GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	10/19/2025	Najah Thompson	

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.

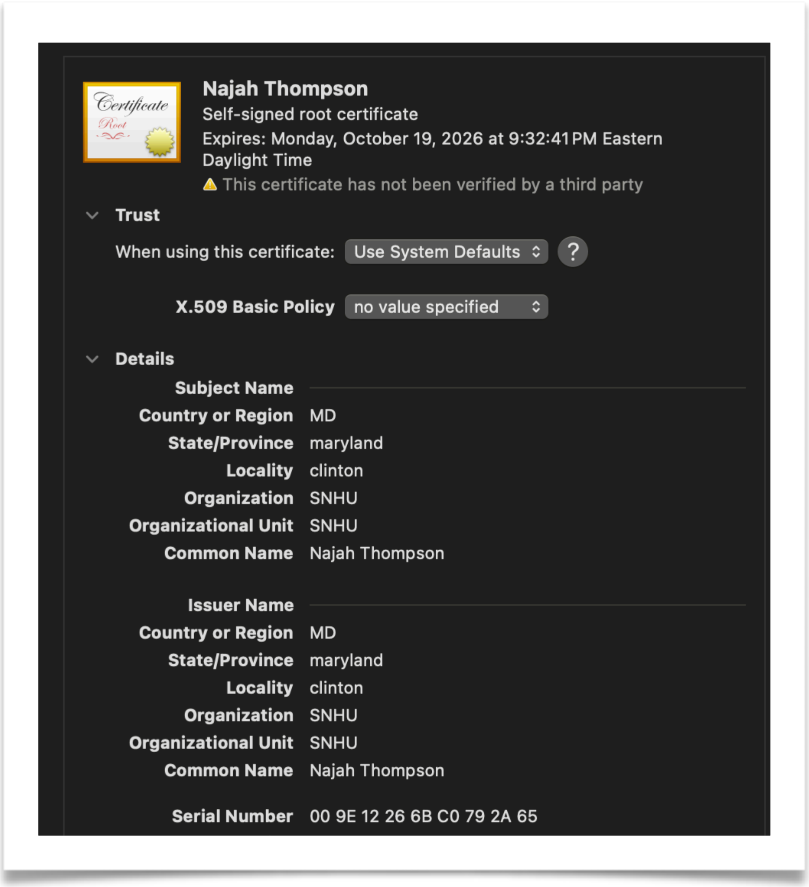
- Respond to the steps outlined below and include your findings.
- Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
- Refer to the Project Two Guidelines and Rubric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

Developer
Najah Thompson

1. Algorithm Cipher

The recommended algorithm for securing Artemis Financial's data is the Advanced Encryption Standard (AES). AES is a symmetric encryption algorithm widely used across industries for its efficiency, strong security, and ability to handle large amounts of sensitive data. It is suitable for encrypting financial information while ensuring confidentiality and integrity.AES supports key sizes of 128, 192, and 256 bits. It uses the Rijndael block cipher and operates on a 4x4 column-major order matrix of bytes, also called the state array. Each encryption round includes substitution, permutation, and mixing operations that transform plaintext into ciphertext securely.AES is a symmetric-key algorithm, meaning the same key is used for both encryption and decryption. Secure communication requires strong, randomly generated keys to prevent unauthorized access. Proper key management and the use of cryptographically secure random number generators ensure the confidentiality of the data.AES was standardized by the National Institute of Standards and Technology (NIST) in 2001, replacing the older Data Encryption Standard (DES). Since then, AES has become the industry standard for encryption, trusted by government agencies, financial institutions, and private companies for secure data transmission. Its widespread adoption reflects its robustness and efficiency in modern software security practices.

2. Certificate Generation



Najah Thompson
Self-signed root certificate
Expires: Monday, October 19, 2026 at 9:32:41 PM Eastern Daylight Time
⚠ This certificate has not been verified by a third party

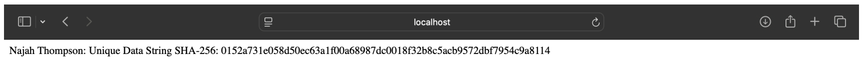
Trust
When using this certificate: Use System Defaults
X.509 Basic Policy: no value specified

