Your Password Is No Longer Secret

Stoyan Rachev

Your Password Is No Longer Secret

Stoyan Rachev

This book is for sale at http://leanpub.com/yourpasswordisnolongersecret

This version was published on 2013-03-17

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Password Vulnerability 1
Password Strength 2
Modern Password Cracking 4
How Strong a Password? 9
Conclusion 10

% Your Password Is No Longer Secret, Part 1 % Stoyan Rachev %

Written by Stoyan Rachev[1]

Of course, the title is a trick. Your password is still secret, for now. To be sure that it will remain so, try to answer the following questions to yourself:

- **How strong** are your passwords?
- How strong they **should be** in order to prevent other people from revealing them?
- Are your password habits really **adequate**?

Here, I assume that you are an Internet user with some experience. You don't use simple or common passwords. Your passwords are at least 8 characters long. You mix letters, numbers, and special symbols. You never use the same password for multiple accounts, at least not for important accounts.

Still, answering the above questions with certain confidence can be somewhat challenging. Also, the answers that would have been valid just a few years ago no longer hold. Modern computing advancements have invalidated many former assumptions, to the point that the entire concept of passwords can be seen as severely compromised.

In this short blog series, I will be exploring passwords somewhat deeper. In this first post, I will try to help you answer the first two questions above by taking a closer look at password strength, cracking a few passwords on my home PC, and finally putting it all together to come to a definite answer. In the next post, I intend to examine more closely password creation and password habits in general.

Password Vulnerability

There are two factors to consider when determining the vulnerability of a password to various types of attacks:

- the *password strength* itself, that is the average number of guesses the attacker must test to crack it, usually measured by its *entropy*
- the *speed* with which an attacker can check the validity of each guess, usually measured by "password guesses per second" (p/s).

While the first factor is under your (the user) direct control, the second factor is determined entirely by how the password is stored and used, and is therefore beyond your control. Most security systems introduce measures to severely limit the testing speed, usually by imposing a timeout after a small number of failed attempts. In such circumstances, the testing speed rarely exceeds 1000 p/s (usually

[1] http://twitter.com/stoyanr

much lower). However the system must store the passwords in some form and if this information is stolen, the situation gets much worse.

To reduce the above risk, most systems store only a cryptographic hash[2] of the password instead of the password itself, using hashing algorithms such as [MD5][md5] or SHA1[3]. Such hashes are very hard to reverse, so an attacker who gets hold of the hash cannot directly recover the password. However, knowing the hash allows the attacker to test guesses offline much faster.

Should you, as an user, worry about the above worst-case scenario? Well, it is not very likely for any individual account, but certainly far from impossible. Only in 2012, hackers got hold of millions of password hashes in [security breaches at Yahoo, LinkedIn, eHarmony, and last.fm][5*worst_security_breaches*2012], among others. I personally had to change three of my passwords during the year due to these incidents. If you, like many Internet users, have accounts at tens or hundreds of Web sites, then the probability at least one of those hashes to fall into the wrong hands just in the next one year is actually quite high.

There are also situations in which quick guessing is simply always possible. This is when the password is used to form a cryptographic key, e.g. for [PGP][pgp] or [WPA][wpa].

The conclusion is that since you, as a user, don't have control over how your password is stored and used, you should assume that sooner or later, a hacker will be able to attempt to crack it offline. The two factors that influence the outcome are again the password strength, as well as the performance of the computing equipment that the hacker has at his disposal.

[md5]: http://en.wikipedia.org/wiki/MD5 "" [5*worst_security_breaches*2012]: http://www.symform.com/blog/5-worst-security-breaches-2012/ "" [seven_incidents]: http://www.crn.com/slide-shows/security/240003727/breaches changes-seven-incidents-that-remind-us-about-password-integrity.htm "" [pgp]: http://en.wikipedia.org/wiki/Prett Good_Privacy "PGP (Wikipedia)" [wpa]: http://en.wikipedia.org/wiki/Wi-Fi_Protected_Access "WPA (Wikipedia)"

Password Strength

According to Wikipedia, Password strength[4] is a measure of the effectiveness of a password in resisting guessing and brute-force attacks. It estimates how many trials an attacker would need, on average, to crack it. The strength of a password depends on the following factors:

