****

# 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** | **October 6, 2025** | **Hassan Lindsay** | **Module 7 Edits** |

## 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

[Insert your name here.]

## Algorithm Cipher

## Artemis Financial requires strong encryption to protect client information both in transit and at rest. For this project, the Advanced Encryption Standard (AES-256) and the Secure Hash Algorithm (SHA-256) were implemented. AES-256 provides high-performance symmetric encryption widely approved by NIST for government-level data protection. SHA-256 offers a one-way, collision-resistant hashing method that verifies data integrity without exposing the original content.

## Within the /checksum endpoint, the application uses the Java MessageDigest library to generate a SHA-256 hash for the string *“Hello World Check Sum!”*, confirming that sensitive data can be validated securely. Together, AES and SHA-256 form a layered defense that ensures confidentiality, integrity, and authenticity of financial transactions.

A screenshot of a computer program

AI-generated content may be incorrect.

## Certificate Generation

A self-signed digital certificate was generated using the Java Keytool utility to establish SSL/TLS encryption. The following command created the keystore and exported the certificate file:

keytool -genkeypair -alias selfsigned -keyalg RSA -keysize 2048 -validity 365 -keystore keystore.jks -storepass changeme

keytool -export -alias selfsigned -storepass changeme -file server.cer -keystore keystore.jks

This process produced the *keystore.jks* and *server.cer* files, which authenticate the server and enable encrypted HTTPS sessions.

**Citation:** Oracle. (n.d.). *Keytool – Key and Certificate Management Tool.* Oracle Docs.

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

To ensure file integrity, a checksum was created for the exported certificate using the command

shasum -a 256 server.cer.

The resulting SHA-256 hash verified that the certificate had not been altered since generation.

This step demonstrates proper deployment verification and confirms that the encryption artifact used by the web server is authentic and untampered.

A screenshot of a computer

AI-generated content may be incorrect.

## Secure Communications

The Spring Boot application was configured to run exclusively over HTTPS on port 8443 using the generated keystore.

The application.properties file defines the SSL parameters:

server.port=8443

server.ssl.key-store=keystore.jks

server.ssl.key-store-password=changeme

server.ssl.key-store-type=JKS

When executed, Tomcat initialized successfully with (https) in the startup log, and the browser displayed the secure route https://localhost:8443/checksum.

This confirms that all communication between client and server is encrypted using SSL/TLS.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

After securing the application, an additional OWASP Dependency-Check scan was performed to assess remaining vulnerabilities.

The secondary test completed with a *BUILD SUCCESS* result and showed a reduction in detected vulnerabilities compared with the initial scan.

This validated that dependency updates and secure-code modifications effectively mitigated high-risk exposures while maintaining build integrity.

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

Functional testing confirmed that the refactored code executed successfully with no runtime errors.

IntelliJ logs displayed *“Tomcat started on port(s): 8443 (https)”*, and the /checksum endpoint returned the expected text and SHA-256 hash.

These results verify that implementing SSL/TLS encryption and refactoring the codebase did not impact core functionality or performance.

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

## Summary

Following a series of secure-coding improvements, Artemis Financials’ web service now demonstrates a mature and stable security posture.

The project achieved secure communication through SSL/TLS, validated data integrity via SHA-256 hashing, and confirmed certificate authenticity through checksum verification.

Secondary and functional testing proved that vulnerabilities were reduced and that the application continues to perform as expected.

Overall, this initiative strengthened confidentiality, integrity, and availability, the core principles of cybersecurity, and exemplified how iterative testing and remediation produce resilient software in financial environments.

## Industry Standard Best Practices

The project aligns with multiple industry frameworks and best practices, including the OWASP Top 10, the NIST Cybersecurity Framework, and the Secure Software Development Life Cycle (SSDLC).

Key principles applied include:

• Encrypting data in transit with TLS 1.2+

• Using strong cryptographic algorithms (AES-256, SHA-256)

• Performing regular dependency-vulnerability scans

• Implementing input validation and exception handling

• Applying the principle of least privilege

• Maintaining certificate lifecycle management through Java KeyStore

Adhering to these standards ensures Artemis Financials’ software remains compliant, trustworthy, and secure against evolving threats.

**References:**

National Institute of Standards and Technology. (2018). *Advanced Encryption Standard (AES) – FIPS PUB 197.* U.S. Department of Commerce. <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>

National Institute of Standards and Technology. (2015). *Secure Hash Standard (SHS) – FIPS PUB 180-4.* U.S. Department of Commerce. <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>

Oracle. (n.d.). *Keytool – Key and Certificate Management Tool.* Oracle Documentation. <https://docs.oracle.com/en/java/javase/17/docs/specs/man/keytool.html>

OWASP Foundation. (2021). *OWASP Top Ten Web Application Security Risks – 2021.* <https://owasp.org/www-project-top-ten/>

National Institute of Standards and Technology. (2018). *Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1.* <https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf>