Cybersecurity and Defense-in-Depth

Lab 1: Password Cracking
Prepared by: Ahmed Hussein
TA: Mina Sadik
Student(s) per team: 1-3 students (2 is recommended)
All teams submit by: 30 October 2024
Total points: 60
Total possible points: 70

Usage simplicity and ease of implementation makes passwords the most used technique for user authentication. As the password length increases and the password includes characters from a larger set of choices, the potential number of passwords would be large enough to make brute-force attack infeasible even on a supercomputer. However, most users choose easily guessed passwords which makes them vulnerable. Crackers often utilize dictionaries of commonly used and leaked passwords, which are available online, to break passwords and gain initial access to vulnerable accounts. In this lab, you will experiment with manual cracking, then you are asked to automate the cracking process by writing a program. You will also learn about offline dictionary attack.

In the following questions, it is recommended to use Python. However, feel free to use any preferred programming language to complete this lab.

Question 1: (10 points)

In this lab, we will examine and try to crack a simple login program, *Q1/Login.pyc*. This login program takes two arguments namely, username and password. Before cracking this program, first look at *Q1/LoginTemplate*.py script to understand how this login program works. Note that *hashlib* Python library has been imported and the SHA-256 algorithm is used for password hashing. Assume that user `Magdy` is using a very weak password. Try to crack Magdy's password by trying the passwords in the given short list of commonly used passwords stored in *Q1/MostCommonPWs* file. Write a simple script to find and report Magdy's password. Also, report the run time, that is the time taken for cracking his password. Verify the password is correct by manually calling the login executable.

Question 2: (10 points)

Assume that a user from the short users list stored at *Q2/users* (other than Magdy), is using a weak password as well. Write another script that will find and report that user along with the corresponding password. Also, report the run time and verify your answer by calling the login script.

Question 3: (10 points)

Assume that another user (other than those already exposed), is using a password among the top 100,000 leaked passwords. Write a program to launch a dictionary attack on the login script to discover and report that user and the cracked password. Use `Q3/LeakedPWs100k` as the dictionary file. Don't forget to print the runtime as well.

Question 4: (10 points)

In this question, you are given an exposed passwords file, *Q4/PwnedPWfile*. Assume that an additional user has reused one of the passwords listed in *Q4/PwnedPWfile*. Write a program to find this user and as usual report the execution time and the cracked password.

Question 5: (10 points)

In this question, you are given an exposed hashed passwords file, *Q5/HashedPWs*. Write a program that uses this hashed password file to find credentials for additional users (from *users* file). Assume that users have picked a random password from *Q5/LeakedPWs100k* and concatenated to it two random digits. As usual, report the execution time, exposed user and password, and verify using the login script.

Question 6: (10 points)

In this question, you are given a file *Q6/SaltedPWs* which lists, for each user *x*, the pair ($salt_x$,$h(salt_x + PW_x)$). Recall that $salt_x$ is a random value picked for user *x*. Note that the + sign denotes string concatenation. Write a program that uses the file *SaltedPWs* to find the passwords of an additional user that uses a password from Q6/*LeakedPWs100k* concatenated with one random digit.

Bonus (10 points): Use multi-threading for faster execution of the slowest program of your choice among the previous questions.