****

# 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** | **8/10/2023** | **Brian Engel** |  |

## 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

Brian Engel

## Algorithm Cipher

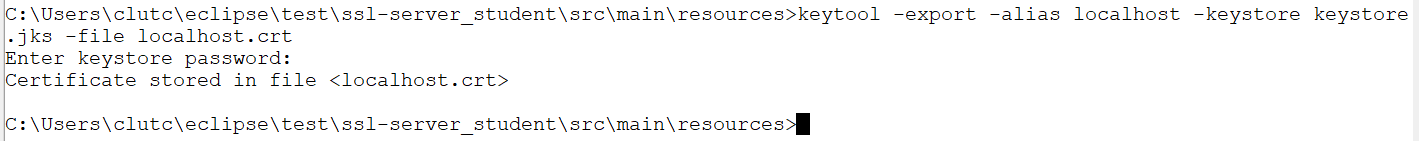
I am choosing SHA-256 is because it is very resistant to collision attacks, meaning it is extremely unlikely to have 2 different inputs that produce the same output. It is also resistant to preimage attacks, which is when you can find an input that gives out a specific output, and is also resistant to length extension attacks, where you can add on to the input and have it produce a valid output. It is also widely supported and was easy to use. Using SHA-256 means that the key is 256 bits, which means that there are 2^256 possible combinations, which is virtually impossible to crack without an enormous amount of time and resources. Using random numbers helps to keep the key unpredictable, which keeps the cipher as strong as it can be. Symmetric keys are used for both encryption and decryption and should be used when you don’t have to transmit data, otherwise you would have to transmit the key as well and that is a huge security concern. Asymmetric keys usually have a public key for encrypting, so anyone can send an encrypted message to the receiver, and then a private key for decrypting, so only the receiver can decrypt the message. This is the case with credit card transactions. All merchants encrypt the card info, but only the credit card company can decrypt it. SHA is a trusted encryption algorithm that many organizations and governments around the world use. The thing that surprised me in an industry that technology evolves so quickly, this cipher is old. It was developed in 2001 by the NSA and NIST. It is also one of the easiest to use, secure, and fastest, which makes it easy to understand the cipher’s longevity of use.

## 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.

A computer screen shot 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.

I had initially forgotten to run a dependency check on the code before I had refactored it for the checksum and added the certificate, so I created a new project with the code base and ran a dependency check on that since it would be the same as before I had changed anything. Both dependency checks ended up with 104 vulnerabilities, so that shows that the code that I added didn’t add any new vulnerabilities.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a check

Description automatically generated

## Functional Testing

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

On further inspection of my code, I realized that some of the code might be vulnerable to injection attacks, so I changed the strings that are concatenated together for the return statement. Actually the return statement is basically the same, but I cleaned the strings with the htmlEscape function so that it would be much harder to use a injection attack if a bad actor got access to the strings.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

In this project, the main areas of security that I addressed were input validation, API’s, cryptography, and client / server security. Input validation was addressed in the strings that are returned in the html. While the program doesn’t take input at this point, I set it up so it would be easy to change it so you can input the string to encrypt and the name of the cipher you want to use. Adding input validation in anticipation of a change seemed like the best thing to do so it wouldn’t be overlooked later. API security is addressed in running the dependency check report, and addressing known issues of the API’s that are used. Cryptography is addressed in the use of the checksum cipher algorithm SHA-256, which encodes data and is virtually impossible to decrypt without the key. Client / server security is addressed through creating certificates that show that the server is what it says it is and not a bad actor pretending to be your site. This is usually done through a certificate authority, but for our project we just used self signed certificates since it is easy and free. The process of adding layers of security is basically finding a area of concern, and then adding another layer of security to protect it. One area that I feel could have been addressed but wasn’t is adding a layer of security for authentication and authorization, but might have been out of the scope of this project.

## Industry Standard Best Practices

I used industry standard best practices to maintain the application’s current security by addressing input validation, API’s, cryptography, and client / server security. This helps to prevent injection attacks, cross site scripting, and insecure data storage. Applying industry standard best practices for secure coding is essential for the company’s overall wellbeing, especially when it comes to an organization that deals with other people’s financial well-being. Keeping security in mind while coding lowers the risk and minimizes the cost of having to fix code later. It also is necessary for complying with government regulations, increases the quality and performance of the code, and instills trust from the client and their customers.