



Password Strength Analyzer with AI

GenAI-powered Password Strength Analyzer
with Security Insights

By **Bit_Bandits**
(Third Year - PICT, Pune)



Current Scenario



With increasing cyber threats, **weak and easily guessable passwords** remain a primary cause of data breaches. Many users rely on predictable patterns or **reuse passwords** across multiple platforms, making them easy targets for attackers.

Existing **password strength meters** often **provide vague feedback** without explaining vulnerabilities, leaving users unaware of real security risks. This gap in awareness and lack of intelligent guidance motivated us to develop a more insightful and AI-driven password analysis tool.



Our Solution

- Develop an AI-powered Password Strength Analyzer using **GenAI** and **Machine Learning**.
- Estimates **time-to-crack** and **strength-score** for each password and identifies vulnerabilities.
- Provides **AI-generated feedback** with **personalized security insights**.
- Works **entirely offline**, ensuring user privacy and security.
- Supports **multiple languages** for wider accessibility.



Existing Solutions and Challenges Faced

- **Current solutions leverage only single factors while checking strength.**

e.g. **zxcvbn** checks for pattern matching and entropy but ignores real-world cracking techniques.

- **Existing password checkers only verify against known breached databases.**

e.g. **Have I Been Pwned** (HIBP) only flags passwords already leaked, but doesn't analyze similar weak patterns.

- **Strength assessment is often vague, with no real-world time-to-crack estimation.**

e.g. **Google Password Manager** labels passwords as "weak" or "strong" without estimating cracking time.

- **No personalized feedback on why a password is weak and how to improve it.**

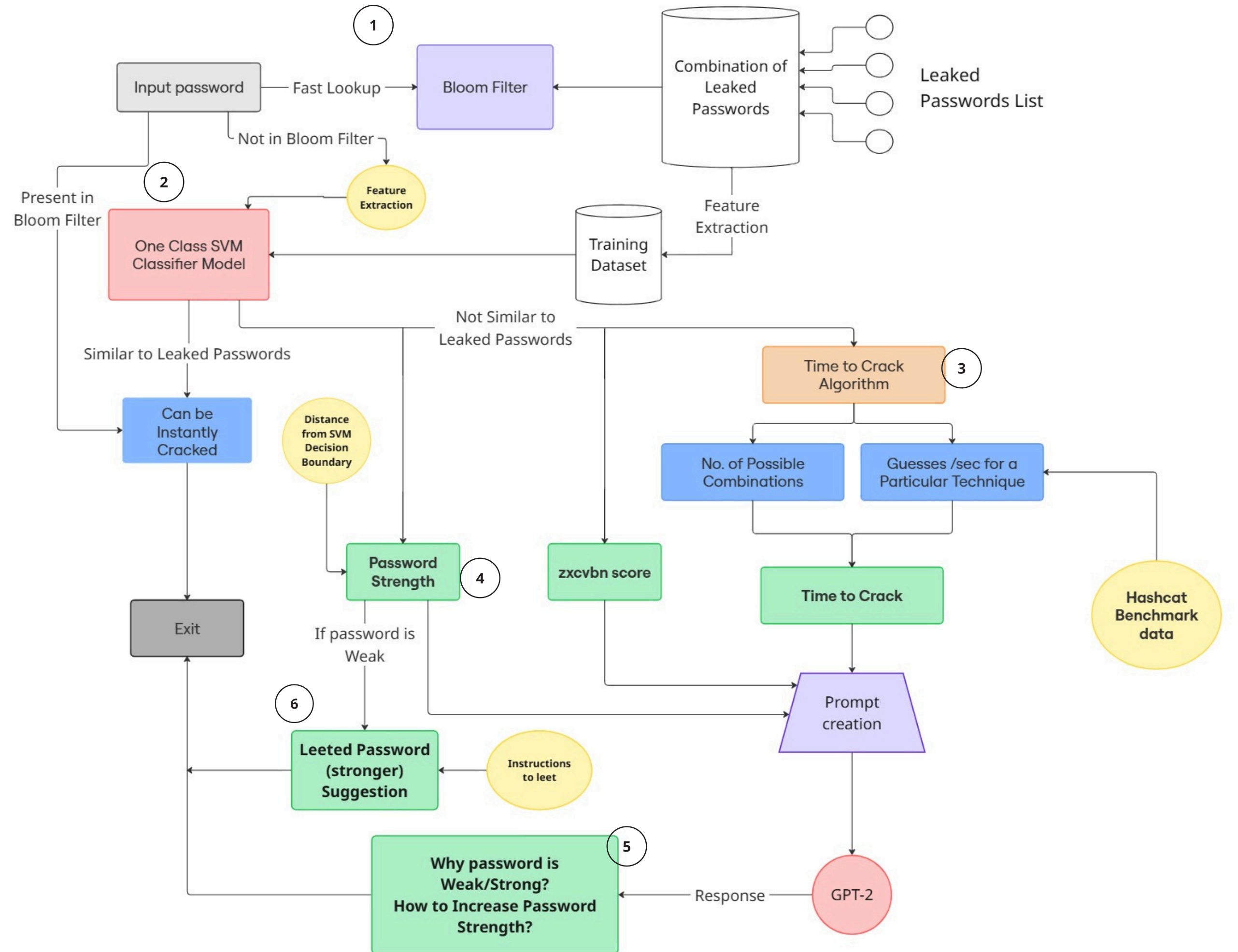
e.g. **NCSC Guidance** suggests general best practices but doesn't provide user-specific improvements.

