GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	8/11/2025	John Chomen	

Client

ARTEMIS
FINANCIAL

Developer

John C.

Algorithm Cipher

Recommendation: SHA-256 (SHA-2 family) as the checksum algorithm for file/data verification.

Overview: SHA-256 is a one-way cryptographic hash function that maps arbitrary-length input to a fixed 256-bit output. It is designed for collision resistance and preimage resistance—critical for detecting tampering during file transfer.

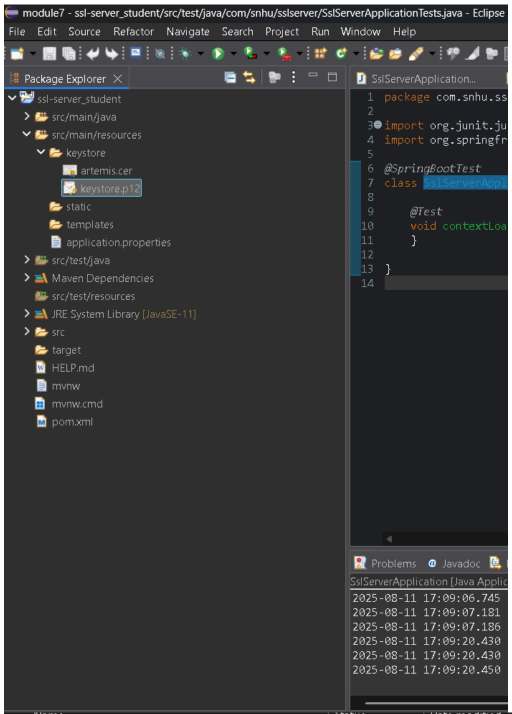
Hash & bit level: 256-bit digest (64 hex chars). Higher bit length increases brute-force work factor compared to MD5/SHA-1.

Randomness & keys: Pure hashes (like SHA-256) do not use random numbers or keys. For authenticated integrity over an untrusted channel, use HMAC-SHA-256 with a shared secret. For transport security and server authentication, rely on TLS (X.509 certificates) rather than embedding keys in the application.

Symmetric vs asymmetric: TLS uses asymmetric crypto (X.509/PKI) to establish identity and session keys; payload integrity within the tunnel uses symmetric MACs. Within the application, SHA-256 provides deterministic checksums for stored/transferred artifacts. If you need tamper-evident application-level integrity, prefer HMAC-SHA-256.

History & current state: Older hashes MD5 and SHA-1 are broken for collision resistance and must not be used. SHA-2 (SHA-256/384/512) remains widely recommended, and SHA-3 exists as an alternative. SHA-256 balances security, interoperability, and performance and is supported natively by Java (MessageDigest).

