How to Install and Use Hashcat to Decrypt MD5? (Tutorial)

infosecscout.com/hashcat-tutorial-decrypt-md5

Patrick Fromaget

Decrypting MD5 hashes is probably a goal for most of you on this website, and hashcat seems a pretty good tool to help you in this task.
The problem is that it's not the easier software to understand and use, whatever your operating system is.
In this tutorial, I will explain everything and show you effective ways to use hashcat.

Hashcat is one of the best password recovery tool, available for free on almost any operating system.
It can use several methods to find the clear password corresponding to an encrypted hash. It also supports most hash formats.

Today, the goal is firstly to introduce Hashcat and explain all the technical language around it, then to install it on your system, and finally give you a few examples on the most effective way to use Hashcat to decrypt MD5 encrypted passwords.

Download now

Hashcat glossary

What is Hashcat?

Hashcat is a free software for password recovery.
Basically, security companies and hackers are the ones using this tool.

In a database, passwords are generally encrypted with a specific algorithm, for example MD5, SHA1, etc.
These poor cryptographic hashing methods are fast to use, so even if they are not reversible, **a powerful computer can attempt several billions words each second**, and short passwords can be recovered quickly.
If you want to learn more about this, you can check this article: Why MD5 is not secure?

In this tutorial, you'll install & use hashcat on your computer, so you can really understand how it works.

Hashes

A "hash" is a word that you'll see many times in this tutorial, so just in case: **a password hash is the result of a hashing function (ex: MD5)**.
When you encrypt a password with MD5, it gives a hexadecimal string with 32 characters, whatever the word was.

For example, **the corresponding MD5 hash for "MD5Online" is "d49019c7a78cdaac54250ac56d0eda8a"**.
You can use this MD5 encryption tool to try this if you want.

The goal with hashcat will be to give it the hash as an input, and see if it can recover the correct word (MD5Online).

Dictionary

Hide your IP address and location with a free VPN:
Try it for free now, with advanced security features.
2900+ servers in 65 countries. It's free. Forever.
Even if we know that strong passwords are long and including all cases, numbers and special characters, most people still use some pretty basic words, just to comply with the website requirements.

Currently, passwords like "123456", "qwerty" or simply "password" are still the most used ones online.
So, when you have a list of password hashes to recover, a good method is often to start with a list of the most common passwords to save time with hashcat.

This kind of passwords list is named a dictionary.
You can find some dictionaries files easily on the Internet (we'll see a few examples later), or create one based on the information you already have.

So, creating a dictionary file and asking hashcat to compare it to your hashes list is the first method you can try.

Attack modes

But hashcat can do much more of that, many attack modes are supported by this tool, and we'll see a few of them today.
Here are the three main ones:

- **Dictionary attack**: you already know what it is
- **Brute-force attack**: in brute-force mode, hashcat will try all combinations in a given charset (for example, "try all combinations of 8 letters words with digits only")
- **Mask attack**: almost the same thing as brute-force, but with a dynamic charset (ex: "all combinations for words between 6 and 10 characters, with lowercase letters, starting with an uppercase)

Most of the time, you will try all of them, of combine them in one command.

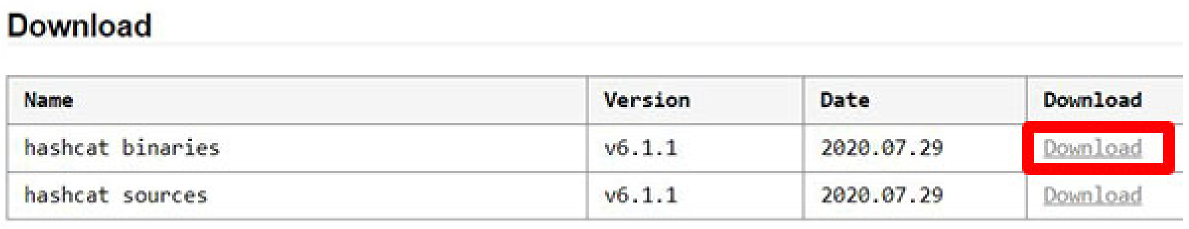
Install Hashcat on your computer

I hope the theory is a bit clearer for you now, so we can move to the practice, and install Hashcat on your computer.
It's available for any operating system, so it should not be so complicated.

