****

# 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** | **04/10/2023** | **Christine Emerson** |  |

## 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

Christine Emerson

## Algorithm Cipher

The encryption algorithm cipher that avoids collisions recommended for the Checksum Verification Project is SHA-256.

Security hash algorithm 256, referred to as SHA-256, is used to verify message, file, and data integrity. This 256-bit hash function converts data into a new, unrecognizable string using a 256-bit key. A string of random characters and numbers, also known as a hash value, is also 256 bits long (Thakkar, 2022).

As a result, the information is protected from external threats. Cryptography experts consider it one of the most secure algorithms ciphers around, and it is arguably impossible to crack. Since most financial institutions also use this cipher, it's best to follow their example. The SHA-256 hash function is compromised in terms of randomness. Hash values are generated by developing a compressed value from our input. In this case, 256 bits represent the number of combinations within the encryption. The lack of predictability makes it almost impossible to hack. As technology advances in 2023, it will always be a battle between protecting information and people trying to abuse it. The need for security and the ability to exploit security breaches will never disappear.

To fully justify why AES is the right choice, I will go into more detail about what is involved with ciphers and why it is the right choice for protecting Artemis Financial. Making a hash function consists of taking the input value and converting it into a compressed one. In essence, a hash function transforms regular data into an unreadable cipher that is difficult to crack even by the most advanced hackers. Encryption algorithm ciphers have bit levels based on the number of characters in their keys. As the AES key length increases, so does the number of encryption rounds it undergoes. AES provides multiple security levels, including 128, 192, and 256 bits.

The AES encryption cipher uses symmetric keys. Basically, the same key is used to encrypt and decrypt data. The keys can be cracked with ease if they are easy to remember, as opposed to if they are random. For this reason, random keys are necessary to ensure security. Nevertheless, this presents a challenge. What is the best way to send the key securely? According to Crawford, "Asymmetric encryption systems solve this problem by securing data using a public key which is made available to everyone. It can only be decrypted by an intended recipient who holds the correct private key” (2018).

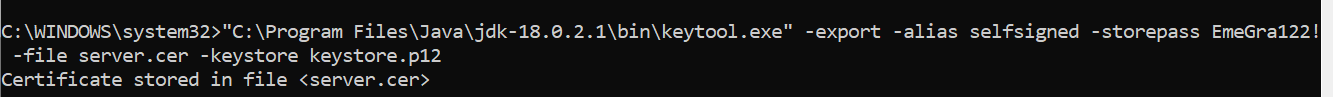
Since asymmetric encryption does not require the sender to know the recipient's private key, it is more effective at securing data in transit. Therefore, the security of data stored on your hard drive is superior when encrypted using symmetric ciphers like AES because of their low computational requirements for this application. As a result, symmetric encryption makes it much quicker to encrypt and decrypt data. In addition, symmetric ciphers are much faster and ideal for encrypting large volumes of data in bulk. Therefore, using asymmetric encryption, such as RSA, is only sensible for a minimal amount of data, such as keys for symmetric cryptography. A hard drive full of data of little use in today's connected world. By using asymmetric encryption, remote key exchanges can be handled securely over the internet, allowing it to be transferred safely over the web. An asymmetric TLS key exchange is advisable to establish a secure connection between your PC and the VPN server to transfer encrypted data securely.

