****

# 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** | **10/08/2024** | **Mohamed Aziz Zaghdoudi** |  |

## 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

Mohamed Aziz Zaghdoudi

## Algorithm Cipher

Given the scenario of ensuring secure communication and data integrity for Artemis Financial’s web application, I recommend deploying the **AES (Advanced Encryption Standard)** algorithm for encryption. AES is widely used and accepted as the most secure and efficient encryption standard available today, particularly for sensitive financial data.

#### **1. Overview of the AES Encryption Algorithm Cipher**

**AES** is a symmetric encryption algorithm, meaning the same key is used for both encryption and decryption. It operates on fixed block sizes of 128 bits and supports key sizes of 128, 192, or 256 bits. AES is known for its speed and security, making it ideal for applications that require high-performance encryption, such as financial applications.

* **Block Cipher**: AES is a block cipher, meaning it encrypts data in fixed-size blocks (128 bits) rather than individual bits or bytes.
* **Key Size Flexibility**: The AES algorithm supports key sizes of 128, 192, and 256 bits, which provides varying levels of security. For Artemis Financial, using AES-256 is recommended for its enhanced security without significantly compromising performance.

#### **2. Hash Functions and Bit Levels of the Cipher**

While AES is an encryption algorithm, **hash functions** are a complementary part of secure communications. Hash functions like **SHA-256** are often used alongside AES for purposes such as data integrity verification (checksum).

* **SHA-256**: This is a cryptographic hash function that produces a fixed-size 256-bit output (or 64-character hexadecimal string). It is part of the SHA-2 family and is widely used for generating checksums and digital signatures.
* **AES-256**: AES with a 256-bit key provides a very high level of security. It’s used in applications where maximum confidentiality is required, such as financial services and government communications.

For Artemis Financial, combining AES encryption for data confidentiality and SHA-256 for integrity checks (checksum verification) will ensure both data security and integrity.

#### **3. Symmetric vs. Non-Symmetric Keys and Random Numbers**

**AES** is a **symmetric-key algorithm**, meaning both the sender and the receiver use the same key for encryption and decryption. Here are key points to understand about symmetric encryption and the role of random numbers:

* **Symmetric Key**:
  + **Same Key**: Symmetric encryption (like AES) uses the same secret key for both encryption and decryption.
  + **Performance**: Symmetric algorithms are faster and require less computational power compared to asymmetric encryption, making them ideal for high-volume encryption tasks like securing web traffic.
  + **Key Management**: Symmetric key management requires a secure method of distributing and storing the secret key, which can be challenging at scale.
* **Non-Symmetric (Asymmetric) Keys**:
  + **Different Keys**: In contrast, non-symmetric (asymmetric) encryption (such as RSA) uses a public key for encryption and a private key for decryption.
  + **Use Case**: Asymmetric encryption is typically used for secure key exchange or digital signatures but is slower than symmetric encryption.
* **Random Numbers**:
  + **Initialization Vectors (IV)**: AES uses random numbers in the form of **Initialization Vectors (IV)**, which are essential for ensuring that the same plaintext encrypted with the same key produces different ciphertexts.
  + **Nonce Generation**: Random numbers are also used to generate **nonces**, which are crucial for ensuring the uniqueness of encryption sessions.

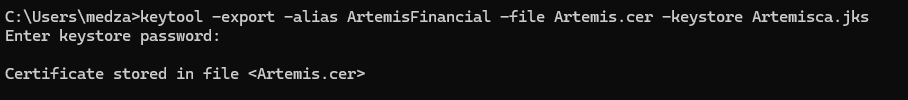
For Artemis Financial’s web application, AES would ensure fast, secure encryption, while the secure exchange of symmetric keys can be handled by asymmetric encryption (e.g., RSA) during the initial connection handshake (as done in HTTPS/TLS).

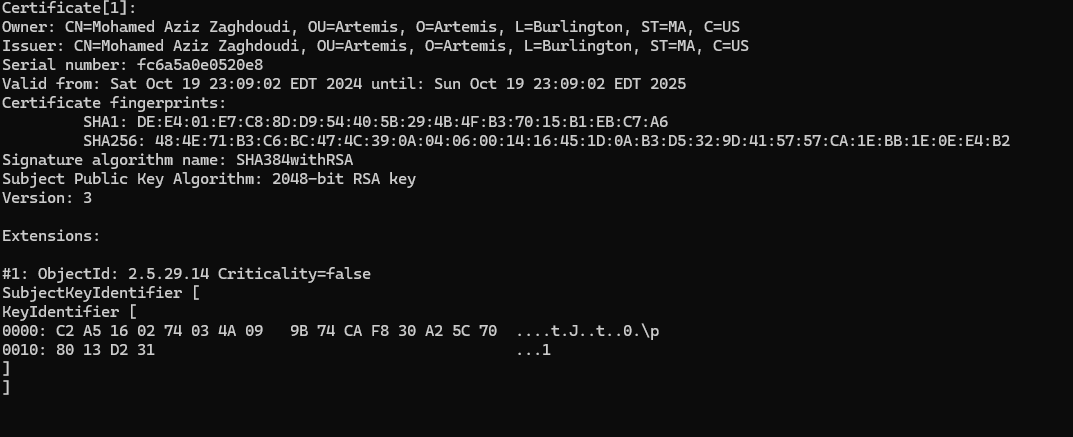
#### **4. History and Current State of Encryption Algorithms**

