# The Usability of Passwords

**Source :** http://www.baekdal.com/tips/password-security-usability

Written by Thomas Baekdal | Saturday, August 11, 2007

Security companies and IT people constantly tells us that we should use complex and difficult passwords. This is bad advice, because you can actually make usable, easy to remember and highly secure passwords. In fact, usable passwords are often far better than complex ones.

So let's dive into the world of passwords, and look at what makes a password secure in practical terms.

**Update: Read the FAQ (updated January 2011)**

**Update – April 21, 2011: This article was "featured" on Security Now, here is my reply!**

## How to hack a password

The work involved in hacking passwords is very simple. There are 5 proven ways to do so:

1. **Asking**: Amazingly the most common way to gain access to someone's password is simply to ask for it (often in relation with something else). People often tell their passwords to colleagues, friends and family. Having a complex password policy isn't going to change this.
2. **Guessing**: This is the second most common method to access a person's account. It turns out that most people choose a password that is easy to remember, and the easiest ones are those that are related to you as a person. Passwords like: your last name, your wife's name, the name of your cat, the date of birth, your favorite flower etc. are all pretty common. This problem can only be solved by choosing a password with no relation to you as a person.

3. **Brute force attack**: Very simple to do. A hacker simply attempts to sign-in using different passwords one at the time. If you password is "sun", he will attempt to sign-in using "aaa, aab, aac, aad ... sul, sum, **sun (MATCH)**". The only thing that stops a brute force attack is higher complexity and longer passwords (which is why IT people want you to use just that).

4. **Common word attacks**: A simple form of brute-force attacks, where the hacker attempt to sign-in using a list of common words. Instead of trying different combination of letters, the hacker tries different words e.g. "sum, summer, summit, sump, **sun (MATCH)**".

5. **Dictionary attacks**: Same concept as common word attacks - the only difference is that the hacker now uses the full dictionary of words (there are about 500,000 words in the English language).

## When is a password secure?

You cannot protect against "asking" and "guessing", but you can protect yourself from the other forms of attacks. A hacker will usually create an automated script or a program that does the work for him. He isn't going to sit around manually trying 500,000 different words to see if one of them is your password.

The measure of security must then be "**how many password requests can the automated program make – e.g. per second**". The actual number varies, but most web applications would not be capable of handling more than 100 sign-in requests per second.

This means it takes the following time to hack a simple password like "**sun**":

- Brute-force: **3 minutes**
- Common Word: **3 minutes**
- Dictionary: **1 hour 20 minutes**

Note: "sun" has 17,576 possible character combinations. 3 letters using the lowercase alphabet = $26^3$

This is of course a highly insecure password, but how much time is enough for a password to be secure?

- a password that can be hacked in **1 minute** is far too risky
- **10 minutes** - still far too risky
- **1 hour** - still not good enough
- **1 day** - now we are getting somewhere. The probability that a person will have a program running just to hack your account for an entire day is very little. Still, it is plausible.
- **1 month** - this is something that only a dedicated attacker would do.

- **1 year** - now we are moving from practical risk to theoretical risk. If you are NASA or CIA then it is unacceptable. For the rest of us, well - you do not have that kind of enemies, nor is your company data that interesting.
- **10 years** - Now we are talking purely theoretical.
- **A lifetime: 100 years** - this is really the limit for most people. Who cares about their password being hacked after they have died? Still it is nice to know that you use a password that is "secure for life"

But let's take a full swing at this. Let's look at "100 year - secure for life". It has good ring to it and it makes us feel safe. There is still the chance that the hacker gets lucky. That he accidently finds the right password after only 15 years instead of 100. It happens.

Let's step that up too and go for the full high-end security level. I want a password that takes **1,000 years to crack**- let's call this "**secure forever**". That ought to be good enough, right?

## Making usable and secure passwords

Now that we have covered the basics, let's look at some real examples, and see just how usable we can make a password, while still being "secure forever".

Note: The examples below are based on 100 password request per second. The result is the approach that is the most effective way to hack that specific password - either being by the use of brute-force, common words or dictionary attacks.

First let's look at the common 6 character password - using different methods:

| Type | Password | Method | Time | Security level |
|---|---|---|---|---|
| **6 random characters** | jskerv | Brute-force | 1 month | risky |
| **6 random characters with numbers** | ergs43 | Brute-force | 8 months | Low risk |
| **6 random characters with mixed case, symbols and numbers** | J4fS<2 | Brute-force | 219 years | Secure for life |
| **6 character common word** | orange | Common words | 3 minutes | useless |
| **6 character uncommon word** | woosaa | dictionary | 1 hour 22 minutes | useless |

In this example complexity clearly wins. Using a password with mixed case characters, numbers and symbols is far more secure than anything else. Using a simple word as your password is clearly useless.

