The Fox Hunt

Handling Passwords - II

By Whil Hentzen

Last month, we discussed the issues involved in encrypting passwords so that users peering into the USERS table can't read or hack the passwords for different users.

Once we've developed a mechanism for encrypting the password, however, we've still several other issues to consider.

The biggest problem with passwords is that if they're easy to guess, the cleverest encryption scheme won't help. Here are some issues. First, consider requiring a minimum length for the password. Systems in common use often use four or six characters as a minimum.

Next, consider making the password case sensitive. Of course, this provides all sorts of opportunities for the office jokester to cause havoc by discretely hitting the CAPS LOCK key on an unsuspecting user's keyboard, but it also greatly increases the number of combinations of passwords available. If you don't make passwords case sensitive, be sure to convert all password input to either upper or lower case, or your encryption routine will cause trouble.

The third consideration is filtering the actual strings entered by the user. Depending on the level of security required and the nosiness of the user base, you may want to prohibit entries with repeated characters (such as the infamous FJFJFJ password) or all numeric entries (to eliminate birthdates and social security numbers.)

Finally, you may want to keep a dictionary of "illegal" passwords, like user names and company terminology. You may even want to go as far as populating the "illegal words" dictionary with a complete dictionary, so that users must create gibberish passwords such as JF8RW2. This dictionary can also be used to store common four letter words that some users delight in using as passwords, and old passwords, so that users can't hop back and forth between two passwords.

Another issue you'll want to address is whether or not you want to require the changing of passwords on a regular basis. This can be accomplished by keeping a "Password last changed" date in the USERS table. This date would be checked at each logon, and if the date has expired, require the user to enter a new password on the spot.

The last issue we'll cover is handling initial passwords for new users. Some systems use the user's login name as default, others use a common string, such as "12345" to begin with. This string can be kept in your system table, so that you can change it for each new application that you create. The following code snippet from a "Add New User" routine presumes a cPWDef field in the SYS_SYST table. If empty, passwords for new users are initialized to be the same as their login name. If not empty, the entry in the field is used as the default for each new user.

```
use SYS_SYST in 0
if empty(SYS_SYST.cPWDef)
 m.cUserPW = f_encrypt(alltrim(m.cNameUser) ,1)
else
 m.cUserPW = f_encrypt(alltrim(SYS_SYST.cPWDef),1)
endif
use in SYS_SYST
```

These are all issues you need to consider in order to make your password system more than an interesting curiosity that every user knows how to get around.

As the saying goes, security systems are intended to keep honest people honest. The determined hacker will find their way around any type of security system you can build - the best you can hope for is to make it difficult enough that most will get discouraged and go off to ply their trade somewhere else.