

Command Structure & Parameters

Complete reference for hashcat command-line syntax, flags, and arguments.

General Syntax

Every hashcat invocation follows the same pattern:

```
hashcat [options...] hash_file [dictionary|mask|dictionary+mask]
```

The binary name varies by platform:

- **Linux / macOS:** `hashcat` (or `./hashcat.bin` if running from the extracted directory)
- **Windows:** `hashcat.exe`

Options (flags) can appear in any order before or after the hash file, but the positional arguments — hash file, wordlists, and masks — must appear in the correct sequence after all flags.

Hash Type: `-m`

The `-m` parameter specifies the hash algorithm hashcat should use for cracking. Using the wrong mode is the most common beginner mistake — hashcat will run but find nothing because it is interpreting the hash bytes incorrectly.

Common Hash Modes

Mode	Algorithm	Typical Source
<code>0</code>	MD5	Web application databases, legacy systems
<code>100</code>	SHA-1	Older web applications, Git commits
<code>1000</code>	NTLM	Windows SAM database, Active Directory

Mode	Algorithm	Typical Source
1800	sha512crypt (\$6\$)	Linux /etc/shadow
2500	WPA/WPA2 (legacy)	Wi-Fi capture files
3200	bcrypt (\$2*\$)	Modern web applications
13100	Kerberos TGS-REP (RC4)	Kerberoasting attacks
22000	WPA-PBKDF2-PMKID+EAPOL	Modern Wi-Fi auditing

Tip: Use `hashcat --example-hashes` to see sample hashes for every supported mode. This helps you identify unknown hash types by visual comparison. There are over 350 supported modes.

Attack Mode: `-a`

The `-a` parameter selects the attack strategy. It determines what kind of arguments (wordlists, masks, or both) must follow the hash file.

Mode	Name	Required Arguments	Description
0	Dictionary (Straight)	wordlist	Tests every word from a wordlist against the hashes. The most common starting point.
1	Combinator	wordlist1 wordlist2	Combines every word from list 1 with every word from list 2. Good for compound passwords like "sunflower123".
3	Mask (Brute-force)	mask pattern	Generates candidates based on character position masks. Full brute-force when using <code>?a</code> (all printable).
6	Hybrid: Dictionary +	wordlist mask	Appends mask-generated characters to each dictionary word. Example: <code>password</code>

Mode	Name	Required Arguments	Description
	Mask		+ <code>?d?d</code> = <code>password00</code> through <code>password99</code> .
7	Hybrid: Mask + Dictionary	mask wordlist	Prepends mask-generated characters to each dictionary word. Example: <code>?d?d</code> + <code>password</code> = <code>00password</code> through <code>99password</code> .

Mask Charsets

In mask mode (`-a 3`), each position in the mask is defined by a charset placeholder:

Placeholder	Characters	Count
<code>?l</code>	Lowercase letters (a-z)	26
<code>?u</code>	Uppercase letters (A-Z)	26
<code>?d</code>	Digits (0-9)	10
<code>?s</code>	Special characters (!@#\$... etc.)	33
<code>?a</code>	All printable ASCII (?!u?d?s)	95
<code>?b</code>	All bytes (0x00-0xff)	256

Output Control

Output File: `-o`

By default, hashcat displays cracked hashes on screen and stores them in `hashcat.potfile`. Use `-o` to write results to a specific file:

```
hashcat -m 0 -a 0 -o cracked.txt hashes.txt rockyou.txt
```

Output Format: `--outfile-format`

Controls what gets written to the output file. The value is a bitmask — add the numbers together to combine fields:

Value	Content	Example Output
1	Hash only	5f4dcc3b5aa765d61d8327deb882cf99
2	Plain password only	password
3	Hash:password (default)	5f4dcc3b...cf99:password
4	Hex-encoded plain	70617373776f7264
5	Hash:hex-plain	5f4dcc3b...cf99:70617373776f7264

Rules: `-r`

Rules transform dictionary words on the fly, dramatically expanding the effective wordlist without increasing file size. Each rule defines a mutation (capitalize, reverse, append digits, leet-speak substitution, etc.).

```
# Single rule file
hashcat -m 0 -a 0 -r rules/best64.rule hashes.txt rockyou.txt

# Multiple rule files (chained — every combination is applied)
hashcat -m 0 -a 0 -r rules/best64.rule -r rules/toggles1.rule hashes.txt rockyou.txt
```

Included rule files: Hashcat ships with several rule files in the `rules/` directory. Popular choices: `best64.rule` (64 most effective rules), `rockyou-30000.rule` (30k rules extracted from the RockYou breach), `d3ad0ne.rule` (34k+ aggressive rules).

Incremental Mode: `-i`

By default, a mask attack only tries candidates matching the exact mask length. The `-i` (or `--increment`) flag tells hashcat to iterate through increasing lengths.

```
# Without increment: only tries 8-character combinations
hashcat -m 0 -a 3 hashes.txt ?a?a?a?a?a?a?a

# With increment: tries 1-char, 2-char, ..., up to 8-char
hashcat -m 0 -a 3 -i hashes.txt ?a?a?a?a?a?a?a?a

# With min/max bounds: tries 4-char through 8-char only
hashcat -m 0 -a 3 -i --increment-min 4 --increment-max 8 hashes.txt ?a?a?a?a?a?
```

Warning: Incremental mode with `?a` (all printable) generates enormous keyspaces. An 8-character `?a` mask alone produces $95^8 = 6.6$ quadrillion candidates. Adding lengths 4 through 7 adds another ~7.4 billion on top. Always set `--increment-min` to skip trivially short passwords.

Workload Profile: `-w`

The workload profile controls how aggressively hashcat uses the GPU. Higher values yield faster cracking but may cause system unresponsiveness.

Level	Name	Use Case
1	Low	Desktop use — system remains responsive
2	Default	General purpose
3	High	Dedicated cracking rig, headless server
4	Nightmare	Maximum throughput; may freeze GUI/display

Device Selection: `-D`

By default, hashcat auto-detects available compute devices. Use `-D` to restrict to specific device types:

- `-D 1` — CPU only
- `-D 2` — GPU only
- `-D 1,2` — Both CPU and GPU

Use `-I` (or `--backend-info`) to list all detected devices and their capabilities before choosing.

Session Management

Long-running attacks can be paused and resumed using sessions.

```
# Start a named session
hashcat --session audit_q1 -m 1000 -a 0 hashes.txt rockyou.txt

# Pause: press 'c' during execution (checkpoint), or 'q' to quit gracefully

# Resume the session later
hashcat --session audit_q1 --restore
```

Session data is stored in `*.restore` and `*.induct` files in the hashcat working directory. Without `--session`, hashcat uses a default session name and will refuse to start a second instance.

Complete Command Examples

Example 1: Simple MD5 dictionary attack

```
hashcat -m 0 -a 0 -o results.txt hashes.txt rockyou.txt
```

Cracks MD5 hashes using the RockYou wordlist. Results written as hash:password pairs to `results.txt`.

Example 2: NTLM mask attack with incremental mode

```
hashcat -m 1000 -a 3 -o results.txt --session mysession \
--increment --increment-min 4 --increment-max 8 \
-w 3 hashes.txt ?a?a?a?a?a?a?a
```

Cracks NTLM hashes using a brute-force mask attack. Tries all printable character combinations from 4 to 8 characters. Uses high workload on GPU. Named session allows pause/resume.

Example 3: Dictionary + rules attack on SHA-512

```
hashcat -m 1800 -a 0 -r rules/best64.rule -o cracked.txt \
-w 2 shadow_hashes.txt rockyou.txt
```

Cracks Linux shadow file hashes (sha512crypt) using a dictionary with the best64 rule file applied to each word.

Example 4: Hybrid dictionary + mask

```
hashcat -m 0 -a 6 -o results.txt hashes.txt rockyou.txt ?d?d?d?d
```

Appends 4 digits to every word in the dictionary. Catches patterns like `password2024` or `admin1234`.

Example 5: Combinator attack

```
hashcat -m 0 -a 1 -o results.txt hashes.txt words1.txt words2.txt
```

Concatenates every word from `words1.txt` with every word from `words2.txt`. Effective for compound passwords.

Quick Reference Card

Flag	Purpose	Common Values
<code>-m</code>	Hash type	0 (MD5), 1000 (NTLM), 1800 (SHA-512), 3200 (bcrypt)
<code>-a</code>	Attack mode	0 (dict), 1 (combo), 3 (mask), 6 (hybrid d+m), 7 (hybrid m+d)
<code>-o</code>	Output file	Any file path
<code>--outfile-format</code>	Output content	1 (hash), 2 (plain), 3 (hash:plain)
<code>-r</code>	Rules file	rules/best64.rule, rules/rockyou-30000.rule

Flag	Purpose	Common Values
<code>-i</code>	Incremental mode	(no value — it is a toggle)
<code>--increment-min</code>	Min length for increment	1–N
<code>--increment-max</code>	Max length for increment	1–N
<code>-w</code>	Workload profile	1 (low), 2 (default), 3 (high), 4 (nightmare)
<code>-D</code>	Device type	1 (CPU), 2 (GPU), 1,2 (both)
<code>--session</code>	Session name	Any string
<code>--restore</code>	Resume session	(no value — it is a toggle)

Further Reading

- [Hashcat Wiki](#) — official documentation and hash mode reference
- [Mask Attack Reference](#) — detailed mask charset documentation
- [Rule-Based Attack Reference](#) — rule syntax and examples
- `hashcat --help` — the built-in help is comprehensive and always up to date