Understanding password strength

author: Alberto De Bortoli
reviewer: Andrei Adrian Voineagu

Abstract—This short paper aims to pragmatically and briefly describe the issue of data security related to the choice of passwords regarding their strength.

1 Introduction

Choosing and handling password for Internet services, bank accounts or simply to protect sensible documents is something that nowadays every human being has to face with. The consequent problem of achieving the desired security is raised. Most of the people hardly understand the importance of protecting their data and this leads them to choose weak passwords for their accounts.
We define as *weak* a password which is possible to crack (i.e. to discover) in reasonable time using the best technology at the state of the art. Using the same technology, we say that *strong* passwords are hard to crack in human time, whilst *unbreakable* passwords are meant to be impossible to crack in human time. Password strength is a measure of the effectiveness of a password in resisting, guessing and brute-force attacks. A brute-force attack is meant to be a hacker attack that tries repeatedly all the possible strings in the solution space to search for the correct one that matches the password. What the paper aims to do is to explain and help the reader to understand how to write strong passwords. Different sets of possible characters combinations are identified and reasonable considerations are made on them (using only numbers, only lower case alphabet letters etc...).

August, 2011

2 Hint to choose passwords

We can anticipate, as common sense suggests, that a *strong* password is often made of a mix of digits, lower and upper case alphabetic characters and symbols. Memorize a complex strong password could be tricky so a hint is suggested to introduce the argument. As the first step, a common sense sentence need to be figured out. Here is an example:

Bob and Alice never had a better chance to fool Trudy, our very enemy.

We can then extrapolate the first char of every word and the punctuation symbols in the sentence. We retrieve the following string.

`B a A n h a b c t f T, o v e.`

Some modifications can be made, remembering the original meaning of the sentence. So we get to the following:

`B&Anh1bct4T,ove.`

A password with digits, symbols, lower and upper case letters is obtained. Using a method like this for remembering a password is easy and quick for everyone. We'll see that the password mentioned above belongs to what we'll define be set #10 and, most of all, it is *unbreakable*.

3 Sets of characters

In table 1 the different sets of characters are shown. With [a-z] we denote all the lower case alphabet letters, and with [A-Z] all the upper case alphabet letters. Considered symbols are:

```
! " # $ % & ' * + , - . / : ; \
< = > ? @ ( ) [ ] ^ _ ` { | } ~
```

Set #	Set	Chars
1	Digits	10
2	[a-z] or [A-Z]	26
3	Symbols	32
4	Digits + ([a-z] or [A-Z])	36
5	Symbols + Digits	42
6	[a-z] + [A-Z]	52
7	Symbols + ([a-z] or [A-Z])	58
8	Digits + [a-z] + [A-Z]	62
9	Symbols + [a-z] + [A-Z]	84
10	Digits + Symbols + [a-z] + [A-Z]	94

TABLE 1

Length	Possible combinations
1	10
2	100
3	1.000
4	10.000
...	...
10	10.000.000.000

TABLE 2

Consider a trivial example with only digits (set #1). Table 2 expresses how many possible combinations (i.e. how large is the solution space) there are using a specific length. The common exponential rule is applied as shown in the table 2.

It should be perfectly clear that more characters a password is made of, more the space of combinations becomes incredibly (exponentially) large.

4 Solution space

The user has to deal with a state of the art technology to be sure that a password is actually unbreakable. At the time of writing the fastest computer on earth has reached a speed of 76.1 billion password crack attempts per second [1]. With this in mind, calculations on cracking time needed are made.

Any kind of password shorter than 8 chars is considered (for us) *weak*. So considering a 10 length password from set #8 (digits + [a-z] + [A-Z]) there are exactly 62^10 possible combinations. This number is about 839.299.365.868.340.000. A pretty huge number for sure, but if we assume that somewhere in the world the incredible machine mentioned before is working with the target to crack this password, it would take about 128 days to examine the entire solution space.

