CSE 3140 Lab 1 - Password Cracking and Crypto Hash Functions

Tom McCarthy (tkm20002), Filip Graham (fvg20002).
Section Z81
(Tom) IP of VM: 172.16.49.96, (Filip) IP of VM: 172.16.49.88

Question 1 : Adam's password. Million2 (Filip's VM)

- Break1.py code

```python
import subprocess
import time

start_time = time.time() # Record the start time

passwords = [] # list that contains the common passwords

# Open the file for reading
with open('./MostCommonPWs', 'r') as file:
    for line in file:
        password = line.strip() # Remove leading and trailing whitespace (or "\n" newline characters)
        passwords.append(password) # Append the cleaned password to the list

usernameToHack = 'Adam'
for password in passwords:
    result = subprocess.run(['python3', 'Login.pyc', usernameToHack, password], stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)

    # Check if the command was successful
    if result.returncode == 0:
        if result.stdout == 'Login successful.\n':
            print(f"The password for {usernameToHack} is: {password}")
            break
            # print("result is: ", result) # result is:  CompletedProcess(args=['python3', 'Login.pyc', 'Adam', 'Million2'], returncode=0, stdout='Login successful.\n', stderr='')

end_time = time.time() # Record the end time

elapsed_time = end_time - start_time # Calculate the elapsed time

print(f"Break1.py took {elapsed_time:.2f} seconds to run")
```

- Break1.py output

```
cse@cse3140-HVM-domU:~/Lab1/Q1$ python3 Break1.py
The password for Adam is: Million2
Break1.py took 0.92 seconds to run
```

Question 2.A : Name of the gang member exposed. Andrew (Filip's VM)

Question 2.B : Password of the gang member exposed. 123456789

- Break2.py code

```python
import subprocess
```

```python
import time

start_time = time.time() # Record the start time

passwords = [] # list that contains the common passwords

# Open the file for reading
with open('./MostCommonPWs', 'r') as file:
    for line in file:
        password = line.strip() # Remove leading and trailing whitespace (or "\n" newline characters)
        passwords.append(password) # Append the cleaned password to the list

usernames = []
with open('./gang', 'r') as file:
    for line in file:
        username = line.strip() # Remove leading and trailing whitespace (or "\n" newline characters)
        if username != "Adam":
            usernames.append(username) # Append the cleaned password to the list

def hack():
    for usernameToHack in usernames:
        for password in passwords:
            result = subprocess.run(['python3', 'Login.pyc', usernameToHack, password], stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)

            # Check if the command was successful
            if result.returncode == 0:
                if result.stdout == 'Login successful.\n':
                    print(f"The password for {usernameToHack} is: {password}")
                    return # lab mentions that there is one other gang memeber, so we can just break
                    # print("result is: ", result) # result is:  CompletedProcess(args=['python3', 'Login.pyc', 'Adam', 'Million2'], returncode=0, stdout='Login successful.\n',
stderr='')
hack()

end_time = time.time() # Record the end time

elapsed_time = end_time - start_time # Calculate the elapsed time

print(f"Break2.py took {elapsed_time:.2f} seconds to run")
```

- Break2.py output

```
cse@cse3140-HVM-domU:~/Lab1/Q2$ python3 Break2.py
The password for Andrew is: 123456789
Break2.py took 8.88 seconds to run
```

Question 3.A : Name of the gang member exposed. **Al** (Tom's VM)
Question 3.B : Password of the gang member exposed. **neverguess**

- Break3.py code

```python
import subprocess
import time

start_time = time.time() # Record the start time

passwords = [] # list that contains the common passwords

# Open the file for reading
with open('./PwnedPWs100k', 'r') as file:
    for line in file:
        password = line.strip() # Remove leading and trailing whitespace (or "\n" newline characters)
        passwords.append(password) # Append the cleaned password to the list

usernames = []
with open('./gang', 'r') as file:
    for line in file:
        username = line.strip() # Remove leading and trailing whitespace (or "\n" newline characters)
        usernames.append(username) # Append the cleaned password to the list

for usernameToHack in usernames:
    if usernameToHack == 'Adam' or usernameToHack == 'Andrew':
        continue
    for password in passwords:
        result = subprocess.run(['python3', 'Login.pyc', usernameToHack, password], stdout=subprocess.PIPE,
stderr=subprocess.PIPE, text=True)

        # Check if the command was successful
        if result.returncode == 0:
            if result.stdout == 'Login successful.\n':
                print(f"The password for {usernameToHack} is: {password}")
                # break
                # print("result is: ", result) # result is:  CompletedProcess(args=['python3', 'Login.pyc', 'Adam', 'Million2'],
returncode=0, stdout='Login successful.\n', stderr='')

end_time = time.time() # Record the end time

elapsed_time = end_time - start_time # Calculate the elapsed time

print(f"Break3.py took {elapsed_time:.2f} seconds to run")
```