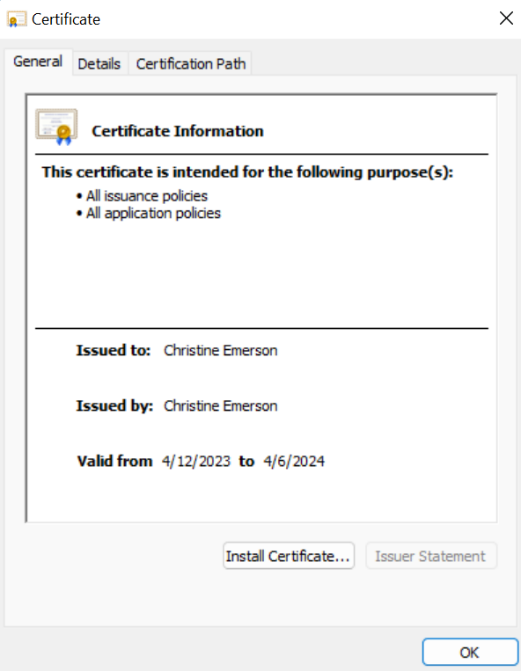
A gold standard in cryptography for AES was DES (Data Encryption Standard), which was developed in the 1970s. Due to its high level of security, it was adopted by the U.S. government. Despite its suitability for the 70s and 80s, this block cipher cannot be used today due to technological advances. The 56-bit key can now be brute-forced within hours thanks to modern computers; after being hacked in the 1990s, it was replaced in the early 2000s with AES**.**With AES, DES is significantly improved and makes the algorithm substantially safer. As discussed earlier, it has variants that feature 128-bit, 256-bit, and 512-bit keys, although, as Axel pointed out, "AES 256-bit encryption is the official standard for U.S. government agencies such as the NSA” (2018). A brute force attack would likely take billions of years. So, you can rest assured that AES is the right choice to secure Artemis Financials’ archive files.

## Certificate Generation

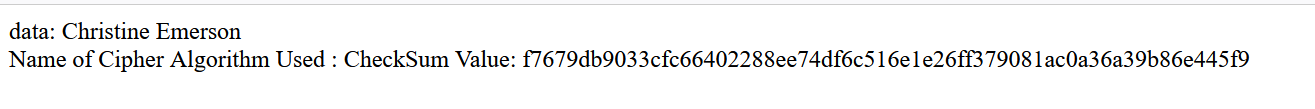
Insert a screenshot below of the CER file.

Text

Description automatically generated

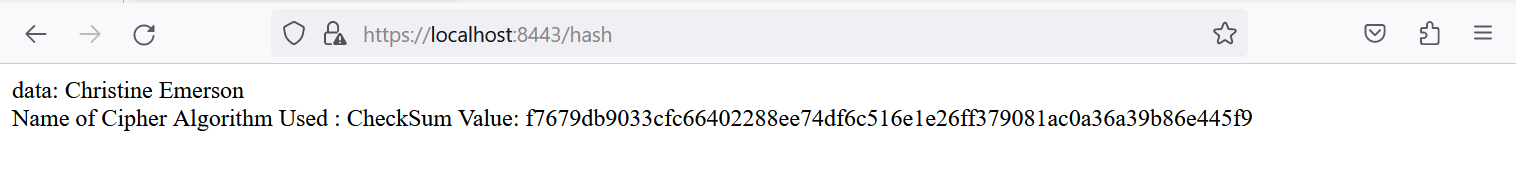


## Deploy Cipher



## Secure Communications

Using the correct navigation path, <https://localhost:8443/hash>, you can view the encrypted data and the checksum value displayed on your browser (as shown below). Since I used a self-signed certificate, the padlock representing a secure connection has a warning. A closer look at the security details reveals that the connection uses SHA256 encryption with TLS 1.3. This warning is due to the certificate being self-signed; a certificate would need to come from an approved Certificate Authority for the browser to remove the warning.



Graphical user interface, text, application, email

Description automatically generated

## Secondary Testing

***pom.xml***

Graphical user interface, text, application

Description automatically generated

***Maven Dependency Report before refactoring code***

**Text

Description automatically generatedGraphical user interface, text, application, Teams

Description automatically generated**

***Refactored Code***

## Graphical user interface, text, application Description automatically generated

***Maven Dependency Report after refactoring code (note no new security vulnerabilities were introduced)***

## Text Description automatically generated

## Graphical user interface, text, application, Teams Description automatically generated

## Functional Testing

***Pom.xml*** ***(upgraded Spring Boot 3.0.5, Maven 8.2.1, and snakeyaml 2.0)***

***\*No errors are present; build is a success.***

Graphical user interface, text, application, email

Description automatically generated

***MavenWrapperDownload.java before refactoring (6 warnings present)***

***Graphical user interface, text, application

Description automatically generated***

***MavenWrapperDownloader.java after refactoring (moved the curly bracket so the resources weren’t closed in the try block)***

