Password Cracking & Credential Attack Suite
(Project Documentation)

1. Project Overview / Description

This project focuses on designing and developing a practical toolkit for password policy testing and credential security assessment.

Weak passwords remain one of the most exploited vulnerabilities in cybersecurity. Attackers use dictionary attacks, credential dumping,

and brute-force techniques to compromise systems.

This project provides an ethical, controlled environment to understand how password cracking works, how credentials are stored,

and how security teams can reinforce authentication mechanisms.

The toolkit includes:

- Dictionary generator

- Hash extraction (Linux shadow & Windows SAM)

- Brute-force simulation module

- Password strength analyzer

2. Practical Motivation

Passwords remain the first line of defense for user authentication. However, poor password practices lead to:

- Account takeovers

- Privilege escalation

- Data breaches

- Credential stuffing attacks

This project enables hands-on learning of:

- How password hashes are stored and protected

- How attackers attempt to crack passwords

- How strong password policies are enforced

- How defenders audit and strengthen authentication systems

3. Project Objectives

1. Develop a dictionary generator for password testing.

2. Extract password hashes from Linux shadow files and Windows SAM database (ethical environment only).

3. Build a brute-force simulation engine to test password robustness.

4. Analyze password strength based on complexity, entropy, and predictability.

5. Generate a detailed audit report on password vulnerabilities and mitigation steps.

4. Practical Scope of the Project

A. Dictionary Generator:

- Generate custom wordlists based on patterns (name+DOB, keyboard patterns, common passwords, hybrid lists).

- Include mutation rules (leet-speak, uppercase variations, appended numbers).

B. Hash Extraction Module:

- Extract /etc/shadow entries from Linux.

- Extract SAM and SYSTEM registry hives from Windows (offline method).

- Identify hashing algorithms used (MD5, SHA-512, NTLM, etc.).

C. Brute-Force Simulator:

- Simulate brute-force cracking attempts.

- Support incremental mode (a–z, A–Z, 0–9, symbols).

- Provide estimated time-to-crack metrics.

- Demonstrate ethical cracking only on controlled lab environments.

D. Password Strength Analyzer:

- Check complexity requirements.

- Estimate entropy.

- Identify dictionary-based weaknesses.

- Provide improvement recommendations.

E. Report Generation:

- Summary of weak passwords found.

- Simulation results.

- Recommended password policies.

5. Tools & Technologies Used

Programming Languages:

- Python (recommended)

- Bash (optional)

Modules/Tools Used:

- hashlib (hashing operations)

- crypt / passlib (Linux password hash processing)

- reg.exe (Windows hive export)

- John the Ripper / Hashcat (optional reference tools, not mandatory)

Documentation Tools:

- Word / Google Docs

- Draw.io (flowcharts and architecture diagrams)

6. Practical Techniques Implemented

Red Team Techniques:

- Password dictionary generation and mutation

- Credential harvesting in a controlled lab

- Hash cracking simulation

- Brute-force and rule-based cracking methodology

Blue Team Techniques:

- Understanding secure password storage

- Detecting weak passwords using audit tools

- Implementing strong password policies

- Monitoring and mitigating brute-force attempts

7. Workflow / Architecture (Practical Explanation)

STEP 1: User Input

- Provide username list, password samples, or hash files.

STEP 2: Dictionary Generation

- Build wordlists based on patterns or imported data.

STEP 3: Hash Extraction (Optional Demo)

- Extract Linux shadow hashes or Windows SAM offline.

STEP 4: Brute-Force & Dictionary Attack Simulation

- Test password resistance using dictionary or brute-force methods.

STEP 5: Password Strength Analysis

- Evaluate password complexity, entropy, and predictability.

STEP 6: Report Generation

- Include weak passwords, cracking times, risks, and mitigation suggestions.

8. Flowchart (Text Version)

START

↓

Input Data → Generate Dictionary

↓

Extract Password Hashes

↓

Simulate Dictionary/Brute-Force Attack

↓

Analyze Password Strength

↓

Generate Final Audit Report

↓

END

9. Expected Practical Output

Deliverables include:

- Working dictionary generator

- Hash extraction demonstration

- Brute-force simulation results

- Password strength analysis output

- Complete security audit report

Expected Output Examples:

- Generated wordlist file

- Extracted hash list

- Estimated time required to crack passwords

- Weak passwords detected and rated by severity

10. Learning Outcomes

This project enables understanding of:

- How passwords are stored, hashed, and protected

- Ethical password cracking methodologies

- Authentication security auditing

- Red team vs blue team password assessment techniques

- Best practices for secure password creation and enforcement

11. Project Deliverables

1. Project documentation (Word/PDF)

2. Fully working password cracking toolkit (simulation only)

3. Screenshots of all modules running

4. Flowcharts and architecture diagrams

5. Final presentation (PPT)