Password Cracker Internship Project

By Purab Awasthi

Intern ID: ITID0485

Project Overview

This project is a Python-based password cracker tool designed to demonstrate common password cracking techniques such as dictionary attacks and brute-force attacks. It supports cracking hashed passwords using popular hash functions like MD5, SHA-1, and SHA-256.

Features

- Supports multiple hash algorithms: MD5, SHA-1, and SHA-256.
- Performs dictionary attacks using a user-provided wordlist.
- Performs brute-force attacks by generating password combinations dynamically.
- Uses multi-threading to speed up the cracking process.
- Accepts command-line inputs for flexible use.
- Allows customization of brute-force password length and character sets.

How to Use

1. Dictionary Attack
   Use a wordlist file (e.g., rockyou.txt) to attempt cracking a hashed password.
   python password_cracker.py <hash> <hash_type> --wordlist wordlist.txt

2. Brute-force Attack
   Attempt to crack a hashed password by generating all possible passwords within a length range.
   python password_cracker.py <hash> <hash_type> --min 1 --max 4

Example

Cracking the MD5 hash of "123456":

python password_cracker.py e10adc3949ba59abbe56e057f20f883e md5 --min 1 --max 6

Key Concepts Learned

- Understanding of cryptographic hash functions and their vulnerabilities.

Password Cracker Internship Project

- Implementation of dictionary and brute-force password cracking.
- Use of multi-threading to optimize performance.
- Command-line argument parsing in Python.
- Basic cybersecurity and ethical hacking concepts.

Requirements

- Python 3.x
- Standard Python libraries: hashlib, argparse, itertools, threading, queue, string

Notes

- Make sure the wordlist file (if used) is present in the working directory or provide the full path.
- Brute-force attacks can be slow for large password lengths or complex character sets.
- This tool is for educational purposes and ethical hacking only.

Code: password_cracker.py

```
import hashlib
import argparse
import itertools
import string
import threading
from queue import Queue

HASH_FUNCTIONS = {
    'md5': hashlib.md5,
    'sha1': hashlib.sha1,
    'sha256': hashlib.sha256
}

def worker(queue, target_hash, hash_func, found_flag):
    while not queue.empty() and not found_flag[0]:
        password = queue.get()
        hashed = hash_func(password.encode()).hexdigest()
        if hashed == target_hash:
            found_flag[0] = True
            print(f"\n[FOUND] Password found: {password}")
        queue.task_done()

def dictionary_attack(target_hash, hash_func, wordlist_path, threads=4):
    print("[*] Starting dictionary attack...")
    found_flag = [False]
    queue = Queue()

    try:
```

Password Cracker Internship Project

```
        with open(wordlist_path, 'r', encoding='utf-8', errors='ignore') as file:
            for word in file:
                queue.put(word.strip())
    except FileNotFoundError:
        print(f"[!] Wordlist not found: {wordlist_path}")
        return

    for _ in range(threads):
        thread = threading.Thread(target=worker, args=(queue, target_hash, hash_func, found_flag))
        thread.start()

    queue.join()
    if not found_flag[0]:
        print("[-] Password not found in wordlist.")

def brute_force_attack(target_hash, hash_func, min_len, max_len, charset, threads=4):
    print("[*] Starting brute-force attack...")
    found_flag = [False]
    queue = Queue()

    for length in range(min_len, max_len + 1):
        for combination in itertools.product(charset, repeat=length):
            queue.put(''.join(combination))

    for _ in range(threads):
        thread = threading.Thread(target=worker, args=(queue, target_hash, hash_func, found_flag))
        thread.start()

    queue.join()
    if not found_flag[0]:
        print("[-] Password not found using brute-force.")

def main():
    parser = argparse.ArgumentParser(description="Python Password Cracker - By Inlighn Tech")
    parser.add_argument("hash", help="Target hashed password")
      parser.add_argument("hash_type", choices=HASH_FUNCTIONS.keys(), help="Hash algorithm (md5,
sha1, sha256)")
    parser.add_argument("--wordlist", help="Path to wordlist file (dictionary attack)")
           parser.add_argument("--min", type=int, default=1, help="Minimum password length
(brute-force)")
           parser.add_argument("--max", type=int, default=4, help="Maximum password length
(brute-force)")
            parser.add_argument("--charset", default=string.ascii_lowercase + string.digits,
help="Characters for brute-force")
    parser.add_argument("--threads", type=int, default=4, help="Number of threads to use")

    args = parser.parse_args()
    hash_func = HASH_FUNCTIONS[args.hash_type]

    if args.wordlist:
```

```
        dictionary_attack(args.hash, hash_func, args.wordlist, args.threads)
    else:
        brute_force_attack(args.hash, hash_func, args.min, args.max, args.charset, args.threads)

if __name__ == "__main__":
    main()
```

Optional Sample Wordlist: sample_wordlist.txt

password
123456
qwerty
letmein
admin
welcome

How to Run

Open your terminal or PowerShell, navigate to the project folder and run:

Dictionary attack example:
python password_cracker.py e10adc3949ba59abbe56e057f20f883e md5 --wordlist sample_wordlist.txt

Brute-force attack example:
python password_cracker.py e10adc3949ba59abbe56e057f20f883e md5 --min 1 --max 6