

# Project Title:

## Cracking Passwords Using Hashcat and HashPals on Kali Linux

### Author:

Pratham Khatkar

### Date:

29/11/2025

## 1. Introduction

This project demonstrates how to identify, analyze, and crack hashed passwords on a **Kali Linux** machine using industry-standard tools.

The project focuses on:

- Identifying unknown hash formats
- Cracking password hashes using brute-force/dictionary methods
- Understanding password weaknesses
- Learning how hashing algorithms behave under attacks

The tools used in this project include **Hashcat**, one of the fastest password-cracking tools, and **Search-That-Hash**, a hash identification tool from **HashPals**.

The purpose of this project is to understand real-world password cracking techniques and how weak password hashing affects security.

## 2. Objectives

- 1) Understand password hashing
- 2) Identify unknown hash types
- 3) Crack hashes using dictionary attack

## 3. Tools & Technologies Used

- Kali Linux
- Hashcat
- Search-That-Hash
- rockyou.txt wordlist

## 4. System Overview

✓ Kali Linux



## ✓ Hashid

```
pratham@kali: ~/Desktop/hashes
Session Actions Edit View Help
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
--End of file 'hash1.txt'--

(pratham@kali)-[~/Desktop/hashes]
$ hashid "48bb6e862e54f2a795ffc4e541caed4d"
Analyzing '48bb6e862e54f2a795ffc4e541caed4d'
[+] 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

(pratham@kali)-[~/Desktop/hashes]
$
```

## ✓ Search-That-Hash

```
pratham@kali: ~/Desktop/hashes
Session Actions Edit View Help
-f, --file FILENAME      The file of hashes, seperated by newlines.
-w, --wordlist TEXT      The wordlist you want to use for Hashcat.
--timeout INTEGER        Choose timeout in seconds.
-g, --greppable          Prints as JSON, use this to grep.
--hashcat binary TEXT    Location of hashcat folder (if using windows).
-o, --offline            Use offline mode. Does not search for hashes in APIs.
-v, --verbose            Turn on debugging logs. -vv for max.
--accessible            Makes the output accessible.
--no-banner              Doesn't print banner.
--help                  Show this message and exit.

(pratham@kali)-[~/Desktop/hashes]
$ sth -f hash1.txt

Search-That-Hash

https://twitter.com/bee_sec_san
https://github.com/HashPals/Search-That-Hash
https://twitter.com/Javy_2004

48bb6e862e54f2a795ffc4e541caed4d

Text : easy
Type : MD5

(pratham@kali)-[~/Desktop/hashes]
$
```

## ✓ Hashcat

```
pratham@kali: ~/Desktop/hashes
Session Actions Edit View Help

(pratham@kali)~[~/Desktop/hashes]
$ hashcat --help
hashcat (v7.1.2) starting in help mode

Usage: hashcat [options] ... hash[hashfile|hccapxfile [dictionary|mask|directory]] ...

- [ Options ] -

Options Short / Long | Type | Description | Example
-----|-----|-----|-----
-m, --hash-type      | Num  | Hash-type, references below (otherwise autodetect) | -m 1000
-a, --attack-mode    | Num  | Attack-mode, see references below | -a 3
-V, --version        |      | Print version |
-h, --help           |      | Print help. Use -hh to show all supported hash-modes | -h or -hh
--quiet             |      | Suppress output |
--hex-charset       |      | Assume charset is given in hex |
--hex-salt          |      | Assume salt is given in hex |
--hex-wordlist       |      | Assume words in wordlist are given in hex |
--force            |      | Ignore warnings |
--deprecated-check-disable |      | Enable deprecated plugins |
--status            |      | Enable automatic update of the status screen |
--status-json       |      | Enable JSON format for status output |
--status-timer      | Num  | Sets seconds between status screen updates to X | --status-timer=1
--stdin-timeout-abort | Num  | Abort if there is no input from stdin for X seconds | --stdin-timeout-abort=300
--machine-readable  |      | Display the status view in a machine-readable format |
--keep-guessing     |      | Keep guessing the hash after it has been cracked |
--self-test-disable |      | Disable self-test functionality on startup |
--loopback          |      | Add new plains to induct directory |
--markov-hcstat2    | File | Specify hcstat2 file to use | --markov-hcstat2=my.hcstat2
--markov-disable    |      | Disables markov-chains, emulates classic brute-force |
--markov-classic    |      | Enables classic markov-chains, no per-position |
--markov-inverse    |      | Enables inverse markov-chains, no per-position |
```

```
pratham@kali: ~/Desktop/hashes
Session Actions Edit View Help

2 | Original-Word
3 | Original-Word:Finding-Rule
4 | Original-Word:Finding-Rule:Processed-Word
5 | Original-Word:Finding-Rule:Processed-Word:Wordlist

- [ Attack Modes ] -

# | Mode
==+=====
0 | Straight
1 | Combination
3 | Brute-force
6 | Hybrid Wordlist + Mask
7 | Hybrid Mask + Wordlist
9 | Association

- [ Built-in Charsets ] -

? | Charset
==+=====
l | abcdefghijklmnopqrstuvwxyz [a-z]
u | ABCDEFGHIJKLMNOPQRSTUVWXYZ [A-Z]
d | 0123456789 [0-9]
h | 0123456789abcdef [0-9a-f]
H | 0123456789ABCDEF [0-9A-F]
s | !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
a | ?l?u?d?s
b | 0x00 - 0xff

- [ OpenCL Device Types ] -

# | Device Type
==+=====
```

