ITP 125 FInal Report - Yusuke Kobayashi

Results

Cracked => Z || Time Required => <0.0001s
Cracked => AD || Time Required => 0.001s
Cracked => God || Time Required => 0.0802s
Cracked => 1234 || Time Required => 8.1684s
Cracked => AbCdE || Time Required => 237.0784s
(Waited for more than 24 hours but was not able to crack the rest)
(When I ran it without the time calculation, it cracked the Trojan in around 24 hours.)

Analysis

<u>1: Longer the length, Longer it takes + Complexity</u>

First, these are the factors considered when calculating the time it takes to crack the password.

1. **Characters** - the type of characters consists password(lowercase, uppercase, digits, symbols)
2. **Length** - the length of the password
3. **Hashing Type** - hashing type(in this case MD5)
4. **Hashrate** - how many hashes the PC can hash per second (depends on each machine.)
5. **Algorithm** - algorithm to crack the password(brute force, dictionary attack, the way the system generates passwords, etc…)

Second, this is the calculation formula to get the time it takes in a brute force attack. (Check all the combinations)

Conditions:

Characters = **95**
Lower letters (‘a’ to ‘z’ => 26)
Upper letters(‘A’ to ‘Z’ => 26)
Numbers (‘0’ to ‘9’ = > 10)
Symbols(33)
Length = **L** characters
HashRate = **H** hashes(per second)
Time Required = **T** seconds

With the above conditions, a simple calculation formula would be the following.

T = pow(95, L) / H

Now, with this formula, it is obvious that **the more letters, the longer it takes**. In addition to that, since in this case, we are considering **95** letters, however, if attackers know that the password only consists of (Lower letters => 26), the time it takes to crack significantly decreases. This means t**he more complex the password, the longer it takes to crack.**

2. How To Make The Cracking Faster

There are multiple ways to make the cracking faster.

A: Utilize multi-core CPU /GPU

From the calculation formula, it is clear that if we simply boost the hash rate, it will decrease the time. By utilizing, a multi-core CPU, the processing speed will become faster since each core does the task separately. This makes the hash rate faster. In terms of GPU, it uses a process called "parallel computing", which processes the task simultaneously. This also makes the hash rate faster.

B: Change Algorithm

The main reason why the script takes much time is because it checks every single possible combination of the characters. So, if we change the algorithm of checking every single combination to the only possible combination, the time it takes will significantly decrease.
For example, if we know beforehand that passwords consist of 8 characters and require at least one capital letter and number, then we can assume that it is likely that the password will be something like "Jack1230" (Name + Birthday). So if we write a program that checks the combination of Capital letters only for the first character (26 possibilities) 3 lower letters(26), and 4 numbers(10).
Possible Combinations would be pow(26,4) * pow(10, 4) total combinations, which is significantly less than normal 8 letter combinations, pow(95,8)
In this way, by modifying the algorithm, the time it takes will decrease.

C: Try Other Attacking Methods

To crack password way faster, using other attacking methods are effective.
In the case of a dictionary attack, where the system hashes possibly given a combination of the passwords and checks with the hashes, the time it takes to crack the password is way faster than the method that I mentioned in the previous chapter. Because in this way, the system does not need to check the combinations which is unlikely to be the password. For example, the algorithm that I used as an example in the previous chapter checks combinations like Npgl9878 (which is unlikely to be the password when we are considering the Name + Birthday combination) By doing dictionary attacks, the system can effectively check only the possible combinations, removing a significant amount of tasks. In addition, attacking method like rainbow tables, which does not require a hashing process, is effective too.

Sources/References Used for this project:

itertools — Functions creating iterators for efficient looping — Python 3.12.0 documentation

How to get MD5 Sum of a String in Python? - Studytonight.

Python time Module (with Examples)

What is GPU Parallel Computing? | OpenMetal IaaS.

How Long Does It Take a Hacker to Brute Force a Password in 2023 - NetSec.News

Brute Force Algorithms Explained.