maketecheasier

Recover Lost PDF Passwords with pdfcrack (Linux)

Attila Orosz 22nd Jul 2015

Securing your important PDF documents with a password can be a great way to ensure your privacy remains unbroken – until you lose, or forget the password.

pdfcrack is a simple command line utility that will attempt to crack the password of any PDF file with a single command, if used the right way, **and on files you have permission to crack**.

Disclaimer

Cracking PDF passwords might not be legal in your country; you should always check for laws and legislation before attempting to do anything of the sort.

Also, pdfcrack is meant to be used on files you own or have obtained permission to crack from the owner. You must **never** use password cracking software such as this to obtain passwords of protected documents you don't have the permissions for or for illegal means.

The following information is provided for educational purposes only – to help users recover documents they own or have obtained permission to crack. The author or anybody representing Make Tech Easier are in no way responsible for the way readers will use the information provided here.

Usage

pdfcrack is readily available from most Linux distros' main repositories. On Debian (and so Ubuntu) derivatives, you can install it with

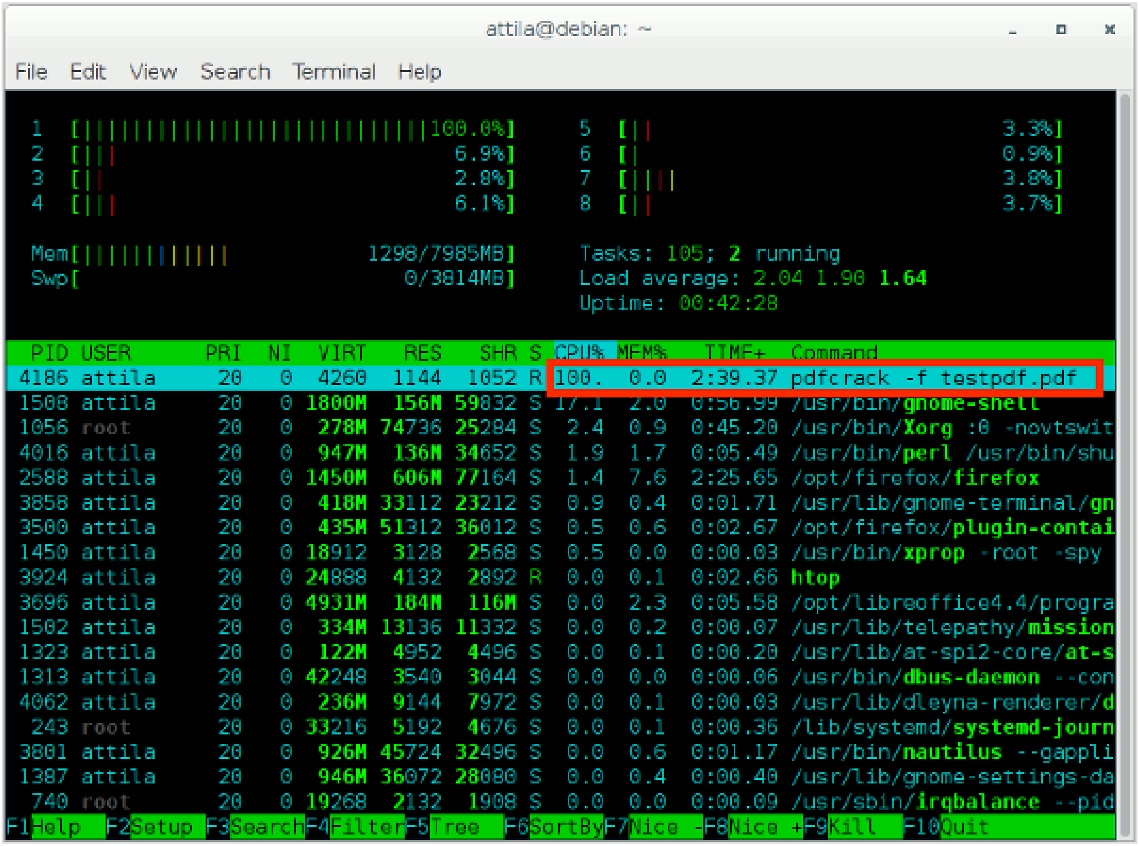
```
sudo apt-get install pdfcrack
```

To crack the password of a PDF file you own, or have permission for, you can use the command

```
pdfcrack -f filename.pdf
```

Password cracking is a long and tedious process. pdfcrack will attempt to crack your file with a series of words (combinations of characters) which can take a long time, especially since it appears to be a single threaded process utilizing only one CPU core regardless of your setup.

Besides using only one core, the single CPU will be run at one-hundred percent for the full duration of the cracking.