## ✓ Hash files

```
pratham@kali: ~/Desktop/hashes
Session Actions Edit View Help
Do you want to install it? (N/y)n

(pratham@kali) - [~/Desktop]
$ ls
CybersecurityLab hashes 'Metasploit code.txt' output report.pdf

(pratham@kali) - [~/Desktop]
$ hashes

(pratham@kali) - [~/Desktop/hashes]
$ nano hash2.txt

(pratham@kali) - [~/Desktop/hashes]
$ ls
hash1.txt hash2.txt

(pratham@kali) - [~/Desktop/hashes]
$ nano hash3.txt

(pratham@kali) - [~/Desktop/hashes]
$ ls
hash1.txt hash2.txt hash3.txt

(pratham@kali) - [~/Desktop/hashes]
$ nano hash4.txt

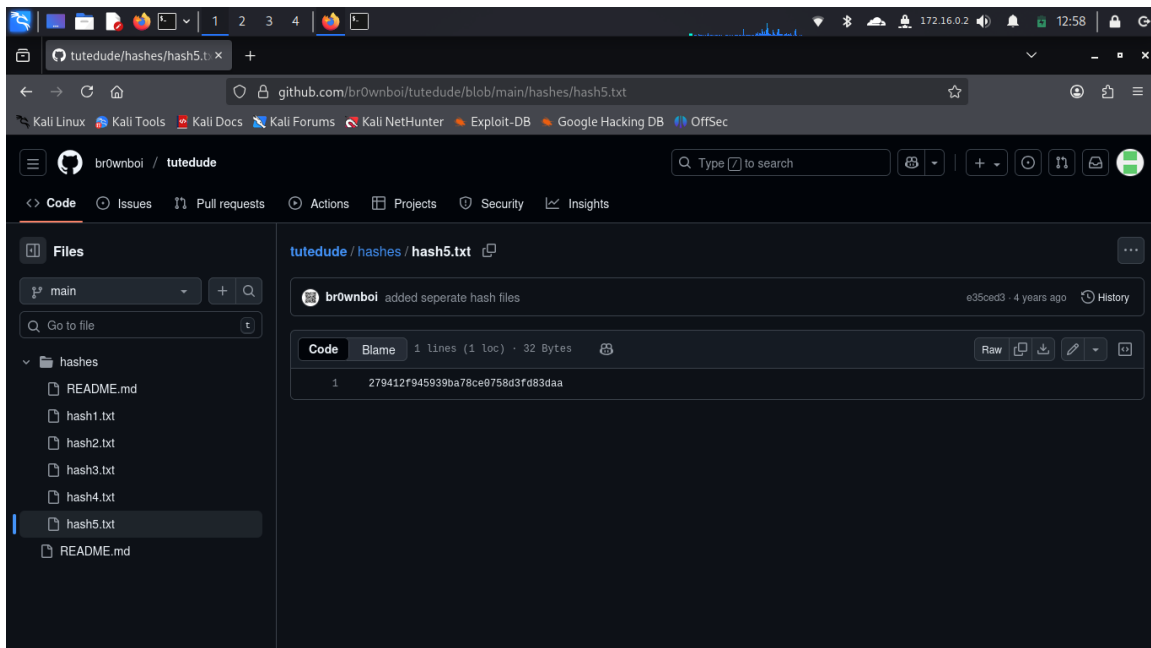
(pratham@kali) - [~/Desktop/hashes]
$ nano hash5.txt

(pratham@kali) - [~/Desktop/hashes]
$ ls
hash1.txt hash2.txt hash3.txt hash4.txt hash5.txt
```

