Abstract

Every person using different online services is concerned with the security and privacy for protecting individual information from the intruders. Many authentication systems are available for the protection of individuals' data, and the password authentication system is one of them. Due to the increment of information sharing, internet popularization, electronic commerce transactions, and data transferring, both password security and authenticity have become an essential and necessary subject. But it is also mandatory to ensure the strength of the password. For that reason, all cyber experts recommend intricate password patterns. But most of the time, the users forget their passwords because of those complicated patterns. In this paper, we are proposing a unique algorithm that will generate a strong password, unlike other existing random password generators. This password will he based on the information, i.e. (some words and numbers) provided by the users so that they do not feel challenged to remember the password. We have tested our system through various experiments using synthetic input data. We also have checked our generator with four popular online password checkers to verify the strength of the produced passwords. Based on our experiments, the reliability of our generated passwords is entirely satisfactory. We also have examined that our generated passwords can defend against two password cracking attacks named the "Dictionary attack" and the "Brute Force attack". We have implemented our system in Python programming language. In the near future, we have a plan to extend our work by developing an online free to use user interface. The passwords generated by our system are not only user-friendly but also have achieved most of the qualities of being strong as well as non- crackable passwords.

1. Introduction

Having a weak password is not good for a system that demands high confidentiality and security of user credentials. It turns out that people find it difficult to make up a strong password that is strong enough to prevent unauthorized users from memorizing it.

Text-based username-password is the most commonly employed authentication mechanism in many multiuser environments. These multiuser applications, while registering users to their application, some applications allow users to create password their own and others generate random password and supply to users.

This application can generate random passwords, with the combination of letters, numerics, and special characters. One can mention the length of the password based on the requirement and can also select the strength of the password.
Internet security is recently becoming a significant issue with the increasingly wide range of internet applications. Bank and commercial exchanges are now being carried out online in the form of internet banking and commercial electronic transactions. The level of information transmission is becoming more critical for occurring information leakage, and damages due to such leakages are more significant. User authentication is a necessary security element in the open network environment, and the use of simple authentication information also has some severe problems.

One problem is that it is easy for the attackers to guess passwords whenever the users often choose personal information, such as their ID or telephone number as passwords. The users do this to remember them quickly.
Sometimes they use the same password for many web sites. They do this so that they do not have to remember too many passwords. Another problem is that users rarely choose passwords that are both hard to guess and easy to remember. To help users in choosing good passwords, many experts proposed different kinds of guidelines for following many policies. Another problem is that many of the deficiencies of password authentication systems arise from human memory limitations. We know that a maximally secure password with maximum entropy consists of a string having numerous random special characters as long as the system permits. But human memory can not remember such long as well as complex passwords. When humans do remember a sequence of items, those items must be familiar pieces of information such as words or familiar symbols. However, most people are much better at recalling the information when we encode them in multiple ways. So, password authentication involves a trade-off. Some passwords may be easy to remember (for example, best friends name, pets name, kids name, etc.) but also easy to crack through dictionary searches. Other passwords may be secure against guessing but challenging to recall. Besides, some users tend to create different passwords for their various online accounts. When users create different passwords for different accounts, they need to remember several passwords which may be problematic or confusing
for them during use. In this case, for remembering all the passwords, they sometimes keep insecure written records of them. Having a written document of passwords is a terrible idea because this act is not free of password cracking attacks.
Eventually, an attacker can easily guess the password.

2. Literature Survey

Author	**Title**	**Year**	**Source**	**Findings/Output**
Michael D.	A comparative	2007	IEEE	This paper

Leonhard; V.N. Venkatakrishnan	study of three random password generators.			compares three random password generation schemes, describing and analyzing each. Qualities discussed include security, memorability, and user affinity.
Farhana Zaman Glory; Atif Ul Aftab; Olivier Tremblay-Savard; Noman Mohammed	Strong Password Generation Based On User Inputs	2019	IEEE	In this paper, they implemented UI based password generator that takes inputs from the users.

3. Objectives

Every individual who works with different modem online services is concerned with his/her security and privacy for protecting personal information from the attackers. Password authentication is one of the popular authentication systems that has been used for many years for defending online accounts or services. At the same time, the users also need to create a strong password for protecting their services from a real-world attacker. Thus it is recommended to create a unique password with a healthy pattem so that they can protect it from the intruders. But usually, the users forget their passwords because it is not easy to remember a string with a robust and intricate pattem. To recall their passwords, they save them in their own devices, in notebooks or on sticky notes from where the passwords can get easily compromised by the intraders. So, it is always expected that the users should use strong passwords which they can easily remember without the help of writing down or saving them in their own devices or notebooks. Thus, the idea of auto-generating strong passwords based on their inputs will be convenient for them to remember easily.

