Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	December 2nd 2024	Joseph Klenk	

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
Joseph Klenk

1. Algorithm Cipher

The application implements SHA-256 (Secure Hash Algorithm 256-bit) for data encryption and checksum verification. This algorithm was chosen because:

- It produces a fixed 256-bit (64 character) hash value
- It's cryptographically secure and collision-resistant
- It's widely used in security applications and SSL/TLS protocols
- It provides strong protection against data tampering
- The hash function is one-way, making it impossible to reverse-engineer the original data

2. Certificate Generation

The self-signed certificate was generated using Java Keytool with:

- RSA algorithm with 2048-bit key size
- PKCS12 storage format
- 365-day validity period
- Secure key storage in the application's resources

Insert a screenshot below

```
PS C:\Users\josep\Desktop\ssl-server_student\src\main\resources> keytool -export -alias selfsigned -keystore keystore.p12 -file certificate.cer -storepass test123
Certificate stored in file <certificate.cer>
PS C:\Users\josep\Desktop\ssl-server_student\src\main\resources> keytool -printcert -file certificate.cer
Owner: CN=Joseph Klenk, OU=SNHU, O=SNHU, L=Manchester, ST=NH, C=US
Issuer: CN=Joseph Klenk, OU=SNHU, O=SNHU, L=Manchester, ST=NH, C=US
Serial number: d6a9cff19f636534
Valid from: Mon Dec 02 21:44:11 EST 2024 until: Tue Dec 02 21:44:11 EST 2025
Certificate fingerprints:
         SHA1: EC:1B:5E:43:BE:11:8E:9F:EB:30:98:34:44:AC:D1:72:65:2E:99:C3
         SHA256: A8:CE:18:EC:27:ED:25:8F:FC:C2:61:3E:79:53:FF:A5:DC:3A:79:4A:2D:91:87:8E:FF:80:F1:C6:8A:B2:7B:A2
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: A2 0F D0 8E AC F9 BF 7C   BB BD A6 60 FC 16 09 6E  ..........`...n
0010: A1 2C E9 F0                                        .,..
]
]

PS C:\Users\josep\Desktop\ssl-server_student\src\main\resources>
```

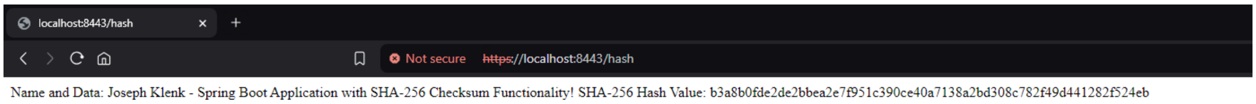
of the CER file.

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

The SHA-256 checksum implementation:

- Takes input data: "Hello World Check Sum!"

