Password Strength Evaluation Report - Kali Linux

Objective:

Understand what makes a password strong by creating, testing, and evaluating various passwords using Kali Linux tools.

Files to Upload to GitHub:

1. `README.md` (This Report)

Contains:

* Steps followed

* Tool commands

* Results

* Screenshots (to be added)

2. `passwords.txt`

List of passwords tested:

```
12345678

password123

P@ssw0rd

G$8vT&2bL!q9

Th1sIs$Tr0ng!
```

3. `hash.txt`

Password hash of strong password generated using:

```bash
echo 'Th1sIs$Tr0ng!' | openssl passwd -6 -stdin > hash.txt
```

4. Screenshots Folder (Optional but Strongly Recommended)

Create a folder named `screenshots/` and add:

* `cracklib-check.png` (Image showing password test results using cracklib)

* `john-result.png` (Image showing hash not cracked by John)

* `hydra-success.png` (Image showing successful Hydra brute-force attack)

Tools Used:

* `cracklib-check` – to test password strength locally

* `john` (John the Ripper) – to simulate dictionary attacks

* `hydra` – to simulate SSH brute-force attacks

* `seclists` – common password lists

Password Testing Results:

Password	cracklib-check Result	John Cracked?	Notes
12345678	Too simplistic/systematic	Yes	Very weak password
password123	Based on dictionary word	Yes	Common, weak
P@ssw0rd	Based on dictionary word	Yes	Weak (obvious substitution)
G$8vT&2bL!q9	OK	No	Strong
Th1sIs$Tr0ng!	OK	No	Strong passphrase

Hydra SSH Brute Force Result

Command Used:

```bash
hydra -l testuser -P /usr/share/seclists/Passwords/Common-Credentials/10k-most-common.txt ssh://127.0.0.1
```

Result:

```bash
login: testuser
password: 123456
```

SSH password successfully cracked.

Best Practices for Strong Passwords

* Use 12+ characters

* Include uppercase, lowercase, numbers, and symbols

* Avoid dictionary words and common substitutions

* Use passphrases that are long but easy to remember

* Never reuse passwords between accounts

Tips & Learnings

* Even "complex-looking" passwords like `P@ssw0rd` are weak if based on common patterns.

* Strong passwords take too long to crack even with tools like `john` or `hydra`.

* Simple passwords are quickly cracked with public wordlists.

* Testing your password locally helps prevent using weak ones online.

Common Password Attacks (and Tools)

Attack Type	Description	Tool(s)
Brute Force	Tries all combinations	`hydra`
Dictionary Attack	Tries known common passwords	`john`, `hydra`
Weak Password Check	Validates if password follows complexity rules	`cracklib-check`

Summary: Why Complexity Matters

Password complexity drastically improves security. Longer, more random passwords with mixed characters make brute-force and dictionary attacks nearly impossible within a reasonable time.