

# Proof of concepts :

## Recherche de collision de Hash

[Prérequis](#)

[Procédure](#)

[1\) Récupération du format](#)

[Name-that-hash](#)

[2\) Brute Force](#)

[2.1\) Hashcat](#)

[2.2\) John the ripper](#)

[Remédiation](#)

[Autres](#)

[Outils en ligne](#)

[Hashes.com](#)

[Crackstation.net](#)

## Prérequis

name-that-hash

→ pip3 install name-that-hash

Hashcat

→ apt install hashcat

JohnTheRipper

→ apt install john

Mot choisi pour le test :

test

→ md5 : 098f6bcd4621d373cade4e832627b4f6

# Procédure

## 1) Récupération du format

Name-that-hash

```
[*]$ nth --text 32ed8707730d74150e77279c0bde07ca

https://twitter.com/bee_sec_san
https://github.com/HashPals/Name-That-Hash

32ed8707730d74150e77279c0bde07ca

Most Likely
MD5, HC: 0 JtR: raw-md5 Summary: Used for Linux Shadow files.
MD4, HC: 900 JtR: raw-md4
NTLM, HC: 1000 JtR: nt Summary: Often used in Windows Active Directory.
Domain Cached Credentials, HC: 1100 JtR: mscach

Least Likely
Domain Cached Credentials 2, HC: 2100 JtR: mscach2 Double MD5, HC: 2600 Tiger-128, Skein
```

Maintenant nous savons que ce hash est surement en MD5.

Name-That-Hash nous donne directement le flag a utiliser (HC: 0), sinon :

hashcat -h

## 2) Brute Force

### 2.1) Hashcat

Hashcat fonctionne de la façon suivante :

hashcat -m [ModelID] [HASH] [Wordlist]

- -m : définit le mode que l'on veut utiliser, 0 pour MD5, 1000 pour du NTLM etc...
- HASH : le hash que l'on as en amont. Peut êtres contenue dans un .txt
- Wordlist : liste de mots que l'on veut utiliser pendant le brute force.

```
[*]$ hashcat -m 0 "098f6bcd4621d373cade4e832627b4f6" Documents/rockyou.txt
```

```

098f6bcd4621d373cade4e832627b4f6:test

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
Hash.Target.....: 098f6bcd4621d373cade4e832627b4f6
Time.Started.....: Wed Feb 25 08:25:21 2026 (0 secs)
Time.Estimated...: Wed Feb 25 08:25:21 2026 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (Documents/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#2.....: 6492.3 kH/s (0.13ms) @ Accel:512 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 167936/14344385 (1.17%)
Rejected.....: 0/167936 (0.00%)
Restore.Point....: 165888/14344385 (1.16%)
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#2....: tyson4 -> playpen

Started: Wed Feb 25 08:25:20 2026
Stopped: Wed Feb 25 08:25:23 2026

```

On retrouve bien notre “test”, on peut voir que pour un mot aussi court, hashcat a pris 3 secondes.

## 2.2) John the ripper

John the ripper est plus “intelligent” et peut souvent détecter lui-même le format du hash. Il fonctionne de la façon suivante :

`john --wordlist=[wordlist] [hashfile] (--format=[format])`

- wordlist : liste de mots que l’on veut utiliser pendant le brute force.
- hashfile : John a besoin que le hash soit stocker dans un fichier txt.

Si il ne trouve pas le type de hash que vous lui avez donner, ajouter : `--format=[format]`

- format : type de hash que vous lui donnez.

```

└─ [★]$ echo "098f6bcd4621d373cade4e832627b4f6" > hash.txt

└─ [★]$ cat hash.txt
098f6bcd4621d373cade4e832627b4f6

└─ [★]$ john --format=raw-md5 --wordlist=Documents/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
test (???)
1g 0:00:00:00 DONE (2026-02-25 08:59) 50.00g/s 8313Kp/s 8313Kc/s 8313KC/s tyson4..tauruz
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.

```

# Remédiation

## 1) Utilisez des versions modernes (argon2, bcrypt...)

Ces algorithmes sont plus complexes, nécessitent plus de puissance pour être calculés et de ce fait, sont plus chronophage lors des brutes force.

## 2) Ajouter "salt" et "pepper"

Ajoutez une valeur aléatoire au hash existant, ainsi, 2 utilisateurs ayant un même mot de passe auront un hash différent.

Salt : valeur ajoutée directement au hash

Pepper : clé secrète stocker ailleurs

## 3) MFA

Evidemment. Même si le mot de passe venait à être cassé, l'utilisation de facteurs multiples compliquerait grandement l'authentification.

# Autres

## Outils en ligne

[Hashes.com](https://hashes.com)

Hashes.com

[Home](#) [FAQ](#) [Deposit to Escrow](#) [Purchase Credits](#) [API](#) [Tools](#) [Decrypt Hashes](#) [Escrow](#) [Support](#) [English](#) [Register](#) [Login](#)

Decrypt MD5, SHA1, MySQL, NTLM, SHA256, MD5 Email, SHA256 Email, SHA512 hashes

Enter your hashes here and we will attempt to decrypt them for free online.

Hashes (max. 25 separated by newline, format 'hash[:salt]') [\(QEscrow\)](#)

098f6bcd4621d373cade4e832627b4f6

☐ Show plains and salts in hex format

☐ Show algorithm of finds



Capcha Check

SUBMIT & SEARCH

Hashes.com

[Home](#) [FAQ](#) [Deposit to Escrow](#) [Purchase Credits](#) [API](#) [Tools](#) [Decrypt Hashes](#) [Escrow](#) [Support](#) [English](#) [Register](#) [Login](#)

▲ Proceeded!

1 hashes were checked: 1 found 0 not found

✓ Found:

098f6bcd4621d373cade4e832627b4f6:test

SEARCH AGAIN

HASHES.COM

Support

API

DECRYPT HASHES

Free Search

Mass Search

Reverse Email MD5

TOOLS

Hash Identifier

Hash Verifier

Email Extractor

\*2john Hash Extractor

Hash Generator

File Parser

List Matching

List Management

Base64 Encoder

ESCROW

View jobs

Upload new list

Manage your lists

CrackStation

Defuse.ca · Twitter

CrackStation

Password Hashing Security

Defuse Security

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

098f6bcd4621d373cade4e832627b4f6

I'm not a robot

This site is exceeding 3000000 requests per day

hCaptcha

Privacy · Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-ha1, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1|sha1\_hex), Qubsv3.1BackupDefaults

[Download CrackStation's Wordlist](#)

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).

CrackStation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 15GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries [here](#), and the lookup table implementation (PHP and C) is available [here](#).

CrackStation

Defuse.ca · Twitter

CrackStation

Password Hashing Security

Defuse Security

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

098f6bcd4621d373cade4e832627b4f6

I'm not a robot

This site is exceeding 3000000 requests per day

hCaptcha

Privacy · Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-ha1, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1|sha1\_hex), Qubsv3.1BackupDefaults

| Hash                             | Type | Result |
|----------------------------------|------|--------|
| 098f6bcd4621d373cade4e832627b4f6 | md5  | Text   |

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

[Download CrackStation's Wordlist](#)

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).