In conclusion, we can say that existing password strength analyzers rely on limited factors, outdated rules, and predefined datasets, whereas our solution leverages AI-driven analysis, real-world attack simulations, and personalized feedback to provide a more accurate and adaptive security assessment.

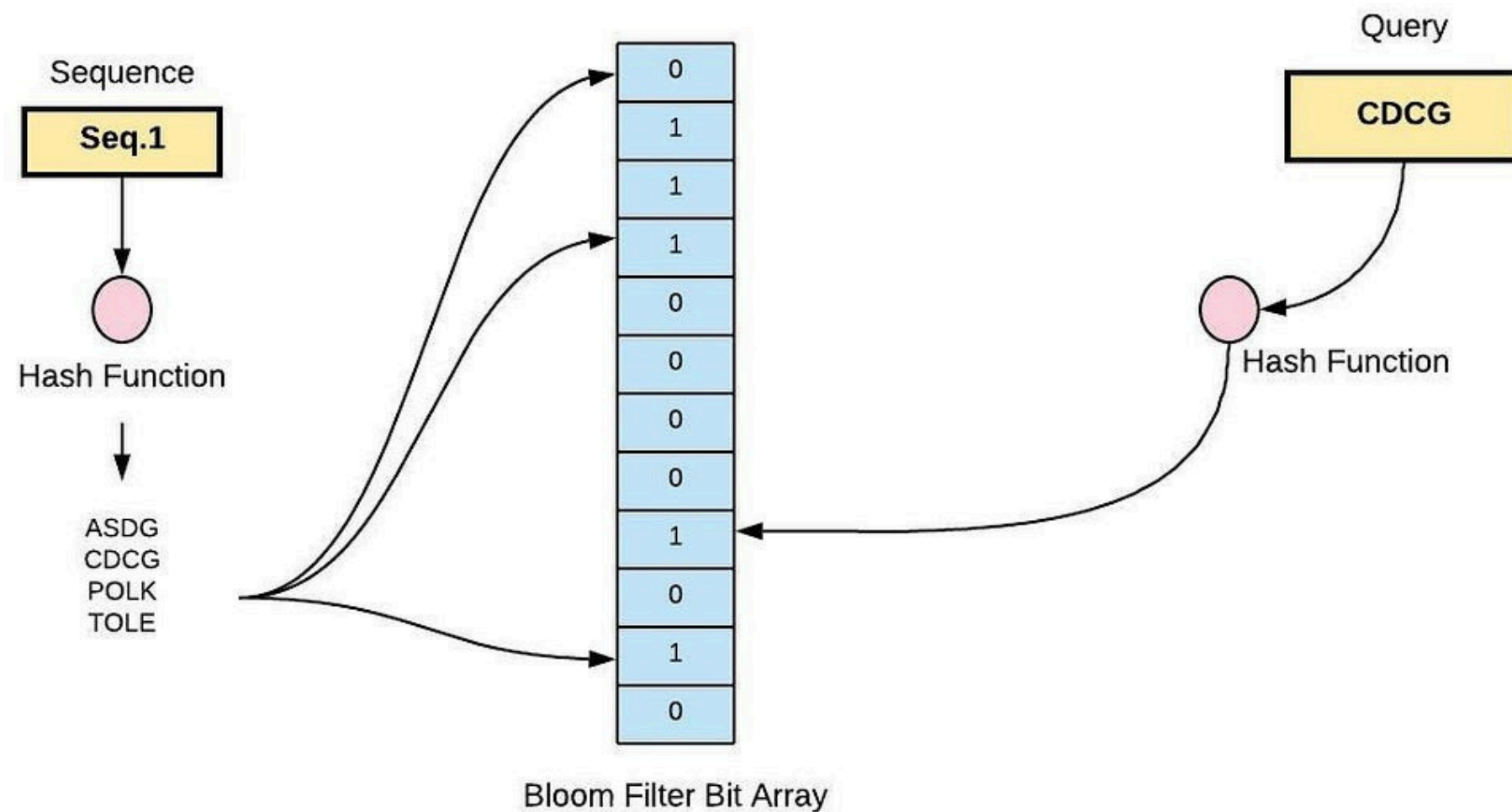
System Design

Important Phases of the System:

1. Fast Lookups & Bloom Filters
2. One-Class SVM Classifier
3. Time-to-Crack algorithm
4. Strength and zxcvbn score
5. GPT-2 prompt and output
6. Leeting Weak Passwords



I. Fast Lookups & Bloom Filters



source: [Lucky Sharma - Medium Blog](#)

Combining Leaked Datasets

We merge multiple leaked password lists (RockYou, HIBP, LinkedIn and Facebook Leaks, etc) into a single dataset to improve coverage.

Building the Bloom Filter

We hash each password using multiple hash functions and set corresponding bits in a large bit array (The Bloom Filter).

Fast Lookup Process

To check if a password is leaked, we apply the same hash functions to the input password and verify if all corresponding bits of the Bloom Filter are set.

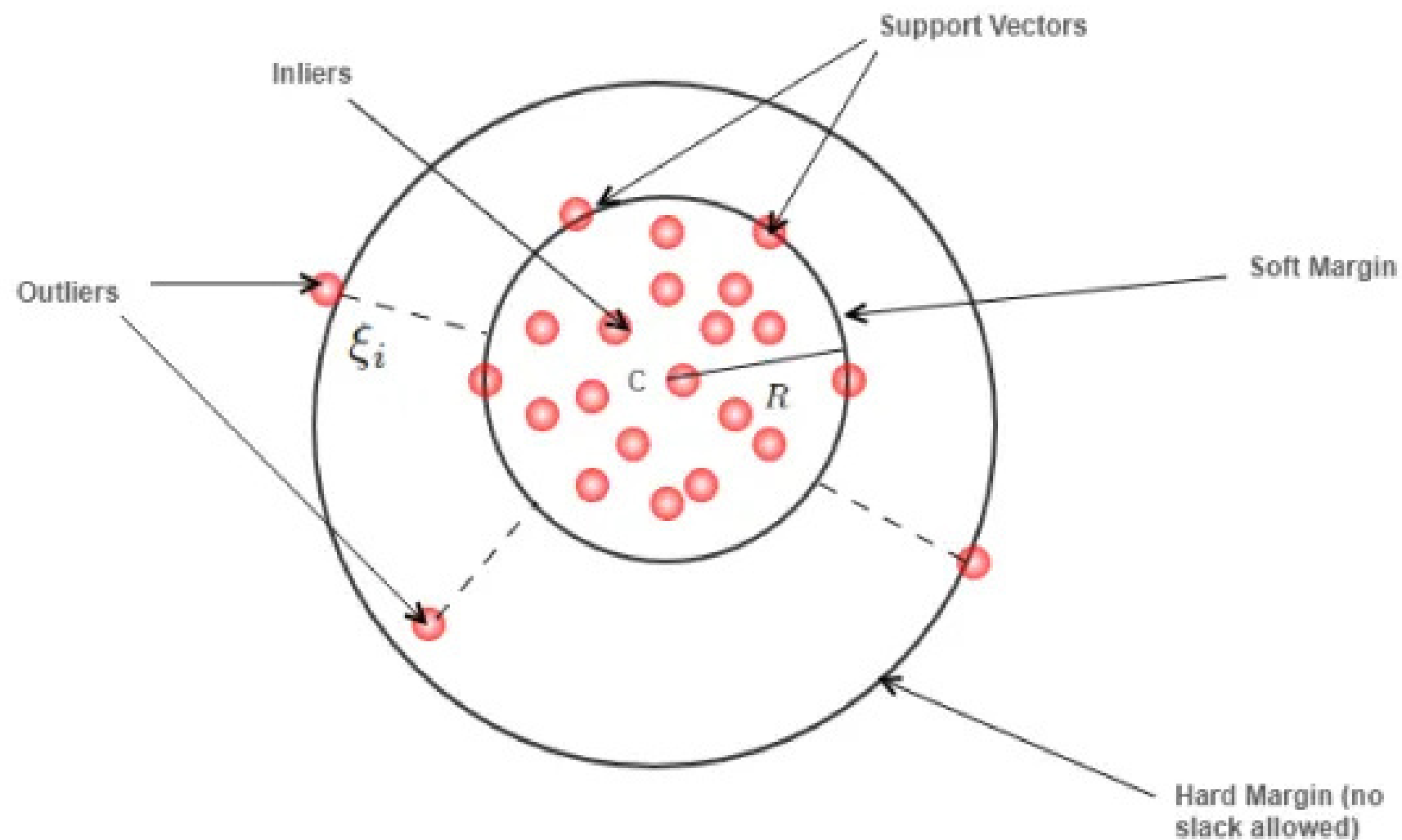
Why It's Fast & Efficient

Bloom filters use constant time ($O(1)$) lookups and require significantly less memory than storing raw passwords.

False Positives but No False Negatives

A password flagged as leaked might be a false positive, but if it is flagged as not leaked, it's definitely safe.

2. One-Class SVM Classifier



source: [Garima - Medium Blog](#)

Note: One Class SVM prediction happens only when input password is not found in the leaked password's dataset by the Bloom Filter.

What is One-Class SVM?

A machine learning model that learns patterns from leaked passwords and detects similar weak passwords.

How It Works?

1. **Training on Leaked Passwords** – Model is trained using a dataset of known leaked passwords.
2. **Feature Extraction**
3. **Classification** – When a new password is entered, SVM checks if it follows similar weak patterns.

