Report

Information Assurance and Security

Password Strength Tester

Prepared by :
Arij Kadhi
Tasnim Ben Brahim

Submitted to :
Dr. Manel Abdelkader

Introduction:

Topic: Password Security and Strength Assessment.

Problem: Weak or compromised passwords remain a primary vector for security breaches. Attackers exploit predictable patterns, reused credentials, and insufficient complexity.

Importance: Why robust password policies and user awareness are critical components of cybersecurity.

Project Goal: To introduce a practical tool based on OWASP principles for evaluating password strength, promoting better password habits.

What Defines a Strong Password?

Length: The single most important factor in resisting brute-force attacks. Exponential increase in complexity with length.

Complexity/Character Set Size: Using a mix of uppercase, lowercase, numbers, and symbols increases the search space for attackers.

Unpredictability: Avoiding dictionary words, common phrases, personal information, sequences, and repetitions.

Entropy: A measure of randomness or unpredictability. Calculated typically as **Log2(N^L)** where **N** is the size of the pool of possible characters and **L** is the password length. Higher entropy = stronger password.

Passphrases: Encouraged as an alternative to complex, hard-to-remember strings. By using multiple words, passphrases can easily achieve significant length (the most important factor) while remaining memorable, often resulting in high entropy if words are chosen randomly.

Core Features of the Tester:

1. Input:

Takes a password string as input. Can take a username or related public information (optional) to check against.

2. ***OWASP-Inspired Checks:***

a. **Minimum length:** Enforce a minimum length (e.g., 12 or 15 characters, aligning with modern recommendations).

b. **Complexity:** Check for the presence of multiple character sets:

- Uppercase letters (A-Z)
- Lowercase letters (a-z)
- Numbers (0-9)
- Special characters

c. **Avoidance of Personal Information:** Check if the password contains significant parts of the provided username, email, or other specified public info.

d. **Commun passwords check:** Check against a small, curated list of notoriously common and weak passwords (e.g., "password", "123456", "qwerty").

e. **Sequence check:** Detect common keyboard sequences ("qwerty", "asdfg", "12345")

f. **Repetition check:** detect heavily repeated characters (“fffff” , “6666”)

3. ***Feedback:*** Provide specific feedback on which rules passed or failed.

4. ***Final Verdict:*** Output a clear "Thumbs Up" (Strong) only if all configured checks pass, otherwise "Thumbs Down" (Weak) with reasons.

Main components:

1. Input Interface: Describe how the user interacts (command-line prompt, simple web form).

2. Password Input Module: Receives the password string.

3. Optional Context Input Module: Receives username/email ..

4 . Validation Engine: The core logic containing individual check functions.

- check_length()
- check_character_types() (using regex or character iteration)
- check_against_context() (simple substring checks)
- check_common_passwords() (lookup in a predefined list/set)

- check_sequences() (regex or iterative comparison)
- check_repetitions() (regex or iterative comparison)
- Feedback Aggregator: Collects results from each check function.

5. Output Module: Displays detailed feedback and the final "Thumbs Up" / "Thumbs Down" verdict.

Functional Flow:

1. User types password into the UI input field.
2. User provides username/contextual info. (optional)
3. UI sends password (and context) to the Validation Engine on each keystroke.
4. Validation Engine iterates through enabled Rule Modules, passing the password/context.
5. Each Rule Module performs its specific check and returns a pass/fail status and potentially a particular feedback message.
6. Validation Engine collects all results.
7. Feedback Mechanism receives results and updates the UI:
    a. Shows specific failure messages for any failed rules.
    b. If all required rules pass, it displays the "Strong" indicator. (for example)
    c. If any rule fails, it displays the "Weak" indicator and the relevant failure messages.
8. The feedback is updated in near real-time as the user types.

Existing solution :

1. **Online Checkers:** Websites that let you paste a password for immediate feedback (Kaspersky, security.org).
    a. **Pros:** Very convenient and accessible.
    b. **Cons**: Major privacy risk (never use real passwords), often lacks transparency on why a password fails specific standards, may not align perfectly with OWASP.
2. **Libraries/Frameworks:** Code libraries (like Dropbox's zxcvbn) that developers integrate into applications.
    a. **Pros:** Can offer sophisticated analysis (entropy, pattern matching, estimated crack time), saves development effort.

   b. **Cons**: It might focus more on entropy scores than strict OWASP rule compliance, still requires integration effort.
3. **Built-in OS/Browser Checks:** Basic strength

Advantages and Limitations of This Project :

- **Advantages:**
  - **Simplicity:** Easy to understand, implement, and use.
  - **Clear OWASP Alignment:** Directly implements core, understandable rules based on recognized standards.
  - **Educational:** Helps users understand *why* certain password characteristics are important.
  - **Deterministic:** Provides a clear pass/fail ("Thumbs Up") based on explicit criteria, avoiding ambiguous scores.
  - **Offline Potential:** Can be implemented as a standalone script/application, avoiding online privacy risks.
- **Limitations:**
  - **No True Entropy Calculation:** Doesn't provide a quantitative measure of strength (like bits of entropy or estimated crack time).
  - **Basic Pattern Matching:** Doesn't detect keyboard walks, complex sequences, or substitutions (e.g., P@ssw0rd might pass complexity but is a known pattern).
  - **Limited Dictionary:** Relies on a potentially small list of common passwords.
  - **Rudimentary Context Check:** Basic username substring check is easy to bypass. Doesn't check against other personal info.
  - **No Adaptive Checks:** Doesn't adjust requirements based on password length (e.g., a very long passphrase might be secure even without meeting all complexity rules).

Conclusion:

This Password Strength Tester provides a practical tool for evaluating passwords against fundamental OWASP-aligned security principles. Enforcing rules for length, complexity, and uniqueness (avoiding username/common passwords) guides users toward creating more robust credentials. While not as sophisticated as entropy-calculating tools like zxcvbn, its clear pass/fail

mechanism based on essential security hygiene rules offers significant educational value and encourages better password practices, including the use of longer passphrases. It serves as a solid foundation for understanding and implementing basic password security checks.