GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	10/14/2025	Joseph Limbert	

Client

ARTEMIS
FINANCIAL

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.

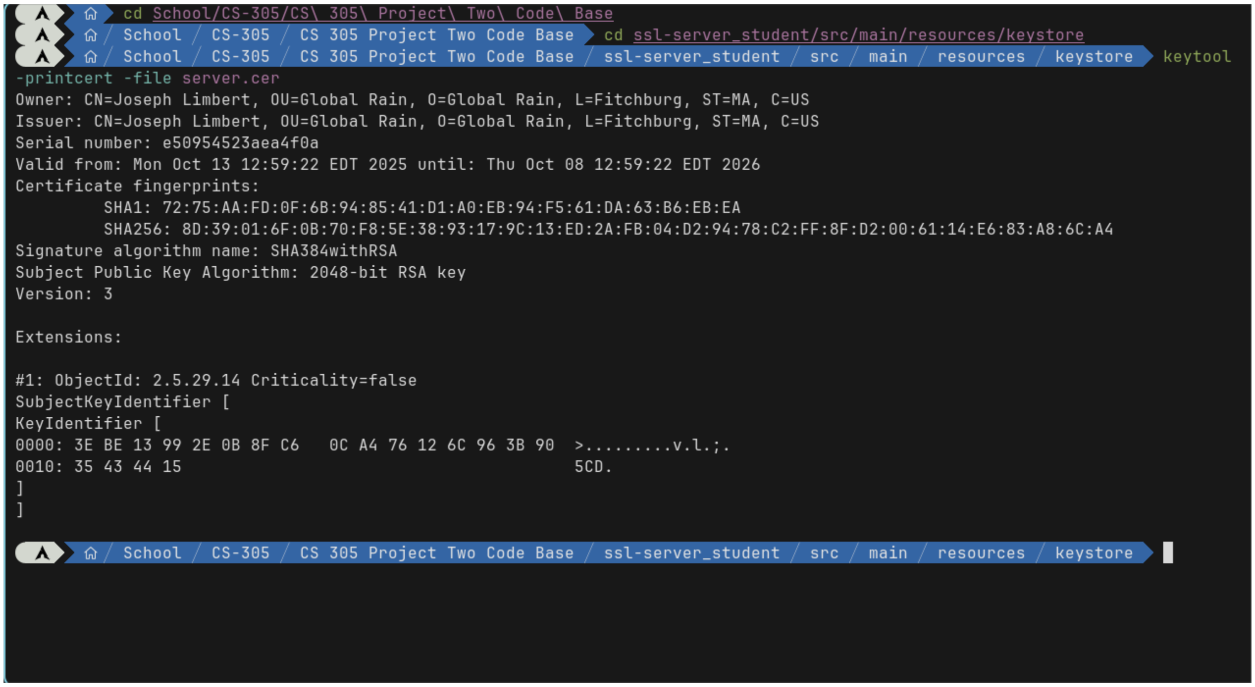
Developer
Joseph Limbert

1. Algorithm Cipher

I recommend using the SHA-256 algorithm cipher for encryption in this application. The SHA 256 algorithm is a hashing algorithm that was designed by the NSA to be a secure hashing algorithm. It is widely used in many industries to verify file integrity, verify SSL and TSL certificate integrity, and password hashing. It works by padding, then dividing the input into multiple blocks and performing various operations on the blocks to transform and compress them to create a 256-bit hash value. Collision in a cryptographic hash function refers to having two different inputs that create the same hash value. Avoiding collision is very important to ensure the integrity of the hash. The checksum hash works for verification because only the unique input is capable of creating the unique hash. If more than one input can create the same hash, then there is no way to truly verify that the data hasn't been corrupted. SHA-256 was designed to be collision resistant and secure. This makes it a good choice for this application. Encryption algorithms have gone through many changes over the years as computing power has increased. No algorithm is impenetrable however the strength of an algorithm is in the time it conceivably would take to be broken. As computers become more sophisticated and powerful, the need for stronger algorithms will increase.