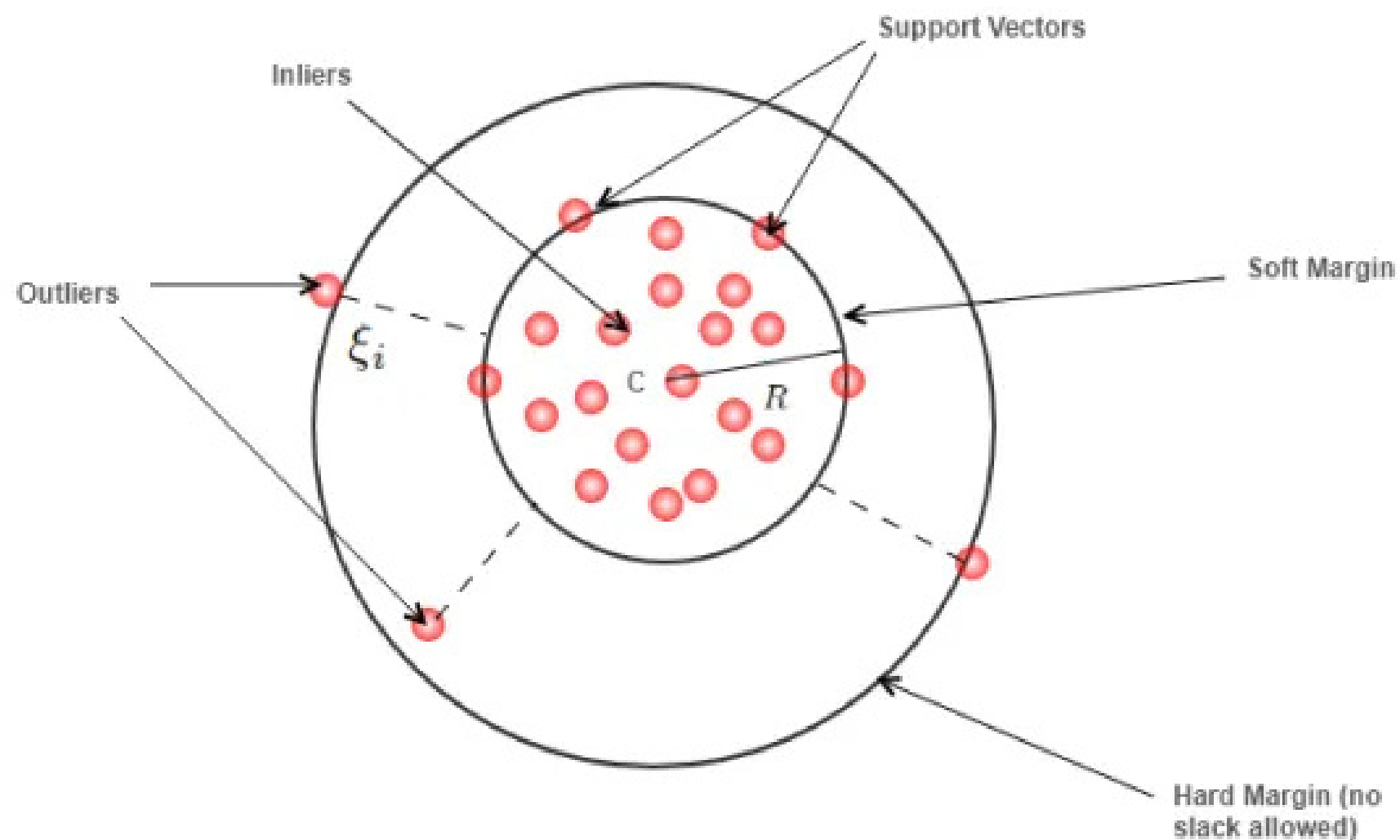
Model Predictions

- Similar to Leaked Passwords → Weak Password
- Not Similar → Potentially Strong Password

Why One-Class SVM?

- Detects weak passwords even if they haven't been leaked.
- Identifies passwords following predictable patterns.

2. One-Class SVM Classifier (preprocessing)



source: [Garima - Medium Blog](#)

Here, **inliners** are the leaked passwords on which the model trains, where's **outliers** are the strong passwords unknown to the dataset

Training Dataset

The model is trained using the merged leaked password dataset, treating all passwords as a single class.

Feature Extraction

Each password is converted into meaningful numerical features:

1. **Length** – Shorter passwords are easier to crack.
2. **Character Diversity** – More unique characters increase strength.
3. **Character Ratio** – Balance of alphabets, numbers, symbols.
4. **Shannon Entropy** – Measures randomness and complexity.
5. **N-grams** – Identifies common structures in weak passwords.

Training the One-Class SVM


The model learns the characteristics of weak passwords and flags new passwords that follow similar patterns.

Generalizes Beyond Known Leaked Passwords

Unlike direct matching (e.g., Bloom filters), this method detects weak passwords that haven't been leaked yet but follow risky patterns.

