****

# 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** | **4/20/2025** | **Umar Asif** |  |

## 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

Umar Asif

## Algorithm Cipher

For this project, the **SHA-256 (Secure Hash Algorithm 256-bit)** cryptographic hash function was selected and implemented. SHA-256 is part of the SHA-2 family, developed by the **National Security Agency (NSA)** and published by the **National Institute of Standards and Technology (NIST)** in 2001. It is widely recognized for its strong security properties and is a standard for many secure applications today.

#### Overview

SHA-256 is a **one-way cryptographic hash function**, meaning it transforms input data into a fixed-size 256-bit (32-byte) hash value. Unlike encryption algorithms, which are reversible, hash functions like SHA-256 are designed to be **irreversible**—ensuring that original data cannot be retrieved from the hash value. This makes it suitable for data integrity verification, password storage, and digital signatures.

#### Hash Functions and Bit Levels

SHA-256 generates a **256-bit** (64 hexadecimal characters) digest regardless of input size. The size and complexity of the output make it computationally infeasible to reverse the hash or generate collisions (two different inputs producing the same hash). The bit level (256 bits) provides a high degree of security, resisting brute-force and preimage attacks far better than older algorithms like SHA-1 or MD5.

#### Random Numbers, Symmetric vs. Asymmetric Keys

SHA-256 itself is **not an encryption algorithm**—it does not use keys. However, in encryption practices:

* **Symmetric key algorithms** (like AES) use the same key for encryption and decryption. They are fast and suitable for large data.
* **Asymmetric key algorithms** (like RSA) use a public and private key pair, often used for secure key exchange or digital signatures.
* **Hash functions**, like SHA-256, do **not** use keys but may incorporate **random numbers** (nonces, salts) when used in conjunction with other protocols (e.g., hashing passwords).

In secure systems, these elements are often used together—for example, digital signatures use hash functions and asymmetric encryption.

#### History and Current State of Encryption Algorithms

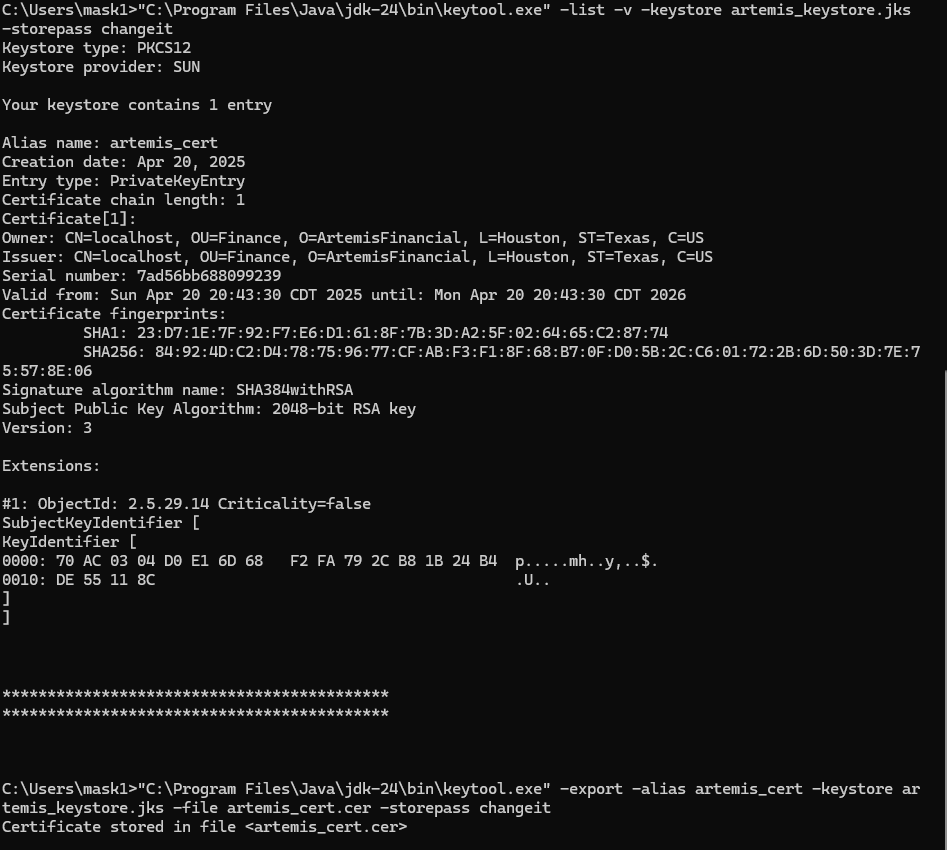
Encryption and hashing algorithms have evolved significantly:

* **Early algorithms**: DES (56-bit key) was used in the 1970s but is now considered insecure due to its small key size.
* **MD5 and SHA-1**: Popular in the 1990s but now deprecated due to collision vulnerabilities.
* **SHA-2 family (SHA-256, SHA-512)**: Currently trusted and widely used in SSL certificates, blockchain technologies, and secure messaging.
* **Modern encryption**: AES (Advanced Encryption Standard) is the current symmetric encryption standard, while RSA and ECC (Elliptic Curve Cryptography) are widely used for asymmetric encryption.

**SHA-256 remains one of the most reliable and widely adopted hashing algorithms** in the world today, supported by industry standards, browsers, and security frameworks.

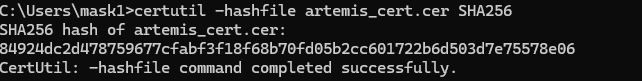
## Certificate Generation

Insert a screenshot below of the CER file.



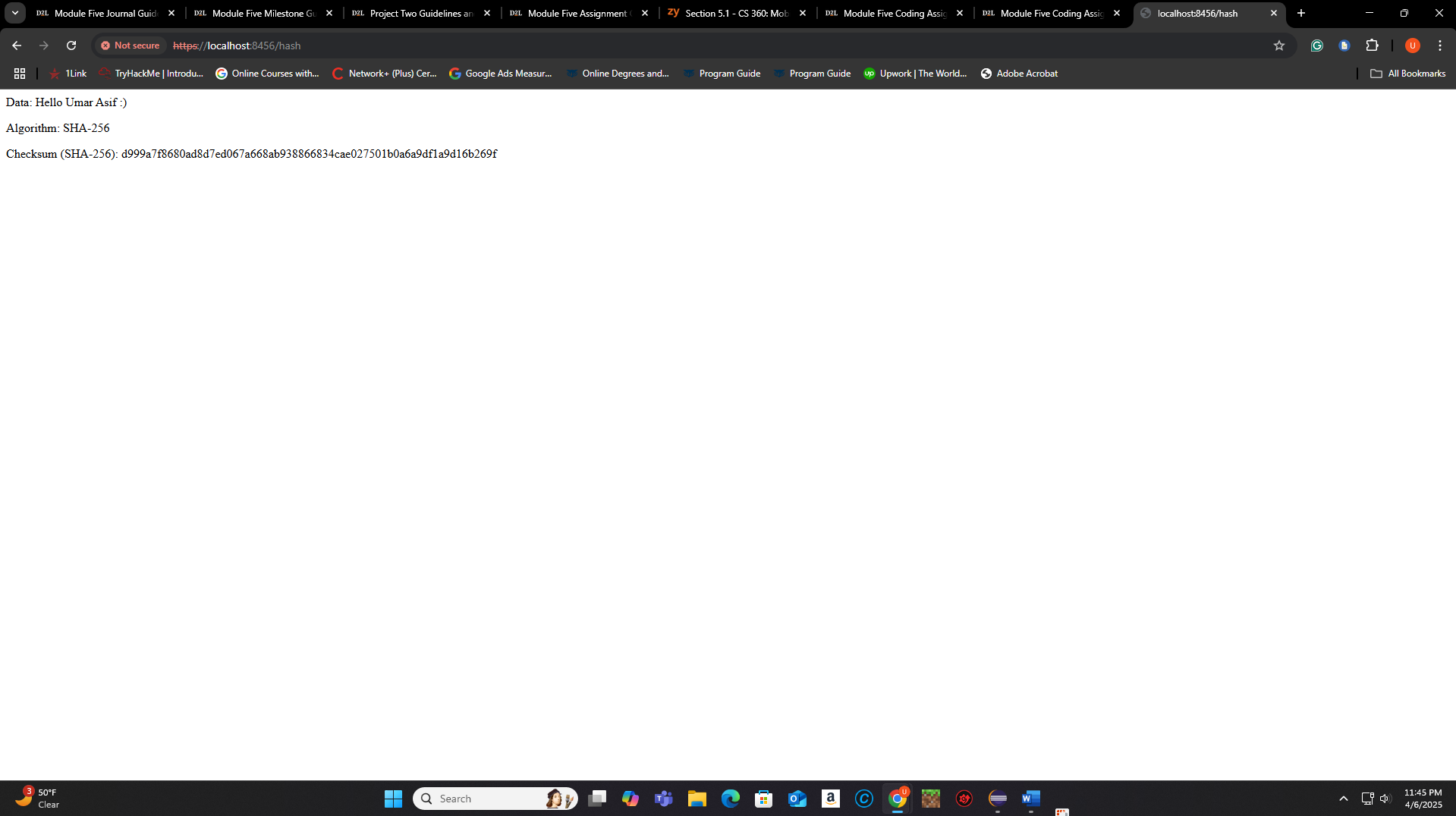
## Deploy Cipher

Insert a screenshot below of the checksum verification.