2. Certificate Generation

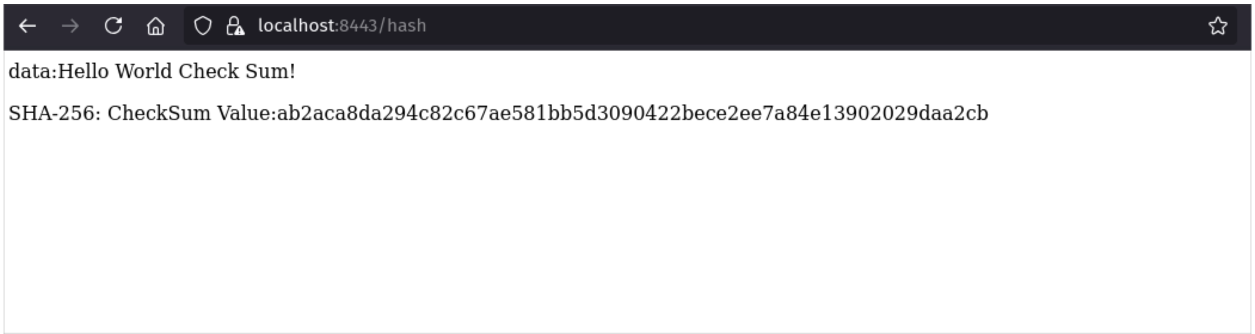
Insert a screenshot below of the CER file.



```
⌂ cd School/CS-305/CS\ 305\ Project\ Two\ Code\ Base
⌂ / School / CS-305 / CS 305 Project Two Code Base  cd ssl-server_student/src/main/resources/keystore
⌂ / School / CS-305 / CS 305 Project Two Code Base / ssl-server_student / src / main / resources / keystore  keytool
-printcert -file server.cer
Owner: CN=Joseph Limbert, OU=Global Rain, O=Global Rain, L=Fitchburg, ST=MA, C=US
Issuer: CN=Joseph Limbert, OU=Global Rain, O=Global Rain, L=Fitchburg, ST=MA, C=US
Serial number: e50954523aea4f0a
Valid from: Mon Oct 13 12:59:22 EDT 2025 until: Thu Oct 08 12:59:22 EDT 2026
Certificate fingerprints:
         SHA1: 72:75:AA:FD:0F:6B:94:85:41:D1:A0:EB:94:F5:61:DA:63:B6:EB:EA
         SHA256: 8D:39:01:6F:0B:70:F8:5E:38:93:17:9C:13:ED:2A:FB:04:D2:94:78:C2:FF:8F:D2:00:61:14:E6:83:A8:6C:A4
Signature algorithm name: SHA384withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 3E BE 13 99 2E 0B 8F C6   0C A4 76 12 6C 96 3B 90  >.........v.l.;.
0010: 35 43 44 15                                        5CD.
]
]

⌂ / School / CS-305 / CS 305 Project Two Code Base / ssl-server_student / src / main / resources / keystore
```

3. Deploy Cipher

Insert a screenshot below of the checksum verification.

