****

# 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** | **October 15th, 2023** | **Aaron Ciminelli** |  |

## 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

Aaron Ciminelli

## Algorithm Cipher

I recommend using GCM with AES-256 for its established reputation and industry-best practices for strong symmetric encryption. GCM has been thoroughly vetted over decades of worldwide use and provides excellent security with efficient performance. However, the longevity of AES also instills confidence, making it a reliable choice for protecting sensitive information. Ultimately, the optimal balance of security, speed, and interoperability for Artemis Financial is presented by AES-GCM.

1. **Provide a brief, high-level overview of the encryption algorithm cipher.**

Advanced Encryption Standard (AES) is a symmetric encryption algorithm that replaced Data Encryption Standard (DES) as the preferred method for securing electronic data. AES operates on data blocks and comes in various vital lengths, of which 128, 192, and 256 bits are the most common. AES-256, in particular, uses a 256-bit key size, offering a higher security level.

Galois/Counter Mode (GCM) is an authenticated encryption operation that ensures confidentiality and authenticity. When combined, AES-256-GCM provides strong encryption with added authentication, ensuring that the data remains confidential and has not been tampered with during transit.

1. **Discuss the hash functions and bit levels of the cipher.**

AES does not use hash functions; it's a block cipher. However, the GCM mode uses a cryptographic to compute an authentication tag for encrypted data. As the name implies, the bit level for AES-256 is 256 bits. This means it uses a 256-bit key to encrypt and decrypt information. This longer key length offers a larger keyspace, making brute-force attacks computationally infeasible with current technology.

1. **Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.**

In encryption, especially with AES-GCM, random numbers are crucial for generating unique initialization vectors (IV) for each encryption operation. This ensures that the ciphertext will differ even when the same data is encrypted multiple times, increasing security. The symmetric encryption algorithm uses the same key for both encryption and decryption. This differs from asymmetric (or non-symmetric) encryption, where a pair of keys is used: one for encryption (public key) and one for decryption (private key). Due to its symmetric nature, key management becomes paramount. The key must remain secret and protected, and it's essential to distribute it to intended recipients without exposure.

1. **Describe the history and current state of encryption algorithms.**The history of encryption algorithms dates back to ancient civilizations but in the context of modern cryptography. Introduced in the 1970s, DES was the standard symmetric encryption algorithm. However, by the late 1990s, it was deemed insecure due to its short key length, making it vulnerable to brute-force attacks. 2001 AES was established as the new standard by the U.S. National Institute of Standards and Technology (NIST) after a public competition. It was designed to address the vulnerabilities of DES and has since become the de facto standard for symmetric encryption worldwide. AES remains robust and widely accepted, with AES-256 often recommended for high-security situations. Other encryption algorithms and modes of operation have emerged, but AES, especially in GCM mode, remains a popular choice for ensuring data confidentiality and integrity.

In conclusion, encryption algorithms are vital in ensuring data security, privacy, and confidentiality. AES with GCM, in particular, is a widely accepted and trusted encryption standard for symmetric encryption. It offers a high level of security, is computationally infeasible to crack with current technology, and provides efficient performance. AES-GCM's ability to offer confidentiality and authenticity makes it a popular choice for securing sensitive information for businesses like Artemis Financial. Encryption algorithms will continue to evolve as new technologies emerge, but AES-GCM remains the optimal choice for businesses that require high-level data security.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen shot of a black screen

Description automatically generated

A screenshot of a certificate

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer program

Description automatically generated

A white background with black text

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

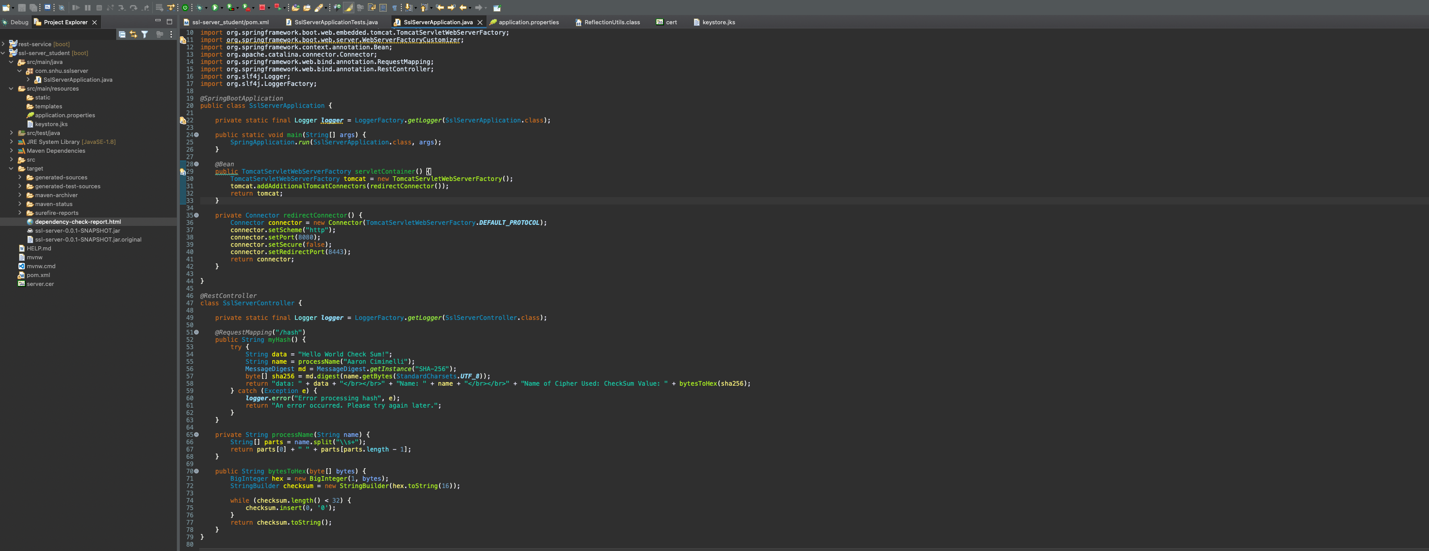
A screenshot of a computer program

Description automatically generated  
  
A screenshot of a document

Description automatically generated

## Functional Testing

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



A screenshot of a document

Description automatically generated

## Summary

The refactored code has improved clarity by isolating the logic behind processing the name into a separate method called processName. This makes the purpose of the logic more transparent, which is to format the name consistently as "FirstName LastName," regardless of any middle names or additional spaces.

