****

# 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 5](#_Toc102040759)

[3. Deploy Cipher 5](#_Toc102040760)

[4. Secure Communications 5](#_Toc102040761)

[5. Secondary Testing 6](#_Toc102040762)

[6. Functional Testing 7](#_Toc102040763)

[7. Summary 9](#_Toc102040764)

[8. Industry Standard Best Practices 10](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **07/12/2024** | **Brad Wells** | **Initial Creation** |

## 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

Brad Wells

bradley.wells@snhu.edu

## Algorithm Cipher

Artemis Financial is looking to encrypt archived files within their system. As a company dealing with its customer’s personal data and financials this is an important step to take in upholding the security of their data and keeping the data confidential in the event of an unforeseen attack where data could potentially be compromised. Encryption has been around since the Roman Empire using ciphers to encode and decode messages and the technology has developed through the years using rotary machines during World War 2, and Data Encryption Standard (DES) being developed in the 1970’s. Advanced Encryption Standard (AES) was established in 2001 by the National Institute of Standards and Technology (NIST) as a replacement for DES due to its stronger security features. It is still widely supported and used today, proving it has stood the test of time and new emerging technology. Based on best practices for security and specifically encryption, I recommend AES. Encryption layered with access control, secure software development, and network security are all best practices to securing data and preventing unwanted access, upholding the CIA Triad of confidentiality, integrity, and availability.

AES is a 128 bit block cipher that supports keys that are 128, 192, and 256 bits. It offers symmetric encryption, meaning that the same key is used to both encrypt and decrypt data in its fixed blocks of 128 bits. The key lengths vary in size, offering more encryption for larger key sizes but this also results in more processing power being required. The varying key sizes also make it scalable to fit the needs of the user while offering security and resiliency. For this cipher, hash functions ensure data integrity by generating a unique has value for the original present data, with changes resulting in a different hash, in turn signaling that the data has been tampered with. Random numbers are used to randomize initialization vectors so the same plaintext will encrypt to a different ciphertext each time due to the random “path” that the data goes on. As long as the key is present that was there to encrypt the data, it can still be decrypted due to the nature of symmetric keys. While there are many advantages to using AES, it does not come without some risks.

Risks involved with the recommendation of AES for Artemis Financial include proper key management and the amount of processing power required. Key management is essential to AES as keys that are secured can be exposed, which would compromise the security of the data should an attacker gain access to the key. Additionally, if a key is lost, since it uses symmetric encryption, the data would not be able to be decrypted until the right key is found. This could lead to loss of data, which is not as severe as compromised data but the act of losing customer data could result in loss of trust as well as having far reaching effects to how the business operates. With variable key sizes, up to 256 bit, large files may require more processing power than is available during encryption and decryption. This can be prevented by using the proper key size to meet the business needs and what processing power is available. While these risks need to be understood and evaluated, AES still is the best choice to support the needs of Artemis Financial and comply with government regulations.

In the financial industry there are many regulations surrounding customer data and the protection of that data. NIST has standards for encryption information contained in Federal Information Processing Standards Publications (FIPS). Particularly FIPS 197 mentions the selection of AES for cryptographic services (National Institute of Standards and Technology, 2001). AES is certainly compliant with government regulations here in the United States as well as being compliant with regulations abroad when looking at the General Data Protection Regulation in the EU.

The AES algorithm cipher can be used by Artemis Financial for archive files as it excels in encrypting data at rest. Implementing AES, each file will be encrypted with a unique key, which is derived from the master key. In code, it is implemented using the Java Cryptography Extension where an instance for AES is created and initialized to the desired size. A method is then used for both encryption and decryption using the same key that is generated where the string is converted to a base64 string where it is then decoded to bytes, and finally decrypted and converted back to a string. While considering what is the “best” cipher, it is really dependent on the use case. AES-256 is incredibly secure, and most would regard it as the best cipher available when balancing both security and performance. The most secure cipher is not always the correct way forward as mentioned earlier, larger key sizes can require more resources to encrypt data so while a cipher may be extremely secure, it also might not fit the needs of the customer if the resources for that cipher are insufficient. This is important to remember regarding AES as Artemis Financial can choose to use 128, 192, or 256 bit encryption depending on the resources available.

## Certificate Generation

Screenshot of the CER file:

A computer screen with text on it

Description automatically generated

## Deploy Cipher

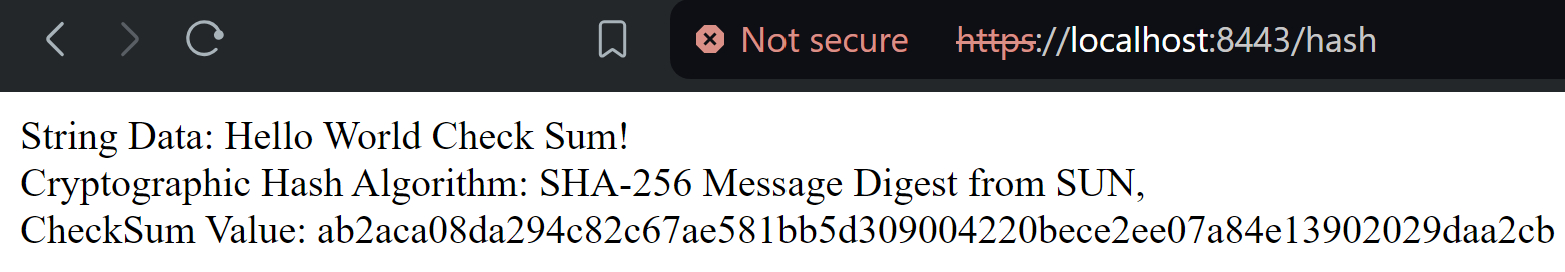
Screenshot of the checksum verification:

A screen shot of a computer

Description automatically generated

## Secure Communications

Screenshot of the web browser that shows a secure webpage with unique string “Hello World Check Sum!”:



## Secondary Testing

Dependency-Check Report before refactoring code:

A screenshot of a computer

Description automatically generated

Dependency-Check Report after refactoring code with no new vulnerabilities introduced:

A screenshot of a computer

Description automatically generated

Successful code run as a Java Application (without errors):

A screenshot of a computer program

Description automatically generated

## Functional Testing

Refactored SslServerApplication.java:

A screen shot of a computer

Description automatically generated

Refactored application.properties:

A screen shot of a computer code

Description automatically generated

Successful code run as a Java Application (without errors):

A screenshot of a computer program

Description automatically generated

Successful Maven build with Dependency-Check Tool:

A screenshot of a computer program

Description automatically generated

## Summary

Referenced files from “ssl-server\_student.zip”

The code for the SSL server application has been refactored to comply with security testing protocols, ensuring a secure application for Artemis Financial. This process involved significant changes to the SslServerApplication.java file, application.properties, pom.xml, and the incorporation of a secure certificate and keystore.

In the refactoring process, areas from the vulnerability assessment flowchart were addressed, including views, models, controllers, data access, services, plug-ins, and APIs. The views were secured by ensuring that data displayed on the web interface is properly hashed using a secure algorithm. The models and data access were indirectly secured through the encryption and hashing mechanisms, ensuring data integrity and confidentiality. The controllers were refactored to include secure hashing of data, using SHA-256 to ensure that any data sent or received by the application is protected against tampering. The services were enhanced by configuring secure communication protocols, specifically by enforcing HTTPS over HTTP to ensure that all data in transit is encrypted. Plug-ins were updated where necessary to their most current version. Although APIs were not explicitly modified in this process, the interactions with the API endpoints were encrypted as a result of using HTTPS. Many of the security improvements made in this code review have multiple order effects, improving security posture beyond the primary component that they intend to address.

Areas of security addressed in the refactoring include encryption of data in transit and hashing of sensitive data. By converting the application to use HTTPS, it ensures that communications between the client and server are encrypted, protecting against eavesdropping and man-in-the-middle attacks. The use of SHA-256 hashing algorithm in the controller ensures that data integrity is maintained, as any changes to the data would result in a different hash value, making tampering detectable*.*

Layering of security in software application began with identifying the areas of vulnerability. The SslServerApplication.java was refactored to include a new controller method that implements SHA-256 hashing. The application.properties file was updated to configure HTTPS, specifying the keystore details to enable SSL. The pom.xml was checked to ensure all necessary dependencies were included and up to date. Additionally, a certificate (server.cer) and a keystore (keystore.jks) were created and configured to establish a secure communication channel.

## Industry Standard Best Practices

In refactoring the SSL server application, industry-standard best practices for secure coding were applied to mitigate known security vulnerabilities and enhance the application's security. To maintain the existing security of the software application, there was more of a focus on ensuring secure communication and employing strong cryptographic hashing, while not introducing additional vulnerabilities. By configuring the application to use HTTPS instead of HTTP, it guaranteed that all data transmitted between the client and server is encrypted, effectively protecting against eavesdropping and man-in-the-middle attacks. Implementing the SHA-256 algorithm for hashing sensitive data ensured data integrity, as SHA-256 is an industry standard cryptographic hash function known for its security and resistance to collision attacks.

The value of applying these industry-standard best practices for secure coding extends beyond just the immediate security improvements. For Artemis Financial, adhering to these practices significantly enhances the company's overall well-being, reputation, and trust in the public space. Secure coding practices help prevent data breaches, which can have severe financial and reputational repercussions. By ensuring that client data and financial information are well-protected, the company builds trust with its clients, which is crucial for maintaining and growing its customer base. The cost of a single data breach for a company, especially one in the financial space can be devasting. Importantly, these practices align with regulatory requirements and industry standards, reducing the risk of legal penalties and ensuring compliance. Applying secure coding practices enhances the security posture within the organization, emphasizing the importance of protecting sensitive information and maintaining high standards in software development.

References

Ellis, S. (2018, Dec 13). *The beautiful hash algorithm.* Medium. <https://steviecellis.medium.com/the-beautiful-hash-algorithm-f18d9d2b84fb>

Monico, J., & Detlefsen, A. (2014). *Iron-Clad Java: Building secure web applications*. McGraw-Hill Education. <https://learning.oreilly.com/library/view/iron-clad-java/9780071835886/?sso_link=yes&sso_link_from=SNHU>

National Institute of Standards and Technology. (2001). *Advanced Encryption Standard (AES) (FIPS PUB 197)*. U.S. Department of Commerce. <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf>

Oracle. (n.d.). *Java Security Standard Algorithm Names*. <https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names>