6331322721 Pawan Kanjeam

Computer Security
Dept. of Computer Engineering,
Chulalongkorn University.

Activity I : Hacking Password

Created by : Krerk Piromsopa, Ph.D

Overviews

This activity demonstrates the fundamentals of password security. Several hacking techniques will be demonstrated throughout the exercises. In particular, we will learn: brute-force attack, rainbow-table attack, and password analysis.

We will use a free password dictionary from the given url as our dictionary.
https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10k-most-common.txt

Exercises

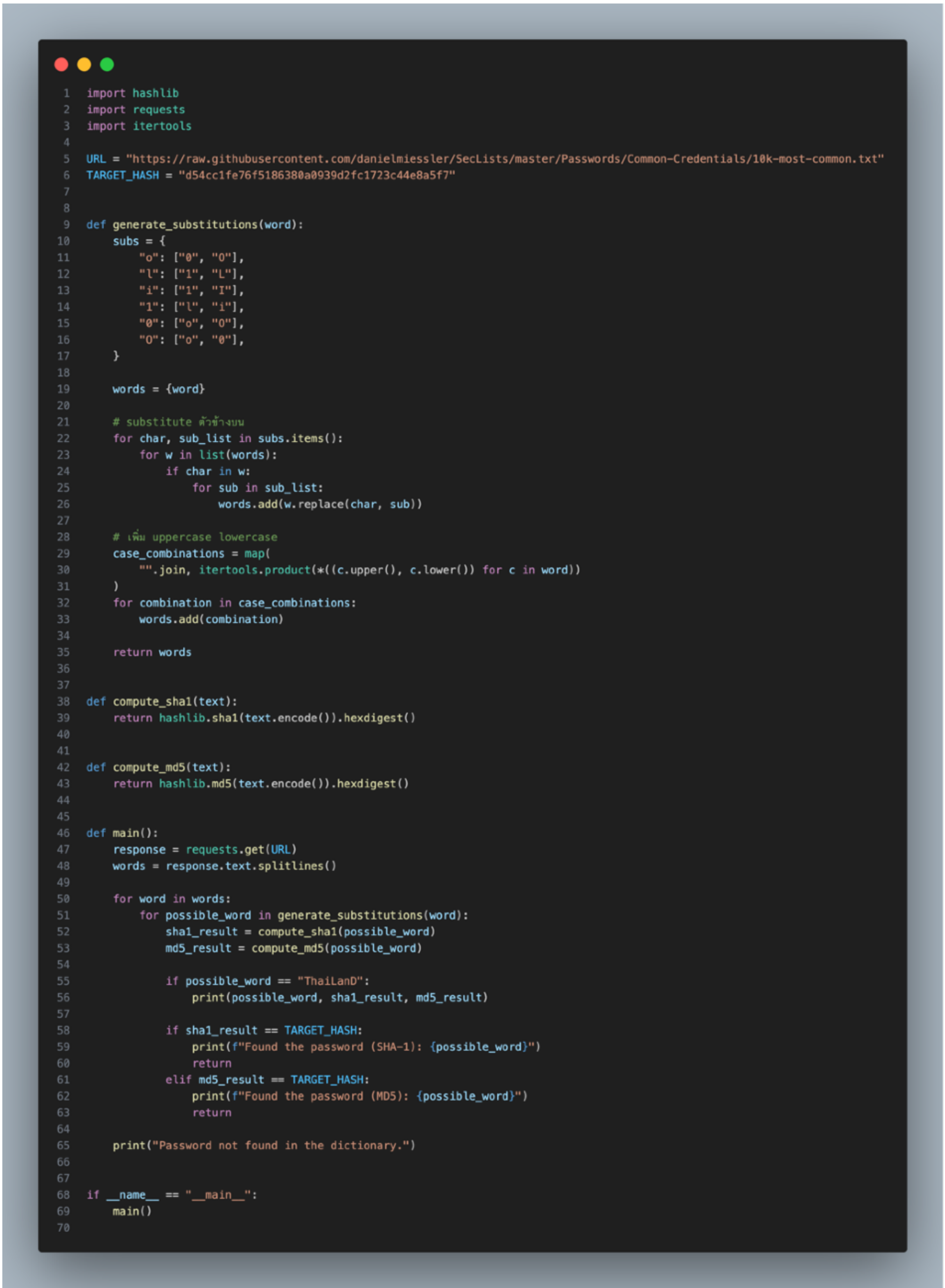
1. Write a simple python program to use the word from the dictionary to find the original value of **d54cc1fe76f5186380a0939d2fc1723c44e8a5f7**.
Note that you might want to include substitution in your code (lowercase, uppercase, number for letter ['o' => 0 , 'l' => 1, 'i' => 1]).
Hint: Here is a snippet for sha1 and md5 functions.

```
import hashlib
m=hashlib.sha1(b"Chulalongkorn").hexdigest()
print(m)
m=hashlib.md5(b"Chulalongkorn").hexdigest()
print(m)
```