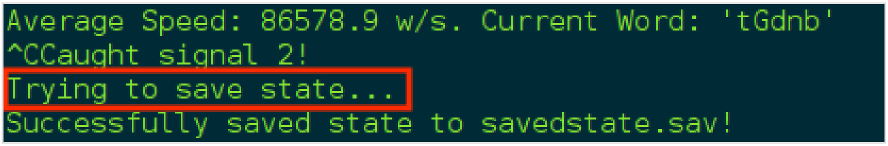
```
  1  [|||||||||||||||||||||||||||||100.0%]     5  [||                              3.3%]
  2  [|||                           6.9%]     6  [|                               0.9%]
  3  [||                            2.8%]     7  [||||                            3.8%]
  4  [|||                           6.1%]     8  [||                              3.7%]

  Mem[|||||||||||||           1298/7985MB]     Tasks: 105; 2 running
  Swp[                           0/3814MB]     Load average: 2.04 1.90 1.64
                                               Uptime: 00:42:28

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 4186 attila     20   0  4260  1144  1052 R 100.  0.0  2:39.37 pdfcrack -f testpdf.pdf
 1508 attila     20   0 1800M  156M 59832 S 17.1  2.0  0:56.99 /usr/bin/gnome-shell
 1056 root       20   0  278M 74736 25284 S  2.4  0.9  0:45.20 /usr/bin/Xorg :0 -novtswit
 4016 attila     20   0  947M  136M 34652 S  1.9  1.7  0:05.49 /usr/bin/perl /usr/bin/shu
 2588 attila     20   0 1450M  606M 77164 S  1.4  7.6  2:25.65 /opt/firefox/firefox
 3858 attila     20   0  418M 33112 23212 S  0.9  0.4  0:01.71 /usr/lib/gnome-terminal/gn
 3500 attila     20   0  435M 51312 36012 S  0.5  0.6  0:02.67 /opt/firefox/plugin-contai
 1450 attila     20   0 18912  3128  2568 S  0.5  0.0  0:00.03 /usr/bin/xprop -root -spy
 3924 attila     20   0 24888  4132  2892 R  0.0  0.1  0:02.66 htop
 3696 attila     20   0 4931M  184M  116M S  0.0  2.3  0:05.58 /opt/libreoffice4.4/progra
 1502 attila     20   0  334M 13136 11332 S  0.0  0.2  0:00.07 /usr/lib/telepathy/mission
 1323 attila     20   0  122M  4952  4496 S  0.0  0.1  0:00.20 /usr/lib/at-spi2-core/at-s
 1313 attila     20   0 42248  3540  3044 S  0.0  0.0  0:00.06 /usr/bin/dbus-daemon --con
 4062 attila     20   0  236M  9144  7972 S  0.0  0.1  0:00.03 /usr/lib/dleyna-renderer/d
  243 root       20   0 33216  5192  4676 S  0.0  0.1  0:00.36 /lib/systemd/systemd-journ
 3801 attila     20   0  926M 45724 32496 S  0.0  0.6  0:01.17 /usr/bin/nautilus --gappli
 1387 attila     20   0  946M 36072 28080 S  0.0  0.4  0:00.40 /usr/lib/gnome-settings-da
  740 root       20   0 19268  2132  1908 S  0.0  0.0  0:00.09 /usr/sbin/irqbalance --pid
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

To speed up the process, you can feed pdfcrack with a subset of characters to try, using the `-c` option. The below example would use the letters ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’ and numbers ‘0’, ‘1’, ‘2’, ‘3’, ‘4’.

```
pdfcrack -f filename.pdf -c abcdef01234
```

This could help if you have a rough idea of what your password looks like, e.g. when you use a few regular passwords to secure your files, but you are unsure which one it is, or you’re afraid you have misspelled one (twice) at the time of setting it up.

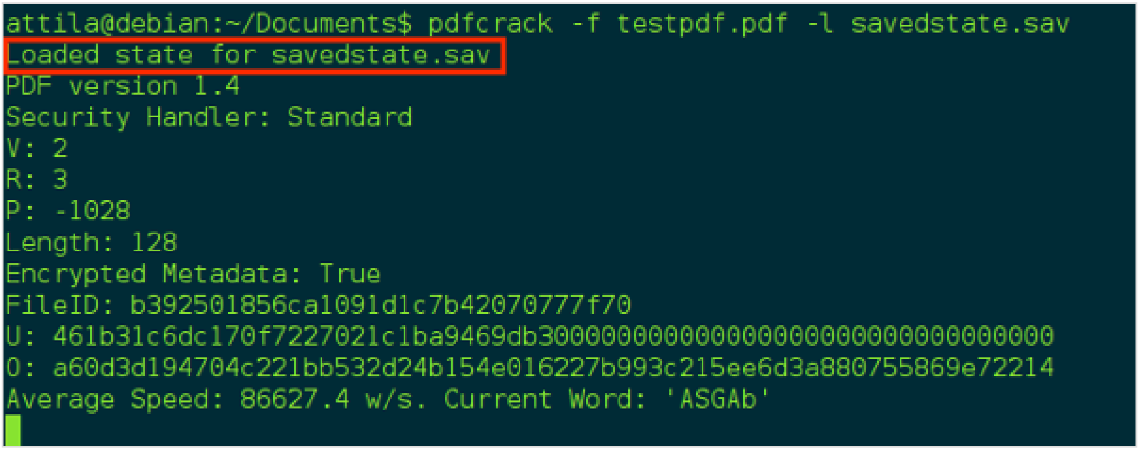
If you are afraid of your CPU getting damaged by the heavy use, you can cancel the process at any time (Ctrl + C). pdfcrack will attempt to save the state of the process,



