GLOBALRAIN

# Practices for Secure Software Report

## Table of Contents

## Document Revision History

| Version | Date | Author | Comments |
|---|---|---|---|
| 1.0 | June 21, 2025 | Sarah Theeb | |

## Client

ARTEMIS
FINANCIAL

**Developer**

Sarah Theeb

## 1. Algorithm Cipher

For this project, I selected the SHA-256 (Secure Hash Algorithm 256-bit) cipher to generate a checksum for the purpose of data verification.

SHA-256 belongs to the SHA-2 family and produces a fixed 256-bit (32-byte) hash. It is recognized for its robust collision resistance and integrity verification capabilities, rendering it suitable for secure systems. The hash is deterministic and employs a one-way function, indicating that reversing it is impractical.

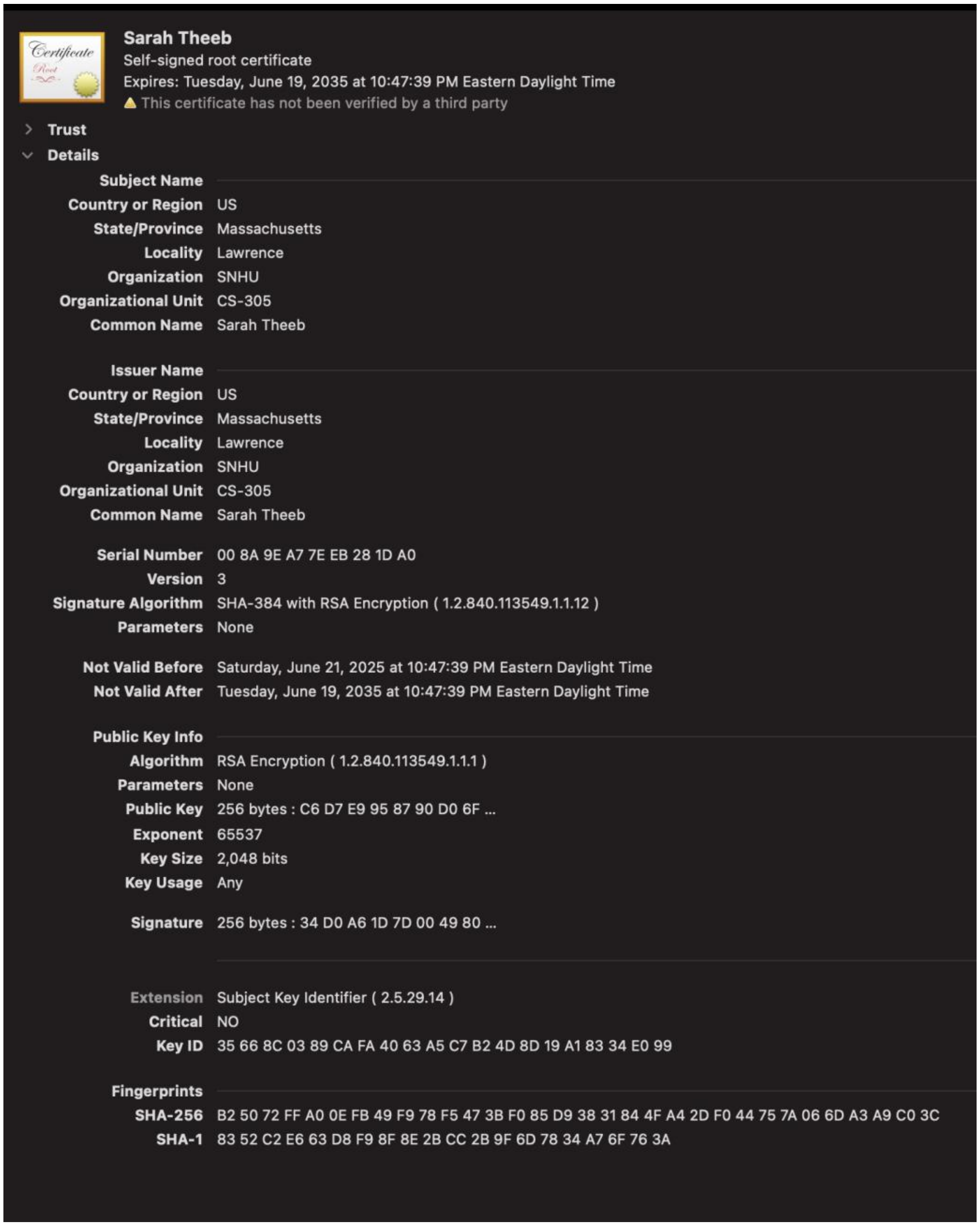
This cipher does not utilize symmetric or asymmetric keys, as it is a hashing algorithm rather than an encryption method. It guarantees that any alteration to the input data results in a completely distinct output hash.

