All about passwords (excerpts only)

A password in the virtual word is something like a lock in the world of reality. We all encounter passwords when entering Internet , using networks, receiving e-mail etc. Naturally, there are people out there who want to get your passwords. Here is some brief information about how they do it and how you can protect yourself. At the end of this page are some practical advices on how to choose a reliable password.

Sometimes a protection system has a breach of some sort which lets breaking it easily. Such was the password storage system in the original Windows '95 (this error corrected in OSR2). Yet, if the protection system you use had been designed properly the only remaining way of hacking is searching all passwords (although this search may be quite intelligent - read further about this). For a password search it is important to know how many passwords are possible and what is the search speed. The quantity of password variants equals to N^M, where N is the quantity of characters in a charset (i.e., the quantity of characters that may be virtually present in a password) and M is the maximum length of a password. By dividing this number by the speed of a search we obtain the T - the time for a complete search, that is, in the 'worst case' the time we have to wait to get a password. In the 'best case' the very first password will be the right one. The probability to find a password during the time of t equals to $p=t/T$, for example, a probability to find the password during the half of time indicated equals to 50%.

A password usually only contains letters. In this case the quantity of characters in a charset is 26 or 52, depending on usage of registers - both of them or just one. Some systems (Windows, for example) don't make any difference between lower-case and upper-case letters. Notice: in the first case the search time is 2^M times, not just 2 times, lesser (M is the length of the password). With an 8-characters' long password the difference would amount to 256 times, which is really significant. The following table will help you estimate what the aforementioned formulas actually mean.

The password search time with respect to the password and charset size. The search speed is supposed equal to 100,000 passwords per second (a very decent speed).				
psw length / charset	26 (no case, letters only)	36 (no case, letters&digits)	52 (case sensetive)	96 (all printable)
4	0	0	1 minute	13 minutes
5	0	10 minutes	1 hour	22 hours
6	50 minutes	6 hours	2.2 days	3 monthes
7	22 hours	9 days	4 monthes	23 years
8	24 days	10.5 monthes	17 years	2287 years
9	21 monthes	32.6 years	881 years	219,000 years
10	45 years	1159 years	45,838 years	21 million years

A user choosing a password and a hacker trying to increase the search speed compete on very unequal grounds. Should this user increase the password by one character, the hacker would have to increase the search speed 26 times (in the optimal case when only letters from one register are being used. In the worst case the time would grow 96 times. It is evident that no program can be optimized in this way, except for the most flagrant situations.

Nevertheless, things don't look so bleak for hackers. The point is that passwords are made up by living people, and many people are alike. Therefore people use some words more frequently than others. Hackers know these words. There exist frequency glossaries that list the most popular words. Good glossaries contain hundreds thousand words. Almost for sure, any word you can come up with is contained in such glossaries. Even if you take a professional term as a password, it can still be contained in a glossary (I have just looked up a dictionary for some terms, having remembered my physical education. ASTIGMATISM, HAMILTONIAN, DISPERSION, etc). Being original is the hardest thing in the world. Remember that a good frequency dictionary is not just a plain dictionary. A usual dictionary does not contain names of corporations, movie titles, trademarks etc. A good frequency dictionary is composed of actual passwords made up by people just like you. So, do not aspire to be

special. You could try introducing an error into a chosen word, yet the most popular modifications like FILEZ still can be found in a dictionary.

By the way, if you speak another language (even if a little bit) use words out of it as passwords. Hackers use English frequency dictionaries in English-speaking countries. In non-English speaking countries two dictionaries are to be used, English dictionary (as an international one) and a dictionary of a corresponding language. A password from some other language would be unexpected (you should only try to avoid using international words that are common for many languages).

In theory, the best option would be to have a computer make up a random password for you, but this option is usually unacceptable. For example, it's hard to remember a password, such as 'zmg74u5p'. Having such password leads to inconveniences; users tend to forget such passwords and in order not to forget they write them down. Such notes may be lost and be obtained by a hacker. Some cases are known when complex passwords are written by users on reverse of their keyboards, which does not promote security in any way. Therefore, we have to acknowledge that either the right to choose a password should be left to the end user, or a computer-aided password-generation system must be intelligent and make up 'humane' passwords.

The aforementioned facts are known not only to hackers, but to system administrators also. As a mean to counter the usage of a dictionary the latter use such a method: a password is composed of two words separated by a character, for example, a '$' sign. Another way is using a word and a number, like 'smart34'. Such a password is definitely better than a single word. But is it much better? Possessing such a technique, a hacker might complicate the search procedure, that is - pick words out of a dictionary and combine them. If the dictionary contains N words and the search speed is S passwords per hour than it would take $N*N*44/S$ hours ($44 = 128 - 32 - 26*2$, the quantity of printable non-text ASCII characters) to try all of the variants of the following type: '<word1><character><word2>'. This means pretty much time. Using huge dictionaries consisting of hundreds thousand words becomes impossible. Here's an example: it would take two days to search at the speed of 100,000 terms a second with a dictionary of 20,000 words. In case if a password is a word with an added number the search time would be equal to $100*N/S$, which is much lesser than in the previous case.

