****

# 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** | **[Date]** | **Adam Vosburg** | **Final Version** |

## Client



## 

## Developer

Adam Vosburg

04/14/2024

## Algorithm Cipher

I recommend using the Advanced Encryption Standard (AES) cipher for encryption. AES is a symmetric-key algorithm widely adopted as a secure encryption standard due to its efficiency, support for various key lengths, and proven resistance to cryptanalytic attacks (Stallings, 2017).

AES supports hash functions like SHA-256 and SHA-3 for generating message digests. It operates on fixed block sizes of 128 bits and supports key lengths of 128, 192, and 256 bits, with larger key sizes providing higher security at the cost of performance (Daemen & Rijmen, 2002). As a symmetric-key algorithm, AES relies on the generation of secure, random keys for both encryption and decryption. This use of random numbers is crucial for ensuring keys are varied and unpredictable, which is important for maintaining the security of the encryption process

Despite its age, AES is still considered secure and the standard for encryption in many applications. Its strength lies in its robust design, which has withstood extensive cryptanalysis efforts. The algorithm's use of substitution-permutation networks, multiple rounds of transformation, and key expansion techniques make it resistant to various cryptanalytic attacks, including linear and differential cryptanalysis.

Even with the advent of new computing technologies like quantum computing, AES is expected to remain secure for the foreseeable future. Its large key space and the computational complexity required to break it make it a formidable encryption algorithm, even for powerful quantum computers. However, it's important to note that as computing power continues to increase, larger key sizes (such as 256-bit keys) may become necessary to maintain the desired level of security (NIST, 2001).

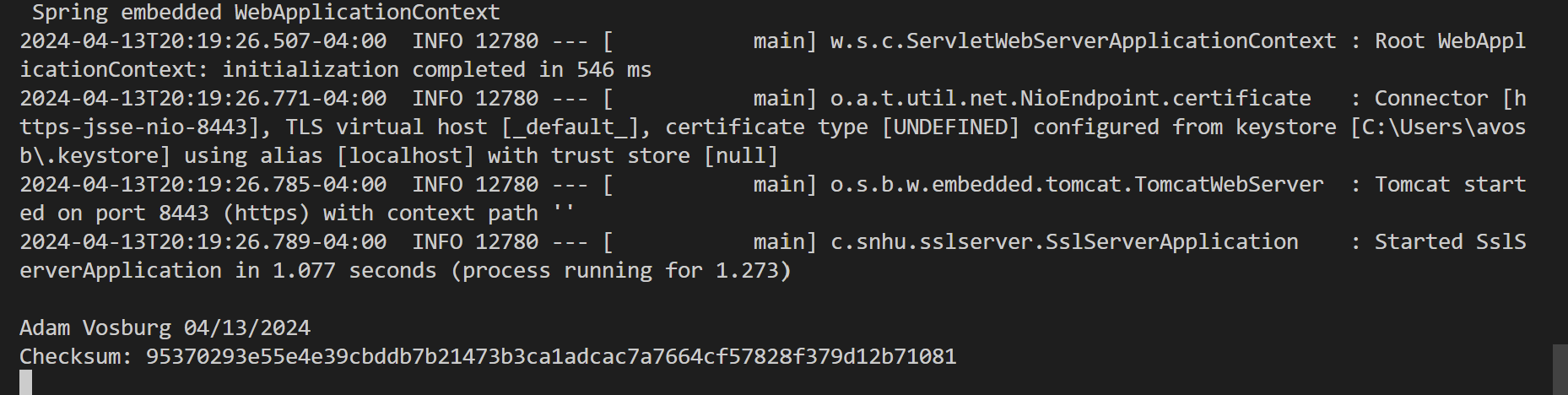
Overall, AES has proven to be a reliable and trustworthy encryption standard, earning the confidence of governments, industries, and organizations worldwide. Its widespread adoption and adherence to rigorous security standards have solidified its position as the go-to encryption algorithm for protecting sensitive data in a wide range of applications.