- Break3.py output

```
cse@cse3140-HVM-domU:~/Lab1/Q3$ python3 Break3.py
The password for A1 is: neverguess
Break3.py took 97421.42 seconds to run
```

Question 4.A : Name of the gang member exposed. Kim (Filip's VM)
Question 4.B : Password of the gang member exposed. Sc8Z6nxT

- Break4.py code

```
import subprocess
import time

start_time = time.time()
usernames = set()
with open('./gang', 'r') as file:
    for line in file:
        username = line.strip()
        usernames.add(username)

passwords = []

with open('./PwnedPWfile', 'r') as file:
    for line in file:
        stripped = line.strip()
        splitted = stripped.split(',')
        if splitted[0] in usernames:
            result = subprocess.run(['python3', 'Login.pyc', splitted[0], splitted[1]], stdout = subprocess.PIPE, stderr =
subprocess.PIPE, text = True)
            if result.returncode == 0:
                if result.stdout == 'Login successful.\n':
                    print(f"The password for {splitted[0]} is: {splitted[1]}")

end_time = time.time()
elapsed_time = end_time - start_time
print(f"Break4.py took {elapsed_time:.2f} seconds to run")
```

- Break4.py output

```
cse@cse3140-HVM-domU:~/Lab1/Q4$ python3 Break4.py
The password for Kim is: Sc8Z6nxT
Break4.py took 0.14 seconds to run
```

Question 5.A : Name of the gang member exposed. Jack (Filip's VM)
Question 5.B : Password of the gang member exposed. sadman51

- Break5.py code

```
import subprocess
import time
import hashlib
from collections import defaultdict

start_time = time.time() # Record the start time
```

```
# Open the file for reading


usernames = set()
with open('./gang', 'r') as file:
    for line in file:
        username = line.strip()
        if username == 'Adam' or username == 'Kim' or username == 'Andrew' or username == 'Charles':
            continue
        usernames.add(username)

hashed = defaultdict(list)
with open('./HashedPWs', 'r') as file:
    for line in file:
        stripped = line.strip()
        splitted = stripped.split(',')
        if splitted[0] in usernames:
            hashed[splitted[1]].append(splitted[0])

with open('./PwnedPWs100k', 'r') as file:
    for line in file:
        password = line.strip() # Remove leading and trailing whitespace (or "\n" newline characters)
        for i in range(10):
            for j in range(10):
                hash = hashlib.sha256()
                updatedPassword = password + str(i) + str(j)
                hash.update(bytes(updatedPassword, "utf-8"))
                hex = hash.hexdigest()
                if hex in hashed:
                    for member in hashed[hex]:
                        result = subprocess.run(['python3', 'Login.pyc', member, updatedPassword], stdout =
subprocess.PIPE, stderr = subprocess.PIPE, text = True)
                        if result.returncode == 0:
                            if result.stdout == 'Login successful.\n':
                                print(f"The password for {member} is: {updatedPassword}")
end_time = time.time() # Record the end time

elapsed_time = end_time - start_time # Calculate the elapsed time

print(f"Break5.py took {elapsed_time:.2f} seconds to run")
```

- Break5.py output

```
cse@cse3140-HVM-domU:~/Lab1/Q5$ python3 Break5.py
The password for Jack is: sadman51
Break5.py took 21.05 seconds to run
```

Question 6.A : Name of the gang member exposed. **Bonnie**

Question 6.B : Password of the gang member exposed. bella178

- Break6.py code

```
import subprocess
import time
import hashlib
from collections import defaultdict

start_time = time.time() # Record the start time

# Open the file for reading


usernames = set()
with open('./gang', 'r') as file:
    for line in file:
        username = line.strip()
        if username == 'Adam' or username == 'Kim' or username == 'Al' or username == 'Charles' or username ==
'Kevin':
            continue
        usernames.add(username)

hashed = []
with open('./SaltedPWs', 'r') as file:
    for line in file:
        stripped = line.strip()
        splitted = stripped.split(',')
        if splitted[0] in usernames:
            hashed.append([splitted[0], splitted[1], splitted[2]])

with open('./PwnedPWs100k', 'r') as file:
    for line in file:
        password = line.strip() # Remove leading and trailing whitespace (or "\n" newline characters)
        for i in range(10):
            updatedPassword = password + str(i)
            for h in hashed:
                hash = hashlib.sha256()
                hash.update(bytes(h[1] + updatedPassword, "utf-8"))
                hex = hash.hexdigest()
                if hex == h[2]:
                    result = subprocess.run(['python3', 'Login.pyc', h[0], updatedPassword], stdout = subprocess.PIPE,
stderr = subprocess.PIPE, text = True)
                    if result.returncode == 0:
                        if result.stdout == 'Login successful.\n':
                            print(f"The password for {h[0]} is: {updatedPassword}")
end_time = time.time() # Record the end time

elapsed_time = end_time - start_time # Calculate the elapsed time
```

```
print(f"Break6.py took {elapsed_time:.2f} seconds to run")
```