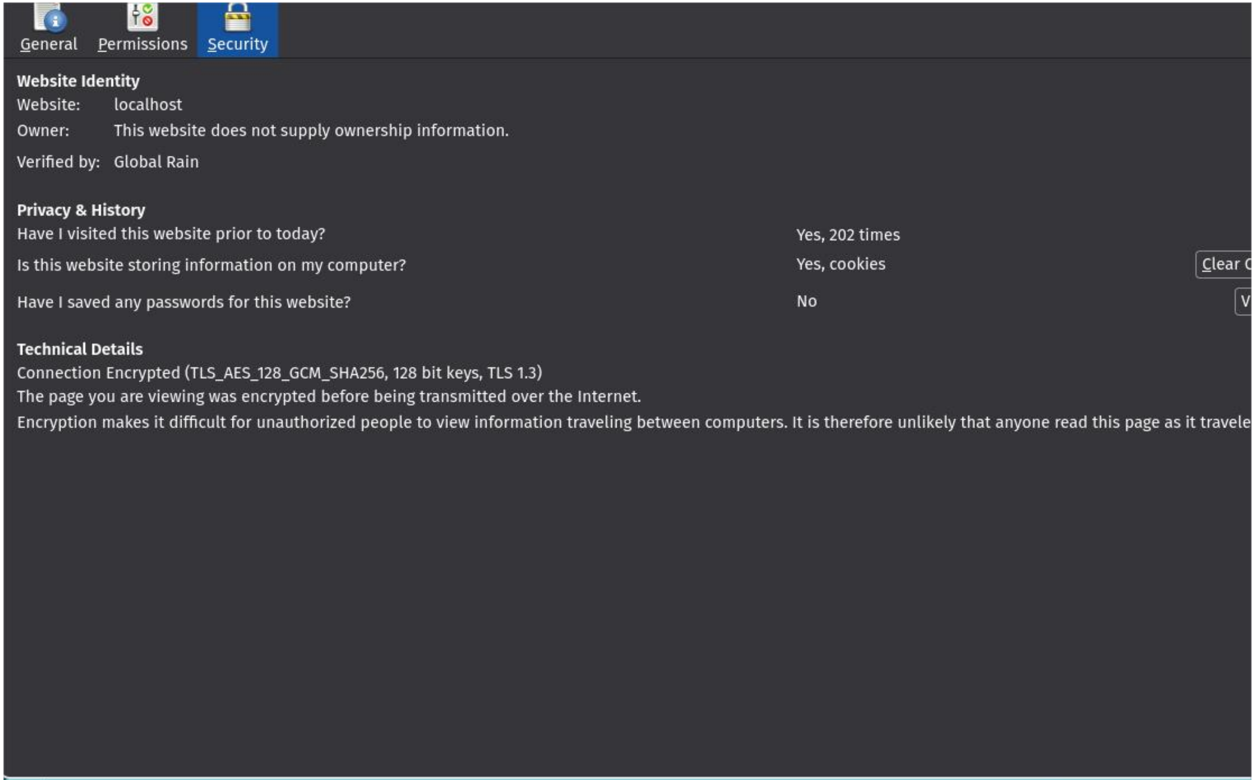


data:Hello World Check Sum!

SHA-256: CheckSum Value:ab2aca8da294c82c67ae581bb5d3090422bece2ee7a84e13902029daa2cb

4. Secure Communications

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



Website Identity

Website: localhost

Owner: This website does not supply ownership information.

Verified by: Global Rain

Privacy & History

Have I visited this website prior to today? Yes, 202 times

Is this website storing information on my computer? Yes, cookies

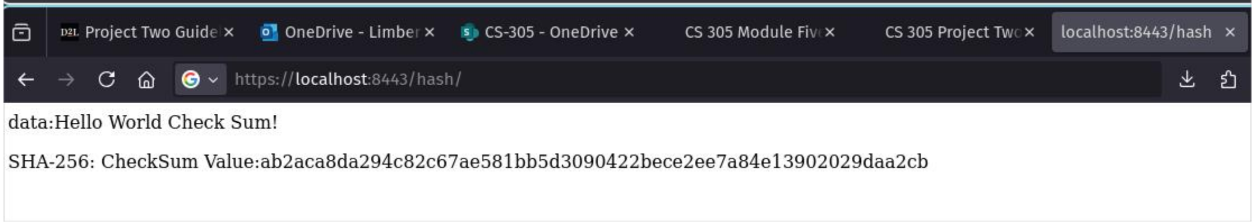
Have I saved any passwords for this website? No

Technical Details

Connection Encrypted (TLS_AES_128_GCM_SHA256, 128 bit keys, TLS 1.3)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it travele

