****

# 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** | **August 17, 2025** | **Christopher Prempeh** |  |

## 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

Christopher Prempeh

## Algorithm Cipher

For this project, I’m recommending AES (Advanced Encryption Standard). It’s a symmetric encryption algorithm that’s fast, reliable, and used pretty much everywhere from government systems to financial apps. With key sizes of 128, 192, or 256 bits, AES-256 is the strongest option and is the one you’d go with if you’re serious about locking things down.

Since AES is symmetric, the same key is used to encrypt and decrypt the data, which makes it faster than asymmetric methods like RSA. That said, you’ve got to make sure the key is kept secure, because if someone gets access to it, it’s game over.

AES also relies on secure random numbers when generating keys or initialization vectors (IVs), which helps protect against brute-force or replay attacks. It’s been around for years, tested like crazy, and is still the go-to choice for secure data encryption today.

## Certificate Generation

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

A screenshot of a computer

AI-generated content may be incorrect.

## Secure Communications

A screenshot of a computer screen

AI-generated content may be incorrect.

## Secondary Testing

**A screenshot of a computer

AI-generated content may be incorrect.**

## Functional Testing

A screenshot of a computer

AI-generated content may be incorrect.

## Summary

I refactored the original application to improve its security using layered practices. First, I added AES encryption to support secure data handling. Then I generated a self-signed certificate with Java Keytool and configured Spring Boot to run over HTTPS using that certificate. I also created a hash endpoint that calculates SHA-256 checksums to verify file integrity during transmission.

Once the app was set up, I tested it in both HTTP and HTTPS modes, verified the checksum logic worked as expected, and confirmed the app ran with no errors. I wrapped up by running the OWASP dependency check plugin, which flagged any vulnerable libraries. This gave me a full picture of the system’s current security state and what could be improved moving forward.

## Industry Standard Best Practices

To keep this app secure, I used industry standards like:

* AES encryption to handle sensitive data securely
* SHA-256 hashing to verify file integrity
* HTTPS configuration with a self-signed certificate to encrypt communication
* Dependency-check static analysis to flag known vulnerabilities in libraries

These steps align with secure software development best practices, like defense in depth, using strong encryption, and regularly scanning for vulnerabilities.

Applying these standards helps protect user data, limits the risk of data breaches, and builds trust with the client. Artemis Financial works in a field where security matters a lot, and this setup helps lay a strong foundation going forward.