Windows Password Recovery

User manual

Passcape Software

Introduction

1 Introduction

1.1 About the program

Welcome to **Windows Password Recovery**, a network security analyzer and Windows password recovery utility. Windows Password Recovery is the only solution that implements the most advanced, patented password recovery technologies developed by Passcape Software programmers, such as *Artificial Intelligence* or *Pass-phrase* attack.

Compared to similar products, Windows Password Recovery features a number of competitive advantages:

For home users - easy set up and use. Easily recovers or resets forgotten passwords to any Windows account.

For system administrators - password audit reveals security breaches, helping the administrators to ensure the reliability and security of the corporate network. Checks the security level of Windows operating systems.

For forensics, industry and government security experts - analyzes and audits system security policies, issues recommendations on improving the stability of the operating systems' password protection.

1.2 Features and benefits

- Contemporary, easily customizable graphical user interface.
- Load hashes from 9 different programs.
- Imports directly from SAM or ntds.dit; even if the files are locked by the system, the program still reads them.
- Imports hashes from remote computers.
- Import hashes from system shadow copies, restore points, backup and repair folders.
- Can backup\save local registry files and Active Directory database.
- Imports password history hashes.
- Recovers some account passwords on the fly (when importing locally).
- Supports Active Directory (domain accounts).
- Supports importing from 64-bit systems.
- Exports hashes to the PWDUMP file.
- The software has 17 types of different attacks; 10 of them are unique, developed by our company, implemented upon patented technologies.
- The program supports multithreading, fully leveraging the power of modern computers.
- Dictionary attack supports text dictionaries in the ASCII, UNICODE,UTF8, PCD, RAR and ZIP formats.
- Broad choice of online dictionaries for dictionary attacks (about 2 GB)
- Some of the program's functions - e.g., word mutation - are unique. For example, the total number of mutation rules exceeds a hundred and fifty. Not any other similar application features that!
- Supports unlimited number of inspected hashes.
- Intelligent analysis of found passwords.
- High search speed on modern computers - over 100 million of passwords per second for 4-core CPUs and over billion passwords using GPU power.

- Includes auxiliary tools: hash generator, password strength check, rainbow table creation, etc.
- Extended toolset to work with wordlists: create, sort, convert, etc.
- Addon modules for forensics and researchers: LSA secrets editor, domain cached credentials viewer, Active Directory and SAM explorers, DPAPI offline decoder.
- Advanced password reports

Program's interface

2 Program's interface

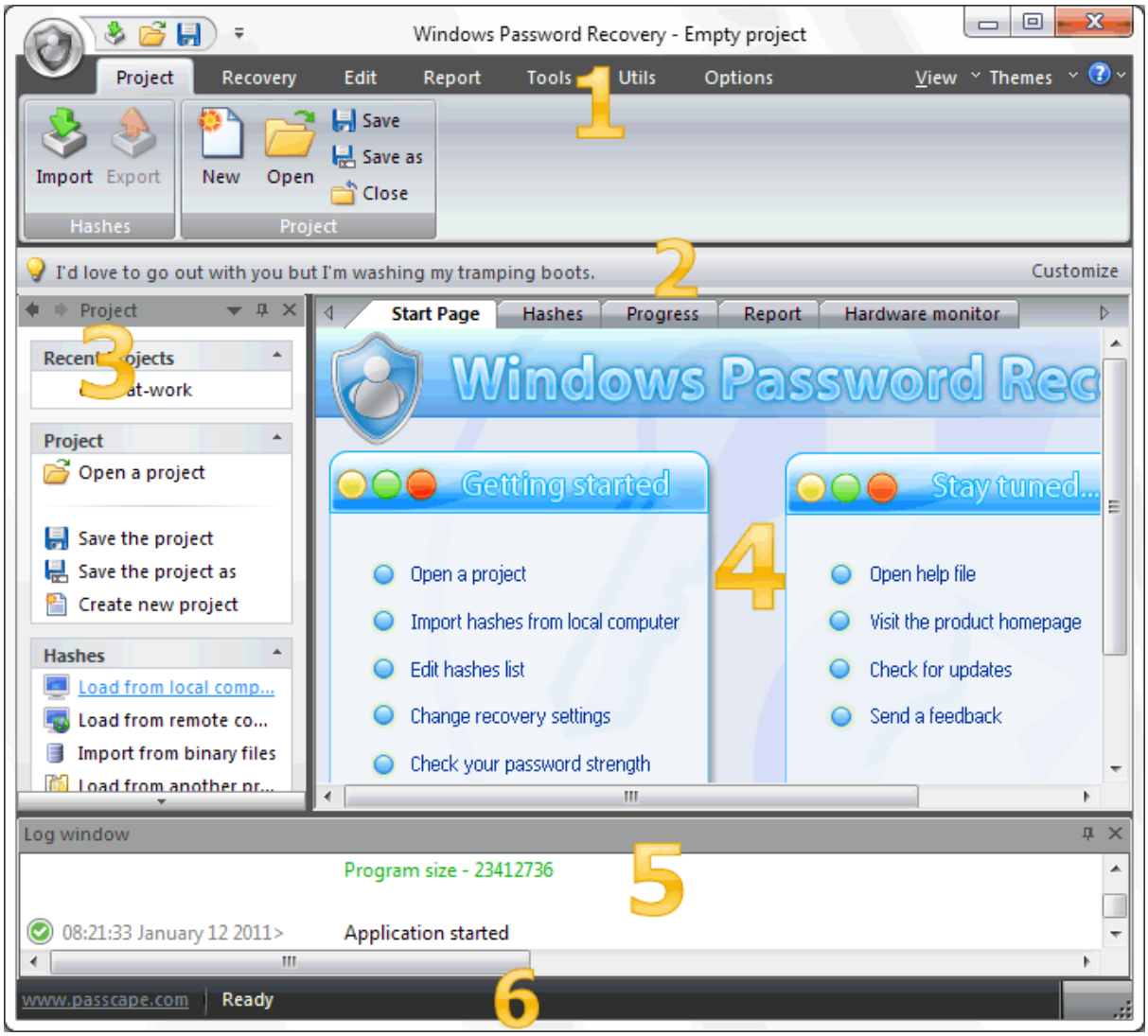
2.1 Overview

The program's interface is made in the form of the SDI architecture, i.e. it allows working with only one project at a time. The program's operation can be conventionally divided into 4 stages:

1. Creating a project
2. Importing (loading) password hashes to the project. Editing the hashes: deleting, adding, selecting, etc.
3. Recovering the hashes. Includes selecting, configuring and launching the selected one or several attacks.
4. Analyzing the results.

The entire interface can be conventionally divided into several components:

- Menu Bar
- Information Bar - for displaying brief information texts - like tips, warnings, etc.
- Task Bar - duplicates and compliments the menu bar, providing quick access to the most common operations. Consists of three parts:
  - Project - includes the main operations over project - like opening, closing, creating a new project, and importing hashes.
  - Hash Editor. Duplicates the most common editing operations.
  - Tools - includes a clock, calendar, and calculator.
- Main Window - bears the main burden and consists of 5 parts. The first tab is the welcome window. The second tab contains the list of hashes to be analyzed and recovered. Then there goes a tab with the current attack state (progress) indicator and a tab with the statistics and reports. And finally - a tab with the hardware monitor.
- Log Window - displays information on the current state of the application, current operation, etc. The program's log can be copied to clipboard or saved to a file (right-clicking opens the corresponding menu).
- Status Bar is designed for informational purposes.

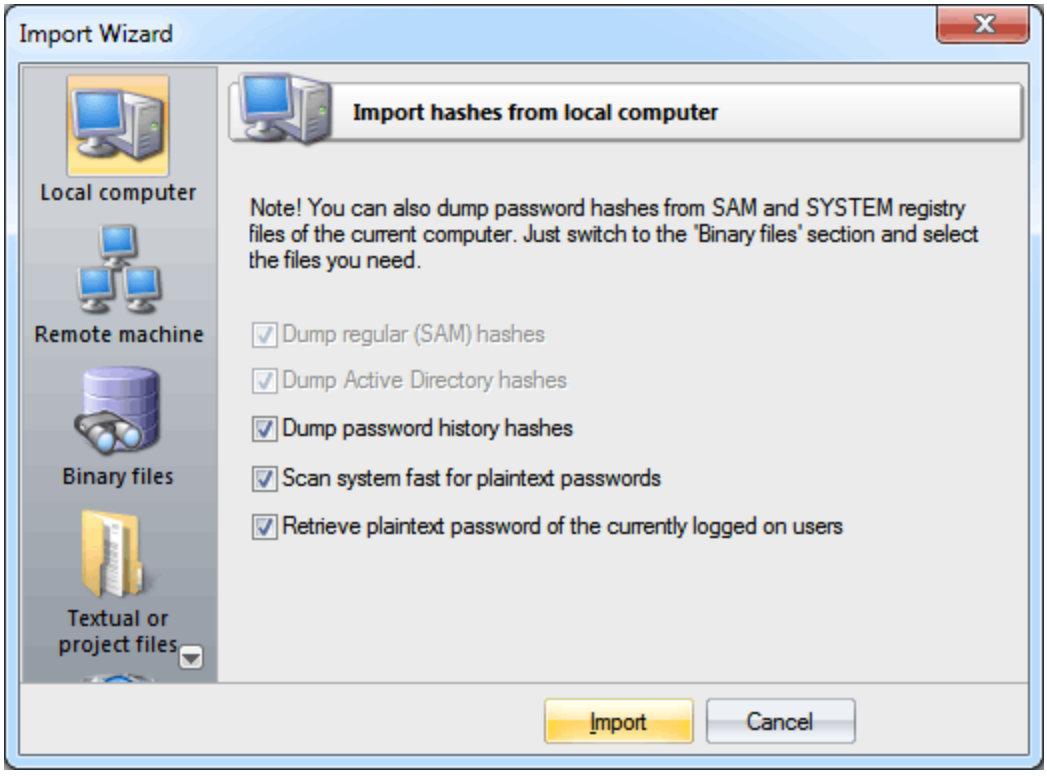


2.2 Project menu

2.2.1 Import

Windows Password Recovery offers a broad range of options for loading hashes depending on your capabilities. There are 5 major ways to import hashes to the program.

2.2.1.1 Import local hashes



Import hashes from the local computer - the most preferable method, as it implies the deepest overall analysis of the system and the passwords. Besides that, the hashes that are imported from the local computer can undergo the sophisticated *Intelligent attack*, which allows to relatively quickly recover the passwords to some accounts.

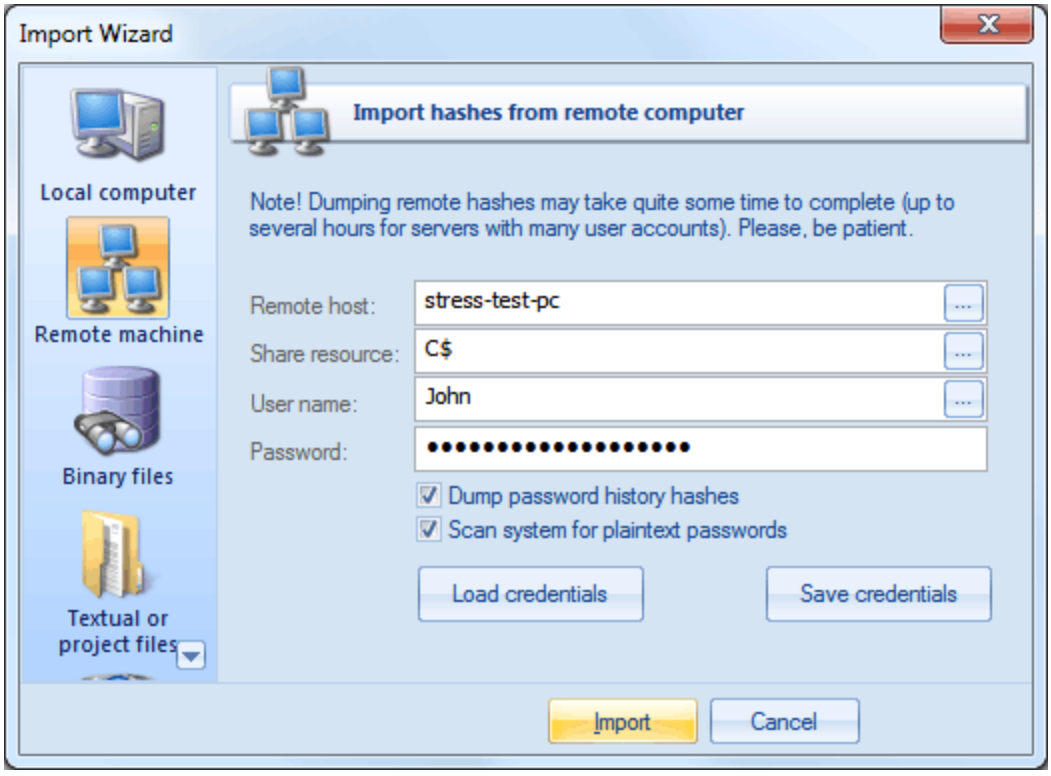
Importing local hashes runs well regardless to where the hashes are localized: in SAM or in Active Directory. This item has two additional options: dumping password history hashes and searching for plain-text passwords that are stored in the system. The very process of searching for plain-text passwords is divided into 4 steps and consists of the actual searching for the passwords that are stored in the system using the reverse encryption, searching for the text passwords for the system accounts, searching for start-up passwords and an extra step, when the program analyzes some of the uncovered accounts, passwords to which can also be recovered from the system (for example, for the HomeGroupUser$ account in Windows 7).

If disabling the last 2 options is not desired, as it allows to relatively painlessly and quickly recover the complete passwords to some of the system's accounts, the password history dump is completely opposite - disabling it is often very useful. For example, when the number of passwords to be imported exceeds hundreds of thousands or even millions. On the other hand, the program has a power of artificial intelligence, so if during an attack it finds one of the history passwords, it will take every effort to recover the remaining passwords by analyzing the user's preferences for the recovered password.

One of the latest version of the program can also dump user history hashes from DPAPI CREDHIST file. So setting the option is recommended now.

The local import functionality requires administrative privileges.

2.2.1.2 Import hashes from remote computer



Import hashes from a remote host. The program has means for dumping hashes from a remote host without employing third-party utilities. This does not compromise the remote system, as it still requires supplying the credentials for the remote host user.

Dumping from a remote host works as follows. First, you should enter the remote host name in the Remote Host field. You can use the [...] button to browse the network. Once you have selected the remote host, set up a shared resource (allowed for both reading and writing), through which the data will be transmitted. Usually, that is either C$ or ADMIN$. Here too, you can take advantage of the browse button to the right of the edit box. Next, in the two fields at the bottom type in the remote host account name and the password.

The '*Save Credentials'* button saves current settings. Respectfully, the *'Load Credentials'* button allows loading existing settings, so that you don't have to enter them manually every time you need them. The password is stored in the encrypted form!

This import option also requires administrative privileges on the target PC.

You may, however, experience some troubles connecting to remote PC, even if you have an Administrator account. When connection to the target PC with Windows Vista/7/8/10, you may get the following error:

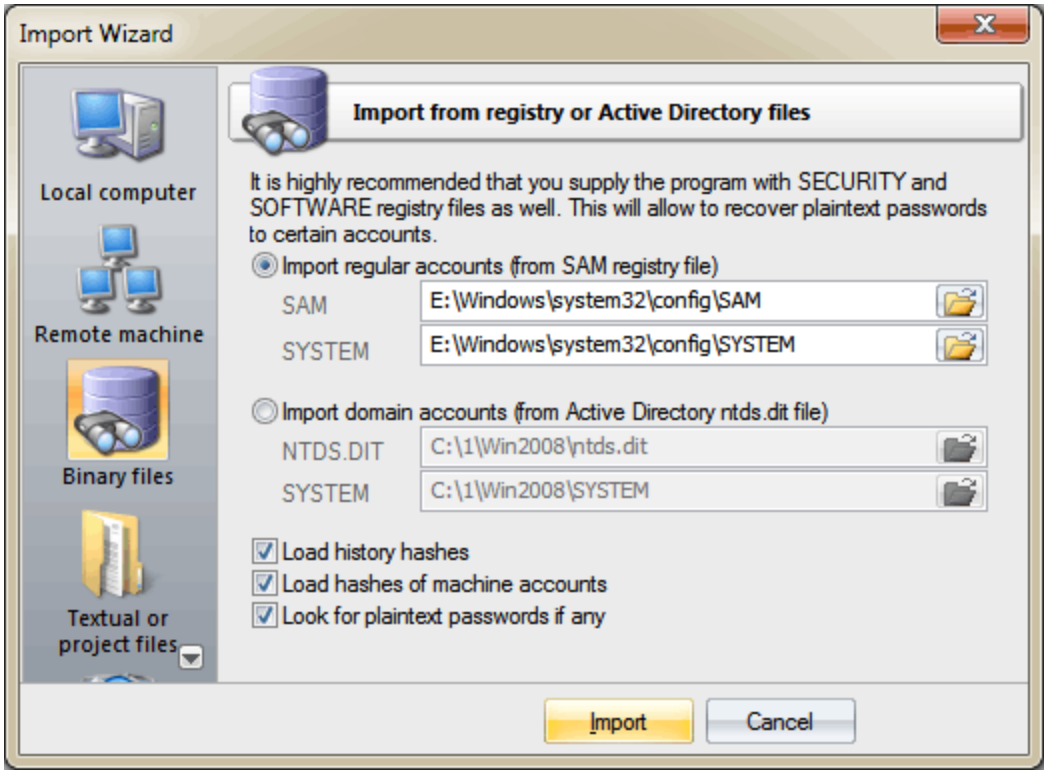
16:34:18 June 11 2015> Application started
16:35:27 June 11 2015> Importing from remote machine
16:35:27 June 11 2015> COMP: JOHN-PC
16:35:27 June 11 2015> SHARE: C$
16:35:27 June 11 2015> USER: John
16:35:30 June 11 2015> system error 5
16:35:32 June 11 2015> Failed to run remote service: can't connect remote machine.

The error 5 indicates that access is denied (even if the target account has Administrator privileges). The problem is that any remote connection in Windows Vista and higher OSes by default cannot perform administrative tasks. Microsoft documentation clearly states the following:

> *"When a user with an administrator account in a Windows Vista computer's local Security Accounts Manager (SAM) database remotely connects to a Windows Vista computer, the user has no elevation potential on the remote computer and cannot perform administrative tasks. If the user wants to administer the workstation with a SAM account, the user must interactively log on to the computer to be administered."*

There's a however a flag in the Windows registry that allows to change the default behavior. Just launch the registry editor of the target PC and open the following key:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\system
Then create DWORD value *LocalAccountTokenFilterPolicy* and set it to one (1). So you will be able to connect to the admin share.

2.2.1.3 Import hashes from binary files



Import hashes from binary files. Windows Password Recovery can extract password hashes directly from binary files. Even those of them that are currently used by the system (i.e. locked).

Normally, password hashes are stored in the registry file SAM, which resides in the '%WINDOWS% \System32\Config' folder. The same folder contains the SYSTEM registry, which is necessary for the recovery. If you have specified path to the registry in the current system, parsing it will take a bit longer (normally by a few seconds).

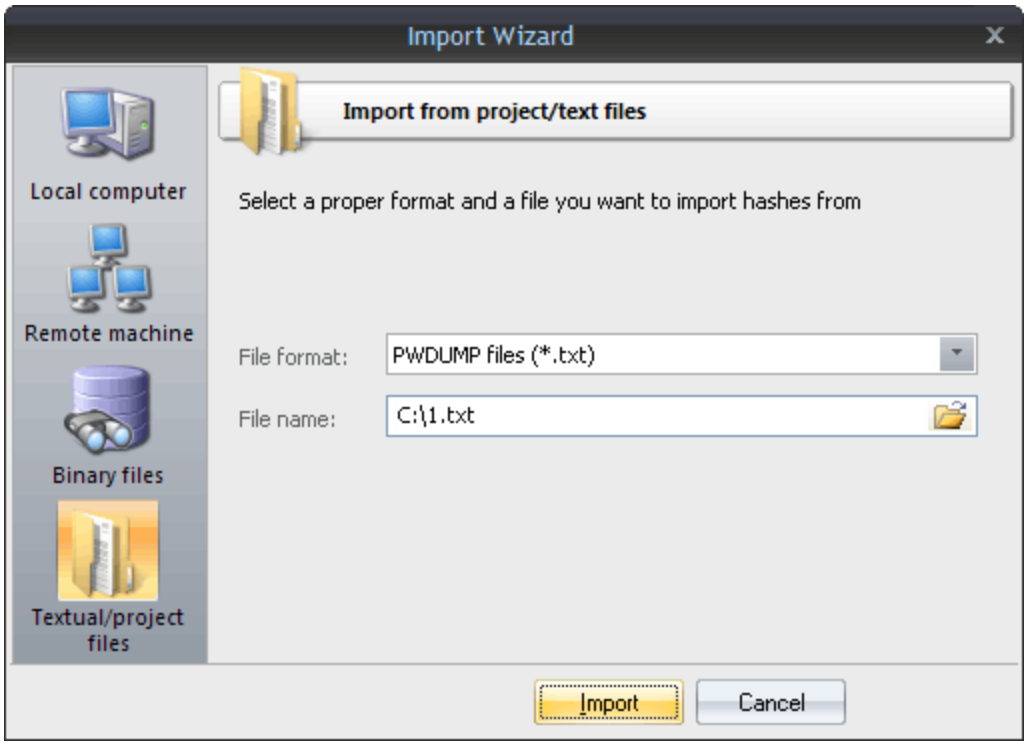
Password hashes for domain accounts are stored in the Active Directory database; or, to be more specific, in the very heart of it, in the ntds.dit file, which resides in the folder: '%Windows%\ntds'. The recovery of domain accounts also requires the SYSTEM registry file. Be careful! Dumping from the current system's AD database may take some time, especially when ntds.dit is of a considerable size.

The program works properly and supports all the SYSKEY encryption options: Registry SYSKEY, SYSKEY startup diskette, SYSKEY startup password.

If you are copying the files from other system, besides the SAM (ntds.dit) and SYSTEM files, it is also highly recommended to copy the SECURITY and SOFTWARE registries (they should be located in the same folder with the SYSTEM file); that would allow you to recover the passwords to some user accounts quicker.

Using additional options you can:
- Turn on/off loading history hashes. Turning off history loading will increase database parsing. From the other hand, when processing (attacking) hashes, guessing history passwords may give a clue to figure out the password for the primary account the hashes belong to.
- Discard loading machine accounts (ones end up with $ character).
- Switch on/off instant check for plaintext passwords, if any.

2.2.1.4 Import from project/text files



Finally, you can load the hashes to your project by **importing them from other applications**. The software supports the following formats:

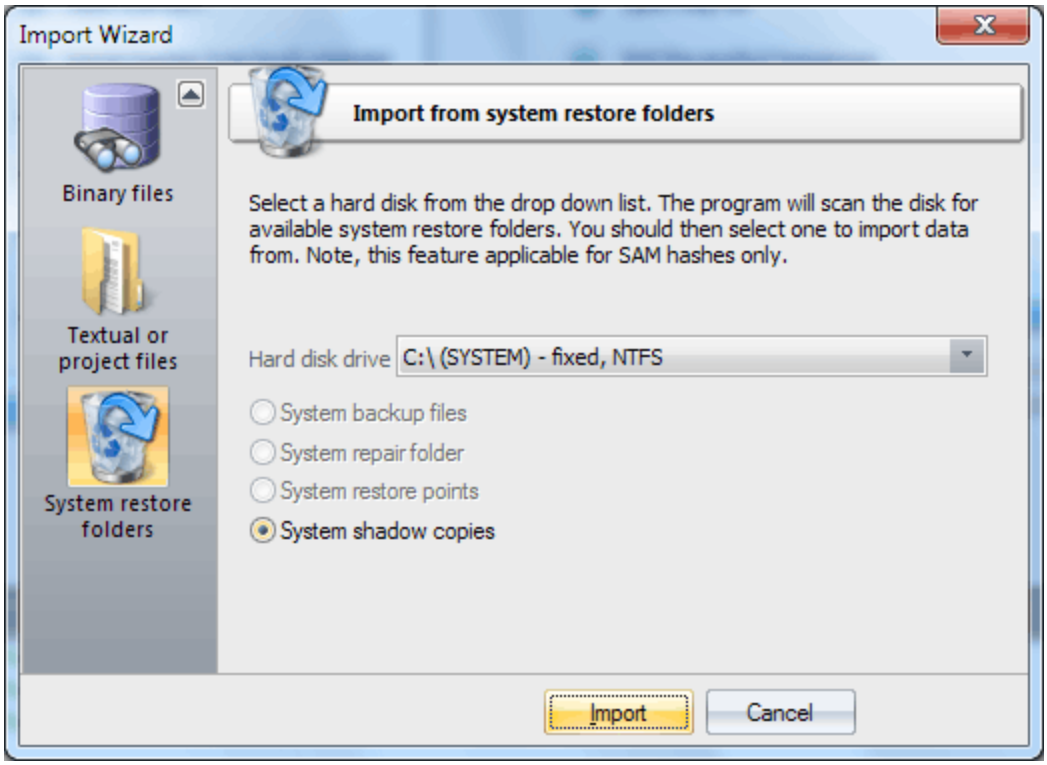
- **PWDUMP** - despite many disadvantages, this is a de facto standard format for storing password hashes. Important note: This format does not fully support national characters. Therefore, some user

names or comments may not display correctly. Windows Password Recovery also supports textual PWDUMP files in UNICODE.

- **LophtCrack (*.lcs)** - this file format is used by the LophtCrack software. The program supports all versions of LCS files, beginning with v4.
- **Project files *.hdt**, which are used by Proactive Password Auditor (used to be PWSEX) from ElcomSoft. Also supports all versions of the format, beginning with v3.
- ***.hsh** files, which are exported by Proactive System Password Recovery from the same notorious company.
- **Hash lists *.lst**, created by Cain & Abel. Windows Password Recovery supports lst files beginning with v.4.9.12. The earlier versions of LST files used the ';' delimiter instead of 'TAB'. Unfortunately, the LST file does not have a marker that specifies the version; therefore, if the LST file is unreadable, you may have to manually replace all the field delimiters with the 'TAB'.
- ***.winpsw files**, created by WinPassword, from good old LastBit. Supports all versions of WINPSW, beginning with v6.
- **SamInside project files (*.hashes)**. This format is similar to text PWDUMP, but it is more flexible and uses the 0x7f character instead of colon, which is more reasonable.
- **PasswordPro project files (*.hashes)**. This format is similar to text PWDUMP, except several changes. It is used by PasswordsPro product.
- **Passcape Universal Configuration Files (*.puc)**. This container is used in Reset Windows Password software and can contain several different dumps.
- **Plain hashes (*.*)**. Raw hashes in plain text format (32 or 16 characters on a line).

After importing hashes, the program automatically marks all the LM or NT hashes and launches the preliminary attack. This action is optional and can be disabled in the general settings. This option is enabled by default.

2.2.1.5 Importing hashes from system restore folders



Yet another, not a less helpful option is **importing hashes from the system restore folders**. All you would need for that is to specify the path to one of the disks. The program will automatically find the

recovery folders and, if it finds the necessary files, import the hashes.

The search is performed, first of all, in the system directory. Second, in '%Windows%\Repair' folder, which normally contains system registry backups. Third, in the 'System Volume Information' folder, which is used for undoing changes made to the system. This technology has been available since Windows XP and is also known as System Restore (XP) or Shadow Coping (Vista+).

Be careful though, the registry backups may contain obsolete data!

2.2.2 Export

All project hashes, along with the settings, are stored in the project file (*.wpr); however, for the sake of greater flexibility and compatibility with other software, the program can export hashes to a PWDUMP or POT files. If 'Export to POT' is chosen, all found passwords along with corresponding NTLM hashes will be saved to file in the following format:
hash:password
The passwords are UTF8 encoded.

2.2.3 New

Saves current project and creates a new one.

2.2.4 Open

Loads/opens a new project. The application's projects have the *.wpr extension and contain program settings and hashes. However, for speeding up the search speed, the program stores the current state of the attack in a separate file progress.ini.

2.2.5 Save

Saves current project. It is recommended to save critical projects from time to time.

2.2.6 Save as

Saves current project under a different name (renames it).

2.2.7 Close

Closes current project.

2.3 Recovery menu

This menu item allows selecting and launching an attack. The '*Attack*' pane allows selecting the type of the attack and toggle between attacking LM or NT hashes. Take a note that before actually launching the attack you must have selected/marked the necessary hashes. You can do that through the **Edit-Select** menu. Launching the attack assumes that you have also made all the required settings (on the **Options-Attack Options** menu).

2.3.1 Run

Launches selected attack. When the attack is running, all other items on the menu are disabled. Please note that when the attack is over, the program runs a special mutation and password analysis routine over the found passwords. This option is enabled by default, but it can be disabled in the general settings.

2.3.2 Continue

Resumes attack from the last stored point. Please remember that the last stored point is automatically erased when changes are made to the attack's options.

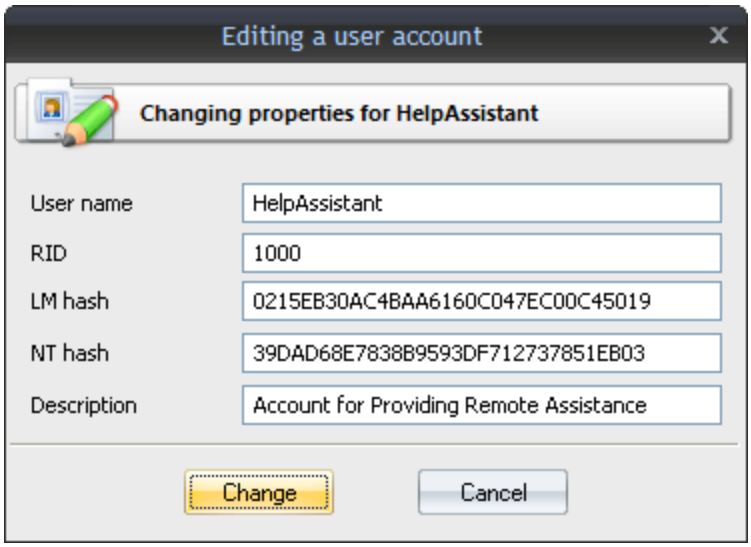
2.3.3 Stop

Pauses current attack.

2.4 Edit menu

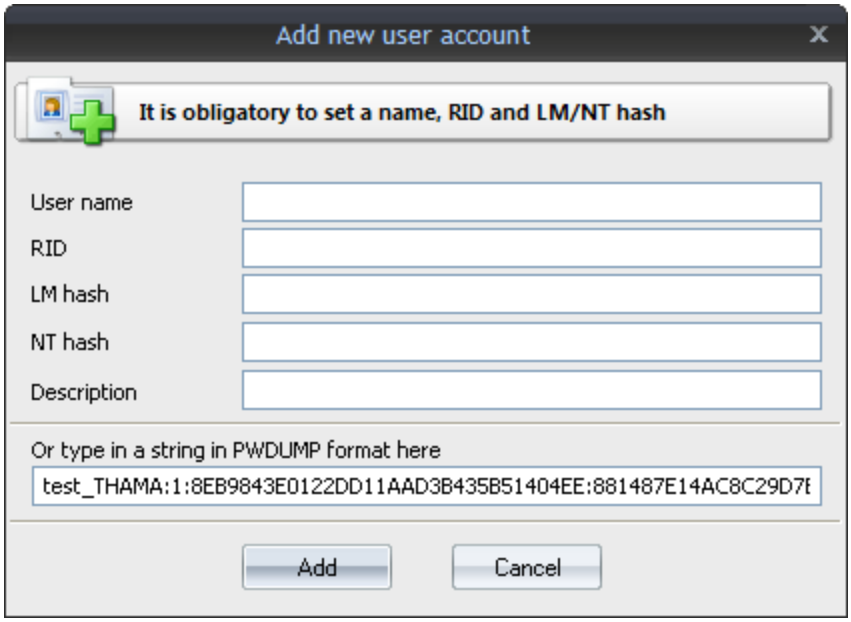
The Edit menu is available only when the *'Hashes'* tab is active; it includes four items: Edit, Copy, Select, and Search.

2.4.1 Edit



Selecting this item opens the dialog where you can manually edit the following fields for the selected account: user name, user RID, LM and NT hashes, plus the comment to the account.

2.4.2 Add



This item allows adding items manually. It allows entering PWDUMP-like strings.

2.4.3 Delete

Deletes entries from list: highlighted (i.e. the one being under the cursor), marked or all at once.

2.4.4 Reset passwords

Drops all found passwords for list.

2.4.5 Copy

Copies current (highlighted) entry to Windows clipboard. Copies only the selected portion of the entry, not the entire entry. For example, user name or the found password.

2.4.6 Select

Selects hashes to be attacked (ones with checkbox option is on). If during the attack the password for the selected hash is found, the checkbox will be automatically cleared, and the record will be marked green. To select the NT hashes, you must first have deselected the LM hashes, and the other way around.

2.4.7 Search

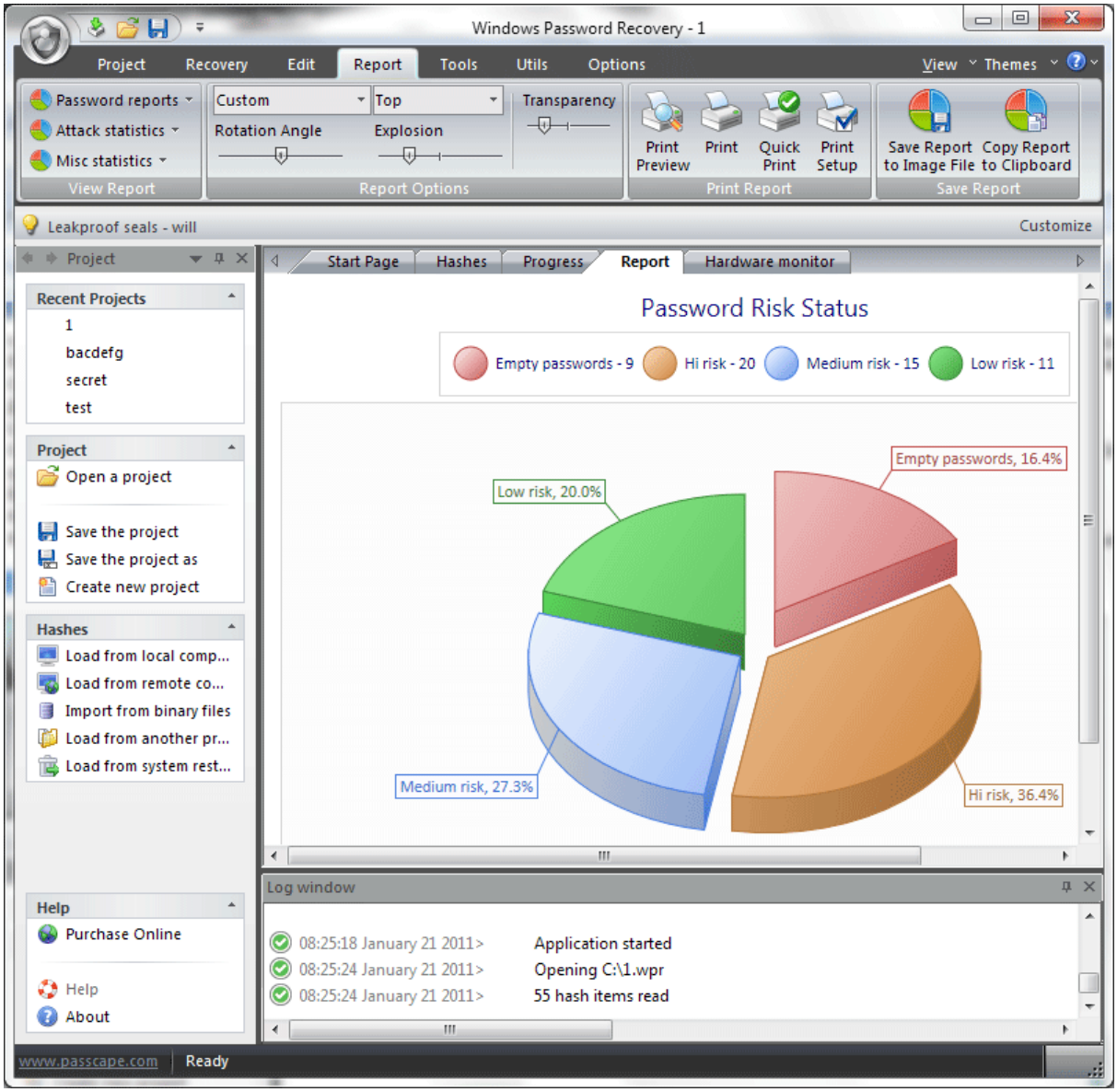
When the number of entries exceeds a hundred of thousands, finding a specific entry often takes quite a

bit of an effort. To make the job easier, the program offers the search of two types: searching a specific field - e.g., user name - and quick-searching of serial entries. In the latter case, the program scans the entire entry, character by character.

2.5 Reports Menu

You can create, print or save one of the program's reports here. The following reports are available:

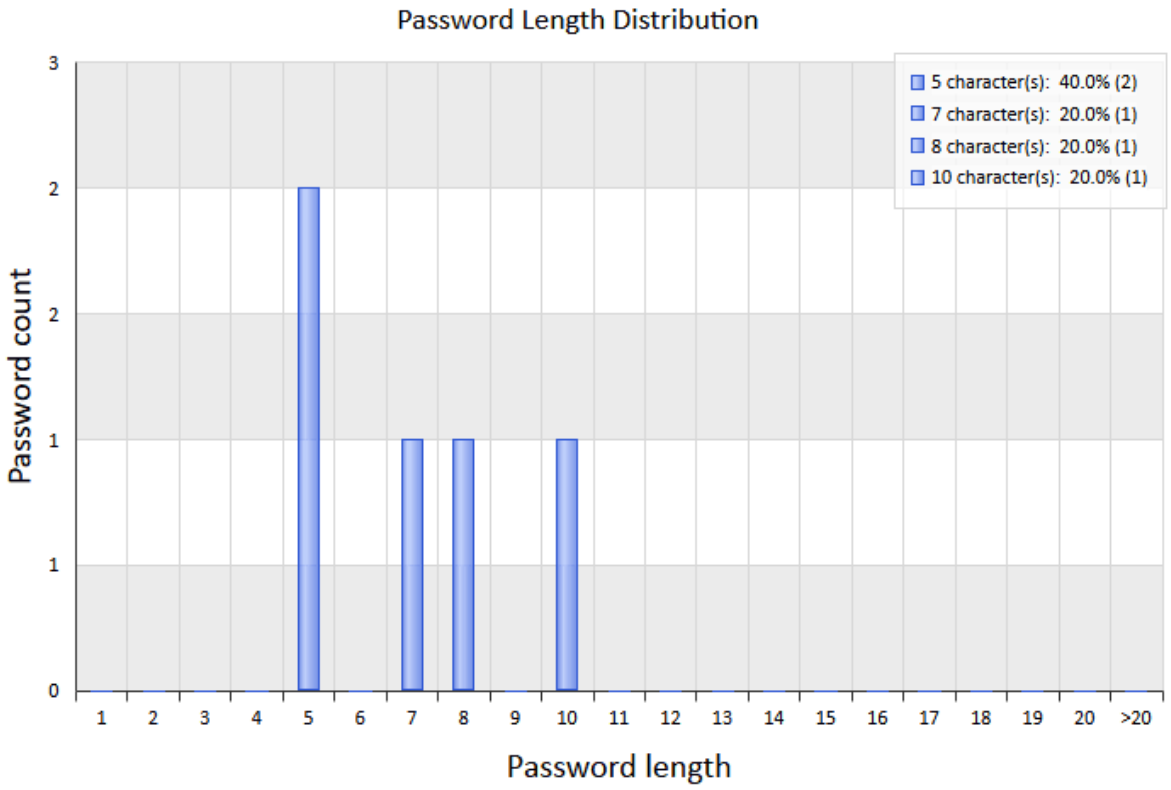
- Password reports
- Attack statistics
- Miscellaneous statistics
- Account statistics
- Password-list analysis
- Group information



2.5.1 Password reports

The following reports are available here:

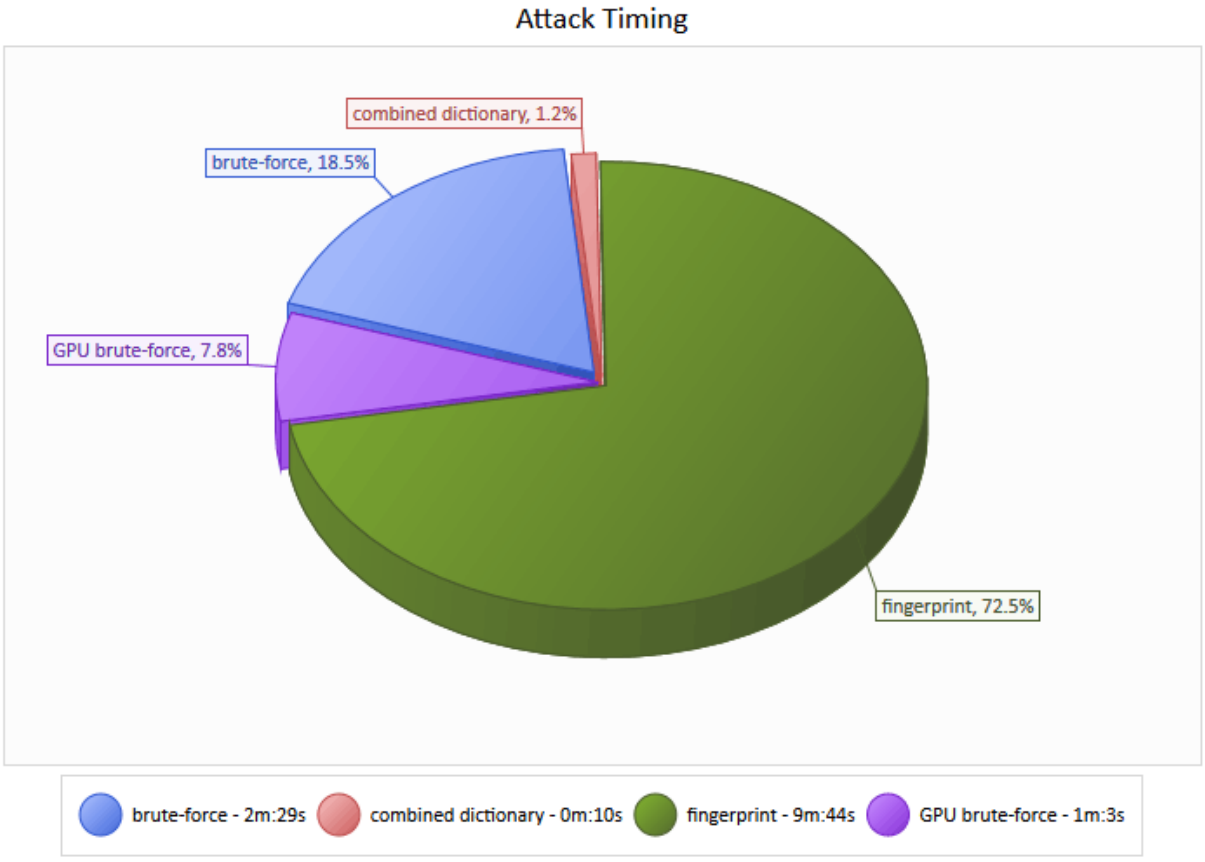
- **Password risk status** - displays empty, found, and not recovered passwords
- **Password complexity** - reports the number of passwords and various character sets being audited
- **Password length distribution** - shows overall length of the broken passwords
- **Password uniqueness** - this report shows unique against reused passwords chart.
- **Top reused passwords** - displays top 20 of the most popular passwords.
- **LM vs NT** - reports the number of LM and NT hashes
- **Regular vs history passwords** - reports the number of common and history passwords (only for hashes imported from SAM\NTDS.DIT files; eg. imported from a local computer)
- **Password recovery time** - time took to crack a certain password(s). Most vulnerable passwords are marked in red palette.
- **Recovered vs unbroken passwords** - displays the number of discovered and not-found passwords
- **Passwords found** - shows a bit detailed report on found passwords



2.5.2 Attack statistics

Attack statistics includes the following items:

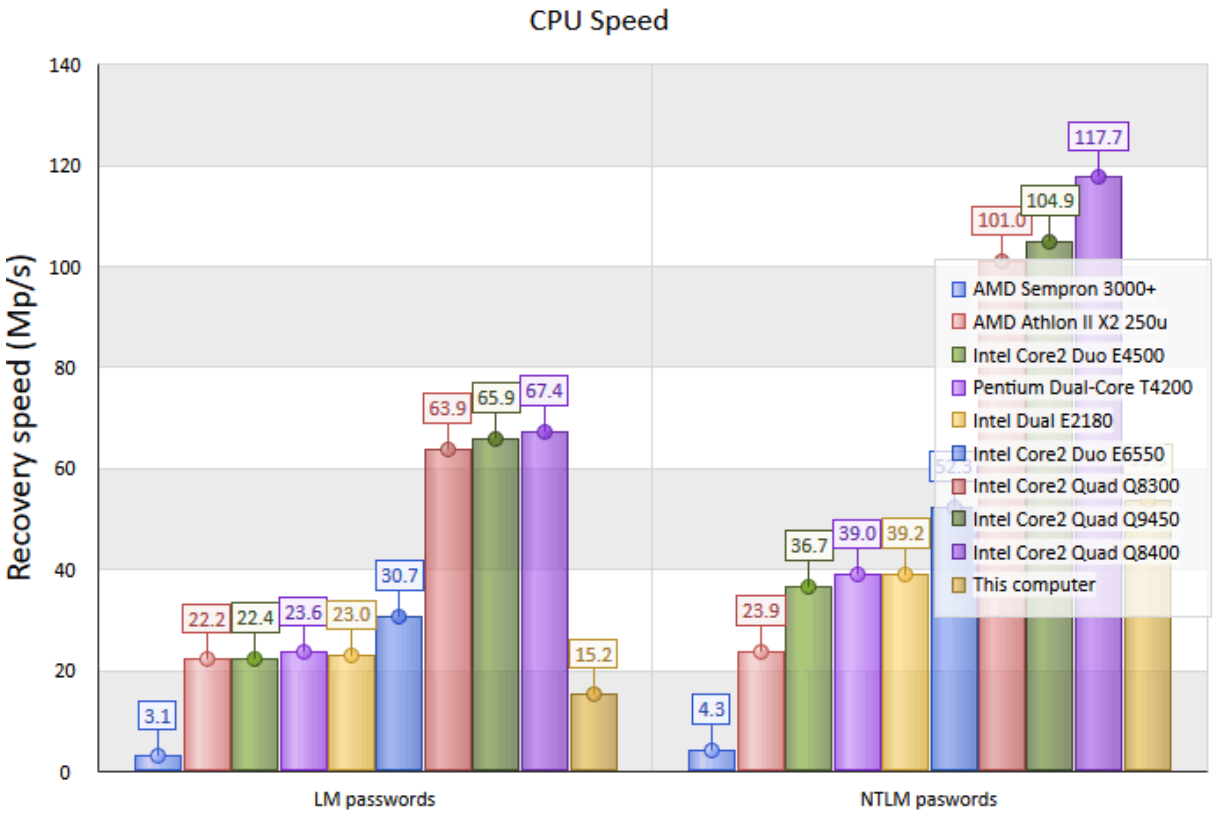
- **Preferred attack** - statistics on number and type of used attacks.
- **Attack time** - analysis of time spent on each attack.
- **Attack efficiency1** - efficiency analysis: time spent vs. passwords found during attack ratio.
- **Attack efficiency2** - efficiency analysis: overall efficiency for each attack.



2.5.3 Miscellaneous statistics

Some additional stuff like:

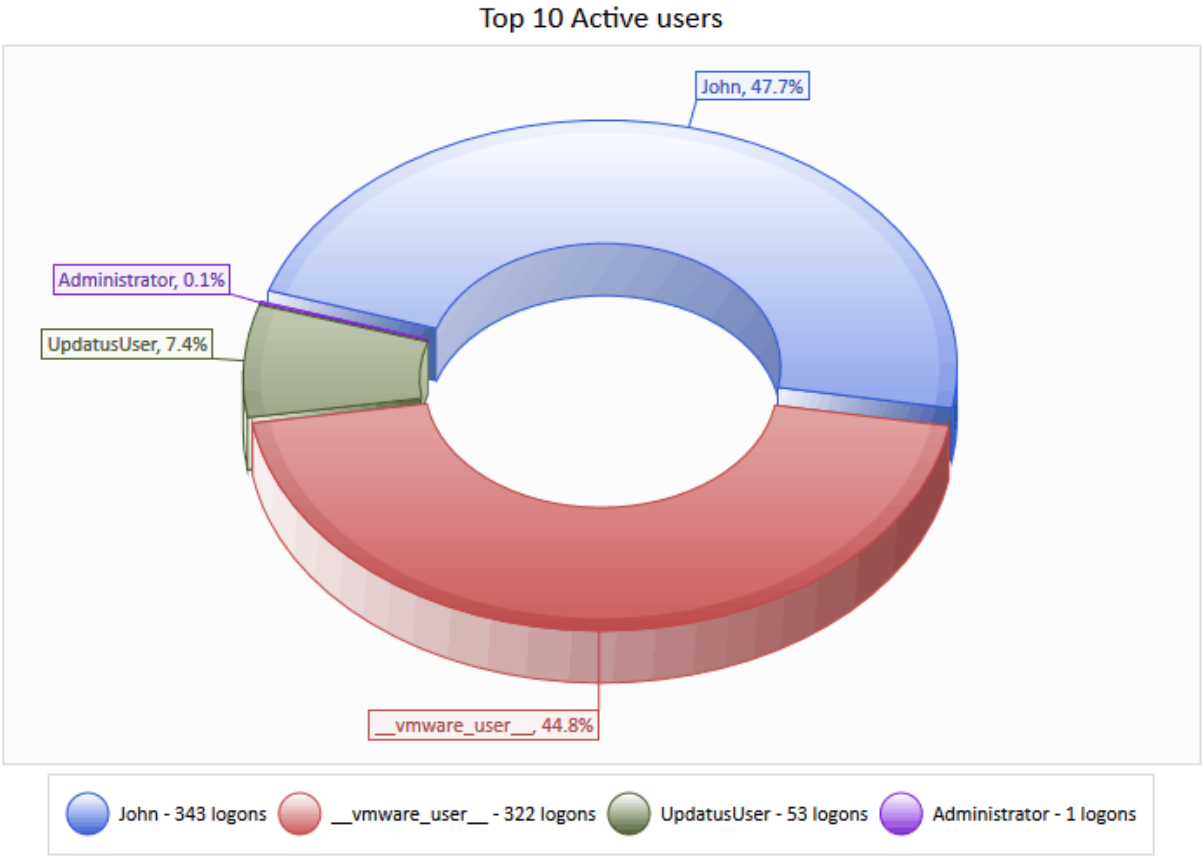
- **CPU speed** - password recovery speed comparison (for brute-force attack).
- **GPU speed** - shows and compares password recovery speed for your GPU device. You can benchmark your CPU or GPU performance using the Pass-o-meter tool.
- **Cracked users** - displays the number of cracked users. The full list of cracked user accounts can be saved to text file additionally.
- **Cracked users and passwords** - displays the list of cracked accounts with passwords.



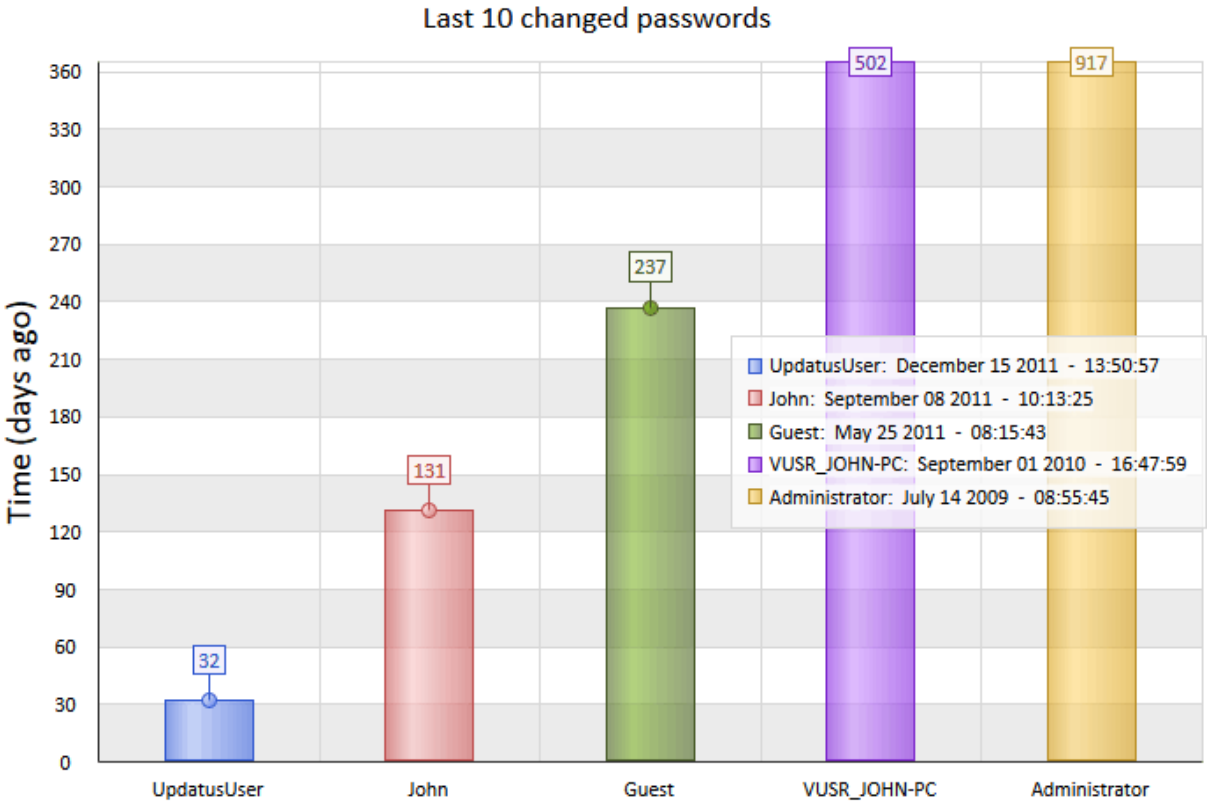
2.5.4 Account statistics

Account statistic are available for both local and domain accounts. To generate a report, first select the data source: local or external database, SAM or Active Directory. These are the reports available in this category:

- **Regular vs. disabled accounts.** This report shows the ratio of regular vs. disabled user accounts.
- **Regular vs. locked accounts.** Ratio of regular vs. blocked\locked accounts.
- **With/without password.** Shows the number of accounts with blank and set passwords.
- **User vs. machine accounts.** Ratio of user vs. system accounts.
- **Active vs. expired passwords.** Report with statistics on accounts with active vs. expired passwords.
- **Regular vs never expired passwords** - compares regular user accounts against those with 'Password never expires' flag or unlimited password live date set.
- **Administrators vs. limited users.** This report gives comparative statistics on accounts with administrative rights vs. restricted user accounts.
- **Account types** shows how much machine, user, administrator, etc. accounts.
- **Account status** displays active against disabled accounts. The same as the first report in the list but contains no additional pane on disabled accounts.
- **Top 10 active users.** Report on top 10 most active OS users. The statistics is gathered from the system's internal user logon counter.
- **Bad password logons.** Top 10 users with the highest rates in the failed logon counter.



- **Last 10 failed logons** - dispays the list of user accounts last tried to logged on unsuccessfully.
- **Last 10 changed passwords** - shows the time of last 10 users who changed their passwords.
- **Last 10 logons** - displays the time of last 10 users who logged on the system successfully.
- **Last 10 logoffs** - the time at which the last 10 accounts logged off.
- **Expired soon accounts** - user accounts that will expire soon.
- **Logon activity** groups users by time passed since last logon to system.
- **Password age** groups users by time passed since last password set/change.



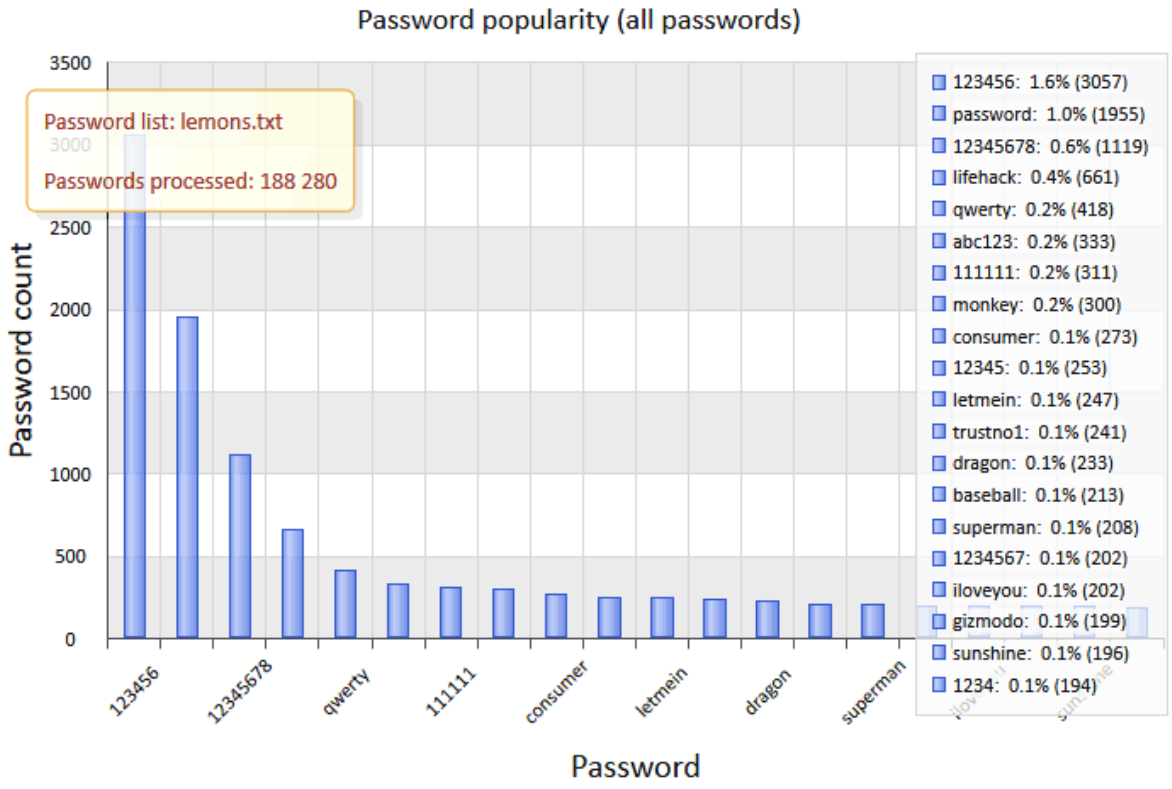
2.5.5 Password-list analysis

Password-list reports display various statistics and perform a deep analysis for input wordlists. As a source wordlists you can use, for example, the list of passwords recovered by the program. You can generate reports for all words of the input list as well as for passwords with a certain length only. The following reports are available here:

- **Password length distribution** - displays the overall length of the password in a given wordlist.
- **Password uniqueness** - this report shows unique against identical passwords chart.
- **Password popularity** - displays the most popular passwords and their percentage of the total number of passwords.
- **Password format** - statistics on the 20 most popular formats. The password format is defined by a character mask. For example, the DDUUUUDD mask corresponds to passwords consisting of two leading and two trailing digits, with four capital letters in the middle. You can save popular password masks into a file so that you can easily use them in a mask-based attack later.
- **Character set exclusivity** - this report displays the number of passwords consisting of one unique character set and the percentage of these passwords to those consisting of several ones.
- **Character set diversity** -the percentage ratio of passwords consisting of one, two, or more character sets.
- **Character sets** - lists all charsets the input passwords are made of.
- **Character set ordering** - the most popular password templates corresponding to the character set order. For example, the *digit-string-special* template includes the following passwords: 123password!@#, 1ove*****, and 12monkey^, etc.
- **Character frequency** - statistics on the frequency of characters in the input words. The 20 most frequent characters are displayed.
- **Unique characters** - the 20 least frequent characters.
- **Frequently used leading characters** - statistics on the most frequent combinations of 1 to 3 characters in the beginning of words.
- **Frequently used trailing characters** - statistics on the most frequent combinations of 1 to 5

characters in the end of words.

- **Frequent combinations** - the 20 most frequently used combinations of 4 to 8 characters.

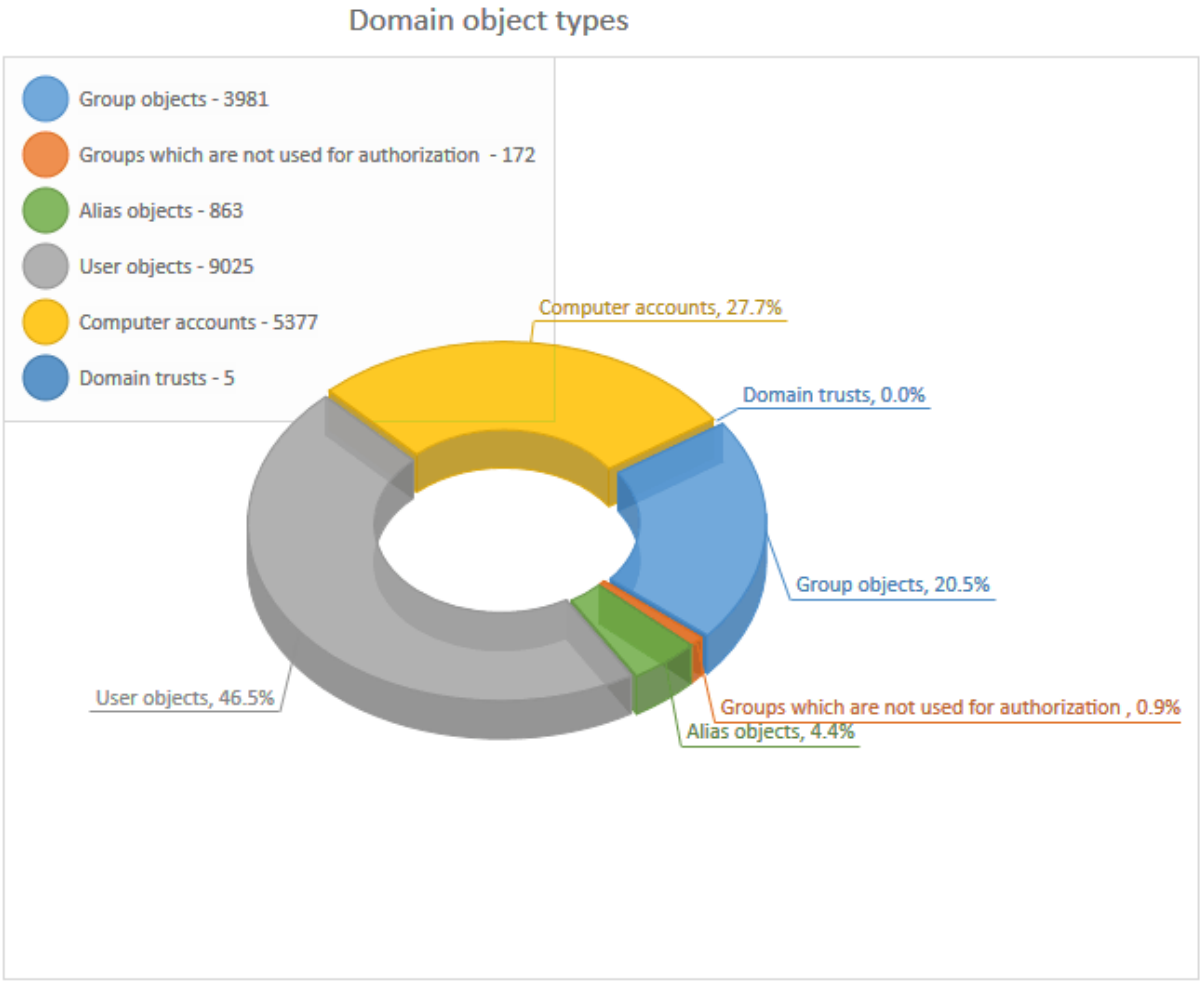


2.5.6 Group information

This section is aimed mainly to analyze various information about Active Directory groups and aliases. Some reports however can be used to display statistics of a local PC by reading information from SAM registry file. The following reports are available here:

- **Last 10 created groups.** 10 recently created group accounts.
- **Last 10 changed groups.** 10 recently changed group accounts.
- **Group types.** This report shows different types group accounts belong to.
- **Most populated groups** - displays top 10 groups with the largest number of users.
- **Sparsely populated groups** - displays top 10 groups with the smallest number of users. Groups without users are not displayed here.
- **Active vs inactive groups.** The program assumes that active groups have at least one member while inactive groups have no users at all.
- **Admin vs non-Admin groups** - shows statistics about Administrator privileges of the groups.
- **Last 10 created aliases.** 10 recently created alias accounts.
- **Last 10 changed aliases.** 10 recently modified alias accounts.
- **Alias types.** This report shows different types alias accounts belong to.
- **Most populated aliases** - displays top 10 aliases with the largest number of users.
- **Sparsely populated aliases** - displays top 10 aliases with the smallest number of members. Aliases without users are not displayed.
- **Active vs inactive aliases.** The program assumes that active aliases have at least one user while inactive aliases have no members at all.
- **Admin vs non-Admin aliases** - shows how many aliases have Administrator privileges.
- **Domain object types** - shows information about all found objects in a domain. For example: users,

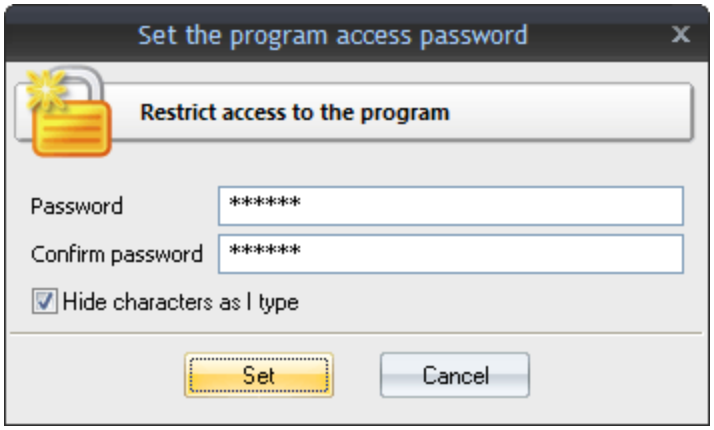
groups, computer accounts, domain trusts, etc.



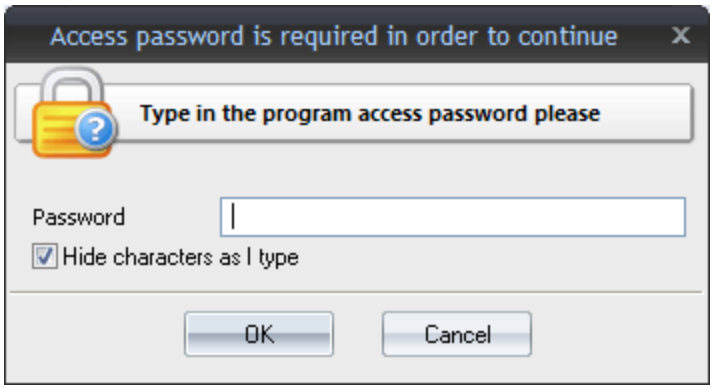
2.6 Tools menu

The Tools consists of two parts: tools for controlling access to the application and tools for working with passwords.

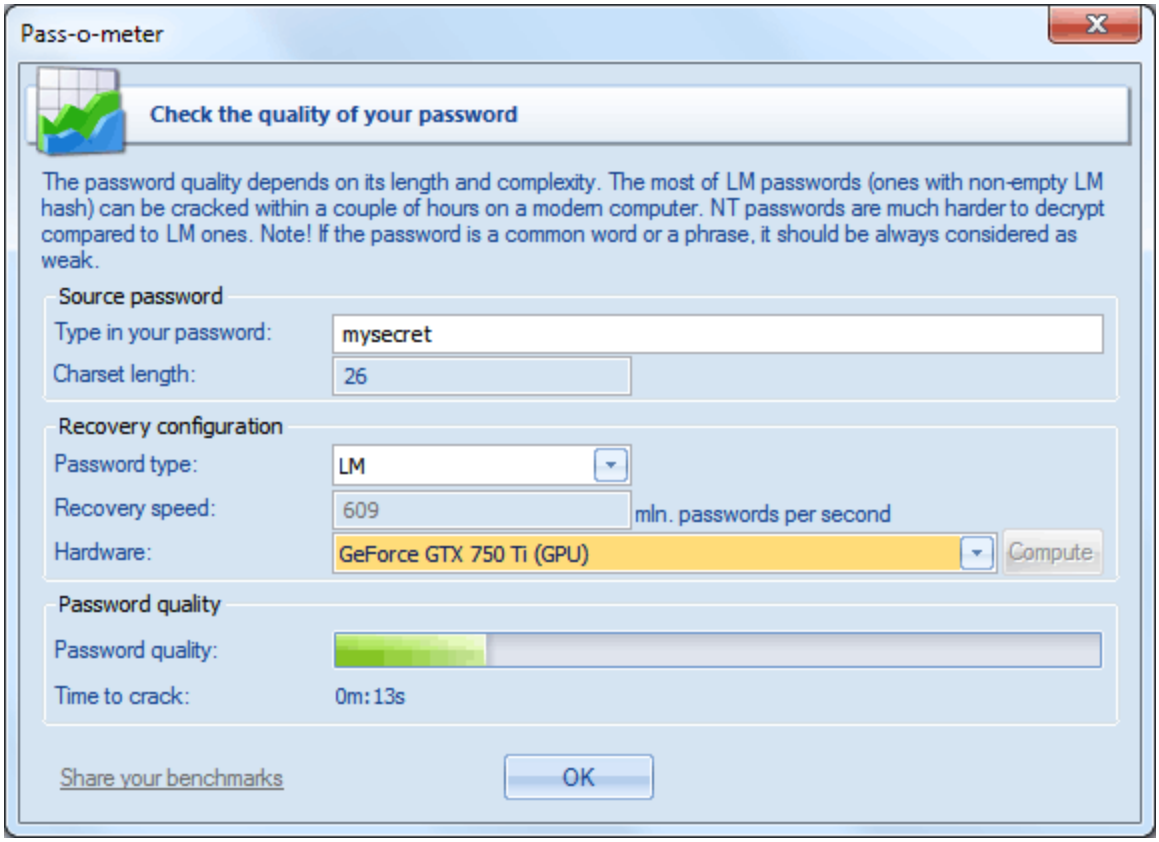
2.6.1 Program access



If anyone besides you can access your computer or account, you can password-protect the application. In this case, when starting the program, user will be prompted for the password, and the application will fail to continue unless the valid password is supplied.



2.6.2 Pass-o-meter



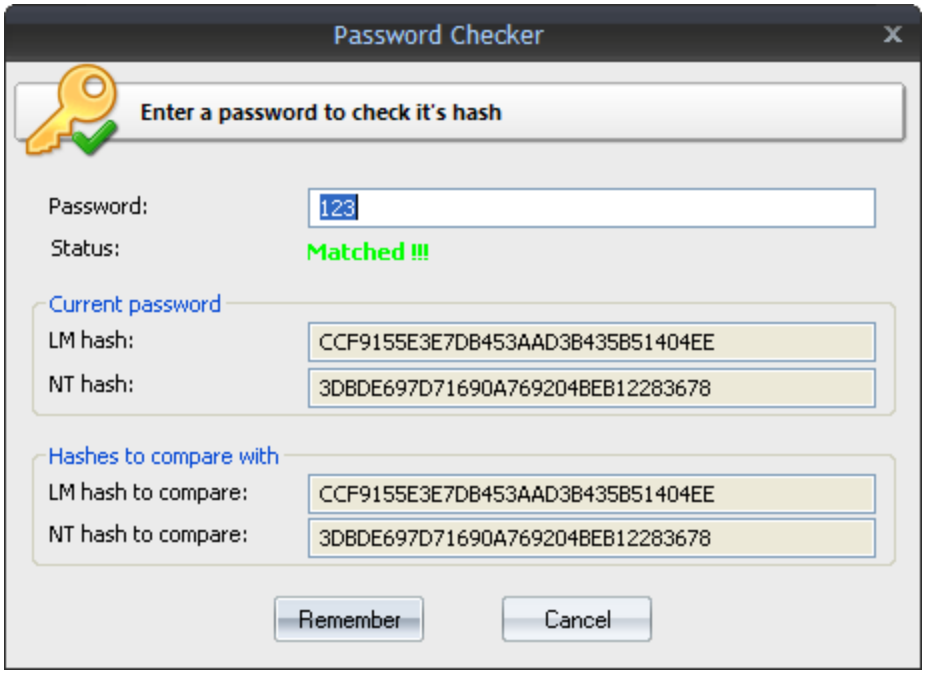
A tool for measuring password strength. During its first start, the program asks you to test your computer's performance. To check the quality of a password:

- Enter the password in the corresponding field.
- Select the hash type: LM or NT. Please remember that beginning with Windows Vista operating systems store passwords as NT hashes by default.
- Select the computer type. *'This computer'* indicates your computer's search speed.
- If you want to test the speed of your GPU device, select '*This computer (GPU)'* from '*Hardware*' combo box and click '*Compute'* button. Note, that you can do it from *Reports* menu as well.

The quality of your password, along with the time that would take your computer with the selected configuration to break it will be shown at the bottom. For example, breaking any LM hash of an alphanumeric password would take about 10 minutes on a modern CPU (at the search speed of over 100 mln. passwords per second). The search speed on a GPU can raise by another order of magnitude.

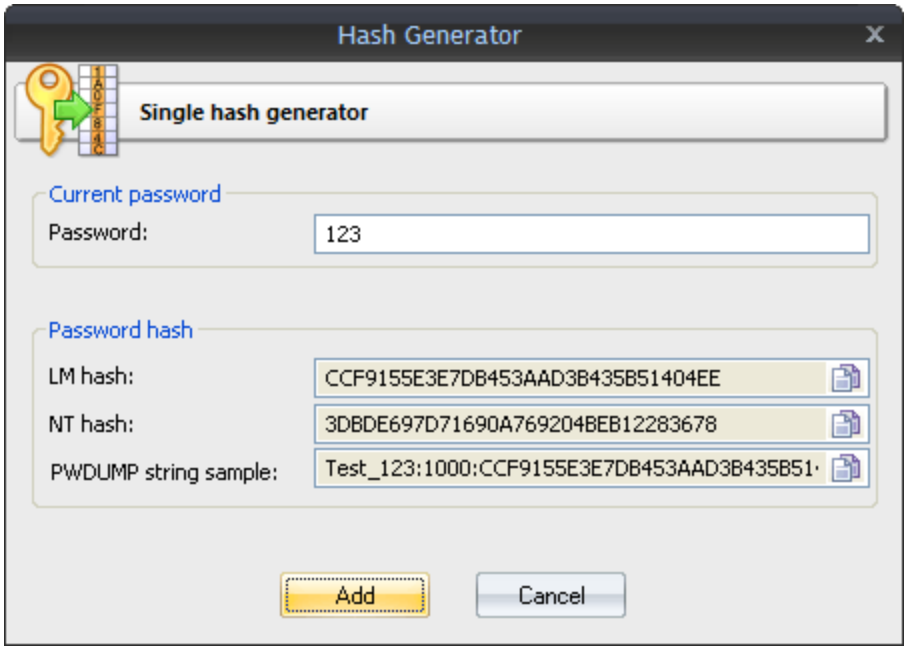
We would be grateful if you let us know the speed you've reached on your PC.

2.6.3 Password Checker

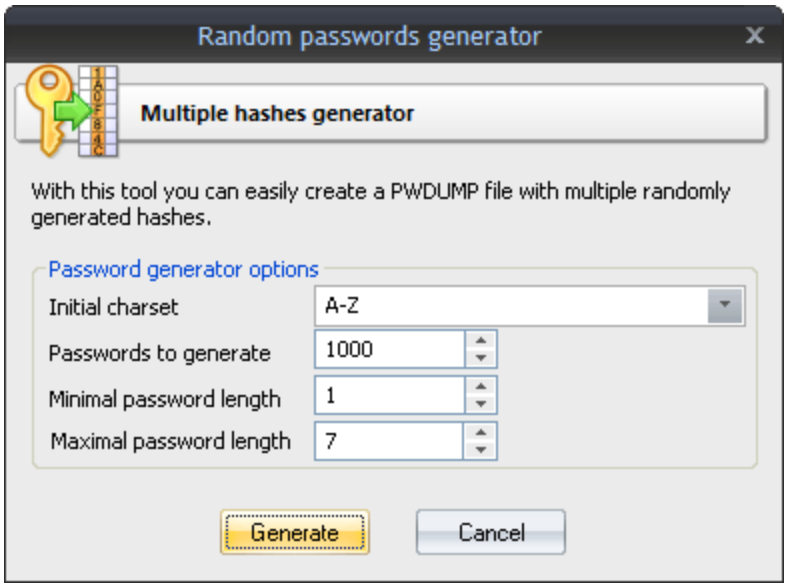


This tool allows checking the password of a selected hash manually. The tool is often necessary for validating certain hashes. For example, when an LM hash, for one or the other reason, doesn't match the password's NT hash.

