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

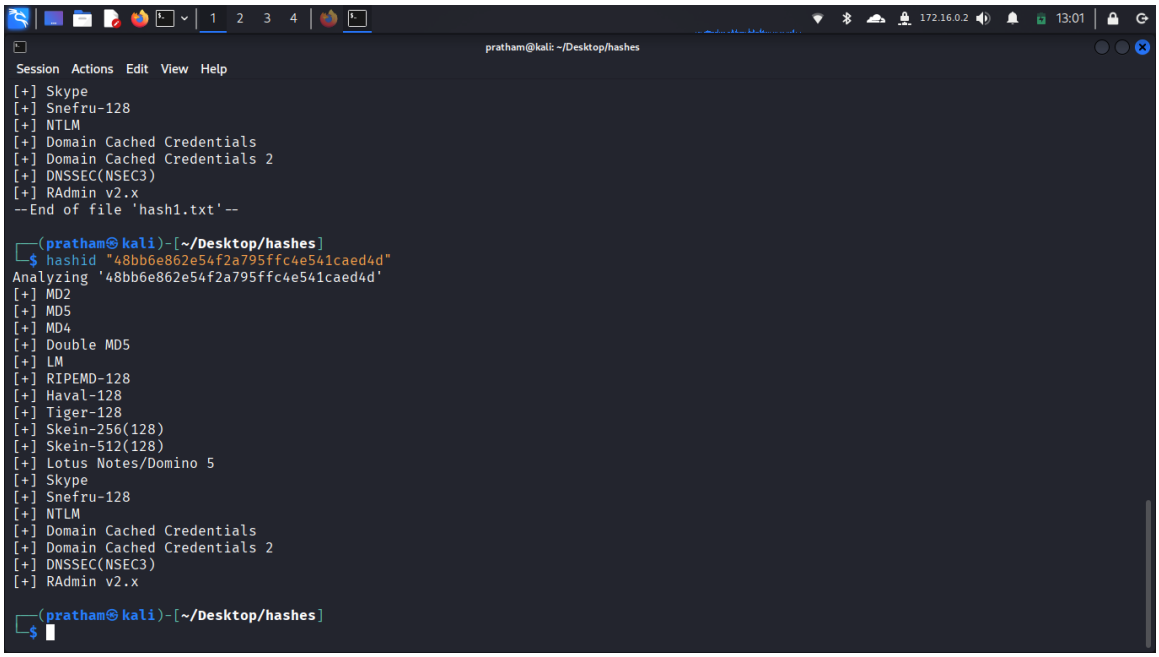
• Search-That-Hash

• rockyou.txt wordlist

