****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **August 14, 2021** | **Alyssa Navarro** |  |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Alyssa Navarro

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

The best thing I would recommend would be to use AES encryption algorithm to meet Artemis Financial’s needs. Since AES is one of the newest and strongest encryption algorithms in the industry right now it would make sense for the company to use it. It was specifically designed to protect classified documents and the encryption is extremely secure. There are up to 256 encryption keys which makes it hard for someone to try and hack into the document if it is that secure.

AES takes any regular text like a note or an email then when it is encrypted it becomes all jumbled up. No one will be able to read it or even put it together unless they have the encryption key. The encryption key is given to those authorized to view the document and its contents. Once the correct key is entered the text goes back to normal and is readable for the user. Both parties the sender and receiver must have the same encryption key for this to work.

Keeping the last statement in mind, this would be a great tool for a company that deals with sensitive information. Using AES can prevent people from snooping into other people’s information such as their bank accounts or something else along those lines. AES is the most modern form of encryption and is arguably the best option.

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.

Text

Description automatically generated

The screenshot above shows the certificate for the keystore file. The certificate provides the information provided for the keystore and the start and end time for the valid period.

## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.

Graphical user interface, text, application

Description automatically generated

## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

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

Graphical user interface, text, application, website

Description automatically generated

## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

Text

Description automatically generatedText

Description automatically generated

The above screenshots display the refactored code without error as well as the pom file to highlight the change that needed to happen since there was dependency issues. To fix them I needed to update the POM file to the latest version of the dependency check.Graphical user interface, application, Teams

Description automatically generated

## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.

[Insert screenshot(s) here.] Text

Description automatically generated

## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

Upon review I found that these areas are necessary to review:

* APIs
* Input Validation
* Code
* Cryptography

These are all important parts to make sure the code being worked on is fully functional. This is especially important when there needs to be a refactored code and a checksum. To add layers of security first there is a need to know what we are working with. This means getting to know the product and what the product is supposed to do. Then the data needs to be identified by what type of data it is and how the data should be transferred effectively. Doing these things helps to determine what areas of security need to be increased or what is needed for the code to run. There are numerous ways to maintain software security since it is being used everyday and implemented into so many things. I think that one of the best things is to keep the software up to date and make sure there aren’t any patches that need to be downloaded. This will prevent any vulnerabilities to be exploited or even identify some that haven’t been found yet.