****

# 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** | **12/07/23** | **William Rosales** | **Revised and filled documentation report. Included screenshots of system.** |

## 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

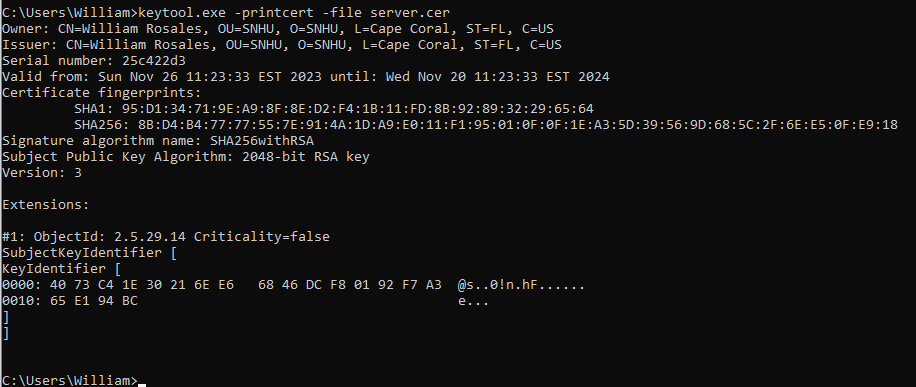
William Rosales

## Algorithm Cipher

The recommended encryption algorithm cipher that I chose is SHA-256. This version of Secure Hash Algorithm allows for strong security features and is able to be implemented in many different types of applications. It is very common to use since it was first introduced in 2001 starting with SHA-1. SHA-256 generates a 256-bit hash that is very resistant to collisions or interferences. Since the bit value is so high, it has very little chance for the same hash output to be shown with two different inputs. This method is used for verification processes, signatures, and for password storage. SHA-256 has become a very dependable choice when it comes to protecting data and maintaining the integrity of a system.

## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

A screenshot of a computer

Description automatically generated

## Secure Communications

A screenshot of a computer

Description automatically generated

## Secondary Testing

A screen shot of a computer program

Description automatically generated

## Functional Testing

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

## Summary

While working with this base code for Project Two, we had to edit some classes and use the Keytool function again. In the SslServerApplication.java file, I created the ServerController class in order to identify issues in the vulnerability assessment diagram. We were also able to generate a dependency report and be able to analyze the file for levels of severity for our dependencies in our system. Choosing the SHA-256 hashing cipher would help minimize the collisions in this system since it would bring down the chances for it significantly. This is important for security. With the dependency checks, there must be frequent visits to generate new reports. These checks are very useful to us even after the system is completed. During the stages of maintenance, we are able to see if a dependency pops up due to an update being available or if something is too outdated. I also self-signed certificates using Keytool in command prompt. This allows for the secure communication for web browsers over HTTPS. The self-signing process includes using an alias and a designated password for the certificate. So, not any person can use that certificate for the program. They would have to use the location of Keytool, the alias name, and the unique password that has been used.

## Industry Standard Best Practices

These practices are for securing code to minimize any security vulnerabilities. We use these practice to not only avoid these scenarios, but to also protect user information and system information. The type of information held in a system should not determine the security effort put into a system. There is sensitive information all around every system, which in this case it has a lot to do with money and investments. Even though as developers, we aren’t working for the purpose of this system, we must use Industry Standard Practices to ensure that the future users are not compromised from future attacks. A big part of maintaining a high level of security with Artemis Financial is to also make their customers feel that there is a high level of security for them too. The sensitive data in the project includes savings, retirement, investments, and insurance. Securing this type of data is crucial to both the company and its users since it can have real world consequences if their information is not secure. This is why us as developers must take additional steps to maintain and update systems in order to keep that high level of security.