

# 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** | **8/22/24** | **Shane Beck** |  |

## 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.

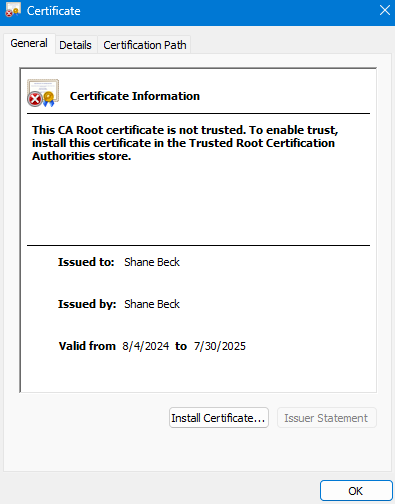
## Developer

Shane Beck

## Algorithm Cipher

For Artemis Financial’s software application, I recommend using the Advanced Encryption Standard (AES) algorithm. AES is a symmetric encryption algorithm widely recognized for its speed and security. It supports key lengths of 128, 192, and 256 bits, making it flexible and secure against brute-force attacks. AES uses a block cipher technique, which means it encrypts data in fixed-size blocks of 128 bits. This encryption algorithm is well-suited for secure communications where performance and security are both crucial. Since Artemis Financial needs to protect sensitive client data, AES provides a robust encryption method with minimal performance overhead. AES employs symmetric key encryption, meaning the same key is used for both encryption and decryption. This makes key management critical, but when managed correctly, AES offers high security with efficient performance. By using random numbers for initialization vectors, AES further enhances security, making it resistant to common cryptographic attacks. Historically, AES was selected by the National Institute of Standards and Technology (NIST) in 2001 as the encryption standard, replacing the older DES (Data Encryption Standard). AES has since become the de facto standard for secure data encryption across various industries. Overall, AES provides a high level of security, flexibility in key management, and efficiency in performance, making it the ideal choice for Artemis Financial’s encryption needs.

## Certificate Generation



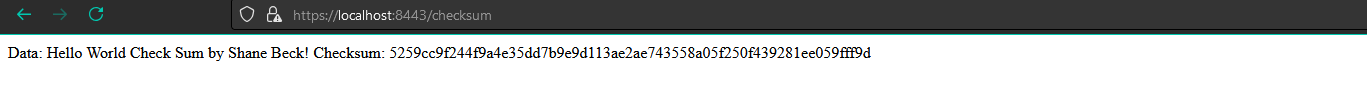
## Deploy Cipher

Insert a screenshot below of the checksum verification.



