****

# 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** | **April 18, 2025** | **Julianne Takaya** |  |

## 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

Julianne Takaya

## Algorithm Cipher

Artemis Financial handles many users’ sensitive and confidential financial and personal information, and so they have a duty to ensure that the information they possess is kept confidential. Encrypting the data they send and receive is a must. For this I recommend using the hashing algorithm SHA-256. This algorithm was developed by the NSA in 2001 and is regarded as one of the most secure algorithm ciphers. The computational power that would be needed to break this algorithm does not exist.

The algorithm works by taking the original message and adding padding to make it 64 bits short of a multiple of 512, then a modulus value is added to make the message a multiple of 512. The entire message is broken down into blocks of 512 bits each and these blocks are then processed through several computation cycles and compressed into intermediate hash values. These block hash values are then added together to create the final 256-bit hash value. The fixed size block hashing guarantees a unique hash value for each input, making collisions, or different input data with the same hash value, impossible. To decode the data, one will need access to the decryption key. There are two types of keys, symmetric, or private key, and asymmetric, or public key. Symmetric key encryption has one key that encrypts and decrypts the data, so care must be taken to ensure that the key is kept secure. Asymmetric keys have two keys, one secret key and one public key. These keys are used to sign and validate data.

While cryptography dates back to ancient times, modern cryptography can be traced to the efforts made during the second world war and the Enigma machine used by the German military to encode messages. Modern, computer-based encryption started in the seventies by IBM, who designed a block cipher to protect their customers’ data. This cipher, named the Dat Encryption Standard(DES) was adopted as the standard until it was cracked in 1997. In the 2000’s the Advanced Encryption Standard (AES) replaced the DES as the standard. AES is royalty-free worldwide and approved for use in classified US government information.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

Unfortunately I am unable to get confirmation that the checksum verification works, as this is the response I get when I run the code

A computer screen with many lines

AI-generated content may be incorrect.

