****

# 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** | **6/21/2025** | **Victor Ngetich** |  |

## 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

Victor Ngetich

## Algorithm Cipher

The core mission of Artemis Financial is to deliver financial services and programs to a global clientele. To support this goal securely, I recommend implementing the SHA-256 algorithm as the chosen encryption standard. SHA-256 is recognized for its robustness, making it extremely resistant to unauthorized access or cryptographic attacks. Due to the algorithm’s complexity, decrypting data through brute-force methods would require an impractical amount of time and computational power.

This level of security is particularly critical when dealing with sensitive communications in the financial sector, where SHA-256 is a widely accepted standard. The algorithm uses a hashing process where input data is compressed into a fixed-size hash, with the output (hash value) dependent on a 256-bit configuration. The strength of this approach lies in its large number of possible combinations, which significantly reduces the risk of unauthorized data breaches. The randomness incorporated into the hashing process increases its unpredictability, further strengthening the encryption.

Encryption techniques generally fall into two categories: symmetric and asymmetric. Symmetric encryption, such as AES-256, uses a single key for both encryption and decryption. One major advantage of this method is its speed and lower computational requirements (Yedakula, K., 2019). In contrast, asymmetric encryption involves the use of a public and private key pair, offering a higher level of security and often used in online communications.

Encryption has a long history, dating back to ancient civilizations as early as 600 BC (A Brief History of Encryption, Thales Group, 2016). Over time, encryption methods have evolved to address the growing need for data protection. Today, ensuring the confidentiality and integrity of information remains a central priority in cybersecurity, especially within the financial industry.

## Certificate Generation

Insert a screenshot below the CER file.

A computer screen with white lines

AI-generated content may be incorrect.

## 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.

A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

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

A screenshot of a computer

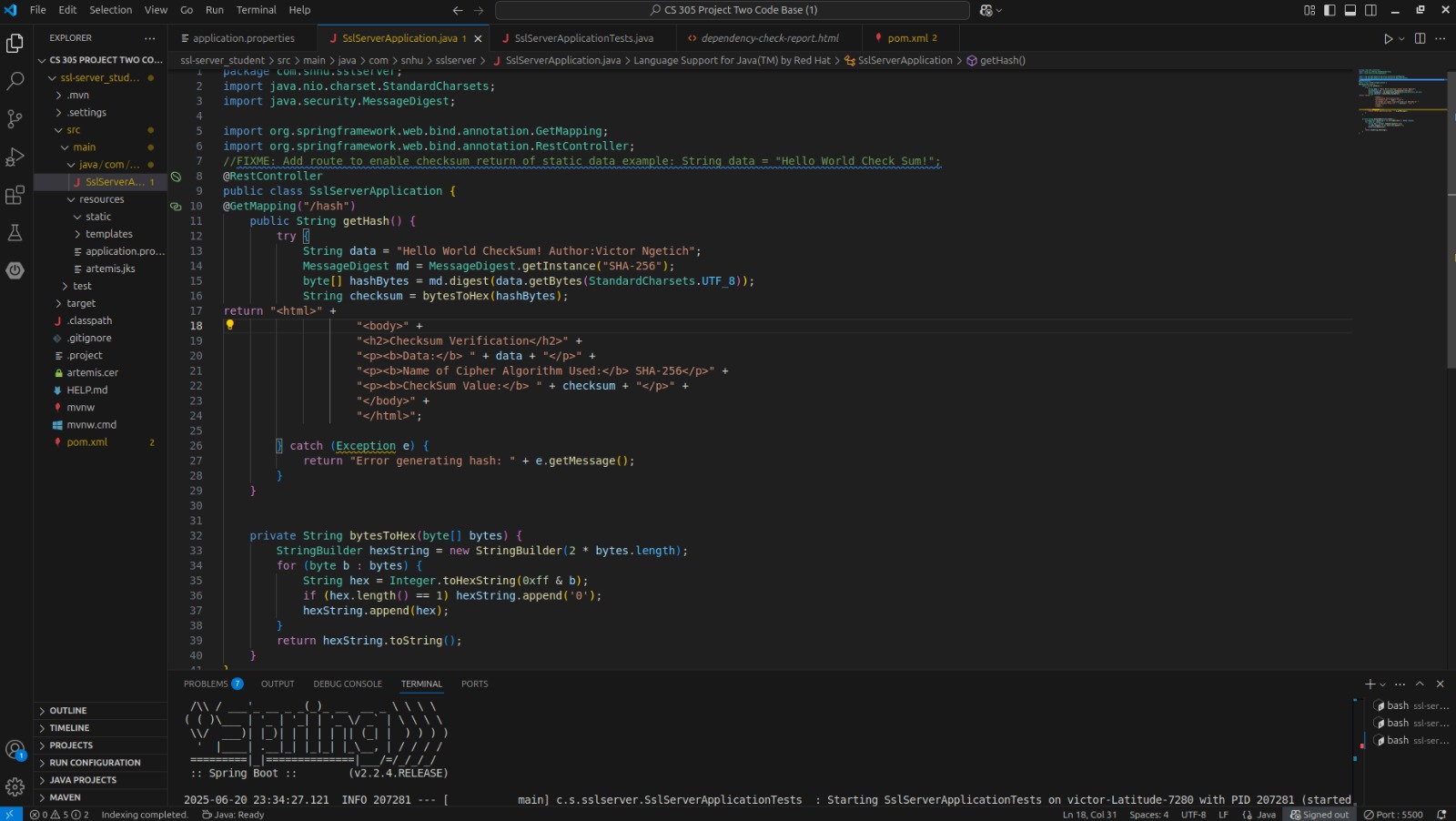
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

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



## Summary

A key security improvement in our application was the implementation of self-sign certificates to enable HTTPS support. This measure ensured that data exchanged between the server and clients is encrypted, strengthening the overall security of the system. Additionally, we updated and refactored the `pom.xml` file to address and resolve vulnerabilities identified during the OWASP Dependency-Check analysis.

The first step in this process involves properly generating self-sign certificates. This was critical to establishing a secure HTTPS connection once the application was deployed. Using HTTPS not only secures communication but also assures users that they are interacting with our legitimate platform and not an imposter, thereby reinforcing trust in our service.

Next, we verified that our hashing functionality was working as expected by cross-checking data with checksums. This helped confirm that client data is being hashed securely, making it much harder to reverse-engineer or expose sensitive information. This layer of security is essential to protecting user data and upholding the integrity of our application.

Finally, we ensured that all detected security vulnerabilities were remediated. This step is vital in maintaining a secure and reliable application, as it ensures all components are up-to-date and functioning as intended.

Maintaining system security requires continuous effort. One key best practice is regularly updating and patching software to eliminate known vulnerabilities. This helps protect the system from threats that exploit outdated components. Another crucial measure is applying the principle of least privilege—ensuring that users have access only to the resources necessary for their roles. Although our current system may not fully implement this yet, it is a security priority for future improvements to reduce internal risks.

## Industry Standard Best Practices

**Implementation of Secure Coding Best Practices**

To enhance the security of the application, the following best practices were applied:

* Integrated robust encryption techniques, specifically AES-256 for data encryption and SHA-256 for data integrity verification.
* Configured HTTPS to secure all data transmissions and prevent unauthorized access during client-server communication.
* Considered the principle of least privilege for future development, ensuring that users will only have access to the information necessary for their roles.
* Conducted both static and dynamic code analysis to uncover and address potential security flaws introduced during development.

**Importance to Artemis Financial**

Adopting these industry-standard security measures contributes significantly to Artemis Financial’s overall safety and reliability. It helps safeguard sensitive customer information, maintain compliance with regulatory standards, and foster user confidence. Moreover, actively identifying and resolving security risks strengthens the application's defense against cyber threats, which ultimately minimizes future maintenance burdens and reduces potential legal exposure.