****

# 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** | **06/15/24** | **Ty Wheeler** | **First revision** |

## 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

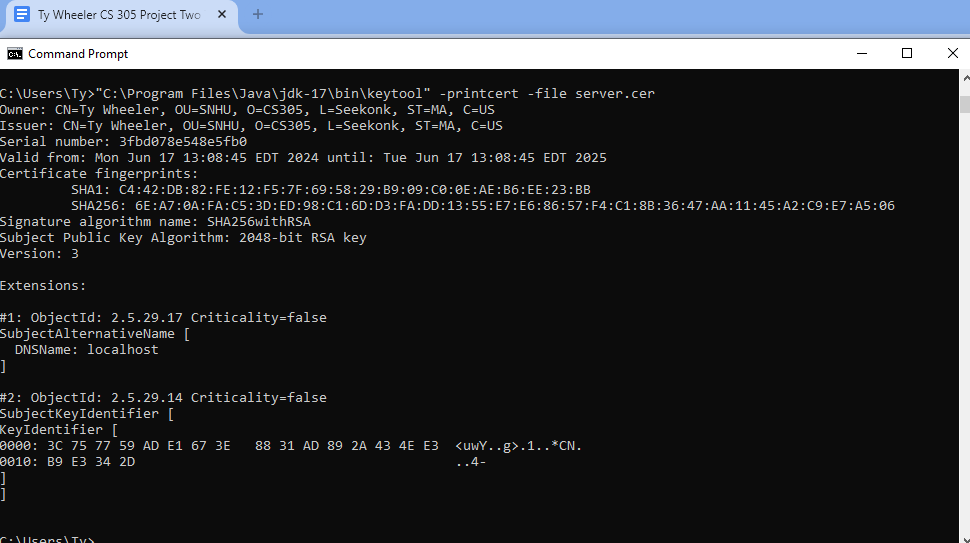
Ty Wheeler

## Algorithm Cipher

I recommend using the SHA-256 algorithm for Artemis Financial's web-based application. SHA-256 is a cryptographic hash function known for its security and efficiency, generating a 256-bit hash value that is highly secure against brute-force attacks due to the vast number of possible hash values. The algorithm processes input data through a series of logical operations, resulting in a fixed-size hash value unique to the input data, suitable for verifying data integrity and authenticity. Unlike AES-256, which is a symmetric encryption algorithm for encrypting and decrypting data, SHA-256 is a hash function used for one-way encryption, where the original input cannot be derived from the hash value. Asymmetric keys make it easier to distribute public keys since there are separate keys for encryption and decryption or public and private keys. This comes at the cost of performance however, as this system is slower due to it requiring more resources. Encryption algorithms have an extensive history that date far back in history, way before computers existed. Julius Caesar used a simple substitution cipher known as the Caesar cipher to protect his military messages. “Caesar shifted each letter of his message three letters to the right to produce what could be called the ciphertext. The ciphertext is what the enemy would see instead of the true message.” (The Story of Cryptography: History). As history went on, these sorts of algorithms were used to protect sensitive information, especially when it came to sending sensitive information in times of war such as World War II, “The DES can be said to have ‘jump started’ the nonmilitary study and development of encryption algorithms. In the 1970s there were very few cryptographers, except for those in military or intelligence organizations, and little academic study of cryptography” (NIST Technical Series Publications ,1) The Data Encryption Standard (DES) was one of the first widely used encryption standards. As technology advanced DES was eventually replaced by the Advanced Encryption Standard (AES) in the early 2000s offering improved security and efficiency.

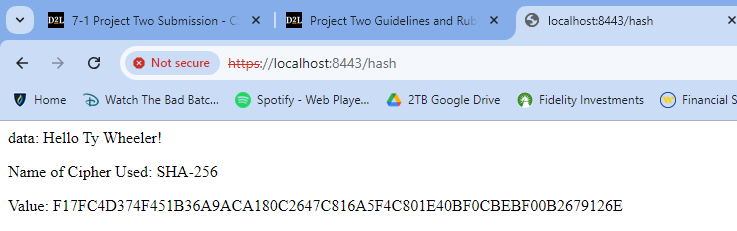
## Certificate Generation

Insert a screenshot below of the CER file.



