****

# Practices for Secure Software Report

**Table of Contents**

[**Document Revision History 3**](#_heading=h.30j0zll)

[**Client 3**](#_heading=h.1fob9te)

[**Instructions 3**](#_heading=h.3znysh7)

[**Developer 4**](#_heading=h.2et92p0)

[**1. Algorithm Cipher 4**](#_heading=h.tyjcwt)

[**2. Certificate Generation 4**](#_heading=h.3dy6vkm)

[**3. Deploy Cipher 4**](#_heading=h.1t3h5sf)

[**4. Secure Communications 4**](#_heading=h.4d34og8)

[**5. Secondary Testing 4**](#_heading=h.2s8eyo1)

[**6. Functional Testing 4**](#_heading=h.17dp8vu)

[**7. Summary 4**](#_heading=h.3rdcrjn)

[**8. Industry Standard Best Practices 4**](#_heading=h.26in1rg)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **12/30/2024** | **Victor Tran** | **Artemis Financial Vulnerability Assessment Report** |

## 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 Tran

## Algorithm Cipher

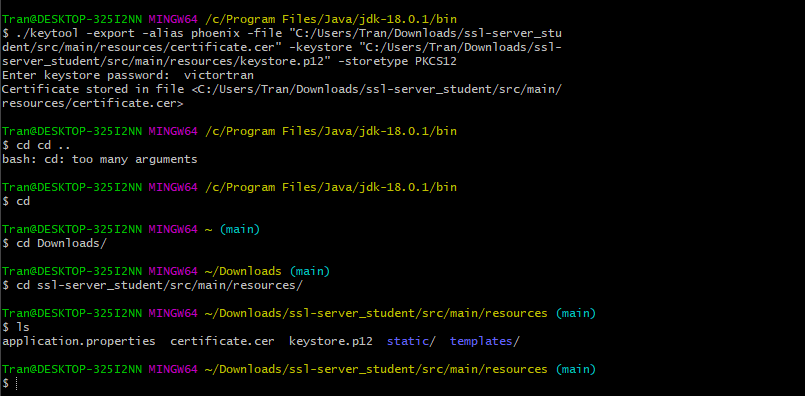
SHA-256 (Secure Hash Algorithm 256-bit) and TLS (Transport Layer Security) are recommended as the encryption solutions for this project. SHA-256 is a cryptographic hash function that converts any input into a fixed 256-bit value, ensuring data integrity by detecting even the smallest changes in the input. It is part of the SHA-2 family, which replaced older, less secure algorithms like SHA-1. Since its introduction in 2001 by the National Institute of Standards and Technology (NIST), SHA-256 has become a widely trusted standard for secure communications, file verification, and digital signatures due to its strong resistance to brute-force and collision attacks.

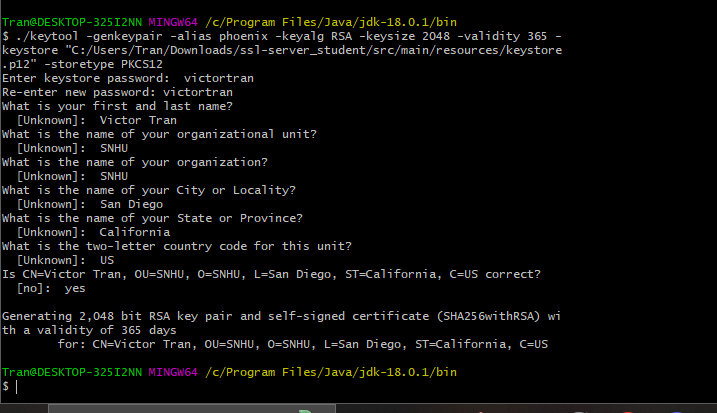
TLS, the successor to SSL, is a protocol designed to secure data in transit. It uses a combination of symmetric encryption, such as AES, and asymmetric encryption, such as RSA or ECDSA, to ensure confidentiality, integrity, and authenticity. Random number generation is a critical part of TLS, enabling secure key generation and preventing replay attacks. The current versions, TLS 1.2 and TLS 1.3, offer robust protection against modern attacks and are globally recognized for securing websites and APIs.

