****

# 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** | **12/22/2024** | **Alex Surprenant** |  |

## 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

Alex Surprenant

## Algorithm Cipher

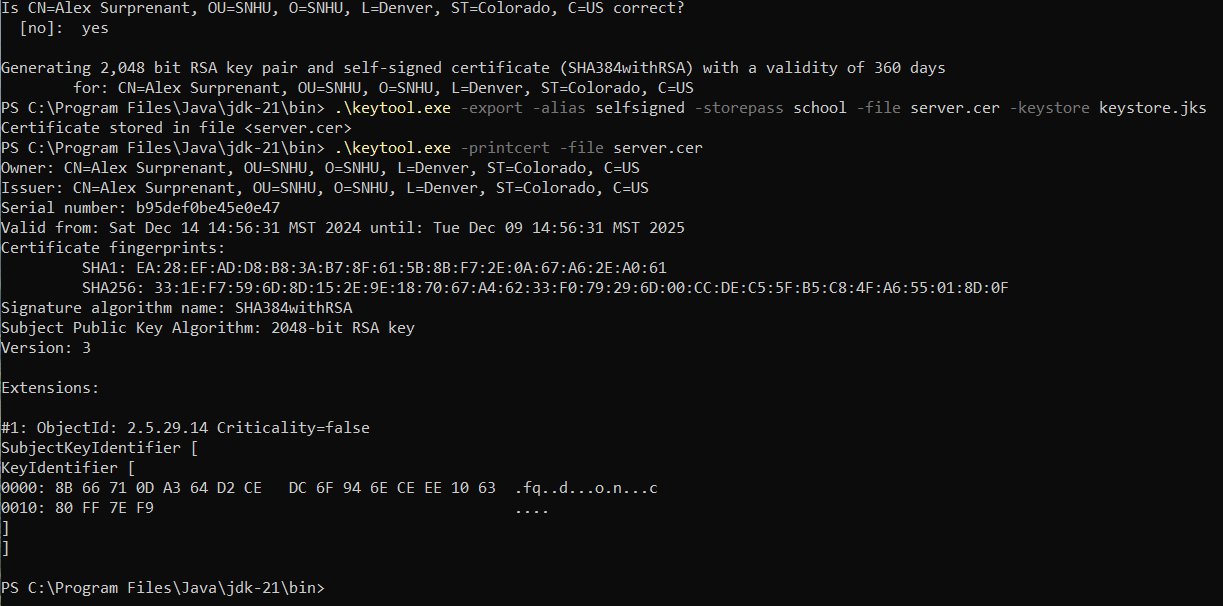
Secure Hash Algorithm-256, or SHA-256 a hash function developed jointly by the NSA and NIST. It is a highly secure algorithm that takes in data input, and returns a hash digest with a bit value of 256 bits, no matter the size of the input, as long as the input is less than 2^64 bits in size.

With a hash function, the security comes from the fact that the output can not be reproduced without the exact original data. Even if the data is off by one single digit or character, the outputted hash value will be completely different. When a user inputs a password, the server that receives the hash value will check to see if there stored value matches, and that’s how the validity is checked, as the same data passed into a hash function will always produce the same output. Using this method, a business like Artemis Financial can be confident in keeping their users data secure.

With SHA-256, in the algorithm, the

## Certificate Generation

Insert a screenshot below of the CER file.



A screenshot of a certificate

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

Description automatically generated

## Secure Communications

Insert a screenshot below of the web browser that shows a secure webpage.

A screenshot of a computer

Description automatically generated

## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

A screen shot of a computer program

Description automatically generated

## Summary

Overall I really struggled with the checksum portion of the project. I feel confident in the functionality of the code, however there was never a point when I could get it to run, and I am sure it has something to do with a different portion of the project or how my project is setup, or the dependencies. Everything else seemed to work ok though.

## Industry Standard Best Practices

[Insert text.]