- length
- complexity, the size of the used character set
- unpredictability, whether it is created randomly or by a more predictable process

[2] http://en.wikipedia.org/wiki/Cryptographic_hash

[3] http://en.wikipedia.org/wiki/SHA-1

[4] http://en.wikipedia.org/wiki/Password_strength

Entropy

Password strength is usually measured in *bits of entropy*. This is simply the base-2 logarithm of the number of guesses needed to find the password with certainty. A password with let's say 42 bits of entropy would require 2 attempts to try all possibilities during a brute force search. Note that on average, an attacker needs to try half the possible passwords to find the correct one.

It is fairly easy to calculate the entropy of truly random passwords. If a password with length L is generated randomly from a set of N possible symbols, the number of possible passwords is N. Therefore the entropy H is given by the formula:

$$H = log_2 N^L = L \log_2 N = L \frac{\log N}{\log 2}$$

When considering human-generated passwords, which are not truly random, the situation gets much more interesting. To remember password more easily, humans tend to use words from a natural language, which has non-uniform distribution of letters, as well as predictable patterns of capitalisation and adding numbers or special symbols. Since all these can be exploited by cracking programs, such passwords have much lower entropy than a truly random password of the same length. To make it worse, it's also much harder to correctly estimate their entropy.

Let's take the password *Admin#123* as an example. This password is of length 9 and uses small and capital letters, numbers, and special symbols, in other words the full set of printable ASCII characters, which is of size 95. A cracking program that only takes this into account has to make 95, or 630,249,409,724,609,400 total attempts, resulting in an entropy of 59.1.

However, this is far from optimal. A smarter cracking program could take advantage of the following facts:

- "admin" is a common 5-letter English word. A program doing a dictionary attack based on up to 5-letter long common English words has to make less than 2000 attempts, resulting in an initial entropy of about 11.
- Only the first letter is capitalised, an extremely common pattern. A program that tests just this pattern has to test each word just twice, which adds just 1 to the total entropy.
- The numbers and special symbols are added at the end of the word, another common pattern. A program that tests for numbers and special symbols at the beginning or the end of the word has to test each word 11 times. This adds another 3.5 bits.
- There is just one special symbol, and it is one of the 10 symbols in the upper row of the keyboard, above the digits. This adds another 3.3 bits.
- The number pattern "123" is also quite common. A program that tests common number patterns of up to this length, e.g. for equal or consecutive digits, would need to make less than 100 attempts per word, which adds another 6.6 bits.

Thus, assuming a fairly smart cracking program, we arrived at an estimation of just about 25.4 bits.

There are other, easier to use methods for estimating the strength of human-generated paswords, but none of them is really proven in practice:

- Using online strength test calculators such as [Rumkin.com][strength_test]. This calculator, which only takes some of the above facts into account, estimates the entropy of *Admin#123* as 35.5.
- The [NIST Electronic Authentication Guideline][nist_eag] proposes a very conservative method, according to which the entropy of *Admin#123* is estimated as 25.5, pretty close to the above.

For best results, I would recommend using more than one method and taking the middle value. In our case, we would come to a final estimation of about 30 bits of entropy.

[entropy_formula]: http://upload.wikimedia.org/math/6/2/7/6270d629826e5df0949332423566dd78.png
“” [strength_test]: http://rumkin.com/tools/password/passchk.php “Strength Test” [nist_eag]: http://csrc.nist.gov/pu
63/SP800-63V1*02*.pdf “NIST Electronic Authentication Guideline”

Modern Password Cracking

Unfortunately for users, the technology progress in just the last 5 years has introduced tools that are radically better at password cracking than anything known to that point. Rather surprisingly, this technology progress happened in a seemingly unrelated area, namely video gaming.

In response to the increasing demand for better 3D gaming experience, nVidia and ATI kept boosting the performance of graphics processing units (GPUs)[5] in common video cards. Modern GPUs have thousands of processing cores and teraflops of computing performance. In 2007, a method of using these devices for password cracking was invented, and software that implements it soon became commonly available.