Together, SHA-256 and TLS provide comprehensive security. SHA-256 ensures data integrity by generating unique hash values, while TLS protects data during transmission by encrypting it. These algorithms follow industry standards and address known vulnerabilities, making them essential for secure software development. Their adoption ensures that the application aligns with modern security requirements and builds trust by safeguarding sensitive information.

## Certificate Generation

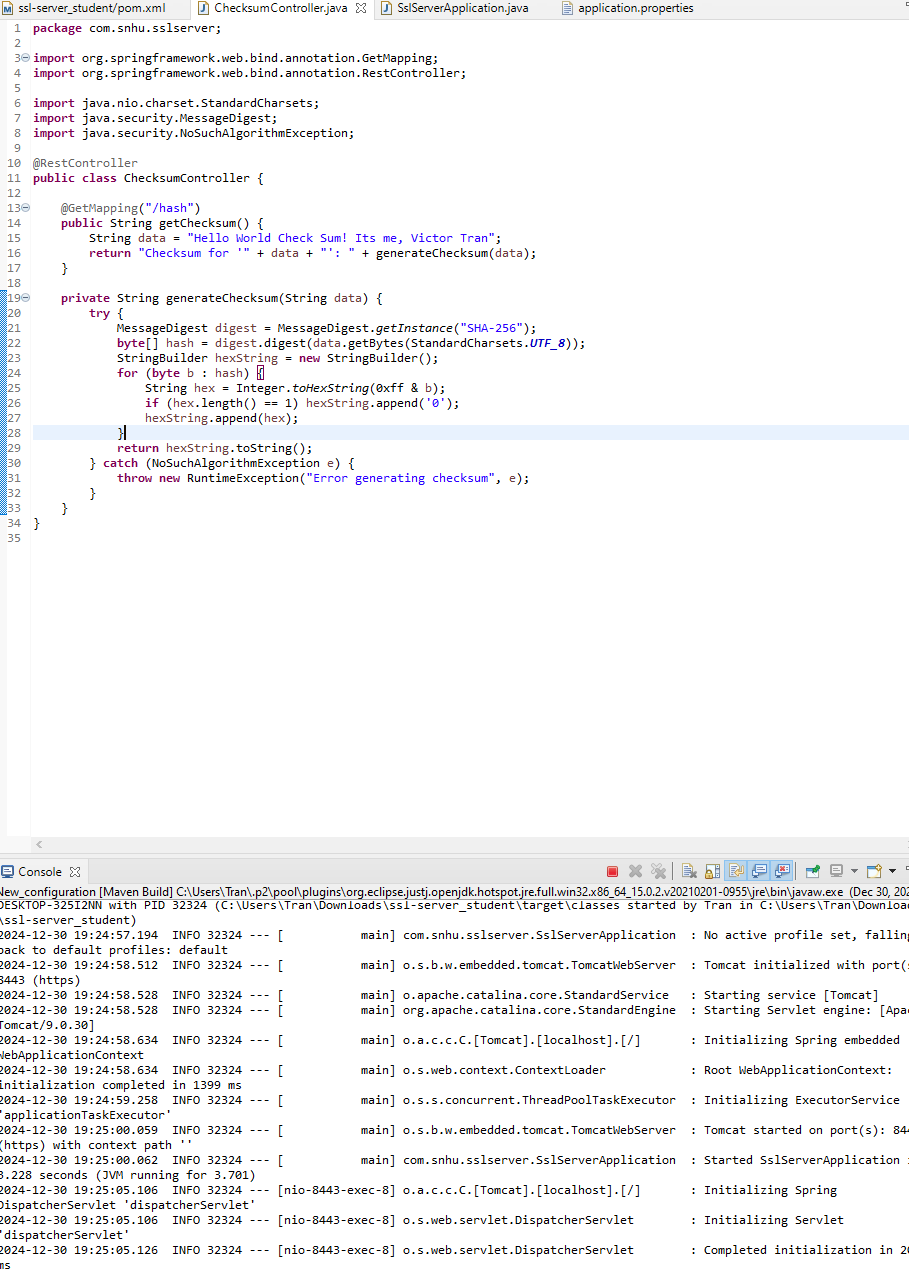
Insert a screenshot below of the CER file.