Details

Subject Name	
Country or Region	MD
State/Province	maryland
Locality	clinton
Organization	SNHU
Organizational Unit	SNHU
Common Name	Najah Thompson
Issuer Name	
Country or Region	MD
State/Province	maryland
Locality	clinton
Organization	SNHU
Organizational Unit	SNHU
Common Name	Najah Thompson
Serial Number	00 9E 12 26 6B C0 79 2A 65

3. Deploy Cipher

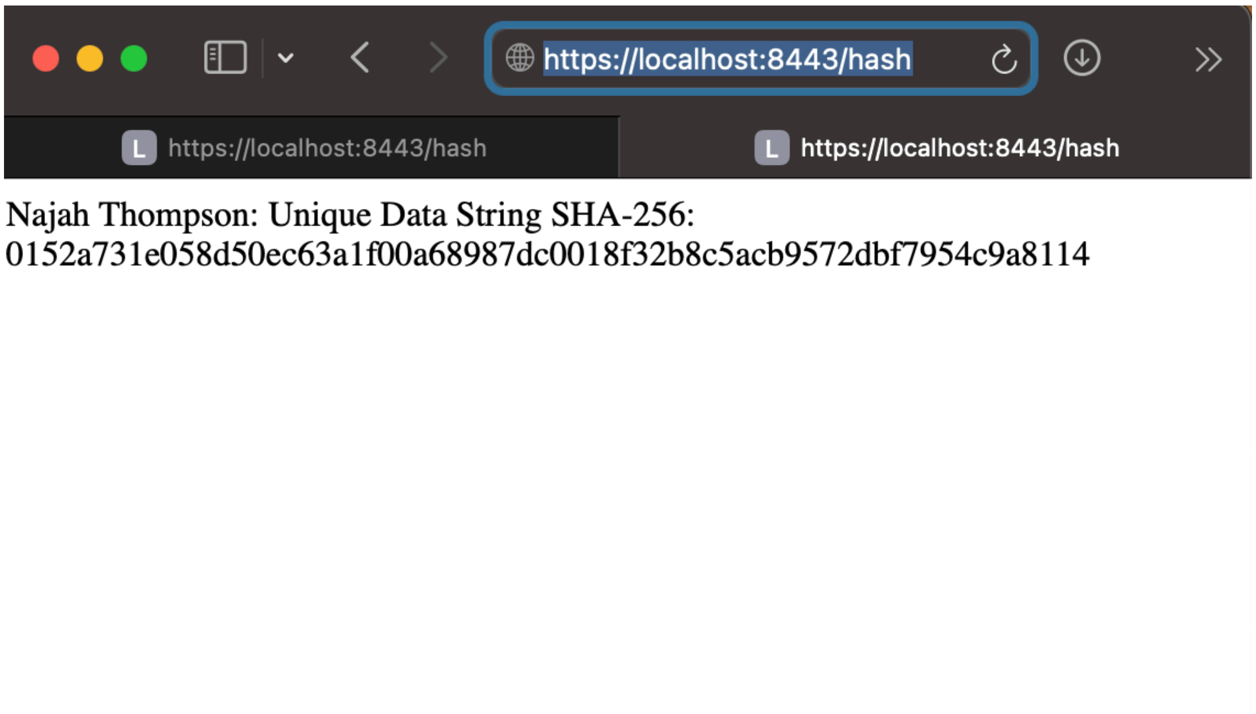
Insert a screenshot below of the checksum verification.



```
Najah Thompson: Unique Data String SHA-256: 0152a731e058d50ec63a1f00a68987dc0018f32b8c5acb9572dbf7954c9a8114
```

