****

# 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** | **June 16, 2024** | **Yancarlo Guzman** | **Start** |
| **1.1** | **June 20, 2024** | **Yancarlo Guzman** | **Algo-ci/Sec-com** |
| **1.2** | **June 21, 2024** | **Yancarlo Guzman** | **Finish** |

## 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: Yancarlo Guzman

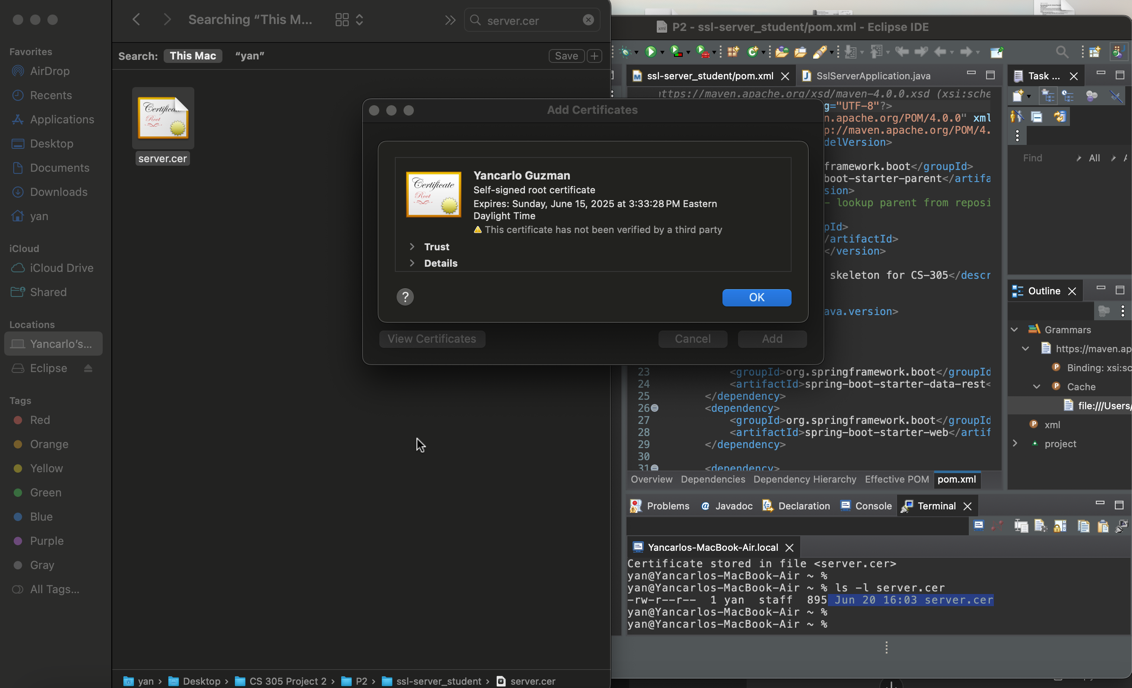
## Algorithm Cipher

A combination of RSA and SHA-256 is what I advise. This is so that only the intended recipient may decode the data, which is a security feature of RSA. SHA-256 creates a distinct hash for every message, ensuring data integrity. Second, SHA-256 and RSA are both well-supported and have been the subject of in-depth security research. It is compliant with legal standards and industry best practices to use RSA and SHA-256 for financial data protection.   
In conclusion, Artemis Financial's online application can benefit from a strong security solution that combines SHA-256 for data integrity and RSA for encryption.

1. **Overview of RSA Encryption Algorithm Cipher:** RSA is an asymmetric encryption algorithm using a public key for encryption and a private key for decryption. This ensures that data encrypted with the public key can only be decrypted by the private key holder.
2. **Hash Functions and Bit Levels:** I recommend using the SHA-256 hash function alongside RSA. SHA-256 generates a 256-bit hash, offering strong security against collisions and pre-image attacks.
3. **Random Numbers, Symmetric vs. Asymmetric Keys:**
   1. **Random Numbers:** RSA relies on large prime numbers and secure random number generation for key creation.
   2. **Symmetric vs. Asymmetric Keys:** RSA uses asymmetric keys (public and private). The public key encrypts data, while the private key decrypts it.
4. **History and Current State of Encryption Algorithms:** RSA, developed in 1977, is a proven, industry-standard encryption method. It remains widely used and trusted for secure data transmission.

## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

Description automatically generated

## Secure Communications

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

After verifying the certificate on safari it now displays as secure!