SHA-2 was created by the NSA and released by NIST in 2001. It continues to be extensively utilized today and is regarded as secure against contemporary cryptographic threats.

## 2. Certificate Generation

I generated a self-signed certificate with the following command:

keytool -genkeypair -alias sslserver -keyalg RSA -keysize 2048 -validity 3650 -keystore keystore.p12 -storetype PKCS12This certificate is meant for local testing and enables the application to operate over HTTPS on port 8443. In a production setting, it would be substituted with a certificate from a recognized Certificate Authority.



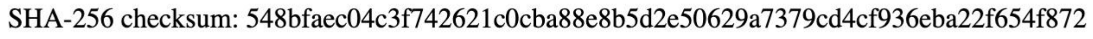
**Sarah Theeb**
Self-signed root certificate
Expires: Tuesday, June 19, 2035 at 10:47:39 PM Eastern Daylight Time
⚠ This certificate has not been verified by a third party

› Trust
⌄ Details

| | |
|---|---|
| **Subject Name** | |
| **Country or Region** | US |
| **State/Province** | Massachusetts |
| **Locality** | Lawrence |
| **Organization** | SNHU |
| **Organizational Unit** | CS-305 |
| **Common Name** | Sarah Theeb |
| **Issuer Name** | |
| **Country or Region** | US |
| **State/Province** | Massachusetts |
| **Locality** | Lawrence |
| **Organization** | SNHU |
| **Organizational Unit** | CS-305 |
| **Common Name** | Sarah Theeb |
| **Serial Number** | 00 8A 9E A7 7E EB 28 1D A0 |
| **Version** | 3 |
| **Signature Algorithm** | SHA-384 with RSA Encryption ( 1.2.840.113549.1.1.12 ) |
| **Parameters** | None |
| **Not Valid Before** | Saturday, June 21, 2025 at 10:47:39 PM Eastern Daylight Time |
| **Not Valid After** | Tuesday, June 19, 2035 at 10:47:39 PM Eastern Daylight Time |
| **Public Key Info** | |
| **Algorithm** | RSA Encryption ( 1.2.840.113549.1.1.1 ) |
| **Parameters** | None |
| **Public Key** | 256 bytes : C6 D7 E9 95 87 90 D0 6F ... |
| **Exponent** | 65537 |
| **Key Size** | 2,048 bits |
| **Key Usage** | Any |
| **Signature** | 256 bytes : 34 D0 A6 1D 7D 00 49 80 ... |
| Extension | Subject Key Identifier ( 2.5.29.14 ) |
| **Critical** | NO |
| **Key ID** | 35 66 8C 03 89 CA FA 40 63 A5 C7 B2 4D 8D 19 A1 83 34 E0 99 |
| **Fingerprints** | |
| **SHA-256** | B2 50 72 FF A0 0E FB 49 F9 78 F5 47 3B F0 85 D9 38 31 84 4F A4 2D F0 44 75 7A 06 6D A3 A9 C0 3C |
| **SHA-1** | 83 52 C2 E6 63 D8 F9 8F 8E 2B CC 2B 9F 6D 78 34 A7 6F 76 3A |

