****

# 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** | **10/13/2023** | **Brent Longstreet** |  |

## 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

Brent Longstreet

## Algorithm Cipher

Given Artemis Financial needs and potential security vulnerabilities, I would recommend the encryption algorithm known as SHA256. The SHA256 is a well-known and tested algorithm used in real-world applications. It is a fast algorithm that offers great security as well as performance. When executing the SHA256 algorithm, a 256-bit hash is generated. Additionally, 256 bits is equivalent to 32 bytes. This level of hashing is important to avoid collisions caused by different value strings being converted into the same hash value. These collisions can cause serious security concerns and lead to invalid input validation. This encryption algorithm cipher is regarded as the industry standard for financial institutions. To brute force and decrypt the hash would take several years to decades.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated

A screenshot of a certificate

Description automatically generated

## 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.

@RestController

class ServerController{

//FIXME:  Add hash function to return the checksum value for the data string that should contain your name.

    @RequestMapping("/hash")

    public String myHash() {

        String data = "Brent Longstreet";

        String algorithm = "SHA-256";

        String hashValue = "";

        try {

            MessageDigest md = MessageDigest.getInstance(algorithm);

            byte[] hashBytes = md.digest(data.getBytes(StandardCharsets.UTF\_8));

            hashValue = bytesToHex(hashBytes);

        } catch (NoSuchAlgorithmException e) {

            e.printStackTrace();

        }

        return "<p>Data: " + data + "</p><p>Name of Cipher Algorithm used: " + algorithm + "</p><p>Checksum: " + hashValue + "</p>";

    }

    private static String bytesToHex(byte[] bytes) {

        StringBuilder sb = new StringBuilder();

        for (byte b : bytes) {

            sb.append(String.format("%02x", b));

        }

        return sb.toString();

    }

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

## Functional Testing

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

package com.snhu.sslserver;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.nio.charset.StandardCharsets;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication

public class ServerApplication {

    public static void main(String[] args) {

        SpringApplication.run(ServerApplication.class, args);

    }

}

@RestController

class ServerController{

//FIXME:  Add hash function to return the checksum value for the data string that should contain your name.

    @RequestMapping("/hash")

    public String myHash() {

        String data = "Brent Longstreet";

        String algorithm = "SHA-256";

        String hashValue = "";

        try {

            MessageDigest md = MessageDigest.getInstance(algorithm);

            byte[] hashBytes = md.digest(data.getBytes(StandardCharsets.UTF\_8));

            hashValue = bytesToHex(hashBytes);

        } catch (NoSuchAlgorithmException e) {

            e.printStackTrace();

        }

        return "<p>Data: " + data + "</p><p>Name of Cipher Algorithm used: " + algorithm + "</p><p>Checksum: " + hashValue + "</p>";

    }

    private static String bytesToHex(byte[] bytes) {

        StringBuilder sb = new StringBuilder();

        for (byte b : bytes) {

            sb.append(String.format("%02x", b));

        }

        return sb.toString();

    }

}

A screenshot of a computer screen

Description automatically generated

## Summary

The creation of a self-sign certificate allowed the webpage to utilize HTTPS. This made sure the website was secure and informed the user their data is encrypted. I refracted the pom.xml to correct the vulnerabilities that were found in the dependency check. It highlighted and informed me of all vulnerability concerns caused by the dependencies used. I utilized the SHA256 encryption algorithm and created a hashing function to encrypt the users’ information. The function showcased it’s ability in action and the output after the encryption takes place. I made sure to change the owasp mavern dependency checker version to the latest to ensure it was up-to-date.

## Industry Standard Best Practices

The pom.xl file should be updated every month or so to ensure that the dependencies used are up-to-date. A owasp mavern dependency check should be performed sooner than that so the company is informed of any new security vulnerabilities and they can be addressed sooner rather than later. When deciding on an encryption algorithm, understand what the situation is calling for and select one that best suits the needs. Performing these industry standard practices will ensure the security of the system and protect itself from future attacks.