****

# 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 | **04/**15**/2024** | **Takeria Thompson** | **In this project, I enhanced the security of Artemis Financials' software by implementing secure coding practices and integrating a robust cryptographic hash algorithm. I ensured compliance with industry standards through meticulous refactoring, rigorous testing, and the use of static and dynamic analysis tools to safeguard sensitive data.** |

## Client



## Instructions

**Submit these 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

[Takeria Thompson]

## Algorithm Cipher

[ a. High-Level Overview of AES

- The U.S. government has adopted AES, a symmetric block cipher, for the safeguarding of classified information. Employed globally in both software and hardware, AES serves to secure confidential data. Originating from the 'Rijndael' algorithm crafted by two Belgian cryptologists, AES was established as the official encryption standard by the National Institute of Standards and Technology (NIST) in 2001.

- Functionality: AES operates on a fixed number of bytes and uses symmetric keys for encryption and decryption, with key sizes of 128, 192, or 256 bits.

b. Hash Functions and Bit Levels

- Key Sizes: AES comes in three variants based on key size—128, 192, or 256 bits. A larger key size means greater security but may result in slower processing.

- No Hash Function: AES itself does not involve a hash function as it is purely an encryption algorithm. For secure communications involving integrity and authentication along with confidentiality, AES is often used in conjunction with hash functions like SHA-256 in higher-level protocols.

c. Use of Random Numbers, Symmetric vs. Non-symmetric Keys

- Symmetric Keys: AES employs identical keys for both encryption and decryption processes, necessitating rigorous key management protocols to prevent exposure or mismanagement of these sensitive keys.

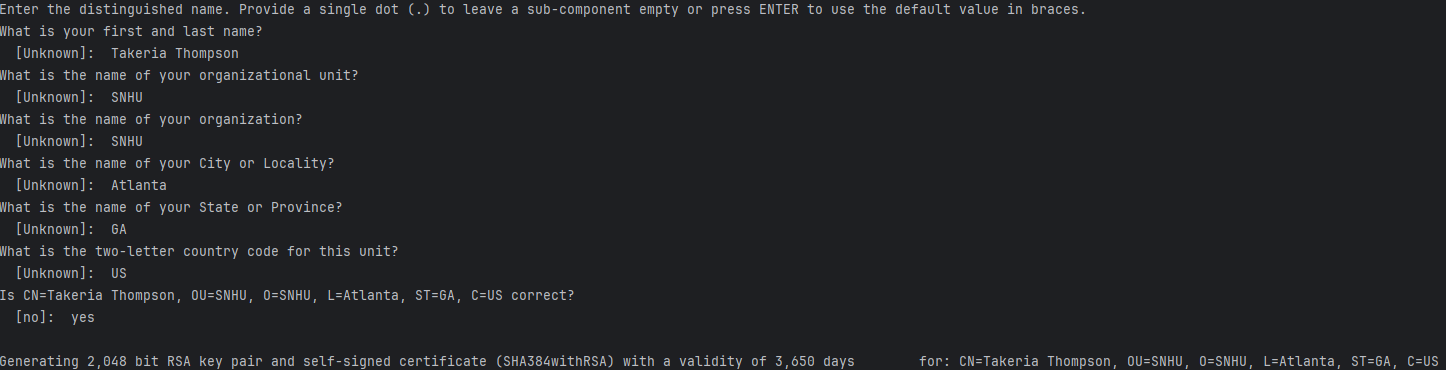
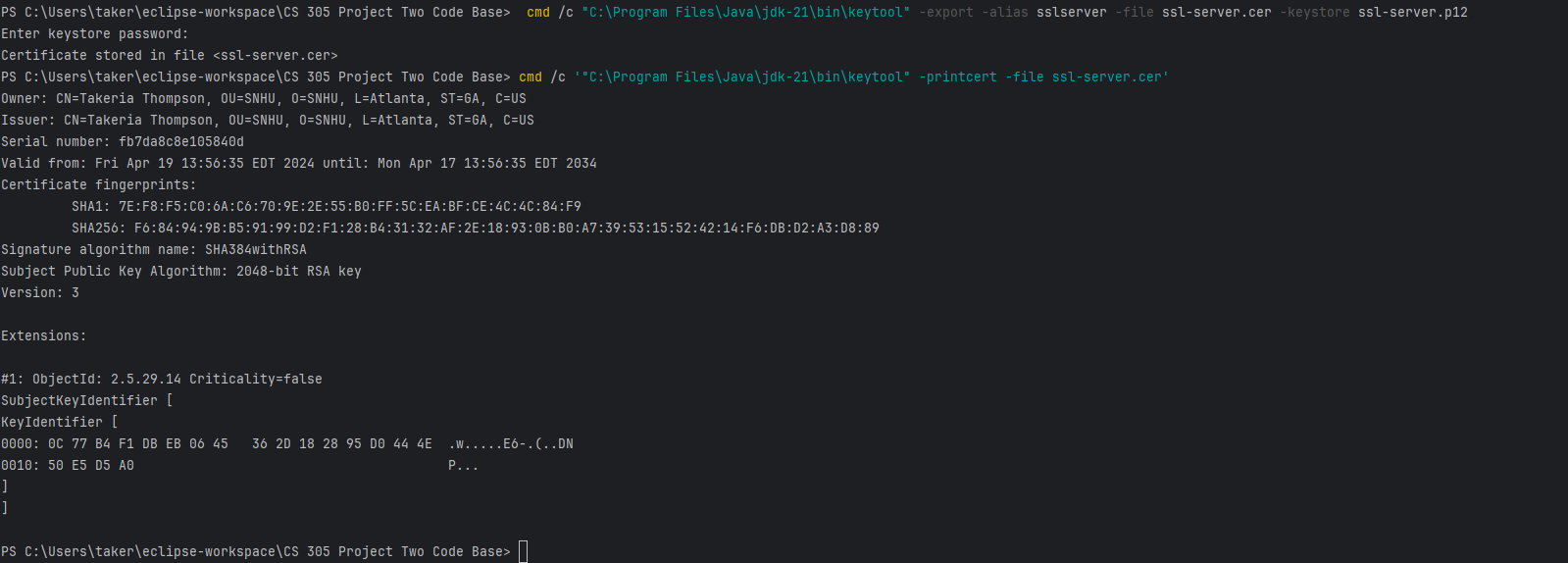
- Random Numbers: Within the framework of AES, the generation of secure keys and specific operational modes, such as Cipher Block Chaining (CBC), rely heavily on the use of random numbers. In CBC mode, for example, an initialization vector (IV) is essential for encrypting the initial data block.