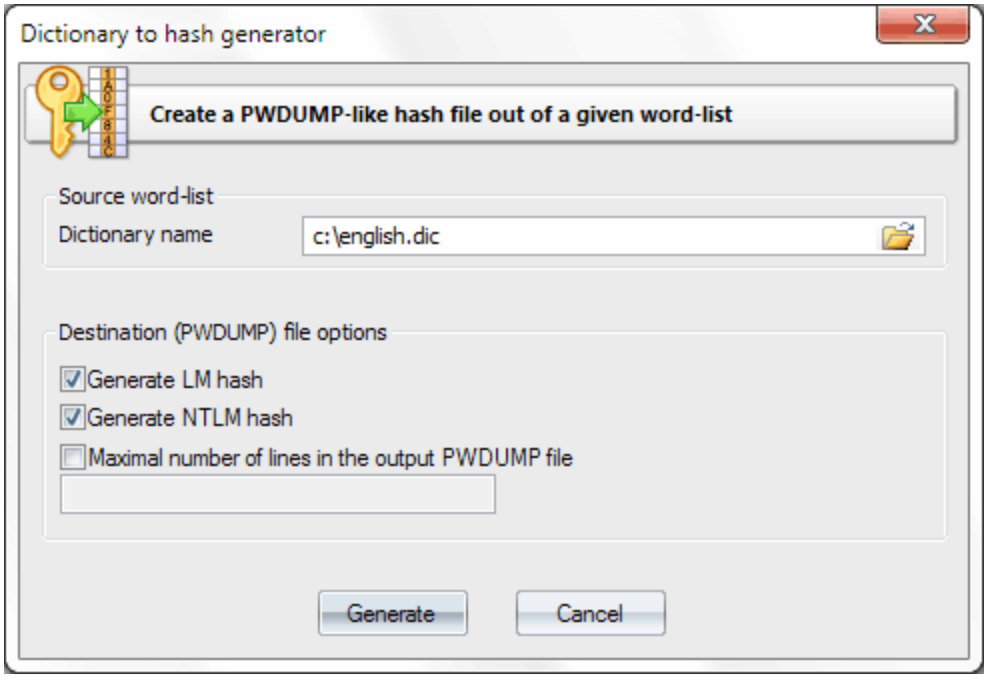
2.6.4 Hash Generator



The single-hash generator allows to quickly generate a test entry for a specified passwords and add it to the hash list.



If you want to create a PWDUMP file with a specific number of randomly generated passwords, use the multiple-hash generator. In the new hash dialog, select the minimum and maximum length, character range and the total number of the hashes to be generated.

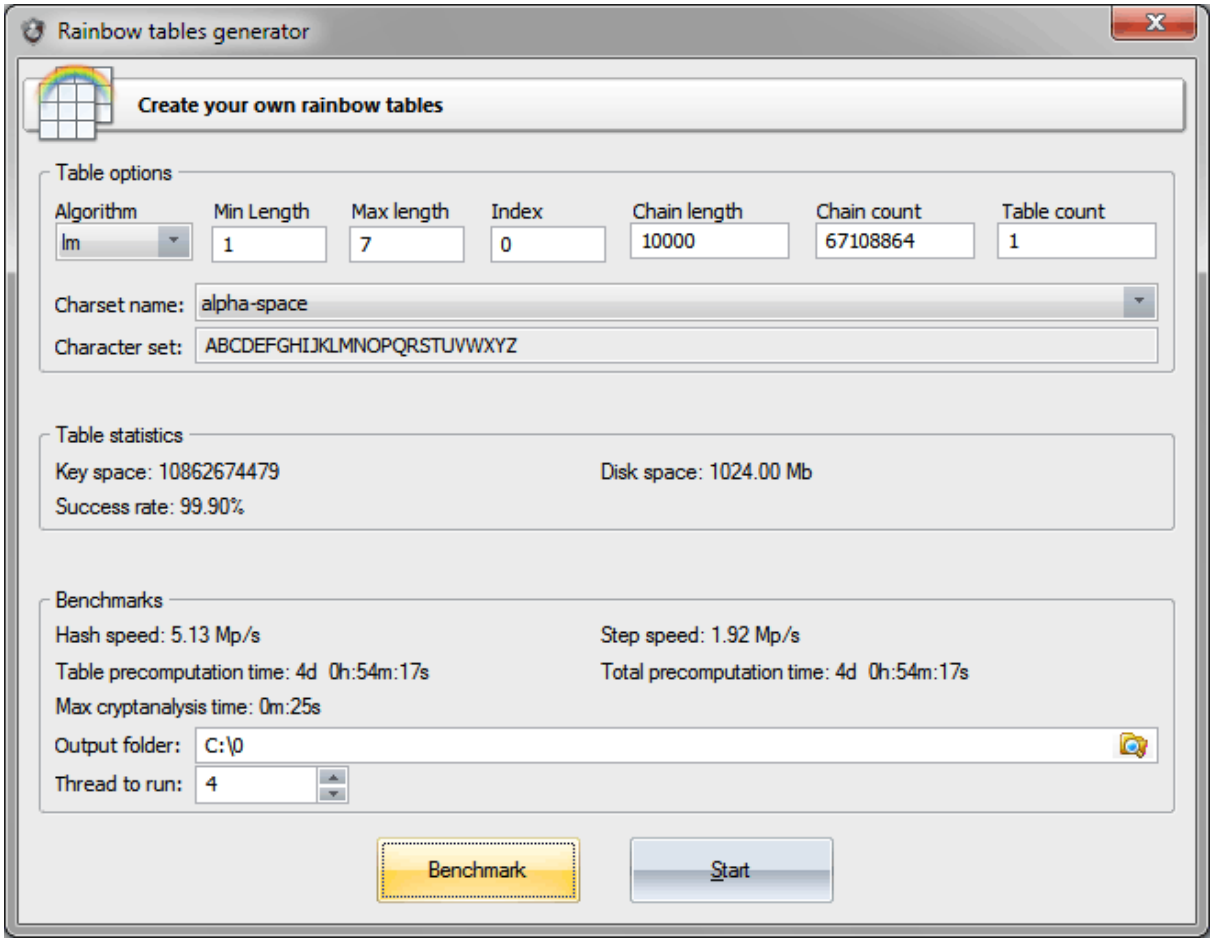


With the dictionary to hash generator you can easily create PWDUMP file out of a given word-list. This tool has a number of additional options here. For example, you can limit the number of output hash items or create PWDUMP file for NTLM hashes only.

2.6.5 Rainbow Tables Generator

Rainbow tables are special search tables used for reversing cryptographic One-Way Functions and cracking plaintext passwords derived from the hash functions. An example of such hashes would be a user password (LM or NTLM hashes) in the Windows OS.

Windows Password Recovery has the password lookup implementation using rainbow tables. The tables it requires can be downloaded off the Internet or created manually with the RT generation tool.



Before you start generating your own tables, it is important to properly configure the respective related options and find their best combination. First, select one of the two algorithms (LM or NTLM) you need and setup a proper **character set** passwords will be limited to. The wider the character range is, the more passwords will be recovered in the rainbow table attack, but the more time it will take to precompute the tables and, perhaps, of greater size they will be.

Rainbow tables are used to recover passwords up to a certain length you should setup in the '**Min length**' and '**Max length**' fields. An LM hash in Windows consists of two 7-character halves; therefore, the maximum password length to be used when generating LM tables must not exceed 7.

'**Chain Length**' affects the following parameters of the table: password recovery rate, table generation time, and time it takes to recover a single password by the attack.

Chain count affects password recovery rate, table generation time, and its size.

Currently, the RT generation tool does not support tables greater than 2 GB in size; however, when creating large tables, you can increase the number of them ('**Table count**' option).

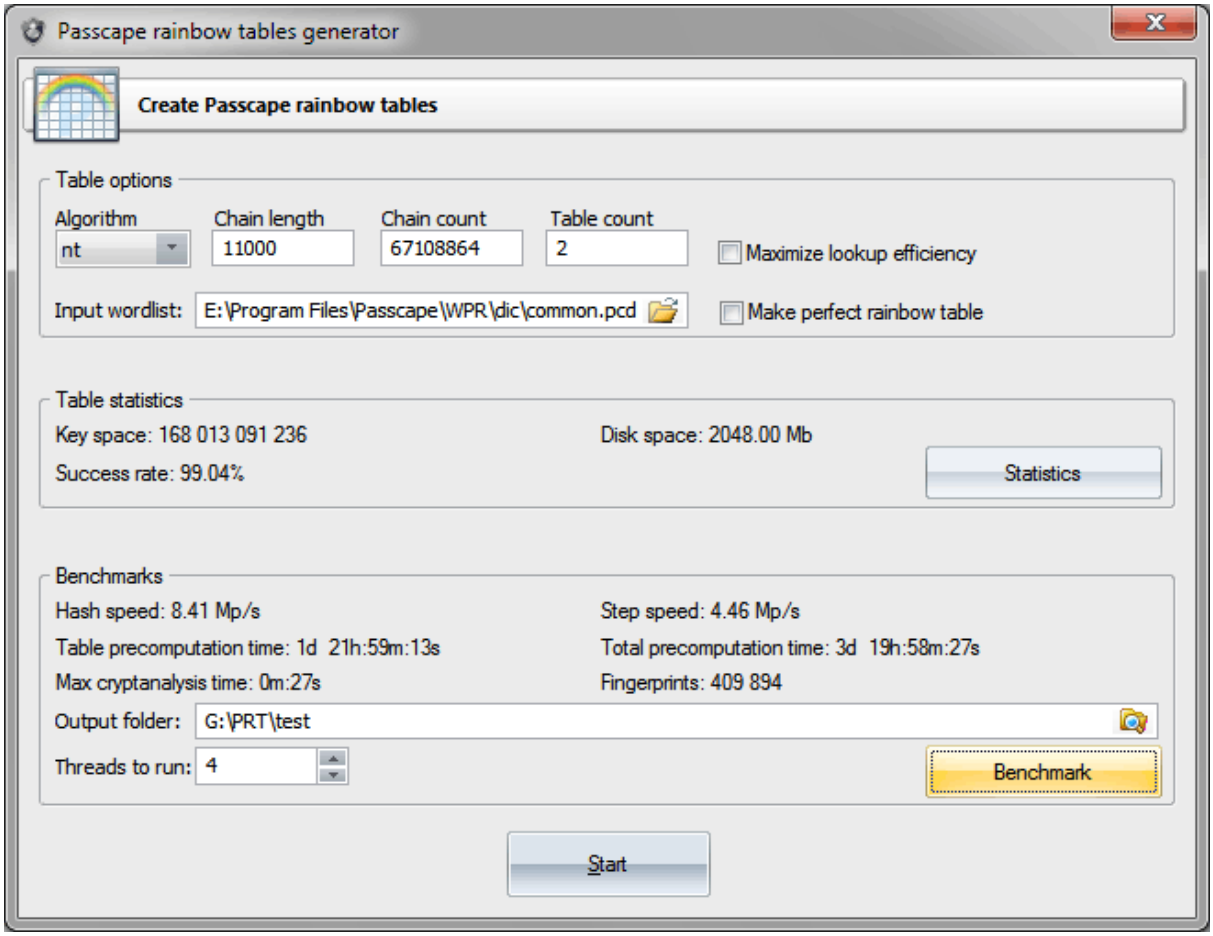
The implementation peculiarity of the rainbow table lookup algorithm is in the fact that the success of the

recovery depends on several parameters, which you need to pick the best ratio for, depending on the size of the tables, the time it takes to generate them and the max time it takes to find a password in the rainbow attack.

The table generation tool supports multithreading, so before launching the precomputation you may want to set an appropriate number of simultaneous threads to be run for creating the tables.

2.6.6 Pascape Rainbow Tables Generator

Passcape Rainbow Tables are used for recovering passwords in Passcape table attack. This tool is intended for creating such tables.



Before you start generating tables, you should set a wordlist that will be used for creating a database of word-prints and specify the table parameters:

- **Chain Length**: affects the probability of finding passwords (e.g. success rate), table generation time and time needed to search for a single password during the attack.
- **Chain Count**: affects the success rate, table generation time and its size.

At the moment, the table generation tool does not support tables of over 2 GB in size. However, you can create several tables if you are working with very large arrays of data (see the '**Table count**' parameter).

Success in recovering a password using the tables depends on several factors, and it's important that you find their best values depending on the size of the tables you work with, their generation time and cryptanalysis time – that is, the time needed for recovering a password during the attack.

Two additional options are used to manipulate table generation efficiency:

- **Maximize password lookup efficiency**: allows you to generate more wordprints from the source wordlist by adding numbers, keyboard and frequently used combinations. This option works well with small wordlists.
- **Make a perfect rainbow table**: as you may know, password chains in rainbow tables can merge. It means that there is a waste of information, time and disk space. This option allows you to create the so-called 'perfect tables' with no merged chains. Perfect tables occupy considerably less disk space and make password recovery a bit faster. However, the payoff for these advantages is a lower success rate in password recovery. To compensate for this lower success rate, you should at least double the number of password chains and increase the number of generated tables.

The table generation tool supports multi-threading, so make sure to set the necessary number of concurrent threads to be run by the program prior to starting the process.

2.6.7 Wordlist tools

Rather a scant number of acceptable tools for working with specialized password dictionaries has inspired the developers of this software to create their own toolkit. With this toolkit, you can easily create new and edit existing wordlists, as well as use them with any password recovery applications.

2.6.7.1 Create new wordlist by indexing files

This tool is designed for creating a new wordlist by selecting (indexing) words from local files on your computer. For example, those could be *.html, *.xml, *.txt, *.doc files, as well as *.mdb, *.pdf, *.exe files, etc.

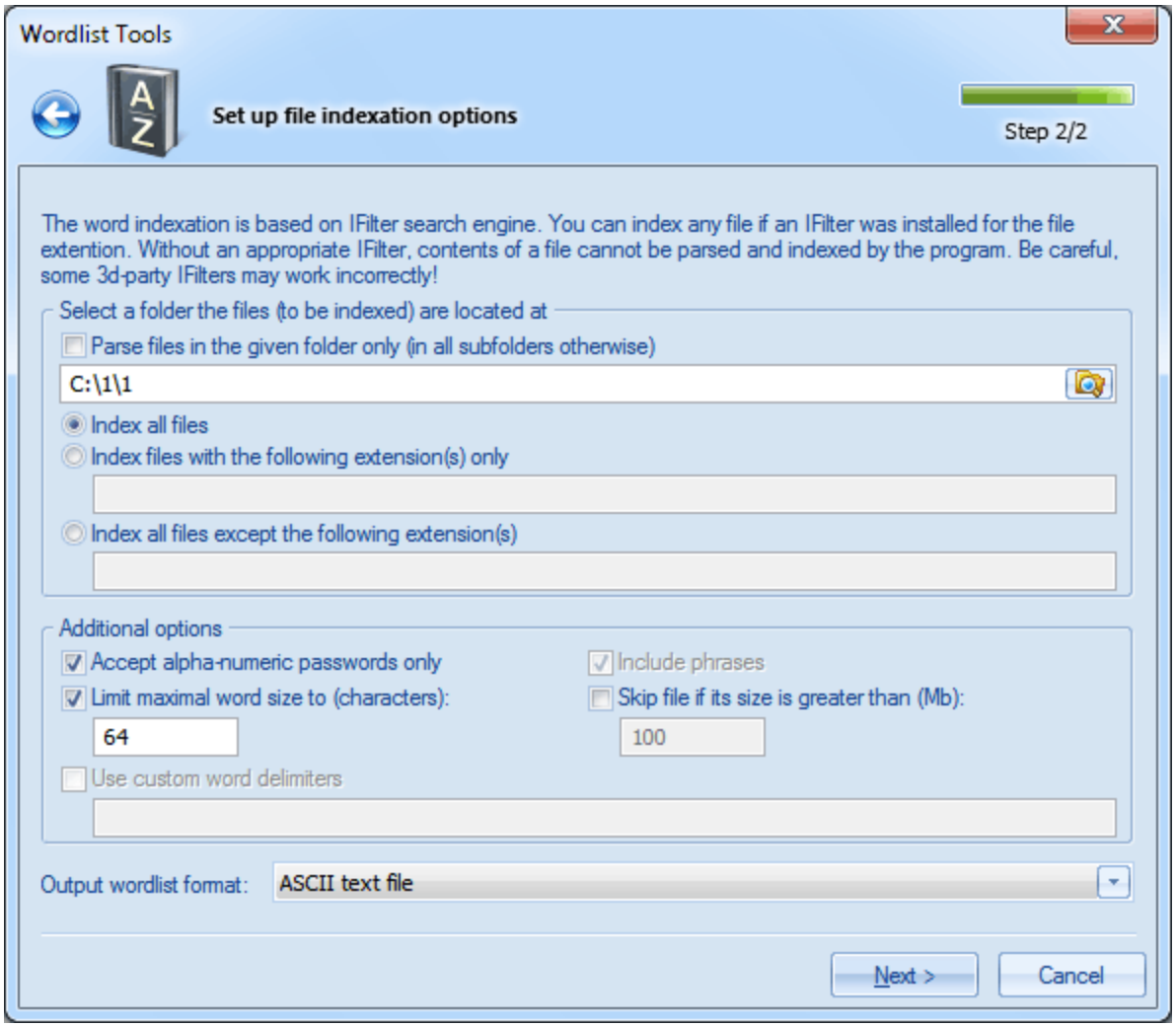
The indexing is based on the **IFilter** technology, which you can read about in Wikipedia. The idea of the technology, developed by Microsoft, comes down to the possibility of indexing the text of any file, which an appropriate IFilter plugin is installed for. This way, you could access the text contained, for example, inside *.exe or *.dll files, e-mail client's database, etc.

Despite the fact that numerous IFIlter plugins, both commercial and free, can be found on the Internet, Windows Password Recovery has internal support for the following types of files:

- Archives: *.zip, *,cab, *.rar
- Programs: *.exe, *.dll, *,cpl, *.ocx, *.sys, *.scr, *.drv
- Text: *.txt, *.dic
- Internet: *.html, *.htm

In other words, files with these extensions can be parsed by the program even without a single IFilter installed on the computer.

Windows 7 has an internal Windows Desktop Search tool, which has a wide range of filters for supporting the majority of popular documents. Under other operating systems, Windows Desktop Search can be installed manually; the setup file can be downloaded from the official website of Microsoft.



The configuration options for this tool consist of two groups. In the first group, you set path to the initial folder, where you need to index the files, and select a file parsing method, namely:

- Parse files in the specified folder only. If this option is not set, the program recursively analyzes all the subfolders and files inside them.
- Index all files
- Index files with certain extensions only
- Index all files except certain extensions

File extensions are to be typed without the dot and to be separated by a comma. Example: txt,dic,xml,chm,htm

The additional options group allows to customize file parsing methods, namely:

- Accept alpha-numeric passwords only. If set, this option will skip all special characters. Only alpha-numeric passwords will be processed.
- Include phrases. This option also allows putting phrases into destination wordlist. A phrase is

considered as a string of characters (of up to 256 symbols) with at least one space character in it.

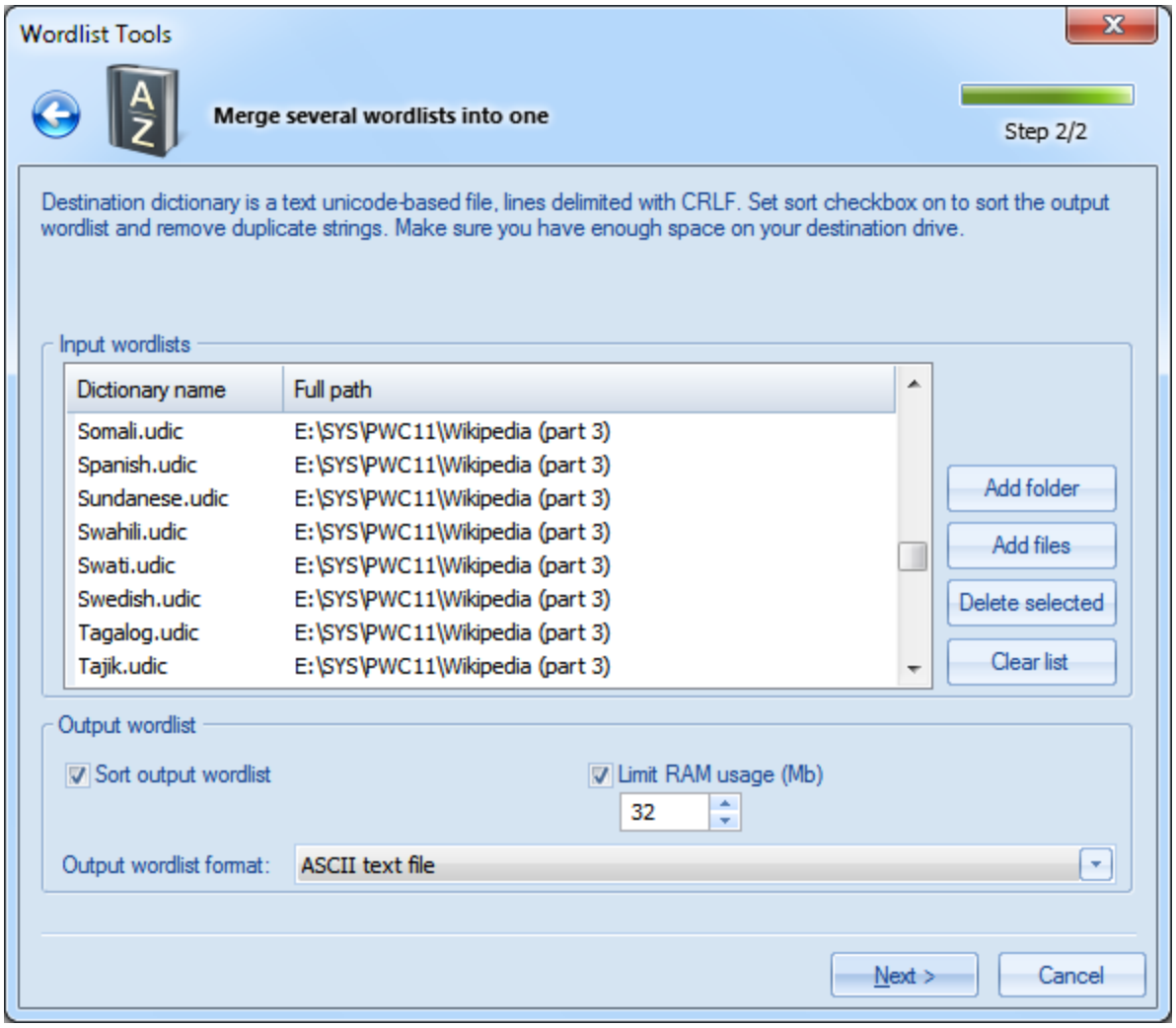
- Limit maximum word size. It is recommended to always set this option. The best maximum word length in a wordlist is 16-64 characters. Cutting the maximum length sometimes radically speeds up the file parsing process. It wouldn't be worthless to remind that the maximum allowed password length in Windows is 128 characters.
- Skip files with size greater than specified. Some IFilters take very long to parse large files; that can cause the program to "hang".
- Use custom word delimiters. You can set your own word delimiters for parsing files. For example, you could use characters like: !"#$%&'()*+,-./:;<=>?@{}[]_ and, of course, space.

Clicking the **Next>** button launches the actual indexing, which may take considerable time. For the sake of speeding up the process, the list of words found during the indexing is created and maintained in the computer memory; that requires significant resources. So, if you get a runtime error of lacking the memory, try decreasing the maximum word length or limiting the number of files being parsed and then try running it over again. Once the operation is completed, and the found words are saved to disk, sort them out to get a truly valuable wordlist. Found words are guaranteed to be unique, i.e. they do not contain duplicates.

Be careful though, some third-party filters could fail to run properly and cause the application to "hang", fail or abnormally terminate. For example, some filters for parsing PDF in Windows XP are known to generate errors.

2.6.7.2 Merge wordlists

A wordlist merging tool is used when you need to combine two or more wordlists in one.

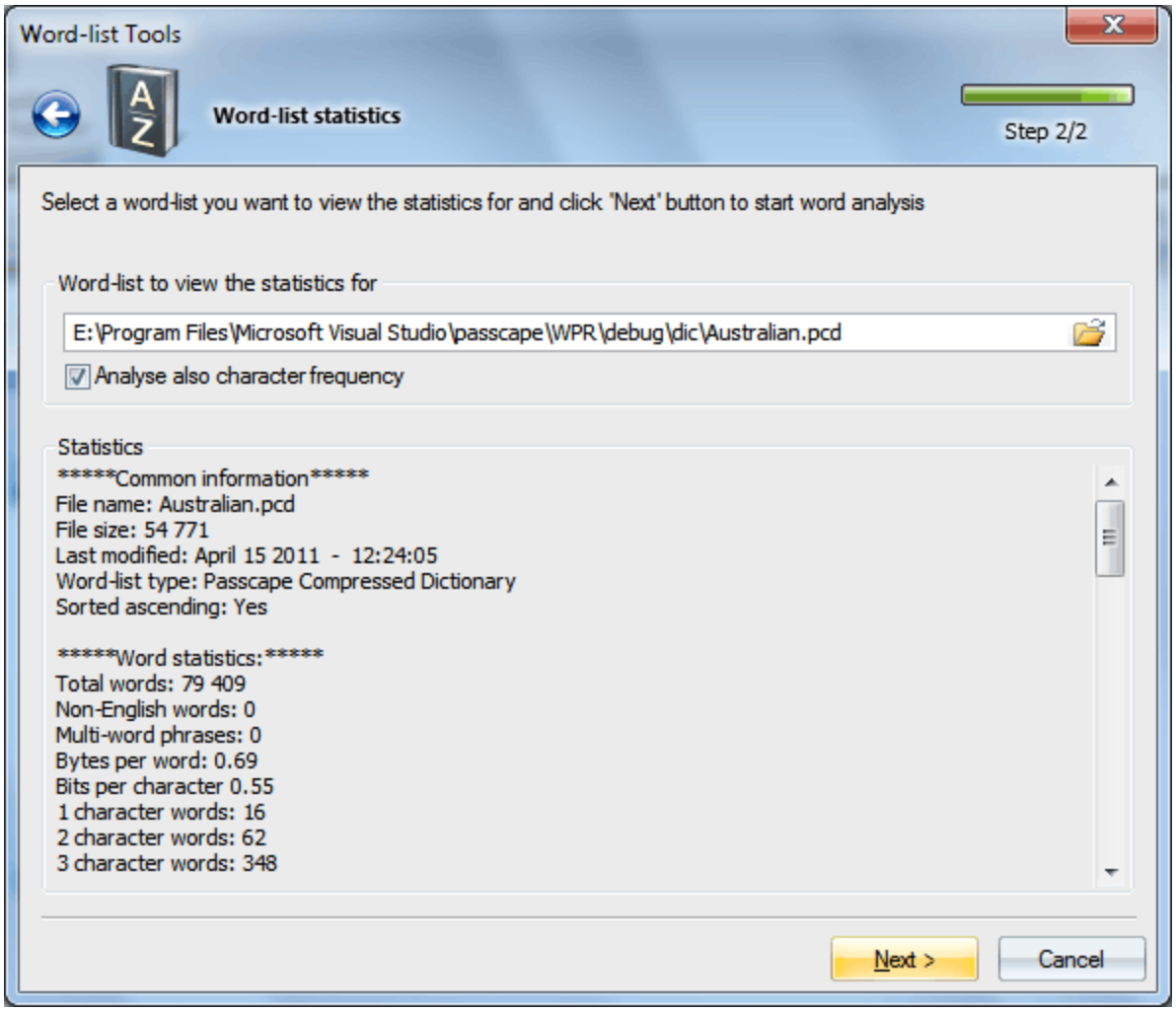


If the '**Sort output word-list**' option is not set, merging comes down to simply adding new words, without sorting or checking for duplicates. In practice, however, more common is merging with sorting; it ensures that all the words in the output wordlist are alphabetically sorted and duplicate-free.

Sorting may take a considerable amount of memory; therefore, it is appropriate to set a limit for the amount of memory that can be used by the process (at the expense of a little downgrade of the operation speed).

2.6.7.3 Wordlist statistics

Wordlist analyzer gathers and shows the following statistics:



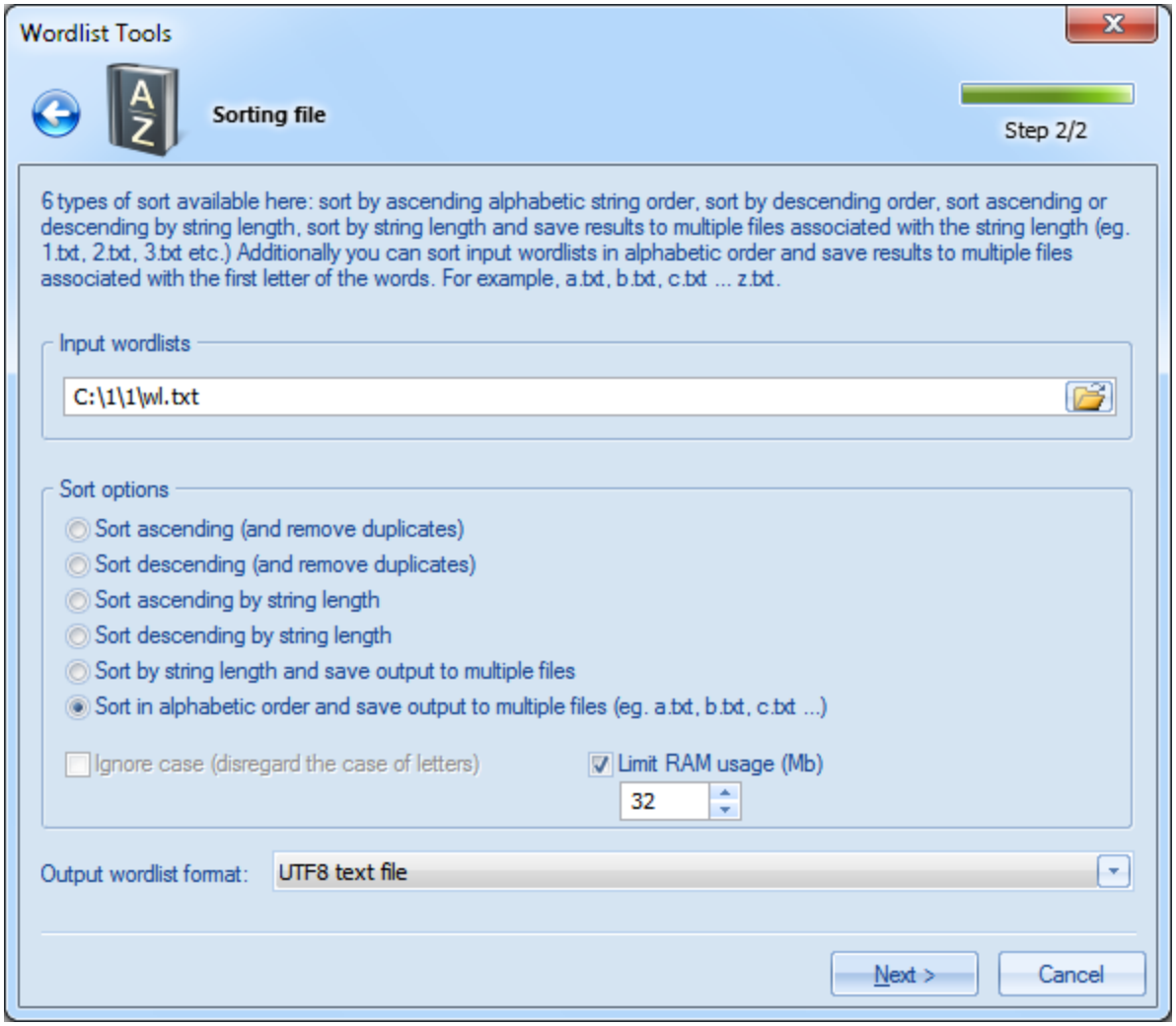
Common information
- Dictionary name
- Size in bytes
- File type
- Last modified date and time
- Whether or not alphabetically sorted (the check takes place only if the file is sorted ascending)

Word statistics
- Total words
- Non-English words
- Multi-word phrases, i.e. words separated with space
- Bytes per word, less word delimiter. Shows average wordlist compression ratio.
- Bits per character. Shows real wordlist compression ratio. For example, in UNICODE the bits per character value tends to 16 (not counting word delimiter), in regular ASCII wordlists - to 8. In certain compressed PCD wordlists one letter can be coded by less than 1 bit (see the screenshot).
- Word statistics - how many words consist of 1, 2, 3, etc. characters.

Character frequency analysis (if the respective option is set)
- Indicates how frequently a certain character appears in a wordlist

2.6.7.4 Sort wordlist

The toolkit offers 6 modes of wordlist sorting; 4 of them are common, and 2 are extended. The common sorting modes include sorting wordlists in the alphabetical order (both ascending and descending) and by word length. When sorting alphabetically or by word length, the program automatically removes word duplicates.



Additionally, you can sort a wordlist by length and save the results in multiple files, associated with word length. For example, file 1.txt would contain 1-character words, 2.txt - two-character, etc.

The sixth sorting mode works similarly. At the same time, the program sorts the source wordlist in the alphabetical order and creates several target wordlists that correspond with the first letter of the word. For example, all words beginning with letter A would be written to file A.txt, words beginning with B - to B.txt, etc. You should keep in mind that certain words may begin with characters that cannot be used in a file name. In this case, the program automatically suggests a replacement by issuing an appropriate warning in the messages window.

If the 'Ignore case' option is set, the sorting is carried out regardless of letter case; i.e., the words *bad*, *Bad* or *BAD* are considered identical, with all the ensuing consequences.

Target wordlist name may be the same as the source; however, that is not recommended.

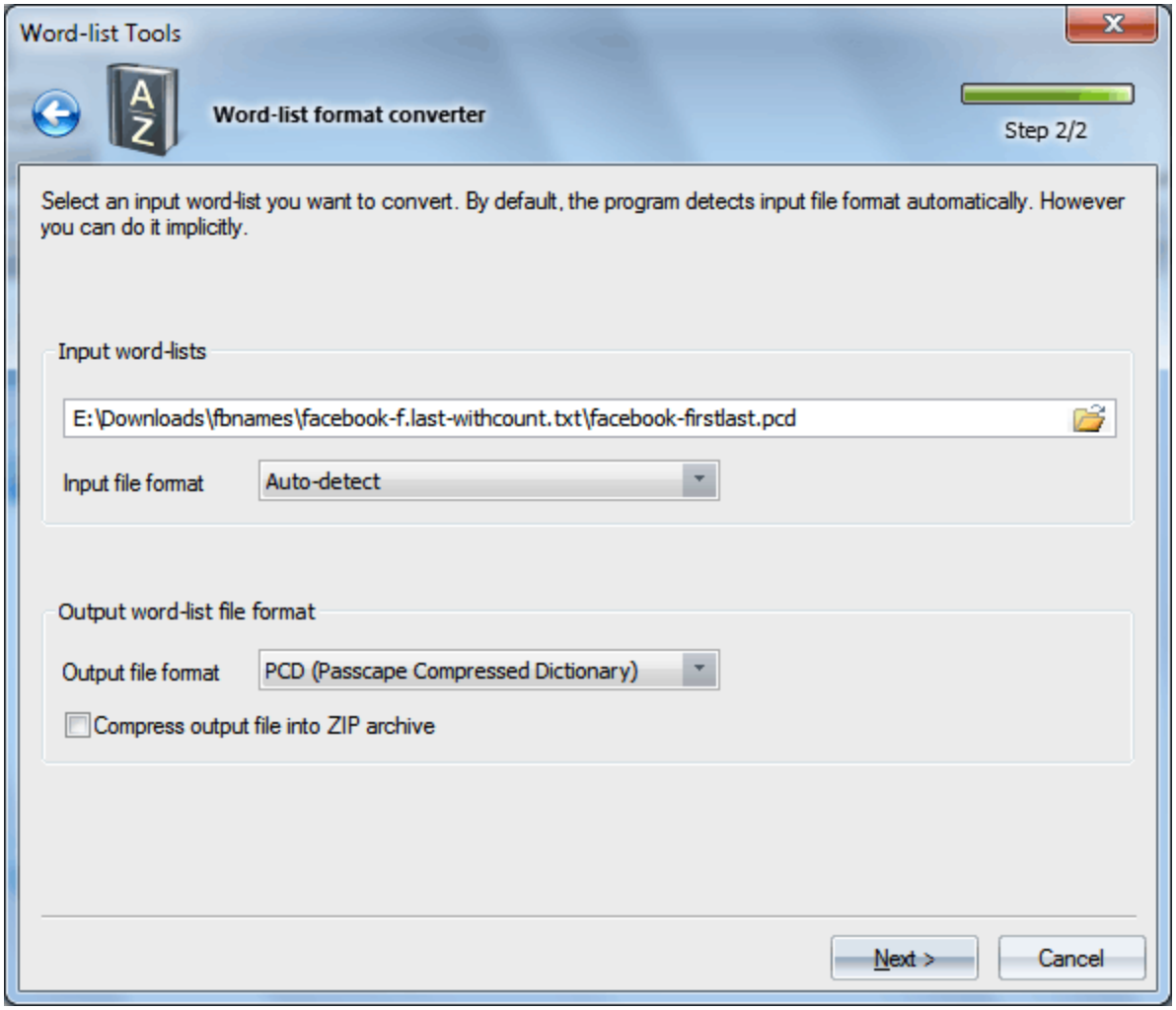
Sorting large files (supports files larger than 4 GB) involves intensive use of RAM; the amount of it can be limited by the respective option. For large files, it is not recommended to set the memory limit less than 16 MB, as that can affect the speed of sorting.

While sorting, the program may create auxiliary files in the application's temporary folder. Make sure that the disk with the temporary folder has enough room for the swap files.

2.6.7.5 Convert/compress wordlist

Numerous wordlists that can be found on the Internet are usually represented by three major formats: **ASCII**, **UTF16** (Unicode) and **UTF8**. With this tool, you can convert a wordlist from one format to another and optionally compress wordlists to ZIP files. Besides the three above mentioned formats, the program supports its own format **PCD** (Passcape Compressed Dictionary), which, in the majority of cases, gives a greater gain in size even compared to a compressed ZIP archive.

Creating large PCD files may take considerable time!



This tool's user interface is pretty easy. In the upper group, select the source wordlist and its format. By default, the program detects the format of the file automatically, but you can also specify it by hand.

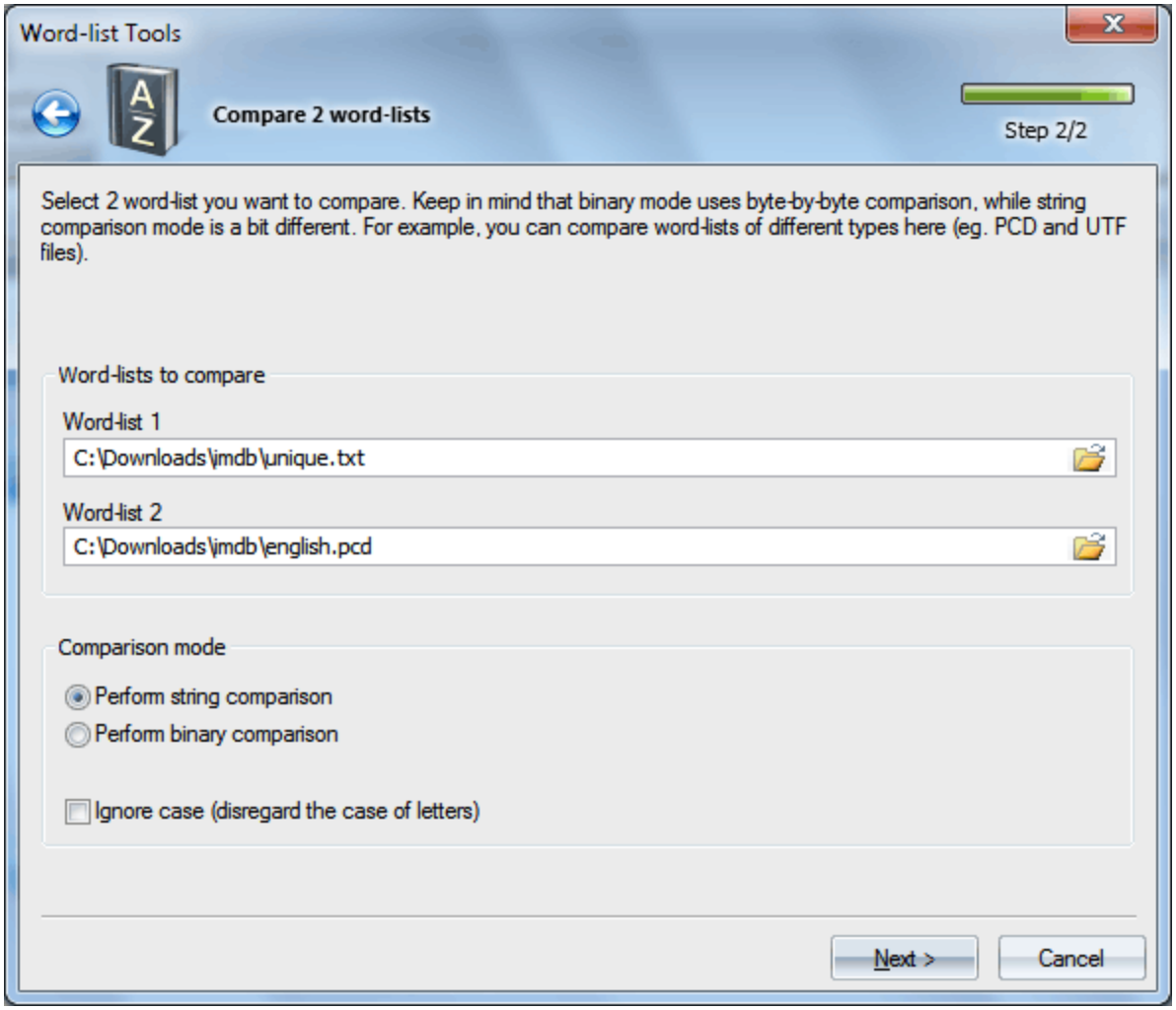
While the format of a PCD can be clearly recognized, with text files it's not that easy. As a rule, text files/wordlists in UTF16 or UTF8 begin with a two- or three-byte marker that describes the type of the file. However, there are Unicode wordlists that do not have any identifying markers. For such "hard" cases, you need to set the type of the source file manually. Otherwise, the program, being unable to see an appropriate identifier, improperly recognizes the file as ASCII.

Target wordlist, similarly, is defined by one of the four above mentioned formats. With the compression option set, the program additionally compresses the file to a ZIP archive.

Target wordlist name may be the same as the source; however, that is not recommended.

2.6.7.6 Compare wordlists

Sometimes, it is necessary to determine whether two wordlists are identical. That is what the wordlist comparison tool for.



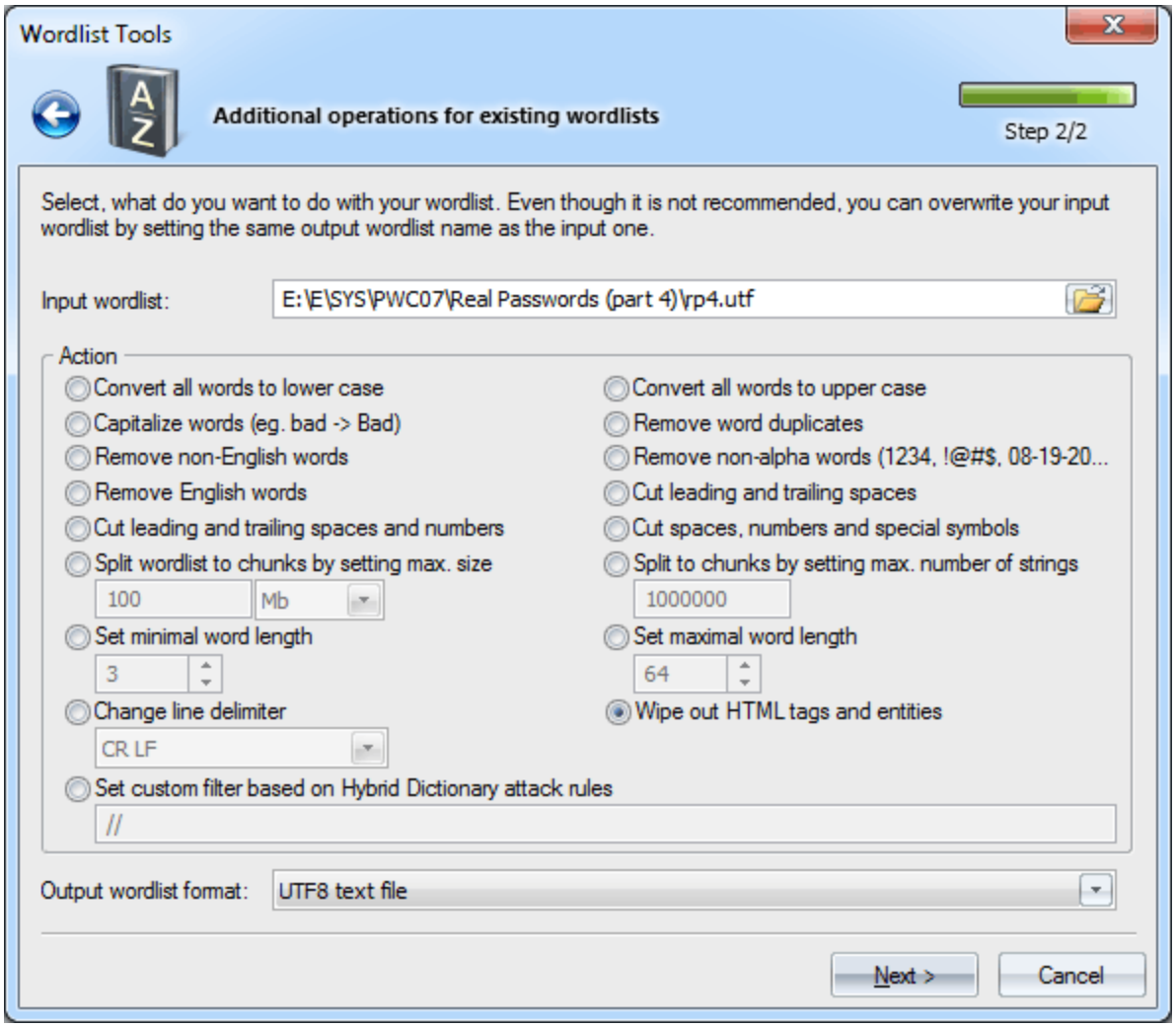
This tool offers two operating modes:
1. Binary comparison, for comparing files by-byte
2. String comparison, which compares words rather than bytes. This mode is noteworthy for its ability to compare wordlists of different formats. For example, PCD and UNICODE, or UNICODE and ASCII.

If the ignore case option is set (string comparison mode only), then, for example, the words *bad* and *Bad*

will be considered identical.

2.6.7.7 Additional operations

The additional tools are designed primarily for editing and tuning up existing wordlists.



The tools include the following operations:

- Convert all words in wordlist to lower case. For example, BAD -> bad.
- Convert all words to upper case. For example, Bad -> BAD.
- Capitalize words - upper-case first letter, lower-case all others. For example, bad -> Bad.
- Remove word duplicates.
- Remove non-English words.
- Remove words that entirely consist of numbers and/or special characters. For example, 12345, !@#$%, 08-19-10, etc.
- Remove English words.
- Cut/remove leading and trailing spaces.
- Cut/remove leading and trailing spaces and numbers.
- Cut/remove leading and trailing spaces, numbers and special characters.
- Split wordlist to chunks by maximum size.
- Split wordlist to chunks by maximum word count.

- Remove words of length smaller than specified.
- Remove words of length greater than specified.
- Change line delimiter.
- Wipe out HTML tags and trash. This menu also converts HTML entities to human-readable form. For example, **&** -> **&**, **@** -> @
- Set your own filter based on Hybrid Dictionary rules

For source wordlist, the program takes ASCII, UTF16, UTF8 and PCD files. Target wordlist can be a text of ASCII, UTF16 or UTF8.

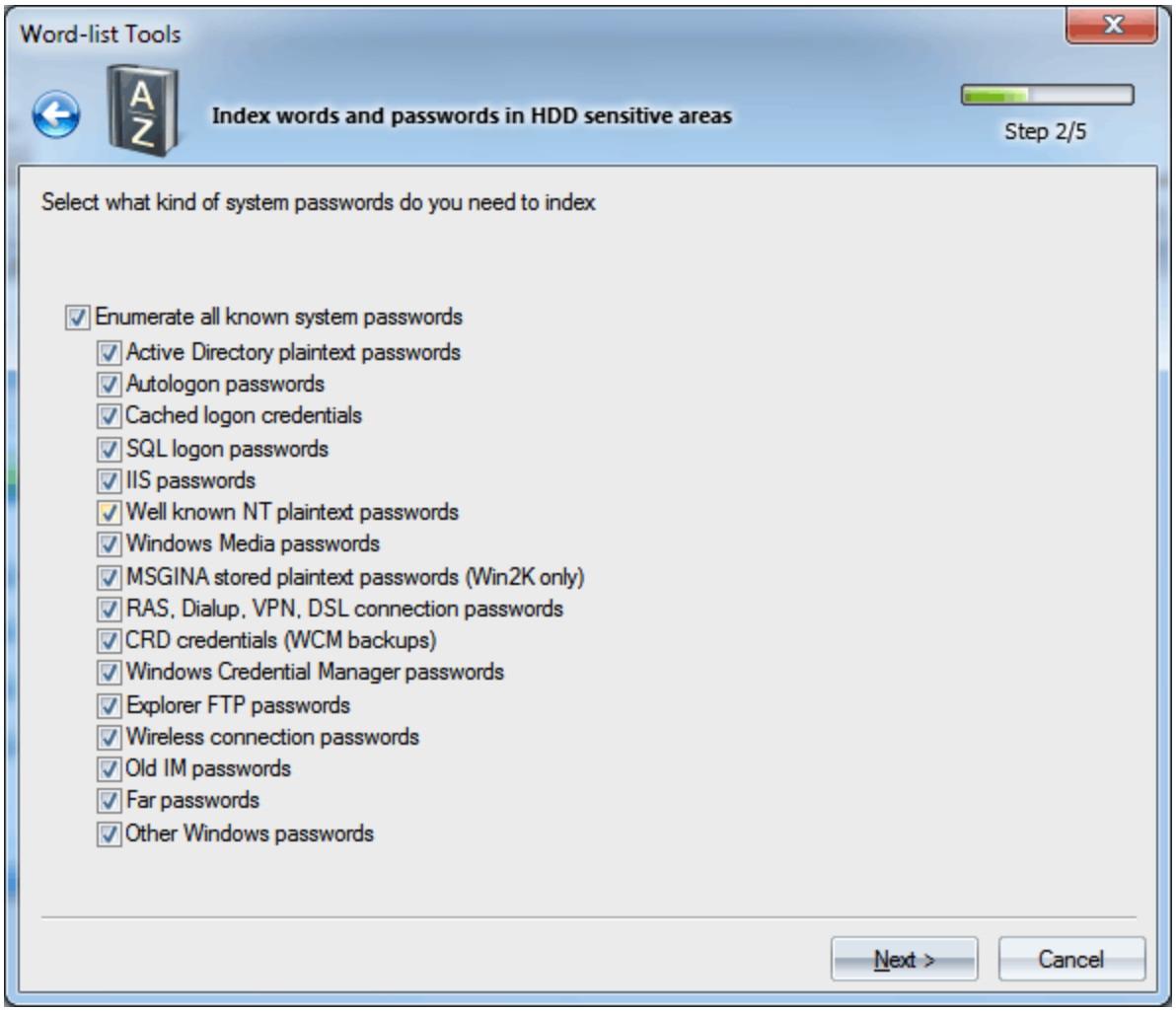
Source and target wordlist name may be identical (not recommended). In this case, the source wordlist will be overwritten.

2.6.7.8 Index HDD sensitive areas

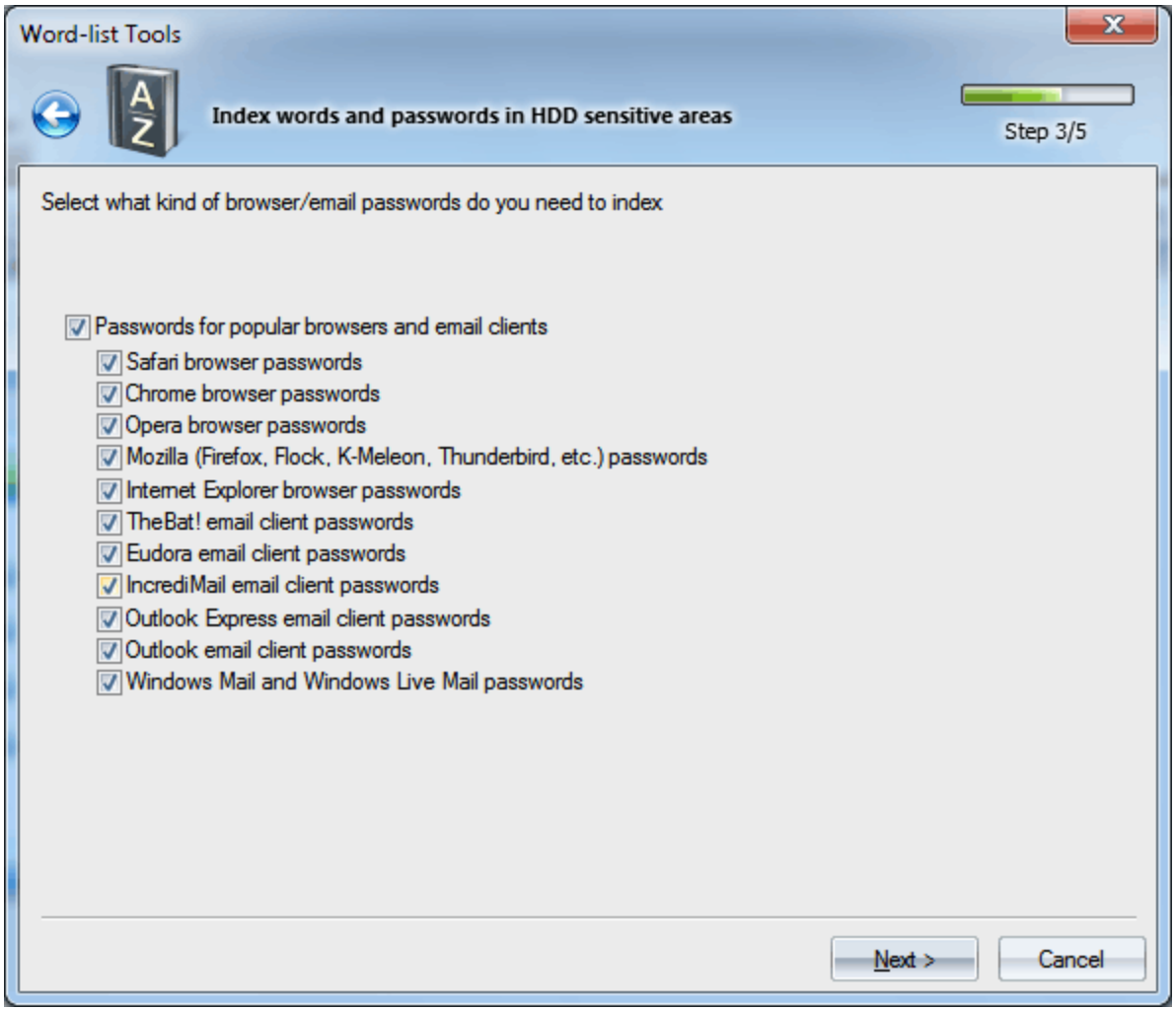
Creating a wordlist by indexing the hard disk (followed by an attack using this wordlist) is a pretty useful and sophisticated tool for decrypting passwords to local Windows accounts.

Often users, instinctively, set same passwords to their Windows accounts, Web, ICQ, etc. The idea of this tool is to create a wordlist of all found formerly used passwords, user's messages, words from recently opened files, etc. and then use the accumulated wordlist for looking up passwords to the local accounts. This technique is engaged in the Artificial Intelligence attack.

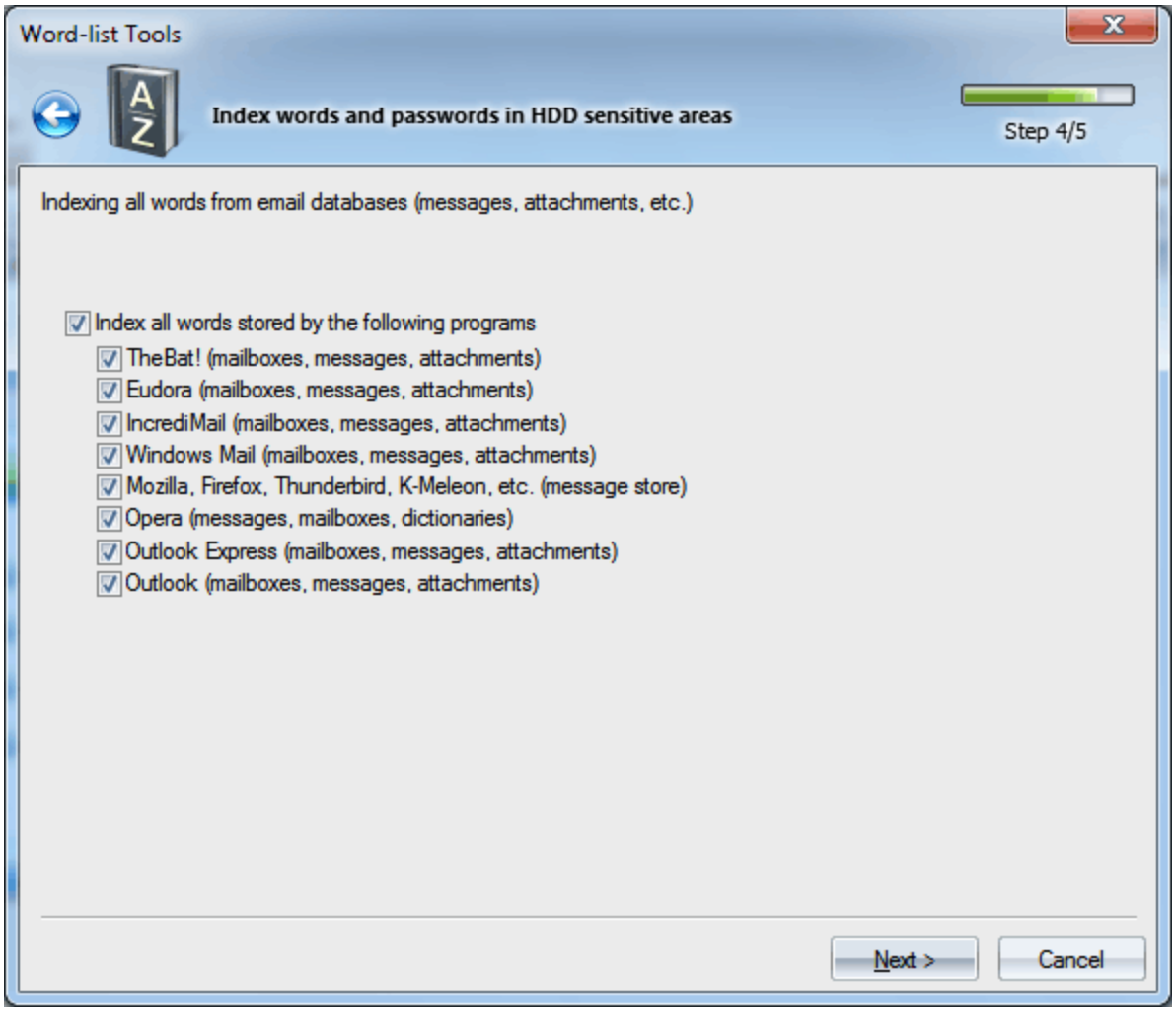
The configuration of the tool conventionally consists of four parts:



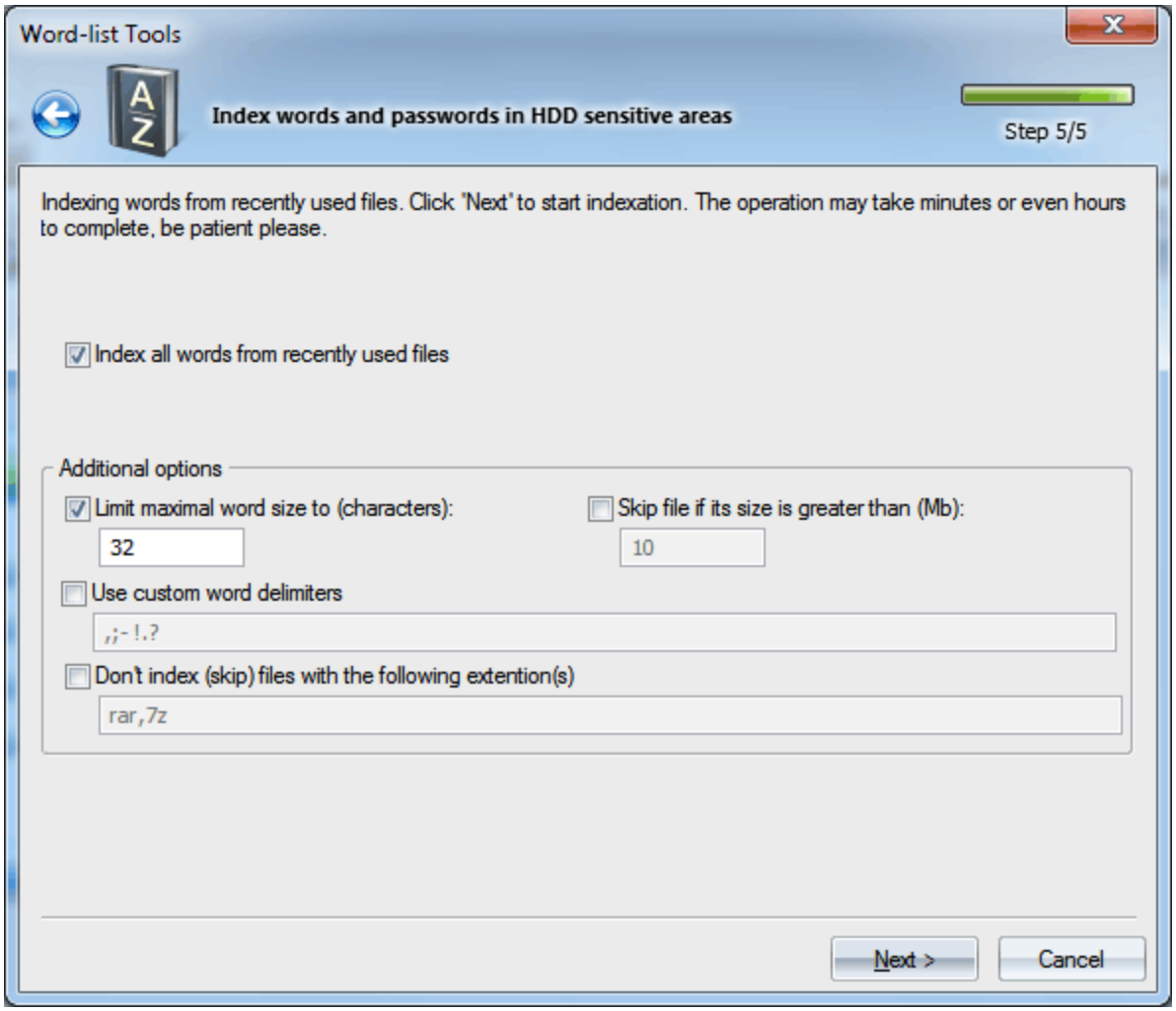
First, select the system modules to be used when generating the wordlist. These modules find and index the following types of passwords on your computer's hard disk: Active Directory plaintext passwords, startup passwords and cached startup passwords, SQL, IIS, Windows Media, Win2K text passwords, RAS, Dialup, VPN, DSL, WEP, WPA, FTP connection passwords, Windows Credential Manager passwords, Instant Messengers, etc. passwords.



In the second part of the configuration, select the browsers and e-mail clients, passwords from which are also to be found and added to the wordlist being created. The program supports the following major web browsers: Safari, Chrome, Opera, Mozilla-based browsers (Firefox, K-Meleon, Flock, etc.), Internet Explorer. E-mail clients are represented by: TheBat!, Eudora, IncrediMail, Outlook Express, Outlook, Windows Mail, and Windows Live Mail.



Besides merely gathering passwords, the program can index user's e-mail communication, scanning all found mailboxes, messages, attachments, etc. The hard disk search is performed for all accounts in a system, so the process may take considerable time, especially when the system hosts many users or when e-mail clients' databases are large. One way or the other, you can enable/disable each module individually.



Finally, in the last dialog, you can set the options for indexing words from all files, recently opened by current user. Available options include:

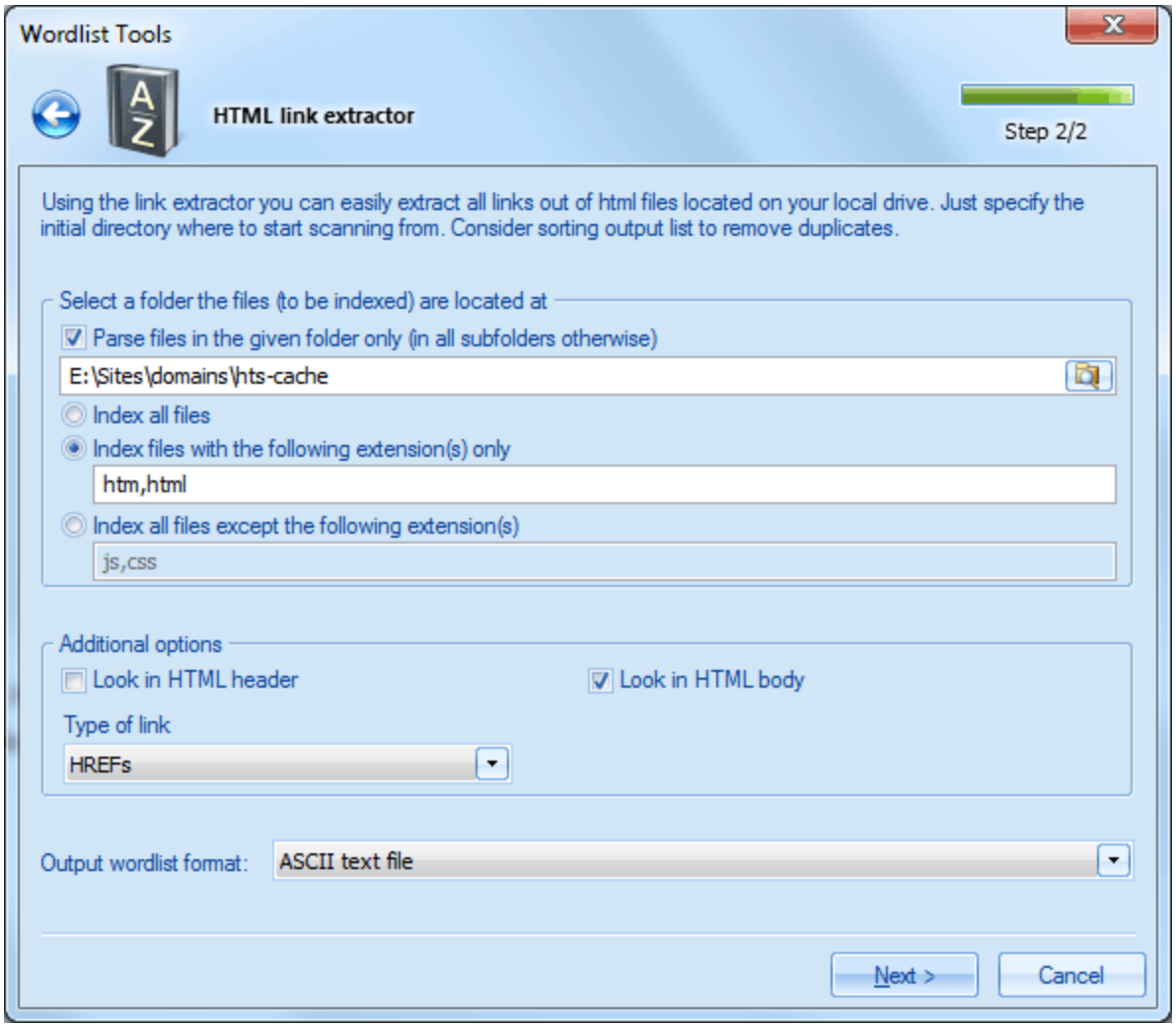
- Set the maximum length of words that can be added to the wordlist. All words with length greater than the specified limit will be skipped.
- Skip files with size greater than specified. The size is specified in MB.
- Use custom word delimiters. By default, word delimiters are all non-alphabetic characters.
- Do not index files with specified extensions. Use this option to skip files that you consider unnecessary.

Clicking the **Next>** button starts the indexing process.

Keep in mind that it can take considerable time!

2.6.7.9 Extract HTML links

This tool is designed for extracting HTML hyperlinks from HTML files.



The configuration options for this tool consist of two groups. In the first group, you should set a path to the initial folder, where the HTML files are located, and select a file parsing method, namely:

- Parse files in the specified folder only. If this option is not set, the program recursively analyzes all the sub-folders and files inside them.
- Index all files
- Index files with certain extensions only
- Index all files except certain extensions

By default, the tool checks *.htm and *.html files only.

The additional options group allows to set the type of links, as well as where to look for them:

- Look in HTML header
- Look in HTML body
- Look links in HREF tag, SRC tag or in both tags.

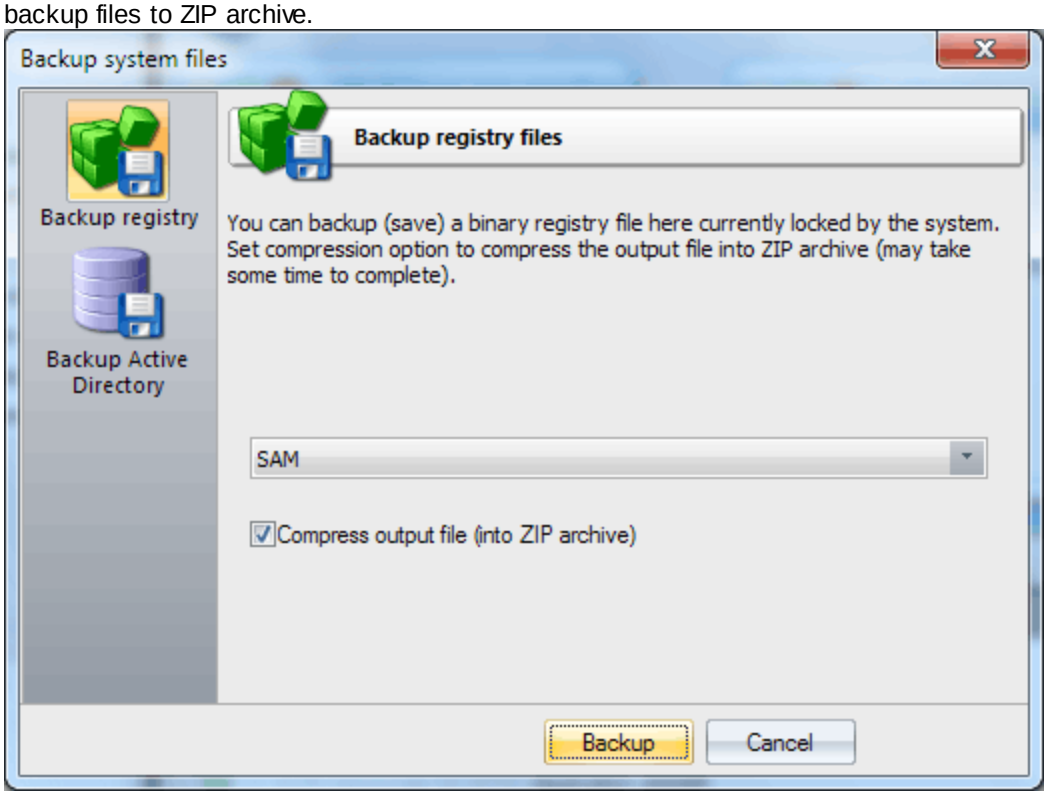
Clicking the **Next>** button launches the search, which may take considerable time. Once the operation is completed, and the found links are saved to disk, consider sort them out to get ride of duplicates.

2.7 Utils menu

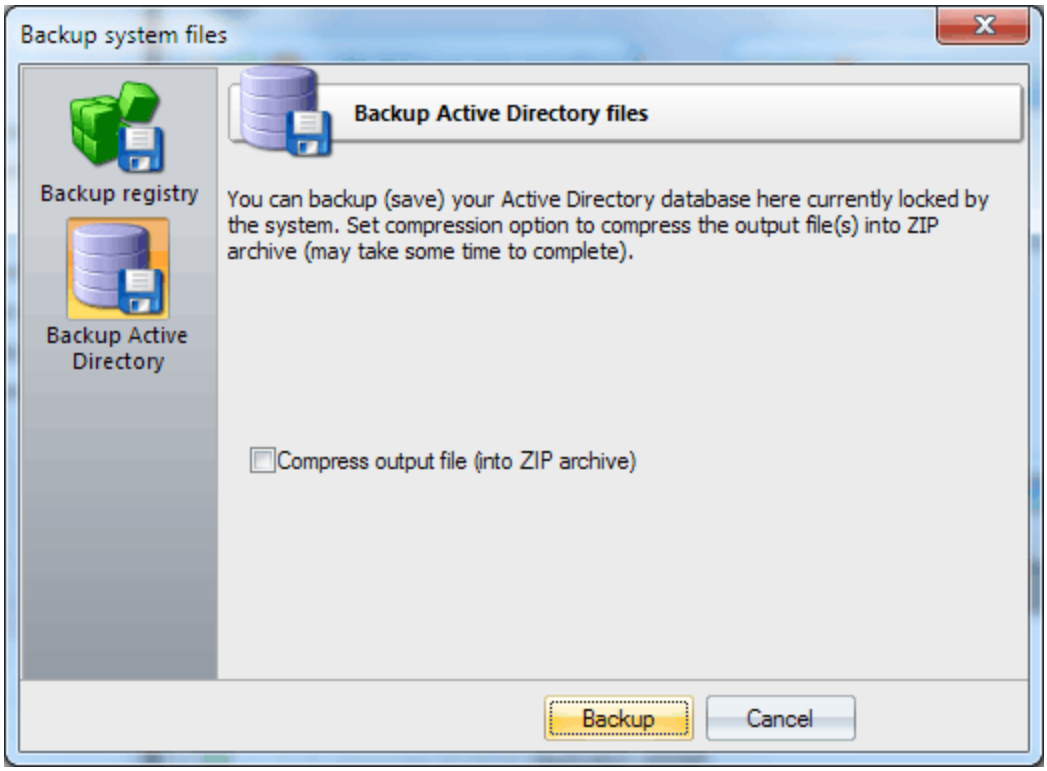
Utilities menu consists of additional addons aimed mainly for advanced users.

2.7.1 Backup system files

Registry backup tool allows easily create a backup copy of your Windows registry. Even if the registry file is locked by operation system. You can set additional option to save extra space and compress the backup files to ZIP archive.

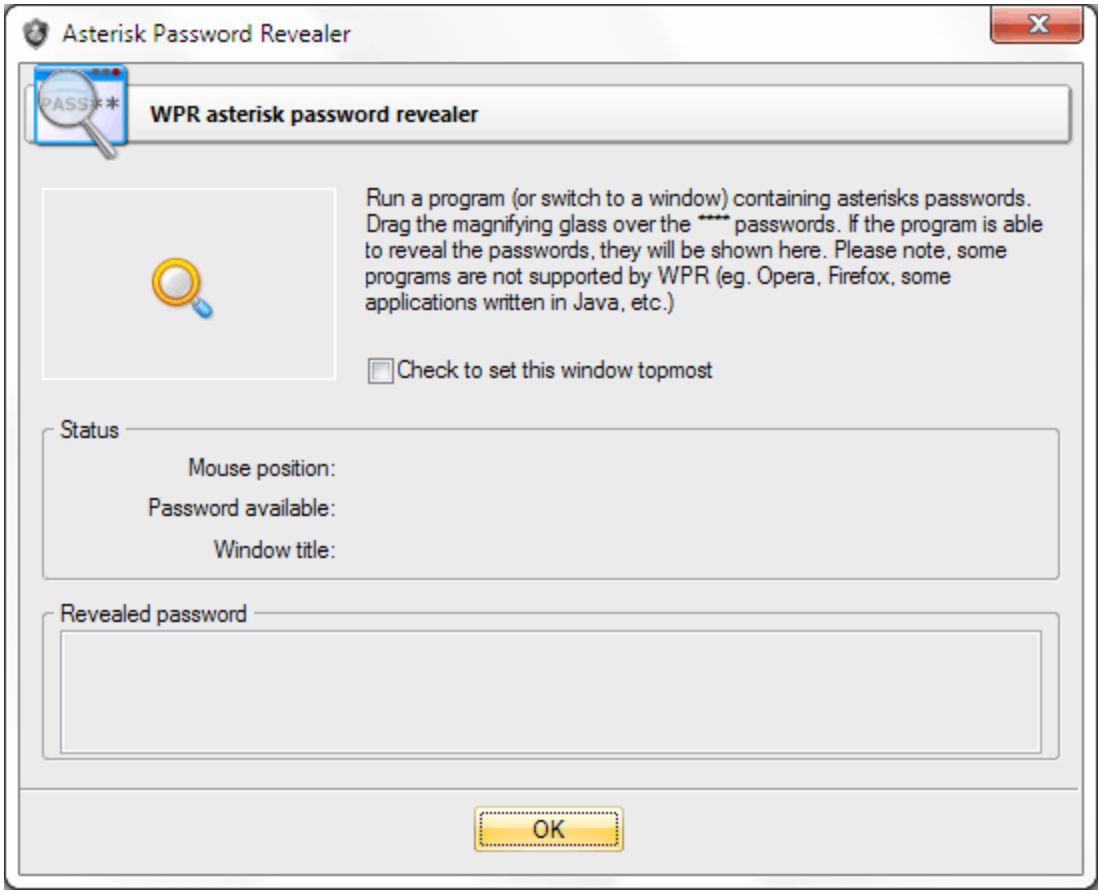


Backing up Active Directory database is much similar to the registry backup, except that the path to Active Directory the program determines automatically.



