****

# 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** | **6/14/2023** | **Dakota Keyes** |  |

## Client



## Developer

Dakota Keyes

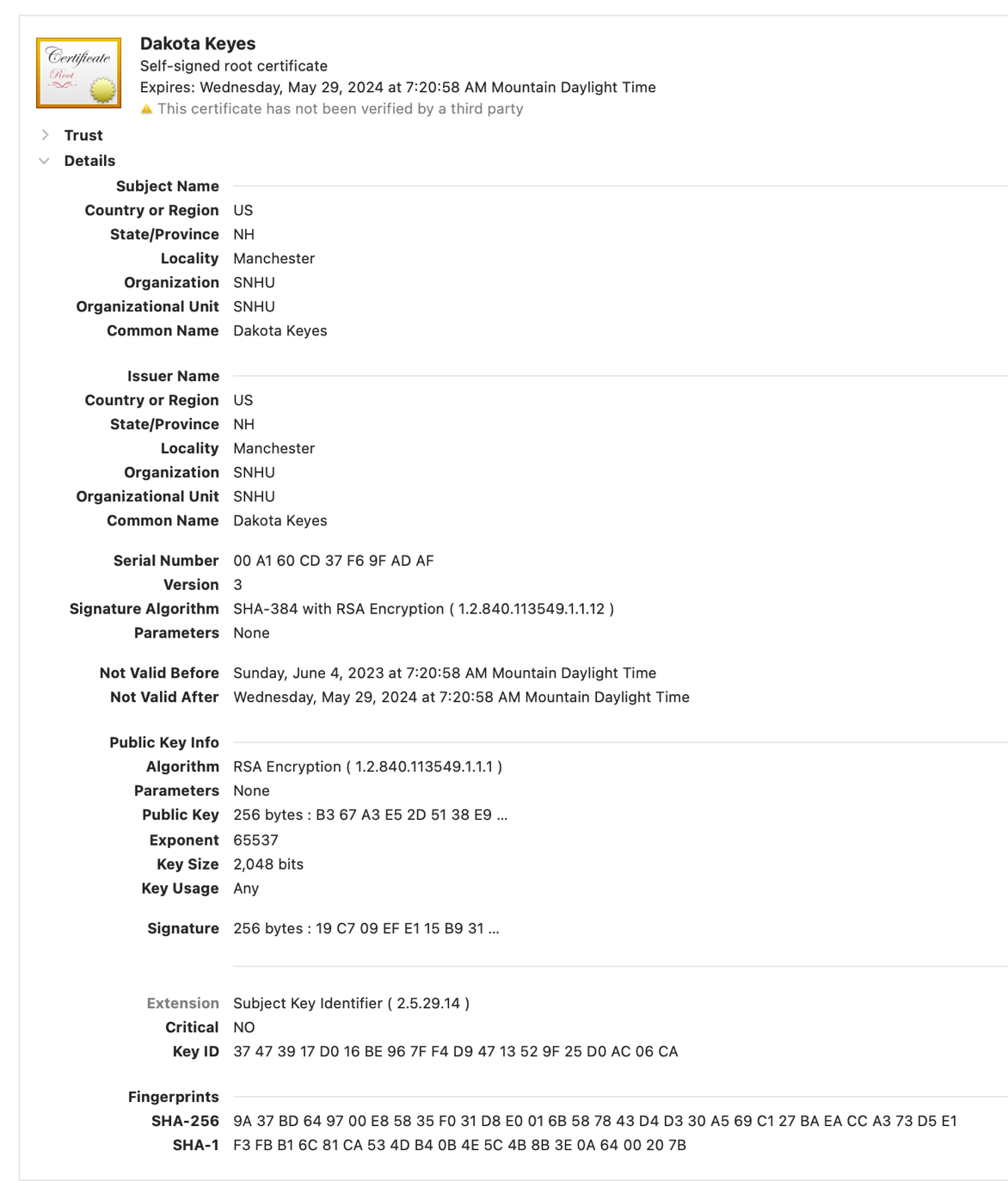
## Algorithm Cipher

I recommend the Secure Hash Algorithm 256-bit encryption algorithm cipher to avoid collision. This encryption cipher generates a 32-byte or 256-bit hash value. The primary purpose of a hash function (in cryptographic regards) is to take input data and produce a fixed output, called a hash. Each hash value must be unique to the input data. For example, any change in the data should create a different hash value. This is regarded as collision resistance.

SHA-256 is used for a variety of platforms such as digital signatures, authentication codes, and can be utilized to securely ensure data authenticity. SHA-256 is part of the SHA-2 family of hash functions, SHA-384, SHA-512/224, SHA-224, SHA-512/256. SHA-256 is widely used and accepted as a secure algorithm cipher to protect data integrity and authenticity and is great choice for many different security applications.

## 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 with medium confidence

## Secure Communications

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

A screenshot of a computer

Description automatically generated with medium confidence

## Secondary Testing

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

**Suppression.xml**

A screenshot of a computer

Description automatically generated with medium confidence

**Pom.xml**

**5.3.0 Version (Before updating to 6.0.5)**

A screen shot of a computer

Description automatically generated

However, I decided to upgrade the Dependency Check version 6.0.5 so the dependency check was more up-to-date and accurate as possible.