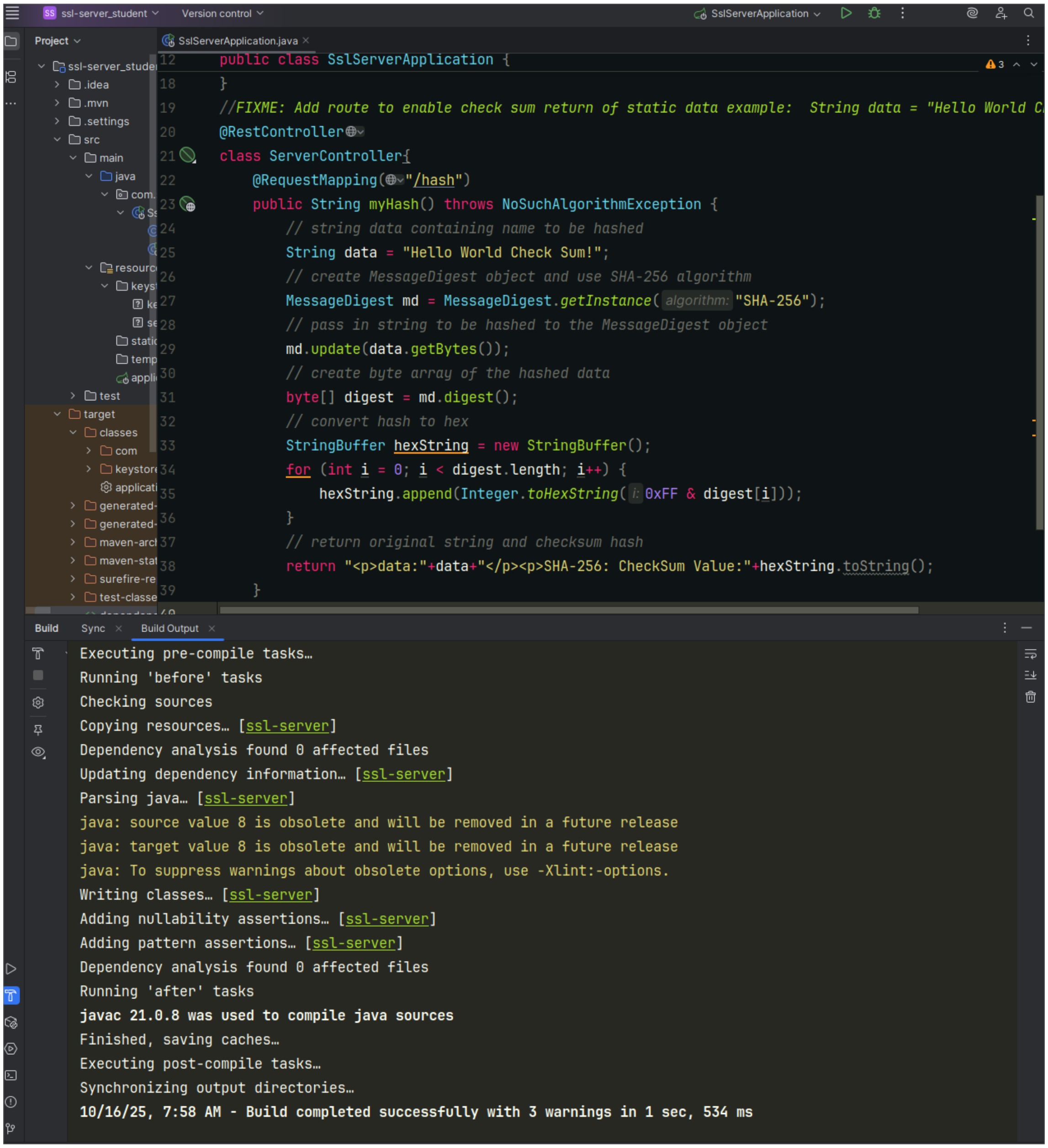


data:Hello World Check Sum!

SHA-256: CheckSum Value:ab2aca8da294c82c67ae581bb5d3090422bece2ee7a84e13902029daa2cb