Administrator or Backup Operator privileges are required to run this plug-in.
Creating and saving Active Directory database may take quite some time: minutes or even hours for huge databases.

2.7.2 Asterisk Password Revealer



This tool allows to recover passwords hidden behind asterisks. It is often helpful when you need to quickly recall a **** password and don't have the necessary recovery tools handy. In order to get the *** password visible, you should have to drag the magic magnifier from the WPR window to the field with asterisks.

This method works both for Windows controls and Internet Explorer windows. It has a number of restrictions though:

- Some applications have their own GUI, and therefore Asterisks Revealer may be unable to interact with such applications. Those include Opera, Mozilla, Firefox, etc.
- Some websites have a built-in protection, which hides either the garbage or the actual asterisks behind the asterisk characters * (asterisks hidden behind asterisks!).
- In some Windows system dialogs asterisks also hide the * character and not the real password.

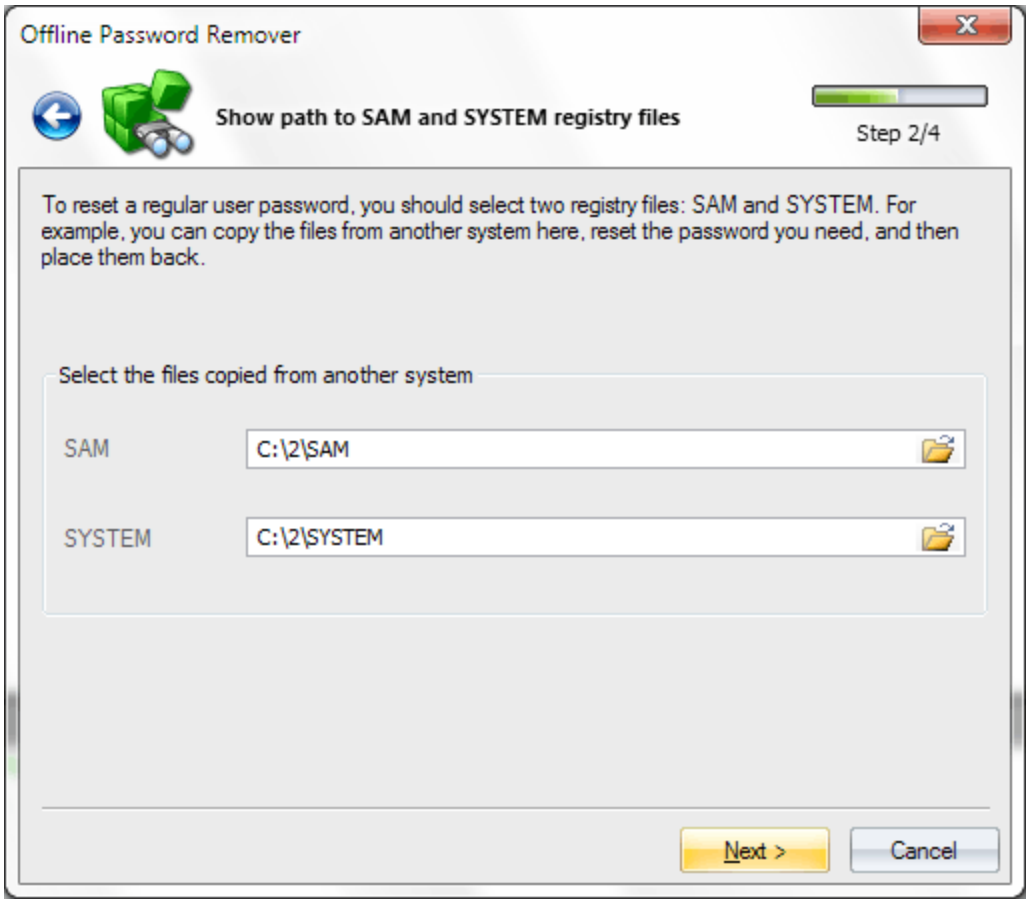
To ensure the proper operation of this tool, you are to have the administrator privileges.

2.7.3 Offline Password Remover

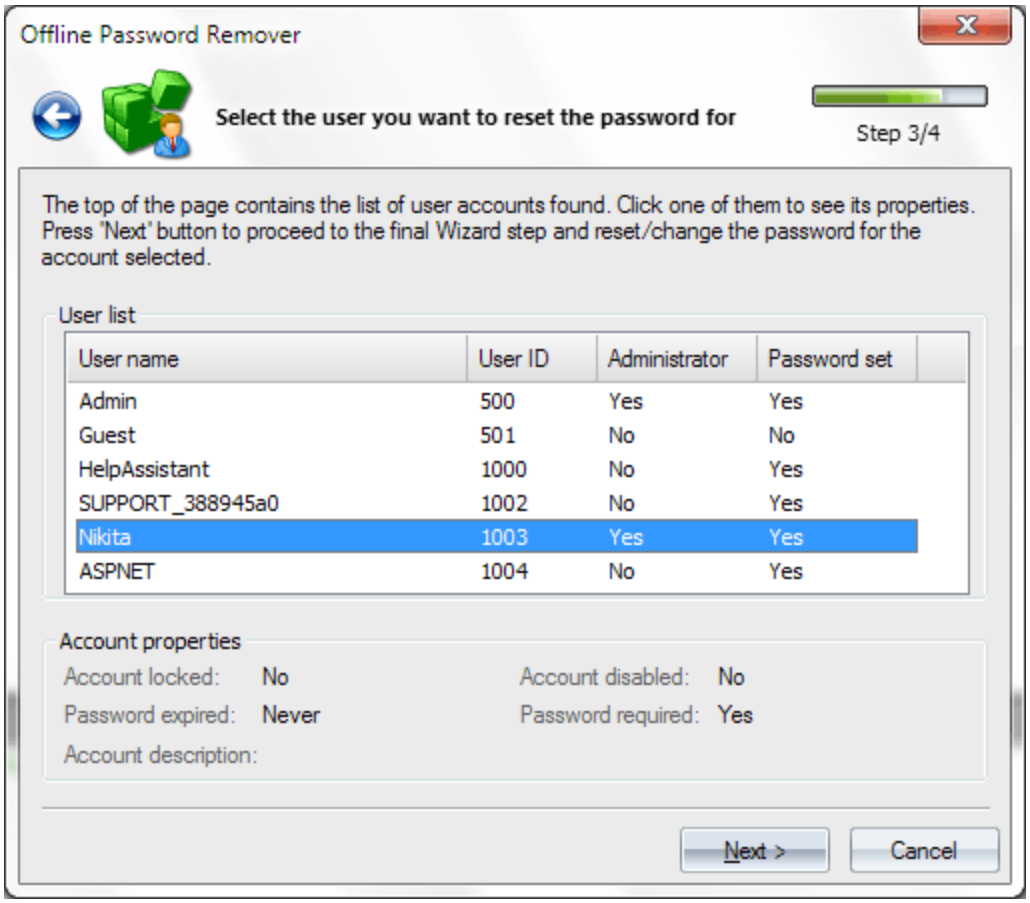
A helpful plug-in for removing and modifying passwords directly in the SAM registry file or in NTDS.DIT. For example, to regain access to a locked system, you do not necessarily have to recover the Windows logon password. Instead, you can just copy the SAM and SYSTEM registry files from the unbootable system, use this plug-in to remove the password for the account (or clear the lockup flag) and copy the files back. The password remover plug-in is made out as a wizard and consists of 4 steps:

1. On the first step, select the password source. That could be either a SAM file - for the regular accounts - or NTDS.DIT - for removing passwords in a domain.

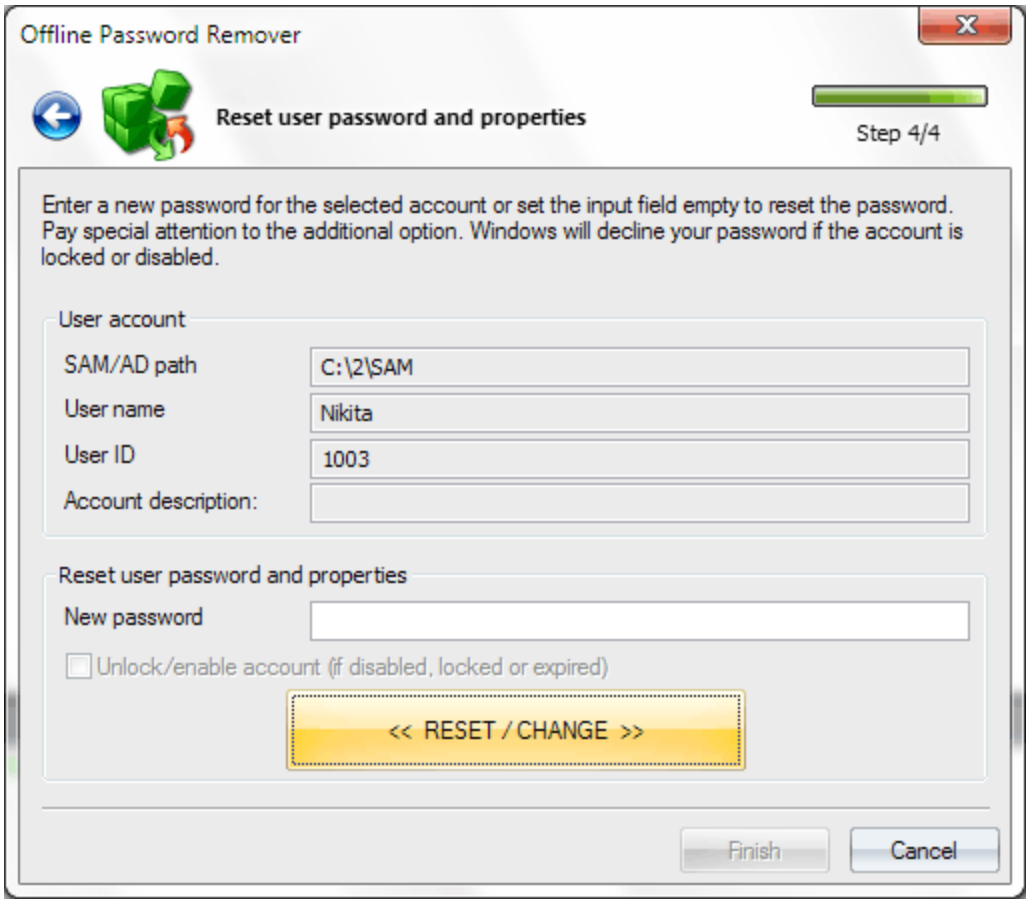
2. On the second step of the wizard, specify the path to the SAM/NTDS.DIT file and to the SYSTEM registry file. By default, NTDS.DIT is located in c:\windows\ntds. Registry files reside in c:\windows \system32\config.



3. On this step, we need to select the account we need to modify the password for. Select the user name and move on to the final step.



4. The 'New password' field is made for the new password (leave it blank to reset the password). If this field is disabled, it means that the password for that account is already empty. The same applies to the advanced option for unlocking locked or disabled user accounts.



Don't forget to save your SAM or NTDS.DIT files before making the final changes to them!

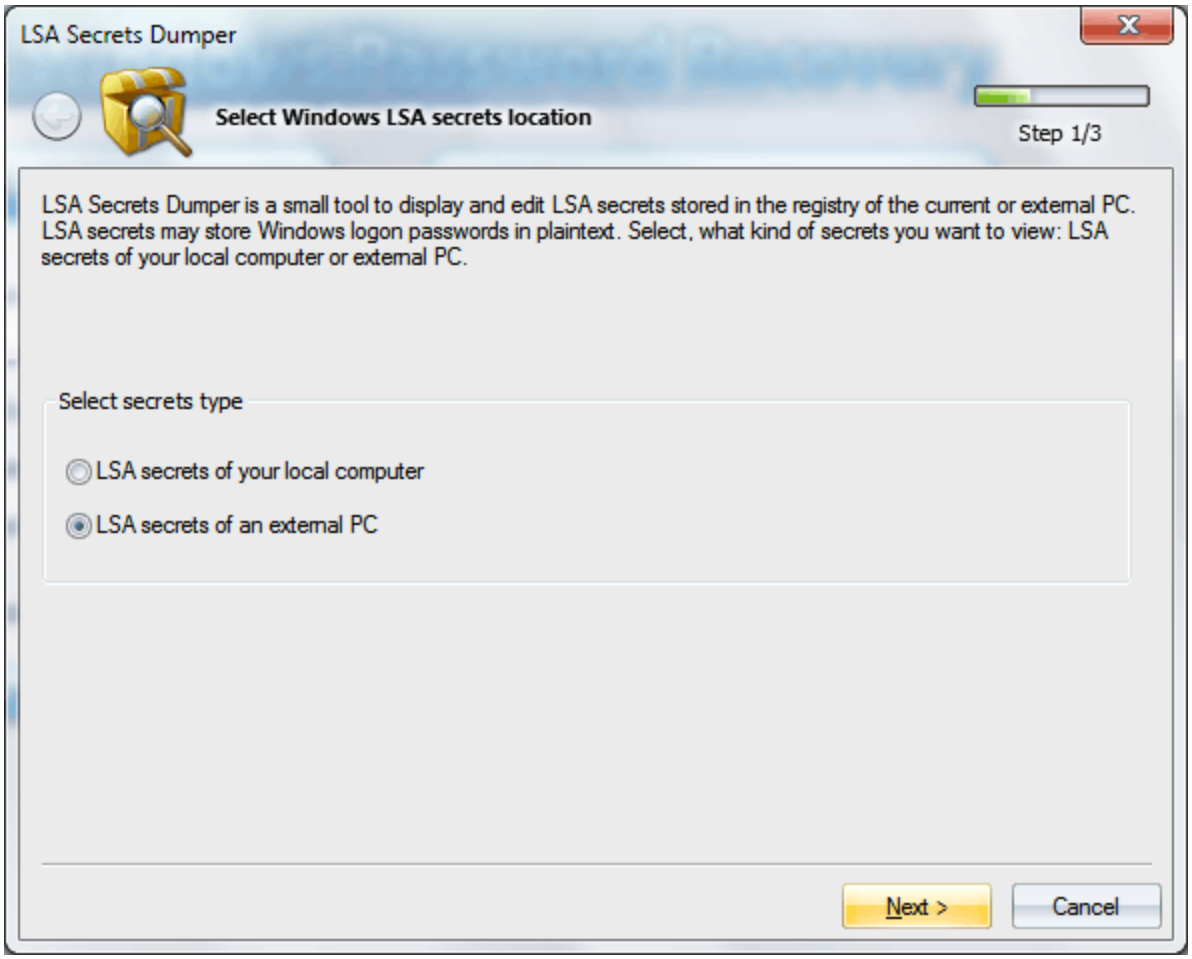
2.7.4 Forensic tools

2.7.4.1 LSA Secrets Dumper

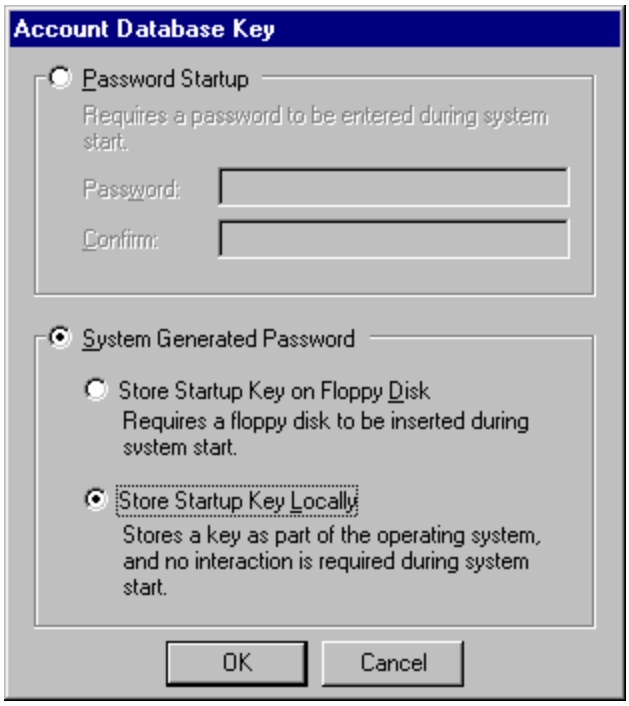
LSA secrets is a special protected storage for important data used by the Local Security Authority (LSA) in Windows. LSA is designed for managing a system's local security policy, auditing, authenticating, logging users on to the system, storing private data. Users' and system's sensitive data is stored in secrets. Access to all secret data is available to system only. However, as shown below, some programs, in particular Windows Password Recovery, allow to override this restriction.

Windows Password Recovery plugin for handling LSA secrets is a small tool for viewing, analyzing and editing LSA secrets. The plugin's wizard-driven user interface is quite simple and contains of just three steps:

1. First, select the type of secrets you are going to deal with. These can be secrets of the local system, where the application is running, or secrets of an external PC.

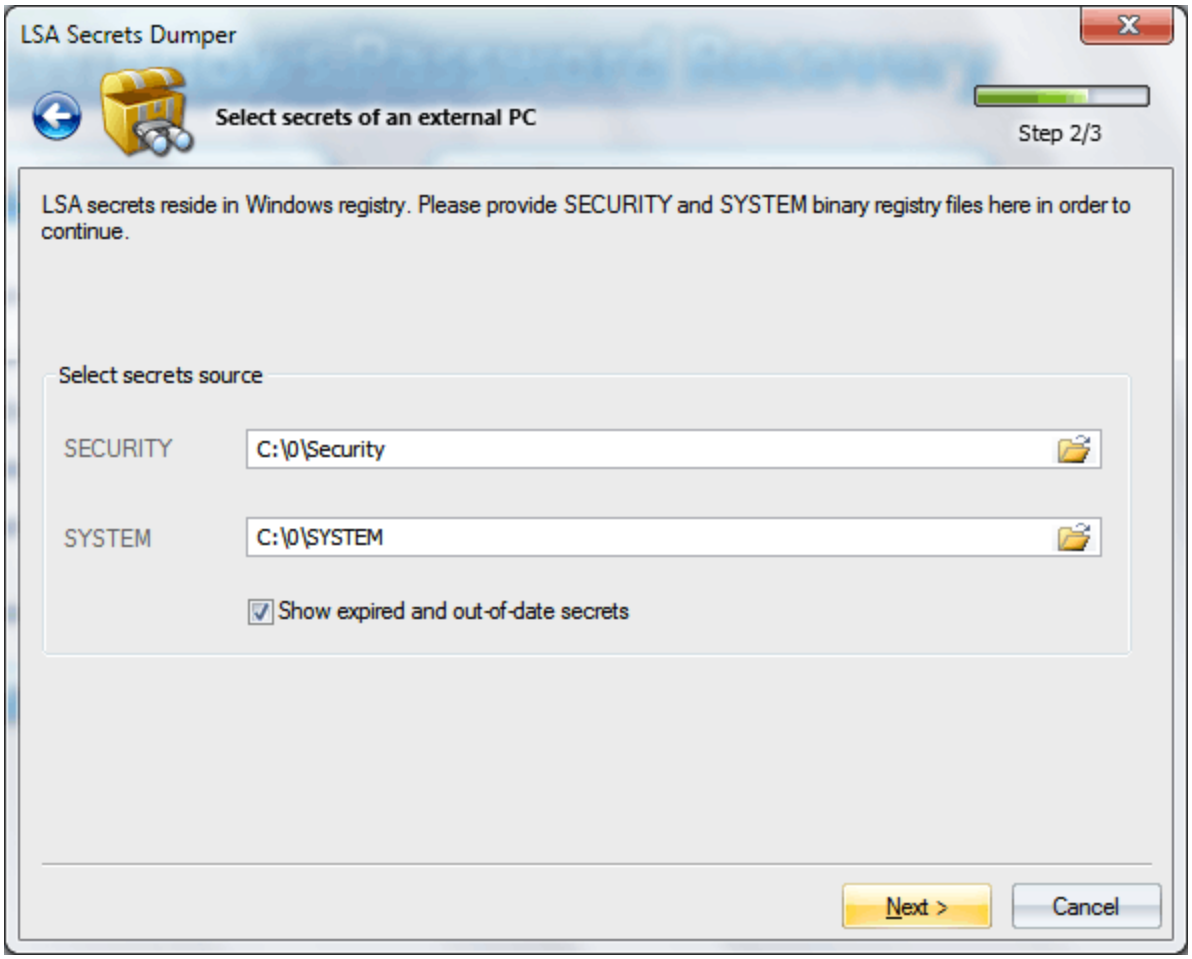


2. When selecting secrets of an external PC, you need to specify path to two registry files: SYSTEM and SECURITY. The SECURITY file contains encrypted secrets, and SYSTEM is necessary for decrypting those. You can find out more on encrypting secrets in our article. Please note that encrypting secrets involves SYSKEY. By default, SYSKEY is configured the way that it can be extracted from the registry (that is what SYSTEM is for).

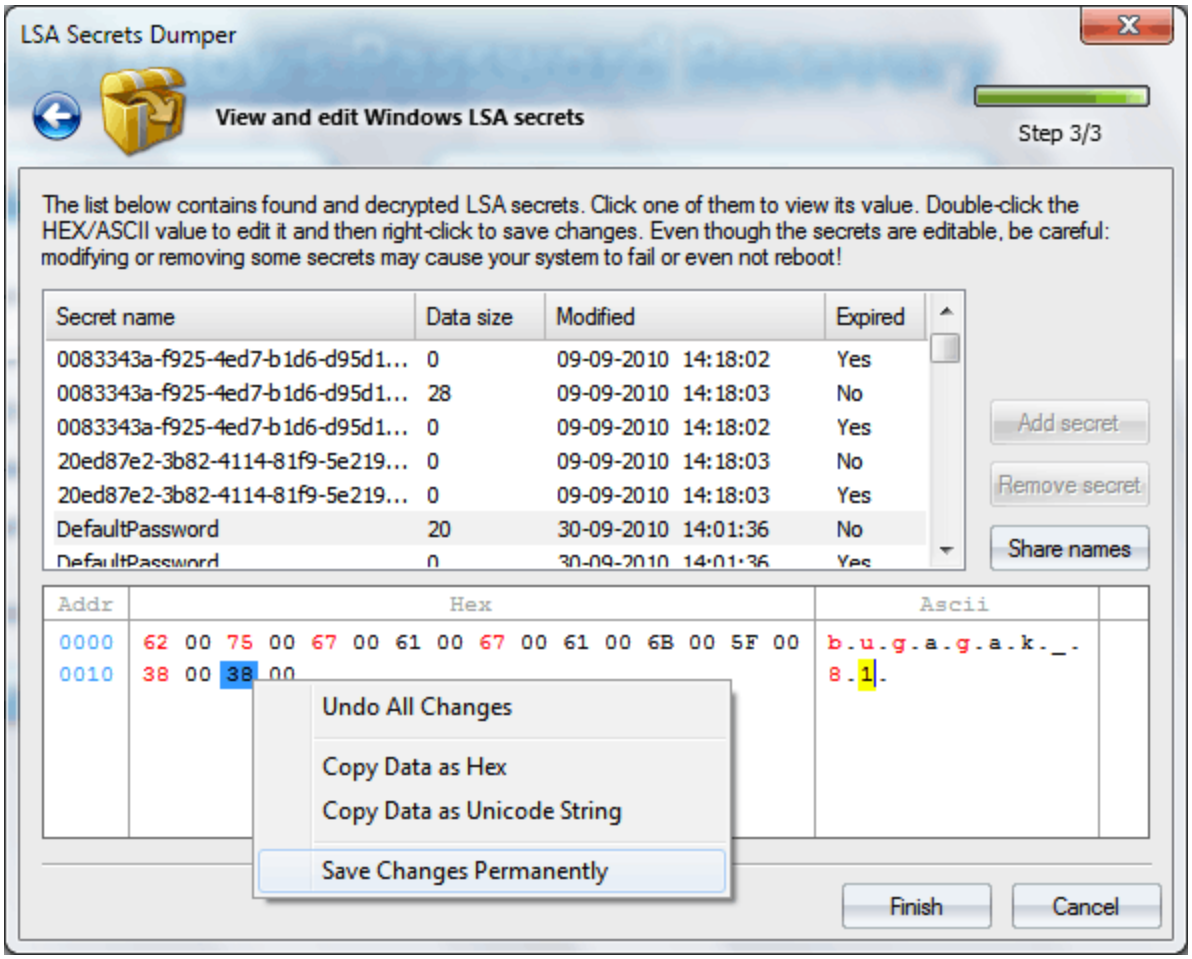


In some cases, it can be configured otherwise: to be either stored on a boot disk or to be derived from user password when the OS starts. One way or the other, the plugin supports all types of SYSKEY encryption.

Data stored in secrets is crucial for the operation of the entire system. Therefore, LSA secrets are stored in two copies: current (active) and previous (former). Modifying a secret places its current copy to the former one and replaces it with the new, modified secret. The plugin has an option for showing both active and former secrets.



3. The last step of the Wizard decrypts secrets and shows them as a list. To show the value of a secret, just click on its name. Enter the edit mode by double-clicking on one of the characters in the Hex or Ascii field (this marks it in yellow), and enter the new value. In the edit mode, use the cursor keys to move to the next character. Modified values are marked in red. To save changes, right-click on the Hex/ Ascii field and then select the save item on the menu that appears.



Keep in mind that certain secrets contain critical data, and modifying them may cause system instability or even impossibility of booting!

The plugin also allows adding and deleting secrets (secrets of current operating system only). Deleting a secret, whether old or new, automatically deletes both its copies.

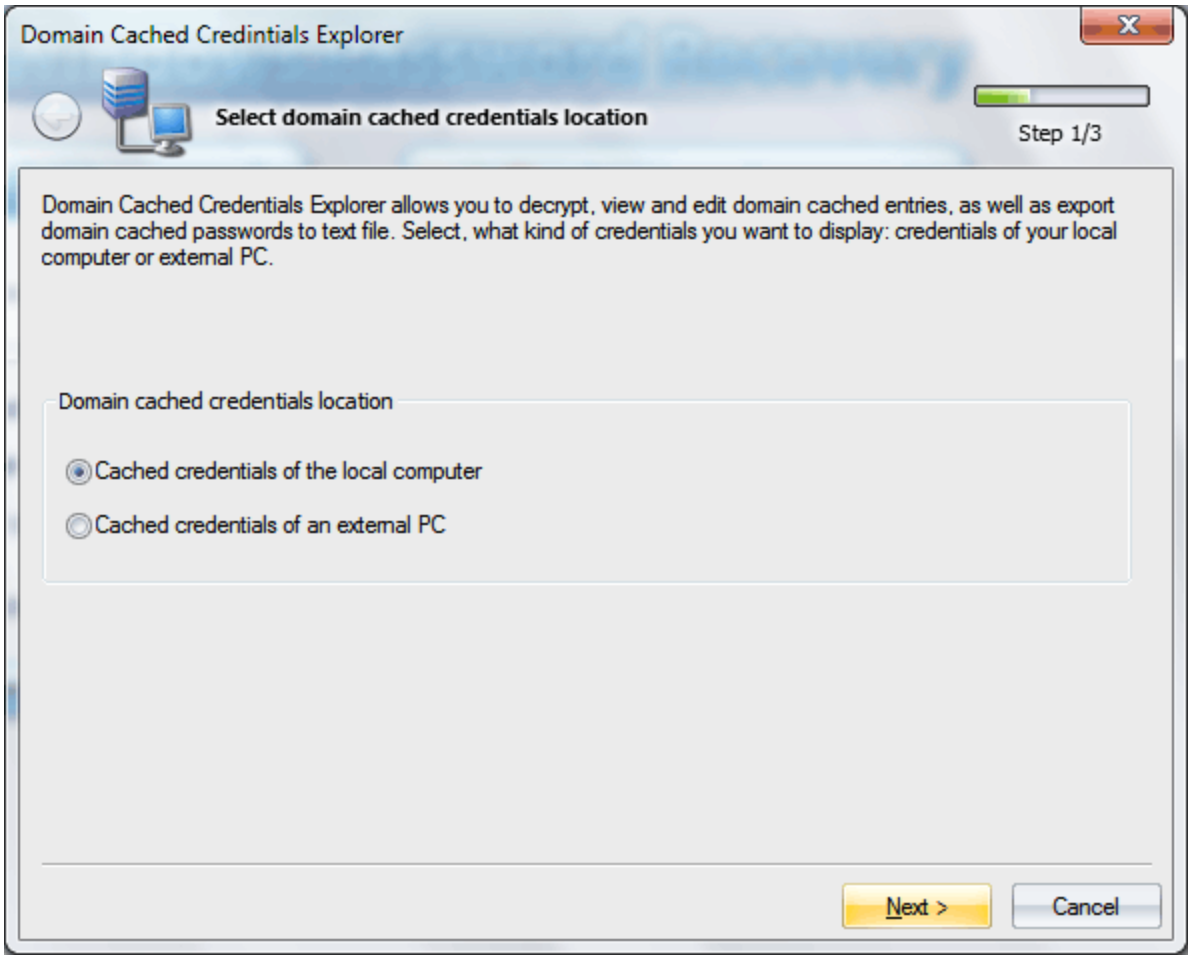
You can share your secrets with developers (Share Names button). This e-mails only the secret names, without the actual data. Analyzing the secret names will help us make the program more efficient.

2.7.4.2 Domain Cached Credentials Explorer

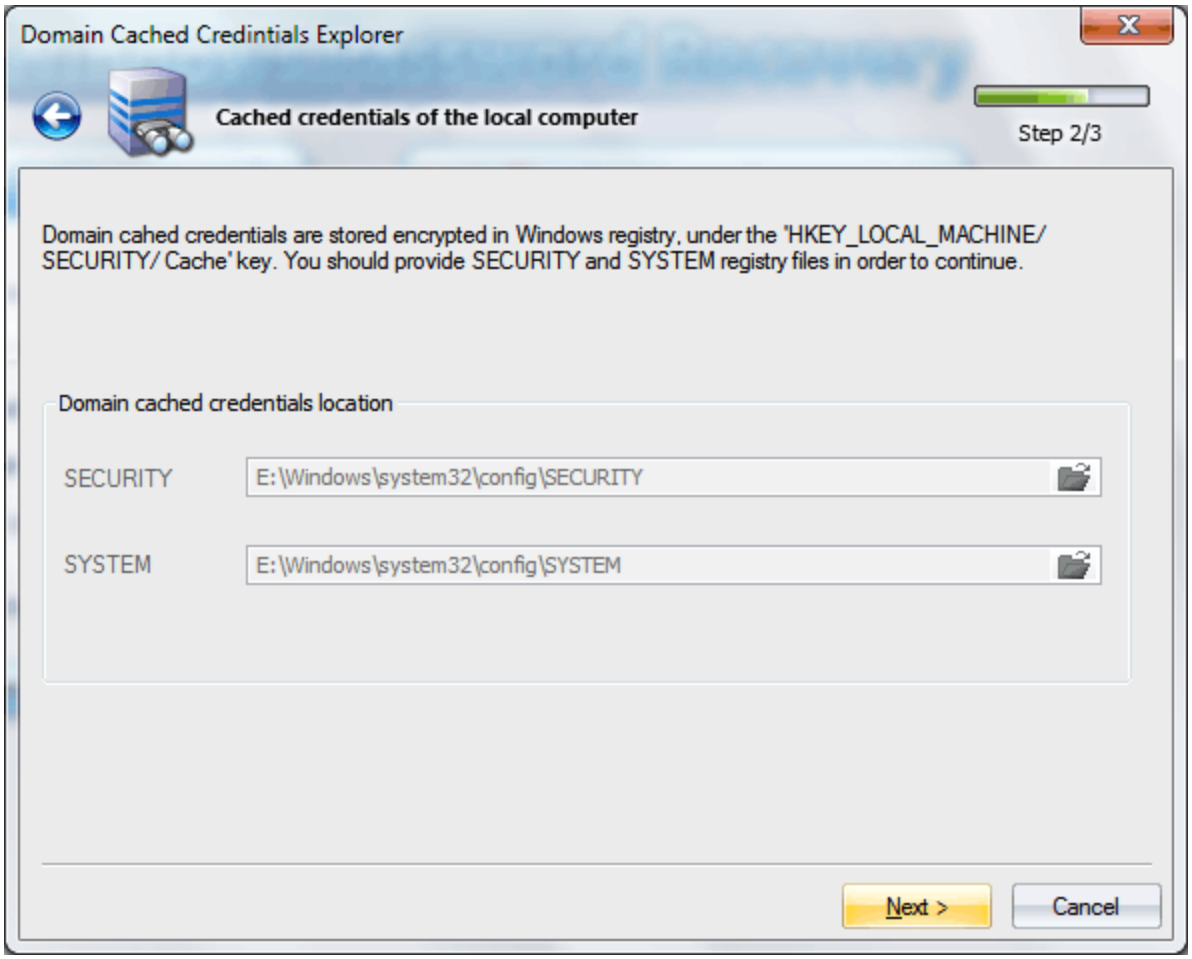
Beginning with version 2.0, the program allows reading cached domain records. Windows uses cached domain records to be able to connect to the server even if the logon server is unavailable for whatsoever reason.

The plugin for handling cached domain records includes three steps.

In the beginning, decide, which records are to be decrypted: cached records of current operating system or of some other computer.

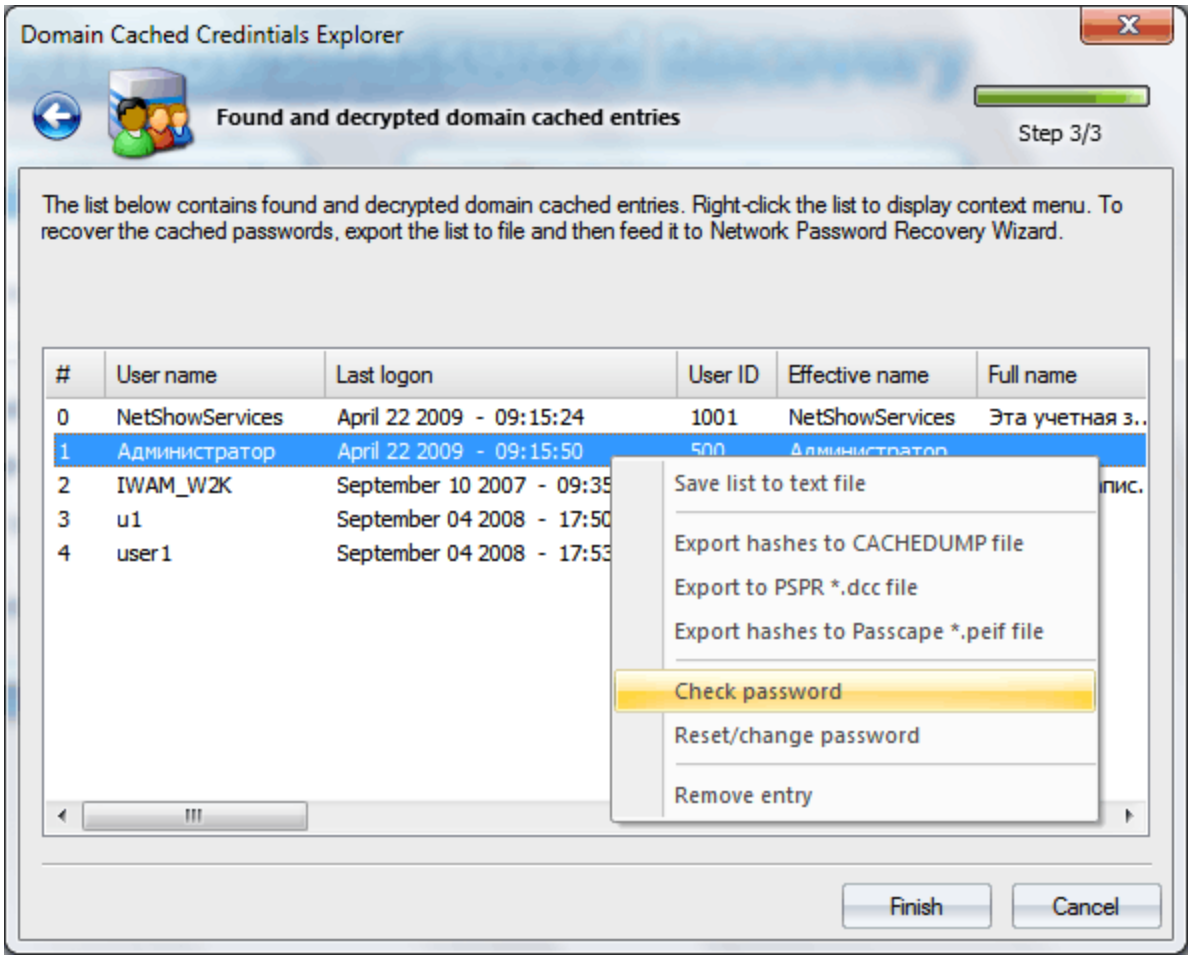


Cached domain records are stored in the SECURITY registry file. Thus, when selecting the option to read records from an external PC, on the next step of the Wizard, you should specify path to both SECURITY and SYSTEM registry used for decrypting the records. When selecting the option to read cached records of the local computer, on the second step of the wizard, the program will automatically locate those files. The registry files are located at the following folder C:\%WINDIR%\system32\config\, where %WINDIR% is the Windows directory.



If the reading was successful, in the final dialog you will see the decrypted domain records. Each record has several attributes. For example, user name, last logon time, group membership, cached user password (actually, hash).

Right-clicking on the list of records opens the context menu, which allows to:
- Save records with all attributes to a text file.
- Export password hashes to a PWDUMP, *.DCC or *.PEIF file. Please note that the PWDUMP format stores records not quite properly; therefore, it is more preferable to store password hashes as *.DCC or *.PEIF files.
- Check or edit the password for a cached domain record.
- Delete record.

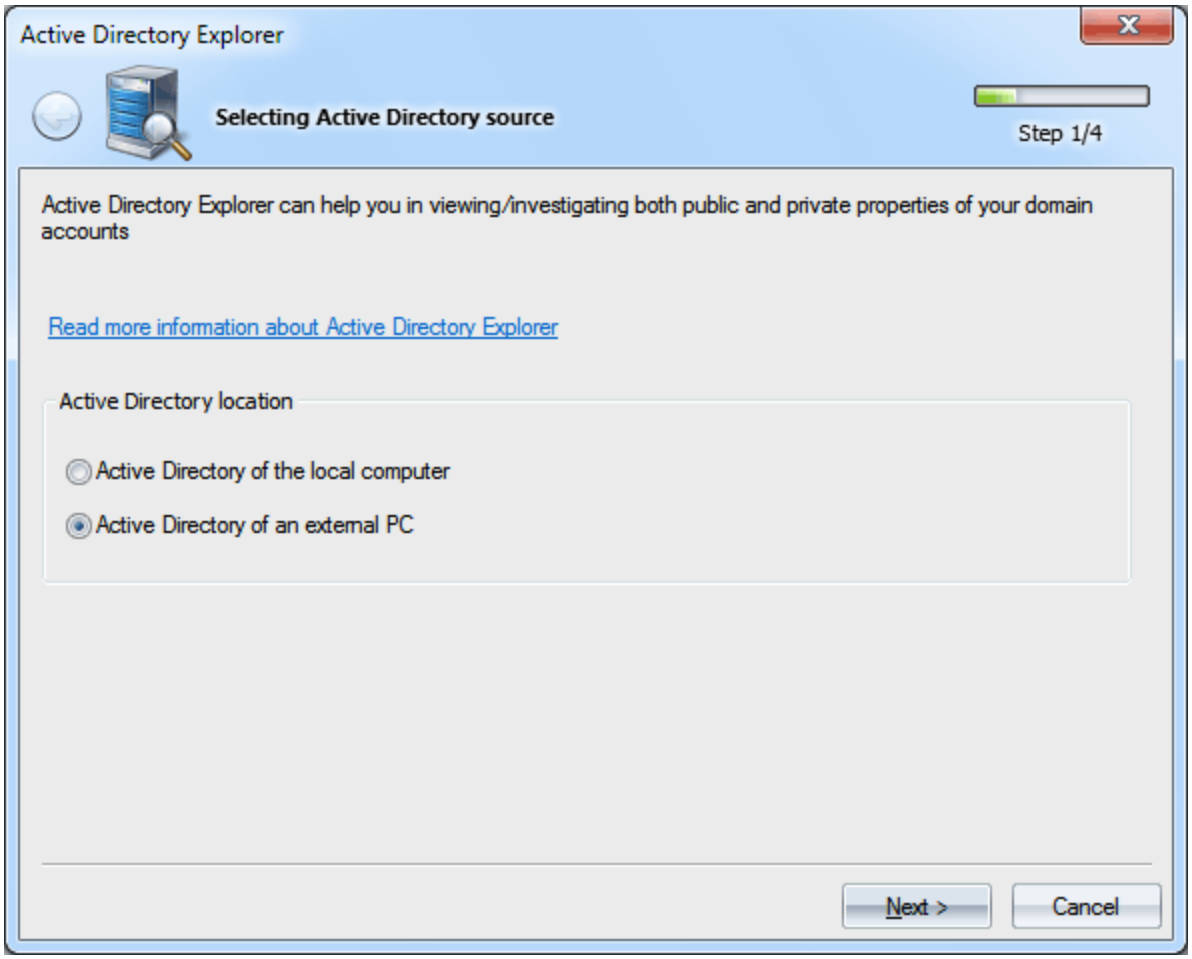


To recover cached domain password, you can take advantage of Network Password Recovery Wizard; just have the hashes exported to a file of one of the above mentioned formats beforehand.

2.7.4.3 Active Directory Explorer

Active Directory Explorer is a small utility for viewing, analyzing and editing properties (attributes) of domain accounts, both public and private.

In the beginning, select the type of the AD database you are going to work with: local or external.

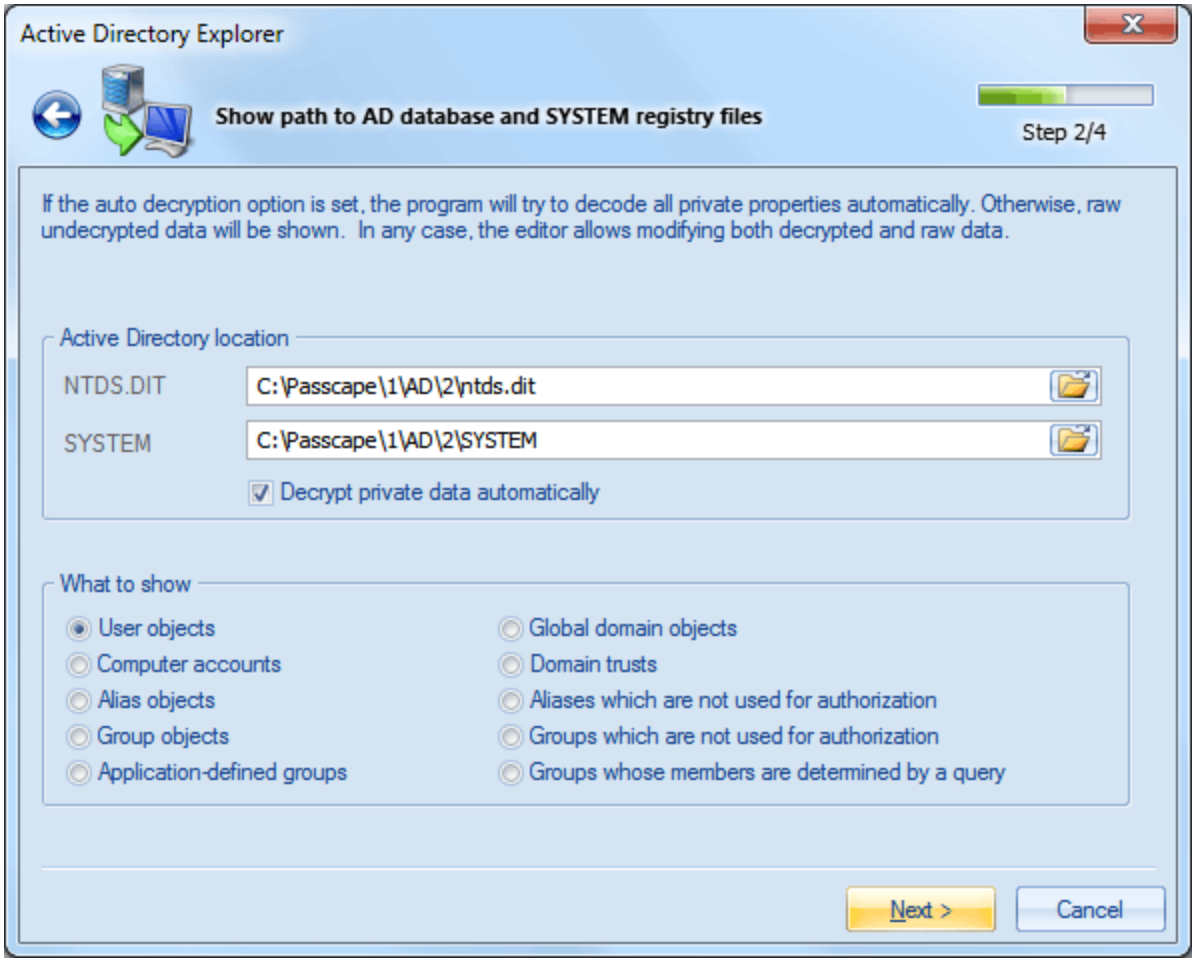


When selecting the external database, specify the path to the **NTDS.DIT** file and to the **SYSTEM** registry. The latter is required for decrypting private data. If the automatic decryption is enabled, all the encrypted attributes of an account will be decrypted on the fly. In any case, the editor allows editing both decrypted and raw data. For safety reasons, the editor mode is available for external databases only!

You should also specify what object you want to display. There are 10 types of domain objects. See the table below.

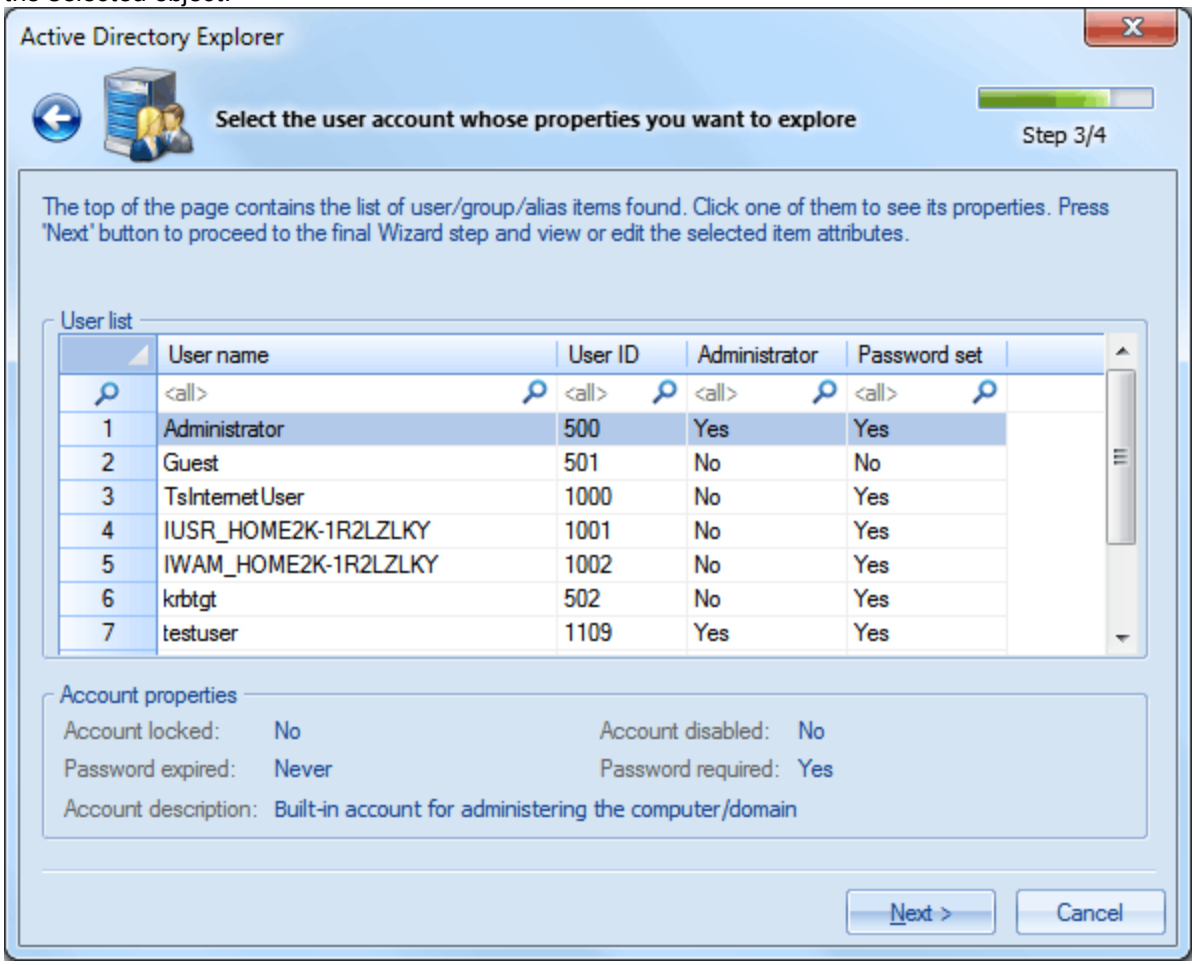
Domain object	Description
User object	An object of class user. A user object is a security principal object; the principal is a person or service entity running on the computer. The shared secret allows the person or service entity to authenticate itself.
Global domain object	Represents a typical domain object that do not conform to other types.
Computer accounts	Represents a computer object that is associated with individual client or server machines in an Active Directory domain.
Domain trusts	Represents a user object that is used for domain trusts. A trusted domain is a domain that is trusted to make authentication decisions for security principals in that domain.
Alias objects	A security or distribution group that can contain universal groups, global groups, other domain local groups from its own domain, and accounts from any domain in the forest. Aliases can be granted rights and permissions on resources that reside only in the same

Domain object	Description
	domain where the domain local group is located.
Aliases which are not used for authorization	Represents an alias object that is not used for authorization context generation.
Group objects	A database object that represents a collection of user and group objects and has a security identifier (SID) value.
Groups which are not used for authorization	Represents a group object that is not used for authorization context generation.
Application-defined groups	An application-defined group.
Query groups	An application-defined group whose members are determined by the results of a query.



Once the data source is selected, move on to selecting accounts. Some Active Directory databases contain tens or even hundreds of thousands of domain records. Reading such large databases and completing the list of users may take some time. Selecting just one record shows brief information on it at the bottom: status, whether a password is set and whether it is expired, account description, etc.

Clicking the '*Next >*' button launches the process of gathering and decrypting all available attributes for the selected object.



Each attribute consists of a name and a value. For example, '**Common-Name**' contains the account name, and '**Unicode-Pwd**' attribute stores its password hash. For a more detailed description of an attribute, select it on the list and then click on the link that appears on the description field. Double-clicking on the data field opens the selected attribute for editing. When done editing, right-click on the text to open the context menu and then save the changes to the ntds.dit file or discard them.

Here is the description of some account attributes. The complete description is available on the website of Microsoft.

Common-Name
The name of the account.

DBCS-Pwd
Contains LAN Manager password of the account.

Unicode-Pwd
The password of the user in Windows NT one-way format (OWF). Note that you cannot derive the clear password back from the OWF form of the password.

Lm-Pwd-History
Contains the password history of the user in LAN Manager one-way function format. The attribute is used for compatibility with LAN Manager 2.x clients, Windows 95, and Windows 98.

Nt-Pwd-History
The password history of the user in Windows NT OWF format.

Primary-Group-ID
Relative identifier (RID) for the primary group of the user. This is Domain Users group, by default.

Bad-Pwd-Count
Contains the number of times the user tried to log on to the account using an incorrect password.

Admin-Count
Indicates that the account is a member of one of the Administrative groups (directly or transitively).

Logon-Hours
The hours that the user is allowed to logon to the domain.

Last-Logon
The last time the user logged on to the account.

Bad-Password-Time
The last time the user attempted to log on to the account with an invalid password. This value is stored as a large 8-byte integer that represents the number of 100 nanosecond intervals since January 1, 1601 (UTC).

Last-Logon-Timestamp
This is the time that the user last logged into the domain.

Pwd-Last-Set
The date when the password for this account was last changed.

Account-Expires
The date when the account expires. A value of 0 or 0x7FFFFFFFFFFFFFFF indicates that the account never expires.

Supplemental-Credentials
Stores the encrypted version of the user's password. Used in authentication.

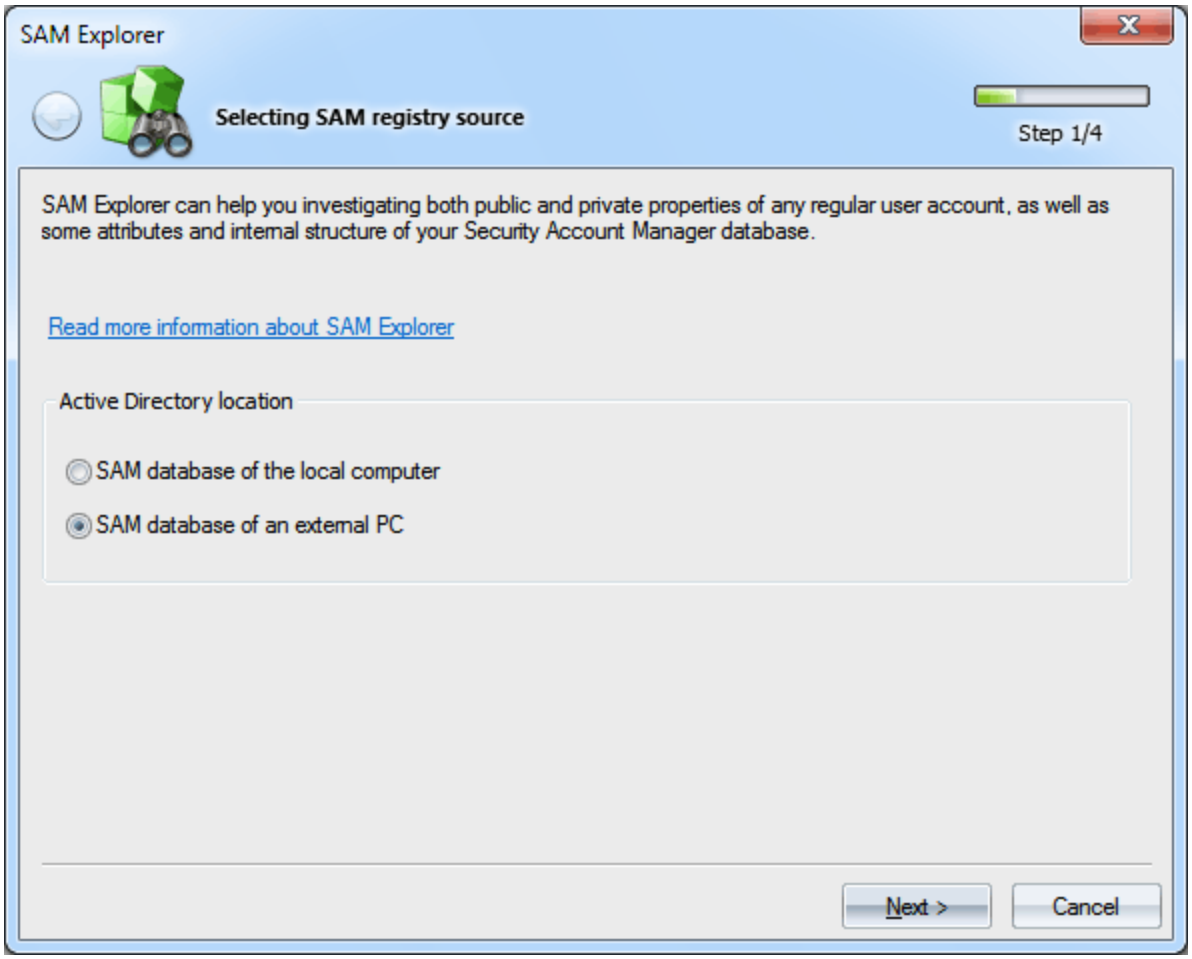
User-Account-Control
Flags that control the behavior of the user account. This value can be a combination of one or more of the following values.
0x00000001 Logon script is executed for the account.
0x00000002 The account is disabled.
0x00000008 Home directory is required.
0x00000010 The account is currently locked out.
0x00000020 No password is required.
0x00000040 The user cannot change the password.
0x00000080 The cleartext password is to be persisted
0x00000100 This is an account for users whose primary account is in another domain.
0x00000200 This is a default account type that represents a typical user.
0x00000800 Trust account for a system domain that trusts other domains.
0x00001000 This is a computer account for a computer that is a member of this domain.

0x00002000 This is a computer account for a system backup domain controller that is a member of this domain.
0x00010000 The password for this account will never expire.
0x00020000 This is an MNS logon account.
0x00040000 The user must log on using a smart card.
0x00080000 The account, under which a service runs, is trusted for Kerberos delegation.
0x00100000 The security context of the user will not be delegated to a service even if the service account is set as trusted for Kerberos delegation.
0x00200000 Restrict this principal to use only Data Encryption Standard (DES) encryption types for keys.
0x00400000 This account does not require Kerberos pre-authentication for logon.
0x00800000 The user password has expired.
0x01000000 The account is enabled for delegation. Enables a service running under the account to assume a client identity and authenticate as that user to other remote servers on the network.
0x04000000 The object is a read-only domain controller (RODC)

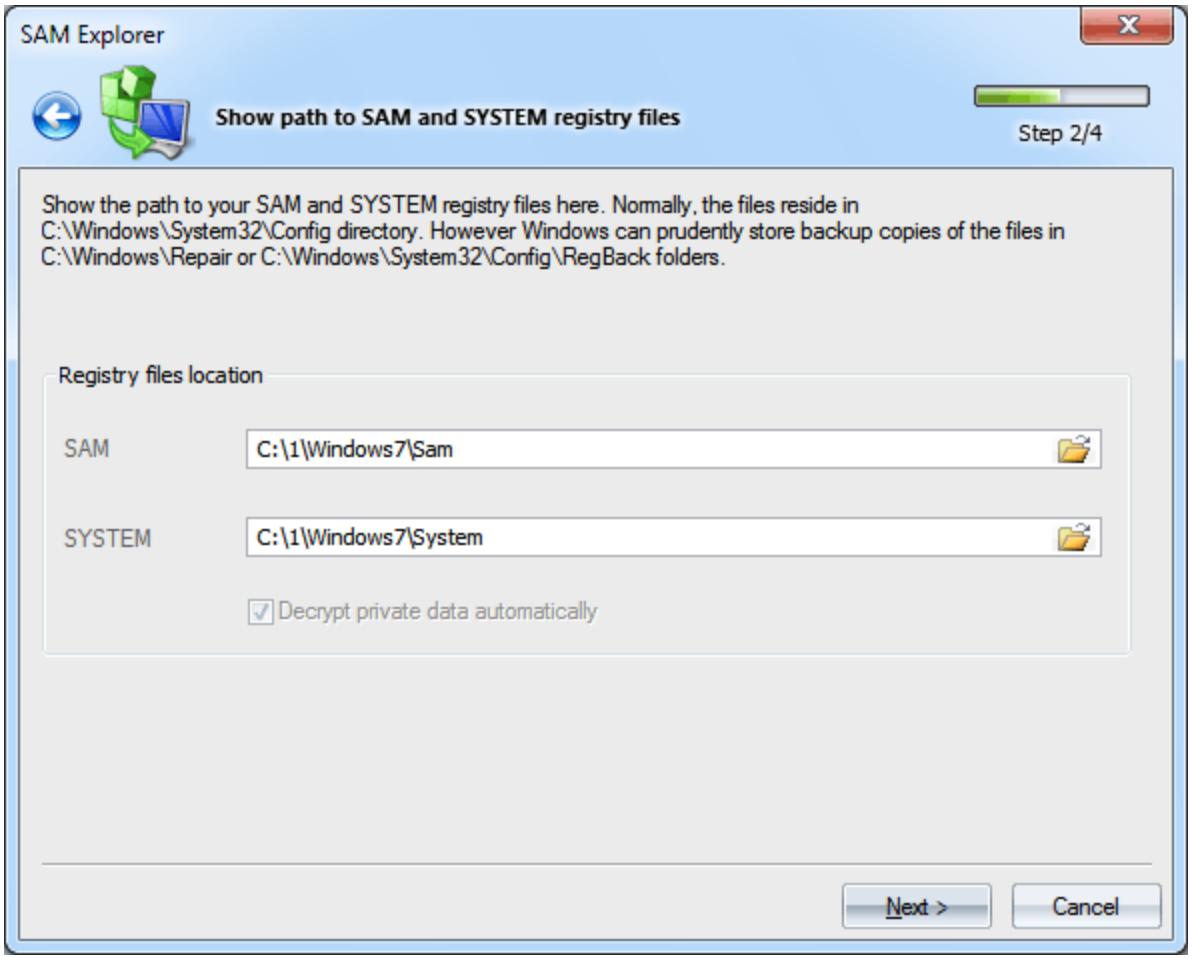
2.7.4.4 SAM Explorer

SAM Explorer allows you to view, analyze and edit the properties and statistics of Windows user accounts. SAM, which is short for **Security Account Manager**, is an RPC server, which manages Windows accounts database and stores passwords and private user data, groups logical structure of accounts, configures security policy (e.g., password or account lockout policy), gathers statistics (last logon time, logon count, failed logon attempt count, etc.) and controls access to the database. The SAM database is stored in the registry (in the key **HKEY_LOCAL_MACHINE\SAM\SAM**), which is inaccessible to anyone, except the system (even to administrators). On the physical level, the SAM database is a binary registry file with the respective name, located in %WINDIR%\System32\Config, where %WINDIR% is the Windows installation folder.

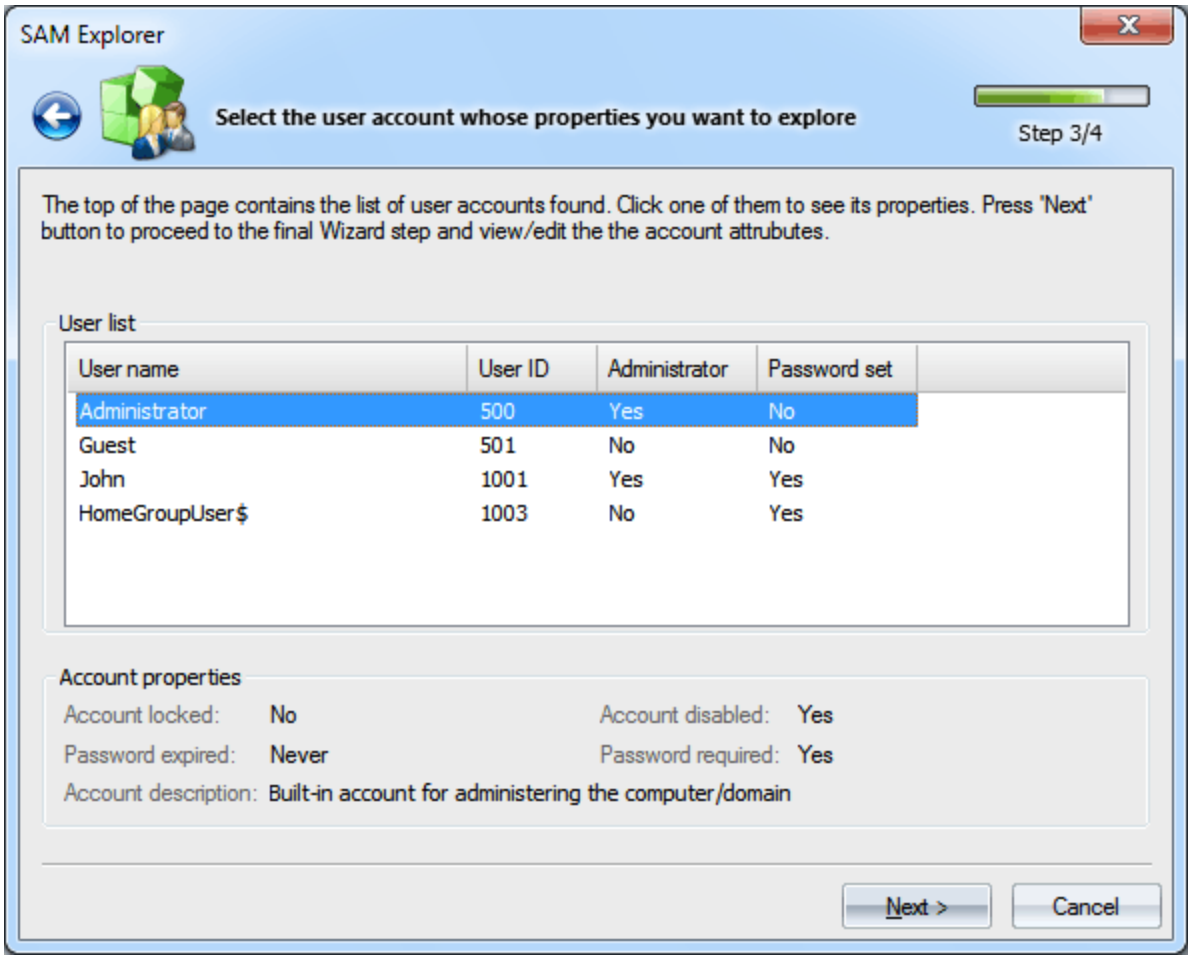
In the beginning, the Wizard prompts you to select the type of the SAM database: local or external. Please note: if you select a local database, for safety reasons, the editor will not be available, and the database will open in the read-only mode.



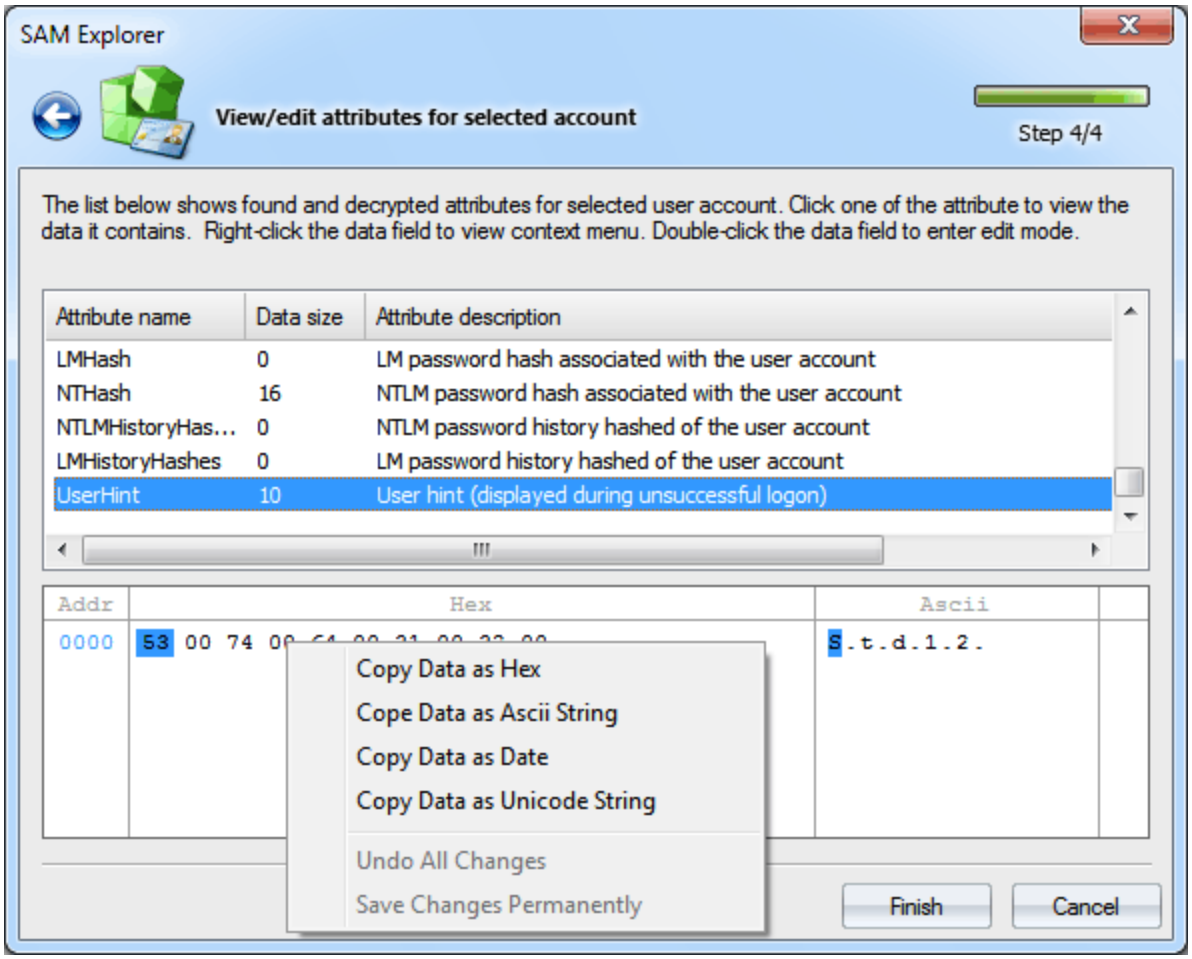
If you select the SAM database on an external computer, on the second step of the Wizard, specify the path to the SAM and SYSTEM registries. By default, both the files are located in **C:\Windows \System32\Config**. Keep in mind that Windows can providently store copies of the registry files in the backup folders, such as **C:\Windows\Repair** or **C:\Windows\ Config\RegBack**.



On the third step, move on to selecting the account you need to get the attributes for. Select the user and then click Next.



That gives you the list of attributes for the selected account. Selecting a certain attribute on the list shows the data common to that attribute at the bottom of the editor. To open it for editing, double-click on the data field; upon completion, select the save changes item on the context menu.



Description of SAM account attributes.

DataRevision
32-bit unsigned interger that stores version of the data structure. It is divided into 2 WORDs: version major and version minor.

LastLogon
A 64-bit value, equivalent to a FILETIME, indicating the time at which the account last logged on.

LastLogoff
A 64-bit value, equivalent to a FILETIME, indicating the time at which the account last logged off.

PasswordLastSet
A 64-bit value, equivalent to a FILETIME, indicating the time at which a password was last updated.

AccountExpires
A 64-bit value, equivalent to a FILETIME, indicating the time at which an account is no longer permitted to log on.

LastBadPasswordTime
A 64-bit value, equivalent to a FILETIME, indicating the time at which an account last tried to logged on

unsuccessfully.

UserID
A 32-bit unsigned integer representing the RID of the account.

PrimaryGroupId
A 32-bit unsigned integer indicating the primary group ID of the acount.

UserAccountControl
A 32-bit flag specifying characteristics of the account.

CountryCode
A 16-bit unsigned integer indicating a country preference specific to this user. The space of values is the international country calling code. For example, the country code of the United Kingdom, in decimal notation, is 44.

CodePage
A 16-bit unsigned integer indicating a code page preference specific to this user object. The space of values is the Microsoft code page designation.

BadPasswordCount
A 16-bit unsigned integer indicating the number of bad password attempts.

LogonCount
A 16-bit unsigned integer indicating the number of times that the user account has been authenticated.

AdminCount
A 16-bit unsigned integer indicating that the account is a member of one of the administrative groups (directly or transitively).

OperatorCount
A 16-bit unsigned integer indicating that the account is a member of the Operators group.

UserName
Unicode string that specifies the name of the user account.

FullName
Unicode string that contains the full name of the user.

AdminComment
Administrator comment associated with the user account.

UserComment
Second user comment associated with the user account.

Parameters
Extended user parameters. Microsoft products use this member to store user configuration information.

HomeDirectory
Unicode string specifying the path of the home directory for the user account.

HomeDirectoryDrive
Specifies the drive letter to assign to the user's home directory for logon purposes.

ScriptPath
Unicode string specifying the path for the user's logon script file. The script file can be a .CMD file, an .EXE file, or a .BAT file.

ProfilePath
Unicode string that specifies a path to the user's profile.

WorkStations
Unicode string that contains the names (separated by commas) of workstations from which the user can log on. Up to eight workstations can be specified. The account flag UF_ACCOUNTDISABLE allows to disable logons from all workstations to this account.

LogonHours
21-byte bit string that specifies the times during which the user can log on. Each bit represents a unique hour in the week, in Greenwich Mean Time. The first bit is Sunday, 0:00 to 0:59; the second bit is Sunday, 1:00 to 1:59; and so on. Note that bit 0 in word 0 represents Sunday from 0:00 to 0:59 only if you are in the GMT time zone. In all other cases you must adjust the bits according to your time zone offset (for example, GMT minus 8 hours for Pacific Standard Time).

Groups
List of groups to which the user account belongs or does not belong.

LMHash
LM password hash associated with the user account.

NTHash
NTLM password hash associated with the user account.

LMHistoryHashes
LM password history hashed of the user account.

NTHistoryHashes
NTLM password history hashed of the user account.

UserHint
User hint (displayed during unsuccessful logon).

UserPicture
Logon picture associated with the account.

2.7.4.5 DPAPI tools

Starting with Windows 2000, Microsoft began equipping their operating systems with a special data protection interface, **Data Protection Application Programming Interface** (DPAPI). Currently DPAPI is very widely spread and used in many Windows applications and subsystems. For example, in the file encryption system, for storing wireless network passwords, in Microsoft Vault and Credential Manager, Internet Explorer, Outlook, Skype, Google Chrome, etc. This system has become popular among programmers first of all for its simplicity of use, as it consists of just a couple of functions for encrypting and decrypting data: CryptProtectData and CryptUnprotectData. However, despite its apparent simplicity, the technical implementation of DPAPI is quite complicated.

Passcape Software first in the world offers a set of 6 tools for comprehensive analysis and decrypting data encrypted with DPAPI. These utilities allow you to:

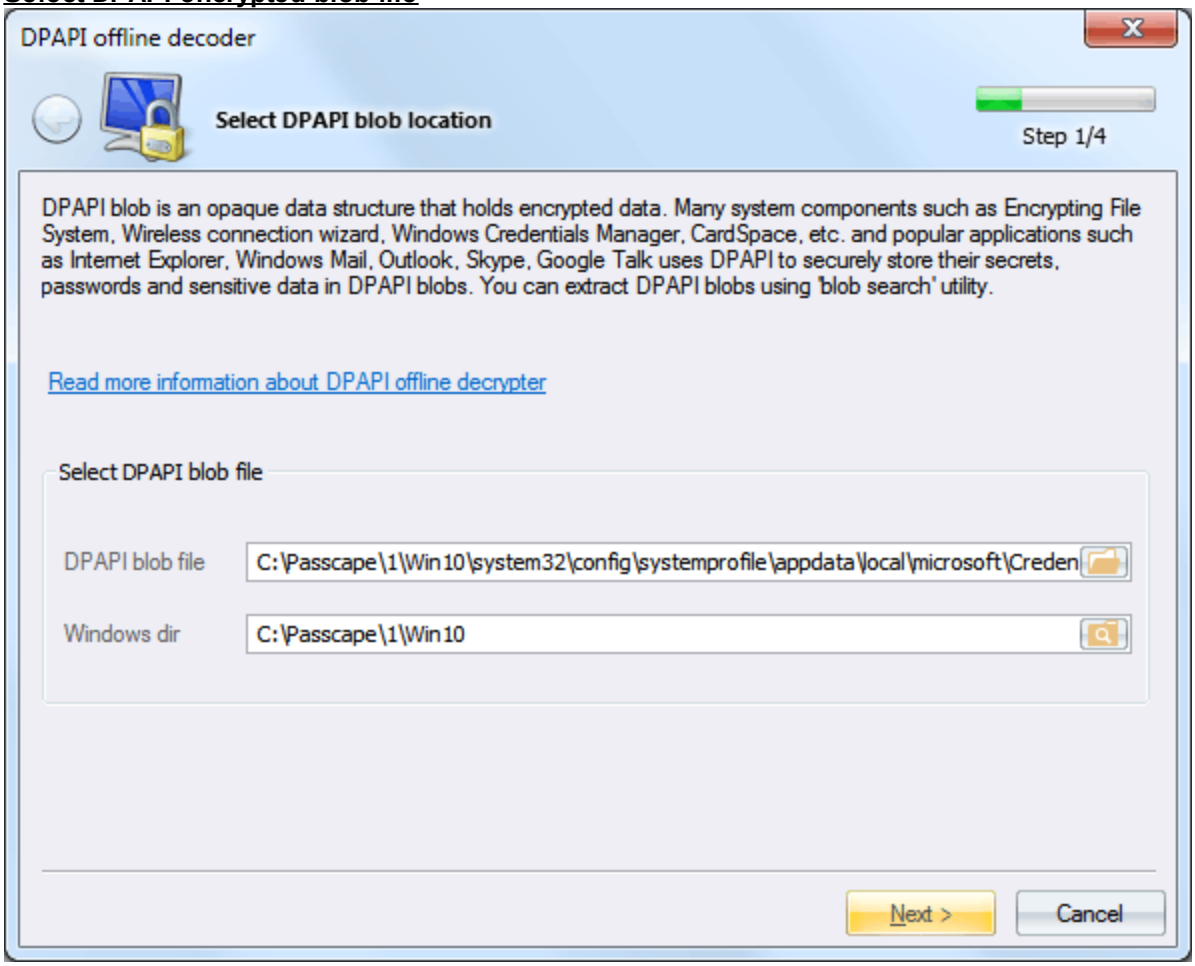
- Decrypt DPAPI blobs for any account

- Search DPAPI blobs on disk
- Decrypt DPAPI blobs encrypted under the SYSTEM account (e.g., WiFi passwords)
- Analyze and decrypt user's Master Keys
- Check user's password without dumping hashes from SAM or NTDS.DIT
- Decrypt history hashes of all passwords entered earlier (without using SAM or NTDS.DIT)

2.7.4.5.1 Decrypt DPAPI blob

The decryption of DPAPI blobs consists of four steps of the wizard.