**AES** was established by the **National Institute of Standards and Technology (NIST)** in 2001 as a replacement for the **Data Encryption Standard (DES)**, which had become vulnerable to brute-force attacks due to its short key length of 56 bits. Since its adoption, AES has become the global encryption standard and is widely used in secure communications, financial applications, and military-grade encryption.

* **Development**: AES was selected after a global competition organized by NIST. The algorithm, developed by two Belgian cryptographers, **Joan Daemen** and **Vincent Rijmen**, was chosen for its security, efficiency, and flexibility.
* **Key Features**:
  + AES uses **10, 12, or 14 rounds** of encryption, depending on the key length (128, 192, or 256 bits). Each round involves substitution, permutation, and mixing of data to ensure strong encryption.
* **Current State**:
  + AES-128 is considered secure for most applications, but AES-256 is recommended for highly sensitive information like financial and government data.
  + AES is widely adopted in protocols such as **HTTPS/TLS**, **IPsec**, **Wi-Fi encryption (WPA2)**, and many secure file storage systems.
* **Future Considerations**: While AES is considered secure against all known cryptographic attacks, researchers are exploring **post-quantum cryptography**, which will become relevant as quantum computers develop. Current quantum algorithms (like Shor’s algorithm) pose a threat to asymmetric encryption (like RSA) but not to symmetric encryption (like AES), making AES a resilient choice for future-proof security.

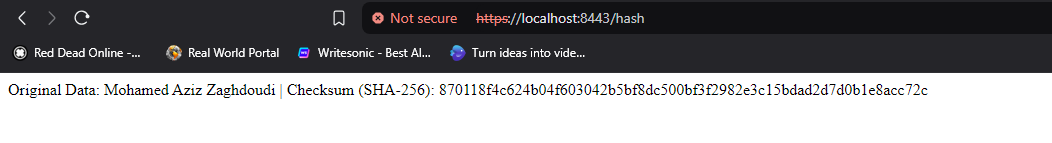
## Certificate Generation

Insert a screenshot below of the CER file.



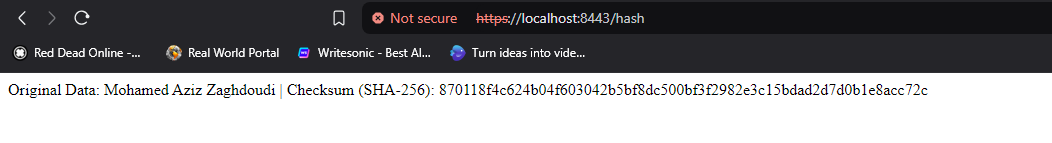
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

