****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.k17ogp2b3dwa)

[**Client 3**](#_heading=h.ec07hoyft89b)

[**Instructions 3**](#_heading=h.cu7n25limpsb)

[**Developer 4**](#_heading=h.m4lbg3o4voeo)

[**1. Algorithm Cipher 4**](#_heading=h.y0ir0qcnyvo1)

[**2. Certificate Generation 4**](#_heading=h.doo29jse3lub)

[**3. Deploy Cipher 4**](#_heading=h.3gn1sfq3ozsg)

[**4. Secure Communications 4**](#_heading=h.gsgam04z7ofb)

[**5. Secondary Testing**](#_heading=h.dc62vf4odaj2) **5**

[**6. Functional Testing**](#_heading=h.iwlazuc5bys4) **5**

[**7. Summary**](#_heading=h.cf9cwlt7vq5m) **5**

[**8. Industry Standard Best Practices**](#_heading=h.2c7sxva2gel2) **5**

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **08/24/2025** | **Casey Hair** |  |

## 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

Casey Hair

## Algorithm Cipher

For our Encryption Algorithm, we will be using AES. This is widely considered one of the most robust encryption algorithms and is used by many companies, as well as government agencies. In short, AES takes our data, called a “string of text”, and turns it into a different, unreadable, “string of text.” This works with what is called an “encryption key”. We use this “key” to encrypt the data, and we use the same key to decrypt the data. We will be the sole owners of this key, and secure storage of this is paramount to the security of the entire system. Specifically, we will use a 256-bit key, which is 256 ones and zeroes. This will be used to securly store user data.

There is also another algorithmic cypher to be used, and that is SHA-256 (Secure Hasing Algorithm). We use this to ensure that our users are downloading the correct software. This is what is know as a “hashing” algorithm. Basically, we take a “string of text” which is our entire program, an put that into this algorithm. Out of the algorithm, we get a 256-bit string of text and make that publicly available. When a user downloads our program, they can run it through the same algorithm, and as long as the program they downloaded is exactly what we wrote, they will get the same 256-bit string of text. Any small change will greatly affect the output, and so if they download a version that has been tampered with, the hash will not line up.

Cryptography has changed a lot through the years. The early algorithms worked because computers were not as fast and effective as modern computers. With modern hardware like high-tech GPUs, we can now crack the old algorithms, and so new ones have been made. This process of the continuation of technology is still going on. With current efforts on perfecting the hardware of quantum computers, we are faced with a similar problem. Once we reach a point where quantum computers are stable and effective, the modern algorithms for encryption will no longer be safe to use. As such, there are efforts being made to create “post quantum encryption algorithms”, which will be designed to be robust even against the brute force of quantum computers.

## Certificate Generation

Certificates are documents that ensure authenticity when connecting to a server. The best method for this is to use a third-party “Certificate Authority (CA)” to handle this. The reason is, if our server is compromised, and we are the ones responsible for generating the certificates, the generation of said certificates could also be compromised. This would give the user a false sense of security.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

The checksum is the value generated from the SHA-256 algorithm. This is the value that we will make public, and the user should run the downloaded program through the algorithm to ensure that it matches. This gives them confidence that the software has not been tampered with.

## Secure Communications

For secure communications with our clients, we would use the HTTPS protocol, which encrypts data while in transport between the server and the client side. This creates what is known as an “encryption tunnel”, which a secure communication line between two entities. This is how our customers can submit and view sensitive information between the application and our server.

## Secondary Testing

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

[Insert screenshots here.]

## Functional Testing

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

[Insert screenshots here.]

## Summary

In summary, the final product uses different types of security measures to ensure the utmost level of security for our clients. AES is used to securly store sensitive user data, SHA-256 is used to validate the application download by the user. As long as their hash is the same as what we post, they can be confident they have installed the proper application. For communication between the application and the server.

## Industry Standard Best Practices

There are standard practices in the industry, and that is simply because they have been tried, tested, and proven to be effective. These include things such as the aforementioned encryption and security techniques, but also other technical aspects of the programming. In the code, we make sure to use those best practices such as encapsulation and consistency across different files. This helps the developers reduce the chance of write code with errors. Then, we test the application and try to break it, to find any errors may have slipped through.