

Dictionary & Combinator Attacks

The most practical and widely-used hashcat attack modes. Dictionary attacks leverage real-world wordlists, while combinator attacks multiply their power by joining words from two lists.

Learning Objectives

- Run a **dictionary attack** (`-a 0`) using popular wordlists
 - Apply **rule-based mutations** to expand wordlist coverage
 - Execute a **combinator attack** (`-a 1`) joining two dictionaries
 - Identify and obtain **popular wordlists** for security auditing
 - Understand when to choose dictionary vs. combinator vs. other attack modes
-

Dictionary Attack (`-a 0`)

The dictionary (or "straight") attack tries every word in a wordlist file against the target hashes. It is attack mode `0` and is hashcat's default.

```
# Basic dictionary attack against MD5 hashes  
hashcat -m 0 -a 0 -o results.txt hashes.txt rockyou.txt
```

Parameter Breakdown

- `-m 0` — hash type: MD5
- `-a 0` — attack mode: dictionary (straight)
- `-o results.txt` — output cracked results to file
- `hashes.txt` — file containing target hashes
- `rockyou.txt` — the wordlist to use

Speed: Dictionary attacks are fast because they only check words that real humans actually use as passwords. With a GPU, hashcat can check billions of MD5 candidates per second.

Rule-Based Attacks

Rules transform each dictionary word into multiple candidates — capitalizing, appending numbers, substituting characters, reversing, and more. This dramatically increases coverage without a larger wordlist.

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

How Rules Work

For each word in the dictionary, hashcat applies every rule in the file. If `rockyou.txt` has 14 million words and `best64.rule` has 64 rules, hashcat generates **~900 million candidates**.

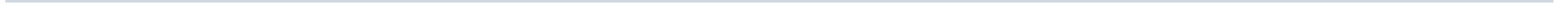
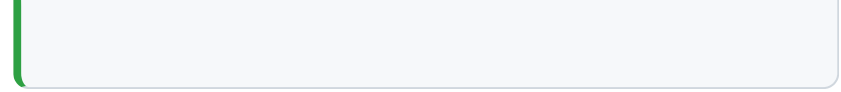
Example Mutations

- `password` → `Password` (capitalize)
- `password` → `password1` (append digit)
- `password` → `p@ssword` (leet-speak)

Built-in Rule Files

- `best64.rule` — 64 high-yield rules
- `d3ad0ne.rule` — ~35K rules
- `dive.rule` — ~99K rules

- `password` → `drowssap` (reverse)



Combinator Attack (`-a 1`)

The combinator attack concatenates every word from the first dictionary with every word from the second dictionary. This catches multi-word passwords like `solarwind` or `bluedragon`.

```
# Combinator attack: dict1 x dict2
hashcat -m 0 -a 1 hashes.txt dict1.txt dict2.txt
```

Candidate Generation

dict1.txt	dict2.txt	Candidates Generated
sun	flower	sunflower
sun	light	sunlight
blue	flower	blueflower
blue	light	bluelight

Candidate count: If dict1 has N words and dict2 has M words, the combinator produces $N \times M$ candidates. $10,000 \times 10,000 = 100$ million candidates.

Popular Wordlists

The quality of a dictionary attack depends entirely on the wordlist. These are the most widely used in the security community.

Wordlist	Size	Source	Notes
rockyou.txt	~14M passwords	2009 RockYou breach	De facto standard; ships with Kali Linux
SecLists	Varies	Daniel Miessler (GitHub)	Curated collection of many wordlists
CrackStation	~1.5B entries	crackstation.net	Massive; 15 GB uncompressed
Custom lists	Varies	You create them	Target-specific: company names, keywords

Tip: Always start with `rockyou.txt` + `best64.rule`. This combination cracks a surprising percentage of weak passwords in minutes.

Attack Strategy Flowchart

Choose your attack mode based on what you know about the target passwords.

Decision Flow

1. **Start simple:** Dictionary attack with `rockyou.txt` (`-a 0`)
2. **Add mutations:** Re-run with `-r rules/best64.rule`
3. **Try combinations:** Combinator attack (`-a 1`) with themed word lists
4. **Escalate rules:** Use `d3ad0ne.rule` or `dive.rule` for deeper coverage
5. **Go broader:** Larger wordlists (CrackStation) or mask/hybrid attacks (Lesson 06)

Real-world stat: Dictionary attacks with rules typically crack 60-80% of password hashes from typical database breaches. Most users pick common words with predictable modifications.

Live Demo

Cracking two well-known MD5 hashes with a dictionary attack.

Demo Steps

1. Create `hashes.txt` with two MD5 hashes:

```
5f4dcc3b5aa765d61d8327deb882cf99  
e10adc3949ba59abbe56e057f20f883e
```

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

3. Check `results.txt` — should find **"password"** and **"123456"**

4. Re-run with rules: `hashcat -m 0 -a 0 -r rules/best64.rule hashes.txt rockyou.txt`

Note: These hashes are intentionally trivial — they exist in every wordlist. In real audits, the easy cracks come first and harder passwords require rules and combinator attacks.

Lab & Homework

Lab 2 — Dictionary Attack

1. Download `rockyou.txt` wordlist
2. Create `hashes.txt` with two known MD5 hashes
3. Run a straight dictionary attack and verify results
4. Re-run with `best64.rule` and compare performance

Homework Tasks

1. Benchmark dictionary vs. rules vs. combinator on provided hashes
2. Create a custom wordlist from a topic and test it
3. Research and document the rule syntax for 5 custom rules

