

## How to crack a PDF password with Brute Force using John the Ripper in Kali Linux

MAY 5TH 2019 7.4K 1 COMMENT

### RELATED ARTICLES - KALI LINUX

#### Performing a genuine slowloris attack (SlowHTTP) of indefinite length in Kali Linux

JUNE 12TH 2019

KALI LINUX

#### How to solve Kali Linux apt-get install: E: Unable to locate package checkinstall

JUNE 11TH 2019

KALI LINUX

#### How to route all the machine Traffic Through TOR in Kali Linux

MAY 21ST 2019

KALI LINUX

#### How to protect your Apache server from DoS attacks (denial-of-service) using the quality of service (QoS) module on Ubuntu 16.04

MAY 19TH 2019

OTHERS

John the Ripper is a fast password cracker, currently available for many flavors of Unix, macOS, Windows, DOS, BeOS, and OpenVMS (the latter requires a contributed patch). Its primary purpose is to detect weak Unix passwords. Besides several crypt(3) password hash types most commonly found on various Unix flavors, supported out of the box are Kerberos/AFS and Windows LM hashes, as well as DES-based tripcodes, plus hundreds of additional hashes and ciphers in "-jumbo" versions.

In this article we will explain you how to try to crack a PDF with password using a brute-force attack with JohnTheRipper.

## 1. Build JohnTheRipper binaries

We will need to work with the Jumbo version of JohnTheRipper. This is a community-enhanced, "jumbo" version of John the Ripper. It has a lot of code, documentation, and data contributed by the user community. This is not "official" John the Ripper code. It is very easy for new code to be added to jumbo: the quality requirements are low. This means that you get a lot of functionality that is not "mature" enough or is otherwise inappropriate for the official JtR, which in turn also means that bugs in this code are to be expected.

Proceed to obtain the source code of JohnTheRipper (The "bleeding-jumbo" branch (default) is based on 1.8.0-Jumbo-1) from the repository at Github with the following command (or download the zip with the content and extract into some directory):

How to perform a DoS attack "Slow HTTP" with SlowHTTPTest (test your server Slowloris protection) in Kali Linux

MAY 19TH 2019

KALI LINUX

ADVERTISE IN OUR CODE WORLD

```
git clone https://github.com/magnumripper/JohnTheRipper.git
```

This will create a directory namely JohnTheRipper in the current directory. You can read more about the "Jumbo" version of JohnTheRipper project in the official website or [visit the un-official code repository at Github here](#). Switch to the src directory of JohnTheRipper with the following command:

```
cd ./JohnTheRipper/src
```

Proceed to download the package lists from the repositories with the following command:

```
sudo apt-get update
```

And install libssl:

```
sudo apt-get install libssl-dev
```

The library requires libssl (openssl) to be installed in your system, so in case you don't have it the previous command will do the trick to accomplish this

requirement. Once the repository has been cloned, proceed to enter into the source directory that contains the source code of JohnTheRipper:

```
cd ./JohnTheRipper/src
```

Inside this directory we will proceed with the build with the following instruction:

```
./configure && make
```

This version of Jumbo has autoconf that supports the very common chain, allowing you to compile the sources on a Unix-like system. Once the build process finishes, switch to the run directory inside the JohnTheRipper directory:

```
cd ..
```

```
cd ./run
```

Inside this directory you will find (after the build) all the tools that the library has to offer (including john itself), you can list the directory to compare:

```
ls
```