Certificate Generation



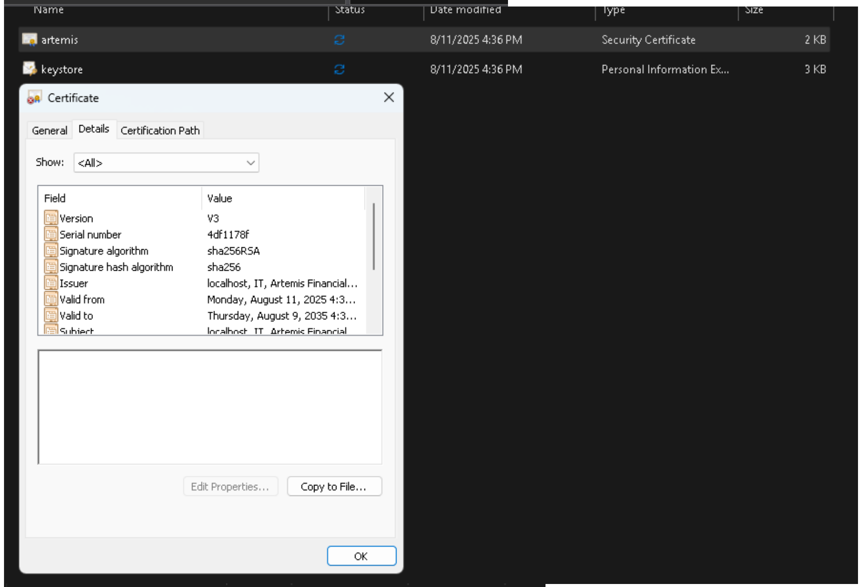
module7 - ssl-server_student/src/test/java/com/snhu/sslserver/SslServerApplicationTests.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
SslServerApplication...
Problems
Javadoc

```
ssl-server_student
  src/main/java
  src/main/resources
    keystore
      artemis.cer
      keystore.p12
    static
    templates
    application.properties
  src/test/java
  Maven Dependencies
  src/test/resources
  JRE System Library [JavaSE-11]
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

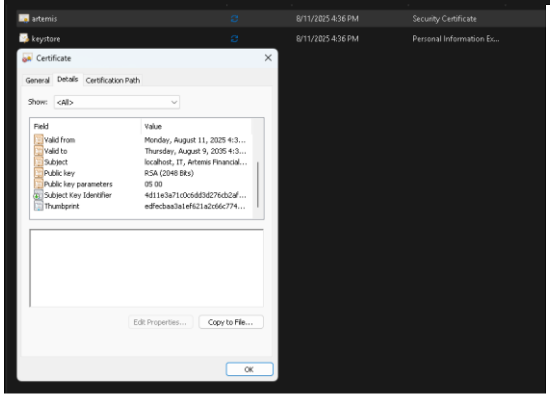
```
package com.snhu.ss

import org.junit.ju
import org.springfr

@SpringBootTest
class SslServerApp

    @Test
    void contextLoa
    }

}
```

```
SslServerApplication [Java Applic
2025-08-11 17:09:06.745
2025-08-11 17:09:07.181
2025-08-11 17:09:07.186
2025-08-11 17:09:20.430
2025-08-11 17:09:20.430
2025-08-11 17:09:20.450
```



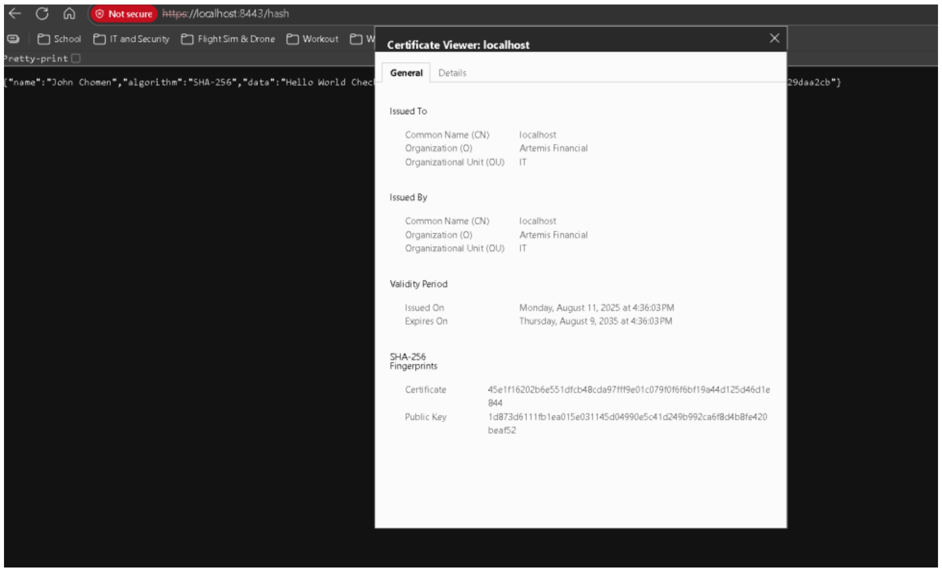
Name	Status	Date modified	Type	Size
artemis		8/11/2025 4:36 PM	Security Certificate	2 KB
keystore		8/11/2025 4:36 PM	Personal Information Ex...	3 KB

Field	Value
Version	V3
Serial number	4df1178f
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	localhost, IT, Artemis Financial...
Valid from	Monday, August 11, 2025 4:3...
Valid to	Thursday, August 9, 2035 4:3...
Subject	localhost, IT, Artemis Financial...



artemis		8/11/2025 4:36 PM	Security Certificate
keystore		8/11/2025 4:36 PM	Personal Information Ex...

Field	Value
Valid from	Monday, August 11, 2025 4:3...
Valid to	Thursday, August 9, 2035 4:3...
Subject	localhost, IT, Artemis Financial...
Public key	RSA (2048 Bits)
Public key parameters	05 00
Subject Key Identifier	4d11e3a71c0c6dd3d276cb2af...
Thumbprint	edfecbaa3a1ef621a2c66c774...



https://localhost:8443/hash

