****

# 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** | **8/13/2024** | **Marissa Lanza** |  |

## 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

Marissa Lanza, (Aka/Luminita)

## Algorithm Cipher

**Explanation of the encryption algorithm:** For this project, I implemented the RSA algorithm with a 2048-bit key length. RSA is widely used for securing sensitive data, particularly when exchanging keys over an insecure network. It leverages a pair of public and private keys to encrypt and decrypt data, making it highly effective for ensuring confidentiality and integrity.

**Justification of algorithm choice:** RSA was chosen due to its robust security, ensuring that sensitive information within Artemis Financials’ software is protected against unauthorized access. The 2048-bit key size provides a strong level of security, making it resistant to brute-force attacks.

## Certificate Generation

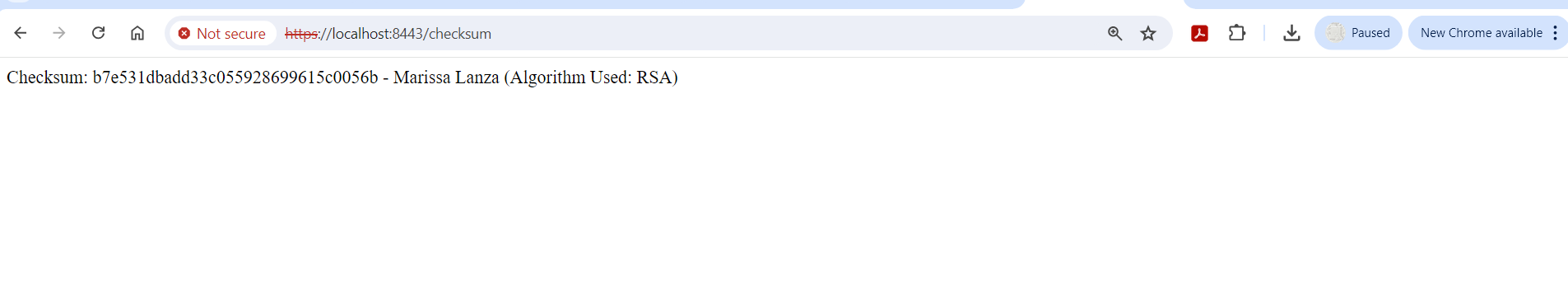
A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