You will see all the tools of JohnTheRipper inside this directory:

```
root@kali: ~/Documents/JohnTheRipper/run
File Edit View Search Terminal Help
root@kali:~/Documents/JohnTheRipper/run# ls
1password2john.py      codepage.pl          hccap2john           latin1.chr           opensbd_softraid2john.py  rulestack.pl
7z2john.pl            cprepair            hccapx2john.py       ldif2john.pl         openssl2john.py          sap2john.pl
adxcsof2john.py       cracf2john.py       hextoraw.pl          leet.pl              oui.txt                  sha-dump.pl
aem2john.py           dashlane2john.py    htdigest2john.py    libreoffice2john.py  padlock2john.py         sha-test.pl
aix2john.pl           deepsound2john.py   ibmiscanner2john.py  lion2john-alt.pl     pass_gen.pl             signal2john.py
aix2john.py           dictionary.rfc2865   ikescan2john.py     lion2john.pl         password.lst            sipdump2john.py
alnum.chr             diskcryptor2john.py ios7tojohn.pl        lm_ascii.chr         pcap2john.py            ssh2john.py
alnumspace.chr       dmg2john.py         iTunes_backup2john.pl lotus2john.py         pdf2john.pl             sspr2john.py
alpha.chr            dm92john.py         john                 lower.chr            pem2john.py             staroffice2john.py
andotp2john.py        dns                 john.bash_completion lowernum.chr          pfx2john.py             stats
androidbackup2john.py DPAPImk2john.py     john.conf            lowerspace.chr        pgpdisk2john.py         strip2john.py
androidfde2john.py   dumbb16.conf        john.log             luks2john.py          pgpsda2john.py          telegram2john.py
ansible2john.py      dumb32.conf         john.pot             mac2john-alt.py       pgpude2john.py          tezos2john.py
apex2john.py         dynamic.conf        john.zsh_completion  mac2john.py          potcheck.pl             tgtsnarf
applenotes2john.py  dynamic_disabled.conf  jtrconf.pm          mailer                prosody2john.py         truecrypt2john.py
aruba2john.py        dynamic_flat_sse_formats.conf  jtr_rulez.pm        makechr               pse2john.py             uaf2john
ascii.chr            ecryptfs2john.py    jtr_rulez.pm        mcafee_epo2john.py   ps_token2john.py        unafs
axcrypt2john.py      ejabberd2john.py    keepass2john         money2john.py         putty2john              undrop
base64conv           electrum2john.py    keychain2john.py    mongo2john.py        pwsafe2john.py          unique
benchmark-unify     encfs2john.py       keyring2john.py     mozilla2john.py      radius2john.pl          unshadow
bestcrypt2john.py   enpass2john.py      keystore2john.py    multitbit2john.py    radius2john.py          upper.chr
bitcoin2john.py     ethereum2john.py    kirbi2john.py       mypasswd              rar2john                uppernum.chr
bitlocker2john.py   filezilla2john.py   korelogic.conf      neo2john.py           raw2dyna                utf8.chr
bitwarden2john.py   fuzz.dic            krb2john.py         netscreen.py          regex_alphabets.conf   vdi2john.pl
bks2john.py         fuzz_option.pl      kwallet2john.py     netntlm.pl           repeats16.conf          vmx2john.py
blockchain2john.py  genincstats.rb     lanman.chr          network2john.lua      repeats32.conf          wpacpac2john
calc_stat           genmkvpwd           lastpass2john.py    office2john.py       rexgen2rules.pl        zip2john
ccache2john.py      gpg2john            latin1.chr          openbsd_softraid2john.py  rules                  ztex
cisco2john.pl       hccap2john          latin1.chr          opensbd_softraid2john.py  rulestack.pl
root@kali:~/Documents/JohnTheRipper/run#
```

Now that you have the tools to proceed, let's get started with the brute force attack.

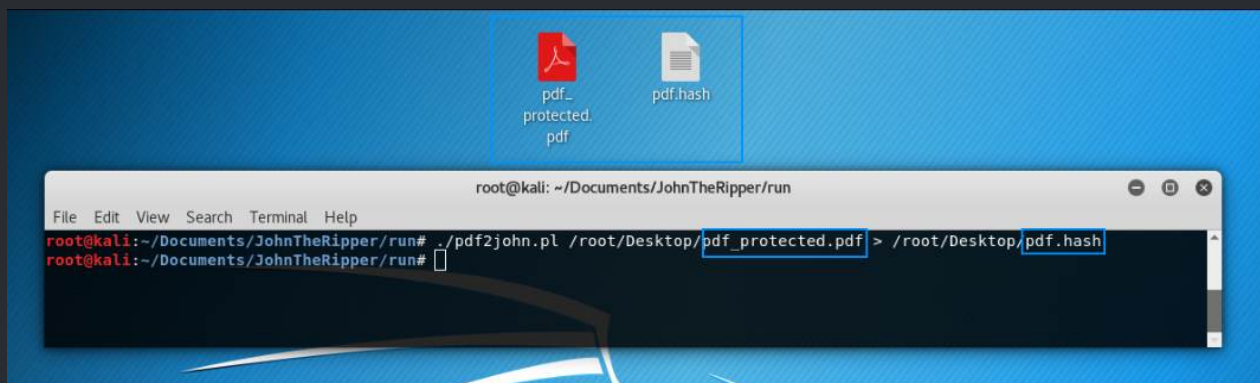
## 2. Generate PDF hash file

JohnTheRipper, as mentioned at the beginning of the article is not related by itself to PDF's, but to passwords and security stuff. That's why you will need to create the hash file of the PDF using the `pdf2john.pl` tool (available in the run directory after compiling from source). This tool allows you to obtain the hash

