****

# 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** | **4/21/2024** | **John Schatzl** |  |

## 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

John Schatzl

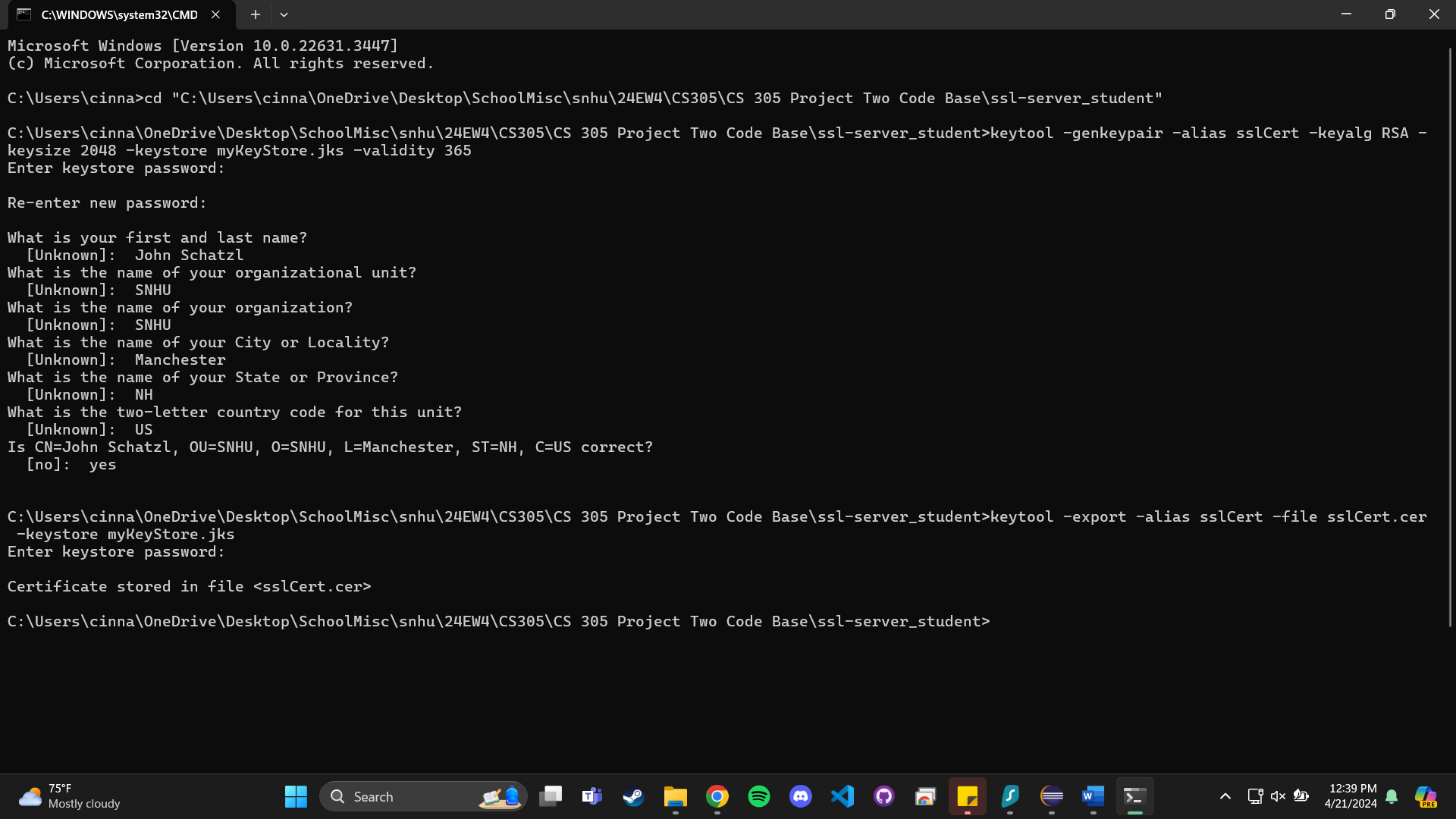
## Algorithm Cipher

**Recommendation:** AES

* Offers robust encryption with key lengths of 128, 192, or 256 bits, it is fast and optimized for various software and hardware platforms, and it is widely used, even by the US Government, regularly maintained, and easily adoptable.

## Certificate Generation

Insert a screenshot below of the CER file.



## Deploy Cipher

## Insert a screenshot below of the checksum verification.

## A screenshot of a certificate 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

A screenshot of a computer screen

Description automatically generatedInsert screenshots below of the refactored code executed without errors and the dependency-check report.

## Functional Testing

A screenshot of a computer screen

Description automatically generatedInsert a screenshot below of the refactored code executed without errors.

## Summary

After referencing the vulnerability assessment process flow diagram to guide my refactoring with security in mind, I made the most out of my refactoring.

I implemented AES to encrypt sensitive data in transit and potentially at rest, enabled HTTPS and configured my application to user a self-signed certificate, which flags most modern browsers, but nonetheless, secures data within this protocol. I also added error logging to commit to more secure coding practices and ensure that there is no error that isn’t caught.

## Industry Standard Best Practices

I have scrutinized the OWASP Top 10 list of common web application vulnerabilities, and my refactoring has addressed as many of these concerns as was applicable to the specifications provided.

As well, the integrated OWASP dependency-check plugin is now in place to proactively find and secure vulnerabilities within the project’s dependencies for tracking and securing vulnerable points of the application and ensuring all is well.