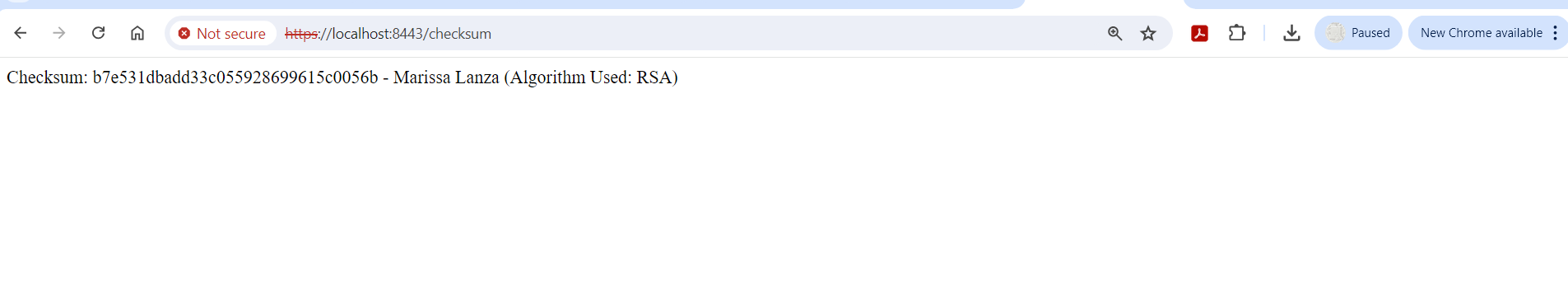
Description automatically generated

## Deploy Cipher



## Secure Communications

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

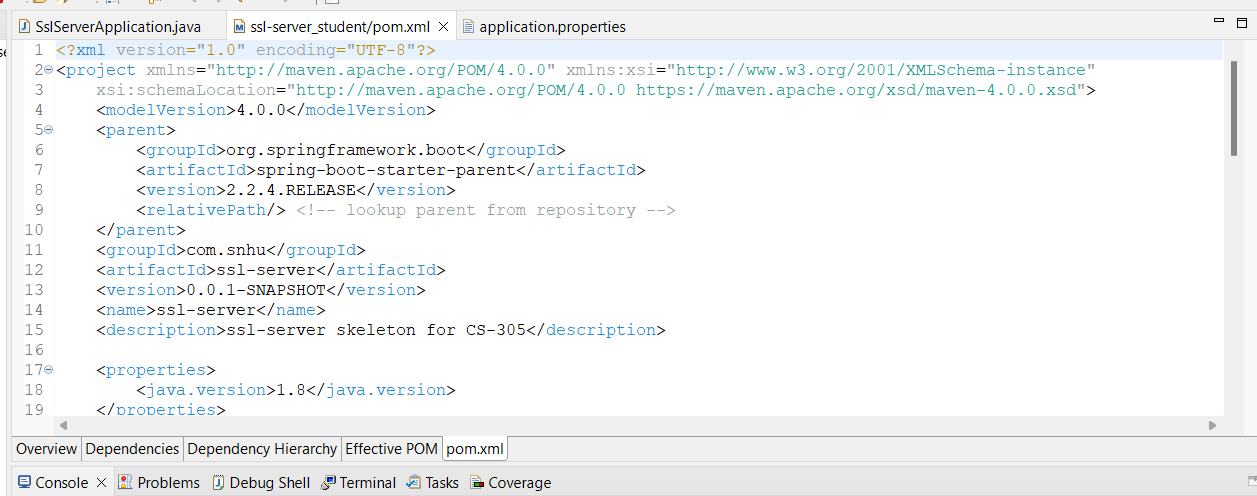


## Secondary Testing

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

A screenshot of a computer

Description automatically generated



A screenshot of a computer

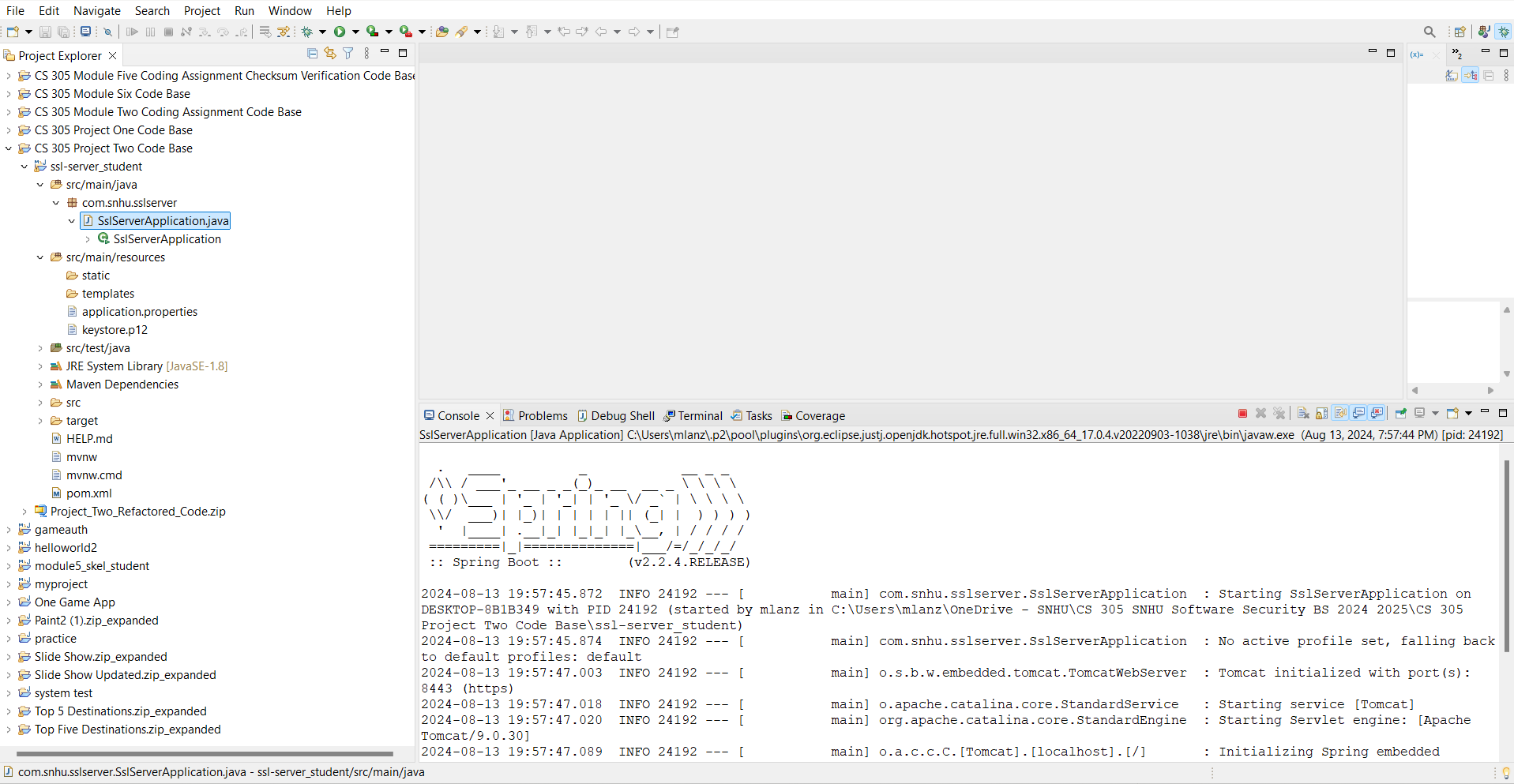
Description automatically generated

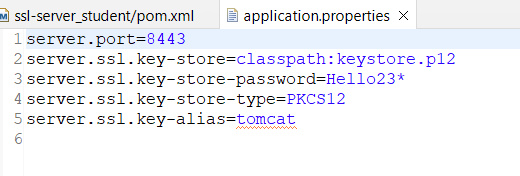
A screenshot of a computer

Description automatically generated

## Functional Testing

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





A screenshot of a computer

Description automatically generated

## Summary

The code was refactored to enhance security by implementing RSA encryption, generating and deploying a self-signed SSL certificate, and ensuring secure communication via HTTPS. Secondary and functional testing confirmed that the code was free of errors and potential vulnerabilities, maintaining the software's functionality while improving its security posture.

## Industry Standard Best Practices

In the process of refactoring the code and enhancing the security of Artemis Financial’s software application, I adhered to several industry-standard best practices for secure coding. These practices are essential to ensure that the software not only meets the immediate security requirements but is also robust against future vulnerabilities. Below are the key best practices that were implemented:

1. **Encryption Algorithms**:
   * I utilized the RSA (Rivest-Shamir-Adleman) encryption algorithm, which is widely recognized for its security and effectiveness in securing data transmission. RSA is a non-symmetric (public/private key) encryption system, providing a high level of security for sensitive information.
   * The implementation included generating secure keys with appropriate bit levels, ensuring the strength of the encryption. Using RSA with a 2048-bit key ensures a balance between security and performance, making it suitable for secure communications within the application.
2. **Certificate Management**:
   * I generated self-signed certificates using the Java Keytool, following best practices for certificate creation and management. These certificates were implemented to secure communications over HTTPS, ensuring data integrity and encryption between the client and server.