3. Time-to-Crack Algorithm

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	3 secs	6 secs	9 secs
5	Instantly	4 secs	2 mins	6 mins	10 mins
6	Instantly	2 mins	2 hours	6 hours	12 hours
7	4 secs	50 mins	4 days	2 weeks	1 month
8	37 secs	22 hours	8 months	3 years	7 years
9	6 mins	3 weeks	33 years	161 years	479 years
10	1 hour	2 years	1k years	9k years	33k years
11	10 hours	44 years	89k years	618k years	2m years
12	4 days	1k years	4m years	38m years	164m years
13	1 month	29k years	241m years	2bn years	11bn years
14	1 year	766k years	12bn years	147bn years	805bn years
15	12 years	19m years	652bn years	9tn years	56tn years
16	119 years	517m years	33tn years	566tn years	3qd years
17	1k years	13bn years	1qd years	35qd years	276qd years
18	11k years	350bn years	91qd years	2qn years	19qn years



> Hardware: 12 x RTX 4090 | Password hash: bcrypt

source: [Martin Brinkmann - ghacks.net](#)

Rough estimates of Time to Crack based solely on character diversity of the password keeping a specific hashing technique in mind

Calculate Total Guesses:

Formula: **Character Set Size ^ Password Length**

Attack Speeds for Different Hashing Algorithms:

We use the [Hashcat benchmark data](#) to determine guesses per second of various hashing techniques (Bcrypt, Scrypt, SHA-256, Argon2, etc.)