Enhanced error handling has been introduced in the refactored version. Instead of only catching the NoSuchAlgorithmException, we catch a more general Exception. This ensures that any unexpected issues are handled gracefully.

The refactored code follows security-focused error responses. Instead of exposing internal system details or configurations by returning the raw exception message to the client, the code logs the detailed error server-side. It only returns a generic error message to the client. This approach minimizes the exposure of internal system details, adhering to best practices for security.

The refactored code uses logging mechanisms introduced through SLF4J. Proper logging plays a critical role in secure software development. It helps monitor diagnose issues, and even forensic analysis post a security incident.

A layered security approach is crucial when addressing the vulnerability assessment process flow. Input validation, configuration management, secure communication, error handling, logging and monitoring, and regular updates and patching are all essential layers of security.

Secure coding practices include validating and sanitizing inputs, avoiding exposing sensitive details in error messages, and handling errors gracefully without crashing the application. Infrastructure security ensures that the underlying systems, servers, and databases are secure. This includes setting up firewalls, using intrusion detection systems, and regularly patching and updating software.

Secure communications are achieved by using encrypted communication protocols like HTTPS to protect data in transit. Authentication and authorization mechanisms should be implemented, and users should only have access to the data and functionalities they are authorized to use.

Regular audits and testing are critical to identifying vulnerabilities. This includes automated testing, manual code reviews, and third-party penetration testing.

In the context of the provided software, the primary focus was on secure communications (using HTTPS), secure error handling, and secure coding practices. The Vulnerability Assessment Process Flow Diagram serves as a roadmap to ensure all potential vulnerabilities are considered and addressed during development. By following this roadmap and continuously updating it based on the evolving threat landscape, we can develop functional and secure software.

## Industry Standard Best Practices

Ensuring secure coding practices is crucial in protecting software against common vulnerabilities. Even if the current code does not handle external user input, validating and sanitizing all inputs in more complex applications is essential. This approach safeguards against input-based attacks, such as SQL injection and XSS.

The refactored code ensures that raw errors that reveal system internals are not exposed to end-users. Instead, a general message is presented to users, while detailed errors are logged internally to prevent potential information disclosure vulnerabilities. By enforcing HTTPS, all communication between users and the server is encrypted, protecting data in transit from eavesdropping and man-in-the-middle attacks.

Introducing logging mechanisms using SLF4J helps monitor and diagnose issues, aiding preventive measures and post-incident analysis. It is best not to hardcode sensitive details such as passwords but instead manage them securely through environment variables or secret management tools.

The application's security can be maintained by adhering to industry-standard best practices in several ways. Periodically reviewing the code, especially after updates or adding new features, ensures no new vulnerabilities are introduced as the threat landscape evolves. Staying updated with the latest security research, vulnerabilities, and patches helps keep the application secure against emerging threats. Regularly updating libraries and dependencies protects against known vulnerabilities in third-party software.

Customers and partners can trust the company by ensuring the software is secure and reliable. A single security breach can lead to significant reputational damage. Data breaches and security incidents can lead to substantial financial costs, both direct and indirect. Many industries have regulations dictating specific security standards. By following best practices, companies can remain compliant and avoid penalties. A secure application guarantees uninterrupted business operations, as incidents can lead to downtime, data loss, or permanent closure. A strong security posture can be a significant market differentiator in an era of increasing data breaches.

In conclusion, applying industry-standard best practices for secure coding is a critical business necessity. It provides a foundation for trust and ensures long-term success and resilience in an increasingly digital world.

**Resources:**

Certitude Security. (n.d.). *Four steps for Cyber Attack Prevention and Security*. https://www.certitudesecurity.com/blog/borderless-network-security/4-steps-for-cyberattack-prevention-and-security/

Fluent, S. (2023, May 2). *Nine key reasons why data security is crucial in Mobile apps*. DejaOffice Blog. https://www.dejaoffice.com/blog/2023/05/02/9-key-reasons-why-data-security-is-crucial-in-mobile-apps/

MozDevNet. (n.d.). *HTTP*. MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP

Muller, A., & Meucci, M. (n.d.). *Owasp Web Security Testing Guide*. OWASP Web Security Testing Guide | OWASP Foundation. https://owasp.org/www-project-web-security-testing-guide/

NIST. (2023, October 10). *Cybersecurity framework*. NIST. https://www.nist.gov/cyberframework

*Secure Coding Guidelines for Java SE*. Secure coding guidelines for Java SE. (n.d.). https://www.oracle.com/java/technologies/javase/seccodeguide.html

Stephen, F. (2018). *Python - secure coding guidelines*. Medium. https://felsen88.medium.com/python-secure-coding-guidelines-73c7ce1db86c

Van Der Stock, A., Glas, B., Smithline, N., & Gilgler, T. (2021). *Owasp Top Ten*. OWASP Top Ten | OWASP Foundation. https://owasp.org/www-project-top-ten/