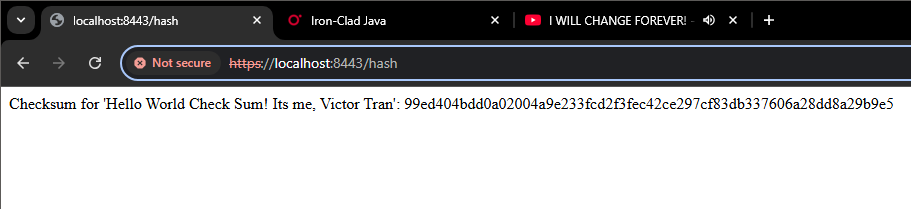
## Deploy Cipher

Insert a screenshot below of the checksum verification.



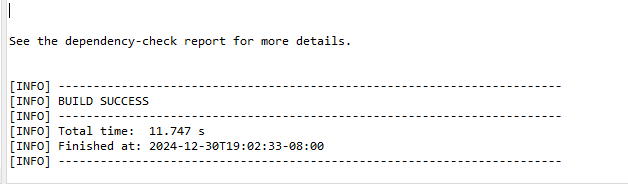
## Secure Communications

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



## Secondary Testing

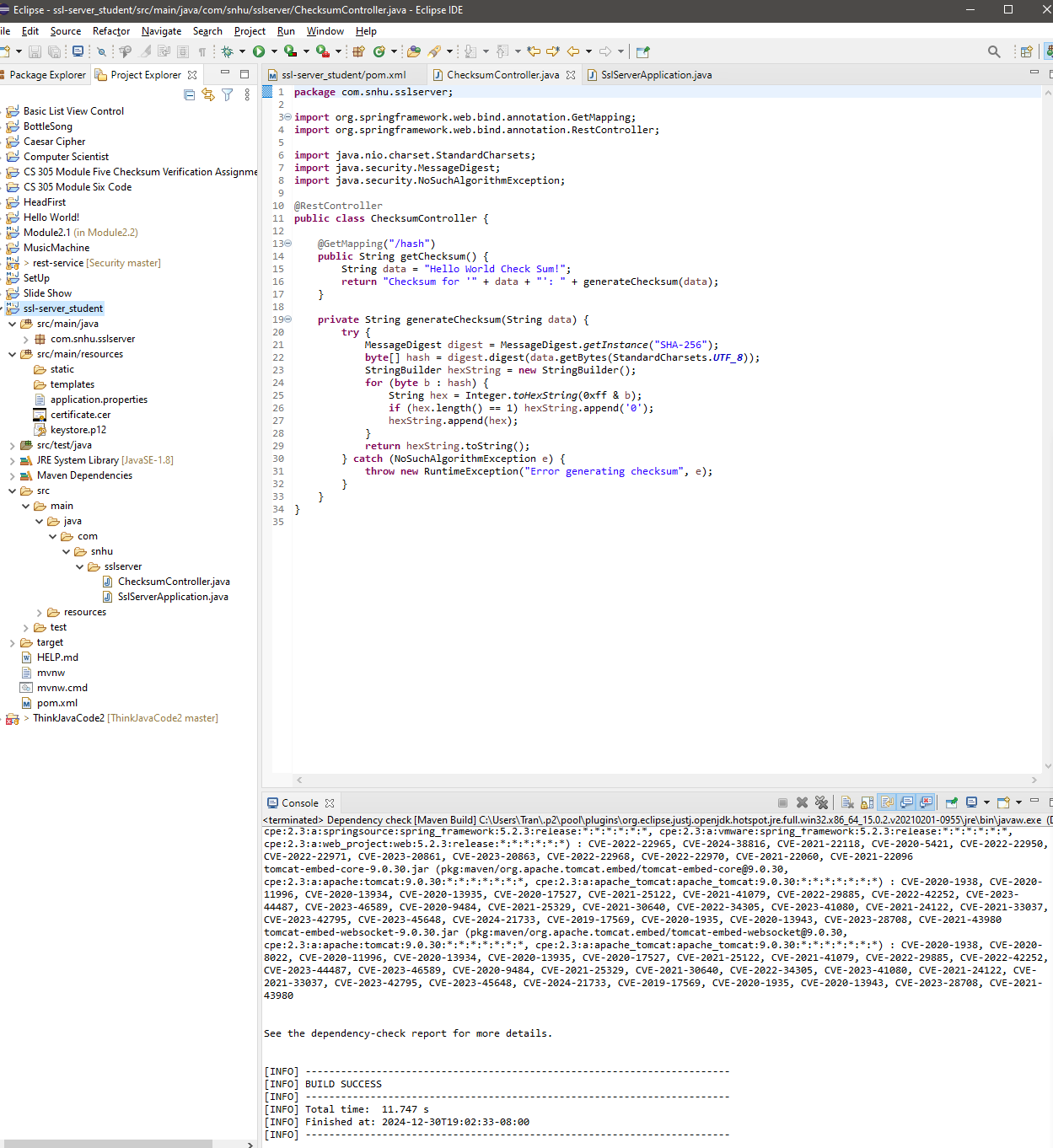
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