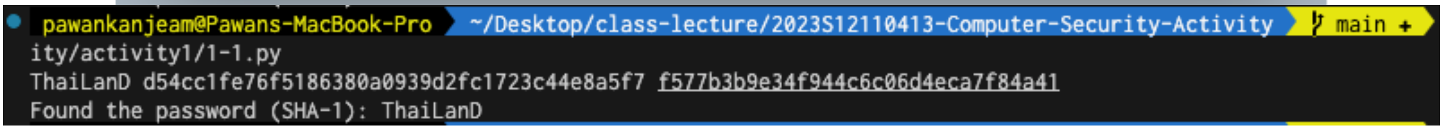
2. For the given dictionary, create a rainbow table (including the substituted strings) using the sha1 algorithm. Measure the time for creating such a table. Measure the size of the table.
3. Based on your code, how long does it take to perform a hash (sha1) on a password string? Please analyze the performance of your system.
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.)
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).
6. What is salt? Please explain its role in protecting a password hash.

https://github.com/pwanstax/2023S12110413-Computer-Security-Activity/tree/main/activity1

1.



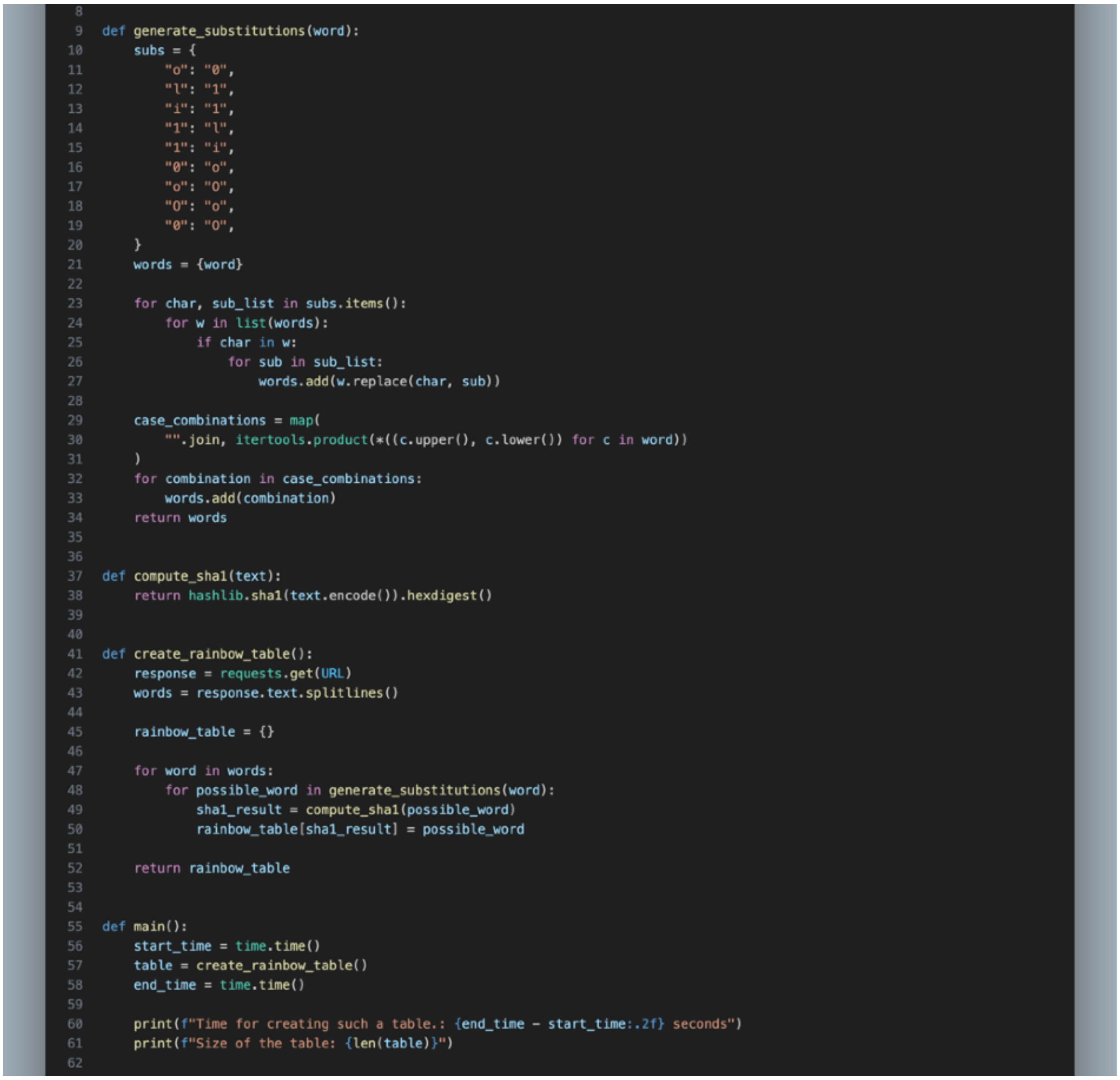
```python
import hashlib
import requests
import itertools

URL = "https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Common-Credentials/10k-most-common.txt"
TARGET_HASH = "d54cc1fe76f5186380a0939d2fc1723c44e8a5f7"


def generate_substitutions(word):
    subs = {
        "o": ["0", "O"],
        "l": ["1", "L"],
        "i": ["1", "I"],
        "1": ["l", "i"],
        "0": ["o", "O"],
        "O": ["o", "0"],
    }

    words = {word}

    # substitute ตัวข้างบน
    for char, sub_list in subs.items():
        for w in list(words):
            if char in w:
                for sub in sub_list:
                    words.add(w.replace(char, sub))

    # เพิ่ม uppercase lowercase
    case_combinations = map(
        "".join, itertools.product(*((c.upper(), c.lower()) for c in word))
    )
    for combination in case_combinations:
        words.add(combination)

    return words


def compute_sha1(text):
    return hashlib.sha1(text.encode()).hexdigest()


def compute_md5(text):
    return hashlib.md5(text.encode()).hexdigest()


def main():
    response = requests.get(URL)
    words = response.text.splitlines()

    for word in words:
        for possible_word in generate_substitutions(word):
            sha1_result = compute_sha1(possible_word)
            md5_result = compute_md5(possible_word)

            if possible_word == "ThaiLanD":
                print(possible_word, sha1_result, md5_result)

            if sha1_result == TARGET_HASH:
                print(f"Found the password (SHA-1): {possible_word}")
                return
            elif md5_result == TARGET_HASH:
                print(f"Found the password (MD5): {possible_word}")
                return

    print("Password not found in the dictionary.")


if __name__ == "__main__":
    main()