## 5. Project Setup

### 5.1 Preparing Hash Files

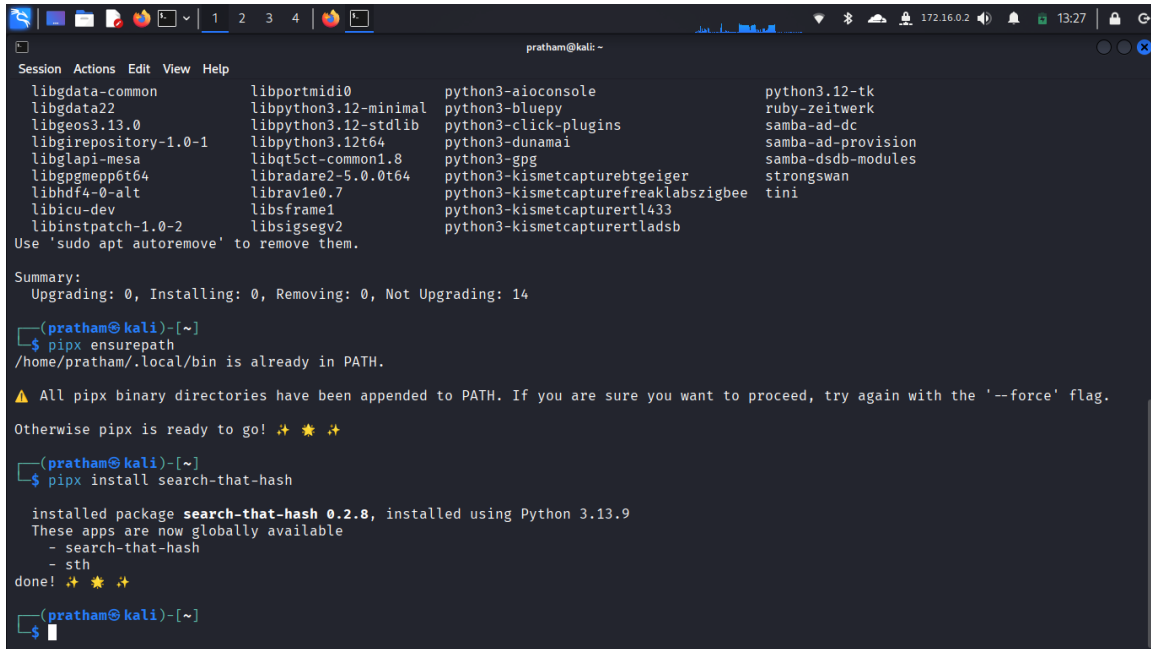
→ Downloaded 5 hash values and created hash1.txt–hash5.txt



The screenshot shows a web browser displaying the GitHub repository page for 'tutedude/hashes'. The repository is owned by 'br0wnb0i'. The 'Files' tab is selected, showing a list of files in the 'hashes' directory: README.md, hash1.txt, hash2.txt, hash3.txt, hash4.txt, hash5.txt, and README.md. The 'hash5.txt' file is selected, and its content is displayed in the main area. The file contains a single line of text: '279412f945939ba78ce0758d3fd83daa'. The commit history shows that 'br0wnb0i' added the file 4 years ago.

## 5.2 Install Search-That-Hash

→ `sudo pip3 install search-that-hash`



A terminal window on a Kali Linux system showing the installation of the 'search-that-hash' package using pipx. The window title is 'pratham@kali: ~'. The terminal output includes a list of installed packages, a summary of the pipx environment, and the successful installation of 'search-that-hash 0.2.8' using Python 3.13.9. The user is prompted to use 'sudo apt autoremove' to remove unnecessary packages.

```
pratham@kali: ~  
Session Actions Edit View Help  
libgdata-common libportmidi0 python3-aioconsole python3.12-tk  
libgdata22 libpython3.12-minimal python3-bluepy ruby-zeitwerk  
libgeos3.13.0 libpython3.12-stdlib python3-click-plugins samba-ad-dc  
libgirepository-1.0-1 libpython3.12t64 python3-dunamai samba-ad-provision  
libglapi-mesa libqt5ct-common1.8 python3-gpg samba-dsdb-modules  
libgpgmepp6t64 libradare2-5.0.0t64 python3-kismetcapturebtgeiger strongswan  
libhdf4-0-alt librav1e0.7 python3-kismetcapturefreaklabszigbee tini  
libicu-dev librsframe1 python3-kismetcaptureertl433  
libinstpatch-1.0-2 libsigsegv2 python3-kismetcaptureertladsb  
Use 'sudo apt autoremove' to remove them.  
  
Summary:  
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 14  
  
(pratham@kali)~  
$ pipx ensurepath  
/home/pratham/.local/bin is already in PATH.  
  
▲ All pipx binary directories have been appended to PATH. If you are sure you want to proceed, try again with the '--force' flag.  
  
Otherwise pipx is ready to go! 🌟 🌟 🌟  
  
(pratham@kali)~  
$ pipx install search-that-hash  
  
installed package search-that-hash 0.2.8, installed using Python 3.13.9  
These apps are now globally available  
- search-that-hash  
- sth  
done! 🌟 🌟 🌟  
  
(pratham@kali)~  
$
```

