Creating unique custom dictionaries for password recovery

Passcape Software

1 Abstract

A lot of water has flown under the bridge since the first password recovery tool has appeared, but a dictionary attack (or wordlist recovery) is still one of the fundamental methods of password hacking used in the majority of such programs. However, if in 1988 all that the first widely-known computer worm needed for quick propagation was 400 most commonly used words, the present-day password recovery utilities require larger and more complex dictionaries.

2 Creating custom wordlists

How you can tell a professionally-written password recovery program from a makeshift utility slapped together by an amateur? It is really very simple. If there are unique dictionaries in the program installation kit or on the website, you can be fairly sure that it is a professional program. Amateur programmers usually do not bother to create new dictionaries and simply use the existing ones; especially, as there are plenty of them on the Internet. The most amazing thing is that the majority of large downloadable wordlist have a lot of unusable trash and their makers do not take the time to clean them up, simply merging all existing stuff and files into one huge heap. As a result, you get 10-20 GB of trash with HTML tags, long strings of unintelligible text, and even copyrights of other dictionaries, unusable in serious projects.

Creating quality, themed word lists is a complicated and thankless task – it takes forever to create new files. In this article we will try to show how you can quite effectively automate some routine operations when creating your own unique dictionaries. In order to do that you will need a bit of imagination and a set of utilities for dictionary processing. We'll start small and try to collect our set from several popular languages. Each dictionary will contain the most popular words. In the second part we will quickly throw together a list of 1,000,000 most popular Internet domains. Finally, in the third part, we will create a huge (over 1.5 GB), actually the largest known, unique dictionary that will have over 124 million records of user names as the output. So, here goes!

2.1 Creating international dictionaries

Recently, the latest version of the Android KitKat dictionary databases were published. These databases are used for word auto-completion. The English database, for example, contains over 150 thousand words. We can use them to create our own English and international word-lists. Each database is a compressed file with records like these:

```
word=days,f=152,flags=,originalFreq=152
word=developed,f=152,flags=,originalFreq=152
word=east,f=152,flags=,originalFreq=152
word=election,f=152,flags=,originalFreq=152
word=formed,f=152,flags=,originalFreq=152
```

It's easy to clear each record of unwanted information using a special filter. Let's open the dictionary creation tool, specify the path to the unzipped database file, and set the filter option on the basis of the special rules. Let's enter the following text in the rule field:

=0 =1w=2o=3r=4d=5=e,E=

Click on 'Next ' button and have the dictionary ready. Let's review in detail what happened here. The first part (**=0 =1w=2o=3r=4d=5=**) rejects the lines that does not begin with the character sequence we need, i.e. with ' word='. Then the 'e' command takes the character substring from the first position to the first comma. Then '**E=**' takes whatever there is after the equal sign, i.e. the word we need. As you can see, it's all fairly simple. Once you've done it for the rest of the files, you have over two dozen international dictionaries ready for use.

2.2 Generating a list of ~1 mln. popular Internet domains

In recovering archive passwords, there are often passwords that look like the name of a URL address or a domain. Why not create a dictionary with the most popular domain names; after all there is no such dictionary on the net? No sooner said, than done. All we need is the 22 MB database of the 1 million popular domains. Each line of the database consists of the position in the popularity list and a URL address separated by a comma. Give the simplest command 'E,' and we get a clean list of domains without the positions. If we do not need country-code top-level domains (.org, .com, etc.), we can modify the filter a bit for '**E,e.**'. In that case, the file we get after processing will have to be cleared of duplicates, because, for example, the source lines google.com, google.com.hk, google.fr, etc. will simply become google. We'll get around 11 MB or a bit over 900,000 names (without the country domains). You can download an existing dictionary from our website with over 1.1 million records compressed from 14 to 3.5 MB.

2.3 Creating a dictionary with user names

The Adobe website was hacked in the fall of 2013, as the result of which over 150 million encrypted records with e-mail addresses and encrypted passwords of registered users leaked out to the net – poor old Adobe.

Despite the fact that the encryption algorithm was rather outdated (3DES, according to the Company), it was used quite professionally and luckily for the users, their passwords have not been decoded yet. Let's try to create a dictionary of user names by taking out the first part of the e-mail addresses. In fact, such dictionaries are pretty good for password recovery in programs that use special rules of password mutation.

The leaked password database is over 9 GB and consists of records that look approximately like this:

103239615-|--|-techlaf@yahoo.com-|-getc/eE3AqY=-|-normal|--
103239616-|--|-sepro777@yahoo.co.jp-|-QBk2p9UC6XA=-|-|--

103239617-|--|-jack_galante@yahoo.com-|-wiNNHCMhG64=-|-coche|--

First, we should truncate the text we don't need and leave just the e-mail address database. Let's launch the dictionary generation utility and enter the following command as the filter:

%1|E|E|e|][e@

What it does in detail:
%1| - skip the lines that don't have the '|' symbol
E|E|e| - delete all fields except the e-mail address
][- truncate the first and the last symbol (there is a '-' symbol at the beginning and at the end of every e-mail address).
e@ - delete everything after '@'

As an example, let's take a look at what will happen after every rule is applied to an input word:
Input word: **103239615-|--|-techlaf@yahoo.com-|-getc/eE3AqY=-|-normal|--**
Input rules: **%1|E|E|e|][e@**

Rule	Result output
%1|	103239615-|--|-techlaf@yahoo.com-|-getc/eE3AqY=-|-normal|--
E|	--|-techlaf@yahoo.com-|-getc/eE3AqY=-|-normal|--
E|	-techlaf@yahoo.com-|-getc/eE3AqY=-|-normal|--
e|	-techlaf@yahoo.com-
]	-techlaf@yahoo.com
[	techlaf@yahoo.com
e@	techlaf

Once the command is executed for the input wordlist, we get a list of usernames that we have to sort in order to get rid of duplicates. Even after deleting the duplicates the list is quite impressive – over 124 million words. See below for some interesting statistics of the final dictionary.