Name: Truc Ho Nguyen

NetID: thv20002

Section: 8

VM IP Address: 172.16.49.54

1. To find the answer for Question 1, I first had to write the Break1.py file using sudo vim Break1.py. It contained a for-loop that tested every password in the file MostCommonPWs. After writing the code, I used python3 Break1.py to find the password (iloveyou) which took about 0.73 seconds.



2. To find the answer for Question 2, I modified Break1.py to have a nested loop that also tested every name in the file gang. After writing the code, I used python3 Break2.py to find the name (Kim) and password (12345) which took about 6.34 seconds.



3. To find the answer for Question 3, I modified Break2.py to read from PwnedPWs100k instead. I removed Adam and Kim from the gang file, then used python3 Break3.py to find the name (Charles) and password (green07) which took about 6284.28 seconds.



4. To find the answer for Question 4, I modified Break2.py to split the passwords from the usernames in PwnedPWfile. I removed Adam, Kim, and Charles from the gang file, then used python3 Break4.py to find the name (Donald) and password (Dyzi50WN) which took about 3249.72 seconds.



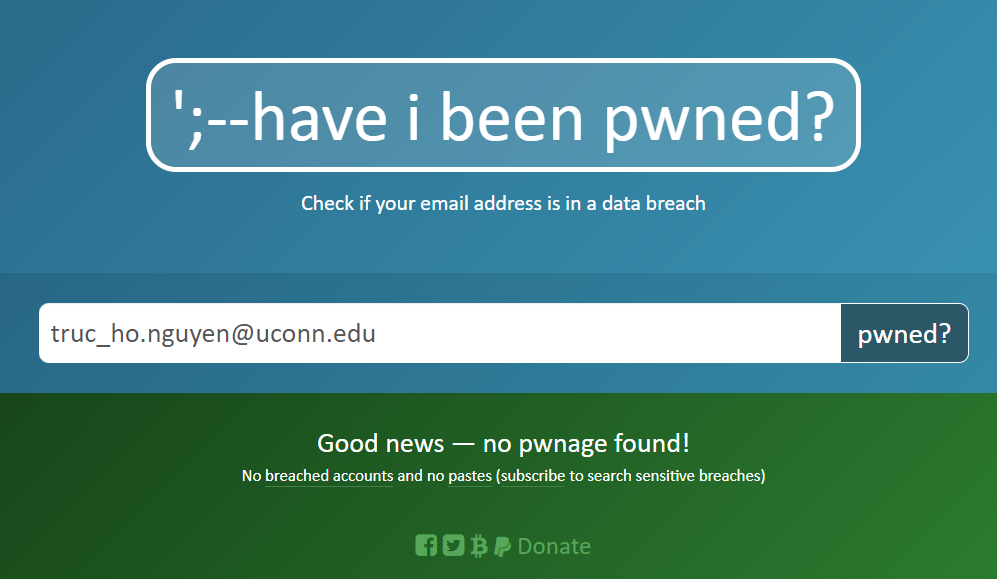
5. To find the answer for Question 5, I modified Break2.py to first create a dictionary of each name, paired with all of its hashes inside HashedPWs. Then, a loop would test every name in the gang file and check if it was also in the dictionary. If so, it would test different combinations of passwords concatenated with 2 digits for the name. If the SHA-256 hash of these combinations were found in the name’s dictionary value set, then it would attempt a login. I removed Adam, Kim, Charles, and Donald from the gang file, then used python3 Break5.py to find the name (Ted) and password (01058598) which took about 32.06 seconds.



6. To find the answer for Question 6, I modified Break5.py to store tuples of (salts, salted passwords) in the dictionary instead. Then, a loop would test every name in the gang file and check if it was also in the dictionary. If so, it would test different combinations of salts concatenated with the passwords concatenated with 1 digit for the name. If the SHA-256 hash of these combinations were found in the name’s salted password position of the tuple, then it would attempt a login. I removed Adam, Kim, Charles, Donald, and Ted from the gang file, then used python3 Break6.py to find the name (Clyde) and password (emery15) which took about 9.71 seconds.