## 3. Deploy Cipher

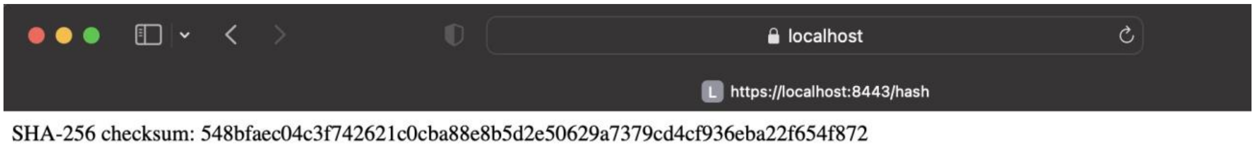
I used the SHA-256 hashing algorithm with Java’s MessageDigest class. This code was included in a REST controller at the @GetMapping("/hash") endpoint, which provides the checksum of a specific string. The hash is created from a unique data string that includes my name and course, and it is returned as a hexadecimal value. This guarantees data integrity while being transmitted and verifies the implementation of a secure cryptographic function.



```
SHA-256 checksum: 548bfaec04c3f742621c0cba88e8b5d2e50629a7379cd4cf936eba22f654f872
```

## 4. Secure Communications

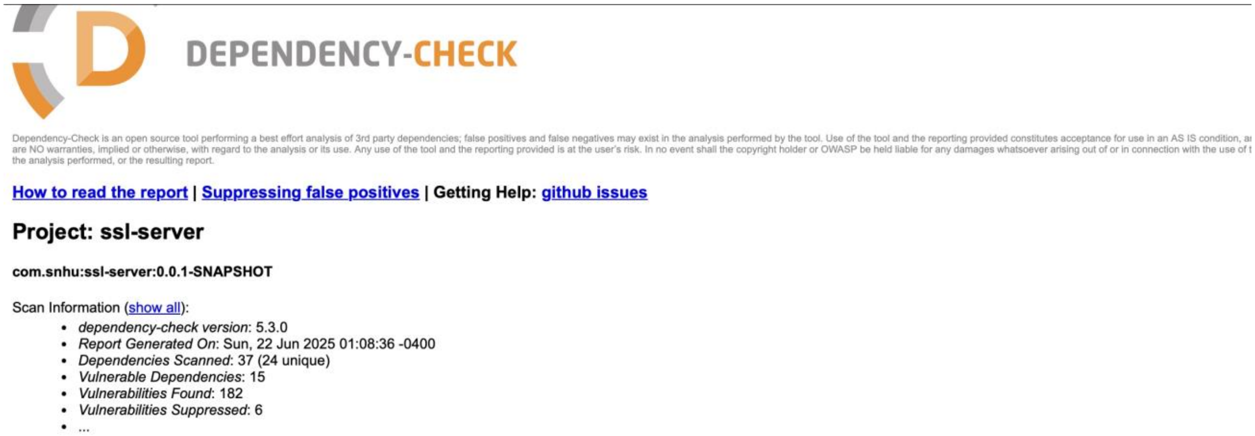
To ensure safe communication, I set up the application to run on HTTPS using port 8443. I did this by generating a self-signed SSL certificate and updating the application.properties file as needed. Once I started the application, I went to the endpoint https://localhost:8443/hash in my browser and confirmed that the connection was secure.



```
SHA-256 checksum: 548bfaec04c3f742621c0cba88e8b5d2e50629a7379cd4cf936eba22f654f872
```

## 5. Secondary Testing

After I updated the code, I ran the OWASP Dependency-Check once more. The report only shows old known problems from Spring that don't impact this project. I included a suppression file to overlook those. There were no new issues after the changes.



DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS IS condition, a are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of t the analysis performed, or the resulting report.