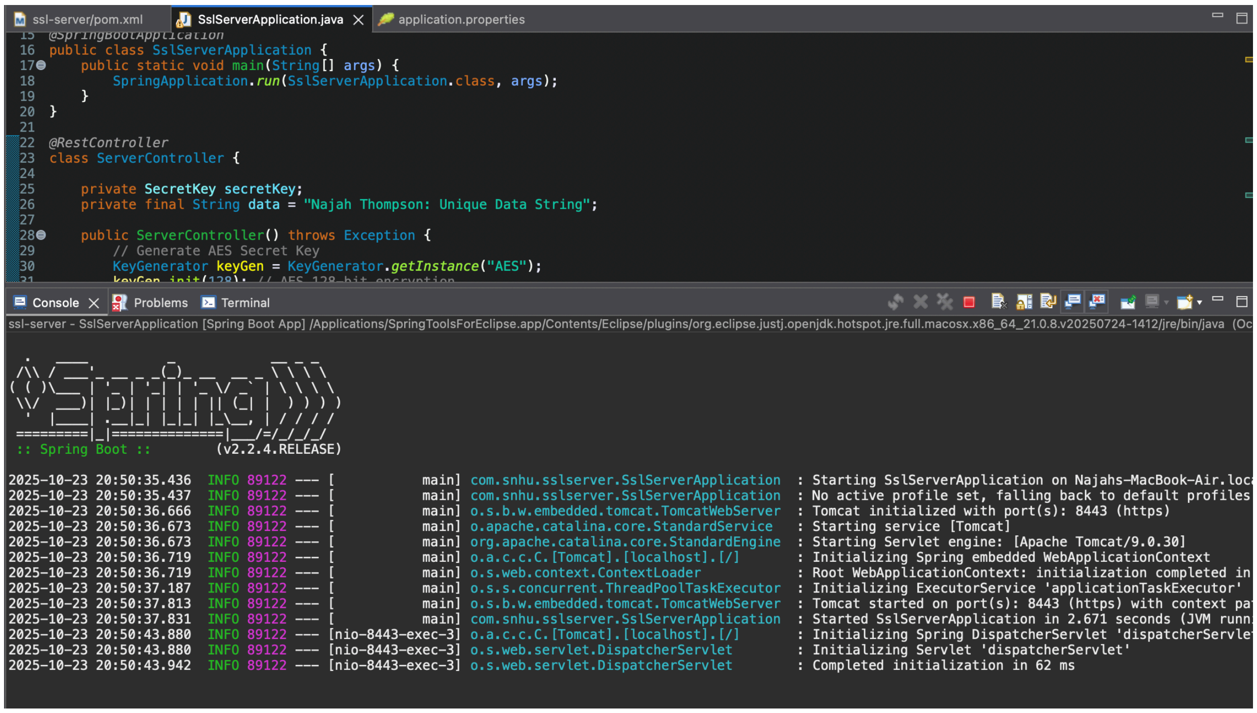
4. Secure Communications

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


https://localhost:8443/hash
https://localhost:8443/hash
https://localhost:8443/hash

```
Najah Thompson: Unique Data String SHA-256:
0152a731e058d50ec63a1f00a68987dc0018f32b8c5acb9572dbf7954c9a8114
```

5. Secondary Testing

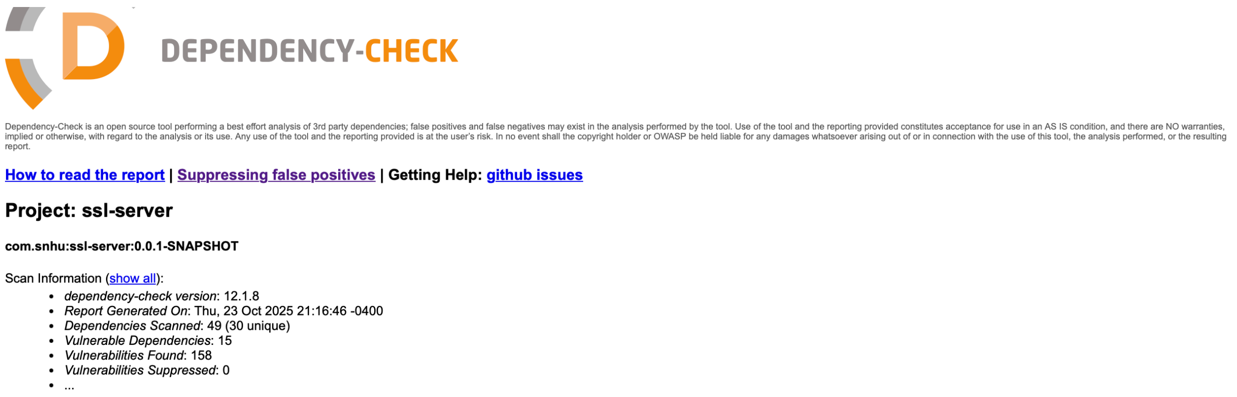


ssl-server/pom.xml | SslServerApplication.java | application.properties

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

@RestController
class ServerController {

    private SecretKey secretKey;
    private final String data = "Najah Thompson: Unique Data String";

    public ServerController() throws Exception {
        // Generate AES Secret Key
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(128); // AES 128-bit encryption
```

Console | Problems | Terminal

