Brute Force Cracker & Dictionary Attack Report

Since a typical brute force password cracker takes an incredibly long time to decipher a password, I decided to implement a brute force/dictionary attack hybrid to help facilitate the crack speed. I attempted to use Kali Linux’s ‘rockyou.txt’ file that contains 14 million passwords but encountered an issue when trying to encode in UTF-8. I instead went with another file of about 10 million passwords and gathered six passwords from the beginning, middle, and end of it. After running each password these were the times that were recorded, and as expected passwords found toward the end of the file took longer to crack than those in the middle and beginning. If a password is not found in the file, then the brute force cracker takes over.

The brute force cracker’s main issue is prevalent due to the amount of guessing that can be done before the password can be successfully cracked. The number of guesses needed can be calculated by using the formula **c = (m^n)/2** where (m) is the possibilities for each character, (n) is the amount of characters in the password, and (c) is the estimated number of guesses. For example, a password that is 8 characters long and contains an upper-case letter, lower case letter, and number then 1.0917E+14 guesses can be made. If we assume that it takes 1 millisecond per guess, then this password would take ~3549 years to crack. In my implementation of the brute force cracker I gave the user the option to only use certain character types in the search but guesses still take a long time to guess. For my brute force guessing experimentation, I had decided to just ask for one of my roommate’s passwords and see how long it took to crack it. The password they provided me was nine characters long with one upper-case at the beginning followed by seven lower-case letters and a number. I had ran the brute forcer for about 4 days to no avail, but it did bring my interest into how efficient different hardware is at computing the guesses. On average my laptop can make about 30,000 guesses per second when it is plugged in, but if I run the program using my NVidia GTX 1080 graphics card to run the program it can make 3,449,247 guesses per second on average which is about 100 times more efficient.