Password cracking algorithms can be trivially parallelised, which makes GPUs particularly suitable for this task. To make it even worse, parallel computing frameworks such as OpenCL[6] and virtualisation libraries such as VCL[7] allow such algorithms to use a high number of graphics cards in a cluster, with cracking performance scaling nearly linearly with each GPU added.

Security experts have only started to [take notice][gpu_cracking_article] of this threat, and companies have so far largely failed to respond appropriately. We still use the same password creation policies and guidelines as 5 or 10 years ago. How do “strong” 8-character passwords with a max entropy of 52 bits (much lower if human-generated) fare against GPU-powered crackers? In short, they suck. In the next sections, we will see how much exactly.

[gpu_cracking_article]: http://www.defendingthenetwork.com/blog-home/2011/6/19/how-to-crack-your-ntlm-password-in-under-a-second-the-power.html “”

[5] http://en.wikipedia.org/wiki/Graphics_processing_unit
[6] http://en.wikipedia.org/wiki/OpenCL
[7] http://www.mosix.org/txt_vcl.html

Cracking Software

There are several GPU-powered password cracking programs available, both commercial and free. Here, I would like to mention just two of the free alternatives:

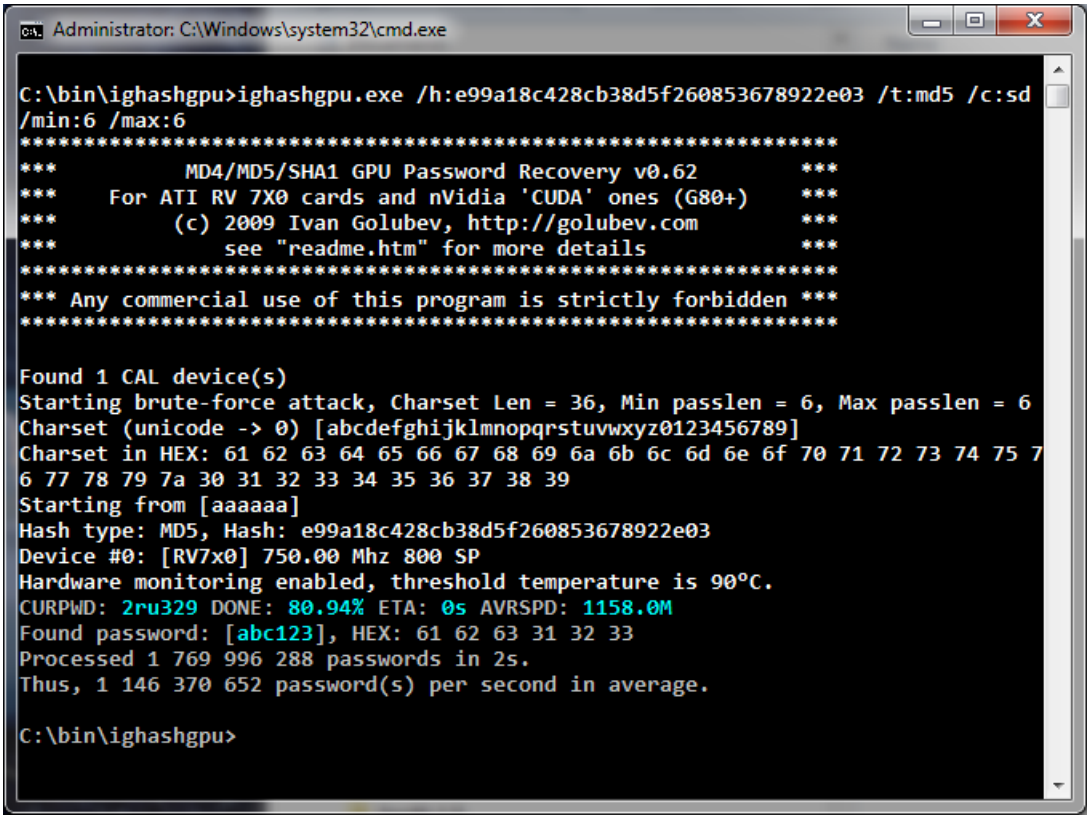
- IGHASHGPU[8] was one of the first such tools developed in 2009 - 2010 by Ivan Golubev[9]. It supports a limited number of hashing algorithms and attack modes, and is no longer actively developed or maintained.
- oclHashcat-plus[10] is arguably the best non-commercial password recovery tool available today. It supports a much larger set of hashing algorithms and attack modes, including a rule engine to script your own sophisticated attacks. It is actively developed and maintained and has a growing ecosystem of related tools, many of which open source.