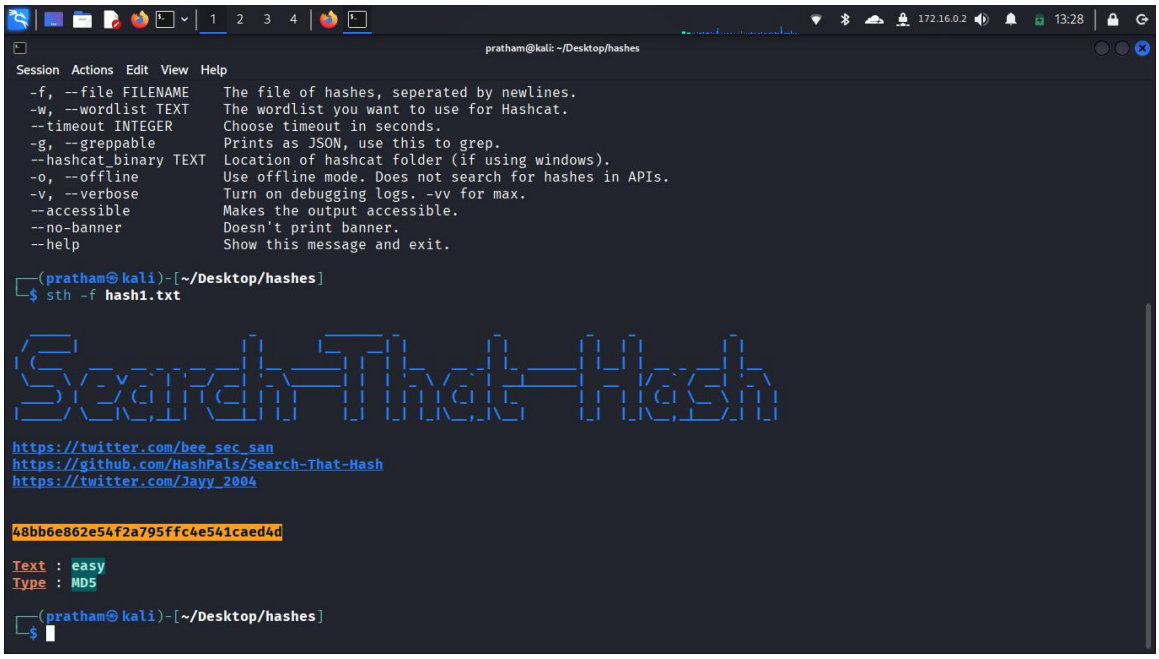
4. System Overview

✓ Kali Linux

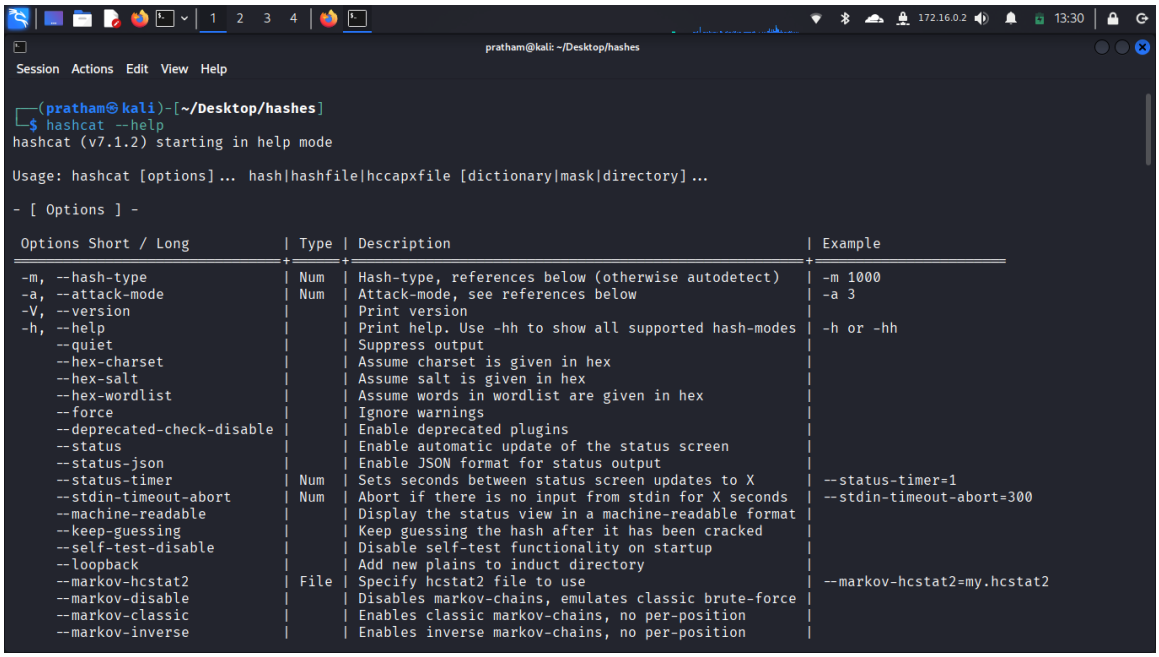
✓Hashid

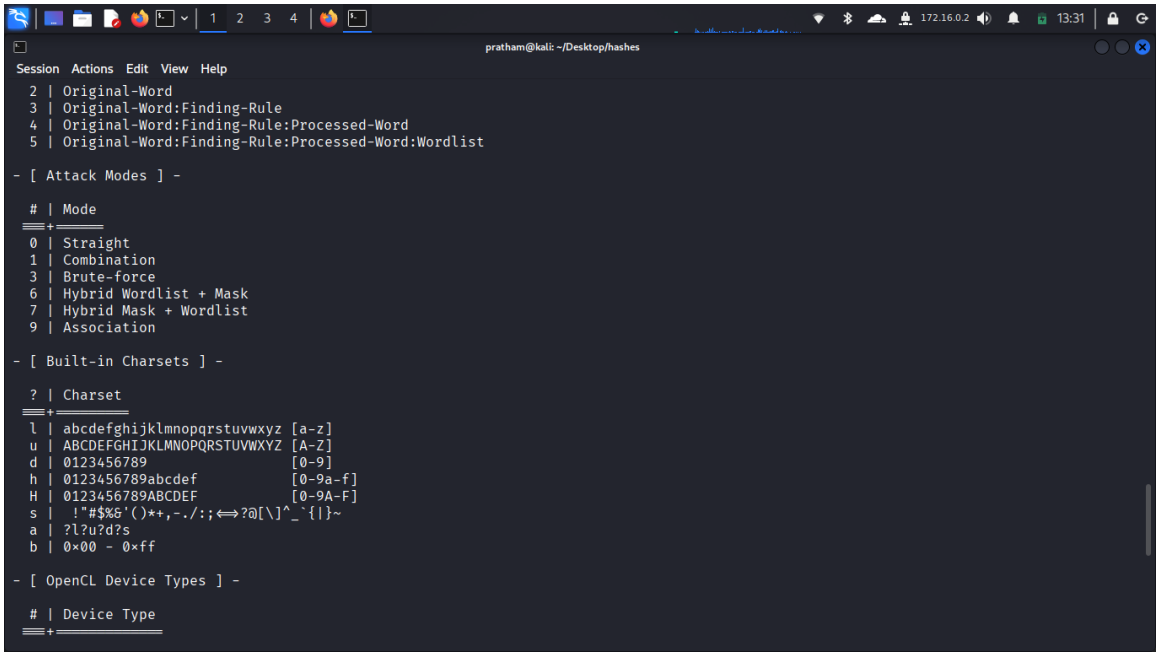