(Read meta information) of the file through this perl script, which can be extracted into a new file with the following command:

```
pdf2john.pl /root/Desktop/pdf_protected.pdf > /root/Desktop/pdf.h
```

This command will create a `.hash` file in the defined directory. This is the file that we will use to work with JohnTheRipper tool:



The `pdf.hash` file contains a text like:

```
/root/Desktop/pdf_protected.pdf:$pdf$4*4*128*-4*1*16*d22933dd5306
```

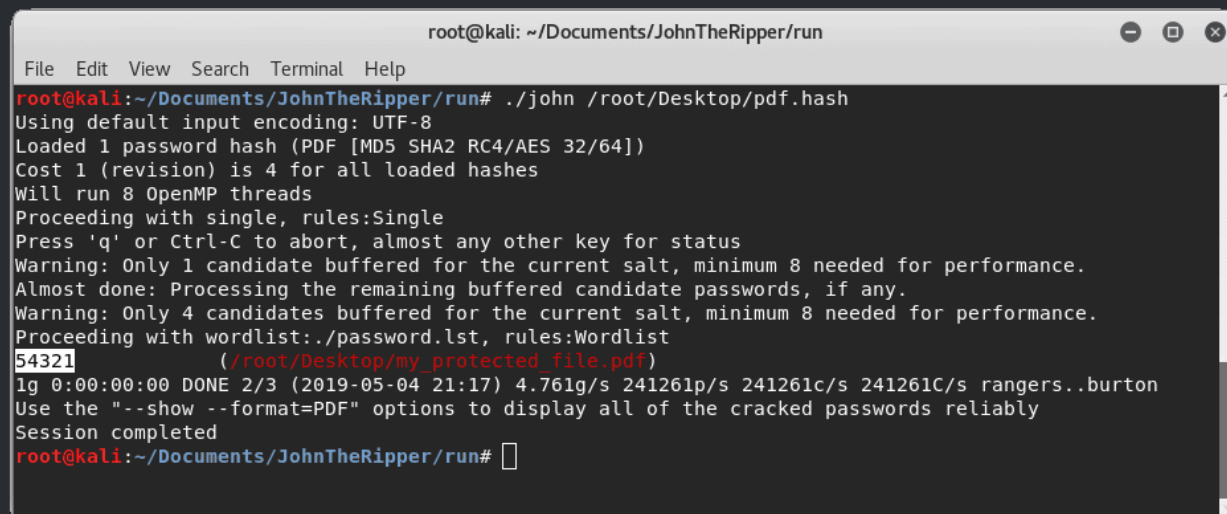
Now that we have the hash file, we can proceed with the brute forcing using the john CLI tool.

### 3. Brute Force with John

Now that we have the .hash file of the PDF with password that we want to unlock, we just need to pass the file as argument to the CLI tool of JohnTheRipper (in the `run` directory):

```
john protected_pdf.hash
```

This will use UTF-8 as the default input encoding and will start to guess the password of the PDF file using the default wordlist of the library. If it's found, it will display the password and the path to the protected PDF:

A terminal window titled 'root@kali: ~/Documents/JohnTheRipper/run' showing the execution of the 'john' command. The output displays various status messages, including the input encoding (UTF-8), the number of hashes loaded (1), and the cost of the hashes (4). It also shows the progress of the brute force attack, including a warning about the number of candidates buffered. The final output shows the password '54321' and the path to the protected PDF file, followed by a completion message.

```
root@kali: ~/Documents/JohnTheRipper/run
File Edit View Search Terminal Help
root@kali:~/Documents/JohnTheRipper/run# ./john /root/Desktop/pdf.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
Cost 1 (revision) is 4 for all loaded hashes
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 1 candidate buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 4 candidates buffered for the current salt, minimum 8 needed for performance.
Proceeding with wordlist:./password.lst, rules:Wordlist
54321 (/root/Desktop/my_protected_file.pdf)
1g 0:00:00:00 DONE 2/3 (2019-05-04 21:17) 4.761g/s 241261p/s 241261c/s 241261C/s rangers..burton
Use the "--show --format=PDF" options to display all of the cracked passwords reliably
Session completed
root@kali:~/Documents/JohnTheRipper/run#
```