Compute Time to Crack for Each Hashing Algorithm:

Formula: **(Total Guesses) / (Attack Speed)**

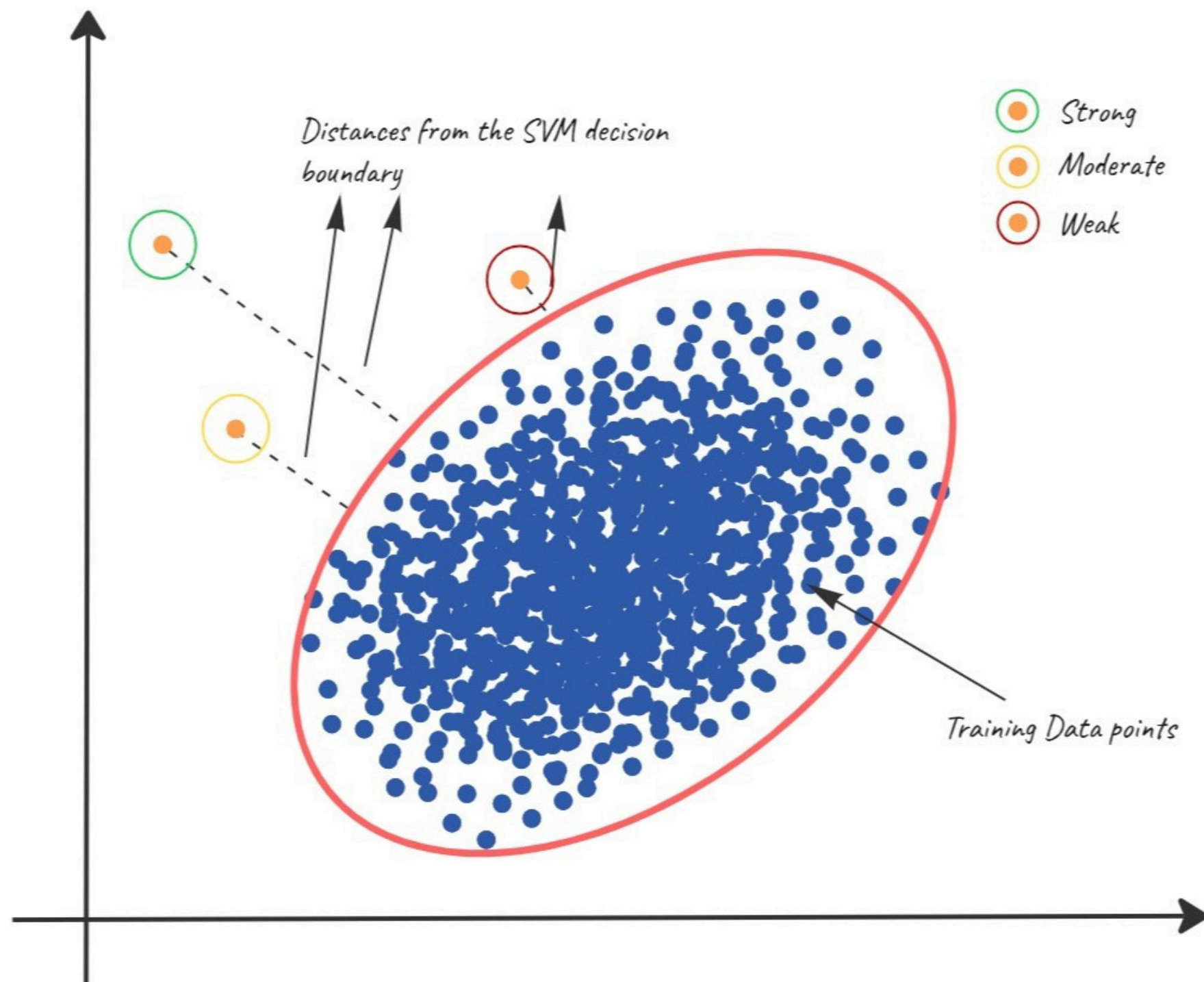
Convert Time to Human-Readable Format and display:

Display in years, months, days, hours, minutes, seconds format.

Key Details:

- Time to crack is calculated for different hashing techniques (bcrypt, SHA-256, etc.).
- Time to crack is only computed if the SVM model determines the password is not similar to leaked ones.

