****

# 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** | **12/22/2024** | **THOMAS SEIBERT** |  |

## 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

TOM SEIBERT

## Algorithm Cipher

In this scenario, since Artemis Financial is looking to protect its client data and financial information, as well as add a file verification step to its web application in the form of a checksum, I will recommend as an encryption cipher a member of the Secure Hash Algorithm (SHA), specifically SHA-384. It is a cryptographic hash function that produces a 384-bit (48-byte) hash value. It is commonly used in various security applications and protocols and given the sensitive nature of the financial data being handled, it would be a suitable choice.

In terms of hash functions and bit levels of the cipher, SHA-384, as the name suggests, generates a hash value that is 384 bits long. The hash function works by taking an input and returning a fixed-size string of bytes. The output is unique to each unique input - a small change in the input will produce such a drastic change in output that the new hash value appears uncorrelated with the old hash value.

*\*Addressing the use of random numbers, symmetric versus non-symmetric keys, etc.\**

In the context of SHA-384, random numbers are often used as salts. A salt is random data that is used as an additional input to a hash function to safeguard passwords against hash table attacks.

SHA-384, like all hash functions, does not use keys and therefore does not distinguish between symmetric and asymmetric keys. However, it can be used in conjunction with other cryptographic algorithms that do use keys. For example, in a digital signature algorithm, the message might be hashed using SHA-384, and then the hash could be encrypted using a private key.

*\*History and current state of encryption algorithms\**

Mankind has created and employed various encryption algorithms for just about all recorded history. They range from primitive substitution ciphers from hundreds, or maybe thousands of years ago to modern electronic encryption algorithm ciphers. Modern encryption algorithms began with the Data Encryption Standard of the 20th century. DES was later found to be insecure against brute-force attacks, leading to the development of more secure algorithms such as the Advanced Encryption Standard.

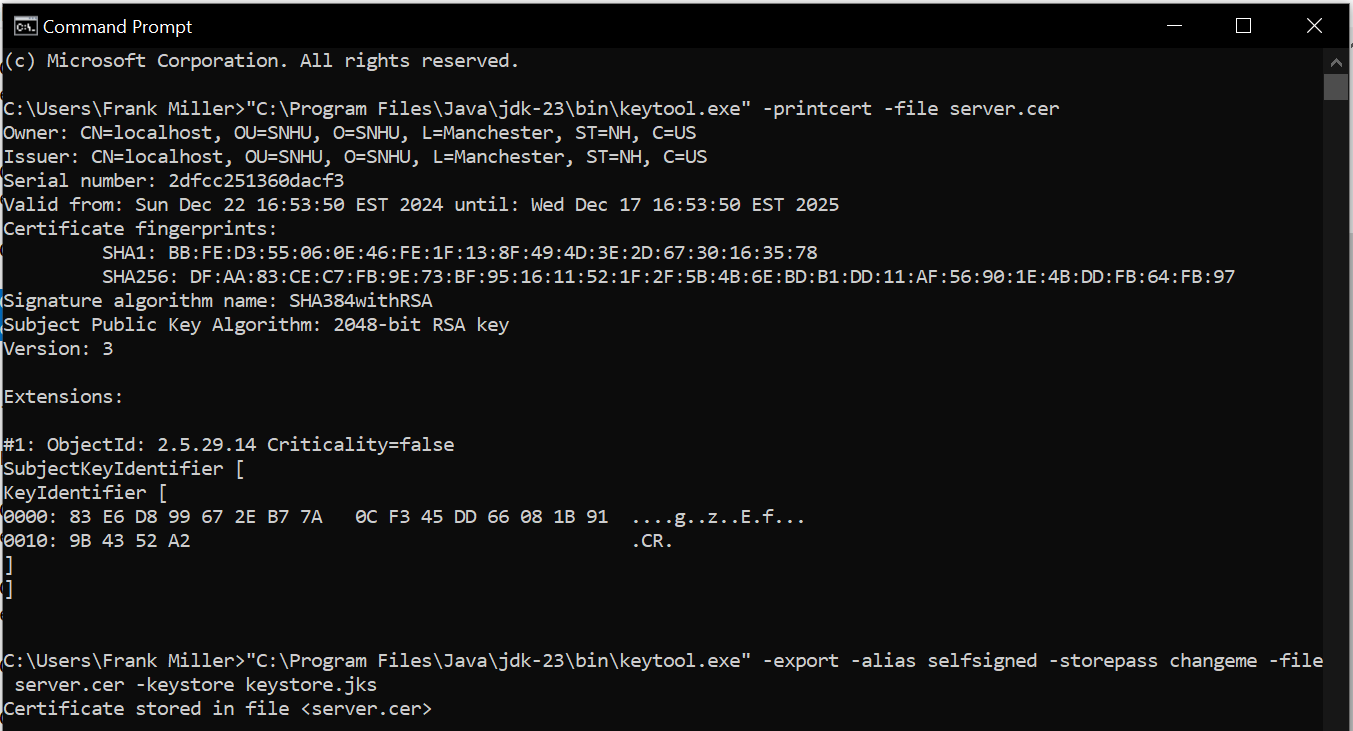
SHA-384 is part of the SHA-2 family, which was published by the National Institute of Standards and Technology in 2001. SHA-1 is currently being phased out in favor of SHA-2 and SHA-3.