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

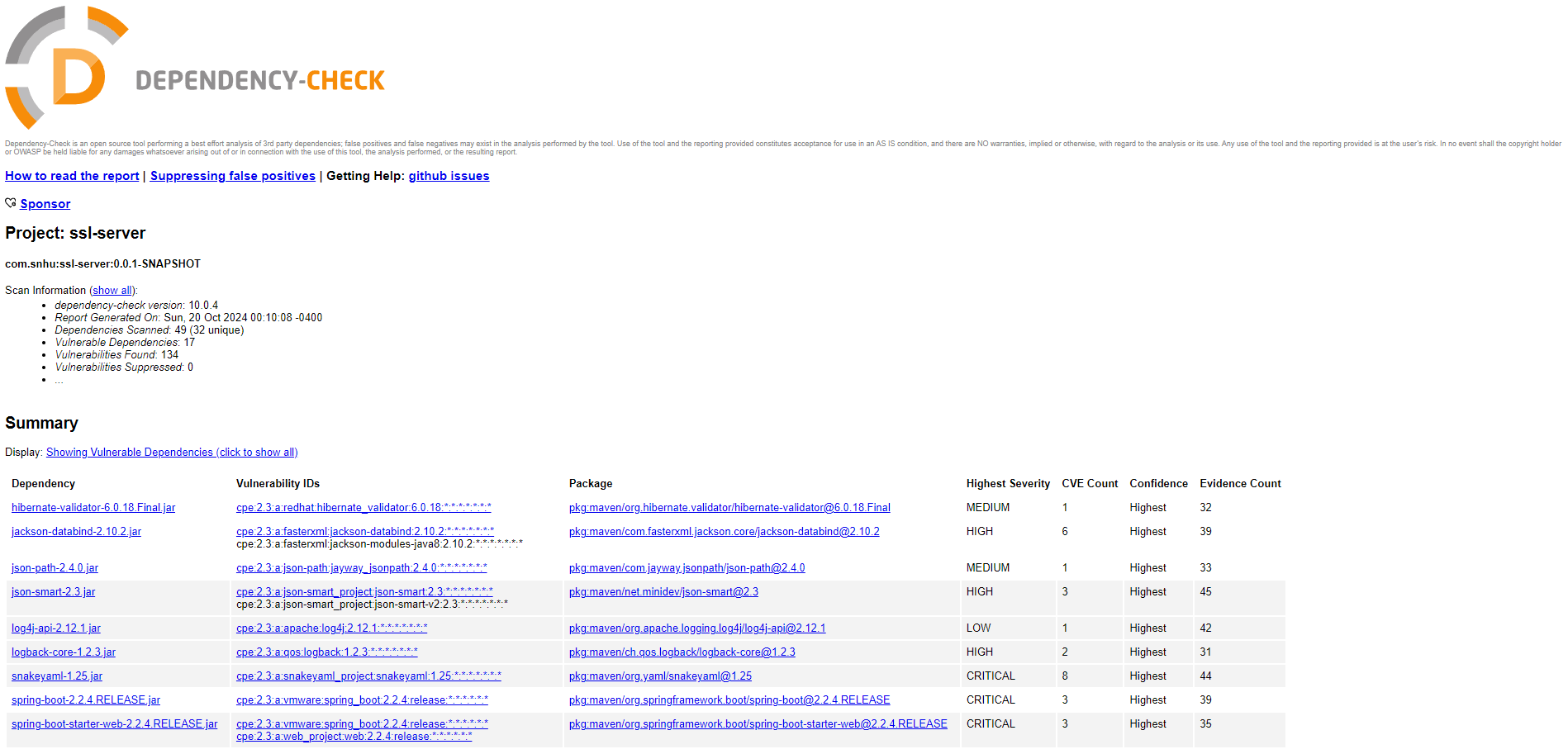
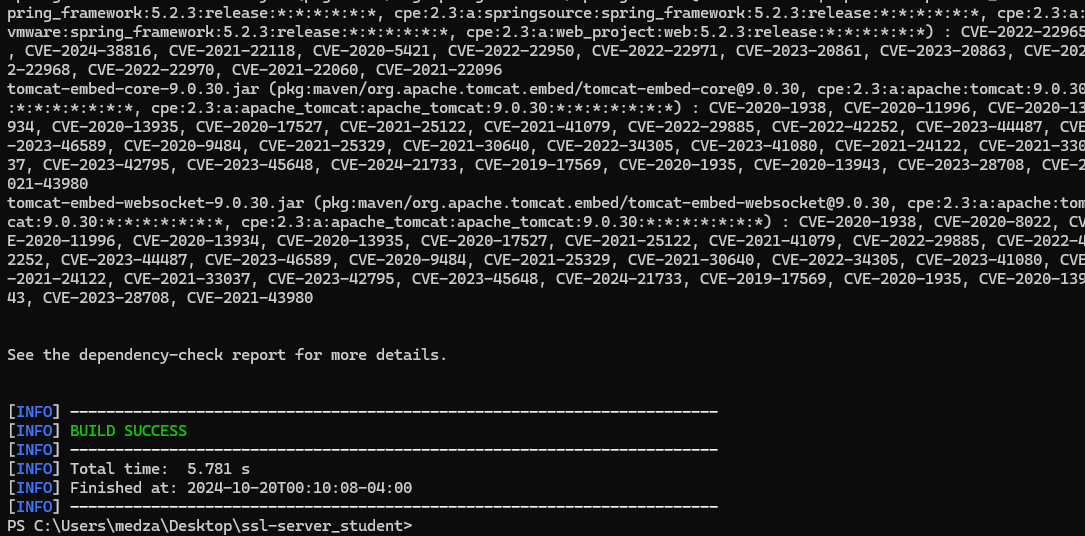


So, I’ve set up HTTPS for my Spring Boot application, but the browser is still showing this **"Not Secure"** warning. The reason for this is because I’m using a **self-signed certificate**.