```
Average Speed: 86578.9 w/s. Current Word: 'tGdnb'
^CCaught signal 2!
Trying to save state...
Successfully saved state to savedstate.sav!
```

and you can later resume the crack from the saved file with the `-l` switch. The default file name for saving is “savedstate.sav.”

```
pdfcrack -f filename.pdf -l savedstate.sav
```



```
attila@debian:~/Documents$ pdfcrack -f testpdf.pdf -l savedstate.sav
Loaded state for savedstate.sav
PDF version 1.4
Security Handler: Standard
V: 2
R: 3
P: -1028
Length: 128
Encrypted Metadata: True
FileID: b392501856ca1091d1c7b42070777f70
U: 461b31c6dc170f7227021c1ba9469db300000000000000000000000000000000
O: a60d3d194704c221bb532d24b154e016227b993c215ee6d3a880755869e72214
Average Speed: 86627.4 w/s. Current Word: 'ASGAb'
```

You can also set a minimum or maximum length for the attempted passwords. This is useful when you know you always use a password longer or shorter than a given length.

To set the minimum length, use the `-n=LENGTH` flag. For example, to only check for passwords longer than five characters you can use

```
pdfcrack -f filename.pdf -n=5
```

To set the maximum length, use the `-m=LENGTH`. For example, to only check for passwords shorter than ten characters, you can use

```
pdfcrack -f filename.pdf -m=10
```

You can of course combine the options. To check for passwords shorter than ten characters but longer than five and only containing the letters “a, b, c, d, e, f,” use

```
pdfcrack -f filename.pdf -m=10 -n=5 -c abcdef
```

Using these options have the potential to give it a considerable performance boost if used the right way.

Other options include using a word list (`-w`), working with an owner password (`-o`) or a user password (`-u`, although this is the default behaviour), or providing a user password to ease the process of obtaining an owner password (`-p`). Permutation (`-s`) is currently limited to switching the first character to uppercase.

You can also use `-b` to perform a benchmark to have an idea of how well pdfcrack might perform

To see how all the options work, just type `pdfcrack`, and it will print its usage.

Performance

For testing purposes, a small sample PDF file was created with LibreOffice Writer 4.4 in two versions. The first version had a random seventeen-character long password with a combination of upper and lower case letters, numbers and non-alphanumeric characters. The sort of password even the most paranoid security geek would decide was “acceptable”.

pdfcrack proceeded with attempting to crack the password for many minutes.

Unfortunately the test had to be canceled as the CPU’s temperature eventually reached 69°C/156°F despite all the cooling attempts (oversized fans, water, cold beer), and any damage for the AMD chipset at that temperature was considered unworthy for seeing how well pdfcrack might perform.

For the second attempt a shorter and much simpler password – “crackme” – was given. pdfcrack delivered the result in under fifteen minutes.

On the third attempt a subset of characters – “a,c,d,e,k,m,r,p,t” – was specified, containing all the characters of the actual password and then some. The result was almost instantaneous.

Of course it is only good to err on the too many characters side. If you don’t provide enough, the password could never be cracked, and the missing characters would never be tried.

The last attempt will probably represent a more realistic scenario where you know the sort of passwords you usually use for your documents, but you’ve forgotten which one the PDF was secured with (if it’s your own document, you should probably know such details), and you can help the password cracker guess for you.

A little trick to speed things up further

pdfcrack is single threaded, meaning it will not be able to use multiple CPU cores, yet you can use bash to run multiple instances simultaneously. If you are uncertain which options to use, or if you have multiple guesses as to the successful setup, this trick would save you some time and effort.

In the following example, three instances are started at once. One is a simple command with no options, the second one includes a subset of letters (which of course includes all the letters of the password “crackme”), and the last one uses a different subset of characters with an option to only check for words longer than five characters. Simultaneous execution (multiple processes, not multiple threads) will be achieved by the `&` operator.

```
pdfcrack -f testpdf.pdf & pdfcrack -f testpdf.pdf -c rkmetacti & pdfcrack -f testpdf.pdf -c hjktr -n=5
```

This will start three processes running on three separate cores

As the snapshot below shows, one of the commands has in fact found the right password very quickly, but the other two kept on trying, so it’s worth keeping an eye on the output.

Another way to make sure this happened would be to filter `top`/`htop`/`atop` output to only show `pdfcrack`. When one of the processes disappears, it probably means it has found a password.

Once you’ve found the password, or if you wish to terminate the run, you need to make sure all started processes die. The easiest way to ensure this would be to use

```
killall pdfcrack
```

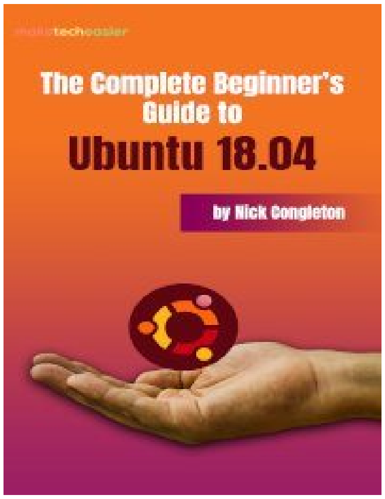
from another terminal window.

Conclusion

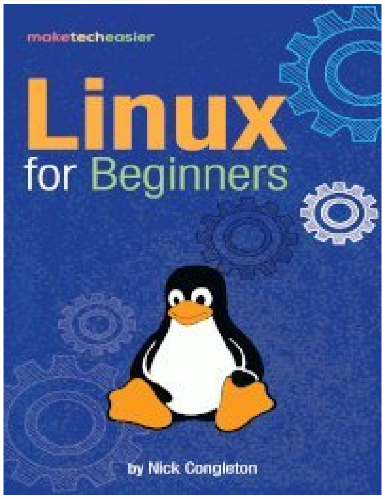
pdfcrack is a simple tool to recover lost passwords of your PDF files or of files you have permission to crack (if regulations allow, of course). It cannot be used to alter any permissions set in the PDF but only to crack a password. While it would eventually discover the most elaborate password, this could take a very long time. With a little knowledge (of your own password setting habits) and setting appropriate options, the process could be considerably set up. So whether you forgot or just mistyped your password while setting it up, you can easily recover them with this single command.

Is this article useful? Yes No

Related Ebooks



The Complete Beginner's Guide to Ubuntu 18.04



Linux for Beginners

The Beginner's Guide to KDE

7 comments

AC2

Check out https://github.com/shreepads/pdfcrack-mp/tree/baseopenmp

Work-in-progress enhancement to support simple patterns (e.g. [abc][abc][abc][123][123][123]) and multi-core support for user and owner passwords – including Rev 5.

Also includes tests. Run 'make test' to get an idea of what is supported…

Sep 6, 2015 at 11:02 am

Attila Orosz

That looks interesting. I'm going to give it a go later on.

Sep 7, 2015 at 12:52 am

AC2

Now also handles patterns with ranges and partial support for character classes.

See Issue #3 and #4 at https://github.com/shreepads/pdfcrack-mp/issues

Pls note that the master branch is to be kept in sync with the original so all multi-core/ pattern changes are in other branches…

Sep 26, 2015 at 11:05 am

Attila Orosz

Nice work. When do you assume it would be release-grade?

Sep 28, 2015 at 2:31 am

AC2

I can’t believe I had to create this one, but…

http://i57.tinypic.com/w7eu6s.jpg

Oct 8, 2015 at 9:33 am

Attila Orosz

Yes, you do.

Oct 12, 2015 at 12:25 pm

AC2

:-)

Am pretty much done with the patterns related work. It now supports regex like character classes with support for ranges of repitition for each (in addition to the basic list of chars patterns).

You can now run something like this test in the Makefile:

./pdfcrack -t 4 -e “[:upper:]{1,2}[:lower:]{2,3}[:digit:][:punct:]” ./testpdfs/TestPDF7.pdf

This will use 4 threads of execution in parallel to find a password that

* Starts with an uppercase letter
* Followed by 1 to 2 lower case letters
* Followed by 2 to 3 digits
* And one punctuation mark

Next up, am planning to add a character class [:word:] which will include words from a wordlist so you can have a pattern like this

{4,8}[:word:]{2,3}[:digit:][:punct:]

:word: is going to be special

* It can only appear once
* The leading range {4,8} indicates that it should use words from the wordlist that are at least 4 chars and if longer than 8 chars truncate to 8 chars

I also want to add support for the new PDF security revision 6 but need a test file. Pls see : https://github.com/shreepads/pdfcrack-mp/issues/16

Nov 22, 2015 at 9:51 am

Comments are closed.

About Contact Advertise Write For Us Terms of Use Privacy Policy RSS Feed Terms