Python Plain Text Password Cracker Tutorial

Contents

Time required: 30 minutes

What is pwned?

pwned, in a security context, means that your account has been the victim of a data breach.

The word itself takes its name from player-to-player messaging in online computer gaming. When one player is defeated, another might type out a message to say 'You've been owned'.

This was so frequently misspelt as 'pwned', the word itself took off.

Has Your Password been Pwned?

Check out your current email account or cell phone number. Has your information been compromised in a data breach? If you find your info in the list, change your passwords now.

https://haveibeenpwned.com/

Password Lists

Password lists are text files of common passwords used to crack password hashes. These are from data breaches and are REAL passwords that people use.

The following sites have password lists. We will use some in this lab.

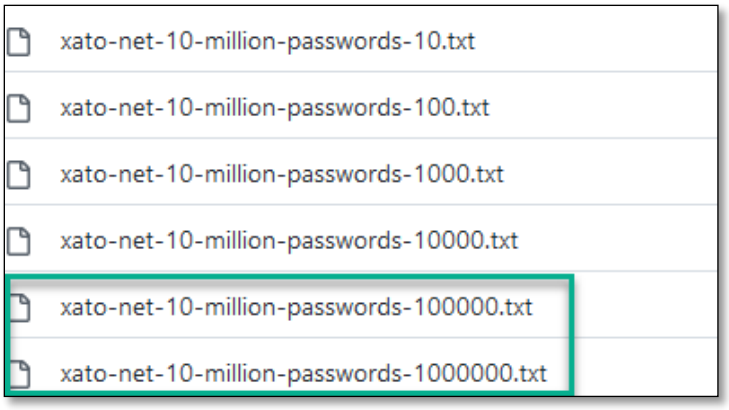
WARNING: Some of the passwords might be offensive. Remember, these are real passwords that people have and do use.

- https://github.com/danielmiessler/SecLists/tree/master/Passwords
- https://weakpass.com/

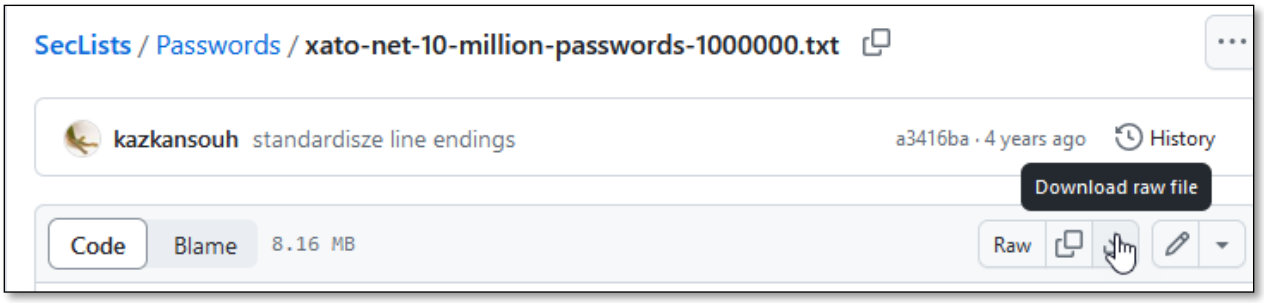
Tutorial 1: Get a Password List

We are going to use a password list that has real passwords from data breaches and popular passwords in it.

1. Go to https://github.com/danielmiessler/SecLists/tree/master/Passwords
2. I suggest using one of the two highlighted lists. I will be using the last one. 10 has the 10 most popular passwords, 100 has the 100 most popular passwords, etc. The bigger the list, the better chance of finding a password.



3. Click the password file. Click View Raw.



4. In the same folder that you will save your password cracker program: Save the file as **passwords.txt**

We are now ready to create our password cracking tool in Python.

Tutorial 2: Python pwned Password Cracker

The following program uses a password list to find a password. We aren't really cracking a password . . . yet. We are going through the comparison process with a password list to get used to the idea.

1. Create a Python program named **password_cracker.py**
2. Enter the following code, comments and all.

```
#!/usr/bin/env python3
"""
    Name: password_cracker_plain_text.py
    Author:
    Created:
    Crack a password using a plain text password list
    https://github.com/danielmiessler/SecLists/tree/master/Passwords
"""



# ------------------------ PRINT TITLE ---------------------------------#
def print_title():
    print(" +-------------------------------------------------+")
    print(" |    ---   Bill's Best Password Cracker   ---     |")
    print(" |         . . . Use at your own risk . . .        |")
    print(" +-------------------------------------------------+")
```

1. Get the password to be tested from the user.
2. Enter the password list filename. This method of input handling provides a default value for the input if you press the Enter key without entering a value.

```
19  def main():
20      # Boolean variable to track whether the password has been found
21      password_found = False
22
23      print_title()
24
25      # Get password to get tested from user
26      input_password = input(" Enter a password: ")
27
28      # Enter the password list filename
29      password_list_file = (
30          input(" Enter password list filename: ") or "1000000_passwords.txt"
31      )
```

3. Read each individual password in the file into a list. The with context handler automatically closes the file handle when the with statement ends.

```
31      # ----------- Read all passwords into a Python list ----------------- #
32      try:
33          # Try to open the password file using the 'with' context handler
34          # with automatically closes the file when you exit the block
35          with open(password_list_file, "r") as file:
36
37              # Read file --> splitlines() removes \n newline
38              # Read each line into a list item
39              password_list = file.read().splitlines()
40
41          # The file is automatically closed when with exits
42
43      except Exception as e:
44          # If there is an error reading the file, we handle it here
45          print(f" Error: {e}")
46          print(f" {password_list_file} is not found.")
47          quit()
```

4. Compare each password in the list with the input password.

The enumerate() function in Python adds a counter to an iterable (like a list, tuple, or string) and returns it as an enumerate object. This allows you to loop through the iterable while keeping track of both the index and the value of each item.

```python
    # ----- Compare each password in the list with the input password ------ #
    # Loop through each password in the password list one at a time
    # Use enumerate to get index and password
    for index, password in enumerate(password_list):

        # Compare the input_password with the
        # current password in the password_list
        if input_password == password:
            print(f" Password found. The password is: {password}")
            print(
                f" Password position in the list: {index}"
            )  # Print the index of the password
            # If password is found, set password_found to True
            password_found = True
            break

    # If the password is not found
    if password_found == False:
        print(f" Password not found in {password_list_file} file")


main()
```

Example run:

```
+-----------------------------------------------+
|    ---   Bill's Best Password Cracker   ---    |
|        . . . Use at your own risk . . .       |
+-----------------------------------------------+
Enter a password: zebra
Enter password list filename:
Password found. The password is: zebra
Password position in the list: 6573
```

Assignment 1: Try Your Own Passwords

Try some of your own passwords on this list. If you find your password on a 5 year old, you better change it fast!

Assignment Submission

1. Attach all program files.
2. **NOTE:** Do not attach the password file, I will test your program with my own files.

3. Attach a screenshot of your functioning program.
4. Attach to the assignment in BlackBoard.