4. Strength and zxcvbn Score



source: [Soham Phatak - miro Board](#)

Strength Score (OC-SVM Distance)

- We calculate the strength score of an uncommon password by checking how far away it lies from the decision boundary of the One-class SVM model.
- A higher distance from the decision boundary indicates a stronger password.
- The distance is then normalized on a 0-100 scale to get the strength score and better interpretability.

ZXCVBN Score

- zxcvbn is an open-source password strength estimator developed by Dropbox which analyzes passwords using pattern-matching and entropy calculations.
- The zxcvbn score is a numeric value (0 to 4) that represents the estimated strength of a password
- It produces a score between 0 (very weak) to 4 (very strong) by analyzing password patterns, entropy, and dictionary-based weaknesses.
- Helps identify common words, predictable sequences, and keyboard patterns.

5. GenAI integration for personalized password insights

Why GPT-2

No Internet Dependency – Newer models typically require cloud-based APIs, introducing privacy concerns. GPT-2 can be deployed locally, ensuring privacy.

Efficiency & Speed – Advanced GenAI models require more computation, making them slower for real-time password analysis. GPT-2 is lightweight and faster, ensuring low-latency responses

Fine-Tuning Capability – Unlike API-restricted models, GPT-2 can be fine-tuned on password-related datasets, allowing it to generate customized security insights based on time-to-crack, strength scores, and common patterns.

Lower Hardware Requirements – Running advanced models locally requires high-end GPUs. GPT-2 can work efficiently on consumer-grade hardware, making it practical for real-world applications.

Structured Yet Creative Outputs – Despite being smaller, GPT-2 is still capable of generating insightful and coherent feedback, making it ideal for real-time password strength analysis without unnecessary complexity.

Overview

- GPT-2 is used to provide personalized insights to the input password based on its calculated time to crack, Zxcvbn score, and SVM-based strength score.
- It is asked to generate human-readable insights on why the password is strong/weak and how it can be made stronger still.

GenAI Response

- Explaining Weaknesses:** Identify vulnerabilities like dictionary words, common patterns, and low entropy.
- Predicting Real-World Crackability:** Estimates how easily hackers can brute-force or guess the password.
- Provide Strengthening Suggestions:** Suggests randomization, leeting, and increased length to enhance security.

Why is this unique?

- Moves beyond static strength meters by using AI-generated contextual feedback.
- Educates users with detailed insights instead of just assigning a strength label.
- Ensures personalized recommendations for stronger passwords, improving real-world security.

6. Leeting Weak Passwords

pict123

Weak

characters containing: Lower case Upper case Numbers Symbols

Time to crack your password:
3.29 minutes

P!c7@lZE4

Very Strong

characters containing: Lower case Upper case Numbers Symbols

Time to crack your password:
283 years

source: passwordmonster.com

Note: This website calculates time to crack based solely on comparing the entered password against **common password dictionaries**

What is Leeting?

- Leeting (from "leet speak") means to replace characters with other characters which are visually similar to them.
- Helps make passwords less predictable and harder to crack.
- We leet only weak passwords in our implementation.

Steps We Follow:

1. **Leet Substitution** – Replace alphabets & numbers with symbols

Example: p → P, i → !, t → 7, 1 → @, 2 → Z, 3 → E

2. **Add Random Symbols** – Insert symbols before/after keywords

Final Password Example: "P!c7@lZE4"

- We add Random Symbols to make leeted password stronger, as normal leeting is still easily crackable.

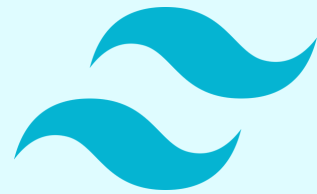
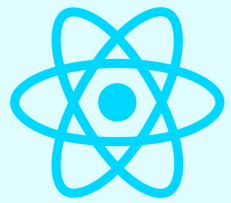
Impact on Strength:

- Increases entropy, making the password harder to guess
- Breaks dictionary & pattern-based attacks by avoiding common structures
- Stronger against brute-force attacks due to more variations

This method significantly strengthens passwords against modern cracking techniques.

Tech Stack to Use

Frontend



Backend



Deployment



Model-Building



HUGGING FACE



Tech Stack to Use

Frontend Technologies:

- **React** - JavaScript library for building the user interface.
- **HTML** - To structure the web page.
- **Tailwind CSS** - A Styling tool to make web pages look good with pre-built styles.