Windows

For Windows, the easiest way is to download the binary file on the official website:

- **Go to the Hashcat website** here
- **Click on Download** in the "hashcat binaries" line



Download

Name	Version	Date	Download
hashcat binaries	v6.1.1	2020.07.29	Download
hashcat sources	v6.1.1	2020.07.29	Download

- You'll get a compressed file, probably a .7z
- **Extract all the files** with WinRAR or 7zip

Remember the files location, we'll need it later.

Linux

On Linux, you can follow the same steps as for Windows, but the best way is probably to use your package manager.
This way the command will be perfectly integrated, and you'll automatically install new updates.

For example, **if you are on a Debian-like system, you can use**:
`sudo apt install hashcat`

That's it, the hashcat command is now available anywhere in your system.
It's easy to use on Linux, I like that. The only problem might be to get the correct drivers for your graphic card (Hashcat is more powerful with a good GPU than a high-end CPU).

If you are using Ubuntu, I have a complete guide on how to install Hashcat on Ubuntu here.

macOS

On macOS, I think the best way is to use Homebrew to install everything easily:

- **Install Homebrew** if you don't have it yet. In a terminal, paste the following command:
  `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"`
- Once installed, you can **install hashcat with:**
  `brew install hashcat`

That's it!
More detail here about Homebrew if you are not (yet) familiar with it.

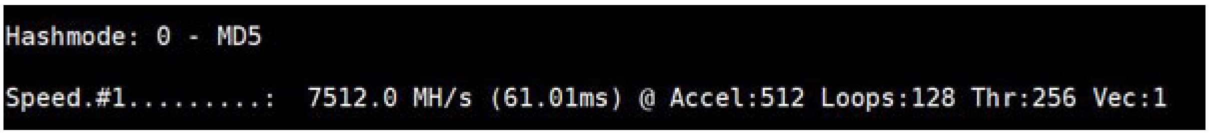
Tests

Once installed, you can start a short test to check that everything is ok, before we start to really use it in the next part.
Hashcat includes a benchmark mode that you can run with no data:

- **Windows:**
  - Open a command prompt
  - Go to the hashcat extracted folder, probably something like:
    `cd Downloads\hahscat-<version>\`
  - Run hashcat:
    `hashcat.exe -b`
    On older versions, you may have the choice between 32 and 64 bits:
    `hashcat64.exe -b`
    The .exe extension is not mandatory, I will skip it in the following part, to make the command work also on Linux / macOS.

- **Linux / macOS:**
  You can use hashcat anywhere, so just open a terminal and type the same command:
  `hashcat -b`

You should get something like this for each hash mode you try:



```
Hashmode: 0 - MD5

Speed.#1.........:  7512.0 MH/s (61.01ms) @ Accel:512 Loops:128 Thr:256 Vec:1
```

If you have any error, try to fix it now.
You can check the Hashcat wiki and forums, or just Google the error 🙂

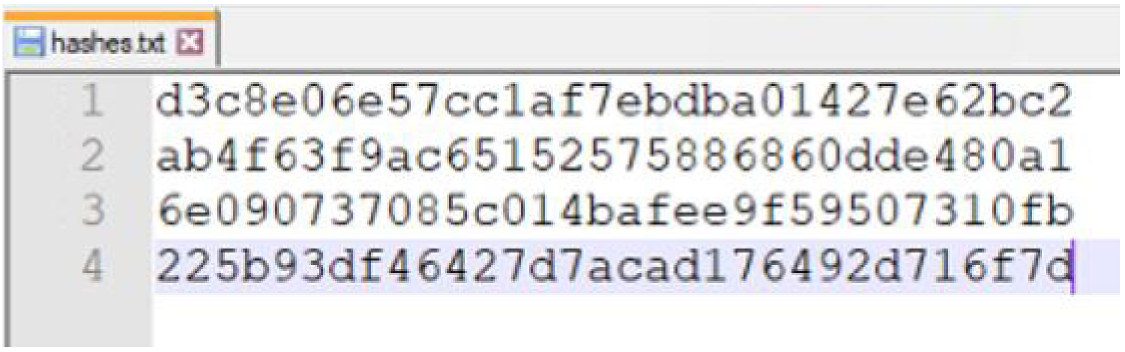
How to use Hashcat

Let's see now how it works with a concrete example.