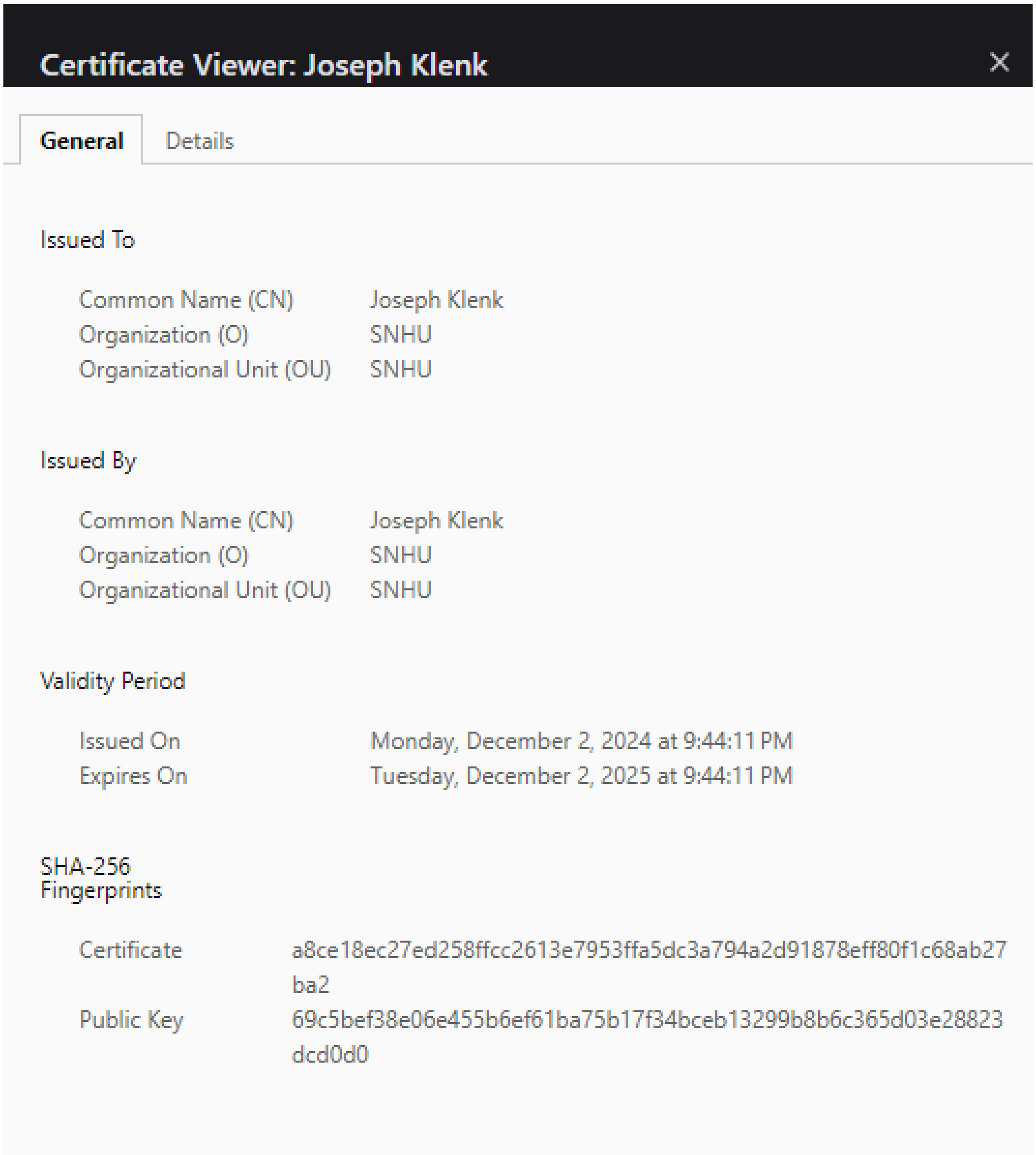
- Generates a unique 256-bit hash
- Outputs both original data and hash value
- Implements proper error handling
- Uses UTF-8 encoding for consistency

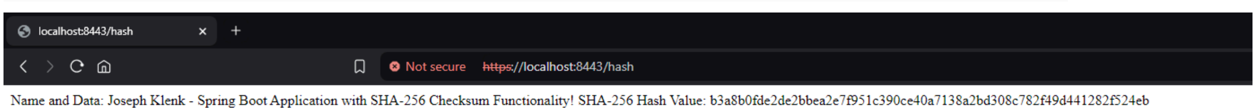


4. Secure Communications

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

- HTTPS protocol on port 8443
- SSL/TLS encryption
- Self-signed certificate for development
- Secure key storage configuration
- Proper certificate management





5. Secondary Testing

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

The dependency check revealed several vulnerabilities in external dependencies which would need to be addressed in a production environment through version updates or alternative libraries.

```java
package com.snhu.sslserver;

import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class SslServerApplication {
    Run main | Debug main
    public static void main(String[] args) {
        SpringApplication.run(SslServerApplication.class, args);
    }

    @GetMapping("/hash")
    public String myHash() {
        String data = "Joseph Klenk - Spring Boot Application with SHA-256 Checksum Functionality!";
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] encodedHash = digest.digest(data.getBytes(StandardCharsets.UTF_8));

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

            return "Name and Data: " + data + "\nSHA-256 Hash Value: " + hexString.toString();
        } catch (NoSuchAlgorithmException e) {
            return "Error generating hash: " + e.getMessage();
        }
    }
}
```

