When it comes to cracking passwords, there are three types of attacks:

- Brute force: Which attempts to guess the password by sequentially working through every possible letter, number, and special character combination. This is a painfully slow process, but effective.
- Dictionary: This attack leverages a file containing lists of common passwords (usually taken from a breach of some kind) to guess a given password. Can be helpful in CTFs, but nowadays it can be difficult to apply this type of attack in the real world.
- Rainbow table: Rainbow tables are a series of pre-computed hashes. The idea is that these rainbow tables include all hashes for a given algorithm. So instead of cracking the hash/ password/etc. you perform a look up of the hash in the table. Do note that this takes considerable processing power to achieve.

For this article, lets perform a dictionary attack. To do that, first we need a dictionary to attack with. The easiest to acquire is *rockyou.txt*. rockyou.txt is a set of compromised passwords from the social media application developer RockYou. *Note: you can download rockyou.txt.gz from here, if you're not using Kali Linux.*

NOTE/USAGE of JohnTheRipper

```
 To  use  John,  you  just need to supply it a password file and the
desired options. If no
       mode  is  specified,  john  will  try  "single"  first,  then
"wordlist"   and   finally
       "incremental".

       Once  John  finds  a  password,  it  will be printed to the
terminal and saved into a file
       called ~/.john/john.pot. John will read this file when it restarts
so it  doesn't  try  to crack already done passwords. That's why if you
want to retry to crack the same file, it's better to go to "/.john/" and
delete the content inside that file; the "john.pot".
```

Command to bruteforce a hashed password inside a text file with JtR:

```
/usr/sbin/john --wordlist=/usr/share/wordlists/rockyou.txt JeanText-file-which-contains-the-hash-password.txt
```

OR just type into a terminal:

```
john --wordlist=/usr/share/wordlists/rockyou.txt JeanText-file-which-contains-the-hash-password.txt
```

To see the cracked passwords, use

john -show passwd

Important: do this under the same directory where the password was cracked (when using the
cronjob, /var/lib/john), otherwise it won't work.

While cracking, you can press any key for status, or Ctrl+C to abort the session, saving
point information to a file (~/.john/john.rec by default). By the way, if you press
Ctrl+C twice John will abort immediately without saving. The point information is also
saved every 10 minutes (configurable in the configuration file, ~/.john/john.ini or
~/.john/john.conf) in case of a crash.

To continue an interrupted session, run:

john -restore

Now, you may notice that many accounts have a disabled shell, you can make John ignore
these (assume that shell is called /etc/expired):

john -show -shells:-/etc/expired passwd

You might want to mail all the users who got weak passwords, to tell them to change the
passwords. It's not always a good idea though (unfortunately, lots of people seem to
ignore such mail, it can be used as a hint for crackers, etc), but anyway, I'll assume you
know what you're doing. Get a copy of the 'mailer' script supplied with John, so you won't
change anything that's under /usr/sbin ; edit the message it sends, and possibly the mail
command inside it (especially if the password file is from a different box than you got
John running on). Then run:

./mailer passwd

Anyway, you probably should have a look at /usr/share/doc/john/OPTIONS for a list of all
the command line options, and at /usr/share/doc/john/EXAMPLES for more John usage examples
with other cracking modes.

OPTIONS

All the options recognized by john start with a single dash (`-'). A summary of options is included below.

-external:MODE
Enables an external mode, using external functions defined in ~/john.ini's [List.External:MODE] section.

-format:NAME
Allows you to override the ciphertext format detection. Currently, valid format names are DES, BSDI, MD5, BF, AFS, LM. You can use this option when cracking or with '-test'. Note that John can't crack password files with different ciphertext formats at the same time.

-groups:[-]GID[,..]
Tells John to load users of the specified group(s) only.

-incremental[:MODE]
Enables the incremental mode, using the specified ~/john.ini definition (section [Incremental:MODE], or [Incremental:All] by default).

-makechars:FILE
Generates a charset file, based on character frequencies from ~/.john/john.pot, for use with the incremental mode. The entire ~/.john/john.pot will be used for the charset file unless you specify some password files. You can also use an external filter() routine with this option.

-restore[:FILE]
Continues an interrupted cracking session, reading point information from the specified file (~/.john/john.rec by default).

-rules Enables wordlist rules, that are read from [List.Rules:Wordlist] in /etc/john/john.conf (or the alternative configuration file you might specify on the command line).
This option **requires** the **-wordlist** option to be passed as well.

-salts:[-]COUNT
This feature sometimes allows you to achieve better performance. For example you
can crack only some salts using '-salts:2' faster, and then crack the rest using
'-salts:-2'. Total cracking time will be about the same, but you will get some
passwords cracked earlier.

-savemem:LEVEL
You might need this option if you don't have enough memory, or don't want John to
affect other processes too much. Level 1 tells John not to waste memory on login
names, so you won't see them while cracking. Higher levels have a performance
impact: you should probably avoid using them unless John doesn't work or gets into
swap otherwise.

-session:FILE
Allows you to specify another point information file's name to use for this
cracking session. This is useful for running multiple instances of John in
parallel, or just to be able to recover an older session later, not always continue
the latest one.

-shells:[-]SHELL[,..]
This option is useful to load accounts with a valid shell only, or not to load
accounts with a bad shell. You can omit the path before a shell name, so
'-shells:csh' will match both '/bin/csh' and '/usr/bin/csh', while
´-shells:/bin/csh' will only match '/bin/csh'.

-show Shows the cracked passwords in a convenient form. You should also specify the
password files. You can use this option while another John is cracking, to see what
it did so far.

