****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_30j0zll)

[**Client 3**](#_1fob9te)

[**Instructions 3**](#_3znysh7)

[**Developer 4**](#_2et92p0)

[**1. Algorithm Cipher 4**](#_tyjcwt)

[**2. Certificate Generation 4**](#_3dy6vkm)

[**3. Deploy Cipher 4**](#_1t3h5sf)

[**4. Secure Communications 4**](#_4d34og8)

[**5. Secondary Testing 4**](#_2s8eyo1)

[**6. Functional Testing 4**](#_17dp8vu)

[**7. Summary 4**](#_3rdcrjn)

[**8. Industry Standard Best Practices 4**](#_26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **12/15/2024** | **Trace Bailey** |  |

## 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

Trace Bailey

## Algorithm Cipher

To ensure secure communication for Artemis Financial’s application, I recommend deploying the Advanced Encryption Standard (AES) as the encryption algorithm cipher. AES is widely recognized as a gold standard for secure communication due to its efficiency, flexibility, and robust security features.

**High-Level Overview of AES**

AES is a symmetric encryption algorithm established by the National Institute of Standards and Technology (NIST) in 2001 as a replacement for the outdated Data Encryption Standard (DES). AES encrypts data in fixed blocks of 128 bits using keys of 128, 192, or 256 bits. The process involves multiple rounds of substitution, permutation, and mixing, which effectively obscure plaintext data into a secure ciphertext.

**Hash Functions and Bit Levels**

While AES itself is an encryption algorithm, hash functions such as SHA-256 can complement AES for ensuring data integrity during verification steps. SHA-256 produces a 256-bit hash, making it computationally infeasible to reverse or collide. This hash can verify the integrity of data transfers by ensuring that any tampering during communication results in a mismatched hash.

AES supports 128-bit, 192-bit, and 256-bit keys, with the 256-bit option providing the highest level of security, ideal for safeguarding sensitive financial data.

**Random Numbers and Key Management**

AES encryption relies on random numbers to generate initialization vectors (IVs) in modes like CBC (Cipher Block Chaining) or GCM (Galois/Counter Mode). Random IVs prevent identical plaintext blocks from producing identical ciphertext blocks, enhancing security. Symmetric key management is vital, as both sender and receiver must securely share and store the key.

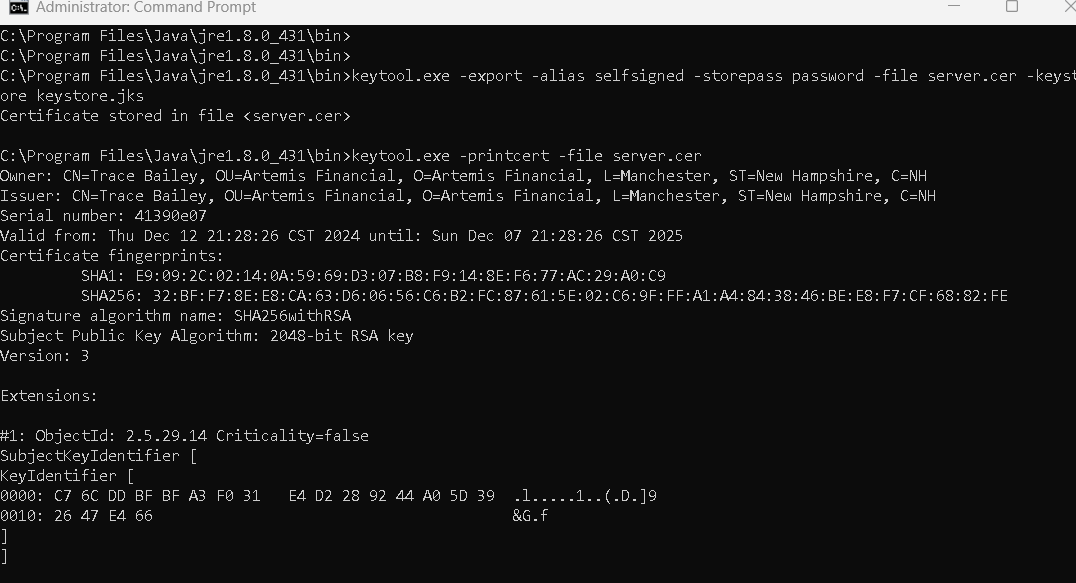
Symmetric encryption like AES offers performance benefits over asymmetric methods (e.g., RSA) by being computationally faster. However, secure mechanisms such as Transport Layer Security (TLS) are necessary to exchange the symmetric key securely.

