Drew Burns Final Password Generator 12/10/16

I look forward to gaining knowledge in my major by completing an application project on security. This semester I am also taking a data communications class where our current main focus Is security. I find cybersecurity to be a very interesting topic, and think I will eventually take it as a minor. Therefore, the idea of a strong password generator really caught my eye. With this generator I can make my passwords as well as other people passwords more secure from dictionary attacks and other kind of reusable password attacks from hackers. This code can test the use of diverse characters within the password, as well as the use of capitalization, characters, and numbers. As I continue this project I am excited to see how these types of codes are written and what types of flaws I may come across as well as how I could customize the code to defend against certain types of password hacking programs. I think that my strong password generator has a lot of significance in the real world and to everyone's computer. My program will test a user's password and make sure it passes a series of tests. Checking their passwords will help people to tell if their information is secure against potential hacks.

My code has 2 specific functions. First I ask the user to input their current password, from here my code will ask the user for their first and last name. Then, the password must meet a list of parameters. If the password fails to even one, the user will have to start over and try a new password. The password must be 12 characters long, contain an uppercase letter, a lower case letter, and two numbers. On top of this the password cannot contain the users first or last name in any part of their password. Also,

the password has to be unique. Meaning, the password cannot possess string of characters from the top most used passwords. This will help to keep the passwords more secure by switching up the use of characters and numbers and prevent the user from inputting easily hackable phrases. Next I give the user a new password. One of the main problems with creating a strong password that will trick hackers is that you end up tricking yourself and not being able to remember your password. My program creates a strong password and does so in a memorable way for the user. I ask the users a series of questions that I have them answer with brief phrases. My program takes the first letter of each phrase they answered with and inputs it into an array. The array stores the one letter answer as well as randomly generated numbers after each letter. To make sure the new password matches all the steps from the previous part of the code, it capitalizes the first and last number in the array. This acronym style password is a unique way to generate strong and memorable passwords for people. The user will know the meaning of the characters but it would be improbable for the hacker to.

If a user was to input their password as password a number of things would go wrong and it would pass zero of my tests. The user will be given a warning that their password doesn't contain 12 characters. If they were to retry with "passwordpassword", it would pass the first test of having 12 characters. Therefore, my program will prompt the user to enter their first and last name. Once entered the program will go through the password checklist and make sure each qualifying test is passed. The first test failed will pop up to the user and make them retry their password. This user will get a "Your password must contain at least one uppercase character" because that is the first of many failed tests. Once the user incorporates a capital letter into their password for

example “Passwordpassword” the program will again ask the user for their first and last name to make sure this new password doesn’t include them. This time the user will get an error for not including two numbers inside of their password. If the user inputs a password like “helloWorld77text” all initial tests will be passed, and the user will be greeted with a message reading “Your password passes all the initial tests!!” from here the program will start asking questions to the user. Let’s pretend the user answers every question with “blue frog” this would not be a useful way of using the program but hypothetically could still work. The user’s password will come out as B3b7b1b1b9B4. The reason for this is because all six questions were answered with the same phrase. My code took the first letter and made it uppercased for the first and last question, and interweaved randomly generated numbers between the letters. Although this is not an effective way of using the program, the code will still work and give you a decent password. Normally user’s answers will be much more diverse and personal to them. Making the password both more memorable and more difficult to hack.

I felt my code worked best as a single class and could work most simply without the use of methods. Overall I felt this was a good experience for me. I gained knowledge and confidence in my coding and also learned a lot about the kids of attacks hackers use and how to best prevent passwords from being hacked. I enjoyed being able to create something that I found interesting and important for myself and others. Having my program tested by others was a great experience for the production of the remainder of code, as well as teaching people the importance of security. I feel the program I created is paramount in today’s day in age. The average person has multiple passwords for close to a hundred different accounts. The security of some is very

important and would set many people back financially if their more important online accounts were hacked.

Works Cited

Liang, Y. Daniel. *Introduction to Java Programming: Brief Version*. Boston: Pearson, 2013. Print.

Tsukayamka, Hayley. "How to Keep Track of Your Passwords without Going Insane." *The Washington Post*. WP Company, n.d. Web. 10 Dec. 2016.