

Soong Cun Yuan /1002074

Instructor's Name

19 February 2019

Security Lab3

Brute-Force and Dictionary Attack

Brute force Attack

Create a python script to find corresponding inputs by computing hash values for each possible combination. For this exercise, only require to consider passwords with 5 lowercase and or numeric characters.

```
{'aseas': 'ddaaafa5d551a582bc924d09cc8d33ee5', 'cance': '96f6065d8f2dd1376eff88fba65d1d83', 'di5gv': '836626589007d7d5304c8d22815fffc', 'dsmt0': 'a74edf83748e3c4fa5f31ec10bad79db', 'egunb': '1b31905c59f481958d2eb72158c27ac7', 'hed4e': 'a8218c67a5b4e652e30a59372e07df59', 'lou0g': '81466b6bb4be5a48e2230be1338bcde6', 'mlhdi': '6e313b70d12de950443527a33d802b76', 'nized': '78c1b8edd1bc3ffc438432479289a9e1', 'ofr0r': 'de952f5454fb0ee79bca249f80e9fe8f', 'opmen': 'a92b66a9802784ca8616c4b092378272', 'owso9': '644674d142ba2174a80889f833b32563', 'sso55': '1b4babaa3ae3be69857b323cf6b7fc80', 'tpoin': '0d5b558d5f6744deaaf5b016c6c77a57', 'tthel': 'd4efdba5e9725e77c9b9051fa8136f0a'}['aseas', 'cance', 'di5gv', 'dsmt0', 'egunb', 'hed4e', 'lou0g', 'mlhdi', 'nized', 'ofr0r', 'opmen', 'owso9', 'sso55', 'tpoin', 'tthel']The time it takes to run is 174.053426188 seconds.  
real    2m54.146s  
user    2m16.541s  
sys     0m31.058s  
Zengers-MacBook-Pro:Lab3 zenger$
```

The output of the brute force method is shown above, it took 174 seconds to run the code. The brute force method is shown below.

```
def bruteForce():
    ans={}
    counter=0
    for a in range(len(key)):
        tried[0] = key[a]
        for b in range(len(key)):
            tried[1] = key[b]
            for c in range(len(key)):
                tried[2] = key[c]
                for d in range(len(key)):
                    tried[3] = key[d]
                    for e in range(len(key)):
                        tried[4] = key[e]
                        new_try = ''.join(tried)
                        new_try_code = hashlib.md5(new_try.encode('utf-8')).hexdigest()
                        print(new_try_code)
                        if new_try_code in hashlist:
                            print("====")
                            print("{} : {}".format(new_try,new_try_code))
                            print("====")
                            ans[new_try]=new_try_code
                            counter+=1
                            if counter>=15:
                                # Return a dictionary of the unciphered clear password and
                                print(ans)
                                print(list(ans.keys()))
                                return 0
```

3b. Dictionary Attack

Dictionary attack is a form of brute force attack for deciphering. The difference is that dictionary attack on try those possibilities which are deemed likely to succeed.

My dictionary attack took short time to find 14/15 of the passwords, however was unable to find the 15th password. This may be because the 15th password is not a common word, and not included inside words5.txt. Hence not able to find it at all.



```
dictAttack.py
1  #!/usr/bin/python3
2
3  import hashlib
4  import itertools
5
6  dgts = set()
7  f5 = open('hash5.txt')
8  for d in f5:
9      dgts.add(d.strip())
10 f5.close()
11
12 fout = open('FOUND.txt', 'w')
13 foundCount = 0
14 digits = '0123456789'
15 fin = open('words5.txt')
16 checkedCharSet = set()
17 alreadyChecked = set()
18 wc = 0
19
20 for w in fin:
21     wc += 1
22     print(wc)
23     w = w.strip()
24     sortedchars = tuple(sorted(w)) # If the character are the same, skip
25     if sortedchars in checkedCharSet:
26         continue
27     else:
28         checkedCharSet.add(sortedchars)
29
30     for t in itertools.permutations(w+digits, 5):
31         ww = ''.join(t)
32         if ww in alreadyChecked:
33             continue
34         md5 = hashlib.md5(ww.encode('ascii')).hexdigest()
35         if md5 in dgts:
36             print('----->found:', ww)
37             print(ww, '-->', md5, file=fout, flush=True)
38             foundCount += 1
39             if foundCount == 15:
40                 break
41         alreadyChecked.add(ww)
```

For each word in words5.txt, we permute it and hash it, to find if there is a match, if there is, we found the hashed words. We also need to ensure there is no repeated words when the word is permuted, this to save time-complexity.

