****

# 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** | **10/14/2022** | **Cameron Lee** |  |

## 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

Cameron Lee

## Algorithm Cipher

The cipher that I choose for this project is the Advanced Encryption Standard (AES). There are a few reasons for choosing this cipher. It is fast, secure, and backed by the United States government. The AES cipher is a symmetric one, meaning that it uses the same key to encrypt and decrypt the data. There are three different lengths of the AES encryption cipher: 128-bit, 192-bit, 256-bit. This gives options regarding the level of encryption, with each increasing bit length sacrificing lack of computing power. Using this cipher would take an input string and encrypt it into random numbers based on the amount of bit length chosen. With malicious attacks seemingly being more common through technology today, the need for encryption ciphers has grown.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text, application, email

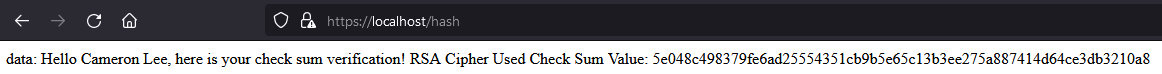
Description automatically generated

Text

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

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

Text

Description automatically generated

## Secondary Testing

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

**Run Proof – 0 Errors**

Text

Description automatically generated

**Dependency Check Report**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email, website

Description automatically generated

## Functional Testing

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

Text

Description automatically generated

## Summary

The code has been refactored in several different areas of this project. The first, involved adding a way to display a set data on a localhost server. This code was done in a previous assignment, but after feedback, I made some changes in my bytesToHex function that allowed the localhost server to display my unique string and the check sum verification with the hash value. From there, I ran a dependency check report and got several vulnerabilities. This is where I really struggled in the project. A lot of the vulnerabilities were due to older versions of maven dependencies, but even after updating several to the latest versions by importing JAR files that I downloaded through their respective creators I still got the same vulnerabilities popping up on the dependency check report. Despite this, I did have the program running with zero failures or errors. In the future I would like to figure out what was preventing the vulnerabilities from going away by successfully updating the versions. The biggest area of security that I addressed was the code quality. This is mentioned in the vulnerability diagram because bad code quality can lead to security issues. I feel like using the feedback from my previous milestone helped me create a better class to handle the functions.

## Industry Standard Best Practices

I feel like I used industry standard best practices to maintain not only the software’s security, but also it’s ability to run without errors, and to convert the local server to a secure webpage. Between that, creating a secure certificate that is password locked, and a keystore with a separate password, this entire project revolved around using solid practices to provide a secure final product. This would contribute to a company’s wellbeing immensely as most companies nowadays deal with sensitive customer information. Having good industry standard security practices can protect that information.