Select DPAPI-encrypted blob file



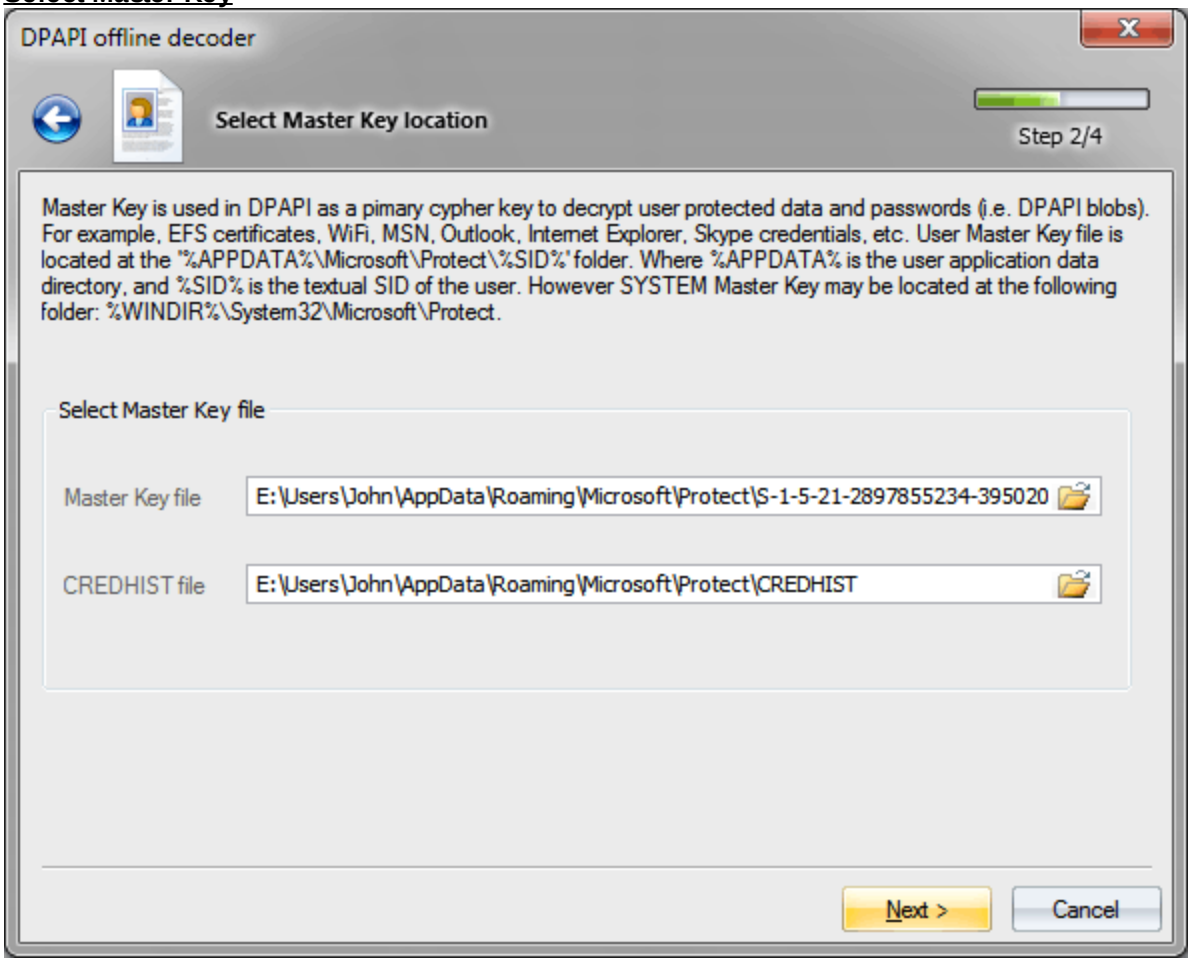
On the first step, specify the path to the DPAPI blob and Windows directory. It must be said that actual DPAPI objects may be stored in different locations of the operating system; for example, in individual xml files, in the registry, in Active Directory; and in different formats: binary, ASCII, UNICODE. There is a special tool for locating, extracting and saving DPAPI blobs to files. With that utility, for example, you can save all DPAPI blobs from a user's registry to individual files and use them in the program.

Here are storage locations for some DPAPI objects.

- Internet Explorer and Outlook passwords, WiFi passwords (XP only): user's registry, **%APPDATA%\ntuser.dat**
- Google Chrome: **%LOCALAPPDATA%\Google\Chrome**
- WiFi passwords (Windows Vista and higher): **%PROGRAMDATA%\Microsoft\Wlansvc**
- Network connection passwords (Windows Credential Manager): **%LOCALAPPDATA%\Microsoft\Credentials** or **%APPDATA%\Microsoft\Credentials**

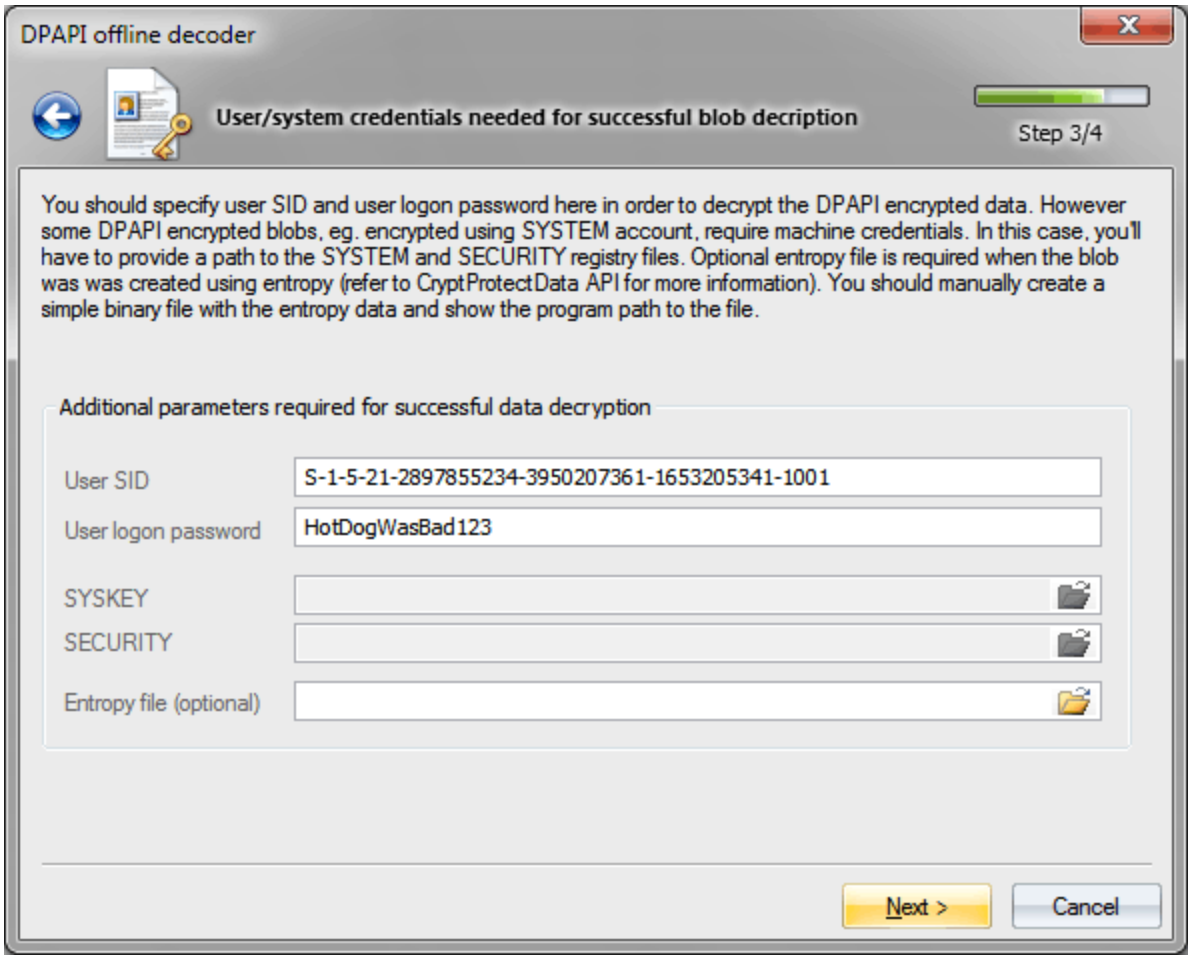
Use the finder utility to extract DPAPI data from there.

Select Master Key



Master Key is a set of 64 random bytes, used as the primary key when decrypting DPAPI blobs. Master Key is encrypted with user's password (or system's password if that is a system Master Key). User's Master Key is always located in **%APPDATA%\Microsoft\Protect\%SID%**, while a system account's Master Keys are stored in **%SYSTEMDIR%\Microsoft\Protect**. It must be noted that there can be several Master Keys, and only one of them is suitable for decrypting a certain object, the one with the name stored inside the DPAPI blob. When searching for a Master Key, the program may filter out unnecessary names. The folder **%APPDATA%\Microsoft\Protect** also contains the **CREDHIST** file, which is optional parameter, and in the majority of cases is not required for the decryption.

Decrypt Master Key

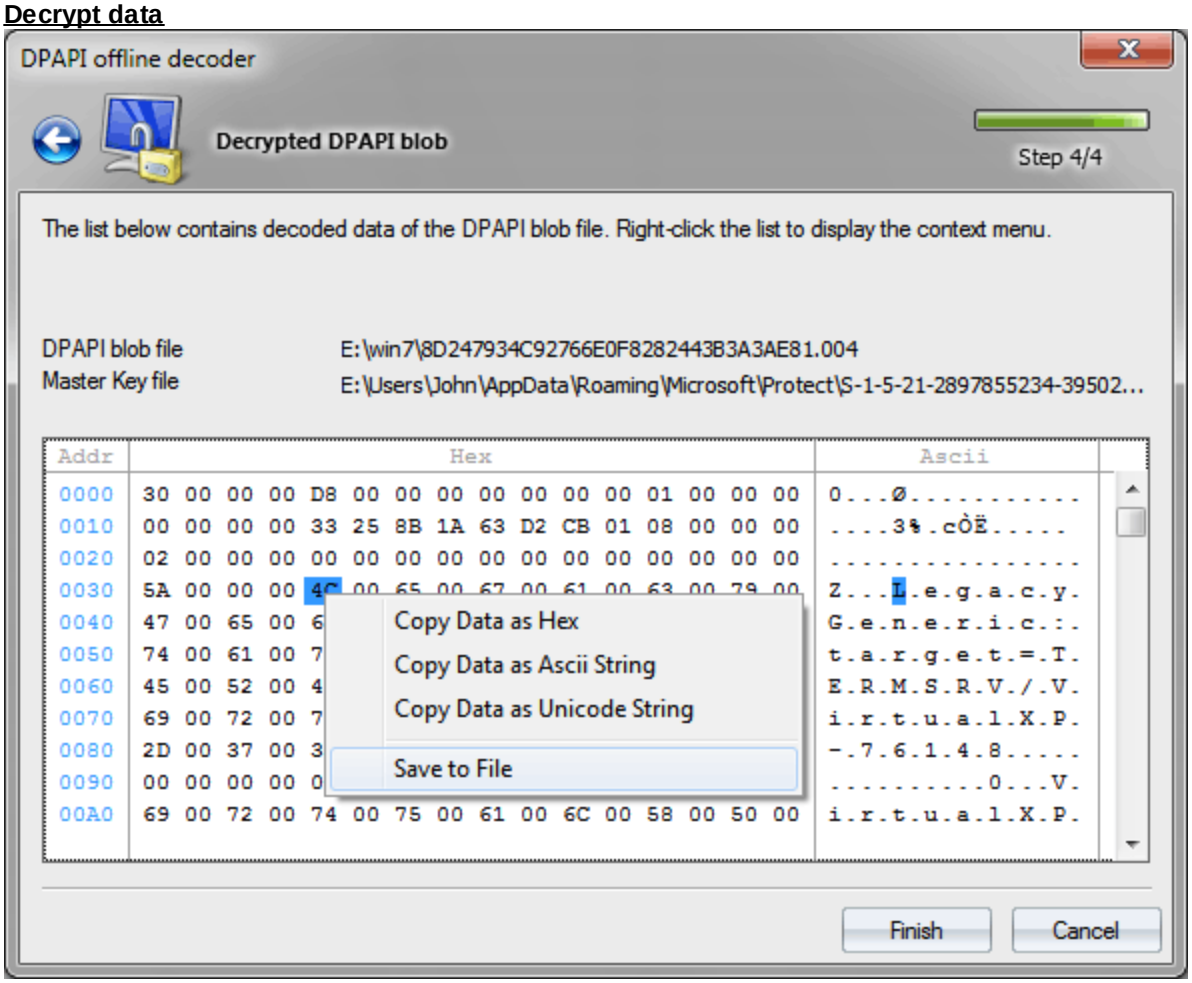


At least two parameters must be set in order to decrypt user's Master Key: user's logon password and his security identifier (SID), which is normally specified in the path to the Master Key or flashed in CREDHIST. One way or the other, Windows Password Recovery calculates user's SID automatically. To decrypt a system's Master Key, as it has been said already, setting a password doesn't make sense, as the program retrieves all data necessary for the recovery from two registry files: SYSTEM and SECURITY. If additional entropy was used when creating the DPAPI blob, you must manually create the binary entropy file and specify the path to it. For example, when encrypting Internet Explorer passwords, the UNICODE-formatted website name is used as entropy.

It is curious that Windows 2000 has a critical vulnerability, which allows decrypting any(!) DPAPI blob on a standalone PC without necessarily specifying user's logon password! I.e. all the data protected with DPAPI are actually vulnerable. This is a major fault in the implementation of DPAPI, which is known to Microsoft; however, other operating systems do not have this drawback. If the **CRYPTPROTECT_LOCAL_MACHINE** flag was set in the CryptProtectData function when protecting data, the decryption of that data is also possible without the user's logon password (for example, wireless network passwords). However, this is a peculiarity of an interface implementation and is not a bug.

Windows Password Recovery starting with version 9.7 utilizes some new vulnerabilities in DPAPI Master Key protection which were detected by our company. Thus to decrypt a Master Key of a domain user, the owner logon password is not necessary any longer.

Decrypt data

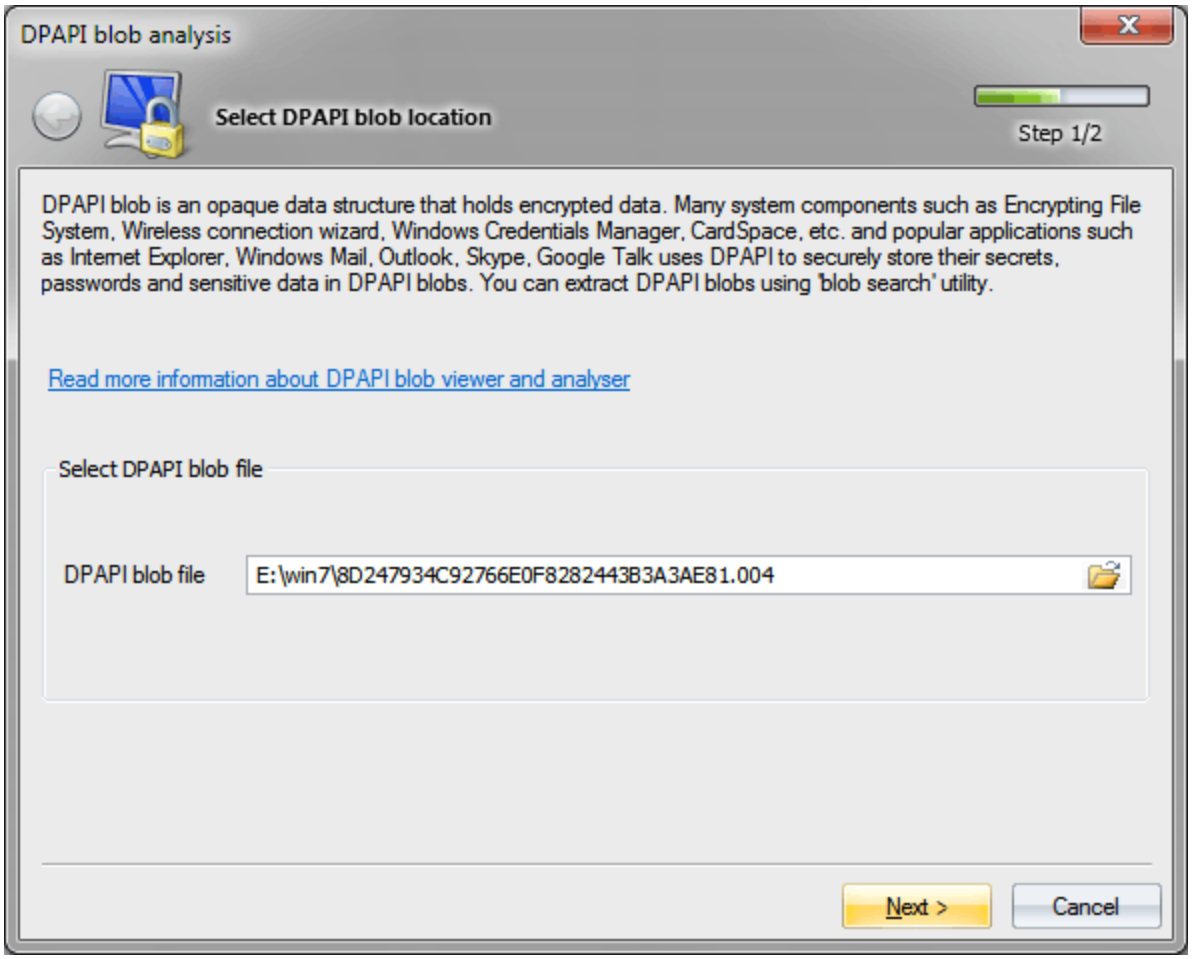


Having all that is necessary, Windows Password Recovery performs the final decryption of the DPAPI blob data, which you can then copy to clipboard or save to file. If the final step of the decryption ends up with an error, it is most likely because you have not set properly or not set at all the additional entropy. For example, Internet Explorer and Vista Ftp Manager uses the source page where the password was entered as entropy. Windows Credential Manager, similarly, uses certain string constants, and so on.

2.7.4.5.2 Analyse DPAPI blob

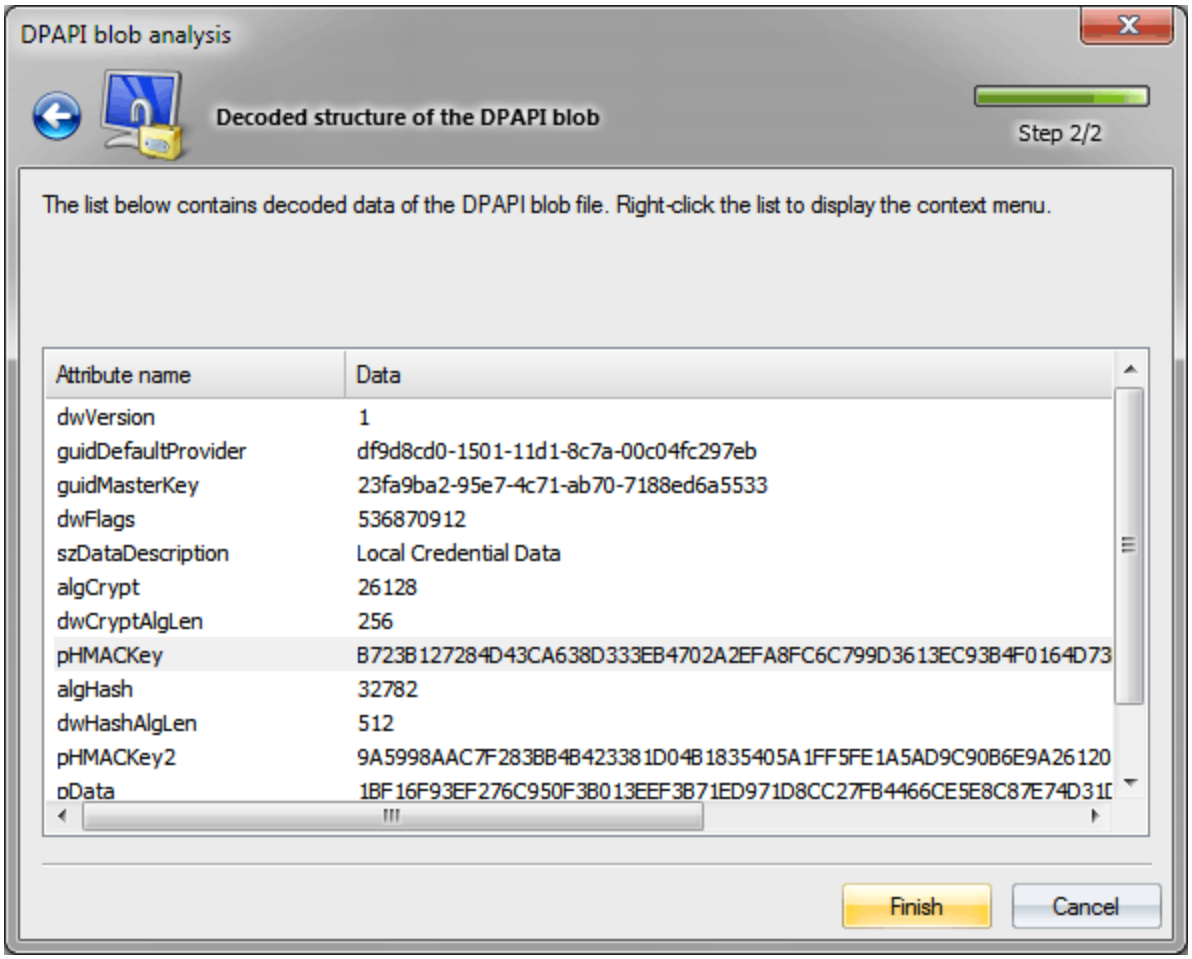
A DPAPI blob is an opaque binary structure, which contains application's private data encrypted using DPAPI. Many Windows applications and subsystems store passwords, secrets and private data in DPAPI blobs. To create files with DPAPI blobs (for further analysis), use our DPAPI blob look-up utility.

Specify path to DPAPI blob



This is the file that was created by the blob search tool.

And proceed to analyzing data



DPAPI blob is a binary data structure, which consists of the following consecutive attributes:

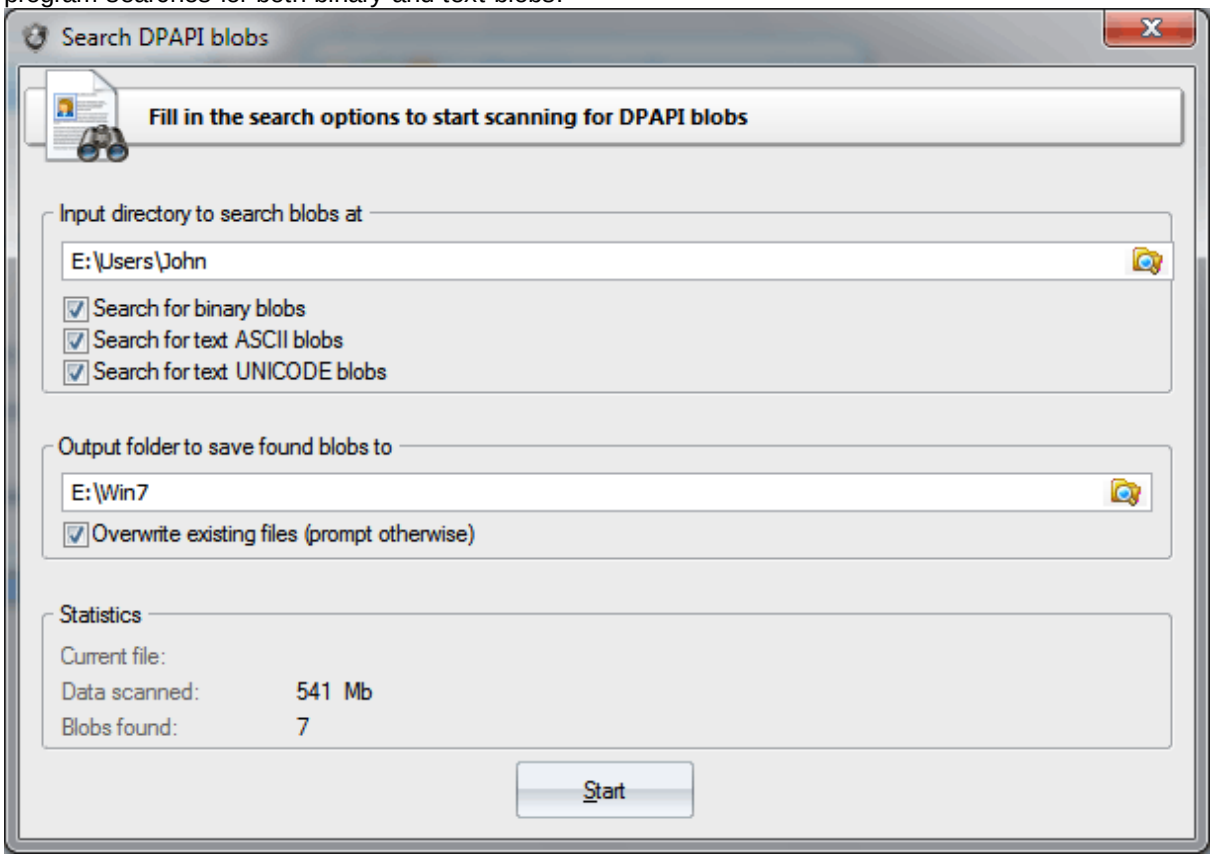
- **dwVersion** — data structure version. Current data version - 1.
- **guidDefaultProvider** — data encryption provider, used in encryption function calls, ensures compatibility of versions and organizes simple cryptological primitives. For example, you can set Blowfish or RC5 as a block cipher. Currently, Windows has the following default crypto provider: **df9d8cd0-1501-11d1-8c7a-00c04fc297eb**, which corresponds with the registry key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Protect\Providers\df9d8cd0-1501-11d1-8c7a-00c04fc297eb.
- **guidMasterKey** — Master Key GUID, which data is encrypted with. To decrypt data within a DPAPI blob, first of all you must decrypt the Master Key with the name set in the binary structure guidMasterKey. Only one Master Key can be attached to a DPAPI blob.
- **dwFlags** — various flags. For example, when the bit 3 is set, it indicates that the decryption of the data is to be carried out under the SYSTEM account. The bit (dwFlags&0x20000000) is set at all times.
- **szDataDescription** — data descriptor, which is set by the optional parameter LPCWSTR *szDataDescr* in the function CryptProtectData.
- **algCrypt** — data encryption algorithm. By default, Windows 7 uses AES 256 (which corresponds to 0x6610 in the hexadecimal or 26128 in the decimal notation), Windows XP - 3DES, Windows 2000 - RC4.
- **dwCryptAlgLen** — key length in the encryption algorithm.
- **pHMACKey** — HMAC key 1.
- **pSalt** — salt (optional).
- **algHash** — hashing algorithm. By default, Windows 7 uses SHA 512, Windows XP and Windows

2000 - SHA1.

- **dwHashAlgLen** — hash length in the hashing function.
- **pHMACKey2** — HMAC key 2.
- **pData** — actual encrypted data.
- **pSignHash** — digital signature for verifying data integrity.

2.7.4.5.3 Search DPAPI blobs

The DPAPI blob search dialog is rather trivial. All you need to specify is the source folder, which the program would search for DPAPI blobs, and the target folder, where found blobs are to be stored. The program searches for both binary and text blobs.



Example of a path, where you can find files, containing binary DPAPI blobs:
C:\Users\John\AppData\Roaming\Microsoft\Credentials

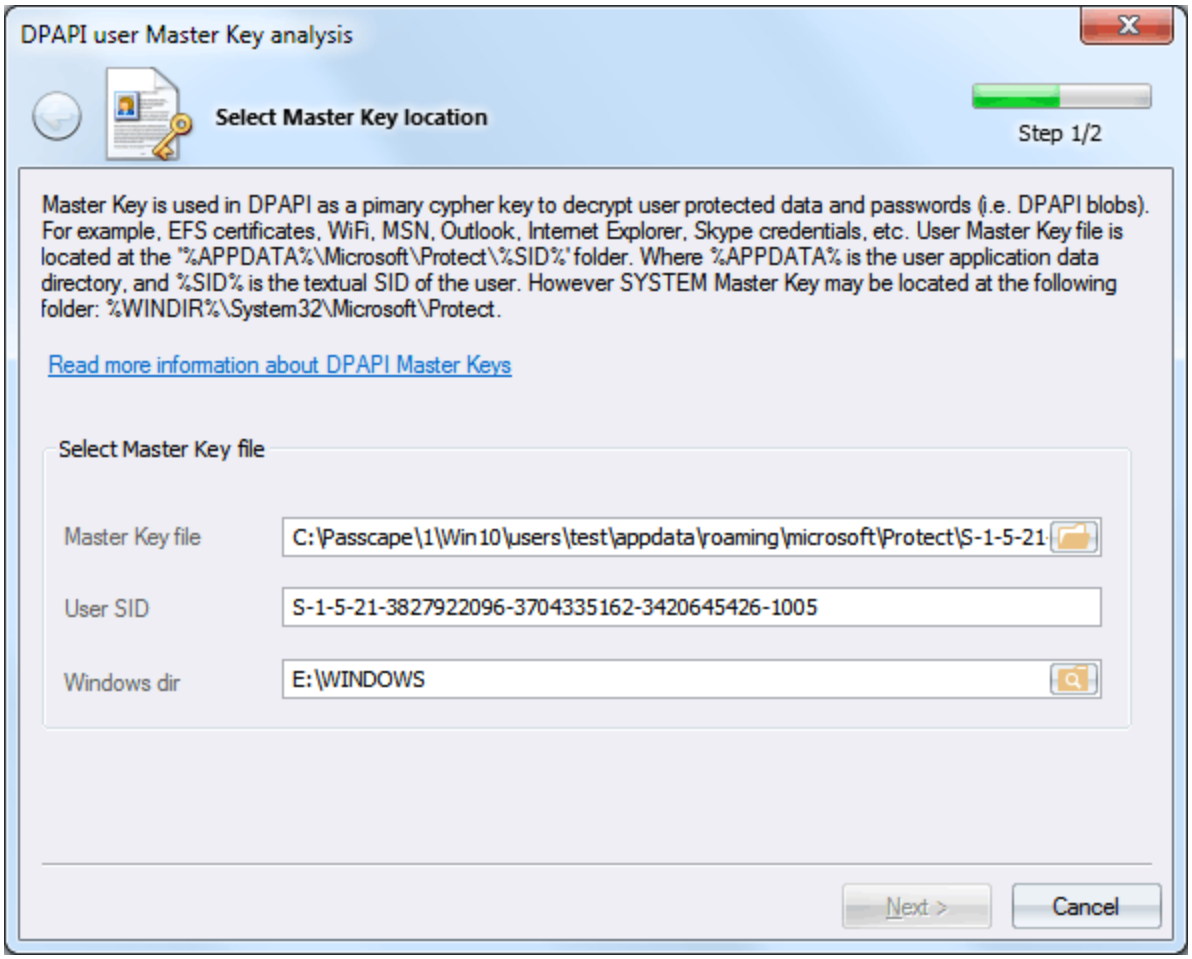
Example of a path, where you can find files, containing textual DPAPI blobs:
C:\ProgramData\Microsoft\Wlansvc

Keep in mind that if you want to search for blobs in current user's registry or in Active Directory database, you should first back up the files to a separate directory.

2.7.4.5.4 Master Key analysis

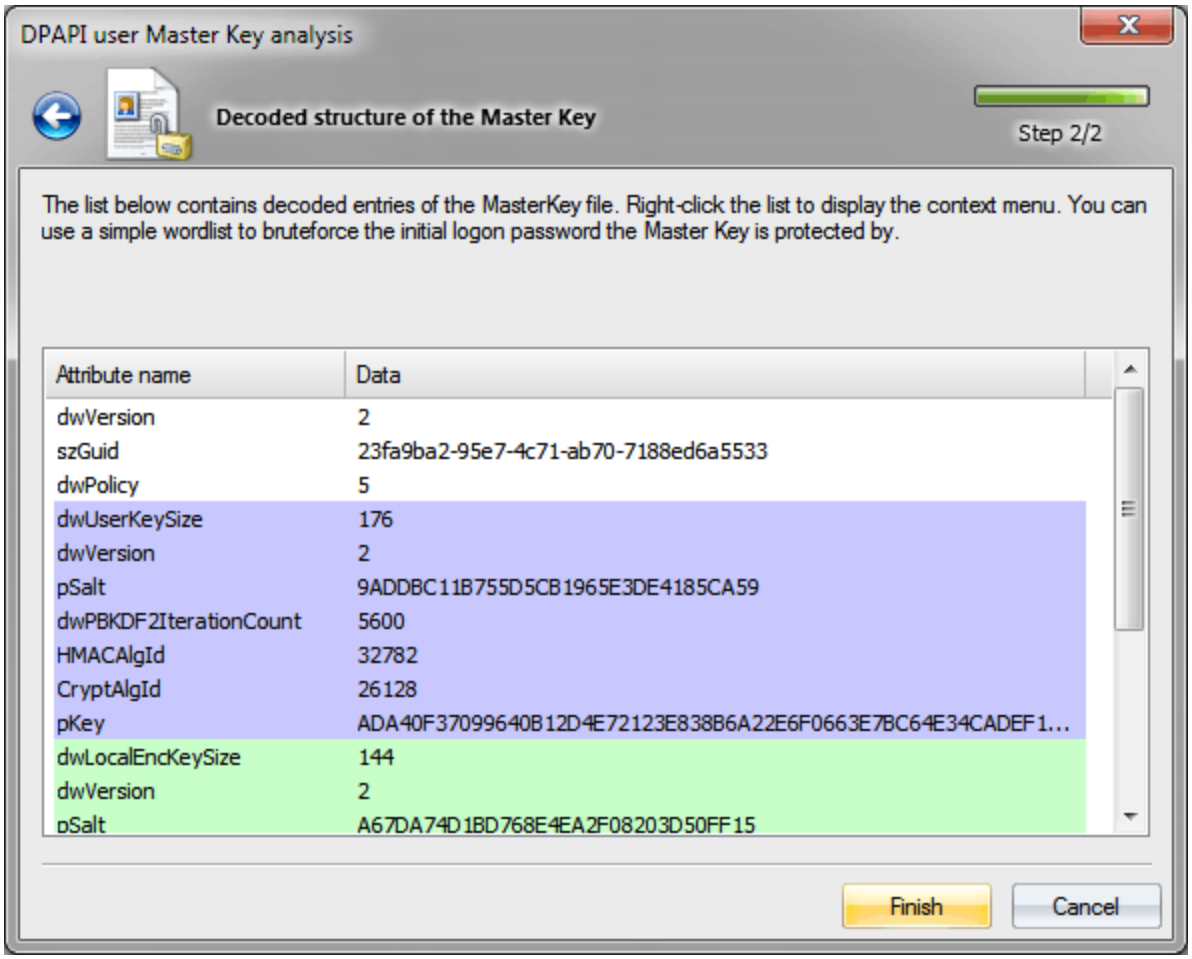
Master Key is 64 bytes of data, which are used as the primary key when decrypting a DPAPI blob. A user's Master Key is encrypted with the user's logon password.

Set path to Master Key file and specify user SID, which the program normally calculates automatically from the specified path.



All of that user's Master Keys are located in **%APPDATA%\Microsoft\Protect\%SID%**. For example, C:\Users\John\AppData\Roaming\Microsoft\Protect\S-1-5-21-2897849034-3956381361-16091305341-1001\23ab9bc1-9397-4cb1-ab74-7166ed6a8713
The system's Master Keys are stored in **%SYSTEMDIR%\Microsoft\Protect** folder.

Analyzing Master Key



Master Key file is a binary structure, which consists of a service header and four slots, namely: the actual user's Master Key, local encryption key (for unprotecting local backup key), local backup key (in Windows 2000) or CREDHIST GUID (in Windows XP and higher) and domain backup key.

The Master Key structure list consists of attribute names (i.e. binary fields) and values that corresponds with them. Each section is uniquely colored:

- field with header attributes
- slot with user's Master Key attributes
- slot with Local Encryption Key attributes
- slot with Local Backup Key or CREDHIST file's GUID attribute
- slot with Domain Backup Key attributes

Now, a little more detail.

Header attributes

- **dwVersion** - Master Key file version.
- **szGuid** - Master Key textual GUID. It normally matches the file name.
- **dwPolicy** - various flags. For example, if bit 3 is set, the program uses the SHA1 password hash when decrypting user's password; otherwise, it uses MD4. Thus, in Windows 2000 this bit is always cleared. A set bit 2 tells us that backup is require for the Master Key.

User's Master Key attributes

- **dwUserKeySize** - current slot length.
- **dwVersion** - data structure version. Version 1 implements only attribute with salt.

- **pSalt** - pSalt - salt, i.e. 16 random bytes of data, involved in the decryption of the Master Key and preventing data attacks using rainbow tables.
- **dwPBKDF2IterationCount** - iterations in the PBKDF2 encryption key generation function.
- **HMACAlgId** - hashing algorithm identifier.
- **CryptAlgId** - encryption algorithm used.
- **pKey** - user's encrypted Master Key.

Local Encryption Key attributes
- **dwLocalEncKeySize** - current slot length.
- **dwVersion** - data structure version. Win2K uses only one attribute with salt.
- **pSalt** - salt.
- **dwPBKDF2IterationCount** - iterations in the PBKDF2 encryption key generation function.
- **HMACAlgId** - hashing algorithm identifier.
- **CryptAlgId** - encryption algorithm used.
- **pKey** - encrypted Local Encryption Key, used for decrypting Local Backup Key in Windows 2000.

Local Backup Key attributes (Windows 2000)
- **dwLocalKeySize** - current slot length.
- **dwVersion** - data structure version.
- **pSalt** - salt.
- **pKey** - encrypted Local Backup Key.

CREDHIST file's GUID attributes (Windows XP and higher)
- **dwLocalKeySize** - current slot length.
- **dwVersion** - data structure version.
- **guidCredHist** - CREDHIST file binary identifier.

Domain Backup Key attributes
- **dwDomainKeySize** - current slot length.
- **dwVersion** - data structure version.
- **pSalt** - 16 random bytes of data, involved in the decryption of the Master Key and preventing data hacks using rainbow tables.
- **dwPBKDF2IterationCount** - iterations in the PBKDF2 encryption key generation function.
- **HMACAlgId** - hashing algorithm identifier.
- **CryptAlgId** - encryption algorithm used.
- **pKey** - encrypted Domain Backup Key. Its decryption requires the domain controller RSA private key, stored in Active Directory database.

To decrypt user's Master Key, you must know that user logon password. From the context menu, you can check the password for that Master Key and even try to guess one using a dictionary. However, don't flatter yourself too much. While in Windows 2000 the search speed is ranged in tens and even hundreds of thousand passwords per second, in Windows 7 the count goes by single items. See the table below (the speed is measured for a single-core of CPU Intel Q8400 2.66GHz).

Operating System	Encryption algorithm	Hash function	PKCS#5 PBKDF2 rounds	Password check speed (p/s)
Windows 2000	RC4	SHA1	1	**95000**
Windows XP	3DES	SHA1	4000	**76**
Windows Vista	3DES	SHA1	24000	**12**
Windows 7	AES256	SHA512	5600	**10**

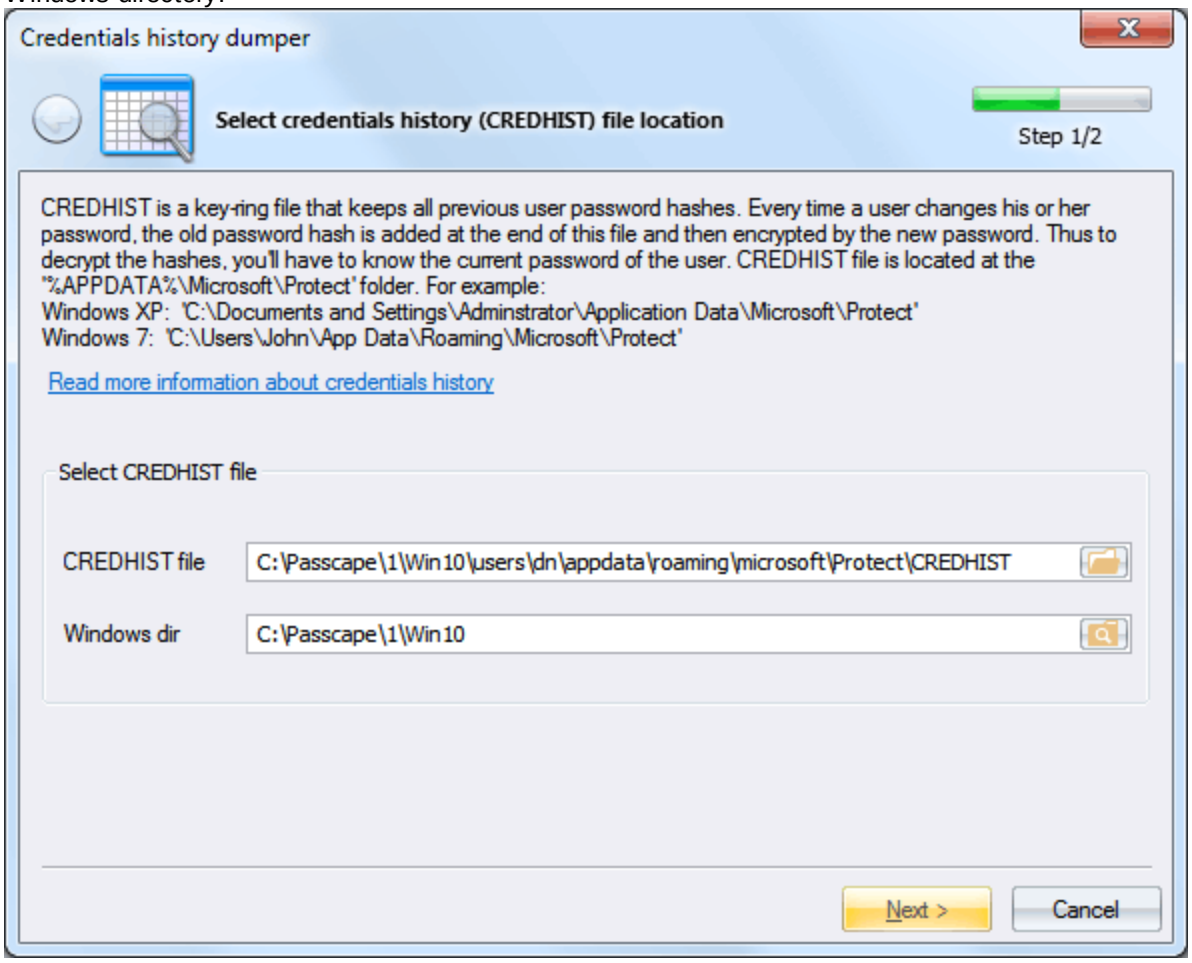
2.7.4.5.5 Dump user credentials history hashes

Due to peculiarities of DPAPI implementation, in order to guarantee the successful decryption of all DPAPI blobs, Windows must store all user's previous passwords in the system. User's password history is located in the following file:
%APPDATA%\Microsoft\Protect\credhist

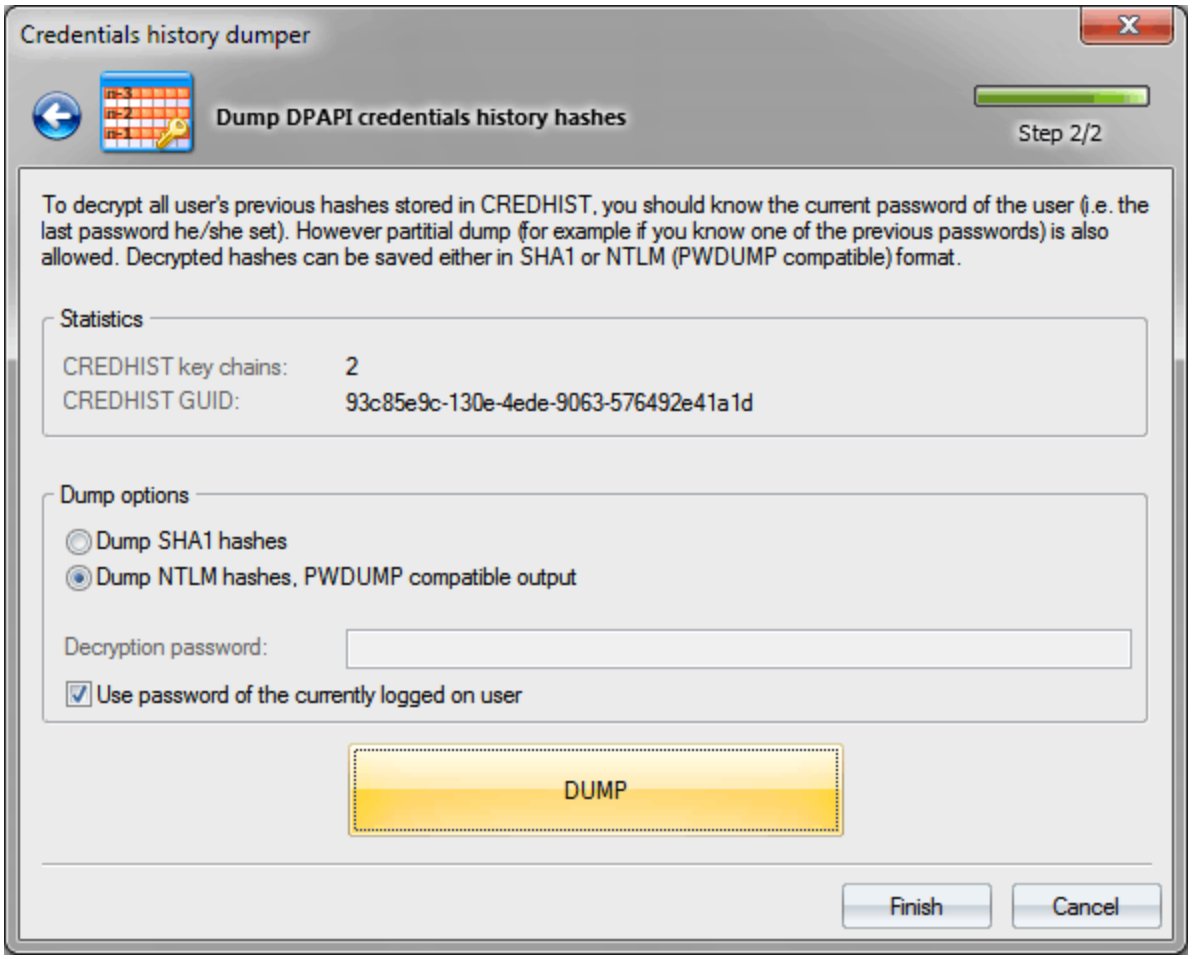
All user's older passwords (along with certain service data) are stored as pairs of hashes: **SHA1** and **NTLM**. Moreover, in order to decrypt the last pair, you must know the hash of user's current password, to decrypt the previous hashes, you need the last decrypted pair, and so on, along the line.

Windows Password Recovery is the world's first utility, which allows to decrypt password history hashes from CREDHIST files.

To do so, on the first step of the application's wizard, specify the path to your CREDHIST file and Windows directory.



Then you can decrypt and save hashes from CREDHIST to a textual PWDUMP-like file, if saving as **NTLM** is selected, or to a plain-text file, if the **SHA1** hash format is selected.



It is important to know that in order to decrypt CREDHIST hashes you must know user's current password. If you are decrypting CREDHIST of a currently logged on user, make sure to set the respective option. In this case, you will not have to enter the decryption password, it will be retrieved from the system cache.

The program supports partial dump of history hashes. That means that if user's current password is unknown, but at least one of the older passwords is available, the program can decrypt the passwords the user used earlier, i.e. before that old password was entered.

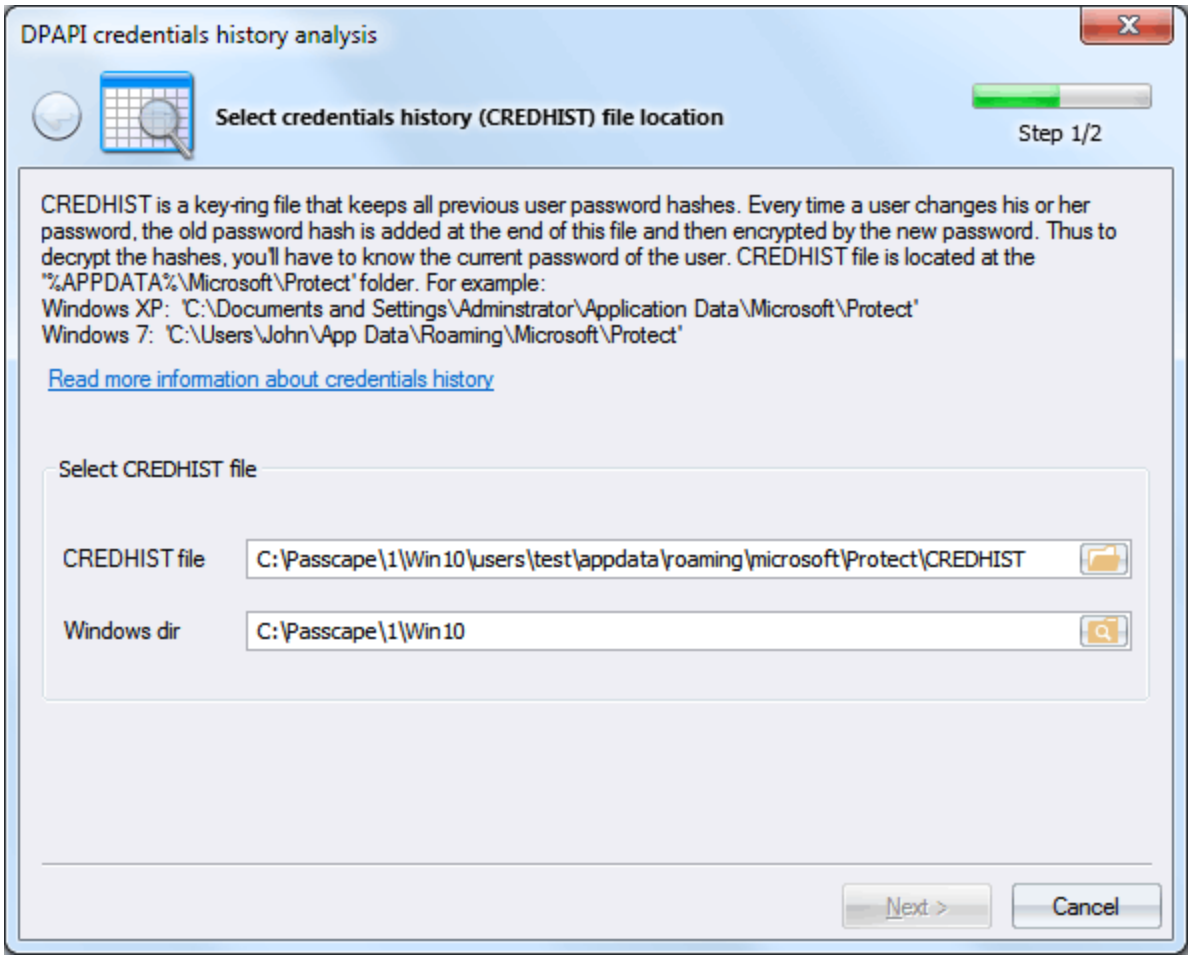
Be aware, in Windows 8 and higher OSes the dumped hashes for LiveID accounts are not correspond to those ones derived from LiveID logon passwords.

2.7.4.5.6 Analyse credential history

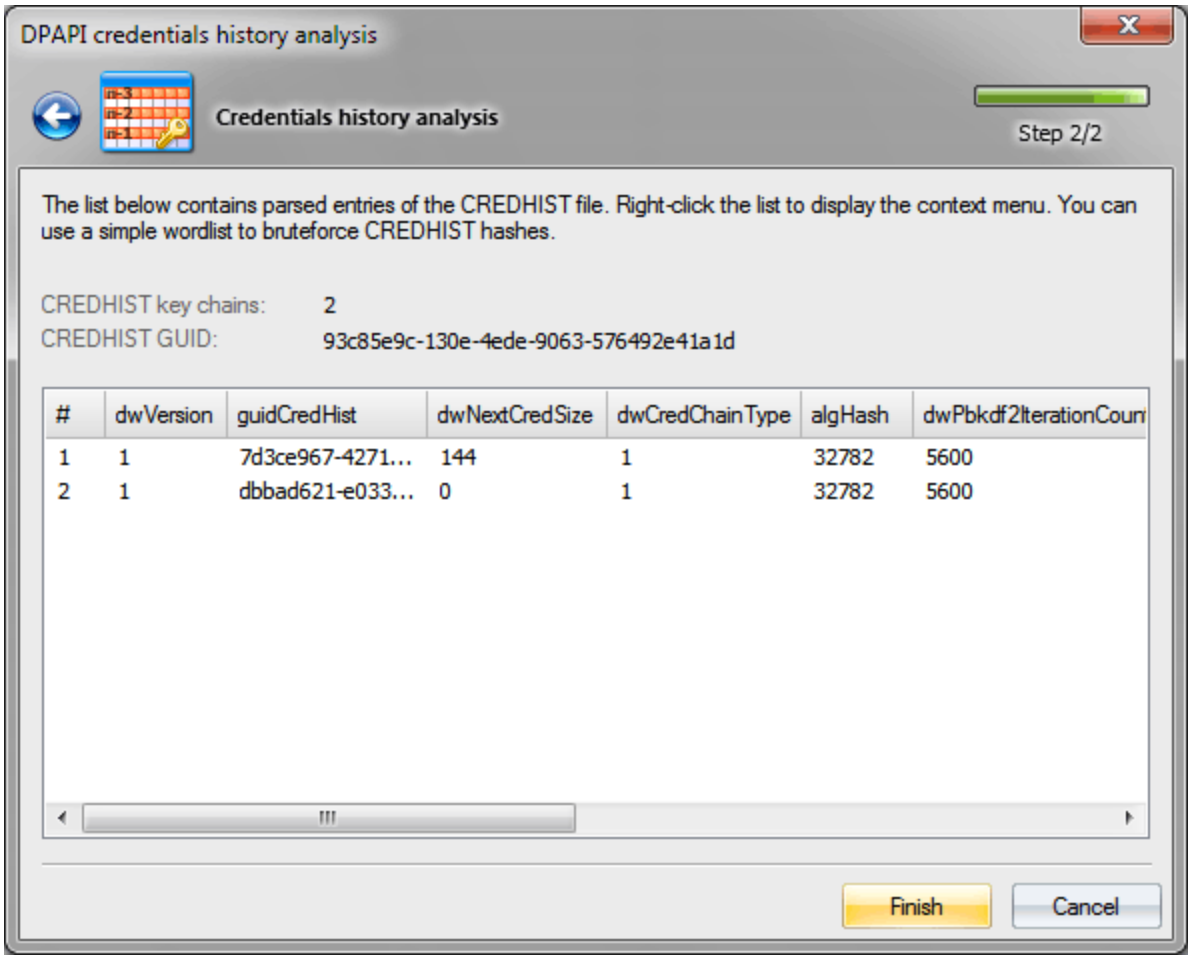
CREDHIST is a password history file, made out as a chain, where each link represents user's older password hashes. Each time user changes the password, the old password hash is appended to the file and encrypted with a new password. Therefore, to decrypt all the hashes in a chain, you must know user's current password.

Along with hashes, the chains store other service data, which is also analyzed by this utility.

Select CREDHIST file



And proceed to analyzing its content



On the screenshot, you can see that the CREDHIST identifier is 93c85e9c-130e-4ede-9063-576492e41a1d. This is the identifier (GUID) all user's Master Keys in the context of the data owner are attached to. The number of links in the hash chain is 2.

The list below contains all attributes and their values for each link of our CREDHIST.

Attribute description

- **dwVersion** - data structure version
- **guidLink** - current link unique identifier
- **dwNextLinkSize** - next link size
- **dwLinkType** - link type
- **algHash** - hashing algorithm used when decrypting the link
- **dwPbkdf2IterationCount** - iterations in the PKCS#5 PBKDF2 key generation routine
- **dwSidSize** - owner security descriptor (SID) size
- **algCrypt** - encryption algorithm
- **dwShaHashSize** - SHA1 hash size
- **dwNtHashSize** - NTLM hash size
- **pSalt** - salt used in the encryption
- **sidUser** - data owner SID
- **pShaHash** - SHA1 hash
- **pNtHash** - NTLM hash

To guess the original CREDHIST password, right-click on the attributes and then select '*Use wordlist to*

check password...' on the context menu that appears. You can validate password for both currently selected and all the records. The validation time increases proportionally to the number of the records (i.e. links).

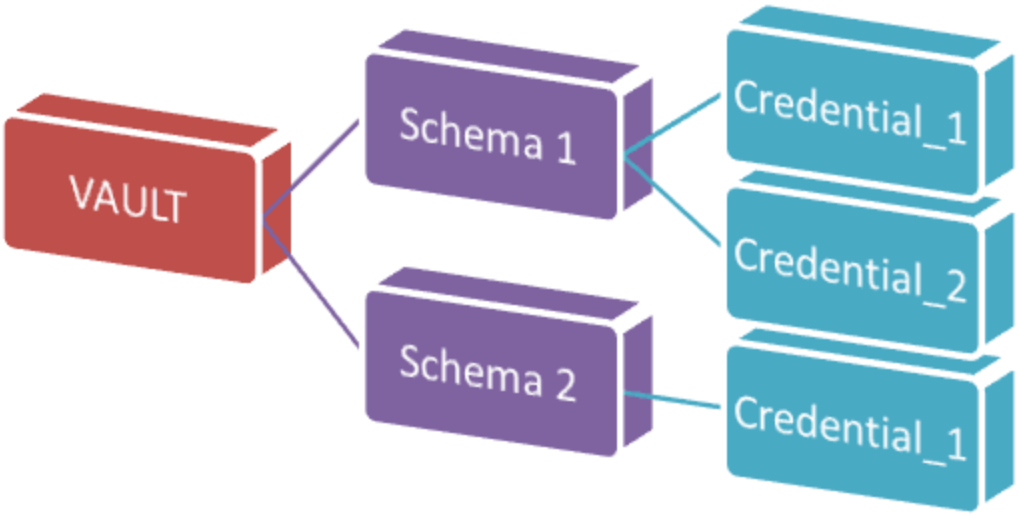
See the original CREDHIST password search speed comparative table. The speed is measured for a single-core of CPU Intel Q8400 2.66GHz for default OS configurations (for example, in Windows 7 the number of iterations in PBKDF2 may differ).

Operating System	Encryption algorithm	Hash function	PBKDF2 counter	Password check speed (p/s)
Windows XP	3DES	SHA1	4000	**76**
Windows Vista	3DES	SHA1	24000	**12**
Windows 7	AES256	SHA512	5600	**10**

2.7.4.6 Windows Vault Explorer

What is Windows Vault

Windows Vault is a protected storage for user or system secrets, passwords, network keys, web password and other personal information. Data stored in Windows Vault is structured and represents a set of records belonging to a certain Vault schema (see pic. below).



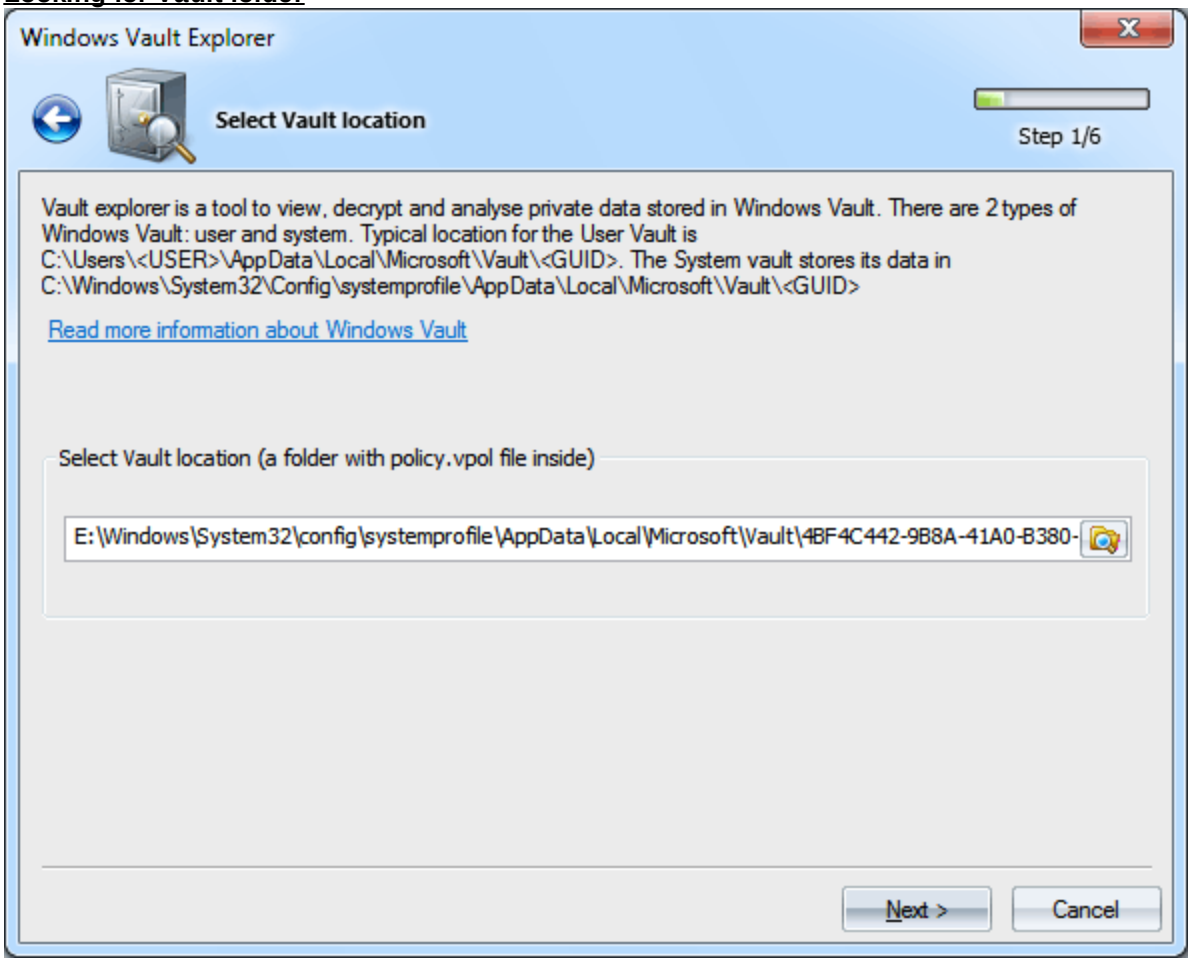
On the physical level, Vault is a disk-based folder with a set of the following files:
Policy.vpol - set of encryption keys for Vault records (credentials). These keys can be protected using two basic methods: either using DPAPI or using a specific user password. The latter protection method is not used in Windows 8 and currently is not supported by the software.
<GUID>.vsch - Vault schema that contains data description, flags and other system information.
<GUID>.vcrd - Vault credential that stores the original encrypted data associated with a certain schema. The data may consist of and normally consists of several fields. Description of the fields is stored in <GUID>.vsch.

Windows Vault Explorer

Windows Vault Explorer is a utility for offline analyzing and decrypting Vault credentials. The decryption Wizard splits the entire process into the following steps:
1. Looking for Vault folder

2. Looking for user's or system's Master Key
3. Setting registry files and other information necessary for decrypting the Master Key
4. Selecting Vault Schema
5. Looking for Vault records belonging to selected schema
6. Decrypting selected Vault credential

Looking for Vault folder



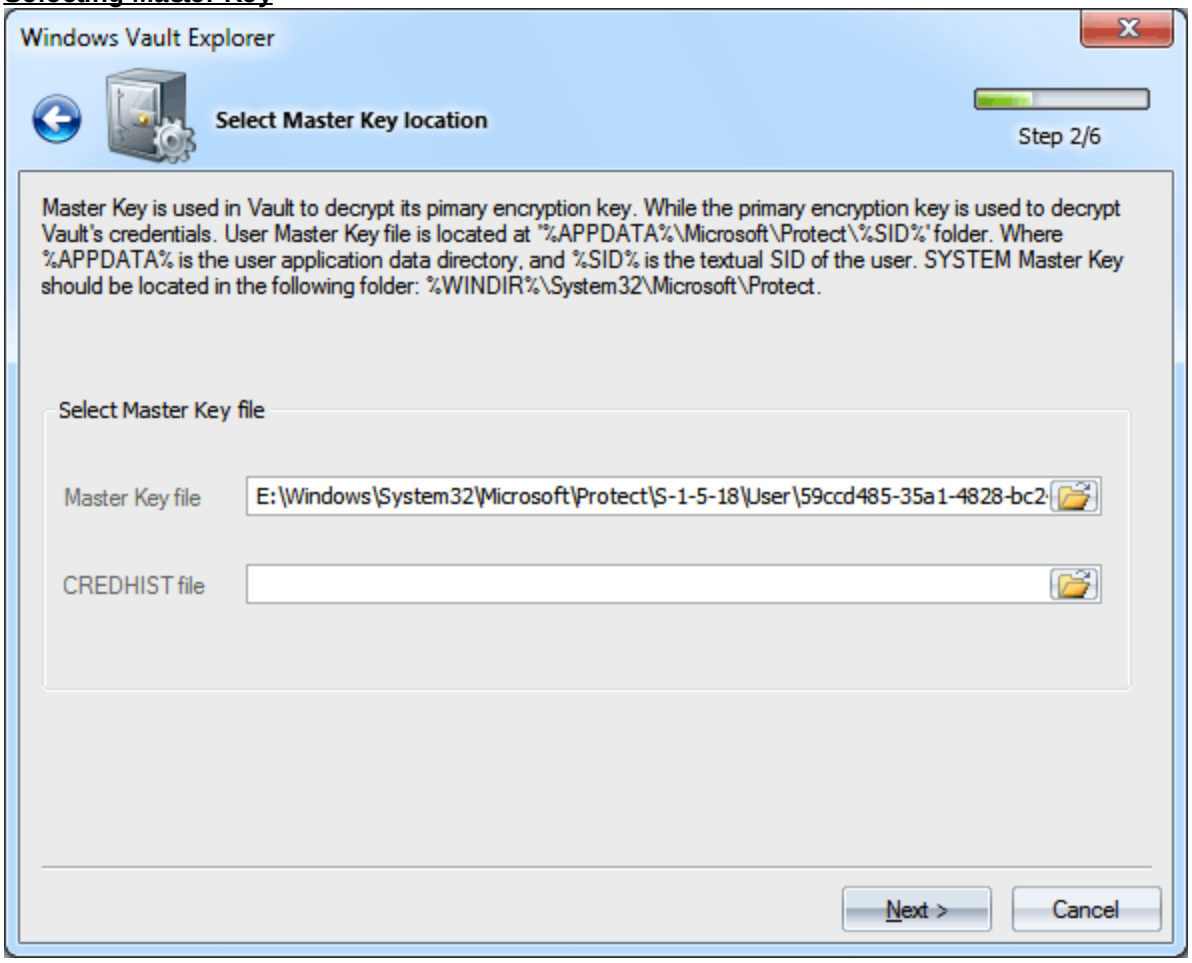
There are currently two types of Vault storage: system and user. The user Vault storage can be located in the following folders:
<USER_APP_DATA>\Microsoft\Vault\<GUID>
<USER_LOCAL_APP_DATA>\Microsoft\Vault\<GUID>
For example,
C:\Users\John\AppData\Local\Microsoft\Vault\18289F5D-9783-43EC-A50D-52DA022B046E
C:\Users\Helen\AppData\Roaming\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704DDB28

The default location of the system Vault storage is:
<SYSTEM_APP_DATA>\Microsoft\Vault\<GUID>
<SYSTEM_LOCAL_APP_DATA>\Microsoft\Vault\<GUID>
<PROGRAM_DATA>\Microsoft\Vault\<GUID>
For example,
C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704DDB28

C:\Windows\System32\config\systemprofile\AppData\Roaming\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704DDB28
C:\ProgramData\Microsoft\Vault\AC658CB4-9126-49BD-B877-31EEDAB3F204

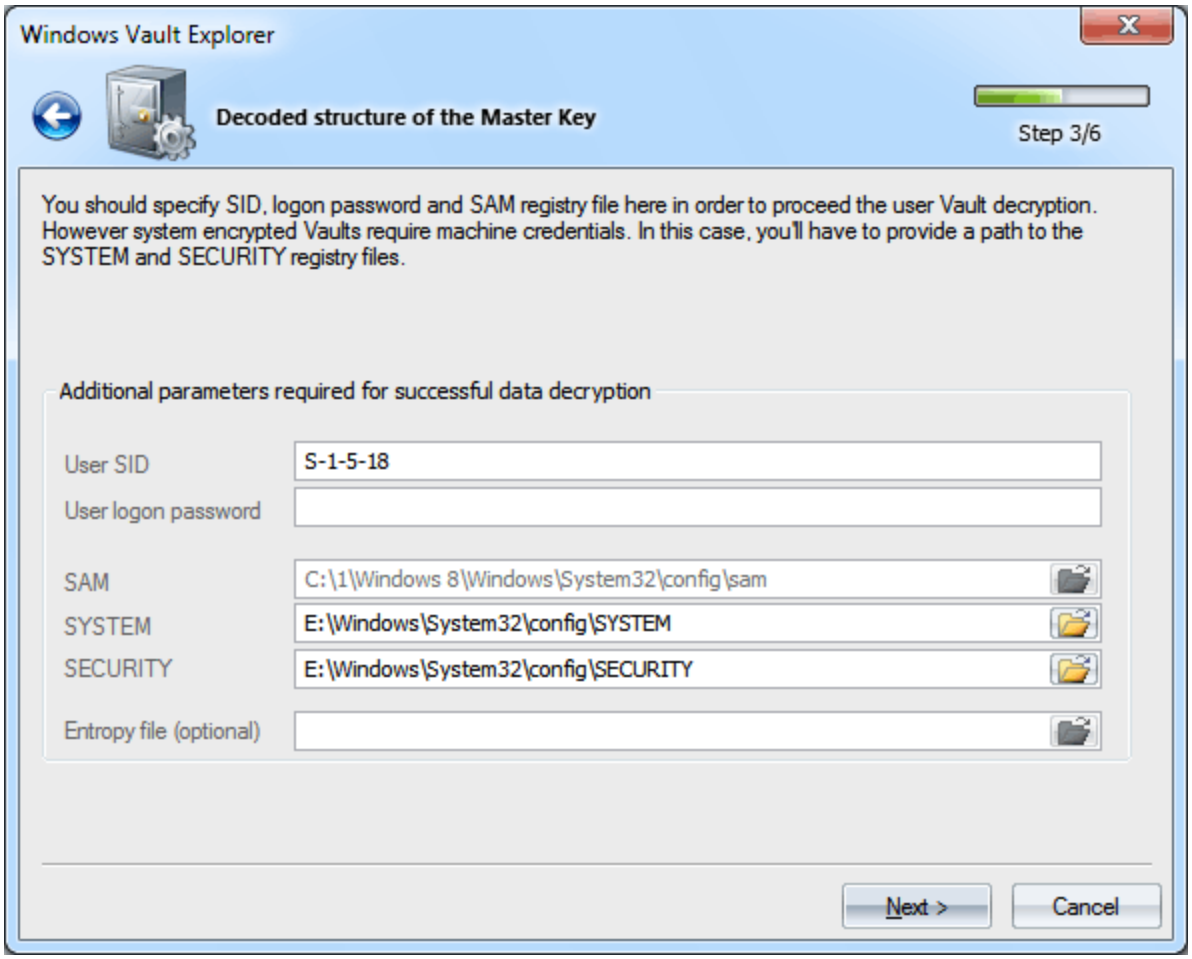
Note that some of the specified folders have the system attribute set on, which makes these folders hidden.
Windows has VaultCmd.exe utility for creating and managing your own Vault storages.

Selecting Master Key



Once a certain Vault folder is selected, you need to specify path to the Master Key used in the protection of the Vault encryption keys. The user's Master Key always resides in the folder **%APPDATA%\Microsoft\Protect\%SID%**, and the system account's Master Keys are stored in **%SYSTEMDIR%\Microsoft\Protect**. It must be noted that there could be a number of Master Keys, while a specific object could be decrypted using only one key, the name of which is stored in the Policy.vpol file. When searching for the Master Key, the program can filter out unnecessary names.

Decrypting Master Key

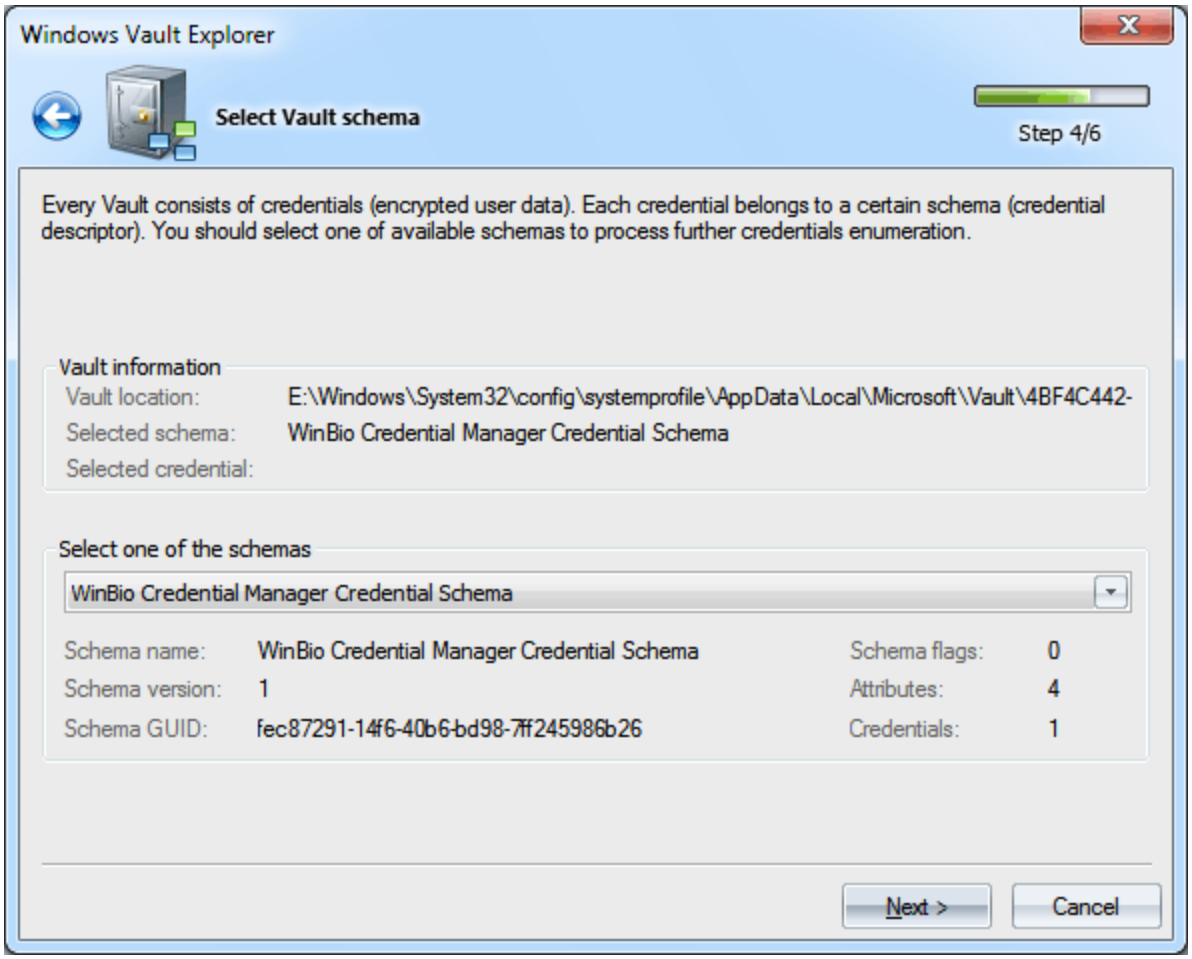


To decrypt a user's Master Key, you need to provide at least two parameters: the user's logon password and his security identifier (SID), which is normally included in the path to the Master Key. The program finds user's SID automatically. If that hasn't been done for whatsoever reason, set it up manually. To decrypt the system's Master Key, we don't need to specify the password; the program will extract all the necessary information from the two registry files: **SYSTEM** and **SECURITY**.

In some cases, the decryption of the Master Key requires specifying path to the **SAM** registry file. That's the case only when the account of the data owner in Windows 8 has the **LiveID** type.