Considering the 76.1 billions crack attempts per second we have

$$839.299.365.868.340.000 \;/\; 76.100.000.000 = 11.028.901 \text{ seconds} \approx 128 \text{ day-calculus computations.}$$

where a day-calculus is a full day working using the super machine mentioned above.

Things get more complicated as the length of the password grows as we can see in the stylesheet available at the bibliography URL[4].

At the time of writing, a good threshold for choosing a password and be sure that it is *unbreakable* is consider everything that takes more than 10.000 day-calculus to crack it. We mean at most 10.000 day-calculus as this is the time needed to examine the entire the solution space, but the password can be discovered in less time. Also taking into account the obsolete Moore's law [8], we should not worry about the unbreakableness of the passwords.
For example, a 13 length password from set #4 is considered unbreakable as a 12 length password from set #8 is, whilst a 9 length password from set #7 is definitely not.
Passwords longer than 8 characters that take less than 10.000 day-calculus to be cracked are just "strong" (for example, a 18 length from set #1 password can be cracked in 152 day-calculus).

The password mentioned in section 2 (`B&Anh1bct4T,ove.`) is 16 length from set #10, and the solution space is 94^16 large. This boils down to 5.651.285.632.241.030 day-calculus needed to explore the whole space, and we can conclude that it's pretty unbreakable.

Obviously a password can be stolen or retrieved using other ways (social engineering, computer theft...) to access user data but this is far away from the purpose of this paper. Moreover, in our scenario the cracker have some extra knowledge on the password, he knows the set the password belongs to, so he is aware what algorithm is best to use to do the cracking.

5 PRACTICAL SUGGESTIONS

Some softwares like "1Password"[3] generate passwords for user accounts and also manage them. Tipically the passwords are store in a "ad hoc" database that is crypted with a master key. All that the user has to do is remember only the master key to access the database and consequently the passwords. The master key should be chosen according to the hint proposed in this paper to be long enough to be considered unbreakable and to be easy to remember. Other software like "FileVault" (embedded in MacOS X) or "TrueCrypt" can crypt the user account home folder with a master key, so even if a thief stole the machine he would not find any plain data.

6 CONCLUSIONS

With these useful data in our hands it's easy to check the strength of a chosen password with the provided table in reference [4]. The choice of using a *strong* or *unbreakable* password is obviously up to the user.

REFERENCES

[1] Lockdown.co.uk, The Home Computer Security Centre. *Password Recovery Speeds*; http://www.lockdown.co.uk/?pg=combi, accessed December, 2010.

[2] Fischer Thorsten. Information Risk Management, *Everyday password cracking*; http://www.irmplc.com/downloads/whitepapers/Everyday_Password_Cracking.pdf2007, accessed December, 2010.

[3] Product page for "1Password" software, http://agilewebsolutions.com/products/1Password, accessed December, 2010.

[4] Table for "Brute-force Password Cracking", http://www.albertodebortoli.it/CS/PswStrength/Brute-force_password_cracking.xls 2010.

[5] Help Net Security. *Cracking one billion passwords per second with NVIDIA video cards*; http://www.net-security.org/secworld.php?id=6616 2008, accessed December, 2010.

[6] John P. One Man's blog, *How I'd Hack Your Weak Passwords*; http://onemansblog.com/2007/03/26/how-id-hack-your-weak-passwords, accessed December, 2010.

[7] Debasis Mohanty, *Demystifying Google Hacks*; http://www.hackingspirits.com/eth-hac/papers/Demystifying%20Google%20Hacks.pdf, accessed December, 2010.

[8] Wikipedia page about Moore's law, http://en.wikipedia.org/wiki/Moore's_law, accessed December, 2010.

Alberto De Bortoli Master graduate in Computer Science at the University of Padua (Italy). He is currently working as iOS developer at H-Umus, a company growing in H-Farm (Italy).
website: www.albertodebortoli.it

Andrei Adrian Voineagu Master graduate in Economics & Management at university Ca' Foscari, Venice (Italy). He is currently attending a 2nd master's degree in International Finance. He is currently working at Rabobank, Fixed-Income Product Control (Holland).