✓Search-That-Hash

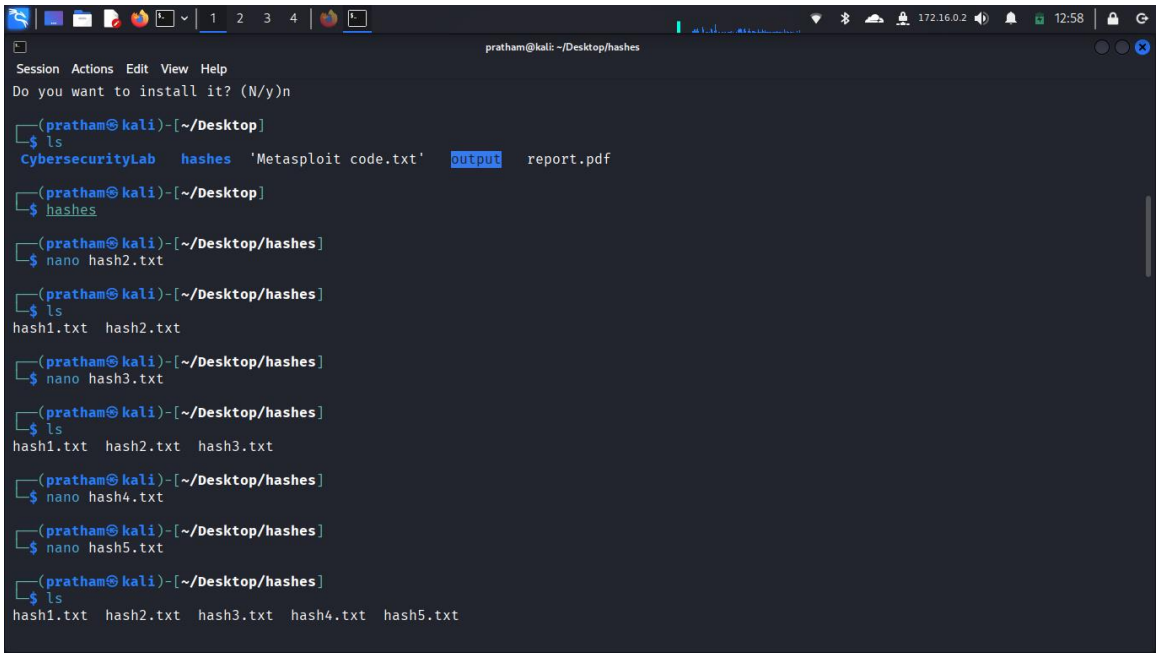


✓Hashcat


1 2 3 4
172.16.0.2
13:30
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 |


1 2 3 4
