****

# 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** | **10/22/24** | **August Moews** |  |

## 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

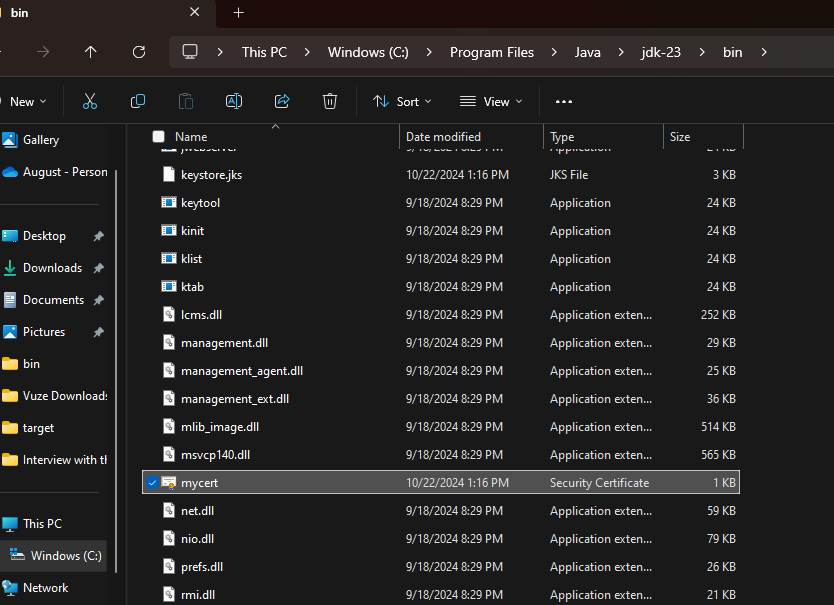
August Moews

## Algorithm Cipher

For Artemis Financials' application, I recommend using SHA-256 (Secure Hash Algorithm 256-bit) as the cryptographic hash algorithm to enhance security. SHA-256 is part of the SHA-2 family of hash functions, developed by the National Security Agency (NSA) and is widely recognized as one of the most secure and reliable algorithms available today. SHA-256 operates by taking an input message of any length and producing a fixed 256-bit (32-byte) hash value. This property makes it ideal for verifying data integrity. One of the key strengths of SHA-256 lies in its resistance to collision attacks, where two different inputs produce the same hash output. As of now, SHA-256 is considered computationally infeasible to break, making it a strong candidate for secure applications that require data integrity checks (NIST, 2012). SHA-256 is commonly used in digital signatures, certificate generation, and blockchain technologies due to its robust security profile. SHA-256 uses random number generation to ensure each hash value is unique. SHA-256 is not reversible – it is a one-way function. This means that, while you can generate a hash from an input, you cannot retrieve the original input from the hash. (Katz & Lindell, 2020) Introduced in 2001, SHA-256 has become a widely used and trusted hashing algorithm and is an excellent choice for Artemis Financial.

## Certificate Generation

Insert a screenshot below of the CER file.



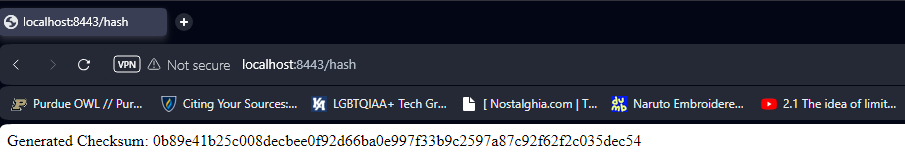
## Deploy Cipher

Insert a screenshot below of the checksum verification.



