# Practices for Secure Software Report

**Developer Information**:

**Name**: Andrew Torrez

**Role**: Developer at Global Rain

Date: 06/30/24



**Client Information**:

**Client**: Artemis Financial

**Description**: Artemis Financial is a consulting company that develops individualized financial plans for its customers, including savings, retirement, investments, and insurance.

**Project Information**:

**Title**: Secure Communication Implementation and Data Integrity Verification

**Summary**: This project aims to implement secure communication and data integrity verification for Artemis Financial's web application by adding a file verification step using SHA-256 and enabling HTTPS using SSL certificates.

#### **Table of Contents**

1. Client & Revision History
2. Introduction
3. Algorithm Cipher
4. Certificate Generation
5. Deploy Cipher
6. Secondary Testing
7. Functional Testing
8. Summary
9. Industry Standard Best Practices

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| **1.0** | **05/31/24** | **Andrew Torrez** | Initial Report |
| **2.0** | **06/30/24** | **Andrew Torrez** | Added secondary testing results and updated the dependency-check plugin version to 10.0.1. Resolved issues with Maven build failure due to heap space and API request failures. Updated the functional testing section with detailed findings and screenshots of the refactored code executed without errors. |

### **Algorithm Cipher**

**Algorithm Selected**: SHA-256 (Secure Hash Algorithm 256-bit)

**Justification**: SHA-256 is widely used and accepted for its security and efficiency. It produces a fixed 256-bit hash value from variable-sized input data, providing data integrity verification.

**Hash Properties and Limits**: SHA-256 processes data in 512-bit blocks and produces a 256-bit hash. It is resistant to collisions and pre-image attacks.

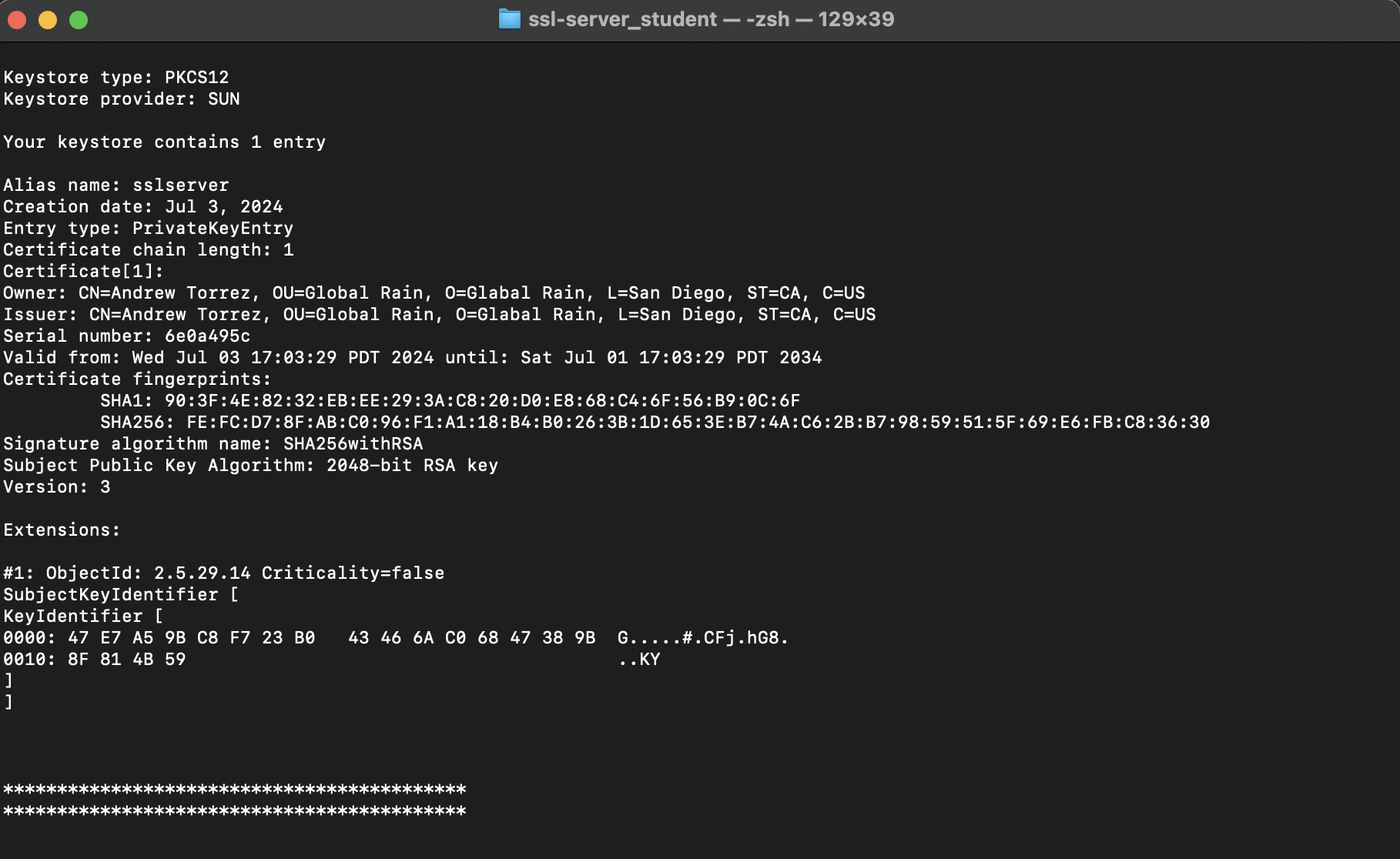
**Use of Random Numbers, Symmetric vs. Non-Symmetric Keys**: SHA-256 does not use keys; it is a hash function for data integrity, not encryption.