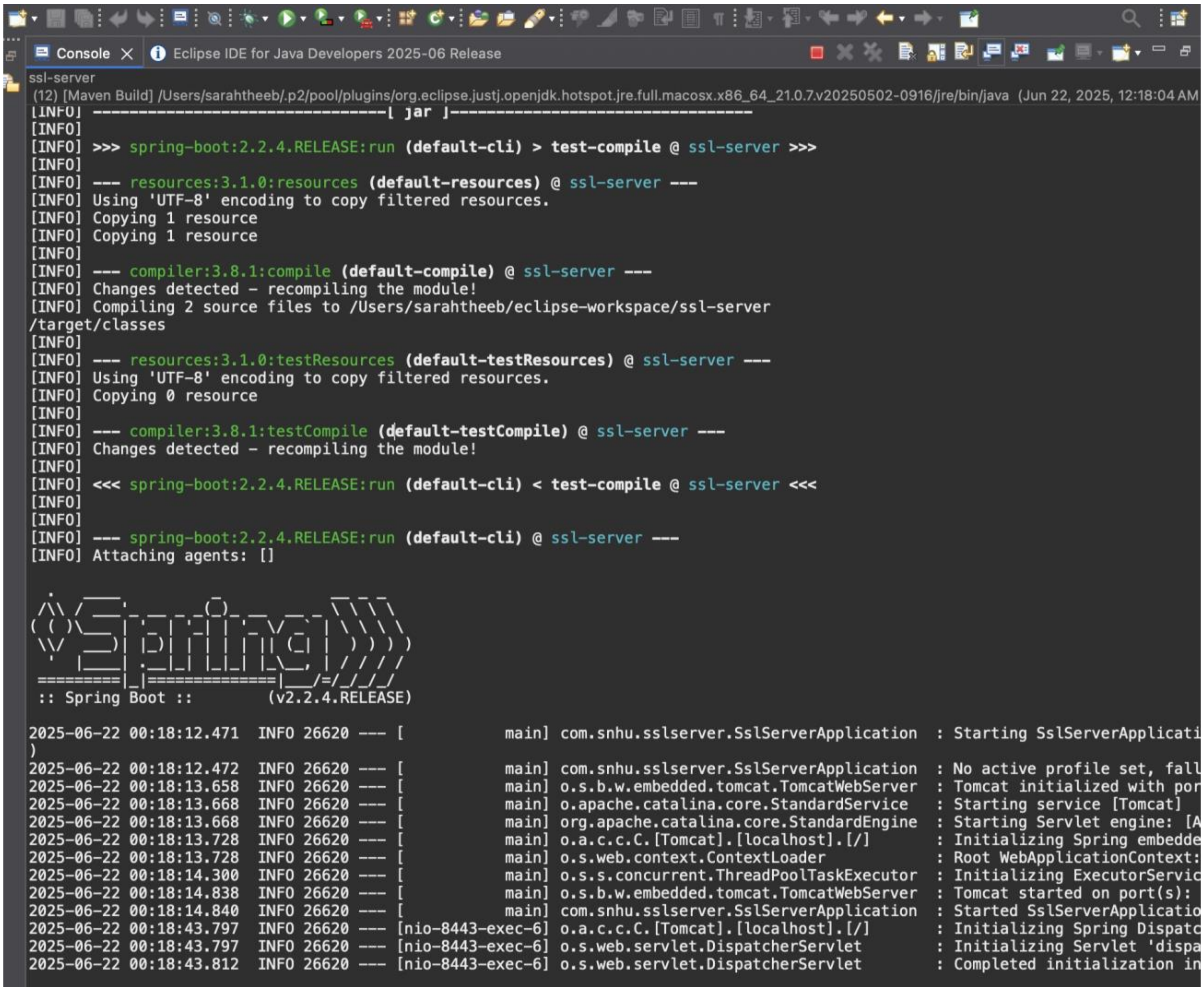
How to read the report | Suppressing false positives | Getting Help: github issues