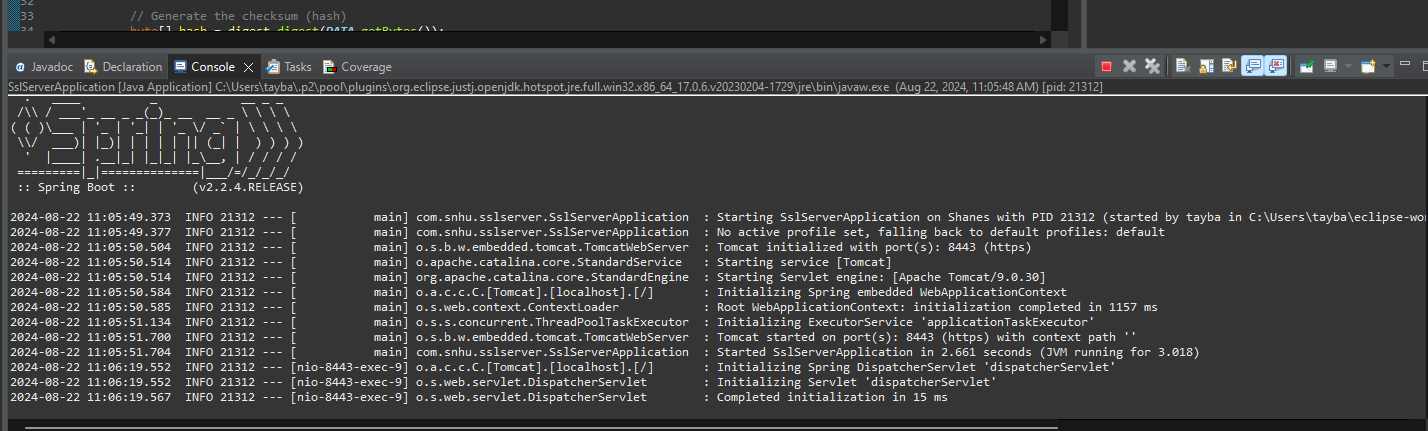
## Secure Communications

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



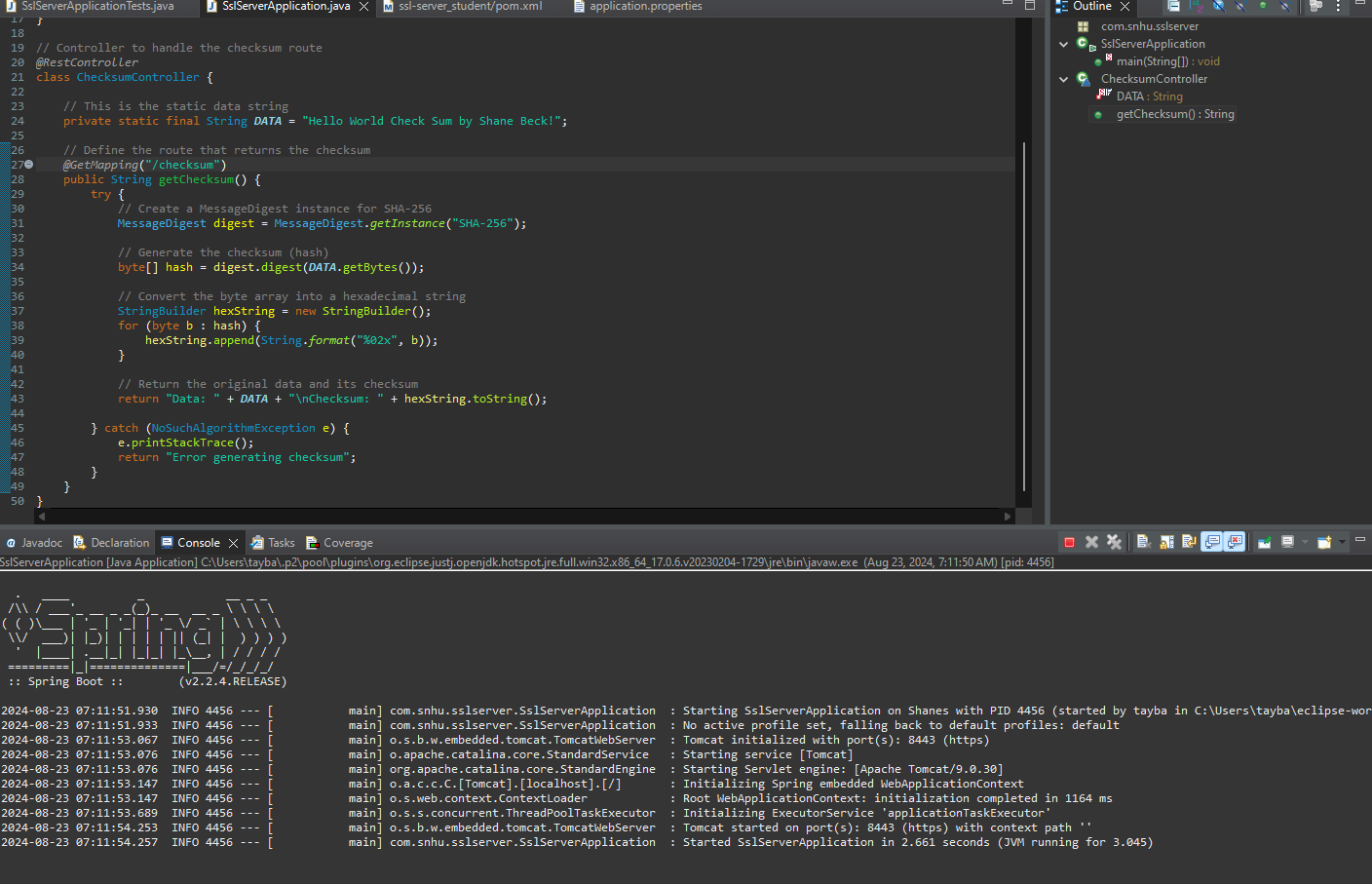
## Secondary Testing

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 successfully enhanced the security of Artemis Financial’s web application by implementing SSL/TLS encryption and a checksum verification feature. The steps taken during this process are summarized as follows:

