Ben Heinze
CSCI 476 Lab 9: Hashing
Professor Reese Pearsall

Task 1: Brute force attack against a hashed password:
In this task, you had to write a program to hash each word in *passwords.txt* and check if it matched the given hash of **437233c74e25fe505293cd2e8ecc2696**. Here is the program I wrote to accomplish this:

```
import hashlib
#437233c74e25fe505293cd2e8ecc2696 is the hash to break
file = open("passwords.txt", "r")

for line in file:
    line = line.replace("\n", "")
    #print(line)
    result = hashlib.md5(line.encode())

    if "437233c74e25fe505293cd2e8ecc2696" in result.hexdigest():
        print("the hexidec equiv of a hash is:, ", end="")
        print(result.hexdigest())
        print(line)
~
~
```

And this is the result:

```
[04/30/23]seed@VM:~/.../09_hashing$ python3 hashbreak.py
the hexidec equiv of a hash is:, 437233c74e25fe505293cd2e8ecc2696
pyramid
[04/30/23]seed@VM:~/.../09_hashing$
```

Task 2.1: Generating two different files with the same md5 hash

First we need to generate some prefix, “I am a prefix” saved in prefix.txt.

```
[04/30/23]seed@VM:~/.../09_hashing$ md5collgen -p prefix.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: d5fd278a37934aa1cfd4c41fd6f8bde2

Generating first block: ............
Generating second block: S00.
Running time: 5.80782 s
[04/30/23]seed@VM:~/.../09_hashing$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[04/30/23]seed@VM:~/.../09_hashing$ md5sum out1.bin
84118cdac4f01349c0c9625725ae55b6  out1.bin
[04/30/23]seed@VM:~/.../09_hashing$ md5sum out2.bin
84118cdac4f01349c0c9625725ae55b6  out2.bin
[04/30/23]seed@VM:~/.../09_hashing$
```

Although the two files have the same md5 hash of **84118cdac4f01349c0c9625725ae55b6**, the diff command reveals that the two files are actually different.

The hex dump of both **out1.bin** and **out2.bin** are indeed different, the character “Z” exists in the last line of **out1.bin**, but does not exist in **out2.bin**.

```
[04/30/23]seed@VM:~/.../09_hashing$ xxd out1.bin
00000000: 4920 616d 2061 2070 7265 6669 780a 0000  I am a prefix...
00000010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 5d43 8975 f573 b15a 6520 1616 a1b0 9198  ]C.u.s.Ze ......
00000050: bf49 4fc0 3706 4087 5432 40a6 7040 3df3  .IO.7.@.T2@.p@=.
00000060: 2716 03ed dec7 dd6a 8f64 77e2 d2af 7864  '......j.dw...xd
00000070: 0c92 96e7 bda8 35e4 3a84 d87f 910f bd14  ......5.:.......
00000080: 6c84 b803 7790 ec0a dfcd 64cb 2992 fa1c  l...w.....d.)...
00000090: cc14 1244 0af8 405e dd17 17c1 dc21 79ae  ...D..@^.....!y.
000000a0: 8f53 1ef6 6d18 73a4 6774 b10f 8f1e 7fa4  .S..m.s.gt......
000000b0: dc06 509c f392 3188 610c ed5a 3be6 0803  ..P...1.a..Z;...
[04/30/23]seed@VM:~/.../09_hashing$ xxd out2.bin
00000000: 4920 616d 2061 2070 7265 6669 780a 0000  I am a prefix...
00000010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 5d43 8975 f573 b15a 6520 1616 a1b0 9198  ]C.u.s.Ze ......
00000050: bf49 4f40 3706 4087 5432 40a6 7040 3df3  .IO@7.@.T2@.p@=.
00000060: 2716 03ed dec7 dd6a 8f64 77e2 d22f 7964  '......j.dw../yd
00000070: 0c92 96e7 bda8 35e4 3a84 d8ff 910f bd14  ......5.:.......
00000080: 6c84 b803 7790 ec0a dfcd 64cb 2992 fa1c  l...w.....d.)...
00000090: cc14 12c4 0af8 405e dd17 17c1 dc21 79ae  ......@^.....!y.
000000a0: 8f53 1ef6 6d18 73a4 6774 b10f 8f9e 7ea4  .S..m.s.gt....~.
000000b0: dc06 509c f392 3188 610c edda 3be6 0803  ..P...1.a...;...
[04/30/23]seed@VM:~/.../09_hashing$
```

Task 2.2:

Because the length of prefix.txt is not a multiple of 64, the md5 program added padding to make it so. This is because md5 uses block hashing, so it needs sufficient blocks in order to create a successful hash.

