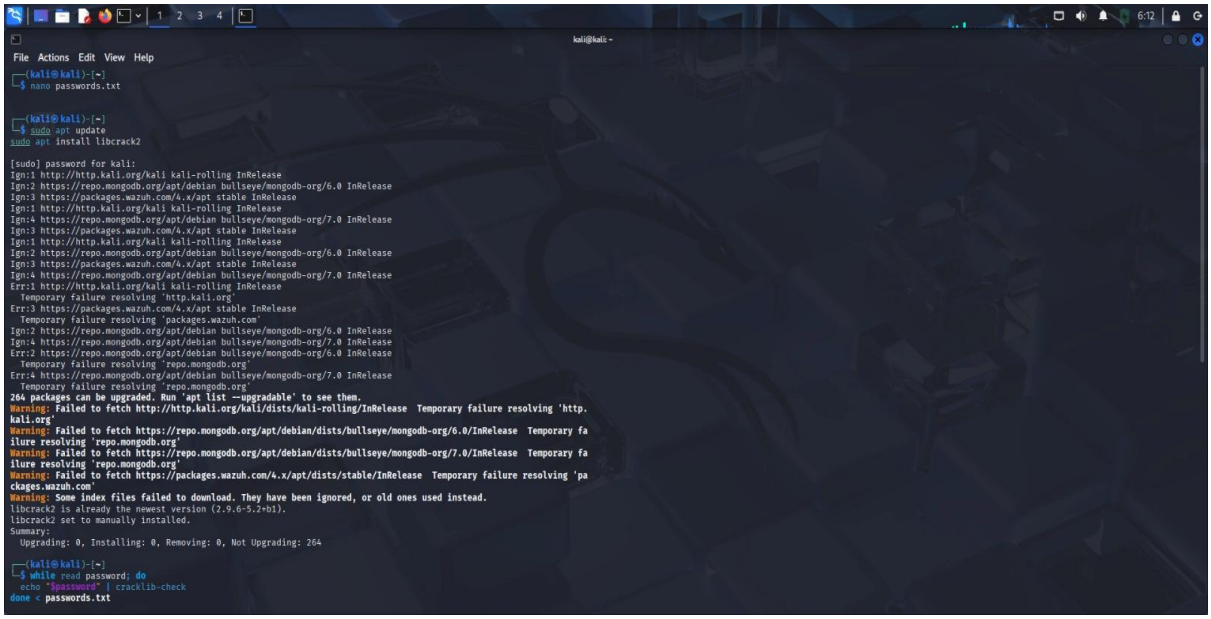
Create a Strong Password and Evaluate Its Strength.

Step 1: Create Multiple Passwords with Varying Complexity*

Create a file with sample passwords:

nano passwords.txt



Paste these samples (or your own):

password

Password123

Pass@2025!

$3cur3P@ssw0rd!

Very$tr0ng&LongP@ss2025!

Save with Ctrl + O, then Enter, then exit with Ctrl + X.

Step 2: Use Uppercase, Lowercase, Numbers, Symbols, and Length Variations*

Make sure your file contains:

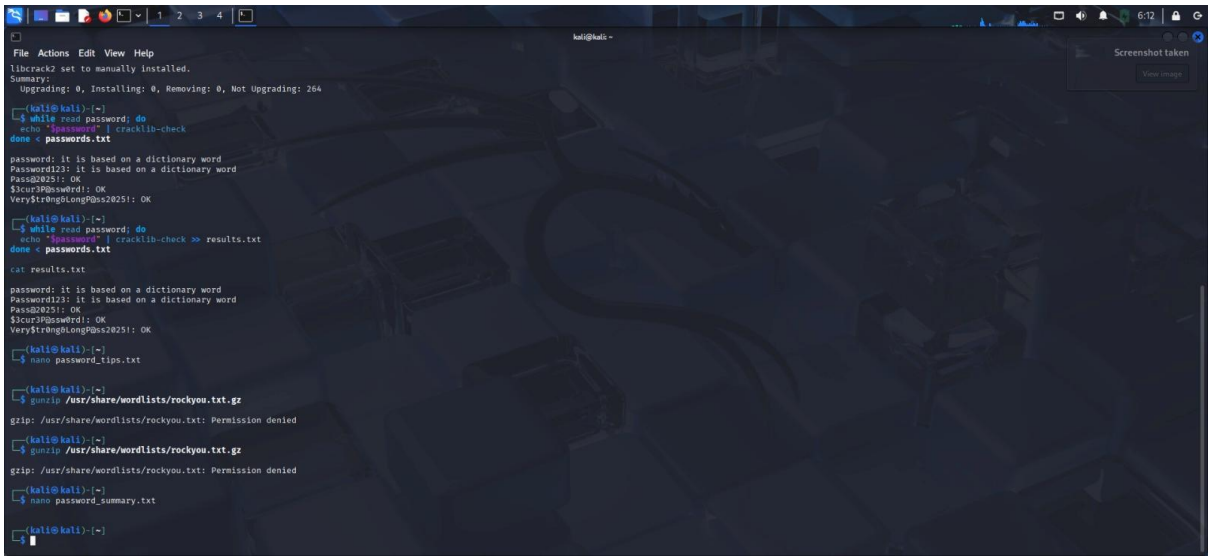
* Only lowercase: password

* Mixed case with numbers: Password123

* Symbols and numbers: Pass@2025!

* Long & complex: Very$tr0ng&LongP@ss2025!

This helps test password complexity variety.



Step 3: Install Password Strength Checker (cracklib)*

sudo apt update

sudo apt install libcrack2

Step 4: Test Passwords Using cracklib-check*

while read password; do

  echo "$password" | cracklib-check

done < passwords.txt

 Output Example:

password: it is based on a dictionary word

Password123: it is based on a dictionary word

Pass@2025!: OK

Step 5: Save Scores and Feedback to a File*

while read password; do

  echo "$password" | cracklib-check >> results.txt

done < passwords.txt

cat results.txt

Now you can review which passwords were strong or weak.

Step 6: Write Down Tips Learned from the Evaluation*

nano password_tips.txt

Add tips like:

* Always use 12+ characters

* Use symbols like @, #, !, &

* Mix upper, lower, numbers

* Avoid dictionary words

* Avoid keyboard patterns (e.g., qwerty)

Step 7: Research Common Password Attacks*

Use built-in tools and wordlists in Kali to understand:

Brute Force Attack (Example using hydra or john)

Tries all combinations – takes longer for complex passwords.

Dictionary Attack (using rockyou.txt)

Unzip wordlist

gunzip /usr/share/wordlists/rockyou.txt.gz

View sample

head /usr/share/wordlists/rockyou.txt

This file contains millions of real leaked passwords.

Step 8: Summarize How Complexity Affects Security

Open a summary file:

nano password_summary.txt

Write something like:

Password complexity greatly increases security.

- Longer passwords reduce chances of brute-force success.

- Adding symbols defeats pattern-based guesses.

- Avoiding dictionary words blocks simple dictionary attacks.

Files I'll Create

| File Name | Description |

| ---------- | ---------- |

| passwords.txt | List of sample passwords |

| results.txt | Output from cracklib-check |

| password_tips.txt | Best practices/tips from analysis |

| password_summary.txt | Impact of password complexity |