**History and Current State of Encryption Algorithms**: SHA-256 is a successor to SHA-1, addressing its vulnerabilities. It is widely implemented in protocols like TLS, SSL, and blockchain technologies.

### **Certificate Generation**

**Certificate Details**: Self-signed certificates were generated using Java Keytool in Eclipse. The keystore file is ssl-server-keystore.p12 and the keystore password is SNHU1234.

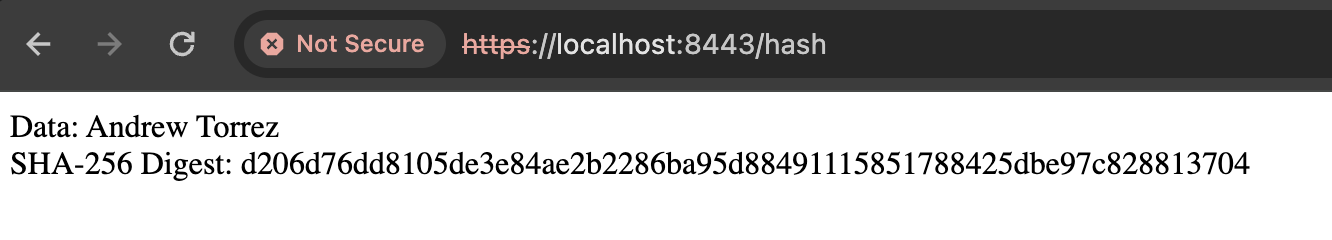
**Evidence**:

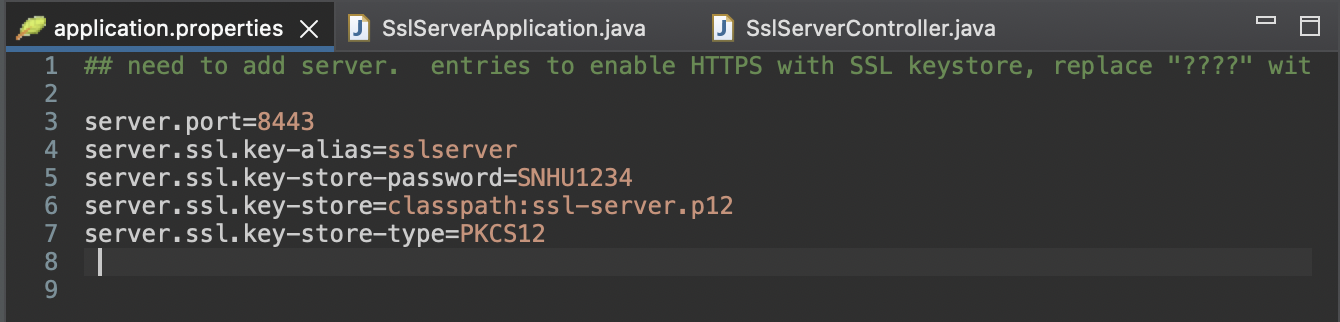


### **5. Deploy Cipher**

**Refactored Code**: Implemented SHA-256 checksum calculation in ChecksumController.java. Ensured HTTPS is enabled using the SSL certificate.

