****

# 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** | **06/30/2024** | **Akhad Alimov** |  |

## 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

Akhad Alimov

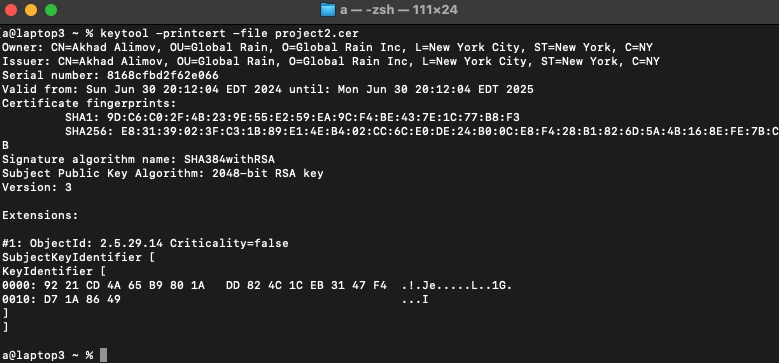
## Algorithm Cipher

As Global Rain a developer we have a mission “Security is everyone’s responsibility” and our client Artemis Financial will need strong security and security practices for safe communication and data handling. I would recommend the standard AES 256-bit encryption and is widely used amongst other financial related organizations. It is a symmetric encryption which means that the key to encrypt and decipher are the same. The bit size is 256 therefore the key size is 256 bits. Having actual random number is critical for encryption to create certainty that the encryption key Is random and cannot be guessed or brute forced by a computer.

Hashing unlike encryption is very difficult or impossible to reverse. Therefore, it is applicable to use in scenarios to verify authenticity and detect tampering as any change to data would create different hash. SHA-256 bit is commonly used for verification purposes.

## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

Insert a screenshot below of the checksum verification.

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 screenshot of a computer

Description automatically generated

## Functional Testing

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

[Insert screenshots here.] A computer screen shot of a program code

Description automatically generated

## Summary

For this specific application I don’t belive I could factor code further without adding more complexity such as taking an input instead of hardcoding the string, adding random info into the data being hashed (also referred as salt), and more robust error handling. Otherwise, I believe the code is ok for the specified application purpose. However, adding network defense features to rate limit or prevent DDOS attack would be critical if it were to deploy amongst many computers or to the internet. Logging the usage can also be very helpful as everything is recorded, and you can analyze all incoming and outgoing traffic.

## Industry Standard Best Practices

I configured the application properties so that it would run in HTTPS instead of http meaning that all communication between server and user is encrypted. In this case it was local host, so it was encrypted within. Then, I decided to use SHA-256 infamously known to be impossible to reverse with current computational capabilities. Therefore, data verification can be certain to high level that data was not manipulated as any would output a different SHA-256 hash. Also running static analysis further proved that there was not any major security risk. Using industry best standards is important because many of these practices were made so due to previous common vulnerabilities. NIST recommends the AES-256 bit encryption as it’s very secure and quick enough to not take up too much of the compute power.