7. My results using my UConn email:

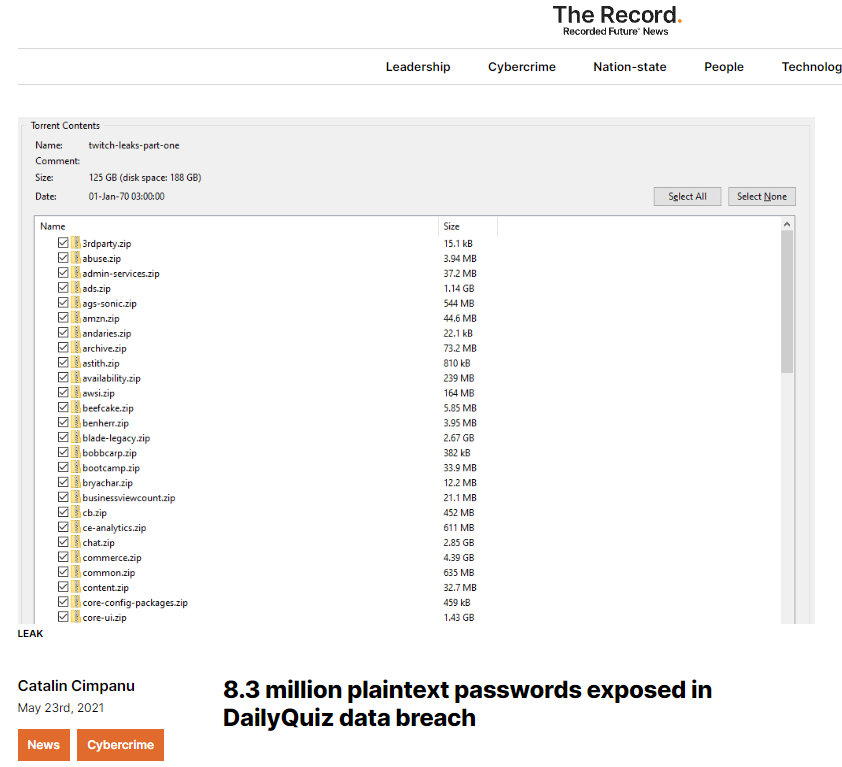


8. The two worst password exposures I could find on the internet were:

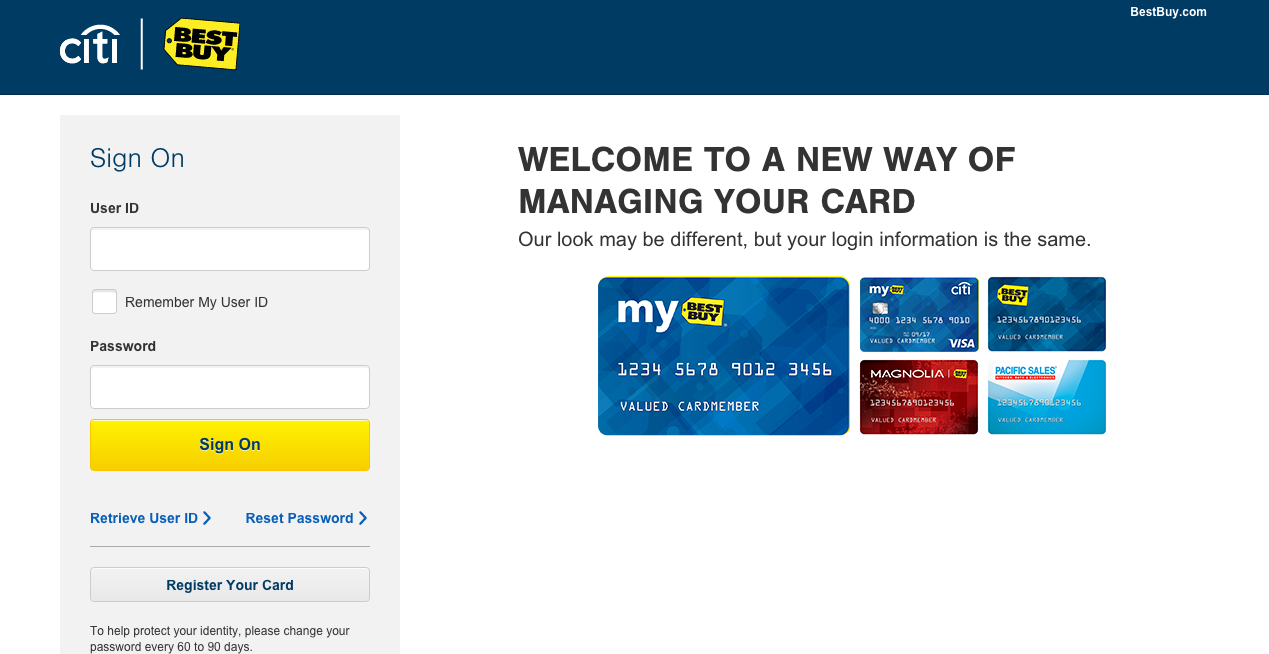
* Yahoo (2013): The breach affected 3 billion different user accounts, leaking the encrypted information of not only passwords, but also names, birth dates, phone numbers, security questions, and backup email addresses.



* DailyQuiz (2021): The breach affected 13 million different user accounts after a hacker stole the content from the website’s database. The information included passwords, emails, and IP addresses all stored in plain text. The data was also put up for sale on different forums and channels, where it was bought using cryptocurrencies.



9. One popular website that does not currently use 2FA is Best Buy, which I chose because my family regularly purchases their technology products here. 2FA is not always necessary on retail websites because if a user’s account is compromised, they will receive emails of fraudulent purchases and have time to cancel the order and recover their information.



One popular website that does currently use 2FA is X, formerly known as Twitter, which I chose because it is one of the most widely used social media platforms at the moment. 2FA is relatively important on social media websites because hackers could easily spread harmful information using accounts with a large following.