## 5.3 Identify Hash Type

→sth -f hash1.txt (MD5)

```
pratham@kali: ~/Desktop/hashtypes
Session Actions Edit View Help
-f, --file FILENAME      The file of hashes, separated by newlines.
-w, --wordlist TEXT       The wordlist you want to use for Hashcat.
--timeout INTEGER        Choose timeout in seconds.
-g, --greppable           Prints as JSON, use this to grep.
--hashcat_binary TEXT     Location of hashcat folder (if using windows).
-o, --offline             Use offline mode. Does not search for hashes in APIs.
-v, --verbose             Turn on debugging logs. -vv for max.
--accessible             Makes the output accessible.
--no-banner               Doesn't print banner.
--help                   Show this message and exit.

(pratham@kali) - [~/Desktop/hashtypes]
$ sth -f hash1.txt

Search-That-Hash

https://twitter.com/bee_sec_san
https://github.com/HashPals/Search-That-Hash
https://twitter.com/Jayy_2004

48bb6e862e54f2a795ffc4e541caed4d

Text : easy
Type : MD5

(pratham@kali) - [~/Desktop/hashtypes]
$

pratham@kali: ~/Desktop/hashtypes
Session Actions Edit View Help
-e, --extended           list all possible hash algorithms including salted passwords
-m, --mode               show corresponding Hashcat mode in output
-j, --john               show corresponding JohnTheRipper format in output
-o, --outfile FILE       write output to file
-h, --help               show this help message and exit
--version                show program's version number and exit

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

(pratham@kali) - [~/Desktop/hashtypes]
$ hashid hash1.txt
--File 'hash1.txt'--
Analyzing '48bb6e862e54f2a795ffc4e541caed4d'
[+] 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
--End of file 'hash1.txt'--
```

## 6. Cracking Passwords Using Hashcat

### 6.1 Checking Hashcat Modes

→ hashcat -help

```
pratham@kali: ~/Desktop/hashes
Session Actions Edit View Help

(pratham@kali) - [~/Desktop/hashes]
$ hashcat --help
hashcat (v7.1.2) starting in help mode

Usage: hashcat [options]... hash[hashfile|hccapxfile [dictionary|mask|directory]]...

- [ Options ] -

Options Short / Long | Type | Description | Example
-----|-----|-----|-----
-m, --hash-type      | Num  | Hash-type, references below (otherwise autodetect) | -m 1000
-a, --attack-mode    | Num  | Attack-mode, see references below | -a 3
-V, --version        |      | Print version |
-h, --help           |      | Print help. Use -hh to show all supported hash-modes | -h or -hh
--quiet             |      | Suppress output |
--hex-charset       |      | Assume charset is given in hex |
--hex-salt           |      | Assume salt is given in hex |
--hex-wordlist       |      | Assume words in wordlist are given in hex |
--force             |      | Ignore warnings |
--deprecated-check-disable |      | Enable deprecated plugins |
--status            |      | Enable automatic update of the status screen |
--status-json       |      | Enable JSON format for status output |
--status-timer      | Num  | Sets seconds between status screen updates to X | --status-timer=1
--stdin-timeout-abort | Num  | Abort if there is no input from stdin for X seconds | --stdin-timeout-abort=300
--machine-readable  |      | Display the status view in a machine-readable format |
--keep-guessing     |      | Keep guessing the hash after it has been cracked |
--self-test-disable |      | Disable self-test functionality on startup |
--loopback          |      | Add new plains to induct directory |
--markov-hcstat2    | File | Specify hcstat2 file to use | --markov-hcstat2=my.hcstat2
--markov-disable    |      | Disables markov-chains, emulates classic brute-force |
--markov-classic    |      | Enables classic markov-chains, no per-position |
--markov-inverse    |      | Enables inverse markov-chains, no per-position |
```