Let us clarify our goal by describing an example. Suppose a user has been provided with a password like “#XeV65a$PzH08!” from a random online password generator in the web for his usage. These kinds of passwords are definitely secure, but are almost impossible to remember for the average user. On the other hand, our system will ask the user to provide five input texts and any two numbers for generating passwords. Let us assume the user provides this input: “mango, cat, red, kathy, ice, 67, 81”. After that using them, our system will generate a password like “81KathY%m@ngO|”. We see that both passwords have healthy patterns, so they are safe to use. Now if the user is given a choice of choosing one password from these two passwords “#XeV65a$PzH08!” and “81KathY%m@ngO|”, the user should go for the second one. The reason is that the first password from the random generator is clueless to the user and very tough to remember. On the contrary, the second password from our system has two words and one number which they can not easily forget because they are based on the user’s suggestions. In summary, we aim to generate reliable and non-crackable passwords for the online services of the user based on the information provided by the individual which can be remembered easily.

4. Problem statement

Random Password Generator in Python

5. Existing system

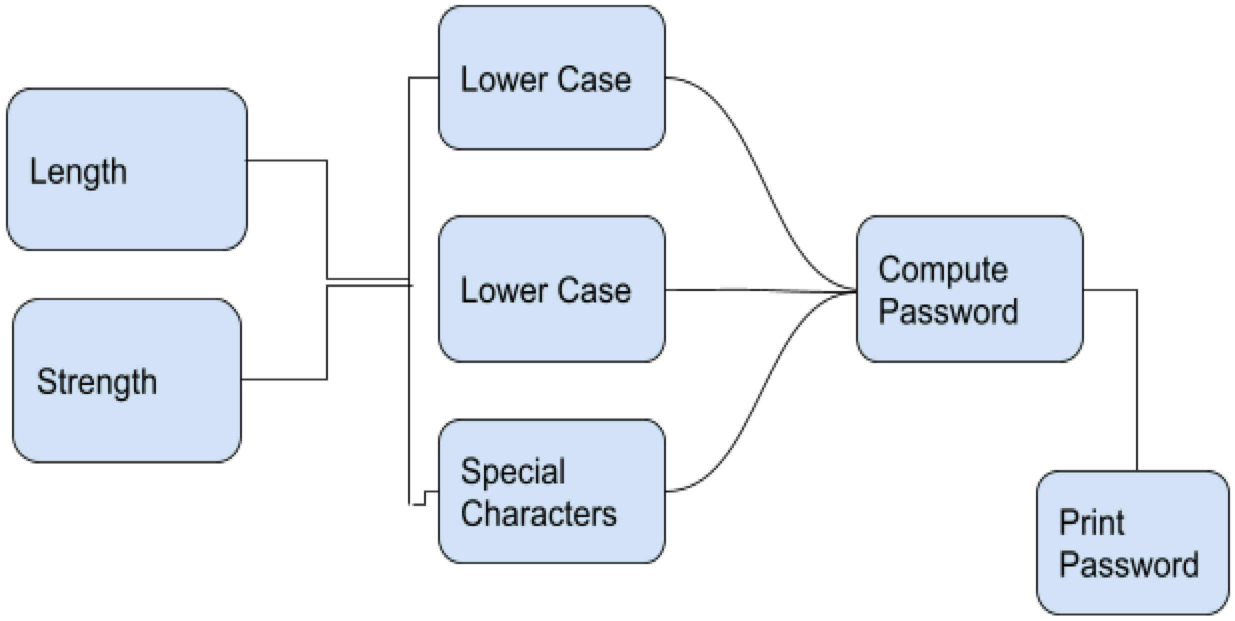
- When the existing system was studied, it was found to have some problems, the user interface was not good as there is no option to specify the strength of the passwords.

- Some of the systems were only built to generate uppercase and lowercase strings that make passwords vulnerable for brute-force attacks.
- Users need to manually type the generated passwords as the password is a little complex it will be time-consuming.
- While working on this project we have taken into consideration all the above drawbacks and included them in our project.

6. Proposed system

- The proposed system is designed to generate random passwords based on the length given by the user.
- Current systems consider the following parameters while constructing the password.
    - i. Length of the password.
    - ii. Strength of the password.
- Users can input the required length to get the password.
- The strength of the password is the complexity of the password. There are 3 options given to the user.
- Easy - Includes only lower case letters.
- Medium - Includes both upper and lower case letters.
- Strong - Includes Upper case, Lower case, and Symbols.
- The generated password can be copied as a string.
- The user has the interface to exit from the system.

7. Architecture Diagram



8. Applications

- To prevent your passwords from being hacked by social engineering, brute force, or dictionary attack method, and keep your online accounts safe.
- One-click to generate and copy the password.
- Includes lowercase, uppercase, and symbols to make the password more complex and difficult to memorize.
- Written in python that makes the executions of the program speed and reliable.
- Users have options to select the strength of the passwords based on their requirements.