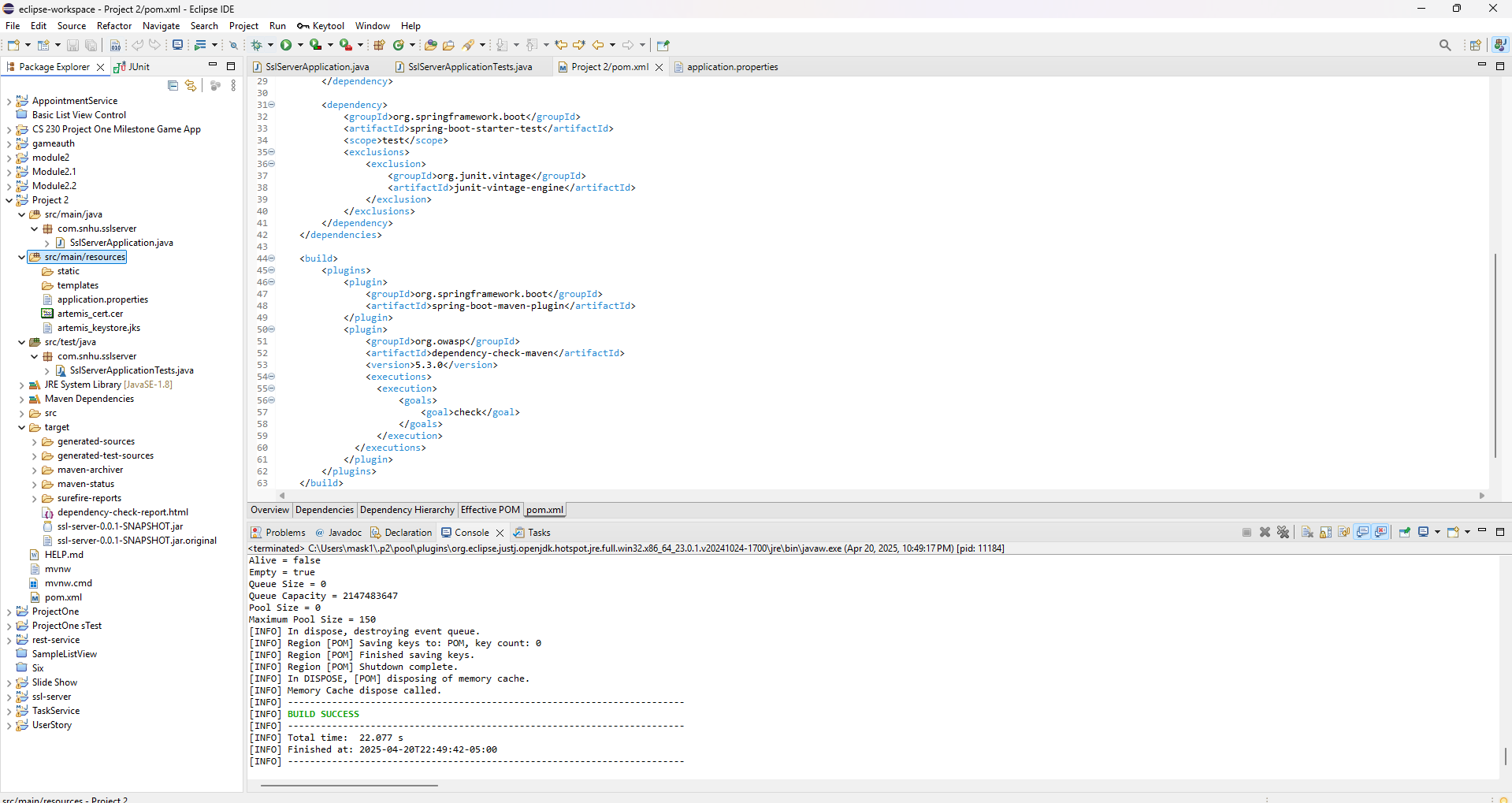
## Secure Communications

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



## Secondary Testing

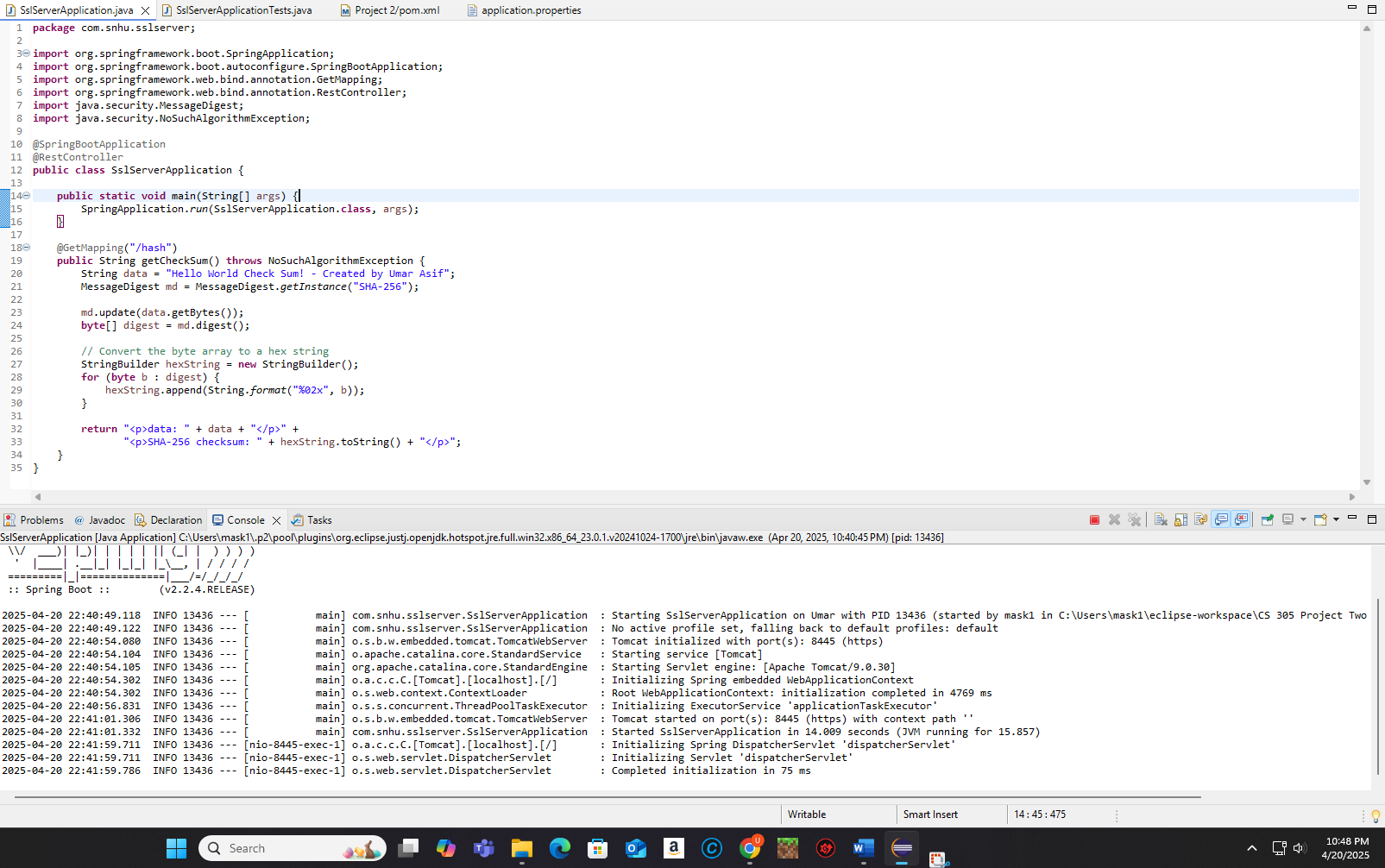
Insert screenshots below of the refactored code executed without errors and the dependency-check report.



<file:///C:/Users/mask1/eclipse-workspace/CS%20305%20Project%20Two%20Code%20Base/ssl-server_student/target/dependency-check-report.html>

## Functional Testing

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



## Summary

In this project, the Artemis Financial application was refactored to address security vulnerabilities using modern software security principles. The following steps were completed:

* Implemented a SHA-256 hash algorithm to create a checksum of a static string: "Hello World Check Sum!".