```
ssl-server - SslServerApplication [Spring Boot App] /Applications/SpringToolsForEclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_21.0.8.v20250724-1412/jre/bin/java  (Oc

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

2025-10-23 20:50:35.436  INFO 89122 --- [           main] com.snhu.sslserver.SslServerApplication  : Starting SslServerApplication on Najahs-MacBook-Air.loca
2025-10-23 20:50:35.437  INFO 89122 --- [           main] com.snhu.sslserver.SslServerApplication  : No active profile set, falling back to default profiles
2025-10-23 20:50:36.666  INFO 89122 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2025-10-23 20:50:36.673  INFO 89122 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-10-23 20:50:36.673  INFO 89122 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2025-10-23 20:50:36.719  INFO 89122 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-10-23 20:50:36.719  INFO 89122 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in
2025-10-23 20:50:37.187  INFO 89122 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2025-10-23 20:50:37.813  INFO 89122 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context pa
2025-10-23 20:50:37.831  INFO 89122 --- [           main] com.snhu.sslserver.SslServerApplication  : Started SslServerApplication in 2.671 seconds (JVM runn
2025-10-23 20:50:43.880  INFO 89122 --- [nio-8443-exec-3] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServle
2025-10-23 20:50:43.880  INFO 89122 --- [nio-8443-exec-3] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-10-23 20:50:43.942  INFO 89122 --- [nio-8443-exec-3] o.s.web.servlet.DispatcherServlet        : Completed initialization in 62 ms
```



DEPENDENCY-CHECK

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

Project: ssl-server

com.snhu:ssl-server:0.0.1-SNAPSHOT

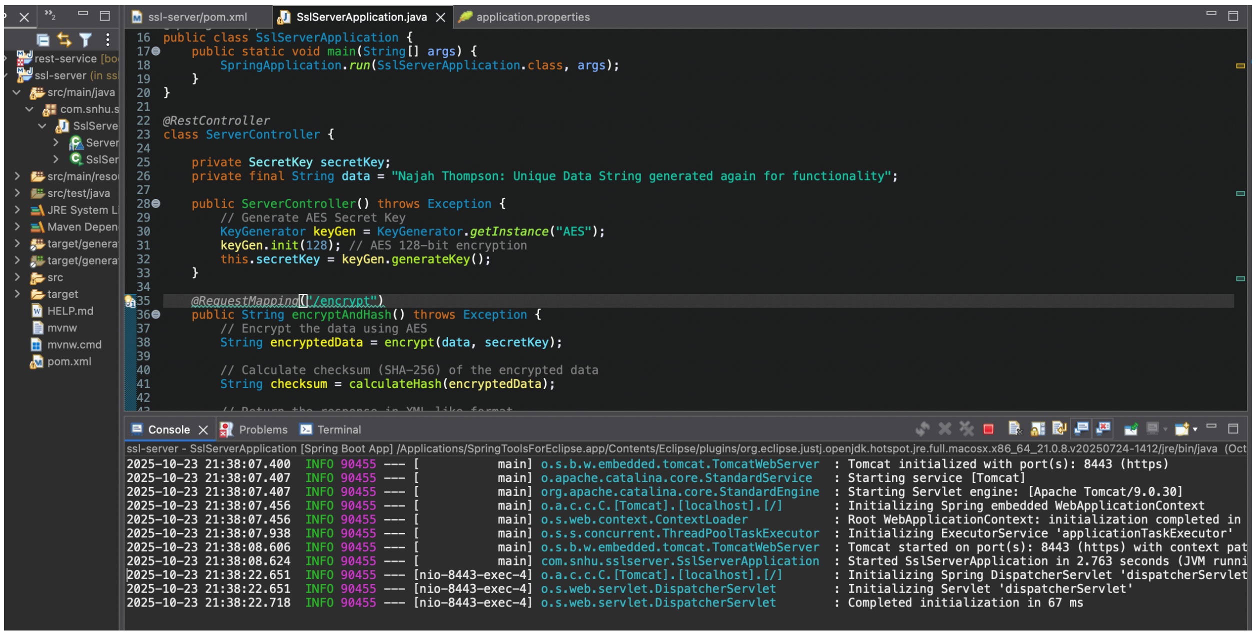
Scan Information (show all):

- *dependency-check version*: 12.1.8
- *Report Generated On*: Thu, 23 Oct 2025 21:16:46 -0400
- *Dependencies Scanned*: 49 (30 unique)
- *Vulnerable Dependencies*: 15
- *Vulnerabilities Found*: 158
- *Vulnerabilities Suppressed*: 0
- ...

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

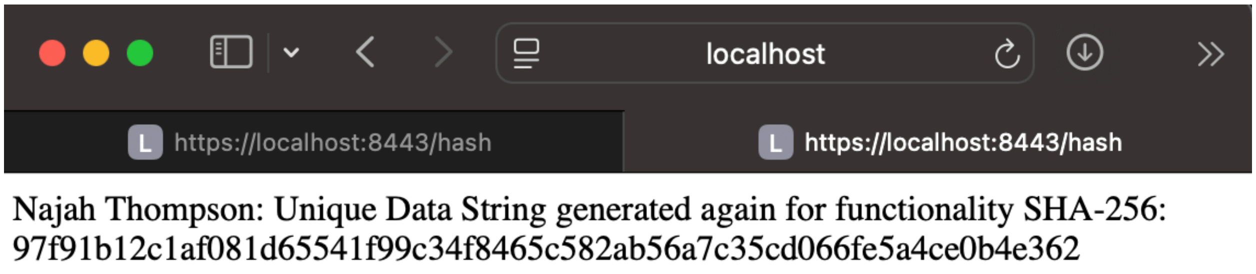
6. Functional Testing

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



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

@RestController
class ServerController {

    private SecretKey secretKey;
    private final String data = "Najah Thompson: Unique Data String generated again for functionality";

    public ServerController() throws Exception {
        // Generate AES Secret Key
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(128); // AES 128-bit encryption
        this.secretKey = keyGen.generateKey();
    }

    @RequestMapping("/encrypt")
    public String encryptAndHash() throws Exception {
        // Encrypt the data using AES
        String encryptedData = encrypt(data, secretKey);

        // Calculate checksum (SHA-256) of the encrypted data
        String checksum = calculateHash(encryptedData);