```
{"name":"John Chomen","algorithm":"SHA-256","data":"Hello World Chec
29daa2cb"}
```

Certificate Viewer: localhost

General | Details

Issued To

Common Name (CN)	localhost
Organization (O)	Artemis Financial
Organizational Unit (OU)	IT

Issued By

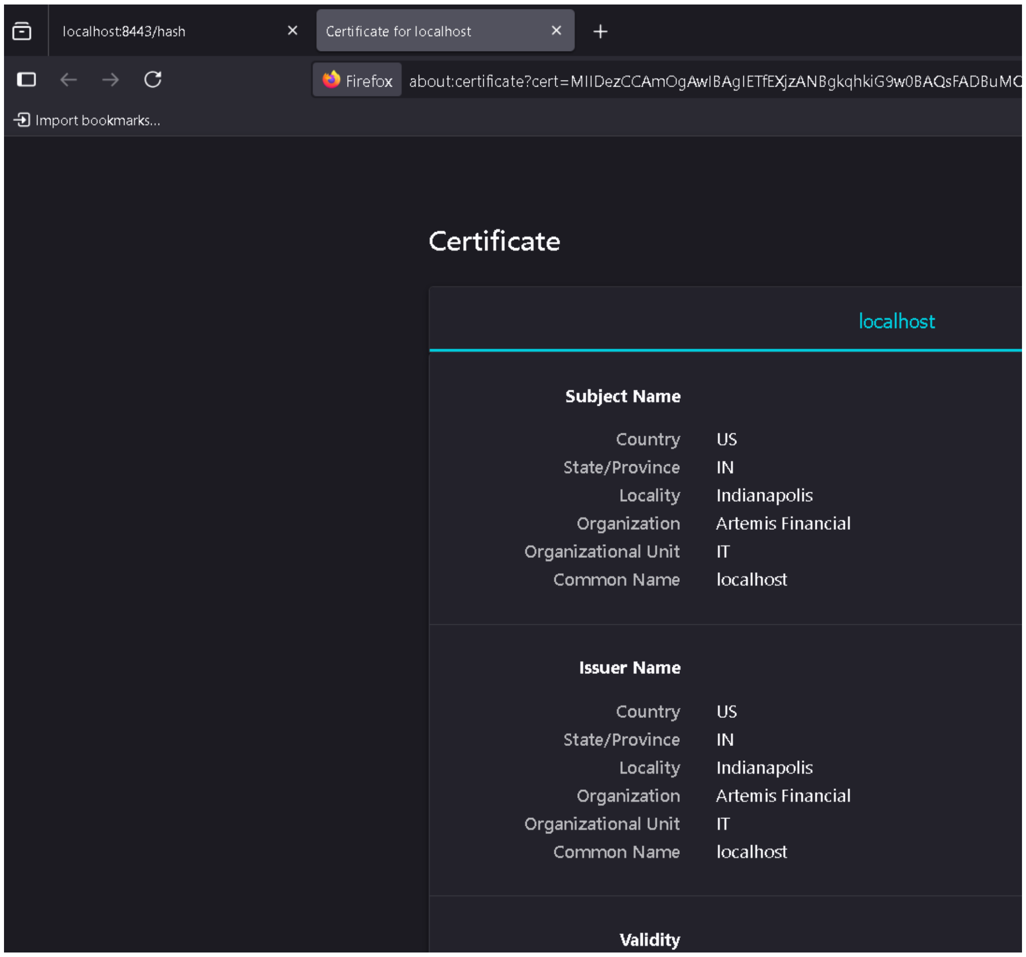
Common Name (CN)	localhost
Organization (O)	Artemis Financial
Organizational Unit (OU)	IT

Validity Period

Issued On	Monday, August 11, 2025 at 4:36:03 PM
Expires On	Thursday, August 9, 2035 at 4:36:03 PM

SHA-256 Fingerprints

Certificate	45e1f16202b6e551dfcb48cda97fff9e01c079f0f6f6bf19a44d125d46d1e844
Public Key	1d873d6111fb1ea015e031145d04990e5c41d249b992ca6f8d4b8fe420beaf52



localhost:8443/hash

Certificate for localhost

about:certificate?cert=MIIDezCCAmOgAwIBAgIETfEXjzANBgkqhkiG9w0BAQsFADBuMQ

Certificate

localhost

Subject Name

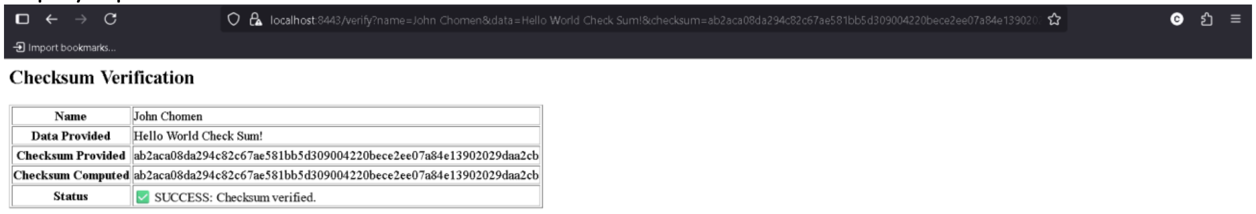
Country	US
