# Hashcat Password Cracking  [Total Headers : 3]

## Hash Modes

| Algorithm | Hashcat Mode | Notes |
|---|---|---|
| MD5 | 0 | Most basic & fast hash |
| SHA1 | 100 | Still common, less secure |
| SHA2-256 | 1400 | More secure than SHA1 |
| SHA2-512 | 1700 | Part of the SHA family |
| SHA3-224 | 17300 | Keccak-based algorithm |
| SHA3-256 | 17400 | Used in newer cryptography |
| SHA3-384 | 17500 | Used in your question |
| SHA3-512 | 17600 | Very secure, slow |
| MD4 | 900 | Rare today, used in NTLM |
| NTLM | 1000 | MD4 variant used in Windows |
| bcrypt | 3200 | Very slow, used in web apps |
| PBKDF2-HMAC-SHA1 | 12000 | Used in WPA, Office |
| PBKDF2-HMAC-SHA256 | 10900 | More secure than SHA1 version |
| WPA-PBKDF2-PMKID+EAPOL | 22000 | Used for Wi-Fi password cracking |

```
 9400 | MS Office 2007
 9500 | MS Office 2010
 9600 | MS Office 2013
25300 | MS Office 2016 - SheetProtection
 9700 | MS Office ≤ 2003 $0/$1, MD5 + RC4
 9710 | MS Office ≤ 2003 $0/$1, MD5 + RC4, collider #1
 9720 | MS Office ≤ 2003 $0/$1, MD5 + RC4, collider #2
 9810 | MS Office ≤ 2003 $3, SHA1 + RC4, collider #1
 9820 | MS Office ≤ 2003 $3, SHA1 + RC4, collider #2
 9800 | MS Office ≤ 2003 $3/$4, SHA1 + RC4
```

```
┌──(kali㊉kali)-[~]
└─$ hashid '8968e60ae465a52918e33412b32d1
9818e18072637d9ce65749c68878716f01a'
Analyzing '8968e60ae465a52918e33412b32d19
818e18072637d9ce65749c68878716f01a'
[+] Snefru-256
[+] SHA-256
[+] RIPEMD-256
[+] Haval-256
[+] GOST R 34.11-94
[+] GOST CryptoPro S-Box
[+] SHA3-256
[+] Skein-256
[+] Skein-512(256)
```

## Mask Modes

| Symbol | Meaning | Example |
|--------|---------|---------|
| ?d | Digit (0–9) | ?d?d?d?d = 4-digit PIN |
| ?l | Lowercase letter (a–z) | ?l?l?l = 3 lowercase letters |
| ?u | Uppercase letter (A–Z) | ?u?u = 2 uppercase letters |
| ?s | Special char (!, @, #) | ?s = 1 special character |
| ?a | All printable ASCII | Combo of everything above |
| ?h | Hex lowercase (0–f) | Used in hex-based hashes |
| ?H | Hex uppercase (0–F) | Uppercase hex format |

## Common Attack Modes

| Mode | Name | Description |
|------|------|-------------|
| 0 | Dictionary | Tries each word in the wordlist |
| 1 | Combination | Joins words from 2 wordlists |
| 3 | Brute-force | Tries every character combination |
| 6 | Hybrid [ Wordlist + Mask ] | Wordlist on the left, mask on the right |
| 7 | Hybrid [ Mask + Wordlist ] | Mask on the left, wordlist on the right |

### Command Layout

**hashcat [options] -m <hash-mode> -a <attack-mode> <hash-file> <wordlist> <mask>**   ←Syntax
**hashcat --show .hash.**   ←Hash Type

**sudo apt install wordlists**   ←Install
**sudo gunzip /usr/share/wordlists/rockyou.txt.gz**
**ls -lh /usr/share/wordlists/rockyou.txt**

1. hashcat -m 17600 -a 7 hash.txt ?d?d?d?d rockyou.txt **# ls for digit+list**   ←Examples

