John Ryan

CMPT 220

Professor Rivas

12 December 2016

Final Writeup

One of the biggest problems in today’s digital society is the threat of being hacked or having your identity stolen. One of the easiest ways into a person’s personal life is through an 8-digit combination that too many people take for granted the fact that it is known only to them. Having a person’s password can lead to stolen personal information, money, and many other much worse things. Sadly, people do not generally help themselves when it comes to making a password. All too often, you see someone’s password include their name, address, birthday, general words, or other personal attributes. Leaving such things in your password makes you vulnerable to hacks. The most common type of hacking recently has been dictionary hacking. This type of hacking takes a long time but is generally successful when it comes to passwords not intended to protect the user. Dictionary hacking usually uses a computer program which can make up to 1000 attempts at your password a minute by going through a list of common words, phrases, or letter combinations (Gil). Sadly, even I am guilty of not having a unique password or even different passwords for different accounts. Thus, the program I want to create consists of a random password generator that allows the user to personalize their password in a way that does not include personal information. With many websites requiring different things such as a symbol or numbers and letters, this program will allow the user to select the desired length, then choose the number of letters, numbers and symbols they would like.

The way I plan on approaching this project is first by creating three arrays, which consist of numbers 0-9, letters, and different symbols, respectively. Then I have prompted the user to input the number of characters he would like the password to be. Following that, the user is asked individually how many numbers they want, how many letters they want, and how many symbols they want. After the user inputs these four numbers, the conditions for the program are checked using an if else statement. First, the number of characters is checked to ensure that it meets the requirement of it being more than 3 and less than 20. If it is not within those bounds, it reiterates the length requirement and the program exits. Next, the program checks to see whether the sum of letters, numbers, and symbols equals the total password length that the user originally stated. If not, the program displays that the sum of those three must equal the total password length. The final condition is that the input of the number of letters, numbers, and symbols must not be less than 0, preventing the user from putting in a negative number. If the user does put a negative number, the program displays a message to tell them that. Finally, if the user’s input passes all of the requirements, the program begins to create the password.

With this information, a for loop is created for each of the individual characteristics. The length of the loop is however long the user input was for that specific characteristic. For example, if the user wants 4 numbers to be in his password, the loop will run four times. In the for loop, there will be a math.random function multiplied by the length of the original array. Then, these numbers serve as index numbers for the original array, and a new array is created for each. For example, if the user wants 3 letters, and the three random index numbers are {3,25,2}, then the new array using these index numbers is {c,z,b}. Then the program does the same for the numbers and symbols based on the users input.

Now, with the three arrays containing the random letters, numbers, and symbols, I needed to join the three arrays and make them into one. In order to do this, I created a for loop for each characteristic. I declared a new array labeled “passwordArray” and in each for loop, it adds each element into that new array and the loop would run for however long the original array is with the random characters. For example, if the number array is {2,7,5}, then the for loop would add that to the final array. Then if the letter array is {f, p, d}, the final array would then be {2,7,5,f,p,d}, and so on.

The program now left with an array which has the password in order. While at this point, the password is effectively generating random numbers, letters, and symbols, it would be better equipped to be used if there was no order for the arrangement of the final array. This was definitely the hardest part of the project in terms of conceptually figuring out how to randomly arrange the final array. However, after many failed attempts of using collections, I was able to figure it out using a for loop. I first made the loop run the length of the array multiplied by two. I did this in order to ensure that the method within the loop was done enough times based on the length of the password where it was truly random. Next in order to randomize the array, each time the loop ran, it would select two random numbers, from 0 to however long the length of the array was minus 1. These two numbers, which correspond to two index numbers in the array, would be switched using a temporary character. Then the two index numbers are put back into the array, switching the two elements using their two randomized index numbers. Now, with the randomized final array, the system prints it out for the user.

After running the program tens of times trying to figure out what else could make the program stronger and less apt to errors, I finally remembered to ensure that what the user actually puts in is a number. With that in mind, I remembered back to javascript with the isNaN function. However, when researching how to apply that function or a similar function to java, I was unsuccessful. So, I had to get creative with how I approached the error. When I saw the error itself, I decided to try to use a try-catch function which would catch the error and instead of the user simply seeing an error message, they would see “You must enter a number”. Thus, the try is around the whole code right before the scanner and the catch is after the password is printed.

While talking to other people who worked on the same project and decided to also make a password generator, I found that my idea was noticeably different than the others. Some of them asked for the user’s current password to check some conditions before actually giving them a password, then if they needed one according to the program, it would print a new one which consisted of letters. I feel that mine is better compared to the others because of the customization and also the amount of security offered by the randomization of my program. By the user being able to select the length of the password and the number of letters, numbers, and symbols, it allows the user to ensure that the password is acceptable by the website and it is something that they can remember. Also, with both the randomness of the characters selected and the order of the final password also being shuffled, the security of the password from this program as compared to others is a big difference.

In making this project, I hope to create a program which helps prevent people from making generic passwords and being subject to hacking or theft. This program sets itself apart from other password generators, and while it does not produce a password that is necessarily easy to remember, it does offer a lot of protection when it comes to hacking. With the personalization and the truly random process for selecting the characters and their order, this password generator allows for multiple uses and at the same time, is simple enough to be used by anyone.

Bibliography

Gil, Paul. "What Is 'Brute Force' Dictionary Hacking?" *Lifewire*. N.p., 16 Sept. 2016. Web. 11 Nov. 2016.

UML Diagram

