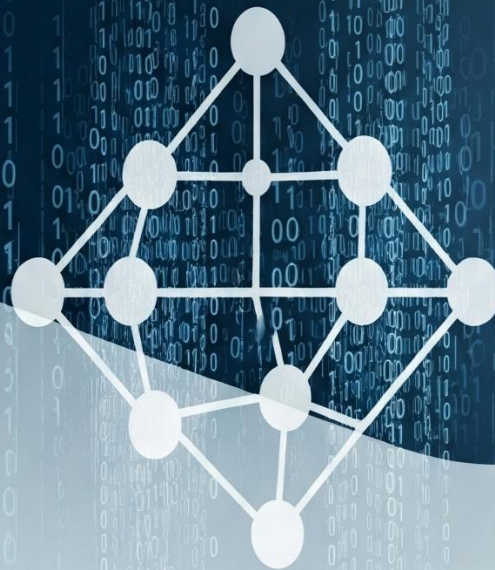


Made by Moez Javed



HASH-IDENTIFIER MANUAL

Moez Javed



Hash-Identifier

Manual

Audience: Undergraduate/graduate students in cybersecurity

1. Introduction — why this matters

Hashes are fixed-length fingerprints of data. They are used everywhere: password storage, digital signatures, file integrity checks, and forensic artefacts. Being able to **identify** a hash type quickly is a core skill for incident response, password auditing, and forensics: it tells you which cracking tools and attack modes to use, whether a hash is salted or not, and which platforms or applications likely produced it.

Learning objectives: - Understand the purpose and limits of hash identification tools. - Install and run hash-identifier and the modern alternative hashid. - Interpret output and map identified hashes to cracking tools (Hashcat / JohnTheRipper). - Practice safe, ethical use in labs and assessments.

2. Prerequisites

- Basic Linux command-line skills (terminal, copy/paste).
- A Kali / Debian / Ubuntu environment (real or VM).
- Python 3 installed (for running the tool from source or using modern replacements).

Safety & Ethics: Only analyze hashes you are authorized to examine. Do not attempt to crack or misuse credentials from systems you do not own or have explicit permission to test.

3. Tools covered in this manual

- hash-identifier — the classic, interactive tool (commonly available on Kali). Ideal for quick interactive identification.

- hashid / hashID — a more modern Python-based tool that supports many more hash types and can print Hashcat modes and John formats. Good for scripting and batch work.
- Short mentions of other tools (name-that-hash, online analyzers) as alternatives.

4. Installation

4.1 Install on Kali (recommended for labs)

```
sudo apt update  
sudo apt install hash-identifier
```

This installs the interactive hash-identifier package that starts a text UI.

4.2 Install hashid (modern alternative — useful for automation)

```
# Option A: pip (quick)  
pip install hashid
```

```
# Option B: clone & run from source (if you prefer the repo)  
git clone https://github.com/psypana/hashid.git  
cd hashid  
# run with python3 ./hashid.py or install with pip from the repo
```

4.3 If a package is not available on your distro

Clone the official repository for hash-identifier (blackploit) and run the script with Python 3 from the repo directory:

```
git clone https://github.com/blackploit/hash-identifier.git  
cd hash-identifier  
# view files; the script file is typically named hash-id.py or similar  
python3 hash-id.py
```

5. Quick — how to run (interactive)

1. Open a terminal.
2. Launch the tool:

hash-identifier

3. At the prompt, paste the hash you found (or type it) and press Enter.
4. The tool prints **Possible Hashs** and **Least Possible Hashs** with human-friendly hints (e.g., MD5, Wordpress, MySQL, NTLM, etc.).

Example: - Paste 5f4dcc3b5aa765d61d8327deb882cf99 → hash-identifier will list MD5 among possible matches.

6. Using the modern hashid tool (non-interactive & scripted)

hashid supports CLI flags to show more metadata and to work with files.