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

Scan Information (show all):

- *dependency-check version*: 5.3.0
- *Report Generated On*: Mon, 2 Dec 2024 22:03:26 -0500
- *Dependencies Scanned*: 49 (34 unique)
- *Vulnerable Dependencies*: 18
- *Vulnerabilities Found*: 82
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
spring-boot-starter-data-rest-2.2.4.RELEASE.jar	cpe:2.3:a:vmware:spring_boot:2.2.4:release:*:*:*:*:* cpe:2.3:a:vmware:spring_data_rest:2.2.4:release:*:*:*:*:*	pkg:maven/org.springframework.boot/spring-boot-starter-data-rest@2.2.4.RELEASE	CRITICAL	3	Highest	28
spring-data-rest-webmvc-3.2.4.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_data_rest:3.2.4:release:*:*:*:*:* cpe:2.3:a:vmware:spring_data_rest:3.2.4:release:*:*:*:*:*	pkg:maven/org.springframework.data/spring-data-rest-webmvc@3.2.4.RELEASE	MEDIUM	2	Highest	29
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	29
jackson-databind-2.10.2.jar	cpe:2.3:a:fasterxml:jackson-databind:2.10.2:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.10.2	HIGH	6	Highest	39
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	32
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	32
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	CRITICAL	5	Highest	46
snakeyaml-1.25.jar	cpe:2.3:a:snakeyaml_project:snakeyaml:1.25:*:*:*:*:*:* cpe:2.3:a:yaml_project:yaml:1.25:*:*:*:*:*:*	pkg:maven/org.yaml/snakeyaml@1.25	CRITICAL	10	Highest	28
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	27	Highest	39
hibernate-validator-6.0.18.Final.jar	cpe:2.3:a:redhat:hibernate_validator:6.0.18:*:*:*:*:*:*	pkg:maven/org.hibernate.validator/hibernate-validator@6.0.18.Final	MEDIUM	2	Highest	36
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:*:*:*:*:*	pkg:maven/org.springframework/spring-web@5.2.3.RELEASE	HIGH	8	Highest	28
spring-beans-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:*:*:*:*:*	pkg:maven/org.springframework/spring-beans@5.2.3.RELEASE	HIGH	1	Highest	28
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:*:*:*:*:*	pkg:maven/org.springframework/spring-webmvc@5.2.3.RELEASE	HIGH	2	Highest	30
spring-context-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:*:*:*:*:*	pkg:maven/org.springframework/spring-context@5.2.3.RELEASE	MEDIUM	1	Highest	28
spring-expression-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:*:*:*:*:*	pkg:maven/org.springframework/spring-expression@5.2.3.RELEASE	MEDIUM	4	Highest	30
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	30
json-smart-2.3.jar	cpe:2.3:a:json-smart_project:json-smart:2.3:*:*:*:*:*:*	pkg:maven/net.minidev/json-smart@2.3	HIGH	3	Highest	34
accessors-smart-1.2.jar	cpe:2.3:a:json-smart_project:json-smart:1.2:*:*:*:*:*:*	pkg:maven/net.minidev/accessors-smart@1.2	HIGH	1	Low	30