Cracking Tests on My Home PC

I happen to have a 4-year old [ATI Radeon HD 4870][ati_radeon_4870] video card on my home PC. So far it didn't see much use in 3D gaming, but lately it experienced a few hours of full load when I did some password cracking tests on it with IGHASHGPU.

The first thing I tried was a simple brute force search of *abc123* hashed as MD5. This ended pretty quickly.

[8] http://golubev.com/hashgpu.htm
[9] http://www.golubev.com/blog/
[10] http://hashcat.net/oclhashcat-plus/



Well, *abc123* isn't exactly a secure password, and MD5 is a relatively weak algorithm compared to SHA1. So I proceeded with more complex passwords, using both algorithms. The results for MD5 are given in the table below.

Password Charset Size Length Entropy Speed(Mp/s) ETA Actual Time ———————- ———————- ———- ———- ———————- ———————- ———————- abc123 36 6 31.0 1100 2s 2s abcd1234 36 8 41.4 1100 41m 32m 7s Admin#123 72 9 55.5 1100 1y 200d ? Admini$>123 95 11 72.3 1100 162000y ? Admini$tr>123 95 13 85.4 1100 Next Big Boom ? Admini$trat0r>12 95 16 105.1 1100 Next Big Boom ? ———————- ———————- ———- ———- ———————- ———————- ———————-

In the last 2 cases, the program actually displayed "Next Big Boom" as an estimated time. I decided not to wait that much.

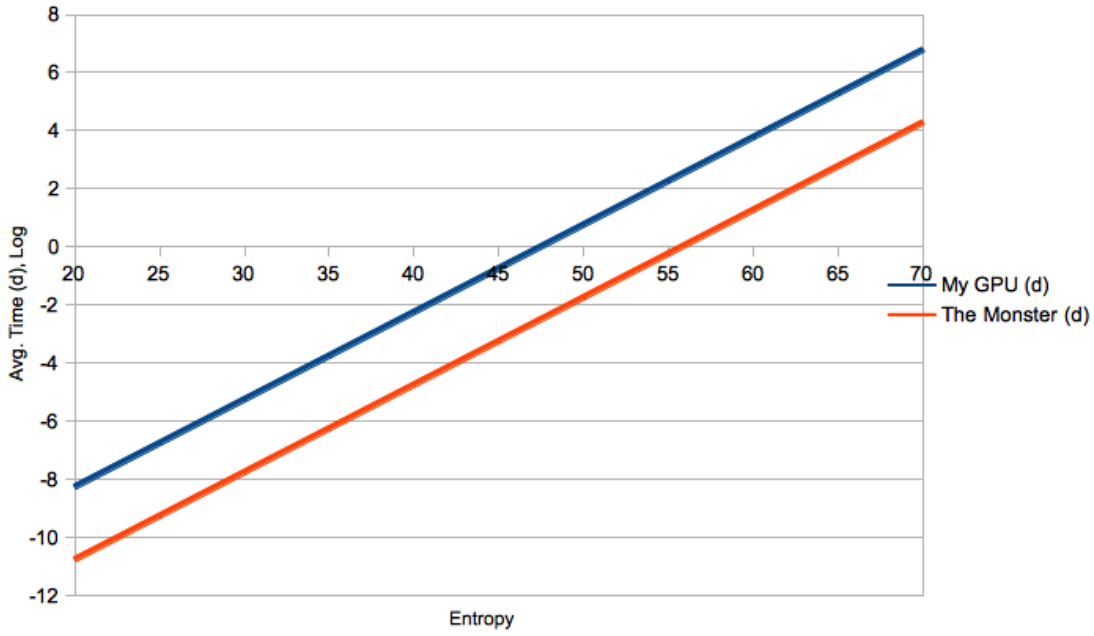
The times for SHA1 were consistently about 3 times longer. The important thing to notice is the speed: on my outdated GPU it is **1100 Mp/s for MD5** and **360 Mp/s for SHA1**. This is far better than what could be achieved on the fastest Intel i7 processor currently available.