Backend Technologies:

- **Flask** - A lightweight Python framework which will handle requests from the frontend and process them.
- **Python** - The main programming language, which will help write the backend logic as well as create the ML model.

Model-Building Technologies:

- **GPT-2 (from Hugging Face)** - A pre-trained GenAI model that can generate text based on an input prompt.
- **Scikit-Learn** - A library which contains pre-built ML models, like classifiers and regressors.
- **NumPy** - A Python library which helps with numerical calculations, especially with large datasets.
- **Pandas** - A Python library which helps with handling and analyzing data more easily.

Deployment Technologies:

- **AWS (Amazon Web Services)** - A cloud platform to host your application.
- **Docker** - Packs your project into containers so it runs the same way on any system.
- **Git** - A version control tool to track changes in your code and collaborate with others.

Multi-Lingual Support

USECASES

- **Corporate Users** – Will help global organizations implement password security policies across different languages, ensuring compliance and better employee awareness.
 - **Individual Users** – Will enable non-English speakers to understand password strength analysis and receive security recommendations in their preferred language.
 - **Government & Regulatory Bodies** – Will be able to assist in implementing cybersecurity awareness programs and enforcing regional cybersecurity regulations.
 - **Educational Institutions** – Can help students and educators learn about password security without language barriers, improving cybersecurity education.
-

HOW WILL IT WORK

1. **Language Selection Will Be Enabled** – Users will be able to choose their preferred language for password analysis reports.
2. **GPT-2 Will Be Fine-Tuned** – The model will be adapted to generate security insights in multiple languages while maintaining accuracy.
3. **Localized Feedback Will Be Provided** – Password strength explanations and improvement suggestions will be clear and culturally relevant.
4. **Seamless User Experience** – The interface will dynamically adjust to display reports in the selected language.
5. **Accessibility Will Be Improved** – Expands the tool's reach, making it useful for non-English speakers and global organizations.

Bulk Password Analysis

USE CASES IN CORPORATE

- **Employee Credential Security** – Organizations can analyze bulk passwords to identify weak or compromised credentials and enforce stronger password policies.
 - **Data Breach Prevention** – Helps detect leaked or easily crackable passwords before they become an entry point for cyberattacks.
 - **Compliance & Policy Enforcement** – Ensures passwords meet security standards like NIST, GDPR, and ISO 27001.
 - **Integration with IT Systems** – Can be integrated into corporate login systems to enforce real-time password strength validation.
-

HOW WILL IT WORK

1. **Batch Processing** – Multiple passwords will be analyzed simultaneously for efficiency.
2. **Time to Crack** – Each password's cracking time will be calculated based on the selected hashing technique (option will be given to the user to select).
3. **Strength Scoring** – Passwords will be evaluated using OC-SVM distance and Zxcvbn score.
4. **Results Will Be Exportable** – The output will be structured in downloadable CSV/JSON format for further analysis.
5. **Scalability Will Be Ensured** – The system will be optimized to handle large datasets efficiently with minimal delays.

Conclusion

- This project revolutionizes password strength analysis by integrating Machine Learning, GenAI, and cryptographic benchmarking to provide a comprehensive and intelligent evaluation of password security. Unlike traditional solutions that rely on static rule-based checks, this approach:
 1. **Combines Multiple Factors** – Evaluates entropy, common patterns, time-to-crack, and ML-based similarity detection.
 2. **Uses One-Class SVM & Bloom Filters** – Efficiently detects leaked passwords without exhaustive lookups.
 3. **Benchmarks Hashing Algorithms** – Displays time-to-crack estimates based on real-world attack speeds.
 4. **Leverages GenAI for User Education** – Provides contextual feedback and stronger password suggestions in real time.
- By integrating these elements, the project not only assesses password security but actively enhances user awareness and guides them toward stronger passwords.
- This multi-layered approach ensures higher accuracy, deeper insights, and real-world applicability, making it a powerful and scalable solution for modern password security challenges.



Our Team



Prafulla Bhand

Fullstack Developer



Payal Burade

Frontend Developer



Sohan Choudhary

Backend Developer
ML Engineer



Ayush Khairnar

Backend Developer
GenAI Expert



Soham Phatak

ML Engineer
UI/UX Designer





Thank you