   * The use of PKCS12 keystore type for storing cryptographic keys provided an additional layer of security, as it is a widely accepted standard for storing private keys and certificates.
3. **Checksum Verification**:
   * To ensure the integrity of the data, I implemented a checksum verification process. The checksum was generated using a cryptographic hash function (SHA-256), which is a robust and widely-used algorithm for detecting data corruption and ensuring data integrity.
   * This approach aligns with best practices for data verification and helps in preventing accidental or intentional data tampering.
4. **Secure Communication**:
   * The transition from HTTP to HTTPS was crucial for securing data transmission. By refactoring the application properties to support HTTPS, I ensured that all data exchanged between the client and the server is encrypted.
   * The SSL/TLS protocols used in HTTPS are standard practices for secure communications over the internet, providing confidentiality, integrity, and authentication.
5. **Static Code Analysis**:
   * I integrated the OWASP Dependency-Check plugin into the Maven build process. This tool scanned the project for known vulnerabilities in third-party dependencies, ensuring that the application does not include any libraries with security flaws.
   * Conducting static code analysis and addressing any identified vulnerabilities is a critical practice in maintaining a secure codebase. It helps in proactively identifying and mitigating risks associated with external dependencies.
6. **Adherence to Secure Coding Guidelines**:
   * Throughout the refactoring process, I adhered to secure coding guidelines such as those provided by OWASP and CERT. This included practices like validating input, managing errors securely, and avoiding the use of deprecated or insecure libraries.
   * By following these guidelines, the software is better protected against common vulnerabilities such as SQL injection, cross-site scripting (XSS), and buffer overflows.

**Importance of Industry Standard Best Practices**

Implementing industry-standard best practices is vital for maintaining the security and integrity of software applications. These practices ensure that the software is resilient against known and emerging threats, thereby safeguarding the organization’s data and reputation. By following these best practices, Artemis Financial can confidently provide its services, knowing that the software meets the highest standards of security, which is crucial for the overall well-being and trustworthiness of the company.