9. Code

```
import random
from tkinter import *
from tkinter import messagebox
from tkinter.ttk import *

def low():
  entry.delete(0, END)
  length = var1.get()
  lower = "abcdefghijklmnopqrstuvwxyz"
                                    upper                    =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
                                   numbers                   =
"""ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01234567
89!@#$%^&*()_-+/\:;"""
  password=''
  if var.get()==1:
     for i in range(length):
        password+=random.choice(lower)
     return password

  elif var.get()==0:
     for i in range(length):
        password+=random.choice(upper)
     return password

  elif var.get()==3:
     for i in range(length):
        password+=random.choice(numbers)
     return password

  else:
          messagebox.showwarning('Warning','Select  One  of  the
options')

def copy1():
  random_pwd = entry.get()
  root.clipboard_clear()
  root.clipboard_append(random_pwd)

def generate():
  p = low()
  entry.insert(10,p)

#GUI Window
root = Tk()
var,var1 = IntVar(), IntVar()

root.title('Random Password Generator')
pk = Label(root,text="Password")
```

```
pk.grid(row=0)
entry = Entry(root)
entry.grid(row=0,column=1)

c_label = Label(root,text="Length")
c_label.grid(row=1)

copy = Button(root,text="Copy",command=copy1)
copy.grid(row=0,column=2)

generate = Button(root,text="Generate",command=generate)
generate.grid(row=0,column=3)

exit = Button(root,text="Exit",command=root.quit)
exit.grid(row=0,column=4)

#Radio Buttons to set password length
radio_low = Radiobutton(root,text="Low",variable=var,value=1)
radio_low.grid(row=1,column=2,sticky='E')
radio_middle                                                     =
Radiobutton(root,text="Middle",variable=var,value=0)
radio_middle.grid(row=1,column=3,sticky='E')
radio_strong                                                     =
Radiobutton(root,text="Strong",variable=var,value=3)
radio_strong.grid(row=1,column=4,sticky='E')
combo = Combobox(root,textvariable=var1)


combo['values'] = (8,9,10,11,12,13,14,15,16,
              17,18,19,20,21,22,23,24,25,
              26,27,28,29,30,31,32,33)
combo.current(0)
combo.bind('<<ComboboxSelected>>')
combo.grid(column=1,row=1)
root.mainloop()
```

Modules needed:

import random
import pyperclip
from tkinter import * from tkinter.ttk import *

low() function is used to calculate password.
generate() function for generation of password
copy1() function is used for copy to clipboard

10. Merits & Demerits

Merits :

- High set of potential or huge pool of high security passwords. Beneficial for one-time authentication. Difficult passwords are hard to crack.
- Easy user interface.
- Generated the passwords that are hard to crack by social engineering, brute-force attacks.
- Allows users to decide length and strength of the passwords.
- Mixture of capital letters, small letters and special characters makes password guessing difficult.

Demerits:

- Remembering the generated password is difficult.
- Managing multiple passwords for multiple sites or account is difficult.

11. Algorithm 1 Password Generating Algorithm

```
Require:length and strength as input.
l: Begin Procedure
4: specialchars 4— { @ , $ , !, # , % , & , (, ), 0, 3, 8, < , |}
5: punctuation 4 - {<§>,*,+ ,-,:,” ,/,\,~ ,? ,[ ,] ,{ ,} ,$ ,! ,# ,% ,& ,( ,) ,_ ,< ,|}
6: while N > 0 do
7:      stringl 4— randomly selected from small characters
8:      string2 4— randomly selected from capital letters
9:      Number 4— randomly selected from two numbers
10:     stringl 4— capitalize (stringl) some letters randomly
11:     string2 4— capitalize (string2) some letters randomly
```

```
12:     Final_string 4— merge stringl and string2 randomly
13:     Alphabets 4— {a, s, i, r, x, q, c, j, o, e, b, k, 1}
14:     Final_string 4— Final_string. replace(Alphabets, specialchars)
15:     Final_string 4— randomly insert Number in the final string
16:     either in the middle, or in the beginning or in the end of it
17:     Final_string 4— randomly append one special character from
18:     punctuation at the end of the Final string
19:     C = len (Finaljstring)
20: if C < length then
21:     Final_string 4— append (length - C) special characters randomly
22:     print Final_string
23: end if
24: N 4— N — 1
25: end while
26: End Procedure
```

In our methodology, at first, we take input information from the user who is asking for a password. First we get the length and strength of the password from the user. Based on the length and strength of password we call renerate function where random function picks the characters from the array depending on the strength. If the strength of the password is **low** then only small characters are picked from the array and by looping over the length. If the strength of the password is medium then random function picks the characters from small and medium characters array. If the strength of the password is hard then it will pick the mixture of the small case, capital and special characters from the array by looping over the length.
Finally we will construct the string password and display it to the users.

