****

# 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** | **[Date]** | **[Daniel Evers]** |  |

## 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

[Daniel Evers.]

## Algorithm Cipher

[The encryption algorithm I would use would be AES. AES or Advanced encryption standard is an encryption algorithm that is used widely today. Financial institutions, the military and U.S government all use AES for their encryption. AES encryption was introduced in 2001 and approved by the NIST. It was made to replace the old system called DES. AES encryption is very hard to crack, it would take years for someone to crack AES and decrypt the ciphertext. Today, AES is still used however, a new algorithm called post-quantum cryptography is becoming popular. AES encryption has multiple bit levels, 128, 192 and 256. AES is a symmetric key, meaning that the same shared key can encrypt and decrypt ciphertext. An asymmetric key algorithm like RSA, needs a pair of keys to encrypt and decrypt data. A hashing algorithm I would use would be SHA-256. SHA-256 hashes the data into a 256 bit hash, ensuring that the data is secure and untouched. Using both of these methods can help make data secure. ]

## Certificate Generation

Insert a screenshot below of the CER file.

[A computer screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated.]

## Deploy Cipher

Insert a screenshot below of the checksum verification.

[A screenshot of a computer

Description automatically generatedCode for it.

package com.snhu.sslserver;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

@SpringBootApplication

public class SslServerApplication {

public static void main(String[] args) {

SpringApplication.run(SslServerApplication.class, args);

}

@RestController

public static class ChecksumController {

@GetMapping("/checksum")

public String getChecksum() {

String data = "Hello World Daniel Evers!"; // Static data for checksum

try {

MessageDigest digest = MessageDigest.getInstance("SHA-256");

byte[] hash = digest.digest(data.getBytes());

StringBuilder hexString = new StringBuilder();

for (byte b : hash) {

String hex = Integer.toHexString(0xff & b);

if (hex.length() == 1) hexString.append('0');

hexString.append(hex);

}

return "Data: " + data + "\nChecksum (SHA-256): " + hexString.toString();

} catch (NoSuchAlgorithmException e) {

return "Error calculating checksum: " + e.getMessage();

}

}

}

}]

## Secure Communications

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

[A close-up of a computer screen

Description automatically generated

Code

package com.snhu.sslserver;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

@SpringBootApplication

public class SslServerApplication {

public static void main(String[] args) {

SpringApplication.run(SslServerApplication.class, args);

}

@RestController

public static class ChecksumController {

@GetMapping("/checksum")

public String getChecksum() {

return calculateChecksum("Hello World Daniel Evers!");

}

@GetMapping("/hash")

public String getHash() {

return calculateChecksum("Hello World Daniel Evers!");

}

private String calculateChecksum(String data) {

try {

MessageDigest digest = MessageDigest.getInstance("SHA-256");

byte[] hash = digest.digest(data.getBytes());

StringBuilder hexString = new StringBuilder();

for (byte b : hash) {

String hex = Integer.toHexString(0xff & b);

if (hex.length() == 1) hexString.append('0');

hexString.append(hex);

}

return "Data: " + data + "\nChecksum (SHA-256): " + hexString.toString();

} catch (NoSuchAlgorithmException e) {

return "Error calculating checksum: " + e.getMessage();

}

}

}

}

## Secondary Testing

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

[Code

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.2.4.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.snhu</groupId>

<artifactId>ssl-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>ssl-server</name>

<description>ssl-server skeleton for CS-305</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-rest</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

<plugin>

<groupId>org.owasp</groupId>

<artifactId>dependency-check-maven</artifactId>

<version>11.1.1</version>

<executions>

<execution>

<goals>

<goal>check</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

A screenshot of a computer

Description automatically generatedAfter Fixing

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.1</version>

<relativePath/>

</parent>

<groupId>com.snhu</groupId>

<artifactId>ssl-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>ssl-server</name>

<description>SSL server skeleton for CS-305</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-rest</artifactId>

<version>3.2.1</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<version>3.2.1</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.owasp</groupId>

<artifactId>dependency-check-core</artifactId>

<version>11.1.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

<plugin>

<groupId>org.owasp</groupId>

<artifactId>dependency-check-maven</artifactId>

<version>11.1.1</version>

<executions>

<execution>

<goals>

<goal>check</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

A screenshot of a computer

Description automatically generated.]

## Functional Testing

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

[A close-up of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated.]

## Summary

[One thing I changed in the code was using the hash endpoint. I also added a checksum in to generate a hash. I also made sure to use https. I have a self signed certificate to encrypt the data and make it secure. To prevent information from being leaking in errors, I put in the line “NuSuchAlgortihmException.” I manually reviewed the code twice, running it through the maven dependency tool. ]

## Industry Standard Best Practices

[The first best practice I used was using HTTPS. I made sure HTTPS was used and had an SSL certificate. This ensures that communication between the client and server is safe and secure. Attacks like man in the middle attacks can be prevented. Another best practice I used was SHA-256. This is a strong hashing algorithm. Sha-256 is one of the best choices when it comes to hashing. A third best practice I used was error handling. Sensitive data can sometimes be leaked through errors and logs, by using the NoSuchAlgortihmExecption, it can prevent leakage. The fourth practice I used was keeping the code simple. By keeping the code simple, it allows for quick and easy error identifying. Some benefits for the company is, data being safe and secure, customer trust, prevents risk, cost effective and can have an advantage over the competition.