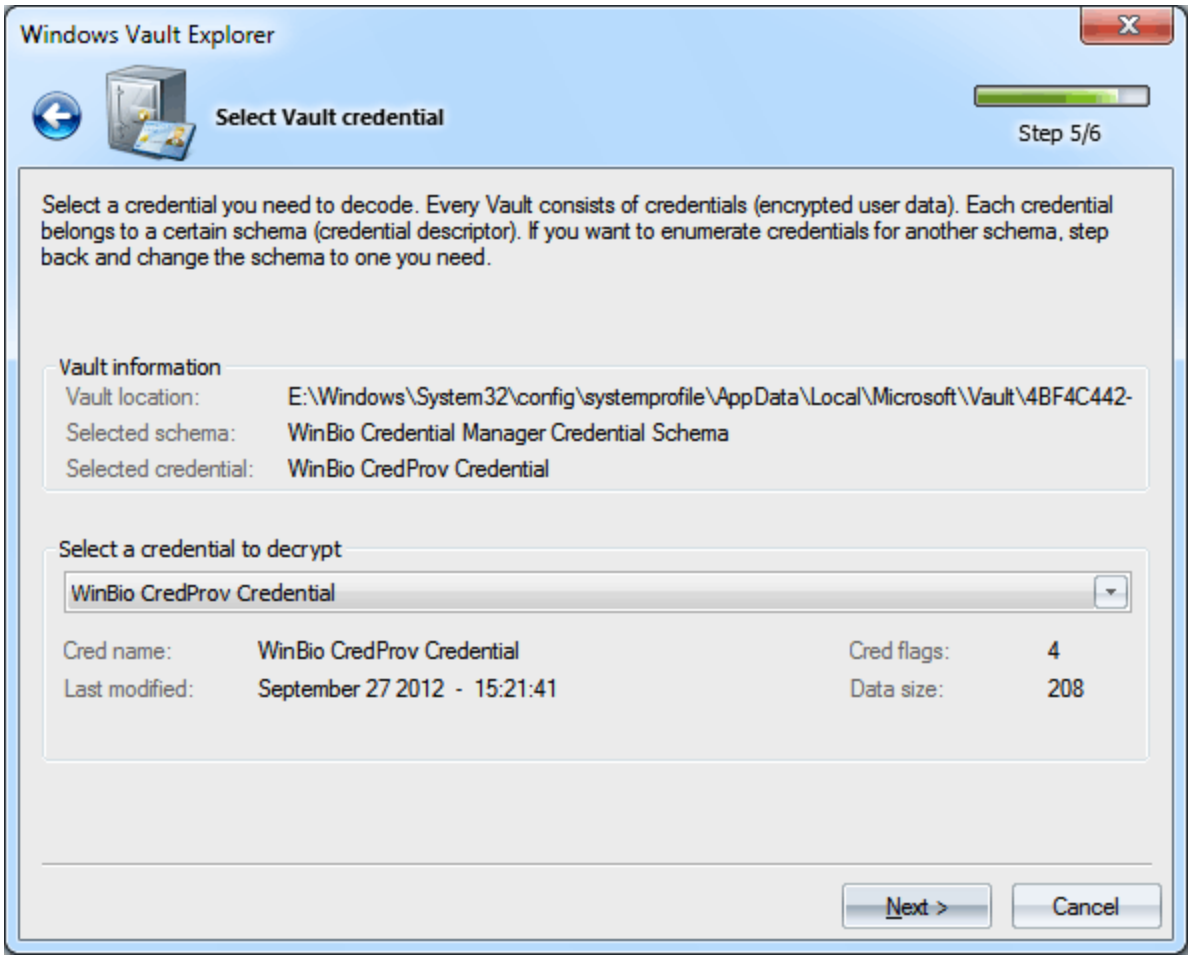
Windows Password Recovery starting with version 9.7 utilizes some vulnerabilities in DPAPI Master Key encryption. Thus to decrypt ANY Vault entry of a domain user, the owner logon password is not needed any longer.

Selecting Vault Schema



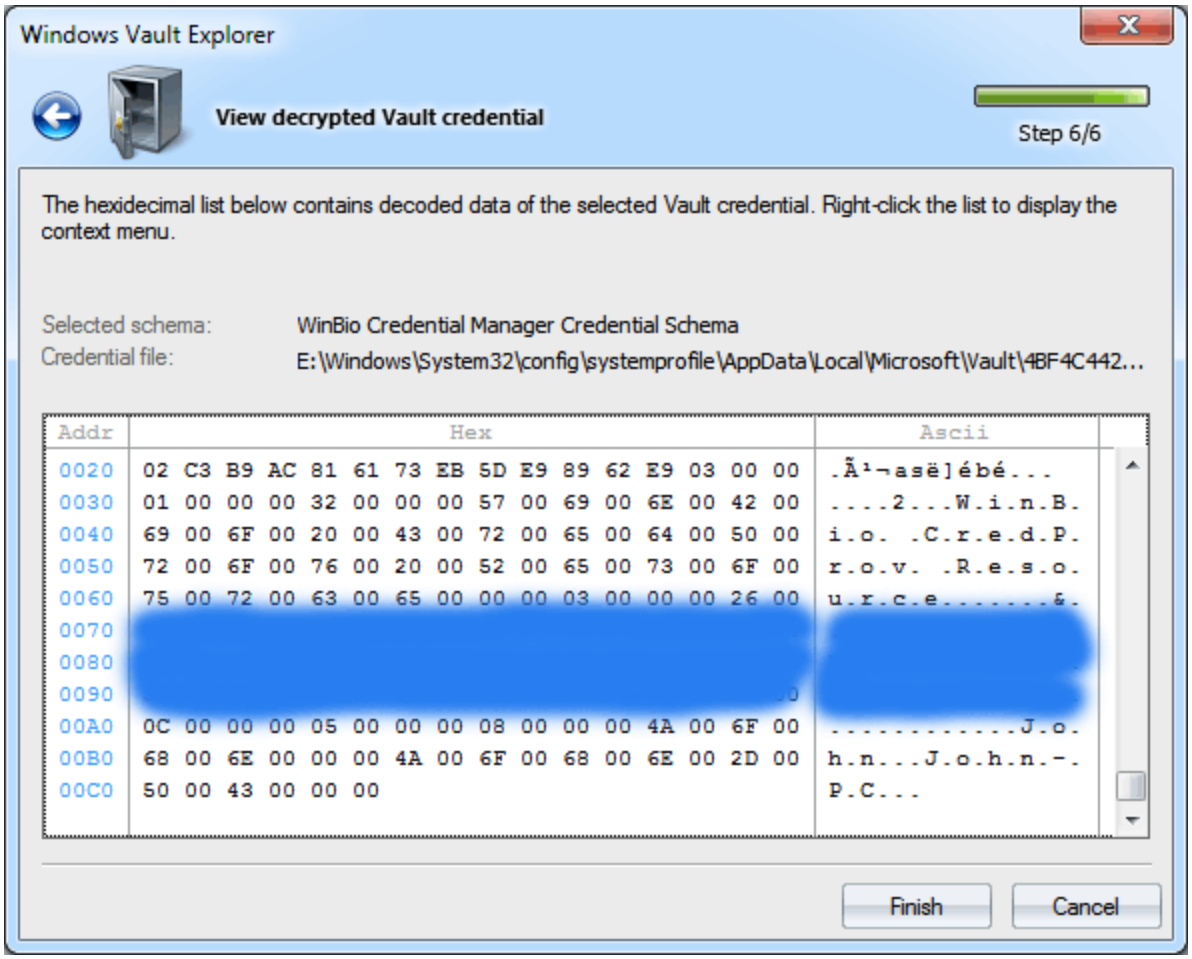
On the fourth step, if the previous ones passed successfully, the program prompts you to select one of the schemas belonging to our Vault from the dropdown list. Just below the list, we can see the general characteristics of the selected schema: its name, version, GUID, flags, number of attributes and credentials.

Selecting Vault credential



In a similar manner, select one of the credentials of interest that belongs to the schema we have selected during the previous step.

Decrypting Vault credential



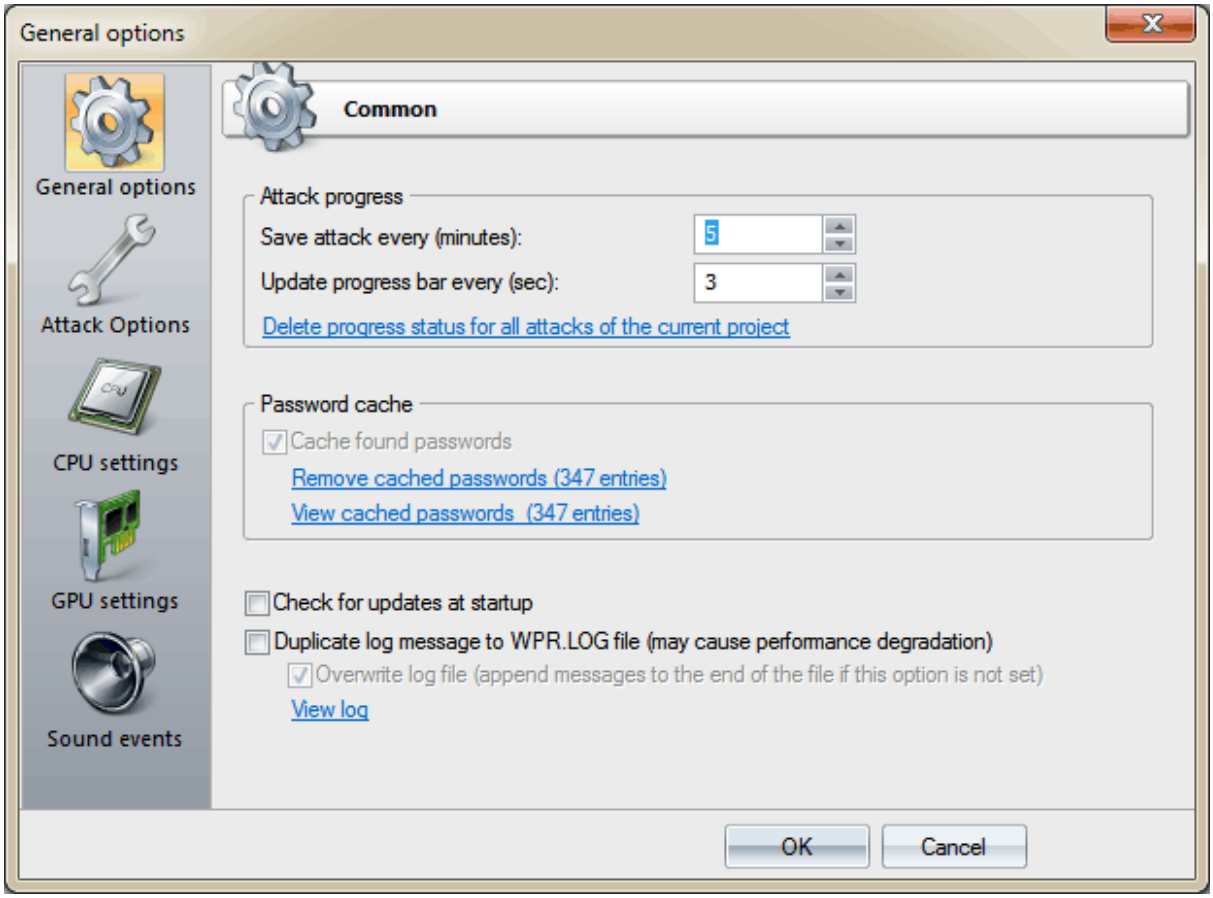
And at last the final step, where you can view the decrypted record, copy it to clipboard or save to file for further analysis. The figure shows decrypted plain-text password (it is clobbered) of the administrator account configured to logon using biometric information (fingerprint).

2.8 Settings menu

2.8.1 General settings

The general settings are divided into five parts.

2.8.1.1 General options



Attack progress
The first group of settings allows setting the save and update intervals for the current state of an attack. By default, an attack saves its state every 5 minutes (further on, you can resume the attack from the last saved point) and updates the screen every 3 seconds.

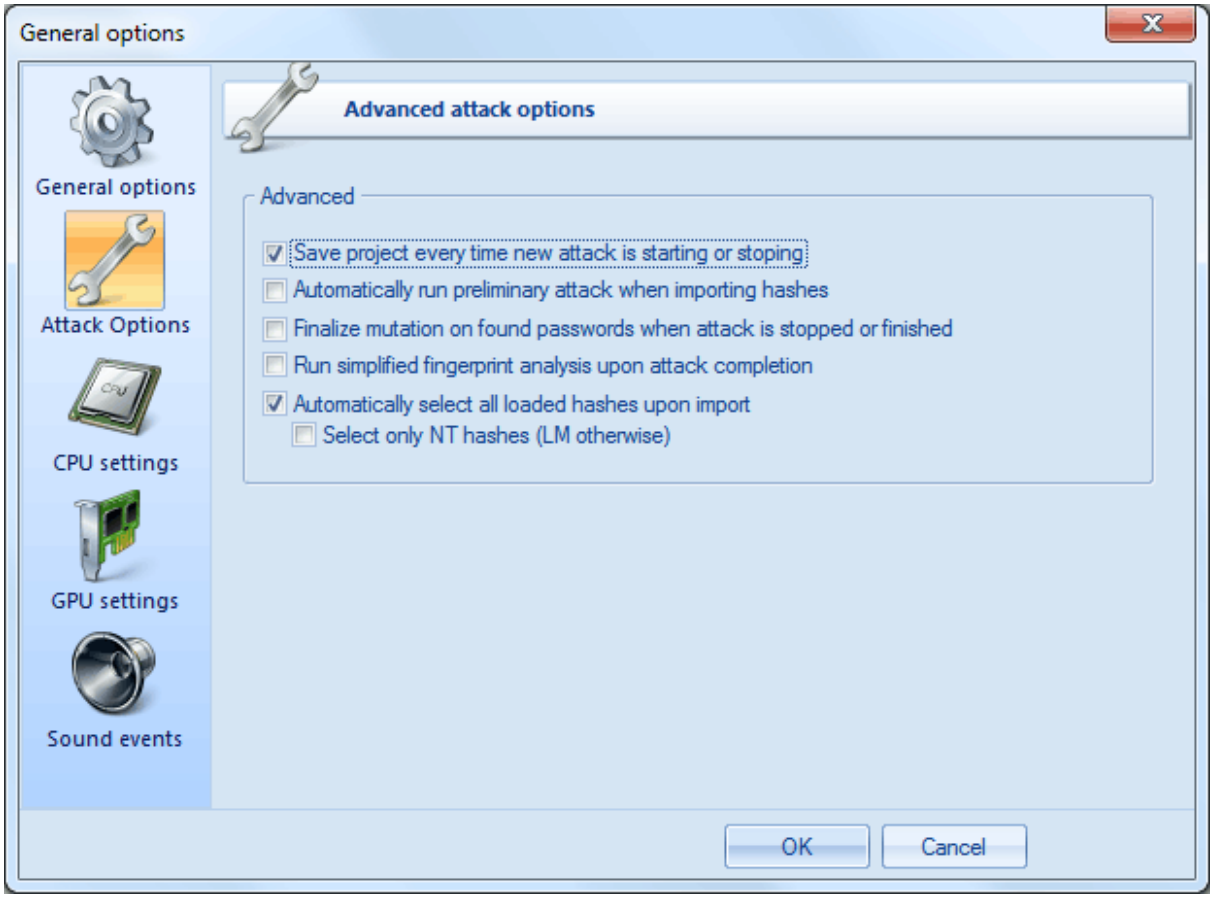
Password Cache
All passwords found by the program are cached by default. A very helpful thing that is engaged in many subsystems. For example, in the intellectual or preliminary attack. Deleting password cache is recommended in cases of the extreme need only. For example, when their number exceeded ten thousand. In this case, the search speed for some attacks can drop significantly.
Additionally, you can duplicate found passwords to text file. So even if the program fail unexpectedly or in case of sudden power failure, the found passwords guaranteed to be written to file.

Check for updates at startup - check if an update is available every time the program starts. The option works only if PC is connected to internet.

Duplicate log messages to wpr.log file - this option when set, writes all messages the log window holds to WPR.LOG file. Setting this option may cause performance degradation on big list of hashes because the wpr.log flushes its content to disk every time new message is arrived. It can however be helpful when the program stalls or works unstable. WPR.LOG is located in the program's installation directory.

Overwrite log file - overwrite the log file every time the program starts. Otherwise new messages will be appended to the end of the log file.

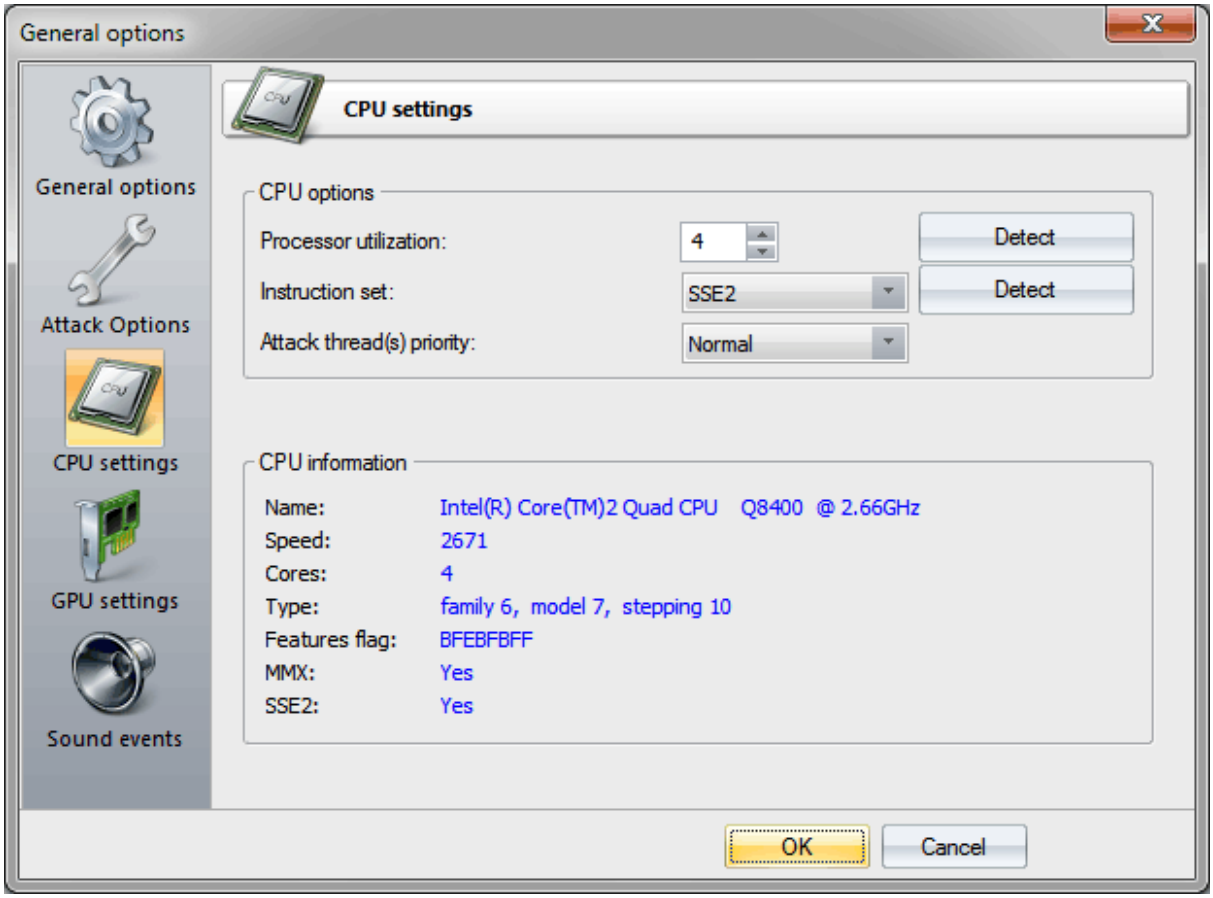
2.8.1.2 Attack options



Advanced

- *'Save project every time ...'* - setting this option will force the program automatically save project every time new attack is starting or stopping (including sub-attacks of a Batch attack).
- *'Automatically run preliminary attack when importing hashes'* - automatically launch preliminary attack upon import. This attack recovers extremely weak passwords within seconds.
- *'Finalize mutation on found passwords when attack is stopped or finished'* - activate password analysis and mutation module for found passwords after attack. This option can be extremely useful; for example, for recovering similar passwords.
- '*Run simplified fingerprint analysis upon attack completion*' - activate second analysis module. It launches on attack completion, creates new fingerprint dictionary out of found passwords, trying to retrieve more passwords. Useful on big list of hashes, history hashes, etc.
- *'Automatically select all loaded hashes upon import'* - automatically select entries to be searched after importing.

2.8.1.3 CPU settings

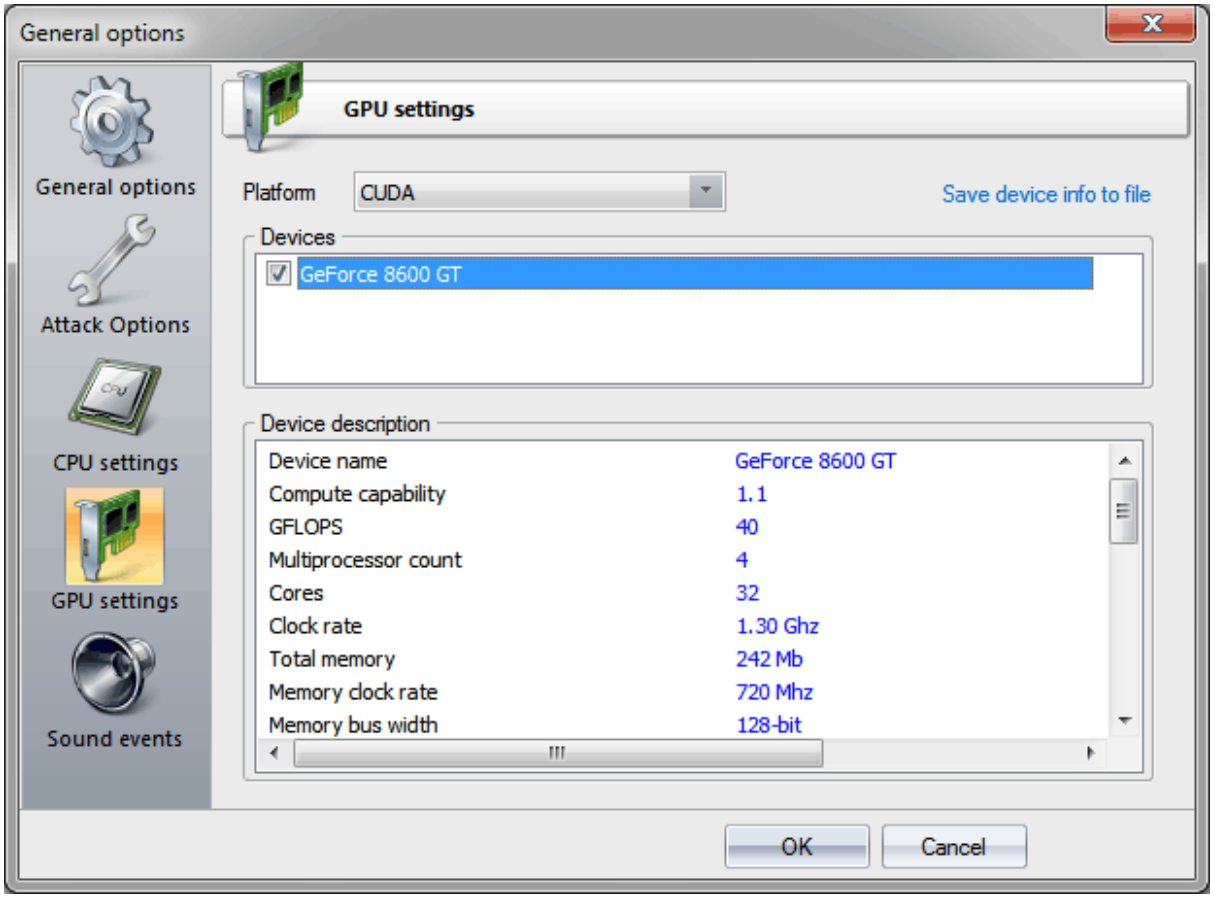


Since the majority of the attacks supports multithreading, you can set the number of search threads to be run simultaneously. In the majority of cases, it should match the number of cores in your CPU. However, if the CPU supports the Hyper-Threading technology, you can even double the number of search threads that run simultaneously.

The DES and MD4 search algorithms in Windows Password Recovery are optimized for three CPU architectures: X86, MMX and SSE2. Naturally, on CPU that support newer architecture, the search would run faster.

It is not recommended to set the attack priority above normal; otherwise, you may observe a considerable reduction of performance of the entire system.

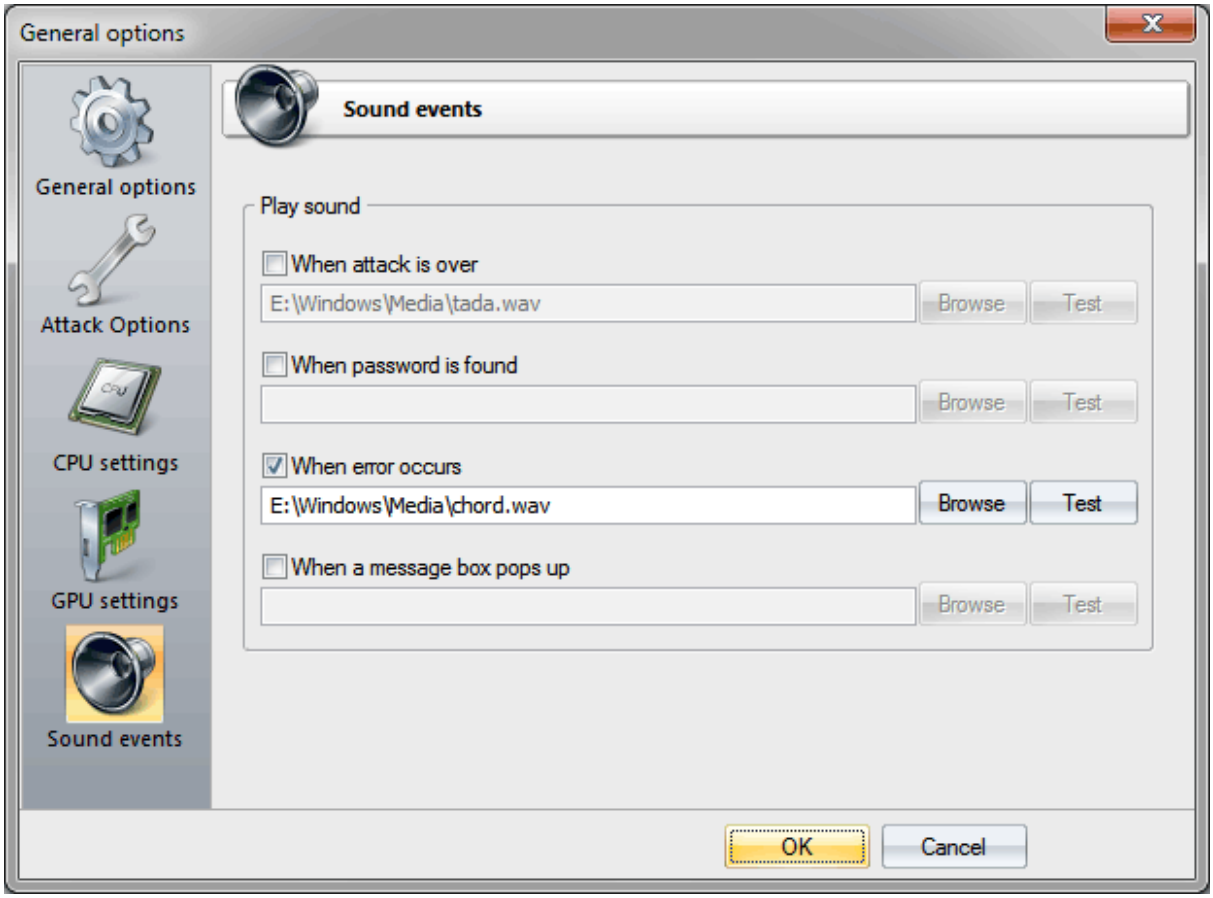
2.8.1.4 GPU settings



Before running an attack on a GPU, select it in the application's general settings by simply ticking the check box by the GPU name. All the main characteristics of the device are displayed in the property table.

The software supports NVidia (built on the CUDA platform) and AMD (built on OpenCL platform) GPUs.

2.8.1.5 Sound notifications

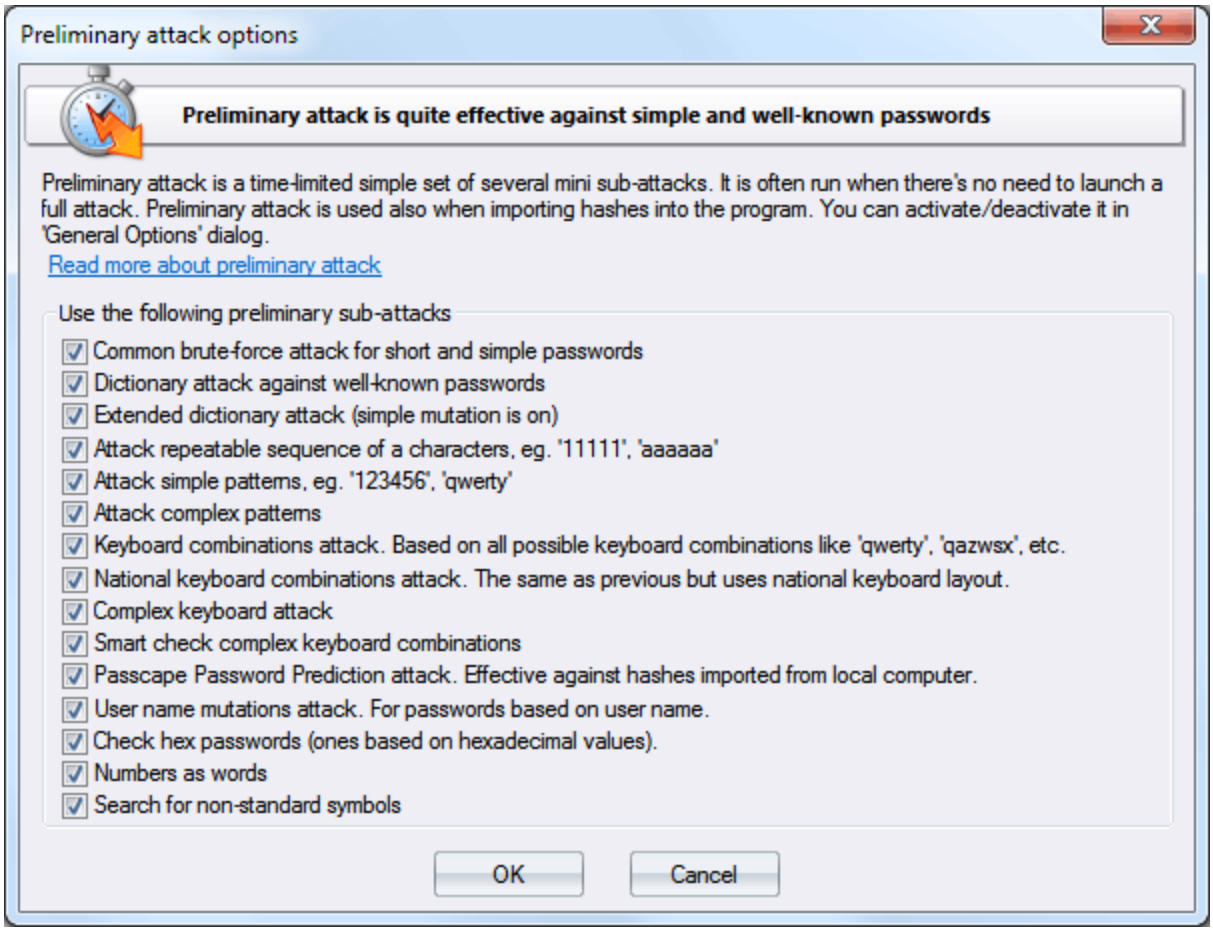


The software allows setting up sound notifications for certain events. For example, when the attack is over or when a password is found.

2.8.2 Attack Settings

2.8.2.1 Preliminary attack

Preliminary attack (developed in Passcape) is quite effective against short, simple, dictionary, repetitive, keyboard, etc. passwords and consists of several mini-attacks. Each mini-attack can be enabled/ disabled individually.



Preliminary attack run about 10-20 minutes or even faster. It consists of at least the following sub-attacks:

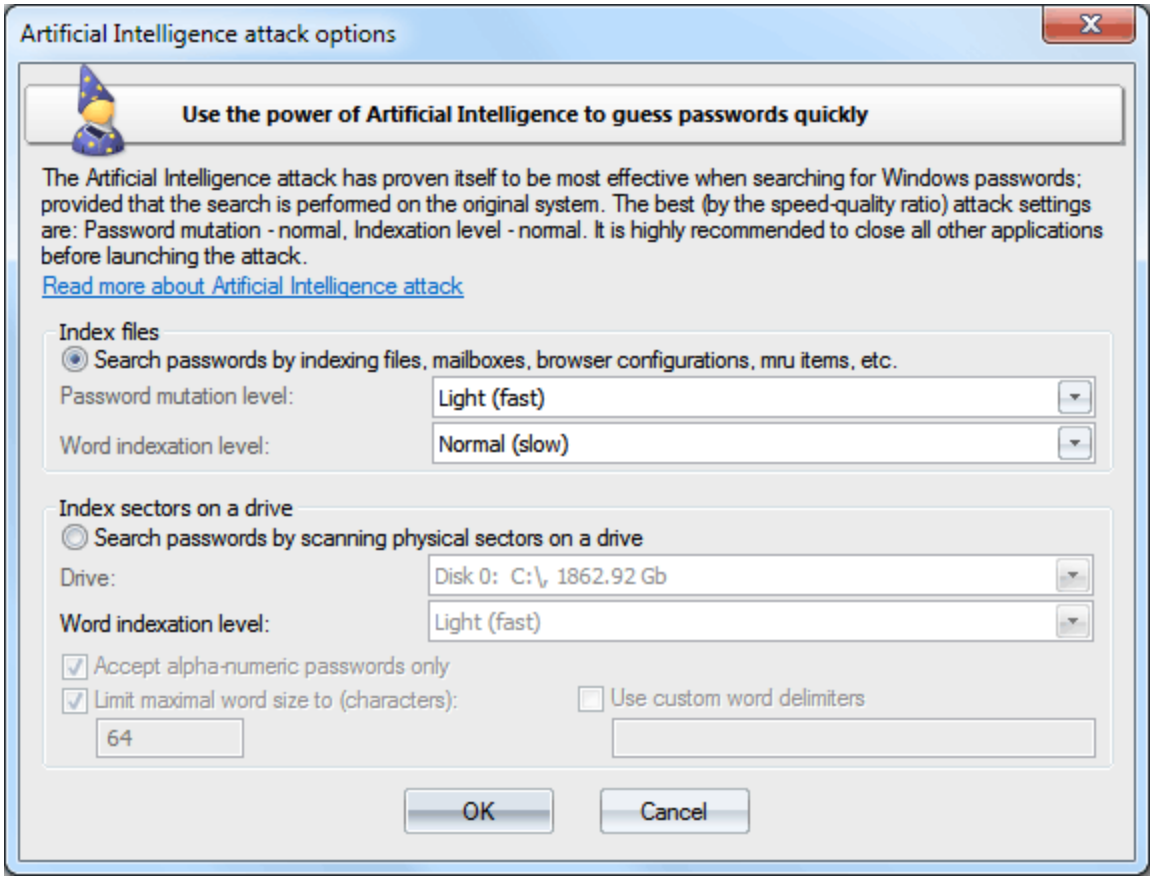
- Common brute-force attack. Performs several simple brute-force attacks based on predefined character sets.
- Simple dictionary attack. Fast check the password by verifying all words from a given dictionary.
- Extended dictionary attack. It's almost the same as above but with some smart mutation options set on.
- Attack on repeatables. Checking passwords as a repeatable sequence of a character. Eg. '1111111' or 'xxxxxxx'.
- Attack on simple patterns, like '123456' or 'qwerty'.
- Attack on complex patterns. The same as above, for compound patterns.
- Keyboard attack checks for keyboard passwords and all possible combinations. Eg. 'qwer', 'qazwsx', 'asdzxc', etc.
- National keyboard attack. The same as above, but checks passwords typed in national keyboard layout.
- Complex keyboard attack is the same as previous 2 attacks, for compound keyboard patterns.
- Passcape Password Prediction attack is the most complicated and state-of-art password prediction tool.
- Attack on name-based passwords.
- Attack on hex passwords (eg. 7A49F3).
- Attack passwords based on numbers (as words).
- Search for non-standard symbols and short passwords that were created using non-standard UNICODE symbols.

2.8.2.2 Artificial intelligence attack

Artificial Intelligence Attack is a new type of attack developed in our company. It is based upon a social engineering method and has never been implemented in password recovery applications yet.

This one is mostly used when the hashes are imported from the local computer. Intellectual attack scans the local computer, indexes and creates the list of found words and passwords, analyzes them, upon the results of the analysis produces user's preferences, performs the mutation of the found words and, based on all that, attempts to recover the passwords.

This attack allows, without resort to time-consuming and costly computations, to almost instantly recover certain passwords encrypted with hash functions. The basic idea behind the Artificial Intelligence attack is that an average user very often chooses similar words and word combinations or follows the same password generation rule when creating one's passwords. With that in mind, we could attempt to figure that rule out and pick the original password.



Although this sounds somewhat abstractive, in the reality the attack clearly splits into four successive steps.

1. Initiating the collection of private data. Here comes into action the password retrieval and indexation module, which looks for all available and hidden in the system passwords entered by user at any moment of time. Those include network access passwords, ICQ, email, FTP, Windows account passwords, server passwords, LSA Secrets, etc.
2. Launches the data collection and indexation module. During the execution of this step, we analyze the activity of the user (or all users, if the indexation module selected is different than Light) in the

system. Next, basing upon that, we generate the list of words - potential passwords selected from the text files, archives, internet browsers' history, email correspondence, etc.
3. Includes the semantic analysis module for the database of found passwords and the list of potential passwords.
4. On the final stage, the data analysis module will perform the mutation of the words and attempt to pick the passwords.

In the beginning of the attack, the program will search the system for all passwords it knows of. For that purpose, there are currently 32 mini modules for decrypting system, mail, browser, messenger, archive and other passwords. Then there goes the file and data indexation, along the course of which the program generates a potential attack dictionary. The third module breaks the passwords and words into pieces, out of which in the last module it will assemble new combinations for picking and guessing the original password.

In average, with the least indexation and mutation levels, the attack time may vary between 1 minute and 10-15 minutes, depending on the network activity of the user. On a home computer, the entire route normally takes not more than 2-3 minutes. Naturally, the more complex is the mutation and indexation level, the more efficient will be the search. However, reaching the topmost indexation and analysis level may take hours and even days, depending on the speed of the password validation algorithm and the number of users in the system.

The Artificial Intelligence attack has proven itself to be most effective when the search is performed on the original system. Only two options are available here: password mutation depth and word indexing level. The most preferred options for running a speedy attack are *Light:Light*. For a deeper (and at the same time slower) search, set these options to *Normal* or even *Deep*. The duration of an intellectual attack also depends on the configuration of your system, your network load, and other factors.

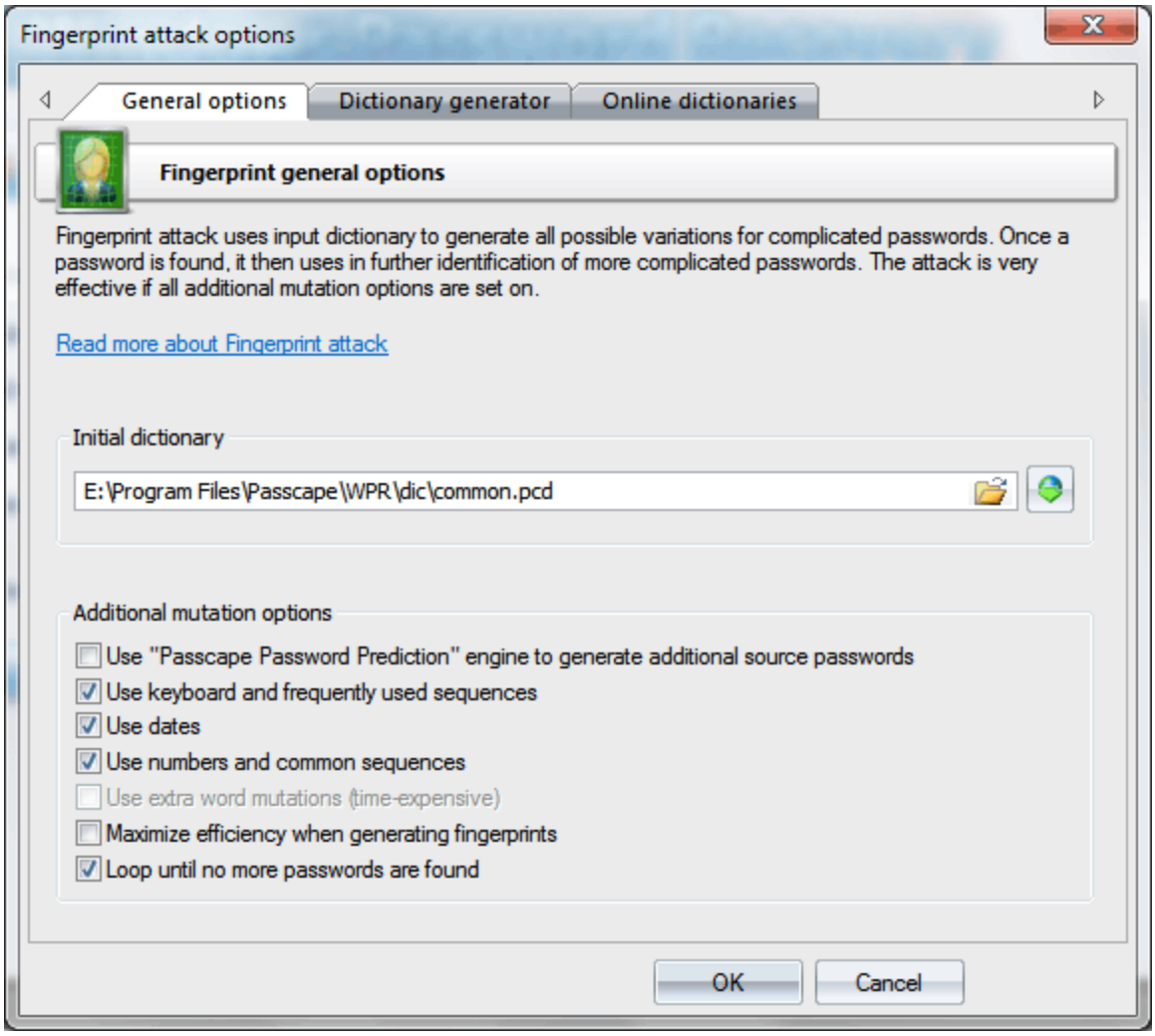
It is highly recommended to shut down all other programs before launching the attack. If your Artificial Intelligence attack runs very slow, you may need to remove your program's cached password (eg. total amount of the cached passwords exceeds 10000).

Windows Password Recovery version 9.5 now comes with a brand new feature which allows password searching by indexing raw sectors on selected drives. This feature works for both LM and NTLM hashes, looking for both ASCII and UNICODE passwords. You can change some advanced search options here. For example, '*Word Indexation level*' sets additional mutation on all found passwords. Be careful, walking through all sectors of the target drive with this option set to 'Hard' may take quite a time. Note that the sector-based scanning algorithm is not effective against drives which have a full-disk encryption set on. Like Bitlocker or TrueCrypt, for example.

2.8.2.3 Fingerprint attack

Fingerprint attack is a relatively new tool for recovering complex passwords, which could not be decrypted by other attacks. The idea of the attack is that here, to recover a password, we take neither individual words from the source dictionary, like in the dictionary attack, nor even word combinations, like in the combined attack, but so-called "fingerprints". Now, every source word from the dictionary is used for generating several fingerprints. If some password is found during the attack, it participates in generating new fingerprints, and the attack goes another round.

Before launching the attack, specify the source dictionary to be used for creating the fingerprint bank. The software comes with a dictionary, common.pcd, optimized for this attack, but you can use yours or download one off the Internet ('Online dictionaries' tab). There are no certain requirements to the dictionary, except one: the source dictionary must not be too large; otherwise, the attack will take significant time. You can use dictionaries with national passwords, if you suspect that the sought password contains characters in a national encoding.



Here is the way to generate fingerprints: first, break each word from the source dictionary into one-character passwords, then - into 2-character, etc. For instance, break the source word **crazy** into one-character fingerprints. We get:
c
r
a
z
y
Now, two-character:
cr
ra
az
zy
Next, three-character:
cra
raz
azy
And, finally, four-character:
craz
razy

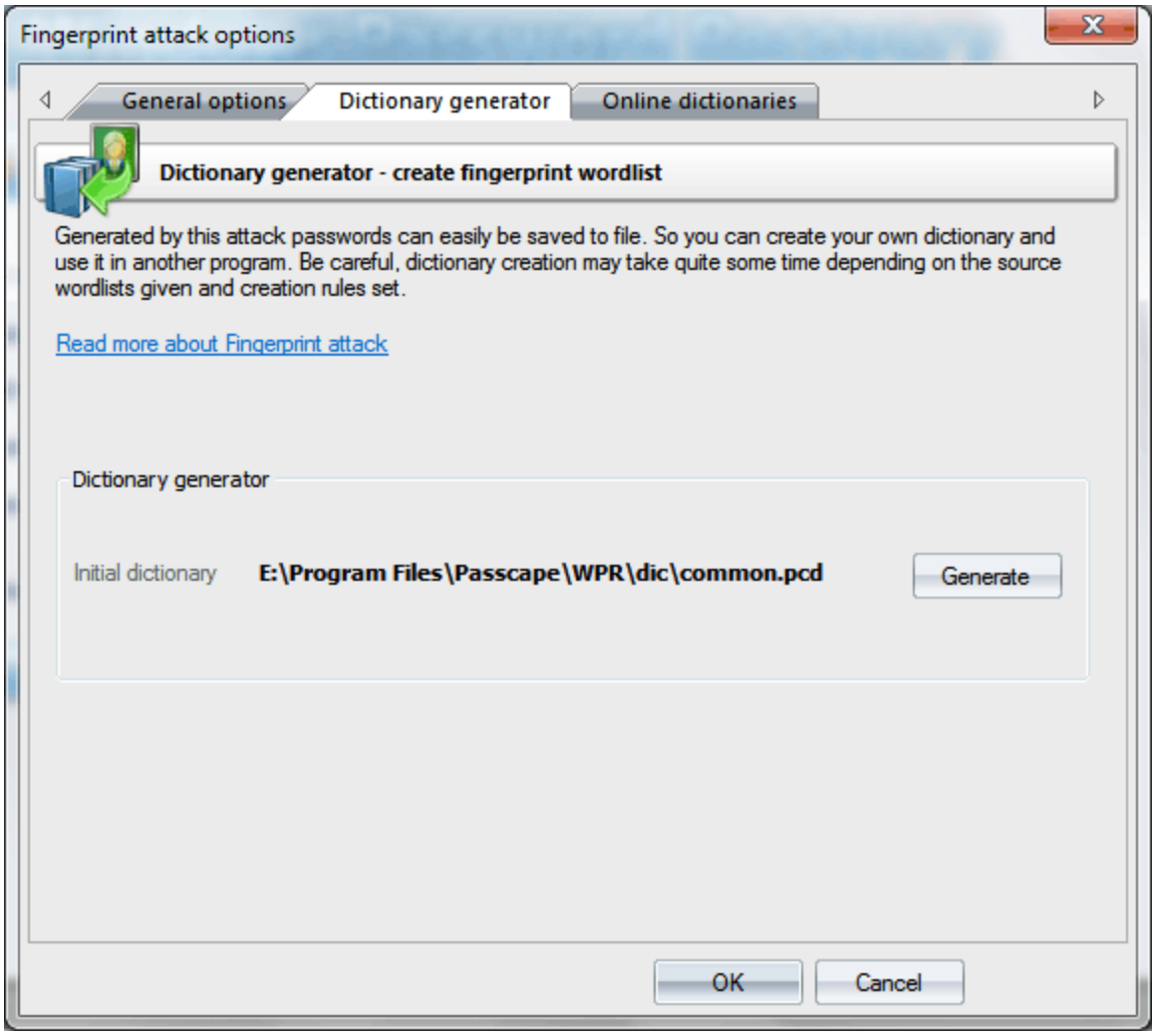
We have got 5+4+3+2=14 fingerprints, not counting the source word.
Repeat this for each word of the source dictionary. After this, all the fingerprints are dumped into a single database, naturally, discarding duplicates. We have got a database of fingerprints that would be used for checking passwords by gluing all the fingerprints with each other and finding the match.

The real fingerprint generation algorithm is much more sophisticated. Moreover, there is an option in the attack settings, **Maximize effeciency when generating fingerprints**, which uses a more sophisticated algorithm, which maximizes the efficiency (at the expense of speed) by generating additional fingerprints.

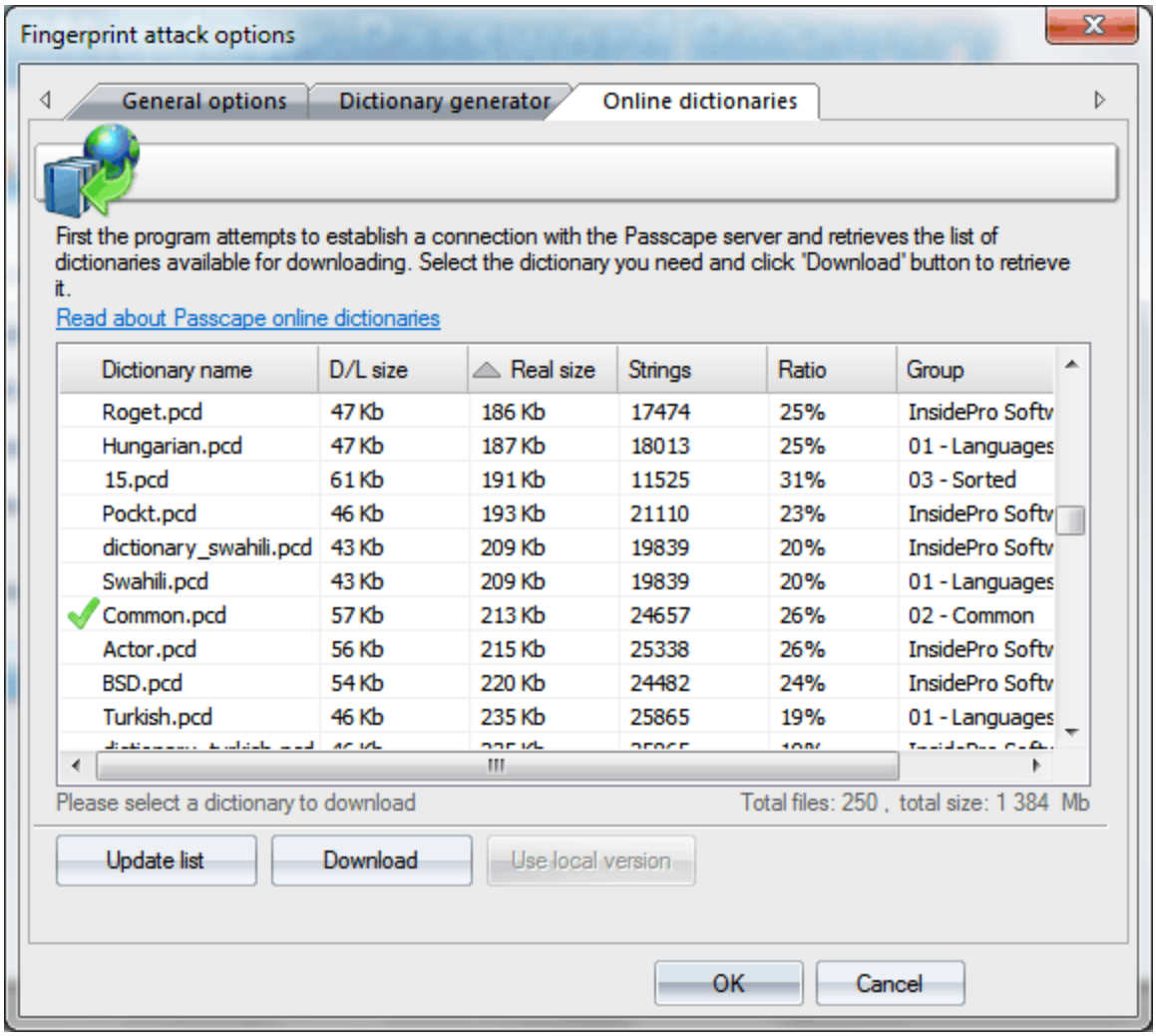
Let's take a look at the remaining options.
Use PPP engine to generate additional passwords - use passwords found in other attacks when generating fingerprints.
Use keyboard and frequently use sequences - add keyboard combinations and common sequences to fingerprint bank.
Use dates - add dates to fingerprints.
Use numbers and common sequences - use digits and simple combinations of letters.

The most careful attention should be paid to the option **Loop until no more passwords are found**. That is where fingerprint attack can really show itself off. Here is how it works: if at least one password is found during an attack, when the attack is over, the password participates in generating new fingerprints, and the attack runs again. This option works great on large lists of hashes and on password history hashes. However once the option is set, you will not be able to proceed the attack from the last saved position.

The second tab with the settings allows to create and record a custom dictionary using current options of fingerprint attack. Be careful; that dictionary may take up a lot of space on your computer's hard disk.



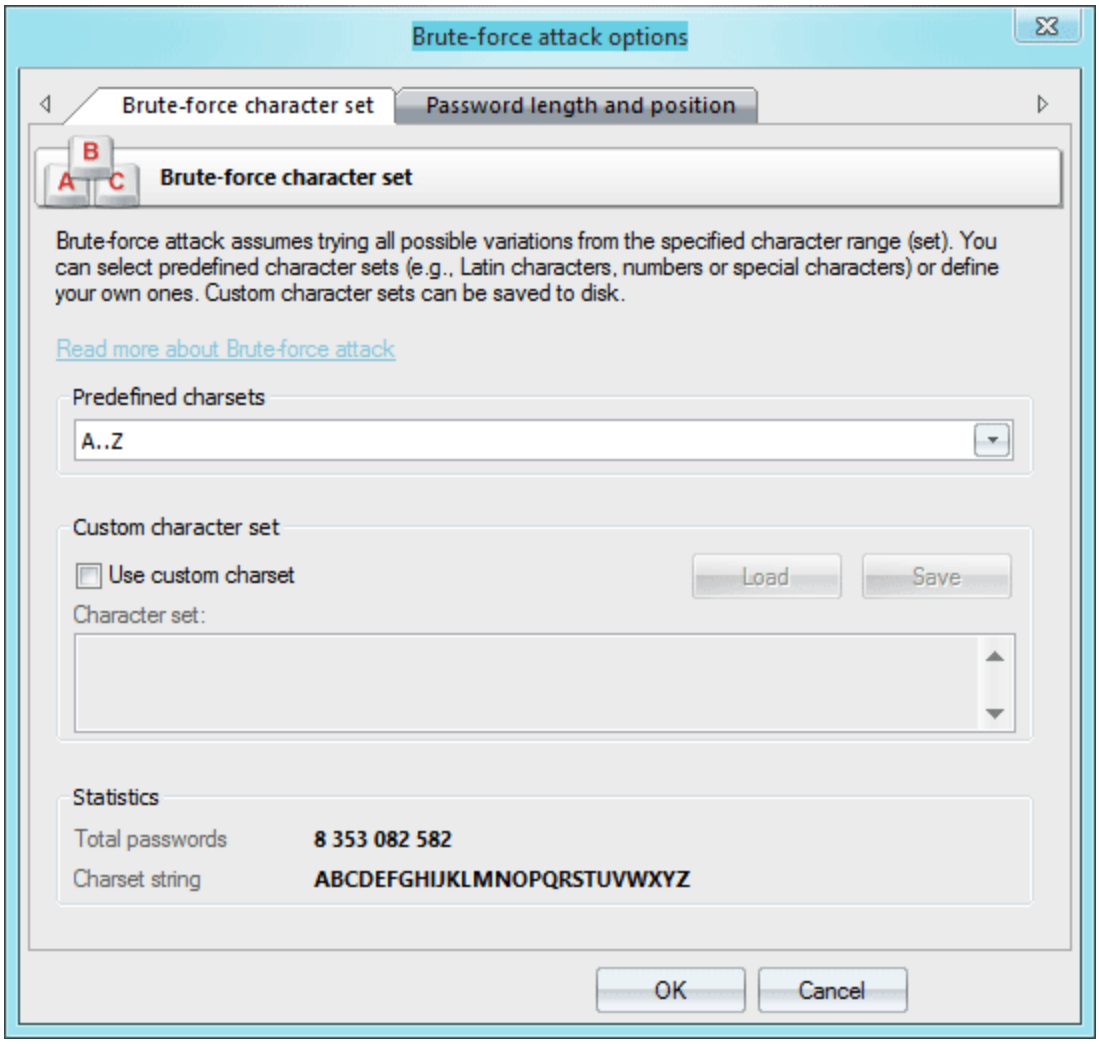
On the third tab, you can download source dictionaries for fingerprint attack from the Internet.



2.8.2.4 Brute-force attack (exhaustive search)

In cryptanalysis, a brute force attack is a method of defeating a cryptographic scheme by trying a large number of possibilities; for example, exhaustively working through all possible keys in order to decrypt a message. This definition was taken from Wikipedia site.

Well, to put it in simple words, brute-force attack guess a password by trying all probable variants by given character set. Eg. checking all combination in lower Latin character set, that is *'abcdefghijklmnopqrstuvwxyz'*. Brute-force attack is very slow. For example, once you set lower Latin charset for your brute-force attack, you'll have to look through 217 180 147 158 variants for 1-8 symbol password. It must be used only if other attacks have failed to recover your password.



The brute-force attack options consist of two tabs.

The first tab is for setting the range of characters to be searched. You can use the predefined sets or create your own ones. To define your own character set, select the option *'Custom charset'*. This will enable two fields for defining a custom character set: the first one - for entering ASCII characters, second one - for entering non-printable characters. You can save your custom character set on disk. The program comes with several examples of user-defined character sets.

On the second tab, set the minimum and maximum length of the passwords to be searched. Please note that for attacking LM hashes the maximum password length should not exceed 7 characters. You can also set a starting password, which would start the search.

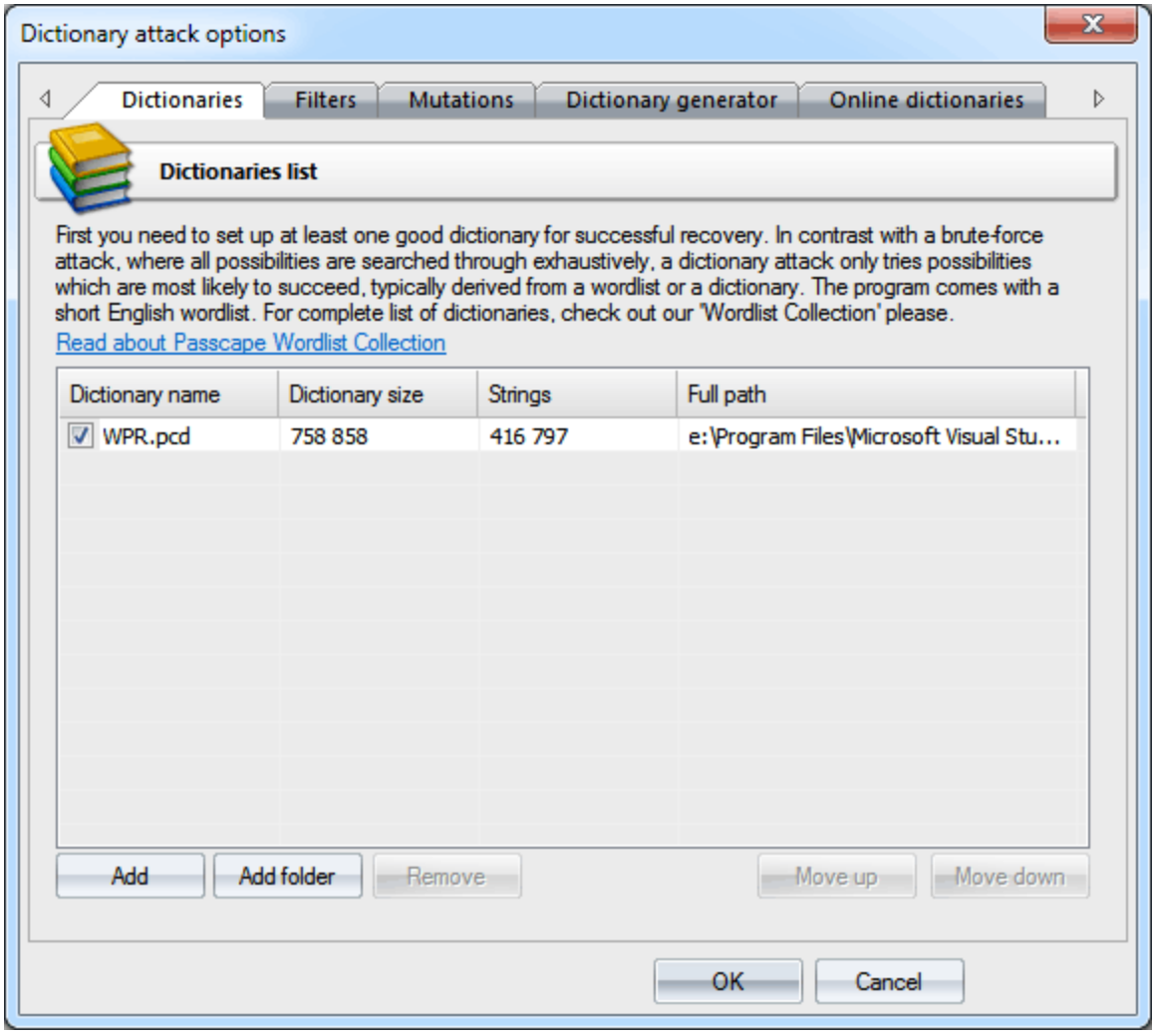
Below is a table that shows password strength depending on the password length and complexity. Assuming that the recovery speed is 100M passwords per second.

Character set	Password length	Password example	Time to crack (exhaustive brute-force search)
A .. Z	5	CRUEL	Instantly
A .. Z	6	SECRET	3s

A .. Z	7	MONSTER	1m 23s
A .. Z	8	COOLGIRL	36m 11s
A .. Z, 0 .. 9	5	COOL3	Instantly
A .. Z, 0 .. 9	6	BANG13	22s
A .. Z, 0 .. 9	7	POKER00	13m 26s
A .. Z, 0 .. 9	8	LETMEBE4	8h 3m 37s
A .. Z, a .. z, 0 .. 9	5	P0k3r	9s
A .. Z, a .. z, 0 .. 9	6	S3cr31	9m 37s
A .. Z, a .. z, 0 .. 9	7	DidIt13	9h 56m 33s
A .. Z, a .. z, 0 .. 9	8	GoAway99	25d 16h 26m 34s

2.8.2.5 Dictionary attack

In contrast with a brute-force attack, where all possibilities are searched through exhaustively, a dictionary attack only tries possibilities which are most likely to succeed, typically derived from a wordlist or a dictionary. Generally, dictionary attacks succeed because many people have a tendency to choose passwords which are short, single words in a dictionary, or are simple variations that are easy to predict.



On the *'Dictionaries'* tab, set up the list of dictionaries to be used in the attack. Supported are plain-text dictionaries in the formats ASCII, UNICODE and UTF8, as well as encrypted/compressed dictionaries in

the native PCD format, developed in Passcape Software. ZIP and RAR packed wordlist are supported as well with some restrictions. To deactivate a dictionary, simply clear the checkbox by its name. In this case, the dictionary, although it remains on the list, will be skipped during an attack. The software comes with a 360000-word dictionary. For complete list of dictionaries, check out our wordlist collection please. Or you can use our online dictionaries as an alternative.

The *'Filters'* tab filters the words from a dictionary by the include/exclude principle. If the first, inclusive, filter is enabled, the attack will accept only the words that contain at least one of the characters entered in the filter. If the second, exclusive, filter is set, the program will skip the words that contain at least one of the entered characters.

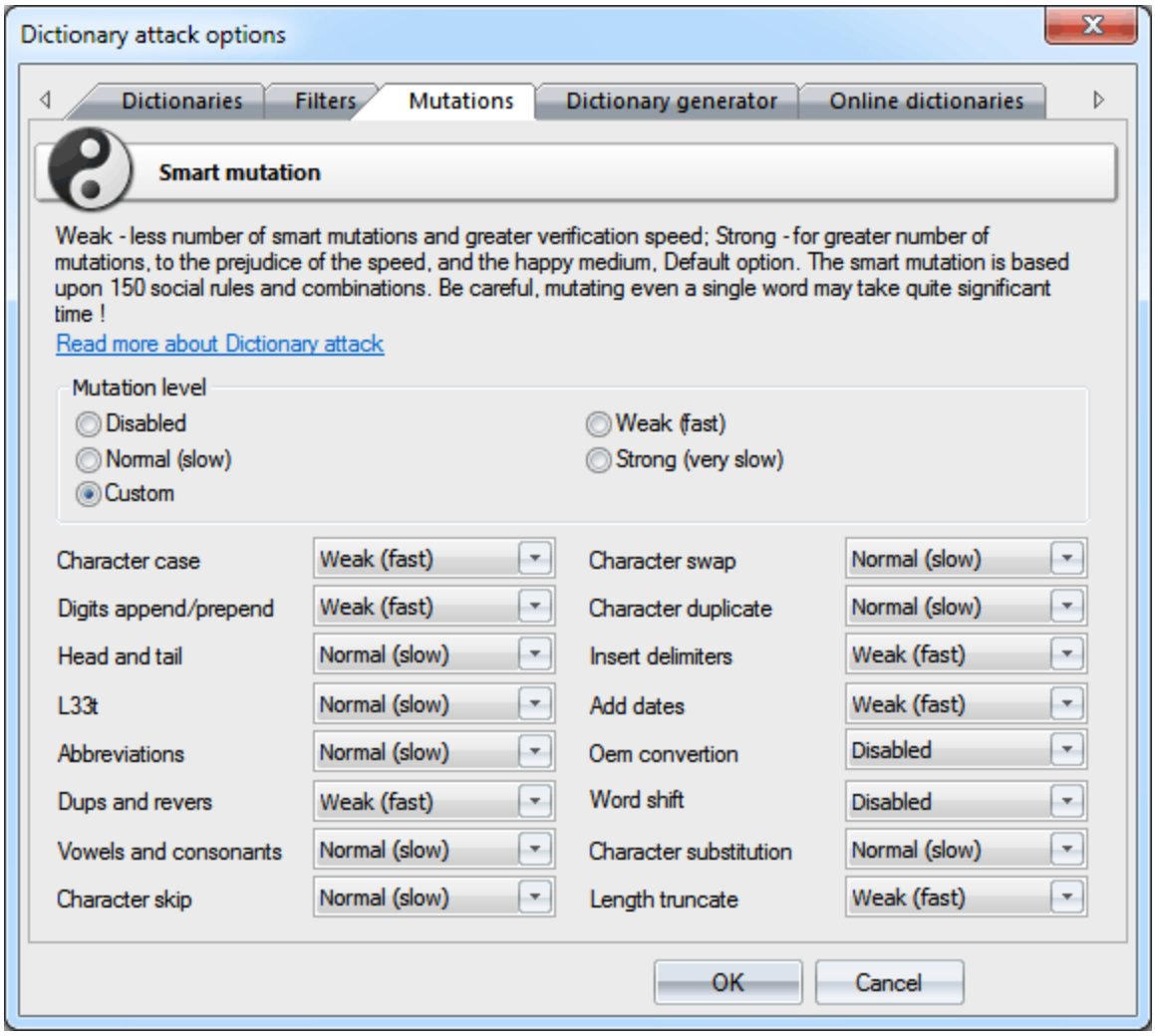
The *'Mutation'* tab allows setting all kinds of possible combinations of the words to be searched. For example, if you set a strong mutation, the program will create several hundreds of analogs for each word from the dictionary. For example, secret - Secret - s3cr3t - secret123, and so on. You can set up to three mutation rules: *Weak* - less number of mutations and, in its turn, greater verification speed; *Strong* - for greater number of mutations, to the prejudice of the speed, and the happy medium, default option (*Normal*).

You can use *Dictionary Generator* to create your own wordlists based on options of the first three tabs.

Online dictionaries. The program has a great feature that allows downloading and using existing dictionaries available on the Passcape website. We have accumulated quite a large dictionary collection - over 250 items. That should get you rid from the extra hassle on finding the required content on the Net.

Customizing mutations

Starting with version 4.0, the program has ability to customize the smart mutation of the Dictionary attack. All mutation rules are clustered into 16 primary groups. You can set one of three mutation levels or disable mutation separately for each group.



For example, you can turn off OEM mutation (and thus double your Dictionary attack speed) if you sure the password you're looking for contains Latin characters only. Simple description of what all these mutation groups mean is given below:

Group name	Description	Examples (for word '**password**')	Comments
Character case	Checks case combinations of the input word.	Password, PasswordD, PaSsWoRd	Maximal (Strong) level of the mutation group DOES NOT generate all possible case combinations of input words. To check all possible case variants, consider using Hybrid dictionary attack (**aN** rule)
Digits append/ prepend	Adds digits to the beginning or to the end of the word.	password99, 2Password, PASSWORD3	Maximal level adds 2 digits.
Head and tail	Almost the same as previous one, but appends or prepends words, abbreviations, characters, keyboard combinations, etc.	#Password#, password12345, 4everPASSWORD, Passwordqwerty	

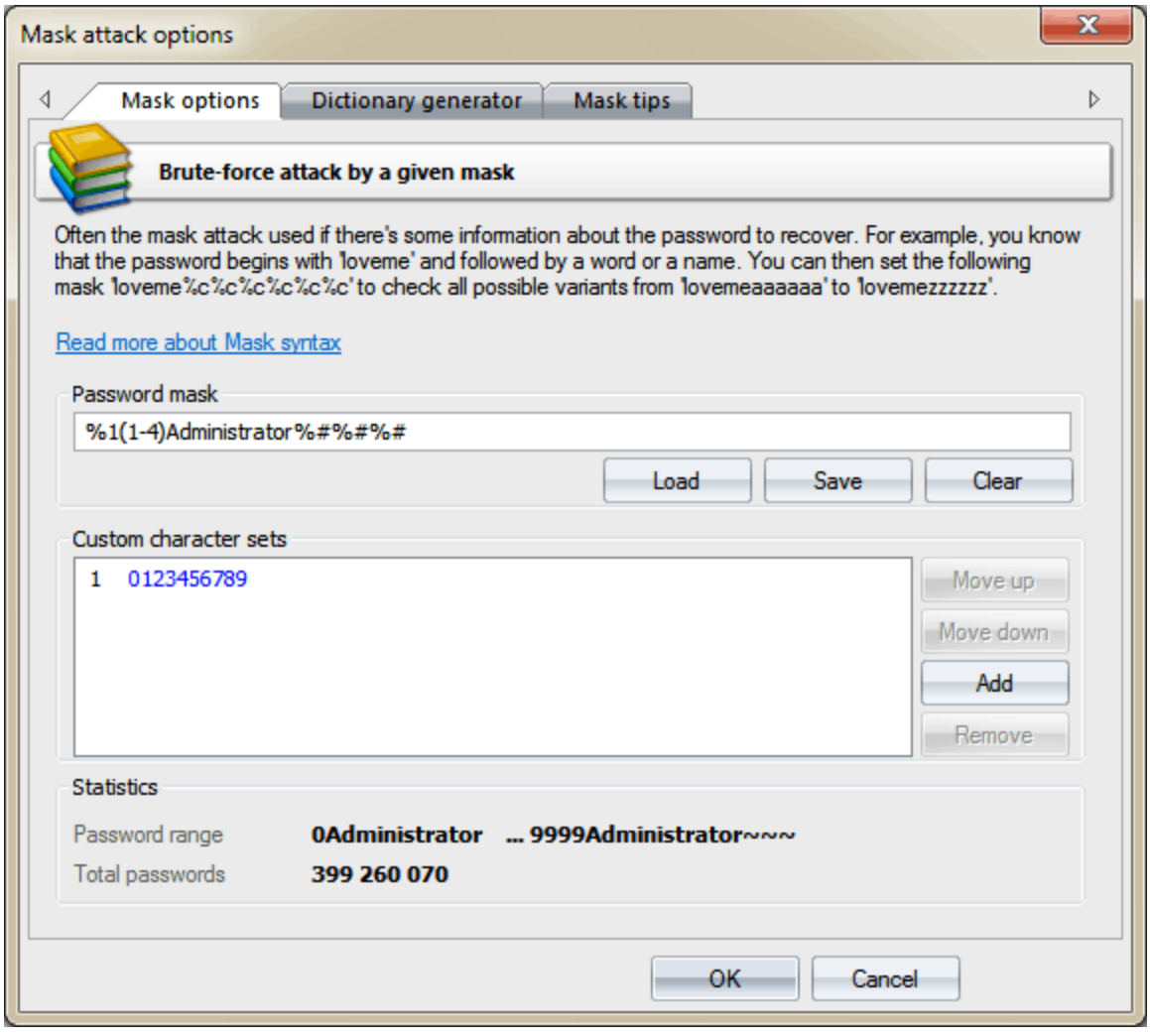
Group name	Description	Examples (for word '**password**')	Comments
l33t	Creates different combinations using leet language.	p@ssword, P@$$w0rd, P@$$W0RD	
Abbreviation	Converts several character combinations (if the initial word contains any) into abbreviations.	ihateyou -> ih8you, Ih8u	
Dups and revers	Revers, duplicates the word, etc.	drowssap, passwordpassword, PasswordDrowssap	
Vowels and consonants	Mutates vowels and consonants (English characters only).	Psswrd, PaSSWoRD, pAsswOrd	
Character skip	Skips a single character of the original word.	assword, Passwrd, Pasword	
Character swap	Exchanges two adjacent characters.	apssword, Passowrd	
Character duplicate	Duplicates characters.	ppasword, ppaasswwoorrdd, Passwordddd	
Delimiters	Separates characters with delimiters.	p.a.s.s.w.o.r.d, P-a-s-s-w-o-r-d	Maximal level uses 10 delimiters.
Dates	Adds dates to the end of the word.	Password2010, password1980	Even though the mutation engine can generate more complicated variations (for example, password03171998 or Password19710830), this feature if turned off here even in maximal mutation level.
Oem convertion	Converts English word into another language and vice-versa using alternative keyboard layout (second language of the OS).	If your OS has 2 languages installed (let it be English and Russian), the program will convert initial word **password** into Russian **зфыыцщкв**, and Russian **пароль** will be converted into **gfhjkm**.	The program works correctly for 2 or even more languages. So if you have 5 languages installed locally (including English one), there will be 4 different combinations of the input word.
Word shift	Stupidly shifts all characters of the word to the right or to the left.	asswordp, dpasswor	
Character substitution	Replaces a character of the initial word.	oassword, passqord	This is quite helpful rule assuming the fact that the characters for substitution are taken from a special table. For example, the character 's' will be replaced with the following ones: 'a', 'w', 'e', 'd', 'x', 'z'. You can notice that all of these characters are located near 's' on any qwerty keyboard.

Group name	Description	Examples (for word '**password**')	Comments
Length truncate	Truncates word length to probe all possible length combinations.	passwor, passwo, Pass	

2.8.2.6 Mask attack

Mask attack is an irreplaceable tool when you know a fragment of the password or have any specific details about it. For example, when you know that the password consists of 12 characters and ends with the *qwerty*, it is obvious that searching the entire 12-character range of passwords is unreasonable. All what would be required in this case is to pick the first 6 characters of the sought password. That is what mask attack is for.

In our case, we could define the following mask: **%c%c%c%c%c%cqwerty**. That means that the program would serially check the following combinations: aaaaaaqwerty .. zzzzzzqwerty. If the original password is *'secretqwerty'*, it perfectly hits our range.



The mask entry field is used for setting the rule, by which the program will try to recover the password. If the mask is set correctly, below you will see the range of characters generated by the mask. User-defined masks can be saved to disk. You can also use the mask tool to generate a dictionary (may not

be available in some editions).

The mask syntax is quite trivial and consists of static (unmodifiable) and dynamic (modifiable) characters or sets. Dynamic characters/sets always have a leading %. For example, if you set the mask secret%d(1-100), the program will generate 100 passwords (secret1, secret2: secret100).

