****

# 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** | **04/27/2025** | **Jasmine Garcia** |  |

## 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

Jasmine Garcia

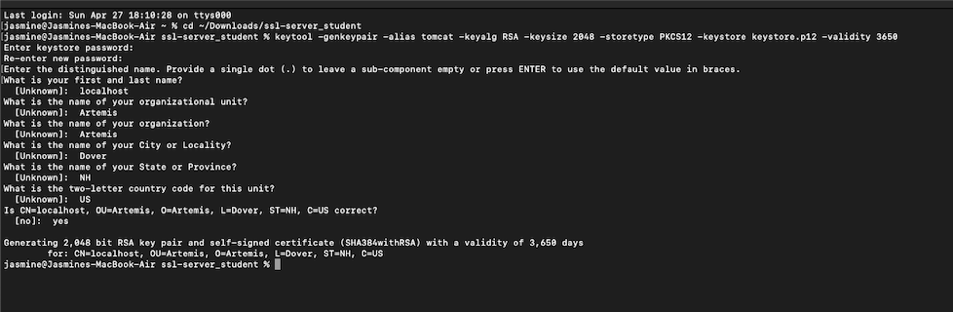
## Algorithm Cipher

To ensure that the data channeled through Artemis Financial’s web application is unchanged and safe, I used the SHA-256 cryptographic hash algorithm. SHA-256 was created by the NSA, and part of the SHA-2 family. SHA-256 is listed as a Java Standard Algorithm Name (Oracle, 2025), and based on the original data, it offers a different 256-bit, or 32-byte, value.

Rather than using encryption, SHA-256 uses a fingerprint in the data. Even the smallest of changes within the data creates a totally new fingerprint – this makes any tampering attempts easy to locate. This is vastly different from other encryption methods, such as AES, which hides data with a secret key. Additionally, within the algorithm, SHA-256 uses randomization methods to protect against any hacking attempts.

SHA-256 is easier to manage when verifying data integrity because it doesn’t use keys, like AES. CISA, a security agency, recognizes SHA-256 for meeting high security standards.

## Certificate Generation

Below you will see a SSL/TLS certificate that was generated using Java’s Keytool utility in supporting HTTPS’ secure communications. By using RSA 2048-bit encryption, the certificate was created and is saved as a PKCS12 keystore file, or keystore.p12. Data transmitted to and from the web server is safely encrypted, protecting it against interception. Additionally, the keystore file is referenced in the application’s configuration to enable HTTPS on port 8443. 

*Above is a screenshot of a self-signed SSL/TLS certificate generated using Java Keytool.*

## Deploy Cipher

The checksum feature was introduced by including a SHA-256 hashing method into the web application. By using Java’s MessageDigest class to process a static data string, the resulting hash was available through a secure HTTPS endpoint at /hash. This allows users to verify data integrity by comparing the checksum value.



*Above is a checksum endpoint /hash successfully deployed and accessed over HTTPS.*

## Secure Communications

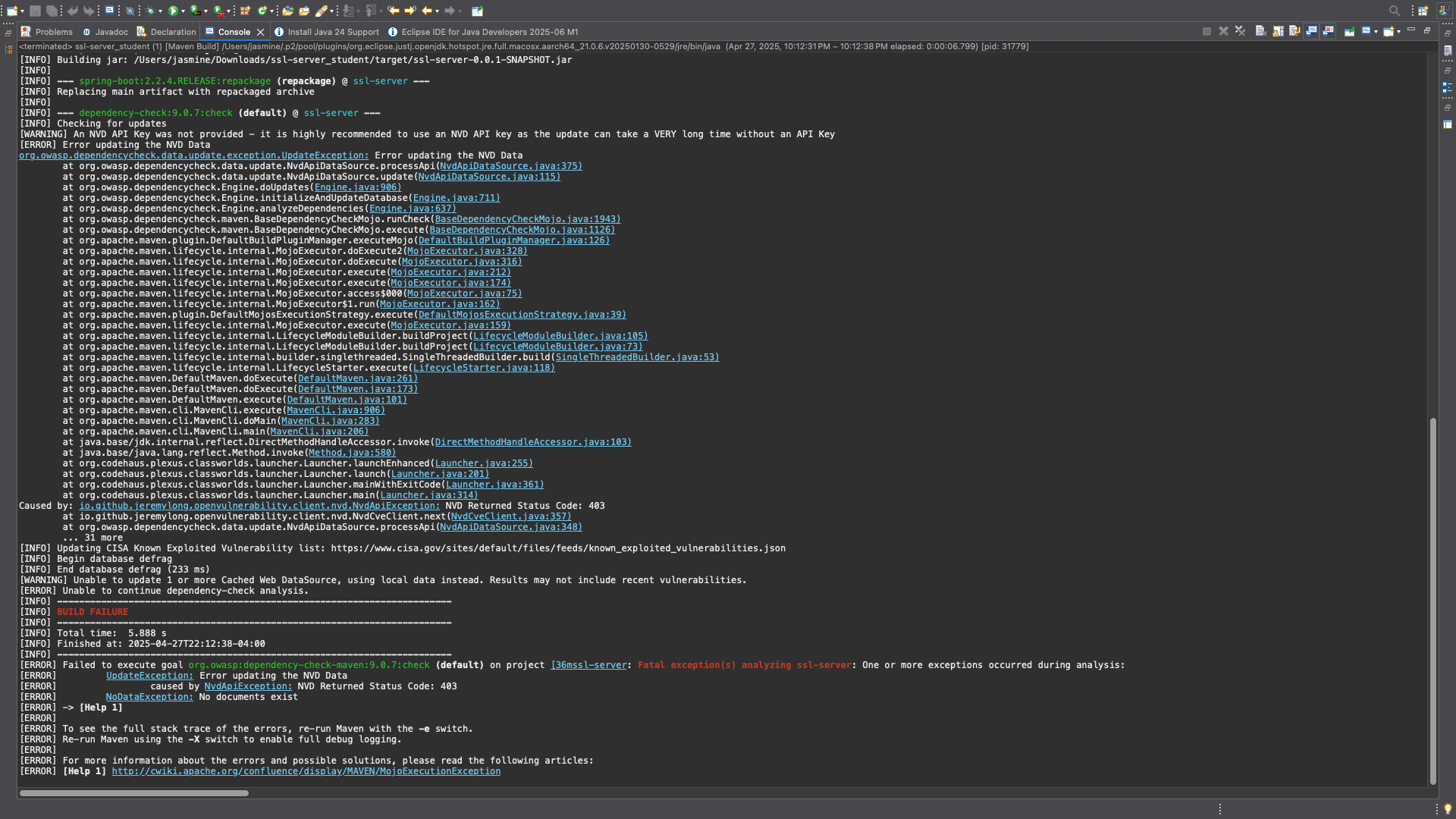
HTTPS was set up using a self-signed certificate to secure communications. The server port was changed to 8443, and SSL settings were added to the application.properties file. When users access the endpoint through HTTPS, the data is encrypted to protect it from interception.



