GLOBALRAIN

Practices for Secure Software Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	2/21/2026	AARON VOGEL	

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
AARON VOGEL

1. Algorithm Cipher

Artemis Financial needs a dependable way to protect financial and client data. AES-256 is a good choice because it provides strong security without slowing down systems. The 256-bit key size boosts protection and lowers the chance of brute-force attacks.

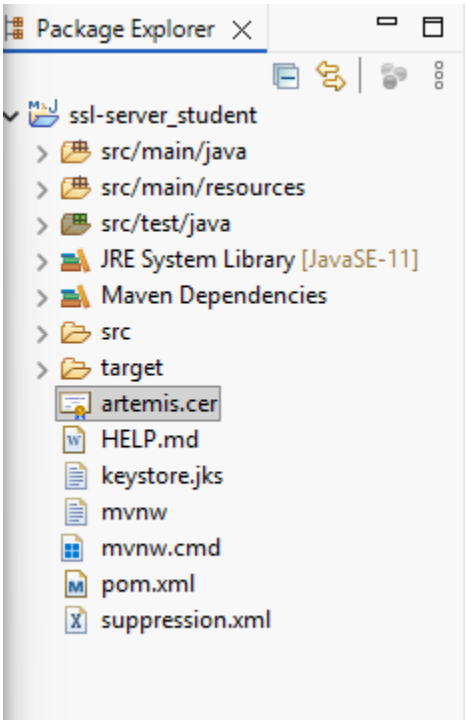
AES uses the same secret key to encrypt and decrypt data, so managing that key carefully is crucial to keep information secure.

Since AES uses the same key for encryption and decryption, keeping that key safe is very important. Encryption also depends on good random numbers, like initialization vectors (IVs), to make sure the encrypted data stays unpredictable.

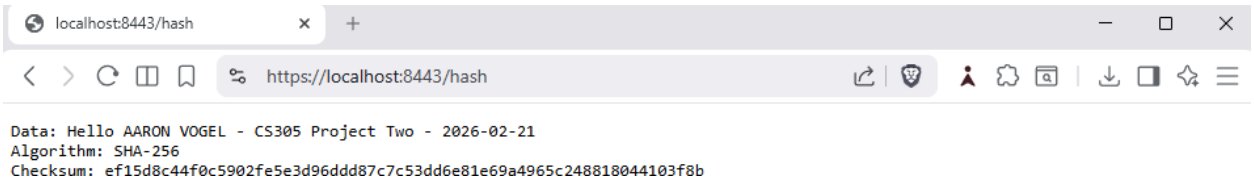
To verify data integrity, the application uses the SHA-256 hashing algorithm. SHA-256 creates a unique 256-bit hash that represents the input data. Unlike encryption, hashing is one-way and helps detect any changes instead of hiding the data.

Older algorithms like DES and 3DES are now outdated because their keys are weaker and computers are more powerful. AES is the current industry standard thanks to its strong security, good performance, and wide use.

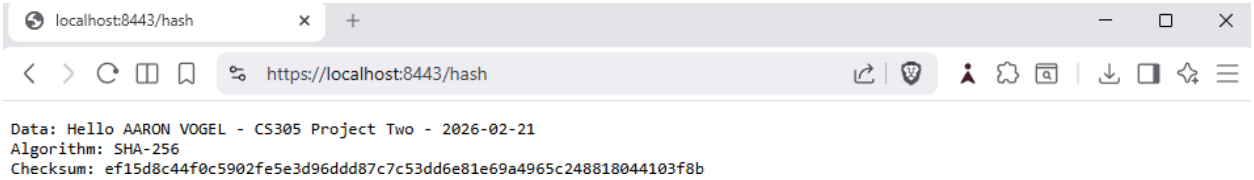
2. Certificate Generation



3. Deploy Cipher



4. Secure Communications



5. Secondary Testing

6. Functional Testing

7. Summary

In this project, we updated the Artemis Financial app to boost security and enable safe communications. I set up HTTPS by configuring SSL and creating a self-signed certificate with Java Keytool. This means data sent between the client and server is now encrypted.

I added a checksum verification feature using the SHA-256 hashing algorithm. This helps the app check data integrity by creating a unique hash for each data string. If the data changes, the checksum will be different, so tampering can be detected.

I ran security tests with the OWASP Dependency-Check tool. The scan showed that our changes did not add any new vulnerabilities. I also completed functional tests to make sure the app works properly without errors.

Overall, these updates improved transport security, enhanced data integrity verification, and aligned the application with modern secure coding standards.

8. Industry Standard Best Practices

I followed industry best practices during the refactoring process. I chose modern cryptographic algorithms like AES-256 for encryption and SHA-256 for hashing. These are widely trusted and secure when used correctly.
I secured communication by switching the application from HTTP to HTTPS. SSL/TLS protects sensitive financial data from being intercepted during transmission. Correctly setting up certificates makes sure connections are both authenticated and encrypted.
I added static security testing with OWASP Dependency-Check to find any known vulnerable dependencies. This helps me manage risks proactively and keep the software secure over time.
Following these best practices protects client data, lowers organizational risk, and ensures it meets current security standards.