Create your hashes list

The first thing you need to do, is to create a file with all the MD5 hashes inside (or with another hash function, but I will show you with the MD5 algorithm).

- **Create a folder for hashcat**, or use the one created after the download if you took the binaries.
- **Create a new empty file in it: hashes.txt** for example.
- **Put one hash on each line**, no spaces, no separators, nothing.
  Here is my file content if you want an example:
  ```
  d3c8e06e57cc1af7ebdba01427e62bc2
  ab4f63f9ac65152575886860dde480a1
  6e090737085c014bafee9f59507310fb
  225b93df46427d7acad176492d716f7d
  ```
  And as you can see, I don't put anything except the hashes, one per line:



```
d3c8e06e57cc1af7ebdba01427e62bc2
ab4f63f9ac65152575886860dde480a1
6e090737085c014bafee9f59507310fb
225b93df46427d7acad176492d716f7d
```

- **Save the file and exit** the editor.

Dictionary attack

The first thing we'll try is to use a word list.
It's the same kind of file, but with one word (not hash) per line.

You can either create a file from scratch with the passwords you think you'll find, or download one with the most common passwords in it.

Here are a few links you can try to find good dictionaries:

- Daniel Miessler GitHub
- Crackstation
- You can also simply pick a list of words, like the most used words in English here

Whatever your method, try to have a file with at least the most common password, and ideally have one of them in your hashes list to check that hashcat is working 🙂

In my case, I took this one for the test.
Once you have downloaded or created the file, use the following command with hashcat:
```
hashcat -a 0 -m 0 -O hashes.txt wordlist.txt
```
Obviously, you need to change the hashes and wordlist name, or add the path if not in the same folder as your prompt.

Here are the options I used:
-a 0 : to tell hashcat to use a dictionary attack
-m 0 : to tell hashcat to use only the MD5 function
-O : use the optimized version

And the log I got:

```
>hashcat.exe -a 0 -m 0 -O hashes.txt wordlist.txt
hashcat (v6.1.1) starting…
OpenCL API (OpenCL 2.1 AMD-APP (3004.8)) - Platform #1 [Advanced Micro Devices,
Inc.]
Device #1: Tahiti, 3008/3072 MB (2393 MB allocatable), 28MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 31
Hashes: 4 digests; 4 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
Applicable optimizers applied:
Optimized-Kernel
Zero-Byte
Precompute-Init
Meet-In-The-Middle
Early-Skip
Not-Salted
Not-Iterated
Single-Salt
Raw-Hash
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 555 MB
Dictionary cache built:
Filename..: wordlist.txt
Passwords.: 100000
Bytes…..: 781896
Keyspace..: 100000
Runtime…: 0 secs
The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework
Approaching final keyspace - workload adjusted.
```

ab4f63f9ac65152575886860dde480a1:azerty

```
Session……….: hashcat
Status………..: Exhausted
Hash.Name……..: MD5
Hash.Target……: hashes.txt
Time.Started…..: Thu Sep 17 06:04:05 2020 (0 secs)
Time.Estimated…: Thu Sep 17 06:04:05 2020 (0 secs)
Guess.Base…….: File (wordlist.txt)
Guess.Queue……: 1/1 (100.00%)
Speed.#1………: 5532.1 kH/s (0.10ms) @ Accel:1024 Loops:1 Thr:64 Vec:1
Recovered……..: 1/4 (25.00%) Digests
Progress………: 100000/100000 (100.00%)
Rejected………: 0/100000 (0.00%)
Restore.Point….: 100000/100000 (100.00%)
Restore.Sub.#1…: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1….: 123456 -> 070162
Hardware.Mon.#1..: Temp: 34c Fan: 33% Util: 45% Core:1000MHz Mem:1250MHz Bus:16
Started: Thu Sep 17 06:03:59 2020
Stopped: Thu Sep 17 06:04:05 2020
```

As you can see in the middle, hashcat recover one password in my list ("azerty"), and added it to the hashcat.potfile file (so you can see all recovered passwords in it, even if there are many results).

