IMPROVING PASSWORD GENERATION USING BIAS FROM LEAKED DATA

Ivan Drankov

University of Mining and Geology “St. Ivan Rilski”, 1700 Sofia; E-mail: ivan.drankov@mgu.bg

ABSTRACT. Different brute force algorithms can be optimised by creating a better method for “guessing” the right password. Improving the performance of brute force algorithms can help in generating hard-to-guess passwords based on opposing non-bias rules when creating a password. Passwords created by users have a bias with specific keystrokes that can be analysed. Brute force can be improved by bias. Bias prediction rules can be summarised and used in brute force generation and password generation. This article uses various leaked data with a total amount of millions for generating templates and bias rules.

Keywords: Brute force, leaked data, human bias, big data.

ПОДОБРЯВАНЕ ГЕНЕРИРАНЕТО НА ПАРОЛИ, ИЗПОЛЗВАЙКИ ПОТРЕБИТЕЛСКИ ПРЕДПОЧИТАНИЯ, ИЗВЛЕЧЕНИ ОТ ИЗТЕКЛИ ДАННИ

Иван Дрънков

Минно-геоложки университет „Св. Иван Рилски“, 1700 София

РЕЗЮМЕ. Различните brute force алгоритми могат да бъдат оптимизирани чрез създаването на по-добър метод за „познаване“ на точната парола. Подобряването на ефективността на тези алгоритми е в помощ при генериране трудни за „познаване“ пароли, основаващо се на обратните правила – за „непредпочитание” при създаването на парола. Паролите, създадени от потребителите, имат шаблон на предпочитания към конкретни натискани клавиши, който може да се анализира. Brute force алгоригмите може да се усъвършенстват при отчитане на склонността да се натискат определени клавиши. Правилата за прогнозиране на потребителските предпочитания могат да се обобщят и да послужат за създаване и добавяне на нови правила за сложна генерация на пароли. Тази статия използва различни изтекли данни с обща големина над милиони записи за генериране на шаблони и правила на потребителските пристрастия.

Ключови думи: Brute force, изтекла информация, човешки фактор, big data.

Introduction

Randomly generated passwords do not have bias. They are not truly random. Their generation depends on a randomly selected seed. Generating passwords from different seeds is the key to increasing randomness. Human-created passwords are not random. Most people tend to use passwords which are easy to remember and read. Therefore, biased behaviour can be observed, such as:

-Some characters are not used in the process. Different languages have different character frequencies.

- Uncommon characters are rarely used just because most people do not even know they are allowed in passwords or are hard to remember.

-Repeating common phrases. Some users use passwords that contain a base phrase or repeating pattern.

Observing this behaviour can lead to a more efficient way to generate new password dictionaries for brute-forcing attacks. Brute force is still an effective method for exploiting vulnerable IoT devices.

Fig.1 Heat map of about 4 million users’ password keystrokes.

Brute force algorithms

Many relevant search problems, from artificial intelligence to combinatorics, explore large search spaces to determine the presence or absence of an ascertained object. These problems are hard due to combinatorial explosion and have traditionally been called infeasible. The brute-force method, which at least implicitly explores all possibility-ties, is a general approach to systematically searching through such spaces (Heule, 2017).

Brute forcing passwords can be categorised in the following categories:

-Random generation. This method is generating random passwords to “guess”. For longer passwords, this method is ineffective.

-Dictionary attack (rainbow list). This method uses dictionaries containing real leaked passwords. If the attacked device has one of those dictionary passwords, guessing it will be a fast process.

-Pattern attack from the dictionary. Creating substrings from existing passwords when the dictionary attack cannot succeed.

The “guessing” time of the password depends on the password complexity and processing power. Most passwords can contain as many as 94 characters. Those characters can be categorised into different subsets from the total set of 94, as shown in Table 1.

