Justin Guerrero

MD5 collisions Lab

Task 1: Generating two different files with the same MD5 hash

Here we want to generate two different files with the same MD5 hash values. This will be the first part we need in order to produce a collision. We use md5collgen, a linux program which allows us to provide prefixes to our files.

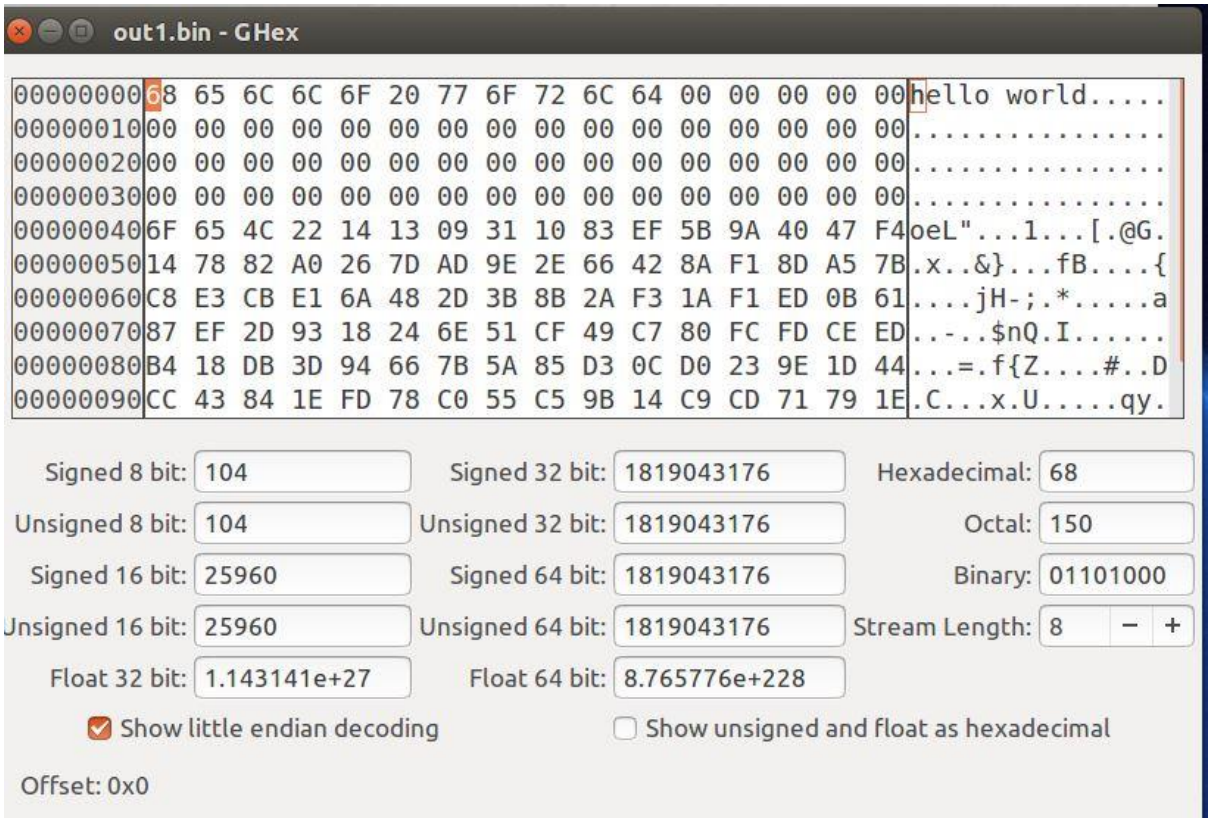
```
04/17/20]seed@VM:~/Documents$ mkdir lab10
04/17/20]seed@VM:~/Documents$ cd lab10/
04/17/20]seed@VM:~/.../lab10$ ls
04/17/20]seed@VM:~/.../lab10$ echo -n "hello world" > prefix.txt
04/17/20]seed@VM:~/.../lab10$ md5collgen -p prefix.txt -o out1.bin out2.bin
D5 collision generator v1.5
y Marc Stevens (http://www.win.tue.nl/hashclash/)

sing output filenames: 'out1.bin' and 'out2.bin'
sing prefixfile: 'prefix.txt'
sing initial value: 7a1a5e4371dccff7da465331445bcce5

enerating first block: .........................
enerating second block: S00.....
unning time: 25.0974 s
04/17/20]seed@VM:~/.../lab10$
```

Next we want to check the differences between them, using the diff command. Which doesn't give us much information because they're in Binary.

```
[04/17/20]seed@VM:~/.../lab10$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
```

Task 1A

What we want to do here is take a look at the differences between them in a hex editor to see what's going on. We notice that they entire file gets padded with Zeros.