2. hashcat -m 17600 -a 6 --session sha3_session --status
   --status-timer 10 --force -o cracked_passwords.txt hash.txt
   /usr/share/wordlists/rockyou.txt ?d?d?d?d **# ls for list + digit**

```bash
bash                                          Copy    Edit

hashcat -m 0 -a 0 -r /usr/share/hashcat/rules/best64.rule hashes.txt wordlist.txt
```

This command:

- Uses `rockyou.txt` as the base wordlist.
- Applies the 64 most effective rules from `best64.rule`.
- Tries many variations like:
    - `password` → `Password`, `Password1`, `P@ssword`, `password123`, etc.

```
(root㉿kali)-[/home/aditverma400]
# hashcat custom_dict.txt -r /usr/share/hashcat/rules/best64.rule --stdout >> custom_dict.txt

(root㉿kali)-[/home/aditverma400]
# wc -l custom_dict.txt

161382 custom_dict.txt
```

**[Options]**

| Option | Description | Example |
|---|---|---|
| hashcat -h | Lists out all hashcat help commands and modes | hashcat -h |
| -o <output-file> | Specify output file for cracked passwords. | -o cracked_passwords.txt |
| --session <session-name> | Name the session for saving and resuming later. | --session my_session |
| --status | Display real-time status (progress, speed, cracked passwords, etc.). | --status |
| --status-timer <seconds> | Set the interval (in seconds) for status updates. | --status-timer 10 |
| --force | Force Hashcat to run even with warnings or minor issues. | --force |
| -w <workload-profile> | Set the workload profile (speed/efficiency). Options: 1 (low), 2 (default), 3 (high), 4 (insane). | -w 3 |

| | | |
|---|---|---|
| --potfile-path <path> | Specify the location of the potfile (password storage). | --potfile-path /path/to/potfile |
| --remove | Automatically remove cracked hashes from the hash file after they're cracked. | --remove |
| -r <rule-file> | Apply a rule file to modify wordlist entries. | -r usr/share/hashcat/rules/best64.rule |
| --max-combinations <number> | Limit the total number of combinations Hashcat will try in brute-force or hybrid mode. | --max-combinations 1000000 |
| --skip <number> | Skip a specific number of hashes from the hash file before starting the crack. | --skip 1000 |
| --restore | Resume a session that was previously stopped or interrupted. | --restore |
| --benchmark | Run a benchmark to test the performance of Hashcat on the current hardware. | --benchmark |
| --stdout | Output the results of the attack to the terminal instead of a file. | --stdout |
| -u <username> | Specify a username if cracking hashes that involve a user context (e.g., SMB, HTTP). | -u username |
| --hash-info | Show detailed information about the hash file before starting the crack. | --hash-info |
| --debug-mode <level> | Enable debugging output for troubleshooting. | --debug-mode 1 |
| -t <attack-time-limit> | Limit the attack to a specific number of seconds. | -t 600 (10 minutes) |
| -b | Perform benchmark using only the selected algorithm. | -b |
| --quiet | Suppress output to only show essential information. | --quiet |

```
┌──(kali㊇kali)-[~]
└─$ hashcat -m 17600 -a 1 hash.txt /usr/share/wordlists/rockyou.txt a.txt
#rockyou_word + Wordlist2_word interchanging the sequence of the .txt file
will give Wordlist2_word + rockyou_word

hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 1
8.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
═══════════════════════════════════════════════════════════════════════

* Device #1: cpu-skylake-avx512-AMD Ryzen AI 9 HX 370 w/ Radeon 890M, 6924
/13913 MB (2048 MB allocatable), 24MCU

/usr/share/hashcat/OpenCL/m17600_a1-optimized.cl: Pure kernel not found, f
alling back to optimized kernel
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 31

INFO: All hashes found as potfile and/or empty entries! Use --show to disp
lay them.

Started: Mon Apr 21 20:55:18 2025
Stopped: Mon Apr 21 20:55:18 2025

┌──(kali㊇kali)-[~]
└─$ hashcat -m 17600 --show hash.txt

d550390367149fbfbf93f554aa4efba0ced10b66b814ed646298360e0293b9e4fe6bc014c9
4cc45b0f2f10aca1617f92db1c3f236dc6f9a1d7d1774393b89807:$HEX[50415353574f52
442132333421643a6e78]

┌──(kali㊇kali)-[~]
└─$ echo 50415353574f52442132333421643a6e78 | xxd -r -p

PASSWORD!234!d:nx
```