Today, encryption algorithms are moving closer to ever more secure and efficient methods. Advanced computing technology may pose a potential threat to many of today’s encryption methods, and as such, the field is moving towards more advanced algorithms.

In conclusion, using SHA-384 for file verification in the form of a checksum would provide a high level of security for Artemis Financial, ensuring the integrity of data during transfer.

## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

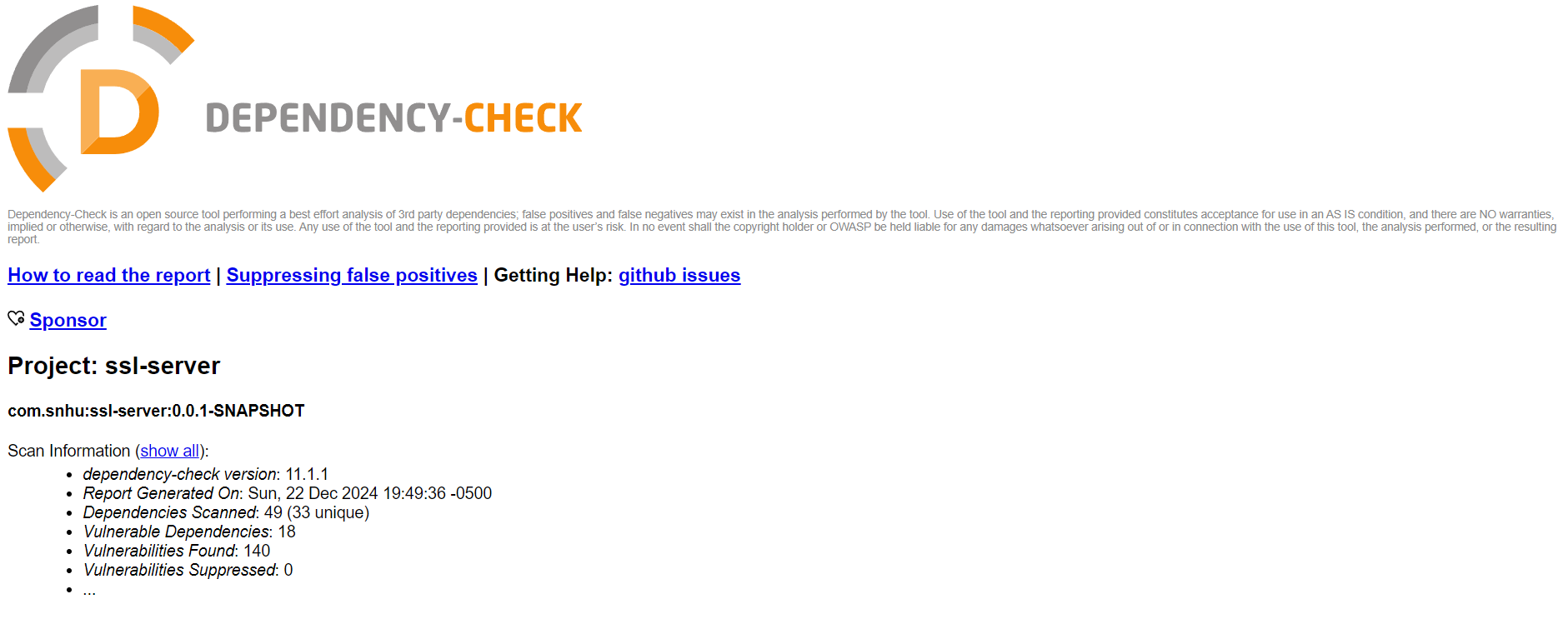
Insert a screenshot below of the checksum verification.

