# Choosing Secure Passwords

schneier.com/blog/archives/2014/03/choosing_secure_1.html

As insecure as passwords generally are, they're not going away anytime soon. Every year you have more and more passwords to deal with, and every year they get easier and easier to break. You need a strategy.

The best way to explain how to choose a good password is to explain how they're broken. The general attack model is what's known as an offline password-guessing attack. In this scenario, the attacker gets a file of encrypted passwords from somewhere people want to authenticate to. His goal is to turn that encrypted file into unencrypted passwords he can use to authenticate himself. He does this by guessing passwords, and then seeing if they're correct. He can try guesses as fast as his computer will process them -- and he can parallelize the attack -- and gets immediate confirmation if he guesses correctly. Yes, there are ways to foil this attack, and that's why we can still have four-digit PINs on ATM cards, but it's the correct model for breaking passwords.

There are commercial programs that do password cracking, sold primarily to police departments. There are also hacker tools that do the same thing. And they're *really* good.

The efficiency of password cracking depends on two largely independent things: power and efficiency.

Power is simply computing power. As computers have become faster, they're able to test more passwords per second; one program advertises eight million per second. These crackers might run for days, on many machines simultaneously. For a high-profile police case, they might run for months.

Efficiency is the ability to guess passwords cleverly. It doesn't make sense to run through every eight-letter combination from "aaaaaaaa" to "zzzzzzzz" in order. That's 200 billion possible passwords, most of them very unlikely. Password crackers try the most common passwords first.

A typical password consists of a root plus an appendage. The root isn't necessarily a dictionary word, but it's usually something pronounceable. An appendage is either a suffix (90% of the time) or a prefix (10% of the time). One cracking program I saw started with a dictionary of about 1,000 common passwords, things like "letmein," "temp," "123456," and so on. Then it tested them each with about 100 common suffix appendages: "1," "4u," "69," "abc," "!," and so on. It recovered about a quarter of all passwords with just these 100,000 combinations.

Crackers use different dictionaries: English words, names, foreign words, phonetic patterns and so on for roots; two digits, dates, single symbols and so on for appendages. They run the dictionaries with various capitalizations and common substitutions: "$" for "s", "@" for "a," "1" for "l" and so on. This guessing strategy quickly breaks about two-thirds of all passwords.

Modern password crackers combine different words from their dictionaries:

> *What was remarkable about all three cracking sessions were the types of plains that got revealed. They included passcodes such as "k1araj0hns0n," "Sh1a-labe0uf," "Apr!l221973," "Qbesancon321," "DG091101%," "@Yourmom69," "ilovetofunot," "windermere2313," "tmdmmj17," and "BandGeek2014." Also included in the list: "all of the lights" (yes, spaces are allowed on many sites), "i hate hackers," "allineedislove," "ilovemySister31," "iloveyousomuch," "Philippians4:13," "Philippians4:6-7," and*

> *"qeadzcwrsfxv1331." "gonefishing1125" was another password Steube saw appear on his computer screen. Seconds after it was cracked, he noted, "You won't ever find it using brute force."*

This is why the oft-cited XKCD scheme for generating passwords -- string together individual words like "correcthorsebatterystaple" -- is no longer good advice. The password crackers are on to this trick.

The attacker will feed any personal information he has access to about the password creator into the password crackers. A good password cracker will test names and addresses from the address book, meaningful dates, and any other personal information it has. Postal codes are common appendages. If it can, the guesser will index the target hard drive and create a dictionary that includes every printable string, including deleted files. If you ever saved an e-mail with your password, or kept it in an obscure file somewhere, or if your program ever stored it in memory, this process will grab it. And it will speed the process of recovering your password.

Last year, Ars Technica gave three experts a 16,000-entry encrypted password file, and asked them to break as many as possible. The winner got 90% of them, the loser 62% -- in a few hours. It's the same sort of thing we saw in 2012, 2007, and earlier. If there's any new news, it's that this kind of thing is getting easier faster than people think.

Pretty much anything that can be remembered can be cracked.

There's still one scheme that works. Back in 2008, I described the "Schneier scheme":

> *So if you want your password to be hard to guess, you should choose something that this process will miss. My advice is to take a sentence and turn it into a password. Something like "This little piggy went to market" might become "tlpWENT2m". That nine-character password won't be in anyone's dictionary. Of course, don't use this one, because I've written about it. Choose your own sentence -- something personal.*

Here are some examples:

- WIw7,mstmsritt... = When I was seven, my sister threw my stuffed rabbit in the toilet.
- Wow...doestcst = Wow, does that couch smell terrible.
- Ltime@go-inag~faaa! = Long time ago in a galaxy not far away at all.
- uTVM,TPw55:utvm,tpwstillsecure = Until this very moment, these passwords were still secure.

You get the idea. Combine a personally memorable sentence with some personally memorable tricks to modify that sentence into a password to create a lengthy password. Of course, the site has to accept all of those non-alpha-numeric characters and an arbitrarily long password. Otherwise, it's much harder.

Even better is to use random unmemorable alphanumeric passwords (with symbols, if the site will allow them), and a password manager like Password Safe to create and store them. Password Safe includes a random password generation function. Tell it how many characters you want -- twelve is my default -- and it'll give you passwords like y.)v_|.7)7Bl, B3h4_[%}kgv), and QG6,FN4nFAm_. The program supports cut and paste, so you're not actually typing those characters very much. I'm recommending Password Safe for Windows because I wrote the first version, know the person currently in charge of the code, and trust its security. There are ports of Password Safe to other OSs, but I had nothing to do with those. There are also

other password managers out there, if you want to shop around.

There's more to passwords than simply choosing a good one:

1. Never reuse a password you care about. Even if you choose a secure password, the site it's for could leak it because of its own incompetence. You don't want someone who gets your password for one application or site to be able to use it for another.
2. Don't bother updating your password regularly. Sites that require 90-day -- or whatever -- password upgrades do more harm than good. Unless you think your password might be compromised, don't change it.
3. Beware the "secret question." You don't want a backup system for when you forget your password to be easier to break than your password. Really, it's smart to use a password manager. Or to write your passwords down on a piece of paper and *secure that piece of paper*.
4. One more piece of advice: if a site offers two-factor authentication, seriously consider using it. It's almost certainly a security improvement.

*This essay previously appeared on BoingBoing.*