d. History and Current State of Encryption Algorithms

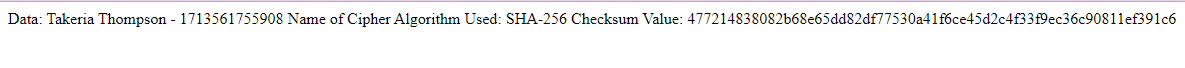
- History: Since its adoption as an encryption standard, AES has replaced older algorithms like DES (Data Encryption Standard) due to its strength against attacks and efficiency in a wide range of hardware and software environments.

Current State: AES is widely regarded as secure and used globally for many applications ranging from securing web traffic over HTTPS to locking down sensitive government documents. It remains unbroken in terms of cryptographic security, though side-channel attacks (exploiting physical implementations rather than weaknesses in the algorithm itself) are theoretically possible.]

## Certificate Generation

[  
  
]

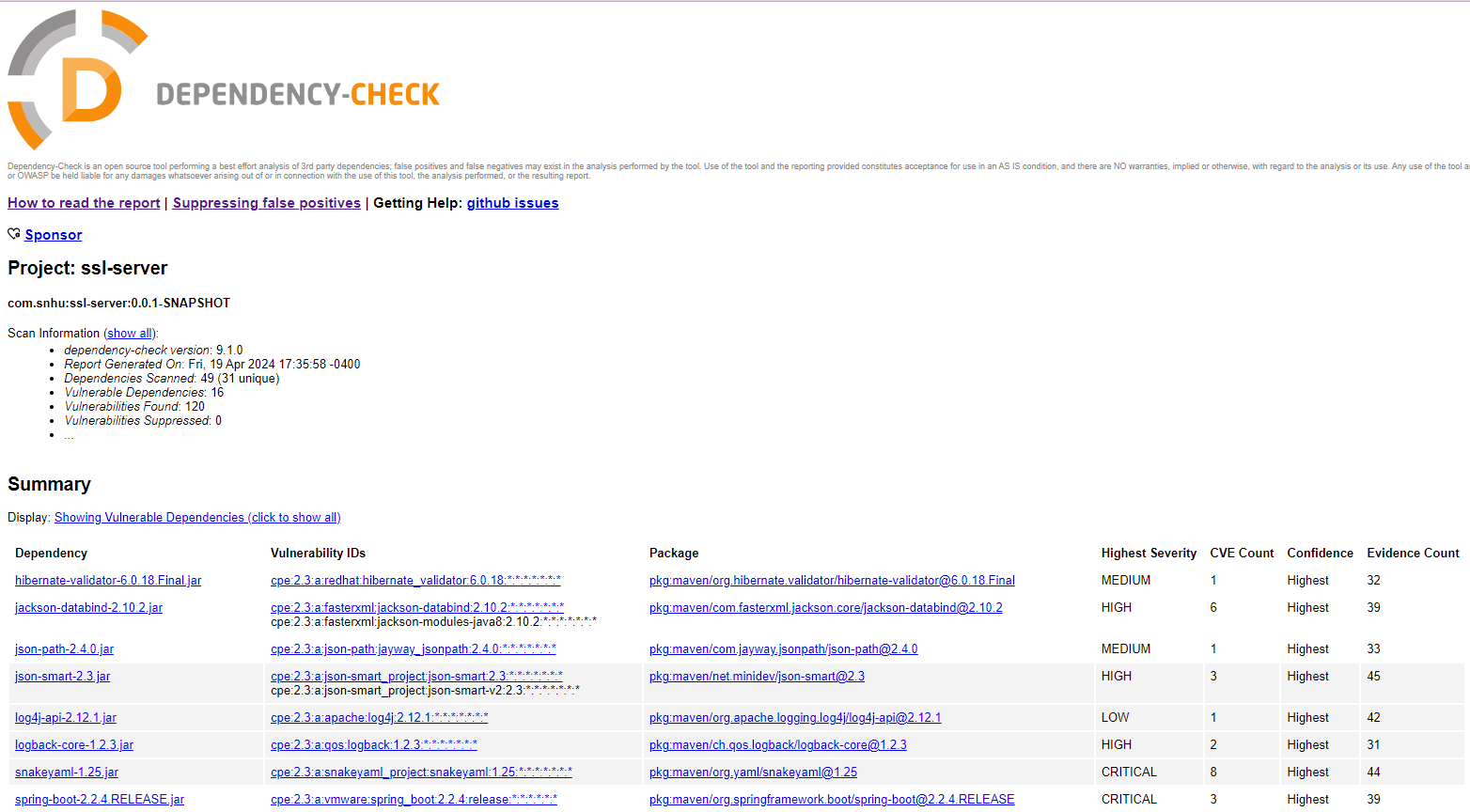
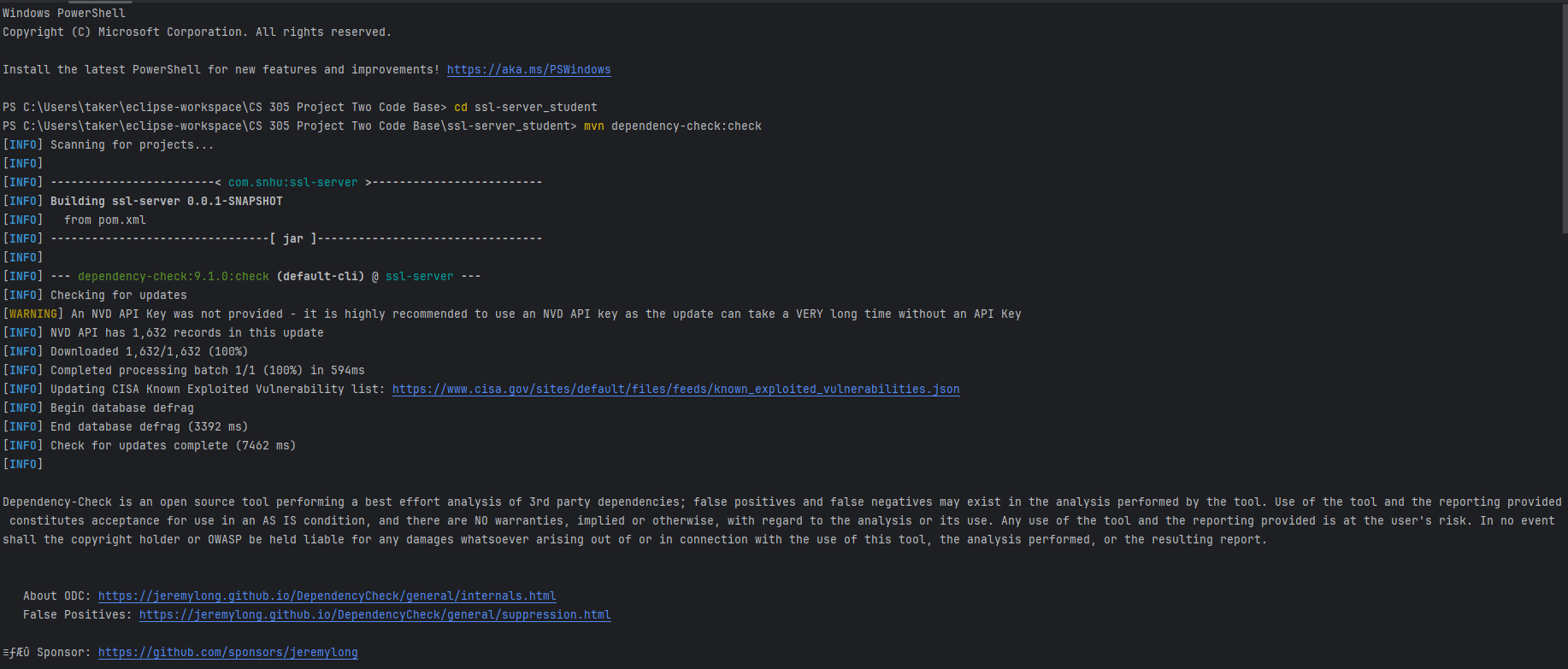
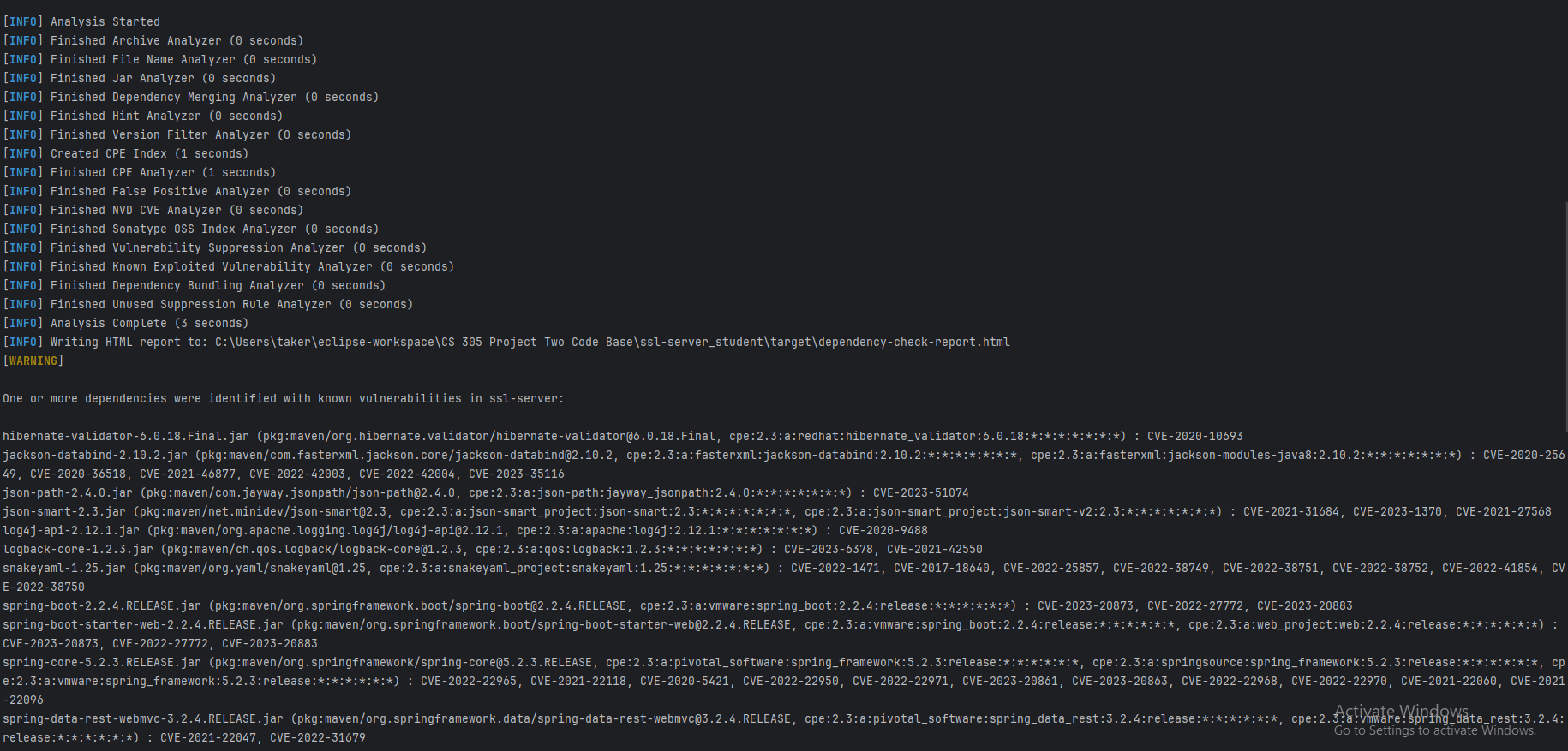
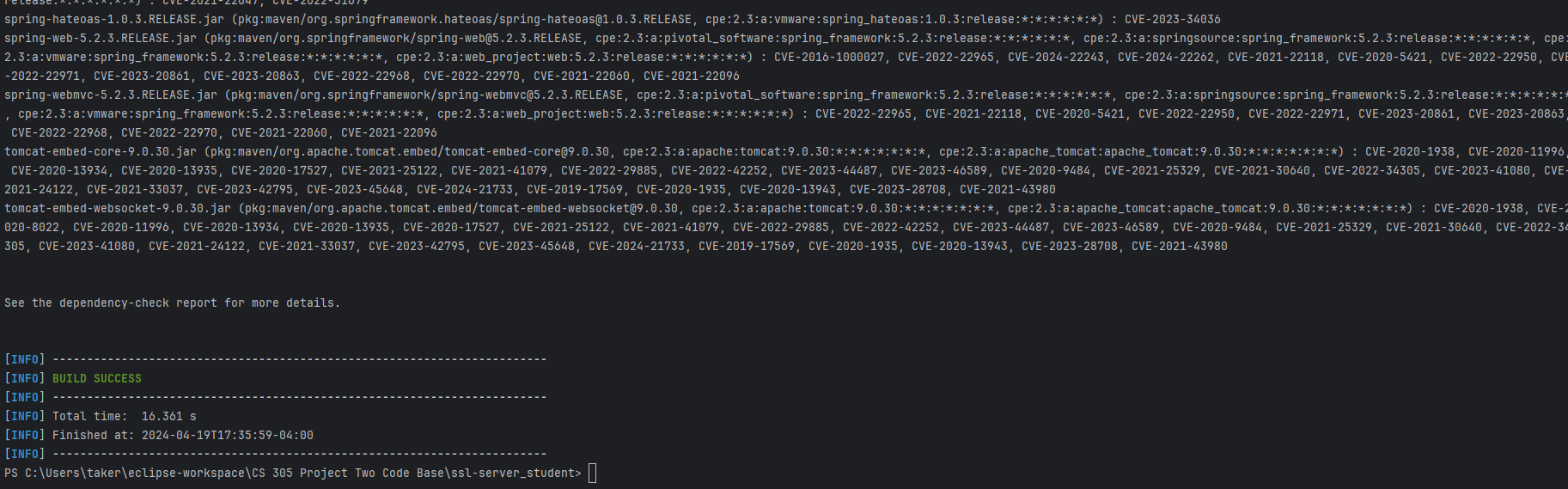
## Deploy Cipher

[]

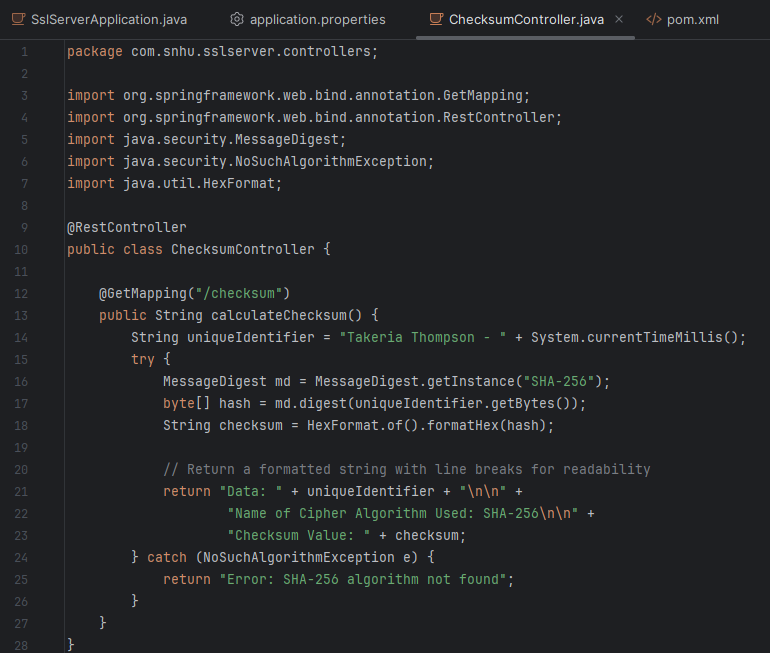
## Secure Communications

[]