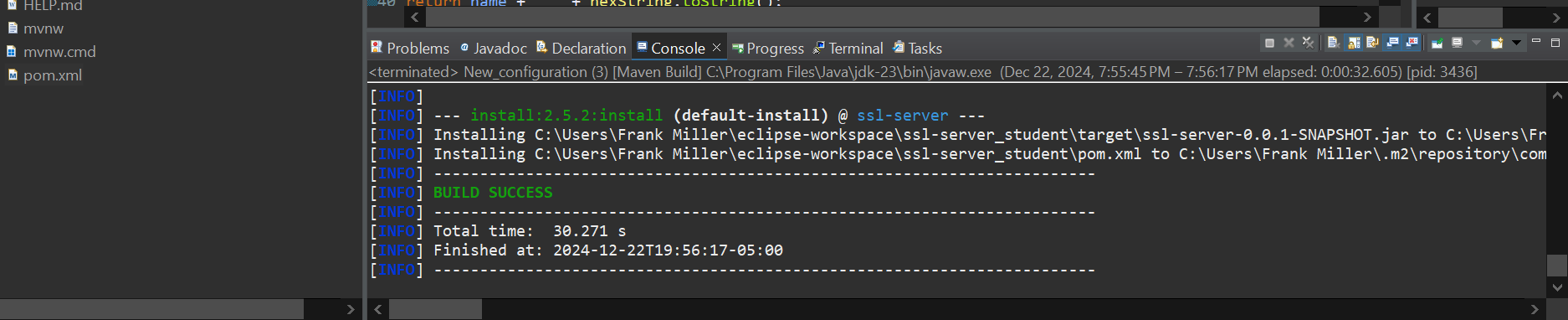
## Secure Communications

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

## Secondary Testing

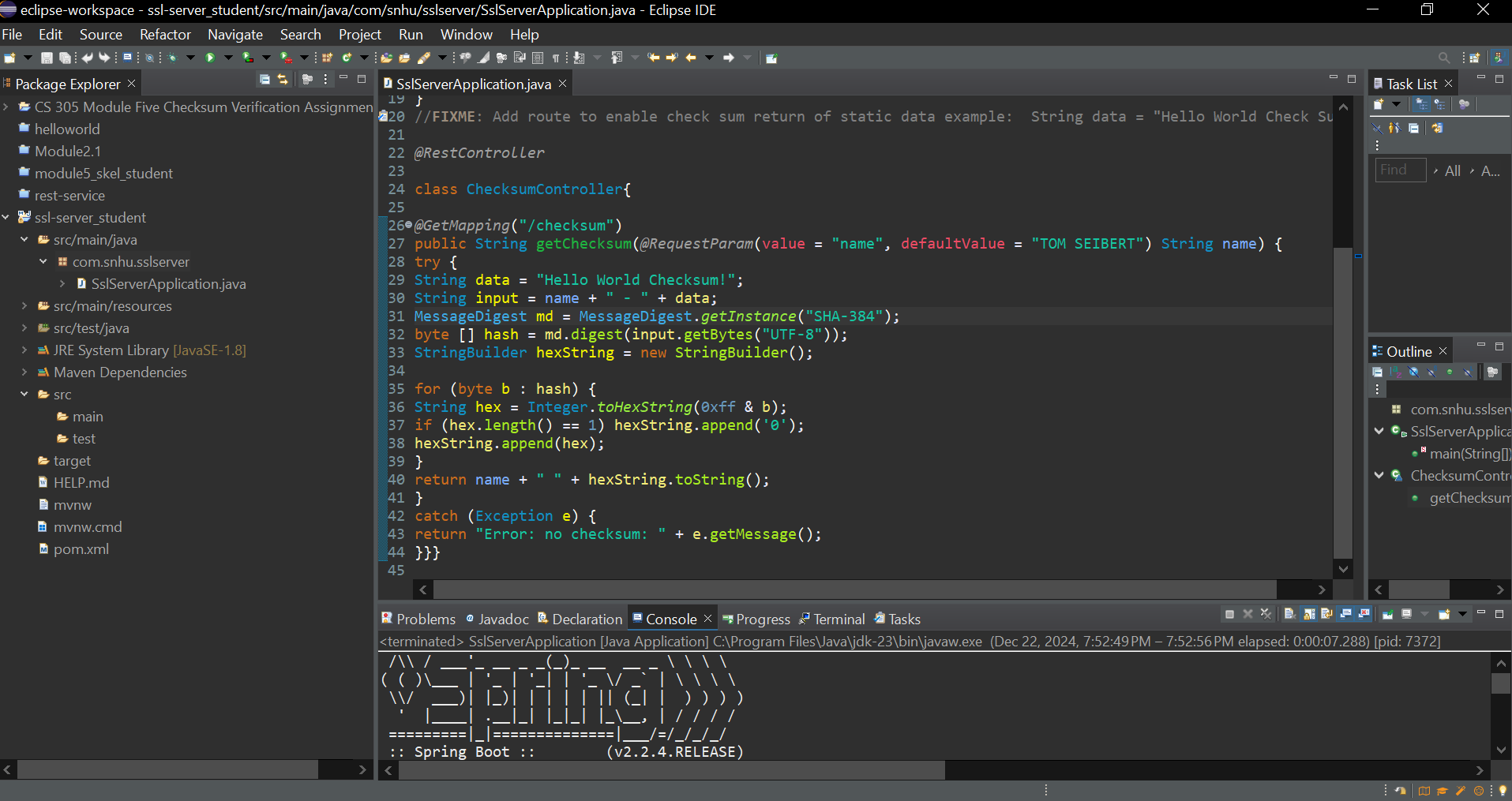
Insert screenshots below of the refactored code executed without errors and the dependency-check report.





## Functional Testing

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



## Summary

In refactoring the code, I focused on security enhancements in the areas of data transfer and data integrity. The Secure Hash Algorithm (SHA-384) was used in the process of checksum validation. This algorithm guaranteed the security of data in its transfer. Also implemented HTTPS which further prevents the potential for various exploitation of vulnerabilities in the future. After these vulnerabilities were addressed, I then implemented an OWASP Dependency Check to further reduce the potentiality of plug-in exploitation. Altogether, these changes amount to a more secure program for Artemis Financial.

## Industry Standard Best Practices

Implementing industry standard best practices are a requirement when dealing with clients such as Artemis Financial, or any other client in the financial industry for that matter. Nothing less is acceptable. The measures which were taken and detailed in the paragraph above will mitigate potential vulnerabilities.