Table 1. *Password combinations for a 10-character-long password*

TOTAL combinations	94^{10}	5,38615E+19
(A-Z)	26^{10}	1,41167E+14
(a-z)	26^{10}	1,41167E+14
(0-9)	10^{10}	10000000000
(Sp)	32^{10}	1,1259E+15
(A-Z) + (a-z)	$(26+26)^{10}$	1,44555E+17
(A-Z) + (0-9)	$(26+10)^{10}$	3,65616E+15
(A-Z) + (Sp)	$(26+32)^{10}$	4,30804E+17
(a-z) + (0-9)	$(26+10)^{10}$	3,65616E+15
(a-z) + (SP)	$(26+10)^{10}$	4,30804E+17
(0-9)+(SP)	$(32+10)^{10}$	1,70802E+16
All 10-character-password combinations for a Google Account	94^{10}-26^{10}-26^{10}-10^{10}-32^{10}-$(-26+10)^{10}$-$(26+32)^{10}$-$(26+10)^{10}$-$(26+32)^{10}$-$(32+10)^{10}$	5,28295E+19

Most services enforce additional rules to increase password complexity based on those subsets. This can be described in Regex. One common rule is: Minimum eight characters, at least one upper case English letter, one lower case English letter, one number, and one special character.

In this case, a password must contain:

- At least one upper case English letter:(?=.*?[A-Z])
- At least one lower case English letter:(?=.*?[a-z])
- At least one digit :(?=.*?[0-9])
- At least one special character or space: (?=.*?[#?!@$ %^&*-])
- Minimum eight in length: .{8,}

```
^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[#?!@$ %^&*-]).{8,}$
```

Fig. 2 Password Regex.

Human bias attack

Character frequency can be used to determine the language (Kerwin, 2006). In this case, analysing data sets from user passwords can be used in creating a template for generating passwords. To create a generation method from human-made passwords, large password sets are needed. Those sets can be separated by type of the service they protect. For example, brute-forcing IoT devices will not benefit from templates created from password sets used in web forums. So, choosing the right data is a key factor, too, thus increasing the odds of success. After creating proper templates, the most common characters from those data sets have priority over the uncommon ones in brute force password generation.

Fig 3. Character frequency from 3723515 leaked E-mail passwords

In the generated template, the following can be observed:

- The most common character is “1” used exactly 2541116 times with a chance of ≈68.25% to appear in any given password.
- The least common character is “@” used exactly 5 times with a chance of ≈0.000134 to appear in any given password.

The results can be divided from the different password subsets that can be described in one string. The order of characters is not random. It represents the frequency of the characters. The first character is always the most common and the last one is the rarest.

Table 2. *Passwords subsets ordered by frequency*

subset	Frequency string
(A-Z)	AESIORNDLKYMUTVFBGCPZHJXWQ
(a-z)	aerintosklyfmdvhbqwucgzpjx
(0-9)	1029385764
(Sp)	.-_,!*()$+#/(%=^"`;&]' \?{<~}|>@

The overall subset frequency indicates the percentage of subsets of characters used in password creation. The subsets (0-9) and (a-z) come on top. This means that every password contains at least 3.7 numbers, 4.2 lowercase letters, 0.27 uppercase letters, and 0.02 special characters.

Table 3. *Passwords subsets ordered by overall frequency*

0-9	a-z	A-Z	SP
3,799974	4,294155	0,272275	0,028605

The most used password length for a password is 10 characters. The second most used length is 8 characters. Most passwords will be distributed in this range (7-13 characters).