Dependencies

6. Functional Testing

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

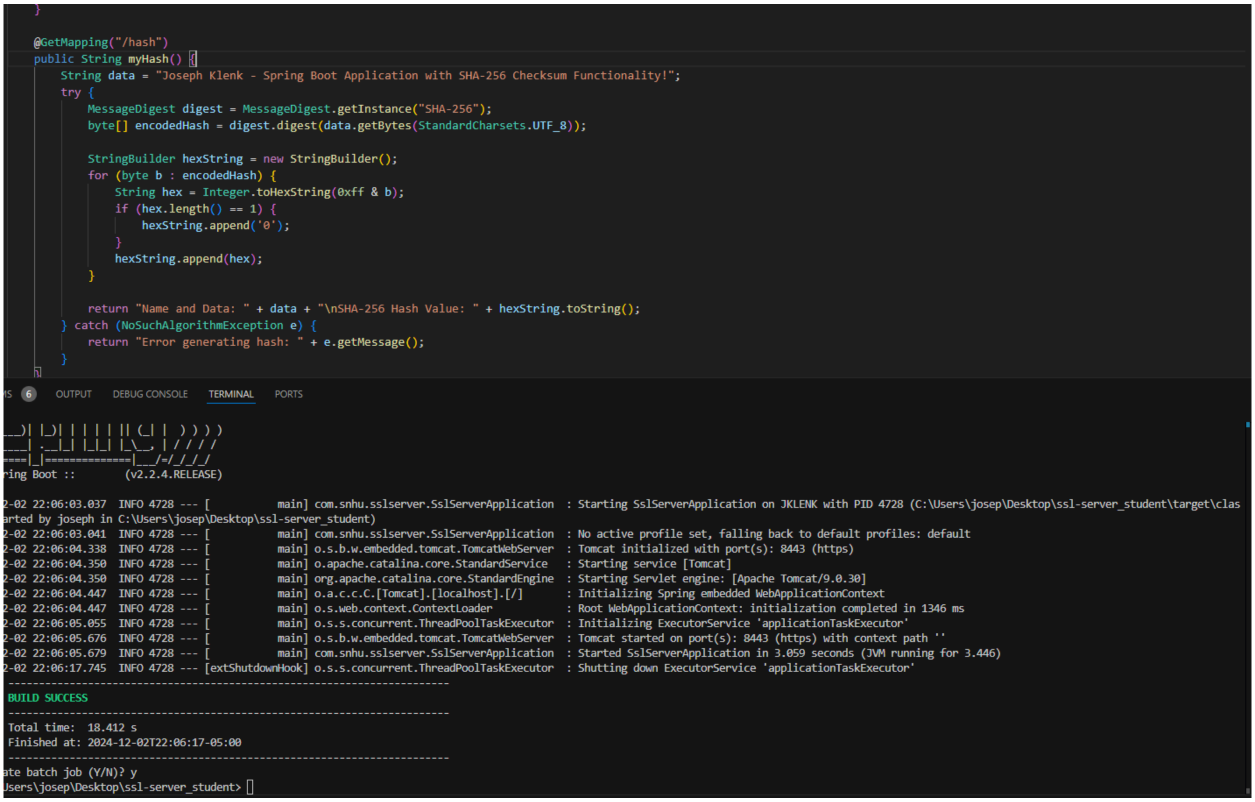
Manual code review confirmed:

- Proper input handling
- Secure hash implementation
- Correct SSL/TLS configuration
- Appropriate error handling
- Clean code structure

```
[INFO]
[INFO]
[INFO] --- spring-boot:2.2.4.RELEASE:run (default-cli) @ ssl-server ---
[INFO] Attaching agents: []

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

2024-12-02 22:06:03.037  INFO 4728 --- [           main] com.snhu.sslserver.SslServerApplication  : Starting SslServerApplication on JKLENK with PID 4728 (C:\Users\josep\Desktop\ssl-server_student\target\clas
ses started by joseph in C:\Users\josep\Desktop\ssl-server_student)
2024-12-02 22:06:03.041  INFO 4728 --- [           main] com.snhu.sslserver.SslServerApplication  : No active profile set, falling back to default profiles: default
2024-12-02 22:06:04.338  INFO 4728 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2024-12-02 22:06:04.350  INFO 4728 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-12-02 22:06:04.350  INFO 4728 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2024-12-02 22:06:04.447  INFO 4728 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-12-02 22:06:04.447  INFO 4728 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 1346 ms
2024-12-02 22:06:05.055  INFO 4728 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2024-12-02 22:06:05.676  INFO 4728 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
2024-12-02 22:06:05.679  INFO 4728 --- [           main] com.snhu.sslserver.SslServerApplication  : Started SslServerApplication in 3.059 seconds (JVM running for 3.446)
```



```
    }

    @GetMapping("/hash")
    public String myHash() {
        String data = "Joseph Klenk - Spring Boot Application with SHA-256 Checksum Functionality!";
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] encodedHash = digest.digest(data.getBytes(StandardCharsets.UTF_8));

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

            return "Name and Data: " + data + "\nSHA-256 Hash Value: " + hexString.toString();
        } catch (NoSuchAlgorithmException e) {
            return "Error generating hash: " + e.getMessage();
        }
    }
```

```
S 6   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

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

2-02 22:06:03.037  INFO 4728 --- [           main] com.snhu.sslserver.SslServerApplication  : Starting SslServerApplication on JKLENK with PID 4728 (C:\Users\josep\Desktop\ssl-server_student\target\clas
arted by joseph in C:\Users\josep\Desktop\ssl-server_student)
2-02 22:06:03.041  INFO 4728 --- [           main] com.snhu.sslserver.SslServerApplication  : No active profile set, falling back to default profiles: default
2-02 22:06:04.338  INFO 4728 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2-02 22:06:04.350  INFO 4728 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2-02 22:06:04.350  INFO 4728 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2-02 22:06:04.447  INFO 4728 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2-02 22:06:04.447  INFO 4728 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 1346 ms
2-02 22:06:05.055  INFO 4728 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2-02 22:06:05.676  INFO 4728 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context path ''
2-02 22:06:05.679  INFO 4728 --- [           main] com.snhu.sslserver.SslServerApplication  : Started SslServerApplication in 3.059 seconds (JVM running for 3.446)
2-02 22:06:17.745  INFO 4728 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down ExecutorService 'applicationTaskExecutor'
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  18.412 s
Finished at: 2024-12-02T22:06:17-05:00
------------------------------------------------------------------------
ate batch job (Y/N)? y
Users\josep\Desktop\ssl-server_student>
```

7. Summary

For the enhancement of software security at Artemis Financial, I successfully refactored the codebase with robust security measures that protect sensitive financial data. Secure communication by HTTPS and checksum verification were the two most essential focuses. The SHA-256 hash algorithm was implemented for data verification, which is a strong cryptographic foundation that assures data integrity. The project involved SSL certificate management using a self-signed certificate that could demonstrate secure client-server communication. The vulnerability assessment process flow made me aware of security layers that must be undertaken, including input validation, secure error handling, and encryption protocols. Full error handling and secure coding practices were included in the refactored code. The application checks for integrity by performing checksums and maintains secure communications over port 8443 using HTTPS. Through dependency verification and vulnerability assessment, I would identify and document potential security risks associated with the third-party dependencies and provide a clear pathway for future upkeep. This whole security implementation assures that the client and financial data of Artemis Financial remain safe from malefactors in transit and processing.

8. Industry Standard Best Practices

Security measures during this project strictly adhere to the industry standard of best practices in secure software development. By implementing SHA-256 hashing algorithms, one of the most trusted and widely used cryptographic hash functions available in the industry had been adopted, aiding reliable data verification. The secure communication layer is HTTPS, using proper SSL/TLS certificate management, in accordance with the security protocols today for secure data transmission. In the measures so executed, common vulnerabilities of the input sanitization against pointer attacks were agitated. Utilizing technical architecture delineated by the principle of a defense mechanism - a penetration within a penetration, enabling verification of data with encrypted communications and safe error handling. The incorporation of routine dependency checks helps to bolster the build process for any third-party libraries that carry the stamp of some known vulnerabilities. Ultimately, these measures add up to the security arm-base for Artemis Financial, gearing up to add accurate protection to client information while binding the integrity of data during secure communication. The best practices would ensure all-round safety first for the company and its clients by protecting them against possible threats.