GOOD PRACTICES FOR SELECTING STRONG PASSWORDS

Using passwords for authentication is inherently a fairly weak mechanism for security. Remembering complicated user ID and password combinations for an average of 30 or 40 different applications and websites is darn near impossible for any normal person. So, we tend to use one combination that we can drill into our head and we don't like to change it if we don't have to. The problem is that passwords can be lifted by key-loggers, shoulder surfers or, if not sufficiently strong enough, they can be brute force cracked from the secret hash that is stored on a system for when a credential is validated for an access request. There are other super stealth methods but the point of this post isn't to cover all of the remote possibilities of attack. The point is that, unless the application we want to use has some form of 2 factor authentication, we are pretty much stuck with the old school simple password option and we need to know how to achieve the strongest credentials in that scenario.

Steve Gibson puts it quite succinctly in his Password Haystacks article:

Every password you use can be thought of as a needle hiding in a haystack. After all searches of common passwords and dictionaries have failed, an attacker must resort to a "brute force" search – ultimately trying every possible combination of letters, numbers and then symbols until the combination you chose, is discovered.

If every possible password is tried, sooner or later yours will be found.

The question is: Will that be too soon . . . or enough later?

System Security

We like to hope the application/systems administrator has done their homework and properly secured all of their systems and infrastructure and is regularly patching and testing for vulnerabilities to ensure everything is locked down tightly and no unauthorized person can make off with the keys to the kingdom. But you never really can know or be certain of that. This potentially

can provide an opportunity for your credential to be attacked and learned. The worst case scenario is that you use this same user ID and password for all of your critical accounts. Once it is compromised in one location, you can bet the attacker is trying it with your user ID combination on every known public Internet site you can imagine to see what other accounts they can compromise and pilfer or use to their advantage. So, we can't control the security of the system where our usernames and passwords are being stored, albeit, hopefully, in some form of encrypted state. We also can't perfectly control against keyloggers or shoulder surfers or man in the middle type attacks. However, the odds of these are quite low in comparison to the risk of someone lifting a password hash table somewhere and running a brute force attack to uncover user ID and password combinations. So, what we can do is ensure that the passwords we use for these systems are sufficiently strong enough that any typical brute force attack to uncover your password will take an inordinate amount of time and processor power to achieve. The question is "How can you know what is a sufficiently strong enough password to counter the typical brute force attack?" That is the answer I hope to provide here.

Password Strength & Complexity

Entropy. You may hearken back to your days spent staring through your eyelids at the ceiling in Chemistry class when you may have vaguely heard your professor drone on about entropy as it related to the laws of thermodynamics. Well that is not the entropy I am referring to. Entropy as it relates to information theory is the measure of the uncertainty of a random variable. It is a measure of disorder or unpredictability. In this case, the random variable being the password or even each character position within the password itself. The goal of a strong password is to provide as much entropy as possible such that a brute force attack to guess the value of the password would take such a large enough number of guesses that it would require an enormous amount of processing power to accomplish the task in any reasonable amount of time.

So, how do we establish as much entropy as possible within a password and how impossible will it be for a normal human to memorize and recall the password for frequent use? Well, it is not very difficult to do at all and it can be quite easy to create a password that is memorable and fairly simple to use frequently. The key is the human brain's ability to use mnemonics and shortcuts. It is like our own indexing mechanism for bits of information. But, I'll get into that later. At this point, let's just focus on the mathematics and probability of guessing a password.

There are 2 factors that weigh into the strength of a password. First and foremost is the length of the password. How many characters make up the password itself. The more characters there are in a password, the stronger it is, by a factor of at least 26 to 84 times per additional character, depending on the range of characters chosen from. Length is actually the single most significant factor in password strength. The second factor is the range of possible characters that could be found in any given character position within the password. Generally, most people stick to

passwords that use just letters and numbers. If you only make use of lower case letters and numbers that leaves you with only 36 possible characters to choose from. In a typical 8 character password with only 36 characters to choose from, if I had access to the encrypted hash value, I could easily crack it in under an hour on my MacBook Pro. If you minimally add upper case letters into the mix, that processing time increases to over 2.5 days. That is a multiple of 65.85 times the duration it took to crack the password without any upper case letters. So, in combination with a longer password, the goal is to include as many character possibilities as possible to increase the entropy or difficulty of predicting the value of the password.

On a normal keyboard, there are typically 84 character options you can easily include in a password with a single keystroke or holding the Shift key. Password crackers know that the majority of people only use a range of 36 characters to create their passwords as they do not use symbols or upper case characters. This fact alone allows them to vastly reduce the time it takes for them to crack a password. If everyone just followed the simple recommendations you always see anytime you go to create a password (use at least 1 upper case, 1 lower case, 1 number and 1 special character like @,+,$,! etc. and you would render most trivial attacks useless requiring an attacker to use a brute force method.

Password length. Each additional password character significantly increases the number of password permutations. If you minimally used a mix of upper case, lower case, and numbers that is a factor of 62 times or almost 26 for each character. An 8 character password would be equivalent to 218,102,555,120,846 possibilities. If you threw in a special character that would be a factor of 84 times for each character. An 8 character password would then be equivalent to 2,479,550,445,438,080 possibilities or a factor of 11.4 times more permutations. If you add one more character and made it a 9 character password it would increase the permutations by a factor of 62 to 84 times, depending on the range of characters you are selecting from.

Excerpt from NIST 800-118

Increasing the character set from 26 characters to 95 characters on a four character length password increases the keyspace almost 200 times. However, if the length of the password is increased from four to 12, given a character set of only 26 characters, the keyspace increases by almost 200 billion times. Although both have significant effect on the overall strength of a password in resisting brute force attacks, outside of cryptographic attacks, length seems to be the dominating factor in determining password strength.

The problem is that even a fairly modest computer with a Graphical Processor (GPU) can attempt brute force password cracking at rates in the neighborhood of 500 Million to a Billion guesses per second or more. Without sufficient length and character options, a small computer can make swiss cheese out of your password in a very short period of time. With the advent of parallel computing,

an attacker can easily gather a small cluster of computers and crack 70-80% of typical passwords stored in hash table. They can even use Botnets, thousands of unsuspecting Internet connected computers under their control, to perform these tasks. So, how can you create a reasonably difficult password that can stand up to the ever increasing capability of computer hardware and the prospect of parallel computing or a Botnet attack? You have to select a minimum password length and character set that creates a large enough pool that will require so much horsepower and take so much time that it would not be worth it for the attacker to continue dumping resources into the task.

Source : https://rolande.wordpress.com/2012/04/21/strongpasswords/