echo 50415353574f52442132333421643a6e78
echo is a shell command that prints text to the terminal.
50415353574f52442132333421643a6e78 is a hexadecimal string representing ASCII characters.
This part outputs the hex string.

| (pipe)
The pipe takes the output from the echo command and passes it as input to the next command (xxd).

xxd -r -p
xxd is a utility for creating hex dumps or reversing them.
-r tells it to reverse the process (hex to ASCII or binary).
-p indicates plain hex format without offsets or formatting.
This part decodes the hex string into its ASCII representation.

**1. Hashcat Attack Modes**
Hashcat supports **6 core attack modes**, each suited for specific scenarios depending on the available information about the password or system.

---

**1.1 Straight Mode (Attack Mode 0)**
**Description:**
Applies every word from a wordlist (dictionary) directly to the hash.
**Scenario:**
You have a list of common passwords or leaked credentials.

**Example:**
hashcat -m 0 -a 0 hashes.txt rockyou.txt
- -m 0: Hash type MD5
- -a 0: Straight mode
- rockyou.txt: Dictionary of common passwords

---

**1.2 Combination Mode (Attack Mode 1)**
**Description:**
Concatenates two wordlists (wordlist1 + wordlist2) and tries every possible combination.
**Scenario:**
The password might be composed of two common words like summer + 2020.

**Example:**
hashcat -m 0 -a 1 hashes.txt wordlist1.txt wordlist2.txt

---

**1.3 Brute-Force Mode (Attack Mode 3)**
**Description:**
Tries all possible character combinations of a given length or pattern.
**Scenario:**
You know the password is a 6-digit number or fixed format like Abcd1234.

**Example 1:** 6-digit PIN
hashcat -m 0 -a 3 hashes.txt ?d?d?d?d?d?d

**Example 2:** Alphanumeric
hashcat -m 0 -a 3 hashes.txt ?l?l?l?l?d?d?d?d
- ?d = digit, ?l = lowercase, ?u = uppercase, ?s = special char

---

**1.4 Hybrid Attack – Wordlist + Mask (Attack Mode 6)**
**Description:**
Appends a mask (like 123) to words in a wordlist.
**Scenario:**
The password is a common word followed by a birth year (e.g., password1995).

**Example:**
hashcat -m 0 -a 6 hashes.txt rockyou.txt ?d?d?d?d

**1.5 Hybrid Attack – Mask + Wordlist (Attack Mode 7)**
**Description:**
Prepends a mask to every word in a wordlist.
**Scenario:**
The password begins with a fixed pattern like 123 followed by a common word (e.g., 123summer).

**Example:**
hashcat -m 0 -a 7 hashes.txt ?d?d?drockyou.txt

---

**1.6 Rule-based Attack (Modifier, not a mode)**
**Description:**
Applies transformation rules (e.g., capitalizing, reversing) to words from a dictionary.
**Scenario:**
Users modify common passwords slightly (e.g., Password → Password1, password!).

**Example:**
hashcat -m 0 -a 0 -r rules/best64.rule hashes.txt rockyou.txt

# John Hash Extractor

```
git clone https://github.com/openwall/john.git
cd john/run
office2john.py   >> Helps extract password hashes from the documents
python3 office2john.py   <Document.extension > = Password hash
```

# Hydra Usage

https://www.stationx.net/how-to-use-hydra/?utm_source=chatgpt.com : Hydra Usage online

You can use the CLI or GUI