That's it for the dictionary attack, obviously you will use more or less big files depending on your goals, but you already know the important steps.

Bruteforce attack

The next method I want to show you is the brute force method.
It's the base of the most powerful one you can dream with hashcat (we'll see it just after this one).
The idea is to try a huge quantity of possible words, in a short period of time.

Basically, hashcat will start from the smallest word and increase the password length gradually.
As it's rarely the better way to find a password, the hashcat developers consider it as outdated, **the mask attack we'll see after that replaces it**.

Master Ethical Hacking Skills!
Join the Complete Ethical Hacking Course Bundle and step into the world of cybersecurity.
Learn to think like a hacker and protect systems with this comprehensive course.
But if you want to give a try for the exercise, here is how to do this:
`hashcat -a 3 -m 0 hashes.txt`
You should already know most of the options here.
Just "-a 3" is the brute force attack mode (the dictionary mode was 0).

When you run this command, hashcat will display the status regularly in your console, so you can see what it's checking at the moment, for example:
`Candidates.#1….: 0qw6bx -> Xqqfqx`
Here I know it's already testing 6 characters passwords after a few seconds run on a standard computer.

Mask attack

In this method, rather than trying every possible word, we'll limit the attempts in a predefined charset and password length.

A charset is defined by the characters allowed in your password.
By default, you can already use these possibilities:

- **?l =** abcdefghijklmnopqrstuvwxyz
- **?u =** ABCDEFGHIJKLMNOPQRSTUVWXYZ
- **?d =** 0123456789
- **?h =** 0123456789abcdef
- **?H =** 0123456789ABCDEF
- **?s =** «space»!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

- **?a =** ?l?u?d?s
- **?b =** 0x00 – 0xff

The other thing you can play with, is the password length.
To indicates the password length option, you'll generally repeat the charset character.
Here are a few examples :

- **?l?l?l?l?l?l?l?l:** passwords between "aaaaaaaa" and "zzzzzzzz"
- **password?d:** passwords between "password0" and "password9"
- **?u?l?l:** passwords between "Aaa" and "Zzz"

I think you understood the idea here 🙂

It's awesome. That's why, we don't use a simple brute force method anymore, as most of the time, you already have an idea of the passwords format.
If your passwords come from a list, where passwords shorter than 8 characters are forbidden, why would you try them?

Once you know these rules, **here is the command to use, it's almost the same as seen previously, with the password format added:**
`hashcat -a 3 -m 0 hashes.txt ?l?l?d?l?l?l?l?l?l?`

This command is trying all passwords with 9 characters, lower case, with a digit in third position (I'm just cheating to find one of my other word quickly ^^).
And here is the result I expected:
`d3c8e06e57cc1af7ebdba01427e62bc2:md5online`

You can now play with this attack as you want.
You can go further and create files with several masks in it, you can see some examples in the "masks" subfolder, and replace the password format by the filename in the command, for example:
`hashcat -a 3 -m 0 hashes.txt masks\rockyou-1-60.hcmask`
This file is inspired from the cracked passwords formats from RockYou.

Conclusion

That's it, you now know the basics of password decryption with hashcat.
Hashcat is a compelling tool, with many options to do exactly what you want.
Now that you understand the basic, you can start playing with it, and check the official documentation if you need something more specific.

In any case, I hope this tutorial was useful for you. If it's the case, please share it on your favorite social network or forum, it helps me a lot 🙂

Whenever you're ready for more security, here are things you should think about:

- **Break free from Gmail**: You should be able to choose what happens to your data. With Proton, only you can read your emails. Get private email.