Name Punnapob Jindakul **Student ID** 6330320221

Activity I : Hacking Password

Exercises

1. Write a simple python program to use the word from the dictionary to find the original value of **d54cc1fe76f5186380a0939d2fc1723c44e8a5f7**. Note that you might want to include substitution in your code (lowercase, uppercase, number for letter [‘o’ => 0 , ‘l’ => 1, ‘i’ => 1]).

```python
import hashlib
from itertools import product

checkHash = "d54cc1fe76f5186380a0939d2fc1723c44e8a5f7"

sub = {
    "o": "0",
    "l": "1",
    "i": "1",
}


def hash(word):
    return hashlib.sha1(word.encode("utf-8")).hexdigest()


def checkHash(text, checkHash=checkHash):
    return hash(text) == checkHash


def genCaseCombination(text):
    char_combinations = [(char.lower(), char.upper()) for char in text]
    combinations = product(*char_combinations)
    return ["".join(item) for item in combinations]


def getSubsitueCombintion(text):
    char_combinations = []
    for char in text:
        if char in sub:
            char_combinations.append((char, sub[char]))
        else:
            char_combinations.append((char,))
    combinations = product(*char_combinations)
    return ["".join(item) for item in combinations]
```

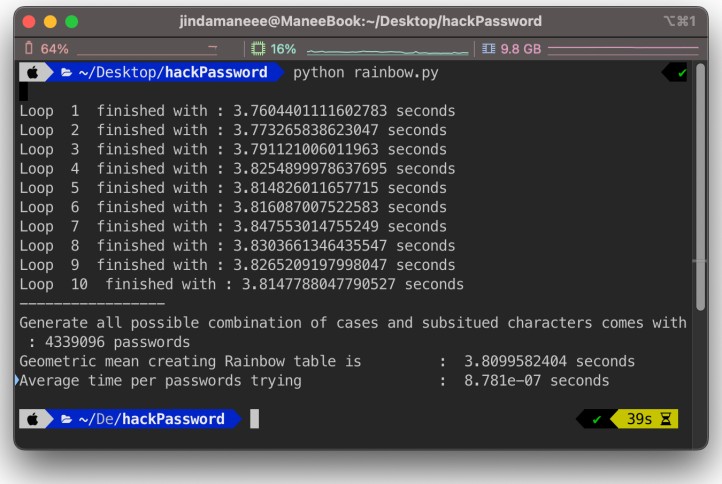
```python
from common import *

lines = []
with open("10k.txt", "r") as f:
    lines = f.readlines()

for line in lines:
    line = line.strip()
    for caseList in genCaseCombination(line):
        for subWord in getSubsitueCombintion(caseList):
            if checkHash(subWord):
                print("--------------------")
                print("Found!", subWord)
                print("--------------------")
                exit(0)
    print("Incorrect!", line)
print("Not found!")
```

```
jindamaneee@ManeeBook:~/Desktop/hackPassword
65%  10%  9.1 GB
Incorrect! ronald
Incorrect! asdf1234
Incorrect! harrison
Incorrect! trigger
Incorrect! truck
Incorrect! danny
Incorrect! home
Incorrect! winnie
Incorrect! beauty
--------------------
Found! ThaiLanD
--------------------
~/Desktop/hackPassword
```

Plaintext for this hash is **ThaiLanD.**

2. For the given dictionary, create a rainbow table (including the substituted strings) using the sha1 algorithm. Measure the time for creating such a table. Measure the size of the table.

```python
from common import *
import time

LOOP_COUNT = 10
DISPLAY_FLOATING_DIGITS = 10

lines = []

with open("10k.txt", "r") as f:
    allWord = f.readlines()


def measureTime():
    rainbow = open("rainbow.txt", "w")
    start = time.time()
    for line in allWord:
        line = line.strip()
        for caseList in genCaseCombination(line):
            for subWord in getSubsitueCombintion(caseList):
                hashed = hash(subWord)
                rainbow.write(subWord + " : \t" + hashed + "\n")
    end = time.time()
    rainbow.close()
    return end - start


timeList = []
for i in range(LOOP_COUNT):
    currTime = measureTime()
    timeList.append(currTime)
    print("Loop ", i + 1, " finished with :", currTime, "seconds")
geometricMean = 1
for i in timeList:
    geometricMean *= i
geometricMean = geometricMean ** (1 / LOOP_COUNT)

with open("rainbow.txt", "r") as f:
    allRainbow = f.readlines()

hashGenerated = len(allRainbow)

print("-----------------")
print(
    "Generate all possible combination of cases and subsitued characters comes with :",
    hashGenerated,
    "passwords",
)
print(
    "Geometric mean creating Rainbow table is \t : ",
    round(geometricMean, DISPLAY_FLOATING_DIGITS),
    "seconds",
)
print(
    "Average time per passwords trying \t\t : ",
    round(geometricMean / hashGenerated, DISPLAY_FLOATING_DIGITS),
    "seconds",
)
```