***\*All errors are removed***

**Graphical user interface, text, application, email

Description automatically generated**

***Preforming the modifications above resulted in 0 vulnerabilities, errors, and warnings:***

**Text, letter

Description automatically generated**

## Summary

***Discuss how the code has been refactored and how it complies with security testing protocols. In the summary of your practices for secure software report, be sure to address the following:***

* ***Refer to the Vulnerability Assessment Process Flow Diagram. Highlight the areas of security that you addressed by refactoring the code.***
* ***Discuss your process for adding layers of security to the software application.***

This project's main focus areas were cryptography, server/client, and code quality. To improve code quality, security practices and patterns were taken into account. A 256-bit hashing algorithm was implemented in the code to protect sensitive information. The information can be anything; however, in this case, it was just my first and last name. Depending on the circumstances, Artemis Financial may be required to choose financial plans, estimates, corporate documents, client information, etc. Encryption is used before storing this data. As part of this project, a security certificate was integrated with the web application, and TLS encryption was enforced. Communication between the client and server is secured by TLS as well as the connection itself. To ensure secure communication, the data, and TLS connection were encrypted. A thorough review and inspection of all code were conducted for errors and vulnerabilities caused either by my code or by its dependencies. Finally, I ensured there were zero vulnerabilities, errors, and warnings by upgrading Spring Boot, Maven, and Snakeyaml, as well as in the MavenWrapperDownloader.java file moving the curly bracket so the resources weren't closed in the try block causing path traversal vulnerabilities and possible leaking of information.

An SSL certificate was my first layer toward securing Artemis Financial. By using this certificate, users are assured that their communications are encrypted and that the site is secure. It increases user trust as well as data and communication security for Artemis Financial. SHA-256 encryption was the second layer of security added. TLS only encrypts the communication session; you must also encrypt the data. In this way, data is protected since if someone obtained access to the server that stores the data, they could access sensitive information. In addition, encrypting the data prevents anyone from deciphering what has been taken if it is stolen from the server. Finally, HTTPS protocol enforcement was incorporated into this application as the last layer of security, which prevents normal HTTP from being used for the session. Both the server and client must be protected when the user interacts with the site through an encrypted session. Due to this, we can prevent things like man-in-the-middle attacks because old browsers cannot use a poorly secured connection.

## Industry Standard Best Practices

***Explain how you applied industry standard best practices for secure coding to mitigate against known security vulnerabilities. Be sure to address the following:***

* ***Explain how you used industry standard best practices to maintain the software application’s current security.***
* ***Explain the value of applying industry standard best practices for secure coding to the company’s overall wellbeing.***

Regular dependency checks to keep track of new vulnerabilities are best practices for maintaining the current security of a software application—secure coding for integrating any additional modules required. In addition, any input taken from a user needs to be validated—creating a solid API that clarifies what and how to do specific things for the client. Finally, all code changes will be reviewed regularly for any errors or unforeseeable interactions with existing code.

There are many ways to achieve these goals, and Artemis Financial's needs can determine the degree of achievement. As an example, if they find in the future that SHA-256 is not a viable security option, they can upgrade it to whichever best standard is currently available. Rather than trying to blacklist every new threat, input validation can be accomplished by allowing only specific input types and whitelisting available inputs. It is important to carry out dependency checks regularly to identify any new vulnerabilities found by the NIST or other entities that review code for vulnerabilities. The reporting sites can be used to mitigate any new threats, or a module that does not depend on the dependency can be switched to. The practice of secure coding should always be practiced, and the implementation of secure coding will be necessary for the future. Creating the API will require care to prevent users from performing any actions that are not explicitly authorized. The application should operate efficiently and fulfill all the functionality Artemis Financial needs for years to come if these best practices are followed.

**REFERENCES**

Crawford, D. (2018). *How does AES encryption work*. ProPrivacy.

[https://proprivacy.com/guides/aes-encryption](https://proprivacy.com/guides/aes-encryption%20).

*Encryption: The past, present, and future*. (2021). AXEL.

[https://www.axel.org/2021/05/28/history-of-encryption/](https://www.axel.org/2021/05/28/history-of-encryption/%20)