****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.1** | **17 Apr 2022** | **Alexis Fuerte** |  |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Alexis Fuerte

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

The encryption that I figured would be the best for Artemis Financial is the symmetric encryption style Advanced Encryption Standard (AES). The first thing I considered was whether to use a symmetric encryption style or an asymmetric. I decided on using a symmetric encryption style because Artemis Financial is trying to encrypt data at rest opposed to data in transmission. Out of the names on the list provided, AES is the most commonly used. There are 3 different key length block ciphers that can be used: 128-bit, 192 bit, and 256 bit. Longer bit key lengths are used to protect more sensitive information. AES is known for being user friendly, as well as providing “fast encryption and decryption times” (Rimkiene, 2020). It is also extremely secure compared to its predecessor, the Data Encryption Standard (DES). According to the National Institute of Standards and Technology, AES replaced the DES key in 2002 as the primary encryption standard. The DES key can be deciphered in 362 seconds by a computer nowadays, while the AES key could “take up to 36 quadrillion years” (Rimkiene, 2020). AES is also free to use and easy to install.

For Artemis Financial, I recommend using AES-192 for a few reasons. One, a 256 bit-key length block cypher is usually used to protect highly sensitive information, such as classified (Secret/Top Secret) documents within the government. Although having a very strong cipher is always a good thing, I would note that it may be a little excessive for the information that Artemis Financial would be handling. The AES-192 will still offer extra protection, and act as a good middle ground compared to the AES-128. I also recommend using the SHA-256 hash function because it can help us add file verification through generated checksums. This will help us avoid collisions which helps to increase the level of security.

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.

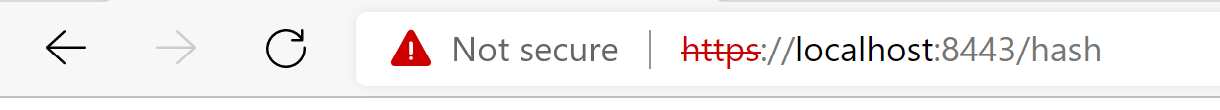
Graphical user interface

Description automatically generated with medium confidence

## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

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



5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

A screenshot of a computer

Description automatically generated with medium confidence

## A screenshot of a computer Description automatically generated with medium confidence

## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.

Graphical user interface, application, Word

Description automatically generated

## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

The following are the vulnerabilities that were tackled:  
API- SHA-256 encryption added

Input validation- Checksum verification added to protect against malicious code

Code Error- Added dependency check

These code refactors should help Artemis Financial meet regulations and keep their customers information safe.

Citations

*Advanced encryption standard (AES)*. GeeksforGeeks. (2022, February 11). Retrieved April 17, 2022, from <https://www.geeksforgeeks.org/advanced-encryption-standard-aes/>

Bernstein, C., & Cobb, M. (2021, September 24). *What is the Advanced Encryption Standard (AES)?* SearchSecurity. Retrieved April 17, 2022, from [https://www.techtarget.com/ search security/definition/Advanced-Encryption-Standard](https://www.techtarget.com/%20%09search%20security/definition/Advanced-Encryption-Standard)

*Data Encryption: The ultimate guide*. Cloudian. (2022, January 24). Retrieved Retrieved April s 17, from <https://cloudian.com/guides/data-protection/data-encryption-the-ultimate-guide/>

Rimkiene, R. (2020, December 11). *What is AES encryption and how does it work?* CyberNews. Retrieved April 17, from <https://cybernews.com/resources/what-is-aes-encryption/>