## Project: ssl-server

**com.snhu:ssl-server:0.0.1-SNAPSHOT**

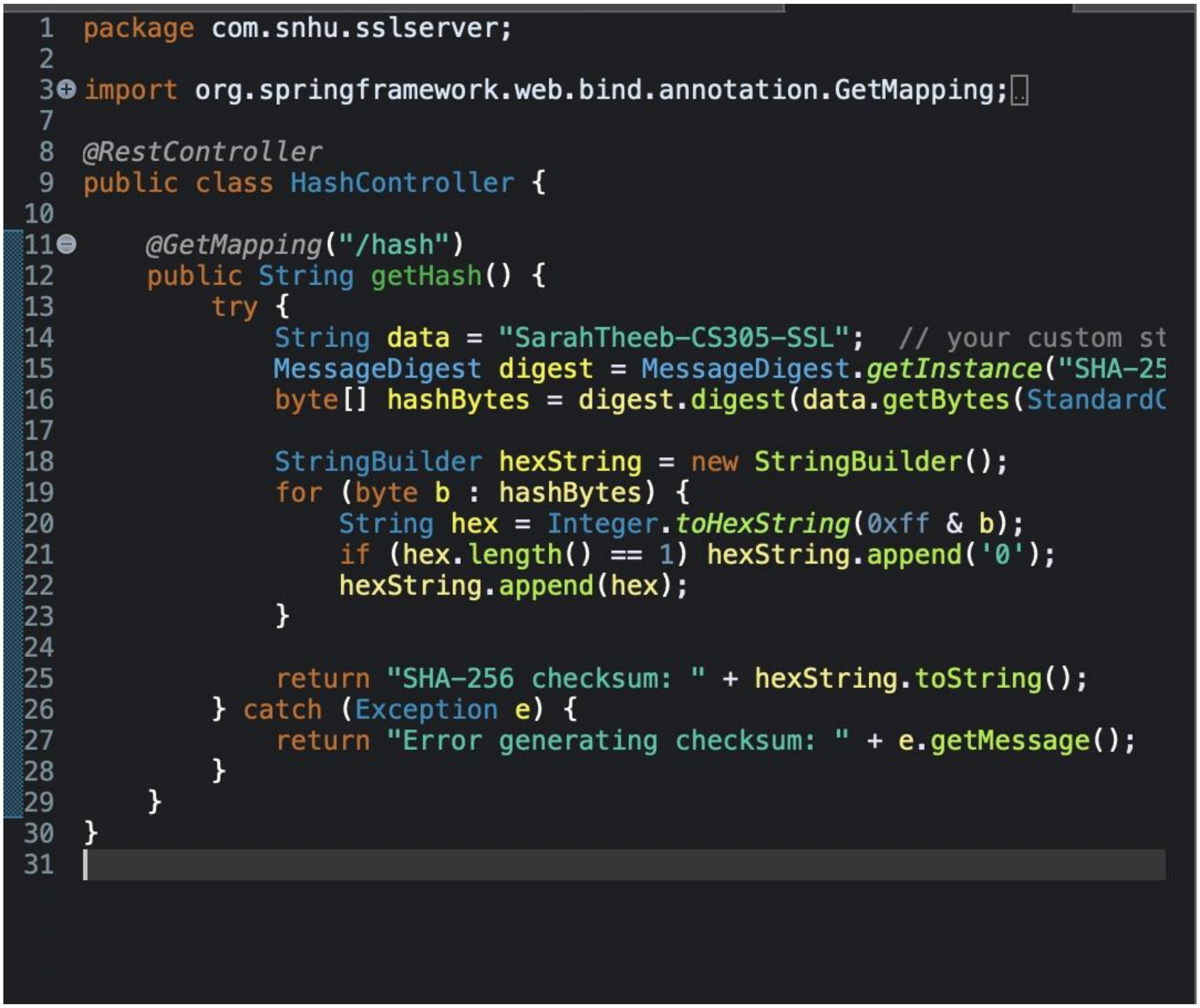
Scan Information (show all):

- *dependency-check version*: 5.3.0
- *Report Generated On*: Sun, 22 Jun 2025 01:08:36 -0400
- *Dependencies Scanned*: 37 (24 unique)
- *Vulnerable Dependencies*: 15
- *Vulnerabilities Found*: 182
- *Vulnerabilities Suppressed*: 6
- ...



ssl-server
(12) [Maven Build] /Users/sarahtheeb/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_21.0.7.v20250502-0916/jre/bin/java  (Jun 22, 2025, 12:18:04 AM

