

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.

1.

```

1  import hashlib
2  import requests
3  import itertools
4
5  URL = "https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Common-Credentials/10k-most-common.txt"
6  TARGET_HASH = "d54cc1fe76f5186380a0939d2fc1723c44e8a5f7"
7
8
9  def generate_substitutions(word):
10     subs = {
11         "o": ["0", "0"],
12         "l": ["1", "L"],
13         "i": ["1", "I"],
14         "1": ["l", "i"],
15         "0": ["o", "0"],
16         "O": ["o", "0"],
17     }
18
19     words = {word}
20
21     # substitute ตัวอักษร
22     for char, sub_list in subs.items():
23         for w in list(words):
24             if char in w:
25                 for sub in sub_list:
26                     words.add(w.replace(char, sub))
27
28     # เพิ่ม uppercase lowercase
29     case_combinations = map(
30         "".join, itertools.product(*(c.upper(), c.lower()) for c in word))
31     )
32     for combination in case_combinations:
33         words.add(combination)
34
35     return words
36
37
38  def compute_sha1(text):
39     return hashlib.sha1(text.encode()).hexdigest()
40
41
42  def compute_md5(text):
43     return hashlib.md5(text.encode()).hexdigest()
44
45
46  def main():
47     response = requests.get(URL)
48     words = response.text.splitlines()
49
50     for word in words:
51         for possible_word in generate_substitutions(word):
52             sha1_result = compute_sha1(possible_word)
53             md5_result = compute_md5(possible_word)
54
55             if sha1_result == TARGET_HASH:
56                 print(f"Found the password (SHA-1): {possible_word}")
57                 return
58             elif md5_result == TARGET_HASH:
59                 print(f"Found the password (MD5): {possible_word}")
60                 return
61
62     print("Password not found in the dictionary.")
63
64
65  if __name__ == "__main__":
66     main()

```

```

pawankanjeam@Pawans-MacBook-Pro ~/Desktop/class-lecture/2023S12110413-Computer-Security-Activity y/
y/activity1/1-1.py
Found the password (SHA-1): ThaiLanD

```

2.

```

8
9 def generate_substitutions(word):
10     subs = {
11         "o": "0",
12         "l": "1",
13         "i": "1",
14         "1": "l",
15         "1": "i",
16         "0": "o",
17         "o": "0",
18         "0": "o",
19         "0": "0",
20     }
21     words = {word}
22
23     for char, sub_list in subs.items():
24         for w in list(words):
25             if char in w:
26                 for sub in sub_list:
27                     words.add(w.replace(char, sub))
28
29     case_combinations = map(
30         "".join, itertools.product(*((c.upper(), c.lower()) for c in word))
31     )
32     for combination in case_combinations:
33         words.add(combination)
34     return words
35
36
37 def compute_sha1(text):
38     return hashlib.sha1(text.encode()).hexdigest()
39
40
41 def create_rainbow_table():
42     response = requests.get(URL)
43     words = response.text.splitlines()
44
45     rainbow_table = {}
46
47     for word in words:
48         for possible_word in generate_substitutions(word):
49             sha1_result = compute_sha1(possible_word)
50             rainbow_table[sha1_result] = possible_word
51
52     return rainbow_table
53
54
55 def main():
56     start_time = time.time()
57     table = create_rainbow_table()
58     end_time = time.time()
59
60     print(f"Time for creating such a table.: {end_time - start_time:.2f} seconds")
61     print(f"Size of the table: {len(table)}")
62

```

```

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

```

3. คำตอบจากข้อที่แล้ว

ใช้เวลาเฉลี่ย $(0.97+1.06+0.98+0.97)/4 = 0.995$ วินาที

ขนาด 1,135,910 hashes

แสดงว่าเวลาที่ใช้ต่อ 1 hash = $0.995/1,135,910 = 0.000000876 = 0.876 * 10^{-6} = 0.876$ microseconds

จึงสรุปได้ว่าระบบมีประสิทธิภาพสูงเนื่องจากใช้เวลาเพียง **0.876 microseconds ต่อ hash**

4. ถ้าดูจาก password จะเห็นว่ามีทั้ง lowercase uppercase และ number

จึงได้ combination เป็น $26(\text{lowercase}) + 26(\text{uppercase}) + 10(\text{number}) = 62$ แบบที่เป็นไปได้ โดยระยะเวลาที่ใช้ขึ้นอยู่กับความยาวของ password ด้วย

เวลาที่ต้องใช้จึงเท่ากับ $= (62^n) * 0.000000876$

เช่น ความยาว 4 จะมีทั้งหมด $62^4 = 14,776,336$ hashes

จะต้องใช้เวลาทั้งหมด $14,776,336 * 0.000000876 \approx 12.944$ seconds เป็นต้น

5. แปลง 1 ปีให้อยู่ในหน่วยวินาทีเพื่อที่จะได้หาจะนวน combination ที่ต้องการ

$1 \text{ year} = 365 \text{ days} = 8,760 \text{ hours} = 525,600 \text{ minutes} = 31,536,000 \text{ seconds}$

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

ถ้าแทน n ด้วย 7 จะใช้เวลาประมาณ 3 ล้านวินาที

แทน n ด้วย 8 จะใช้เวลาประมาณ 191 ล้านวินาที หรือประมาณ 6 ปี

ดังนั้นควรตั้งรหัสอย่างน้อย 8 หลัก

6. salt คือ ข้อมูลที่ใส่เพิ่มเข้าไปใน password ก่อนนำไป hash เพื่อให้โจมตีได้ยากยิ่งขึ้น