Passwords

Passwords are an example of an authentication mechanism based on what a person knows: The person supplies a password, and the system validates that password.

Def: A password is information associated with an entity that confirms the entity’s identity.

Talk about UNIX password system and how it works.

The UNIX password system uses one of 4,096 one-way hash functions to compute the encrypted plaintext password in the /etc/passwd. Later UNIX systems store the encrypted password in the /etc/shadow file.

1 of 4096 hash functions

\[
f: A \rightarrow C
\]

\[
f: \text{8 characters} \rightarrow \text{11 characters + 2 characters for hash function}
\]

stored

e.g.

A UNIX password is composed of up to 8 ASCII characters minus the ASCII NUL character. The set A is composed of strings of up to 8 characters, each chosen from 127 possible characters. This is \(67,675,234,241,018,881\) (\(6.7 \times 10^{16}\)) possible passwords. The set C contains strings of exactly 13 characters chosen from an alphabet of 64 characters or \(302,231,454,903,657,293,676,544\) (\(3 \times 10^{23}\)) possible strings. A sample shadow password file where these encrypted passwords are stored might be

```
noaccess:NP:6445::::::
pcguest:NP:::::::
vcs:SHUdAMEOLreF2:::::::
deadmail:D8yS4WAYr9kVc:::::::
grossman:SF8ZphUEkPyWQ:10163::::::
techval:/LMOlO4sa0aiE:13553::::::
tecadmin:4yopWKO9JWksc:10886::::::
```

How can one protect the password?

1. Hide enough information so that A, C, or f cannot be found
   This is the current approach using /etc/shadow

2. Prevent access to the authentication function L
   Restrict where a user (root) can login in from
   Restrict when a user (root) can login in

Attacking a Password System

The simplest attack against a password-based system is to guess passwords.

Def: A dictionary attack is the guessing of a password by repeated trial and error.

Type 1: If one knows the complementary information and complementation functions, a dictionary attacks takes each guess and computes \(f(g)\) for each \(f \in F\). If \(f(g)\) corresponds to the complementary information for the user, then \(g\) authenticates that user. Note: In a UNIX system you would need the shadow file to crack passwords.
Type 2: If one does not know either the complementary information or the complementation functions, then the authentication function $l \in L$ can be used. In this case, if the guess results in $l$ returning true, then the guess is the correct password.

Examples

University
minimum of 8 characters
at least one letter
at least one number
at least one special character
no spaces
no more than two repeated characters
must change every 365 days

Password Strength Checker
try the Microsoft password strength meter at
http://www.microsoft.com/protect/yourself/password/checker.mspx

Counter Password Guessing

What can be done to counter password guessing? Password guessing requires either the set of complementation functions and complementary information or access to the authentication functions. In either case, the goal of the defenders is to maximize the time needed to guess the password.

1. Random Selection of Passwords

Theorem 1: Let the expected time required to guess a password be $T$. Then $T$ is a maximum when the selection of any of a set of possible passwords is equiprobable.

The above theorem guides us in the selection of passwords. For example, randomly selected passwords are stronger than user selected passwords in general.

PDP-11 System Story

The PDP-11 password system has randomly generated passwords composed of 8 capital letters and digits. Therefore, there should have been $36^8$ possible passwords. However, the PDP-11 was a 16 bit machine. Hence, there were only $2^{16} - 1$ possible random passwords. Thus, what appears to be $2,821,109,907,456$ passwords actually turned out to be 65,535 passwords.

Written Passwords

When passwords get too difficult to remember, which is often the case with randomly generated passwords, the user will write those passwords down and attach them to their computer monitor. This obviously is not an acceptable security practice, but often occurs. One approach to this is to write a form of the password down and then apply a transformation algorithm to the written password. For example, if the randomly generated password is “sw4Ggle3”, one might write down “sw4ggle3” and use the transformation that the 3rd letter is always capitalized. This is far from idea, but it does obscure the password.

2. Pronounceable Random Computer Generated Passwords

A compromise between using random, unmemorizable passwords and writing passwords down is to use random pronounceable passwords. Basically, pronounceable words can be constructed with $c v c$ combinations, where $c$ represents a consonant and $v$ represents a vowel. For example, we might have something like
Throw in a couple of digits, and you have a pretty good randomly generated password. Using this approach, we have $20^6 \times 6^2 \times 10^2$ or 576,000,000 possible passwords.

3. User Selection of Passwords

Rather than randomly generating passwords for users, one can allow users to select their password, but constrain that selection process. This leads to what is called proactive password selection.

**Def:** Proactive password selection is a process that enables a user to propose a password that they can remember, but rejects any passwords that are deemed to easy to guess.

Good passwords can be constructed in several ways. A password containing at least one digit, one lower-case letter, one upper-case letter, one punctuation symbol, and one control character is usually quite strong.

**Def:** A proactive password checker is software that enforces specific restrictions on the selection of new passwords.

There are a number of criteria that a proactive password checker should possess. Those can be found in the text.

What does university do
Anyone aware of other proactive processes?

4. Guessing Through Authentication Functions

In this approach, one guesses passwords to see if one is successful. The actual complements or the complementation functions are not known. Luck and extreme patience are required for this attack approach. Several techniques are available to counter such attacks.

*Backoff* techniques comprise one such approach. The most common approach is the *exponential backoff*, where the authentication process is slower to respond as successive unsuccessful guesses are submitted to the authentication process. In general, if n unsuccessful attempts have been made, the authentication process waits $x^n$ seconds before prompting for the next password guess.

A second technique is to *disconnect* the user after n failed authentication attempts. This approach works best over a dial-up connection but is less effective for a network connection.

Another approach is called *disabling or lockout*. In this approach the account is locked after n tries. Only a specially authorized human can unlock the account. This is very effective but has the potential of offending important users or locking out special user ids such as root.

The final approach that we will discuss is called *jailing*. In this approach the user is admitted to the system after n unsuccessful attempts but is placed into a special containment where their actions are recorded. This is potentially useful in determining what the attacker is interested in. In intrusion detection system a *honeypot* is often used to entice the user into thinking that they have gotten important data or access to the desired machine. In general, this is not effective.

**Password Aging**
Another approach to inhibit the guessing of passwords is to ensure that the passwords are changed frequently.

**Def:** *Password aging* is the requirement that a password be changed after some period of time has passed or after some event has occurred.

- minimum time between change
- maximum time must change
- remember n past passwords

Talk about problems