172.16.0.2
13:31
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

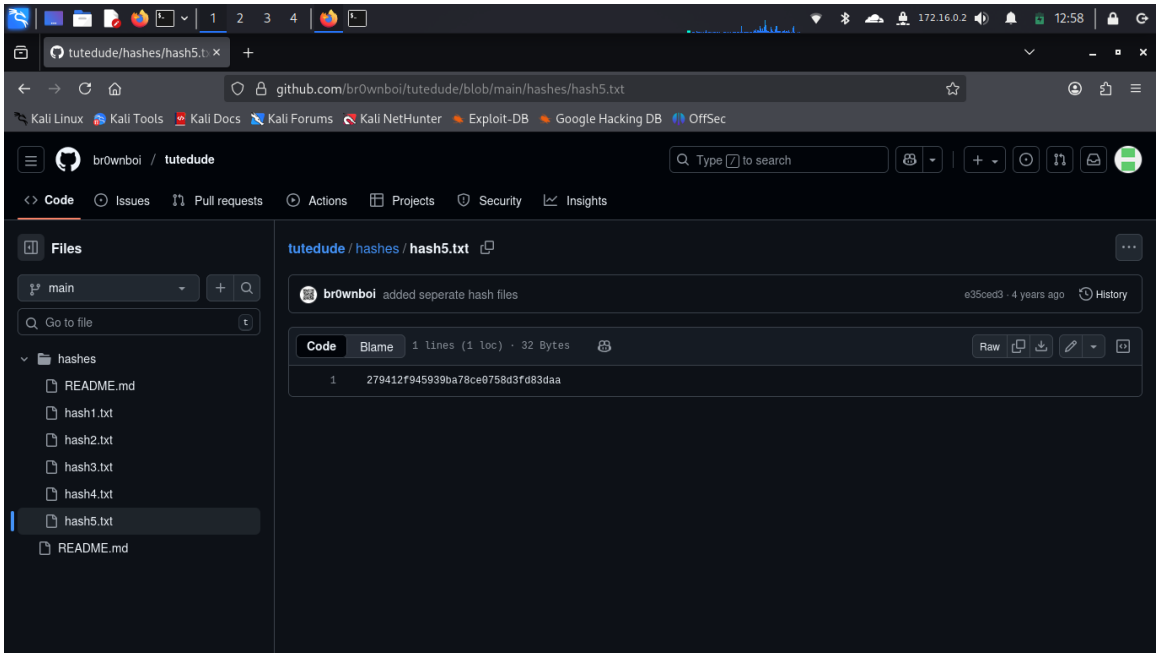
✓Hash files



5. Project Setup