<groupId>org.owasp</groupId>

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

<version>6.0.5</version>

I also upgrade BouncyCastle from 1.4.6 to 1.7.0

<!-- https://mvnrepository.com/artifact/org.bouncycastle/bcprov-jdk15on -->

<dependency>

<groupId>org.bouncycastle</groupId>

<artifactId>bcprov-jdk15on</artifactId>

<version>1.7.0</version>

</dependency>

Final pom.xml with upgraded dependency check version and bouncy castle update, with suppressions for false positives.

<?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>rest-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>rest-service</name>

<description>Project for CS-305 Module 7</description>

<properties>

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

</properties>

<dependencies>

<dependency>

<groupId>com.jayway.jsonpath</groupId>

<artifactId>json-path</artifactId>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.bouncycastle/bcprov-jdk15on -->

<dependency>

<groupId>org.bouncycastle</groupId>

<artifactId>bcprov-jdk15on</artifactId>

<version>1.70</version>

</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>8.2.1</version>

<configuration>

<suppressionFiles>

<suppressionFile>suppression.xml</suppressionFile>

</suppressionFiles>

</configuration>

<executions>

<execution>

<goals>

<goal>check</goal>

</goals>

</execution>

</executions>

</plugin>

<plugin>

<groupId>org.owasp</groupId>

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

<version>8.2.1</version>

<executions>

<execution>

<goals>

<goal>check</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

**Dependency Check without Suppressions**

**5.0.3 – Original Version of Dependency Check**

A screenshot of a computer

Description automatically generated

As you can see, there are 112 vulnerabilities found an no vulnerabilities have been suppressed. By creating the suppression.xml file, we can filter out false positives.

For example CVE-2020-15522 stems from a vulnerability within bouncy castle.

<!-- CVE-2020-15522 -->

<!-- Race Condition -->

<suppress>

<notes><CDATA[

file name: bcprov-jdk15on-1.46.jar

]]></notes>

<sha1>991c96a4e31e6c19e2b9136c8955bd423f2dc4c7</sha1>

<cve>CVE-2020-15522</cve>

</suppress>

**Dependency Check with Suppressions**

**Updated to Version 6.0.5 for dependency check.**

![A screenshot of a computer

Description automatically generated with low confidence

You can see that 5 vulnerabilities have been suppressed by creating the suppression.xml file and configuring it to remove false positives from the dependency check.

Final dependency check after updating bouncy castle from version 1.4.6 to 1.7.0, and dependency check version to 6.0.5. However, after updating bouncy castle from 1.4.6 to 1.7.0 the suppression.xml file to suppress CVE-2020-15522, becomes irrelevant with the updated version.

A picture containing text, screenshot, font

Description automatically generated

## Functional Testing

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

A screenshot of a computer

Description automatically generated with medium confidence

From these screenshots, you can see the code executing without any errors.

A screenshot of a computer

Description automatically generated

SslServerApplicationTests.java is running as well without any errors, executing in 0.91 seconds.

A screenshot of a computer

Description automatically generated with medium confidence

Code for SslServerApplication.java file

A screenshot of a computer program

Description automatically generated with medium confidence

## Summary

In the refactored code, I have added a Rest Controller, which is a simple and secure controlled for the RESTful API hash endpoint. For this project, I have selected a SHA-256 hashing cipher algorithm function for secure communication.

Spring Boot is one Java’s many server-side OAuth2 libraries that are available for use. However, spring-based implementation is a great choice because it is well integrated into the architecture of Spring Security and will limit the need to manage and handle low-level details of use. These security-related libraries are handled by Maven with the help of Spring Boot.

The Spring Boot requires an explicit version inside the pom.xml file which is used for configuration of Maven dependencies. Spring Boot protects against authorization and authentication attacks.

In order to maintain security of this application, I would recommend running regular dependency checks and check for new vulnerable dependencies. Also, staying up to date with the newest versions and releases by regularly checking for updates will help increase security and reduce possible security threats. These plugins can be found in the pom.xml file of the zipped project file.

By refactoring the code, I have addressed the code quality, through the ServerController class, APIs, cryptography and client/server. Through creating self-signed certificates and generated keys which will be allow us to connect with the SHA-256 hashing cipher algorithm. This encryption (cryptography) allows for secure communication by allowing only the indented audience to interpret the data that is being transferred. Unsecure communications pose a major risk for Artemis Financial and using the SHA-256 hash algorithm, communications can be secured to protect sensitive data from being leaked or intercepted by hackers or attackers.

## Industry Standard Best Practices

Industry standard best practices differ for different types of industries. For example, a gaming company will have different best practices than a financial company. For a financial company like Artemis Financial, security is a primary concern. According to BrowserStack, adhering to industry-specific coding standards makes writing correct code that matches product expectations easier. Furthermore, writing code that will satisfy the end-users by meeting business requirements becomes simpler (2023). Implementing an enterprise security framework is an important aspect for this project. Spring Security is a framework that focuses on providing authorization and authentication for Java applications (*Spring Security*, n.d.) Using Maven, we have implemented Spring Boot security features through dependencies within the pom.xml file.

**References**

Bose, S. (2023). Coding Standards and Best Practices to Follow | BrowserStack. *BrowserStack*. https://www.browserstack.com/guide/coding-standards-best-practices

*Spring Security*. (n.d.). Spring Security. https://spring.io/projects/spring-security