Size of rainbow table is **4339096 lines** which is **294.4 MB.**

Geometric mean for creating rainbow table is **3.80996 seconds** evaluated from 10 tests.

3. Based on your code, how long does it take to perform a hash (sha1) on a password string? Please analyze the performance of your system.

   From the exercise 2,

   The rainbow table contains **4339096 lines** of hash and their plaintext
   which take around **3.80996 seconds** to create.
   So, for one try hashing, it needs to take $\frac{\mathbf{3.80996}}{\mathbf{4339096}}$ **= 8.781e-7 seconds.**

4. If you were a hacker obtaining a password file from a system, estimate how long it takes to break a password with brute force using your computer. (Please based the answer on your measurement from exercise #3.)

   From the exercise 3,

   For one try hashing, it needs to take **8.781e-7 seconds.**
   Define,
   C = The possible characters for creating passwords
   N = The number of digits in a password
   Therefore, it will take around $\mathbf{8.781 \times 10^{-7} \times C^N}$ **seconds** to decrypt.

5. Base on your analysis in exercise #4, what should be the proper length of a password. (e.g. Take at least a year to break).

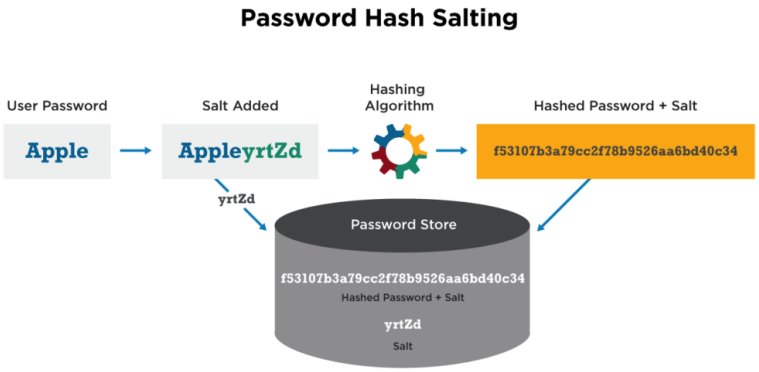
   From the exercise 4, for one password, we need $\mathbf{8.781 \times 10^{-7} \times C^N}$ seconds to decrypt.
   The possible characters for creating a password are
   - Lowercase letters (a-z): **26** characters
   - Uppercase letters (A-Z): **26** characters
   - Digits (0-9): **10** characters
   - Special characters (e.g., !, @, #, $, etc.)
     From [Password Special Characters | OWASP Foundation] there are around **33** characters.

   So, there are **95** characters for creating a password.
   And there are **31,536,000** seconds in 1 year.

   Therefore, it will take $\frac{\log_{\frac{\mathbf{31,536,000}}{\mathbf{8.781 \times 10^{-7}}}}}{\log 96}$ = 6.83825 ≈ **7 digits**

6. What is salt? Please explain its role in protecting a password hash

   Salt เป็นค่าหนึ่งที่ generate ขึ้นมาแล้วนำไป hash พร้อมกับ password ซึ่งค่า salt นี้จะเป็นค่า unique ของแต่ละ password เวลาเก็บใน database password ก็จะเก็บในลักษณะ salt กับ hash ต่อกัน ทำให้แม้ database หลุดไป hacker ก็จะรู้แค่ salt กับ hash แต่ว่า hash นี้เป็น hash ของ salt กับ password ทำให้ rainbow table ที่ compute ไว้ก่อนหน้าที่มักจะเป็น password ทั่วๆไป ไม่สามารถนำมาใช้ถอดค่า hash ออกได้ แต่อย่างไรก็ตาม hacker ยังสามารถนำค่า salt ที่รู้ไป brute-force ต่อได้ ซึ่งก็ใช้เวลานานอยู่ดีหาก password นั้นยาว



รูปจาก [Password Salting - CyberHoot Cyber Library]