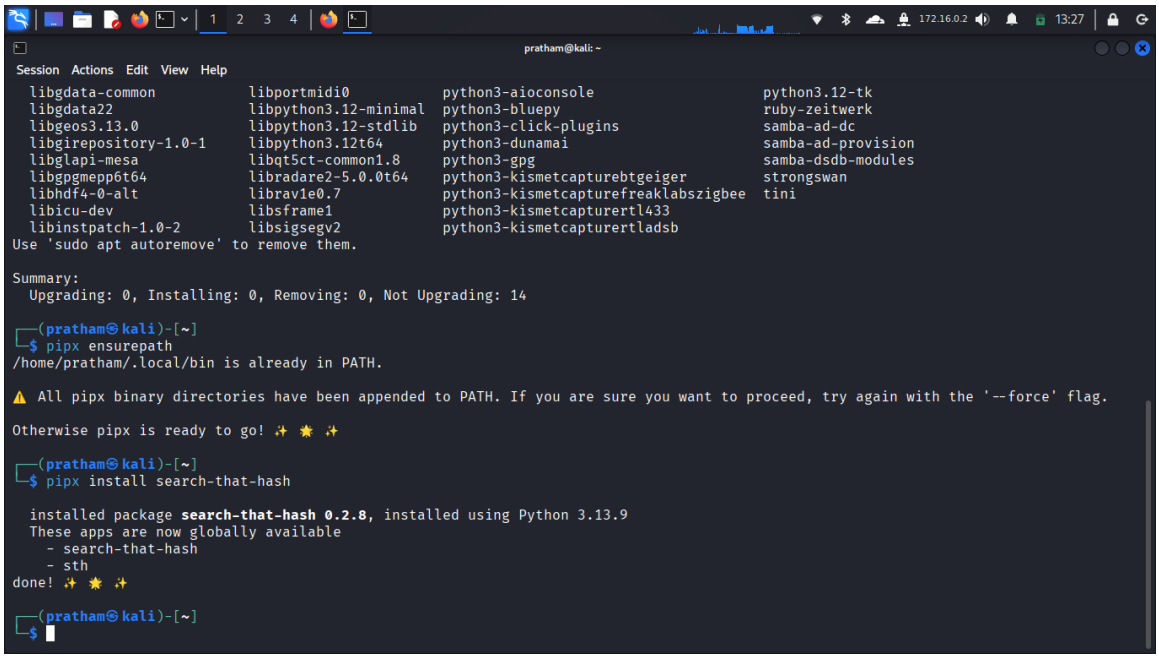
5.1 Preparing Hash Files

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



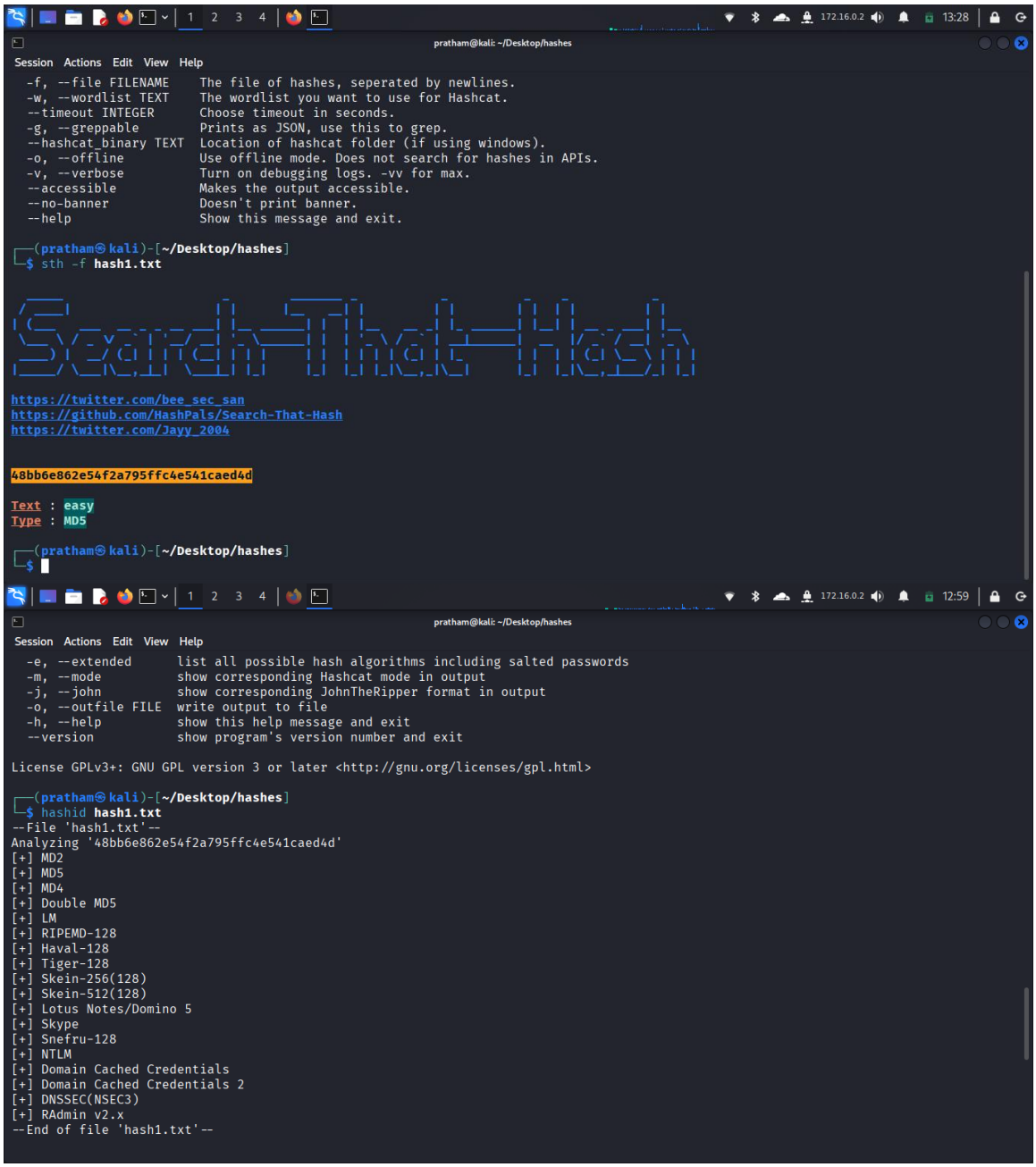
