****

# 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 5](#_Toc102040760)

[4. Secure Communications 6](#_Toc102040761)

[5. Secondary Testing 7](#_Toc102040762)

[6. Functional Testing 7](#_Toc102040763)

[7. Summary 8](#_Toc102040764)

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

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **4/19/25** | **Drew Sibila** |  |

## 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

Drew Sibila

## Algorithm Cipher

For this project, I used the SHA-256 algorithm. SHA-256 is a hashing function that takes input data and turns it into a 256-bit hash value. This helps confirm that the data hasn’t been changed or messed with. It’s part of the SHA-2 family, which is still widely trusted today. It’s a one-way function, meaning once the hash is made, you can’t reverse it. I used it in the checksum method to make sure the input string gives a unique output.

## Certificate Generation

Insert a screenshot below of the CER file.

To secure the application, I created a self-signed certificate using the Java Keytool. I filled in the info like my name, org, and location, then exported the .cer file to use in the app. This lets the application prove its identity and run over HTTPS.

A black screen with white text

AI-generated content may be incorrect.A black screen with white text

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

I wrote a method in the ChecksumUtil class to generate a SHA-256 hash for a string input. Then I tested it by printing the checksum to the console. This helps verify data integrity.A screen shot of a computer

AI-generated content may be incorrect.

## Secure Communications

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

I moved the keystore file into the resources folder and edited the application.properties file to switch the app from HTTP to HTTPS. When I ran the app and went to the browser, it showed the secure message with my name. Even though the browser says “not secure,” that’s just because it’s a self-signed cert.A screenshot of a computer

AI-generated content may be incorrect.

## Secondary Testing

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

I ran the OWASP Dependency Check using Maven. It scanned all the libraries and found a few vulnerabilities from third-party dependencies. These came from Spring and Jackson libraries, which are outside the code I added. My own code didn’t introduce any new vulnerabilities, so the refactor passed the check.A screenshot of a computer

AI-generated content may be incorrect.

C:\Users\drew2\Downloads\modseven\.metadata\ssl-server\_student\target\dependency-check-report.html

## Functional Testing

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

I ran the program and verified that it executed with no errors. The console printed the input string and the generated SHA-256 hash just like it should.A screen shot of a computer

AI-generated content may be incorrect.

## Summary

I updated the application to make it more secure by adding SHA-256 hashing, setting up HTTPS, and creating a self-signed certificate. I also ran an OWASP dependency scan to make sure no new vulnerabilities were added. All my updates followed secure coding standards and worked without errors.

## Industry Standard Best Practices

I followed security best practices by hashing data with SHA-256 and switching from HTTP to HTTPS. Using HTTPS encrypts the communication, which helps keep user data safe. Running the OWASP scan was also part of best practices so I could catch any library vulnerabilities. These kinds of steps are what developers are expected to do to build safer software.