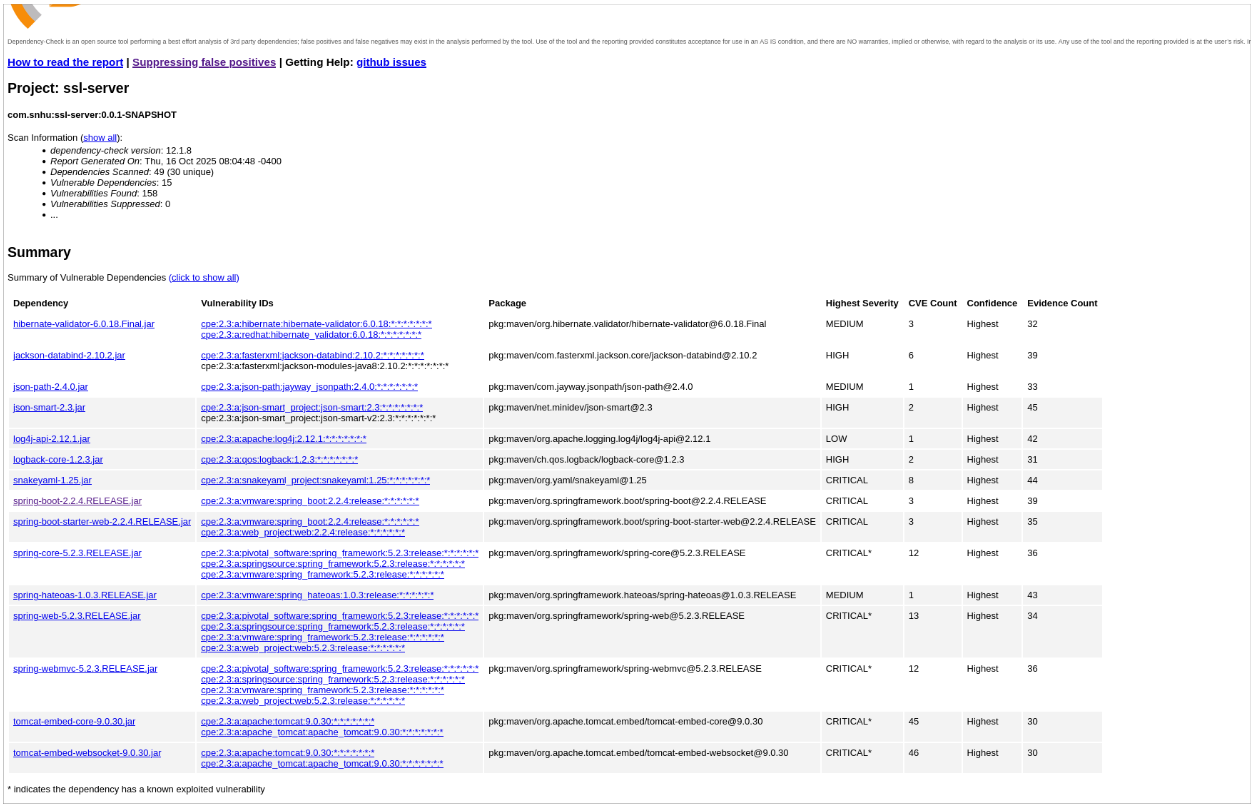
5. Secondary Testing

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



```
public class SslServerApplication {
}
//FIXME: Add route to enable check sum return of static data example:  String data = "Hello World C
@RestController
class ServerController{
    @RequestMapping("/hash")
    public String myHash() throws NoSuchAlgorithmException {
        // string data containing name to be hashed
        String data = "Hello World Check Sum!";
        // create MessageDigest object and use SHA-256 algorithm
        MessageDigest md = MessageDigest.getInstance( algorithm: "SHA-256");
        // pass in string to be hashed to the MessageDigest object
        md.update(data.getBytes());
        // create byte array of the hashed data
        byte[] digest = md.digest();
        // convert hash to hex
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < digest.length; i++) {
            hexString.append(Integer.toHexString( i: 0xFF & digest[i]));
        }
        // return original string and checksum hash
        return "<p>data:"+data+"</p><p>SHA-256: CheckSum Value:"+hexString.toString();
    }
```

```
Executing pre-compile tasks…
Running 'before' tasks
Checking sources
Copying resources… [ssl-server]
Dependency analysis found 0 affected files
Updating dependency information… [ssl-server]
Parsing java… [ssl-server]
java: source value 8 is obsolete and will be removed in a future release
java: target value 8 is obsolete and will be removed in a future release
java: To suppress warnings about obsolete options, use -Xlint:-options.
Writing classes… [ssl-server]
Adding nullability assertions… [ssl-server]
Adding pattern assertions… [ssl-server]
Dependency analysis found 0 affected files
Running 'after' tasks
javac 21.0.8 was used to compile java sources
Finished, saving caches…
Executing post-compile tasks…
Synchronizing output directories…
10/16/25, 7:58 AM - Build completed successfully with 3 warnings in 1 sec, 534 ms
```



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, and there 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. I

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*: 12.1.8
- *Report Generated On*: Thu, 16 Oct 2025 08:04:48 -0400
- *Dependencies Scanned*: 49 (30 unique)
- *Vulnerable Dependencies*: 15
- *Vulnerabilities Found*: 158
- *Vulnerabilities Suppressed*: 0
- ...