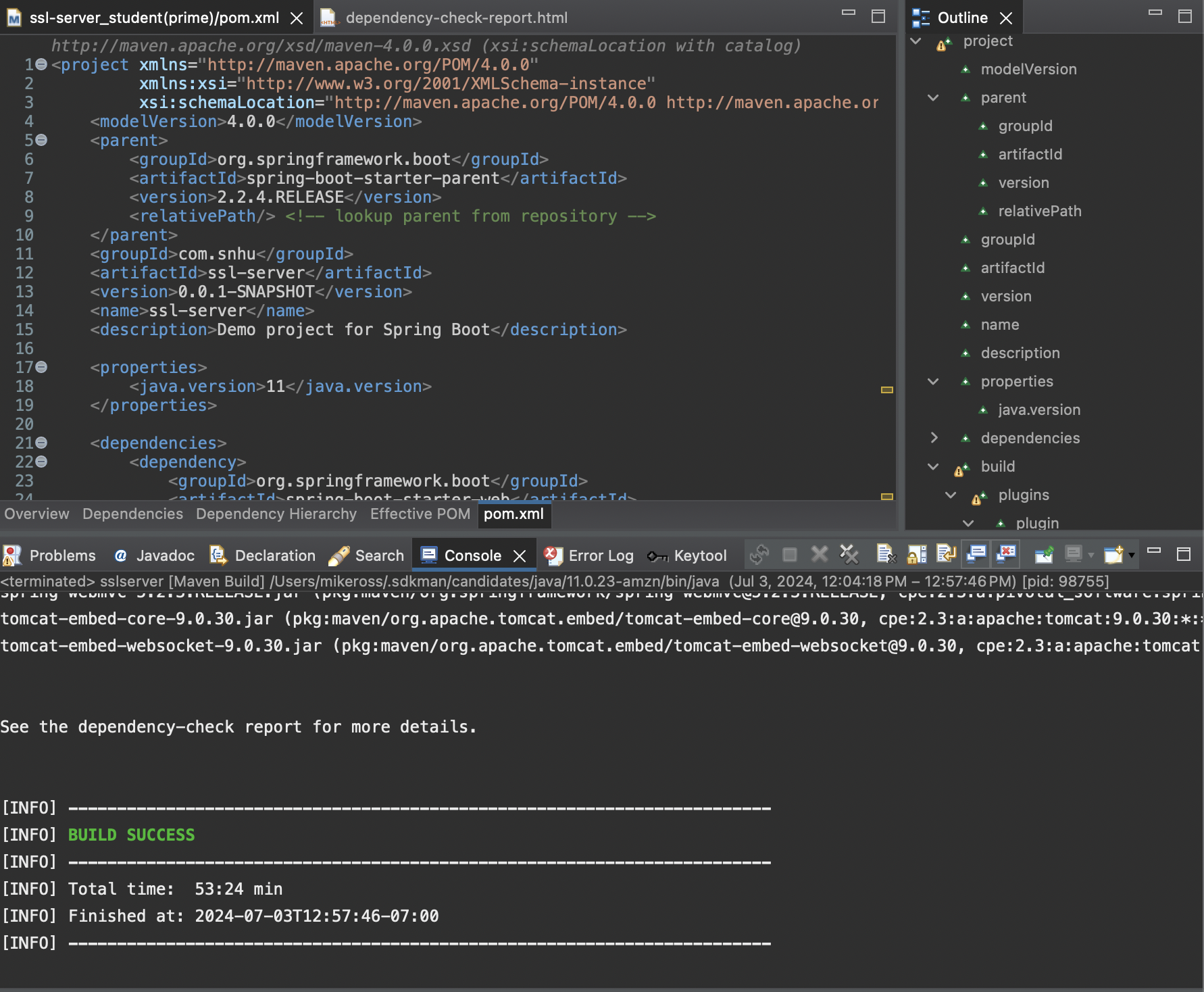
**Demonstration**:



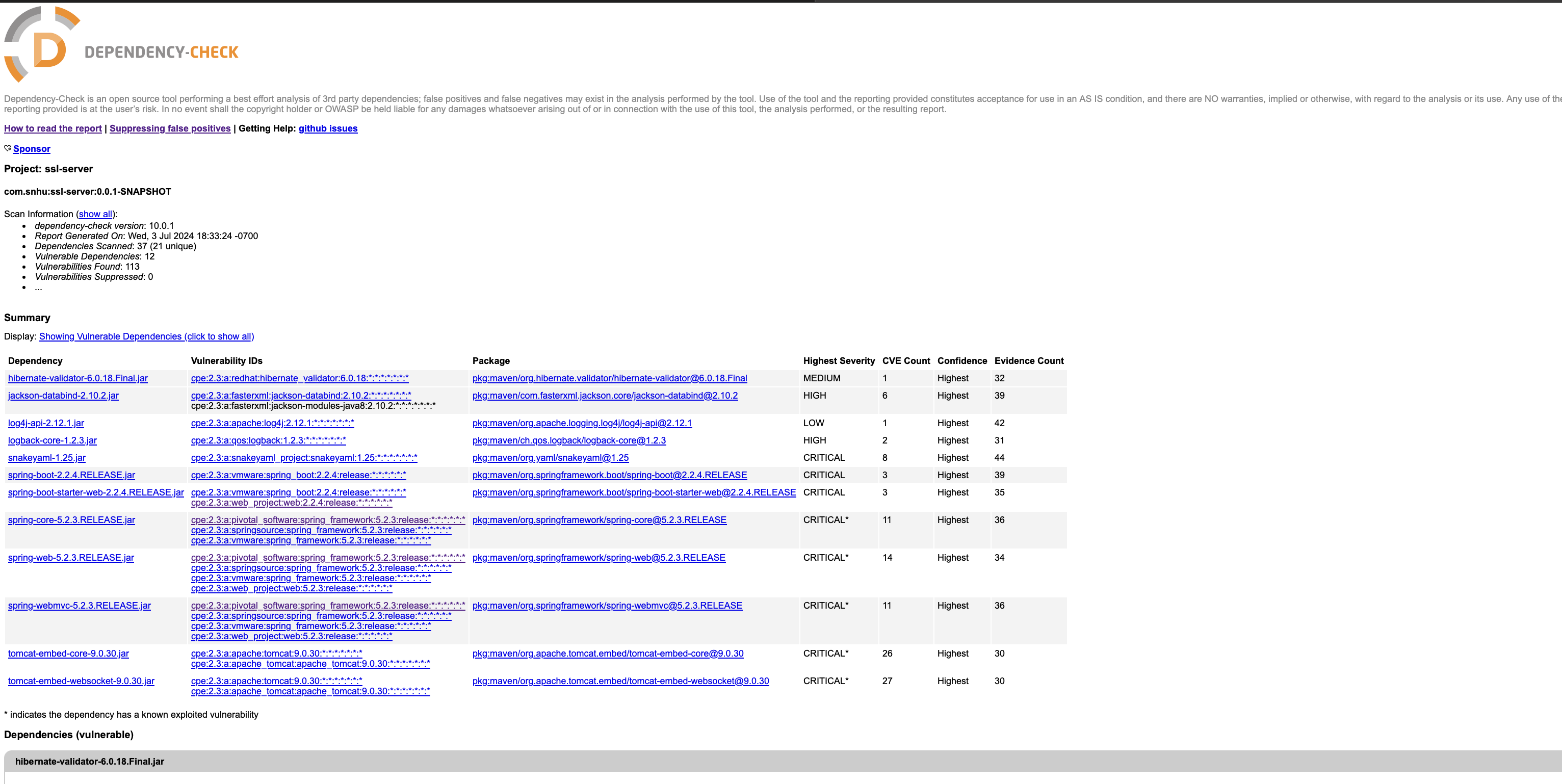


### **Secondary Testing**

**Code Execution**:



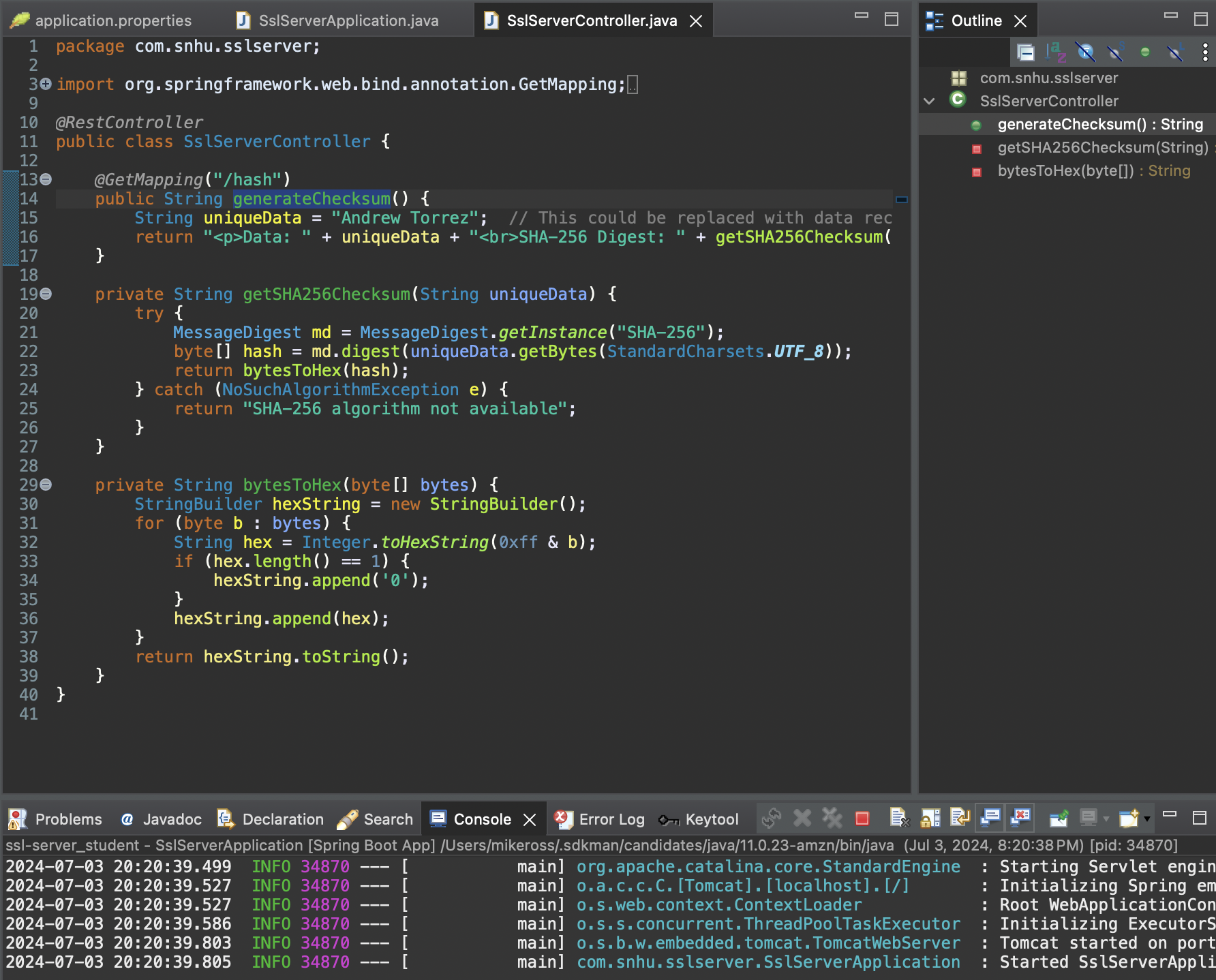
**Dependency-Check Report**:



### **Functional Testing**

**Review Process**: Manually reviewed code for syntactical, logical, and security vulnerabilities. Ensured the application is functioning correctly and securely.

**Evidence**:



**Description of Testing Steps and Outcomes**:

* Verified that the application calculates the checksum correctly.
* Ensured the SSL configuration is correctly set up and the application runs without security warnings or errors.

### **Summary**

The code for Artemis Financial's application has been refactored to comply with security testing protocols. The primary areas of security addressed by refactoring include the implementation of SHA-256 cryptographic hashing for data integrity and enabling HTTPS for secure communications. The vulnerability assessment process involved identifying potential vulnerabilities through code reviews and static analysis tools, analyzing the impact of these vulnerabilities, refactoring the code to address them, and conducting thorough testing to ensure no new vulnerabilities were introduced.

To add layers of security, SHA-256 hashing was implemented to ensure data integrity, HTTPS was configured to protect data in transit, and a dependency check was performed to identify and mitigate vulnerabilities in third-party libraries. This multi-layered approach enhances the overall security posture of the application.

### **Industry Standard Best Practices**

Industry-standard best practices were applied to mitigate known security vulnerabilities in Artemis Financial's application. By implementing strong cryptographic techniques like SHA-256 and securing data transmission channels with HTTPS, the security of the application was significantly improved. Regular dependency checks using the OWASP dependency-check plugin ensure that vulnerabilities in third-party libraries are continuously monitored and addressed.

Applying these best practices enhances the security, compliance, and trustworthiness of the application. It ensures that the application meets regulatory requirements and maintains the integrity and confidentiality of financial data. Continuous monitoring, regular security audits, and ongoing security training for the development team are recommended to maintain a robust security posture.