[ati_radeon_4870]: http://www.amd.com/us/products/desktop/graphics/ati-radeon-hd-4000/hd-4870/Pages/ati-radeon-hd-4870-overview.aspx ""

Cracking Speeds on Modern GPUs

As you would expect, the performance of my 4-year old video card is nothing special compared to modern equivalents, especially if they are assembled together in large clusters. The following [article][gpu_cluster_article] from December 2012 describes a 25-GPU cluster of modern ATI video cards which reportedly can do brute force search on NTLM hashes with a speed of **350 Bp/s**. Since NTLM hashes are only slightly weaker than MD5 hashes, this is actually about **300 times** faster than my GPU.

The diagram below charts the cracking times in days of my GPU and that monster per entropy. With simple brute force search, the time doubles with each bit of entropy. To visualize this better, the base-10 logarithm of the time is used.



In a single day, my GPU could crack a password of 47 bits of entropy, while the Monster would happily crunch a much more complex password of 56 bits of entropy. The average time to find any 8-character password of 52 bits of entropy on the Monster is just about 2 hours.

[gpu_cluster_article]: http://www.techspot.com/news/51044-25-gpu-cluster-can-brute-force-windows-password-in-record-time.html “”

Speed Per $

People that build 25-GPU clusters could also build much larger ones, given sufficient resources. Therefore, instead of looking at the speed of any GPU cluster with a given size, we should rather look at the *speed per $* which is achievable today and in the near future.

To do this, I first combined the GPU speed estimations[11] for various ATI video cards provided by Ivan Golubev with the comparison of ATI GPUs[12] in Wikipedia. For SHA1, I came to the following picture:

Card Year Price ($) Units Clock GFLOPS Speed,Mp/s Speed / GFLOPS ———————– —- ———- ——— ——- ———- ——— ——————— ATI Radeon HD 4770 2008 640 750 960 352 0.37 ATI Radeon HD 4870x2 2008 560 1600 750 2400 880 0.37 ATI Radeon HD 5870 2009 1600 850 2720 1360 0.50 ATI Radeon HD 5970 2009 599 3200 725 4640 2320 0.50 ATI Radeon HD 6970 2010 369 1536 880 2703 1408 0.52 ATI Radeon HD 6990 2011 699 3072 830 5100 2656 0.52 ATI Radeon HD 7950 2012 449 1792 800 2867 1493 0.52 ATI Radeon HD 7970 2012 549 2048 925 3789 1973 0.52 ATI Radeon HD 8970 2013 499 2048 1000 4096 ? ?
———————– —- ———- ——— ——— ———- ——— ———————

The price above is the release price of the card. The GFLOPS value is calculated by multiplying the processing units to their clock and then by 2, and finally dividing by 1000. The last value, *Speed / GFLOPS*, is a factor which represents how well the available GFLOPS can be used on any given GPU for password cracking. It tends to be the same for GPUs of the same family. Over the last 5 years, this number has initially increased quickly and then reached a plateau.

Based on the above table, one could easily calculate the speed per $ for any given year after 2008. I decided to go further and extend this for the next 5 years as well. The results below are again for SHA1.

Year GFLOPS Price ($) GFLOPS / $ Speed / GFLOPS Speed / $ —- ——- ———- ———- ——————- ———- 2008 2400 560 4.29 0.37 1.59 2009 4640 599 7.75 0.50 3.87 2010 2703 369 7.33 0.52 3.81 2011 5100 699 7.30 0.52 3.79 2012 3789 549 6.90 0.52 3.59 2013 4096 499 8.21 0.52 4.27 2014 8.66 0.53 4.55 2015 9.12 0.53 4.83 2016 9.58 0.54 5.12 2017 10.03 0.54 5.42 2018 10.49 0.55 5.72 —- ——- ———- ———- ——————- ———-

To come to the above numbers, I have made some arbitrary assumptions:

- The *GFLOPS / $* will increase every year at the rate it increased between 2013 and 2011. This is conservative compared to let's say 2009 and 2008.
- The *Speed / GFLOPS* will increase every year at the rate it increased between 2013 and 2009. This is also somewhat conservative.