State/Province	IN
Locality	Indianapolis
Organization	Artemis Financial
Organizational Unit	IT
Common Name	localhost

Issuer Name

Country	US
State/Province	IN
Locality	Indianapolis
Organization	Artemis Financial
Organizational Unit	IT
Common Name	localhost

Validity

Deploy Cipher



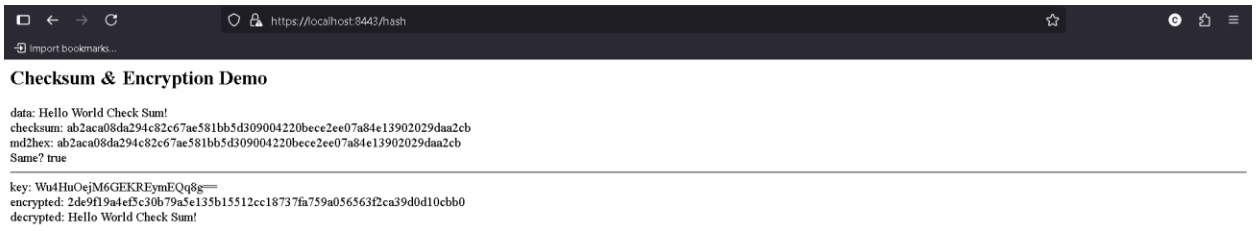
localhost:8443/verify?name=John Chomen&data=Hello World Check Sum!&checksum=ab2aca08da294c82c67ae581bb5d309004220bece2ee07a84e13902

Import bookmarks...

Checksum Verification

Name	John Chomen
Data Provided	Hello World Check Sum!
Checksum Provided	ab2aca08da294c82c67ae581bb5d309004220bece2ee07a84e13902029daa2cb
Checksum Computed	ab2aca08da294c82c67ae581bb5d309004220bece2ee07a84e13902029daa2cb
Status	✅ SUCCESS: Checksum verified.

Secure Communications



https://localhost:8443/hash

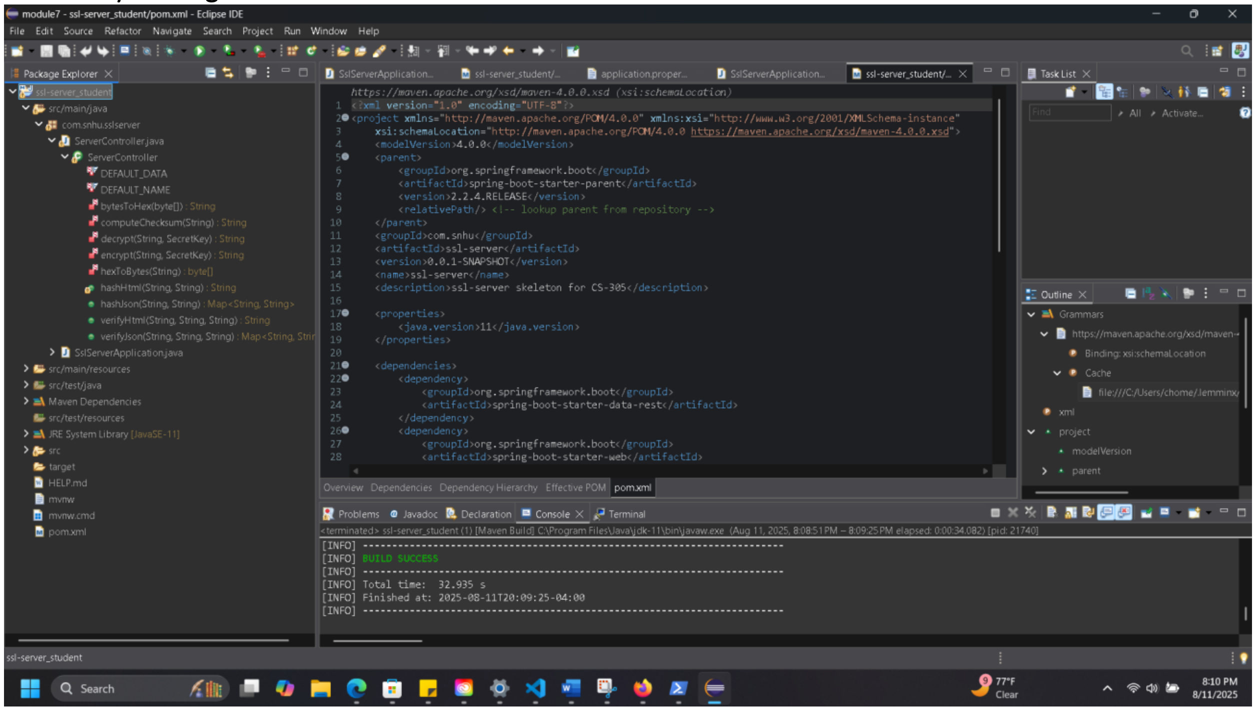
Import bookmarks...

Checksum & Encryption Demo