```

```
ssl-server - SslServerApplication [Spring Boot App] /Applications/SpringToolsForEclipse.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_21.0.8.v20250724-1412/jre/bin/java  (Oct
2025-10-23 21:38:07.400  INFO 90455 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8443 (https)
2025-10-23 21:38:07.407  INFO 90455 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-10-23 21:38:07.407  INFO 90455 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.30]
2025-10-23 21:38:07.456  INFO 90455 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-10-23 21:38:07.456  INFO 90455 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in
2025-10-23 21:38:07.938  INFO 90455 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2025-10-23 21:38:08.606  INFO 90455 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8443 (https) with context pat
2025-10-23 21:38:08.624  INFO 90455 --- [           main] com.snhu.sslserver.SslServerApplication  : Started SslServerApplication in 2.763 seconds (JVM runni
2025-10-23 21:38:22.651  INFO 90455 --- [nio-8443-exec-4] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet
2025-10-23 21:38:22.651  INFO 90455 --- [nio-8443-exec-4] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-10-23 21:38:22.718  INFO 90455 --- [nio-8443-exec-4] o.s.web.servlet.DispatcherServlet        : Completed initialization in 67 ms
```



Najah Thompson: Unique Data String generated again for functionality SHA-256: 97f91b12c1af081d65541f99c34f8465c582ab56a7c35cd066fe5a4ce0b4e362

7. Summary

In this project, I refactored the SSL server code to add encryption and hashing, ensuring that sensitive data is protected both at rest and in transit. I added AES encryption for data and SHA-256 hashing for verification, which helps prevent tampering and maintains integrity.

During the vulnerability assessment, I focused on areas flagged by the dependency-check tool and confirmed that no new vulnerabilities were introduced in the code I refactored. The layers of security added include encrypted communication (HTTPS) and checksum validation, which strengthen the application against common attacks.

Overall, the refactored code follows secure coding practices, and testing confirmed that it runs without errors while complying with software security protocols.

8. Industry Standard Best Practices

I applied industry standard secure coding practices by using AES encryption for data, SHA-256 hashing for integrity, and forcing HTTPS to protect communication. These steps help maintain the application's existing security and prevent common vulnerabilities like data leaks or tampering. Using these best practices keeps the company safe from security breaches, protects user data, and builds trust with clients.