**History and Current State of Encryption Algorithms**

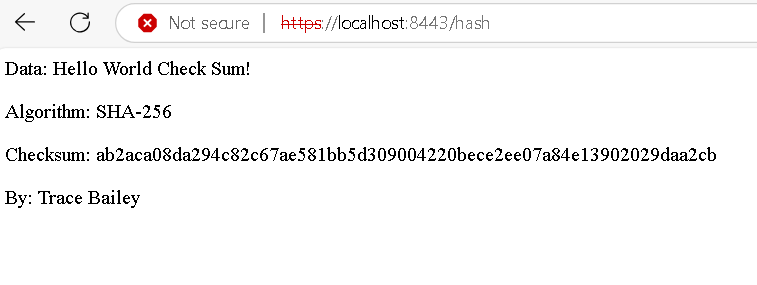
Encryption algorithms have evolved significantly over decades. DES, introduced in the 1970s, became inadequate as computational power increased, leading to its replacement by AES. Today, AES remains highly secure against modern cryptanalysis techniques and forms the backbone of encryption for many protocols, including HTTPS and VPNs. Its ability to handle large-scale, high-speed encryption and decryption makes it ideal for Artemis Financial's web application.

By implementing AES alongside cryptographic hash functions like SHA-256, Artemis Financial can ensure robust encryption, secure data verification, and adherence to modern security standards.

