****

# 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** | **9/24/23** | **Brittney miller** |  |

## Client



## Instructions

Submit these completed practices for a 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 have them, insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for detailed instructions about each template section.

## Developer

Brittney Miller

## Algorithm Cipher

I chose to use the AES\_256 cipher for this because of its variety in crucial size. This is also one of the best standards in place to use for ciphers. I chose the 256-bit since it is the hardest for attackers to hack into and crack. It allows for encrypted data as needed and delivers keys to clients.

AES 256 encrypts up to 256 bits, which can be nearly impossible to break since the number of different values that can be created is so large. That is why I chose the 256-bit for the highest level of security.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

My macOS would not open the local host page for this screenshot and the next one. I messed up my command code as I was moving quickly and didn’t read the directions. I tried looking up help and everything but couldn’t open it properly.

## Secure Communications

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

Same as the previous question.

## Secondary Testing

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

A screenshot of a computer program

Description automatically generated

## Functional Testing

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

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

The code will be attached in a zip drive.

In the code, we made changes, covered the APIs, cryptography, and client/server, and checked the code quality to ensure security. We first made a self-signed certificate and generated keys, which I messed up, which is why I struggled with this so much. I used the AES\_256 cipher for the most secure connection, which created the most keys, making it nearly impossible to crack. This ensures application security when sending and receiving communications. This will ensure that any data or information sent internally and externally is protected from hackers or people not meant to see the information. Security breaches that happen with businesses can ruin potential future clients and hurt trust in current clients and how their data is being protected and kept safe. This can also cause financial stress on the company as it will be seen as unsafe, and information is unprotected against hackers. To have the highest level of security, maintaining and updating security features is critical to stay up to date with the most current software versions, so we close any security gaps.

## Industry Standard Best Practices

Best practices are continuously checking in on current security to ensure all software is up to date and using recent versions, as there are sometimes security fixes in updated versions. Also, check code quality and vulnerability to ensure there are no areas of weakness in which hackers can easily make their way through the code to steal valuable information. Check through any updated code or new developments to ensure we secure the code and program properly before it is launched and goes live.