Summary

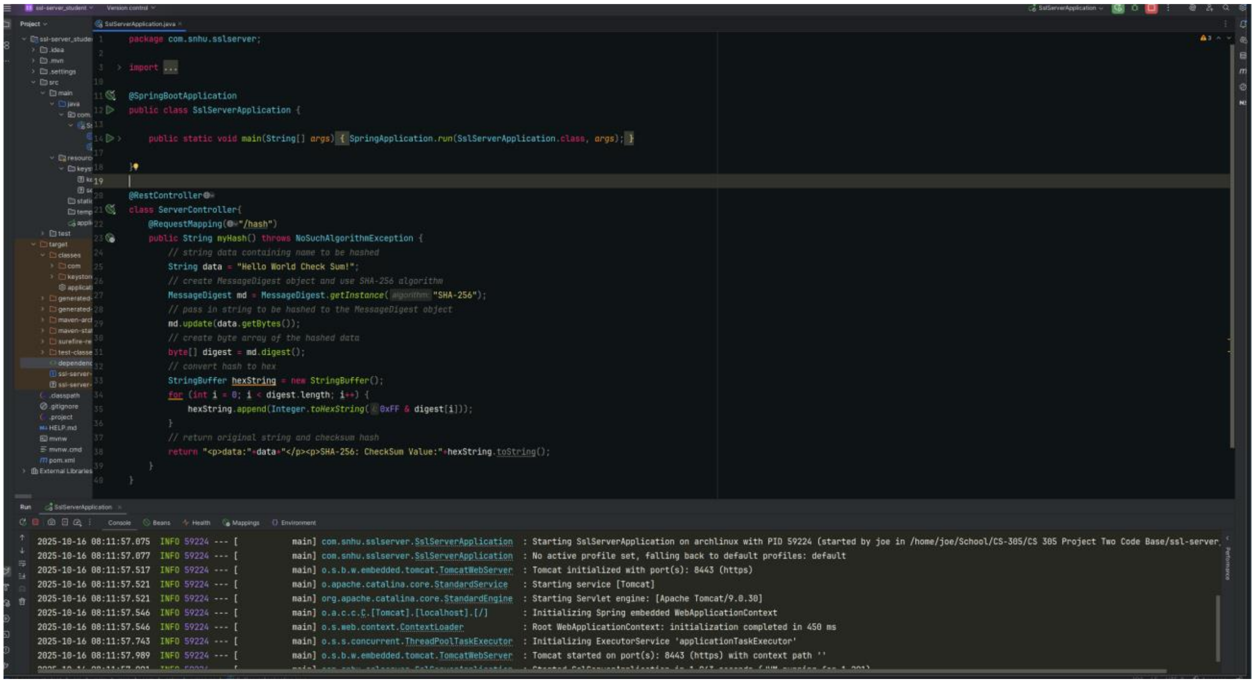
Summary of Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
hibernate-validator-6.0.18.Final.jar	cpe:2.3:a:hibernate:hibernate-validator:6.0.18:*:*:*:*:*:*:* cpe:2.3:a:redhat:hibernate_validator:6.0.18:*:*:*:*:*:*:*	pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final	MEDIUM	3	Highest	32
jackson-databind-2.10.2.jar	cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:*:*:*:* cpe:2.3:a:fasterxml:jackson-modules-java8:2.10.2:*:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2	HIGH	6	Highest	39
json-path-2.4.0.jar	cpe:2.3:a:json-path:jayway_jsonpath:2.4.0:*:*:*:*:*:*:*	pkg:maven/com.jayway.jsonpath/json-path@2.4.0	MEDIUM	1	Highest	33
json-smart-2.3.jar	cpe:2.3:a:json-smart_project:json-smart:2.3:*:*:*:*:*:*:* cpe:2.3:a:json-smart_project:json-smart-v2:2.3:*:*:*:*:*:*:*	pkg:maven/net.minidev/json-smart@2.3	HIGH	2	Highest	45
log4j-api-2.12.1.jar	cpe:2.3:a:apache:log4j:2.12.1:*:*:*:*:*:*:*	pkg:maven/org.apache.logging.log4j/log4j-api@2.12.1	LOW	1	Highest	42
logback-core-1.2.3.jar	cpe:2.3:a:qos:logback:1.2.3:*:*:*:*:*:*:*	pkg:maven/ch.qos.logback/logback-core@1.2.3	HIGH	2	Highest	31
snakeyaml-1.25.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.25	CRITICAL	8	Highest	44
spring-boot-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot@2.2.4.RELEASE	CRITICAL	3	Highest	39
spring-boot-starter-web-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:*:* cpe:2.3:a:web_project:web:2.2.4:release:*:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot-starter-web@2.2.4.RELEASE	CRITICAL	3	Highest	35
spring-core-5.2.3.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:*	pkg:maven/org.springframework/spring-core@5.2.3.RELEASE	CRITICAL*	12	Highest	36
spring-hateoas-1.0.3.RELEASE.jar	cpe:2.3:a:vmware:spring_hateoas:1.0.3:release:*:*:*:*:*:*	pkg:maven/org.springframework.hateoas/spring-hateoas@1.0.3.RELEASE	MEDIUM	1	Highest	43
spring-web-5.2.3.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:web_project:web:5.2.3:release:*:*:*:*:*:*	pkg:maven/org.springframework/spring-web@5.2.3.RELEASE	CRITICAL*	13	Highest	34
spring-webmvc-5.2.3.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:vmware:spring_framework:5.2.3:release:*:*:*:*:*:* cpe:2.3:a:web_project:web:5.2.3:release:*:*:*:*:*:*	pkg:maven/org.springframework/spring-webmvc@5.2.3.RELEASE	CRITICAL*	12	Highest	36
tomcat-embed-core-9.0.30.jar	cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:* cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30:*:*:*:*:*:*:*	pkg:maven/org.apache.tomcat.embed/tomcat-embed-core@9.0.30	CRITICAL*	45	Highest	30
tomcat-embed-websocket-9.0.30.jar	cpe:2.3:a:apache:tomcat:9.0.30:*:*:*:*:*:*:* cpe:2.3:a:apache_tomcat:apache_tomcat:9.0.30:*:*:*:*:*:*:*	pkg:maven/org.apache.tomcat.embed/tomcat-embed-websocket@9.0.30	CRITICAL*	46	Highest	30

