****

# 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** | **08/14/2024** | **Cody Adams** |  |

## 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

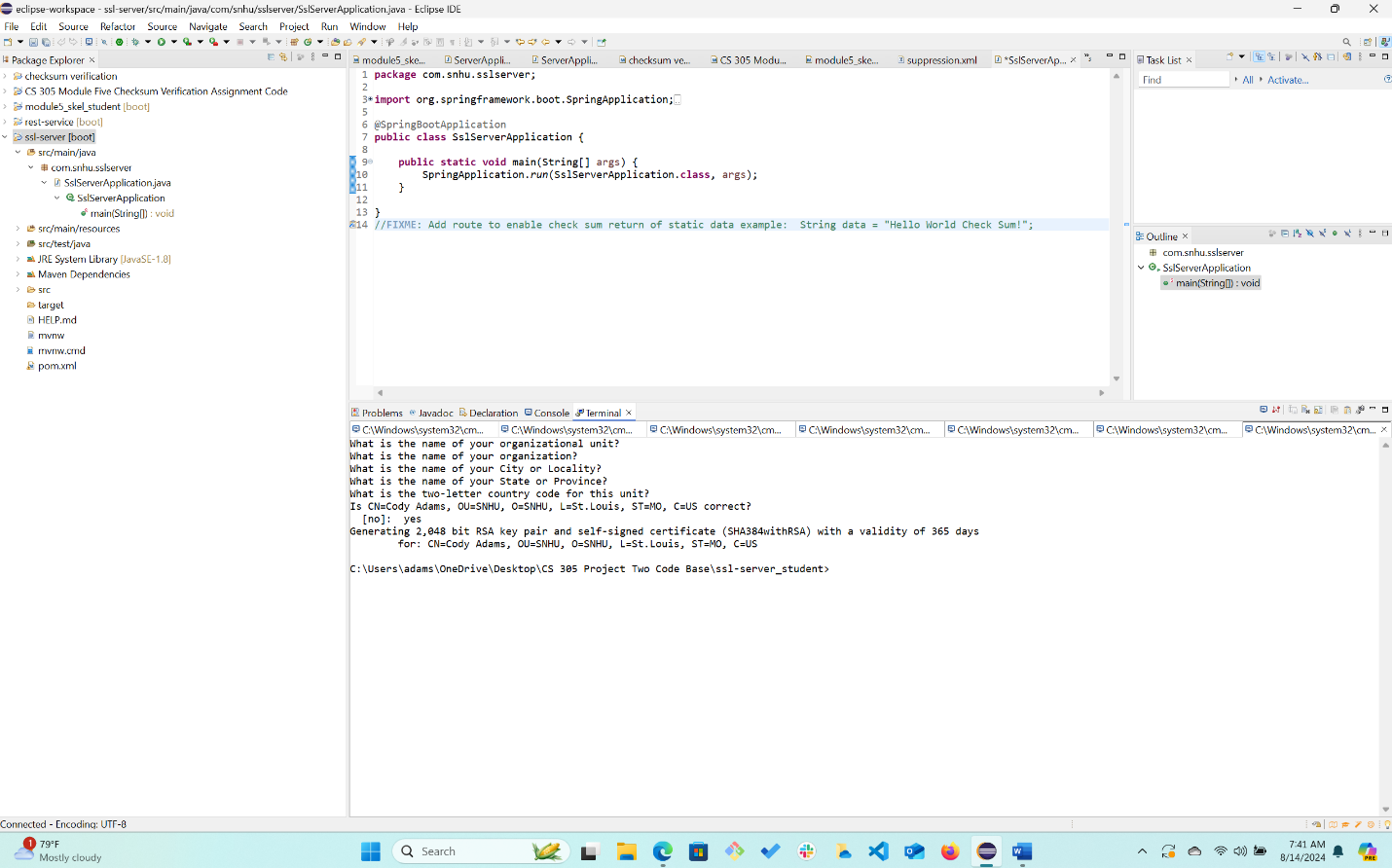
Cody Adams

## Algorithm Cipher

AES

## Certificate Generation

Insert a screenshot below of the CER file.



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer code

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

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

A screenshot of a computer

Description automatically generated

Focused on addressing CRITICAL dependencies. I updated the dependencies and added them into the pom.xml file.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

Overall I focused on using AES as my cipher. This allowed for the application to utilize encryption. The ChecksumController class was added to create a path to our (/hash), which gives us our encrypted checksum based off our my static data. In this case I used my name (Cody Adams) as asked in the assignment. The AES cipher allows us to encrypt data and transmit that data which is really handy because it can help keep our data secure.

Another thing I focused on was after having a successful maven build, was to navigate to our dependency check report. Here I was able to see vulnerabilities. I did not address every single one of the vulnerabilities. I focused in on a couple of the CRITICAL vulnerabilities such as datablind, log4j-api, logback-core, and tomcat-embed-core. I was able to add the updated dependencies within the pom.xml file.

Another thing that I updated was the Spring Boot framework version to 2.7.6. This was able to address other vulnerabilities, and creates a more solid foundation of security within our application.

I added a HTTPS configuration as well that will ensure that the data that is being transmitted is indeed encrypted and protected. The data is moving from server to client , and this security will help against people trying to get access to the data through attacks.

This application was tested for static and functional security compliance. After testing, I have witnessed the application running correctly with all of the new security measures established as mentioned above.

## Industry Standard Best Practices

Like I mentioned above I utilized AES Encryption. This is a widely used encryption , and allows us to encrypt transmitting data which makes it hard to intercept, and also hard to decipher.

On top of AES I used another really important industry standard, and that is dependency management. This ensures that the projects dependencies are up to date, and ensures that the newest and best up to date dependencies are being used to decrease overall vulnerability. Keeping dependencies up to date is a foundational industry standard best practice.

To identify these potential vulnerabilities I used the OWASP dependency check tool. Once I did this I was able to identify CRITICAL vulnerabilities. I was also utilizing Spring Boot Framework, in which I updated the version , giving our application a solid more up to date foundation to start on. This makes our application more stable because it has newer ways of operating that might be different than a previous version. The newer framework might have critical patches, newer vulnerabilities, bug fixes, performance enhancements, and framework compatibility. Another overlooked one is community contributions, and support using the framework.

Overall after testing this application in different areas, and utilizing industry standard best practices, the application has been refactored to be more secure than it was before.