```



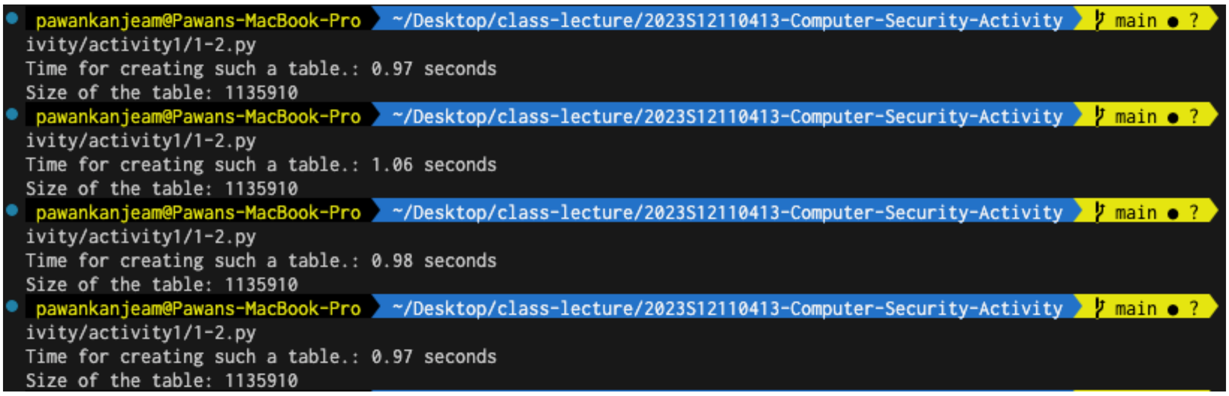
```
● pawankanjeam@Pawans-MacBook-Pro  ~/Desktop/class-lecture/2023S12110413-Computer-Security-Activity  main +
ity/activity1/1-1.py
ThaiLanD d54cc1fe76f5186380a0939d2fc1723c44e8a5f7 f577b3b9e34f944c6c06d4eca7f84a41
Found the password (SHA-1): ThaiLanD
```

https://github.com/pwanstax/2023S12110413-Computer-Security-Activity/tree/main/activity1