Does that mean that the IT-departments and security companies is right? Nope, it just means that a 6 character password isn't going to work. None can remember a password like "J4fS<2", which evidently mean that it will be written on a post-it note.

To make usable passwords we need to look at them differently. First of all what you need is to use words you can remember, something simple and something you can type fast. Like these:

| Type | Password | Method | Time | Security level |
|---|---|---|---|---|
| **2 common word password** | alpine fun | Common word | 2 months | Low risk |
| **3 common word password** | this is fun | Common word | 2,537 years | Secure forever |

Using more than one simple word as your password increases you security substantially (from 3 minutes to 2 months). But, by simply using 3 words instead of two, you suddenly got an extremely secure password.

It takes:

- 1,163,859 years using a brute-force method
- 2,537 years using a common word attack
- 39,637,240 years using a dictionary attack

**It is 10 times more secure to use "this is fun" as your password, than "J4fS<2".**

If you want to be insanely secure; simply **choose uncommon words** as your password - like:

| Type | Password | Method | Time | Security level |
|---|---|---|---|---|
| **3 uncommon word password** | fluffy is puffy | Dictionary | 39,637,200 years | Secure forever |
| **5 uncommon word password** | du-bi-du-bi-dub | Brute-force | 531,855,448,467 years | Secure forever |

A usable and secure password is then not a complex one. It is one that you can remember - a simple password using 3+ words.

It is not just about passwords

One thing is to choose a secure and usable password. Another thing is to prevent the hacker from hacking password in the first place. This is a very simple thing to do.

All you need to do is to prevent automatic hacking scripts from working effectively. What you need to do is this:

1. **Add a time-delay between sign-in attempts.** Instead of allowing people to sign-in again and again and again. Add a 5 second delay between each attempt.

   It is short enough to not be noticeable (it takes longer than 5 seconds to realize that you have tried a wrong password, and to type in a new one). And, it forces the hacker to only be able make sign-in requests 1 every 5 seconds (instead of 100 times per second).

2. **Add a penalty period** if a person has typed a wrong password more than - say - 10 times - of something like 1 hour. Again, this seriously disrupts the hacking script from working effectively.

A hacker can hack the password "alpine fun" in only 2 months if he is able to attack your server 100 times per second. But, with the penalty period and the 5 second delay, the same password can suddenly sustain an attack for 1,889 years.

| No of attacks | Password | Time | Security level |
|---|---|---|---|
| **100 times per sec** | alpine fun | 2 months | Low risk |
| **1 time every 5 sec** | alpine fun | 63 years | Secure |
| **1 time every 5 sec with a 1 hour penalty period after 10 attempts** | alpine fun | 1,889 years | Secure forever |

Remember this the next time you are making web applications or discussing password policies. **Passwords can be made both highly secure and user-friendly**.

**Update: Read the FAQ (updated January 2011)**

Back in 2007, I wrote an article about how to make usable and secure passwords. It is one of the most popular articles I have ever written, with more than 340,000 unique readers (and counting).

The article has been linked to by a large number of sites, including several of the big security companies, and last week it was picked up by ReadWriteEnterprise.

Over the years, people have been asking the same questions time and again. So in this article, I am going to answer those questions once for all.

## Why did I write that article?

The article came to life after yet another discussion with IT, who believed that everyone should be forced to use password with a minimum of eight characters, including two uppercase characters, numbers and a least one special character.

I was absolutely furious for several reasons. First, I knew it was like kicking every employee in the groin every morning they showed up for work, that it would do squat for actual security (it is likely to make it worse), and that it would completely destroy the plans I had for password free web application I was working on at the time.

So I wrote this article about how to create passwords that were really easy to use and remember. I wanted to demonstrate that complexity is a sickness of IT, and has nothing to do with actual security.

I illustrated how the simple password "**this is fun**" is **10 times more secure than "J4fS<2"**.

With this the problem was solved. Gone was the ridicules IT security crap, and we could all go on with our lives. But then people started asking these questions.

**Q: What about the problem that many sites store people's passwords in clear text?**

A: What about it? It has nothing to do with this. How a password is stored in a database is a server problem, something that must be solved a server level. No level of complexity by the user can solve this problem.

Every server admin has the responsibility to store people's passwords in a secure and encrypted way.

**Q: Most password can be cracked using rainbow tables (or similar), will a higher complexity not solve that?**

A: Yes and no.

Let me briefly explain what this is. The way hackers hack passwords today is to look it up in password tables. This was how people's password was hacked in the Gawker incident.