## Certificate Generation

## 

## 

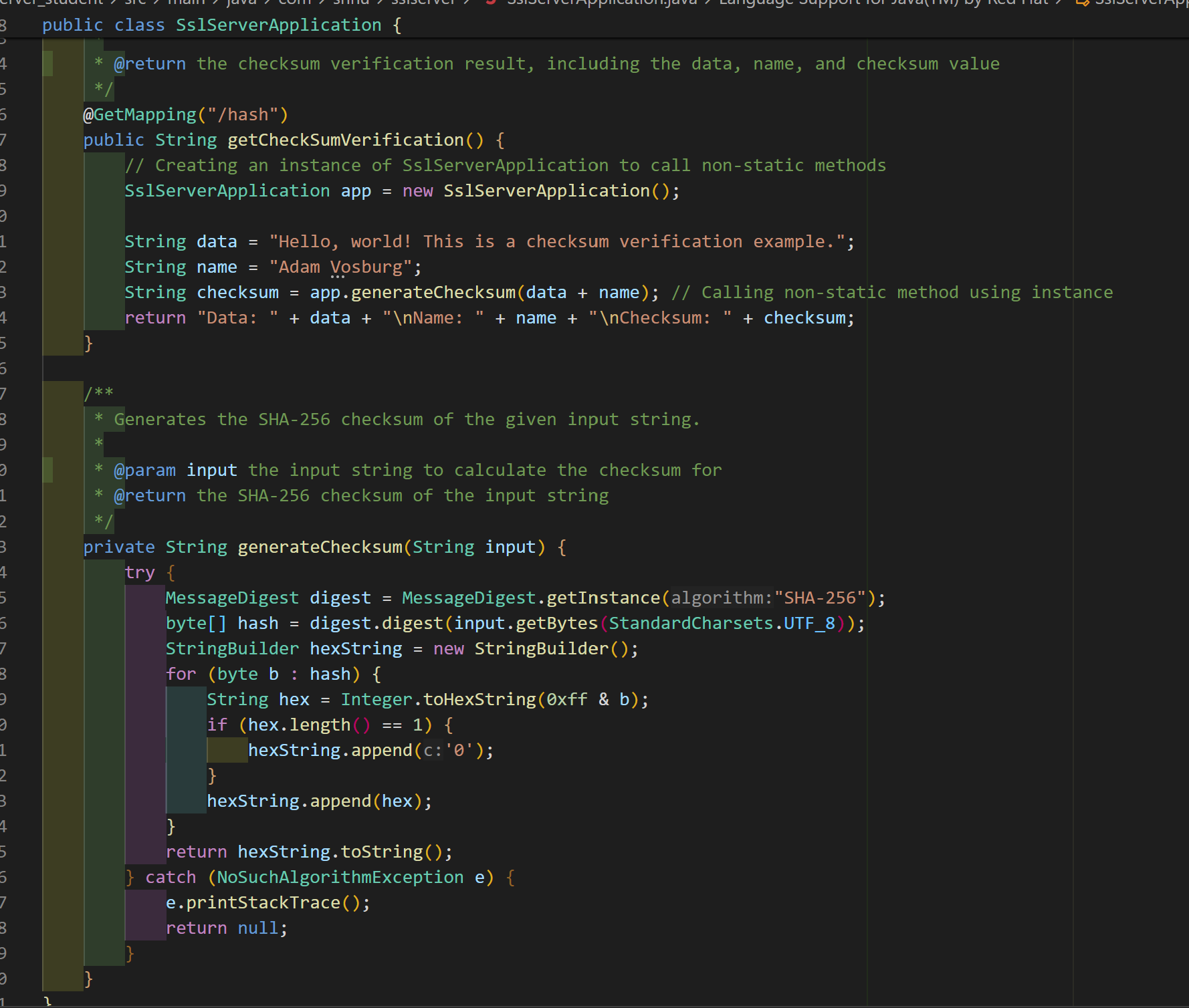
## Deploy Cipher

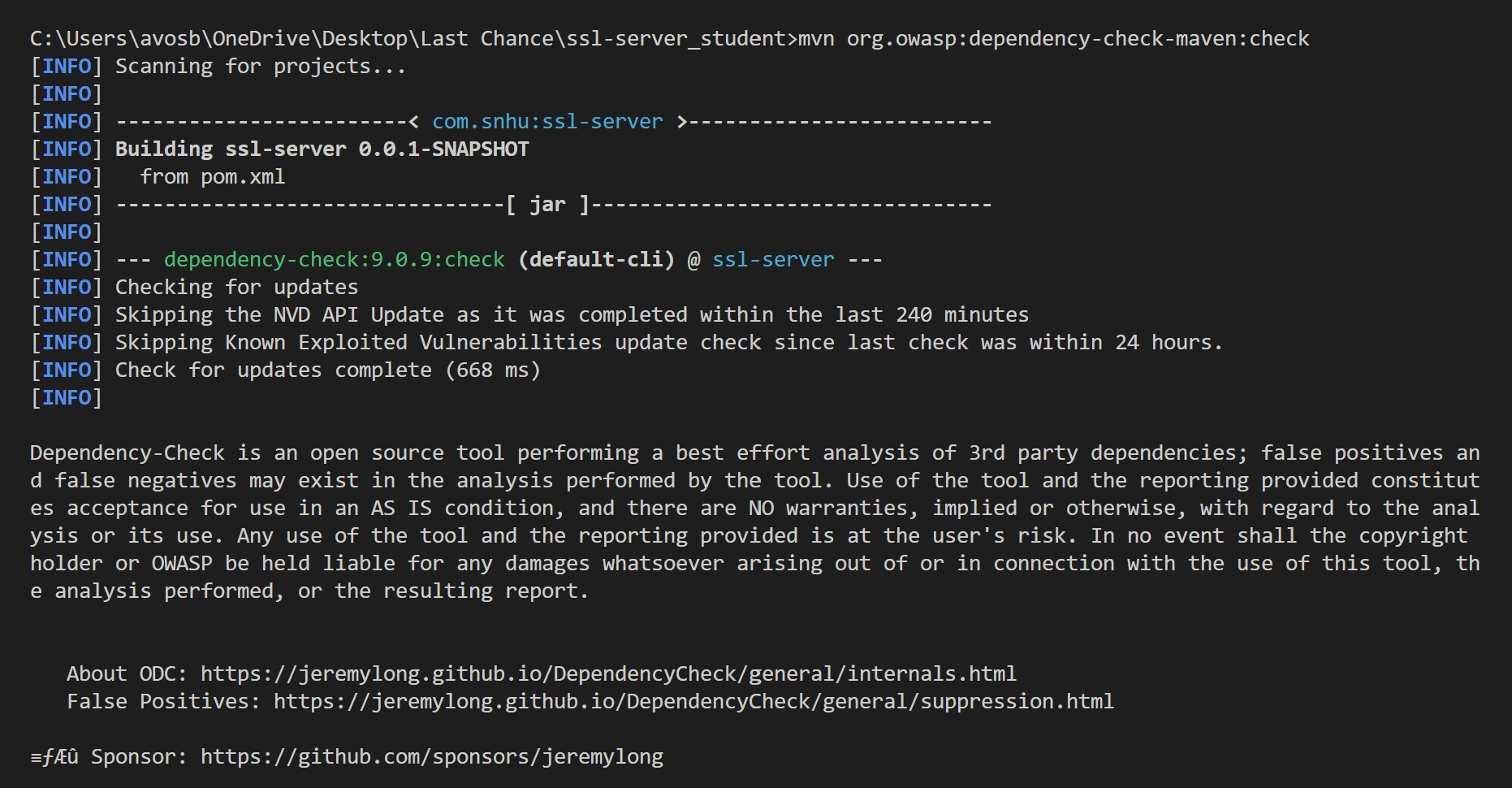


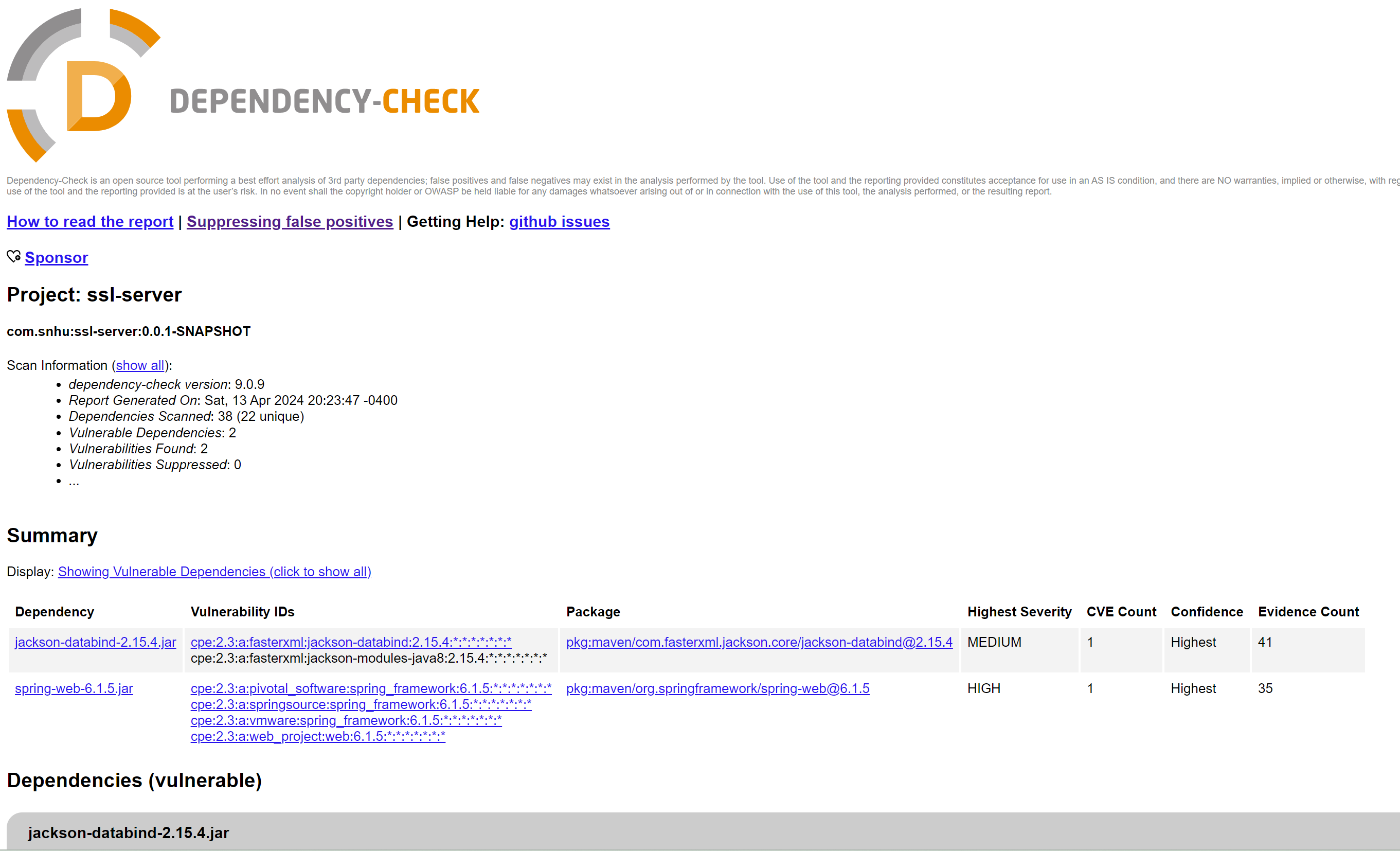
## Secure Communications

## 

## Secondary Testing







## Functional Testing

## 

## 

## Summary

During the process of refactoring the SSL Server application, I made significant progress in enhancing its security by addressing key areas highlighted in the Vulnerability Assessment Process Flow Diagram. Starting with the Architecture Review phase, I thoroughly analyzed the application's structure, identifying potential vulnerabilities and areas for improvement. This laid the foundation for the subsequent stages of the refactoring process.

Next, in to the Cryptography phase, I implemented the Advanced Encryption Standard (AES) cipher to encrypt sensitive data, ensuring its protection against unauthorized access. I generated secure certificates using the Java Keytool in VS Code, adhering to industry best practices for encryption. The SslServerApplication class demonstrates the application of cryptographic best practices, such as using the SHA-256 hash function from the Java Cryptography Architecture (JCA) in the generateChecksum method to ensure data integrity.