2.



```python

def generate_substitutions(word):
    subs = {
        "o": "0",
        "l": "1",
        "i": "1",
        "1": "l",
        "1": "i",
        "0": "o",
        "o": "O",
        "O": "o",
        "0": "O",
    }
    words = {word}

    for char, sub_list in subs.items():
        for w in list(words):
            if char in w:
                for sub in sub_list:
                    words.add(w.replace(char, sub))

    case_combinations = map(
        "".join, itertools.product(*((c.upper(), c.lower()) for c in word))
    )
    for combination in case_combinations:
        words.add(combination)
    return words


def compute_sha1(text):
    return hashlib.sha1(text.encode()).hexdigest()


def create_rainbow_table():
    response = requests.get(URL)
    words = response.text.splitlines()

    rainbow_table = {}

    for word in words:
        for possible_word in generate_substitutions(word):
            sha1_result = compute_sha1(possible_word)
            rainbow_table[sha1_result] = possible_word

    return rainbow_table


def main():
    start_time = time.time()
    table = create_rainbow_table()
    end_time = time.time()

    print(f"Time for creating such a table.: {end_time - start_time:.2f} seconds")
    print(f"Size of the table: {len(table)}")

```



```
● pawankanjeam@Pawans-MacBook-Pro  ~/Desktop/class-lecture/2023S12110413-Computer-Security-Activity  main ● ?
ivity/activity1/1-2.py
Time for creating such a table.: 0.97 seconds
Size of the table: 1135910
● pawankanjeam@Pawans-MacBook-Pro  ~/Desktop/class-lecture/2023S12110413-Computer-Security-Activity  main ● ?
ivity/activity1/1-2.py
Time for creating such a table.: 1.06 seconds
Size of the table: 1135910
● pawankanjeam@Pawans-MacBook-Pro  ~/Desktop/class-lecture/2023S12110413-Computer-Security-Activity  main ● ?
ivity/activity1/1-2.py
Time for creating such a table.: 0.98 seconds
Size of the table: 1135910
● pawankanjeam@Pawans-MacBook-Pro  ~/Desktop/class-lecture/2023S12110413-Computer-Security-Activity  main ● ?
ivity/activity1/1-2.py
Time for creating such a table.: 0.97 seconds
Size of the table: 1135910
```

https://github.com/pwanstax/2023S12110413-Computer-Security-Activity/tree/main/activity1

3. คำตอบจากข้อที่แล้ว
ใช้เวลาเฉลี่ย (0.97+1.06+0.98+0.97)/4 = 0.995 วินาที
ขนาด 1,135,910 hashes

แสดงว่าเวลาที่ใช้ต่อ 1 hash = 0.995/1,135,910 = 0.000000876 = 0.876 * 1^-6 = 0.876 microseconds
จึงสรุปได้ว่าระบบมีประสิทธิภาพสูงเนื่องจากใช้เวลาเพียง 0.876 microseconds ต่อ hash

4. ถ้าดูจาก password จะเห็นว่ามีทั้ง lowercase uppercase และ number
จึงได้ combination เป็น 26(lowercase) + 26(uppercase) + 10(number) = 62 แบบที่เป็นไปได้
โดยระยะเวลาที่ใช้ขึ้นอยู่กับความยาวของ password ด้วย

เวลาที่ต้องใช้จึงเท่ากับ = (62^n) * 0.000000876

เช่น ความยาว 4 จะมีทั้งหมด 62^4 = 14,776,336 hashes
จะต้องใช้เวลาทั้งหมด 14,776,336 * 0.000000876 ≈ 12.944 seconds เป็นต้น

5. แปลง 1 ปีให้อยู่ในหน่วยวินาทีเพื่อที่จะได้หาจะนวน combination ที่ต้องการ
1 year = 365 days = 8,760 hours = 525,600 minutes = 31,536,000 seconds

จะได้ว่า (62^n) * 0.000000876 = 31,536,000

ถ้าแทน n ด้วย 7 จะใช้เวลาประมาณ 3ล้านวินาที
แทน n ด้วย 8 จะใช้เวลาประมาณ 191 ล้านวินานที หรือประมาณ 6 ปี
ดังนั้นควรตั้งรหัสอย่างน้อย 8 หลัก

6. salt คือ ข้อมูลที่ใส่เพิ่มเข้าไปใน password ก่อนนำไป hash เพื่อให้โจมตีได้ยากยิ่งขึ้น

https://github.com/pwanstax/2023S12110413-Computer-Security-Activity/tree/main/activity1