Assignment 1, Part 2

Instructions
Three files are given inside the zip. Spend time trying to compromise passwords. Use whatever language/libraries you use to compromise the passwords, but do not use password crackers. It is ok to use existing data breaches from the internet but the use of rainbow tables is not allowed.

What to submit:

A zip file named as your NetID (e.g. ab123) with:

Three text files (one for each password list) where each line contains the original line from the password file and the plaintext password separated by a space. No more than 100 lines in each file.
Files containing any source code used.
A pdf file explaining:

- The technique you used to obtain the passwords
- Other techniques you considered
- How were the passwords stored? Compare the difficulty in cracking passwords protected with each type of storage.

*Rubric (Very necessary for you!) :

+40 submitting working code which we can build and run, which follow the assignment directions
+20 turning in password files in the correct format and following all other submission rules
+15 points cracking at least 1 password from one database breach
+10 points cracking at least 1 password from two database breaches
+4 points cracking at least 1 password from three database breaches
+5 points cracking 100 passwords total
+4 points cracking 200 passwords total
+2 point cracking 300 passwords total

_

Overview

Wordlist: weakpass_3w.txt (https://weakpass.com/wordlist/1950)
Script Language/ Version: Python/ 3.11.1

File hierarchy:
Password Lists
```
├── crack.py
├── formspring
│   └── formspring.txt
├── linkedin
│   └── SHA1.txt
├── weakpass_3w.txt
└── yahoo
    ├── copylines.py
    ├── password.file
    └── pure_pass.json
```

Source code: Attached (crack.py and copylines.py)
Run: Download wordlists, place files according to hierarchy and run.

General Procedure

1. For formspring.txt (formspring) and SHA1.txt (linkedin) it is encoded while the password.file (yahoo) is available in text.
2. I used https://www.tunnelsup.com/hash-analyzer/#google_vignette to determine the hashing method for the files that were hashed.
3. Then I brute forced the passwords. If the passwords were originally hashed then I hashed the word in the wordlist then compared. Otherwise I directly compared the wordlist word to password.
4. Then printed the passwords that were found with the original line in the desired format.

*Specific procedures for password.file (yahoo). I wrote copylines.py which would copy all the lines containing the passwords and write a json file {password: line format} to retain the original line.

Techniques Considered

1. Tried to use rockyou.txt -> Got only a few results
2. I noticed that formspring passwords had varying letter cases. Not all of them were in lower case like rockyou.txt. So I tried to make variations to rockyou.txt by flipping cases randomly. -> Took too long, wasn't much of use
3. Perhaps weakpass_3w.txt is not the best for this procedure since it is specifically for wifi passwords, however 8-24 character length seems like a good length for a reasonable person setting their passwords.

Storage Technique

- formspring.txt (formspring): SHA 2 - 256

- SHA1.txt (linkedin): SHA 1
- password.file (yahoo): Some sort of key value pair in plain text

These are in hard to crack to easy to crack order.

- SHA-256 is quite secure due to its complexity and resistance to collisions and pre-image attacks. However, if the password is weak or commonly used, a brute-force attack could still be successful (like in our case).
- SHA-1 is considered weaker compared to SHA-256 due to vulnerabilities that allow for collision attacks.
- Plain text passwords are the easiest to crack because they are not hashed or encrypted.