****

# 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** | **04/16/2023** | **Matthew Tyson** | **Commit** |

## 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

Matthew Tyson

## Algorithm Cipher

The encryption algorithm cipher that I recommend for Artemis Financial is AES (Advanced Encryption Standard). AES is a symmetric-key block cipher that operates on fixed-length blocks of data (128 bits) and uses a secret key of variable length (128, 192, or 256 bits). AES is one of the most widely used and secure encryption algorithms in the world. It has several advantages over other ciphers, such as:

* It is fast and efficient, both in software and hardware implementations.
* It has a high level of security, as no successful attacks have been reported against it.
* It has a simple and elegant design, based on mathematical operations such as substitution, permutation, and XOR.
* It has been standardized by the National Institute of Standards and Technology (NIST) and adopted by many governments and organizations.

AES uses a hash function to generate the secret key from a passphrase or a random number. A hash function is a one-way function that maps an input of any size to an output of fixed size, such that it is easy to compute the output from the input, but hard to find the input from the output. Hash functions are also designed to be collision-resistant, which means that it is unlikely to find two different inputs that produce the same output. Some examples of hash functions are SHA-1, SHA-2, and SHA-3.

AES uses symmetric keys, which means that the same key is used for both encryption and decryption. Symmetric keys are faster and simpler than asymmetric keys, which use different keys for encryption and decryption. However, symmetric keys also have some drawbacks, such as:

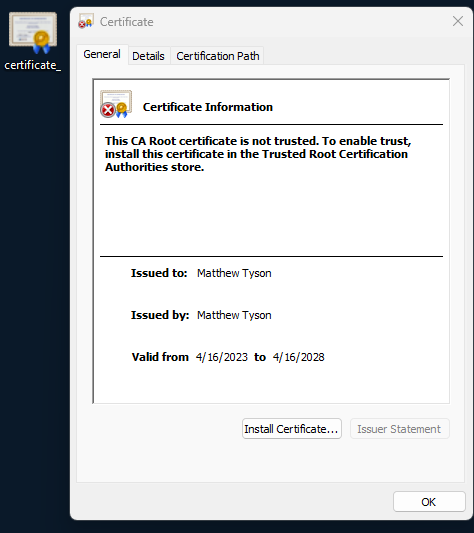
* They require a secure way to distribute and store the keys among the parties.
* They do not provide digital signatures or public-key certificates, which are useful for authentication and non-repudiation.

To overcome these drawbacks, AES can be combined with other cryptographic protocols, such as SSL/TLS (Secure Sockets Layer/Transport Layer Security), which use asymmetric keys to establish a secure channel and exchange symmetric keys for data encryption.

AES has a history of more than 20 years. It was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, in 1998. It was selected by NIST as the winner of a public competition to replace the previous standard, DES (Data Encryption Standard), which was considered insecure and outdated. AES was officially published as a federal standard in 2001 and has been widely adopted ever since.

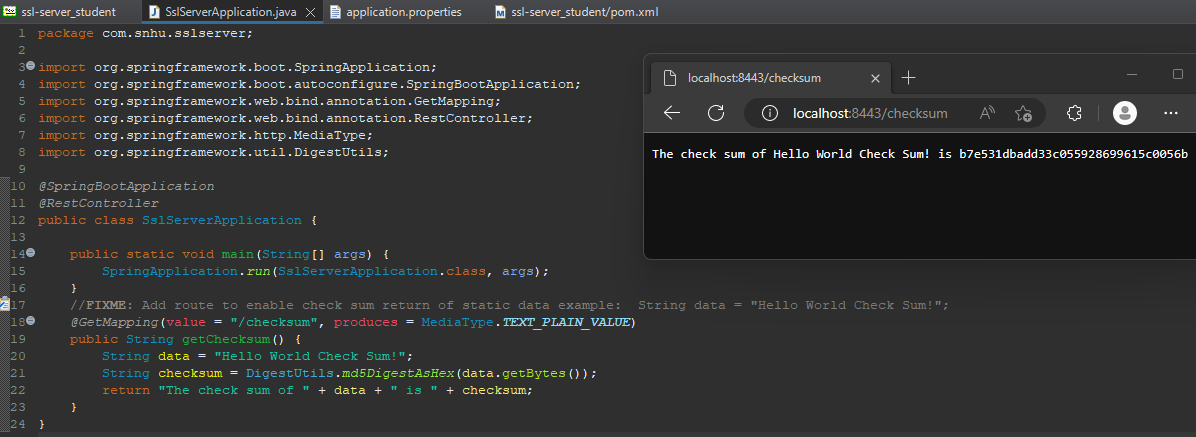
## Certificate Generation

Insert a screenshot below of the CER file.



