Task 6 — Password Strength Evaluation (Kali Linux)

Summary

Objective: create multiple passwords, evaluate strength using local tools and online meter, and summarize best practices.

Files in this repo

- `passwords_plain.txt` — example passwords (DO NOT PUBLISH real passwords)
- `pwgen_generated.txt` — generated passwords (examples)
- `passwd_shadow.txt` — sha512crypt-style hashed passwords (used for local cracking demo)
- `screenshots/` — screenshots showing tool outputs
- `README.md` — this document

Passwords tested (redacted)

#	Example (redacted)	Local tool result (cracklib)	John cracking result
1	pass***	FAILED / weak	cracked (found)
2	hello***	FAILED / weak	cracked (found)
3	Fateh@2025	OK / moderate	not cracked (slow)
4	Linux#Power9	OK / moderate	not cracked
5	Mokanojiya@…#	OK / strong	not cracked

> Note: Real passwords must never be uploaded to public repositories. Store only hashes or redacted examples.

Tools used

- `cracklib-check` — dictionary & rule checks
- `john` (John the Ripper) — offline cracking / wordlist tests
- `pwgen` — password generation
- `openssl` — hashing
- `scrot` / `gnome-screenshot` — screenshots
- `passwordmeter.com` — online strength check (screenshots included)

Key findings

- **Length** and **entropy** matter most: 12+ characters with mixed classes are recommended.
- **Dictionary words** and common substitutions are still crackable if short.

- Passwords that appear in common lists like `rockyou.txt` are quickly cracked.
- Use **passphrases** (4+ random words) or long generated strings for better security.
- Use a **password manager** (e.g., KeePassXC) to store unique, long passwords.

Best practices

1. Use 12–16+ characters (or longer).

2. Mix uppercase, lowercase, numbers, and symbols.

3. Avoid single dictionary words and common patterns.

4. Use unique passwords per account.

5. Enable MFA (2FA) on accounts where possible.

6. Use a reputable password manager.

Common attacks (short)

- **Brute force:** tries all combinations — length/complexity increases time exponentially.
- **Dictionary attack:** uses lists of common passwords (e.g., rockyou).
- **Phishing/social engineering:** trick user into revealing credentials.
- **Credential stuffing:** reuse of breached credentials across sites.

Interview questions — suggested short answers

1. **What makes a password strong?**
   Length, unpredictability (entropy), mix of character types, and uniqueness.
2. **Common password attacks?**
   Brute force, dictionary, credential stuffing, phishing.
3. **Why is length important?**
   Each extra character increases possible combinations exponentially, making brute force impractical.
4. **What is a dictionary attack?**
   Using a precompiled list of likely passwords (words, leaked passwords) to guess credentials.
5. **What is multi-factor authentication (MFA)?**
   An authentication method requiring two or more independent credentials (something you know, have, or are).

6. **How do password managers help?**
   Generate and store strong unique passwords, auto-fill credentials, and reduce password reuse.
7. **What are passphrases?**
   A sequence of words (e.g., four random words) that is easier to remember and long & strong.
8. **Common password mistakes?**
   Reuse across sites, predictable substitutions, short length, writing passwords in plain text.

How to reproduce locally (commands)

See `commands.txt` (or read this README).