****

# 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** | **04/15/2024** | **Joel De Alba** |  |

## 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

Joel De Alba

## Algorithm Cipher

As a developer entrusted with enhancing Artemis Financial's security measures, I've recommended the adoption of the Advanced Encryption Standard (AES) to bolster their encryption protocols. AES works with fixed-length data blocks and offers key lengths of 128, 192, or 256 bits. Additionally, AES has superseded the earlier Data Encryption Standard (DES) due to its greater security features and computational efficiency. AES does not directly incorporate hash functions; however, it relies on secure hash functions like SHA-256 for additional security measures. SHA-256 is a cryptographic hash function that produces a 256-bit (32-byte) hash value. It ensures data integrity and authentication by generating a unique checksum for each input.

AES employs symmetric key encryption, meaning the same key is used for both encryption and decryption. This approach offers simplicity, speed, and efficiency. Random numbers play a crucial role in generating encryption keys, enhancing the security of AES. Symmetric key encryption ensures confidentiality and integrity of data during transmission and storage. This combined approach benefits the specific application Artemis Financial by providing a comprehensive security framework. It addresses both confidentiality and integrity requirements, ensuring that sensitive financial data remains protected from unauthorized access and tampering.

The history of cryptography traces back thousands of years, from ancient methods of secret communication to the modern cryptographic systems used today. One significant milestone occurred during World War II with the development of the Enigma machine, a cryptographic device used by the Germans to encode messages. Following the war, cryptography evolved rapidly with the advent of computer technology. In 1977, the Data Encryption Standard (DES) was introduced as a standard encryption algorithm by the U.S. government. However, as computing power increased, DES became vulnerable to brute-force attacks. This led to the development of the Advanced Encryption Standard (AES) in 2001, chosen through an extensive selection process for its superior security and efficiency. AES has since become the de facto standard for symmetric encryption, widely adopted across industries and applications. Concurrently, cryptographic hash functions like SHA (Secure Hash Algorithm) were developed to ensure data integrity and authentication. SHA-256, introduced in 2001, is a widely used cryptographic hash function known for its robustness and resistance to collision attacks. Together, AES and SHA-256 represent the culmination of centuries of cryptographic research and innovation, providing the foundation for secure communication and data protection in the digital age.

## Certificate Generation

Image 1 Shows Certificate Generation through Java and Terminal

A screenshot of a computer

Description automatically generated

## Deploy Cipher

A screenshot of a computer

Description automatically generated

## Secure Communications

When setting up a local server environment using a self-signed certificate authority (CA), it's essential to understand that self-signed certificates are not inherently trusted by web browsers. This means that when you try to establish an HTTPS connection to your server, web browsers will typically display a security warning indicating that the connection may not be secure.

This warning occurs because web browsers rely on trusted certificate authorities to validate SSL certificates. Since self-signed certificates are not issued by a trusted CA, browsers cannot verify their authenticity.

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.

Running Spring Application

A screenshot of a computer

Description automatically generated

ServerApplication.java SHA-256 Hash Gen

A screenshot of a computer program

Description automatically generated

ServerApplicationTests.java

**A screen shot of a computer program

Description automatically generated**

Application.Properties

A screenshot of a computer program

Description automatically generated

## Functional Testing

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

Despite incorporating the latest releases and updating CVEs in POM.xml, certain CVE vulnerabilities persist. This underscores the potential for false negatives, either due to limitations in other packages or the emergence of new vulnerabilities. Additionally, conflicts between incompatible dependencies can exacerbate the issue. Moreover, some dependencies can only be updated to a certain extent before causing application failures. Addressing these challenges may necessitate significant changes to Java versions, runtime environments, and other aspects to align with current best practices.

A screenshot of a computer

Description automatically generated

## Summary

Imported and reviewed dependencies may have triggered false positives in security testing tools. This process involved meticulously updating each dependency to the latest compatible version, ensuring compatibility with Java 1.8, and resolving any conflicts or errors that arose during the process.

Refactoring the code involved identifying and addressing vulnerabilities highlighted in the Vulnerability Assessment Process Flow Diagram. This included implementing OWASP dependency check to identify and mitigate security risks posed by outdated or vulnerable dependencies. Additionally, layers of security were added by updating Spring Boot packages to the latest releases, which often include security patches and enhancements.

## Industry Standard Best Practices

By adhering to best practices for maintaining software security, such as regularly updating dependencies, conducting thorough security testing, and staying informed about the latest security threats, we ensure the ongoing security of the software application. Communicating these practices to customers emphasizes the importance of proactive security measures in safeguarding against potential threats and vulnerabilities. Ultimately, prioritizing security not only protects Artemis Financial’s assets and data but also fosters trust and confidence among users and stakeholders.

**Sources**

Keytool-Key and Certificate Management Tool. (n.d.). https://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html

*How to Create a Self Signed Certificate using Java Keytool*. (n.d.). <https://www.sslshopper.com/article-how-to-create-a-self-signed-certificate-using-java-keytool.html>

*Java Security Standard Algorithm Names*. (n.d.-b). https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names

Manico, J., & Detlefsen, A. (n.d.-d). Iron-Clad java. O’Reilly Online Learning. <https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/?sso_link=yes&sso_link_from=SNHU>

Mend.io. (2024b, April 17). OWASP Dependency Check: How does it work? | Mend. Mend. <https://www.mend.io/blog/owasp-dependency-check/>

Hoestje, J. (2018b, June 14). OWASP Dependency Check for vulnerability reporting. Keyhole Software. <https://keyholesoftware.com/owasp-dependency-check-for-vulnerability-reporting/>

Mark Fleming. (2012c, July 19). Five minute tutorial : Eclipse Keytool Plugin [Video]. YouTube. <https://www.youtube.com/watch?v=IaXE2FUENFI>

CVE - Search CVE list. (n.d.-b). <https://cve.mitre.org/cve/search_cve_list.html>

*NVD - Search and Statistics*. (n.d.). https://nvd.nist.gov/vuln/search

dependency-check – Suppressing False Positives. (n.d.-b). <https://jeremylong.github.io/DependencyCheck/general/suppression.html>

Silva, J. E. (2003). An overview of cryptographic hash functions and their uses. GIAC, 6.

Cryptography Hash functions. (n.d.). <https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htm>

Wahome. (2022, January 6). Cryptographic hash functions - Wahome - medium. Medium. <https://kwahome.medium.com/cryptographic-hash-functions-2cc49723ea06>