****

# 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** | **[Date]** | **[Your Name]** |  |

## 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

James Emery

## Algorithm Cipher

The recommendation for an encryption algorithm is being made by Artemis Financial in order to encrypt long-term archive material. These files should be encrypted such that, even in the unlikely event that they were stolen, they would be worthless. This is because we should presume that the most likely attack vector for these data will be malicious actors gaining access to them in some way. There is no need to use Asymmetric\* keys because the files will not be transferred. Additionally, since these files will be preserved for a long time, there is no need to encrypt them right away. For this reason, I advise encrypting these data using the SHA-256 cipher method and 256-bit keys. Since SHA-256 encryption offers the highest level of bitwise encryption (256-bit refers to the number of bits in the length of the key), it is the most secure default choice accessible inside all standard installations of Java. More bits translate into more key combinations, which increases the difficulty of brute-forcing the key and reduces the likelihood of collisions. \* Symmetrical\* encryption keys are also used by SHA-256. Additionally, the SHA-256 method effectively utilizes Java's random number generator to guarantee the highest level of security for every encrypted file. The cipher may safely provide a non-reversible checksum that nevertheless confirms the legitimacy of the file or message by using random numbers.

## Certificate Generation

Insert a screenshot below of the CER file.

[Insert screenshots here.]

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

Description automatically generated

## Secure Communications

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

[Insert screenshots here.]

## Secondary Testing

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

A screenshot of a computer

Description automatically generated

## Summary

Any organization must prioritize transaction and communication security, but financial institutions should take particular care to ensure this. When reviewing the Vulnerability Assessment Process Flow Diagram, code quality is one of the first—and, in my opinion, most crucial—points to consider. Of course, there are other, more important aspects of security, but a strong foundation is necessary for any construction, and in the case of a house, that base is the quality of the code. Writing well-written code guarantees that it will be easily readable by others who may need to review it for updates or fixes. Though the team who generated the code might not be the one maintaining it, correct indentation, the use of comments, unambiguous notations, and ensuring that the code communicates between developers are all important. Verifying every input into the system comes next. In this instance, the specific refactoring would have involved adding a checksum for input validation and a self-signed certificate that permits Artemis Application Program Interfaces (API) are addressed by the financial use of the HTTPS protocol. Together with changing the spring-boot version, this helps secure the inputs and increase the code's security against serious threats.

## Industry Standard Best Practices

One of the finest methods for preserving the security of a system nowadays is to rigorously and regularly review the code for vulnerabilities. particularly following the addition of any new features to the system, any modifications to the existing code, and carefully before deploying those changes. In the event that vulnerabilities are discovered or discovered when developing new or updated system functions, the development team should endeavor to appropriately remove those vulnerabilities. Nonetheless, there's a potential that certain vulnerabilities won't have fixes available when they're coding. In this instance, the development team should ascertain whether those vulnerabilities could affect any of the system's several functionalities, possibly in consultation with the customer.