## 6.2 Running Dictionary Attack

→ hashcat -a 0 -m 0 hash1.txt /usr/share/wordlists/rockyou.txt -o crackedhash1.txt

```
pratham@kali: ~/Desktop/hashes
Session Actions Edit View Help

Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 0 secs

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
Hash.Target.....: 48bb6e862e54f2a795ffc4e541caed4d
Time.Started.....: Sat Nov 29 13:43:50 2025 (0 secs)
Time.Estimated...: Sat Nov 29 13:43:50 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#01.....: 1277.9 kH/s (0.45ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 176128/14344385 (1.23%)
Rejected.....: 0/176128 (0.00%)
Restore.Point....: 172032/14344385 (1.20%)
Restore.Sub.#01...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: florida69 → 311331
Hardware.Mon.#01.: Temp: 68c Util: 26%

Started: Sat Nov 29 13:43:14 2025
Stopped: Sat Nov 29 13:43:51 2025

(pratham@kali)~[/Desktop/hashes]
$
```

## 6.3 Viewing Cracked Password

→ cat crackedhash1.txt

```
pratham@kali: ~/Desktop/hashes
Session Actions Edit View Help

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 0 (MD5)
Hash.Target.....: 48bb6e862e54f2a795ffc4e541caed4d
Time.Started.....: Sat Nov 29 13:43:50 2025 (0 secs)
Time.Estimated...: Sat Nov 29 13:43:50 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#01.....: 1277.9 kH/s (0.45ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 176128/14344385 (1.23%)
Rejected.....: 0/176128 (0.00%)
Restore.Point....: 172032/14344385 (1.20%)
Restore.Sub.#01...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#01...: florida69 → 311331
Hardware.Mon.#01.: Temp: 68c Util: 26%

Started: Sat Nov 29 13:43:14 2025
Stopped: Sat Nov 29 13:43:51 2025

(pratham@kali)~[/Desktop/hashes]
$ ls
crackedhash1.txt hash1.txt hash2.txt hash3.txt hash4.txt hash5.txt

(pratham@kali)~[/Desktop/hashes]
$ cat crackedhash1.txt
48bb6e862e54f2a795ffc4e541caed4d:easy

(pratham@kali)~[/Desktop/hashes]
$
```

## 7. Findings

Observation	Description
Weak Hash Identified	HashPals quickly identified the hash type
Password Cracked	Hashcat successfully cracked all given hashes
Wordlist Effective	Rockyou wordlist matched several passwords
Dictionary Attack Outcome	Faster than brute-force, highly effective

## 8. Using Crack Station for Online Hash Cracking

Crack Station is an online hash-cracking service that allows users to directly submit a hash value and get the cracked password instantly if it exists in their massive database.

