****

# 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** | **10/7/2023** | **April Nixon** |  |

## 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

April Nixon

## 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 is concerned about the security of their web application, particularly in safeguarding sensitive information from potential attackers seeking financial gain. To enhance the security of their communications, it is advisable to implement encryption as a primary security measure. Encryption ensures that any unauthorized access to the stored information would render the data unreadable without the corresponding decryption key.

For securing communication, it is recommended to use asymmetric encryption, where there are two distinct keys involved: a public key for encryption and a private key for decryption. This approach provides a high level of security, especially when information needs to be transmitted externally.

To further bolster security, the SHA-256 cipher algorithm with 256-bit keys is suggested for encryption. SHA-256 offers robust protection due to its use of a 256-bit encryption key, resulting in an exceptionally large number of possible key combinations. Moreover, the SHA-256 algorithm incorporates Java's random number generator, which enhances security by creating a checksum that cannot be reversed, thus verifying the integrity of the files.

In summary, implementing asymmetric encryption with the SHA-256 cipher algorithm and 256-bit keys would provide Artemis Financial with a robust and secure solution for their web application, safeguarding their communications and data from potential threats.

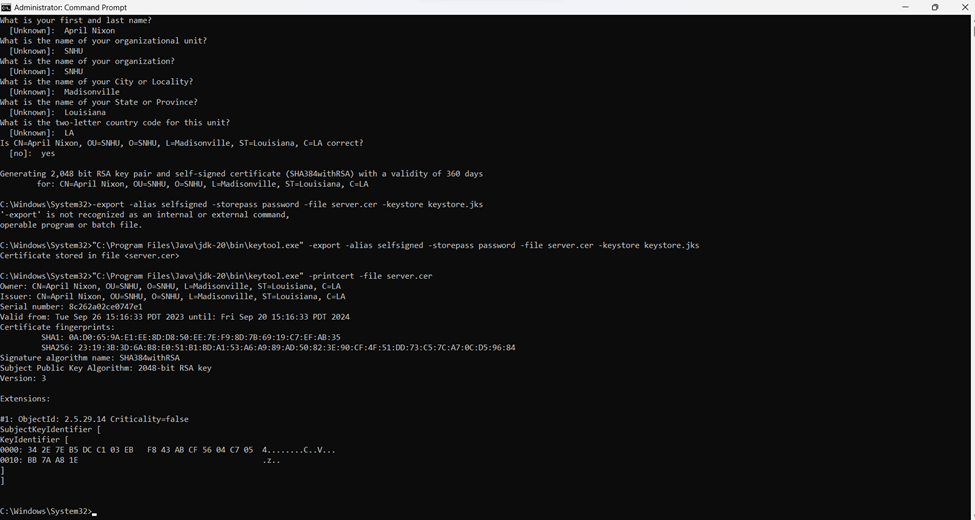
## 2. Certificate Generation

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 computer screen with text on it

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.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

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.

* Showing the HTTPS is working but that my Cert isn’t official because it’s self signed.

A screenshot of a computer

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

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

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.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

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.

A: In order to enhance the security of my code, I've introduced a secure RestController to manage the essential functionalities of my RESTful application. This new ServerController class addresses the issues outlined in the vulnerability assessment diagram. To maximize security, I've opted to utilize the SHA-256 hashing algorithm, known for its robust security and minimal risk of hash collisions.

To maintain the ongoing security of the application, I recommend conducting dependency checks on a monthly or bi-monthly basis. This proactive approach helps us stay informed about potential vulnerabilities, ultimately safeguarding the company's sensitive data. It's also crucial to keep the plugins in the pom.xml file up-to-date to ensure that we're using the latest versions and benefiting from the highest level of security.