****

# 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** | **[8/15/2025]** | **[Paul Garner]** | **Starting the document** |

## 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

Paul R Garner

## Algorithm Cipher

For Artemis Financial, I would have recommended using AES 256 with GCM as the encryption standard. It’s basically the gold standard for symmetric encryption fast, secure, and approved by NIST. AES 256 offers strong protection against brute-force attacks, and GCM mode gives you both encryption and authentication, meaning it keeps the data secret and makes sure no one has tampered with it. This setup uses a single key for both encryption and decryption, which is efficient for large files like long term archives.

## Certificate Generation

(Could not generate due to code issues)

If the code was functional, this is where I would have generated a self signed certificate using Java’s keytool. This certificate would act as a digital ID for secure communications. In a working setup, I’d store the CER file in the project resources and configure the application to use it for HTTPS.

## Deploy Cipher

(Could not run deployment due to project build issues)

Normally, I’d integrate AES 256 GCM encryption into the application’s file handling process, along with checksum verification. The checksum ensures that if a file is altered either by accident or maliciously it will be detected before the file is used.

## Secure Communications

(Could not test HTTPS due to project errors)

Once the certificate is in place, the application would be updated to use HTTPS instead of HTTP. If successful, opening the app in a browser would show the lock icon next to the URL, proving that the data in transit is encrypted and secure.

## Secondary Testing

I couldn’t get the project to run successfully, so I wasn’t able to produce a fresh dependency check report. If the code had built properly, I would have run the OWASP Dependency Check plugin in Maven to identify outdated or vulnerable dependencies, then updated them to secure versions.

## Functional Testing

Again, since the project wouldn’t build, I couldn’t perform a real functional test. Normally, I’d run the application end to end after implementing encryption and HTTPS, making sure all features still work and that the security updates didn’t break anything.

## Summary

While I couldn’t complete the practical work due to build issues in the provided code, the plan was clear: strengthen encryption with AES 256 GCM, use secure HTTPS with a self signed certificate, and run a dependency check to make sure there were no known vulnerabilities. These steps would directly improve Artemis Financial’s data protection.

## Industry Standard Best Practices

* Using strong, modern encryption algorithms (AES 256 GCM).
* Enforcing HTTPS for all network communications.
* Checking dependencies for vulnerabilities before deploying.
* Using checksums to detect tampered files.

These are the kinds of steps that not only align with industry security standards but also build customer trust by keeping sensitive financial data safe.