Problems with passwords

Overview

Most current password systems for the Internet are flawed. Designs that were almost acceptable 10 and 15 years ago have not been updated. Instead of moving to integrating authentication services under a cryptographically sound approach the IT industry has continued to proliferate multiple incompatible systems. Users are increasingly exposed by suppliers who feel no pressure to do anything better. There are parallels with the situation where web site page design methods are increasingly being rejected by security software because they represent known security weaknesses that have been exploited by hackers and viruses.

Introduction

The approach to using a log on identifier and password goes back to the early days implementing security on mainframe systems. This kind of security was introduced as soon as it was possible for people outside the computer room to be able to use computer resources. Up until then access was controlled by physical security.

As we rolled terminals out into user areas, so the ID/password concept was rolled out also. Initially these were held in a file that was not protected, but after some splendid security breaches on Unix systems in particular these files were encrypted to make an attacker work harder to get anywhere.

Passwords were short (6 characters). They were short because the ID would be disabled if the password was entered three times incorrectly. They were also short so you didn't have much to type and would likely get it right. They were short because it gave you less to remember.

Initial design considerations

Experience with short passwords soon threw up a series of flaws for user implementation. In no particular order these included:

- using a 'standard' word such as boss, master, doall, passwd;
- using a dictionary word or the name of the business;
- using repeating letters or numerals (AAAAAA, 111111 and so on).

Six characters were also found to be just about short enough for someone to watch and remember whilst the user typed them in.

To counter the users attempts to make their lives easier, systems were invented that changed passwords on a regular basis (say monthly, and even daily for critical passwords), compelled the new password to be different, and checked it against a list of previously used passwords. More sophisticated systems enforced rules requiring passwords to be structured using letters and digits in non-repeating patterns.

These approaches more or less forced users to break other security rules and write down their passwords – particularly if they had several to 'remember'. (I recall a 'classic' case where a user was being expected to remember more than 20 passwords, some of which were the only way to access encrypted documents. Naturally they did not listen to the ideas of regular change and remembering everything.)

The security people continued to ignore the problems faced by human users. ID/password systems were not integrated following the argument that a compromise of one system must not compromise all systems. (This was then ignored in the attempts to find a system that would securely connect a user to all their applications with just one password.) Applications designers have continued to implement their own ideas about user identification - or none at all by making the assumption that magic would somehow occur outside their control.

There continues therefore to be a central dichotomy between those who want short passwords that are forever changing and those who want one password that a user can remember, but it cannot be short and it must be memorable.

Technical design problems

Early password systems restricted user choice to upper case and numerals, thus giving the attacker a much reduced space of attack (the permutations and combinations of valid input data). Later systems used upper and lower case and this improved things a bit in terms of the number of attempts the attacker had to make before he could find it by 'brute force' (still not all eight bits of each byte since not everything is on the keyboard).

Later systems converted the password into a 'hash' or one way encrypted field so that it could not be readily reverse engineered by an attacker. Unfortunately the hashing systems were not necessarily very effective, and even when they were, the amount of space they give you is not that large and the attacker can choose any password that gives them a valid hash, not just the one the user selected. Please note that when passwords are used on their own (that is without a separate Identity field), the attack space is reduced by the number of passwords that have actually been issued, since for the attacker any valid password is good enough.

Even later some subtle systems combined the user id and the password into a hash. This created the potential for more space, although the length of both parts and the way that they were combined was critical to the quality of the result.

Network systems and services, and the introduction of the PC as a networked device as well as a stand-alone computer, together created the idea that it must be possible to have infinite retries at getting the password right. (In the case of the PC, concern was focused upon the problem of having its owner get locked out with no way to recover the situation. Therefore, some systems had physical password reset buttons to get round this problem.) The attacker was being given a massive advantage!

The Internet, built for resilience and information sharing, included the idea of an ID/password, but did not provide encryption to protect the password and allowed infinite retries to get it right. As a result, passwords are usually transmitted unprotected, and may be sent with every page that needs access to a password protected area as well as allowing the attacker all the time the site is up to try and crack it.

Potential routes forwards

The biggest hurdle to overcome is the ability of a user to hit more than six consecutive keys reliably, given that they cannot 'see' the results of what they are doing. (Actually, this is not new. Anyone with a Remington typewriter No 3 and before would know that the type basket on those models hit the paper directly under the roller, not on the front of the roller, and the user had to lift the roller to see what they had typed.)

Of course a user needs a bit of practice in order to get a longer password right. Constant change makes for bad typing. Using a much longer password, say 30 or so character positions, may not be guaranteed to generate what the cryptologists call entropy, but it has a good chance. If it is combined with using hash algorithms that generate much larger spaces (say SHA-1 512) then the attack space will still be large compared with current results.

A long password should also be harder to crack with short dictionary attacks and more resistant to brute force attacks, because the time to create either the password or the hash becomes significant. This may have a lot to recommend itself. Long passwords are also resistant to being captured by others by mere observation (except when keystroke capturing methods are in use) because there is too much now for the attacker to remember, no matter how often then observe. (Perhaps videos will become more popular in 'public places'.

But how do you educate users into using passwords successfully?

The first thing to remember is that the length must be proportionate to the overall security requirement. If a 'three strikes and you're out' system combined with a token of almost any kind is in use you can live with a 4-digit PIN. If there are multiple systems then a single long password could be used as a system enabler for all services.

Choosing long passwords is not the daunting prospect that so destroys choosing short passwords. Natural language is now to be preferred since it must be memorable. But the expression of the natural language must be left to the capricious nature of the user.

By way of some examples of longer passwords, one could consider the following:

"Table!house*", "Knight(soil)" or "Dem0n**manager". Other examples that could work include, "1066andallthat", "Hangthe****donkey" or "Now is the time forall men". This last one is a quotation, but it's still hard to guess or attack, especially if you don't know where the spaces are! These kinds of passwords are proof against any dictionary attack, and, provided they are not changed often, users are more likely to choose something difficult and unique. Another handy feature is that they are slightly harder to share with friends since there is so much more to remember.

Never forget the real purpose

The password, as we use it today, is more often than not the 'secret' that unlocks systems capabilities or grants authorizations (including access control). In future services it will be used to authorize cryptographic secrets, most likely held in software, and then later in hardware. These 'keystores' may hold various secrets, perhaps even including other passwords that are transparent to the user. Where infinite retries are possible, the use of short passwords will represent a significant, and avoidable weakness which designers may one day be called to account for.

Ultimately, the real purpose of a security system is to try and make the user's life easy whilst making the attacker's life difficult. Systems that ignore the user are going to fail with the very community they are supposed to serve.

Whenever users cannot manage the systems they are given an advantage is being given to the attacker because they will exploit those aspects of the system first. Similarly, a poorly designed system will fail and will compromise the very users it is supposed to protect. Poor design is much harder to fix than bad coding or errors in implementation.