4. Creating Rainbow Table

Generating Rainbow Table

```
zenger@zenger-VirtualBox:~/Downloads/rainbowcrack-1.7-linux64$ ./rtgen md5 loweralpha-numeric 5 5 0 3800 600000
rainbow table md5_loweralpha-numeric#5-5_0_3800x600000_0.rt parameters
hash algorithm:      md5
hash length:        16
charset name:       loweralpha-numeric
charset data:        abcdefghijklmnopqrstuvwxyz0123456789
charset data in hex: 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 30 31 32 33 34 35 36 37 38 39
charset length:     36
plaintext length range: 5 - 5
reduce offset:      0x00000000
plaintext total:    60466176

sequential starting point begin from 0 (0x0000000000000000)
generating...
196608 of 600000 rainbow chains generated (0 m 23.0 s)
393216 of 600000 rainbow chains generated (0 m 22.9 s)
589824 of 600000 rainbow chains generated (0 m 23.0 s)
600000 of 600000 rainbow chains generated (0 m 1.2 s)
```

Sorting Rainbow Table

```
zenger@zenger-VirtualBox:~/Downloads/rainbowcrack-1.7-linux64$ ./rtsort .
./md5_loweralpha-numeric#5-5_0_3800x600000_0.rt:
3088826368 bytes memory available
loading data...
sorting data...
writing sorted data...
```

Rcrack the md5 ciphertext

```
zenger@zenger-VirtualBox:~/Downloads/rainbowcrack-1.7-linux64$ ./rcrack . -l hash5.txt
1 rainbow tables found
memory available: 2473793945 bytes
memory for rainbow chain traverse: 60800 bytes per hash, 912000 bytes for 15 hashes
memory for rainbow table buffer: 2 x 9600016 bytes
disk: ./md5_loweralpha-numeric#5-5_0_3800x600000_0.rt: 9600000 bytes read
disk: finished reading all files
plaintext of 81466b6bb4be5a48e2230be1338bcde6 is lou0g
plaintext of a92b66a9802704ca8616c4b092378272 is opmen
plaintext of 836626589007d7dd5304c8d22815ffff is di5gv
plaintext of 1b21905c59f481958d2eb72158c27ac7 is equnb
plaintext of 1b4babaa3e3be69857b323cf6b7fcdb0 is ss055
plaintext of 78c1b8edd1bc3ffc438432479289a9e1 is nized
plaintext of 0d5b558d5f6744deaf5b016c6c77a57 is tpoin
plaintext of a8218c67a5b4e652e30a59372e07df59 is hed4e
plaintext of a74edf83748e3c4fa5f31ec10bad79db is dsmto
plaintext of ddaafa5d551a582bc924d09cc8d33ee5 is aseas
plaintext of 6e313b70d12de950443527a33d802b76 is mlhdi
plaintext of d4efdba5e9725e77cb90051fa8136f0a is ttbel
plaintext of 644674d142ba2174a0889f833b2563 is ows09
plaintext of de952f5454fb0ee79bca249f80e9fe8f is ofror
plaintext of 96f60065d8f2dd1376eff88fba65d1d83 is cance

statistics
-----
plaintext found:          15 of 15
total time:              5.86 s
time of chain traverse:  3.90 s
time of alarm check:    1.89 s
time of disk read:      0.01 s
hash & reduce calculation of chain traverse: 108243000
hash & reduce calculation of alarm check:   41924830
number of alarm:         144576
performance of chain traverse: 27.74 million/s
performance of alarm check: 22.21 million/s

result
-----
a92b66a9802704ca8616c4b092378272  opmen  hex:6f706d656e
d4efdba5e9725e77cb90051fa8136f0a  ttbel   hex:747468656c
96f60065d8f2dd1376eff88fba65d1d83  cance   hex:63616e6365
78c1b8edd1bc3ffc438432479289a9e1  nized   hex:6e697a6564
0d5b558d5f6744deaf5b016c6c77a57  tpoin   hex:74706f696e
ddaaaf5d51a582bc924d09cc8d33ee5  aseas   hex:6173656173
a74edf83748e3c4fa5f31ec10bad79db  dsmto   hex:64736d746f
1b21905c59f481958d2eb72158c27ac7  equnb   hex:65677556e62
6e313b70d12de950443527a3d802b76  mlhdi   hex:6d6c686469
de952f5454fb0ee79bca249f80e9fe8f  ofror   hex:6f66726f72
a8218c67a5b4e652e30a59372e07df59  hed4e   hex:6865643465
836626589007d7dd5304c8d22815ffff  di5gv   hex:6469356776
644674d142ba2174a0889f833b2563  ows09  hex:6f77736f39
1b4babaa3e3be69857b323cf6b7fcdb0  ss055   hex:73736f3535
81466b6bb4be5a48e2230be1338bcde6  lou0g   hex:6c6f753067
```