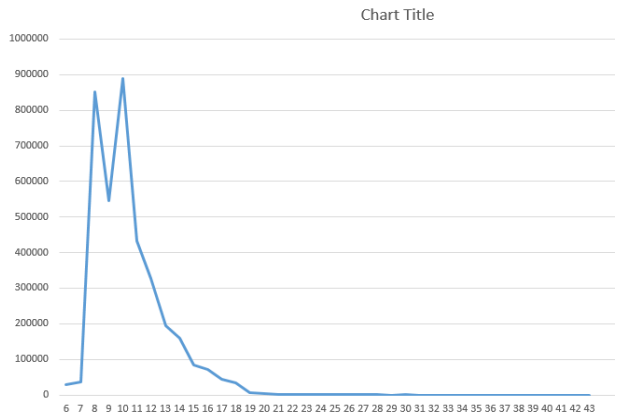

Fig. 4. Password lengths

Character subset frequency by position can indicate the likelihood of any character subset of being included in any chosen position.

Table 4. *Character subset frequency by position*

	pos1	**pos2**	**pos3**	**pos4**	**pos5**
A-Z	0,058	0,029	0,032	0,027	0,027
a-z	0,566	0,541	0,633	0,525	0,492
0-9	0,374	0,428	0,332	0,444	0,477
Sp	0,002	0,002	0,003	0,004	0,004
	pos6	**pos7**	**pos8**	**pos9**	**pos10**
A-Z	0,027	0,032	0,034	0,033	0,033
a-z	0,463	0,484	0,445	0,515	0,491
0-9	0,505	0,479	0,516	0,445	0,469
Sp	0,005	0,005	0,005	0,007	0,007

Improving password generation

These results can be used to create better password generation. Using the gathered data, a new brute force algorithm can be made. This new algorithm is supposedly better. Based on the collected data, this new algorithm will generate a new password to “guess” grounded on the following:

-The password is between 7 and 13 characters;

-The characters used in password generations must have a frequency > 1% (top 80% characters by frequency);

-The order of generation is based on the overall subset frequency and on the character subset frequency by position.

In the case of 10 characters with those rules, long password (5,63135E+18) combinations will be generated. Or about 9% of the total possible combinations are excluded. In other words, this algorithm checks the top (≈91%) of the total combinations for Google accounts (528295E+19) based on the template. The other ≈9% will be generated last. The expected performance improvements are at least (9%).

Testing such algorithm is not feasible due to the long computing time

Brute forcing improvements can be used as a by-product of secure password generations.

To create secure passwords, a variety of factors must be considered:

-Passwords from public leaked data must be excluded. Password generators must use data sets from leaked data to check for unfavourable matches. Services such as HIP (Have I Been Pawned) can be used in this case.

-Excluding passwords with an obvious template. The password generator must check if the generated password matches a given template created from analysed leaked passwords.

Human bias attack is categorised as a template. To exclude passwords from these templates, password generation rules must be added. Examples of these rules are:

-A password must contain at least 1 or more characters with a frequency of under 10%;

-Password length must be over the median for any given template;

- At least one character subset frequency by position must be lower than 30%.

Conclusions

Creating passwords that do not obey the human bias generation rules in those brute-force algorithms can be more secure. Larger sets of data will increase the likelihood of new generation rules. Those rules can lead to more improvements in the speed of “guessing” a password created by a human. Testing such an algorithm is not feasible due to the long computing time. To test and improve the method for the generation of secure passwords suggested in this article artificial intelligence can be created. Using artificial intelligence the computation time can improve and lead to a more feasible computation time. This can be considered in father developments for the human bias attack.

References

Heule, M. J. H., O. Kullmann. 2017. The science of brute force. - *Communications of the ACM, 60, Issue 8*, 70–79.

Kerwin, T. 2006. Classification of natural language based on character frequency. – *Twenty-Eighth International Florida Artificial Intelligence Research Society Conference*.

Manning, C., H. Schutze. 1999. *Foundations of Statistical Natural Language Processing*. 712 p.

Mohammad, A., O. Saleh, R. A. Abdeen. 2006. Occurrences Algorithm for String Searching Based on Brute-force Algorithm. – *Journal of Computer Science, 2(1)*, 82-85.

Navor, P. 2021. *The Effects of Password Length and Complexity on Password Resiliency*. University of Hawai‘i–West O‘ahu.