

Theory

Reading time: ~15 minutes · Prerequisites: Basic command-line familiarity

1. What Is Hashcat?

Hashcat is an open-source password recovery tool that uses modern hash function recognition methods. It is one of the most popular and powerful tools in the cybersecurity field, widely used by security researchers, penetration testers, and system administrators.

The main goal of Hashcat is twofold: to **recover lost passwords** and to **verify the security of existing ones**. By attempting to reverse password hashes back to their plaintext form, security professionals can assess whether an organization's password policies are strong enough to withstand real-world attacks.

How Password Hashing Works

When a user creates a password, the system does not store the password itself. Instead, it runs the password through a **hash function** — a one-way mathematical operation that produces a fixed-length string of characters (the "hash"). For example, the MD5 hash of the word "password" is:

```
5f4dcc3b5aa765d61d8327deb882cf99
```

Hash functions are designed to be **irreversible** — you cannot mathematically derive the original password from the hash. However, you *can* try hashing many possible passwords and comparing the results. This is exactly what Hashcat does, but at extraordinary speed.

Key Insight

Hashcat does not "decrypt" passwords. Hashes are not encrypted data — they are one-way transformations. Hashcat works by generating candidate passwords,

hashing them, and comparing the result against target hashes. When a match is found, the original password is recovered.

2. History and Development of Hashcat

Hashcat was originally created to improve the password recovery process. In its early days, password crackers were slow, limited in the hash types they supported, and ran exclusively on CPUs.

The CPU Era

The earliest versions of Hashcat were CPU-based tools. While functional, CPUs are not well suited for the massively parallel computation that password cracking requires. A typical CPU has 4-16 cores, which limits how many hash computations can be performed simultaneously.

The GPU Revolution

The introduction of GPU acceleration was a turning point. Modern GPUs contain thousands of cores designed for parallel processing. What might take a CPU hours to compute, a GPU can accomplish in minutes or seconds. The GPU-accelerated version, originally called **oclHashcat**, used OpenCL to leverage graphics cards for hash computation.

Unification and Open Source

In 2015, Hashcat became fully open-source. In 2016, with version 3.0, the CPU and GPU versions were merged into a single unified tool. This simplified usage and allowed Hashcat to automatically detect and use the best available hardware.

Development continues actively, with constant updates adding new hash types and optimizing recognition processes. The community contributes new algorithms, performance improvements, and platform support on an ongoing basis.

3. Main Capabilities and Applications

Supported Attack Types

Hashcat supports multiple attack strategies, each effective in different scenarios:

Attack Type	Flag	How It Works	Best For
Dictionary Attack	<code>-a 0</code>	Hashes every word from a wordlist and compares against target	Common passwords, leaked password lists
Combinator Attack	<code>-a 1</code>	Concatenates words from two different dictionaries	Passwords made of two common words
Mask (Brute-Force) Attack	<code>-a 3</code>	Tries all combinations matching a specified pattern	Known password structure (e.g., 8 lowercase letters)
Hybrid Attack (Dict + Mask)	<code>-a 6</code>	Appends mask-generated suffixes to dictionary words	Passwords like "summer2024" or "admin123"
Hybrid Attack (Mask + Dict)	<code>-a 7</code>	Prepends mask-generated prefixes to dictionary words	Passwords like "123admin" or "!password"

Rule-Based Attacks

In addition to the core attack modes, Hashcat supports **rules** — transformation functions applied to dictionary words before hashing. Rules can capitalize letters, append numbers, substitute characters (e.g., "a" to "@"), reverse strings, and much more. A single dictionary of 100,000 words combined with a rule file containing 1,000 rules effectively tests 100 million password candidates.

Cross-Platform Support

Hashcat runs on all major operating systems:

- **Windows** — Pre-compiled binaries available
- **Linux** — Available through package managers or compiled from source
- **macOS** — Supported with Metal backend for Apple Silicon GPUs

Hash Type Support

Hashcat supports over 350 hash types, organized by category. Some of the most commonly encountered types include:

Hash Type	Mode (-m)	Common Use
MD5	0	Legacy web applications, file checksums
SHA-1	100	Legacy systems, some version control
SHA-256	1400	Modern applications, certificates
NTLM	1000	Windows authentication
bcrypt	3200	Modern web applications
WPA/WPA2	22000	Wi-Fi network security

4. Practical Applications

Penetration Testing

During authorized security assessments, penetration testers often extract password hashes from compromised systems. Hashcat is used to recover these passwords, demonstrating the real-world risk of weak password policies to the client.

Password Policy Auditing

Organizations can proactively test their own password databases against common wordlists and attack patterns. If a significant percentage of employee passwords can be cracked within minutes, the password policy needs strengthening.

Forensic Analysis

In digital forensic investigations, encrypted files or password-protected systems may need to be accessed. Hashcat can assist in recovering passwords when authorized by legal process.

Legal and Ethical Warning

Using Hashcat against systems or data you do not own or have explicit written authorization to test is **illegal** in most jurisdictions. Always ensure you have proper authorization before performing any password recovery or security testing activities. This course is for educational purposes and authorized security testing only.

5. Key Concepts Summary

Concept	What It Means	Why It Matters
Hash function	One-way transformation of data to fixed-length string	Passwords are stored as hashes, not plaintext
Dictionary attack	Testing passwords from a pre-built wordlist	Most effective against common/leaked passwords
Mask attack	Brute-forcing all combinations matching a pattern	Covers the entire keyspace for known structures
GPU acceleration	Using graphics card for parallel hash computation	Orders of magnitude faster than CPU-only cracking
Hash type mode (-m)	Numeric identifier for the target hash algorithm	Must match the actual hash format for correct cracking

Further Reading

- [Hashcat FAQ](#) — Official frequently asked questions covering installation, usage, and troubleshooting.
- [Hashcat Wiki](#) — Comprehensive documentation covering all hash types, attack modes, and configuration options.
- [Hashcat Downloads](#) — Official download page for the latest release.
- [Example Hashes](#) — Reference list of example hashes for every supported hash type, useful for testing.

Next lesson: **Lesson 02 — Installation & Setup** — detailed installation instructions for all platforms and hardware configuration.