CSE 127: Computer Security (Fall 2017) **Due November 16, 10:00 P.M. PDT**

Assignment 6

100 pts

The goal of this assignment is to gain hands-on experience with password cracking. The problem consists of two parts: in the first part you need to crack unsalted password hashes, and in the second part you need to crack salted ones. A word list is provided for you so you can do a dictionary attack. Your solution is due on November 16, 10:00 P.M. PDT. You may work with *one* other person in the class on this assignment, however, each student must submit his/her own solution. See Section 4 for additional information on submitting your solution.

1 Problems

It is considered bad security practice for a Web service to store user passwords in the clear. Instead, sites store the hash value of a password computed using a cryptographic hash function. Later, to check if a password provided by a user is correct, the site first computes the hash of the offered password using the same function and compares the result with the stored value. The advantage of storing a hash of the password rather than the password itself is that if the password database is stolen, the attacker will not get the user passwords.

People tend to use create passwords that are easy to remember, such as an English word or a string of digits. Unfortunately, this makes it easier for attackers to determine the password by only trying words from a dictionary or some simple numbers, which dramatically reduces the search space. In the following problems, you will act as an attacker and crack the simple passwords using this dictionary attack.

1.1 Part 1

You have 1,000 password hashes from a password database of a Web service. We know the users of this service are not familiar with password security and create their passwords in only a few ways:

1. An English word as the password (e.g. "`password`").
2. A string of up to 8 digits as the password (e.g. "`12345678`").
3. An English word followed by some digits, together no more than 10 characters (e.g. "`happy12345`").
4. An English word but change some letters to uppercase and change some letters to other symbols, as described below (e.g. "`C@t`").
5. Concatenate two English words together (e.g. "`dogcow`").

When choosing English words, users only use English words from the provided dictionary of the 10,000 most common English words.

1.2 Part 2

You have another password table from the same Web service, this time containing the usernames and passwords for the VIP users. The website salted the password hashes in this table to provide better security, but the users still created their passwords in the same vulnerable way described above. There are only 100 password hashes in this table, and the salt values are provided alongside the hashes.

2 Problem Details

You will be provided with your own set of password hashes that you need to crack. You will receive 1 point for every 20 user passwords from Part 1, and 1 point for every 2 VIP user passwords from Part 2. The list of 10,000 English words will be provided in a file named `words.txt`, where all the words are lowercase.

Note that a numeric password can contain zero in the front, e.g. `00001`. Additionally, you can assume that when users create passwords using the third way above, the English words they use have at least 5 letters, such as in `"hello001"`. Also, an arbitrary number of letters may be uppercase; for example, `"password"`, `"pAsswOrd"`, and `"paSSwoRd"` are all valid. Additionally, letter substitution is a way people change letters in their passwords, making them hard to guess but still easy to remember. In this problem, the possible substitutions are:

Letter	Replacement	Letter	Replacement
`a`	`@` (ASCII 64)	`b`	`8` (ASCII 56)
`c`	`(` (ASCII 40)	`f`	`#` (ASCII 35)
`g`	`9` (ASCII 57)	`i, l`	`1` (ASCII 49)
`o`	`0` (ASCII 48)	`s`	`$` (ASCII 36)

So `"p@ssword"`, `"pa$$word"`, and `"passw0rd"` are all valid. Note that the substitution for the letter `o` is the digit `0` and for the letters `l` (lowercase letter L) and `i` is the digit `1`.

The password hashing function is different for each student. Your password hashing function works by computing the MD5 hash of your PID concatenated with the input. The password files contain the resulting MD5 value expressed in hexadecimal. Suppose the original password is "`password`" and your PID is `A12345678`. Then for the unsalted version, the hash function used to generate your password hashes concatenates the password after PID into `A12345678password`, then calculates MD5 hash of this new string. So if two students get the same passwords, there will still be different hashes. Note that the first letter in the PID is uppercase.

The salted hashes will involve additional salt values. For example, if the original password is "`password`" and the salt value is "`84B03D03`", then the hash function first concatenates two strings together with students' PID into "`A12345678password84B03D03`", then computes the MD5 hash of this new string. Note that the salt is treated as a string, not a hexadecimal value.

3 Homework Download

You can download your homework from the Google drive link below.

https://drive.google.com/drive/folders/1UC_DL3-8f7C96XYxwnxHC5Gl85F-wgH5

The tar files are named using your secret number from GradeSource, so please find your own number and download the corresponding tar file. You must make sure it is your secret number; if you accidentally download somebody else's homework and solve it, you will not receive any credit.

Inside the tar archive there are three files. The `words.txt` contains the 10,000 English words that people use to create their passwords. `hash1.txt` contains the password hashes you need to crack for Part 1, with "`{username}:{hash}`" on each line. For example, if the username is `john` and password hash is `7e23..2b41`, then that line will be:

```
john:7e238a22c982f0d9de093fc7bca92b41
```

The file `hash2.txt` contains the password hashes for Part 2. It has username, password and the salt value on each line, separated by ":". Using the same username and password as above, with a salt value `67ef98c0`, the corresponding line in the password table is:

```
john:55457bde7a1a4816ba636d09017ff30a:67ef98c0
```

You should confirm that the password for both of the examples above is "`cse127ftw`" for PID `A12345678`.

4 Submitting Your Solution

Your solution to this assignment must be a plain text file named "{PID}-hw6.txt" (where {PID} is your PID) containing each username and cracked password in the form "{username}:{password}" one per line, where {password} is the password for username. Figure 1 below shows an example of this format.

```
Foster, Ian
A00000000
Assignment 6
Worked with Maskiewicz, Jacob

brian:123456
guo:password
...
```

Figure 1: Example solution file format.

Your solution must be submitted via email to `cs127f1@ieng6.ucsd.edu` by November 16, 2017, 10:00 P.M. PDT. Even though you can work with one other person on this assignment, *each student must submit his/her own solution.* Each student's solutions will be different.

Sign your solution with your PGP key and encrypt to the `cs127f1@ieng6.ucsd.edu` PGP key, which is provided on the CSE 127 Web page and has key fingerprint:

```
E1BF 1E04 1104 28DA 4F89 6543 B033 B3DC 10D3 7DBD.
```

You must send a plain email with the encrypted and signed archive file as an attachment. The email must have the subject "Homework 6 Submission" and the attachment must be named "{PID}-hw6.txt.asc" (where {PID} is your PID).

Each student must work individually on this homework and submit his/her own solution. You may *not* discuss your solution with any other students except your partner until seven days after the assignment deadline.

To sign and encrypt your submission with GPG, you can use the following command:

```
gpg --encrypt --sign --armor -r cs127f1@ieng6.ucsd.edu {PID}-hw6.txt
```

This will produce a file named "{PID}-hw6.txt.asc" in the same directory. You will need to have imported the `cs127f1@ieng6.ucsd.edu` public key into your GPG keyring first.