****

# 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** | **04/20/2024** | **Ryan Summers** |  |

## 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

Ryan Summers

## Algorithm Cipher

The Advanced Encryption Standard algorithm cypher, or AES for short, should be used for Artemis Financial’s needs. AES is a symmetric block cypher that breaks down chunks of data, then encrypts each chunk. It produces a fixed size output, called a hash. Since it is symmetric, both encryption and decryption use the same key. Asymmetric encryption uses two keys, one public, and one private. Normally the public key can be shared and used for encryption. The private key is kept secret. It is used for decryption. AES has different levels of encryption, 128, 192, and 256-bit. The number represents the length of the key. Historically encryption algorithms have been broken or cracked. Data Encryption Standard (DES) is an example of an outdated encryption algorithm. It was created in 1973. It used a 56-bit key and 64-bit blocks for encryption. AES was created by the National Institute of Standards and Technology (NIST) in 1997. The lowest level of AES encryption is 128-bit. This makes it much stronger since there are more passes involved in the encryption. It is also a longer key. AES offers 256-bit encryption. It is considered uncrackable by today’s standards.

## Certificate Generation

Insert a screenshot below of the CER file.

A computer screen shot of a black screen

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

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

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

The code I added or modified did not introduce any new security vulnerabilities. I was able to eliminate all but one existing security vulnerability by updating the pom file to use the current versions of Spring Boot, 3.2.5 and the OWASP dependency check plugin, 9.1.0. After this there is one remaining vulnerability, CVE-2023-35116. It shows the vendor states this is a false positive. I have suppressed it to reflect a false positive.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

## Summary

For this project I started out by generating a security certificate to enable secure data transmission. Then I refactored the SslServerApplication.java file. I coded a SHA-256 cryptographic hash algorithm. I moved to the application.properties file at this point. I inputted the information for the SNHU certificate used in module 5. The checksum was viewable by going to https://localhost:8443/hash, however https was crossed out, indicating the connection was not secure. In order to get a secure connection, I needed to use the certificate I created. I added the .cer and .jks files to src/main/resources. Then refactored the application.properties to point to my certificate. I added the certificate to my browser as well. After doing this I was able to see a secure connection on https://localhost:8443/hash. I ran a dependency check and found 77 vulnerabilities. To remedy these, the .pom file needed to be updated. Most of the identified issues were due to out-of-date software. I updated the Spring Boot version to the current version 3.2.5 and updated the OWASP version to 9.1.0. This left 1 vulnerability, CVE-2023-35116. Looking into it showed the vender states it is a false positive. I then created a suppression.xml file and added the false positive to it. I had to refactor the .pom file again to point to my suppression.xml file so the false positive would not be flagged. After running the dependency check again, all the vulnerabilities showed to be taken care of.

## Industry Standard Best Practices

Industry standard best practices were kept in mind while doing the whole project. Keeping security in mind from beginning to end is important. Using SHA-256 is one of the most secure means of encryption publicly available. Then using a security certificate to enable https is another good practice. It identifies the site as legitimate. It prevents imposter sites from using a man in the middle attack. Running the OWASP dependency check plugin is also important for best practices. It makes sure there are no known vulnerabilities and provides an opportunity to address any that are found. The combination of these things allows for a secure product.