HW2

Eharmony:

My first attempt on cracking password. At first, I searched it online with the word “crack password”. I acquired information that it is encoded by MD5, all the lower letter are all capitalized before it got hashed (info source: https://defuse.ca/blog/tag/eharmony.html). It is non-reversible and there is a python library for doing MD5 encoding. So my first thought is to create permutation of letters, numbers and symbols to crack the password —— also known as brute force approach.

In order to do that, I went to eharmony’s website to try out what is their password requirement — such as how long it requires for minimum length password, do they enforce special character etc. After I played around, I found the password length for the website is between 8-14, no special character requirement.

So I use the itertool to do the permutation. Sadly, it is highly inefficient. Including alphabets, numbers and symbols just too many tokens. For the shortest permutation(8 characters), there are millions of possibility, and as the length of password grows, the possibility grows exponentially. So I decide to move to other approaches.

Since people usually use some real words for password, and there are some commonly used password list online, so I download one of the common password list. I capitalized every of the common password and MD5-hashed it, finally came to crack some passwords.

Worth mentioning, the file read line has a ‘\n’ be with the password list. That is why primarily my word list attack didn’t work. I spent a long time to figure out why :)

Formspring:

With the success of decoding eharmony password list, I decided to use word list attack for both formspring and linked.

My search on formspring let me know that they used another way for hashing — SHA256 Moreover, they are using salt. Based on my search, salt is a string that added to original password. By hashing salt+actual password, it can be more secure. However, based on my search, form spring didn’t do a good job in salting. They used a random salt from 00-99 —Not to hard to generate and append to wordlist and attack.
(Source: https://tangev.files.wordpress.com/2014/05/final-report-improving.pdf
https://www.itnews.com.au/news/formspring-420000-lost-passwords-were-encrypted-salted-308425)

I prepend and append the 00-99 to my wordlist I used for eharmony attack. Before I try to append, my prepend attack works out already. It shows this is a way to store multiple same password differently with different salt value.

Linkedin:

Based on my search, they are using unsalted SHA-1.
(Source: https://nakedsecurity.sophos.com/2012/06/06/millions-of-linkedin-passwords-reportedly-leaked-take-action-now/)

Actually a close look at linked in’s password file at first helps a lot in this process —— There are a lot of password starts with “00000” Based on the length, I believe they have some mechanism to random choose some hashed password and replace the first five character as “00000”. As result, I tried to hash my common password list I used above in SHA-1, and they replace the first five character with “00000” —- it works out.

Thinking on the difficulties:

I feel the eharmony one is the most insecure one — Hashed without other protection, also it loses case sensitive. If some of the passwords are cracked, just based on comparing the compromised one to the other passwords, a lot of passwords will further more be compromised.

Compare to eharmony, formspring did a better job, but not enough. They used salt values, that protect some user's password when the other same passwords are compromised. However, like I mentioned before —- 00-99 salt value is easy to be generated and tried on. They should use a more complicated salt or do some random salt insertion other than just prepend.

Linkedin —- I don't know how some of the passwords are hashed, but for sure the zero replacement stands out when somebody view the password list. It looks pretty obvious. I think they can do another round of double hashing based on that.