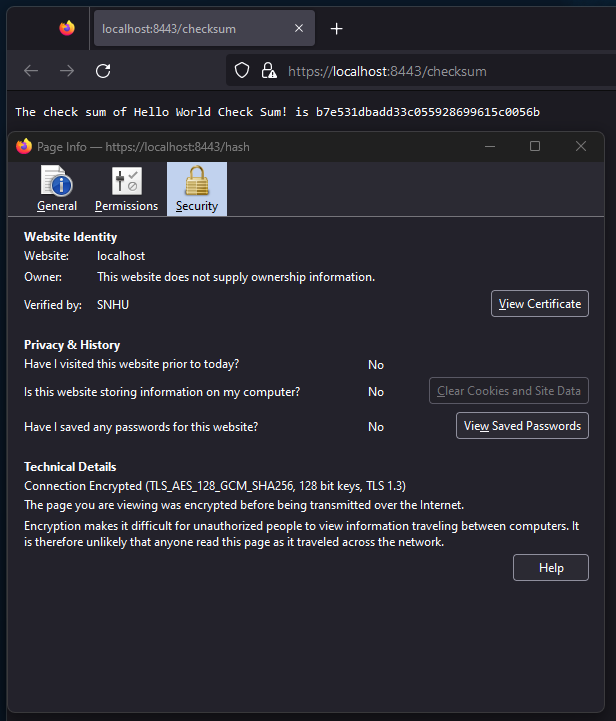
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

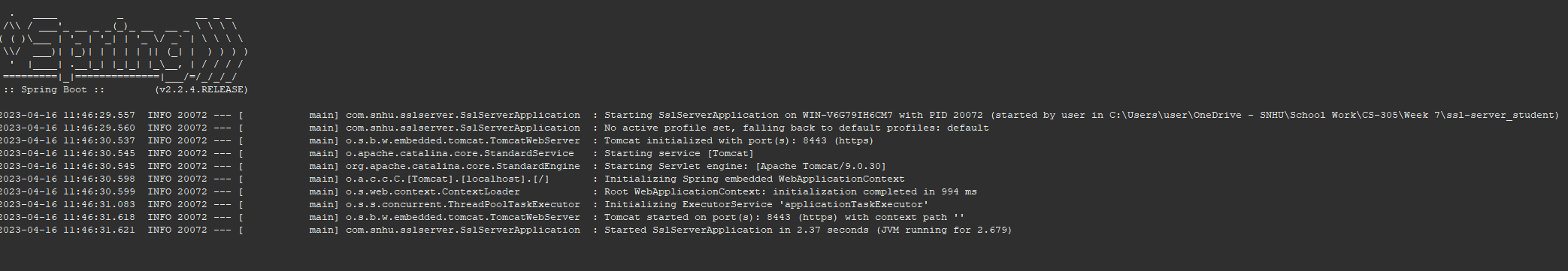
Insert a screenshot below of the web browser that shows a secure webpage.

