****

# 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** | **10-9-2023** | **Nicholas Kreuziger** | **First Pass** |

## 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

Nicholas Kreuziger

## Algorithm Cipher

Artemis Financial is required to encrypt their archive files in compliance with current government regulations and best security practices. The Gramm-Leach-Bliley Act states that if you are a “financial institution” or engaged in activities “financial in nature” you are required to develop, implement, and maintain an information security program with administrative, technical, and physical safeguards to protect customer information (Federal Trade Commission, 2022). To enhance its technical safeguards, Artemis Financial aims to incorporate encryption for its long-term archived data. The recommended algorithm cipher will be implemented to encrypt files before they are archived for long-term storage. This will be achieved by integrating the cipher into Artemis Financials’ existing security infrastructure, where it will work with the system to automatically encrypt data during the archiving process. It ensures that, even in cases of unauthorized access, the data remains inaccessible without the correct encryption key, thereby fortifying data security.

We recommend using the Advanced Encryption Standard. According to the NIST (National Institute of Standards and Technology), the AES (Advanced Encryption Standard) is approved for protection of electronic data (Evans & Brown, 2001). It is endorsed by the United States Government for protection of sensitive data in all its systems. The AES standard is a result of an open competition organized by NIST , where the Rijndael submission by Daemen and Rijmen was selected by NIST as the winner of the competition in 2000 and standardized in 2001 (Mouha, 2021). Since then, AES has evolved to become a benchmark in the encryption landscape, continuously adapting to meet the sophisticated threats of the digital age. It's important to note that in the current state, AES stands robust, supported by continuous developments and advancements that aim to enhance security while accommodating emerging technologies and increasing computational power. AES encryption is native to all modern 64-bit processors and mobile devices, the security provided through these native implementations is estimated to have generated an economic impact of around $250 billion over the past two decades.

The AES cipher offers options of 128, 192, or 256-bit keys; evaluating your current IT infrastructure will be essential to determine the most suitable choice. While the 256-bit key offers superior security, it comes with its downsides. It is more resource-intensive to encrypt and decrypt, potentially slowing down system processes. Moreover, although AES is generally secure, it is not entirely immune to attacks such as side-channel attacks, which exploit the physical implementation of a cryptosystem. Proper management and regular updates can mitigate such risks. Use of this cipher alone is not enough to ensure security best practice; proper management of the security keys is a crucial detail of this implementation. Despite its high level of security, AES-256 might not always be the best choice due to its resource-intensive nature. In environments where resources are limited or where encryption and decryption speed is a priority, a less secure but faster option like AES-128 might be more appropriate. Additionally, organizations with older IT infrastructures might find it challenging to integrate AES-256 without significant upgrades.

The AES cipher operates primarily through symmetric key encryption, utilizing modes of operation like Cipher Block Chaining (CBC) or Galois/Counter Mode (GCM) to bolster data security. The cipher's hash functions play a crucial role in this, helping to maintain data integrity and authenticate information by verifying that it has not been altered during transmission or storage. Different bit levels, such as 128, 192, or 256, offer varying levels of security, with higher bit levels providing increased security at the cost of more intensive computational resources. These modes define how to apply the cipher to messages of more than one block and are essential in bolstering the security provision of the AES algorithm. A 256-bit key offers large keyspace, making brute-force attacks unfeasible. The Encryption process involves the use of Random Numbers to introduce entropy, making the encryption more secure by ensuring that the ciphertext is not predictable. There are two types of keys in cryptographic ciphers, Symmetric or Non-Symmetric. A symmetric key (like AES-256) utilizes a single key for encryption and decryption. This requires secure key management to ensure key confidentiality. Non-symmetric encryption uses a pair of keys: a public key for encryption and a private key for decryption. It is generally slower than symmetric key encryption but allows the distribution of a public key with the security benefits of using a private key for decryption. Our use case does not benefit from the use of a non-Symmetric key.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screen shot of a computer program

Description automatically generated

## Secure Communications

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

A screenshot of a computer

Description automatically generated

## Secondary Testing

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

**Execution Without Errors:**

A screenshot of a computer program

Description automatically generated

**Dependency Check Report:**

A screenshot of a computer

Description automatically generated

*Original Codebase Static Testing Output*: 

*Refactored Codebase Static Testing Output*: 

## Functional Testing

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

Security Concerns:

* SslServerApplication.java
  + SHA-256 is considered secure for most purposes, however if it’ll be used to store passwords additional measures would have to be taken to make it more robust.
  + The hard-coded ‘data’ variable will be replaced with user input at some point, and more secure means of ingesting the string from a user will have to be taken.
  + /hash endpoint will need to be validated, sanitized, and/or escaped to prevent injection attacks once it accepts user data.
* HttpsConfiguration.java
  + The Tomcat connector is listening on port 8080 for HTTP traffic, but not redirecting it to port 8443. Forcing a redirection to port 8443 ensures all traffic is HTTPS. Default configurations may have been doing this, but adding a command to manually redirect ensures it is happening.
  + Ensure the @Configuration header is only accessible by authorized personnel.

