****

# 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** | **10/15/2023** | **Daniel Escobedo** |  |

## 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

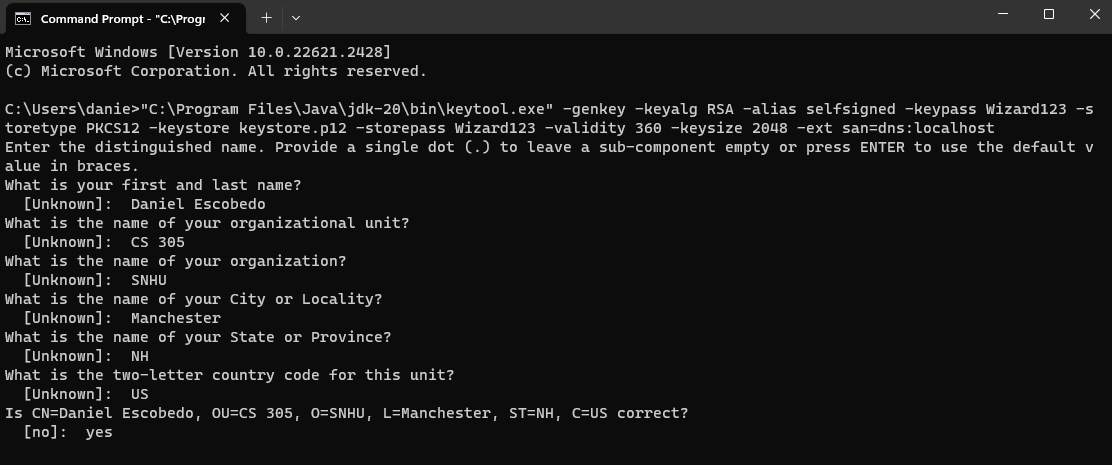
Daniel Escobedo

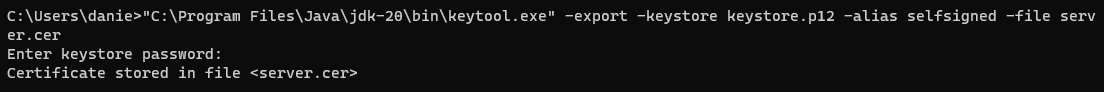
## Algorithm Cipher

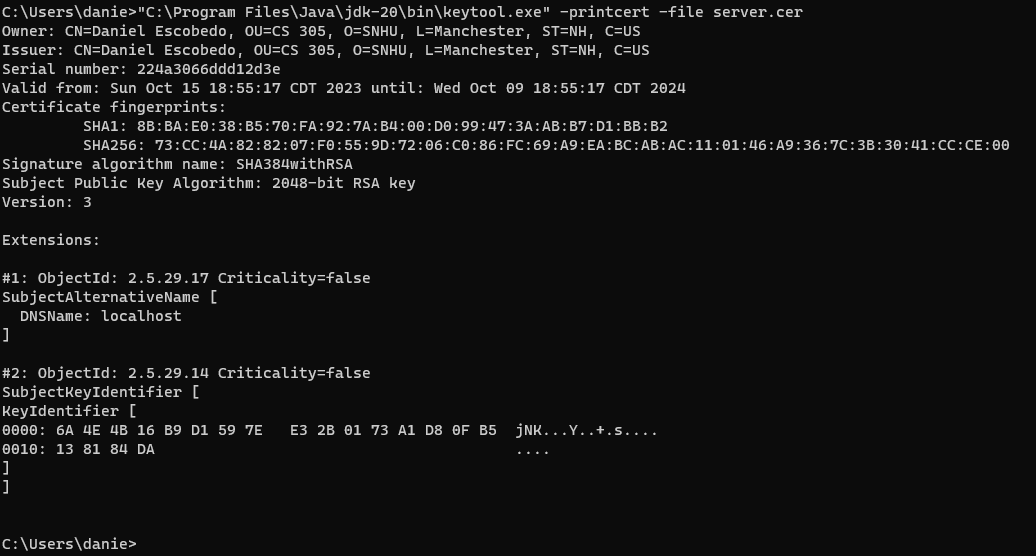
I recommend using the Advanced Encryption Standard (AES) encryption algorithm cipher for Artemis Financials’ web application. AES is a symmetric encryption algorithm that is widely regarded as secure and efficient. It supports key sizes of 128, 192, and 256 bits and is suitable for encrypting large amounts of data. AES works by performing a series of mathematical operations on blocks of data. The specific operations vary depending on the key size, but they all involve substituting, shifting, and mixing the bits of the data. The result is a ciphertext that is very difficult to decrypt without the key. AES does not use hash functions directly. However, hash functions can be used in conjunction with AES to provide additional security. For example, a hash function can be used to generate a unique checksum for each file that is encrypted. This checksum can then be used to verify the integrity of the file when it is decrypted. AES supports key sizes of 128, 192, and 256 bits. The bit level of the key should be chosen based on the level of security required. For most applications, a 128-bit key is sufficient. However, for applications that require the highest level of security, a 