****

# 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** | **26 April 2024** | **Tyler Lowe** |  |

## 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

[Insert your name here.]

## Algorithm Cipher

I chose the Advanced Encryption Standard, or AES, as our go-to encryption algorithm. It’s a top choice because it's not only strong but also efficient, making it perfect for securing sensitive financial data. Unlike hashing functions, AES is a symmetric key cipher, meaning it uses the same key for both encryption and decryption, which enhances both security and speed. AES can operate at 128, 192, or 256 bits, allowing us to adjust the security level based on our needs without compromising performance. Key generation in AES involves creating secure random numbers to ensure the robustness of the encryption process.

AES has been the gold standard in security since it was established as a replacement for DES (Data Encryption Standard) by the U.S. National Institute of Standards and Technology (NIST) in 2001. This adoption followed a meticulous review process, prompted by the vulnerabilities discovered in DES. Today, AES remains widely adopted in various security protocols, including SSL/TLS for secure web browsing, underscoring its robustness and trustworthiness in the face of evolving digital threats. By incorporating AES, we’re ensuring that our application is tightly secured against potential threats, keeping our client data safe and sound.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen with white text

Description automatically generated

## Deploy Cipher

A screenshot of a computer

Description automatically generated

## 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.

A computer screen shot of a program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

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

Ran the app and it worked perfectly – the checksum came out just right every time, no errors.

Dealing with Issues

If something did pop up, my plan was to:

Figure out where it went wrong.

Make the necessary fixes.

Test it again.

Keep at it until all was clear.

Screenshots

Added are the shots that show the code’s doing its job – no bugs, and the checksum output looks good.

A screen shot of a computer program

Description automatically generated

## Summary

I revamped the code to beef up the security. Specifically, I hooked in SHA-256 for hashing, making sure the data stays the same from A to B. Checked out the Vulnerability Assessment Process Flow Diagram, and I zeroed in on secure coding practices and encryption use.

For layering up security, I made sure we’re using up-to-date and strong hash functions. All about keeping our clients' data tight and out of the wrong hands.

## Industry Standard Best Practices

I stuck to the industry playbook to keep our app secure. I made sure the code was clean and up to scratch by using tools to weed out any bad code practices that could lead to security gaps. Keeping everything updated, especially the libraries and frameworks, was key.

Applying these best practices isn’t just about dodging security slip-ups; it’s about trust. When clients see we do things by the book, it tells them their data’s safe with us, and that’s solid gold for our reputation.