Let us consider a case sensitive password. Complete combination search using both upper- and lower-case would increase the quantity of variants by 2^M times (M is the password length). With the average length of a word in a dictionary equal to 9 characters, the quantity of variants grows 512 times. Nine characters is the actual average password length in a frequency dictionary. You may be surprised how come English words are so long, yet there are two reasons for it. Firstly, short one- two- and three-letter words are not included into dictionaries at all; such passwords can be easily found by a simple search. Secondly, many users follow the trend to use long passwords and complex two-word passwords.

Another method may be put to use when approximate spelling of a password is known. For example, a situation when a keyboard is set to CAPS LOCK when a password is being chosen is pretty frequent. As a result a user may think that the password consists of lower-case letters, although it actually is written in upper-case.

Sometimes a password may be cracked in yet another exotic way. Usually every person has one or more favorite passwords. Often the same password is used for several resources. Therefore, having obtained a password in one of the ways a hacker might try to use it with other resources.

Let us now consider a case when a password cannot be guessed in any way. In this case we are left nothing to do but search all of the variants. This way of cracking is usually called a 'brute force attack'. It takes a great amount of resources to crack even a not very long password. The search may be performed in background which might allow using a computer otherwise. Intermediate variants may be saved to disk in order to enable continuing the search the next day. The search may be distributed among several computers which could lead to an increase in the search speed by the number of times that corresponds to the quantity of computers. Here's an exotic option: a search may be performed by a Java applet which is located on some very popular site and is controlled from this site. Dozens thousand visitors may be giving computation power of their computers to crack the password without even suspecting it. Nevertheless, a brute force attack will not help you to break a 10-character password despite your finesse, even if all the letters of the password are in the same case. But why smart guys must use brute force? For the most part there are combinations like jkqmzwd which are totally senseless among billions and trillions of passwords being searched. I attended this problem some time ago. My purpose was creation of a search algorithm which would only attempt 'reasonable' passwords. I named this algorithm 'smart force attack' as opposed to 'brute force attack' methods. There is a separate page dedicated to this technology. Briefly, here are the results. Having the search speed of 100,000 passwords a second and a case insensitive password which only contains letters it takes only 8 days (for a complete search of variants with a non-zero probability) or 10 hours (excluding the low-

probability options) instead of 45 years! Smart force attack only works for letters-only passwords (i.e., not containing any digits or special characters).

I want to emphasize the following sad fact. In the vast majority of cases the time it takes to verify a password is not critical for the system. Namely, if Windows spent half a second to verify a password, users would hardly even notice it. But if we consider this we see that it would take 20 hours to pick a correct password of three letters and 42 days for a four-letter one. It is so easy to implement! It is hard to make an algorithm faster than it is, yet it takes absolutely no effort to slow it down. It would be sufficient to use slow and/or multiply repeated algorithms. Nevertheless, this simple and efficient method is often neglected. Currently I'm working on a program which would replace existing system of password storage in Windows '95, making it safer.

Here's one more important issue. How valuable is your password? What if it gets cracked? A lot rides upon the answers to these questions, like what time of search may be considered acceptable and what computation power may be mobilized to accomplish the task. Would a dozen computers working for a week be acceptable for a hacker? The answer is a definite 'no', if their work results in getting access to your hard disk where different kinds of software junk are stored. And the answer is 'heck why not' if the end result is the key to your savings account.

What to do?

1. Firstly, use password organizers. A good organizer will automatically generate different reliable passwords for all of your resources. You'll only have to come up with and remember one master password. Thus, you are going to ease your life and improve the security. Unfortunately, I do not take on a task of recommending you a good password organizer. The reason is that I don't use them and generally do not follow my own recommendations. Hope you'll be more careful than I. :-)

2. Make your passwords long enough. For a well-built security system four characters should be enough, for Windows at least nine is needed. If you are not sure, prepare for worst and use passwords from nine characters and longer. Use even longer passwords to protect the most valuable information. Somewhere at the level of the characters there is a limit above which there is no sense to increase the length of the password. Keep in your mind that many systems will limit the length of your password to N characters. In this case all of your efforts may be vain.

3. Use different characters. A mark which separates two words seems not to be enough anymore. It is a good idea to use three possibly even shorter words, not just two of them. *I$am@ok!* is a really good password, and a very simple, at that. Do not use a single word as a password use at least two! Use more non-alphabetic characters, and use both cases if the password system is case-sensitive.

4. Remember that the reliability of a system is defined by its weakest component. You may choose the most reliable password out of 20 characters, but if you are going to use it with MS Word v7.0, it's going to be cracked within a fraction of a second.

information available at http://passwordtools.com