Basically, the hacker will take the encrypted password like this one: 4d5257e5acc7fcac2f5dcd66c4e78f9a and simply go to this site (among many) and

paste it in. The site will then return the actual password (which in this case is "mickey"). They are simply looking it up in a database.

The way to solve this is to make the password so complex that it is unlikely to be in any database. More complexity = better encryption security.

**However, this is not a user problem!!!**

You do not ask the user to create a more complex password. You make the password more complex on the server.

Here is a simple example. Let's say that the user decides to use the password "mickey", which is completely insecure.

You then add a one-way complexity algorithm to it, effectively turning it into:

U**c**([u+**e**>q**#i**Z|Xrhl@@H**k**Cfnd=R~5"@and8T:[Z6<A|16n<nw**m**kwV{)H4k'[@f|CR**y**WK;-1

Which you then encrypt, and run trough another one-way complexity algorithm - ending up with this:

Q802AlII0xEKCgjwyXL8lsteSOEftjYgyQc0FW6Dit8F0onuv0gNvv4Xm0cYwCe

This is then what you store in your database on the server. The user can choose whatever password they like. It is your job as a server admin to make sure that it is encrypted correctly.

**Q: Many websites do not allow spaces in password, what then?**

A: True, but again that is a server problem, not a user problem. Fix the damn server!

**Q: If I cannot write "this is fun" because of the spaces, can I not just write "thisisfun"?**

A: Absolutely not! The reason why "this is fun" is 10 times more secure, is simply because it is much longer (11 characters). By removing the spaces, you reduce the length and the complexity substantially. The spaces are effectively special characters, which in itself makes the password much more secure.

Use "this-is-fun" instead.

**Q: If "this is fun" is 10 times more secure, wouldn't it increase security to write it as "Thi3-Is-5un"? ... That would be just as easy to remember.**

A: You are kidding right?

Yes it would be much more secure, but the whole point of this is to reduce complexity. People do not like being kicked in the groin every morning.

There is absolutely no reason for adding that complexity in the first place. It takes 2,537 years to hack "this is fun"

**Q: But modern computers can hack password much faster than they used to...**

A: Yes and no. You need direct access to the database file or the server to hack something faster. If you got that level of access, you do not actually need the password. You can just look up the data directly.

What I am talking about in the article is hacking into remote systems (like web apps). I tested this with Google and I was not able to send more than 25 password request per second to their servers.

And even if you could hack it faster, "this is fun" is still more secure. Just because something is faster doesn't mean the math behind it changes.

**Q: Even so, a BOT net could do it, right?**

A: Read that part about adding delays in the article.

**Q: "Well, I still prefer long and complex passwords. My previous one, for example, had mixed case letters, numbers, symbols, and spaces, and were 32 characters long. But I still could remember it easily, by using the name and ID number of an obscure science-fiction character from an uncommon book combined with a made up language."**

A: Please, go away...

Ohh... and read: A template for every awful Facebook discussion you've ever witnessed.

**Q: When I test "this is fun" it shows up as weak in most password testers.**

A: Yes, this is simply because most password testing tools are completely useless. They measure complexity, not security.

One example. If you head over to The Password Meter, they will tell you that "this is fun" only scores 19% = very weak, while "J4fS<2" scores 60% = strong.

But, this has nothing to do with security. They specifically look for the presence of uppercase letters, numbers and symbols, which they then give a rank using a completely insane algorithm.

These guys are only measuring complexity. It is an utterly useless tool.

Mathematically, "this is fun" is 10 times more secure.

This is one of the big reasons why I hate when IT people talk about password security. They favor complexity over actual security.

Updated: April 18, 2011

**Q: A 3 word password could be hacked a lot faster with a smaller dictionary**

A: Yes, it could. But how would the hacker know what words to put in the smaller dictionary? How would what words people use? Take "this is fun" - and let instead use a dictionary with only the 500 top english words. It wouldn't work. "this" and "is" are in it, but not "fun".

It wouldn't match.

But then people go on to suggest this...

**Q: But a hacker could hack the two first words using a 500 word dictionary, and the last word using the common word dictionary.**

A: You are kidding, right?

I think you have been watching too many Hollywood movies. In Hollywood, passwords are hacked one digit at the time. Meaning the system would return true or false information based on partial matches.

This is not how the real world works. You cannot match a password based on a partial matches. "This * *" is not the same as "this is fun". It would return it as FALSE. You have to match all three words, all at once.

**Q: So, are you saying we should use "this is fun" as our password?**

A: No, I'm saying that you should use a 3+ word pass phrase as your password. Something that isn't linked directly to you or your immediate interests. So don't choose the names of your three kids.

Choose something like "green is wicked", "summer hammocks are fun" or "floppily floors flop". You get the idea :)