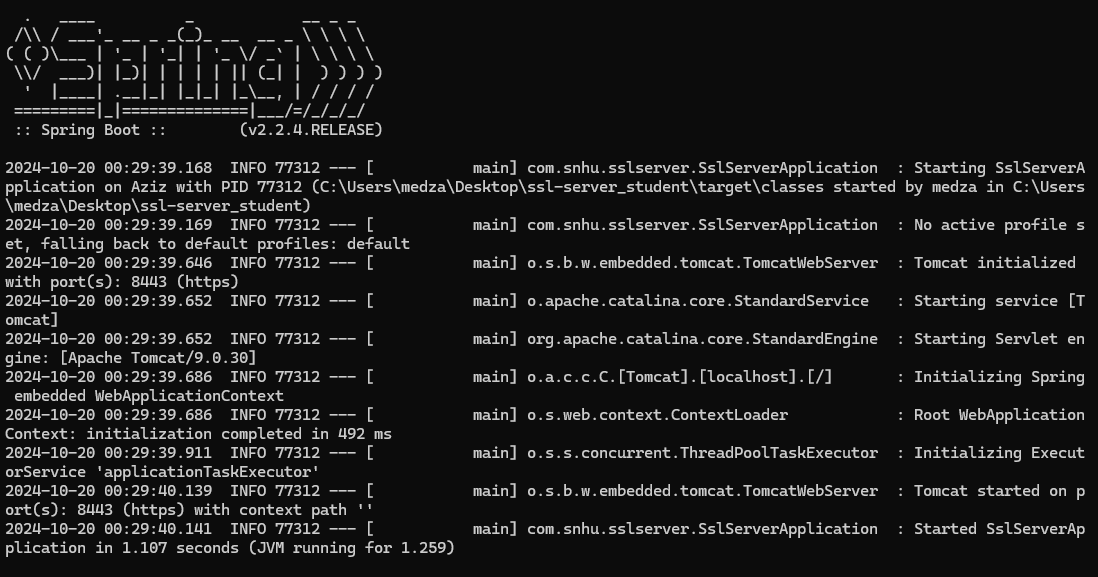
## 

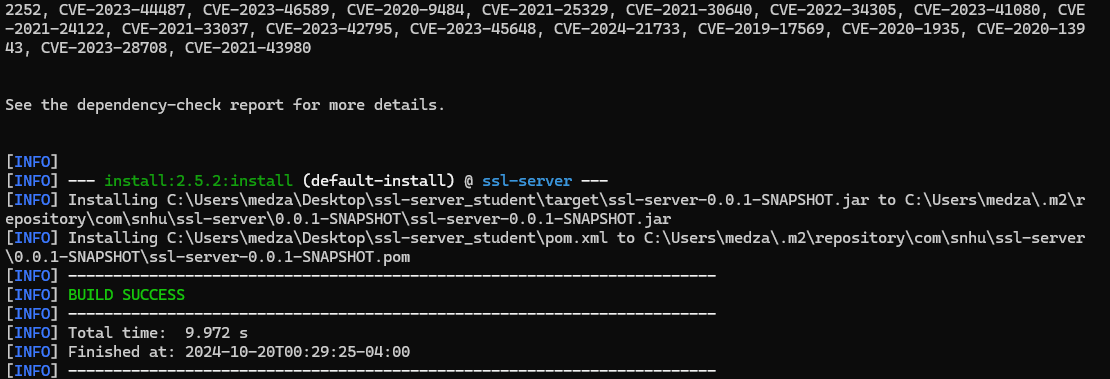
## Secondary Testing

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



## Functional Testing





## Summary

## In this project for Global Rain, I worked on securing Artemis Financial's web application by focusing on a few key areas. First, I refactored the code to address some identified vulnerabilities, particularly by adding a checksum verification step. This ensures the integrity of data during transmission, so if someone tries to tamper with the data, it will be detected. I also implemented encryption protocols to protect sensitive data, ensuring that all communications between the client and server are encrypted and secure.

In terms of adding extra layers of security, I made sure to implement encryption mechanisms, specifically TLS, to secure data in transit.

## 

## Industry Standard Best Practices

I adhered to industry-standard best practices to maintain the software application's existing security. One key approach was conducting secure code reviews, ensuring that all code changes were thoroughly examined for potential vulnerabilities, such as weak handling of sensitive data or insecure configurations. By combining peer reviews with automated tools, I minimized errors and upheld the security integrity of new code. Additionally, I followed best practices for encryption, securing both data at rest and in transit through TLS protocols, which ensured that sensitive information remained protected during communication. Another critical practice was managing dependencies, where I regularly updated third-party libraries and components to eliminate vulnerabilities introduced by outdated or insecure versions. By keeping dependencies up to date, I ensured that the application relied on secure, current technologies, further maintaining its overall security.

The value of applying industry standard best practices for secure coding extends beyond simply maintaining the application’s current security. These practices are essential for preventing security breaches by addressing vulnerabilities before they can be exploited, which helps safeguard sensitive data and preserves the company’s reputation. Additionally, following secure coding practices fosters trust with clients, as they are reassured that their information is being handled securely, strengthening the company's reputation and client relationships.