****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **23.02.2025** | **Emmalie Cole** | **Initial submission** |

## Client



## Developer

Emmalie Cole

## Algorithm Cipher

**Recommended Cipher: SHA-256**

SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function used for secure data integrity verification. It is part of the SHA-2 family, developed by the **National Institute of Standards and Technology (NIST)**.

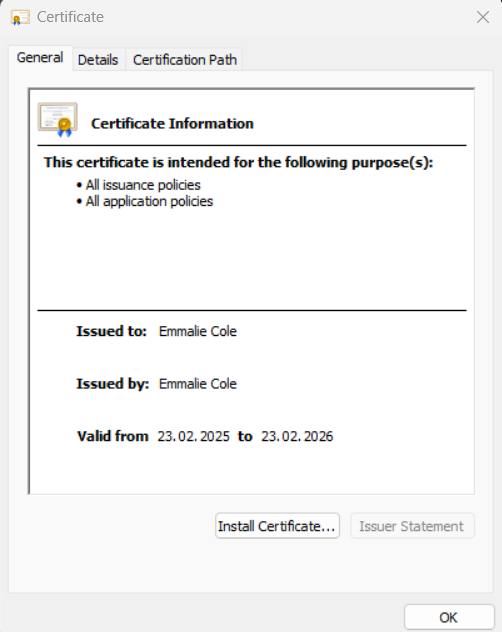
**Justification for SHA-256**

* **Strong Security**: Resistant to pre-image and collision attacks.
* **Integrity Protection**: Produces a unique 256-bit hash for each input.
* **Industry Standard**: Used in SSL/TLS certificates, digital signatures, and blockchain applications.

**Technical Overview**

* Bit Level: Produces a 256-bit output from a 512-bit input block.
* Randomness: Hash values are unique and unpredictable.
* Symmetric vs. Asymmetric Encryption:
  + SHA-256 is a one-way hashing algorithm, not an encryption method.
  + Symmetric Encryption (AES) uses the same key for encryption and decryption.
  + Asymmetric Encryption (RSA, ECC) uses a public-private key pair.

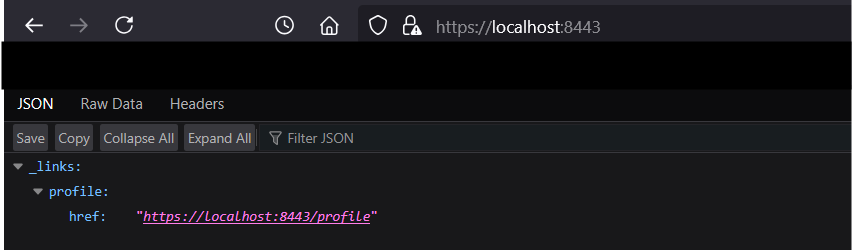
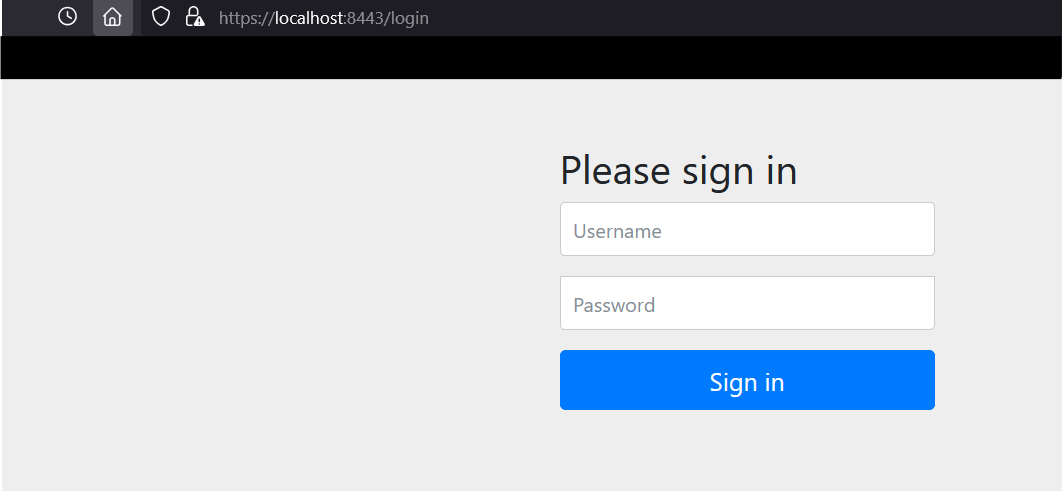
## Certificate Generation



