****

# 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 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/15/2022** | **Rushil Patel** |  |

## 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

[Insert your name here.]

## Algorithm Cipher

Determine and justify an acceptable encryption algorithm cipher to deploy in light of the security risks. Make certain you address the following:

• Provide a high-level summary of the cipher's encryption algorithm.

• Discuss the cipher's hash functions and bit levels.

• Describe the usage of random numbers, symmetric vs non-symmetric keys, and other concepts.

• Explain the evolution of encryption algorithms and their present status.

To maintain safe connections, Artemis Financial has requested increased protection for their online application. Assuming that the most likely attack vector against a financial institution will be a bad actor aiming to achieve financial benefit by accessing the information maintained, encryption is the best advice. This renders the files unusable to any would-be attacker in the absence of a key. I would propose Asymmetric communication to the company to safeguard communication. That is, the key to encrypt is public, but the key to decode is private. To provide the best level of security, as this information may be communicated outside, I recommend encrypting it with the SHA-256 cipher algorithm and 256-bit keys. SHA-256 encryption gives an outstanding high degree of bit encryption with many potential key combinations with a key length of 256 bits. Furthermore, the SHA-256 technique employs Java's random number generator, guaranteeing that the encryption is extremely secure by generating a non-reversible checksum that certifies the file's validity. The hash function will use the SHA-256 cipher to create a checksum of the provided message.

## 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

## 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.

## 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.

## 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.

1. 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, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## Functional Testing

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

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

Graphical user interface, text, application

Description automatically generated

## Summary

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

## 

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

Refactoring in my code I have added a secured RestController to work as the secure controller for my programs hash RESTful stop. The ServerController class works to match the problems presented by the vulnerability assessment diagram. I additionally chose to work with the SHA-256 hashing cipher as it’s very secure and runs a very small chance at collisions. To best maintain the current security of the application I would suggest once or twice monthly dependency checks of the application to keep the most up to date on potential vulnerabilities this will help to protect the company and their sensitive data. Keeping the plugins within the pom.xml additionally would do well to keep the latest iterations of the plugins running ensuring the highest security.