The above calculations only take into account the price of the cards themselves, ignoring other cost components such as electricity, other hardware, personnel, etc. Another thing not taken into

[11] http://golubev.com/gpuest.htm

[12] http://en.wikipedia.org/wiki/Comparison_of_AMD_graphics_processing_units

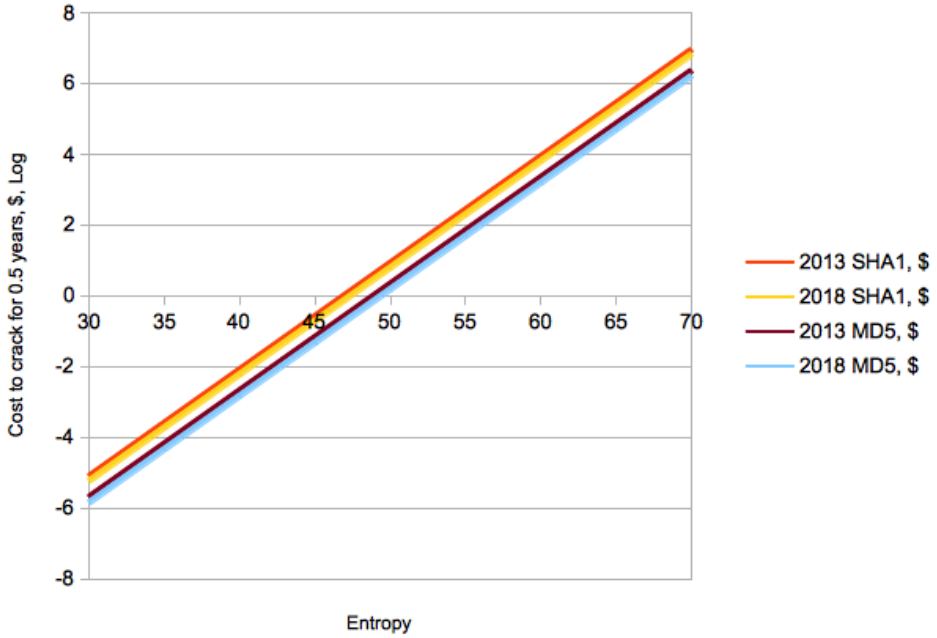
account is that it will be more effective economically to use older cards with more favorable price / performance ratio, rather than brand new ones. Finally, I assume that the hashes are properly salted and cracking has to be done via brute force search and not some other method such as [rainbow tables][rainbow_table].

[rainbow_table]: http://en.wikipedia.org/wiki/Rainbow_table “”

How Strong a Password?

As a user, to estimate how strong should your password be, you should find the answer to the following question: for a given the hashing algorithm *A*, **how much entropy is needed in order cracking the password in *T* time to cost *X* money**, today and in the near future? Note that the variables *T* and *X* can be chosen differently based on your habits (for example, how often do you change your password), and actual needs (for example, the cost of resources that this particular password protects). Since you normally don’t have any knowledge about the hashing algorithm, you should assume a weaker one such as MD5.

The above question can be easily answered using the numbers from the previous section. The diagram below charts the cost in $ against the entropy for *T* of half an year, again in logarithmic units, for SHA1 and MD5, today and in 5 years.



As you can see, in order for cracking your password to cost anything at all, it should be of at least 50 bits. Also, all functions cross the 1,000,000$ mark between 65 and 70 bits. Therefore, passwords of 70 bits or more will cost at least few million dollars to crack for the next few years, independently of the hashing algorithm used.

Based on the above considerations, I consider **passwords of 70 bits of entropy or more to be sufficiently strong** for my needs and that of most Internet users for protecting anything of some importance. On the other side, I consider **passwords of 50 bits of entropy or less to be inadequate** for this purpose.

Conclusion

Now that you know that a password of 70 bits of entropy is sufficiently strong, protecting your important accounts is a simple matter of creating such passwords for all of them, right? Here is the trick: remembering and typing even a single password of genuine 70 bits of entropy is anything but simple. In fact, most users would find it quite hard.

To give you and myself some time to reflect on this, I would like to take a pause here and explore this topic further in my next post.