

Evaluating Password Strength with a Python-Based Cracking Simulation

Nino Aguilar | BSc Networks & Cyber Security | University of Wales Trinity St. David

KEY FINDINGS

- Weak passwords were cracked most often and with the fewest guesses.
- Medium passwords showed greater resistance, but some were still compromised.
- Strong passwords were the most resistant and were not cracked in the representative run.
- Entropy and estimated brute-force time increased from weak to medium to strong.
- Repeated seeded runs showed the same overall pattern across multiple datasets.

AIM

This project aimed to assess and evaluate different levels of password strength using dictionary and brute-force cracking techniques in a safe Python-based simulation. The purpose was to show how weak and predictable passwords increase vulnerability and why stronger password construction matters.

WHY IT MATTERS

- Many users still choose short and predictable passwords.
- Weak password choices can be guessed quickly.
- Understanding password strength helps improve security awareness.

RECOMMENDATIONS

- Avoid short predictable passwords ranging from 5 to 8 characters in length
- Use longer passwords ranging anywhere from 16 to 20n characters in length
- Utilise techniques like pass-phrasing to create passwords to ensure security as well as making it easy to remember
- Utilise password managers to generate and store passwords safe for enhanced security and convenience

EXPERIMENT DESIGN

- Synthetic password dataset generated in Python
- Three categories: Weak, Medium, Strong
- Passwords hashed using SHA-256
- Tested with dictionary attack and bounded brute-force attack
- Results exported to CSV
- Visualised using Matplotlib

CATEGORY	STRUCTURE
WEAK	Short, predictable, word + digits
MEDIUM	Mixed case + digits / separators
STRONG	Long, complex, passphrase-style or mixed-character

ATTACK PROCESS

1. Generate Passwords
2. Hash Passwords
3. Dictionary Attack
4. Brute-Force Attack
5. Record Metrics
6. Export Results

IMPLEMENTATION

- Built entirely in PyCharm Environment
- Main files included: password generator, hashing module, dictionary attack, brute-force attack, result logging, graph visualisation
- Reproducibility supported using seeded datasets
- 10 seeded runs were used to test consistency
- 1 Final seed was used to represent final experiment and results

RESULTS

