****

# 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** | **08/18/2024** | **Joshua Kronenberg** | **Secure Software Report for Artemis Financial** |

## 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

Joshua Kronenberg

## Algorithm Cipher

Recommend an appropriate encryption algorithm cipher to deploy, given the security vulnerabilities, and justify your reasoning. Review the scenario and the supporting materials to support your recommendation. In your practices for secure software report, be certain to address the following actions:

1. Provide a brief, high-level overview of the encryption algorithm cipher.
2. Discuss the hash functions and bit levels of the cipher.
3. Explain the use of random numbers, symmetric versus non-symmetric keys, and so on.
4. Describe the history and current state of encryption algorithms.

SHA-256 as the hash function and AES as the algorithm cipher would be suitable for the client's requirements. Secure messaging apps, financial institutions, and even the US Federal government—which authorized AES as the country's federal standard in May 2002—all make extensive use of AES. Unlike non-symmetric keys, which employ separate keys for both encryption and decryption, AES employs symmetric keys for both operations. It can employ bit keys of 128, 192, or 256.

A hash function with a 256-bit digest is called SHA-256. To put it another way, there are 2256, or 1.1579209e+77, possible hash combinations. At this point, a number that big is practically immune to brute force attacks and extremely unlikely to cause collisions. Furthermore, NIST (Computer Security Division) recognizes it as a standard encryption cipher. This indicates that not only is it regarded as a safe option, but developers have access to resources that can be used to test its effectiveness and setup.

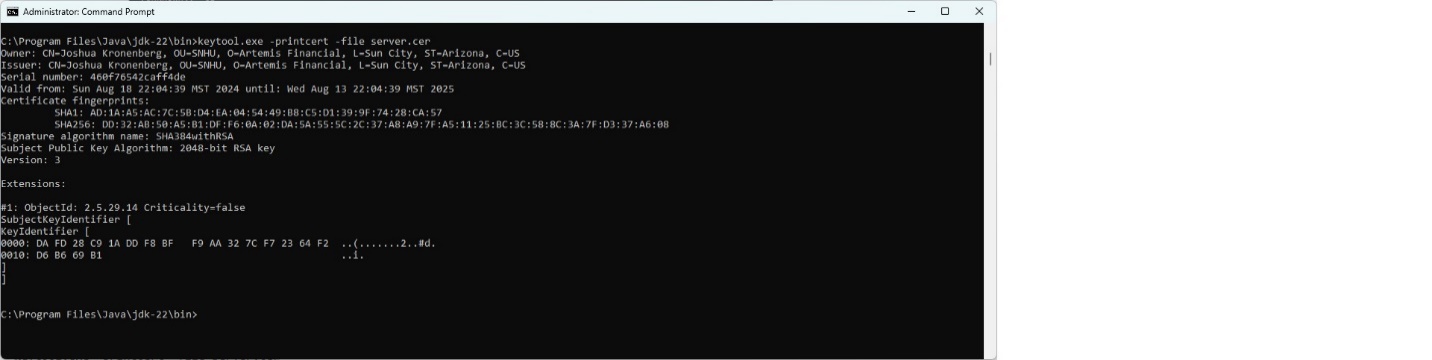
The application's needs determine whether to use symmetric or non-symmetric keys. In contrast to non-symmetric keys, which consist of public and private keys that are known to both the client and the public, symmetric keys are essentially shared between the server and the client. These keys are used to encrypt data, which can only be decrypted with the right key. A drawback of this is that, since there is no way to decrypt encrypted data without the key, any encrypted data that is lost is also essentially lost. Using random number generators, transactions can be given a unique identifier that may be used to identify specific events, such as data transfers or communications.

The US used to consider DES to be the standard for encryption. Since then, it has been broken several times, proving that it is not the safest choice. The fact that its key is only 56 bits—much fewer than AES's—could be one reason for this. AES has taken the place of DES as the accepted standard for data encryption due to the security breaches. AES is currently among the safest solutions accessible.

## Certificate Generation

Insert a screenshot below of the CER file.

The creation of the CER file:



Showing the CER file is installed:



## Deploy Cipher

Insert a screenshot below of the checksum verification.



## 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.

The refactored code executed without errors:

A screenshot of a computer

Description automatically generated

The dependency-check report:

A screenshot of a computer

Description automatically generated

## Functional Testing

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

A screenshot of a computer

Description automatically generated

## Summary

Cryptography was the first area of security that was addressed by refactoring the code. By reworking to incorporate a hash function, the data was encrypted. The RESTful application was developed to improve the security of the API. To enable more secure data transfer, a certificate was added to enhance client/server security. Using a try, catch ensured that safe coding procedures were carried out. Finally, by guarding against known vulnerabilities, using the most recent versions of the spring-boot-parent and tomcat dependencies enhances the application's security.

Since communications between Artemis Financial's systems and their clients will be more secure, hashing the information, utilizing an SSL connection, and developing a RESTful application creates a safe means of communication that adds value. Additionally, the program is free of known vulnerabilities that could result in security flaws because all dependency versions have been updated to the most recent versions that are accessible.

Using the Dependency-Check tool, dependencies used in the application's construction should be examined for any new vulnerabilities to maintain its security. When vulnerabilities are discovered, the dependencies should be updated to the latest versions.

## Industry Standard Best Practices

Reiterating throughout the development process is a crucial practice for industry standard best practices. It was crucial to this project's iterations to keep security in mind. In terms of industry standards, updating the Maven version made sense because it's always advisable to use the newest libraries, extensions, and frameworks. Moreover, a DevSecOps developer can prioritize focusing on the most certain and critical vulnerabilities by suppressing false positives in the dependency check.

Industry best practices are important for the general health of a company. By ensuring that dependencies are up to date, these procedures help the organization defend against the most prevalent and recent vulnerabilities. They help the business write code that is safe from injection attacks and maintains security. Furthermore, by using these procedures, the business can authenticate its own identity and that of the entities it interacts with.