5.2 Install Search-That-Hash

→sudo pip3 install search-that-hash



5.3 Identify Hash Type

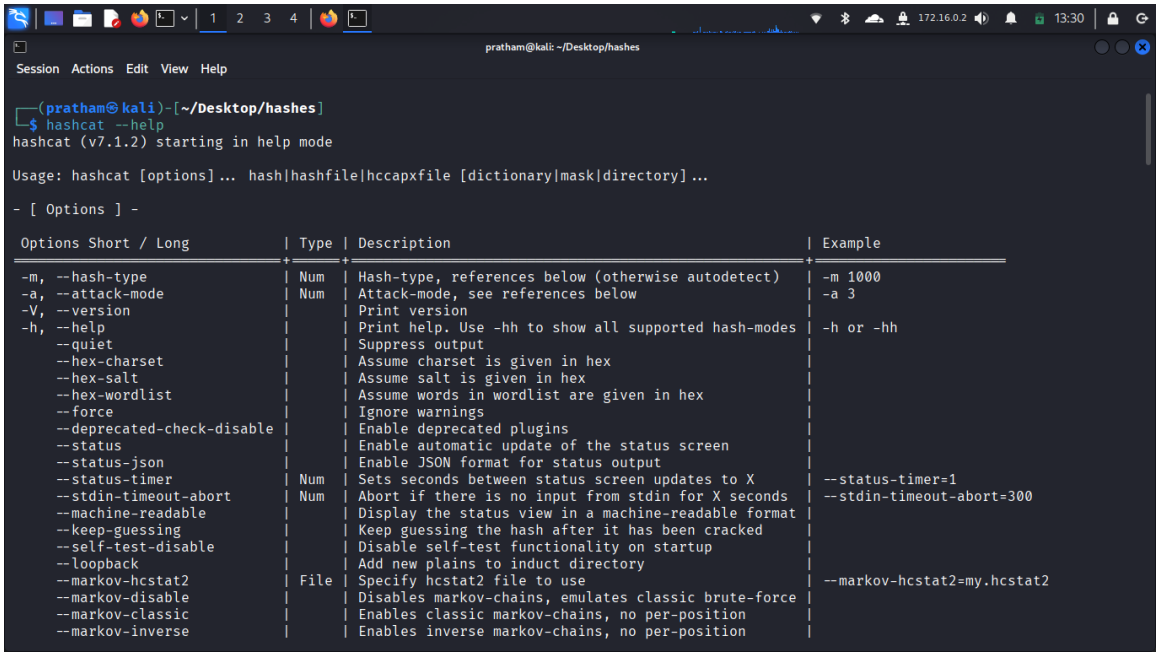
→sth -f hash1.txt (MD5)