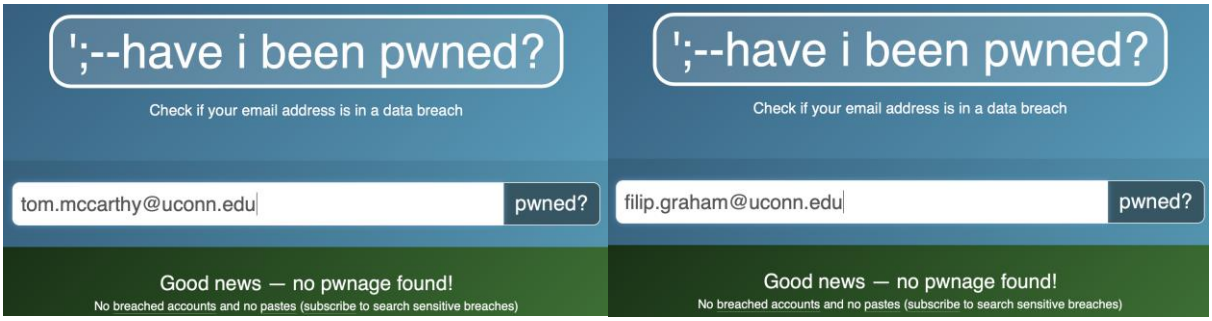
- Break6.py output

```
cse@cse3140-HVM-domU:~/Lab1/Q6$ python3 Break6.py
The password for Bonnie is: bella178
Break6.py took 10.45 seconds to run
```

Explain why the SaltedPWs would (normally) be harder to attack, compared to HashedPWs.

SaltedPWs are more time consuming to attack compared to HashedPWs because there is an extra step involved in identifying matches between passwords and hashes. With only HashedPWs, an attacker can store hashes in a hash table, then go through possible passwords and it takes O(1) time complexity to determine if the hashed password exists in the hash table. However, with a salted PW, the attacker would need to store the associations between a hash and salt. For each possible password, the attacker would need to go through each hash/salt pair, concatenate the password and salt before comparing it with the hash. The same hash table trick cannot be used to improve time complexity so this process takes longer and is harder to attack.

Question 7 requires no submission: Submit in HuskyCT screenshots showing your registration and results.



Question 8 requires no submission: Submit in HuskyCT a short description of the two worst exposures you could identify.

The plain passwords from users of Netflix, Pastebin, Minecraft, and others were exposed. These services are used by millions of people, making this a bad exposure of passwords.

The collective database contains plain text credentials leaked from Bitcoin, Pastebin, LinkedIn, MySpace, Netflix, YouPorn, Last.FM, Zoosk, Badoo, RedBox, games like Minecraft and Runescape, and credential lists like Anti Public, Exploit.in.

"None of the passwords are encrypted, and what's scary is that we've tested a subset of these passwords and most of the have been verified to be true," Casal said. "The breach is almost two times larger than the previous largest credential exposure, the Exploit.in combo list that exposed 797 million records."

(https://thehackernews.com/2017/12/data-breach-password-list.html)

The hashed-but-unsalted passwords from users of LinkedIn were exposed in 2016. It was discovered that around 117 million were hashed but not salted.

- LinkedIn leaked 8 million passwords in 2012. The main reason – passwords were hashed but not salted. When hackers managed to crack the hash used by LinkedIn, all of the stolen passwords were revealed.
- In 2013, Adobe leaked 130 million passwords. Passwords were encrypted but lacked hashing and salting – it was easy for hackers to crack them via brute-force and rainbow table attacks.
- A few years passed, and in 2016 LinkedIn again experienced a data leak. 117 million accounts were hashed but, again, were not salted. This made them easy to reveal.

(https://www.passcamp.com/blog/what-is-password-salting-and-why-should-you-care/#)

Question 9 requires no submission: Submit in HuskyCT examples of websites that use 2FA and ones that don't.

A 'significant' website that does not support 2FA would be leetcode.com. We chose this website because it has over 5,000,000 registered users and because we use this website regularly.

A 'significant' website that supports 2FA would be gmail.com. We chose this website because it has over 1.8 billion users and is the most popular email service.

I use 2FA for important accounts like email and financial services. My parents do not regularly use 2FA unless the service requires it.