**Code executing without error (Refactored HttpsConfiguration File):**

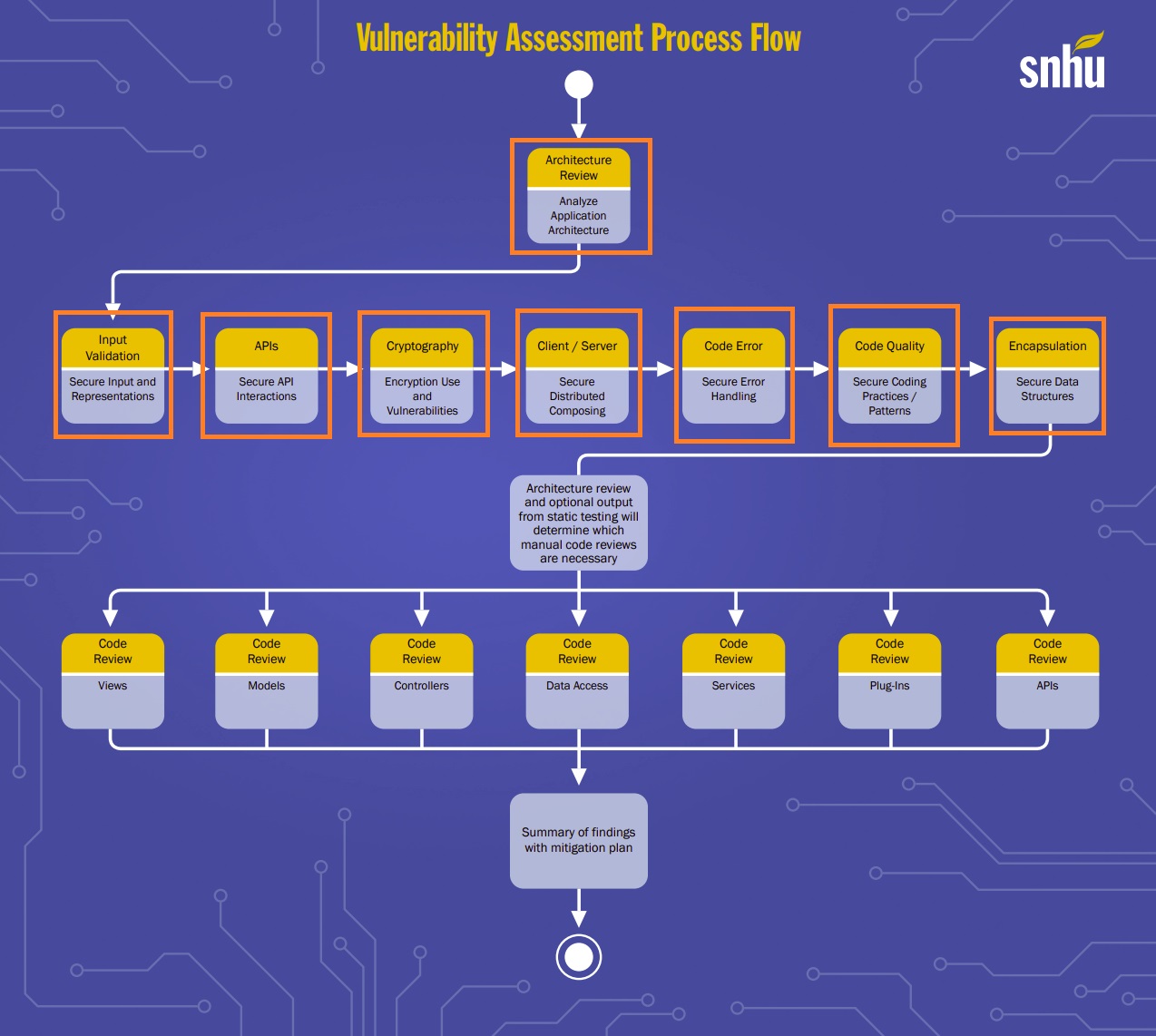
A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary



This report has assessed various areas of security during the refactoring of the code.

* Architecture Review
  + The Code is divided into two sections, adhering to the best principle of a Class per Task. The SSL Server code and the HTTPS Configuration code each serve a separate purpose and will be easier to maintain.
* Input Validation
  + While the code doesn’t have explicit input validation, it uses a static string to create a hash. This report has addressed the additional tasks to complete if it starts accepting user input.
* API
  + The “@RequestMapping(“/hash”)” annotation secures API interactions by mapping HTTP requests to the ‘myHash()’ method.
* Cryptography
  + The code leverages the SHA-256 hash function to create a checksum for a data string. It enhances data integrity by providing a way to validate the content.
* Client/Server
  + The code in HttpsConfiguration.java sets up a server connector for port 8080. It adds a layer of security by guiding the configuration of SSL/TLS, although there's room for improvement, such as redirecting HTTP to HTTPS.
* Code Error
  + The code was manually reviewed to verify that error handling was done in a secure fashion.
* Code Quality
  + The code was manually reviewed to verify separation of tasks and best practices of Object Oriented Programming.
* Encapsulation
  + The ‘bytesToHex’ method is private and only accessible within the ‘ServerController’ class, providing some level of data encapsulation.

## Industry Standard Best Practices

Industry standard best practices are important guidelines to follow when coding against known security vulnerabilities. This project demonstrated industry standard best practices by using static error checking, in combination with Architecture Review, Input Validation, API Security, Cryptography, Client/Server interaction review, Code Review and Encapsulation. While errors and security vulnerabilities are not impossible to avoid, they are a situation developers can work together towards mitigating.

Artemis Financial serves to benefit from continuing application of these secure coding guidelines. As we continue to implement these practices into Secure Dev Ops we will continue to protect against security breaches. One breach can greatly affect our reputation with our clients, continuing to protect against them will only build our clientele in addition to securing the trust and respect of our existing clients.

Citations

Evans, D., & Brown, K. (2001). *FIPS 197 Federal Information Processing Standards Publication Advanced Encryption Standard (AES)*. https://doi.org/10.6028/NIST.FIPS.197-upd1

Mouha, N. (2021). *Review of the Advanced Encryption Standard*. https://doi.org/10.6028/nist.ir.8319

Federal Trade Commission. (2022, April 27). *FTC Safeguards Rule: What Your Business Needs to Know*. Federal Trade Commission. https://www.ftc.gov/business-guidance/resources/ftc-safeguards-rule-what-your-business-needs-know