****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[\_Toc33111301](#_Toc33111301)

[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** | **6/19/2022** | **Brian Hartong** |  |

## 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

Brian Hartong

## 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 deals with secure information being transmitted through communications, so the encryption algorithm cipher to deploy, given the security needs, would be SHA-256. SHA-256 was created by the National Security Agency in 2001. It is used in several popular encryption protocols such as SSL, TLS and SSH. It is also used in Bitcoin to verify transactions. With the extremely large amount of possible hash values, SHA-256 is virtually collision proof. This algorithm cipher has the avalanche effect, which is where a minor change causes the hash value to change so much that the new value does not appear to be derived from the original (N-able, 2019).

The purpose of a hash function is to take the message that is to be encrypted and create a value of a pre-determined size that cannot be reversed. This value can then be stored in a database to later verify user identities. The bit levels of a cipher are the number of digits in the hash. The more bits that a key contains, the more secure the key becomes (ClickSSL, 2012).

The use of random numbers in ciphers is to create unpredictability in the generation of keys. The two main types of encryptions are symmetric and asymmetric. Symmetric encryption uses a single key for both the encrypting and decrypting of the data while asymmetric uses a different key for each (Cyware Hacker News, 2019).

The earliest evidence of the use of encryption was nearly 4000 years ago in ancient Egypt. Since education was limited to the highest circles of society, encryption was used to obscure information (Tresorit Team, 2022). Encryption today has become much more advanced due to the processing power of computers. This has caused much more complex ciphers to be created to help increase the security of the data (Tresorit Team, 2022).

## 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.

A screenshot of a computer screen

Description automatically generated with medium confidence

## 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, text, application, email

Description automatically generated

## 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.

Graphical user interface, text, application, email

Description automatically generated

## 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

Text

Description automatically generated

A screenshot of a computer

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.

Text

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 areas of security that I addressed by refactoring the code are APIs, Cryptography, Client/Server, Code Error, and Code Quality. The first layer of security was adding a self-signed certificate to the site to allow communication using https, which encrypts the site. The next layer of security that was added was the hash value being generated using SHA-256. The last layer is the handling of code errors and making sure the code follows layout standards. The value that these security layers add to the company’s wellbeing is the client’s knowledge that the information that they are sending is secured, allowing them to trust the company.

A best practice for maintaining the current security of the software application is to set up routine checking for updates to all dependencies used the application. This will make sure that all current security risks are handled and do not put the application at a higher risk of data breaches. Another best practice is to keep watch on any error reporting from the application. These error logs can assist in locating an unauthorized access to the application and allow for faster response in mitigating the potential data breach.

References

ClickSSL. (2012, January 27). *What is bit encryption, and how does it work?* ClickSSL Blog - Information about SSL Certificates & Infosec. Retrieved May 27, 2022, from https://www.clickssl.net/blog/what-is-bit-encryption-and-how-does-it-work

Cyware Hacker News. (2019, November 30). *Exploring the Differences Between Symmetric and Asymmetric Encryption*. Cyware Labs. Retrieved May 27, 2022, from https://cyware.com/news/exploring-the-differences-between-symmetric-and-asymmetric-encryption-8de86e8a

N-able. (2019, September 12). *SHA-256 Algorithm Overview*. Retrieved May 31, 2022, from https://www.n-able.com/blog/sha-256-encryption

Tresorit Team. (2022, January 14). *The history of encryption: the roots of modern-day cyber-security*. Tresorit Blog. Retrieved May 27, 2022, from https://tresorit.com/blog/the-history-of-encryption-the-roots-of-modern-day-cyber-security/