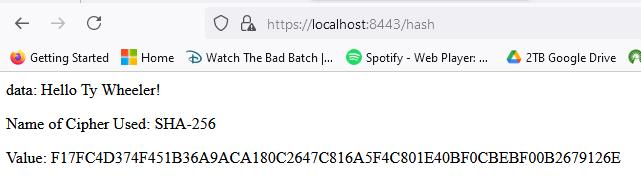
## Deploy Cipher

Insert a screenshot below of the checksum verification.



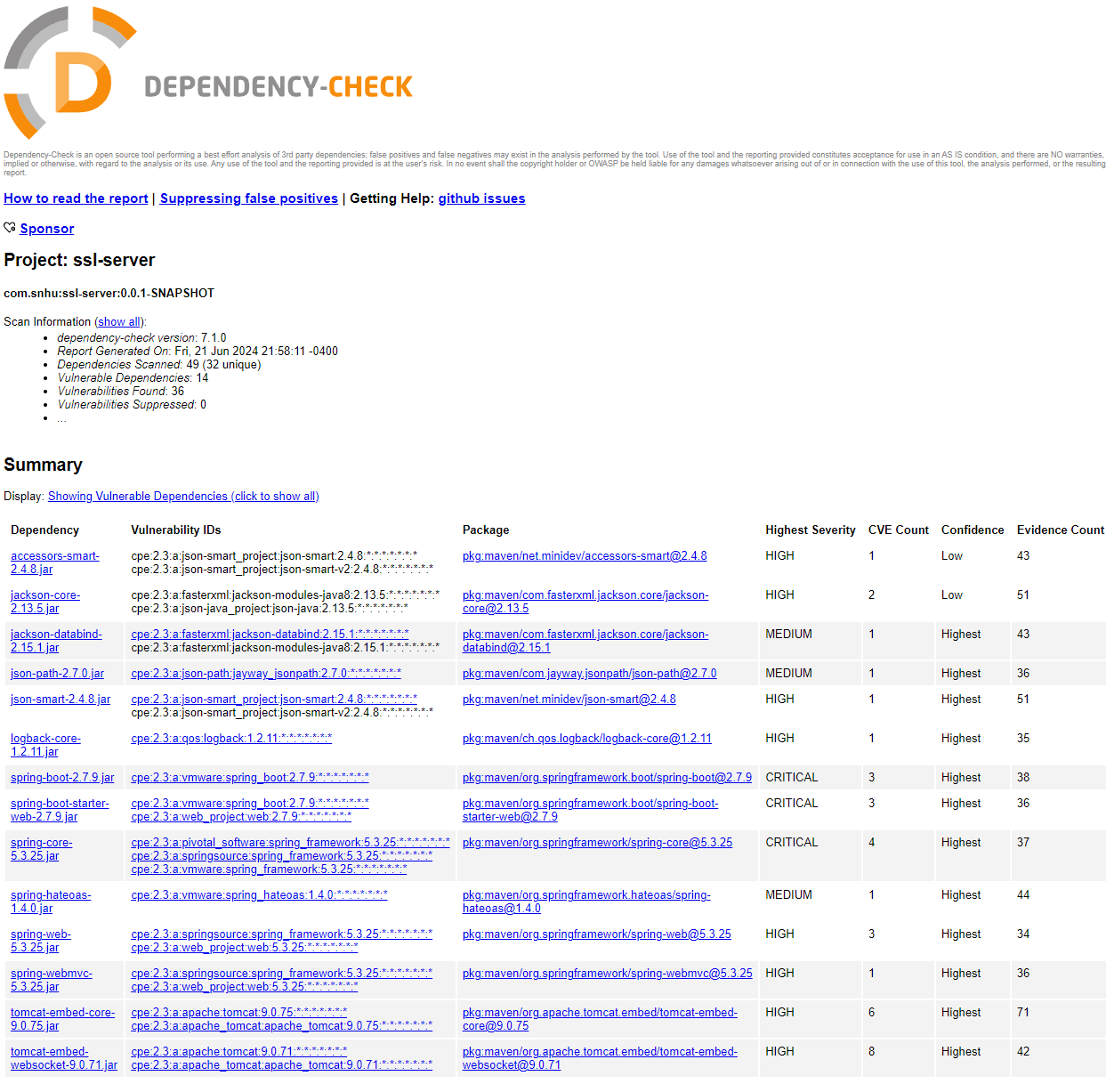
## Secure Communications

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

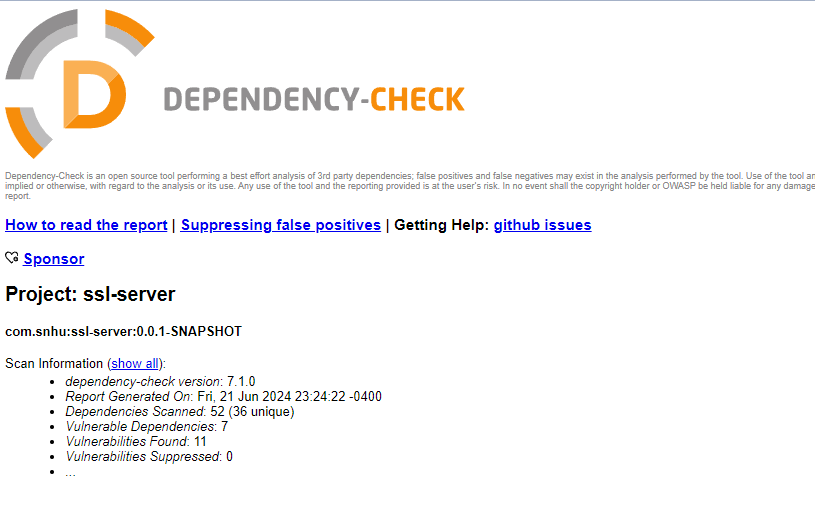


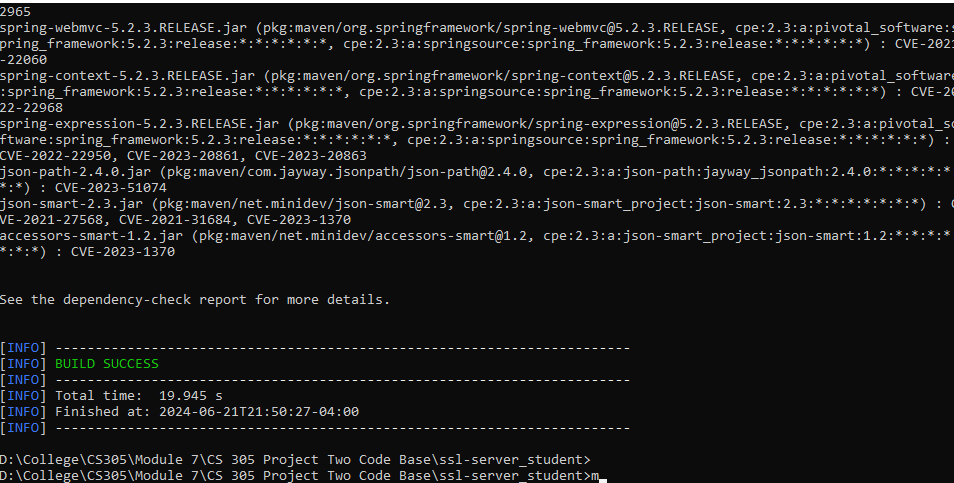
## Secondary Testing

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



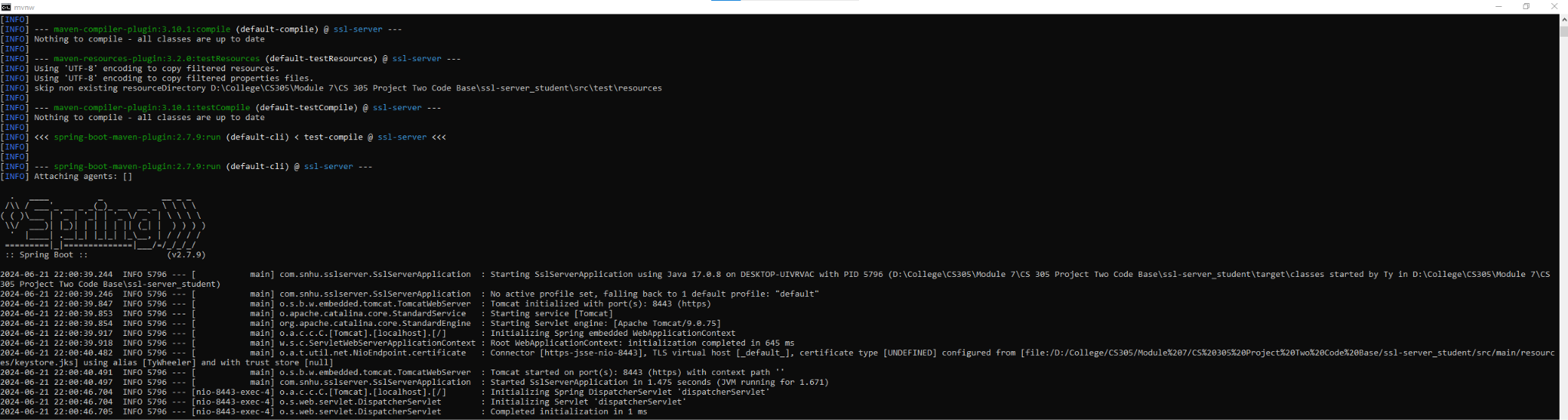






## Functional Testing

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



## Summary

Many areas of security were addressed by refactoring this code. Cryptography was used by implementing SHA-256 to enhance the security of sensitive data and address vulnerabilities related to cryptography. In regards to the client/server, secure distributed composing was employed to ensure secure communication between clients and servers, including implementing secure protocols and verifying data integrity. I also followed best practices by creating self-signed certificates for HTTPS, ensuring that our hashing function operated correctly and vulnerabilities found during the dependency check were patched.

