**CMPUT 333 Assignment 1 Sliding Part**

UNIX Hashes

Our dictionary, dict.lst, containing lowercased English words was obtained from: <http://users.uoa.gr/~sdi0900089/kremala/wordsen.txt>.

Hint: 10 letter word

Running cased10.py will read in the dict.lst and create a new dictionary, cased10length.lst, consisting of all casings of the words of length 10. This was used to obtain $1$ZyPC/.VR$1BhB4OIQgJpjST/uGAHj1/:serialised.

Command Used: john --wordlist=cased10length.lst hashes\_unix.txt

Hint: 10 letter word with 2 leet characters

Running leeter.py will read in cased10length.lst and create a new dictionary, leeted10length.lst, containing words with two 1337 characters, based on the information found on: <https://simple.wikipedia.org/wiki/Leet>. This was used to obtain $1$PARkGLya$.SNH3Lw3iLOwGd91huTJk/:eu7hanas1a.

Command Used: john --wordlist=leeted10length.lst hashes\_unix.txt

List of cities around the world, locations.txt, was obtained from: <https://gist.githubusercontent.com/aqlx86/4beb88bf4cff0cde7a82/raw/bf462d190a2f6c0522c0956ac80d4e9d40b52783/country-city-list.json>.

Hint: 2 word cities (non-alphanumeric connector character (printable ASCII) with capitalization variation independently for the two words

Running cities.py will parse locations.txt, find all of the two word cities, and combine them with one printable, non-alphanumeric character into the dictionary 2wordcities.lst. This was used to obtain $1$gdBSfxVw$rztpKlJcHnoStKEn49p6H.:cedar\_rapids.

Command Used: john --wordlist=2wordcities.lst hashes\_unix.txt

NOTE: if you want to run cities.py on a windows OS, you need to add encoding=”utf-8” in the open file command on line 13. it should look like this:

**open('filename.txt', 'r', encoding='utf-8')**

Hint: 8 symbol character hexadecimal number without leading 0x

We first assumed that the hexadecimal number was either all lowercase or all uppercase, and decided to try all lowercase first. We first made a dictionary, 1h.lst, of 1 character words of valid hex values, 0 – 9 and a – f. Then we added a new rule, Ex, into the john.conf:

[List.Rules:Ex]

$[0-9a-f]$[0-9a-f]$[0-9a-f]$[0-9a-f]$[0-9a-f]$[0-9a-f]$[0-9a-f]

This rule appends 7 characters to the end of each word, and each are also valid hex values 0 – 9 and a – f, therefore testing all possible 8 character lowercase hex values. The lowercase hex obtained the password $1$qCu.TdPv$2NzUJwEQaSABhAqPlQ14r0:aa9d1dd2.

Command Used: john --wordlist=1h.lst --rule=Ex hashes\_unix.txt

Hint : Random 11 character key (alphanumeric with both casings)

The 11 random alphanumeric password wasn’t found, since we aren’t able to eliminate the possibilities of keys. This would mean we have to brute force 6211 keys, since there are 62 different possibilities to choose from for a character, and 11 characters in total.

LANMAN/NT Hashes

<http://www.tobtu.com/lmntlm.php> was used as reference. The hashes provided consists of two halves, a NTLM hash and an LM hash. We created a new file consisting of only the LM portion of the hashes. LM hash encrypts 14 character passwords, padding passwords with null values if the length is smaller than 14, converting them into uppercase and then splitting them into two 7 character halves before encrypting. This is helpful since we only need to crack each half separately, which is only of length 7, and all characters are capitalized, so it removes almost half of the possible characters for each character of the password. We obtained:

$LM$73cc52eafe696961:LATINI

$LM$fc738eb8b0e67616:TOOLBOX

$LM$1164ce4aaefc17ff:MICHELP

$LM$97a9eecee4b5f330:AKSS

$LM$f9c31a96c4c65b1e:X8SJKRO

Command Used: john --format=LM hashes\_lanman.txt

Put them back together to get the password of the corresponding hashes, but all in uppercase:

F9C31A96C4C65B1E97A9EECEE4B5F330:X8SJKROAKSS

FC738EB8B0E67616AAD3B435B51404EE:TOOLBOX

1164CE4AAEFC17FF73CC52EAFE696961:MICHELPLATINI

Note that AAD3B435B51404EE is the hash of an all null 7 character half.

To get the case sensitive password, we had to only try the all the permutations of casings of these passwords. This can be done by creating a new wordlist containing the three uppercase passwords above, lanman\_sol.lst, and using the rule NT, obtaining:

$NT$f3f69a51fc702f24ee994a3a98fca9b0:toolbox

$NT$4b5628becfd9ad0bc1a5e797ff3ca686:x8SJkrOAksS

$NT$d94631ab52f49e06830bd207da175d32:MichelPlatini

Command Used: john --format=NT --wordlist=lanman\_sol.lst --rule=NT hashes\_win.txt

Conclusion:

The Windows passwords are weaker than the Linux passwords in general, as the hashing scheme is weak. You only need to search for 7 length halves, which is subject to easy brute forcing. Therefore, even if the password is a random 14 character password, it is still easily cracked. However, in the case of the random 11 character Linux password, it is cannot be easily brute forced, with the reasoning mentioned above. However, given the hints provided, the other 4 Linux passwords were also cracked with relative ease, as they provided enough information to significantly reduce the set of possible keys required to be tested.