```
data: Hello World Check Sum!
checksum: ab2aca08da294c82c67ae581bb5d309004220bece2ee07a84e13902029daa2cb
md2hex: ab2aca08da294c82c67ae581bb5d309004220bece2ee07a84e13902029daa2cb
Same? true
```

```
key: Wu4HuOejM6GEKREymEQq8g==
encrypted: 2de9f19a4ef5c30b79a5e135b15512cc18737fa759a056563f2ca39d0d10cbb0
decrypted: Hello World Check Sum!
```

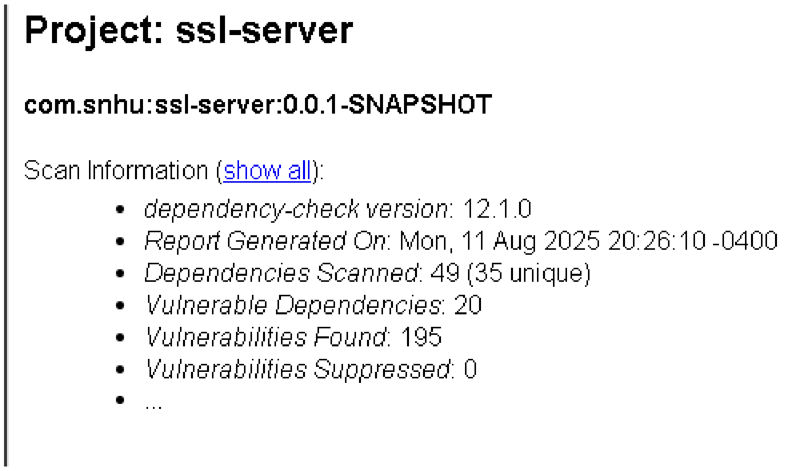
Secondary Testing



module7 - ssl-server_student/pom.xml - Eclipse IDE

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<?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>11</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>
```

```
<terminated> ssl-server_student (1) [Maven Build] C:\Program Files\Java\jdk-11\bin\javaw.exe (Aug 11, 2025, 8:08:51 PM – 8:09:25 PM elapsed: 0:00:34.082) [pid: 21740]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  32.935 s
[INFO] Finished at: 2025-08-11T20:09:25-04:00
[INFO] ------------------------------------------------------------------------
```