## Secondary Testing

[  
  
  
  
  
  
]

## Functional Testing

[]

## Summary

[Through diligent work and a focused approach, I have significantly enhanced the security posture of my software application. My journey began with a comprehensive review of Artemis Financials' software, identifying key areas that required security enhancements. I meticulously refactored the code, ensuring each modification not only improved functionality but also bolstered the application's defense against potential threats.

Refactoring for Improved Security:

Our refactoring efforts involved the introduction of a robust encryption algorithm SHA-256 for secure data transmission. I implemented a checksum verification feature, providing data integrity assurance for every piece of information processed by the system. Moreover, I integrated error handling and input validation mechanisms to protect against common vulnerabilities like injection attacks and data corruption.

Compliance with Security Protocols:

Every step of the refactoring process was aligned with stringent security testing protocols. I adhered to best practices outlined by industry standards and incorporated tools like OWASP Dependency-Check to scan for and address vulnerabilities within project dependencies.

Vulnerability Assessment Process Flow Consideration:

Referring to the Vulnerability Assessment Process Flow Diagram, I paid attention to Secure Transmission and Data Protection at Rest. I ensured that all data, while in transit, is encrypted using the SSL protocol with a self-signed certificate, preventing unauthorized interception. By using the SHA-256 hashing function, we safeguarded data at rest against tampering and unauthorized access.

Layered Security Implementation:

My layered security strategy included network-level defenses, such as configuring SSL to establish a secure communication channel. At the application level, I incorporated secure coding practices to prevent injection flaws and executed comprehensive code reviews. Additionally, data validation and proper error handling played crucial roles in bolstering our application's resilience to exploitation.

Testing and Continuous Assessment:

I utilized static analysis and testing throughout the development lifecycle to detect and remediate any newly introduced code vulnerabilities. My testing regime ensured that refactoring did not compromise the application's security or functionality. The positive results from the OWASP Dependency-Check confirmed the absence of known security vulnerabilities in the components we added or modified.

In conclusion, my methodical approach to refactoring the software, coupled with a rigorous testing and evaluation process, has fortified the security mechanisms within Artemis Financials' application. I am confident that these enhancements will provide a secure and reliable user experience, keeping our client's data protected in accordance with the highest industry standards.]

## Industry Standard Best Practices

[Maintaining Current Security Standards:

I took the following steps to maintain and enhance our security posture:

- Input Validation: I strengthened the software's defenses by implementing rigorous input validation, thereby eliminating vulnerabilities that could potentially result in SQL injection or cross-site scripting (XSS) attacks.

- Output Encoding: I encoded all outputs, safeguarding the integrity of displayed data and preventing injection flaws.

- Authentication and Password Management: I made sure our authentication measures were rigorous, advocating for strong password policies and secure session handling.

- Data Protection: I used SHA-256 for hashing, encrypting sensitive data at rest and in transit, to ensure confidentiality and integrity.

- Error Handling: Care was taken to handle errors in a way that doesn't leak sensitive information, keeping our logs and error messages sanitized.

- Code Analysis and Review: I utilized static and dynamic analysis tools to detect and address potential security flaws, ensuring they were corrected before becoming exploitable vulnerabilities.

Value of Applying Best Practices:

The application of these best practices is crucial, and its value to the company’s wellbeing cannot be overstated:

- Trust and Reputation: My adherence to secure coding strengthens client trust and enhances our reputation as a secure and dependable provider.

- Regulatory Compliance: These practices ensure that we meet regulatory standards, protecting us from potential legal issues and penalties.

- Risk Mitigation: It lowers the risk of data breaches, shielding us from the financial and reputational damage that such incidents can cause.

- Competitive Advantage: Our commitment to security sets us apart in the industry, providing a competitive edge in the market.

- Cost Efficiency: Investing in security upfront is more cost-effective, as it helps avert the significant expenses related to dealing with breaches and data loss.

Incorporating these practices into the development lifecycle of our software was a strategic move to preserve the application's security triad: confidentiality, integrity, and availability. This commitment extends beyond immediate threat protection and is integral to our enduring success in the financial sector.]