Basic usage:

```
# simple single-hash check
```

```
hashid '5f4dcc3b5aa765d61d8327deb882cf99'
```

```
(kali㉿kali)-[~]
└─$ hashid '5f4dcc3b5aa765d61d8327deb882cf99'
Analyzing '5f4dcc3b5aa765d61d8327deb882cf99'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
```

```
# with hashcat mode and john format
```

```
hashid -m -j '5f4dcc3b5aa765d61d8327deb882cf99'
```

```
(kali㉿kali)-[~]
$ hashid -m -j '5f4dcc3b5aa765d61d8327deb882cf99'
Analyzing '5f4dcc3b5aa765d61d8327deb882cf99'
[+] MD2 [JtR Format: md2]
[+] MD5 [Hashcat Mode: 0][JtR Format: raw-md5]
[+] MD4 [Hashcat Mode: 900][JtR Format: raw-md4]
[+] Double MD5 [Hashcat Mode: 2600]
[+] LM [Hashcat Mode: 3000][JtR Format: lm]
[+] RIPEMD-128 [JtR Format: ripemd-128]
[+] Haval-128 [JtR Format: haval-128-4]
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5 [Hashcat Mode: 8600][JtR Format: lotus5]
[+] Skype [Hashcat Mode: 23]
[+] Snefru-128 [JtR Format: snefru-128]
[+] NTLM [Hashcat Mode: 1000][JtR Format: nt]
[+] Domain Cached Credentials [Hashcat Mode: 1100][JtR Format: mscach]
[+] Domain Cached Credentials 2 [Hashcat Mode: 2100][JtR Format: mscach2]
[+] DNSSEC(NSEC3) [Hashcat Mode: 8300]
[+] RAdmin v2.x [Hashcat Mode: 9900][JtR Format: radmin]
```

analyze a file with multiple hashes

hashid hashes.txt

```
(kali㉿kali)-[~]
$ hashid hashes.txt
--File 'hashes.txt'--
Analyzing 'root:*:0:0:root:/root:/usr/bin/zsh'
[+] Unknown hash
Analyzing 'daemon:*:1:1:daemon:/usr/sbin:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'bin:*:2:2:bin:/bin:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'sys:*:3:3:sys:/dev:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'sync:*:4:65534:sync:/bin:/bin/sync'
[+] Unknown hash
Analyzing 'games:*:5:60:games:/usr/games:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'man:*:6:12:man:/var/cache/man:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'lp:*:7:7:lp:/var/spool/lpd:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'mail:*:8:8:mail:/var/mail:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'news:*:9:9:news:/var/spool/news:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'uucp:*:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'proxy:*:13:13:proxy:/bin:/usr/sbin/nologin'
[+] Unknown hash
Analyzing 'www-data:*:33:33:www-data:/var/www:/usr/sbin/nologin'
[+] Unknown hash
```

hashid also supports -e (extended) to list salted variants, and -o <file> to write results to a file.

7. Interpreting output — practical tips

- **Look at length first.** Many hash algorithms have characteristic length (in hex or Base64):
 - MD5 — 32 hex chars
 - SHA-1 — 40 hex chars
 - SHA-256 — 64 hex chars
 - SHA-512 — 128 hex chars
 - bcrypt — typically starts with \$2a\$, \$2b\$ and is ~60 chars
 - NTLM — 32 hex chars (looks like MD5 length — distinguish by context)
- **Watch for prefixes/special markers.** Leading \$ components often indicate salted or application-specific formats: \$P\$, \$\$, \$2a\$, \$6\$ etc (e.g., PHPass, bcrypt, modular crypt formats).
- **Hex vs Base64 vs ASCII.** If the string contains only [0-9a-fA-F] and its length matches known hex lengths, it's likely hex. If you see letters +/- it may be Base64 — decode first before assuming a hex hash.
- **Multiple possible matches:** Tools are heuristic-based. When the tool lists several possibilities, combine: length, character set, presence of markers, and the source (e.g., a WordPress database vs a Linux /etc/shadow). Use contextual clues.
- **When in doubt, use more than one tool.** Compare hash-identifier, hashid, name-that-hash, and online analyzers.

8. Mapping to cracking tools (Hashcat / John)

hashid can print Hashcat mode numbers and John formats — this is extremely useful to prepare cracking jobs.

Example workflow: 1. Identify hash type with hashid -m -j 'HASH'. 2. Suppose hashid returns MySQL and Hashcat Mode: 300 (example). Then you would run Hashcat with the reported mode:

```
hashcat -m 300 -a 0 hashes.txt wordlist.txt
```

3. For John, convert format if needed or directly invoke John with recommended format.

Note: hash-identifier (classic) does **not** always print Hashcat mode numbers — prefer hashid when you need exact modes.

9. Batch / automation suggestions

- For folders of leaks or many extracted hashes, use hashid in a script to produce CSV or plain text results.
- Example one-liner (pseudo):

```
while read -r h; do echo "$h"; hashid -m -j "$h"; done < hashes.txt > hash_report.txt
```

- Normalize inputs: trim whitespace, remove surrounding quotes, handle common separators like : (username:hash) by parsing fields first.

10. Common pitfalls & troubleshooting

- **Hex strings ≠ hash:** A long hex string might be a key, certificate fingerprint, or binary blob — not always a password hash.
- **Salted strings can be mis-identified.** Salted variants can hide the underlying algorithm; look for the salt format and check the application that produced it.
- **Encoding problems:** Some tools expect single-quoted input on Unix shells to prevent interpolation—use single quotes if your hash contains \$.
- **False precision from a single tool:** If hashid lists multiple matches, do not assume certainty — test with the cracking tool on a small sample or search for the application-specific format (e.g., Joomla, Wordpress, MySQL).

