****

# 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** | **10/19/2025** | **Cody Newman** |  |

## 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

Cody Newman

## Algorithm Cipher

The algorithm cipher that I recommend is the Advanced Encryption Standard, known as the AES. I specifically recommend the version that uses 256-bit keys, as this version offers the highest level of security and is commonly implemented by financial institutions and government agencies, and is compliant with U.S. regulatory standards.

AES is a block cipher that uses symmetric keys, meaning that it uses sequences of bits at a fixed length and uses the same key both to encrypt and decrypt data. It offers three different bit levels, each with their own strengths and weaknesses. At the lower end, 128-bit keys offer a moderate level of security but are processed very quickly. This version is commonly used for operations such as payment processing. 256-bit keys are slower to process, but are so complex that the most advanced super-computers on Earth would take an infeasible amount of time to crack them. It’s also worth noting that AES has been extensively studied and tested over the course of nearly 25 years since its introduction, and no significant practical weaknesses have been found. This makes AES a trusted standard when it comes to storing highly sensitive data at rest. AES was originally developed in response to older algorithms being declared insufficient to guard against modern technologies, and the NIST held a competition to select an algorithm for AES from a number of finalists. Although it has remained the gold-standard in encryption since its introduction, AES and similar algorithms may soon be vulnerable to quantum computing, which has the potential to execute brute force attacks (trying every possible combination) at speeds that are previously un-heard of.

## Certificate Generation

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screen shot of a computer

AI-generated content may be incorrect.

## Secure Communications

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

A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

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

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

## Functional Testing

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

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

## Summary

To refactor this code, I implemented a secure REST controller that generates a checksum to verify that application data has not been altered. I also implemented a security certificate that verifies the received response is from the authentic source of the page. The main areas of security that these changes target are:

* APIs: The server uses a REST controller to communicate with the client
* Cryptography: The app uses an encrypted checksum and certificate to verify authenticity to the client

## Industry Standard Best Practices

I maintained the current security of the app by not introducing any new security vulnerabilities in my refactored code or added dependencies, as verified by the dependency check report. I followed the industry-best practices by using a REST API to handle client/server communication and using appropriate error handling in case the checksum algorithm is not retrievable for any reason.