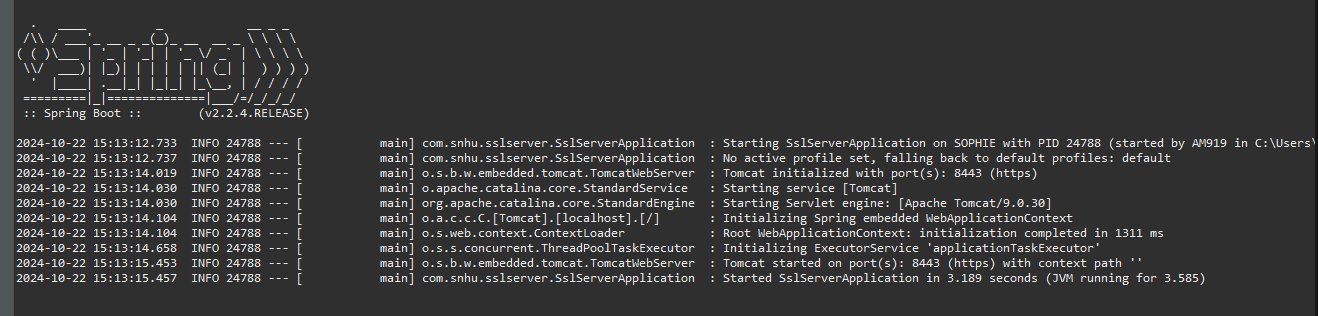
## Secure Communications

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



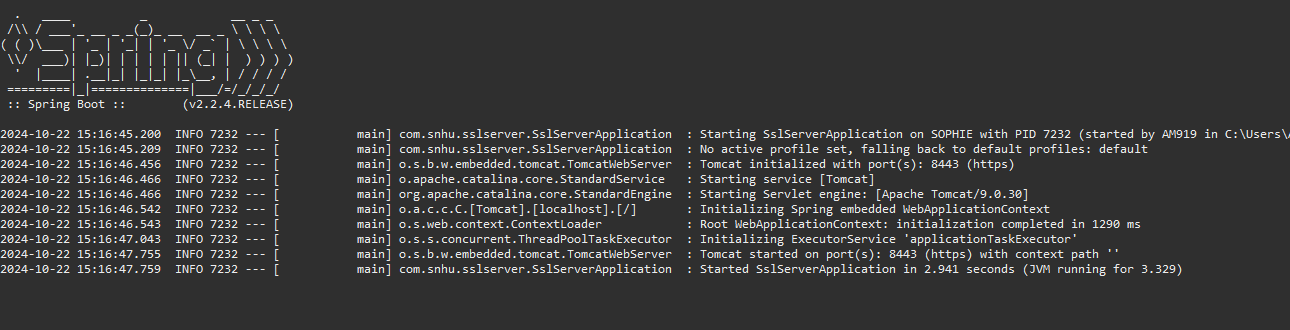
## Secondary Testing

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



## Functional Testing

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



## Summary

In this project, I refactored Artemis Financials' software to enhance security and meet industry-standard protocols. Key improvements included implementing SHA-256 hashing for data integrity verification and securing communications via HTTPS with a self-signed certificate. The application was configured to operate on port 8443 using SSL/TLS, ensuring encrypted communications. By adding checksum verification and securing data transmissions, the application now offers enhanced protection for sensitive financial information.

To ensure software adhered to secure coding standards, I updated critical dependencies such as Jackson Core and Jackson Databind to their latest versions; addressing known vulnerabilities. Additionally, a suppression file was created to handle false positives in OWASP dependency-check reports. This allowed the application to remain free from unnecessary vulnerability warnings, while a thorough review ensured that no new vulnerabilities were introduced during the refactoring process.

The refactoring focused on three primary areas: cryptographic protections, secure communications, and dependency management. By adding multiple layers of security, including checksum validation and encryption, the application is now more resistant to data breaches and attacks.

## Industry Standard Best Practices

In this project, I applied industry-standard best practices for secure coding to mitigate known security vulnerabilities and ensure the ongoing safety of the software application. Key practices included using SHA-256 for data verification, enforcing secure communication through HTTPS, and regularly updating third-party dependencies like Jackson to patch known vulnerabilities. I also employed the OWASP dependency-check tool to identify and suppress false positives, ensuring that only legitimate vulnerabilities were addressed. These practices are essential for preventing potential exploits and maintaining the software’s integrity in an evolving security landscape.

Applying secure coding practices not only strengthens the application’s defenses against attacks but also enhances the company’s overall well-being by protecting sensitive financial data. By adhering to industry standards, such as using modern encryption methods and regularly updating libraries, the application is better equipped to handle emerging threats. This proactive approach not only reduces the risk of data breaches but also builds trust with clients by demonstrating a strong commitment to safeguarding their information.

REFERENCES:

Katz, J., & Lindell, Y. (2020). *Introduction to Modern Cryptography* (3rd ed.). CRC Press.

National Institute of Standards and Technology (NIST). (2012). *SHA-256 Secure Hash Algorithm*. <https://csrc.nist.gov/publications/detail/fips/180-4/final>