5. Salting

Salting is an additional add-on to passwords, that concatenate a random string (salt) to the password before hashing it. This way, it allows for later authentication without keeping and therefore risking the plaintext password.

Salts help to defend hashed passwords against dictionary attacks or a generally pre-computed rainbow table attack. Since salts are different from each case, they also protect common passwords.

In this exercise I used my salting python script to concatenate a single random lowercase alphabet to the plaintext password, as you can see below even when it is cracked by rainbowcrack 6/15 of the passwords, the passwords are still tainted with a random character at the back. The hacker would have to guess what the salt is, and how many character of salt the hash-owner have added, this however some flaws which is for certain examples.

Flaws

Lets say I had a password that is “ILikeApples”, and after concatenate a random string /salt of “3e5d” it becomes “ILikeApples3e5d”, the hacker can guess that “3e5d” is just a random string after using brute-force/rainbow table.

```
zenger@zenger-VirtualBox:~/Downloads/rainbowcrack-1.7-linux64$ ./rcrack . -l hash5.txt
1 rainbow tables found
memory available: 2473793945 bytes
memory for rainbow chain traverse: 60800 bytes per hash, 912000 bytes for 15 hashes
memory for rainbow table buffer: 2 x 9600016 bytes
disk: ./md5_loweralpha-numeric#5-5_0_3800x600000_0.rt: 9600000 bytes read
disk: finished reading all files
plaintext of 81466bb6bb4be5a48e2230be1338bcde6 is lou0g
plaintext of a92b66a9802704ca8616c4b092378272 is opmen
plaintext of 836626589007d7dd5304c8d22815fffc is di5gv
plaintext of 1b31905c59f481958d2eb72158c27a7 is egunb
plaintext of 104babaa3e3be69857b323cf6b7fc80 is ss055
plaintext of 78c1b8edd1bcffc438432479289a9e1 is nized
plaintext of 0d5b558df6744deaa75b016cc77a57 is tpoin
plaintext of a8218c67a5b4e652e30a59372e07df59 is hed4e
plaintext of a74edf83748e3c4fa5f31ec10bad79db is dsmt0
plaintext of ddaafa5d51a582cb924d09c8d33e5 is aseas
plaintext of 6e313b70d12de950443527a3d802b76 is mlhdi
plaintext of d4efdbae59725e77c99051fa8136f0a is ttbel
plaintext of 644674d142ba174a800889f833b2563 is ows09
plaintext of de952f5454fb0ee79bca249f80e9fe8f is ofrror
plaintext of 96f0065d8f2dd1376eff88fba65d1d83 is cance

statistics
-----
plaintext found: 15 of 15
total time: 5.86 s
time of chain traverse: 3.90 s
time of alarm check: 1.89 s
time of disk read: 0.01 s
hash & reduce calculation of chain traverse: 108243000
hash & reduce calculation of alarm check: 41924830
number of alarm: 144576
performance of chain traverse: 27.74 million/s
performance of alarm check: 22.21 million/s

result
-----
a92b66a9802704ca8616c4b092378272 opmen hex:6f706d6566
d4efdbae59725e77c99051fa8136f0a ttbel hex:747468656c
96f0065d8f2dd1376eff88fba65d1d83 cance hex:63616e6365
78c1b8edd1bcffc438432479289a9e1 nized hex:6e697a6564
0d5b558df6744deaa75b016cc77a57 tpoin hex:74706f696e
ddaafa5d51a582cb924d09c8d33e5 aseas hex:6173656173
a74edf83748e3c4fa5f31ec10bad79db dsmt0 hex:64736d746f