6. Cracking Passwords Using Hashcat

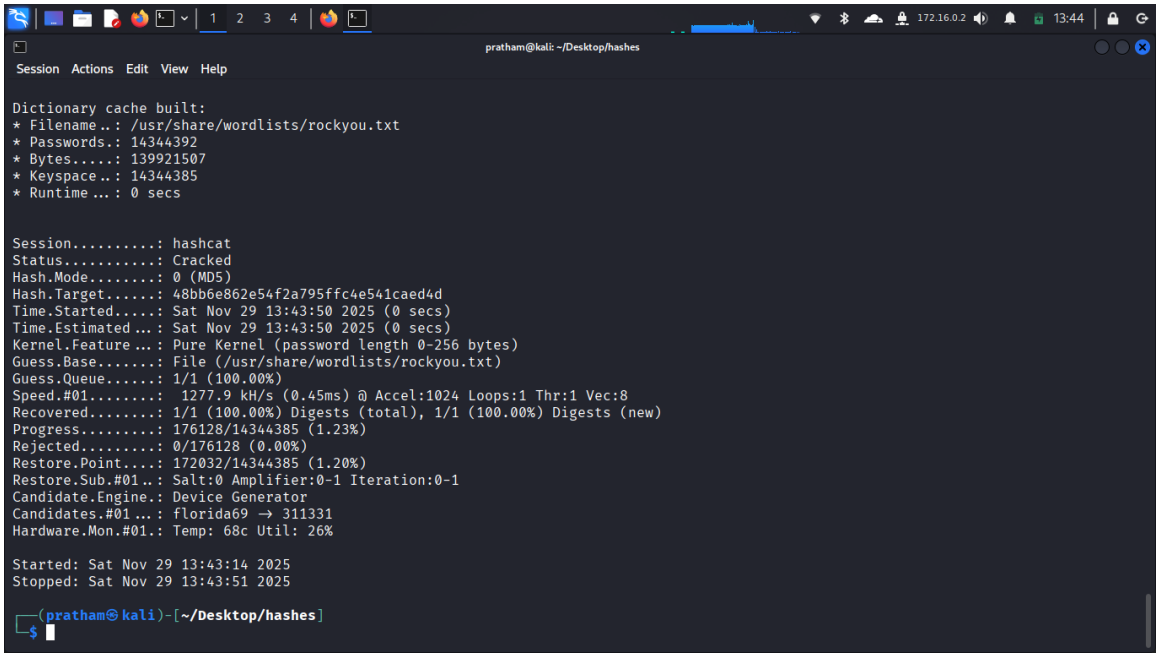
6.1 Checking Hashcat Modes

→hashcat –help



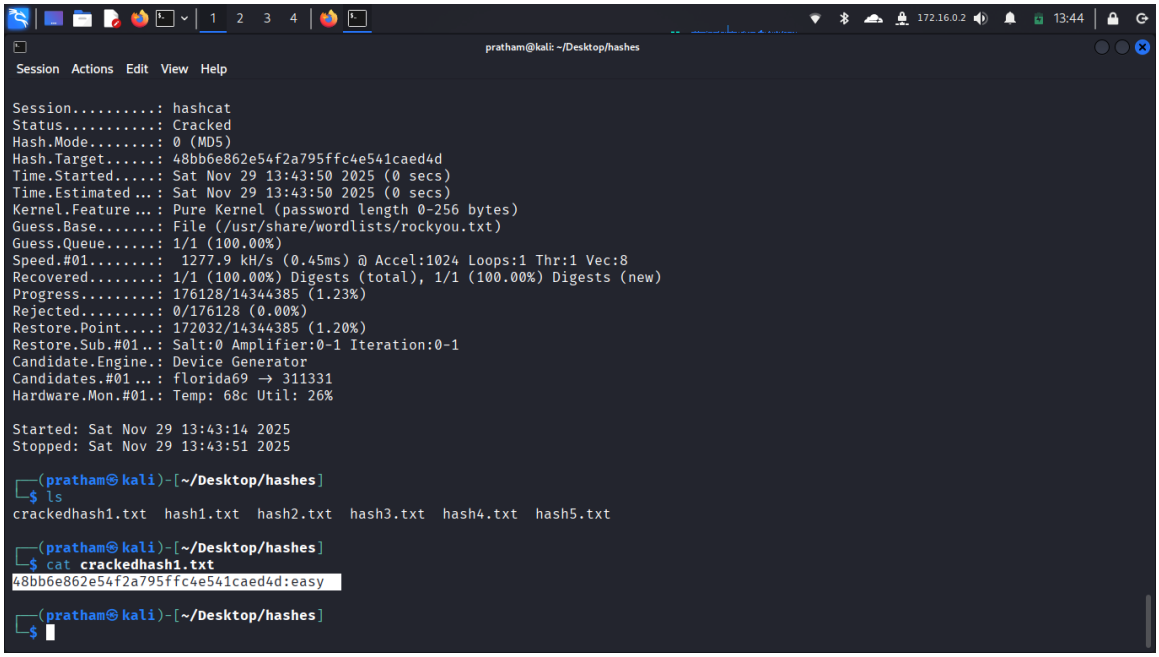
6.2 Running Dictionary Attack

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



6.3 Viewing Cracked Password

→cat crackedhash1.txt



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

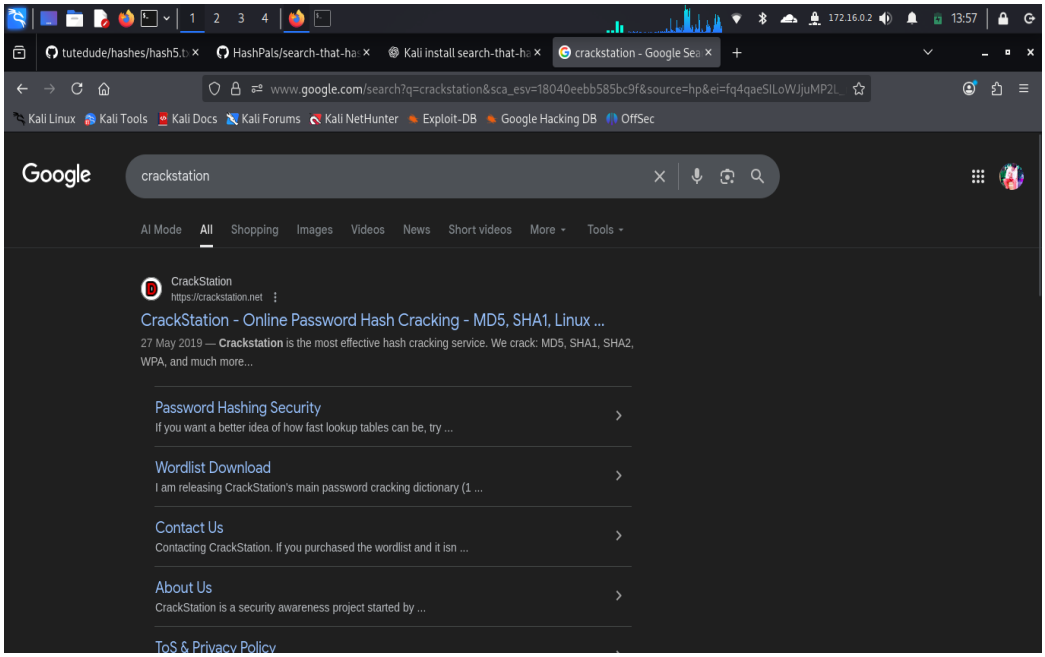