256-bit key should be used. AES uses random numbers to generate the initial state of the cipher and to perform some of the mathematical operations. The quality of the random number generator is important for the security of AES. Therefore, it is important to use a strong random number generator that is certified for cryptographic use. AES is a symmetric encryption algorithm, which means that the same key is used for both encryption and decryption. This contrasts with non-symmetric encryption algorithms, which use two different keys, one for encryption and one for decryption. Symmetric encryption algorithms are generally more efficient than non-symmetric encryption algorithms. However, they are also more vulnerable to certain types of attacks, such as key compromise. AES was developed in the late 1990s and was adopted as the US federal government's standard encryption algorithm in 2002. AES is now the most widely used encryption algorithm in the world. AES has been extensively analyzed by cryptographers and has not been shown to have any significant security vulnerabilities. Therefore, AES is a very secure encryption algorithm.

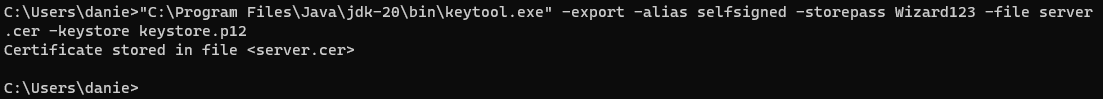
I recommend that Artemis Financial use AES-256 encryption for their web application. This will provide the highest level of security for their client data and financial information. Artemis Financial should also use a strong random number generator to generate the encryption keys and to perform the mathematical operations of AES.