1b31905c59f481958d2eb72158c27a7 egunb hex:6567756e62
6e313b70d12de950443527a3d802b76 mlhdi hex:606c686469
de952f5454fb0ee79bca249f80e9fe8f ofrror hex:6f66726f72
a8218c67a5b4e652e30a59372e07df59 hed4e hex:6056543465
836626589007d7dd5304c8d22815fffc di5gv hex:6469356776
644674d142ba174a800889f833b2563 ows09 hex:6f77736f39
1b4babaa3e3be69857b323cf6b7fc80 ss055 hex:73736f3535
1466bb6bb4be5a48e2230be1338bcde6 lou0g hex:6c6f753067
```

```
zenger@zenger-VirtualBox:~/Downloads/rainbowcrack-1.7-linux64$ ./rcrack . -l hash5.txt
2 rainbow tables found
memory available: 2461754982 bytes
memory for rainbow chain traverse: 60800 bytes per hash, 912000 bytes for 15 hashes
memory for rainbow table buffer: 2 x 9600016 bytes
disk: ./md5_loweralpha-numeric#5-5_0_3800x600000_0.rt: 9600000 bytes read
disk: ./md5_loweralpha-numeric#6-6_0_3800x600000_0.rt: 9600000 bytes read
disk: finished reading all files
plaintext of 6109ad31a7a7098bb1af0515ba854990 is dsmt0
plaintext of b74d955a1d747417565e2a71ac382909 is opmenr
plaintext of 5716db974c19110587f9a1f103fad76b is tpoinw
plaintext of d4346fefade2c208cdccb77b71ddd452 is nizedh
plaintext of 6e83b62e2a32ae2989235fd0718c1f96 is sso55r
plaintext of 29130f4bd6d34744aa7a9a57c43fd957 is di5gtv

statistics
-----
plaintext found: 6 of 15
total time: 65.96 s
time of chain traverse: 14.94 s
time of alarm check: 50.57 s
time of disk read: 0.01 s
hash & reduce calculation of chain traverse: 216486000
hash & reduce calculation of alarm check: 1390055034
number of alarm: 1097307
performance of chain traverse: 14.49 million/s
performance of alarm check: 27.49 million/s

result
-----
64028df466a6c177ff663242d484a0daf <not found> hex:<not found>
70fbcb3b121d33def05bf7ffcaa1991f1 <not found> hex:<not found>
29130f4bd6d34744aa7a9a57c43fd957 di5gtv hex:646935677674
6109ad31a7a7098bb1af0515ba854990 dsmt0h hex:64736d746f68
c246270f3ed86fcf75e15d08ea8d1557 <not found> hex:<not found>
4dfb16ba28661c8b23a257f42b3cf2e4 <not found> hex:<not found>
461c1ee2ffa9dec3da002762259d3e0 <not found> hex:<not found>
b8f9c9311827bf24de109c4ed5447b0b <not found> hex:<not found>
d4346fefade2c208cdccb77b71ddd452 nizedh hex:6e697a656468
b6e97576fed5ba1302097ca6b3ce603c <not found> hex:<not found>
b74d955a1d747417565e2a71ac382909 opmenr hex:6f706d656e72
071d5cf9bd929bba7897e974e3d2735a <not found> hex:<not found>
6e83b62e2a32ae2989235fd0718c1f96 sso55r hex:73736f353572
5716db974c19110587f9a1f103fad76b tpoinw hex:74706f696e77
415e6e0437915b7e3cc47ffcc518ca90f <not found> hex:<not found>
```

The time it takes to crack a salted password as seen above will also increase, as the cracker have to say process 5-X number of characters instead to account for the salts. Seen above with minimum 6, and max 6 hashes, rainbow crack only managed to crack 6/15 hashes in 10 times the time, and a lot of extra space complexity. Thereafter even knowing the password say “dsmtoh”, the cracker will have to guess which is the salt and the original password, and when more salts and random the password string is, it might be too difficult.