## Secure Communications

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

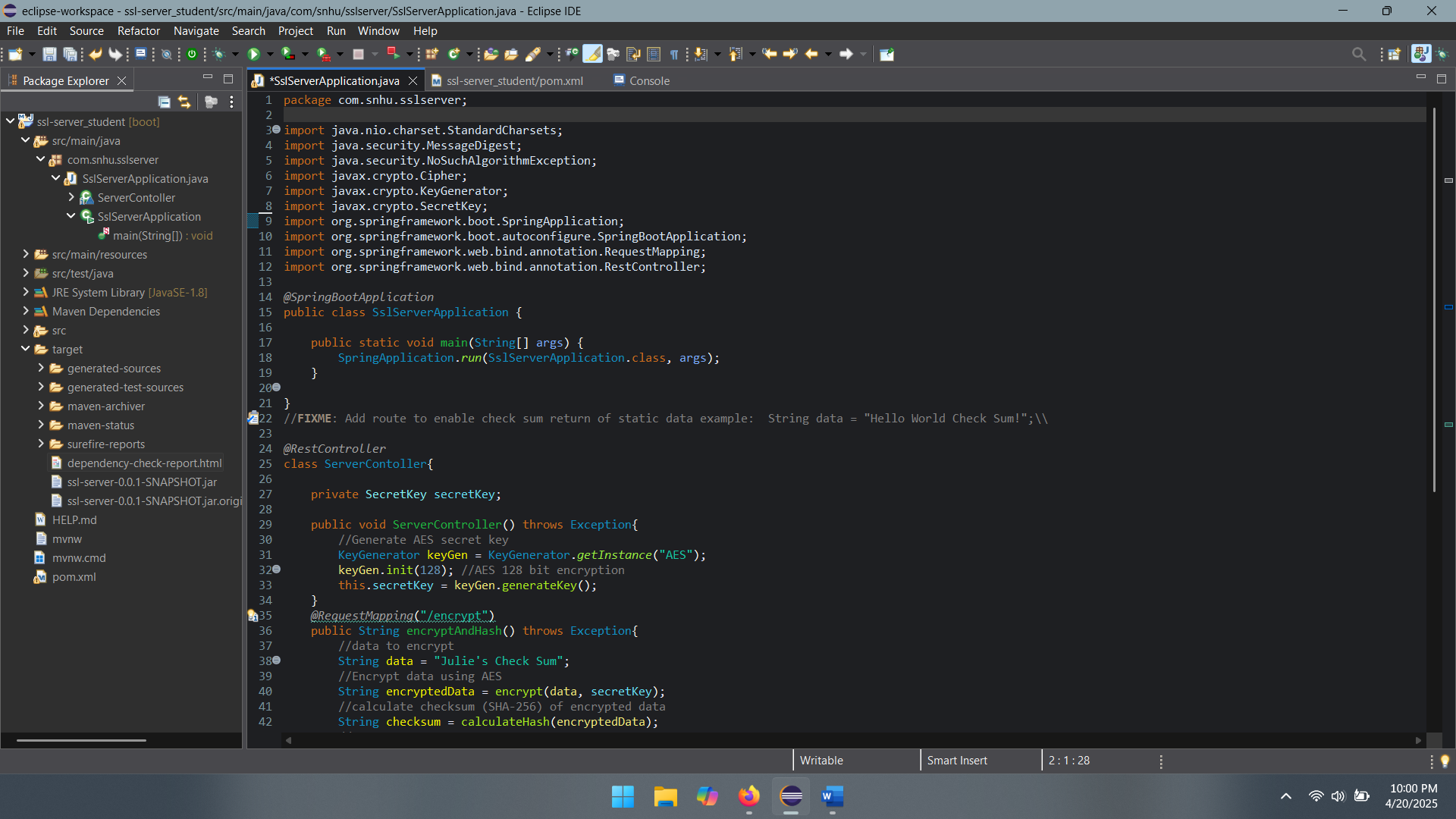
A screenshot of a computer

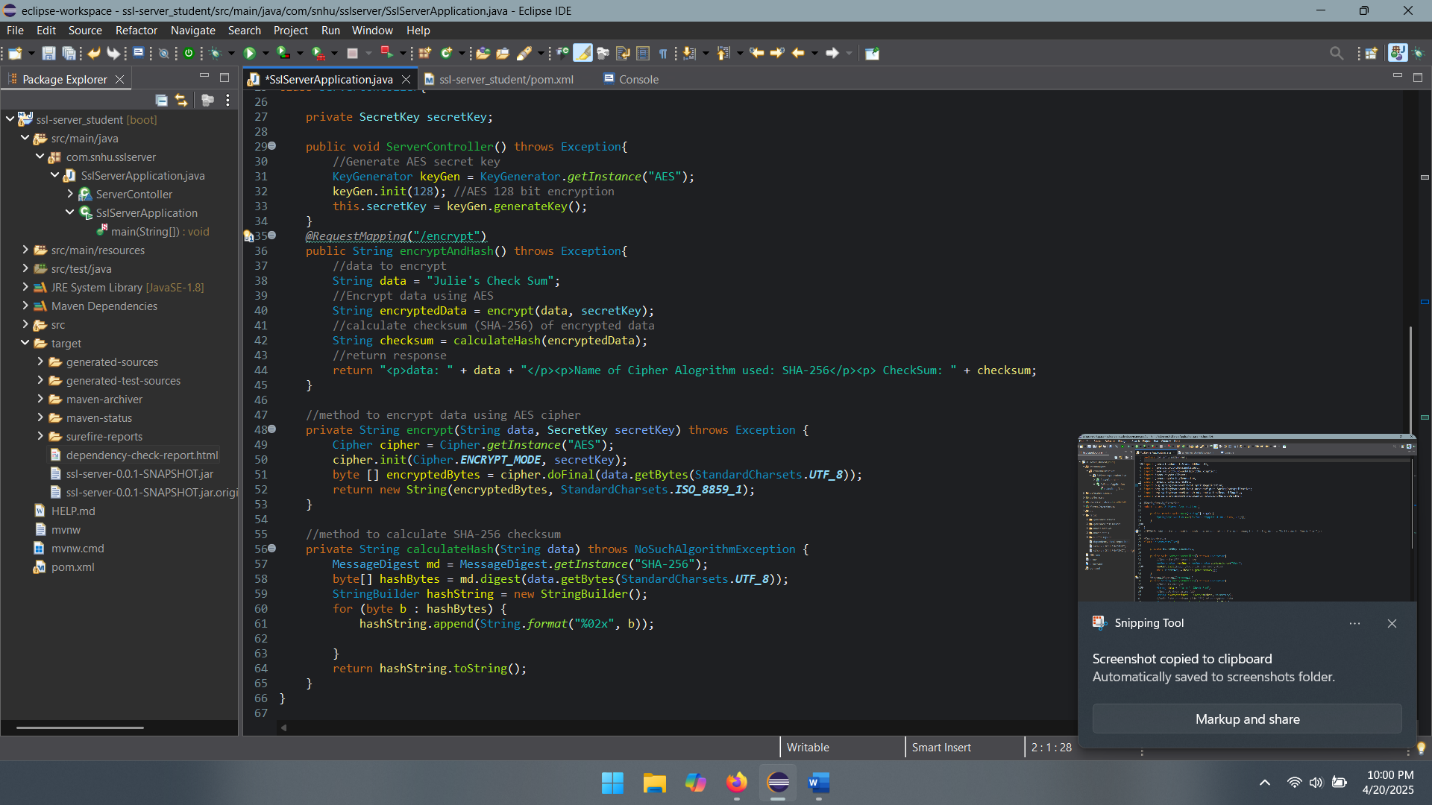
AI-generated content may be incorrect.

