in-the-middle attacks.

Refactoring: To enhance security, we updated the application.properties file to enable HTTPS configuration. By specifying the appropriate server properties, including the SSL certificate details, we ensured secure communication between the client and server.

In summary, the refactoring process has added layers of security to the software application by addressing vulnerabilities related to input validation, error handling, and secure server/client communication.

## Industry Standard Best Practices

In refactoring the code to maintain and enhance the software application's security, several industry-standard best practices for secure coding were applied:

1. Input Validation: Implemented input validation to ensure that user-supplied data is sanitized and validated before processing. This prevents injection attacks such as SQL injection or cross-site scripting (XSS).
2. Error Handling: Enhanced error handling mechanisms to provide informative error messages without leaking sensitive information. Proper error handling helps prevent information disclosure attacks and aids in debugging potential security issues.
3. Secure Hashing: Utilized secure hashing algorithms such as SHA-256 for generating checksums. Secure hashing ensures data integrity and confidentiality, making it challenging for attackers to reverse engineer or tamper with the data.
4. HTTPS Configuration: Enabled HTTPS configuration to encrypt data transmitted between the client and server, thereby safeguarding against eavesdropping and man-in-the-middle attacks. HTTPS ensures that sensitive information remains confidential during transit.
5. Code Readability: Emphasized code readability and maintainability by organizing the codebase, adding meaningful comments, and adhering to coding standards. Well-structured code facilitates easier review for security vulnerabilities.

Applying industry-standard best practices for secure coding is important for the company's overall wellbeing due to several reasons:

1. Reduced Risk of Breaches: By proactively addressing known security vulnerabilities and adhering to best practices, the risk of security breaches and data leaks is significantly reduced. This helps protect your user’s sensitive information, their trust, and your brand reputation.
2. Compliance Requirements: Many industries have regulatory compliance requirements mandating secure coding practices. Adhering to these standards not only ensures legal compliance but also demonstrates the company's commitment to safeguarding user data.
3. Cost Savings: Addressing security vulnerabilities during the development phase is more cost-effective than dealing with security incidents after deployment. Investing in secure coding practices upfront helps minimize the potential financial losses associated with security breaches.
4. Competitive Advantage: Demonstrating a strong commitment to security can differentiate the company from competitors and attract security-conscious customers. It enhances the company's reputation as a trustworthy provider of secure software solutions.