Task2.3:

```
string = ""
for i in range(64):
    string = string+"a"

out = open("64bits.txt","w")
out.write(string)
~
~
```

I wasn't about to guess 64 bits in a file so I just wrote a python program to do it for me.

```
[04/30/23]seed@VM:~/.../09_hashing$ vi 64bitgen.py
[04/30/23]seed@VM:~/.../09_hashing$ python3 64bitgen.py
[04/30/23]seed@VM:~/.../09_hashing$ ls -al
total 260
drwxrwxr-x  3 seed seed   4096 Apr 30 13:58 .
drwxrwxr-x 15 seed seed   4096 Apr 10 12:03 ..
-rw-rw-r--  1 seed seed    104 Apr 30 13:58 64bitgen.py
-rw-rw-r--  1 seed seed     64 Apr 30 13:58 64bits.txt
```

The 64 bit prefix we created with all a's takes up the first 4 rows of the prefix, so no extra padding was needed when compared to the the extra .'s that were used in the first attempt.

```
-rw-rw-r--  1 seed seed    192 Apr 30 14:00 out1.bin
-rw-rw-r--  1 seed seed    192 Apr 30 14:00 out2.bin
-rw-rw-r--  1 seed seed  12261 Apr 10 12:03 passwords.txt
-rw-rw-r--  1 seed seed 184974 Apr 10 12:03 pic_original.bmp
-rw-rw-r--  1 seed seed     14 Apr 30 13:43 prefix.txt
-rw-rw-r--  1 seed seed   1386 Apr 10 12:03 print_array.c
-rw-rw-r--  1 seed seed     51 Apr 10 12:03 README.md
-rw-rw-r--  1 seed seed    749 Apr 10 12:03 sha256_length_extension.c
-rw-rw-r--  1 seed seed    537 Apr 10 12:03 sha256_padding.c
[04/30/23]seed@VM:~/.../09_hashing$ xxd out1.bin
00000000: 6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
00000010: 6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
00000020: 6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
00000030: 6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
00000040: 1b71 9533 16a0 3ca2 088b 2758 0e71 641a  .q.3..<...'X.qd.
00000050: 6bdc 8e39 884b de18 15da 83f6 5512 3519  k..9.K......U.5.
00000060: 96e5 eaed 5a44 4e2d cb25 e359 b123 e555  ....ZDN-.%.Y.#.U
00000070: 9ec3 133c d9c7 9489 3bf2 826f 20fe 75d5  ...<....;..o .u.
00000080: ac7e 3fd4 4610 7fff cd9b 51ec 12d0 64ef  .~?.F.....Q...d.
00000090: ce59 7ac5 3f5b 4a0e 00b9 ef35 51e9 f48f  .Yz.?[J....5Q...
000000a0: d98e ba62 06b1 2069 627b f959 9ecc 5dde  ...b.. ib{.Y..].
000000b0: 0d3d aa25 adc1 03d4 b997 255d c551 16c9  .=.%......%].Q..
[04/30/23]seed@VM:~/.../09_hashing$ xxd out2.bin
00000000: 6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
00000010: 6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
00000020: 6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
00000030: 6161 6161 6161 6161 6161 6161 6161 6161  aaaaaaaaaaaaaaaa
00000040: 1b71 9533 16a0 3ca2 088b 2758 0e71 641a  .q.3..<...'X.qd.
00000050: 6bdc 8eb9 884b de18 15da 83f6 5512 3519  k....K......U.5.
00000060: 96e5 eaed 5a44 4e2d cb25 e359 b1a3 e555  ....ZDN-.%.Y...U
00000070: 9ec3 133c d9c7 9489 3bf2 82ef 20fe 75d5  ...<....;... .u.
00000080: ac7e 3fd4 4610 7fff cd9b 51ec 12d0 64ef  .~?.F.....Q...d.
00000090: ce59 7a45 3f5b 4a0e 00b9 ef35 51e9 f48f  .YzE?[J....5Q...
000000a0: d98e ba62 06b1 2069 627b f959 9e4c 5dde  ...b.. ib{.Y.L].
000000b0: 0d3d aa25 adc1 03d4 b997 25dd c551 16c9  .=.%......%..Q..
[04/30/23]seed@VM:~/.../09_hashing$
```