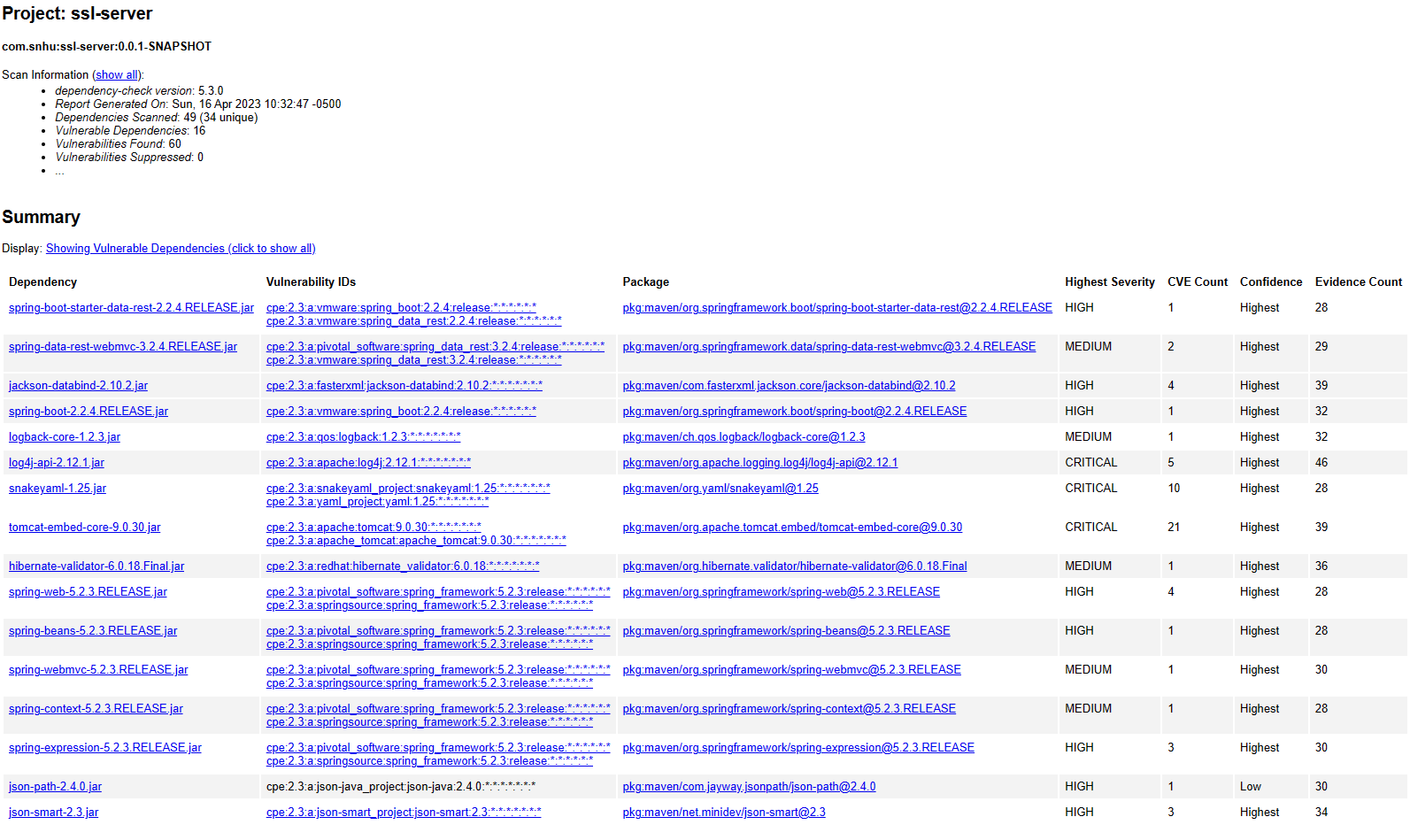


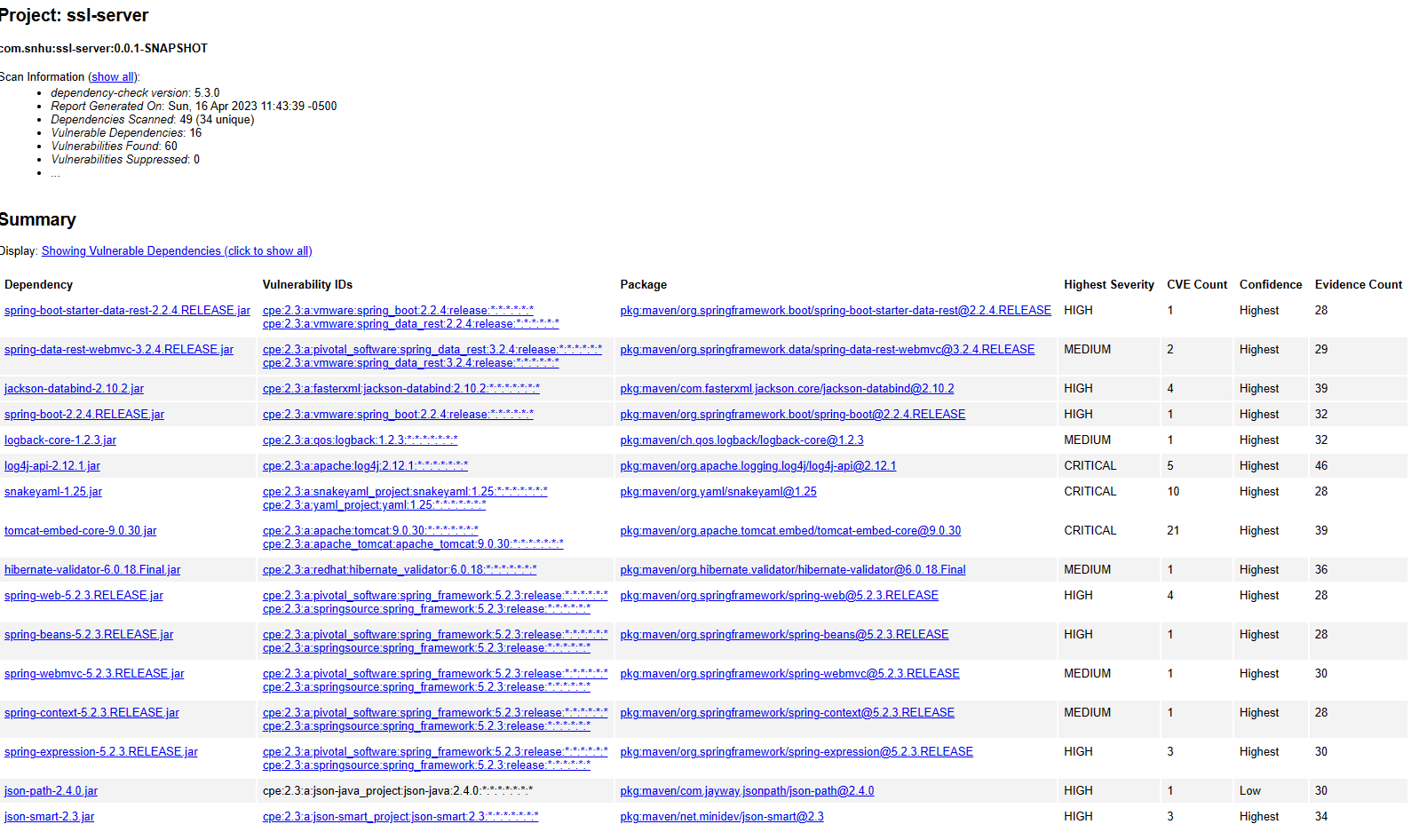
## Secondary Testing

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

Code execution without errors:

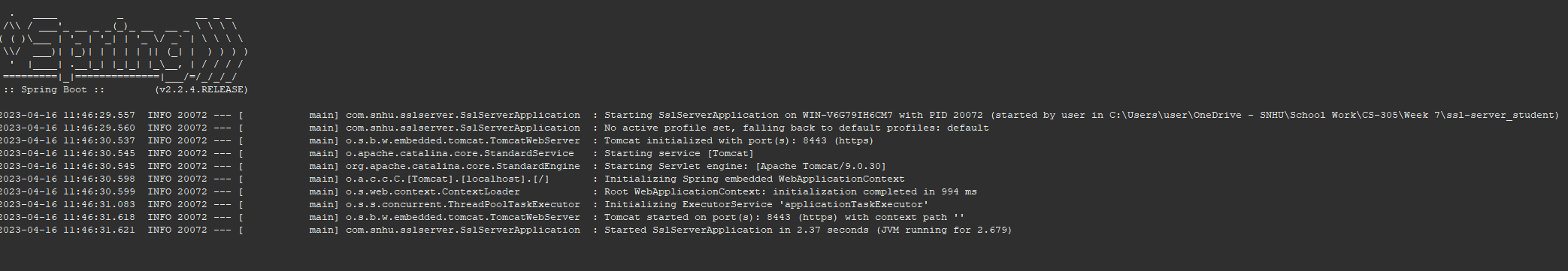


Dependency check before refactor: 

Dependency check after refactor, same scan as before without suppression: 

## Functional Testing

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



## Summary

The code has been refactored to comply with security testing protocols and address potential vulnerabilities. The refactored code includes the following security enhancements:

* SSL/TLS protocol has been implemented to secure communication between the client and server. This is achieved by configuring the server to use HTTPS with a self-signed certificate.
* Checksum has been added to ensure the integrity of the data. The checksum ensures that the data has not been tampered with during transmission.
* The code has been checked for vulnerabilities using the OWASP Dependency-Check Maven. This static testing tool identifies potential security vulnerabilities in the software dependencies.

The process for adding layers of security to the software application includes the following steps:

* Identify potential security vulnerabilities in the code by performing a vulnerability assessment using a tool like OWASP Dependency-Check Maven.
* Refactor the code to address the identified vulnerabilities.
* Implement security protocols, such as SSL/TLS, to secure communication between the client and server.
* Implement security measures, such as checksum, to ensure the integrity of the data.
* Use static testing tools like OWASP Dependency-Check Maven to identify potential security vulnerabilities in the software dependencies.

Overall, the refactored code complies with security testing protocols by addressing potential vulnerabilities and implementing security measures to ensure the integrity and confidentiality of the data.

## Industry Standard Best Practices

HTTPS was implemented to encrypt communication between the client and server, and SSL/TLS was used to provide secure key exchange and prevent man-in-the-middle attacks.

The value of applying industry standard best practices for secure coding to the company's overall well-being cannot be overstated. By implementing these best practices, the company can reduce the likelihood of security breaches, data loss, and other security incidents that could have serious consequences such as financial loss, loss of reputation, and legal liabilities.

Furthermore, following industry standard best practices for secure coding can help the company comply with relevant regulations and standards such as the General Data Protection Regulation (GDPR), the Health Insurance Portability and Accountability Act (HIPAA), and the Payment Card Industry Data Security Standard (PCI DSS), which can help increase customer trust and confidence in the company's services.