CS 577 Cybersecurity Lab

Lab 2 – due 9/24/14 11:59pm

Subject: Password Cracker

Create the best and fastest password cracker you can develop in one week. It should able to break passwords hashed with the MD5 hash function. Your cracker should use the following techniques:

Dictionary attack: Utilize a dictionary.

Brute-force: Brute-force short 3-5 character long passwords.

Heuristics combining the above**:** Combine dictionary and brute-force attacks. For instance, try to break passwords that include a word followed by a small number of digits (cybersecurity14) or replace characters to crack passwords in leetspeak (e.g., replace 'e' with '3', 'a' with '4' and so on). Your password cracker should implement at least 2 such heuristics. They need not be the two examples given here.

You can use any techniques mentioned in the lecture to improve your cracker. For example, rainbow tables, threads, GPUs, etc.

You are given two files with passwords, which you can use for testing your tool during development:

The 1st one has just hashed passwords (passwords_nosalt.txt).

hash = MD5(password)

The 2nd one has passwords hashed after appending a salt with the value "id14" (passwords_salt.txt).

hash = MD5(password||salt)

Deliverables

1. Your password cracker tool, which should accept four command line arguments. [80%]

   cracker *password_file results_file timeout salt*

   file_name - The file to read passwords from.

   results_file – The file to write cracked passwords to

timeout – How many seconds should the cracker run for.

salt - The salt to append to tried passwords. Omitting the argument means there is no salt.

When the cracker completes it should print out how long it run, how many different passwords it generated and tried, and how many user passwords it attempted to crack using at least one generated password. Note that you will not be graded based on the number of passwords cracked, however, the efficiency and speed of your solution will be taken into account.

2. Include a report.txt file with a paragraph explaining your choices when building the cracker and including instructions for building your software. Also describe, what could system designers do to significantly hinder cracking efforts from anyone that obtains their password database. What does adding the salt achieve? [20%]

Submission information

The code you submit for grading must build and run on the linux-lab, even if you use a different machine/environment for developing. Note that linux-lab.cs.stevens.edu is just an alias that connects you to one out of many hosts. Always, make sure you are not messing with anyone else's assignment.

The assignment must be done in C, C++, or Java. All the code must belong to you with the exception of certain built-in libraries (e.g., Java packages part of the JDK) and standard libraries (e.g., for C++). The elegance of the solution will also be taken into account when grading. Permission by the instructor is required for deviating from these rules.

Submit all your files as a tar.gz archive through Canvas.

Helpful information

Dictionaries can be found in: https://wiki.skullsecurity.org/Passwords

Leetspeak information http://en.wikipedia.org/wiki/Leet