Vulnerability of DPAPI data protection in Win2K, Win2K3, Windows Server 2008, and Windows Server 2012

Passcape Software

1 Brief description of the vulnerability

1.1 The problem

One of our previous articles described the operating principles of the DPAPI system, its exceptional reliability, functional value, and cracking resistance. Not long ago, we occasionally found out that some DPAPI blobs in Windows 2003 had decryption issues. After determining the cause of the problem, a rather interesting breach in DPAPI security was revealed which can be reproduced in all server operating systems, beginning with Win2K and ending with Windows Server 2012.

Briefly, the essence of it is the following: By default, the Master Keys of all domain users with interactive logon privileges, except for built-in accounts, are created in the Windows 2000 compatibility mode; therefore, the decryption of the data encrypted with DPAPI doesn't require the owner's logon password.

1.2 Affected software

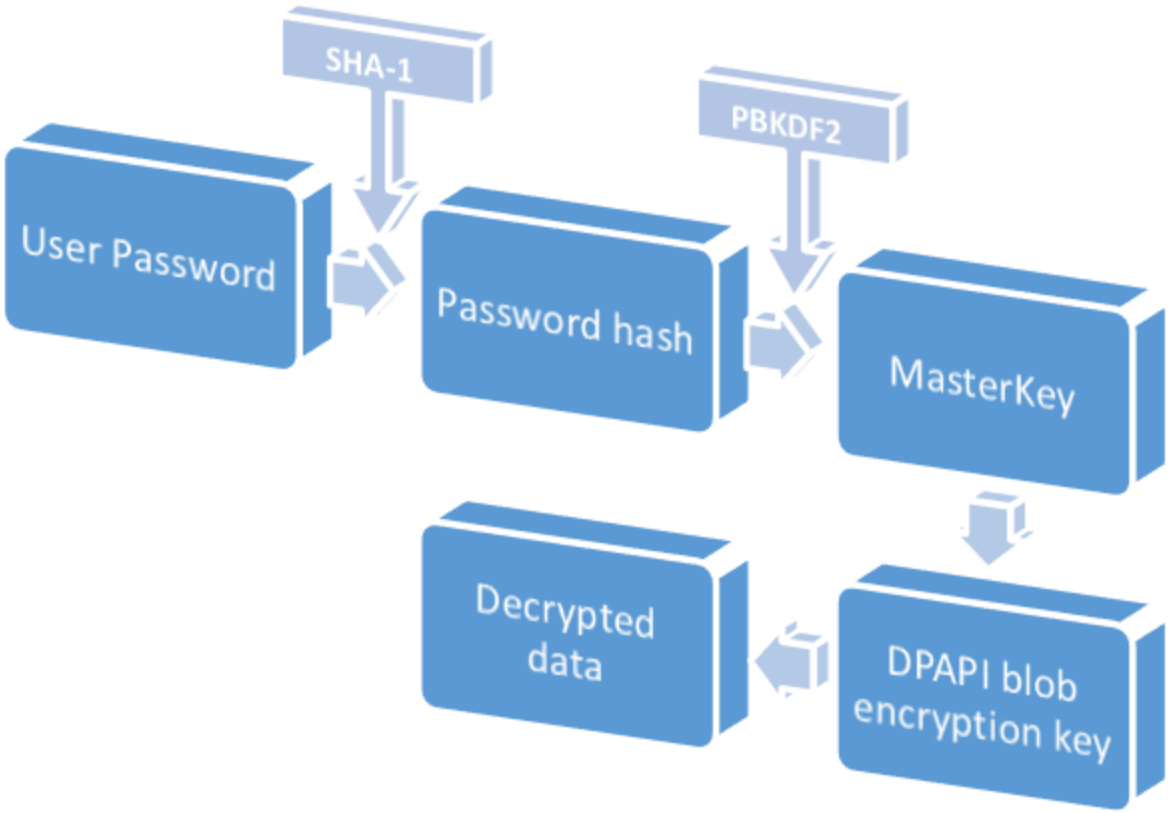
This means that any Administrator or any other user who has physical access to the server can decrypt the following personal data of vulnerable user accounts:

- Passwords and form auto-completion data in the popular browsers: Internet Explorer, Google Chrome, Opera Browser, etc.
- E-mail account passwords in Outlook, Windows Mail, Windows Live Mail, etc.
- Account passwords of the Windows FTP manager
- Access passwords to shared folders and resources
- Keys and passwords to wireless networks
- Encryption keys in Windows CardSpace
- Encryption keys in Windows Vault
- Remote desktop connection passwords
- .NET Passport passwords
- Windows Live ID personal data
- Private keys in encrypting file system (EFS)
- Encryption keys in S-MIME mail
- Users' certificates
- Private data in the Internet Information Services
- EAP/TLS and 802.1x authentication
- Network passwords in the Credential Manager
- Personal data in any application, programmatically protected with the Windows API function CryptProtectData, such as Skype, Windows Rights Management Services, Windows Media, MSN messenger, Google Talk, etc.

2 Technical details

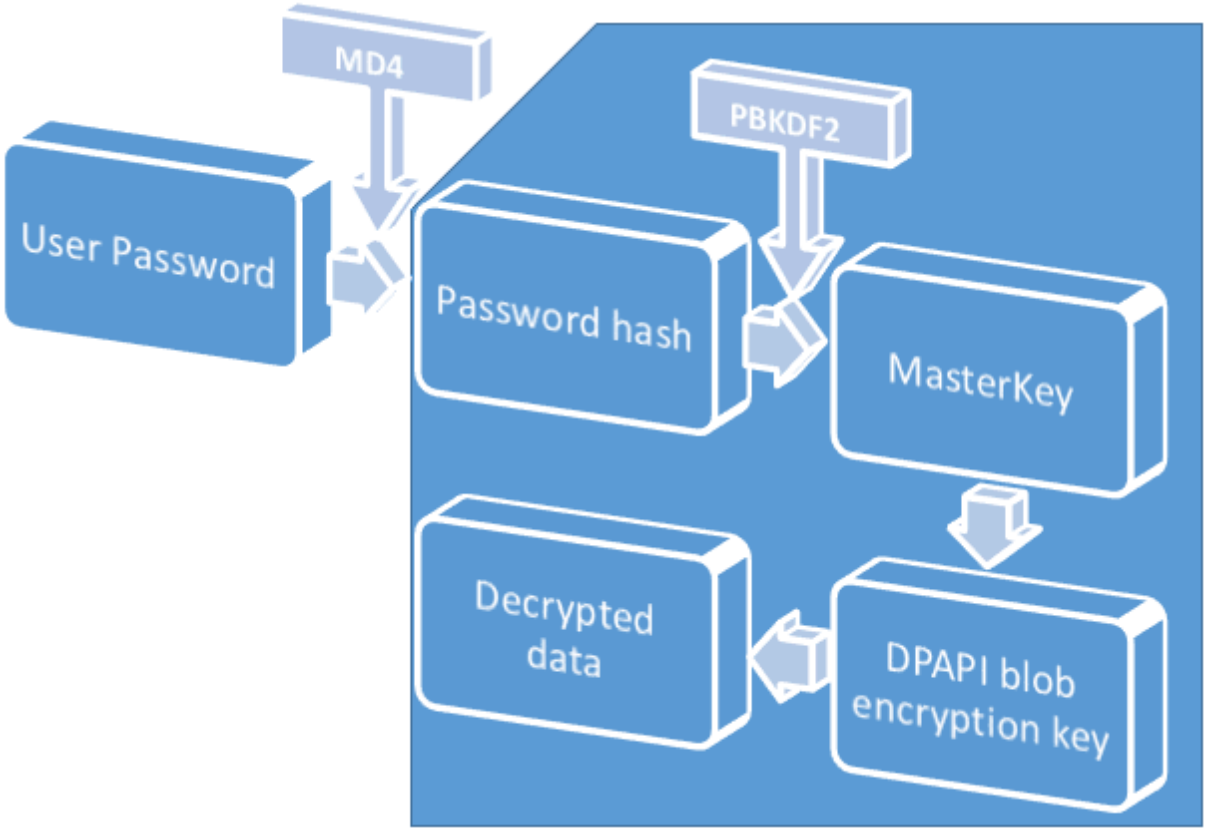
2.1 DPAPI encryption in Windows XP and higher OSes

