**Computer Security Programming Project Deliverables**

Introduction:

The project’s objectives were threefold. Part 1 implements a password authentication mechanism to allow users to create a username and password as well as the ability to log in to a service. Part 2 demonstrates a password cracker that identifies potential password vulnerabilities. Part 3 evaluates password strength and notifies the user of its level of security.

Import Instructions:

To import in Eclipse: Click File 🡪 Click Open Projects from File System 🡪 Click Archive 🡪 Navigate to and select “computerSecurityProgramingProject.zip” 🡪 Click Open 🡪 Click Finish

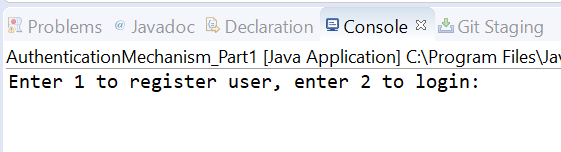
(To run each file, right click and select Run As 🡪 Java Application)

To import in Netbeans: Click File 🡪 Click Import Project 🡪 Click From ZIP… 🡪 Click Browse 🡪 Navigate to and select “computerSecurityProgramingProject.zip” 🡪 Click Open 🡪 Click Import

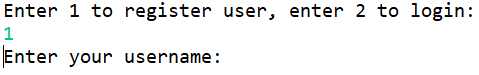
(To run each file, right click and select Run File)

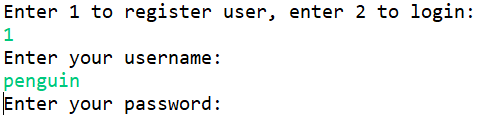
Part 1:

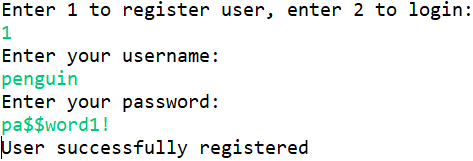
When first ran, the program, *AuthenticationMechanism\_Part1.java*, asks the user if they want to register or login with a currently stored username by entering 1 or 2 accordingly:



If 1 is entered, then the following prompts guide the user to register a new username and password:

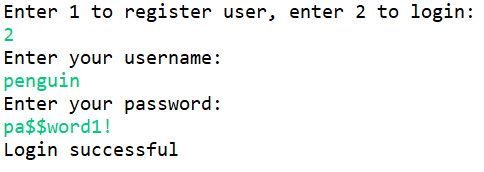




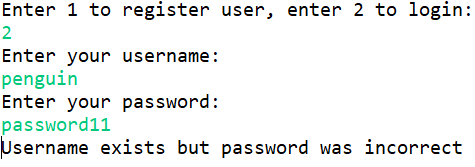


After creating a username and password, run the program again and enter 2 to log in. The program will evaluate your username and password and tell you of your login status:

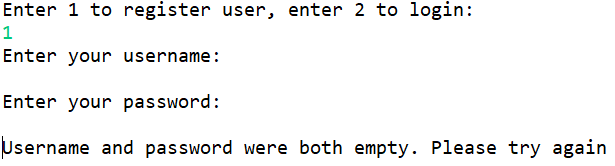
Successful login:



Unsuccessful login:



The program also identifies an attempt to leave both the username and password blank:



In the *userData.txt* file you will find a log of the username and the password hashed with the MD5 message digest scheme.



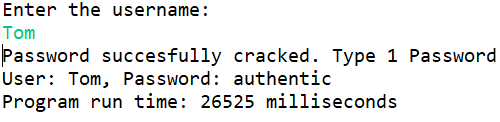
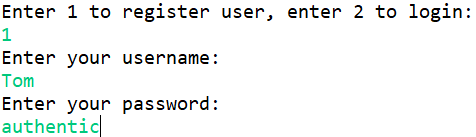
Part 2:

Running *PasswordCracker\_Part2.java* will prompt you to enter a username:

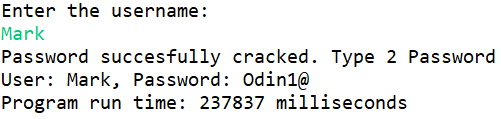
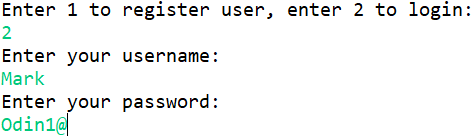


The program will then attempt to crack the password stored for the username utilizing MD5 digests of each dictionary word. If successful the program will display a success message, the type of password cracked, the username and password as well as the total run time. The users “Tom” and “Henry” are as follows:

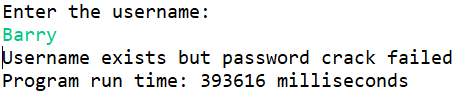
**Type 1** passwords:



The following is an example of a **Type 2** password crack:



This is a failed attempt to crack (password was not a combination of a dictionary word, numerical characters 0-9 and special characters(@,#,$,%,&):



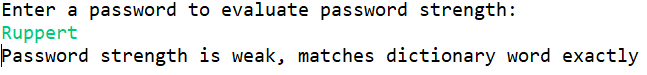
Part 3:

Running *PasswordStrength\_Part3.java* will prompt the user to enter a password to test:

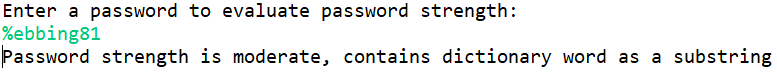


Subsequently, it will test a password and identify whether it is weak, moderate, or strong.

A weak password that contains a word found in the *dictionary.txt* file:



A moderate password that contains a dictionary word as a substring:



A strong password that does not contain any dictionary word:

