

Theory: Mask & Hybrid Attacks

Pattern-based password cracking: when you know the structure but not the exact characters.

Introduction

Dictionary attacks are powerful when users choose common words, but many passwords follow predictable structural patterns: a 4-digit PIN, a capitalized name followed by digits, a word with appended special characters. Mask attacks let you describe these patterns precisely and exhaustively test every combination that matches.

Hybrid attacks combine the best of both worlds: the real-world vocabulary of a wordlist with the systematic character generation of masks. Together, these two modes cover a massive portion of real-world passwords that pure dictionary attacks miss.

Mask Attack Fundamentals (`-a 3`)

A mask attack (attack mode 3) generates password candidates based on a positional character pattern. Each position in the mask specifies which characters are allowed at that position. Hashcat then generates every possible combination.

The mask is read left to right. Each `?x` placeholder represents one character position. The total keyspace is the product of the number of characters in each position's charset.

Built-in Charset Placeholders

Placeholder	Charset	Size	Description
<code>?l</code>	<code>abcdefghijklmnopqrstuvwxyz</code>	26	Lowercase Latin letters
<code>?u</code>	<code>ABCDEFGHIJKLMNOPQRSTUVWXYZ</code>	26	Uppercase Latin letters

Placeholder	Charset	Size	Description
<code>?d</code>	<code>0123456789</code>	10	Decimal digits
<code>?s</code>	<code>!"#\$%&'()*+,-./:;<=>? @[\\]^_`{ }~</code>	33	Special / printable symbols
<code>?a</code>	<code>?l?u?d?s</code>	95	All printable ASCII
<code>?b</code>	<code>0x00 - 0xff</code>	256	Full byte range

Mask Examples

Here are common patterns and how to express them as masks:

- `?d?d?d?d` — 4-digit PIN (0000–9999). Keyspace: 10,000.
- `?u?l?l?l?l?d?d` — Capitalized 4-letter name + 2 digits (e.g., Alex42). Keyspace: 45,697,600.
- `?a?a?a?a` — Any 4 printable characters. Keyspace: 81,450,625.
- `?u?l?l?l?l?l?s?d?d` — Name + special + 2 digits (e.g., Summer!23). Keyspace: ~25 billion.

Basic Command

```
# Crack MD5 hashes using a 4-digit PIN mask
hashcat -m 0 -a 3 -o results.txt hashes.txt ?d?d?d?d
```

Incremental Mode (`-i`)

Often you don't know the exact password length. Incremental mode tells Hashcat to start at a shorter mask and gradually increase the length up to the full mask you provide.

```
# Try lengths 4 through 8 using all printable characters
hashcat -m 0 -a 3 -i --increment-min 4 --increment-max 8 \
hashes.txt ?a?a?a?a?a?a?a
```

The process works as follows:

1. Hashcat starts at length `--increment-min` (4 in this example)
2. It runs the mask truncated to that length: `?a?a?a?a`
3. When that keyspace is exhausted, it increments to length 5: `?a?a?a?a?a`
4. This continues until `--increment-max` (8) is reached

Keyspace explosion: With `?a` (95 chars per position), each additional character multiplies the keyspace by 95. Length 4 = 81 million, length 5 = 7.7 billion, length 6 = 735 billion, length 7 = 69 trillion, length 8 = 6.6 quadrillion. Plan your time and resources accordingly.

If you omit `--increment-min`, Hashcat defaults to starting at length 1. If you omit `--increment-max`, it defaults to the length of your mask.

Custom Charsets

The built-in placeholders are convenient but sometimes too broad or too narrow.

Hashcat supports up to four custom charset definitions using the flags `-1`, `-2`, `-3`, and `-4`.

```
# Define custom charset 1 as lowercase + digits
# Then use ?1 in the mask
hashcat -m 0 -a 3 -1 ?1?d hashes.txt ?1?1?1?1?1?1

# Two custom charsets: ?1 = lowercase+digits, ?2 = uppercase+special
hashcat -m 0 -a 3 -1 ?1?d -2 ?u?s hashes.txt ?2?1?1?1?1?d

# Literal characters in a custom charset
hashcat -m 0 -a 3 -1 aeiou hashes.txt ?1?1?1?1?1
```

Custom charsets let you model real-world password policies precisely. For example, if a policy requires at least one uppercase letter, you can put `?u` in specific positions and `?1?d` in the rest.

Combining charsets: A custom charset definition can include both built-in placeholders and literal characters. For example, `-1 ?1?d_.` creates a charset

containing all lowercase letters, all digits, underscore, hyphen, and period.

Hybrid Attacks

Hybrid attacks combine a wordlist with a mask. They are extremely effective because users commonly take a dictionary word and append or prepend numbers, years, or symbols to meet password requirements.

Mode 6: Wordlist + Mask (Append)

Attack mode 6 takes each word from the dictionary and appends every combination generated by the mask.

```
# Append 3 digits to each word in rockyou.txt
hashcat -m 0 -a 6 hashes.txt rockyou.txt ?d?d?d

# Example: "password" becomes password000, password001, ... password999
# Example: "summer" becomes summer000, summer001, ... summer999
```

The total keyspace is: (number of words in dictionary) x (mask keyspace). For rockyou.txt (~14 million words) with `?d?d?d` (1,000), that is 14 billion candidates.

Mode 7: Mask + Wordlist (Prepend)

Attack mode 7 is the reverse: the mask generates a prefix that is prepended to each dictionary word.

```
# Prepend 3 digits before each word in rockyou.txt
hashcat -m 0 -a 7 hashes.txt ?d?d?d rockyou.txt

# Example: "password" becomes 000password, 001password, ... 999password
```

Argument order matters. In `-a 6`, the wordlist comes first, then the mask. In `-a 7`, the mask comes first, then the wordlist. This matches the order in which they appear in the generated candidate: left-to-right.

Common Hybrid Patterns

Pattern	Mode	Command Fragment
word + 2 digits	<code>-a 6</code>	<code>rockyou.txt ?d?d</code>
word + year	<code>-a 6</code>	<code>rockyou.txt 20?d?d</code>
word + special + digit	<code>-a 6</code>	<code>rockyou.txt ?s?d</code>
digit + word	<code>-a 7</code>	<code>?d rockyou.txt</code>
year + word	<code>-a 7</code>	<code>20?d?d rockyou.txt</code>

Choosing Between Attack Modes

A practical cracking session typically chains multiple attack modes in order of efficiency:

1. **Dictionary attack** (`-a 0`) with rules — catches the most common passwords quickly
2. **Hybrid append** (`-a 6`) — catches "word + digits" patterns
3. **Hybrid prepend** (`-a 7`) — catches "digits + word" patterns
4. **Targeted mask** (`-a 3`) — specific known patterns (PINs, date formats)
5. **Incremental mask** (`-a 3 -i`) — broad search across multiple lengths, last resort

Real-world tip: Before running a mask attack, check the password policy of the target system. If passwords must be 8+ characters with at least one uppercase, one digit, and one special character, you can craft a mask that enforces those constraints and dramatically reduce the keyspace.

Keyspace Calculation

Understanding keyspace helps you estimate how long an attack will take. The formula is straightforward:

$\text{Keyspace} = (\text{chars in position 1}) \times (\text{chars in position 2}) \times \dots \times (\text{chars in position } N)$

At a speed of 10 billion hashes per second (typical for MD5 on a modern GPU):

Mask	Keyspace	Time at 10 GH/s
<code>?d?d?d?d</code>	10,000	Instant
<code>?d?d?d?d?d?d</code>	1,000,000	Instant
<code>?1?1?1?1?1?1</code>	308,915,776	~0.03 seconds
<code>?a?a?a?a?a?a</code>	735,091,890,625	~73 seconds
<code>?a?a?a?a?a?a?a?a</code>	6,634,204,312,890,625	~7.7 days

Further Reading

- [Hashcat Wiki — Mask Attack](#)
- [Hashcat Wiki — Hybrid Attack](#)
- [Hashcat Wiki — Custom Charsets](#)