****

# 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** | **4/17/25** | **Thomas Davis** | **Code Refactor** |

## 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

Thomas Davis

## Algorithm Cipher

To best address Artemis Financial’s need for secure communications I recommend AES-256 as the encryption algorithm cipher that should be adopted for their system. AES-256, or Advanced Encryption Standard, encrypts data using a symmetric 256-bit key. AES is utilized throughout the world and by the United States government to encrypt sensitive data. AES is a block cipher that encrypts data in blocks of 128 bits each (GeeksforGeeks, 2024). Using multiple rounds of encryption, large data is broken down into the 128-bit blocks. The National Institute of Standards and Technology, NIST, approved AES as the first publicly available cipher to protect confidential information (Dworkin et al., 2001).

The hash functions would be handled by a hashing algorithm such as SHA-256. SHA-256 ensures that data remains intact and unmodified by producing a fixed length hash. Hashes produced by SHA-256 are collision resistant meaning that no two files will produce the same hash, this also allows even small changes to the data to produce a different hash (Madan, 2023).

In cryptography random keys are used to add another level of security to a cipher. Using random numbers creates an unpredictable, random key (Cryptomathic, 2019). In a symmetrical encryption requires both the sender and the receiver to have access to the same private key. Asymmetrical encryption creates a private key that is held by the sender and a public key that is shared with the receiver. Symmetrical encryption is typically used to encrypt large amounts of data very quickly, but sharing that data is more difficult since each party needs to have the key. Asymmetrical encryption makes sharing data easier through the Private-Public key method, but encrypting data takes longer. While these two methods of encryption have their own pros and cons, they can be used together (Yackel, 2020). Since symmetrical encryption is faster, it would be used to encrypt most of the data. When that data needs to be shared, the private key for the symmetrical encryption can be sent to the receiver using asymmetrical encryption. This allows the private key to be sent securely and ensures that only the intended recipient gains access to the private key.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen with white text

AI-generated content may be incorrect.

## Deploy Cipher

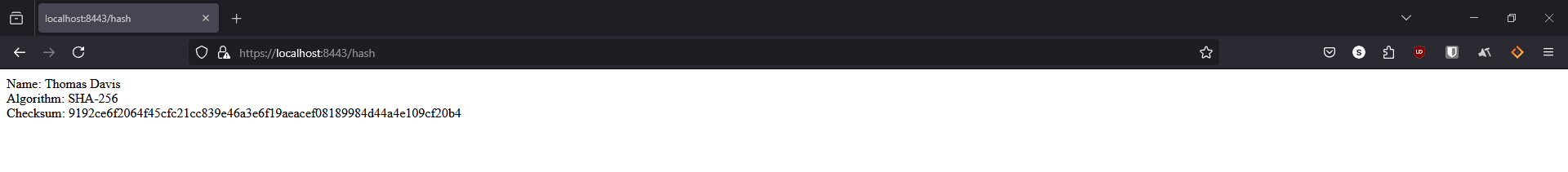
Insert a screenshot below of the checksum verification.

A number on a white background

AI-generated content may be incorrect.

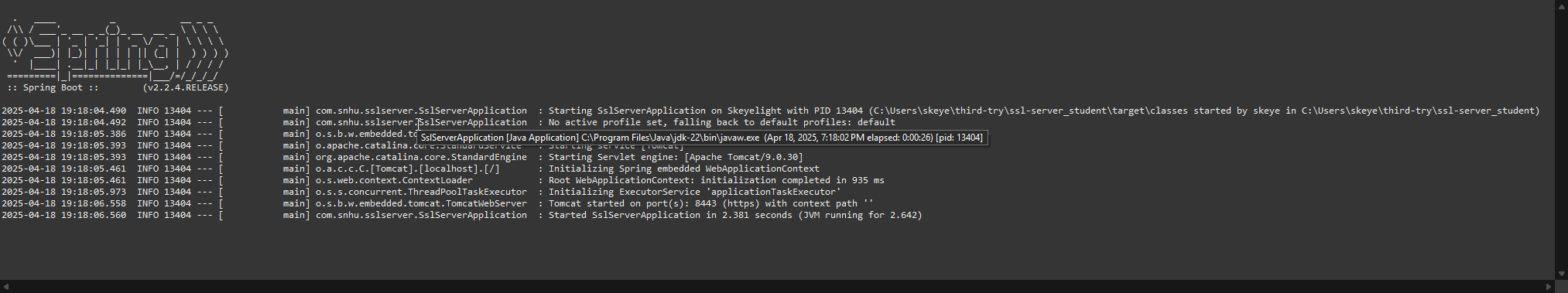
## Secure Communications

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.



Before Changes

A screenshot of a computer error

AI-generated content may be incorrect.

After Changes

A screenshot of a computer error

AI-generated content may be incorrect.

## Functional Testing

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

A screenshot of a computer program

AI-generated content may be incorrect.

## Summary

In my refactored code, I added the SHA-256 hash function to protect the data being transmitted. This prevents data from being viewable by unauthorized parties. By generating a checksum, we can then ensure that the data has not been modified, this can give further confidence that the data was secure. By using encapsulation, the cryptography can be kept isolated from the rest of the program. By using a RESTful API we can keep sensitive data hidden and transmitted securely. By creating a certificate, we can utilize HTTPS and encrypt any data that is being transferred between the client and the server.

## Industry Standard Best Practices

I applied industry best practices by expanding upon the existing SpringBoot tools that were already present in the program. Utilizing SHA-256 in cryptography allows the application to provide proof of data integrity and avoiding collisions. Incorporating error handling through use of NoSuchAlgorithmException to preserve application performance and stability. I used dependency checks to monitor known vulnerabilities that may be present in the application. Using industry standard best practices helps you ensure that your application adheres to any regulations they may be subject to. Following these practices keeps your application secure and easy to maintain.

References

Cryptomathic. (2019, April 23). *The Use of Random Number Generators in Cryptography*. Cryptomathic.com; Cryptomathic. https://www.cryptomathic.com/blog/the-role-of-random-number-generators-in-cryptography

Dworkin, M. J., Barker, E. B., Nechvatal, J. R., Foti, J., Bassham, L. E., E. Roback, & Dray, J. F. (2001, November 26). *Advanced Encryption Standard (AES)*. Federal Inf. Process. Stds. (NIST FIPS) - 197. https://www.nist.gov/publications/advanced-encryption-standard-aes

GeeksforGeeks. (2024, July 16). *Advanced Encryption Standard (AES)*. GeeksforGeeks. https://www.geeksforgeeks.org/advanced-encryption-standard-aes/

Madan. (2023, August 30). *A Deep Dive into SHA-256: Working Principles and Applications*. Medium. https://medium.com/@madan\_nv/a-deep-dive-into-sha-256-working-principles-and-applications-a38cccc390d4

Yackel, R. (2020, June 17). *When to Use Symmetric Encryption vs Asymmetric Encryption*. Keyfactor. https://www.keyfactor.com/blog/symmetric-vs-asymmetric-encryption/