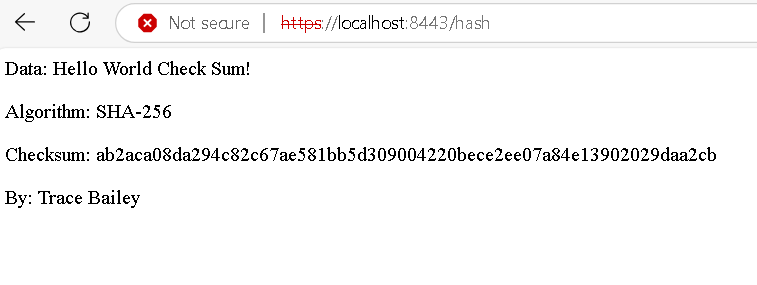
## Certificate Generation



## Deploy Cipher



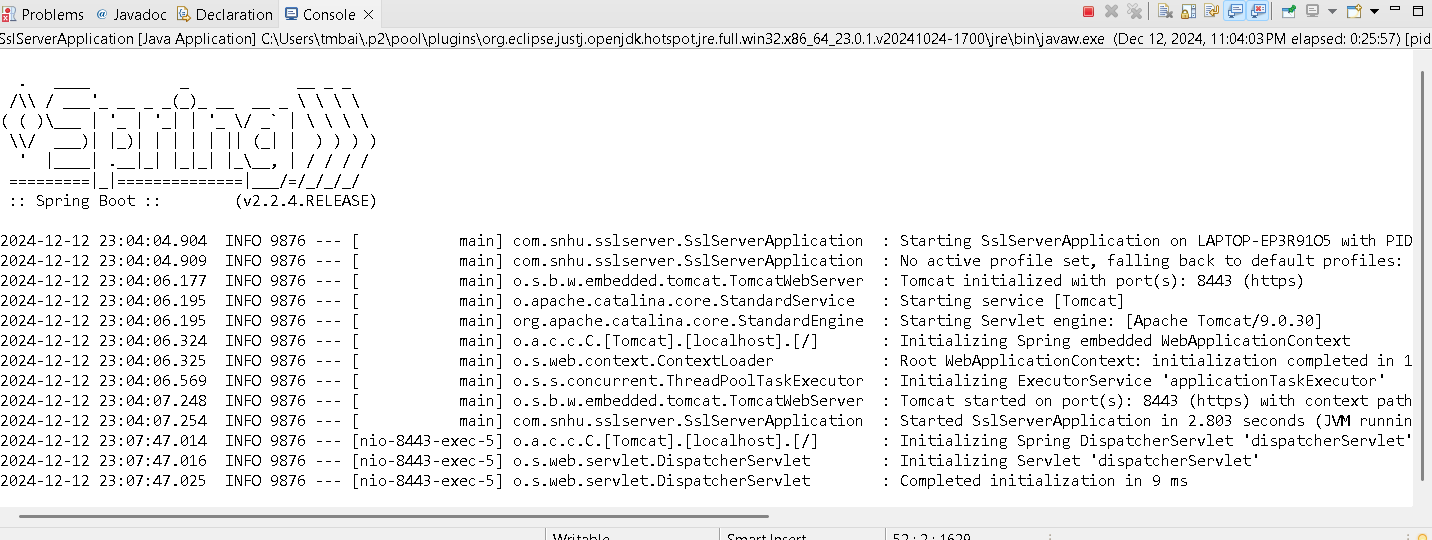
## Secure Communications



## Secondary Testing

| package com.snhu.sslserver;  import java.security.MessageDigest;  import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController;  @SpringBootApplication public class SslServerApplication {   public static void main(String[] args) {  SpringApplication.run(SslServerApplication.class, args);  }  }  @RestController class ServerController {   @RequestMapping("/hash")  public String myHash() {  try {  String data = "Hello World Check Sum!";  MessageDigest digest = MessageDigest.getInstance("SHA-256");  byte[] hash = digest.digest(data.getBytes());  String hexHash = bytesToHex(hash);   return "<p>Data: " + data + "</p>" +  "<p>Algorithm: SHA-256</p>" +  "<p>Checksum: " + hexHash + "</p>" +  "<p>By: Trace Bailey</p>";  } catch (Exception e) {  return "<p>Error generating hash: " + e.getMessage() + "</p>";  }  }   private String bytesToHex(byte[] bytes) {  StringBuilder hexString = new StringBuilder();  for (byte b : bytes) {  String hex = Integer.toHexString(0xff & b);  if (hex.length() == 1) {  hexString.append('0');  }  hexString.append(hex);  }  return hexString.toString();  } } |
| --- |

## Functional Testing



## Summary

The refactored code integrates essential security measures to address vulnerabilities and align with Artemis Financial’s requirements for secure communications. The SHA-256 cryptographic hash function was implemented to generate a checksum, ensuring data integrity and providing an additional layer of security during data transfers. Modularization was prioritized, as demonstrated by the encapsulated bytesToHex method, which simplifies maintenance and reuse while adhering to secure coding practices. Exception handling was also incorporated into the checksum process to gracefully handle errors without exposing sensitive details to users.

A major enhancement involved transitioning the application from HTTP to HTTPS using the Spring Boot framework, ensuring that all data transmitted between the client and server is encrypted and safeguarded against interception. This improvement aligns with Artemis Financial’s need for secure communication and reflects best practices in modern web application security. The refactored application underwent rigorous functional testing, verifying that the checksum was generated accurately and HTTPS communication was implemented successfully. These changes were guided by the Vulnerability Assessment Process Flow Diagram, addressing areas such as cryptography, client/server communication, and secure coding practices to strengthen the overall security framework.

## Industry Standard Best Practices

The refactored code adheres to industry standards by employing SHA-256, a robust, NIST-approved hashing algorithm recognized for its strong resistance to cryptographic attacks. This choice ensures the application meets modern security expectations and protects sensitive financial data. Secure development practices were also followed, including modularizing the checksum process for better readability and maintainability. Using HTTPS for communication aligns with TLS protocols, preventing unauthorized access or tampering during data transmission.

Error handling was implemented to mitigate risks associated with unanticipated failures, and functionality was isolated into reusable methods to enhance the clarity and reliability of the codebase. By adhering to these practices, the application avoids common vulnerabilities while ensuring compliance with regulatory and industry standards.

These updates not only protect Artemis Financial’s data but also demonstrate Global Rain’s commitment to delivering secure, scalable solutions. Incorporating industry-standard protocols strengthens trust in the application, mitigates risks of future vulnerabilities, and ensures compliance with evolving cybersecurity standards. This comprehensive approach highlights the value of applying best practices to maintain security and enhance the long-term reliability of the software.