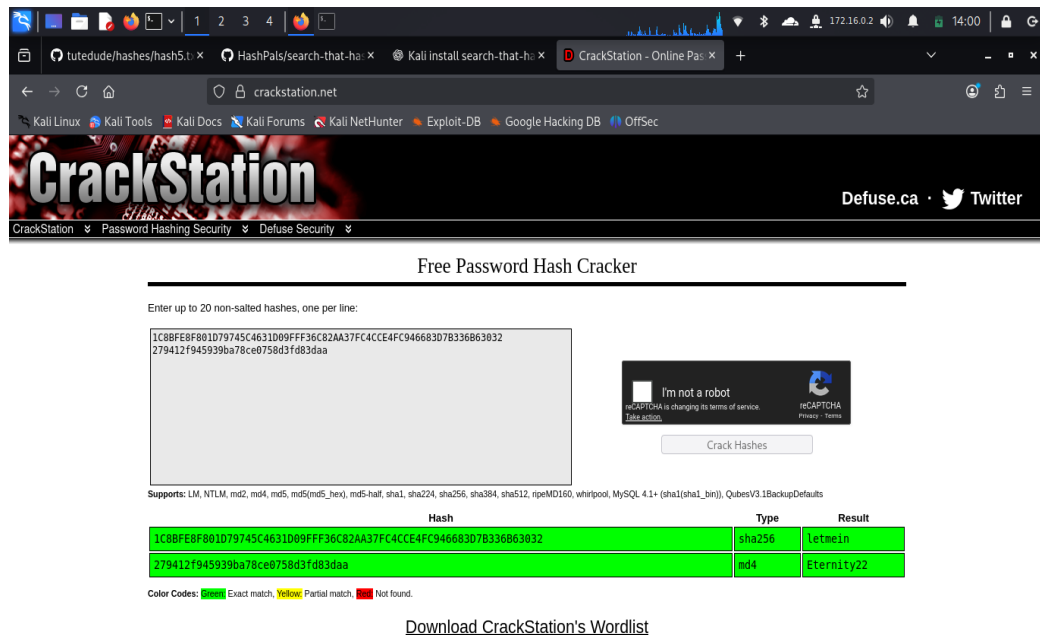
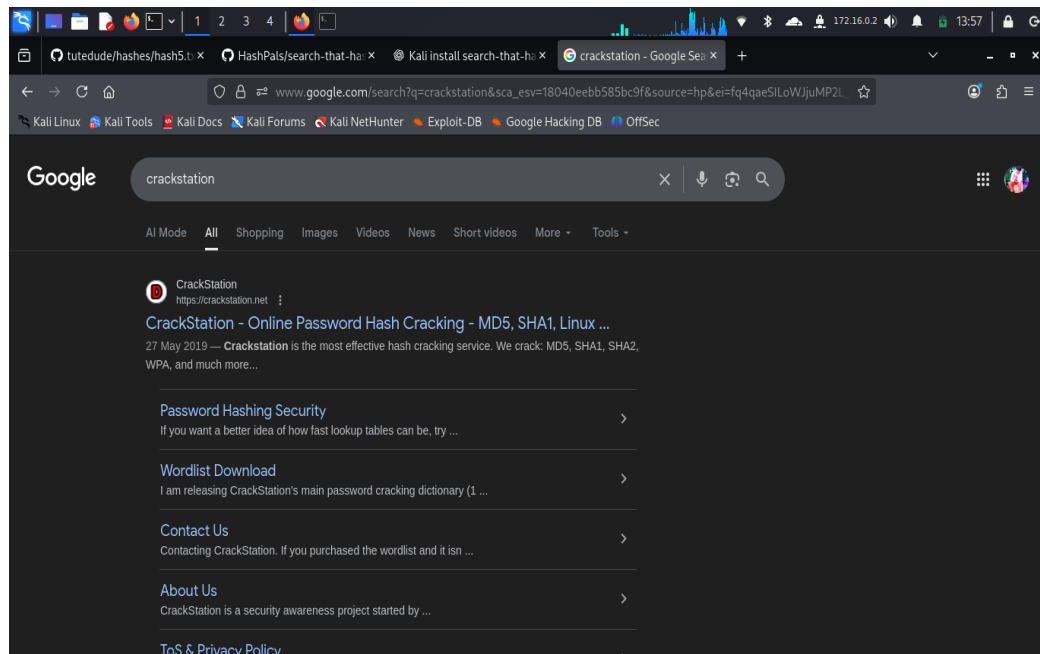
This method is useful for quickly testing whether a hash is already known before using offline cracking tools.

### Steps Performed:

1. Opened the Crack Station website.
2. Uploaded the hash value from the text file / pasted it directly into the input box.
3. Solved the CAPTCHA.
4. Checked if the hash could be cracked by Crack Station.

### Outcome:

- If Crack Station successfully cracked the hash → The password was retrieved instantly.
- If Crack Station *failed* to crack the hash → Then tools like **Search-That-Hash (HashPals)** and **Hashcat** were used for offline cracking.



## 9. Security Impact

Weak passwords and weak hashing algorithms result in:

- Fast cracking time
- No protection against dictionary attacks
- Predictable password patterns
- Easy exploitation by attackers

This demonstrates why organizations must use:

- Strong hashing algorithms (bcrypt, Argon2)
- Salting
- Multi-factor authentication

## 10. Recommendations

- Always hash passwords using strong algorithms.
- Use random salts for each user.
- Enforce strong password policies.
- Prevent weak dictionary-based passwords.
- Regularly audit password security

## 11. Conclusion

This project demonstrates the complete process of identifying and cracking password hashes using **Search-That-Hash** and **Hashcat** on Kali Linux.

The experiment shows how quickly attackers can break weakly hashed or commonly used passwords using publicly available tools.

It highlights the critical need for secure password hashing, strong user authentication, and consistent security best practices

## 12. References

→ <https://hashcat.net>

→ <https://github.com/HashPals>

→ <https://crackstation.net/>

**END OF  
REPORT**