## Secondary Testing

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

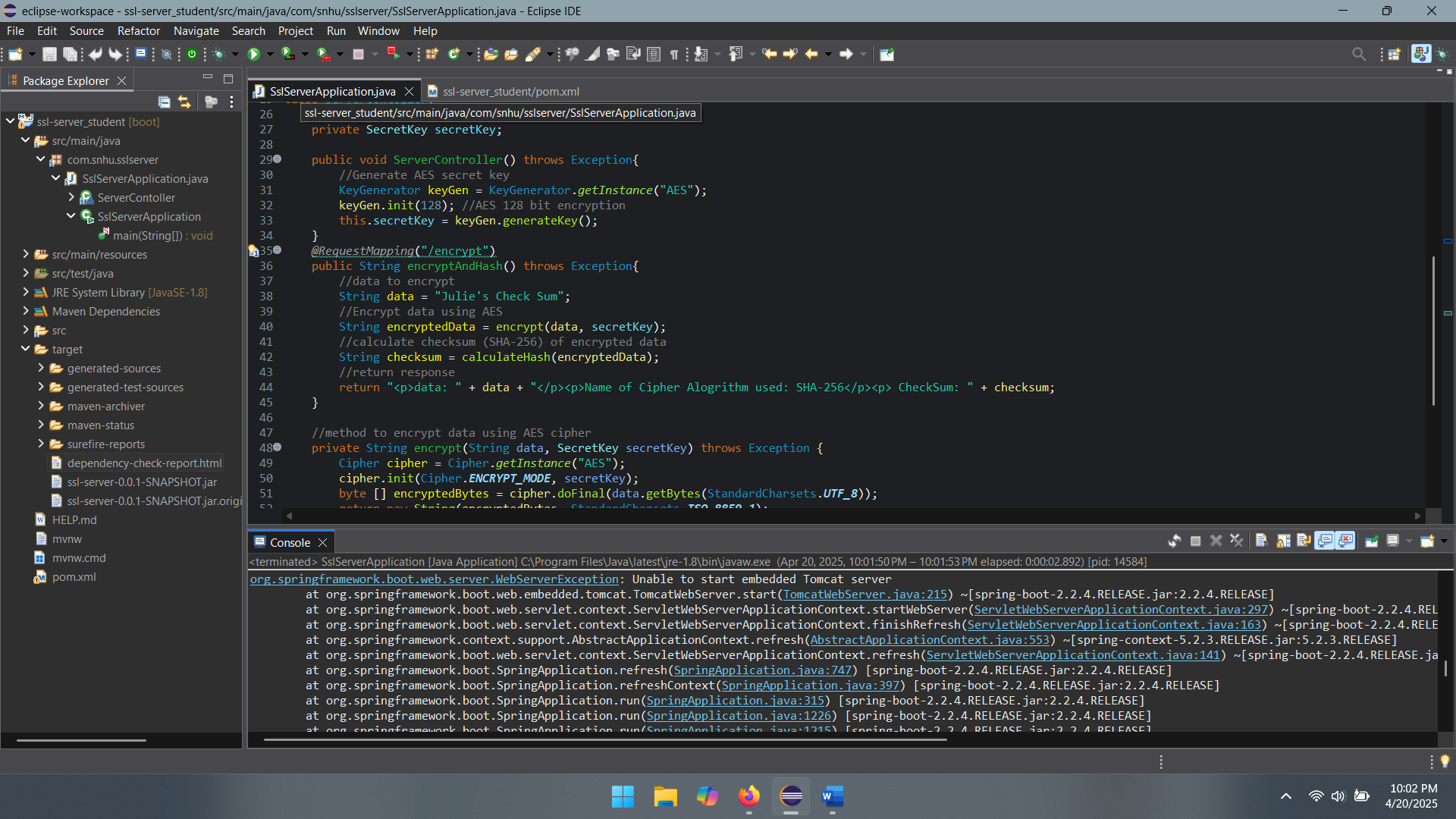
AI-generated content may be incorrect.





## Functional Testing

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



## Summary

While refactoring my code I have added a secured RestController to work as the secure controller for my programs hash RESTful stop. The ServerController class works to match the problems presented by the vulnerability assessment diagram. Additionally, I chose to work with the SHA-256 hashing cipher, as it is very secure and runs a very small chance of collisions. To best maintain the current security of the application, I would suggest running dependency checks once or twice monthly to keep it up to date. This will help protect the company and the users sensitive data.

## Industry Standard Best Practices

To maintain the security of the application I followed the industry standard best practices by specifying character sets that must be used for all input sources, and conducted input validation on the server side. I also used only trusted system objects and ensured outside systems could not use the program. The secret keys are kept in a secure location, and are protected from outside access. I used generic error messages when necessary.