SER335 Module 6 Lab

Objectives:

1. Apply Design Patterns for Security
2. Consider aspects of ethical hacking and privacy rights with respect to security

Activity 1: Secure Design Patterns (60 points)

We are not done making JFM secure yet! Let's add a couple of secure design patterns to augment security.

Task 1: Implement the Observer pattern to log suspicious activity (20 points)

We know that one security technique is to make a logfile of suspicious activity that may be a sign you are being attacked. Use the Observer pattern to log any failed login attempts. Write these out to a file named jfmlog.txt. Each line of jfmlog.txt should have a timestamp and an indication of what user, password, and role combination were used in the attempt.

Task 2: Implement password rules using the Factory and Strategy patterns (40 points)

In an earlier lab you implemented the *validateUser* method to check for logins, and also in this lab you were asked to check for no-name, no-password issues. In systems you are familiar with, you now see many *password rules* that indicate what is a valid password or not. For this task, you will implement *validateUser* using the Strategy pattern to provide different password rules as a filter on the login process. The Factory pattern will assist in determining what implementation object of the Strategy to use.

Steps:

1. Create a Java interface named IValidateUserStrategy in package edu.asu.ser335.jfm. It should have a single method definition, *validateUser*.
2. Create 3 implementations of this interface in the same package:
   a. *StrictValidateUserStrategy* insists on passwords that are at least length 6, uses only letters, numbers, or the special characters !_=@%, and requires at least 1 uppercase letter, 1 lowercase letter, 1 special character, and 1 numeric digit.
   b. *ModerateValidateUserStrategy* insists on passwords that are at least length 3, uses only letters or numbers, and requires at least 1 letter and 1 numeric digit.
   c. *NullValidateUserStrategy* insists on usernames and passwords that are not empty. Currently, when adding a new user, you may add one that has no username or password (leave the fields empty). Similarly, if you modify authentication.json to manually add a no-name, no-password admin user:

      ```
      {"name":"","password":"","role":"admin"}    // add to the array in authentication.json
      ```

   You will find that you can login as an admin by entering no credentials. This strategy allows any non-empty username and non-empty password comprised of only letters and numbers of any length.
3. Use your new strategy to check login attempts. These events should be logged by your logger in Task 1.
4. Use the Factory pattern to determine what Strategy implementation to use for a given run of JFM. To do this:
   a. create a file named jfm335.properties with a single line:

      ```
      validateuserstrategy.property=<classname>
      ```

      where <classname> is the fully qualified implementation class (i.e. edu.asu.ser335.jfm.[StrictValidateUserStrategy | ModerateValidateUserStrategy | NullValidateUserStrategy].

   b. The Factory (call it ValidateUserFactory) should read in this property file, create an instance of an object of the classname type it reads in, and will return that to the caller (the main body of JFM)
   c. If the Factory fails to create a strategy object from the properties file – due to an error (e.g. cannot find the property file) or any other reason, have it choose the most appropriate implementation class from #2 above to use as a default (which one is based on one of your security architecture principles, review!)

We will provide some sample code to show how you can process the property file.

Submission for Activity 1:

For this Activity you may start with the same version of JFM you completed in Lab 2, or you may choose to use a later version based on the labs you have done this semester (after all, our end goal is to have a nice and secure JFM!). *Whichever version you choose to use, please provide the source tree of the code you are starting with as a zipfile named jfm_orig_<asurite>.zip, in addition to your new version for this lab named lab6_act1_<asurite>.zip*.

For submission, please prefix places IN THE CODE IN COMMENTS where you refactored by inserting "SER335 LAB6 TASK1" (All CAPS intended). We will search for these strings to find the locations you modified. There must be a comment in every single place in the code you modified, meaning the diff between your 2 zipfiles should all be accounted for! If we cannot find the locations of changes, we cannot grade it (we have to inspect your code to grade it to ensure the patterns were properly applied) and will not accept grade appeals where you ask us to regrade because you forgot to tell us where to look!

NOTE: Keep in mind what you are implementing is a filter; this means you still have to do the login check you implemented in lab 2 against the json files. What you are doing is adding a rule-check filter.

Activity 2: Ethics and Privacy (40 points)

Task 1: Review the following situations (20 pints)

1. Review the video: https://youtu.be/ibYwQqQ4g_0 and answer the following:
    a. (4) What Alice's act illegal? Why or why not?
    b. (4) Was Alice's act unethical or not? Explain
2. Review the video: https://youtu.be/3XPxK5EHb08
    a. (4) What Alice's act illegal? Why or why not?
    b. (4) Was Alice's act unethical or not? Explain
    c. (4) Were Charlie's rights at work violated? Explain

Task 2: Ethical hacking (20 points)

Ethical hackers need to be familiar with all applicable state laws – the National Conference of State Legislatures (https://www.ncsl.org/research/telecommunications-and-information-technology/computer-hacking-and-unauthorized-access-laws.aspx) provides an excellent and up-to-date reference of the laws in every state. Compare the laws of Arizona to the laws of another state (use the one you live in if you do not live in Arizona). In your comparison, identify:

1. (4) One place where the Arizona law is more stringent (strict)
2. (4) One place where the other state's law is more stringent
3. (6) Your overall assessment of which law is stricter and 2-3 sentences indicating specifically why
4. (6) Your opinion of which law is better. "Better" is of course subjective, I am looking for what criteria you decide to apply, ranging from strictness to economic arguments and enforceability, or whatever criteria you decide to apply. Provide a short writeup that identifies the criteria you use, and which state is better according to those criteria.

Your submission for this Activity should be a document named lab6_<asurite>.[txt|doc|docx|odt] with your answers clearly identified by Task and number.

Your overall submission should be a zipfile named lab6_<asurite>.zip with the contents of Activities 1 and 2.