In this project, I added a checksum feature and secured the application with HTTPS. I tested the application to make sure it works correctly and followed security protocols.

#### **Findings:**

1. **Functional Testing:**
   * The application ran without errors, and the checksum and HTTPS features worked as expected.
2. **Secondary Testing:**
   * A scan with the OWASP Dependency-Check tool found vulnerabilities in third-party libraries, which were already present in the original code.
   * No new vulnerabilities were introduced by the changes I made.

#### **Conclusion:**

The updated application works as expected and meets security requirements. Existing vulnerabilities are from third-party libraries, which can be fixed in the future by updating those libraries.

## Industry Standard Best Practices

I used secure coding practices to protect the application and mitigate vulnerabilities, following advice from **Iron-Clad Java** by Jim Manico and August Detlefsen (2014).

#### **Maintaining Security**

1. **Secure Communication:**
   * I implemented HTTPS to encrypt data sent over the network, ensuring passwords and sensitive information are protected. According to Manico and Detlefsen (2014), HTTPS, based on TLS, is the industry standard for secure data transmission, protecting against eavesdropping and impersonation.
2. **Strong Encryption:**
   * I followed recommendations to use strong encryption methods and avoid weak ciphers like RC4. Manico and Detlefsen (2014) emphasize using modern TLS protocols to enhance security and protect against future threats.

#### **Why Secure Coding Helps**

1. **Protects Client Data:**
   * Secure practices build trust by protecting client information and meeting compliance standards (Manico & Detlefsen, 2014).
2. **Prevents Attacks:**
   * Using strong encryption and secure communication helps prevent common attacks like data theft and impersonation (Manico & Detlefsen, 2014).
3. **Prepares for Future Threats:**
   * By following modern security standards, the application can handle future risks using strong encryption and secure key management (Manico & Detlefsen, 2014).

By using these practices, I made the application safer and more reliable.

### **APA Citation**

Manico, J., & Detlefsen, A. (2014). *Iron-Clad Java*. McGraw Hill Computing.