## Certificate Generation



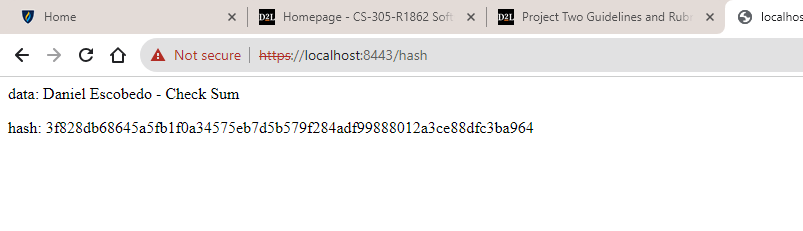






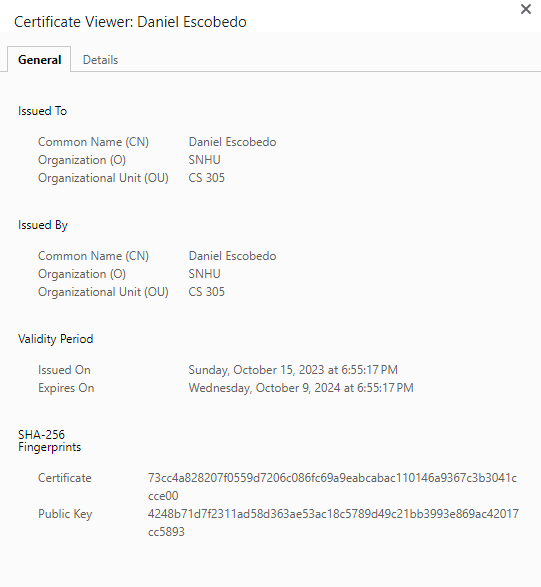
## Deploy Cipher

Insert a screenshot below of the checksum verification.



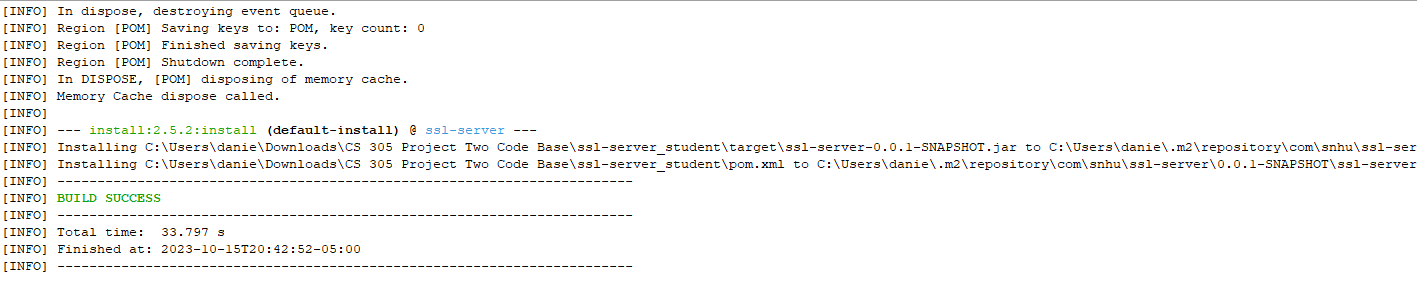
## Secure Communications

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.

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

## Summary

The code was refactored to improve security by adding a verification step to the web application using the SHA-256 cryptographic hash algorithm. This helps to ensure that the data being transferred is authentic and has not been tampered with. I also converted the web application to use the HTTPS protocol. This encrypts all communication between the client and server, protecting it from eavesdropping and tampering. During development the following areas of security were addressed by refactoring the code:

* Data integrity: The file verification step helps to ensure data integrity by comparing the checksum of the data being sent to the checksum of the data being received. If the two checksums do not match, then the data has been tampered with and should not be trusted.
* Communication security: The HTTPS protocol encrypts all communication between the client and server, protecting it from eavesdropping and tampering. This is important for protecting.

After the code was evaluated, I used the following processes to add layers of security to the software application which included, identifying the security vulnerabilities in the existing code, designing, and implementing refactoring changes to address the security vulnerabilities, testing the refactored code to ensure that the security vulnerabilities have been fixed, and deploying the refactored code to production.

## Industry Standard Best Practices

I used industry standard best practices by using the OWASP Dependency-Check Maven plugin to scan the code for known vulnerabilities. This helped to identify any vulnerabilities that may have been introduced by dependencies. I also followed the OWASP Secure Coding Practices Quick Reference Guide to ensure that the code was written in a secure manner. This included using secure coding practices such as input validation, output encoding, and proper error handling.

I believe that applying industry standard best practices for secure coding helps to reduce the risk of data breaches and other security incidents. This can protect the company from financial losses, reputational damage, and legal liability. Secure coding practices can also help to improve the quality of the software application. By making the software more secure, you can reduce the number of bugs and other defects, this can lead to lower maintenance costs and a more reliable product. Applying industry standard best practices for secure coding can also help to improve the company's reputation as a security-conscious organization. This can make it more attractive to customers and investors.