I managed dependencies carefully to ensure that all third-party libraries used in the application were up-to-date and free from known vulnerabilities. I used tools like OWASP Dependency Check to scan for vulnerabilities in third-party dependencies. I configured the application and its components securely, following best practices and guidelines for secure configuration. This included secure storage of credentials, enabling HTTPS, and configuring secure communication protocols.

## Industry Standard Best Practices

The application was configured following best practices and guidelines. This included securely storing sensitive information such as passwords and API keys and ensuring that secure communication protocols such as HTTPS were used. Using strong encryption algorithms to protect sensitive data both at rest and in transit. This helped ensure that even if data was intercepted, it would be unreadable without the encryption key. Implementing secure error handling practices to prevent sensitive information from being leaked in error messages. This included providing generic error messages to users and logging error information for development purposes. Keeping the application and its dependencies up to date with the latest security patches and updates helped protect against newly discovered vulnerabilities.

Applying industry-standard best practices for secure coding is crucial for the company's overall well-being as it helps protect the company's assets, reputation, and customer trust. By following these best practices the company reduces the risk of security breaches, data leaks, and other security incidents that could result in financial losses, legal liabilities, and damage to its reputation. Secure coding practices also help ensure compliance with industry regulations and standards. This demonstrates to customers and stakeholders that the company takes security seriously. Overall, investing in secure coding practices is essential for safeguarding the company's digital assets and maintaining a good reputation.

References

Data Encryption Standard - NIST Technical Series Publications. (n.d.). https://nvlpubs.nist.gov/nistpubs/sp958-lide/250-253.pdf

*The story of Cryptography: History*. The Story of Cryptography : Historical Cryptography. (n.d.). https://ghostvolt.com/articles/cryptography\_history.html#:~:text=The%20ciphertext%20is%20what%20the,called%20a%20%E2%80%9Cshift%20cipher.%E2%80%9D