Crack Station is an online hash-cracking service that allows users to directly submit a hash value and get the cracked password instantly if it exists in their massive database.
This method is useful for quickly testing whether a hash is already known before using offline cracking tools.

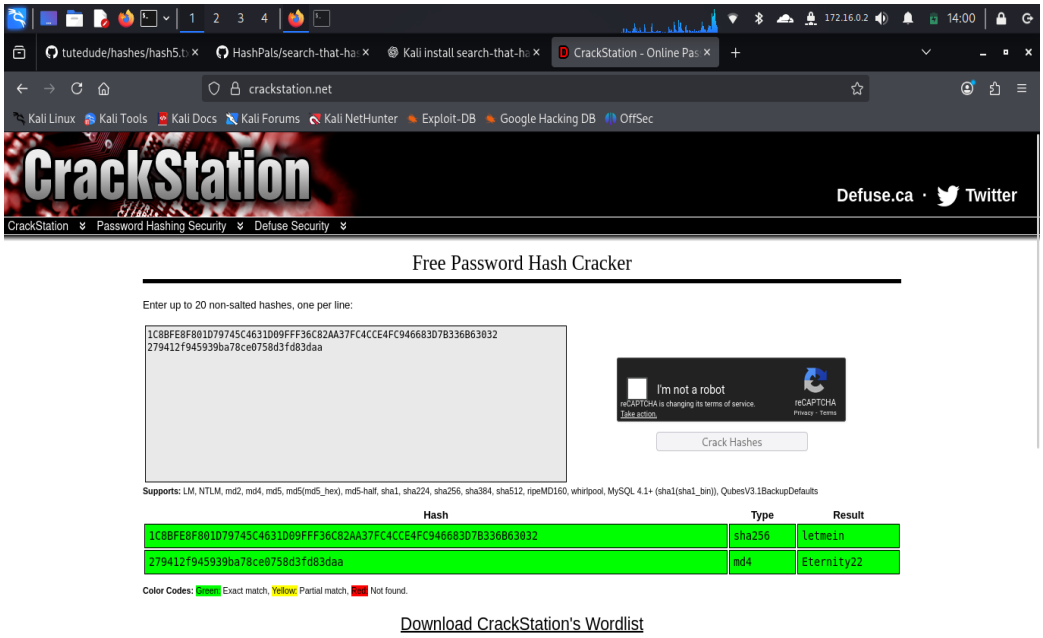
Steps Performed:

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