The screenshot shows a web browser window for "MD5 Decrypter - Over 829.726" on the "hashkiller.co.uk/md5-decrypter.aspx" page. The main header is "GPUHASH.me" with "online WPA/HASH cracker". The navigation menu includes Home, Forums, Decrypter / Cracker (which is selected), Database Info, Hash Min Max, WPA Crack, Lists and Competition, Contest, Tools, and Hashcat GUI. A donation message at the top right says "HashKiller relies on donations so please donate! BTC: 15qF9WUeFUD63ishxyAMIEgGqTcYzk49b". Below the menu, there's a message about mining Etherum and an ETH Wallet address: "ETH Wallet: 0x6D5aE69f127ad3cc7bcbe140a0AF60c75E5D54a". A note below the menu states: "HashKiller.co.uk allows you to input an MD5 hash and search for its decrypted state in our database, basically, it's a MD5 cracker / decryption tool." A section titled "How many decryptions are in your database?" claims over 829.726 billion unique decrypted MD5 hashes since August 2007. The main content area has a status bar: "Status: We found 15 hashes! [Timer: 2184 m/s] Please find them below...". On the left, a sidebar says "MD5 Hashes: 64028df466ac177ff663242d484a0daf 70fbc3b121d33def05bf7ffcaa1991f1 29130f4bd6d34744aa7a9a57c43fd957 6109ad31a7a7098bb1af0515ba854990 c2462707f3ed86cf75e15d08ea8d1555 4dfb16ba28661c8b23a257f42b3cf2e4 b8f9c9311827bf24de109c4ed5447b0b d4346efade2c208cdccb77b71ddd452 b6e97576fed5ba1302097ca6b3ce603c b74d955a1d747417565e2a71ac382909 071d5cf9bd929bba7897e974e3d2735a 6e83b62e2a32ae2989235fd0718c1f96 5716db974c19110587f9a1f103fad76b 415e6e8437915b7e3cc47ffc518ca90f Max: 64 Please use a standard list format" followed by a long list of decrypted MD5 hashes. The right side of the content area displays a large list of decrypted MD5 hashes, each preceded by a colon and a password.

```

Status: We found 15 hashes! [Timer: 2184 m/s] Please find them below...

MD5 Hashes:
64028df466ac177ff663242d484a0daf
70fbc3b121d33def05bf7ffcaa1991f1
29130f4bd6d34744aa7a9a57c43fd957
6109ad31a7a7098bb1af0515ba854990
c2462707f3ed86cf75e15d08ea8d1555
4dfb16ba28661c8b23a257f42b3cf2e4
b8f9c9311827bf24de109c4ed5447b0b
d4346efade2c208cdccb77b71ddd452
b6e97576fed5ba1302097ca6b3ce603c
b74d955a1d747417565e2a71ac382909
071d5cf9bd929bba7897e974e3d2735a
6e83b62e2a32ae2989235fd0718c1f96
5716db974c19110587f9a1f103fad76b
415e6e8437915b7e3cc47ffc518ca90f

Max: 64
Please use a standard list format

64028df466ac177ff663242d484a0daf MD5
: aseasq
70fbc3b121d33def05bf7ffcaa1991f1 MD5
: canceb
29130f4bd6d34744aa7a9a57c43fd957 MD5
: di5gvt
6109ad31a7a7098bb1af0515ba854990 MD5
: dsmtoh
c2462707f3ed86cf75e15d08ea8d1555 MD5
: egunbl
4dfb16ba28661c8b23a257f42b3cf2e4 MD5
: hed4ee
461clee2ffa9dec3da002762259d3e0 MD5
: lou0gv
b8f9c9311827bf24de109c4ed5447b0b MD5
: mlhdia
d4346efade2c208cdccb77b71ddd452 MD5
: nizedh
b6e97576fed5ba1302097ca6b3ce603c MD5
: ofrorg
b74d955a1d747417565e2a71ac382909 MD5
: opmenr
071d5cf9bd929bba7897e974e3d2735a MD5
: ows09m
6e83b62e2a32ae2989235fd0718c1f96 MD5
: sso55r
5716db974c19110587f9a1f103fad76b MD5
: tppoinw
415e6e8437915b7e3cc47ffc518ca90f MD5
: tthelz

```

HASHKILLER.CO.UK

Incomes hashkiller, this website contains a database of 829.726 billion decrypted MD5 hashes, with a tool like this, due to its space complexity, can easily break any hashes. However the hacker still do not know which part of it is the salt, unless it gets the file containing the salt tagged to the original password.