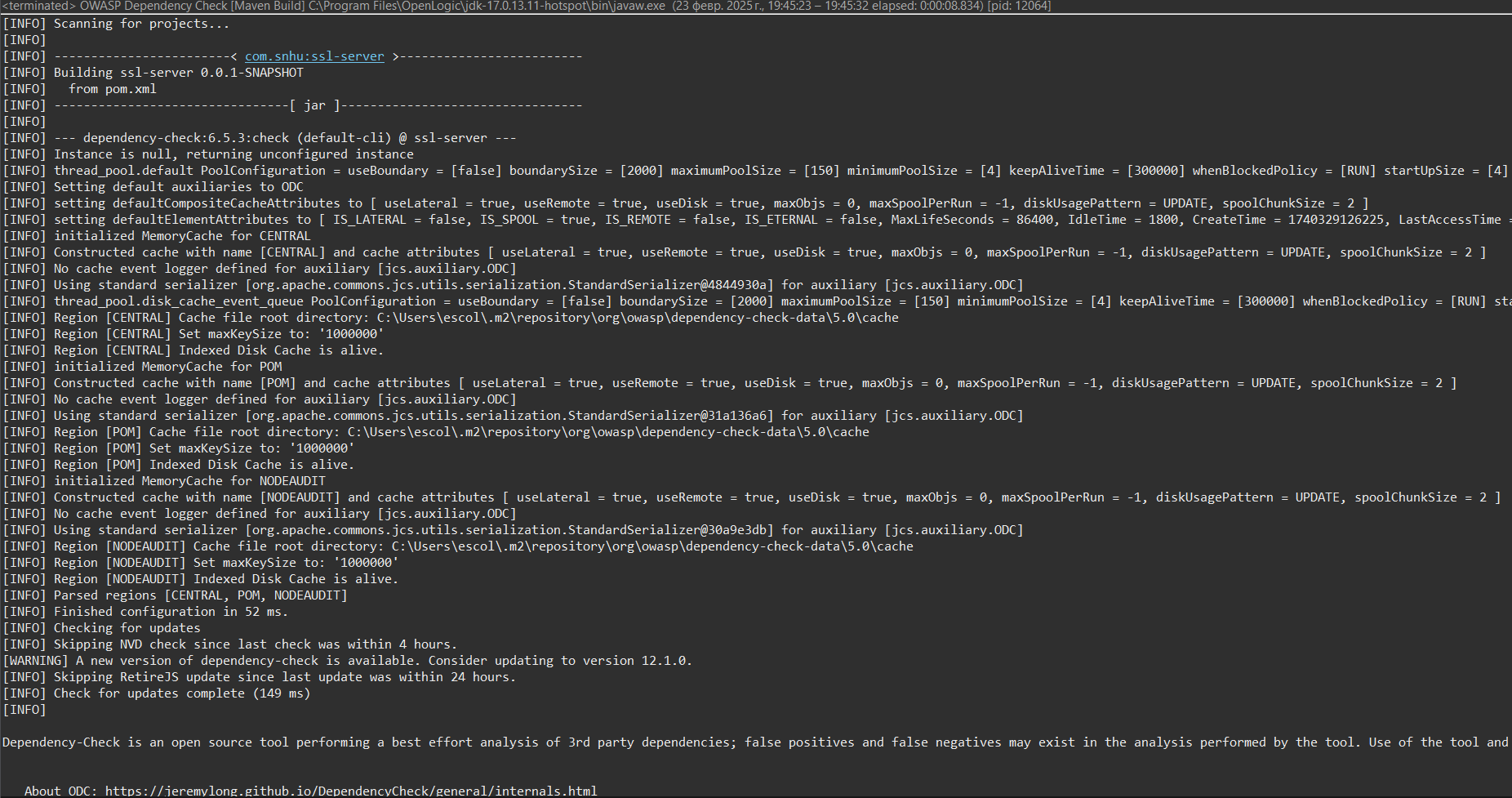
