GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	2/17/2026	Kasra Pratt	

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.

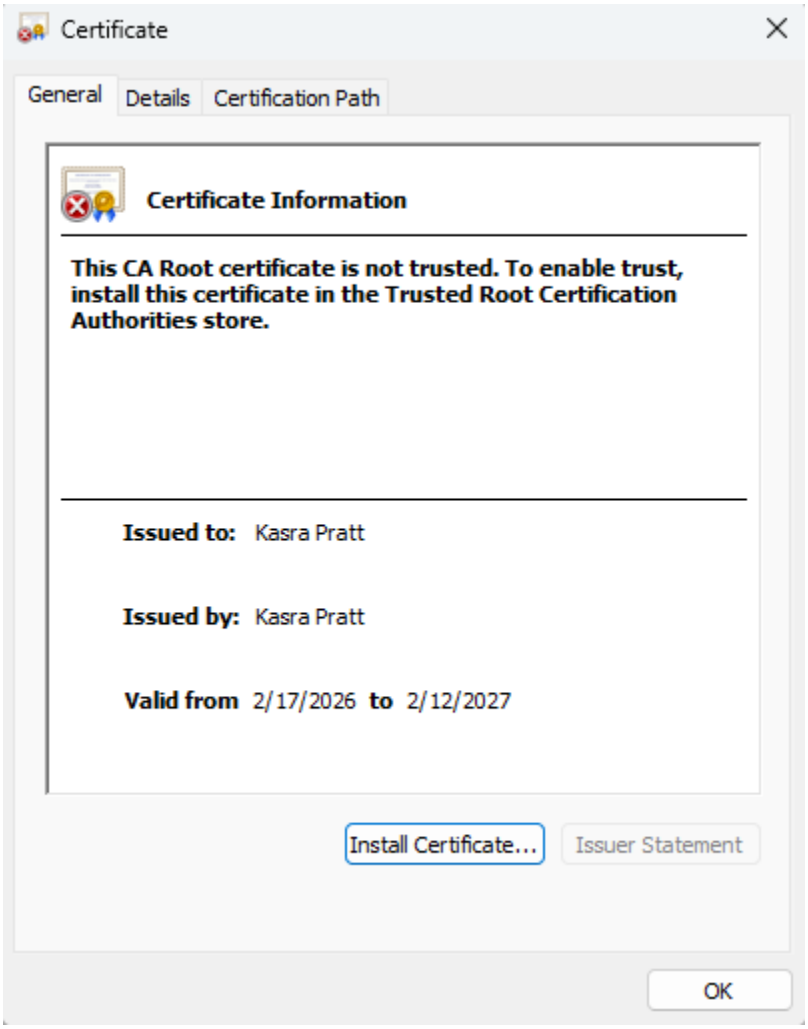
- Respond to the steps outlined below and include your findings.
- Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
- Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

Developer
Kasra Pratt

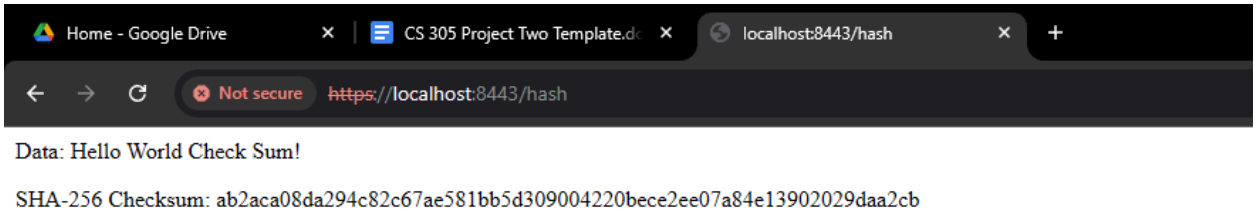
1. Algorithm Cipher

I recommend using SHA-256 as the encryption algorithm cipher for deploying the checksum verification in Artemis Financial's application. SHA-256 is a cryptographic hash function that produces a 256-bit hash value, typically rendered as a hexadecimal number, 64 digits long.