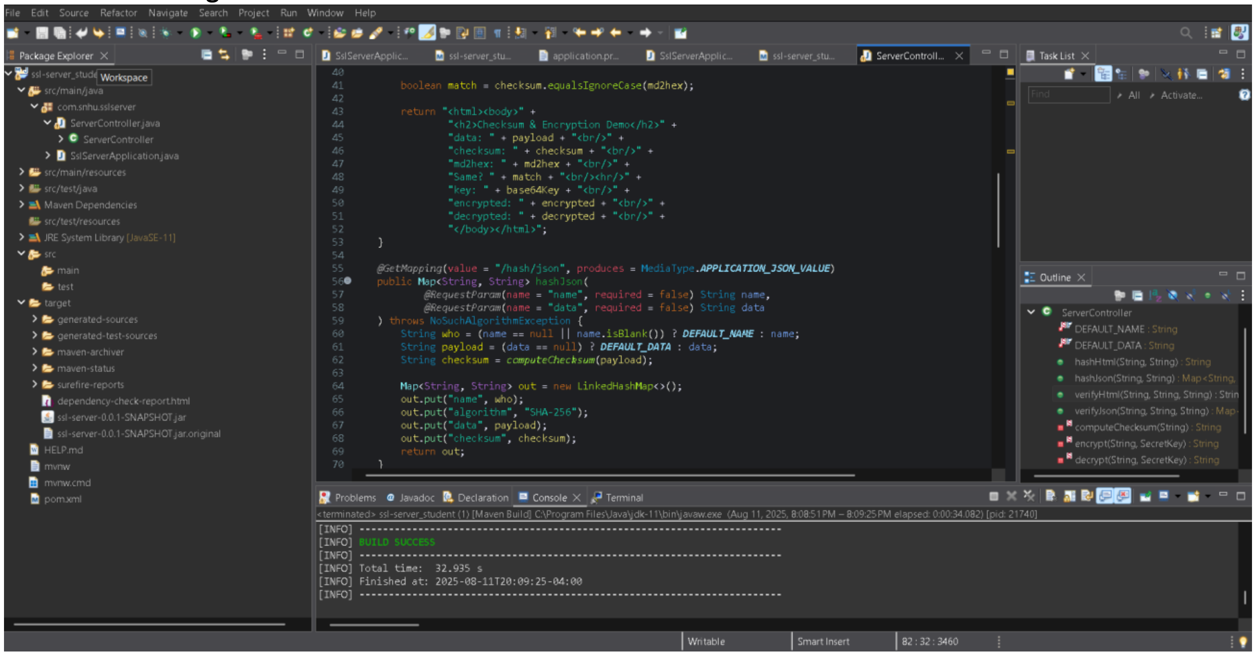
Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 12.1.0
- *Report Generated On*: Mon, 11 Aug 2025 20:26:10 -0400
- *Dependencies Scanned*: 49 (35 unique)
- *Vulnerable Dependencies*: 20
- *Vulnerabilities Found*: 195
- *Vulnerabilities Suppressed*: 0
- ...