This is what the process of decrypting private data encrypted with DPAPI looks like (some details are omitted): Initially, the data owner's logon password is passed through SHA-1 to get the password hash; the password hash and the owner's SID are then fed to PBKDF2 function. At the output, that produces a prekey, which participates in the decryption of the Master Key. The decrypted Master Key in its turn is used for the decryption of the actual DPAPI blobs. Here is what it looks like in a chart:



2.2 DPAPI encryption in OS Windows 2000

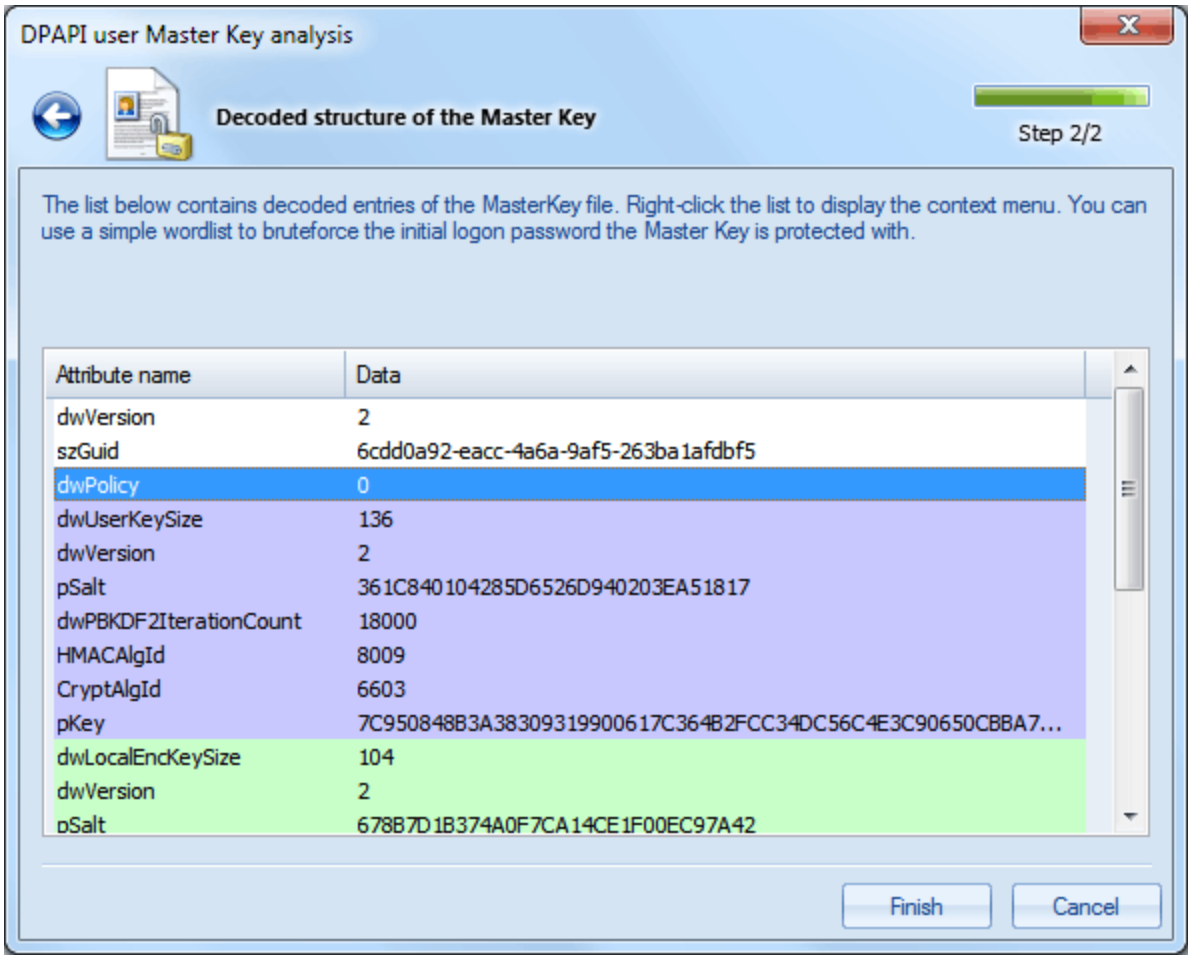
The first implementation of DPAPI used in Windows 2000 had other Master Key encryption algorithms. But that wasn't what has made it extremely vulnerable; it was the fact that in order to obtain the user's password hash, it used MD4 hashing function instead of SHA-1. Here is what it looks like:



Amusingly, the same hashing function is also used in validating users' logon passwords, the hashes of which are stored either in the SAM registry or in the Active Directory (for server operating systems). Thus, the user's plaintext password is not really necessary do decrypt the Master Key. It is sufficient to just take an existing MD4 hash of the respective user from SAM or NTDS.DIT and use it as data-in. The rest is already known.

2.3 DPAPI encryption flaw in Windows 2003, 2008, 2012 server OSes

Which algorithm is used – SHA-1 or MD4 – is specified in the header of the Master Key. Bit 4 of the dwPolicy flag indicates that the Master Key uses the SHA-1 algorithm. Newly created users with interactive logon privileges in Windows server OSes do not have this flag set by default. Respectively, the decryption of their private data doesn't require the logon password.



3 Utilizing the vulnerability

3.1 Creating interactive domain user in Windows 2012 server

Let's move on from theory to practice and try adding a new user to a Windows Server 2012 domain, then create some DPAPI secret and then decrypting it offline without the owner's logon password.

Open the "Active Directory Users and Computers" console and create a new domain user named Test. Grant that user an interactive logon privilege. To do so, you can simply add that user to the local administrators group.

3.2 Creating the Master Key and DPAPI secret for the new user

Now we need to log off the system and then logon under this account. The new account doesn't have the Master Key yet. It will be created during the first call of the CryptProtectData function. We'll speed that process up by forcing the call for the respective function. For that purpose, we have a homonymous utility CryptProtectData.exe, which simply calls the API function CryptProtectData with command-line parameters (the utility source code is available on the website). Launch it with the following parameters: **CryptProtectData mysupersecret out.dat**. At the output, we'll have the out.dat file with a DPAPI blob containing our encrypted text (mysupersecret).

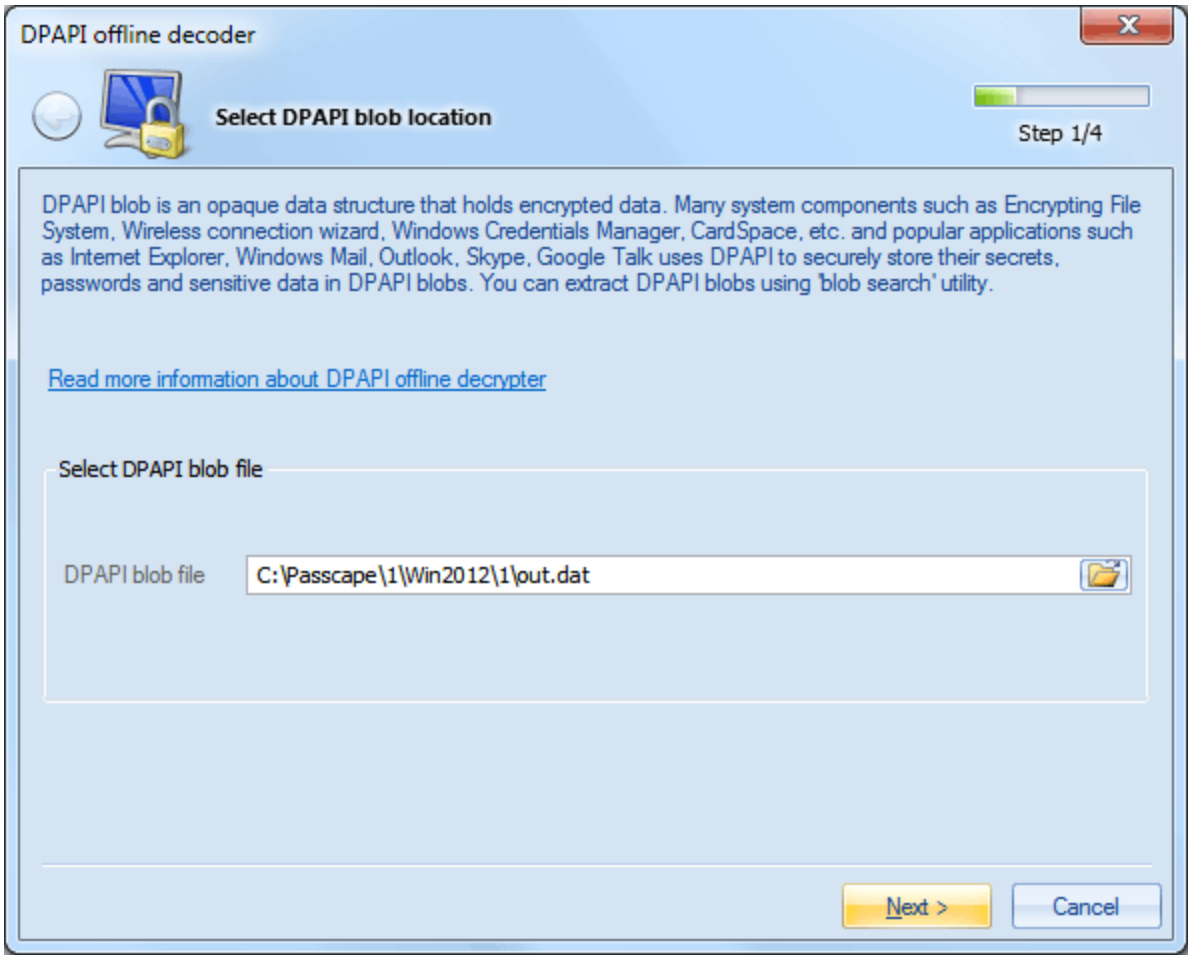
So, the Master Key has been created and stored in the folder C:\Users\test\AppData\Roaming\Microsoft\Protect\<SID>\<mk>
Where <SID> – the owner's sid. We'll still need it, so either memorize it or copy the entire catalog.
<mk> – name of DPAPI Master Key; e.g., 6cdd0a92-eacc-4a6a-9af5-263ba1afdbf5

Other than that, for the offline decryption of the out.dat file we'll need the MD4 hash of the data owner, which is stored in the Active Directory. To get it, we'll take advantage of our utility and make a copy of the NTDS.DIT file, as well as a copy the SYSTEM registry required for obtaining the hash.

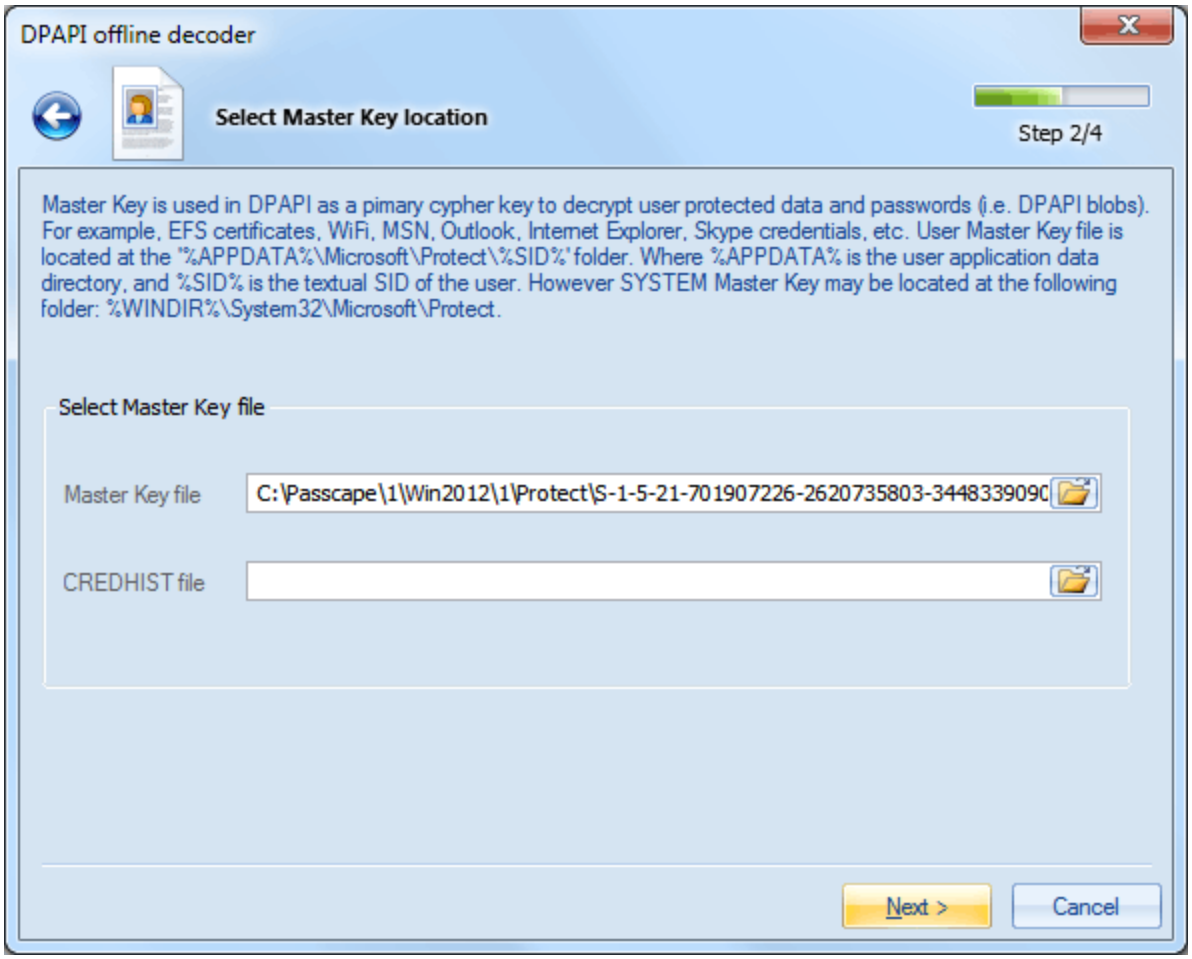
3.3 Decrypting the user DPAPI secret without knowing the owner logon password

As the result, we'll have:

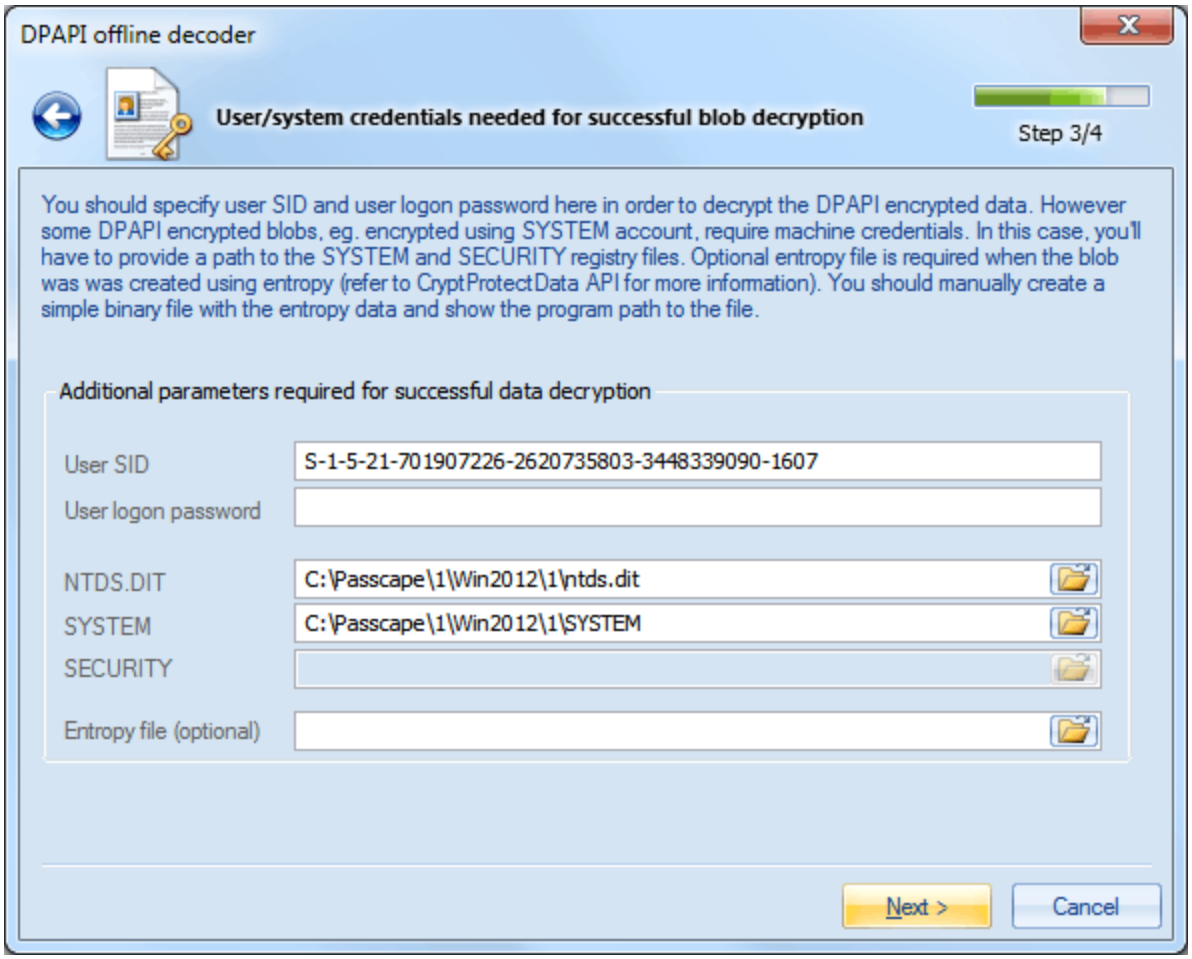
- The out.dat file with the encrypted secret, which we need to decrypt
- The data owner's textual SID
- His Master Key
- The owner's hash (files NTDS.DIT and SYSTEM).



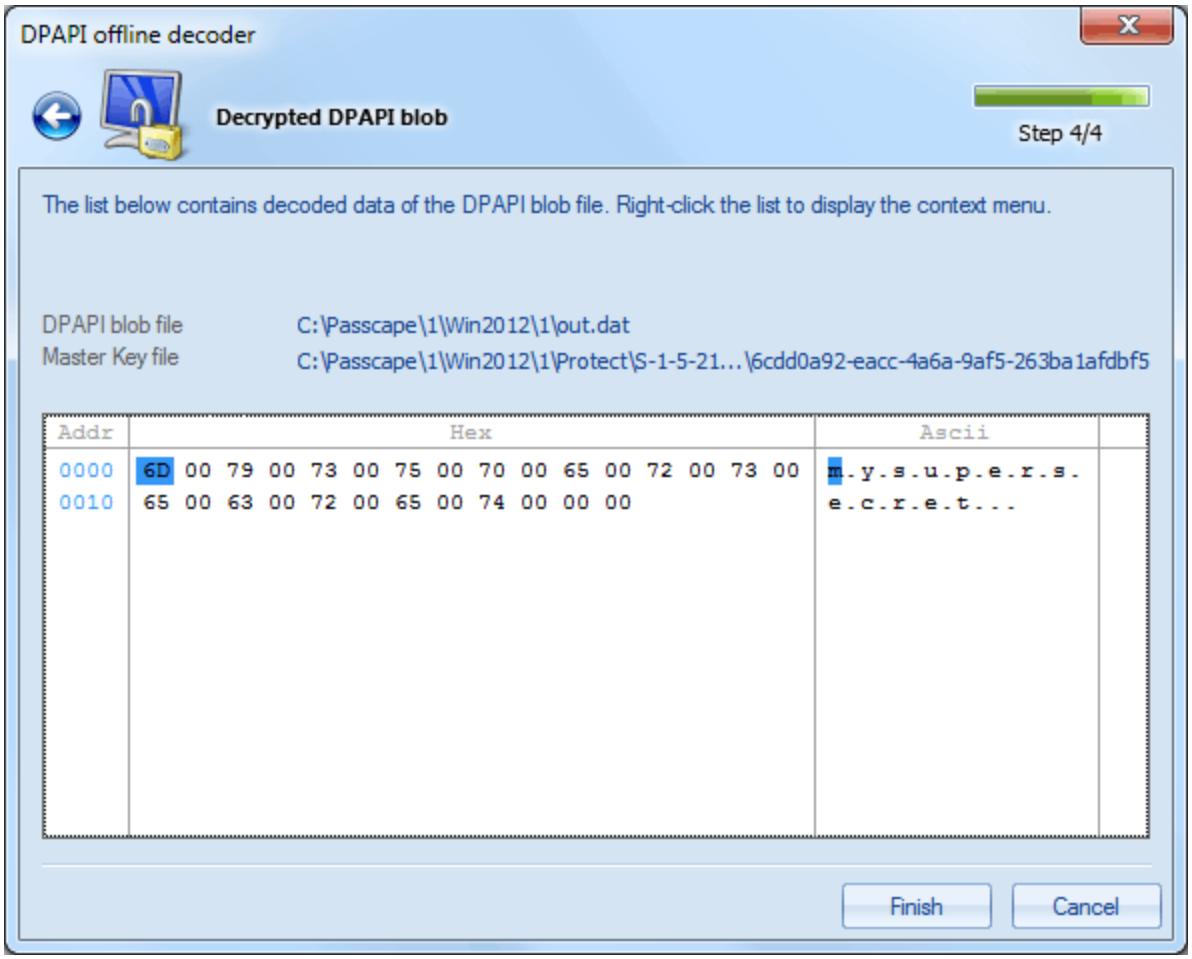
Now, launch the DPAPI offline decryption utility and tell it the path to the out.dat file.



On the second step of the Wizad, specify the path to the data owner's Master Key: select the path and then click Next. The program pops a warning that a vulnerability is found in the Master Key, and therefore the decryption can be carried out two ways: with the user's password and without it. We, certainly, want the latter.



So, on the next step of the Wizard, enter the owner's SID and the path to the NTDS.DIT and SYSTEM files, leaving the password field blank.



Click Next, and here we have the decrypted secret. As you can see, the owner's password wasn't necessary.

4 Conclusion

What remains quite unclear is whether this flaw in the interactive users is related to an error or it shouldn't be treated as such. Perhaps this is a peculiarity of the DPAPI implementation in the server operating systems; for instance, to provide backward compatibility. On the other hand, this seems to be highly unlikely, as the safety of user's confidential data is not properly ensured, while DPAPI was specifically meant to be the data protection system based on the owner's password. Funny, but it turns out that desktop PCs are more resistant to offline password recovery than those running the server operating systems. One way or the other, system administrators should be aware of this vulnerability, as forewarned is forearmed.