4. Human-like password generator

The first team to write “Бро, это чат без преподавателя” in our chat will receive +1 score to their grade.

Do not use this generator to create passwords. This is not a secure generator (obviously), but a tool for very basic security checks of password hashing.

The purpose of this task is to acquaint you with the way real people generate passwords. You cannot enforce everyone to use secure random generators. But you still need to secure their accounts/information/etc.

Upload your results to a public github repository.

Part 1.

1. Create a simple application that generates passwords “just like humans do”:
    a. Search for statistics of top 25/100 passwords. Your app should generate passwords from this list say 5-10% of the time. You may want to research what is the exact percentage in real leaked passwords databases. For the purpose of this task it’s not necessary but encouraged.
    b. Search for 100k-1M most common passwords lists. Your app should generate passwords from this list 50-90% of the time.
    c. Make 1-5% of passwords really random. Make length/symbol space still bearable for people to remember though.
    d. For the rest you are free to choose options regarding length/symbols/generation scheme, but still try to make it look like passwords that humans would create. You may want to combine words, add numbers to the end of words, transliterate words, replace letters with numbers etc.
2. Your app should create a bunch of these passwords at a time.
3. Now choose 2-3 hashing schemes of different security levels. md5, sha-1 + salt, argon2i, bcrypt, you name it. For each scheme generate a bunch of passwords (100k-1M) and create a csv file with hashes (and salts) only. Make it public for everyone in class.
4. Write a short report on how you create your passwords. You may choose to omit describing hashing scheme to make part 2 more challenging. **You will be graded based on your report.**

Part 2.

1. Take anyone but your csv file with hashes.
2. Make yourself familiar with a tool like hashcat or any similar. In short it is a software used to find hash preimages.
3. Try dictionary search, bruteforce or anything else you heard about in lectures. Note how much time your search will take. Also note how effective each method is.

4. Write a short report on how many passwords you managed to recover. What kind of passwords are those (dictionary, combined, random etc). What hashing scheme you found to be pretty strong and why. What attacks you used and what appears to be effective. Include a link to the input file you used.
5. Write some recommendations based on your own experience with this tak on:
    a. which hashing scheme one should use in real life applications
    b. which rules should one enforce on user passwords
    c. anything else you find important
6. Be prepared to reproduce your results live
7. **You will be graded based on your report**

```
5e056c500a1c4b6a7110b50d807bade5
0ea29a0d3cbfb7f98424a0a035859f60
0857e3b78eeb749dd6354c846960d687
2041810a4da800f1488b2aa5074356b4
24ef2299dfb114ef16c39f7fada35ff9
abe0a2e79b3747ef12b70b429ba905ab
2104ceb418482819eb46ac3d952913a4
d4185990ae5d1b1f1c0cc42e49559fab
818a5850ffd55379b827b3099e8744b0
8ec7e84f89ab729805aaae71a45c5ee3
c81329a7e0dfd63c766fc9c565146fff
fa86feed504c85cbf2bd2ac970d707a4
1e94437752 3e4aae7baea648b21fdac9
ee647efa1529f47da13db71b662df247
077b1a78fae644f15d11e88d8a4d1622
```