

Hash Types & Identification

Recognizing hash formats, choosing the correct hashcat mode, and understanding the performance implications of different algorithms.

Learning Objectives

- Identify the **most common hash types** by length, format, and prefix
 - Map each hash type to the correct **hashcat -m mode**
 - Use **hashid**, **hash-identifier**, and **hashcat --example-hashes** for identification
 - Explain the difference between **salted and unsalted** hashes
 - Understand **performance differences** between fast and slow hash algorithms
-

Common Hash Types at a Glance

Algorithm	Mode (-m)	Length	Example Prefix
MD5	0	32 hex chars	5d41402abc4b...
SHA-1	100	40 hex chars	aaf4c61ddcc5...
SHA-256	1400	64 hex chars	2cf24dba5fb0...
SHA-512	1800	128 hex chars	\$6\$rounds=5000\$...
NTLM	1000	32 hex chars	a4f49c406510...
bcrypt	3200	60 chars	\$2a\$12\$... or \$2b\$...
WPA/WPA2	22000	varies	WPA*02*...

Key insight: Hash length alone can narrow possibilities dramatically. 32 hex chars = MD5 or NTLM. 40 hex chars = almost certainly SHA-1.

Identifying Hashes by Format

Many hash formats include **prefixes** that make identification trivial. Look for these patterns before reaching for any tool.

Signature Prefixes

- `$2a$` / `$2b$` / `$2y$` — bcrypt (mode 3200). The number after the second `$` is the cost factor.
- `$6$` — SHA-512crypt (mode 1800). Common in Linux `/etc/shadow`.
- `$5$` — SHA-256crypt (mode 7400).
- `$1$` — MD5crypt (mode 500). Legacy Linux password hash.
- `$apr1$` — Apache MD5 (mode 1600).
- `WPA*02*` — WPA/WPA2 PMKID or handshake (mode 22000).

When there is no prefix and the hash is pure hex, count the characters to determine the algorithm family.

Identification Tools

When manual inspection is not enough, use dedicated tools to narrow down the hash type.

hashid

```
# Install
pip install hashid

# Identify a hash
hashid '5d41402abc4b2a76b9719d911017c592'
# Output: [+] MD5, [+] MD4, [+] NTLM ...
```

hash-identifier

```
# Included in Kali Linux, or install from GitHub
hash-identifier
# Paste the hash when prompted — it suggests possible algorithms
```

hashcat --example-hashes

```
# Search for a specific pattern  
hashcat --example-hashes | grep -i 'bcrypt' -B2 -A3
```

```
# List all modes with example hashes  
hashcat --example-hashes | less
```

Pro tip: `hashcat --example-hashes` is the most authoritative source. It shows the exact format hashcat expects for each mode number.

Salted vs Unsalted Hashes

A **salt** is random data appended or prepended to the password before hashing. It ensures identical passwords produce different hashes.

Why Salts Matter

- **Unsalted:** `MD5("password")` always produces the same hash. Vulnerable to rainbow tables and precomputed lookups.
- **Salted:** `MD5(salt + "password")` produces a unique hash per salt value. Rainbow tables become useless.

How Hashcat Handles Salts

```
# Unsalted MD5 (-m 0): just the hash
5d41402abc4b2a76b9719d911017c592

# Salted MD5 (-m 10): hash:salt format
67b3dba8bc6e4f2a9e1b7c23f8d14e5a:s4lt

# SHA-512crypt (-m 1800): salt embedded in the hash string
$6$rounds=5000$saltsalt$hashvalue...
```

Important: You must supply the salt in the correct format, or hashcat will fail silently. Always check `--example-hashes` for the expected format of salted modes.

Performance: Fast vs Slow Hashes

Hash algorithm choice has an **enormous** impact on cracking speed. This is the single biggest factor in attack feasibility.

Algorithm	Approximate Speed (RTX 4090)	Category
MD5	~164 GH/s	Fast
NTLM	~300 GH/s	Fast
SHA-1	~26 GH/s	Fast
SHA-256	~10 GH/s	Medium
SHA-512crypt	~2.4 MH/s	Slow
bcrypt (cost 12)	~105 kH/s	Very Slow
WPA/WPA2	~1.6 MH/s	Slow

Takeaway: MD5 and NTLM are "fast hashes" — billions of guesses per second. bcrypt is deliberately slow — purpose-built to resist brute-force attacks. Adjust your wordlists and strategy accordingly.

Choosing Your Attack Strategy by Hash Type

The hash type dictates the feasible attack space. Plan your strategy accordingly.

Fast Hashes (MD5, NTLM, SHA-1)

- Full brute-force feasible up to ~8 chars
- Large wordlists + complex rules are practical
- Combinator and hybrid attacks are fast
- Mask attacks with full charset work well

Slow Hashes (bcrypt, SHA-512crypt, WPA)

- Brute-force beyond 6 chars is impractical
- Use small, targeted wordlists
- Focus on common passwords and patterns
- Rules should be minimal and precise

Notes: Always benchmark with `hashcat -b -m MODE` before committing to a long attack. Know your speed before choosing your strategy.

Lab & Homework

Lab Exercises

1. **Hash Identification:** Identify 10 different hashes by type using manual inspection, hashid, and hashcat --example-hashes
2. **Crack MD5 & SHA-1:** Run dictionary attacks on unsalted fast hashes and compare speeds
3. **Crack bcrypt:** Attack a bcrypt hash with a small wordlist and observe the speed difference

Homework Tasks

1. Research and document **5 additional hash types** not covered in class, with their hashcat modes
2. Build a **hash identification cheat sheet** covering length, prefix, and recommended attack strategy
3. Benchmark **3 different hash types** on your hardware and calculate maximum brute-force keyspace