Windows Password Recovery supports the following dynamic mask sets:

- %c lower-case Latin characters (a..z), 26 symbols
- %C upper-case Latin characters (A..Z), 26 symbols
- %# full set of special characters (!..~ space), total 33 symbols
- %@ small set of special characters (!@#$%^&*()-_+= space), 15 symbols
- %? all printable characters with ASCII codes of 32..127
- %* all ASCII characters (codes 1 through 255)
- %d one digit (0..9)
- %d(x-y) numbers between x and y inclusive
- %r(x-y) user-defined characters with serial UNICODE codes between x and y
- %r(x1-y1,x2-y2...xn-yn) set of several non-overlapping sequences of UNICODE characters.
- %[1..9] a character from user defined charset 1..9
- %[1..9](min-max) user-defined range of variable length (from min to max). You can set up to 9 your own custom character sets.
- %% standalone static character %

Examples:

test%d - will generate password range test0..test9, 10 passwords total
test%d(1980-2007) - test1980 .. test2007, 28 passwords
test%r(0x0600-0x06ff) - 256 passwords with Arabic character at the end
%#test%# - _test_ .. ~test~, 1089 passwords
admin%1(1-5) - admina .. adminzzzzz, where %1 is user defined charset 1 (a..z)
%1%1%1pin%2%2%2 - aaapin000 .. zzzpin999, %1 is user character set a..z and %2 is second user-defined charset which contains characters 0..9

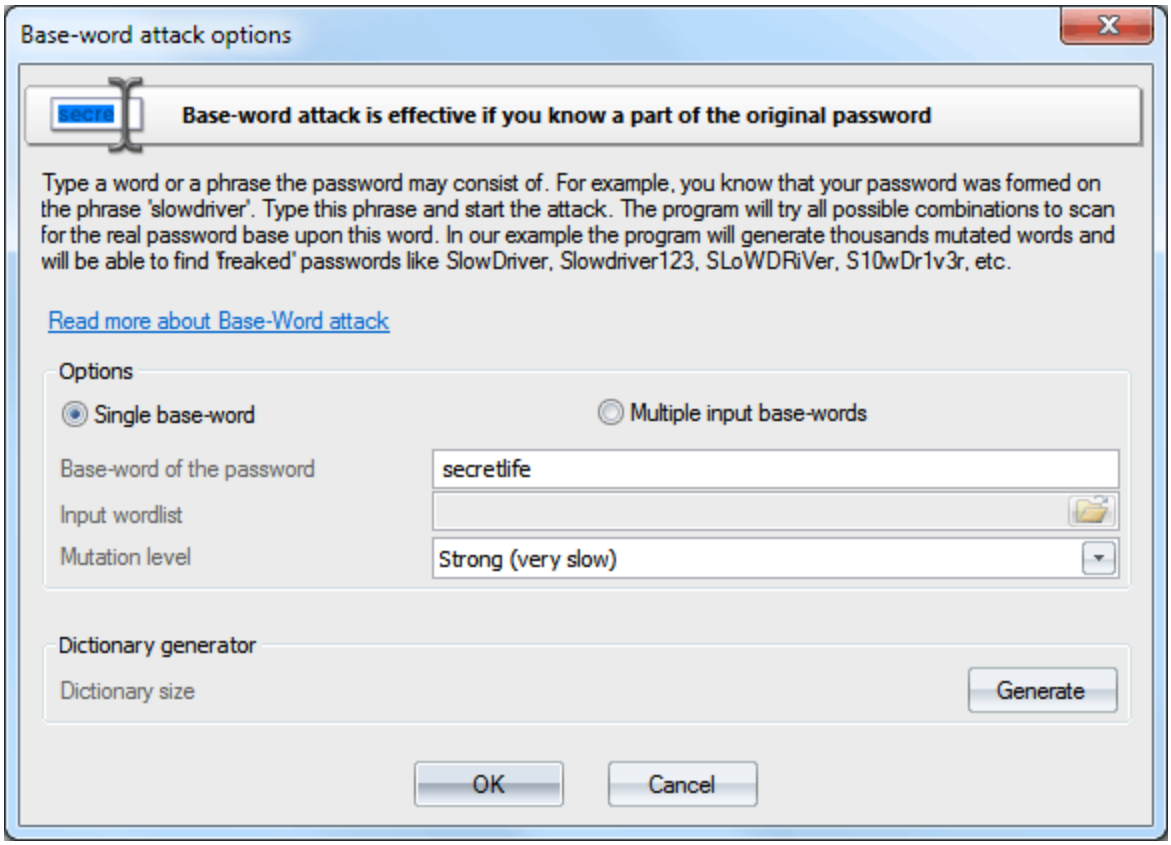
By switching to **Dictionary generator tab**, you can generate your own dictionary by a given mask, and save it to disk. This feature available in Advanced edition of the program only.

Third tab of the mask options contains a short description of the mask syntax and a couple of simple examples.

2.8.2.7 Base-word attack

Base-word attack (developed by Passcape) is in many ways similar to mask attack. However, here you don't need to set up the syntax; simply enter the keyword, which supposedly was the base word for the password. It is an irreplaceable recovery tool when you know a portion of the password or its basic component. Normally, such cases dispose to using mask attack; however, it does not always allow coping with the task set forth. Suppose our password was *'S10wDr1v3r'*. Trying to recover such a complicated password using brute-force attack would be an ungrateful job, even if you are quite sure that it is based upon the *'slowdriver'* word. These are the cases when the base-word attack will rescue you.

With this tool, the program will attempt to recover the original password, trying all possible combinations founded upon 15 groups of rules (total over 150 rules). If you enter *'slowdriver'* in the field, you will see that the program has generated several thousands of different combinations upon this phrase, and one of those combinations could match our password.



If the length of the input phrase exceeds 8-10 characters, the mutation may take significant time. If you remember the original password precisely and simply have forgotten the sequence of the upper-case and lower-case characters in it, you can select the option '*Mutate character case only*'. With this option selected, the program will generate passwords with all possible combinations of upper-case and lower-case characters, total 2^n passwords, where n - is password length. For example, for the password *'slowdriver'* the program will generate 2^10=1024 different combinations for each keyboard layout installed on your computer. You can also generate a dictionary on those mutations and save it on a disk (available not in all editions).

Note, if your password length exceeds 15-16 characters, it may take quite some time to prepare (mutate) the password for the attack.

In Windows Password Recovery version 9.5 the Base-word recovery was split into 2 modes: single input word and many input words. The multiple input words mode acts like the Dictionary attack with maximal mutations set on, but generates much much more passwords (even if the mutation level of the Base-word attack is set to 'Weak'), which maybe useful in a certain situation.

2.8.2.8 Combined dictionary attack

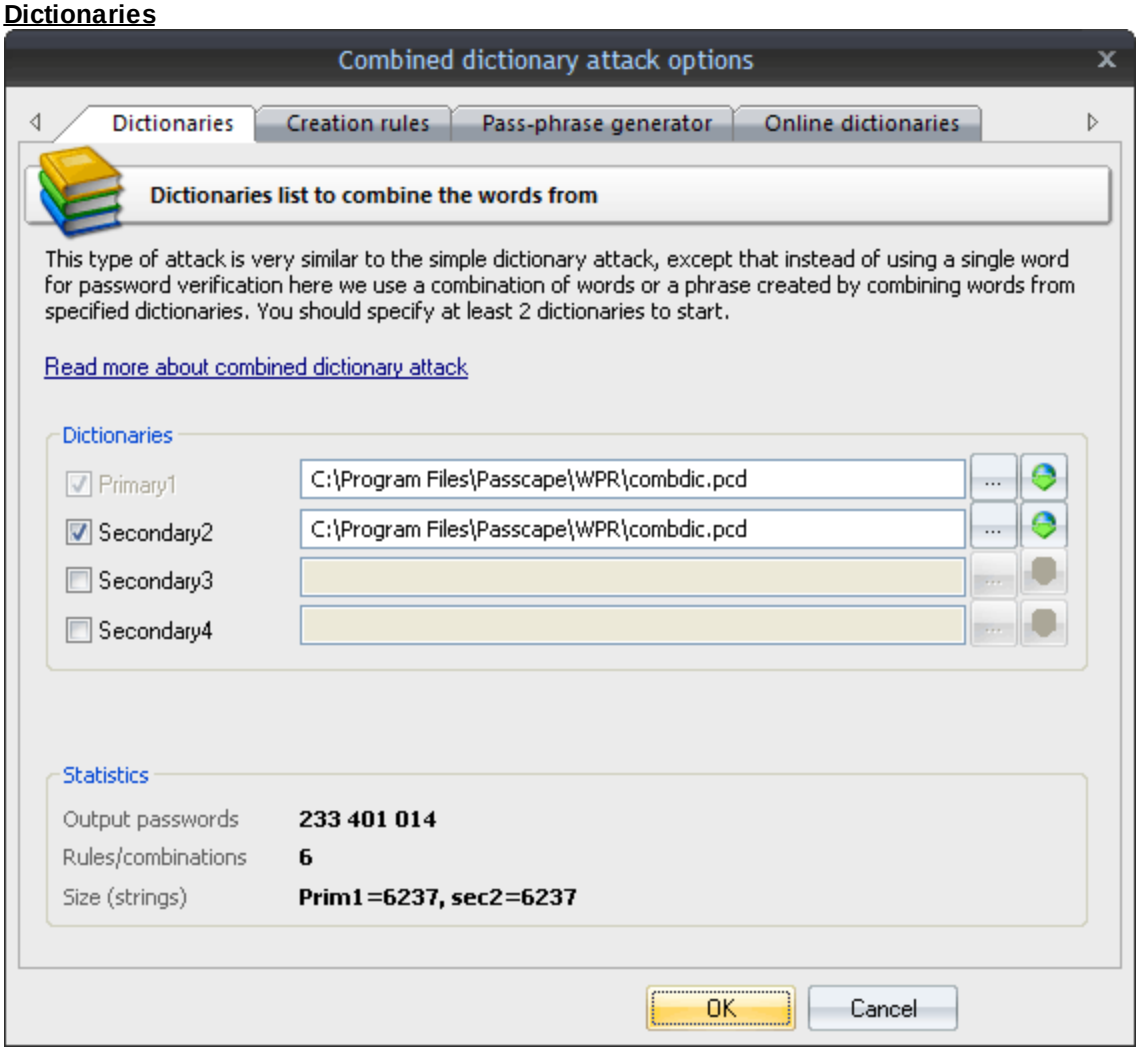
Combined dictionary attack (developed by Passcape Software) is great at recovering passwords that consist of 2,3 and even 4 words. This type of attack on difficult and compound passwords is very similar to the simple dictionary attack, except that instead of using a single word for password verification here we use a combination of words or a phrase created by combining words from specified dictionaries. To successfully utilize this attack, set at least two dictionaries and the rules for generating passwords. You can set the regular dictionaries used in the simple dictionary attack, but it is recommended to use rather small dictionaries with the most common words. Perfect dictionaries for the combined pass phrase attack are those that have different forms of words in them; e.g. jump, jumper, jumped, jumping.

Combined attack sets a certain limit to the number of dictionaries that can be used; that's not more than 4. Thus, the general limitation of this attack is that only password phrases of not more than 4 words can be recovered using this attack.

Another essential drawback is the wide range of phrases generated. And, as the consequence, the proportional increase of the time spent on the validation of a password. Keep in mind that when generating passwords that consist of 3 or 4 words, the generation process takes considerable time

If finding the right dictionary is difficult, don't worry. The software comes with a special dictionary for the combined attack. You can also take advantage of the Online Dictionaries tab or the corresponding button to download such dictionaries from the Passcape website.

Dictionaries



The way the combined attack works is really simple. For example, if you have set two dictionaries, the program will generate the passwords as follows: it will take the first word from the first dictionary and glue it with the first word from the second dictionary, then with the second word, and so on until the end. Then it checks the second word from the first dictionary and goes the same route, and so on.

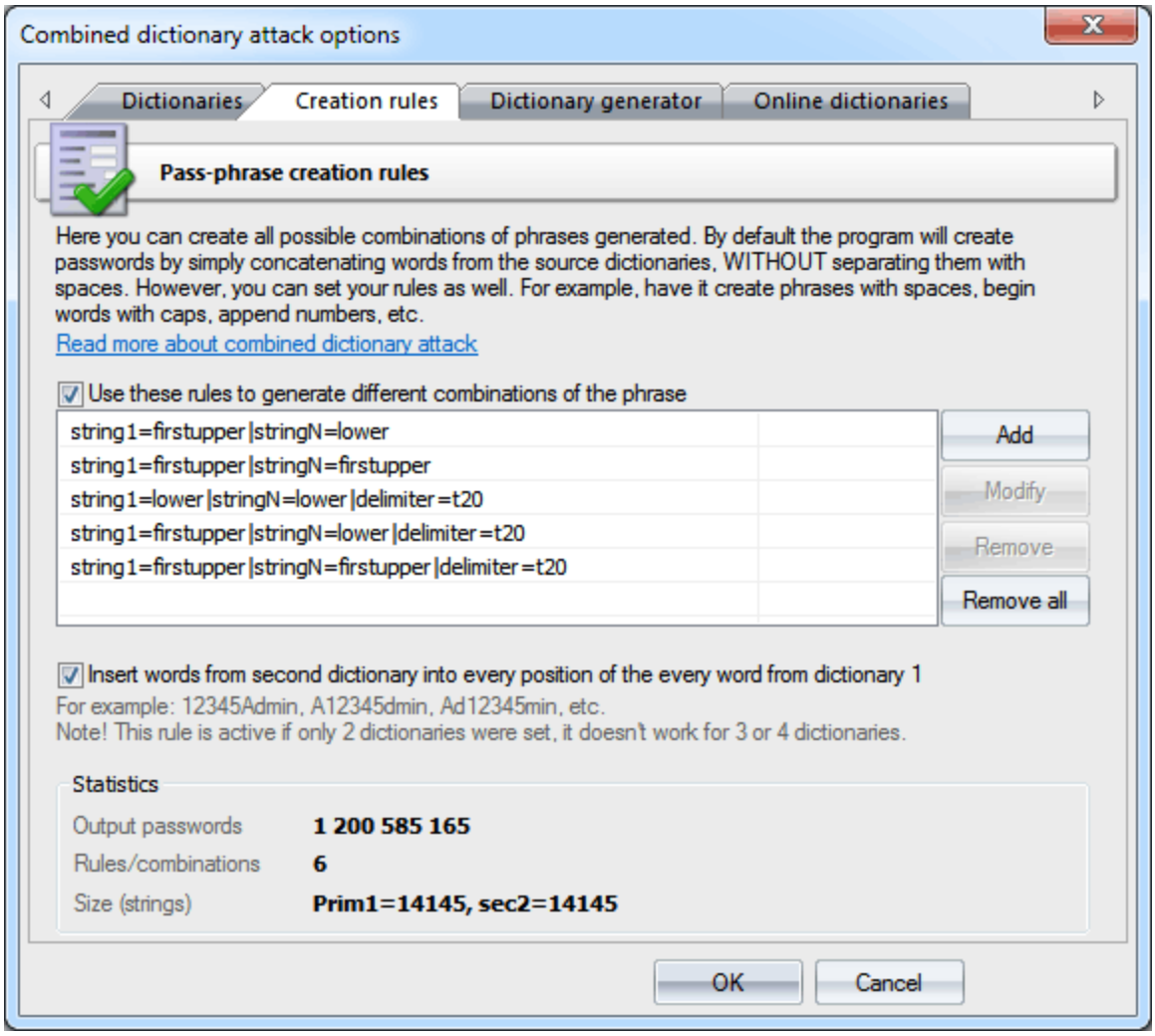
To understand how the combined attack works, let's take a look at a couple of password generation examples that involve, in the first case, the same dictionary and in the second case - two different ones.

1. Suppose we've got a single dictionary with three words: action, bad, and computer. We will set this dictionary as two original sources: primary dictionary & secondary dictionary2 (see the figure). After these dictionaries have been processed, at the output we have the following phrases (they will be used when checking the password sought):
'actionaction', 'actionbad', 'actioncomputer'
'badaction', 'badbad', 'badcomputer'
'computeractio', 'computerbad', 'computercomputer'.
9 phrases total.

2. In the second case, we have got two different dictionaries. For example, the first dictionary consists of three words: action, bad, and computer. The second one also has three words: date, eagle, fail. In this case, we are going to have the following phrases:
'actiondate', 'actioneagle', 'actionfail'
'baddate'', 'badeagle', 'badfail'
'computerdate', 'computereagle', 'computerfail'.

The example is plain but demonstrative. The idea is that for multiple sources you can successfully use both a single dictionary and multiple ones. It all depends on your imagination. The last example shows that a special attention should be paid to the order of the dictionaries if they are different. The order of the words in the phrases to be created depends directly on the order of the source dictionaries. In our second example, if we swap the primary and the secondary dictionaries, at the output we will obtain a completely different set of phrases.

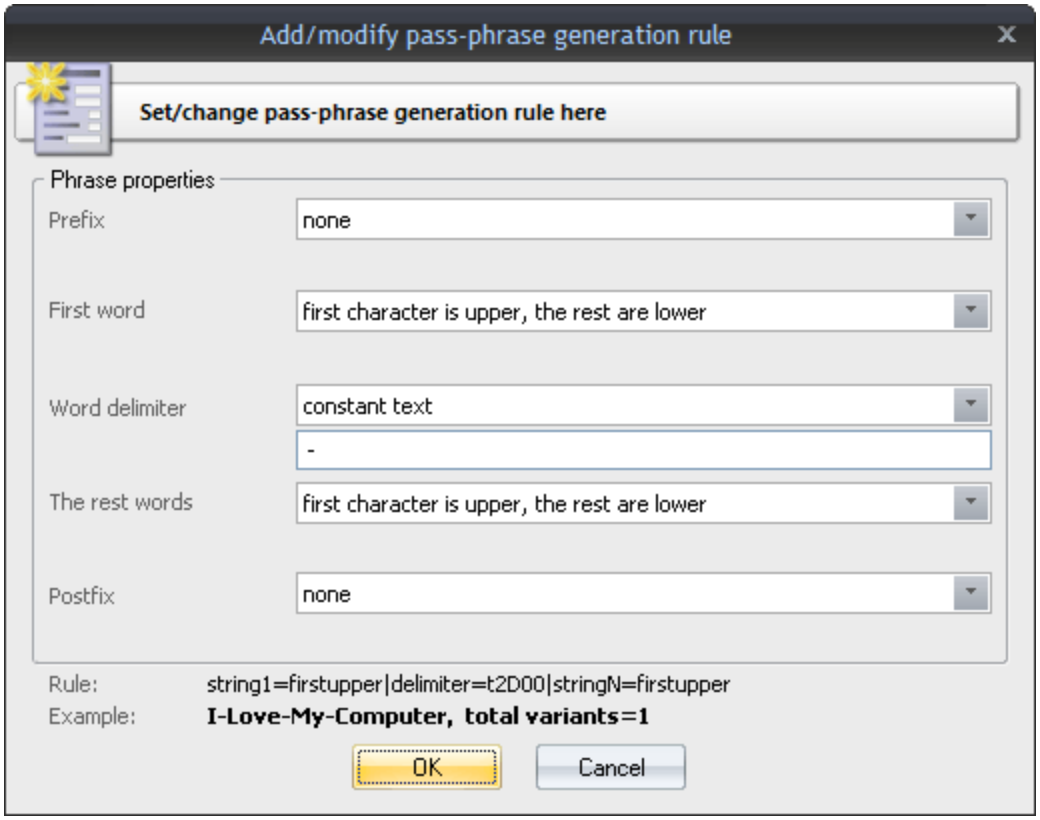
Mutation rules



Passwords created by the combined attack are generated according to special rules that are to be set on the second tab. By default, when password generation rules are disabled, the program generates passwords by simply gluing up the words from the dictionaries, without separating them with a space. For example, of the two words are 'my' and 'computer', you will get 'mycomputer'.

If word insertion option is set, the program additionally creates passwords by inserting words from second dictionary into every position of the word from dictionary 1. For example, if the first dictionary's word is **Admin**, and the word from the second dictionary is **12345**, the program will generate the following passwords:
12345Admin
A12345dmin
Ad12345min
Adm12345in
Admi12345n
And so on for all words of the second dictionary. Then goes another word from dictionary 1, etc. The option is active if only 2 dictionaries were set.

The generation rules are made to extend the password search options. For example: Mycomputer, MyComputer, MY COMPUTER, my-computer, etc. There are special rules available for this purpose; you don't have to know the syntax of them, for the mutation rule creation dialog is simple and intuitive.



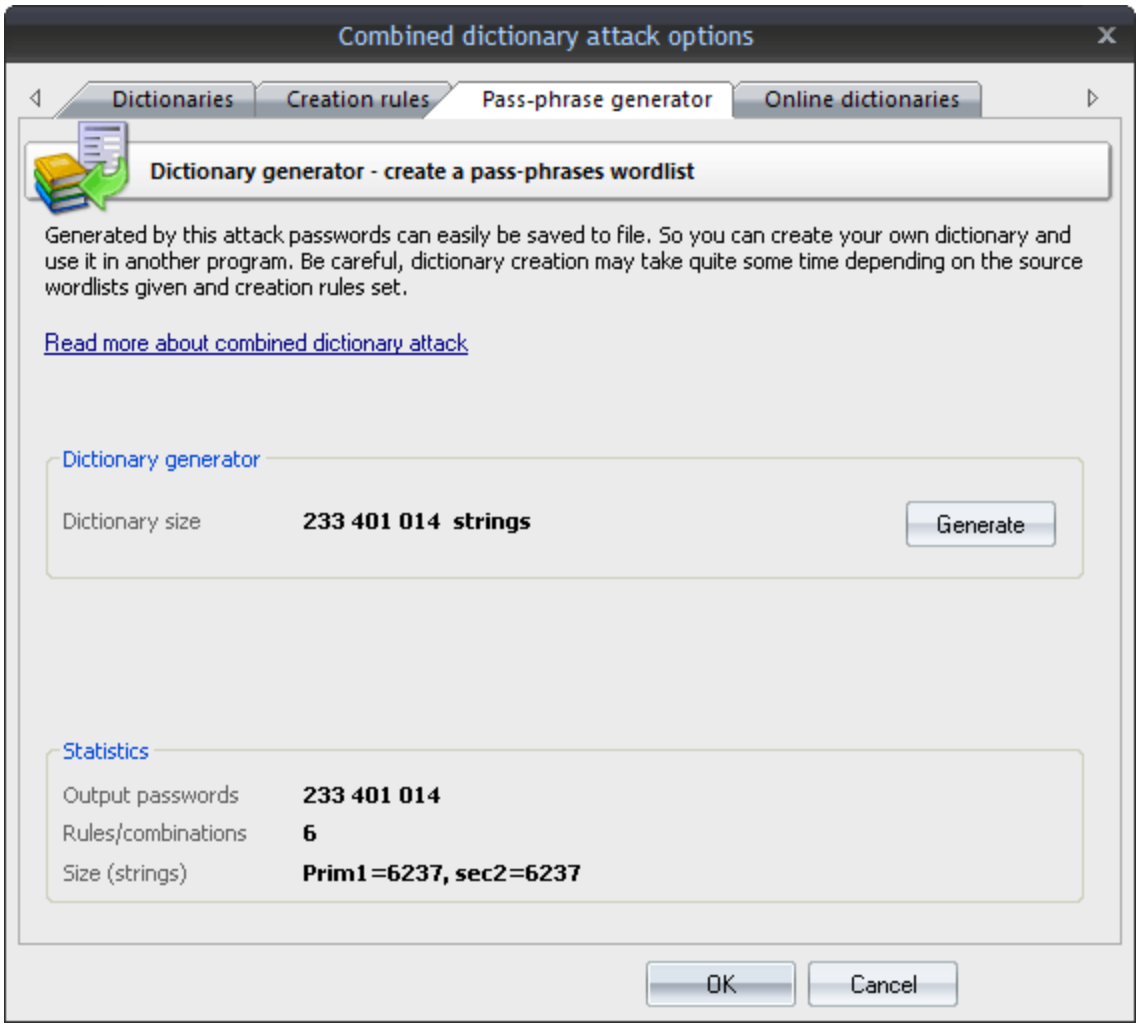
Each mutation rule consists of five elements:

1. *Prefix* - text that will appear before each phrase. This element can be a character, plain text string, one digit between 0 and 9 or a number. For instance, if you set a one-digit prefix, the phrases created with this rules will look as follows: '0 aaa bbb', '1 aaa bbb' : '9 aaa bbb'.
2. *First word* - the action to be performed over the first word of each phrase. There are only four options. Namely: leave intact as is in dictionary, convert all characters to lowercase, convert all characters to uppercase or capitalize only the first letter of the word.
3. *Word separator*. It may be absent. Then all the words will be concatenated. Example: 'aaabbb', 'aaaccc','aaaddd', etc. You can otherwise set a custom separator; e.g. the '-' character: 'aaa-bbb', 'aaa-ccc','aaa-ddd'. Or you can set a range of characters.
4. *Other words*. With this attribute, similarly to 2., you can set rules for the other words of a phrase.
5. *Postfix* - text that will finalize each phrase. For example, if you set Postfix to the '?' or ' ?', all phrases created with this rule will have the question mark at the end.

Certainly, the more password generation rules you set, the more chances you have to pick the right password. But, on the other hand, the more time you will have spent on the attack.

The 'Statistics' group shows the average and recommended average size of a dictionary, number of words in source dictionaries, total number of passwords being generated and other helpful information.

Dictionary generator



The third tab of options serves for creating combined attack-based dictionaries (available not for all editions).

You can also download additional dictionary modules from the Passcape Software Web site.

2.8.2.9 Pass-phrase attack

More and more users choose to make up their pass phrases of entire phrases, passages from poems, movie aphorisms, Latin aphorisms, etc. Attempting to recover such passwords using the traditional techniques is unthinkable, even with the reference to the advancement of the computing power of modern computers. Therefore, the recovery help comes with the predefined and known phrase attack.

Pass-phrase attack is by much similar to the simple dictionary attack, except that here the password search goes phrase by phrase instead of going word by word. The main idea of the attack is to guess the right password by searching through predefined frequently used expressions, phrases and word combinations.

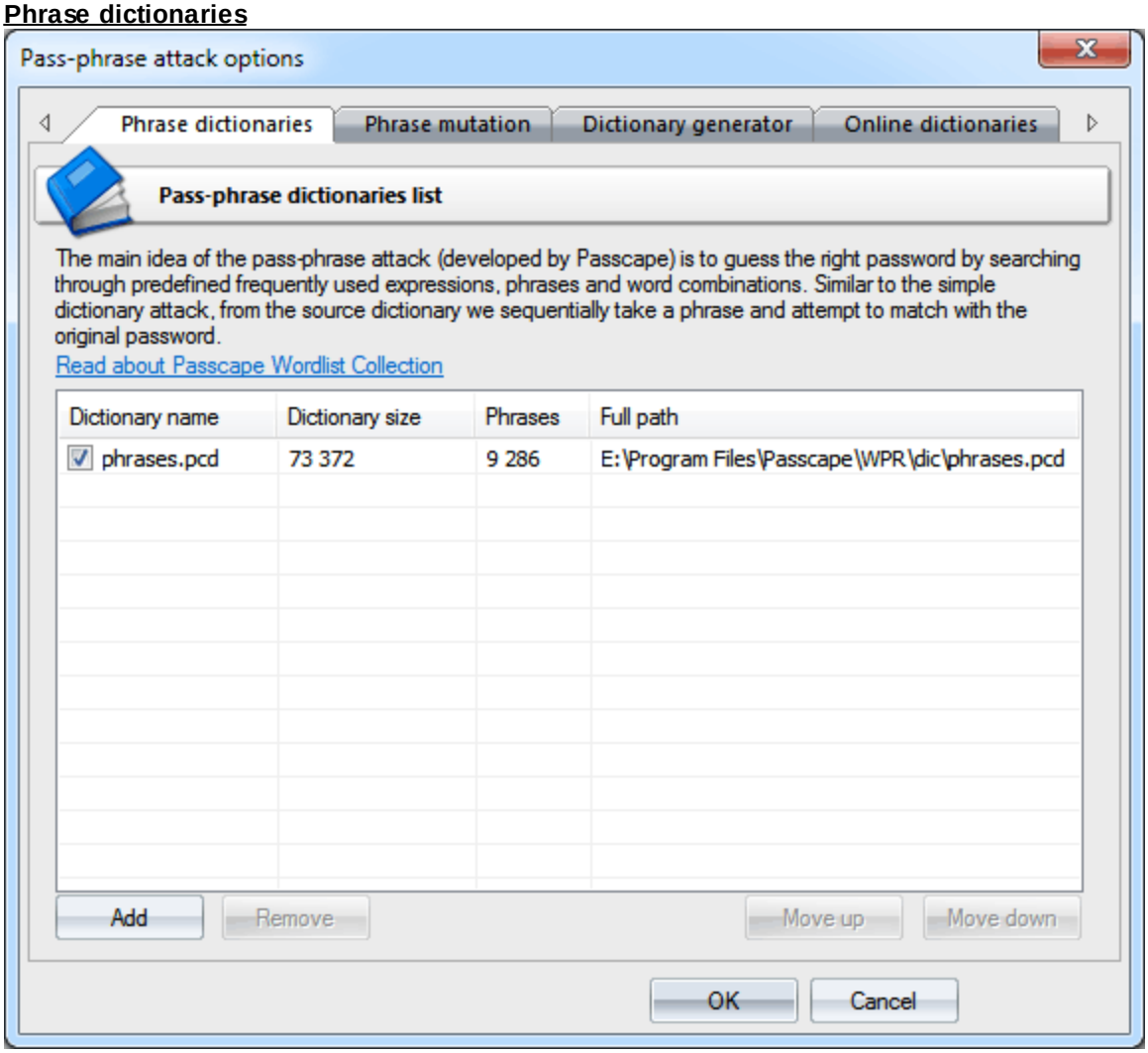
For example, if the sought password is made of the widespread phrase 'To be or not to be', it is obvious that this is the only attack that has the virtue to cope with such a password. In order to do that, you are to specify a special pass-phrase dictionary. A simple phrase dictionary comes with the software, but you can also download the online dictionaries that were compiled specifically for this attack.

It wouldn't be an overestimation to say that 99 percent of the success in the recovery of a password with a dictionary attack depends on the quality of the dictionaries. Most likely, that is the reason why this type of attacks doesn't appear in just about any password cracker. Passcape Software allows utilizing a whole set of online and offline dictionaries (totally over 500 MB) compiled specially for this type of attack.

For example, many users make their passwords of excerpts from their favorite songs or music bands. That's why we have created special, unique (you won't find anything like that anywhere on the Net!) music-oriented key phrase sets. There's also a biblical set, movie phrases, proverbs, etc.

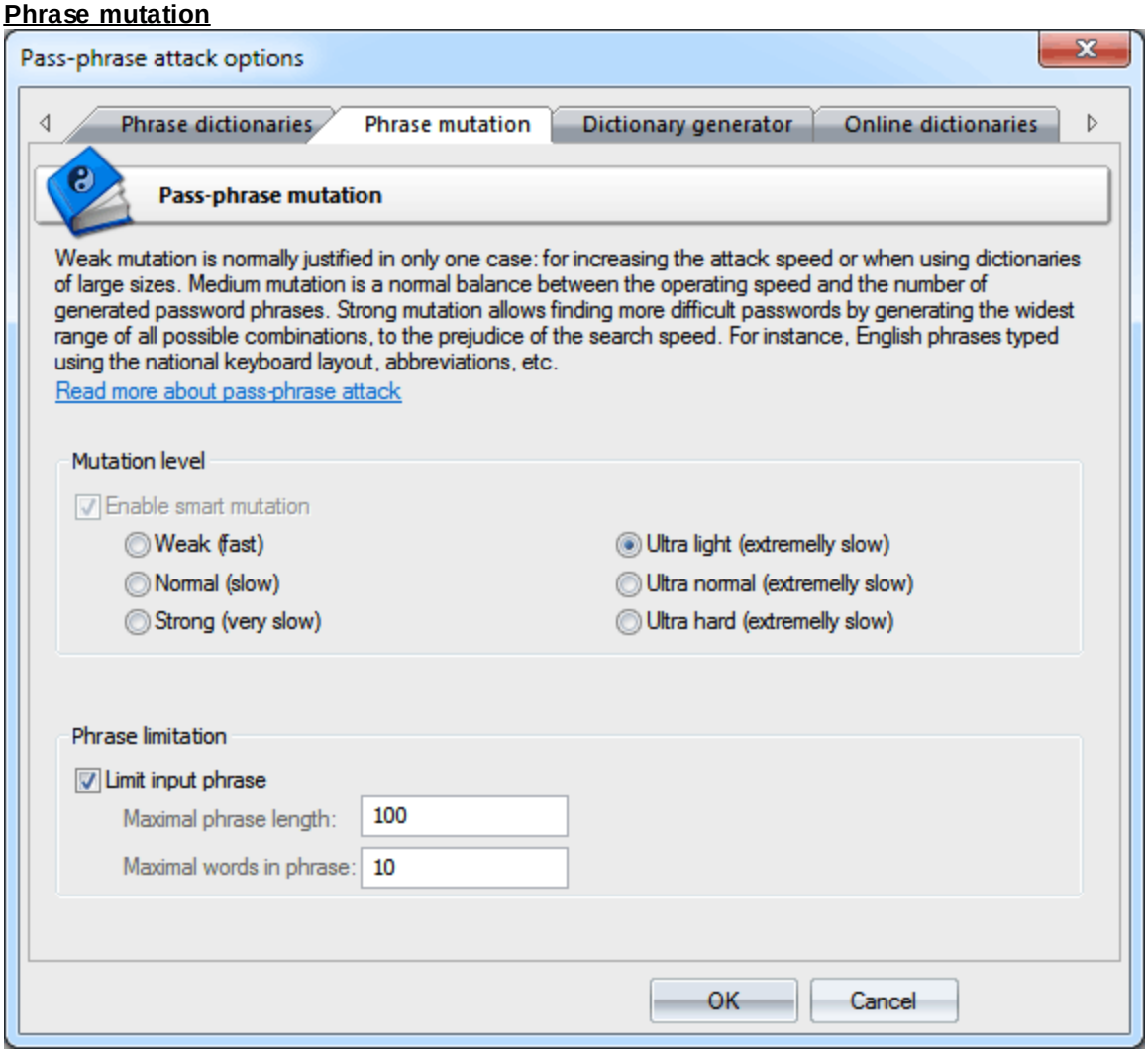
Windows Password Recovery comes with a short dictionary of phrases and aphorisms.

Phrase dictionaries



The password-phrase attack options almost completely repeat the simple dictionary attack options: here, you also are to select one or several dictionaries for the phrase source, it also allows loading additional dictionaries from the Passcape website, and it has the same way for setting phrase mutation rules (creating alternative options).

Phrase mutation



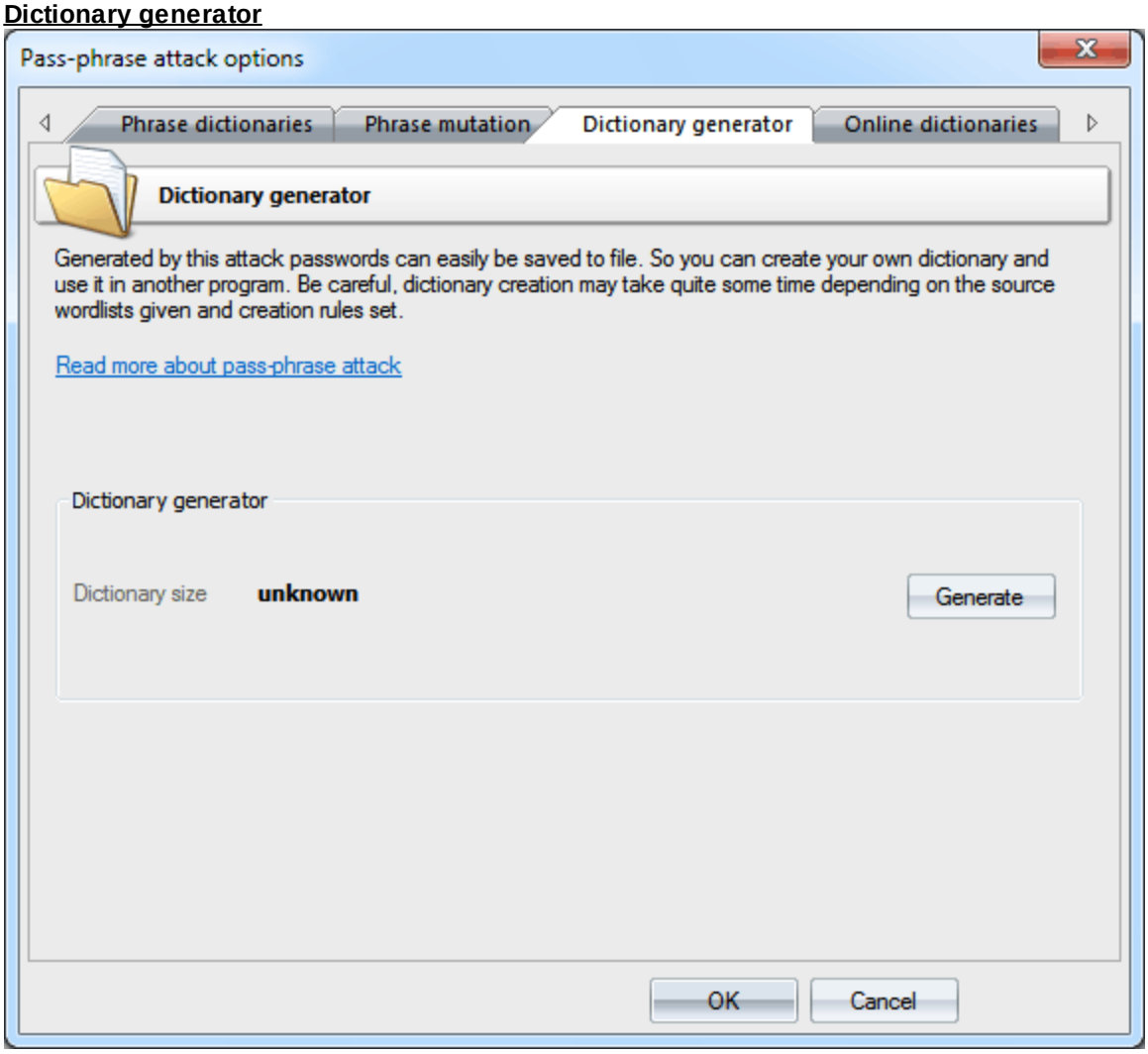
Mutation is worth saying more, since as you should have known strong mutation significantly raises chances for the successful recovery. Weak mutation is normally justified in only one case: for increasing the attack speed or when using dictionaries of large sizes. Medium mutation is a normal balance between the operating speed and the number of generated password phrases. Strong mutation allows finding more difficult passwords by generating the widest range of all possible combinations, to the prejudice of the search speed. The greater is the mutation level, the more passwords the attack will cover. For instance, English phrases typed using the national keyboard layout, abbreviations, etc.

Major difference in mutation levels:

- Weak - simplest thus fastest mutations.
- Normal - the same as Weak, but generates several additional mutations and case combinations.
- Strong - the same as normal plus more mutations and national passwords (according to the installed keyboard layouts, if any).
- Ultra light - this is a 2-step mutation because every generated in Weak mode password goes through the second mutation round (one used in Weak mode of the simple dictionary attack).
- Ultra normal - 2-step mutation. Every password generated in Normal mode is used as a source to generate additional combinations by implementing additional Normal mutation level.
- Ultra hard - every password generated in Strong mode is used as a source to generate additional combinations by using additional Strong mutation level.

Be careful! Ultra modes generate a great number of passwords, thus the attack may be ran extremely slow. To speed up the attack, consider setting up input phrase limits. For example, you can limit input phrases to 10 words and 100 characters.

Dictionary generator

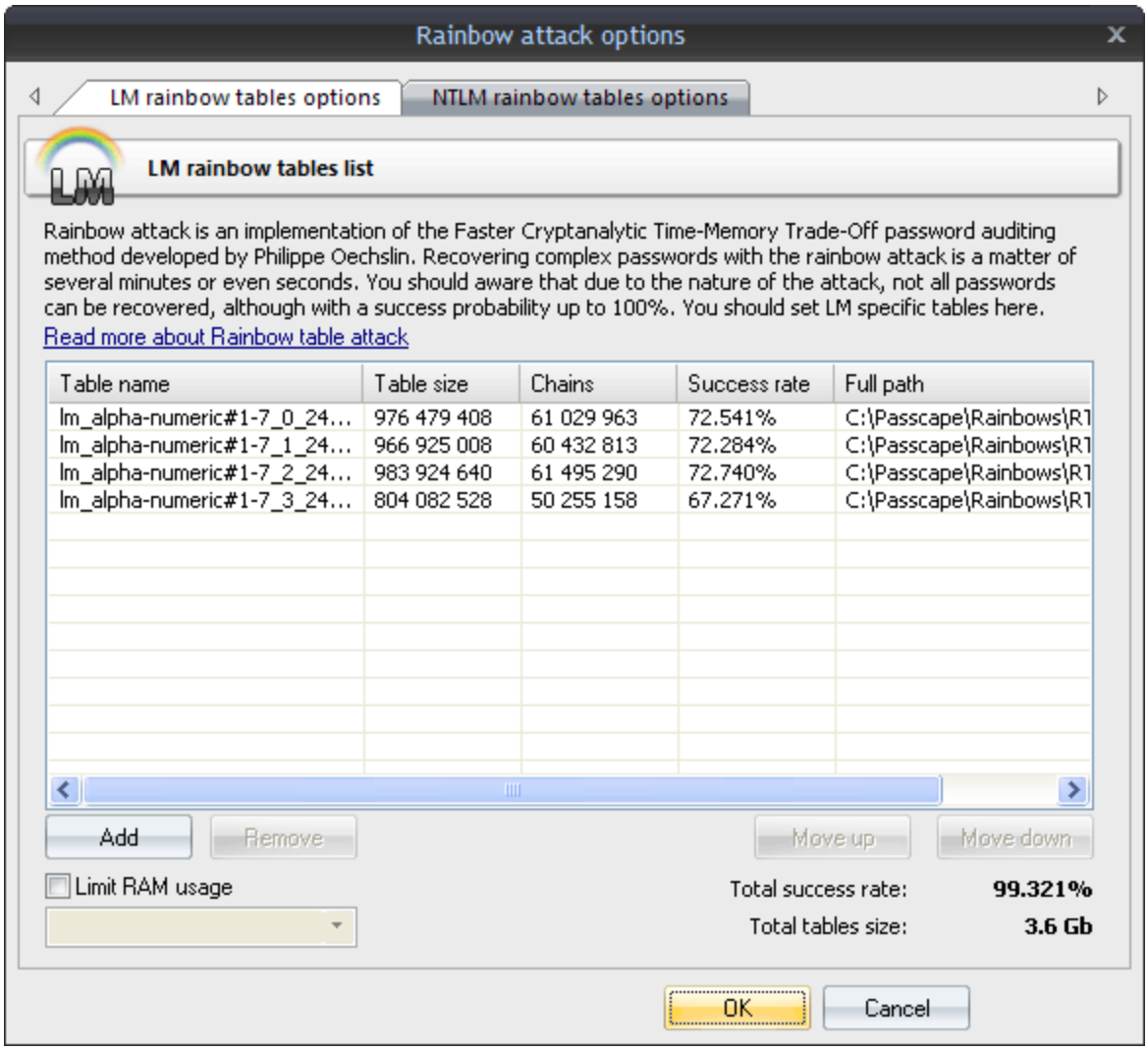


The third tab uses for creating pass-phrase dictionaries.

2.8.2.10 Rainbow tables attack

A rainbow table is a lookup table offering a time-memory tradeoff used in recovering the plaintext password from a password hash generated by a hash function, for example Windows passwords.

This is quite a sophisticated password audit tool. This method was developed by Philippe Oechslin for quick recovery of password using precalculated tables. It's enough to say that the sought password can be recovered within minutes or even seconds.



The program supports the standard *.rt, indexed *.rti and hybrid tables. Multithreading is supported as well.

It must be mentioned that rainbow attack does not guarantee the recovery of all passwords, but the probability of the recovery is close to 100%, depending on the tables you've got.

A specific rainbow table can be implemented for the hash it was created for. Eg. LM specific tables should be used for breaking LM hashes only.

The attack options allow limiting the amount of RAM that can be utilized by the attack when using old computers (the attack assumes using large volumes of RAM for its calculations).

2.8.2.11 Hybrid dictionary attack

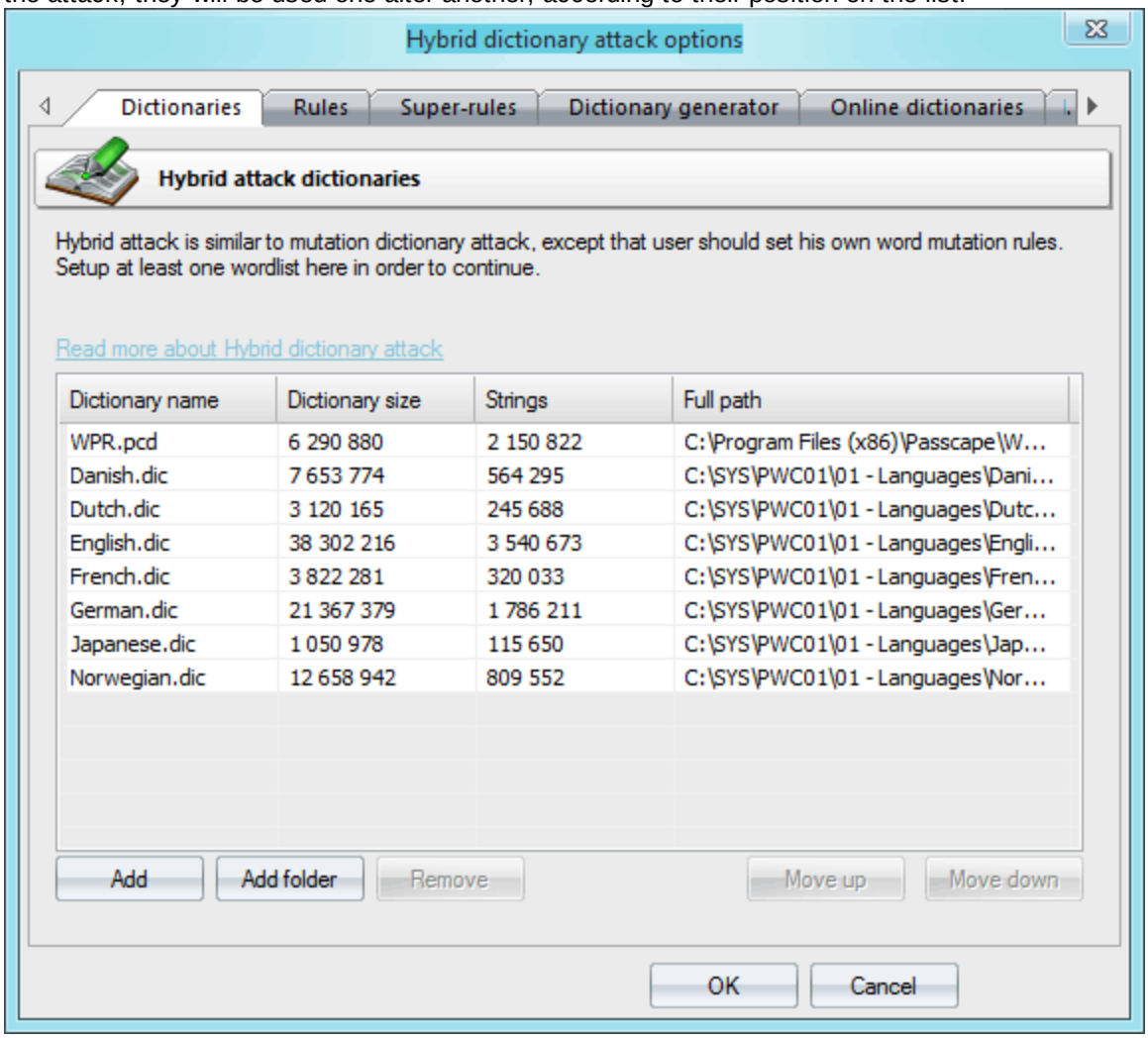
Hybrid dictionary attack is a form of simple dictionary attack. However, unlike the latter, hybrid attack allows user to set his own word mutation (variation) rules and attempt to validate the modified words as source passwords. For example, user could capitalize the first letter of a password being validated, append '2' to it, replace the number 8 in it with the letter B, O with 0, etc.

Actions, performed on source words from the dictionary, are called rules. Multiple rules can be applied to

each source word. The rule definition syntax is compatible with John the Ripper and PassworsPro software. The author of the latter has kindly provided an extended set of rules, slightly edited, which comes with the distribution kit for Windows Password Recovery.

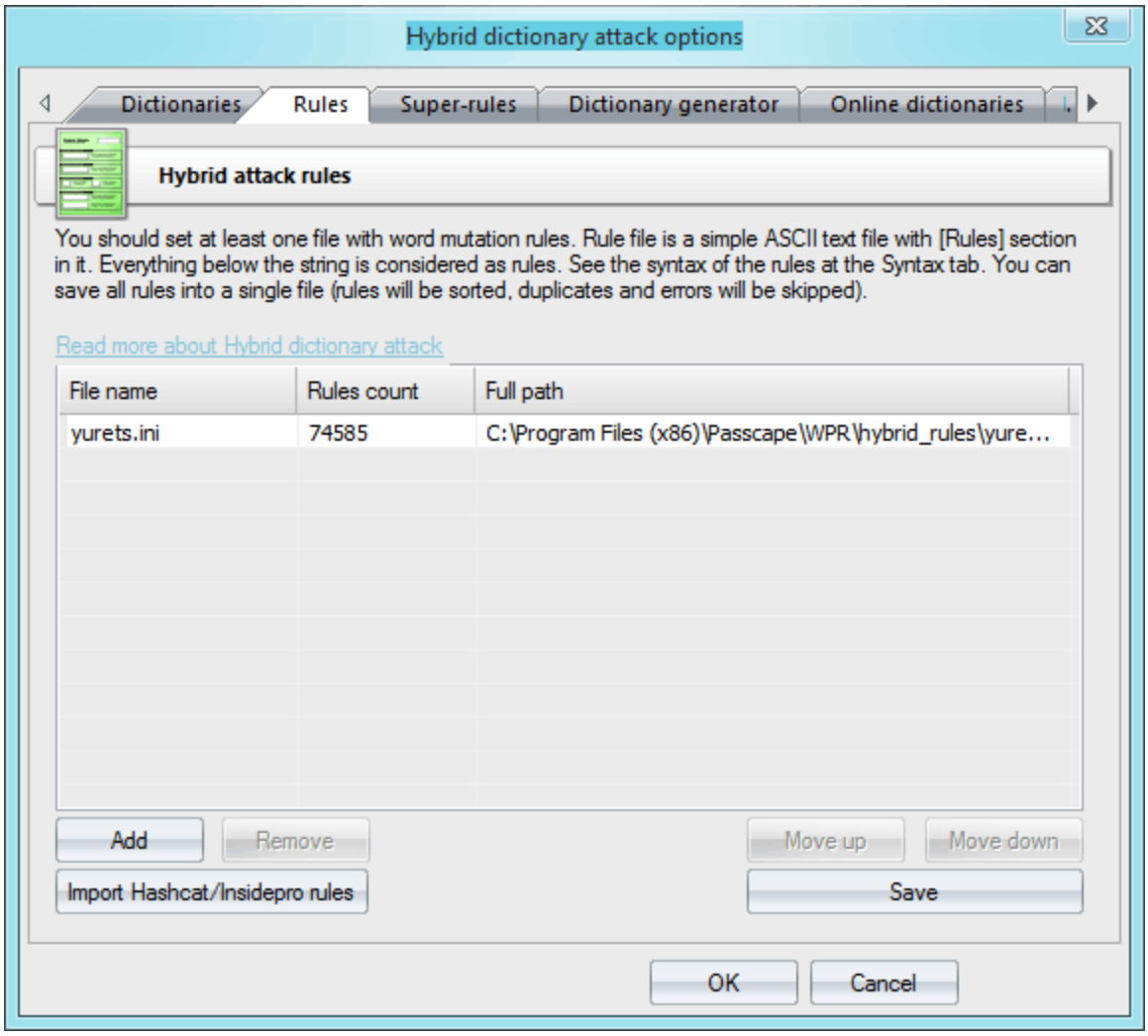
Hybrid dictionary attack settings are grouped in 7 tabs:
1. **Dictionaries** - for setting up source dictionaries.
2. **Rules** - files with set of rules.
3. **Super-rules** - ones to be applied over the top of regular rules
4. **Dictionary generator**, where you can create files of words obtained from the hybrid attack.
5. **Online dictionaries** - for downloading new dictionaries to the application.
6. **Hybrid syntax** - complete description of all rules with examples.
7. **Rule tester**, where you can test your rules.

Wordlists to be used in the attack are set on the first tab. Traditionally, the application supports wordlists in ASCII, UTF8, UNICODE, PCD, RAR and ZIP format. The position of the files on the list can be altered. For example, you may want to move smaller dictionaries up the list or the other way. During the attack, they will be used one after another, according to their position on the list.

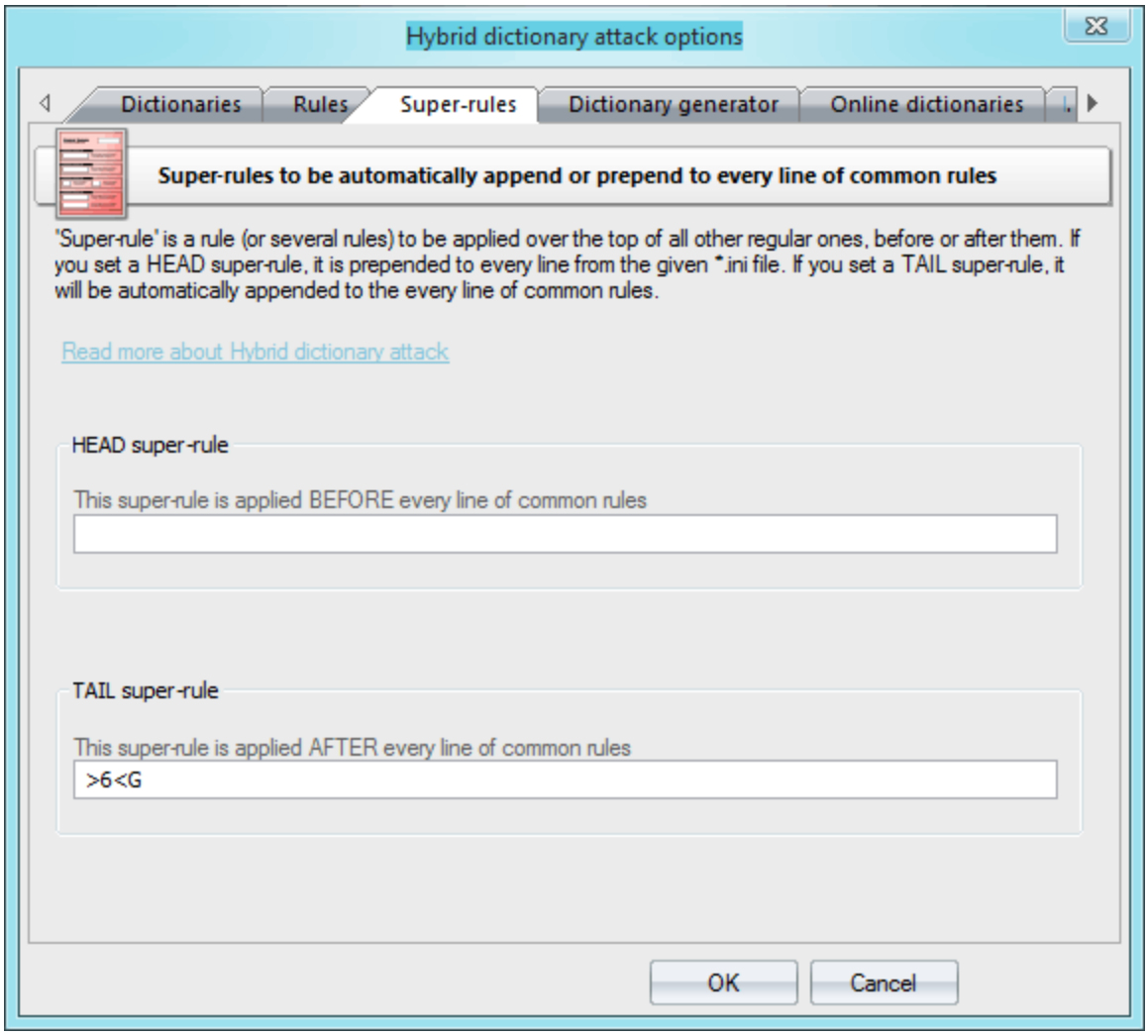


On the '**Rules**' tab, define at least one file with password mutation rules. The format of the rules file is

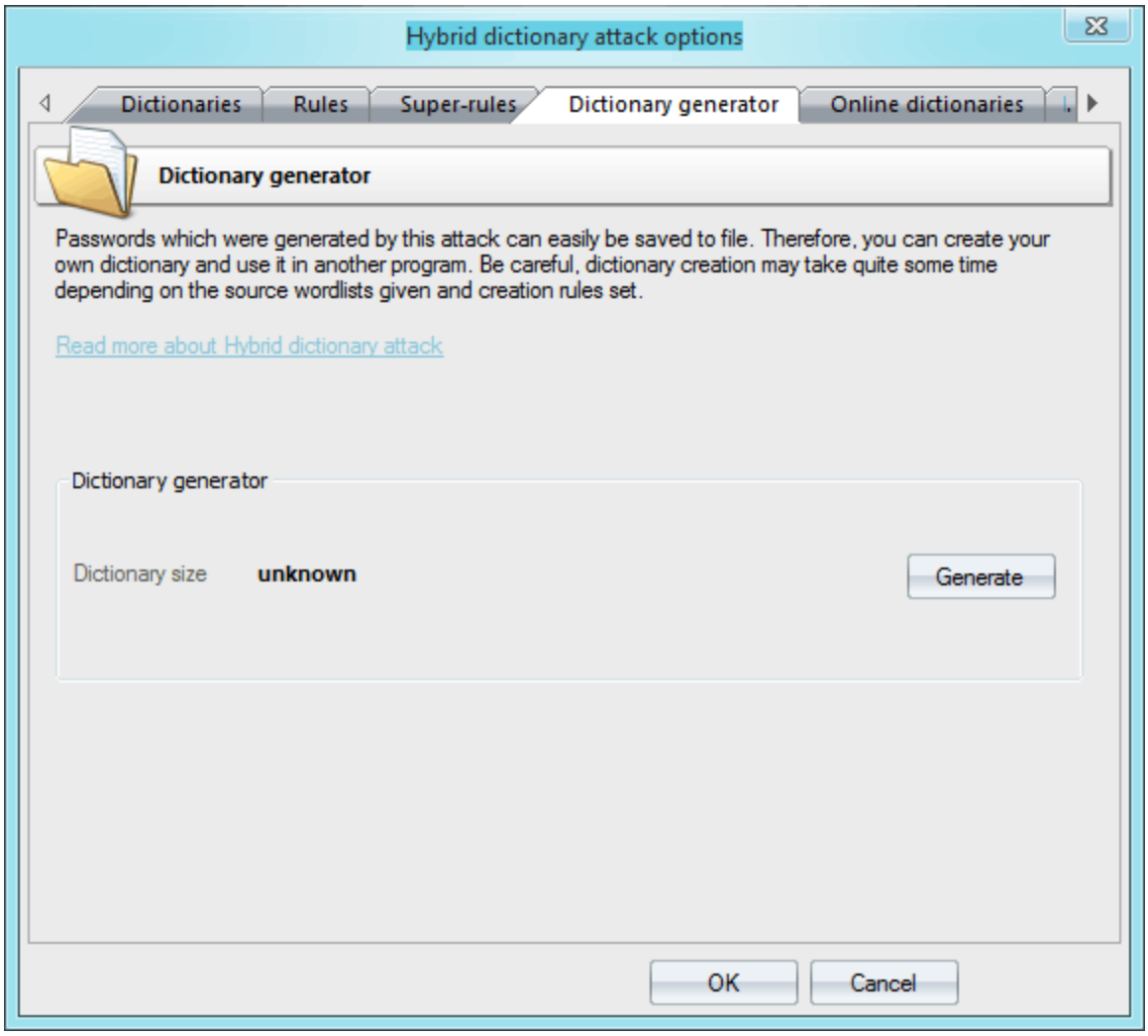
quite trivial; it is a plain-text ASCII file with the '**[Rules]**' string. Anything above this string is considered as comments and ignored by the program. Whatever goes below this string is considered as rules. Each string can contain several rules, applicable to a source word. The exclusion is the **aN** rule. This rule must not be on the same line with other rules. If a string contains multiple rules per word, those rules are parsed left to right. For example, if you apply the rule '@pc$a$b$c' to the source word 'password', at the output you will get 'Asswordabc'. The maximum length of an output word may not exceed **256** characters.



'**Super-rules**' is a rule (or several rules) to be applied over the top of all other regular ones, before or after them. For example, you can set 'a8' tail super-rule to create all possible case combinations after a common mutation has been done. So '/asa4' rule from l33t.ini file will become '/asa4a8', '/csc(' will become '/csc(a8', etc. Yet another one example: setting the '>6<G' head rule allows you to skip all words of less than 6 or greater than 16 characters, before starting a common mutation. This is a helpful feature once you decide to add the same rule to all text lines of the selected *.ini files. There's no need to modify them all. Be careful though, the 'aN' super-rule may increase the total number of generated passwords drastically.

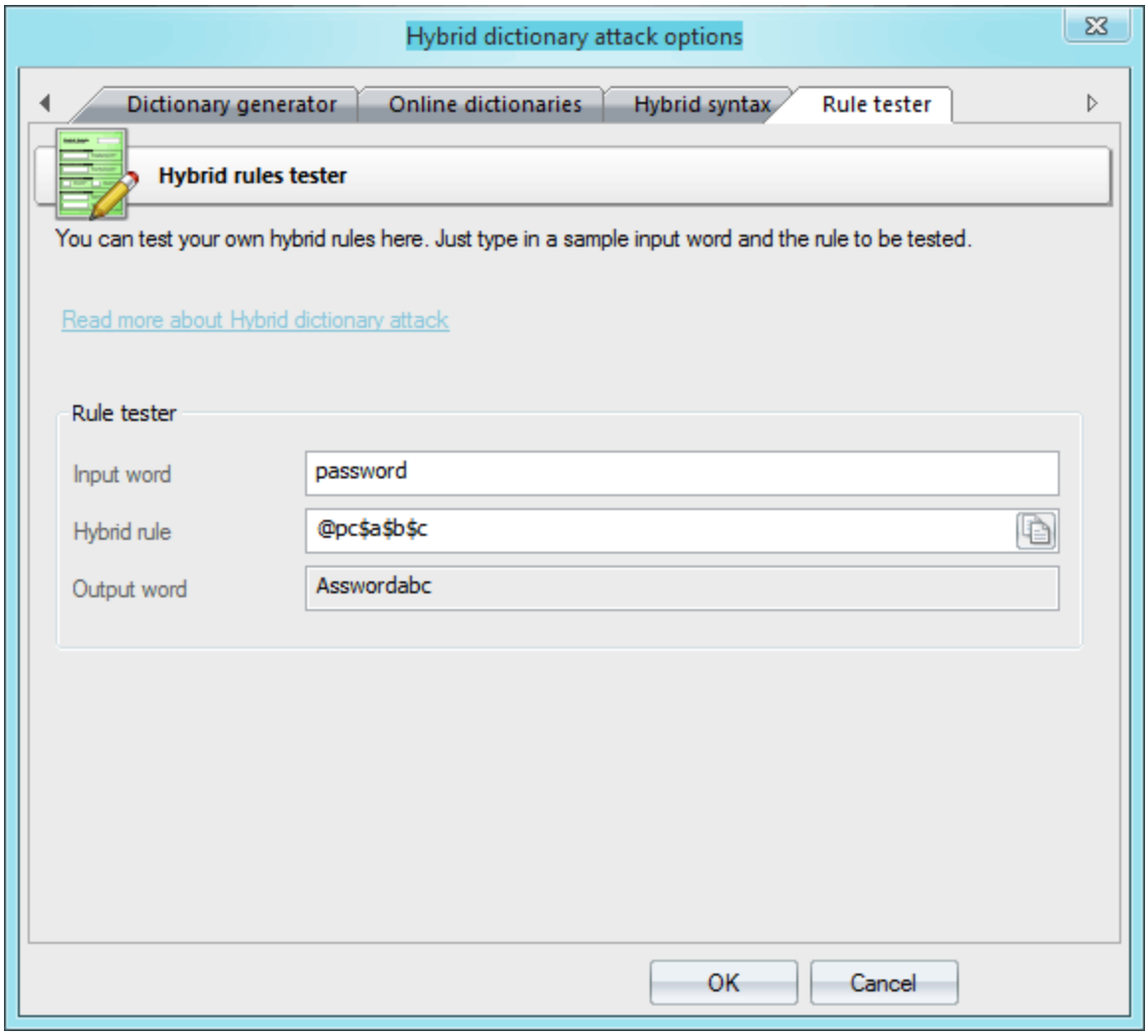


The '**Dictionary generator**' tab is designed for generating dictionaries obtained from an attack. Further on, those dictionaries could be used, for example, in other applications. To generate a dictionary, specify a source dictionary and a set of mutation rules for it. The size of a target file may exceed 2 GB. Be careful, the dictionary generation process may take considerable time!



You can download additional wordlists for the attack using '[Online dictionaries]' tab.

If you want to create your own set of rules, you can use the last two tabs as sources of help. While the '**Hybrid syntax**' tab gives mere descriptions of available rules, on the last tab you can actually test them by specifying a source word and a rule for the hybrid attack. Forward your rule sets to us; if we find them interesting/useful, we will include them in the default distribution of the program.



Rules description for the hybrid dictionary attack

Several rules at a line are allowed to be set.
Rules (if any) are processed from the left to the right.
Maximal line length is limited to **256** characters.
Maximal output word length is limited to **256** characters.
White space is ignored as long as it is not used as a parameter.
A line started with # character considered as a comment.
All text before the '**[Rules]**' line is considered as comment.
N and M always start at 0. For values greater than 9 use A..Z (A=10, B=11, etc.).
The following rules should be at the last position of a line: aN, ?iN[C], ?i[C], ?oN[C], ?o[C], ?iZ[C], ?oZ[C].
Don't change the names of the standard rule files. Some ones are used by the program.
?iN[C], ?i[C], ?oN[C], ?o[C] ?iZ[C], ?oZ[C] rules use the following predefined charsets (you can use custom character sets though):

digits - 0123456789
loweralpha - abcdefghijklmnopqrstuwxyz
upperalpha - ABCDEFGHIJKLMNOPQRSTUVWXYZ
alpha - abcdefghijklmnopqrstuwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
special - !@#$%^&*()-_+=~`[]{}|\:;"'<>,.?/ "

loweralphanumeric - abcdefghijklmnopqrstuwxyz0123456789
upperalphanumeric - ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
alphanumeric - abcdefghijklmnopqrstuwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
printable - abcdefghijklmnopqrstuwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()-_+=~`[]{}|\:;"'<>,.? /

Rules

Rule	Example	Input	Output	Description
:	:	password	password	Do nothing to the input word
{	{	password	asswordp	Rotate the word left
}	}	password	dpasswor	Rotate the word right
[	[	password	assword	Delete the first character
]	]	password	passwor	Delete the last character
c	c	password	Password	Capitalize
C	C	password	pASSWORD	Anti-capitalize (lowercase the first character, uppercase the rest)
d	d	password	passwordpassword	Duplicate word
f	f	password	passworddrowssap	Reflect word
k	k	password пароль	зфыыцщквgfhjkm	Convert word using alternative (first after default) keyboard layout. The rule works in both directions. For example, if there's Russian keyboard layout installed previously in the system, the rule should convert word 'password' to Russian 'зфыыцщкв', and Russian word 'пароль' to 'gfhjkm'. This is very helpful when looking for non-English passwords. If only one language is installed in the system, the rule does nothing.
K	K	password	passwodr	Swap last two characters
l	l	password	password	Convert all characters to lowercase
q	q	password	ppaasssswoorrdd	Duplicate all symbols
r	r	password	drowssap	Reverse word
t	t	PassWord	pASSwORD	Toggle case of all characters
u	u	password	PASSWORD	Convert all characters to uppercase
U	U	my own password	My Own Password	Capitalize all words delimited with space (upper-case the first character and every character after a space)
V	V	password	PaSSWoRD	Vowels elite
v	v	password	pASSWoRD	Vowels noelite

Rule	Example	Input	Output	Description
'N	'4	password	pass	Truncate the word to N character(s) length
+N	+1	password	pbssword	Increment character at position N by 1 ASCII value
-N	-0	password	oassword	Decrement character at position N by 1
.N	.4	password	passoord	Replace character at position N with character at position N+1
,N	,1	password	ppssword	Replace character at position N with character at position N-1. Where N > 0.
<N				Reject (skip) the word if it is greater than N characters long
>N				Reject (skip) the word if it is less than N characters long
aN				Check all possible symbol cases for the word. N is a maximal length of the word to apply this rule for.
DN	D2D2	password	paword	Delete the character at position N
pN	p3	key	keykeykey	Copy word N times
TN	T1T5	password	pAsswOrd	Toggle case of the character at position N
yN	y3	password	paspaswor d	Duplicate first N characters
YN	Y3	password	paswordord d	Duplicate last N characters
zN	z3	password	ppppasswo rd	Duplicate the first character of the word N times
ZN	Z3	password	passwordd ddd	Duplicate the last character of the word N times
$X	$0$0$ 7	password	password0 07	Add character X to the end of the word
^X	^3^2^ 1	password	123passwo rd	Insert character X at the beginning of the word
@X	@s	password	paword	Remove all characters X from the word
!X				Reject (skip) the word if it contains at least one character X
/X				Reject (skip) the word if it does not contain character X
(X				Reject (skip) the word if the first character is not X
)X				Reject (skip) the word if the last character is not X
eX	e@	mike@y ahoo.co m	mike	Extract a substring starting at position 0 and ending up before first occurrence of X character (do nothing if X is not found)
EX	E@e.	mike@y ahoo.co m	yahoo	Extract a substring starting right after first found X character and till the end of the string (do nothing if X is not found)
%MX				Reject (skip) the word if it does not contain at least M instances of the character X
***XY**	*15	password	possward	Swap characters at positions X and Y
=N				Reject (skip) the word if the character at position N is not equal to the X

Rule	Example	Input	Output	Description
X				
iNX	i4ai5bi6c	password	passabcword	Insert the character X in position N
oNX	o4*o5*	password	pass**rd	Overwrite a character in position N with the character X
sXY	ss$so0	password	pa$$w0rd	Replace all characters X with Y
xNM	x4Z	password	word	Extract a substring of up to M characters length, starting from position N.
INX-Y	rl0/-/r	google.com	google.com/	Insert the character X at position N if previous character at position N is not Y.
INX+Y	rl0.+.r	password.	password..	Insert the character X at position N if previous character at position N is Y.
ONX-Y	O0-+p	password	-assword	If the character at position N is not Y, overwrite it with X character.
ONX+Y	O0P+p	password	Password	If the character at position N is Y, overwrite it with X character.
RNM+Y	R01+a	password	assword	Remove character at position N if character at position M is Y
RNM-Y	R40-b	password	passord	Remove character at position N if character at position M is not Y
?iN[C]	?i0[digits]	password	0password, 1password ... 9password	Insert a character from a charset [C] into position N of the word. Where C should be either a predefined charset name or a custom character set itself.
?iZ[C]	?iZ[digits]	password	password0, password1 ... password9	Insert a character from a charset [C] into last position of the word. Where C should be either a predefined charset name or a custom character set itself.
?i[C]	?i[special]	password	~password, !password ... password_, password+	Insert a character from a charset [C] into every position of the word. Where C should be either a predefined charset name or a custom character set itself.
?oN[C]	?o1[upperalpha]	password	pAssword, pBssword ... pZssword	Overwrite a character at position N with a character taken from a charset [C]. Where C should be either a predefined charset name or a custom character set itself.
?oZ[C]	?oZ[upperalpha]	password	passworA, passworB ... passworZ	Overwrite a character at last position with a character taken from a charset [C]. Where C should be either a predefined charset name or a custom character set itself.
?o[C]	?o[-=.]	password	-assword, =assword ... passwor.	Overwrite a character at every position of the word with a character taken from a charset [C]. Where C should be either a predefined charset name or a custom character set itself.

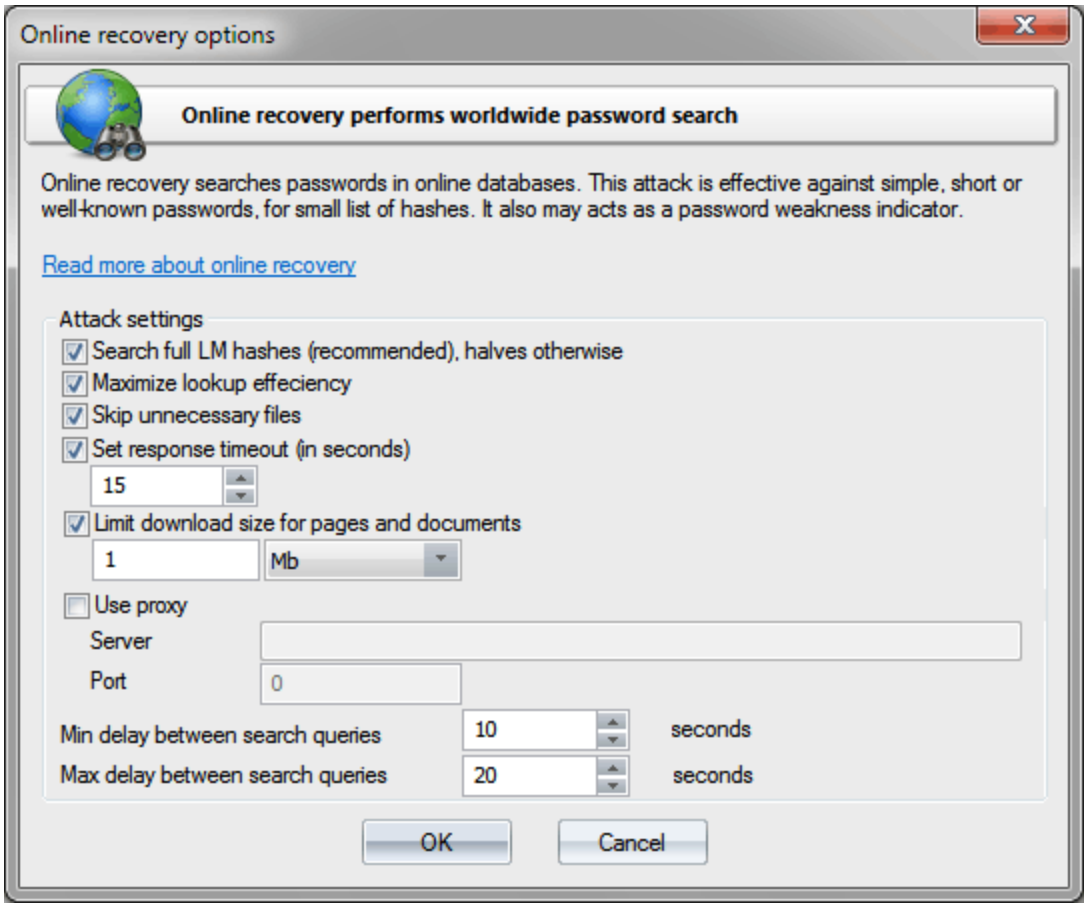
Additional
Windows Password Recovery distribution kit comes with extended sets of password mutation rules:
hybrid_rules/english_words.ini file contains basic rules for English passwords.
hybrid_rules/nonenglish_words.ini holds common rules for non-Eglish passwords.
hybrid_rules/simple_dates.ini - a lot of rules with dates, months, seasons, etc.
hybrid_rules/l33t.ini - rules to freak words (based on leet dictionary). For example, password->p@$$w0rd
...

Looking for a convenient way to handle as much passwords as possible? Downloading the full set of more than 180000 sorted and duplicate-free rules.

2.8.2.12 Online recovery

Online recovery (developed by Passcape Software) finds passwords using Internet search engine servers. It deals fairly well with short and frequently-used passwords. Among its drawbacks are low operating speed and poor suitability for handling large hash lists.

Online recovery has been developed by Passcape Software and is an improved online password finder. To find passwords, the program consecutively submits a special search request for each hash to a search engine and then downloads the password files found and analyzes their contents. Online recovery is relatively slow; therefore, it is appropriate for small hash lists. In addition, the passwords found are usually limited to simple vocabulary and short combinations. One way or the other, this attack can be quite useful; for example, when auditing passwords, as a simple vulnerability detector for certain systems.



Online recovery options

- **Search full LM hashes** - use the entire 16-byte hash when searching LM hashes. If this option is not set, the search will be carried out over the 8-byte halves. To ensure more efficient search and get rid of some stray traffic, it is recommended that this option is set. It is ignored when searching NT hashes.
- **Maximize lookup efficiency** - increase password lookup efficiency not affecting the attack speed. It is also recommended to always set this option.
- **Skip unnecessary files** - do not check some unnecessary files if they are suspected to not contain passwords.
- **Response timeout** - set the maximum allowed Web resource response time.
- **Limit download size** - limit download file size. Some hash databases have enormous size, even despite that often they do not contain passwords. Therefore, for slow Internet connections and to restrict stray traffic, it is recommended to set a limit on the size of download pages. Unfortunately, there is no way to figure out what is in the data to be downloaded; therefore, this option is determined exclusively by your preferences and capabilities.
- **Use proxy** - use proxy server for looking up passwords
- **Min/max delay between search queries** - minimum and maximum delays between two consecutive requests to the search server. Some search servers may reject search requests if they go in series from the same IP address with a very short time interval (normally less than 10 seconds). Despite that Windows Password Recovery has an internal request randomizer, which allows to drop this delay significantly (to as little as 1 and 2 seconds respectively), the safe values when a search request will be definitely processed by the server are min=15 and max=30 seconds. Certainly, the attack speed depends is in direct relation to these two options.

Be careful! Online recovery may generate a lot of Internet traffic!

2.8.2.13 Passcape table attack

Passcape Rainbow Tables are the next logical development of simple pre-calculated tables. They are most suitable for the recovery of meaningful combinations and complex passwords of literally unlimited length.

The original method of simple rainbow tables

The operating principle of simple rainbow tables consists of setting a character range (for example, a..z) and maximum password length, followed by the calculation of all the possible variants and the generation of millions of chains. Each chain is calculated by the formula:

P0 -> hash(P0) -> H1 -> R(H1) ->
P1 -> hash(P1) -> H2 -> R(H2) ->
P2 ...

where **P** – password, **hash** – hashing function, **R** – reduction function. Thus, from the original password, the hashing function produces a hash, which the reduction function then converts into the next password, and the process repeats all over again and generates chains. Each chain stores only the original and final value. Storing only the first and the last hash is an operation leading to compromise and saving memory at the cost of time spent on cryptanalysis.

