

# Activity I: Hacking Password

ในรายงานฉบับนี้จะมีบางส่วนของโปรแกรมปรากฏอยู่ ซึ่งเขียนด้วยภาษา Golang สามารถดู Code เต็ม ๆ ได้ที่ [2110413-COMP-SECURITY Activity 1](#)

## Table of Contents

- [Problem 1](#)
  - [Answer](#)
- [Problem 2](#)
  - [Answer](#)
- [Problem 3](#)
  - [Answer](#)
- [Problem 4](#)
  - [Answer](#)
- [Problem 5](#)
  - [Answer](#)
- [Problem 6](#)
  - [Answer](#)

## Problem 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]).

### Answer

จาก [Dictionary](#) ที่ได้มา พบว่า `ThaiLanD` มีค่า Hash เป็น `d54cc1fe76f5186380a0939d2fc1723c44e8a5f7` เมื่อใช้ SHA1 hashing function

โดยขั้นตอนแรกเราจะ load text เข้ามาใน program ก่อน จากนั้นทำการ **substitute** คำเหล่านั้นเป็นหลาย ๆ คำ เช่น

`hello` -> `HeEll0`

จากนั้นก็ใช้ MD5 และ SHA1 hashing function ลอง hash generated text แล้วลองเทียบกับ target ดู

Function ในการหาคำตอบ

```
func (d *Dictionary) FindOriginalPassword(targetHashedPwd string,
substitutedPasswords []string) (string, method.Method) {
    // Check with MD5 hash function
    if pwd, correct := password.CheckPasswords(substitutedPasswords,
targetHashedPwd, method.MethodMD5); correct {
        return pwd, method.MethodMD5
    }

    // Check with SHA1 Hash function
```

```

    if pwd, correct := password.CheckPasswords(substitutedPasswords,
targetHashedPwd, method.MethodSHA1); correct {
        return pwd, method.MethodSHA1
    }

    return "", ""
}

```

Function ในการหา `substitutedPasswords`

```

func (d *Dictionary) SubstituteText(text string) []string {
    results := make([]string, 0)
    queue := []Queue{
        {text, 0},
    }

    for len(queue) > 0 {
        target := queue[0]
        results = append(results, target.item)
        queue = queue[1:]
        for i := target.start; i < len(target.item); i++ {
            for _, p := range d.patterns {
                for pIdx, s := range p {
                    // Ignore the unmatched pattern
                    if s != string(target.item[i]) {
                        continue
                    }
                    // Change it to many form and then add them to queue
                    removedP := remove(p, pIdx)
                    for _, s := range removedP {
                        newText := target.item[:i] + s + target.item[i+1:]
                        queue = append(queue, Queue{newText, i + 1})
                    }
                }
            }
        }
    }

    return results
}

```

## Problem 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.

Answer

ในการสร้างตาราง Hash Table เราสามารถ map ได้ด้วยวิธีนี้

```
func createTable(words []string) map[string]string {  
    table := make(map[string]string)  
  
    for _, item := range words {  
        hashedItem := hasher.GetSHA1Hash(item)  
        table[item] = hashedItem  
    }  
  
    return table  
}
```

ซึ่งจากการทดลองในเครื่องพบว่าการสร้างตารางทั้งหมดใช้เวลาและขนาดดังนี้

```
-----  
Table Size: 12611732  
Create Rainbow Table elapsed time: 8.871671883s  
Average time per words: 703ns  
-----
```

### Problem 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.

Answer

Assume ว่า Password จะมีเฉพาะอักษร a-z หรือ A-Z เท่านั้น เมื่อลองใช้ Computer ของตัวเองในการ Hash Password ดังกล่าวด้วยความยาวที่ต่างกัน ได้ผลลัพธ์ดังนี้

```
Generate words of length 1 elapsed time: 8.954µs
Number of generated words: 52
Average generation time per words 172ns
Total hashing time: 15.551µs
Hash rate: 299ns
```

```
-----
Generate words of length 2 elapsed time: 170.267µs
Number of generated words: 2704
Average generation time per words 62ns
Total hashing time: 822.348µs
Hash rate: 304ns
```

```
-----
Generate words of length 3 elapsed time: 9.321671ms
Number of generated words: 140608
Average generation time per words 66ns
Total hashing time: 36.359109ms
Hash rate: 258ns
```

```
-----
Generate words of length 4 elapsed time: 624.203534ms
Number of generated words: 7311616
Average generation time per words 85ns
Total hashing time: 1.874895721s
Hash rate: 256ns
```

length	hash rate
1	299 ns
2	304 ns
4	258 ns
5	256 ns

สังเกตว่า Hash Rate หรือความเร็วในการ Hash text ต่อ 1 คำในแต่ละความยาวของคำมีค่าใกล้เคียงกัน ดังนั้นจึงขอสรุปว่า ระยะเวลาที่ใช้ในการ Hash Password 1 ตัวของ Computer เครื่องนี้มีเฉลี่ยประมาณ 280 ns

## Problem 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.)

Answer

จาก Assumption ของ Password ข้างต้น เราจะได้ว่าใน 1 อักษรจะมีความเป็นไปได้ของตัวอักษรทั้งหมด 52 ตัว

ซึ่งการ Bruceforce password จะใช้เวลาขึ้นอยู่กับความยาวของ password ที่ต้องการโจมตี

ดังนั้นเราสามารถทำตาราง "จำนวนคำเมื่อกำหนดความยาวของคำ" และ "คำนวณ Hash Time" ได้ดังตารางนี้

Note: hash time = words \* 280 ns

length	words	hash time
1	$52^1 = 52$	14.56 $\mu$ s
2	$52^2 = 2,704$	757.12 $\mu$ s
4	$52^3 = 140,608$	39.37 ms
4	$52^3 = 7,311,616$	2.04 s
5	$52^5 = 380,204,032$	1.77 min
6	$52^6 = 19,770,609,664$	1.54 hr
7	$52^7 = 1,028,071,702,528$	3.33 d
8	$52^8 = 53,459,728,531,456$	5.77 mo
9	$52^9 = 2,779,905,883,635,710$	25.02 yr

## Problem 5

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

Answer

จาก ตารางในข้อ 4 พบว่า ถ้าต้องการให้ password ใช้เวลาเดามากกว่า 1 ปีด้วย Bruceforce จะต้องใช้ Password ที่มีความยาวมากกว่า 8 ขึ้นไป

## Problem 6

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

Answer

Salt คือ random string ที่เติมเข้าไปใน password ก่อนจะ hash ซึ่งกระบวนการนี้จะการันตีว่า salted password ที่ได้จะ unique เสมอ แม้ว่าจะมี original password เดียวกันก็ตาม โดย salt จะสร้างมาจาก cryptographically secure function ซึ่งจะทำให้การโดนโจมตีด้วย Bruceforce / Hash table ทำได้ยากขึ้น เช่น

- Password: Hello
- Salt: Fopm3
- Salted Password: Fopm3Hello
- Hashed Password:  
7528ed35c6ebf7e4661a02fd98ab88d92ccf4e48a4b27338fcc194b90ae8855c