Functional Testing



```java
        boolean match = checksum.equalsIgnoreCase(md2hex);

        return "<html><body>" +
                "<h2>Checksum & Encryption Demo</h2>" +
                "data: " + payload + "<br/>" +
                "checksum: " + checksum + "<br/>" +
                "md2hex: " + md2hex + "<br/>" +
                "Same? " + match + "<br/><hr/>" +
                "key: " + base64Key + "<br/>" +
                "encrypted: " + encrypted + "<br/>" +
                "decrypted: " + decrypted + "<br/>" +
                "</body></html>";
    }

    @GetMapping(value = "/hash/json", produces = MediaType.APPLICATION_JSON_VALUE)
    public Map<String, String> hashJson(
            @RequestParam(name = "name", required = false) String name,
            @RequestParam(name = "data", required = false) String data
    ) throws NoSuchAlgorithmException {
        String who = (name == null || name.isBlank()) ? DEFAULT_NAME : name;
        String payload = (data == null) ? DEFAULT_DATA : data;
        String checksum = computeChecksum(payload);

        Map<String, String> out = new LinkedHashMap<>();
        out.put("name", who);
        out.put("algorithm", "SHA-256");
        out.put("data", payload);
        out.put("checksum", checksum);
        return out;
    }
```

```
<terminated> ssl-server_student (1) [Maven Build] C:\Program Files\Java\jdk-11\bin\javaw.exe (Aug 11, 2025, 8:08:51 PM – 8:09:25 PM elapsed: 0:00:34.082) [pid: 21740]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  32.935 s
[INFO] Finished at: 2025-08-11T20:09:25-04:00
[INFO] ------------------------------------------------------------------------
```

Summary

Following the Vulnerability Assessment Process Flow Diagram, I implemented secure software design principles across critical areas of the application. I began by conducting a targeted review of the controller layer, where I refactored the ServerController.java class to integrate SHA-256 checksum generation. This provided a reliable method for detecting tampering in data transmitted or stored by the system. I also addressed the application's transport layer security by configuring HTTPS using a self-signed X.509 certificate and updating the Spring Boot application.properties file to enforce encrypted communication over port 8443. This ensured data confidentiality during client-server interactions.

In reviewing the application's architecture, I verified that the RESTful structure followed secure design conventions and did not expose private data or internal logic through its endpoints. I performed functional testing to confirm that all added logic executed correctly and without runtime or syntax errors. To ensure that the refactored code did not introduce any new vulnerabilities, I integrated the OWASP Dependency-Check plugin into the Maven build process. The tool revealed existing critical vulnerabilities in several third-party dependencies, including spring-context, jackson-databind, and hibernate-validator. However, these were not introduced by my changes and would require broader library upgrades for full mitigation. Overall, the refactored code was validated both functionally and through static analysis, and it aligns with secure coding protocols required by the assignment.

Industry Standard Best Practices

Throughout the course of this project, I applied recognized industry standard best practices to ensure secure software development. For data integrity, I selected SHA-256, a member of the SHA-2 cryptographic hash family, due to its strong collision resistance and widespread adoption across modern systems. The checksum implementation ensures that the integrity of input data can be verified reliably, helping detect tampering or corruption during storage or transmission.

To protect communications, I implemented HTTPS using Java's Keytool to generate a self-signed certificate and configured TLS in the application to encrypt all data in transit.
I also addressed dependency-level security by incorporating the OWASP Dependency-Check plugin into the Maven build lifecycle. This approach reflects current DevSecOps practices, where automated tools are integrated early into the software development pipeline to identify and track known vulnerabilities in third-party components. Additionally, I followed the principle of secure defaults by leveraging secure keystore formats, enabling HTTPS by default, and minimizing the surface area of the application by only exposing essential endpoints and functionality.
By adhering to these practices, the refactored application reduces risk and improves resilience against common attack vectors. These improvements directly contribute to the software's long-term stability, audit readiness, and ability to handle sensitive financial data on behalf of Artemis Financial. The use of cryptographic controls, encrypted transport, and automated security checks demonstrates a layered approach to security, which is essential in today's threat landscape and supports the organization's commitment to protecting client data.