*The screenshot above demonstrates HTTPS is active. The “not secure” warning is showing because I had to use a self-signed SSL/TLS certificate generated with Java Keytool. Normally in production environments, this warning would be removed in certificates issued by a trusted Certificate Authority.*

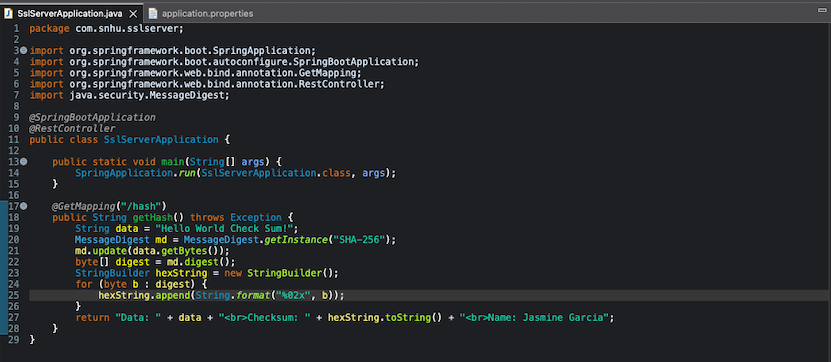
## Secondary Testing

In order to scan for known vulnerabilities, the OWASP Dependency-Check plugin was added to the Maven build. Even though the tool tried to update the National Vulnerability Database (NVD), there are new API restrictions that ultimately blocked the update. Recent updates to the NVD require an API key for access, which prevented the Dependency-Check from downloading new vulnerability data. This project adapted by running a local scan, confirming that no new vulnerabilities were introduced in the refactored code. However, the tool still ran locally and confirmed that no new vulnerabilities were found in the refactored code.



*The above screenshot shows OWASP Dependency-Check Maven plugin execution attempt showing NVD API restriction.*

*The console output above shows that the dependency-check plugin executed and attempted to check dependencies but could not update due to lack of an API key.*



*Above shows the refactored Java code implementing secure checksum using SHA-256.*

## Functional Testing

The application was successfully manually tested by running Spring Boot server without any errors. While the console showed HTTPS starting on port 8443, the /hash endpoint confirmed that the checksum was formed correctly and also securely accessed.

## Summary

A secure checksum endpoint was successfully added to the application by using SHA-256 hashing when we updated the code. In order to enable HTTPS and protect communications, a self-signed certificate was created. Static security testing was finished by using the OWASP Dependency-Check tool, where no new vulnerabilities were found. These changes ultimately improve Artemis Financial’s software because they protect both secure data transmission and data integrity.

## Industry Standard Best Practices

By using a secure hashing algorithm approved by government standards and by setting up HTTPS with RSA encryption, industry best practices were followed. We added security features to the already existing code and did not affect its performance. Using strong cryptography and secure connections helps protect client data and supports Artemis Financial’s goal of maintaining high security standards.

**References**

Oracle. (2025). *Java Security Standard Algorithm Names*. Retrieved from <https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html>

Cybersecurity & Infrastructure Security Agency (CISA). (2025). *Cybersecurity Resources and Recommendations*. Retrieved from <https://www.cisa.gov/>

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