12. Libraries Used

- **Tkinter -** Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. It is a thin object-oriented layer on top of Tcl/Tk.

- Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.
- Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.
    1. import the Tkinter module.
    2. Create the main application window.
    3. Add the widgets like labels, buttons, frames, etc. to the window.
    4. Call the main event loop so that the actions can take place on the user's computer screen.
- **Random Library** - Generate pseudo-random numbers
- Python has a built-in module that you can use to make random numbers.
- The random module has a set of methods:

Method	**Description**
seed()	Initialize the random number generator
getstate()	Returns the current internal state of the random number generator
setstate()	Restores the internal state of the random number generator
getrandbits()	Returns a number representing the random bits
randrange()	Returns a random number between the given range
randint()	Returns a random number between the given range
choice()	Returns a random element from the given sequence
choices()	Returns a list with a random selection from the given sequence

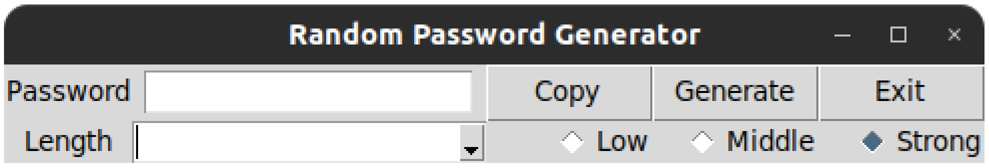
shuffle()	Takes a sequence and returns the sequence in a random order

13. Input samples and generated password from them

Length	Strength	Generated Passsword
8	Low	suptmnyq
6	Medium	COyfvn
7	Hard	J*meipv

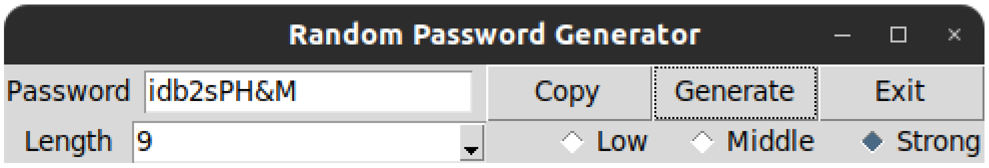
Table 1.0

14. Results



Password:
Length:
Low Middle [✓ Strong]

Fig. 1.0



Password: idb2sPH&M
Length: 9
Low Middle [✓ Strong]

Fig. 2.0

15. Conclusion

In our proposed system, we are generating passwords by prompting the users to provide us with some texts and numbers (which they can easily remember) as inputs. Our generated passwords have ensured the minimum criteria of the password strength. Through various experiments, we have proved that our generated passwords are strong enough. Our generated passwords are different from the passwords of other online random password generators as they are not produced entirely at random. Instead, they are created from the inputs given by the user. They can be easily remembered as they contain only the texts and numbers which the users want to use in their passwords and can not forget easily. We also have experimented that our generated passwords can defend against the dictionary and the Brute Force attacks. However, there is another password cracking attack named "the social attack" which has not been covered in our work. Social attacks take an individual's system and personal information into consideration to try and help speed up the process of dictionary attacks. Say the adversary knows the victim's dog's name as Pickle, mom's name Jane and the birth date 01/07/1980. Then the adversary obviously will give this known data higher priority over other options. So the more the users will use secretive information (which have not been unveiled to any type of social media) as inputs, the more their passwords generated by our system can be safe from the social attack though we have planned nothing yet to defend this attack. In the future, we will extend our work and devise a method so that our generated passwords can defend 0422against the social attack.

For now, we are using now random data sets as input data. By conducting a user survey, we can collect a real data set which we can use in our generator to generate passwords. Through the use of a real data set, we can also check
how much people can remember our generated passwords. Through another user study, we can also get feedback from the users to evaluate whether they can remember the passwords or not. We also have plans to add more features and steps to our algorithm to strengthen our system. We have a goal to add some exceptions in our methodology like (a) if the user inputs compound words like ("living room"; "ice cream"; "flower vase" etc.), our system will not receive these type of data and will prompt the user to provide different contexts as inputs to the system, (b) our framework will not accept the very common words like ("password", "keyword", "pass", "admin", "abc", etc.) as these words are the first choice of the adversary to try for cracking passwords. Our other plan for this work is to experiment with how the length of the input words and numbers

affect the strength of the generated passwords. We will make an online application for our system so that the users do not have to wait for installing our system into their machines for use. We will also make it open source for the user’s convenience.

16. Future scope

Current project allows the users to generate and copy the password, there is no way that user can store and manage passwords.

Future scope of the project will be allowing users to store and manage the generated password in application itself.

17. References

https://docs.python.org/3/tutorial/index.html
https://docs.python.org/3/library/random.html
https://realpython.com/python-gui-tkinter/