6. Hash breaking competition

Again, with modern tools like hashkiller.co.uk, MD5 is no longer secure, as it has such a big database of 829.726 billion decrypted MD5 hashes.

The screenshot shows a web browser window for 'MD5 Decrypter - Over 829.726' at <https://hashkiller.co.uk/md5-decrypter.aspx>. The page title is 'GPUHASH.me online WPA/HASH cracker'. It features a banner 'Industry best 30% WPA success rate'. Navigation tabs include Home, Forums, Decrypter / Cracker (selected), Database Info, Hash Min Max, WPA Crack, Lists and Competition (selected), Contest, Tools, Hashcat GUI, and Downloads. A donation message says 'HashKiller relies on donations so please donate! BTC: 15qF9WUeFUD63ishxyAMiEgGqTcYzk49b'. An ETH Wallet address is also listed. The main content area displays a list of 24 found MD5 hashes with their corresponding salts:

Status:	We found 24 hashes! [Timer: 2724 m/s] Please find them below...
MD5 Hashes:	3f7199846f908ed9897eeada2399c95 3f7d43553778f28cc4f2ba00598d0653 5b2b21ad30dc855e46a484abb180d325 8632c375e9eba096df51844a5a43ae93 41fb027d1c23536f9e0b2dde019e1a37 4060e28193d36aeb17dff58ecd2f4e1d ab56b4d92b40713acc5af89985d4b786 72b302bf297a228a75730123efef7c41 91546a5fcfa4fedaa2b837e4ad3e38fcbb3 fe01d67a002dfa0f3ac084298142eccd 0c5616c3772c470c9ea847e3ce4079dc 5213097ccbffbfbd3dd262860a3c15afc d606757a9c50dedc85e3cc90949b10ae 2ab96390c7dbe3439de74d0c9b0b1767 27c07e1c9bcd1ac7e5e995ed92534d77 c5cb0e6b15ed4ea1103257ac539a015a 82210e61e8f415525262575b20fae48d c822c1b63853ed273b98687ac505f9fa 4e7e00ce3e827e48d68f3309dabb11db 7c6a180b36896a0a8c02787eefaf0e4c 417432b93db6d7654c9612c2cc37d7edd 84df077bc1bd39ab1a3234de0cf655b 463ce859c76afb3b13b40298c2cd4683 981d304c3f23f463adfecf42028f7f0d
	3f7199846f908ed9897eeada2399c95 MD5 : RonRivest 3f7d43553778f28cc4f2ba00598d0653 MD5 : nocryptoforme 5b2b21ad30dc855e46a484abb180d325 MD5 : 59873523 8632c375e9eba096df51844a5a43ae93 MD5 : security1 41fb027d1c23536f9e0b2dde019e1a37 MD5 : 2011992 4060e28193d36aeb17dff58ecd2f4e1d MD5 : @@@@@@@@ ab56b4d92b40713acc5af89985d4b786 MD5 : abcde 72b302bf297a228a75730123efef7c41 MD5 : banana 91546a5fcfa4fedaa2b837e4ad3e38fcbb3 MD5 : randombits fe01d67a002dfa0f3ac084298142eccd MD5 : orange 0c5616c3772c470c9ea847e3ce4079dc MD5 : vladkool 5213097ccbffbfbd3dd262860a3c15afc MD5 : alamat d606757a9c50dedc85e3cc90949b10ae MD5 : makan 2ab96390c7dbe3439de74d0c9b0b1767 MD5 : hunter2 27c07e1c9bcd1ac7e5e995ed92534d77 MD5 : skittles c3cb0e6b73ed4ea1103257ac539a015a MD5 : bubbles1980

It can easily crack any passwords. Though some harder passwords are difficult to crack even with such a large database. We could use a bigger rainbow table, however my computer ran out of space to do so.