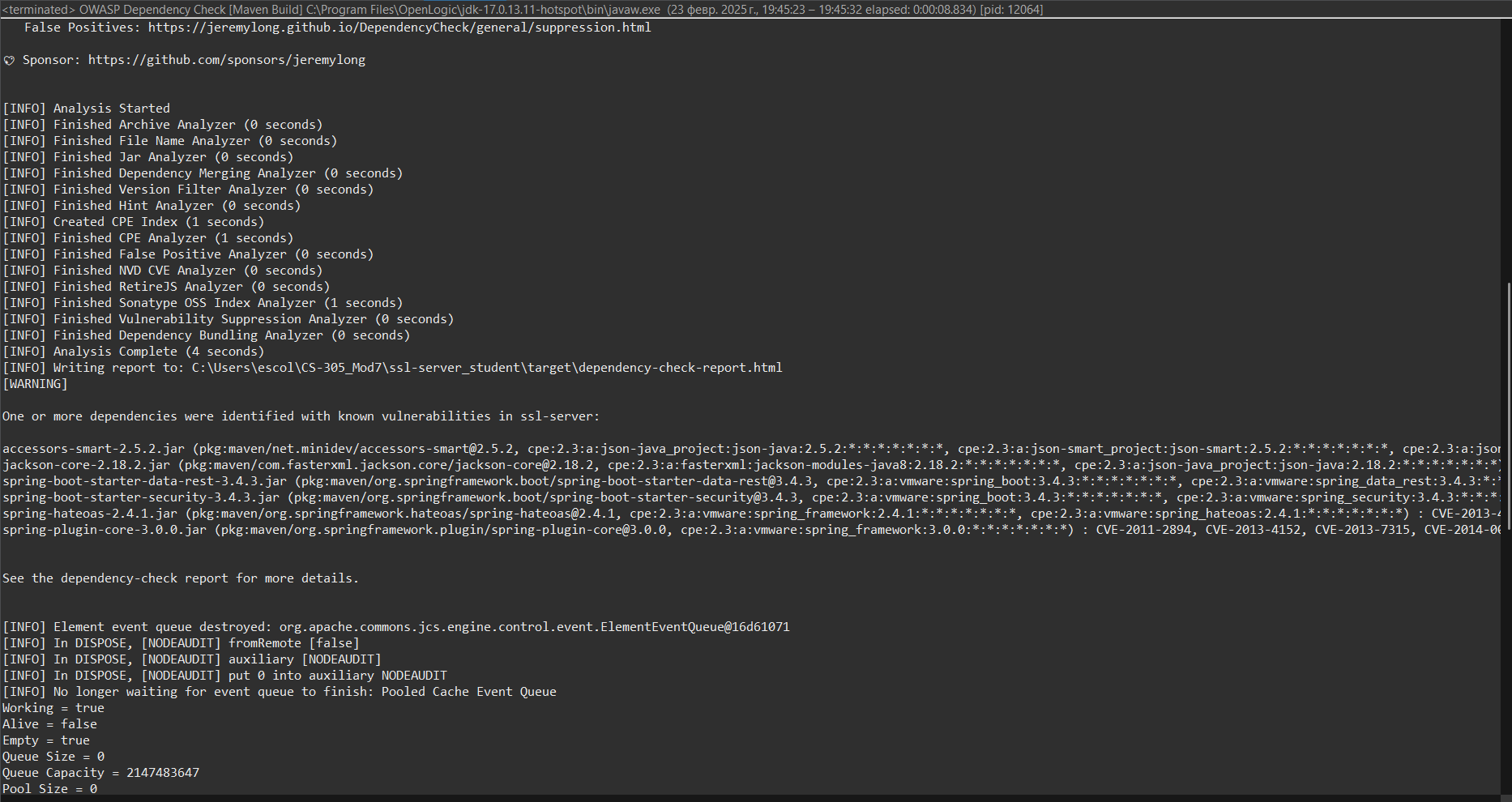