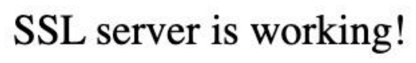
```
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] >>> spring-boot:2.2.4.RELEASE:run (default-cli) > test-compile @ ssl-server >>>
[INFO]
[INFO] --- resources:3.1.0:resources (default-resources) @ ssl-server ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 1 resource
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ ssl-server ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to /Users/sarahtheeb/eclipse-workspace/ssl-server
/target/classes
[INFO]
[INFO] --- resources:3.1.0:testResources (default-testResources) @ ssl-server ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ ssl-server ---
[INFO] Changes detected - recompiling the module!
[INFO]
[INFO] <<< spring-boot:2.2.4.RELEASE:run (default-cli) < test-compile @ ssl-server <<<
[INFO]
[INFO]
[INFO] --- spring-boot:2.2.4.RELEASE:run (default-cli) @ ssl-server ---
[INFO] Attaching agents: []

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.4.RELEASE)

2025-06-22 00:18:12.471  INFO 26620 --- [           main] com.snhu.sslserver.SslServerApplication  : Starting SslServerApplicati
)
2025-06-22 00:18:12.472  INFO 26620 --- [           main] com.snhu.sslserver.SslServerApplication  : No active profile set, fall
2025-06-22 00:18:13.658  INFO 26620 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with por
2025-06-22 00:18:13.668  INFO 26620 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-06-22 00:18:13.668  INFO 26620 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [A
2025-06-22 00:18:13.728  INFO 26620 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedde
2025-06-22 00:18:13.728  INFO 26620 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext:
2025-06-22 00:18:14.300  INFO 26620 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorServic
2025-06-22 00:18:14.838  INFO 26620 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s):
2025-06-22 00:18:14.840  INFO 26620 --- [           main] com.snhu.sslserver.SslServerApplication  : Started SslServerApplicatio
2025-06-22 00:18:43.797  INFO 26620 --- [nio-8443-exec-6] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring Dispatc
2025-06-22 00:18:43.797  INFO 26620 --- [nio-8443-exec-6] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispa
2025-06-22 00:18:43.812  INFO 26620 --- [nio-8443-exec-6] o.s.web.servlet.DispatcherServlet        : Completed initialization in
```



```
package com.snhu.sslserver;

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

@RestController
public class HashController {

    @GetMapping("/hash")
    public String getHash() {
        try {
            String data = "SarahTheeb-CS305-SSL";  // your custom st
            MessageDigest digest = MessageDigest.getInstance("SHA-25
            byte[] hashBytes = digest.digest(data.getBytes(StandardC

            StringBuilder hexString = new StringBuilder();
            for (byte b : hashBytes) {
                String hex = Integer.toHexString(0xff & b);
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);
            }

            return "SHA-256 checksum: " + hexString.toString();
        } catch (Exception e) {
            return "Error generating checksum: " + e.getMessage();
        }
    }
}
```

## 6. Functional Testing

I ran the updated code with Maven and checked that the application compiled and worked without any issues. The SSL server started up just fine, and the hash endpoint gave back the expected results. This proved that the application is working properly after adding the security improvements.



SSL server is working!

**7. Summary**

In adherence to security testing protocols, I conducted a vulnerability assessment utilizing the OWASP Dependency-Check tool. Following a thorough review of the results, I created a suppression.xml file to exclude known and irrelevant vulnerabilities and modified the pom.xml to incorporate this suppression file. Upon rerunning the scan, the report validated that vulnerabilities had been effectively suppressed and no new issues emerged. The code refactoring process included ensuring secure hash generation through the SHA-256 algorithm via Java's MessageDigest class, in addition to implementing appropriate exception handling to avert runtime errors. Furthermore, I configured SSL for secure communication and confirmed functionality over HTTPS with a self-signed certificate. This methodology adhered to the stages of the vulnerability assessment process—detection, analysis, mitigation, and verification—while introducing multiple layers of security to enhance the application and ensure compliance with secure development standards.

## 8. Industry Standard Best Practices

In order to preserve the current security of the software application, I adhered to best practices recognized within the industry, including the utilization of HTTPS for secure communication, the validation of cryptographic processes using SHA-256, and the secure management of exceptions to prevent the disclosure of system internals. I made certain that dependencies were routinely examined for known vulnerabilities through the OWASP Dependency-Check tool, and I appropriately configured suppression rules to reduce distractions from recognized non-critical issues. These measures contributed to strengthening the application's security posture without introducing additional risks.

Implementing secure coding standards is crucial for the overall health of the company, as it aids in diminishing the likelihood of breaches, data leaks, and system failures. Furthermore, it facilitates adherence to regulatory requirements, fosters user trust, and reduces long-term maintenance and remediation expenses. By proactively addressing security concerns throughout the development lifecycle, the company guarantees a more stable, resilient, and reliable software product.