A screenshot of a computer

Description automatically generated

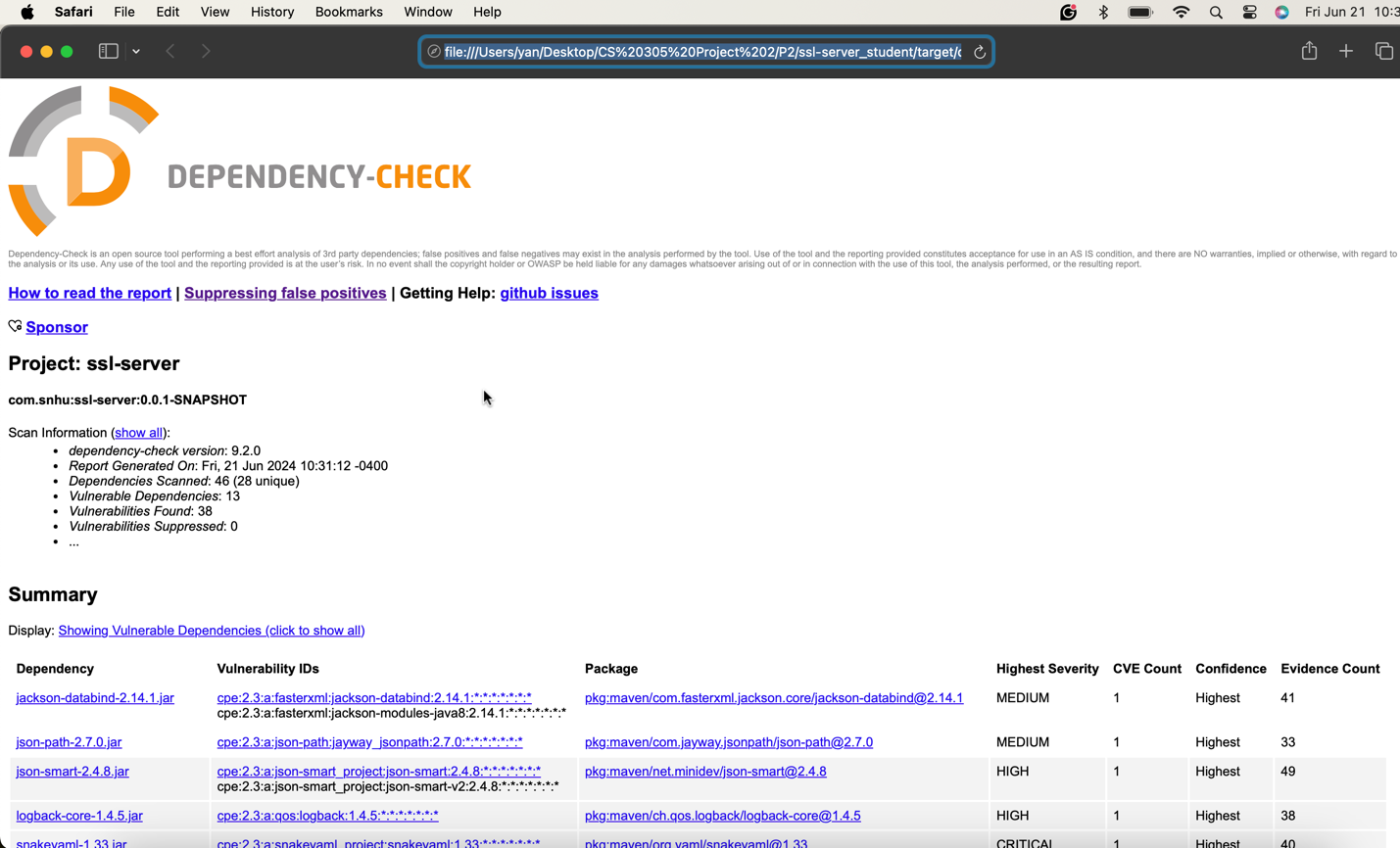
## Secondary Testing

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

A screen shot of a computer

Description automatically generated

Above is screenshot of the refactored code executed without errors. Below is the dependency-check report.



## Functional Testing

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

A screen shot of a computer program

Description automatically generated

## Summary

I had to do a lot of reworking on this project to make sure the code complied with current security testing guidelines. The main goals of this refactoring process were to strengthen the application's security and mitigate any potential vulnerabilities found in the vulnerability assessment process flow diagram that is supplied in the Supporting Materials section.

Security Aspects Addressed  
Secured “RestController” Implementation: Adding a secured “RestController” to safely handle the application's RESTful services was one of the main enhancements. By making sure that every endpoint is adequately secured, this controller lowers the possibility of illegal access and data breaches.  
SHA-256 Integrity Hashing: Because of the SHA-256 hashing algorithm's strong security characteristics and low collision risk, I decided to utilize it. This option makes sure that private information, including user passwords, is safely hashed, making it far more difficult for bad actors to utilize.

Technique for Including Security Layers  
Alignment of Vulnerability Assessment: The vulnerabilities indicated in the assessment diagram were carefully considered and addressed in the “ServerController” class. The program has now provided users with a more secure environment by putting security best practices into practice and making sure that suggested protocols are followed.  
Frequent Dependency Checks: I advise doing dependency checks once or twice a month to keep the application's security posture intact. By taking a proactive stance, such vulnerabilities will be quickly identified and mitigated, protecting the application from new attacks.  
pom.xml's plugin management section: Updating the plugins in the pom.xml file is essential to preserving good security. Updating these plugins on a regular basis guarantees that the most recent security updates and improvements are implemented, shielding the program from known vulnerabilities.

By adhering to these guidelines, the refactored code creates a strong foundation for continuing security management in addition to addressing present security issues. This strategy will support preserving user confidence and protecting critical enterprise data.

## Industry Standard Best Practices

I used industry best practices, such as hashing using SHA-256 and encrypting with RSA, to improve the software application's security. Strong data integrity and confidentiality are ensured by this combination. To handle RESTful services securely and lower the danger of unauthorized access and data breaches, I also added a secured “RestController”. It was advised to perform routine dependency checks and changes to the pom.xml plugins in order to quickly detect and address vulnerabilities and guarantee that the program is safe from new threats.  
  
By implementing these best practices, you may preserve user confidence and safeguard important company information while also fortifying the application's security and adhering to industry and legal regulations. Maintaining continuous adherence to security procedures and protecting the company's reputation depend on this proactive security management strategy.