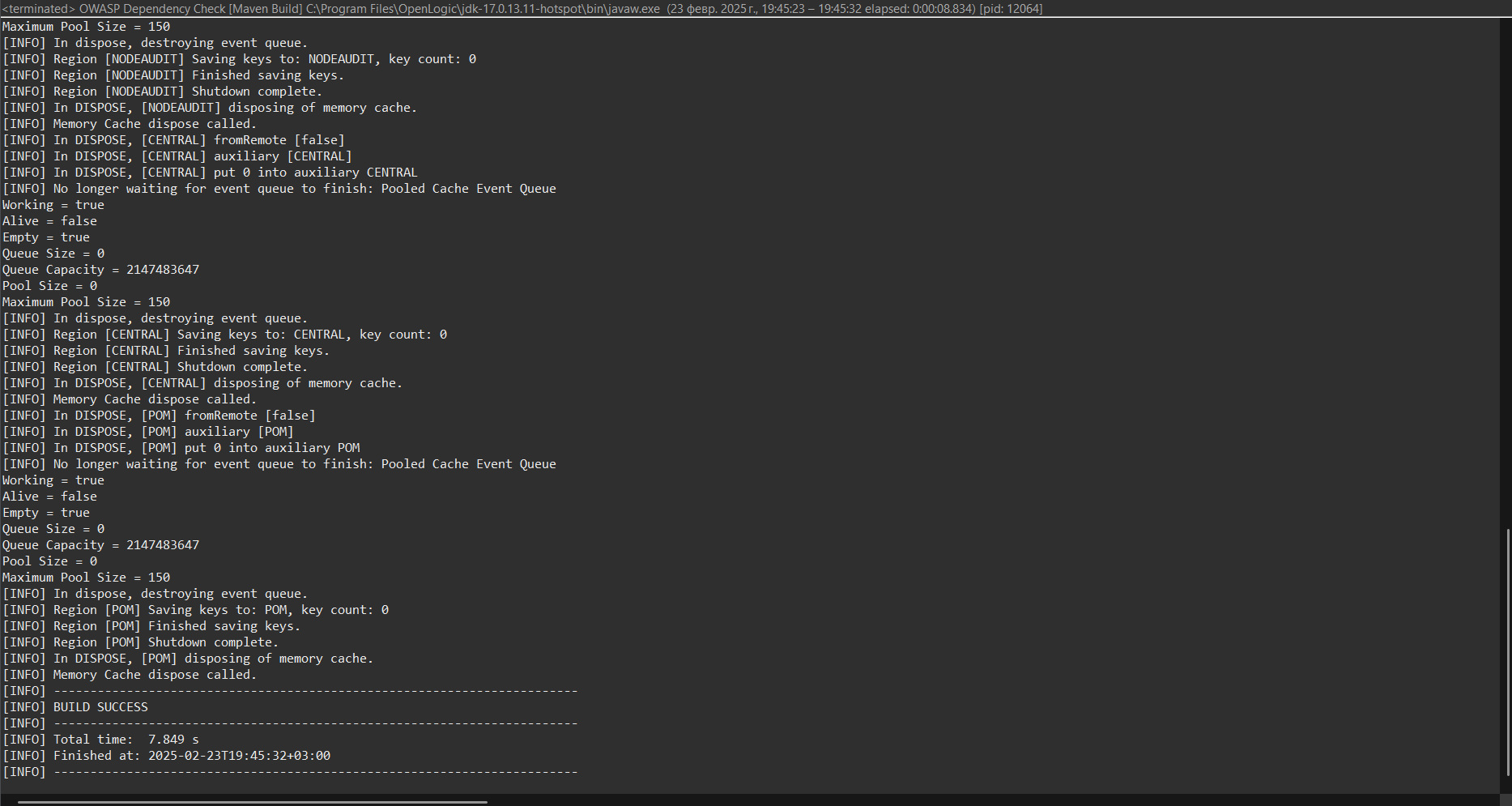
## Deploy Cipher