In the Client/Server phase, I focused on securing client-server communication by refactoring the code to convert HTTP to HTTPS protocol. This step was crucial in maintaining the confidentiality, integrity, and availability of data transmitted between clients and the server. The successful implementation of HTTPS was verified through a screenshot of the web browser displaying a secure webpage.

Throughout the Quality and Encapsulation stages, I leaned on secure coding practices and patterns. The SslServerApplication class showcases the encapsulation of sensitive operations within appropriate class boundaries, mitigating the risk of unintended access or misuse. The adoption of the Spring Security framework further strengthened the application's security by providing robust authentication and authorization mechanisms.

To ensure the effectiveness of the refactoring efforts, I conducted in-depth testing, including secondary static testing using the OWASP Dependency-Check Maven and functional testing. These tests allowed me to identify and address any introduced vulnerabilities or syntactical, logical, and security issues. I repeatedly refined the design and code until no new vulnerabilities were discovered, ensuring a robust and secure application.

By rigorously addressing each stage of the Vulnerability Assessment Process Flow Diagram, I successfully fortified the SSL Server application against potential threats, aligning it with stringent security testing protocols and industry standards for secure software development.

## Industry Standard Best Practices

In the refactored SSL Server application, I adhered to industry standard best practices to mitigate known security vulnerabilities and maintain the application's security. The implementation of the AES cipher, a trusted and widely adopted encryption algorithm, aligns with industry recommendations for secure data protection. The generateChecksum method in the SslServerApplication class exemplifies the application of cryptographic best practices, utilizing the SHA-256 hash function from the Java Cryptography Architecture (JCA) to ensure data integrity. The use of the StandardCharsets.UTF\_8 encoding in the generateChecksum method demonstrates adherence to best practices for character encoding and internationalization, proactively addressing potential character encoding vulnerabilities.

The integration of the Spring Security framework, a widely recognized and robust security solution, showcases the application's alignment with industry standards for authentication and authorization. By leveraging Spring Security, I implemented granular access control based on user roles and permissions, upholding the principle of least privilege and ensuring a secure and modular codebase.

Throughout the refactoring process, I prioritized the use of secure coding practices and patterns. The encapsulation of sensitive operations within appropriate class boundaries, as evident in the SslServerApplication class, minimizes the risk of unintended access or misuse of sensitive functionality. Regular updates to dependencies, as reflected in the pom.xml file, ensure that the application benefits from the latest security patches and bug fixes, proactively addressing known vulnerabilities.

By adhering to these industry standard best practices, Artemis Financial can bring confidence to both its customers and stakeholders. They can do this while still demonstrating a strong commitment to safeguarding sensitive financial data. The implementation of robust encryption, secure communication protocols, and granular access controls not only lowers the risk of data breaches but also helps meet regulatory requirements and maintain the company's reputation as a trusted financial institution.

Investing in secure software development practices, as demonstrated in the refactored SSL Server application, contributes to the overall well-being and success of Artemis Financial. By prioritizing the security and integrity of its software systems, the company can foster long-term customer trust, reduce the potential for financial losses due to security incidents, and maintain a competitive edge in the industry. Adhering to industry standard best practices for secure coding positions Artemis Financial as a responsible and reliable financial institution, bolstering its reputation and ensuring its long-term success.

**References:**

Daemen, Joan & Rijmen, Vincent. (2002). The Design of Rijndael: AES - The Advanced Encryption Standard. [The Design of Rijndael: AES - The Advanced Encryption Standard | Request PDF (researchgate.net)](https://www.researchgate.net/publication/220687895_The_Design_of_Rijndael_AES_-_The_Advanced_Encryption_Standard)

NIST Updates FIPS 197, Advanced Encryption Standard (AES). (2023). *NIST*. <https://www.mybib.com/tools/apa-citation-generator>

Stallings, W. (2006). *Cryptography and network security: Principles and practices* (4th ed.). [Upper Saddle River, NJ: Pearson/Prentice Hall](https://www.amazon.com/Cryptography-Network-Security-Principles-Practice/dp/0133354695)