****

# 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** | **10/18/2024** | **Jason Wright** |  |

## 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

Jason Wright

## Algorithm Cipher

My recommendation for an algorithm cipher is AES-256 (Advanced Encryption Standard with a 256-bit Key). AES is a widely accepted encryption algorithm used around the world to secure sensitive data. AES-256 is also a symmetric key cipher, which means that the same key is used for both encryption and decryption, making it both a secure and efficient cipher.

AES is a block cipher and not a hash function, but it can be used in tandem with a cryptographic hash function such as SHA-256. AES-256 uses a 256-bit key which means that there 2256 different possible key combinations. With that number of possibilities, it is virtually impossible to crack with current technology.

Random numbers are used to create Initialization Vectors (IVs) with AES-256. These IVs are used to prevent the same plaintext from producing the same ciphertext which allows for more security. AES-256 is a symmetric key cipher, meaning it uses a single, shared key for both encryption and decryption. Symmetric encryption is generally faster and more efficient for encrypting large amounts of data as opposed to non-symmetric encryption. Non-symmetric encryption algorithms, such as RSA, use public/private key pairs and are often used for secure key exchanges or digital signatures.

The National Institute of Standards and Technology (NIST) first introduced AES in 2001 as the successor to DES (Data Encryption Standard), which had become vulnerable to brute-force attacks with advancements in computing power. AES was selected for its high security, flexibility, and efficiency after a global competition. It remains the industry standard today.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen shot of a black screen

Description automatically generated

A screenshot of a certificate

Description automatically generated

A screenshot of a computer

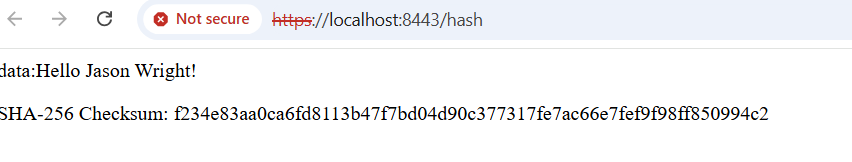
Description automatically generated

A screenshot of a computer

Description automatically generated

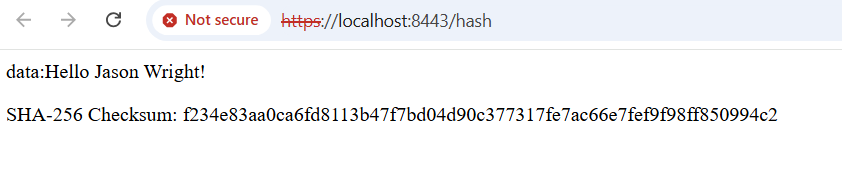
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

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



## 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

A screenshot of a computer error

Description automatically generated

## Functional Testing

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

A screenshot of a computer

Description automatically generated

## Summary

Areas of Security Addressed:

* **Input Validation:** I ensured that any user inputs, particularly those involving URL parameters and request data, were sanitized to prevent injection attacks.
* **APIs:** I refactored the API endpoints to use secure protocols (HTTPS) to ensure the transmission of data between client and server was encrypted and secure.
* **Cryptography:** I integrated a secure cipher algorithm (SHA-256) for encrypting sensitive data and for generating secure checksums for file verification.
* **Client/Server:** I configured the application to connect with secure HTTPS connections and with a self-signed certificate.
* **Code Error:** I ensured that the code ran without errors and implemented exception handling with try catch statements.
* **Code Quality:** I ensured that the code was organized and used a consistent naming convention for fields and functions. I also ran a dependency check to ensure that there weren’t any vulnerabilities with any of the dependencies that I was using.

I added layers of security to my software application by following the different steps laid out in the vulnerability assessment process flow diagram. I first ensured that the inputs were properly validated and would not accept any malicious inputs. Then I implemented the cryptographic hash function into the code. I then ensured that the connection would be made through a secure HTTPS connection. I then read through my code and manually looked for any errors or vulnerabilities. Finally, I ran a dependency check to find any vulnerabilities with the dependencies that I was using.

## Industry Standard Best Practices

To maintain the application’s security, I applied industry-standard best practices such as enabling HTTPS with SSL/TLS for secure data transmission, implementing AES-256 encryption to protect sensitive information, and conducting input validation to prevent injection attacks. By refactoring the code to follow these protocols, I ensured that data handling, transmission, and storage met recognized security standards. Additionally, I used dependency check tools to monitor third party libraries for vulnerabilities, confirming that the application was using up to date and secure components.

Applying these best practices enhances the company’s overall well-being by reducing the risk of security incidents and building trust with clients. Secure coding not only protects the company’s data but also ensures compliance with regulatory standards, helping avoid potential penalties, and reinforcing the company’s reputation in the market. This proactive approach to security supports long term stability, reduces the likelihood of costly breaches, and positions the company as a trustworthy service provider.