If you try to run the command on the same file after the password has been guessed, you will see the following messages: "No password hashes loaded", "No password hashes loaded", or "No password hashes left to crack (see FAQ)". Cracked passwords will be printed to the terminal and saved in the file called `$JOHN/john.pot` (in the documentation and in the configuration file for John, "`$JOHN`" refers to John's "home directory"; which directory it really is depends on how you installed John). The `$JOHN/john.pot` file is also used to not load password hashes that you already cracked when you run John the next time.

If that's the case, you will be able to see the password again of the same file using the `--show` flag:

```
john --show /root/Desktop/pdf.hash
```

So the password will be shown (in our case `54321`):



```
root@kali: ~/Documents/JohnTheRipper/run
File Edit View Search Terminal Help
root@kali:~/Documents/JohnTheRipper/run# ./john --show /root/Desktop/pdf.hash
/root/Desktop/my_protected_file.pdf:54321

1 password hash cracked, 0 left
root@kali:~/Documents/JohnTheRipper/run#
```

## USING A CUSTOM WORD LIST

If you don't want to use the default `password.lst` file of JohnTheRipper, just specify the path to the new file using the `--wordlist` argument:

```
john --wordlist=password.lst protected_pdf.hash
```

As final recommendation, the tool offers to crack a lot of files, so you may want to [read the documentation of the library](#). The rest of documentation is located in separate files, listed here in the recommended order of reading:

- INSTALL - installation instructions
- OPTIONS - command line options and additional utilities

- EXAMPLES - usage examples - strongly recommended
- MODES - cracking modes: what they are
- FAQ - frequently asked questions
- BUGS - list of known bugs
- DYNAMIC - how to use dynamic format in JtR
- DYNAMIC COMPILER FORMATS - List of known hash formats built using the dynamic compiler
- DYNAMIC\_SCRIPTING - how to build/optimize a format that uses dynamic
- README.bash-completion - how to enable bash completion for JtR
- CONTACT (\*) - how to contact the author or otherwise obtain support
- CONFIG (\*) - how to customize
- EXTERNAL (\*) - defining an external mode
- RULES (\*) - wordlist rules syntax
- CHANGES (\*) - history of changes
- CREDITS (\*) - credits
- LICENSE - copyrights and licensing terms
- COPYING - GNU GPL version 2, as referenced by LICENSE above

(\*) most users can safely skip these.

Happy hacking ♥!

@ E-mail

🐦 Tweet

f Like

G +1

📌 Pin it



**Carlos Delgado**



Interested in programming since he was 14 years old, Carlos is the founder and author of most of the articles at Our Code World. Proud Self-taught programmer.

#### THIS COULD INTEREST YOU

#### BECOME A MORE SOCIAL PERSON

##### Our Code World Comment Policy


Our Comments Section is open to every developer, so you can contribute (even code) to the main idea of the Article.


Please read our [Comment Policy](#) before commenting.





What do you think about this article?


3 Responses


 Upvote

 Funny

 Love

 Surprised

 Angry

 Sad

1 Comment

Our Code World

1 Login ▾

Recommend

Tweet

Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?



Name



Islam Sabry • a month ago

when i use the pdf2john part it tells me command not found..

^ | ▾ • Reply • Share ›

Subscribe

Add Disqus to your site

Disqus' Privacy Policy

DISQUS

WHAT IS THIS ALL ABOUT

ADVERTISE WITH US

ADVERTISE WITH US



Our Code World is a free blog about programming, where you will find solutions to simple and complex tasks of your daily life as a developer.

**CONTACT US**

DEV@OURCODEWORLD.COM

**JOIN OUR TEAM!**

DEV@OURCODEWORLD.COM

 FACEBOOK  TWITTER  YOUTUBE

HOME

ALL ARTICLES

CATEGORIES

ADVERTISE WITH US

CONTACT US

ABOUT