To recover a sought password, it undergoes hashing and the reduction function and then is looked up in the table. For that purpose, a key chain is generated beginning with **R(Hn)** up until the maximum chain length. If **Hn** is obtained with the password used when creating the table, we finally get the key that matches the key of the respective chain. This last key was saved in the table along with the first key of the chain. Using the first key of the chain, we can recover the entire chain, in particular, the value right before **R(Hn)**. That is actually the key that was used for generating **Hn**, our sought password.

Operating principle of Passcape rainbow tables

Recovery using Passcape rainbow tables is pretty much the same as recovery using simple rainbow tables. However, unlike the latter, it is sort of a hybrid of Fingerprint and simple table attacks, where instead of setting a specific character range passwords are validated within a so-called 'word footprint' range. The idea of the Fingerprint attack developed at Passcape comes down to taking the source dictionary and creating a bank of word footprints (fingerprints), necessary for validating the password, out of that dictionary; then, during the attack, we search for all possible variants of words that consist of two such footprints.

Similar to the Fingerprint attack, Passcape rainbow tables first create a bank of footprints for words from a user's wordlist. The word footprint bank is an analog to character set in simple rainbow tables. It is used for both creating Passcape tables and validating passwords. Thus, a Passcape rainbow table consists of one or more ***.prt** files (the actual tables) and a bank of word footprints (***.prti**), which can be engaged only with tables that were created with it.

There are a number of advantages in using word footprints instead of character sets when creating tables:

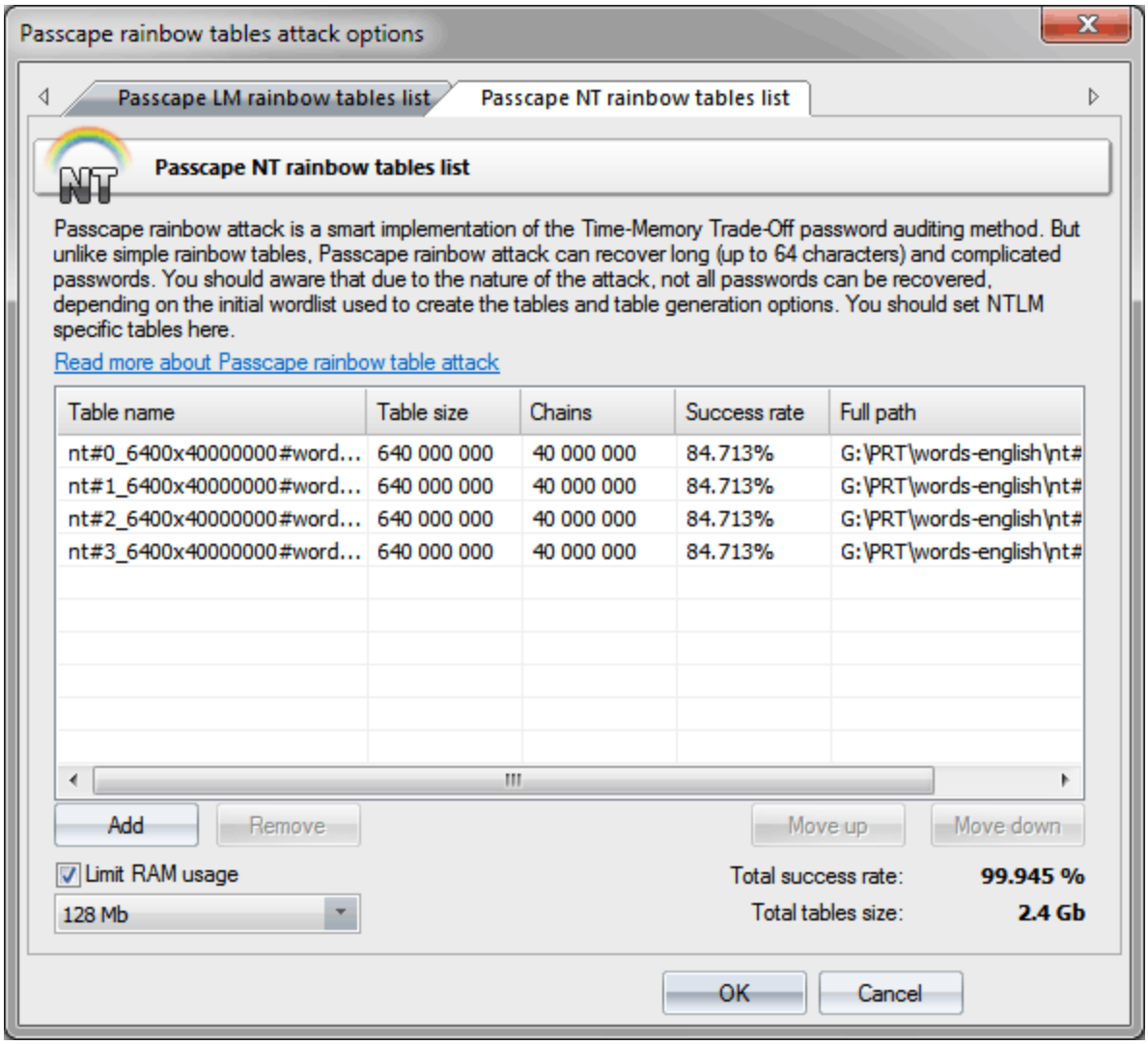
- The length of passwords validated with Passcape tables is literally unlimited. Unlike with simple rainbow tables, which practically cannot be created for passwords longer than 9 characters, with Passcape tables one can recover both one-character and 50-character password with same probability.
- Character set in the regular table greatly affects its critical parameters: the wider the character range, the greater the chain length or the total number of chains for storing success rate (percentage of success in finding password) of the table must be. In a Passcape table, a character set does not

affect the critical parameters of the table.
- Plain tables have certain difficulties when generating tables for validating passwords in national character sets; not all programs properly handle such tables, and not all can create them. With Passcape rainbow tables, when generating tables, for example, for Russian passwords, one can simply specify the source dictionary in Russian.
- With Passcape tables, passwords are searched for using more meaningful combinations; however, that largely depends on the source dictionary.

These can be referred to as drawbacks of Passcape rainbow tables:
- Not all source dictionaries are equally suitable for the tables. Using large dictionaries (normally greater than 1 MB) generates too large of a footprint bank; respectively, creating tables may require significant time and resources.
- Using dictionaries with long words or phrases is discouraged due to the above mentioned reason.
- Rainbow table attack consumes a great deal of resources: the footprint bank must fully fit the computer's RAM.

Passcape rainbow table attack settings

Passcape rainbow table attack settings are rather trivial. Specify one or several *.prt tables, which should reside in the same directory as the footprint bank (*.prti file). Since this attack consumes more RAM than the attack that uses simple rainbow tables, it is recommended to limit the amount of RAM that can be consumed by adjusting the respective option.

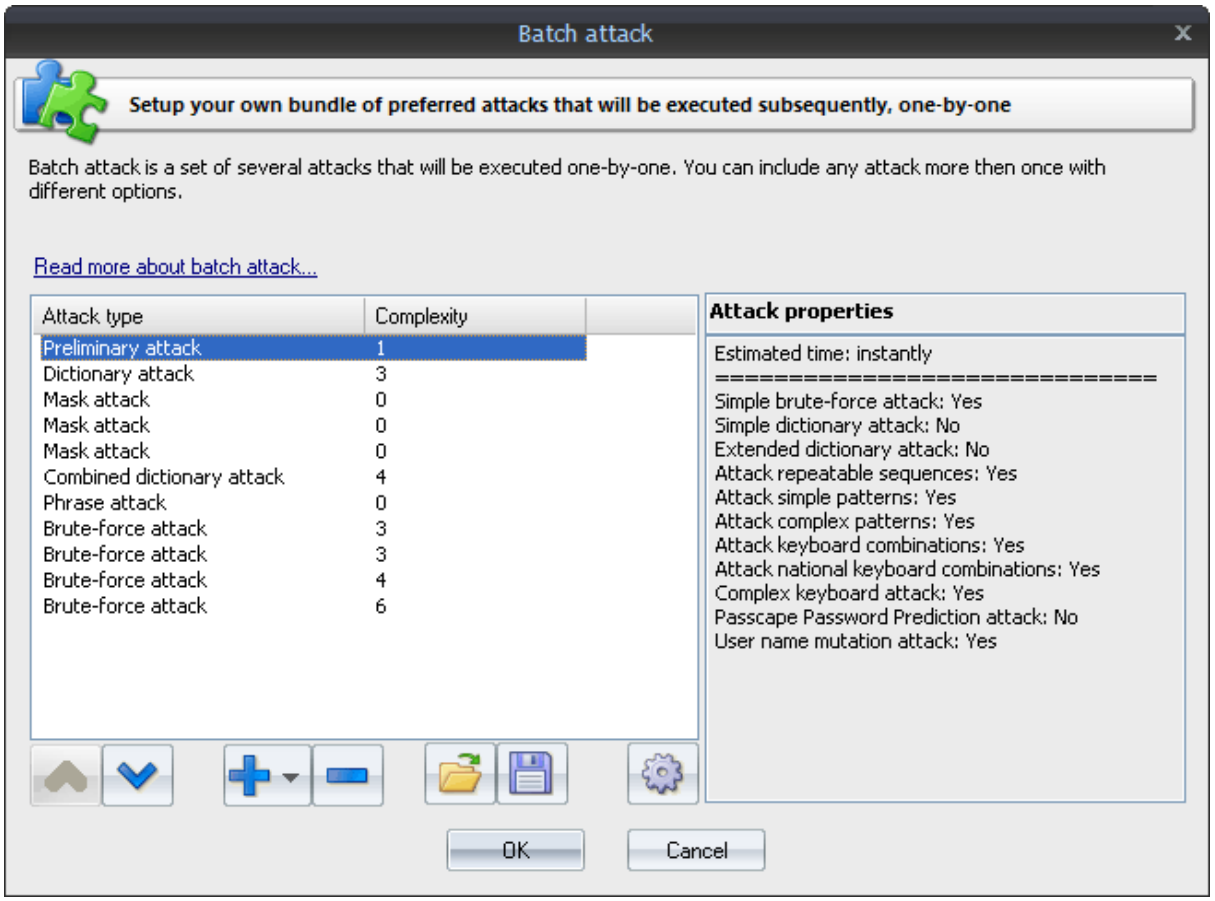


Tables can crack only the hash function they were created for, i.e. NT tables can crack only NT hash!

To create your own tables, you can take advantage of the respective tool.
You can download sample Passcape tables for this attack from our website.

2.8.2.14 Batch attack

Since each attack covers its own password range, sometimes, in order to fully recover password hashes, you have to run several different attacks one after another. The basic idea behind the batch attack (developed by Passcape Software) is to create a list/batch of attacks to be run one after another, so that you could launch all those attacks with a single click of the mouse and not hassle with configuring each of them individually every time you need them.



The batch attack options are available as a list that you can extend or cut (buttons [+] and [-]). Each attack on the list can be moved up or down (buttons [^] and [v]), and its settings can be edited. A batch can include several attacks of the same kind, but of the attacks can have different settings. The pane to the right of the selected entry displays the properties of the selected entry; brief specifications of the attack and the estimated time the attack will take to complete.

2.8.2.15 GPU: Brute-force Attack

A GPU brute force attack is fully identical to a regular brute force attack, except that passwords are searched by the graphics processing unit of your PC instead. It is no secret that the performance of modern graphics cards is an order of magnitude greater than that of CPUs; this makes them a convenient tool for heavy calculations, such as password recovery. It is important to understand that calculations using graphics cards have a number of disadvantages. For example, some algorithms with a great number of conditional jumps and other checks demonstrate extremely poor performance on GPU, and in certain cases it may be even lower than on a regular CPU.

Anyway, the software supports brute force password search using GPU. You can compare the performance indicators of GPU vs. CPU calculations through the respective menu item of the application or present it visually through the '**Reports**' menu.

The configuration of GPU brute force attack consists of three parts:
1. Choosing a character set for the search.
2. Specifying password length.
3. Configuring the graphics processing unit.

Choosing a character set for the search

When choosing a character set for a brute force attack, you are normally guided by empirical considerations. For example, if the expected password consists of lower-case Latin characters and digits, it makes sense to choose the range 'a-z, 0-9'. The smaller the character set, the sooner the attack completes.

On the other hand, there is always a chance to make a wrong choice of the expected character set. If at least one character of the password to be recovered is not included in the specified character set, the password will not be found.

At the bottom of the attack settings dialog, you can see the total number of passwords that match the specified character set and password length.

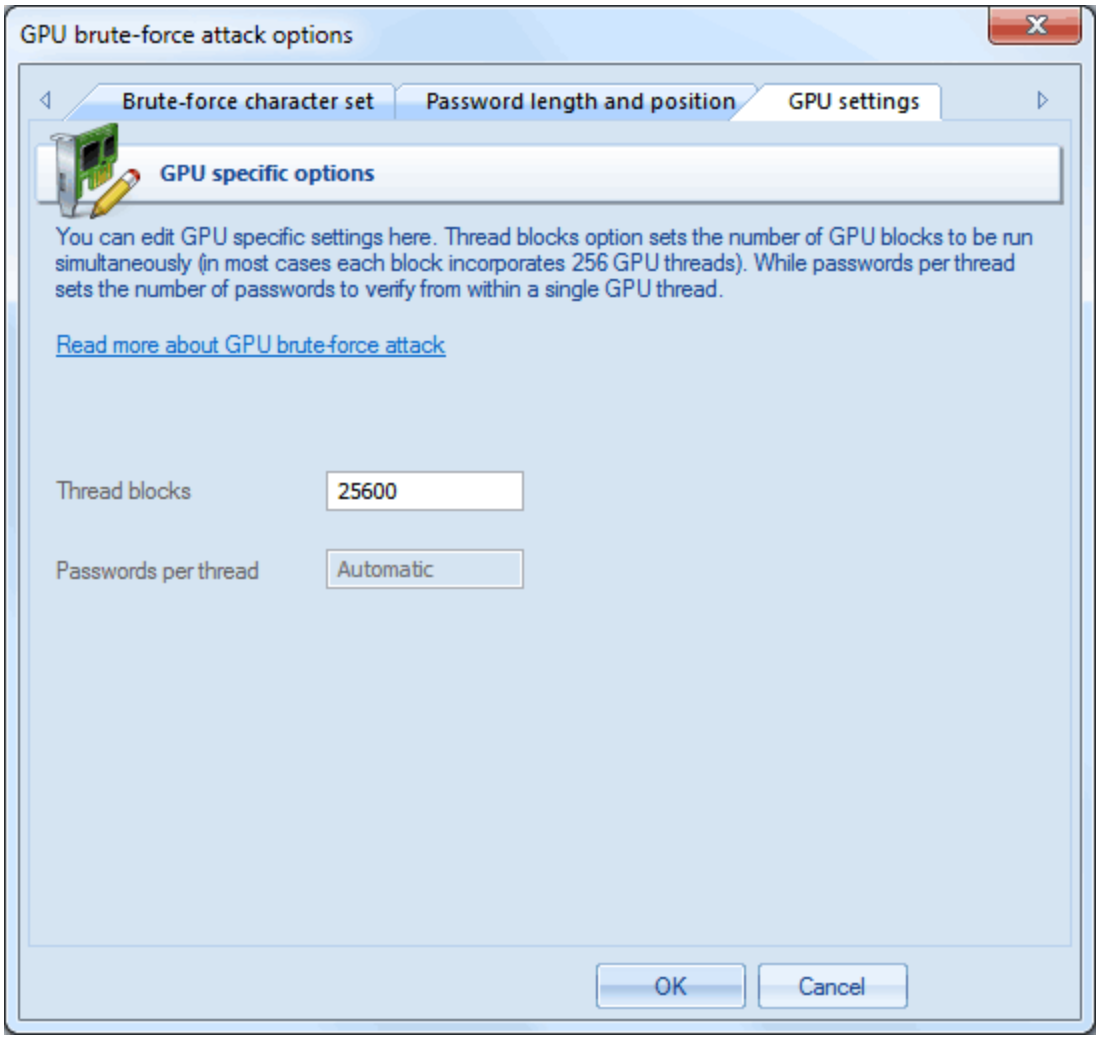
It is important to know that LM passwords in Windows are always converted to upper case; that significantly cuts the range of passwords to be searched!

Specifying password length

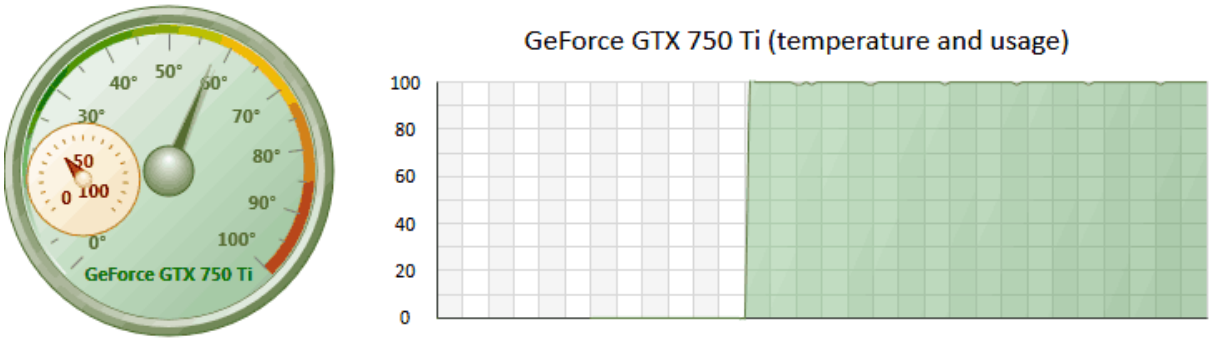
On the second tab of the options page, set the minimum and maximum length of searched passwords. As an alternative to minimum length, you can set the source password, which the search would begin with. The maximum length of LM in Windows operating systems is 7.

Configuring graphics processing unit

Before you can use it in an attack, you must first select the graphics card on the respective menu item.



GPU configuration consists of only 1 parameter: the number of thread blocks to run on GPU. Each block consists of 256 threads. Thus, if you set the number of blocks to 25600, the GPU will run 25600*256=6553600 threads. Each GPU thread can check multiple passwords. The total number of checked passwords greatly depends on other options. Setting the **ThreadBlocks** parameter smaller than 10000 on modern graphics cards, in the majority of cases, leads to poor performance. To avoid performance degradation, after setting up the parameter and running the attack, make sure the GPU load chart has close to 100% plain graphic without peeks (see the screenshot below).

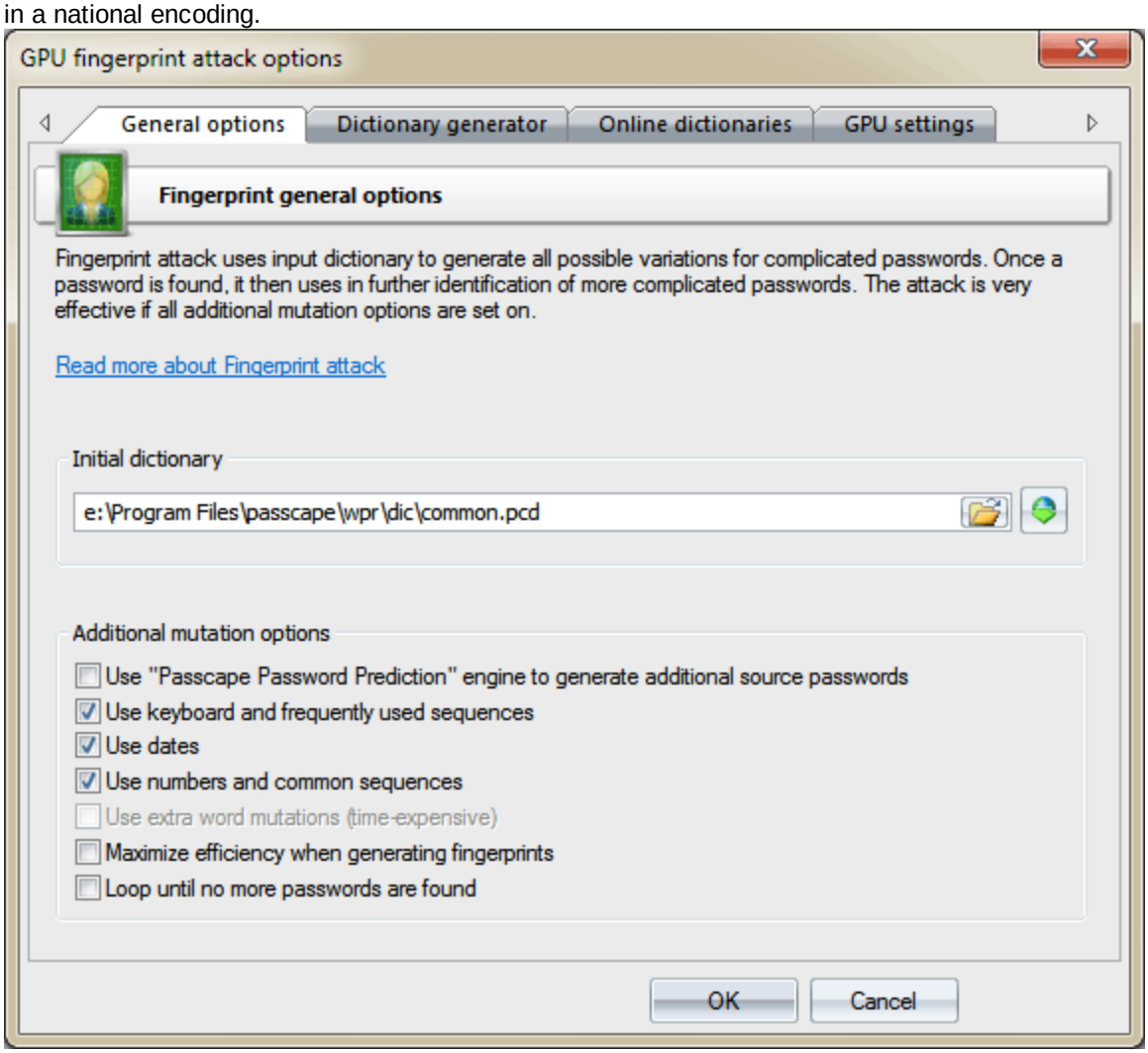


2.8.2.16 GPU: Fingerprint attack

Fingerprint attack is a brand-new tool for recovering complex passwords, which could not be decrypted in a common way. The idea of the attack is that here, to recover a password, we take neither individual words from the source dictionary, like in the Dictionary attack, nor even word combinations, like in the Combined attack, but so-called "fingerprints". So every word from the source dictionary is used for generating several fingerprints. If some password is found during the attack, it participates in generating new fingerprints, and the attack goes another round. Implementing GPU computing power allows to increase the recovery speed drastically. Fingerprint options consist of 4 parts:

General options

Before launching the attack, specify the source dictionary to be used for creating the fingerprints. The software comes with common.pcd dictionary, optimized for this attack, but you can use yours or download one off the Internet (*'Online dictionaries*' tab). There are no certain requirements to the source wordlist, except one: it must not be too large; otherwise, the attack will take significant time. You can use dictionaries with national passwords, if you suspect that the sought password contains characters in a national encoding.



Here is the way the fingerprints are generated: first, a word from the source dictionary is broken into one-character passwords, then - into 2-character, etc. For instance, the source word **crazy** is broken into one-character fingerprints. So we get:

c
r
a
z
y

Now, into two-character:
cr
ra
az
zy

Next, three-character:
cra
raz
azy

And, finally, four-character:
craz
razy

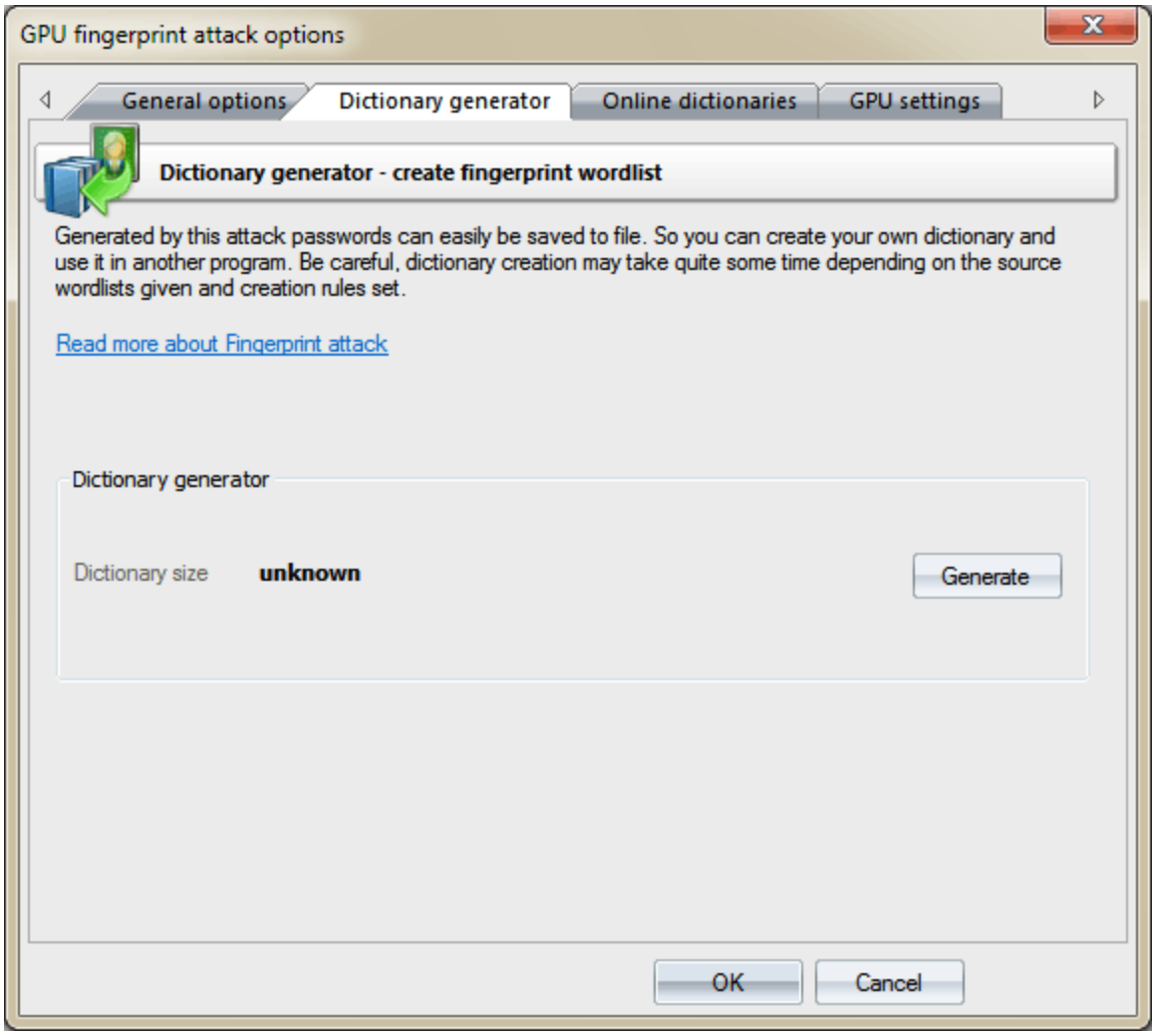
We have got 5+4+3+2=14 fingerprints, not counting the source word.
All word from the source dictionary are broken into fingerprints. After this, all the fingerprints are dumped into a single database, naturally, discarding duplicates. So we have got a database of fingerprints that would be used for checking passwords by gluing all the fingerprints with each other and finding the match.

The real fingerprint generation algorithm is a bit more sophisticated. Moreover, there is an option in the attack settings, **Maximize effeciency when generating fingerprints**, which maximizes the efficiency (at the expense of speed) by generating additional fingerprints.

Let's take a look at the remaining options.
Use PPP engine to generate additional passwords - use passwords found in other attacks when generating fingerprints.
Use keyboard and frequently use sequences - add keyboard combinations and common sequences to fingerprint bank.
Use dates - add dates to fingerprints.
Use numbers and common sequences - use digits and simple combinations of letters.

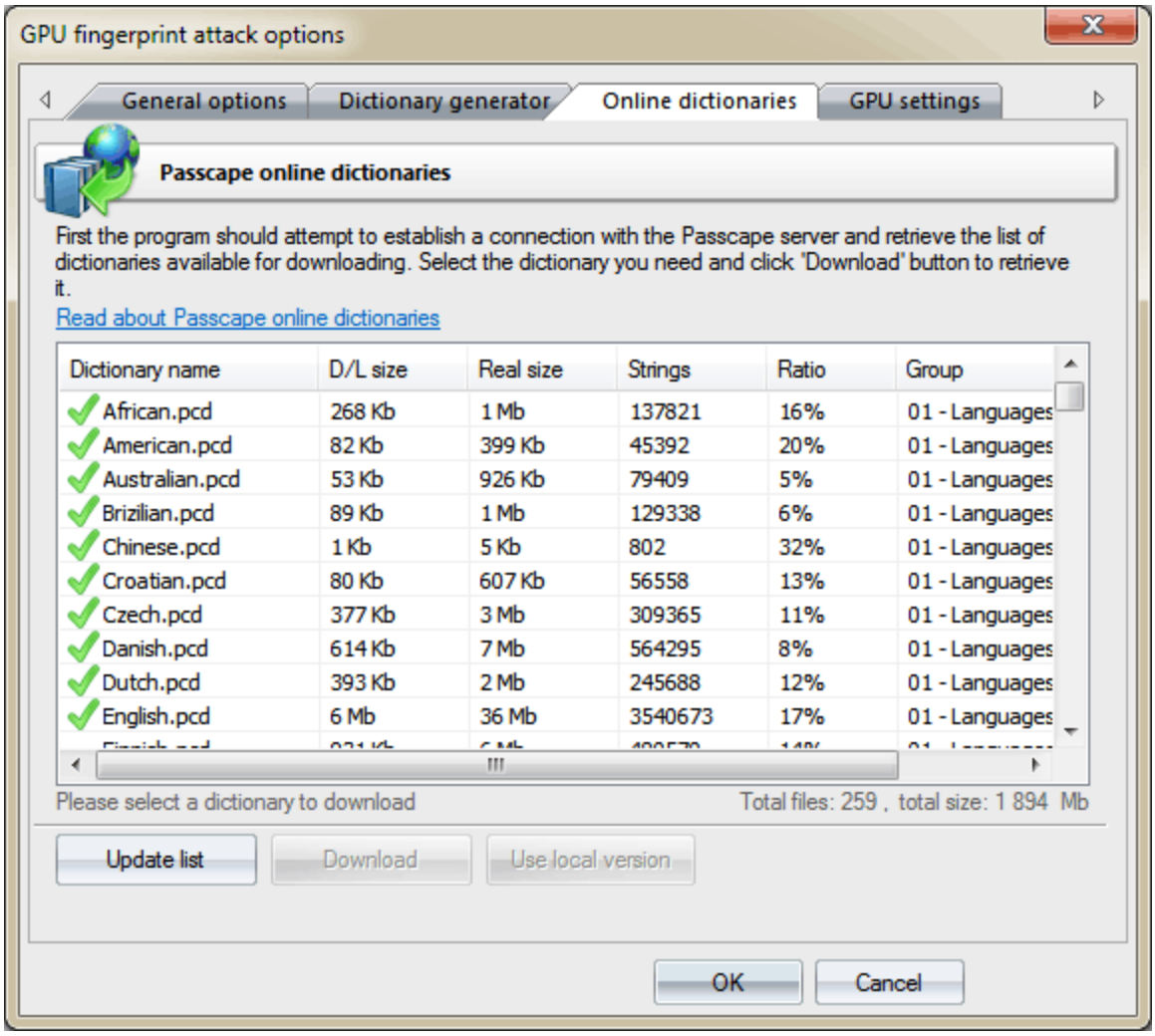
The most careful attention should be paid to the option **Loop until no more passwords are found**. That is where fingerprint attack can really show itself off. Here is how it works: if at least one password is found during an attack, when the attack is over, the password participates in generating new fingerprints, and the attack runs again. This option works great on large lists of hashes and on password history hashes. However once the option is set, you will not be able to proceed the attack from the last saved position.

Dictionary generator
The second tab with the settings allows to create and save a custom dictionary using current options of the fingerprint attack. Be careful; the dictionary may take up a lot of space on your PC's hard disk drive.



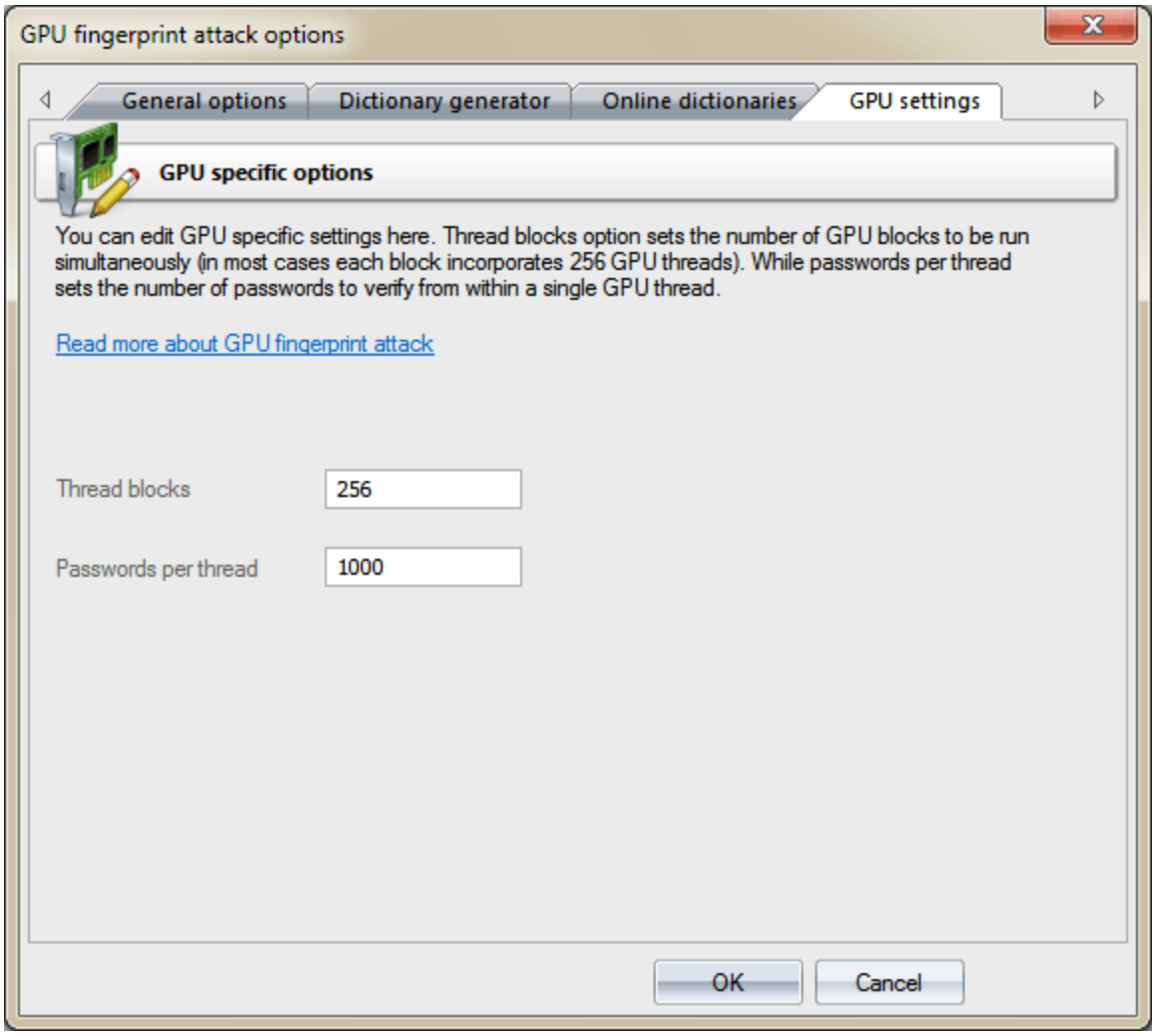
Online dictionaries

On the third tab, you can download source wordlists for fingerprint attack from the Internet. Be careful, not all the dictionaries suit fine for the attack.



GPU settings

Before you can use it in an attack, you must first select the graphics card in the General Options menu.



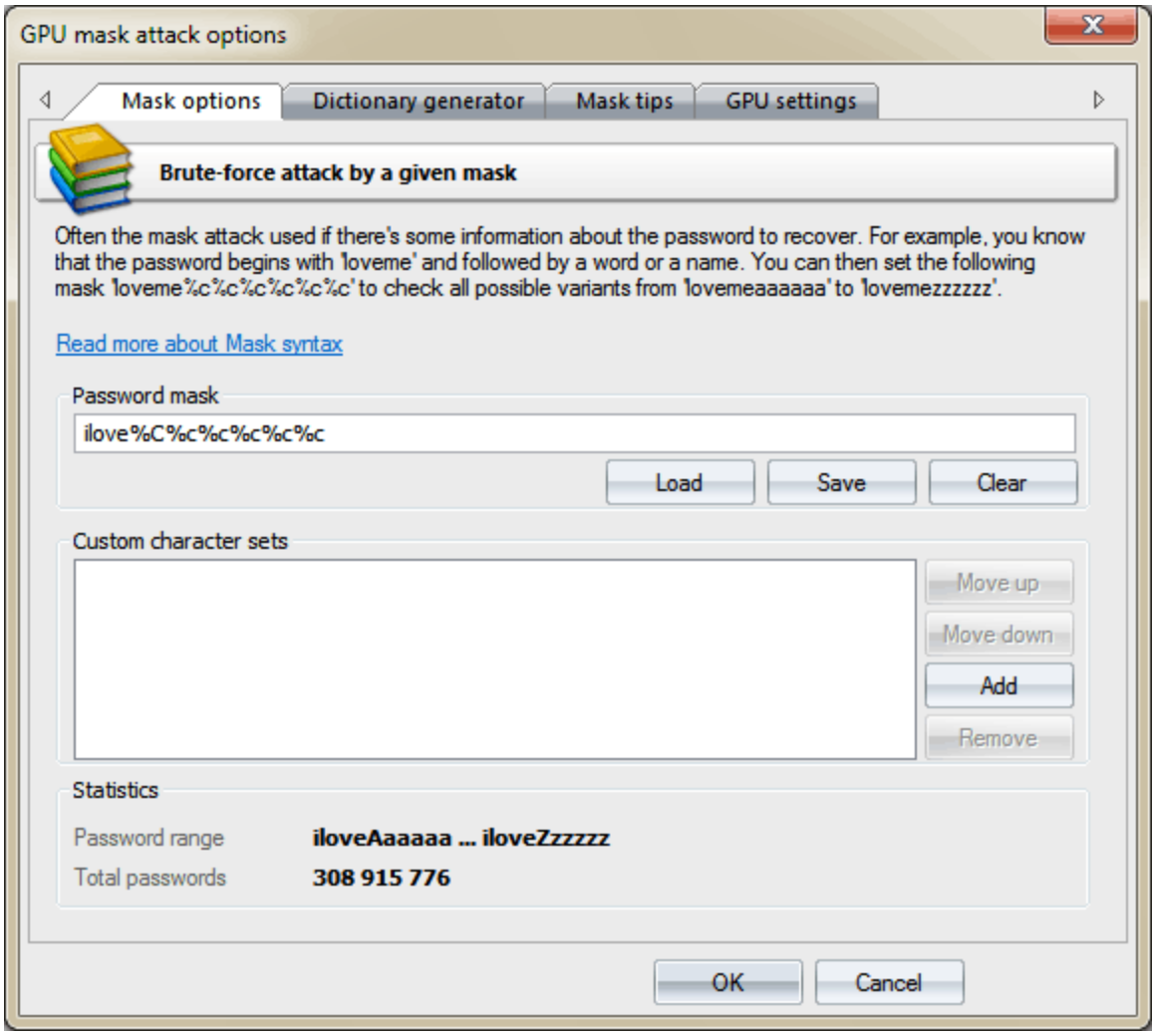
GPU configuration is pretty simple and consists of only two parts:

1. Setting the number of parallel graphics card's blocks, where passwords would be searched. Typically, each block consists of 256 threads. Thus, if you set the number of blocks to 256, the GPU will run 256*256=65536 threads. The total number of checked passwords for one call to GPU kernel will be 256*ThreadBlocks*PasswordsPerThread. In our case 256*256*1000 = 65 536 000 passwords. Setting the **ThreadBlocks** parameter smaller than 256 on modern graphics cards, in the majority of cases, leads to performance degradation.
2. Setting the number of passwords to be search from a single thread. The greater the value, the lower the overhead associated with launching threads, and the higher the search speed. However, setting too great a value may hang the computer or cause significant fluctuations in the current search speed, displayed on the attack status tab. This is caused by the fact that task completion time on the GPU exceeds the time required for refreshing the current state of the attack. Setting too big numbers may cause a system failure.

2.8.2.17 GPU: Mask attack

Mask options

Mask attack is an irreplaceable tool when you know a fragment of the password or have any specific details about it. For example, when you know that the password consists of 12 characters and ends with the *qwerty*, it is obvious that searching the entire 12-character range of passwords is unreasonable (and useless, for it takes ages to complete). All what would be required in this case is to guess the first 6 characters of the sought password. That is what mask attack is for.

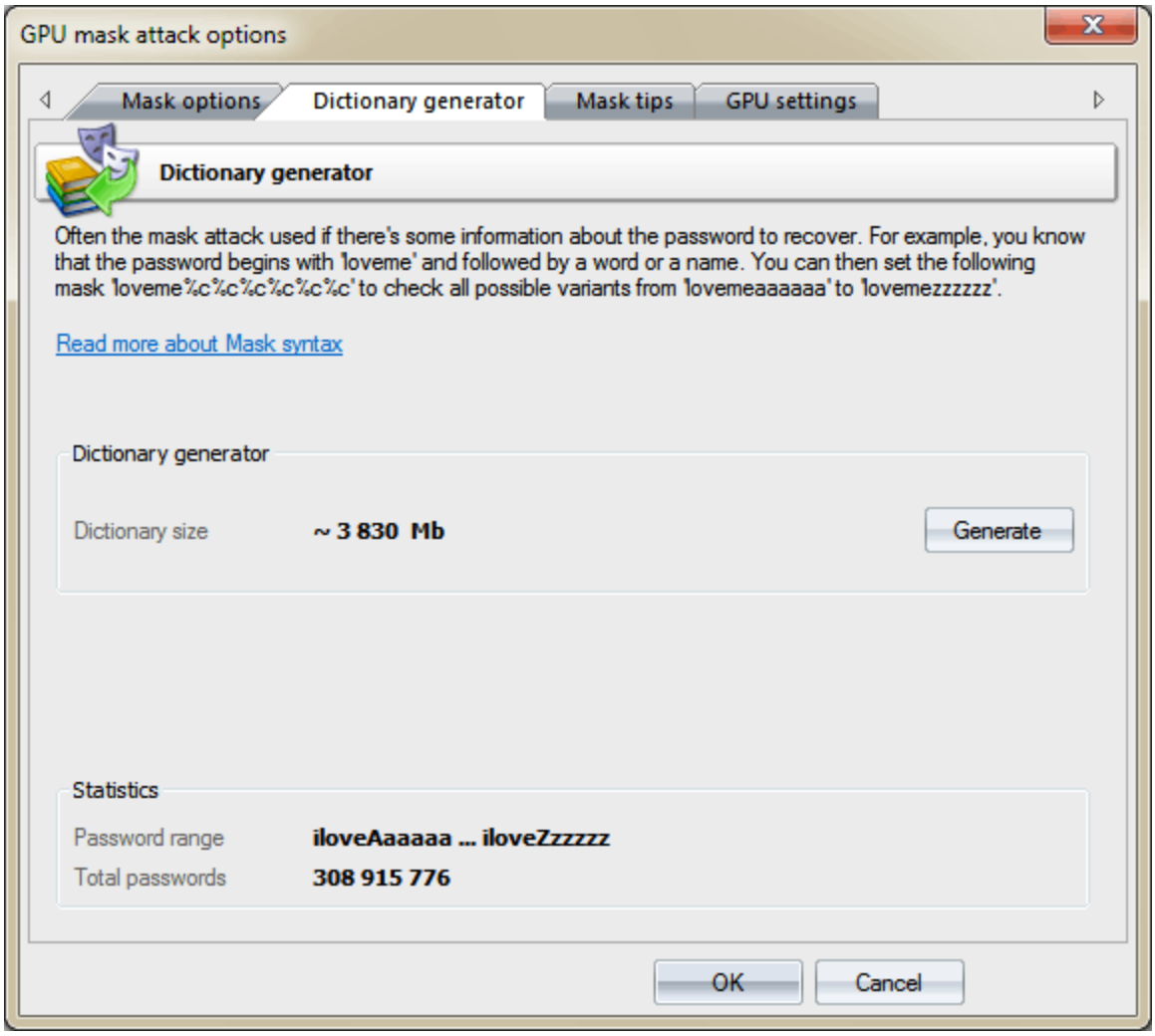


In our case, we could define the following mask: **%c%c%c%c%c%cqwerty**. That means that the program would successively check the following combinations: aaaaaaqwerty, aaaaabqwerty, aaaaacqwerty .. zzzzzzqwerty. If the original password is *'secretqwerty'*, it perfectly hits the range.

The mask input field is used for setting the rule, which will be used by the program to guess the password. If the mask is set correctly, below you will see the range of characters generated by the mask. User-defined masks can be saved to disk.

Dictionary generator

By switching to **Dictionary generator** tab, you can generate your own dictionary by a given mask, and save it to disk. This feature available in Advanced edition of the program only.



Mask tips

Third tab of the mask options contains a short description of the mask syntax and a couple of examples. The mask syntax is pretty simple and consists of static (unmodifiable) and dynamic (modifiable) characters. Dynamic characters always have a leading %. For example, if you set the mask **secret%d%d%d%d**, the program will generate 10000 passwords (secret0000, secret0001, secret0002 .. secret9999).

Windows Password Recovery supports the following dynamic mask sets:

- **%c** lower-case Latin characters (a..z), 26 symbols
- **%C** upper-case Latin characters (A..Z), 26 symbols
- **%#** full set of special characters (!..~ space), total 33 symbols
- **%@** small set of special characters (!@#$%^&*()-_+= space), 15 symbols
- **%?** all printable characters with ASCII codes of 32..127
- **%*** all ASCII characters (codes 1 through 255)
- **%d** one digit (0..9)
- **%r(x-y)** user-defined characters with serial ASCII codes between x and y
- **%r(x1-y1,x2-y2...xn-yn)** set of several non-overlapping sequences of ASCII characters. Useful

for defining custom character sets; e.g., of OEM characters.

- **%1[2,3..9]** a character from user defined charset 1..9
- **%%** standalone static character %

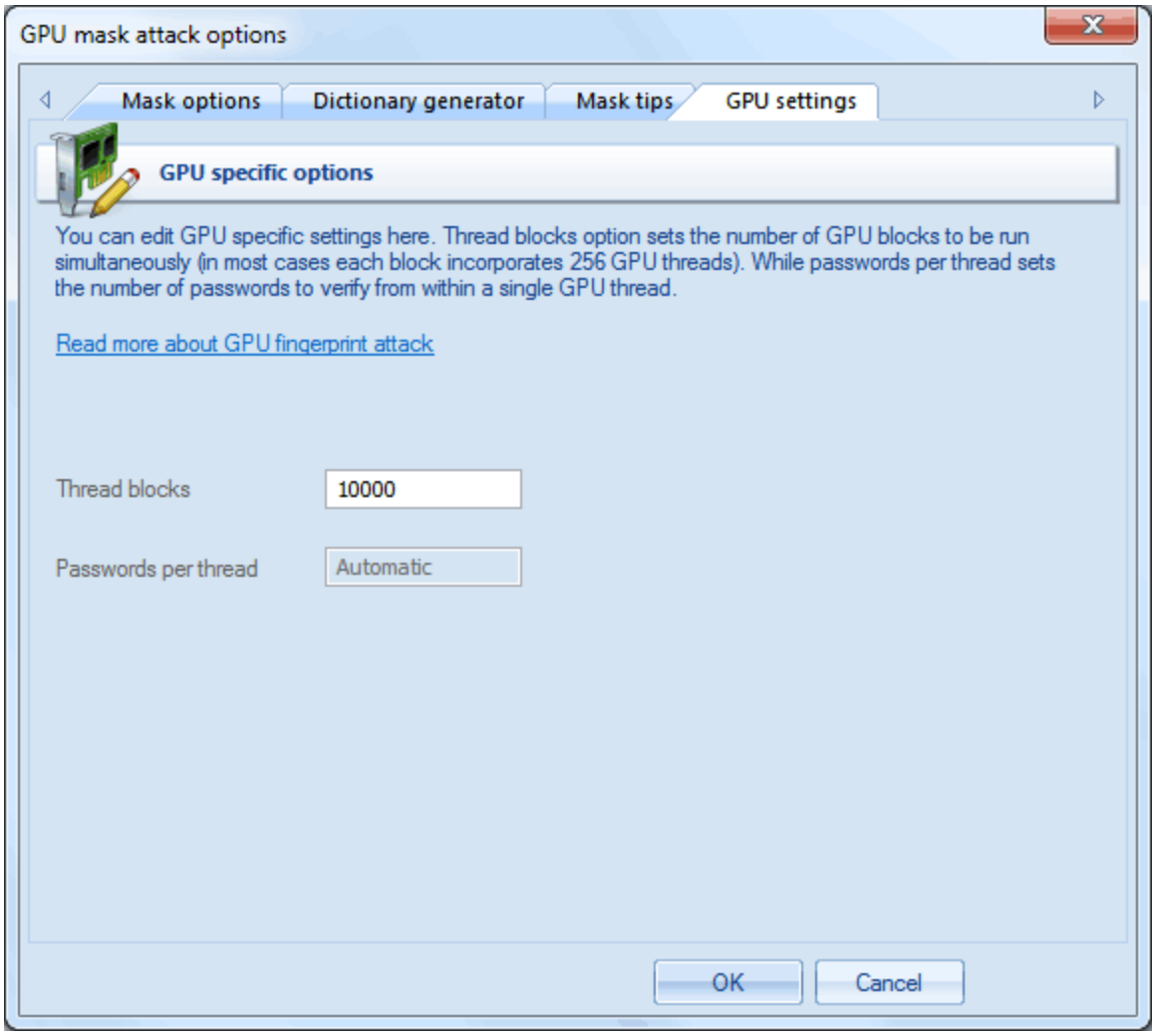
Examples:

test%d - will generate password range test0..test9, 10 passwords total
test%d%d%d%d - test0000..test9999, 10000 passwords
test%r(0x0600-0x06ff) - test_ .. test_, 256 passwords with Arabic character at the end
%#test%# - _test_..~test~, 1089 passwords
%1%1%1pin%2%2%2 - aaapin000.. zzzpin999, где %1 is user-defined charset 1 (a..z), and %2 - the second user-defined charset 0..9
ilove%1%1%1%1%1 - iloveaaaaa .. iloveZZZZZ, %1 is user charset (a..z, A..Z)

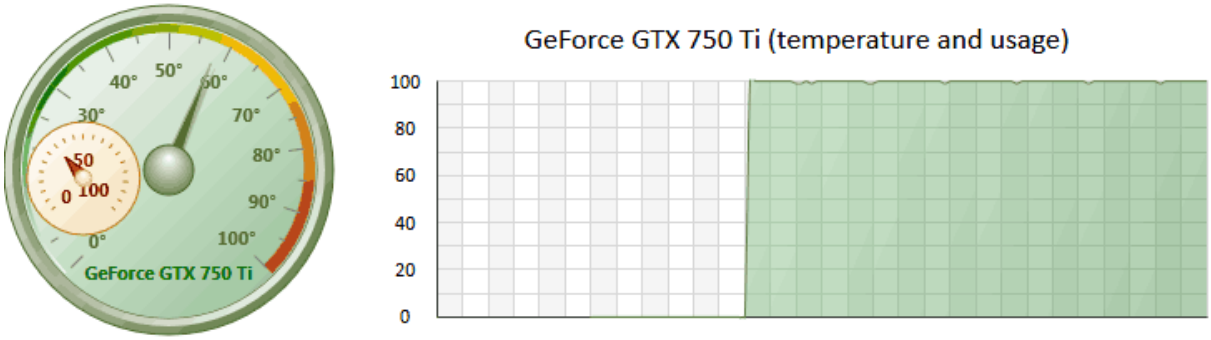
The GPU mask attack syntax differs slightly from one used in a regular mask attack. The main difference is that in GPU-based attack you can not set numbers between x and y and can not set user-defined range of variable length, i.e. the following syntax will not work for GPU mask attack:
%d(x-y)
%1[2,3..9](min-max)

GPU settings

Before you can use it in an attack, you must first select the graphics card in the General Options menu.



GPU configuration for the Mask attack consists of only 1 parameter: the number of thread blocks to run at a single call to GPU. Each block consists of either 128 or 256 threads. Thus, if you set the number of blocks to 10000, the GPU will run 10000*256=2560000 threads for one call to GPU kernel. Each GPU thread can check multiple passwords. The total number of checked passwords depends greatly on other options. Setting the **ThreadBlocks** parameter smaller than 10000, in the majority of cases, leads to poor performance. To avoid performance degradation, after setting up the parameter and running the attack, make sure the GPU load chart has close to 100% plain graphic without peeks (see the screenshot of NVidia GTX 750Ti running with 15000 blocks).



2.8.2.18 GPU: Dictionary-force Attack

Oftentimes, when creating passwords, users add certain characters in the beginning, end or even middle of the word. To recover passwords of this specific kind, we have come up with a GPU-based dictionary attack, which is something between simple dictionary attack and brute force attack.

This attack works as follows:

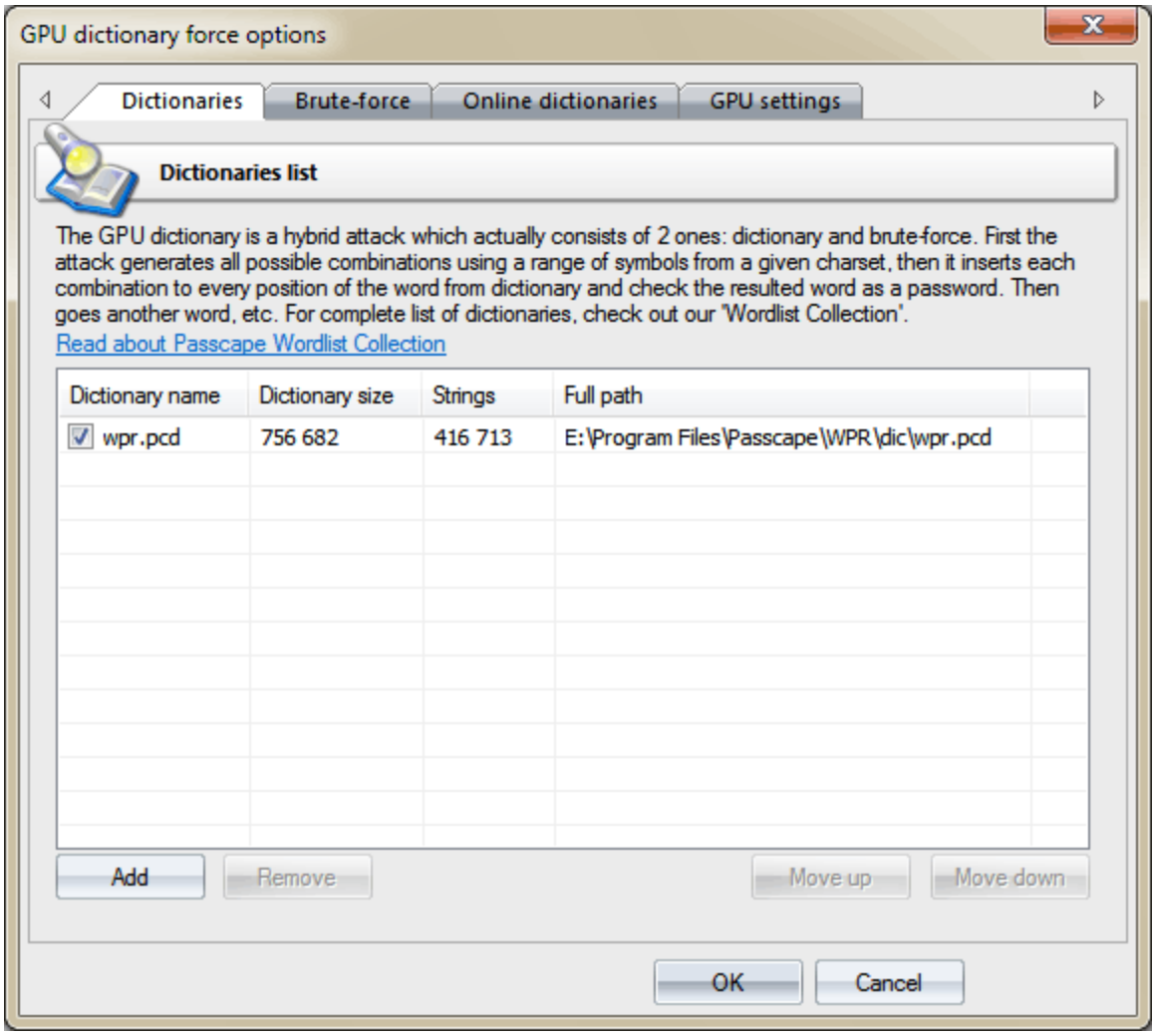
- Reads the first word from the dictionary.
- According to the defined character set and the minimum/maximum length of the search range, generates all the possible variants.
- Those variants (characters) are then added to the beginning, end or middle of the word. The position within the word, where the generated sequences are to be inserted, can be specified at your discretion.
- Then goes next dictionary word, etc.

For example, if we specify a search character range between **0** and **9**, and the range length between **1** and **2**, the program will generate 100 combinations: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 .. 99. Then these sequences will be added to the beginning, middle or end of the word. Thus, for the word **test**, if the sequences are going to be inserted to every listed position, the program will check the following passwords:
0test, 1test .. 99test
t0est, t1est .. t99est
te0st, te1st .. te99st
tes0t, tes1t .. tes99t
test0, test1 .. test99
Total - 100*5=500 variants.

Let's take a closer look at the attack settings.

Dictionaries

On the **Dictionaries** tab, you can specify the list of dictionaries to be used in the attack. The program supports text wordlists in the following formats: ASCII, UNICODE, UTF8, RAR, ZIP, as well as encrypted/packed dictionaries in the native PCD format, developed by our company. To deactivate a dictionary, simply clear the check box by its name. Thus, although the dictionary remains on the list, it will be ignored by the attack. The software comes with the default 400000-word dictionary. You can order the full set of dictionaries, that's over 6 GB in size, on CD or take advantage of the dictionaries available online.

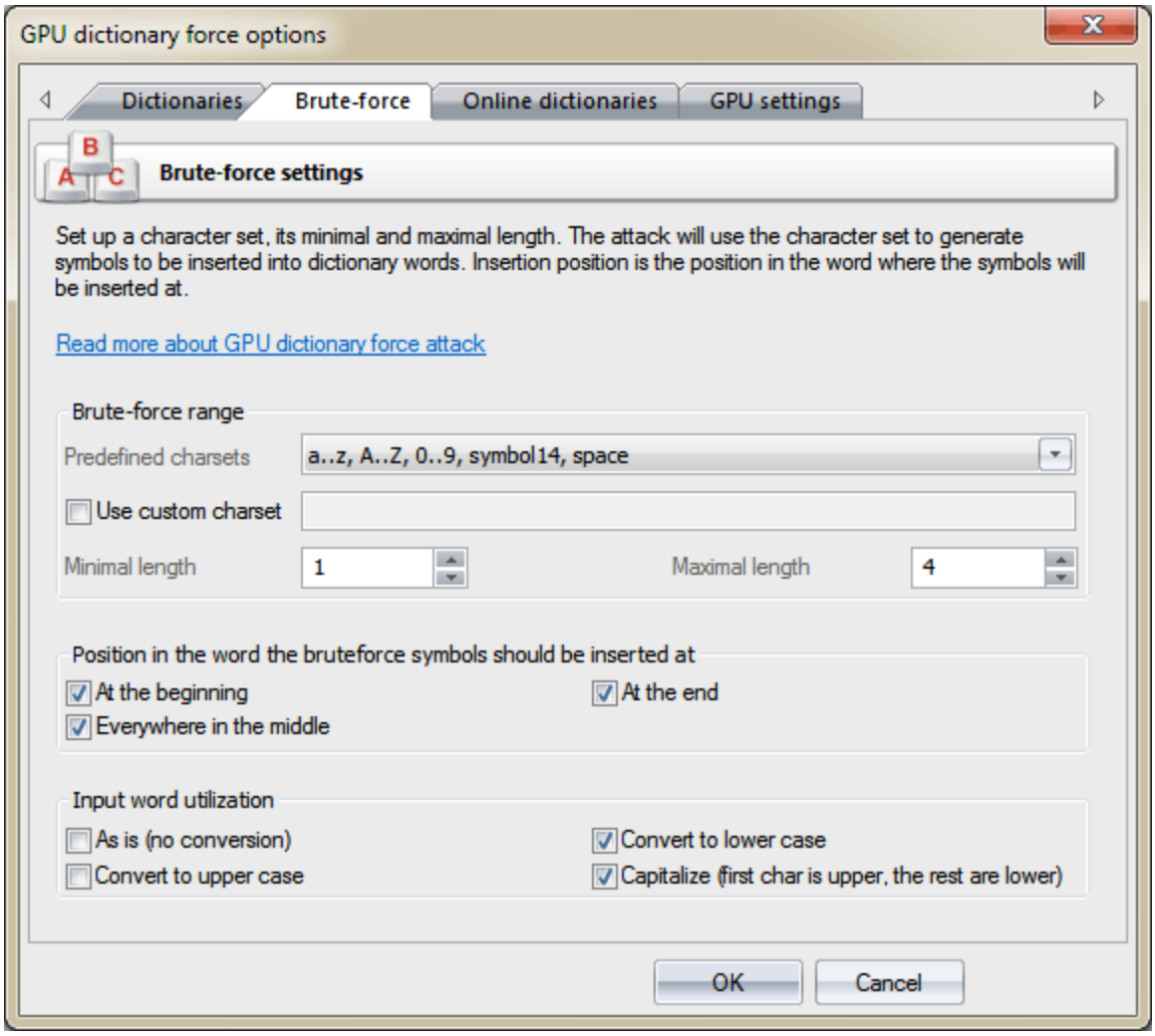


Search range

On this tab, you should set up the range of characters to be inserted into base words, its minimum and maximum length. When setting up a range, you can use the existing templates or, having checked the respective check box, define your own one. When selecting the maximum range length, keep in mind that specifying a too wide or too small value is inadvisable. While in the first case the password search speed may drop down to 0, specifying a too narrow range of characters to be searched raises the overheads related to the irrational use of the computing power of the GPU.

In the second group of options, specify the position in the word, where the characters of the searched range would be inserted.

And, finally, the third group of settings - these are in charge of preprocessing the words from the source dictionary. Selecting the **As is** option makes the program use the source word as is, not converting to upper or lowercase. The number of passwords to be searched grows in direct proportion to the number of options specified in this group. On the other hand, the program is smart enough to not use repeat words. For example, the word **12345678**, even if all the conversion options are set, will be used only once.



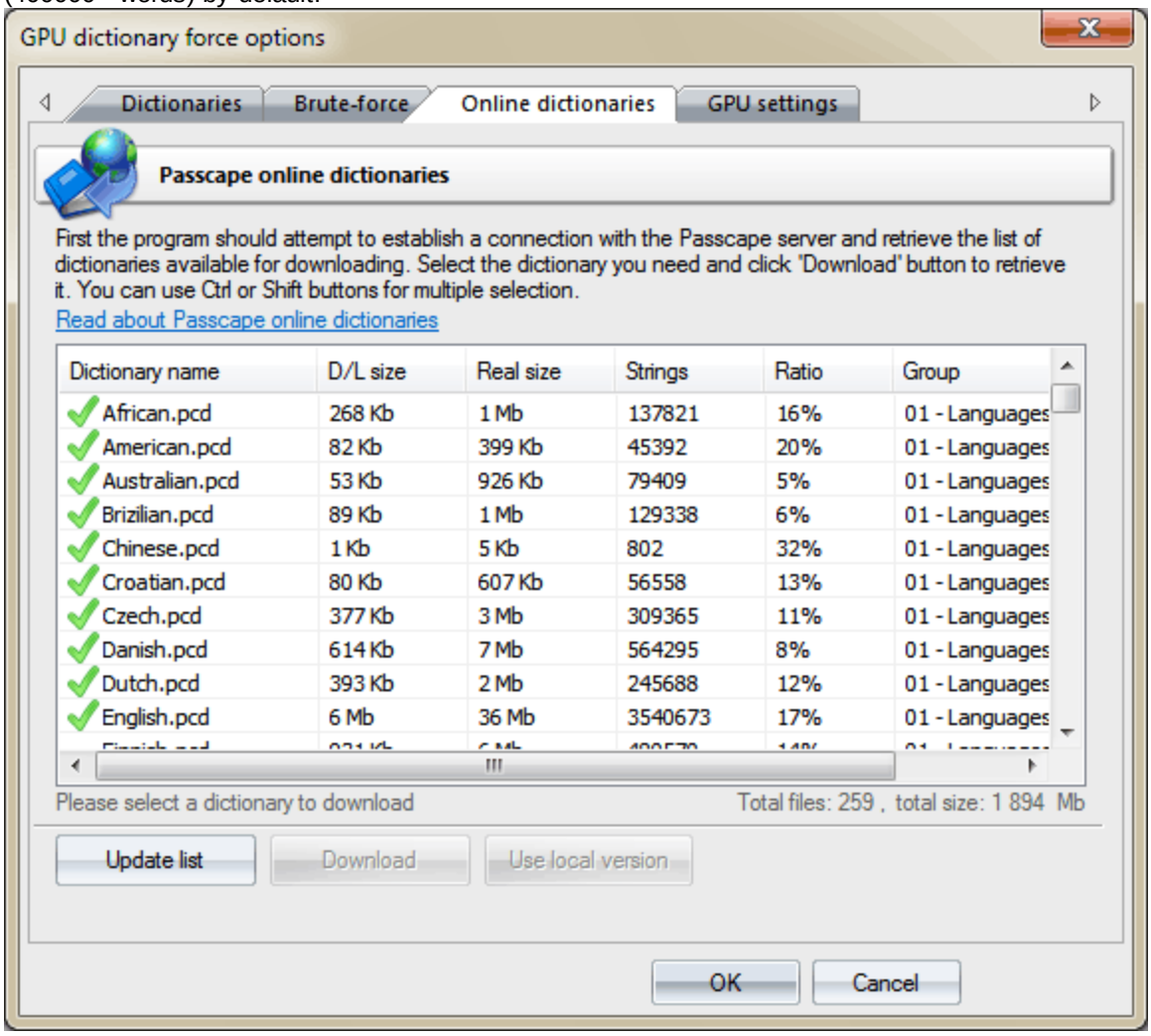
The number of passwords to be searched for a single word can be calculated using the following formula:

passwords = R * L * K
where
R - character range, calculated using the formula: R = charset_length ^ max_length - charset_length ^ (min_length-1) +1
L - positions in word. Calculated as follows: if the insertion is made in the middle of the word, L = password_length - 1; then add plus one if the insertion is made to the beginning and end of the word.
K - number of options specified in the group '**Input word utilization**'.

For example, if the source word we have is **window**, and the options are specified as shown on the image above, i.e. character range **a..z,A..Z,0..9,symbol14,space**, insertion to all positions, conversion to lowercase and capitalizing (first letter in uppercase). Let's calculate how many password we are going to check for this word:
charset_length = 26+26+10+14+1 = 77
R = 77^4 - 77^0 + 1 = 35153041
L = (6-1) + 1 + 1 = 7
K = 2
passwords = 35153041 * 7 * 2 = **492 142 574**

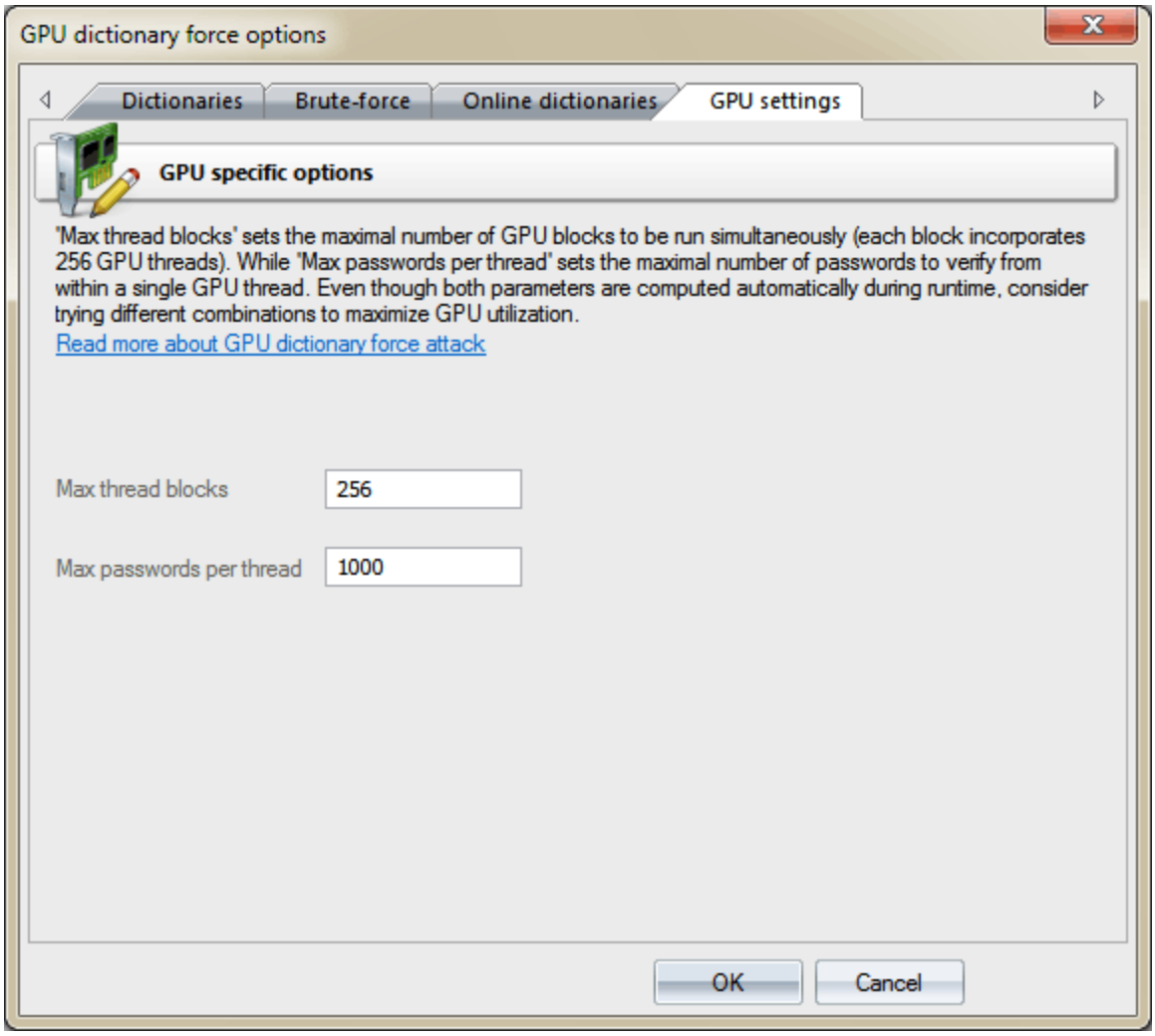
Online dictionaries

On the third tab, you can download source wordlists for the attack. The program uses internal wordlist (400000+ words) by default.



GPU settings

Before you can use a GPU in the attack, you must first select it in the respective item of the main menu.



GPU configuration is pretty simple and consists of only two settings:

1. The number of parallel graphics card's blocks, where passwords would be searched. Typically, each block consists of 256 threads. Thus, if you set the number of blocks to 256, the GPU will run 256*256=65536 threads. The total number of checked passwords for one call to GPU kernel will be 256*ThreadBlocks*PasswordsPerThread. In our case 256*256*1000 = 65 536 000 passwords. Setting the **ThreadBlocks** smaller than 256 on modern graphics cards, in the majority of cases, leads to performance degradation.
2. The number of passwords to be search from a single thread. The greater the value, the lower the overhead associated with launching threads, and the higher the search speed. However, setting too great a value may hang the computer or cause significant fluctuations in the current search speed, displayed on the attack status tab. This is caused by the fact that task completion time on the GPU exceeds the time required for refreshing the current state of the attack.

Depending on the options you have specified, a proper choice of GPU settings can dramatically, often by several times, increase the password search speed. We recommend playing around with GPU settings to achieve the maximum utilization of the GPU in this attack.

2.8.2.19 GPU: Hybrid dictionary attack

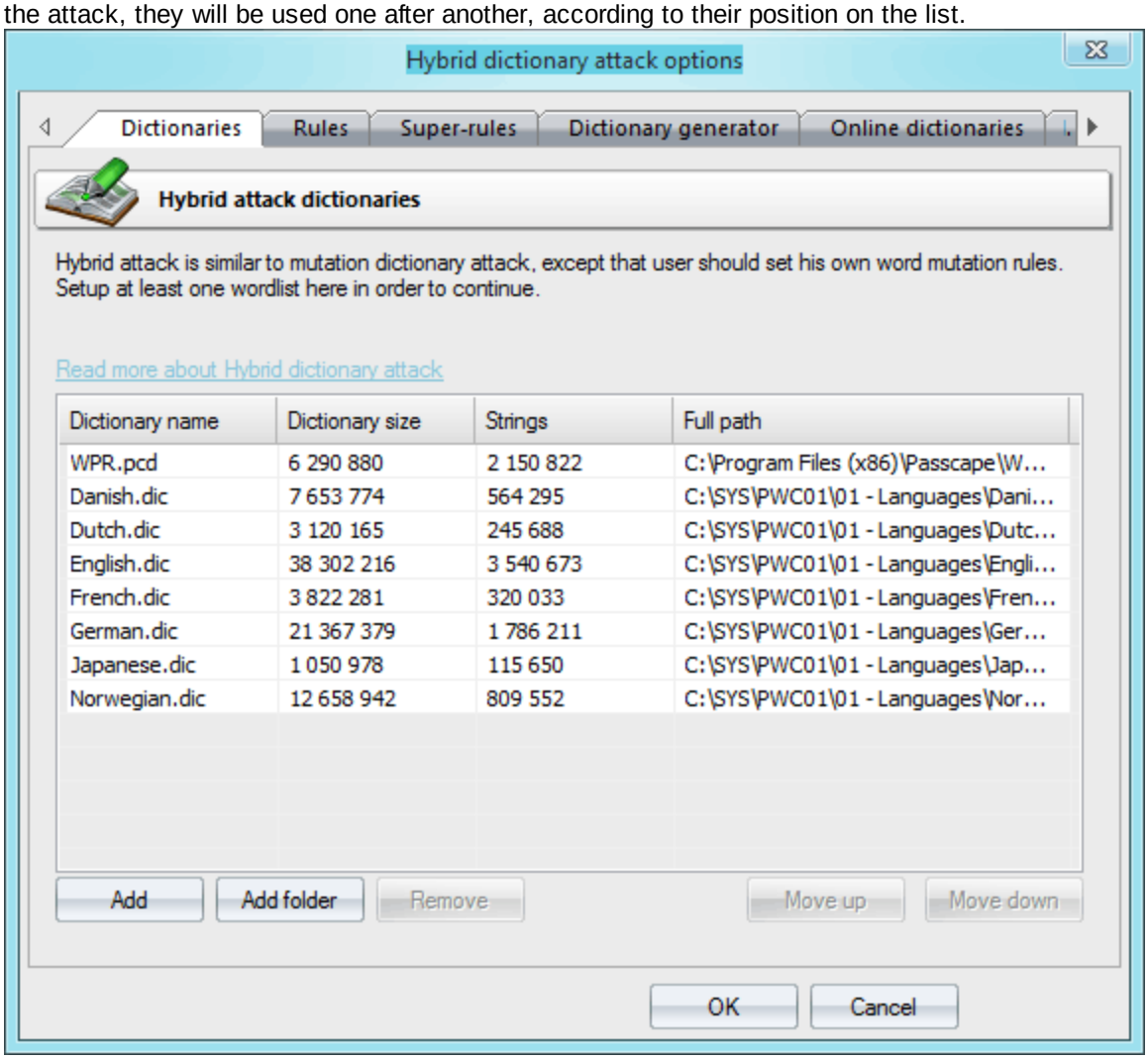
GPU Hybrid dictionary attack is pretty much the same as the Hybrid Dictionary attack, except that it utilizes your GPU power instead of CPU. That makes it extremely fast. Approximately 10 times faster than a simple Hybrid attack. The value greatly depends on options and hardware used though. The

hybrid attack allows user to set his own word modification rules and attempt to validate the modified output words.

