John the ripper :

The basic syntax of John the Ripper commands is as follows. We will cover the specific options and modifiers used as we use them.

john [options] [path to file]

john - Invokes the John the Ripper program

[path to file] - The file containing the hash you're trying to crack, if it's in the same directory you won't need to name a path, just the file.

Automatic Cracking

John has built-in features to detect what type of hash it's being given, and to select appropriate rules and formats to crack it for you, this isn't always the best idea as it can be unreliable- but if you can't identify what hash type you're working with and just want to try cracking it, it can be a good option! To do this we use the following syntax:

john --wordlist=[path to wordlist] [path to file]

--wordlist= - Specifies using wordlist mode, reading from the file that you supply in the following path...

[path to wordlist] - The path to the wordlist you're using, as described in the previous task.

**Example Usage:**

john --wordlist=/usr/share/wordlists/rockyou.txt hash\_to\_crack.txt

#### Identifying Hashes

Sometimes John won't play nicely with automatically recognising and loading hashes, that's okay! We're able to use other tools to identify the hash, and then set john to use a specific format. There are multiple ways to do this, such as using an online hash identifier like [this](https://hashes.com/en/tools/hash_identifier) one. I like to use a tool called [hash-identifier](https://gitlab.com/kalilinux/packages/hash-identifier/-/tree/kali/master), a Python tool that is super easy to use and will tell you what different types of hashes the one you enter is likely to be, giving you more options if the first one fails.

To use hash-identifier, you can just pull the python file from gitlab using:wget https://gitlab.com/kalilinux/packages/hash-identifier/-/raw/kali/master/hash-id.py.

Then simply launch it with python3 hash-id.py and then enter the hash you're trying to identify- and it will give you possible formats!

#### Format-Specific Cracking

Once you have identified the hash that you're dealing with, you can tell john to use it while cracking the provided hash using the following syntax:

john --format=[format] --wordlist=[path to wordlist] [path to file]

--format= - This is the flag to tell John that you're giving it a hash of a specific format, and to use the following format to crack it

[format] - The format that the hash is in

**Example Usage:**

john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txthash\_to\_crack.txt

**A Note on Formats:**

When you are telling john to use formats, if you're dealing with a standard hash type, e.g. md5 as in the example above, you have to prefix it withraw- to tell john you're just dealing with a standard hash type, though this doesn't always apply. To check if you need to add the prefix or not, you can list all of John's formats using john --list=formats and either check manually, or grep for your hash type using something like john --list=formats | grep -iF "md5".

#### Unshadowing

John can be very particular about the formats it needs data in to be able to work with it, for this reason- in order to crack /etc/shadow passwords, you must combine it with the /etc/passwd file in order for John to understand the data it's being given. To do this, we use a tool built into the John suite of tools called unshadow. The basic syntax of unshadow is as follows:

unshadow [path to passwd] [path to shadow]

unshadow - Invokes the unshadow tool

[path to passwd] - The file that contains the copy of the /etc/passwd file you've taken from the target machine

[path to shadow] - The file that contains the copy of the /etc/shadow file you've taken from the target machine

**Example Usage:**

unshadow local\_passwd local\_shadow > unshadowed.txt

**Note on the files**

When using unshadow, you can either use the entire /etc/passwd and /etc/shadow file- if you have them available, or you can use the relevant line from each, for example:

**FILE 1 - local\_passwd**

Contains the /etc/passwd line for the root user:

root:x:0:0::/root:/bin/bash

**FILE 2 - local\_shadow**

Contains the /etc/shadow line for the root user:

root:$6$2nwjN454g.dv4HN/$m9Z/r2xVfweYVkrr.v5Ft8Ws3/YYksfNwq96UL1FX0OJjY1L6l.DS3KEVsZ9rOVLB/ldTeEL/OIhJZ4GMFMGA0:18576::::::

#### Cracking

We're then able to feed the output from unshadow, in our example use case called "unshadowed.txt" directly into John. We should not need to specify a mode here as we have made the input specifically for John, however in some cases you will need to specify the format as we have done previously using: --format=sha512crypt

john --wordlist=/usr/share/wordlists/rockyou.txt --format=sha512crypt unshadowed.txt

#### Single Crack Mode

So far we've been using John's wordlist mode to deal with brute forcing simple., and not so simple hashes. But John also has another mode, called Single Crack mode. In this mode, John uses only the information provided in the username, to try and work out possible passwords heuristically, by slightly changing the letters and numbers contained within the username

Zip2john :

![Graphical user interface, text, application, email

Description automatically generated]()

rar2john:

![Graphical user interface, text, application, email

Description automatically generated]()

ssh2john:

#### SSH2John

Who could have guessed it, another conversion tool? Well, that's what working with John is all about. As the name suggests ssh2john converts the id\_rsa private key that you use to login to the SSH session into hash format that john can work with. Jokes aside, it's another beautiful example of John's versatility. The syntax is about what you'd expect. Note that if you don't have ssh2john installed, you can use ssh2john.py, which is located in the /opt/john/ssh2john.py. If you're doing this, replace the ssh2john command with python3 /opt/ssh2john.py or on Kali, python /usr/share/john/ssh2john.py.

ssh2john [id\_rsa private key file] > [output file]

ssh2john - Invokes the ssh2john tool

[id\_rsa private key file] - The path to the id\_rsa file you wish to get the hash of

> - This is the output director, we're using this to send the output from this file to the...

[output file] - This is the file that will store the output from

**Example Usage**

ssh2john id\_rsa > id\_rsa\_hash.txt