Passwords stored using reversible encryption: how it works (part 2)

Archived: 2026-04-05 23:42:24 UTC

In part one of this article, I described how the reversible encryption of Windows domain passwords works. In this part, we will look at the security of this mechanism.

To decrypt the password you need the following components:
- The encrypted password (G$RADIUSCHAP)
- The 16 byte random (G$RADIUSCHAPKEY)
- The global LSA secret (G$MSRADIUSCHAPKEY)
- A static key hardcoded in RASSFM.DLL

The hardest thing to get is that global LSA secret. This is stored in active directory and synchronized between domain controllers. To access this key, you need domain administrator privileges. An obvious risk here is that once someone gains domain administrator privileges, he won't need to crack any passwords, but can simply decrypt them. Of course, if an attacker gains domain administrator privileges on your domain, you are already in big trouble anyway.
However, the other components are all semi-public information. The static key is hardcoded in RASSFM.DLL which comes with every Windows server, so is easy to get. The G$RADIUSCHAP and G$RADIUSCHAPKEY are stored in active directory in the userParameters structure. If you have a user account on a domain you can use AD Explorer to access the Active Directory database and read this information. Of course, to decrypt the password you will still need that LSA secret.

The encrypted version of the password can be interesting though; by looking at the encrypted password you can derive the length of the plaintext password. Two examples I used in my presentation:

Pwd1 encrypted: 0f53 8420 9418 05ce 01ad

Pwd12 encrypted: 5d69 9375 6f92 1b63 7728 439f

So, by looking at the encrypted passwords you should notice that the encrypted version of Pwd12 is two bytes longer than the encrypted version of Pwd1. So, although we cannot determine from just looking at the encrypted password that they are very similar, we can determine their length. What this means is, that as a domain user, you can determine the length of other people's passwords, which could be quite interesting.

If you obtain the LSA secret somehow (maybe because you temporarily gain domain administrator privileges), from that point on you can decrypt passwords stored using reversible encryption. This could be used as a nice backdoor, just steal the LSA secret, enable reversible encryption (if it hasn't been enabled yet) and you can grab the domain administrator password with just a normal user account.

Someone in the HAR 2009 audience had a very nice question: Is it possible to recreate the LSA secret if you're afraid it has been stolen. Of course the better option is to recreate the entire domain, but this is not always an option. To recreate the LSA secret you need to write a program that sets the LSA secret to NULL. According to the documentation, Windows will delete the LSA secret. It will generate a new LSA secret when it needs to encrypt another password. Of course, Windows won't be able to decrypt the passwords stored before that point anymore. However, it will still try to decrypt them using the new LSA secret, which will result in gibberish most of the time. If you're really unlucky it could decrypt the first two bytes to NULL, which basically means the password is suddenly empty. So if you ever have to do this, resetting all passwords immediately is probably a good idea.