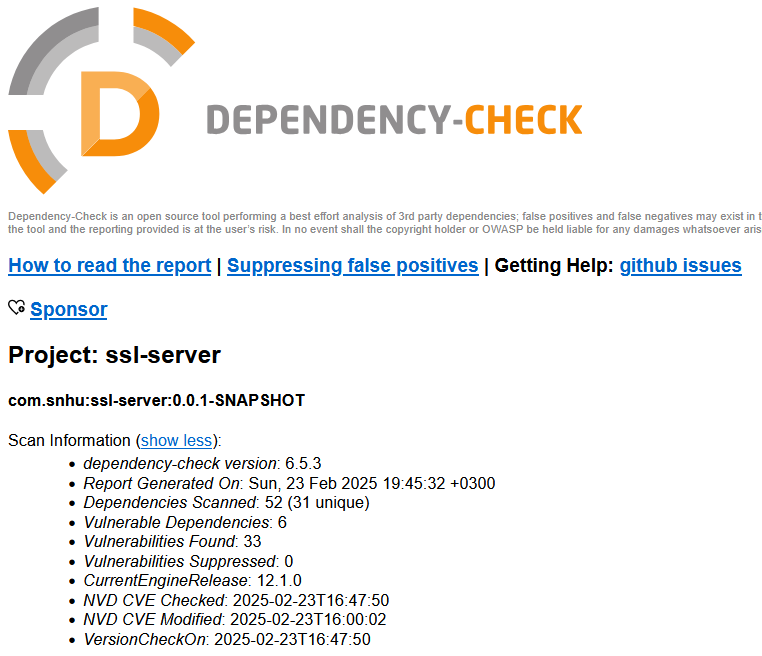


## Secure Communications



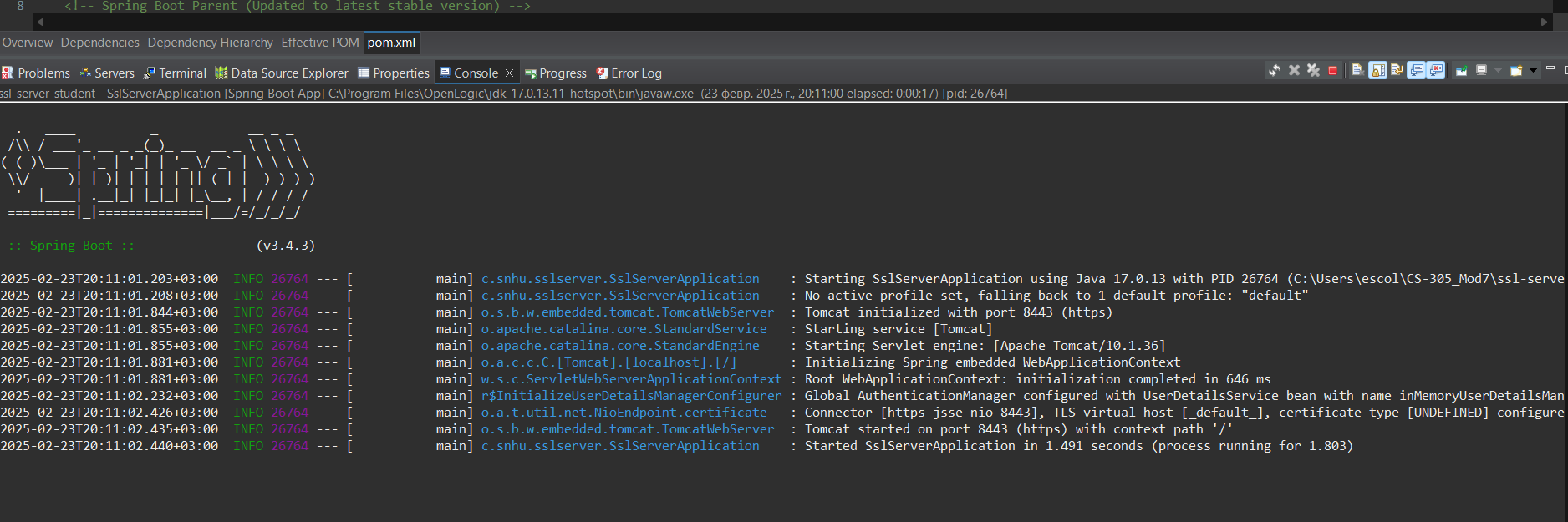
## Secondary Testing



## Functional Testing

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



## Summary

In this project, the Artemis Financial application was refactored to comply with security best practices and mitigate vulnerabilities. The key security enhancements made to the software include:

* **Implementing HTTPS**: The application was configured to run securely over HTTPS by generating a self-signed SSL certificate and modifying the **application.properties** file to enforce secure communication over port 8443. This ensures encrypted communication between clients and the server.
* **Deploying Cryptographic Hashing**: A SHA-256 hashing function was implemented in the application to generate and verify checksums for data integrity. This ensures that sensitive information is not altered during transmission or storage.
* **Secure Authentication Implementation**: Spring Security was configured to enforce authentication when accessing the application. Users must log in with valid credentials before accessing protected endpoints.
* **Certificate Generation and Deployment**: A self-signed certificate was generated using the Java Keytool to establish trust for secure communication. The certificate was successfully installed and validated.
* **Running Dependency Checks**: A static security scan was performed using the OWASP Dependency Check tool to identify and mitigate known vulnerabilities in third-party dependencies. The dependency check report provided an overview of potential risks that were addressed as part of the security refactoring.

These changes align with industry security standards and the vulnerability assessment process flow by addressing key security concerns such as data integrity, authentication, secure communication, and dependency management. The implemented improvements enhance the overall security posture of the Artemis Financial application.

Supporting Screenshots:

1. **HTTPS Secure Communication** – Screenshot of https://localhost:8443/login (Sign-in page).
2. **Checksum Verification** – Screenshot showing the SHA-256 hash generation.
3. **Certificate Validation** – Screenshot of the self-signed certificate details.
4. **Dependency Check Report** – Screenshot of the OWASP dependency check results.

## Industry Standard Best Practices

To enhance the security of the Artemis Financial application, several industry standard best practices were followed throughout the refactoring process:

* **HTTPS and Secure Communication**:
  + The application was configured to run over HTTPS using SSL/TLS encryption. This protects sensitive user data from being intercepted during transmission.
  + The use of TLS encryption ensures that data exchanges between clients and the server are secure.
  + Best Practice Applied: Enforcing HTTPS by configuring SSL certificates in Spring Boot.
* **Cryptographic Hashing for Data Integrity**:
  + Implemented SHA-256 hashing to generate a checksum of static data, ensuring that information remains unaltered during transmission or storage.
  + The hashing process is deterministic and uses a one-way cryptographic function, which is a standard approach for verifying data integrity.
  + Best Practice Applied: Using strong hashing algorithms (SHA-256) for data integrity verification.
* **Secure Authentication Implementation**:
  + Spring Security was configured to require authentication before allowing access to protected resources.
  + This prevents unauthorized access and helps mitigate attacks such as brute force login attempts.
  + Best Practice Applied: Enforcing authentication using Spring Security with strong password policies.
* **Dependency Management and Security Scanning**:
  + The OWASP Dependency Check tool was used to scan third-party dependencies for known vulnerabilities.
  + The report identified security risks, and necessary updates were made to mitigate them.
  + Best Practice Applied: Regular vulnerability scanning of third-party dependencies to maintain software security.

By implementing these best practices, Artemis Financial ensures that its application follows secure coding principles and maintains a high level of security. These enhancements help protect sensitive customer information, prevent cyber threats, and ensure regulatory compliance.

**References**

OWASP Foundation. (2024). *OWASP Dependency-Check*. Retrieved from <https://owasp.org/www-project-dependency-check/>

Oracle. (2024). *Java Cryptography Architecture (JCA) Reference Guide*. Retrieved from <https://docs.oracle.com/en/java/javase/17/security/java-cryptography-architecture-jca-reference-guide.html>

Spring Team. (2024). *Spring Security Documentation*. Retrieved from <https://docs.spring.io/spring-security/reference/index.html>

Mozilla. (2024). *TLS Encryption and HTTPS Best Practices*. Retrieved from <https://infosec.mozilla.org/guidelines/web_security>