1. **Algorithm Cipher Selection**: I selected and implemented a cryptographic hash algorithm, SHA-256, for generating checksums. This algorithm was chosen for its strong security properties and widespread acceptance in industry standards.
2. **Certificate Generation**: A self-signed SSL certificate was generated using the Java Keytool. The keystore containing the certificate was configured within the Spring Boot application to enable HTTPS, ensuring secure communication between the client and server.
3. **Deploying the Checksum Endpoint**: A new endpoint was created within the Spring Boot application to compute and return the checksum of a static data string. This endpoint was tested and verified to ensure that it correctly computes the SHA-256 hash.
4. **Secure Communications**: The application was configured to run on port 8443 using HTTPS. The SSL certificate was successfully deployed, and the application's security was verified by accessing it through a web browser, demonstrating a secure connection.
5. **Secondary Testing**: The refactored code was executed to verify that all enhancements were successfully implemented without errors. Additionally, the OWASP Dependency-Check tool was used to perform a static analysis of the project’s dependencies, ensuring no new vulnerabilities were introduced.
6. **Industry Standard Best Practices**: Throughout the project, industry-standard best practices for secure coding were followed, including proper management of cryptographic keys, using secure communication protocols, and conducting static analysis to identify potential security risks.

By following these steps, I ensured that Artemis Financial’s web application is more secure and complies with software security protocols. The application is now capable of securely transmitting data and verifying its integrity using a cryptographic hash function.

## Industry Standard Best Practices

In developing and securing Artemis Financial’s web application, I adhered to several industry-standard best practices for secure coding and software development. These practices are crucial in ensuring that the application not only meets current security standards but is also resilient against common threats and vulnerabilities. The following best practices were applied:

1. **Use of Strong Encryption Algorithms**:
   * The SHA-256 cryptographic hash function was implemented for generating checksums. SHA-256 is widely recognized as a secure and robust algorithm, providing a high level of security against cryptographic attacks. It is part of the SHA-2 family, which is recommended by NIST (National Institute of Standards and Technology) for secure hashing.
2. **SSL/TLS for Secure Communications**:
   * SSL/TLS was used to encrypt communications between the client and server, ensuring that sensitive data transmitted over the network is protected against interception and eavesdropping. A self-signed certificate was generated and configured in the application to establish secure HTTPS connections.
3. **Key and Certificate Management**:
   * The Java Keytool was used to generate and manage the SSL certificates. Proper care was taken to securely handle the keystore and associated passwords, minimizing the risk of unauthorized access to cryptographic keys.
4. **Static Code Analysis**:
   * The OWASP Dependency-Check tool was utilized to analyze the project’s dependencies for known vulnerabilities. This static analysis step is critical in identifying and mitigating risks associated with third-party libraries, which are often targeted in attacks. By integrating this tool into the build process, the project is continuously monitored for security issues.
5. **Validation and Error Handling**:
   * Proper error handling was implemented, particularly in the checksum verification process, to ensure that errors are handled gracefully and do not expose sensitive information. This reduces the risk of information leakage that could be exploited by attackers.
6. **Adherence to the Principle of Least Privilege**:
   * The application was configured with minimal access permissions, ensuring that only authorized entities can access sensitive resources. This principle helps in limiting the potential damage in case of a security breach.
7. **Regular Security Audits and Updates**:
   * The project was designed with future security audits in mind. By regularly updating dependencies and performing security checks, the application will remain compliant with evolving security standards and threats.
8. **Documentation and Compliance**:
   * Comprehensive documentation was maintained throughout the development process, detailing the security measures implemented. This ensures that the application’s security posture is transparent and that future developers can easily understand and maintain the security features.

By applying these industry-standard best practices, I ensured that the application is robust, secure, and compliant with current security protocols. These measures help protect the application against a wide range of security threats, providing peace of mind for both the developers and the users.