* indicates the dependency has a known exploited vulnerability

6. Functional Testing

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



```java
package com.snhu.sslserver;

import ...

@SpringBootApplication
public class SslServerApplication {

    public static void main(String[] args) { SpringApplication.run(SslServerApplication.class, args); }

}

@RestController
class ServerController{
    @RequestMapping("/hash")
    public String myHash() throws NoSuchAlgorithmException {
        // string data containing name to be hashed
        String data = "Hello World Check Sum!";
        // create MessageDigest object and use SHA-256 algorithm
        MessageDigest md = MessageDigest.getInstance( algorithm: "SHA-256");
        // pass in string to be hashed to the MessageDigest object
        md.update(data.getBytes());
        // create byte array of the hashed data
        byte[] digest = md.digest();
        // convert hash to hex
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < digest.length; i++) {
            hexString.append(Integer.toHexString(0xFF & digest[i]));
        }
        // return original string and checksum hash
        return "<p>data:"+data+"</p><p>SHA-256: CheckSum Value:"+hexString.toString();
    }
}
```

```
2025-10-16 08:11:57.075  INFO 59224 --- [           main] com.snhu.sslserver.SslServerApplication  : Starting SslServerApplication on archlinux with PID 59224 (started by joe in /home/joe/School/CS-305/CS 305 Project Two Code Base/ssl-server
2025-10-16 08:11:57.077  INFO 59224 --- [           main] com.snhu.sslserver.SslServerApplication  : No active profile set, falling back to default profiles: default
2025-10-16 08:11:57.517  INFO 59224 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2025-10-16 08:11:57.521  INFO 59224 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-10-16 08:11:57.521  INFO 59224 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2025-10-16 08:11:57.546  INFO 59224 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-10-16 08:11:57.546  INFO 59224 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 450 ms
2025-10-16 08:11:57.743  INFO 59224 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2025-10-16 08:11:57.989  INFO 59224 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
```

7. Summary

In the refactored code, I created a keystore and self-signed certificate using the java keytool. I then configured tomcat to use the self-signed certificate and enabled an https connection to ensure secure communications. I also created an endpoint for hashing data and returning a checksum for verification using the secure https server. Running a dependency check after did find vulnerabilities however these were vulnerabilities that previously existed and weren't introduced from the refactored code. Manually reviewing the code, I didn't find any security vulnerabilities either. The code is pretty simple containing only the endpoint I created for hashing a checksum. The server itself runs on a secure https connection using the self-signed certificate I generated.

8. Industry Standard Best Practices

I used industry standards best practices by securing the connection with https and ssl. We also make sure we aren't sending any sensitive information through headers and checking dependencies for vulnerabilities. Industry standard best practices exist because they are proven methods for limiting the vulnerabilities in your application and enhancing security. Using the best practices brings a lot of value by securing our application and enabling future developers to understand how the application is secured since they should also be able to identify the best practices used. As these standards evolve, the application will have to be reviewed. Software security is always evolving, and we must remain vigilant to ensure our software stays secure.