Task 2.4:
Yes, although they aren't completely different, there are indeed differences between the two generated output files.

Task 3: Understanding md5's "suffix extension" property
Our goal for this task is to understand that adding a suffix to the end of two different inputs with the same hashes will continue to share the same hash values. I'm going to use our previously generated **64bits.txt** file with 64 bits of "a"'s to append to our files.

```
[04/30/23]seed@VM:~/.../09_hashing$ cat out1.bin 64bits.txt > out1cat.bin
[04/30/23]seed@VM:~/.../09_hashing$ cat out2.bin 64bits.txt > out2cat.bin
[04/30/23]seed@VM:~/.../09_hashing$ md5sum out1cat.bin
88da0da63f82ce57e57c66f76104b70c  out1cat.bin
[04/30/23]seed@VM:~/.../09_hashing$ md5sum out2cat.bin
88da0da63f82ce57e57c66f76104b70c  out2cat.bin
[04/30/23]seed@VM:~/.../09_hashing$
```

As expected, despite concatenating **64bits.txt** two both **.bin** files, they shill share the same hash.

Task 4:

```
[04/30/23]seed@VM:~/.../09_hashing$ cat print_array.c
#include <stdio.h>

unsigned char xyz[200] = {
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
  0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41
};

int main()
{
  int i;
  for (i=0; i<200; i++){
    printf("%x", xyz[i]);
  }
  printf("\n");
}

[04/30/23]seed@VM:~/.../09_hashing$
```

In this task, we used the **print_array.c** to created a bunch of capital A's in hex. Since it's a program and we can't scramble our code, it scrambles a prefix and suffix. We can copy the prefix and suffix of this newly generated pa program, use the md5collgen tool, and add our suffix to both programs, and we still get the same hash values despite the programs being different.

```
[04/30/23]seed@VM:~/.../09_hashing$ gcc print_array.c -o pa
[04/30/23]seed@VM:~/.../09_hashing$ bless pa
Gtk-Message: 14:29:17.955: Failed to load module "canberra-gtk-module"
Could not find file "/home/seed/.config/bless/preferences.xml"
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find file "/home/seed/.config/bless/export_patterns"
Could not find file "/home/seed/.config/bless/history.xml"
[04/30/23]seed@VM:~/.../09_hashing$ head -c 12320 pa > prefix
[04/30/23]seed@VM:~/.../09_hashing$ tail -c +12448 pa > suffix
[04/30/23]seed@VM:~/.../09_hashing$ md5collgen -p prefix -o prefix_and_P prefix_and_Q
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'prefix_and_P' and 'prefix_and_Q'
Using prefixfile: 'prefix'
Using initial value: fa3f7a62525b9c90471862a4a04139a5

Generating first block: ..
Generating second block: S11.....
Running time: 3.51234 s
[04/30/23]seed@VM:~/.../09_hashing$ cat prefix_and_P suffix > program1.out
[04/30/23]seed@VM:~/.../09_hashing$ cat prefix_and_Q suffix > program2.out
[04/30/23]seed@VM:~/.../09_hashing$ diff program1.out program2.out
Binary files program1.out and program2.out differ
[04/30/23]seed@VM:~/.../09_hashing$ md5sum program1.out
c9175472fd9bc4b7753698f0a6a6b3d0  program1.out
[04/30/23]seed@VM:~/.../09_hashing$ md5sum program2.out
c9175472fd9bc4b7753698f0a6a6b3d0  program2.out
[04/30/23]seed@VM:~/.../09_hashing$
```