11. Short cheat-sheet (common hash lengths)

- MD5 — 32 hex characters
- NTLM — 32 hex characters (but different origin than MD5)
- SHA-1 — 40 hex
- SHA-256 — 64 hex
- SHA-512 — 128 hex
- bcrypt — begins with \$2 and ~60 chars
- MD5-based salted / application variants may include additional text (e.g., md5(\$salt.\$pass)).

13. Further reading and references

(Use these to deepen knowledge) - Official hash-identifier repository (GitHub) - hashid / hashID project (GitHub / PyPI) - Hashcat example hashes and wiki

Made by Moez Javed

14. Appendix — quick commands summary

```
# Install on Kali
```

```
sudo apt update && sudo apt install hash-identifier
```

```
# Run classic interactive
```

hash-identifier

```
# Install modern (pip)
```

```
pip install hashid
```

Quick modern usage

```
hashid -m -j 'HASH'
```

```
# Process file with hashid
```

hashid hashes.txt

Easy Example — Hash-Identifier

Step 1: Open terminal

On Kali, type:

hash-identifier

```
(kali㉿kali)-[~]
$ hash-identifier

#####
#                                     #
#   ^__^                            ^__^                               #
#  (oo)\_______                      (__)\_____                  #
#     /====\                             /====\                   #
#    /        \                         /        \                 #
#   /          \                       /          \                #
#  /            \                     /            \               #
# /              \                   /              \              #
#/_              _/                 /_              _/             #
#                          v1.2                                #
#                        By Zion3R                              #
#                        www.Blackexploit.com                   #
#                        Root@Blackexploit.com                  #
#####

HASH: 1234

Possible Hashs:
[+] CRC-16

HASH: 5f4dcc3b5aa765d61d8327deb882cf99

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))

Least Possible Hashs:
[+] RAdmin v2.x
```

Step 2: Tool starts

Made by Moez Javed

Made by Moez Javed

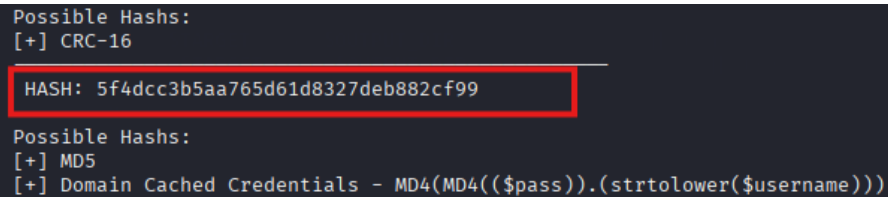
You'll see a banner and then a prompt that says:

HASH:

Step 3: Paste a hash

Let's take a very famous MD5 hash:

5f4dcc3b5aa765d61d8327deb882cf99



```
Possible Hashs:
[+] CRC-16
HASH: 5f4dcc3b5aa765d61d8327deb882cf99
Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
```

Paste it at the HASH: prompt and press **Enter**.

Step 4: Tool output

Hash-Identifier will show something like:

Possible Hashs:

[+] MD5

[+] Domain Cached Credentials

[+] MD4

[+] NTLM

Made by Moez Javed

```
Least Possible Hashs:  
[+] RAdmin v2.x  
[+] NTLM  
[+] MD4  
[+] MD2  
[+] MD5(HMAC)  
[+] MD4(HMAC)  
[+] MD2(HMAC)  
[+] MD5(HMAC Wordpress))  
[+] Haval-128  
[+] Haval-128(HMAC)  
[+] RipeMD-128  
[+] RipeMD-128(HMAC)  
[+] SNEFRU-128  
[+] SNEFRU-128(HMAC)  
[+] Tiger-128  
[+] Tiger-128(HMAC)  
[+] md5($pass.$salt)  
[+] md5($salt.$pass)  
[+] md5($salt.$pass.$salt)  
[+] md5($salt.$pass.$username)  
[+] md5($salt.md5($pass))  
[+] md5($salt.md5($pass))  
[+] md5($salt.md5($pass.$salt))  
[+] md5($salt.md5($pass.$salt))  
[+] md5($salt.md5($salt.$pass))  
[+] md5($salt.md5(md5($pass).$salt))  
[+] md5($username.0.$pass)  
[+] md5($username.LF.$pass)
```

Step 5: Interpret

- The hash is **32 characters long** → typical for MD5/NTLM.
- In most contexts (like an old website), it's MD5.
- If it comes from Windows SAM file, it's NTLM.