-single
Enables the "single crack" mode, using rules from [List.Rules:Single].

-status[:FILE]
Prints status of an interrupted or running session. To get an up to date status information of a detached running session, send that copy of John a SIGHUP before using this option.

-stdin These are used to enable the wordlist mode (reading from stdin).

-stdout[:LENGTH]
When used with a cracking mode, except for "single crack", makes John print the words it generates to stdout instead of cracking. While applying wordlist rules, the significant password length is assumed to be LENGTH, or unlimited by default.

-test Benchmarks all the enabled ciphertext format crackers, and tests them for correct operation at the same time.
This option does **not** need any file passed as argument. Its only function is to benchmark the system john is running on.

-users:[-]LOGIN|UID[,..]
Allows you to filter a few accounts for cracking, etc. A dash before the list can be used to invert the check (that is, load all the users that aren't listed).

-wordlist:FILE
These are used to enable the wordlist mode, reading words from FILE.

MODES

John can work in the following modes:

Wordlist
John will simply use a file with a list of words that will be checked against the passwords. See RULES for the format of wordlist files.

Single crack
In this mode, john will try to crack the password using the login/GECOS information as passwords.

Incremental
This is the most powerful mode. John will try any character combination to resolve
the password. Details about these modes can be found in the MODES file in john's
documentation, including how to define your own cracking methods.

FILES

/etc/john/john.conf
is where you configure how john will behave.

/etc/john/john-mail.msg
has the message sent to users when their passwords are successfully cracked.

/etc/john/john-mail.conf
is used to configure how john will send messages to users that had their passwords cracked.

SEE ALSO

mailer(8), **unafs**(8), **unique**(8), **unshadow**(8),

The programs and the configuration files are documented fully by John's documentation, which should be available in /usr/share/doc/john or other location, depending on your system.
https://manpages.ubuntu.com/manpages/xenial/man8/john.8.html

SSH keys

To test out JtR's SSH key password cracking prowess, first create a set of new private keys. *Note: JtR isn't cracking the file itself (i.e. the number of bytes in the generated key doesn't*

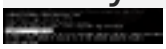
matter), JtR is just cracking the private key's encrypted password.

In this case create the public/private key pair with a predictable password:

```
# Create some private key
ssh-keygen -t rsa -b 4096
# Create encrypted zip
/usr/sbin/ssh2john ~/.ssh/id_rsa > id_rsa.hash
```

Next, all you need to do is point John the Ripper to the given file, with your dictionary:

```
/usr/sbin/john --wordlist=/usr/share/wordlists/
rockyou.txt id_rsa.hash
```



And voila!

Keepass2 database

What about Keepass? If you're not aware, Keepass is an open source, cross-platform, password management vault. For those paranoid individuals who fear storing all their secrets in the cloud (i.e. with LastPass).

So lets create a vault to attack. First, install Keepass CLI ("kpcli").

```
sudo apt-get install -y kpcli
```

Next, create a vault. You don't need to store any passwords in the vault, an empty vault will do.

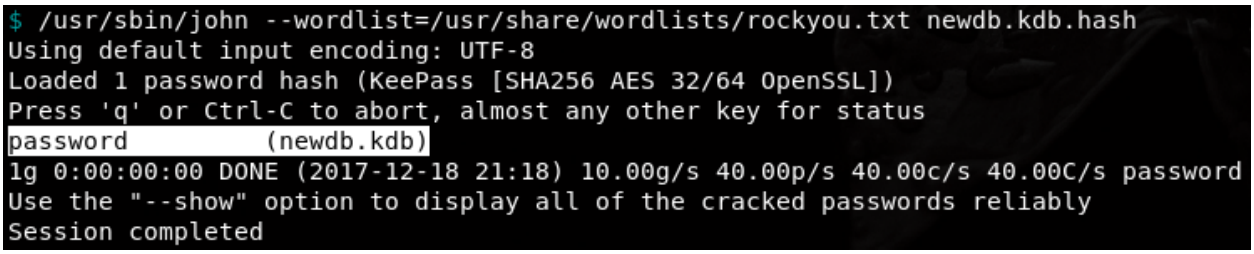
```
$ kpcli
KeePass CLI (kpcli) v3.1 is ready for operation.
Type 'help' for a description of available
commands.
Type 'help <command>' for details on individual
commands.
kpcli:/> saveas newdb.kdb
Please provide the master password:
**************************
Retype to verify: **************************
kpcli:/> exit
```

As with attacking both SSH private keys, and Linux password hashes, convert the Keepass database to a JtR compatible format.

```
/usr/sbin/keepass2john newdb.kdb >
newdb.kdb.hash
```

And attack!

```
/usr/sbin/john --wordlist=/usr/share/wordlists/
rockyou.txt newdb.kdb.hash
```



RAR

Next, lets go after the **R**oshal **Ar**chive ("RAR") format. To create an encrypted RAR archive file on Linux, perform the following:

```
# Install rar
sudo apt-get install -y rar
# Create some dummy file
echo "Hello" > anything.txt

Now follow the 3 steps below:

1) Create an encrypted RAR file with the
password "blablabla"
rar a -hpblablabla encrypted.rar anything.txt
```

2) Next, lets convert it to JtR's cracking format:

```
/usr/sbin/rar2john encrypted.rar >
encryptedBla.hash

OR, just type in a terminal: rar2john
encrypted.rar > encryptedBla.hash
```

3) And fire away!

"john --wordlist=/usr/share/wordlists/ rockyou.txt encryptedBla.hash", without quotes.

That’s it!!!