2. Certificate Generation
Insert a screenshot below of the CER file.

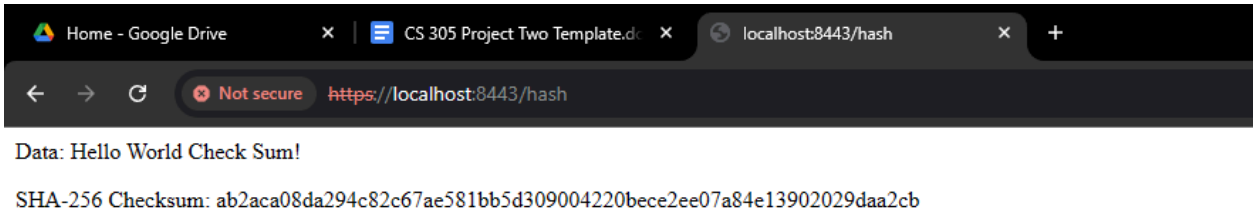


3. Deploy Cipher
Insert a screenshot below of the checksum verification.



4. Secure Communications

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



5. Secondary Testing

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

SslServerApplication.java | ChecksumController.java | ssl-server_student/pom.xml

```java
package com.snhu.sslserver;

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

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

@RestController
public class ChecksumController {

        @GetMapping("/hash")
        public String getChecksum() throws NoSuchAlgorithmException {
            String data = "Hello World Check Sum!";
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            byte[] hash = md.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 + "<p>SHA-256 Checksum: " + hexString.toString();
    }

}
```

```
<modelVersion>4.0.0</modelVersion>
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>4.0.2</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>25</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>12.2.0</version>
            <executions>
              <execution>
                  <goals>
                      <goal>check</goal>
                  </goals>
              </execution>
            </executions>
        </plugin>
```

How to read the report | Suppressing false positives | **Getting Help: github issues**

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 12.2.0
- *Report Generated On*: Tue, 17 Feb 2026 17:12:45 -0500
- *Dependencies Scanned*: 56 (29 unique)
- *Vulnerable Dependencies*: 0
- *Vulnerabilities Found*: 0
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Summary of Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count

Dependencies (vulnerable)

This report contains data retrieved from the National Vulnerability Database.
This report may contain data retrieved from the CISA Known Exploited Vulnerability Catalog.
This report may contain data retrieved from the Github Advisory Database (via NPM Audit API).
This report may contain data retrieved from RetireJS.
This report may contain data retrieved from the Sonatype OSS Index.

6. Functional Testing

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

```java
package com.snhu.sslserver;

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

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

@RestController
public class ChecksumController {

        @GetMapping("/hash")
        public String getChecksum() throws NoSuchAlgorithmException {
            String data = "Hello World Check Sum!";
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            byte[] hash = md.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 + "<p>SHA-256 Checksum: " + hexString.toString();
    }

}
```

Tasks Console ×

erverApplication [Java Application] C:\Users\sidec\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (Feb 19, 2026, 10:45:54 AM) [pid: 8400]

```
.   ____          _            __ _ _
\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 : Spring Boot ::                (v4.0.2)

26-02-19T10:45:55.859-05:00  INFO 8400 --- [           main] c.snhu.sslserver.SslServerApplication    : Starting SslServerApplication using Java 17.0.9 with PID 8400 (C:\Us
26-02-19T10:45:55.871-05:00  INFO 8400 --- [           main] c.snhu.sslserver.SslServerApplication    : No active profile set, falling back to 1 default profile: "default"
26-02-19T10:45:56.992-05:00  INFO 8400 --- [           main] o.s.boot.tomcat.TomcatWebServer          : Tomcat initialized with port 8443 (https)
26-02-19T10:45:57.009-05:00  INFO 8400 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
26-02-19T10:45:57.009-05:00  INFO 8400 --- [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/11.0.15]
26-02-19T10:45:57.055-05:00  INFO 8400 --- [           main] b.w.c.s.WebApplicationContextInitializer : Root WebApplicationContext: initialization completed in 1118 ms
26-02-19T10:45:57.806-05:00  INFO 8400 --- [           main] o.a.t.util.net.NioEndpoint.certificate   : Connector [https-jsse-nio-8443], TLS virtual host [_default_], certi
26-02-19T10:45:57.826-05:00  INFO 8400 --- [           main] o.s.boot.tomcat.TomcatWebServer          : Tomcat started on port 8443 (https) with context path '/'
26-02-19T10:45:57.832-05:00  INFO 8400 --- [           main] c.snhu.sslserver.SslServerApplication    : Started SslServerApplication in 2.374 seconds (process running for 2
26-02-19T10:45:58.919-05:00  INFO 8400 --- [nio-8443-exec-3] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
26-02-19T10:45:58.919-05:00  INFO 8400 --- [nio-8443-exec-3] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
26-02-19T10:45:58.920-05:00  INFO 8400 --- [nio-8443-exec-3] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
```

7. Summary

I refactored by adding the hash endpoint and SSL config, adding layers like cryptographic hashing and TLS encryption. This complies with security testing protocols by ensuring integrity and confidentiality. I also updated all necessary dependencies and reduced the number of known vulnerabilities to 0.

8. Industry Standard Best Practices

Applied best practices such as using secure hash algorithms (NIST-recommended SHA-256), enabling HTTPS for all communications, and running static analysis (dependency-check). Maintained existing security by not altering core dependencies and focusing on additive changes. This mitigates known vulnerabilities like man-in-the-middle attacks and data tampering. For the company, it enhances trust, complies with regulations (e.g., GDPR, PCI-DSS), reduces breach risks, and supports long-term well-being by protecting client financial data.