****

# 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** |
| --- | --- | --- | --- |
| **1.0** | **2/20/2022** | **Antonio Sobalvarro** |  |

## 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

Antonio Sobalvarro

## 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.

Artemis Financial should rely on the best cipher available to be able to protect their client’s information and other sensitive data. The AES, Advanced Encryption Standard, is the safest and most-used cipher that currently exists; the AES-256 is the best of the best. This cipher is able to protect information and data using encryption keys of 128, 192, or 256 bits, which assure protection against many attacks, including brute-force attacks.

Hash functions can be used along with AES to ultimately ensure security. SHA, Secure Hashing Algorithms, are well-known hash functions used by taking in data input in the form of plaintext and then outputting an undecipherable ciphertext, which essentially obscures the original plaintext. In order to view the original plaintext, the recipient must have the same cipher key to unlock the message.

Symmetric keys occur when the same key is utilized for both the encryption and decryption process, which logically means that non-symmetric keys occur when different keys are used for encryption and decryption. Random numbers are used to eliminate any possible patterns that can be used to break a key.

The art of encrypting messages has been practiced for centuries, with some of the earliest examples dating back to early ancient civilizations. Computer-based encryption, however, has been present since the 1970s.

## 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.

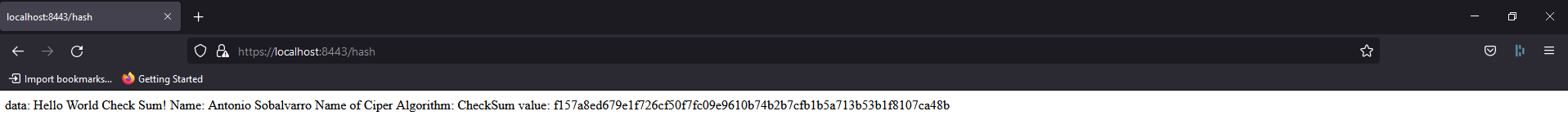
Graphical user interface, text, application, email

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.

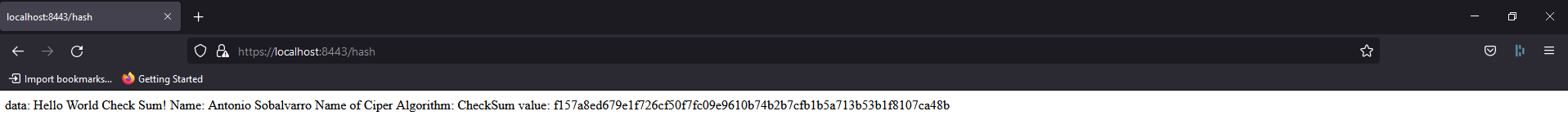
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

## 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

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.

Security was the main focus of this project and in order to fulfill the requirements, checksum has tables were implemented and dependency checks were conducted. I added a self-signed certificate so that HTTPS can be used and allow for an extra layer of protection. Encryption methods were also used by way of a 256-bit encryption algorithm that would prove unbreakable. Layers were added by first creating the correct type of certificate for HTTPS security. This is followed by checking that a hashing function does what is required of it, which is done by using a checksum; this layer hides important data and assures that it cannot be accessed by someone who does not have the right key. Finally, vulnerabilities are patched in order to ensure that the program is protected.

The best practice for maintaining current security of the software application is to stay up to date with any and all changes that pertain to security, from the hacker side to the security expert side. Doing so will ensure that best practices are always being followed and that any vulnerability is immediately patched so as to prevent any attack or access point. I strongly feel that security should not be left to a single person and that it should not be something that a team ignores. This can be achieved by implementing steps at each stage of the process, which in turn makes implementing security measures not feel like a task, but rather, a fundamental part of the process.