Title:
Recovering Internet Explorer Passwords: Theory and Practice

Word Count:
5740

Summary:
The article classifies types of private data stored in Internet Explorer, analyses Internet Explorer password recovery programs, and give us some examples of recovering lost Internet passwords.

Keywords:
internet, explorer, ie, internet explorer, password, recovery, change, find, forgot, remove, lost, software, reset, cracker, browser, asterisk, decrypt, site, web, stealer, cracker, viewer, reader, revealer, cached, best, free, reveal

Article Body:
Recovering Internet Explorer Passwords: Theory and Practice

1. Introduction

Nobody will likely dispute the fact that Internet Explorer is today's most popular Web browser. According to the statistics, approximately 70% of online users prefer to use just this program. Arguments about its pros and cons may last forever; still, this browser is the leader of its industry, and this is a fact that requires no proof. Internet Explorer carries several built-in technologies, designed to make average user's life easier. One of them - IntelliSense - is made for taking care of the routine tasks, like the automatic completion of visited webpage addresses, automatic filling of form fields, users' passwords, etc.

Many of today's websites require registration, which means, user would have to enter user name and password. If you use more than a dozen of such websites, you will likely need a password manager. All modern browsers have a built-in password manager in their arsenal, and Internet Explorer is not an odd. Indeed, why would one have to remember yet another password if it is going to be forgotten some time soon anyway? Much easier would be to have browser do the routine work of remembering and storing passwords for you. It's convenient and comfortable.

This would be a totally perfect solution; however, if your Windows operating system crashed or reinstalled not the way it's supposed to be reinstalled, you can easily lose the entire list of your precious passwords. That's the toll for the comfort and convenience. It's good just about every website has a saving 'I forgot password' button. However, this button will not always take your headache from you.

Each software developer solves the forgotten password recovery problem their own way. Some of them officially recommend copying a couple of important files to another folder, while other send all registered users a special utility that allows managing the migration of private data, and the third ones pretend they are not seeing the problem. Nevertheless, the demand creates the offer, and password recovery programs are currently on a great demand.

In this article, let's try to classify types of private data stored in Internet Explorer, look at programs for the recovery of the data, and study real-life examples of recovering lost Internet passwords.

2. Types of passwords stored in Internet Explorer
- Internet Explorer may store the following types of passwords:
- Internet Credentials
- AutoComplete Data

- AutoComplete Passwords
- FTP Passwords
- Synchronization Passwords for cached websites
- Identities Passwords
- AutoForms Data
- Content Advisor Password

Let's take a closer look at each listed item.

2.1. Internet Credentials for websites
Internet credentials mean user's logins and passwords required for accessing certain websites, which are processed by the wininet.dll library. For example, when you try to enter the protected area of a website, you may see the following user name and password prompt (fig.1 http://www.passcape.com/images/ie01.png).

If the option 'Remember my password' is selected in that prompt, the user credentials will be saved to your local computer. The older versions of Windows 9a stored that data in user's PWL file; Windows 2000 and newer store it in the Protected Storage.

2.2. AutoComplete Data
AutoComplete data (passwords will be covered further) are also stored in the Protected Storage and appear as lists of HTML form field names and the corresponding user data. For example, if an HTML page contains an e-mail address entry dialog: once user has entered his e-mail address, the Protected Storage will have the HTML field name, the address value, and the time the record was last accessed.

The HTML page title and website address are not stored. Is that good or bad? It's difficult to determine; more likely to be good than bad. Here are the obvious pros: it saves free space and speeds up browser's performance. If you think the last note is insignificant, try to imagine how you would have to perform several extra checkups in a multi-thousand (this is not as rare as it may seem to be) auto-fill list.

Another obvious plus is that data for identical by name (and often by subject) HTML form fields will be stored in the same place, and the common data will be used for the automatic filling of such pages. We will see this by this example. If one HTML page contains an auto-fill field with the name 'email', and user entered his e-mail address in that field, IE will put in the storage, roughly, 'email=my@email.com'. From now on, if the user opens another website, which has a page with the same field name 'email', the user will be suggested to auto-fill it with the value that he entered on the first page (my@email.com). Thus, the browser somewhat discovers AI capabilities within itself.

The major drawback of this data storage method comes out of its advantage that we just described. Imagine, user has entered auto-fill data on a webpage. If someone knows the HTML form field name, that person can create his own simplest HTML page with the same field name and open it from a local disk. To uncover the data entered in this field, such person will not even have to connect to the Internet and open the original WWW address.

2.3. AutoComplete Passwords

In the case with passwords data, however, as you might have guessed, the data will not be filled in automatically. Since auto-complete passwords are stored along with the Web page name, and each password is bound to only one specific HTML page.

In the new version, Internet Explorer 7, both AutoComplete passwords and data are encrypted completely different; the new encryption method is free from the shortcoming just described (if that can be classified as a shortcoming.)

It is worth noticing that Internet Explorer allows users to manage auto-fill parameters manually, through the options menu (fig.2 http://www.passcape.com/images/ie02.png).

2.4. FTP passwords

FTP site passwords are stored pretty much the same way. It would be relevant to notice that beginning with Windows XP FTP passwords are additionally encrypted with DPAPI. This encryption method uses logon password. Naturally, this makes it much more difficult to recover such lost passwords manually, since now one would need to have the user's Master Key, SID and the account password.

Starting with Microsoft Windows 2000, the operating system began to provide a Data Protection Application-Programming Interface (DPAPI) API. This is simply a pair of function calls that provide OS-level data protection services to user and system processes. By OS-level, we mean a service that is provided by the operating system itself and does not require any additional libraries. By data protection, we mean a service that provides confidentiality of data through encryption. Since the data protection is part of the OS, every application can

now secure data without needing any specific cryptographic code other than the necessary function calls to DPAPI. These calls are two simple functions with various options to modify DPAPI behavior. Overall, DPAPI is a very easy-to-use service that will benefit developers that must provide protection for sensitive application data, such as passwords and private keys.
DPAPI is a password-based data protection service: it requires a password to provide protection. The drawback, of course, is that all protection provided by DPAPI rests on the password provided. This is offset by DPAPI using proven cryptographic routines, specifically the strong Triple-DES and AES algorithms, and strong keys, which we'll cover in more detail later. Since DPAPI is focused on providing protection for users and requires a password to provide this protection, it logically uses the user's logon password for protection.
DPAPI is not responsible for storing the confidential information it protects. It is only responsible for encrypting and decrypting data for programs that call it, such as Windows Credential manager, the Private Key storage mechanism, or any third-party programs.
Please refer to Microsoft Web site for more information.

2.5. Synchronization Passwords for cached websites

Synchronization passwords free user from having to enter passwords for cached websites (sites set to be available offline.) Passwords of this type are also stored in IE's Protected Storage.

2.6. Identities passwords

So are identities passwords. The identity-based access management mechanism is not widespread in Microsoft's products, except, perhaps, Outlook Express.

2.7. AutoForms Data

A special paragraph must cover the form auto-fill method, which constitutes a hybrid way of storing data. This method stores the actual data in the Protected Storage, and the URL, which the data belong to, is stored in user's registry. The URL written in the registry is stored not as plaintext - it is stored as hash. Here is the algorithm for reading form auto-fill data in IE 4 - 6:

```
===8<===========Begin of original text===========
//Get autoform password by given URL
BOOL CAutoformDecrypter::LoadPasswords(LPCTSTR cszUrl, CStringArray
*saPasswords)
```

```
{
assert(cszUrl && saPasswords);

saPasswords->RemoveAll();

//Check if autoform passwords are present in registry
if ( EntryPresent(cszUrl) )
{
//Read PStore autoform passwords
return PStoreReadAutoformPasswords(cszUrl,saPasswords);
}

return FALSE;
}


//Check if autoform passwords are present
BOOL CAutoformDecrypter::EntryPresent(LPCTSTR cszUrl)
{
assert(cszUrl);

DWORD dwRet, dwValue, dwSize=sizeof(dwValue);
LPCTSTR cszHash=GetHash(cszUrl);

//problems computing the hash
if ( !cszHash )
return FALSE;

//Check the registry
dwRet=SHGetValue(HKCU,_T("Software\\Microsoft\\Internet
Explorer\\IntelliForms\\SPW"),cszHash,NULL,&dwValue,&dwSize);
delete((LPTSTR)cszHash);

if ( dwRet==ERROR_SUCCESS )
return TRUE;

m_dwLastError=E_NOTFOUND;
return FALSE;
}


//retrieve hash by given URL text and translate it into hex format
LPCTSTR CAutoformDecrypter::GetHash(LPCTSTR cszUrl)
```

```
{
assert(cszUrl);

BYTE buf[0x10];
LPTSTR pRet=NULL;
int i;

if ( HashData(cszUrl,buf,sizeof(buf)) )
{
//Allocate some space
pRet=new TCHAR [sizeof(buf) * sizeof(TCHAR) + sizeof(TCHAR)];
if ( pRet)
{
for ( i=0; i<sizeof(buf); i++ )
{
// Translate it into human readable format
pRet[i]=(TCHAR) ((buf[i] & 0x3F) + 0x20);
}
pRet[i]=_T('\0');
}
else
m_dwLastError=E_OUTOFMEMORY;
}

return pRet;
}


//DoHash wrapper
BOOL CAutoformDecrypter::HashData(LPCTSTR cszData, LPBYTE pBuf,
DWORD dwBufSize)
{
assert(cszData && pBuf);

if ( !cszData || !pBuf )
{
m_dwLastError=E_ARG;
return FALSE;
}

DoHash((LPBYTE)cszData,strlen(cszData),pBuf,dwBufSize);
return TRUE;
}
```

```
void CAutoformDecrypter::DoHash(LPBYTE pData, DWORD dwDataSize,
LPBYTE pHash, DWORD dwHashSize)
{
DWORD dw=dwHashSize, dw2;

//pre-init loop
while ( dw-->0 )
pHash[dw]=(BYTE)dw;

//actual hashing stuff
while ( dwDataSize-->0 )
{
for ( dw=dwHashSize; dw-->0; )
{
//m_pPermTable = permutation table
pHash[dw]=m_pPermTable[pHash[dw]^pData[dwDataSize]];
}
}
}
===8<============End of original text============
```

The next, seventh generation of the browser, is most likely going to make this user's data storage mechanism its primary data storage method, declining the good old Protected Storage. Better to say, auto-fill data and passwords, from now on, are going to be stored here.

What is so special and interesting in this mechanism that made MS decide to use it as primary? Well, first of all, it was the encryption idea, which isn't new at all but still simple and genius, to disgrace. The idea is to quit storing encryption keys and generate them whenever that would be necessary. The raw material for such keys would be HTML page's Web address.

Let's see how this idea works in action. Here is IE7's simplified algorithm for saving auto-fill data and password fields:

1 Save Web page's address. We will use this address as the encryption key (EncryptionKey).
2 Obtain Record Key. RecordKey = SHA(EncryptionKey).
3 Calculate checksum for RecordKey to ensure the integrity of the record key (the integrity of the actual data will be guaranteed by DPAPI.) RecordKeyCrc = CRC(RecordKey).

4 Encrypt data (passwords) with the encryption key EncryptedData = DPAPI_Encrypt(Data, EncryptionKey).
5 Save RecordKeyCrc + RecordKey + EncryptedData in the registry.
6 Discard EncryptionKey.

It is very, very difficult to recover password without having the original Web page address. The decryption looks pretty much trivial:

1 When the original Web page is open, we take its address (EncryptionKey) and obtain the record key RecordKey = SHA(EncryptionKey).
2 Browse through the list of all record keys trying to locate the RecordKey.
3 If the RecordKey is found, decrypt data stored along with this key using the EncryptionKey. Data = DPAPI_Decrypt(EncryptedData, EncryptionKey).
In spite of the seeming simplicity, this Web password encryption algorithm is one of today's strongest. However, it has a major drawback (or advantage, depending which way you look at it.) If you change or forget the original Web page address, it will be impossible to recover password for it.

2.8. Content Advisor password

And the last item on our list is Content Advisor password. Content Advisor was originally developed as a tool for restricting access to certain websites. However, for some reason it was unloved by many users (surely, you may disagree with this.) If you once turned Content Advisor on, entered a password and then forgot it, you will not be able to access the majority of websites on the Internet. Fortunately (or unfortunately), this can be easily fixed.

The actual Content Advisor password is not stored as plaintext. Instead, the system calculates its MD5 hash and stores it in Windows registry. On an attempt to access the restricted area, the password entered by user is also hashed, and the obtained hash is compared with the one stored in the registry. Take a look at PIEPR source code checking Content Advisor password:

```
===8<===========Begin of original text===========
void CContentAdvisorDlg::CheckPassword()
{
CRegistry registry;

//read the registry
registry.SetKey(HKLM,
"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\policies\\Ratings");
BYTE pKey[MD5_DIGESTSIZE], pCheck[MD5_DIGESTSIZE];
if ( !registry.GetBinaryData("Key",pKey,MD5_DIGESTSIZE) )
{
MessageBox(MB_ERR,"Can't read the password.");
return;
}

//Get one set by user
CString cs;
m_wndEditPassword.GetWindowText(cs);
MD5Init();
MD5Update((LPBYTE)(LPCTSTR)cs,cs.GetLength()+1);
MD5Final(pCheck);

//Check hashes
if ( memcmp(pKey,pCheck,MD5_DIGESTSIZE)==0 )
MessageBox(MB_OK,"The password is correct!");
else
MessageBox(MB_OK,"Wrong password.");
}
===8<============End of original text============
```

The first thing you may think about is to try to pick the password by using the brute force or dictionary attack. However, there is a more elegant way to that. You can simply remove the hash from the registry. That's it; so simple... Well, it's better to rename it instead, so that if you ever need it, you can restore it back. Some programs also let users check Content Advisor password, "drag out" password hint, toggle password on/off, etc.

3. Brief Overview of Internet Explorer Password Recovery Programs

It's worth noticing that not all password recovery programs suspect there are so many ways to recover passwords. Most likely, this is related to the fact that some passwords (e.g., synchronization passwords) are not often used in the real life, and FTP passwords are not so simple to be 'dragged out'. Here is a brief overview of the most popular commercial products for recovering passwords for the most popular browser on earth :)

Advanced Internet Explorer Password Recovery from the not unknown company, ElcomSoft - does not recognize AutoForm passwords and encrypted FTP passwords. Not to be excluded, the last version of the program may have learnt to do that.

Simple, convenient user interface. The program can be upgraded online automatically.

Internet Explorer Key from PassWare - similarly, does not recognize certain types of passwords. Sometimes the program halts with a critical error when reading some uncommon types of IE's URLs. Displays first two characters of passwords being recovered. The advantages worth noticing are the Spartan user interface and operating convenience.

Internet Explorer Password from Thegrideon Software - not bad, but can recover just three types of Internet Explorer passwords (this is enough for the majority of cases.) Deals with FTP passwords properly. Version 1.1 has problems recovering AutoForm passwords. Has convenient user interface, which in some way reminds one from AIEPR. One can be totally overwhelmed with the beauty and helpfulness of the company's website.

Internet Password Recovery Toolbox from Rixler Software - offers some greater functionality than the previously covered competitors. It can recover encrypted FTP passwords and delete selected resources. However, it has some programming errors. For example, some types of IE records cannot be deleted. The program comes with a great, detailed help file.

ABF Password Recovery from ABF software - quite a good program with friendly user interface. The list of IE record types supported by the program is not long. Nevertheless, it deals with all of them properly. The program can be classified as a multi-functional one, since it can restore passwords for other programs also.

The major drawback of all programs named here is the capability to recover passwords only for user currently logged on.

As it was said above, the general body of stored Internet Explorer resources is kept in a special storage called Protected Storage. Protected Storage was developed specially for storing personal data. Therefore the functions for working with it (called PS API) are not documented. Protected Storage was first introduced with the release of the version 4 of Internet Explorer, which, by the way, unlike the third version, was written from scratch.

Protected Storage provides applications with an interface to store user data that must be kept secure or free from modification. Units of data stored are called Items. The structure and content of the stored data is opaque to the Protected Storage system. Access to Items is subject to confirmation according to a user-defined Security Style, which specifies what confirmation is required

to access the data, such as whether a password is required. In addition, access to Items is subject to an Access rule set. There is an Access rule for each Access Mode: for example, read/write. Access rule sets are composed of Access Clauses. Typically at application setup time, a mechanism is provided to allow a new application to request from the user access to Items that may have been created previously by another application.
Items are uniquely identified by the combination of a Key, Type, Subtype, and Name. The Key is a constant that specifies whether the Item is global to this computer or associated only with this user. The Name is a string, generally chosen by the user. Type and Subtype are GUIDs, generally specified by the application. Additional information about Types and Subtypes is kept in the system registry and include attributes such as Display Name and UI hints. For Subtypes, the parent Type is fixed and included in the system registry as an attribute. The Type group Items is used for a common purpose: for example, Payment or Identification. The Subtype group Items share a common data format.

So, until very recent time, all programs for recovering Internet Explorer passwords used those undocumented API. That's the reason why one significant restriction was applied to the recovery work: PS API can only work with passwords for user that is currently logged on. When the system encrypts data stored in Protected Storage, besides everything else it uses user's SID, without which it is literally impossible (taking into account the current level of computers' calculating performance) to recover stored passwords.

Protected Storage uses a very well thought through data encryption method, which uses master keys and strong algorithms, such as des, sha, and shahmac. Similar data encryption methods are now used in the majority of modern browsers; e.g. in Opera or FireFox. Microsoft, meanwhile, quietly but surely develops and tests new ones. When this article is written, in the pre-Beta version of Internet Explorer 7 Protected Storage was only used for storing FTP passwords.

The analysis of this preliminary version suggests that Microsoft is preparing another 'surprise' in the form of new, interesting encryption algorithms. It is not known for sure, but most likely the new company's data protection technology InfoCard will be involved in the encryption of private data.

Thus, with a great deal of confidence one can assert that with the release of Windows Vista and the 7th version of Internet Explorer passwords will be stored and encrypted with fundamentally new algorithms, and the Protected Storage interface, to all appearances, will become open for third-party developers.

It is somewhat sad, for we think the true potential of Protected Storage was still not uncovered. And this is why we think so:

- First, Protected Storage is based on module structure, which allows plugging other storage providers to it. However, for the last 10 years while Protected Storage exists, not a single new storage provider was created. System Protected Storage is the only storage provider in the operating system, which is used by default.
- Second, Protected Storage has its own, built-in access management system, which, for some reason, is not used in Internet Explorer or in other MS products.
- Third, it is not very clear why MS have decided to decline Protected Storage in storing AutoComplete data and passwords. Decline it as a tried and true data storage, and not data encryption mechanism. It would be more logically proven to keep Protected Storage at least for storing data when implementing a new encryption algorithm. Without fail, there were weighty reasons for that. Therefore, it would be interesting to hear the opinion of MS specialists concerning this subject matter.

4. PIEPR - the First Acquaintance

Passcape Internet Explorer Password Recovery was developed specifically to bypass the PS API's restriction and make it possible to recover passwords directly, from the registry's binary files. Besides, it has a number of additional features for advanced users.

The program's wizard allows you to choose one of several operating modes:
- Automatic: Current user's passwords will be recovered by accessing the closed PS API interface. All current user's passwords currently stored in Internet Explorer will be recovered with a single click of the mouse.
- Manual: Passwords will be recovered without PS API. This method's main advantage is the capability to recover passwords from your old Windows account. For that purpose, you will need to enter path to the user's registry file. Registry files are normally not available for reading; however, the technology used in PIEPR allows doing that (provided you have the local administrative rights.)

User's registry file name is ntuser.dat; its resides in the user's profile, which is normally %SYSTEMDRIVE%:\Documents and Settings\%USERNAME%, where %SYSTEMDRIVE% stands for the system disk with the operating system, and %USERNAME% is normally account name. For instance, path to registry file may look like this: C:\Documents and Settings\John\ntuser.dat

If you have ever been a happy owner of Windows 9x/ME, after you upgrade your operating system to Windows NT, Protected Storage will providently save a copy of your old private data. As a result of that, Protected Storage may contain several user identifiers, so PIEPR will ask you to select the right one before it gets to the decryption of the data (fig.3 http://www.passcape.com/images/ie03.png).

One of the listed SIDs will contain data left by the old Windows 9x/ME. That data is additionally encrypted with user's logon password, and PIEPR currently does not support the decryption of such data.

If ntuser.dat contains encrypted passwords (e.g., FTP sites passwords), the program will need additional information in order to decrypt them (fig.4 http://www.passcape.com/images/ie04.png):
- Logon password of user whose data are to be decrypted
- Full path to the user's MasterKey
- User's SID

Normally, the program finds the last two items in user's profile and fills that data automatically. However, if ntuser.dat was copied from another operating system, you will have to take care of that on your own. The easiest way to get the job done is to copy the entire folder with user's Master Key (there may be several of them) to the folder with ntuser.dat. Master Key resides in the following folder on your local computer: %SYSTEMDRIVE%:\Documents and Settings\%USERNAME%\Application Data\Microsoft\Protect\%UserSid%, where %SYSTEMDRIVE% stands for the system disk with the operating system, %USERNAME% - account name, %UserSid% - user's SID. For example, path to the folder with a master key may look as follows: C:\Documents and Settings\John\Application Data\Microsoft\Protect\S-1-5-21-1587165142-6173081522-185545743-1003. Let's make it clear that it is recommended to copy the entire folder S-1-5-21-1587165142-6173081522-185545743-1003, for it may contain several Master Keys. Then PIEPR will select the right key automatically.

Windows marks some folders as hidden or system, so they are invisible in Windows Explorer. To make them visible, enable showing hidden and system objects in the view settings or use an alternative file manager.

Once the folder with user's Master Key was copied to the folder with ntuser.dat, PIEPR will automatically find the required data, so you will only have to enter user's password for recovering FTP passwords.

Content Advisor
Content Advisor passwords, as it was said already, is not kept as plain text; instead, it is stored as hash. In the Content Advisor password management dialog, it is enough to just delete (you can restore the deleted password at any time later) or change this hash to unlock sites locked with Content Advisor.

PIEPR will also display your password hint if there is one.

Asterisks passwords
PIEPR's fourth operating mode, which allows recovering Internet Explorer passwords hidden behind asterisks. To recover such password, simply drag the magnifier to the window with a **** password. This tool allows recovering passwords for other programs that use IE Frames as well; e.g., Windows Explorer, some IE-based browsers, etc.

We have reviewed the basic Internet Explorer password recovery modes. There is also a number of additional features for viewing and editing cookies, cache, visited pages history, etc. We are not going to cover them in detail; instead, we are going to look at a few password recovery examples done with PIEPR.

5.1. Three Real-Life Examples.
Example 1: Recovering current user's FTP password
When opening an FTP site, Internet Explorer pops up the log on dialog (fig.5 http://www.passcape.com/images/ie05.png).

If you have opened this site and set the 'Save password' option in the authentication dialog, the password must be saved in Protected Storage, so recovering it is a pretty trivial job. Select the automatic operating mode in PIEPR and then click 'Next'. Locate our resource in the dialog with decrypted passwords that appears (the site name must appear in the Resource Name column.)

As we see, the decryption of current user's password should not cause any special difficulties. Oh, if the password is not found for some reason - don't forget to check IE's Auto-Complete Settings. Possibly, you have simply not set the program to save passwords.

5.2. Three Real-Life Examples.
Example 2: We will need to recover Web site passwords. The operating system is unbootable.
This is a typical, but not fatal situation. The necessity to recover Internet Explorer passwords after unsuccessful Windows reinstallation occurs just as often.

In either case, we will have user's old profile with all files within it. This set is normally enough to get the job done. In the case with the reinstallation, Windows providently saves the old profile under a different name. For example, if your account name was John, after renaming it may look like John.WORK-72C39A18.

The first and the foremost what you must do is to gain access to files in the old profile. There are two ways to doing this:
- Install a new operating system on a different hard drive; e.g., Windows XP, and hook the old hard drive to it.
- Create a Windows NT boot disk. There are many different utilities for creating boot disks and USB flash disks available online. For instance, you can use WinPE or BartPE. Or a different one. If your old profile was stored on an NTFS part of your hard drive, the boot disk will have to support NTFS.

Let's take the first route. Once we gain access to the old profile, we will need to let the system show hidden and system files. Otherwise, the files we need will be invisible. Open Control Panel, then click on Folder Options, and then select the View tab. On this tab, find the option 'Show hidden files and folders' and select it. Clear the option 'Hide protected operating system files'. When the necessary passwords are recovered, it's better to reset these options to the way they were set before.

Open the program's wizard in the manual mode and enter path to the old profile's registry file. In our case, that is C:\Documents And Settings\John.WORK-72C39A18\ntuser.dat. Where John.WORK-72C39A18 is the old account name. Click 'Next'.

This data should normally be sufficient for recovering Internet Explorer passwords. However, if there is at least a single encrypted FTP password, the program will request additional data, without which it will not be able to recover such types of passwords:
- User's password
- User's Master Key
- User's SID.
Normally, the program finds the last two items in user's profile and fills that data automatically. However, if that didn't happen, you can do that by hand: copy ntuser.dat and the folder with the Master Key to a separate folder. It is important to copy the entire folder, for it may contain several keys, and the program will select the right one automatically. Then enter path to file ntuser.dat that you have copied to another folder.

That's it. Now we need to enter the old account password, and the recovery will be completed. If you don't care for FTP password, you can skip the user's password, Master Key, and SID entry dialog.

5.3. Three Real-Life Examples.
Example 3: Recovering uncommonly stored passwords.
When we sometimes open a website in the browser, the authentication dialog appears. However, PIEPR fails to recover it in either automatic or manual mode. The 'Save password' option in Internet Explorer is enabled. We will need to recover this password.

Indeed, some websites don't let browser to save passwords in the auto-complete passwords list. Often, such websites are written in JAVA or they use alternative password storage methods; e.g., they store passwords in cookies. A cookie is a small bit of text that accompanies requests and pages as they go between the Web server and browser. The cookie contains information the Web application can read whenever the user visits the site. Cookies provide a useful means in Web applications to store user-specific information. For example, when a user visits your site, you can use cookies to store user preferences or other information. When the user visits your Web site another time, the application can retrieve the information it stored earlier. Cookies are used for all sorts of purposes, all relating to helping the Web site remember you. In essence, cookies help Web sites store information about visitors. A cookie also acts as a kind of calling card, presenting pertinent identification that helps an application know how to proceed. But often cookies criticized for weak security and inaccurate user identification.

If the password field is filled with asterisks, the solution is clear: select the ASTERISKS PASSWORDS operating mode and then open the magic magnifier dialog. Then simply drag the magnifier to the Internet Explorer window (fig.6 http://www.passcape.com/images/ie06.png).

The password (passwords, if the Internet Explorer window has several fields with asterisks) is to appear in the PIEPR window (fig.7 http://www.passcape.com/images/ie07.png).

But it's not always that simple. The password field may be empty or that field may indeed contain *****. In this case, as you have guessed by now, the ASTERISKS PASSWORDS tool will be useless.

We can suppose, the password is stored in cookies. Let's try to locate it. Choose the IE Cookie Explorer tool (fig.8 http://www.passcape.com/images/ie08.png).

The dialog that appears will list the websites that store cookies on your computer. Click on the URL column header to order the websites list alphabetically. This will help us find the right website easier. Go through the list of websites and select the one we need. The list below will display the decrypted cookies for this website (fig.9 http://www.passcape.com/images/ie09.png).

As the figure shows, in our case the login and password are not encrypted and are stored as plain text.

Cookies are often encrypted. In this case, you are not likely to succeed recovering the password. The only thing you can try doing in order to recover the old account is to create a new account. Then you will be able to copy the old cookies in a text editor and replace them with the new ones. However, this is only good when the worst comes to the worst; it is not recommended to use it normally.

Don't forget also that just about all pages and forms with passwords have the 'Forgot password' button.

Conclusion

As this article shows, recovering Internet Explorer passwords is a pretty simple job, which does not require any special knowledge or skills. However, despite of the seeming simplicity, password encryption schemes and algorithms are very well thought through and just as well implemented. Although the Protected Storage concept is over 10 years of age, don't forget that it has proven the very best recommendations of the experts and has been implemented through three generations of this popular browser.

With the release of the next, 7th version of IE, Microsoft is preparing fundamentally new schemes for protecting our private data, where it uses improved encryption algorithms and eliminates shortages peculiar to Protected Storage.

In particular, the analysis of the preliminary beta versions of Internet Explorer 7 has revealed that autoform password encryption keys are no longer stored along with data. They are not stored, period! This is a little know-how, which is to be estimated at its true worth by both professionals and end users, who, finally, will benefits of it anyway.

But the main thing is, the release of the new concept will eliminate the major drawback peculiar to Protected Storage, which is the possibility to recover passwords without knowing the additional information. Better to say, was enough for a potential hacker to gain physical access to the contents of a hard drive, in order to steal or damage passwords and user's other private data. With the release of Internet Explorer 7, the situation will somewhat change.

Meanwhile, we will only have to wait impatiently for the advent of Windows Vista and IE 7 to take a closer look at new encryption mechanisms used in the next generation of this popular browser.