Competency

In this project, you will demonstrate your mastery of the following competencies:

- Write secure communications through the application of current encryption technologies and techniques
- Design and implement code that complies with software security testing protocols

Scenario

You work as a developer for a software company, Global Rain, an engineering company that specializes in custom software design and development for entrepreneurs, businesses, and government agencies around the world. At your company, part of your mission is that “Security is everyone’s responsibility.”

At Global Rain, you are tasked with working with a client, Artemis Financial. Your client is a financial consulting company that develops individualized financial plans for savings, retirement, investments, and insurance for their patrons.

As an important step in Artemis Financial’s desire to modernize its operations and as a crucial part of the success of their custom software, they want to implement and apply the most current and effective software security. Artemis Financial has a public web interface and is seeking Global Rain’s expertise in taking steps to protect their client data and financial information.

Specifically, Artemis Financial is seeking to add a file verification step to their web application to ensure secure communications. When the web application is used to transfer data, they will need a data verification step in the form of a checksum. You have been asked to take their current software application and add secure communication mechanisms to meet their software security requirements. You will deliver a production quality integrated application that includes secure coding protocols.

Directions

You are tasked with examining Artemis Financial's software to address any security vulnerabilities. This will require you to refactor the code base to add functionality to meet software security requirements for Artemis Financial's application. Specifically, you will need to follow the steps outlined below to facilitate your findings, address and remedy all areas, and document your work in the Practices for Secure Software Report.

- ☐ **Algorithm Cipher:** Review the scenario and the Supporting Materials. **Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning**. In your Practices for Secure Software Report, be sure to address the following:
    1. Provide a brief, high-level overview of the encryption algorithm cipher.
    2. Discuss the hash functions and bit levels of the cipher.
    3. Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
    4. Describe the history and current state of encryption algorithms.
- ☑ ~~**Certificate Generation: Generate appropriate self-signed certificates** using the Java Keytool, which is used through the command line.~~
    1. To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file in your Practices for Secure Software Report.
- ☑ ~~**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**.~~
    1. Include a screenshot of the checksum verification in your Practices for Secure Software Report. The screenshot must show your name and a unique data string that has been created.
- ☑ ~~**Secure Communications:** In the application.properties file, refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code. Then once the server is running, you can **verify secure communication** by typing **https://localhost:8443/hash** in a new browser to demonstrate that the secure communication works successfully.~~
    1. Provide a screenshot of the web browser that shows a secure webpage and include it in your Practices for Secure Software Report.
        i. Generate certificate using these instructions
        ii. Copy the keystore file (.jks) to the /target/classes folder
        iii. Include the following in the "application.properties" file:
            - server.port=8443
            - server.ssl.key-alias=selfsigned
            - server.ssl.key-store-password=angelina13!
            - server.ssl.key-store=classpath:keystore.jks
            - server.ssl.key-store-type=jks

- ☑ ~~**Secondary Testing: Complete a secondary static testing of the refactored code using the dependency check tool provided below 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.~~
    1. Include (1) a screenshot of the refactored code executed without errors and (2) a screenshot of the report of the output from the dependency check static tester in your Practices for Secure Software Report.
- ☑ ~~**Functional Testing: Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.**~~
    1. Complete this functional testing and include a screenshot of the refactored code executed without errors in your Practices for Secure Software Report.

What if I receive errors or new vulnerabilities?

You will need to iterate on your design and refactored code, address vulnerabilities, and retest until no new vulnerabilities are found.

- ☐ **Summary: Discuss how the code has been refactored and how it complies with security testing protocols**. In the summary of your Practices for Secure Software Report, be sure to address the following:
    1. Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
    2. Discuss your process for adding layers of security to the software application and the value that security adds to the company's overall wellbeing.
    3. Point out best practices for maintaining the current security of the software application to your customer.

What to Submit

To complete this project, you must submit the following:

Practices for Secure Software Report

Use the template provided and submit one comprehensive report of the steps you have taken to increase the layers of security in Artemis Financial's software application. You will also submit the zipped project files that contain the refactored code. See below for more information. Include details about the code files being "attachments" to the completed report.

CS 305 Project Two Refactored Code Base.zip

Refactor the code provided in the Supporting Materials section. Be sure to zip the refactored code into one zipped project folder that contains all files associated with Artemis Financial's software application. Submit the zipped project folder in addition to the Practices for Secure

Software Report. Include details about the code files being “attachments” to the completed report.

Supporting Materials

The following resource(s) may help support your work on the project:

Java Software Application: CS 305 Project Two Code Base.zip

Refactor the code to meet the software security needs of your customer. You will need to submit your refactored code.

Website: Java Security Standard Algorithm Names

Refer to the standard list of algorithm ciphers provided by Oracle for recommending an appropriate encryption algorithm cipher to use in this project.

Website: How to Create a Self Signed Certificate Using Java Keytool

Access the Java Keytool using the command line for generating certificates. For more information, you may want to refer to this article and the Oracle Java Keytool Commands.

Reading: Integrating the Maven Dependency Check Plug-in Tutorial

Follow the instructions in this tutorial to learn how to integrate the dependency check plugin into Maven. You will need to edit the pom.xml file to add the dependency check plugin to Artemis Financial’s software application. When you compile your code, Eclipse will run the Maven plug-in.

Tool: OWASP Dependency-Check-Maven

OWASP Dependency Check Maven is an open source solution that is used to scan Java applications to identify the use of known vulnerabilities. This will be used for running a dependency check to statically test the code and produce an HTML report as output.