* Generated and imported a self-signed SSL certificate using Java Keytool.
* Updated the application’s configuration (application.properties) to enable HTTPS on port 8443 with the generated keystore.
* Refactored the code to serve data securely over HTTPS via a custom checksum endpoint.
* Ran a secondary static security scan using OWASP Dependency-Check to ensure no new vulnerabilities were introduced during refactoring.
* Conducted functional tests and verified the server ran without errors and delivered the expected cryptographic results.

The code changes were guided by the vulnerability assessment process flow: from cryptographic practices to secure API interactions and secure distributed components.

## Industry Standard Best Practices

To secure Artemis Financial’s application, several industry-standard best practices were applied:

* **Encryption Standards**: Used SHA-256, a secure and widely adopted hash function, for generating verifiable data checksums.
* **Secure Communication**: Transitioned the application from HTTP to HTTPS using an SSL/TLS certificate, encrypting data in transit and protecting against man-in-the-middle attacks.
* **Code Security**: Ensured that the checksum functionality did not expose any sensitive data or internal logic and validated user inputs.
* **Static Code Analysis**: Conducted static testing using OWASP Dependency-Check to verify that third-party libraries did not introduce known vulnerabilities.
* **Separation of Concerns**: Maintained clean, modular code that makes future audits, updates, or enhancements easier and more secure.

These practices enhance Artemis Financial’s resilience to cyber threats, align with compliance standards, and reinforce user trust.