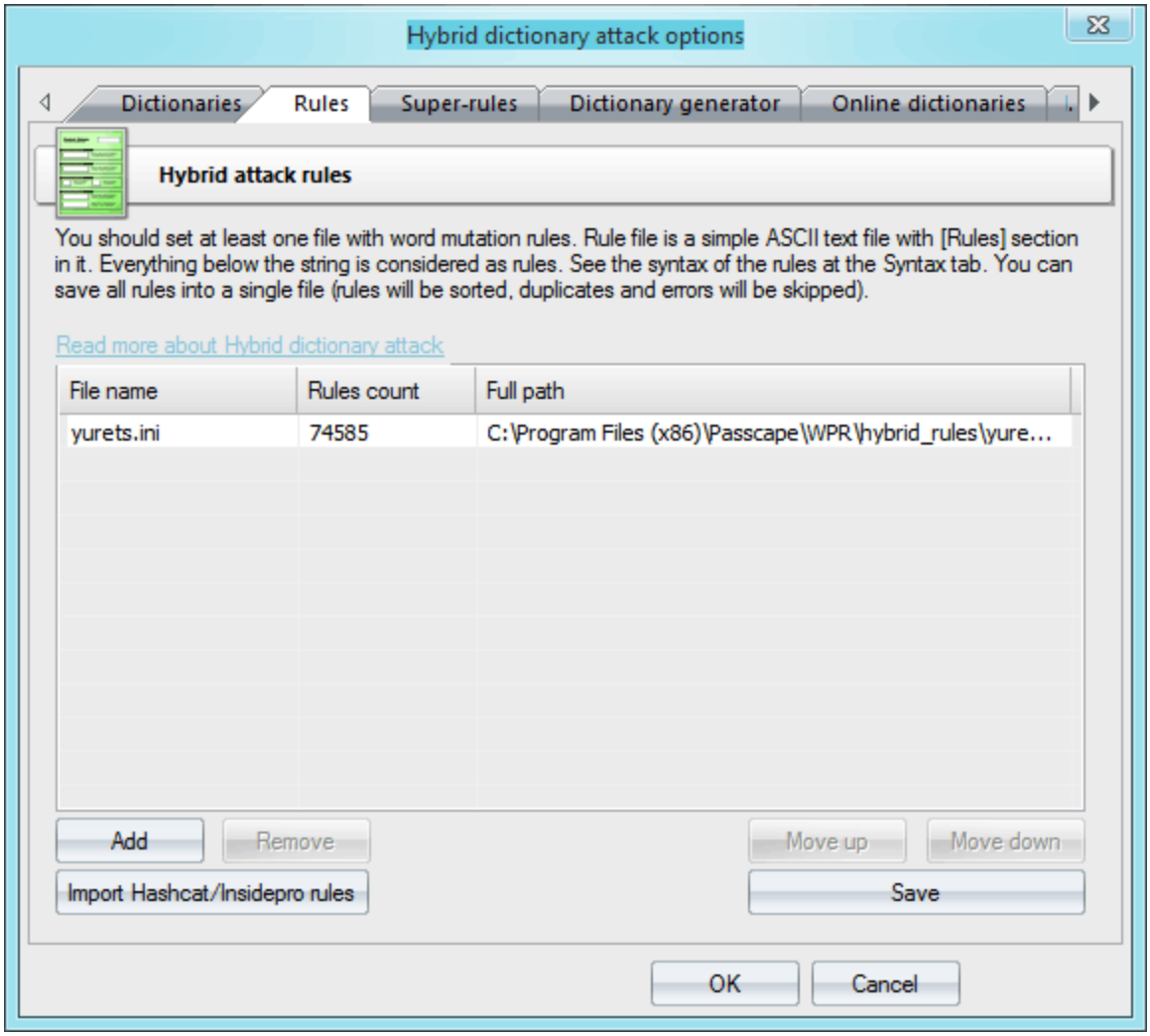
Actions, performed on source words from the dictionary, are called rules. Multiple rules can be applied to each source word.

GPU hybrid dictionary attack settings are grouped in eight tabs:
1. **Dictionaries** - for setting up source dictionaries.
2. **Rules** - files with set of rules.
3. **Super-rules** - ones to be applied over the top of regular rules
4. **Dictionary generator**, where you can create files of words obtained from the hybrid attack.
5. **Online dictionaries** - for downloading new dictionaries to the application.
6. **Attack syntax** - complete description of all rules with examples.
7. **Rule tester**, where you can test your rules.
8. **GPU settings** is used to tune your GPU parameters.

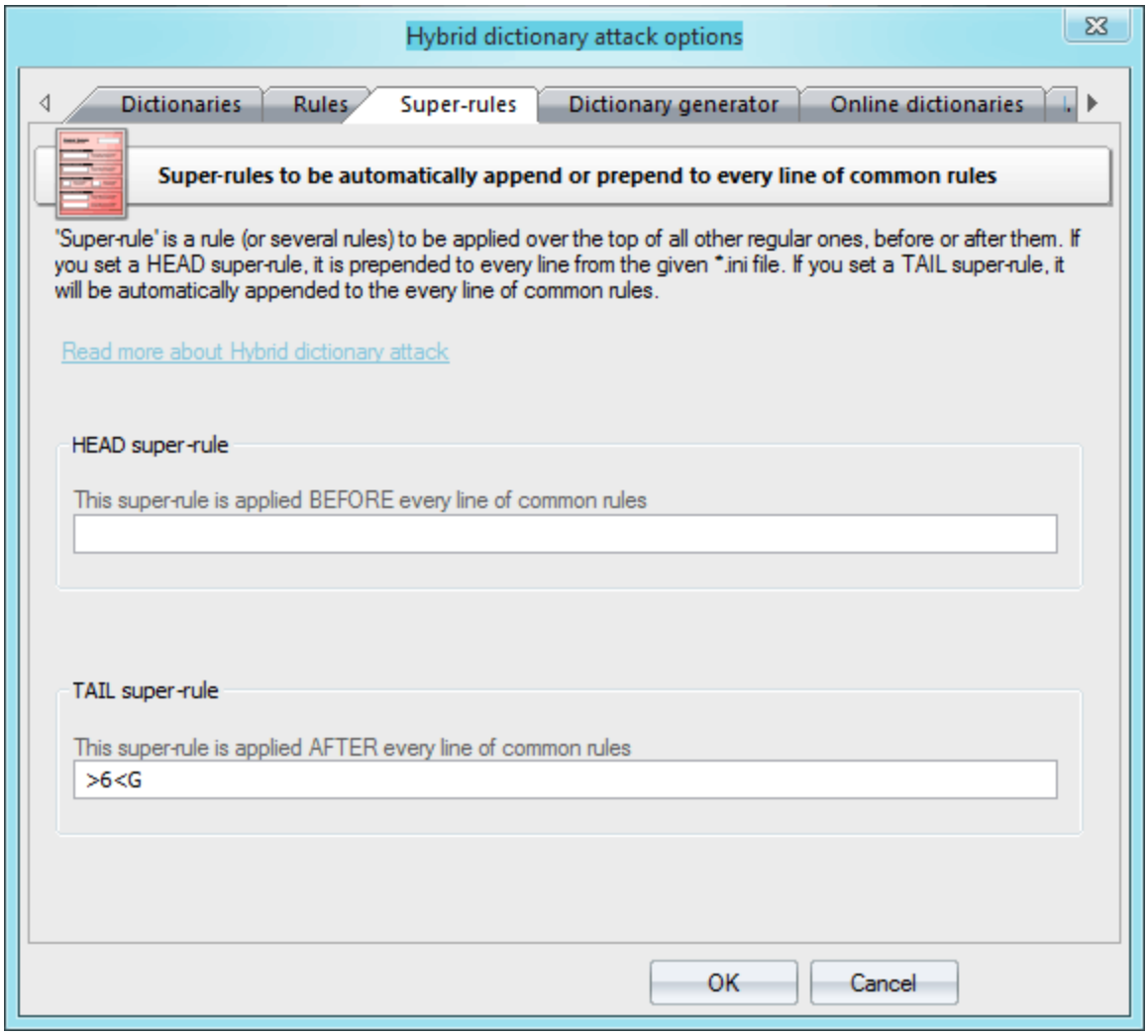
Wordlists to be used in the attack are set on the first tab. Traditionally, the application supports wordlists in ASCII, UTF8, UNICODE, PCD, RAR and ZIP format. The position of the files on the list can be altered. For example, you may want to move smaller dictionaries up the list or the other way. During the attack, they will be used one after another, according to their position on the list.



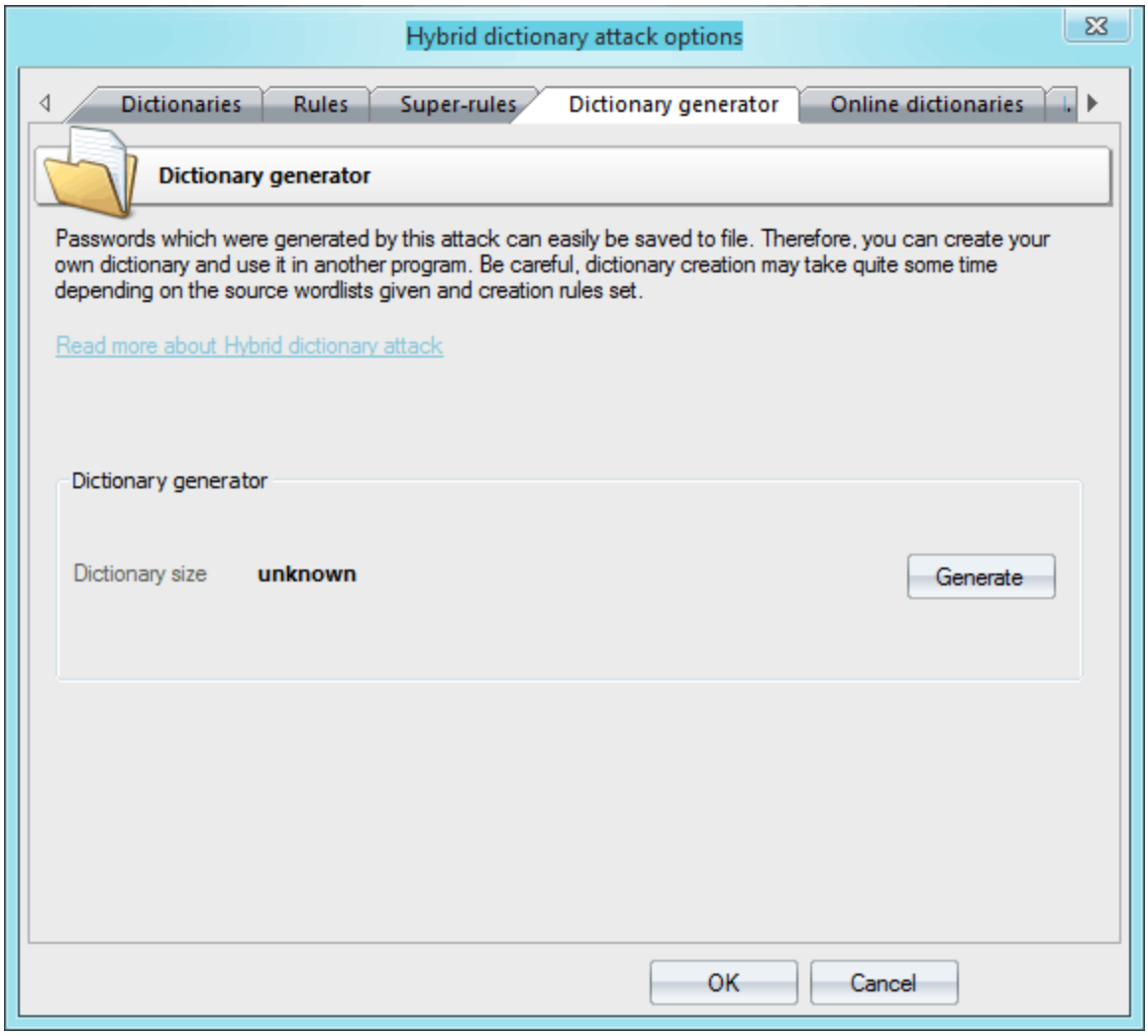
On the '**Rules**' tab, define at least one file with password mutation rules. The format of the rules file is quite trivial; it is a plain-text ASCII file with the '**[Rules]**' string. Anything above this string is considered as comments and ignored by the program. Whatever goes below this string is considered as rules. Each string can contain several rules, applicable to a source word. If a string contains multiple rules per word, those rules are parsed left to right. For example, if you apply the rule '@pc$a$b$c' to the source word 'password', at the output you will get 'Asswordabc'. The maximum length of an output word may not exceed **256** characters.



'**Super-rule**' is a rule (or several rules) to be applied over the top of all other regular ones, before or after them. For example, you can set 'a8' tail super-rule to create all possible case combinations after a common mutation has been done. So the '/asa4' rule from l33t.ini file will become '/asa4a8', '/csc(' will become '/csc(a8', etc. Yet another one example: setting the '>6<G' head rule allows you to skip all words of less than 6 or greater than 16 characters, before starting a common mutation. This is a helpful feature once you decide to add the same rule to all text lines of the selected *.ini files. There's no need to modify them all. Be careful though, the 'aN' super-rule may increase the total number of generated passwords drastically.

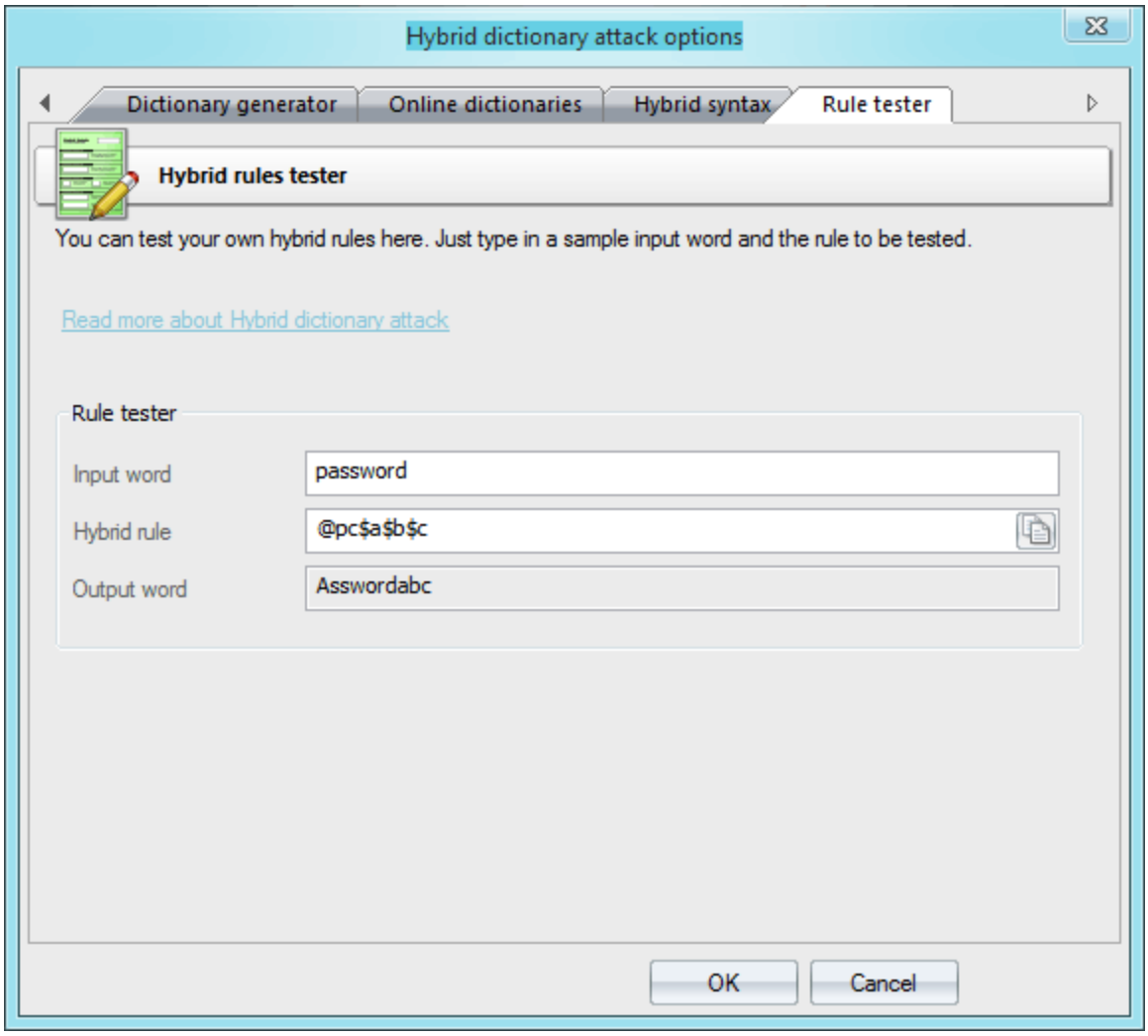


The '**Dictionary generator**' tab is designed for generating dictionaries obtained from an attack. Those custom-made dictionaries could be used, for example, in other applications. To generate a dictionary, specify a source dictionary and a set of mutation rules for it. The size of a target file may exceed 2 GB assuming that you save it to NTFS disk. Be careful, the dictionary generation process may take considerable time and disk space!



You can download additional wordlists for the attack using '[Online dictionaries]' tab.

If you want to create your own set of rules, you can use the next two tabs as sources of help. While the '**Syntax**' tab gives mere descriptions of available rules, on the '**Rule tester**' tab you can actually check them by specifying a source word and a rule. Forward your rule sets to us; if we find them interesting/ useful, we will include them in the program.



Rules description for the hybrid dictionary attack

Several rules at a line are allowed to be set.
Rules (if any) are processed from the left to the right.
Maximal line length is limited to **256** characters.
Maximal output word length is limited to **256** characters.
White space is ignored as long as it is not used as a parameter.
A line started with # character considered as a comment
All text before the '**[Rules]**' line is considered as comment.
N and M always start at 0. For values greater than 9 use A..Z (A=10, B=11, etc.)
The following rules should be at the last position of a line: aN, ?iN[C], ?i[C], ?oN[C], ?o[C], ?iZ[C], ?oZ[C]
Don't change the names of the standard rule files. Some ones are used by the program.
?iN[C], ?i[C], ?oN[C], ?o[C] ?iZ[C], ?oZ[C] rules use the following predefined charsets (you can use custom character sets though):

digits	- 0123456789
loweralpha	- abcdefghijklmnopqrstuwxyz
upperalpha	- ABCDEFGHIJKLMNOPQRSTUVWXYZ
alpha	- abcdefghijklmnopqrstuwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
special	- !@#$%^&*()-_+=~`[]{}|\:;"'<>,.?/ "

loweralphanumeric - abcdefghijklmnopqrstuwxyz0123456789
upperalphanumeric - ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
alphanumeric - abcdefghijklmnopqrstuwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
printable - abcdefghijklmnopqrstuwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()-_+=~`[]{}|\:;"'<>,.? /

Rules

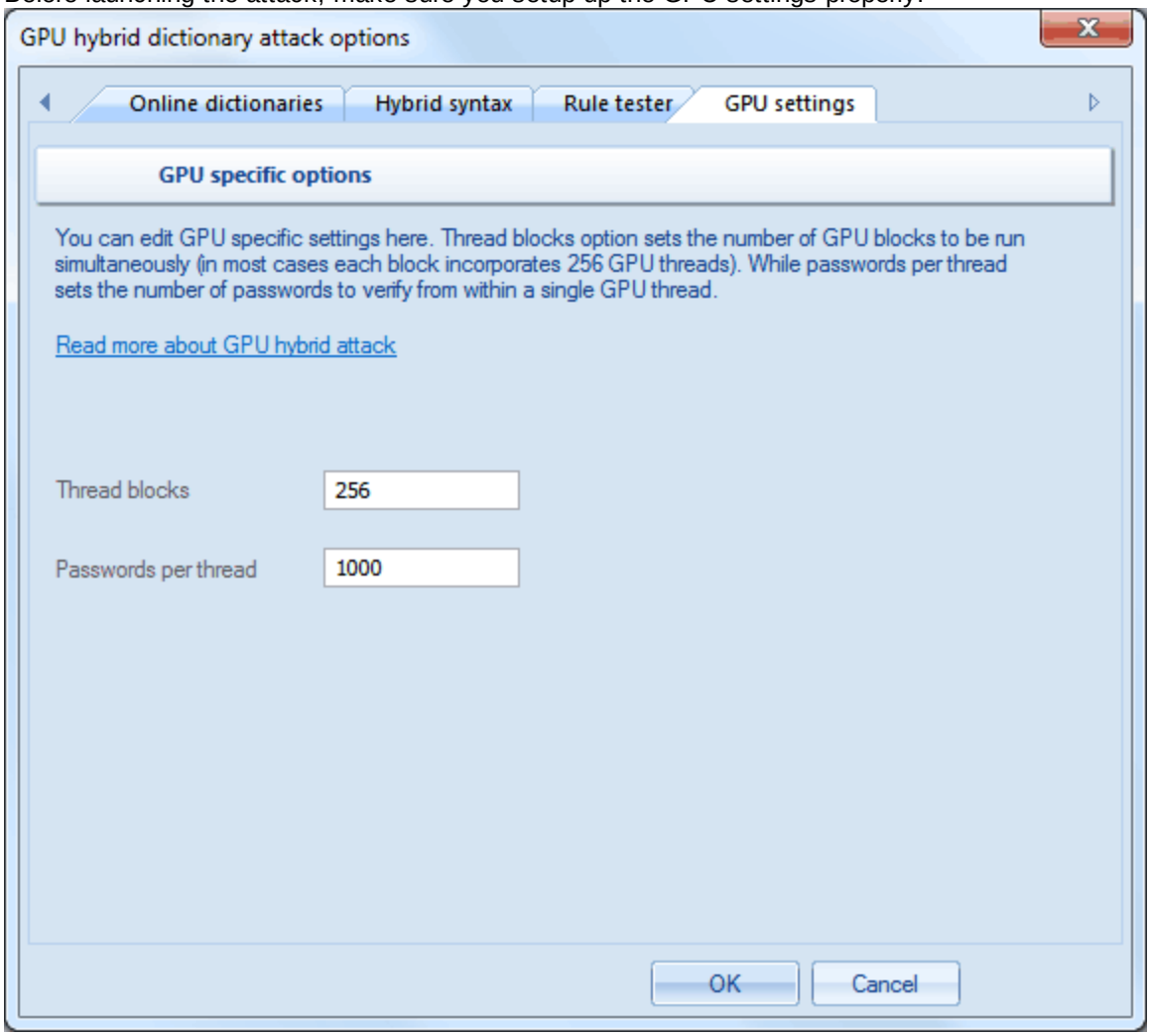
Rule	Example	Input	Output	Description
:	:	password	password	Do nothing to the input word
{	{	password	asswordp	Rotate the word left
}	}	password	dpasswor	Rotate the word right
[	[	password	assword	Delete the first character
]	]	password	passwor	Delete the last character
c	c	password	Password	Capitalize
C	C	password	pASSWORD	Anti-capitalize (lowercase the first character, uppercase the rest)
d	d	password	passwordpassword	Duplicate word
f	f	password	passworddrowssap	Reflect word
k	k	password пароль	зфыыцщквgfhjkm	Convert word using alternative (first after default) keyboard layout. The rule works in both directions. For example, if there's Russian keyboard layout installed previously in the system, the rule should convert word 'password' to Russian 'зфыыцщкв', and Russian word 'пароль' to 'gfhjkm'. This is very helpful when looking for non-English passwords. If only one language is installed in the system, the rule does nothing.
K	K	password	passwodr	Swap last two characters
l	l	password	password	Convert all characters to lowercase
q	q	password	ppaasssswoorrdd	Duplicate all symbols
r	r	password	drowssap	Reverse word
t	t	PassWord	pASSwORD	Toggle case of all characters
u	u	password	PASSWORD	Convert all characters to uppercase
U	U	my own password	My Own Password	Capitalize all words delimited with space (upper-case the first character and every character after a space)
V	V	password	PaSSWoRD	Vowels elite
v	v	password	pASSWoRD	Vowels noelite

Rule	Example	Input	Output	Description
'N	'4	password	pass	Truncate the word to N character(s) length
+N	+1	password	pbssword	Increment character at position N by 1 ASCII value
-N	-0	password	oassword	Decrement character at position N by 1
.N	.4	password	passoord	Replace character at position N with character at position N+1
,N	,1	password	ppssword	Replace character at position N with character at position N-1. Where N > 0.
<N				Reject (skip) the word if it is greater than N characters long
>N				Reject (skip) the word if it is less than N characters long
aN				Check all possible symbol cases for the word. N is a maximal length of the word to apply this rule for.
DN	D2D2	password	paword	Delete the character at position N
pN	p3	key	keykeykey	Copy word N times
TN	T1T5	password	pAsswOrd	Toggle case of the character at position N
yN	y3	password	paspaswor d	Duplicate first N characters
YN	Y3	password	paswordord d	Duplicate last N characters
zN	z3	password	ppppasswo rd	Duplicate the first character of the word N times
ZN	Z3	password	passwordd ddd	Duplicate the last character of the word N times
$X	$0$0$ 7	password	password0 07	Add character X to the end of the word
^X	^3^2^ 1	password	123passwo rd	Insert character X at the beginning of the word
@X	@s	password	paword	Remove all characters X from the word
!X				Reject (skip) the word if it contains at least one character X
/X				Reject (skip) the word if it does not contain character X
(X				Reject (skip) the word if the first character is not X
)X				Reject (skip) the word if the last character is not X
eX	e@	mike@y ahoo.co m	mike	Extract a substring starting at position 0 and ending up before first occurrence of X character (do nothing if X is not found)
EX	E@e.	mike@y ahoo.co m	yahoo	Extract a substring starting right after first found X character and till the end of the string (do nothing if X is not found)
%MX				Reject (skip) the word if it does not contain at least M instances of the character X
***XY**	*15	password	possward	Swap characters at positions X and Y
=N				Reject (skip) the word if the character at position N is not equal to the X

Rule	Example	Input	Output	Description
X				
iNX	i4ai5bi6c	password	passabcword	Insert the character X in position N
oNX	o4*o5*	password	pass**rd	Overwrite a character in position N with the character X
sXY	ss$so0	password	pa$$w0rd	Replace all characters X with Y
xNM	x4Z	password	word	Extract a substring of up to M characters length, starting from position N.
INX-Y	rl0/-/r	google.com	google.com/	Insert the character X at position N if previous character at position N is not Y.
INX+Y	rl0.+.r	password.	password..	Insert the character X at position N if previous character at position N is Y.
ONX-Y	O0-+p	password	-assword	If the character at position N is not Y, overwrite it with X character.
ONX+Y	O0P+p	password	Password	If the character at position N is Y, overwrite it with X character.
RNM+Y	R01+a	password	assword	Remove character at position N if character at position M is Y
RNM-Y	R40-b	password	passord	Remove character at position N if character at position M is not Y
?iN[C]	?i0[digits]	password	0password, 1password ... 9password	Insert a character from a charset [C] into position N of the word. Where C should be either a predefined charset name or a custom character set itself.
?iZ[C]	?iZ[digits]	password	password0, password1 ... password9	Insert a character from a charset [C] into last position of the word. Where C should be either a predefined charset name or a custom character set itself.
?i[C]	?i[special]	password	~password, !password ... password_, password+	Insert a character from a charset [C] into every position of the word. Where C should be either a predefined charset name or a custom character set itself.
?oN[C]	?o1[upperalpha]	password	pAssword, pBssword ... pZssword	Overwrite a character at position N with a character taken from a charset [C]. Where C should be either a predefined charset name or a custom character set itself.
?oZ[C]	?oZ[upperalpha]	password	passworA, passworB ... passworZ	Overwrite a character at last position with a character taken from a charset [C]. Where C should be either a predefined charset name or a custom character set itself.
?o[C]	?o[-=.]	password	-assword, =assword ... passwor.	Overwrite a character at every position of the word with a character taken from a charset [C]. Where C should be either a predefined charset name or a custom character set itself.

GPU settings

Before launching the attack, make sure you setup up the GPU settings properly.



GPU configuration is pretty simple and consists of two parameters:

1. The number of GPU blocks to be run at a single call to GPU. Each block consists of 256 threads. Thus, if you set the number of blocks to 256, the GPU will run 256*256=65536 threads. The total number of checked passwords for one call to GPU kernel will be 256*ThreadBlocks*PasswordsPerThread. In our case 256*256*1000 = 65 536 000 passwords per one call to GPU.
2. The number of passwords to be search in a single GPU thread. The greater the value, the lower the overhead associated with launching threads, and the higher the search speed. However, setting too great a value may hang the computer, make your GPU unresponsive or cause significant fluctuations in the current search speed, displayed on the attack status tab. This is caused by the fact that task completion time on the GPU exceeds the time required for refreshing the current state of the attack.

Be careful setting 'heavy' rules like aN, ?iN, ?oN, etc. These rules may increase the number of generated passwords by hundred times and hang up your system or make your GPU device unresponsive.

2.9 View menu

The View menu enables/disables the auxiliary elements of the interface, change the interface language, minimize the application to the tray or run it in the invisible mode.

2.10 Themes menu

You can select here one of the themes you've liked or create your own theme.

2.11 Help menu

In this section of the menu, you can access the help articles on using the software, visit the program's home on the Web, check availability of updates, submit a bug report, register your copy of Windows Password Recovery, etc.

2.12 Hardware Monitor

On this tab, you can view current CPU load, RAM utilization, GPU temperature and load. By default, the refresh interval is set to 2 seconds. Be careful: gathering these statistics also takes CPU time; therefore, when running "heavy" attacks, such as brute-force, it is recommended to keep the system monitor disabled.

Working with the program

3 Working with the program

3.1 Attacking Windows hashes

Currently the program can decrypt Windows hashes in several ways:

Preliminary attack (developed by Passcape Software) is based upon a social engineering method and consists of several sub attacks. Preliminary attack is very fast and often it is used for guessing simple and short passwords when there's no need to launch a fully scalable attack.

Artificial Intelligence attack - is a brand-new type of attack developed in our company. It is based upon a social engineering method and allows, without resort to time-consuming and costly computations, to almost instantly and painless recover certain passwords.

Dictionary attack. It is the most efficient recovery method, when the program tries each word from the dictionary (or dictionaries if there are several dictionaries) you specify until it finds the original password or until the wordlist is out of words. This method is very efficient since many people use regular words or phrases for password. Besides this type of recovery is performed quite fast compared to brute-force attack, for instance. Additional dictionaries and word-lists can be downloaded from our site or can be ordered on CDs.

Brute-force attack tries all possible combinations from the specified range of characters. For example, for a three-character range of lower-case Latin characters, it will check all possible combinations, starting with 'aaa', 'aab', 'aac', and all the way through 'zzz'. This is the slowest attack, so it is really great for short passwords.

Mask attack is a variation of the brute-force attack, except that some characters for finding the password remain unchanged, and only a portion of the password may change. The special syntax is used for setting a mask or rule for finding a password.

Base-word attack (developed by Passcape). At the first glance, this type of attack reminds the one we just described. It is just as efficient if a portion of the password to be recovered is known to us. However, unlike in the previous attack, here you do not have to set a mask - just provide a basic word. The program will take care of the rest. The phrase attack is based upon the experience of the social engineering to generate a great number of possible combinations of the given password.

Combined dictionary attack (developed by Passcape) uses to find compound passwords. For example, 'nothingtodo' or 'I give up'. It is very similar to the dictionary attack, except that instead of using a single word for password verification it uses a combination of words created by combining words from several dictionaries. You can create your own password generation rules.

Phrase attack (developed by Passcape) is very efficient against complex passwords. The idea of it is to guess the right password by searching through frequently used phrases and combinations. You can download pass-phrase wordlists and dictionaries from our site only.

Rainbow attack (developed by Philippe Oechslin). It is a time-memory tradeoff used in recovering the plaintext password from hashes. This attack is quite fast and effective tool for auditing Windows hashes.

Fingerprint Attack. Developed by Passcape, original idea by Atom. The attack parses input wordlist to generate so-called "fingerprints" used to recover the password. The attack is quite effective in finding difficult passwords for big list of hashes or for password history hashes.

Hybrid dictionary attack is like a simple dictionary attack, but allows user to customize word mutation

and set his own password mutation rules. The rule definition syntax is compatible with some other password recovery software.

Online recovery (developed by Passcape Software) searches passwords in Internet databases. It deals fairly well with simple and frequently-used passwords. Its drawback is pretty low operating speed and poor suitability for handling large hash lists.

Passcape rainbow table attack (developed by Passcape Software). It's the next generation of regular pre-calculated tables. Passcape table attack is most suitable for the recovery of complex passwords of literally unlimited length.

Batch attack (developed in Passcape Software) creates a list/batch of attacks to be run one-by-one, so that you could launch all those attacks with a single mouse-click instead of configuring each of them individually.

GPU brute-force attack is fully identical to simple brute-force except that to guess passwords, it uses video card instead of CPU. The GPU device to be run the attack on, should be set in *General Options*.

GPU fingerprint attack works exactly the same way the simple fingerprint attack does but uses GPU power.

GPU mask attack. This password recovery method is fully identical to the regular mask attack except that the password guessing is processed by a graphical card of your PC, thus the recovery speed is much higher.

GPU dictionary-force. Often, when creating passwords, users add certain symbols in the beginning, end or even middle of the word. To recover passwords of this specific kind, we have come up with a GPU-based dictionary attack.

GPU Hybrid dictionary attack. The same as a simple Hybrid dictionary attack but much faster because uses GPU.

3.2 Attack comparison table

Which attack is the best? How do you choose the attack? The answers to these questions should be found in the attack comparison table.

Attack	Description	Time required	Guaranteed	Pros	Contras	Limitations
Preliminary	A set of light and speedy mini-attacks for finding simple, short or common combinations	A couple of minutes	No	Great quick-find tool for quick recovery of common, simple, short passwords, keyboard combinations, repetitive sequences, etc. Good for finding weak passwords quickly; doesn't require additional settings	Practically useless for serious analysis, when recovering the majority of complex passwords	Finds mainly simple passwords
Artificial Intelligence	The most	Min: 2-3	No	The best tool for finding	During the most	Efficient only

	advanced way of recovering passwords, based on the methods of social engineering.	minutes, Max: over an hour		complex passwords, which other methods cannot cope with. Works great for passwords, words and combinations that the user stored in the system any time in the past.	efficient analysis, when all the options are set to the maximum performance, the attack takes considerable time. Finds not all passwords.	when run on the original system (where the passwords were taken)
Brute-force	Searches all possible combinations within a specified character set	Depends on options	Yes	The only attack (along with the mask attack) that is guaranteed to recover a completely unknown password. Good for any short and medium passwords	Searching long passwords takes considerable time. Hard to guess the right range of characters to be searched.	May take centuries to search long passwords. Does not find passwords when uses wrong character set or password length exceeds the one specified
Dictionary	Finds password by searching words from predefined dictionaries (word-lists)	Almost instantly	No	Good and speedy tool for recovering common passwords	Requires having good dictionaries, does not take into account peculiarities of the language and letter case	Finds only common passwords
Dictionary with smart mutation	Same as dictionary attack, except here each word from the dictionary undergoes all kinds of mutations. For instance, appending numbers, changing letter case, deforming (displacing) letters, etc.	Up to 1000000 times slower than a simple dictionary attack	No	Good for all sorts of variations of common passwords	The maximum (most effective) mutation takes considerable time	Fails to find strong (non-dictionary) passwords, mutation takes considerable time
Mask	Finds passwords by specified mask (password	Depends on options	Yes	Guaranteed to recover the remaining portion of a password. Good	Requires having the exact known portion of the password and	Password will not be found if a wrong

	generation rule)			option when some portion of the original password is known.	its length and specifying the right character set to be searched	character set, incorrect password length or incorrect known portion of the source password is specified
Combined dictionary	Checks complex passwords (composed of two or more words) by gluing words from several dictionaries	Depends on options	No	The only attack that finds long and complex passwords	Limited set of field-specific dictionaries, does not take into account peculiarities of non-English passwords (endings, suffixes, etc.) With a large source dictionary, the attack may take considerable time	Requires to know in advance that the password being searched for consists of two or more words; relatively slow
Combined dictionary with smart mutation	Same as combined attack, plus mutations	Depends on options	No	Same as the previous attack	Same as the previous attack. Requires setting additional mutation rules for the passwords to be generated	Same as the previous attack; mutations require considerable time
Base-word	Takes advantage of a known base word used for making up the password	A couple of seconds if the base-word length is not exceeds 16 characters	No	Good for the cases when you had known the original password but have forgotten its variations, e.g., letter case or trailing numbers	Mutation for long passwords (over 16 characters) may take some time	Does not always work
Phrase	Same as dictionary attack, except that instead of a word this one checks a phrase, popular expression, excerpts from songs, books,	From several minutes up to several hours	No	The only attack against password phrases.	Only a small percentage of users use pass-phrases as passwords. Phrase mutation is imperfect; the mutation and analysis take considerable time. Insufficient number of	Does not take into account peculiarities of the language; limited choice of mutations. Difficulty in the creation of specialized

	etc.				relevant dictionaries; in particular, with non-English phrases and expressions.	dictionaries.
Rainbow tables	Uses precalculated tables	Usually several minutes (or even seconds) for each password	Up to 100% if the password fits into the character set and password length of the table(s)	Currently one of the best attacks for recovering the majority of passwords by the time/efficiency ratio	Requires tables. Precalculation tables may take much room on a hard drive. It is impossible to recover long passwords using this attack.	Cannot recover all passwords simultaneously; generating a new table takes longer than running a brute-force attack. Limited recovery capabilities for long and non-English passwords
Fingerprint	Based on fingerprints that were generated out of the given wordlist	From several hours up to several days (depends on the initial dictionary)	No	Finds complex passwords that were impossible to recover in other attacks	Big input dictionary may generate too much fingerprints. The success depends on the input dictionary.	The attack take too much time to complete when setting a big input wordlist.
Hybrid dictionary	It is much similar to simple dictionary attack, except that the password mutation rules are fully customizable and should be set by user.	Depend on the source wordlist and rules counter. Usually up to several minutes for a small wordlist.	No	Good for all sorts of variations of common passwords	Cannot recover complex passwords.	Fails to find strong (non-dictionary) passwords
Online recovery	Searches passwords via Internet	Depends on options	No	Pretty nice alternative tool for finding out simple and frequently-	Very slow, processes hashes subsequently, feeds	Fails to find most strong passwords.

		set and internet connection speed. Usually less than 1 minute for a single hash.		used passwords.	a lot of Internet traffic.	Works only when internet is available.
Passcape rainbow tables	Uses specially formed precalculated tables to guess strong and complicated passwords	Several minutes (or even seconds) for each password, depending on table parameters.	No	Actually it is very good and advanced attack for recovering strong and complicated passwords which cannot be cracked in other attacks.	A good table precalculation may take much disk space and time. Password recovery success rate greatly depends on input wordlist.	Cannot recover all passwords simultaneously; generating a new table takes longer than running a brute-force attack. Not all initial wordlists suit well for creating Passcape tables.

3.3 Recovering passwords from hashes

Use this simple instruction for the recovery of any passwords in Passcape programs. This instruction is offered in the format of recommendation and is meant primarily for the recovery of passwords encrypted with OWF; e.g., from Windows hashes.

When recovering certain types of passwords the major question is: How to organize the recovery process - which attack should I start with to raise the probability of its successful completion?

For choosing the type and the sequence of the attacks, we advise to follow this algorithm, which is applicable in the majority of cases to all types of passwords to be recovered:

First, enable the preliminary attack option, if it is available. It will help to recover simple and frequently used combinations.

Second, select one or several passwords you need to decrypt first of all and run Online recovery to find out simple and frequently-used passwords.

Third, if you are aware of any specifics of the password you are looking for, it's better to try mask attack or base-word attack first. Specifically, if you know a part of the password - using mask attack would be more effective. If you know the basic component of the password or, for example, know the password but don't remember the sequence of caps and lowercase characters in it, base-word attack would do the job better.

Fourth, if you there's no information on the password you are looking for, which occurs most frequently, be guided by the following sequence of steps:

1. Launch Artificial Intelligence attack with mutation and indexing options set to light.
2. If the password was not found, try once again with mutation option set to normal level and indexing set to deep.
3. Run a rainbow table attack if there are any tables
4. Run a Passcape rainbow table attack.
5. Run dictionary attack with the mutation option disabled.
6. Launch dictionary attack with the mutation option enabled; the depth of mutation depends on the amount of available time and the attack speed. When searching for passwords typed in the national keyboard layout, the depth of mutation should be set to strong.
7. Select and download online dictionaries and repeat steps 5 - 6.
8. Run Hybrid dictionary attack.
9. Repeat Hybrid attack using alternative wordlists.
10. Launch pass-phrase attack with the mutation option disabled.
11. Launch pass-phrase attack with the mutation option enabled and set to the maximum productivity. This will allow finding even passwords typed in the national keyboard layout.
12. Select and download online pass-phrase dictionaries and repeat steps 10 - 11.
13. Launch combined dictionary attack with defined phrase generation rules.
14. Select and download online dictionaries for combined attack and repeat step 13.
15. Run fingerprint attack with default dictionary.
16. Select and download new online dictionary for the fingerprint attack, adjust options, set the new dictionary and repeat step 15.
17. Select a charset and password length for brute-force attack, launch the attack.
18. If necessary, select a new or complete the old character set and repeat the brute-force attack; i.e. step 17.

Based on the given recommendations, it is easy to create your own rules for batch attack.

3.4 Windows passwords FAQ

Q. What is password protection?

A. Perhaps no one would argue that Windows NT-based operating systems today are the most popular all over the world. That makes them very vulnerable targets for various kinds of hackers, intruders and dishonest users. The spread of the global network only exacerbates the situation. To ensure the personalization of stored user or system data and to protect it from unauthorized access by third parties, it was proposed to use the password protection technology. Currently, the primary protection in Windows operating systems is password protection. Access to private data in this case is possible only when user knows the original password, which is normally a word or phrase. Here is what it looks like in the real life: the program or system, on an attempt to access private data, prompts user for the text passwords. That password is checked against the original password, and, if the values match, the system allows access to the private data; otherwise, it denies access. The primary disadvantage of password protection is that the program or system must store the original password somewhere, in order to have something to compare the entered value with.

Q. How do operating systems store passwords?

A. But everything is not so bad; Windows NT was developed in a way that it wouldn't store the original text value of the password. "How is that?" You may ask. - Very easy. There are special cryptographic password wrapper algorithms that work one way only. That's why sometimes they are referred to OWF - one-way functions. Roughly, you can get the hash from a password, but there's no way to get the password from a hash. How does it work in Windows? When creating an account, user enters

the original password, which, however, is not stored as plain text; instead, it is hashed with an OWF function. The password hash returned by the function will be stored in the system. Further on, when attempting to log on, the system will prompt user for the password; it hashes the password again and then compares the generated hash with the original one that is stored in the system. If the two values match, the passwords, naturally, match too. Thus, the original text password is not stored in the system. Moreover, there are new algorithms out there that do not even store hash, and the number of such algorithms keeps growing. An algorithm of such kind, for example, is used for encrypting passwords in Internet Explorer 7-8. You can learn more about it in our article.

Q. How do passwords become encrypted?

A. For hashing user passwords, Windows NT uses two algorithms: LM, which we have inherited from Lan Manager networks, which is based on a simple DES conversion, and NT, based upon the MD4 hashing function. LM, as the weaker and vulnerable one, is not supported by default by the latest Windows Vista and Windows 7; however, you can still enable it. Moreover, there is a tendency to completely eliminate or replace it. It is important to know that when the LM hashing option is on (it is enabled by default in Windows XP), all user passwords are considered quite vulnerable. Cracking the majority of such passwords normally takes just a few minutes. The NT hash is free from the disadvantages, common to the LM hash. Consequently, it is much harder to pick the right password to a known NT hash than to an LM hash. But the current trend of increasing the computing power of modern computers, especially when using GPU, possibly, will make this standard too vulnerable to potential attackers.

Q. Where are password hashes stored?

A. So, we have found out that user passwords in Windows systems are converted to special values - hashes. LM and NT hashes both have a fixed size - 16 bytes - and can be stored in two repositories: SAM - for the regular accounts and Active Directory - for domain accounts.

SAM: The regular accounts that contain user name, password and other auxiliary information are stored in the Windows NT registry; precisely, in the SAM (Security Account Manager) file. That file is located on the hard disk, in the folder %windows%\system32\config. The %windows% stands for the path to your Windows folder. For example, C:\Windows\System32\Config\SAM. The system has priority access to the SAM file, so access to the file is denied to anyone, even administrators, while the system is loaded; nevertheless, Windows Password Recovery bypasses that restriction with ease. Besides that, of great interest for a potential attacker would be the backup of the SAM.SAV file and the compressed archived copy of SAM in the folder %windows%\Repair. Another way to access the SAM file is to launch a special program from a boot disk and then copy the file. Anyway you need a physical access to the computer with password hashes. User passwords or, to be accurate, hashes are additionally encrypted with the SYSKEY utility, which stores its service data in the SYSTEM registry file. Thus, to extract hashes from SAM, you would also need the SYSTEM file, which is located in the same folder as SAM.

Active Directory: Domain accounts are stored in the Active Directory database. Usually, the Active Directory database is located in the file %Windows%\ntds\NTDS.DIT; it is the core of Active Directory. The way user hashes are encrypted here is a bit different than that is in SAM, but the recovery would also require the SYSTEM file. Access to the database is also under the system's complete control; however, unlike SAM, the ntds.dit database is resistant to modifications from the outside.

Q. If everything is so easy, why not simply deny access to SAM or Active Directory to all users?

A. That's the way it's done. By default, only the system has access to those files. However, these restrictions can be easily overridden. For example, WPR can import hashes from the current (locked by the system) files SAM and AD. Besides that, the system stores hashes in the computer memory to speed up access to them, so dumping the computer's memory is also an option.

Q. I didn't quite understand it; what do I need to copy from the computer to recover the passwords?

A. If that's a regular computer, copy these files: SAM, SYSTEM (the SECURITY and SOFTWARE files are also desired). If that's a server, you will need the same files plus ntds.dit one.

Q. How long does it take to pick the password if the LM hash is available?

A. The greatest disadvantage of the LM algorithm is that it splits the password into halves of 7 characters long. If user enters a password that is shorter than 14 characters, the program trails it with zeros to get a 14-character long string. If user password exceeds 14 characters, the LM hash appears the same as for an empty password. Each of the 7-character halves is encrypted independently; that considerably eases and speeds up the password recovery process. Another major disadvantage of the LM hash relates to the fact that during the encryption all the alphabetic characters of the password are converted to uppercase. In other words, the hashes for PASSWORD, password, Password or pAsswOrd will be completely identical. By running a brute force attack against each half, modern personal computers can pick an alphanumeric LM hash within a few minutes (or even seconds, when using the Rainbow attack). Let's do a bit of calculation. To pick a password for any alphanumeric combination, we need to split the password into two 7-character long parts and then search 36+32^2+..36^7=80 603 140 212 combinations. Besides, all the hashes will be searched simultaneously. The search speed in Windows Password Recovery on a computer Intel Core i7 is over 100 million passwords per second. Let's round it downward to 100. 80 603 140 212 / 100 000 000 = 806 seconds. That means, we are guaranteed to get the right password within just a bit over 10 minutes using the brute force.

Q. Can I see the encryption sources?

A. Sure. Let's review a working password encryption program for the LM algorithm.

Q. How much time is it required to guess the password if its NT hash is known?

A. With NT hashes it's a bit more complicated. The NT hash does not have the disadvantages that are common to LM. Therefore, the probability of the recovery of the password completely depends on its length and complexity, and drops like a snowball. Even despite the fact that the NT conversion algorithm is faster. Let's take a look at the following table that demonstrates the how search time depends on password length and complexity. Assuming that the brute-force recovery speed is 10 Bln. p/s (1 top GPU in 2014).

Character set	Password length	Password sample	Time to crack
A .. Z	5	CRUEL	instantly
A .. Z	6	SECRET	instantly
A .. Z	7	MONSTER	instantly
A .. Z	8	COOLGIRL	22s
A .. Z	9	LETMEKNOW	~ 10m
A .. Z, 0 .. 9	5	COOL3	instantly
A .. Z, 0 .. 9	6	BANG13	instantly
A .. Z, 0 .. 9	7	POKER00	8s
A .. Z, 0 .. 9	8	LETMEBE4	~ 5m
A .. Z, 0 .. 9	9	COOLGIRL1	~ 3h
A .. Z, a .. z, 0 .. 9	5	P0k3r	instantly
A .. Z, a .. z, 0 .. 9	6	S3cr31	10s
A .. Z, a .. z, 0 .. 9	7	Didlt13	~ 6m
A .. Z, a .. z, 0 .. 9	8	GoAway99	~ 6h
A .. Z, a .. z, 0 .. 9	9	19Sample3	~ 16d

Q. How much time is it needed to guess NT password by it's LM hash?

A. Almost instantly.

Q. Why can't I just remove/drop the hash, i.e. set a blank password?

A. Who said you couldn't? You can. For instance, using this utility. This way is just fine for those who need to regain access to their (or someone else's - e.g., when talking about the respective authorities) account at any cost. Moreover, with the above mentioned utility, you can do the following: remember the hash, then reset the hash, log on to the account with an empty password, do necessary manipulations with it, and then restore the remembered hash back. But that's not as simple as it seems. Even if you have reset the password and gained access to the account, you still won't be able to recover the majority of other passwords. Why? - Because the user password participates in the creation of the user's master key, which is used in the DPAPI and EFS encryption and other Windows subsystems. In other words, even if you reset the password, you will not be able to recover any of the following data: EFS-encrypted files, Outlook account passwords, Internet Explorer 7-9 passwords, network connection passwords (RAS, DSL, VPN etc.), network passwords to other computers, wireless network keys, MSN Messenger credentials, Google Talk & Google Chrome passwords, Skype, etc.

Q. So, in order to recover, for example, an Internet Explorer password, I would need to get the account password first, right?

A. Exactly.

Q. Are there any backdoors?

A. Like anywhere else. For example, sometimes the account password can be stored in the plain-text form in the secrets. Passwords to many system accounts can also be recovered with ease.

Q. Is that what the SECURITY registry file is requested for when importing hashes from the local computer?

A. Yes. The Security's main purpose is to be a storage for the so-called LSA Secrets. These secrets (but not they alone) can store plain-text passwords. Artificial Intelligence attack implements a check-up for possible vulnerabilities in the system and, as the consequence, chances to recover some passwords.

Q. Can I tuck an existing hash instead of the password when logging on to the system?

O. There are programs that do that. Here is how they work. Before booting up the system, they extract user password hashes from SAM. Then, when loading the account, they tuck the known hash instead of the password. However, the result of such manipulations is the same as of merely resetting the password; i.e. you won't be able to recover the majority of other passwords.

Q. What can I do if the SAM file is hopelessly corrupt? Is there a way to recover the original password in this case?

O. Yes, there is. However, you will no longer have access to the system. You can, for example, pick the password using the user's master key. Passcape Software has means for doing that. If the computer belongs to a domain, the names and hashed passwords of the last ten users registered on the computer are cached in its local system registry, in the SECURITY\Policy\Secrets section. You can take advantage of Reset Windows Password for dumping those hashes (they are also referred to as MSCACHE) and then attack them using Network Password Recovery Wizard.

Q. I need to regain access to my account. Would you draw a picture "for dummies" - what's the best way to do that, and how do I do that?

A. Briefly, there are two ways to regain access to an account.

1. Reset the password; e.g., make the password blank. There are special utilities for doing that; the most powerful one is Reset Windows Password. Its operation principle is simple. Run a boot disk creation program and create an Reset Windows Password boot CD/DVD or USB disk with it. Next, power on the computer with the account you need to regain access to and edit the BIOS settings to enable the computer to boot from CD/DVD /USB. Some computers have this option enabled by

default. Now boot up from the Reset Windows Password boot disk and follow the wizard's instructions to reset the password to the account. However, resetting the password guarantees only access to the account. If you also need to regain access to EFS-encrypted files or recover other passwords (e.g., network ones), this method won't do for you.

2. Recover the original password. By the way, that can be done by that same Reset Windows Password, running the intellectual attack. However, its capabilities are limited by only weak and vulnerable passwords. For restoring the original password, it is recommended to use Windows Password Recovery. In this program, once the hashes are imported, select and launch one of the proposed attacks. If the attack did not succeed, you can alter the settings and run the attack over or replace it with another one. Read on to find out how to choose the best attack for your hashes.

Q. Where can I find word-lists for dictionary attacks?
A. It is not necessary to search it. You can download dictionaries from within the Windows Password Recovery. We have a huge set of dictionaries at our Web site.

Q. How do I make my password more secure?
A. There are several ways how you can secure yourself from picking your passwords by potential attackers:

- Do not use dictionary words in any language, names, numbers, repetitive sequences of letters and numbers, abbreviations, keyboard combinations, personal information, etc. Such passwords can be guessed extremely fast and easy.
- Increase password length. However, there is a reasonable limit for everything. Remember that length is not the main thing (although not with passwords). Finally, making up a too long password will cause you to successfully forget it after a weekend party or vacation. Besides that, an average human's memory cannot hold more than 5-7 passwords at a time. Still, there are network password, Web password, etc. - that are to be remembered also.
- Extend the character set used in the password. For example, replace the 'a' characters in the password with the '@'. Using national characters also strengthens up passwords radically. Use uncommon characters; for instance, '~'. Do not use hard-to-remember passwords that consist of a random set of characters - unless you are a genius.
- Do not use the same password for logging on to Windows, Web sites, services, etc.
- If you have trouble remembering all your passwords, save them in a separate password-protected file in a safe place. A good password protection is implemented, for example, in the Rar archiver. Do not keep that file on the local computer.
- Never enter your password on someone else's computer.
- It's not a good idea to write down your passwords on sticky notes and stick those on the monitor.
- Think about additional protection. For example, if you enable the SYSKEY startup password option, chances are close to 100% that not a single attacker will be able to break your passwords without having guessed the original SYSKEY password first.

3.5 Windows Password Recovery FAQ

Q. What do the question marks in LM passwords mean?
A. As you may have already known, an LM password consists of two halves. If an LM password has 7 leading question marks, that means that only the second half of the password is found. The trailing question marks indicate the first half of the password recovered.

Q. What's the difference between LM and NT passwords? I have found both passwords: MASTERGURU and MasterGuru. Which of them is the right one? Which one should I use?
A.To log on to the system, you need to use the NT password.

Q. When brute forcing an LM password, the program complains and tells me that it truncates

the password to 7 characters. Is that a bug?
A. No. As you know, an LM password is split into two 7-character halves. Therefore, the maximum length of brute forced LM passwords is 7 characters.

Q. I know my NT password, but the program fails to find it for some reason? Why?
A.The NT password is case sensitive. Perhaps, you have set an incorrect search range. Try checking the password manually in (Tools-Password Checker). Password Checker automatically checks all possible combinations of uppercase and lowercase characters.

Q. I have recovered the internal administrator password, but when attempting to log on with it, the system tells me that the password is incorrect. What's the matter?
A. Most likely, you have recovered the local administrator's password, while your computer belongs to a domain. Domain passwords are stored in Active Directory, including the domain Administrator's password. Try logging on to the system in the safe mode.

Q. During a dictionary attack, I have recovered a password that was not in the dictionary. How did that happen?
A. Most likely, you had set the maximum mutation level, when the program also checks dictionary words typed in a non-English, national character set, depending on the keyboard layout. For example, the word 'secret' typed with the Cyrillic layout will produce the word 'ыускуе'. Besides swapping keyboard layouts, the active mutations can mutilate the words to the point where they are hard to recognize. Mutation is used in the preliminary, intellectual, dictionary, and combined attacks, as well as in the key word and phrase attacks.

Q. In a batch attack, can I set the same attack type but with different settings?
A. Yes, you can do that.

Q. I've got a question concerning online dictionaries. I've noticed that they are extremely compressed, to the level greater than those that are produced by the archivers. What is the PCD format?
A. That is a proprietary dictionary storage format developed in Passcape, which uses additional optimization and encryption algorithms. Some dictionaries can indeed be compressed harder than with a regular archiver. For example, the Australian.pcd dictionary in the original format takes 926 KB of space, while in the compressed format it's only 53 KB.

Q. I chose to run a dictionary attack and set the medium mutation level. When I launched the attack, I was unpleasantly surprised with the low speed, only a few thousand passwords per second. Why is it so slow?
A. The program shows the attack speed without mutations. For example, if 1000 words has been processed within a second, it shows 1000 p/s, although the mutation module could have generated 1000 additional words per each word during that time. Thus, the actual search speed is by hundreds or even thousand of times greater than what you see on the screen.

Q. Can I use the regular dictionaries in a combined dictionary attack?
A. Yes, you can.

Q. I know that the password begins with "blue". Which attack would be the best one to use?
A. You can try dictionary attack. For example, the mask blue%c%c%c%c%c%c would search the range from blueaaaaaa through bluezzzzzz.
You can also try running a combined dictionary attack. In order to do that, open notepad, then type 'blue' and save the file as, for instance, 1.dic. Then open the combined attack options and set 1.dic as the primary dictionary and any other - as the secondary dictionary. This way the program would search for disyllable words like bluepig, blueberry, bluegirl, etc. If you add the third dictionary, the program will search through the combination of the three components. For example, bluecoolgirl, blueblackhash,

bluebadboy.

Q. The Artificial Intelligence attack goes too slow. What's the matter?
A. It's either because the password cache is full. In this case, you need to try emptying it. Or because you have set too deep mutation, and the program has found quite many 'suspicious' words; i.e. the words that are considered as the potential passwords.

Q. I am launching the brute force, but the program complains that it has nothing to do. Why?
A. Before launching the brute force, you must first select the hashes. You can do that through the Edit - Select menu.

Q. What are Rainbow tables? And how can they be used for recovering passwords?
A. To launch a rainbow attack, in the attack options you need to load the *.RT or *.RTI files that contain Rainbow tables. The type of the tables must match the type of the hashes selected for the attack. Therefore, the names of the files with the tables must begin correspondingly: "lm_*.rt" for LM hashes, "ntlm_*.rt" for NT hashes. You can get some additional information and download rainbow tables at http://project-rainbowcrack.com.

3.6 GPU FAQ

Q: What are the system requirements for the program?
A: Currently the program supports NVidia video cards with CUDA compute capability 2.0 or higher and AMD Radeon 5xxx or higher GPUs. The full list of CUDA supported devices can be found at http://developer.nvidia.com/cuda-gpus. Compatible AMD Radeon cards are shown here: http://en.wikipedia.org/wiki/Comparison_of_AMD_graphics_processing_units.

Q: What versions of Windows the program supports?
A: GPU acceleration is supported starting up with Windows XP (NVidia GPUs) and Windows Vista (AMD GPUs) on both 32-bit and 64-bit systems.

Q: How do I know which architecture does my video card support?
A: For NVidia devices:
Launch the program, open the menu '*Options - General Options*,' select the '*GPU Settings*' tab, select '*NVidia CUDA*' platform and choose your video card here. The 'Compute capability' field in the description section should display your GPU architecture.
For AMD devices:
Launch the program, open the menu *'Options - General Options*,' select the '*GPU Settings*' tab, select '*AMD OpenCL*' platform and choose your video card here. The 'CL_DEVICE_VERSION' and 'CL_DEVICE_OPENCL_C_VERSION' fields should display your GPU architecture supported.

Q: Where can I get the latest video drivers?
A: You can download the latest drivers from NVidia (http://www.nvidia.ru/drivers) and AMD (http://support.amd.com/us/gpudownload/Pages/index.aspx) web sites.

Q: Where can I read more info about CUDA?
A: Wikipedia site is a good starting point to start from.

Q: Where can I read more info about AMD Radeon cards?
A: http://en.wikipedia.org/wiki/Comparison_of_AMD_graphics_processing_units

Q: After I launch a GPU-based attack, my computer freezes or crashes into BSOD. What's the

problem?
A: The problem may be caused by the following reasons:

- Your video card had been overclocked, and it was malfunctioning at high load. If that's the case, bring the frequencies of the video memory/cores to its defaults.
- Insufficient or ineffective cooling of your card. When you launch a GPU-based attack, the program utilizes the most of the GPU power, and the GPU temperature rises to a critical level. Make sure that your video card is well cooled, the GPU slot and your system unit are free from dirt and dust. An unwise use of some video settings may have a negative impact on the video card's temperature and its stability under high load conditions. For example, some applications reduce the fan speed to minimize the noise, which does result in noise reduction, but also increases the core temperature.
- Power supply problem. Your card can consumes a lot of energy at full load, and the power supply unit may be unable to handle such a high demand for power. If the video card has additional 6-pin or 8-pin power connectors, make sure they are all properly connected.

Q: When I launch a GPU attack, my computer slows down a great deal. How can I fix that?
A: By default, the application is set up for using video cards of medium performance. That's usually 256 threads per block, 256 blocks and 1000 passwords per thread. For older video cards such a configuration is too much and may cause a slowdown. Consider reducing the value of "*Passwords per thread*" to 100 or even less.

Q: What's the best way to find optimal values of "*Thread blocks*" and "*Passwords per thread*" in the GPU attack settings?
A: You can do that either empirically or by doing some maths. For example, if the values are 100 and 100, and the average speed of attack is 1 billion passwords per second, you can calculate that the GPU kernel is called about 390 times per second (the number of passwords calculated each time is usually 256 * ThreadBlocks * PasswordsPerThread). Naturally, the fewer calls, the less the overhead, and the higher the attack speed. On the other hand, you must call the GPU program at least a couple of times per second. So use a calculator, and adjust the parameters. You can also adjust them using a rule of thumb, that is, increasing their values until the speed of attack stops going up and the computer slows down. If you have a GPU monitor installed in your system, it should indicate a load of at least 98-99 percent. Besides, it's important to know some other things too. First, don't set the summary values of those parameters too high. Otherwise your system may malfunction or freeze. Second, you'd better not set the value of "*Passwords per thread*" at less than 100 as it will negatively affect the speed of attack regardless of what kind of video card is used.

Q: Does the PCI-Express bus have any impact on the performance?
A: Actually, this impact is negligible. It's usually masked by other factors. So the generation of your PCI-Express bus and its performance don't matter much.

Q: Does the amount of video memory matter?
A: No, it doesn't. However in most cases, your GPU should have at least 256 Mb of video memory.

Q: A GPU-based attack slows down my PC so I can barely use it. How can I fix it?
A: There are two ways to fix it: temporary and permanent. As a temporary fix to the problem, go to the attack settings and try reducing the number of GPU blocks used or the number of passwords checked per GPU thread. As a permanent fix, install a second video device, provided that you have a second slot on your motherboard and that your power supply unit can handle the additional load. For example, you can use some cheap card as the primary one (for displaying information on your monitor), and a second, more powerful one, for brute-forcing passwords.

Q: I have more than video cards in my computer. Can I use them all for brute-forcing?
A: Yes. You can use all or some of them. Just open general settings and specify the GPU device(s) to be used by the program.

Q: What's the maximal number of GPU devices does your program support?
A: It depends on your hardware. Even though the program supports up to 255 devices, typically, up to 8 devices can be installed into a 4 PCI-E slot motherboard (4 double-GPU cards).

Q: Can I brute-force passwords on devices which performance varies a lot?
A: Yes, you can.

Q: The program can not detect my video card. What can I do?
A: Update your video drivers. If it didn't help, try to extend your desktop to all devices (if you have more than one device). Re-plug your device into another PCI-Express slot.

Q: Your application can't use all of my GPUs.
A: You will have to disable SLI in order to be able to use all devices.

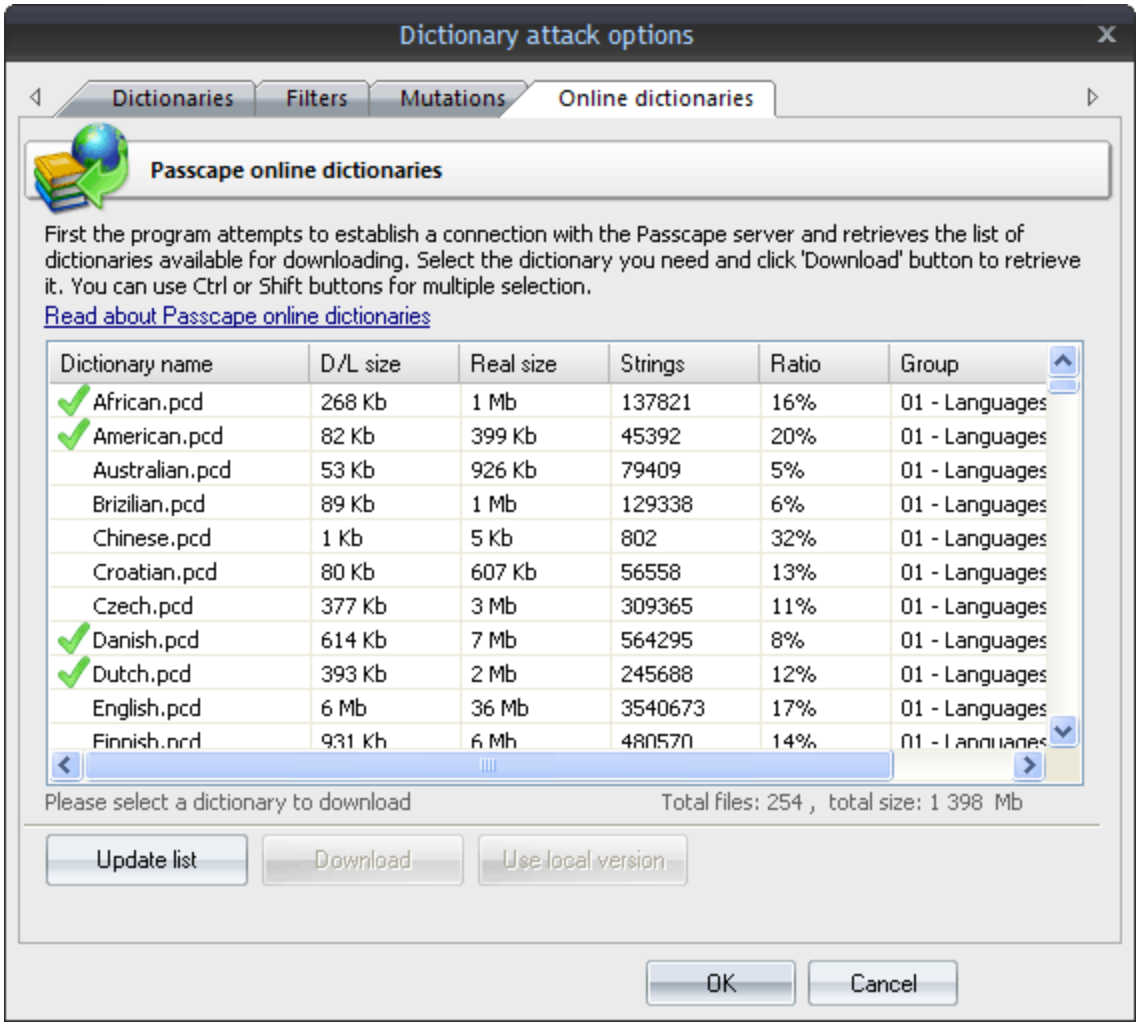
Q: Can I use both NVidia and AMD devices simultaneously?
A: Yes, you can use NVidia and AMD devices simultaneously.

Q: How can I check my GPU utilization?
A: Open 'Hardware Monitor' tab. In 'What to show' drop-box choose the device you need and select 'Show' to display it. You can then click 'Start' or 'Stop' buttons to manage the hardware monitoring. The GPU monitor shows device load (utilization), temperature and fan speed.

Q: My NVidia GPU is absent in hardware monitor.
A: You should install/reinstall NVAPI library. Download the library at https://developer.nvidia.com/nvapi

Q: My AMD GPU shows zeros in hardware monitor.
A: Install/reinstall the latest AMD drivers or ADL component. Make sure other GPU monitoring tools (for example, ATI Catalyst) are functioning correctly.

3.7 Online dictionaries



The online dictionary selection dialog is extremely simple. When it opens up, the program attempts to establish a connection with the Passcape server and then retrieves and displays the list of dictionaries available for downloading.

Select the dictionary you need and then click on the *'Download'* button to retrieve it and use in the program.

Some of the dictionaries are large. For instance, the size of *'music_songs.pcd'* is more than 59 MB in the compressed format. Naturally, retrieving such a large amount of data may take some time, which depends upon file size, bandwidth of your Internet connection and net load.

All online (and some additional) dictionaries can be ordered on CD. The total size of all the dictionaries is over 7.5 GB. You can also share your own dictionary with us by e-mailing us the dictionary or the link where it can be downloaded.

The word-list are used in common dictionary attack, combined dictionary and pass-phrase attacks.

License and registration

4 License and registration

4.1 License agreement

==========
SOFTWARE LICENSE AGREEMENT
==========

IMPORTANT-READ CAREFULLY: This is the End User License Agreement (the "Agreement") is a legal agreement between you, the end-user, and Passcape Software, the manufacturer and the copyright owner, for the use of the "Windows Password Recovery" software product ("SOFTWARE").

All copyrights to SOFTWARE are exclusively owned by Passcape Software.

The SOFTWARE and any documentation included in the distribution package are protected by national copyright laws and international treaties. Any unauthorized use of the SOFTWARE shall result in immediate and automatic termination of this license and may result in criminal and/or civil prosecution.

You are granted a non-exclusive license to use the SOFTWARE as set forth herein.

You can use trial version of SOFTWARE as long as you want, but to access all functions you must purchase the fully functional version. Upon payment we provide the registration code to you.

Once registered, the user is granted a non-exclusive license to use the SOFTWARE on one computer at a time (for every single-user license purchased).

With the personal license, you can use the SOFTWARE as set forth in this Agreement for non-commercial purposes in non-business, non-commercial environment. To use the SOFTWARE in a corporate, government or business environment, you should purchase a business license. With the business license you can run the SOFTWARE on multiple computers of your organization - no matter where they are located.

The registered SOFTWARE may not be rented or leased, but may be permanently transferred together with the accompanying documentation, if the person receiving it agrees to terms of this license. If the software is an update, the transfer must include the update and all previous versions.

You may not create any copy of the SOFTWARE. You can make one (1) copy the SOFTWARE for backup and archival purposes, provided, however, that the original and each copy is kept in your possession or control, and that your use of the SOFTWARE does not exceed that which is allowed in this Agreement.

The SOFTWARE unregistered (trial) version may be freely distributed, provided that the distribution package is not modified. No person or company may charge a fee for the distribution of the SOFTWARE without written permission from the copyright holder.

You agree not modify, decompile, disassemble, otherwise reverse engineer the SOFTWARE, unless such activity is expressly permitted by applicable law.

Passcape Software does not warrant that the software is fit for any particular purpose. Passcape Software disclaims all other warranties with respect to the SOFTWARE, either express or implied. Some jurisdictions do not allow the exclusion of implied warranties or limitations on how lomg an implied warranty may last, do the above limitations or exclusions may not apply to you.

The program that is licensed to you is absolutely legal and you can use it provided that you are the legal owner of all files or data you are going to recover through the use of our SOFTWARE or have permission from the legitimate owner to perform these acts. Any illegal use of our SOFTWARE will be solely your responsibility. Accordingly, you affirm that you have the legal right to access all data, information and files that have been hidden.

You further attest that the recovered data, passwords and/or files will not be used for any illegal purpose. Be aware password recovery and the subsequencial data decryption of unauthorized or otherwise illegally obtained files may constitute theft or another wrongful action and may result in your civil and (or) criminal prosecution.

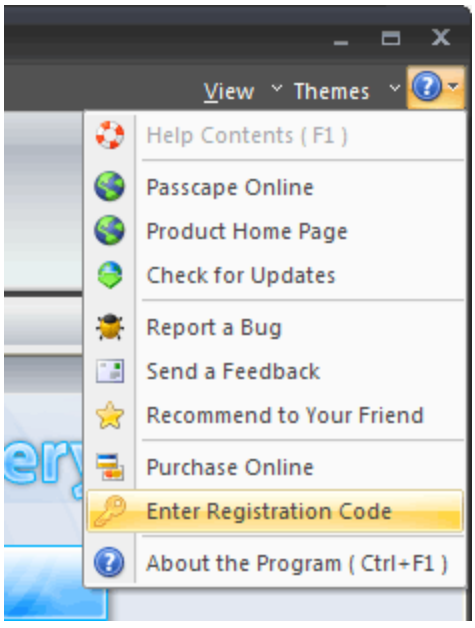
4.2 Registration

The software is available in three editions: Light, Standard and Advanced. The detailed list of features is shown here. You can order fully registered version of Windows Password Recovery at a cost of $65 for Light Edition (personal usage), $345 for Standard Edition (personal usage) or $895 for Advanced Edition (business license).

Detailed instructions for all kinds of orders are available online at WPR order page. Online orders are fulfilled in just a few minutes 24 hours a day 7 days a week. If you purchase our products online, you will receive an automatically generated e-mail message with registration details within several minutes (if the order passes the fraud check system). However some orders can be marked for manual checkout or as 'suspicious'. This may increase order time up to several hours.

Important: when completing the order form, please double-check that your e-mail address is correct. If it will not, we'll be unable to send you the registration code.

To complete the registration:



- Open the registration message and copy the registration code to the Windows clipboard.
- Run the program, select **Help - Enter Registration Code.**
- Type in your registration name and paste the code here.
- Click **Register** button to confirm.

4.3 Limitation of unregistered version

An unregistered version of **Windows Password Recovery** shows only first 3 characters of recovered passwords and has some functional limitations. Registered version of the program eliminates all restrictions. Please refer to this page to view restrictions of a certain edition of the program.

4.4 Editions of the program

Windows Password Recovery comes in three editions: Light, Standard and Advanced. The detailed list of features and compatibility chart is shown below.

FEATURE	Light	Standard	Advanced
Windows 2000/XP/Vista/7/8/10 workstation support	+	+	+
Windows 2000/2003/2008/2012 server support	+	+	+
Windows 64-bit support	+	+	+
Non-US Windows support	+	+	+
International passwords support	+	+	+
Multithreaded recovery	+	+	+
Interface themes support	+	+	+
Load hashes from local computer	+	+	+
Load hashes from remote computer	-	+	+
Dump regular hashes	+	+	+
Dump password history hashes	+	+	+

FEATURE	Light	Standard	Advanced
Search for plaintext passwords	+	+	+
Load hashes from SAM	+	+	+
Load Active Directory hashes	+	+	+
Import hashes from other programs	+	+	+
Load hashes from system restore folders	+	+	+
Export hashes to PWDUMP file	+	+	+
Common attacks	+	+	+
Advanced attacks	+	+	+
Smart attacks	+	+	+
GPU-based attacks	+	+	+
Support for multiple GPU devices	-	-	+
Batch attack	-	-	+
View AI password cache	-	-	+
Smart password mutation	+	+	+
Online dictionaries	+	+	+
Support SYSKEY decryption	+	+	+
Support SYSKEY startup password decryption	+	+	+
Support SYSKEY floppy decryption	+	+	+
Custom wordlist generator tool in dictionary attack	-	-	+
Generate dictionaries by mask	-	-	+
Generate dictionaries by given base-word	-	-	+
Combined dictionaries generator	-	-	+
Pass-phrase dictionary generator	-	-	+
Fingerprint dictionaries generator	-	-	+
Create wordlists based on hybrid dictionary attack	-	-	+
Support for hybrid and indexed (*.rti) rainbow tables	+	+	+
Can restrict access to the program	+	+	+
Password strength measurement	+	+	+
Hash checker	+	+	+
Random hash generator	+	+	+
Multiple hashes generator	-	+	+
Rainbow table generation tool	+	+	+
Passcape rainbow table generation tool	+	+	+
Dictionary to hash generator	-	+	+
Backup system registry files	-	+	+
Backup Active Directory database	-	-	+
Asterisk password viewer tool	+	+	+
Offline password remover tool	-	-	+
LSA secrets Dumper	+	+	+
Domain cached credentials explorer	-	+	+
SAM Explorer	-	+	+
Active Directory Explorer	-	-	+
Windows Vault Explorer	-	-	+
Wordlist tools: create a wordlist by indexing files	-	+	+
Wordlist tools: merge wordlists	+	+	+
Wordlist tools: wordlist statistics	+	+	+

FEATURE	Light	Standard	Advanced
Wordlist tools: sorting	+	+	+
Wordlist tools: convertion/compression	+	+	+
Wordlist tools: wordlist comparison	+	+	+
Wordlist tools: additional operations	+	+	+
Wordlist tools: indexing words/passwords of HDD sensitive areas	-	-	+
Wordlist tools: HTML links extractor	+	+	+
DPAPI: offline DPAPI blob recovery	*	*	+
DPAPI: DPAPI blob analysis	+	+	+
DPAPI: DPAPI blobs search	+	+	+
DPAPI: Master Key analysis	*	*	+
DPAPI: dump password history hashes	-	-	+
DPAPI: analyse password history	*	*	+
Hardware monitor	+	+	+
Password reports	-	+	+
Run in hidden mode	+	+	+
Max. user accounts at a time	500	5000	unlimited
14-days money back guarantee	+	+	+
License	personal	personal	business
Price	$65	$345	$895

* - uses some restrictions

Technical support

5 Technical support

5.1 Reporting problems

If you have a problem, please contact us at support@passcape.com. Please inform us about the following:

- Full name and version of the program
- Windows version including service pack, OEM and language information, etc.
- Registration information if any
- Detailed description of the problem, whether it is a constant or spontaneous error
- If you're reporting a critical error, please attach Crash.log file that was saved during an unhandled exception session.

5.2 Suggesting features

If you have any questions, comments or suggestions about the program or would like more information, email us at info@passcape.com. Please don't forget to mention the program name and version. Also make sure you have the latest program version installed. Your feedback helps us to improve our products and work more effective.

5.3 Contacts

Please don't hesitate to send your questions regarding our products to e-mail support@passcape.com. You will get reply during one or two days. Note, that registered users have priority in technical support.

If you experience any problems during registration process, please send a letter to sales@passcape.com

We will be happy to assist you with the registration.

Please write in English!

You can find other password recovery utilities at http://www.passcape.com

05/10/2015