Which brings us to task 1B

Task 1B

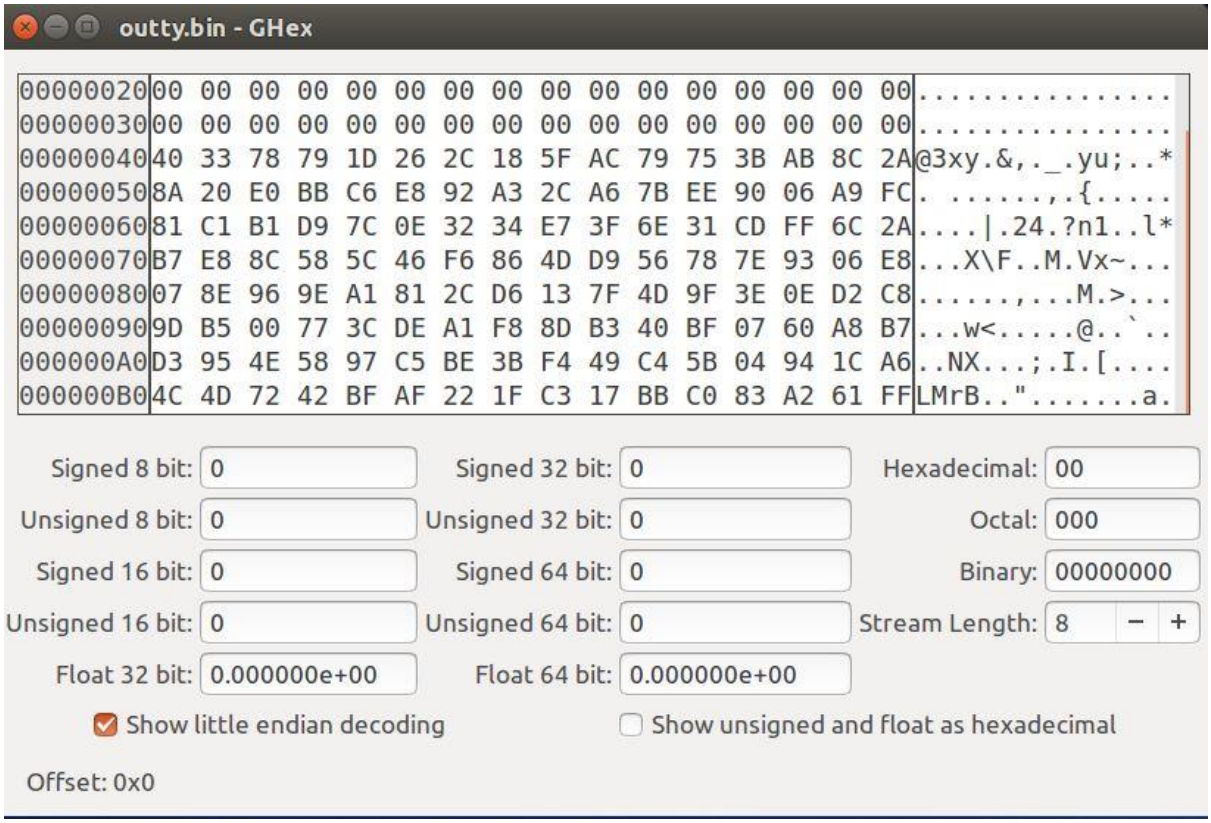
The way that MD5 processing files is in blocks of size 64. Hence if a file is not 64 bits it will be padded in order to reach the desired size.

Task 1C

We want to create a new file that is exactly 64 bit so we use the truncate feature and create a standard 64 bit file.

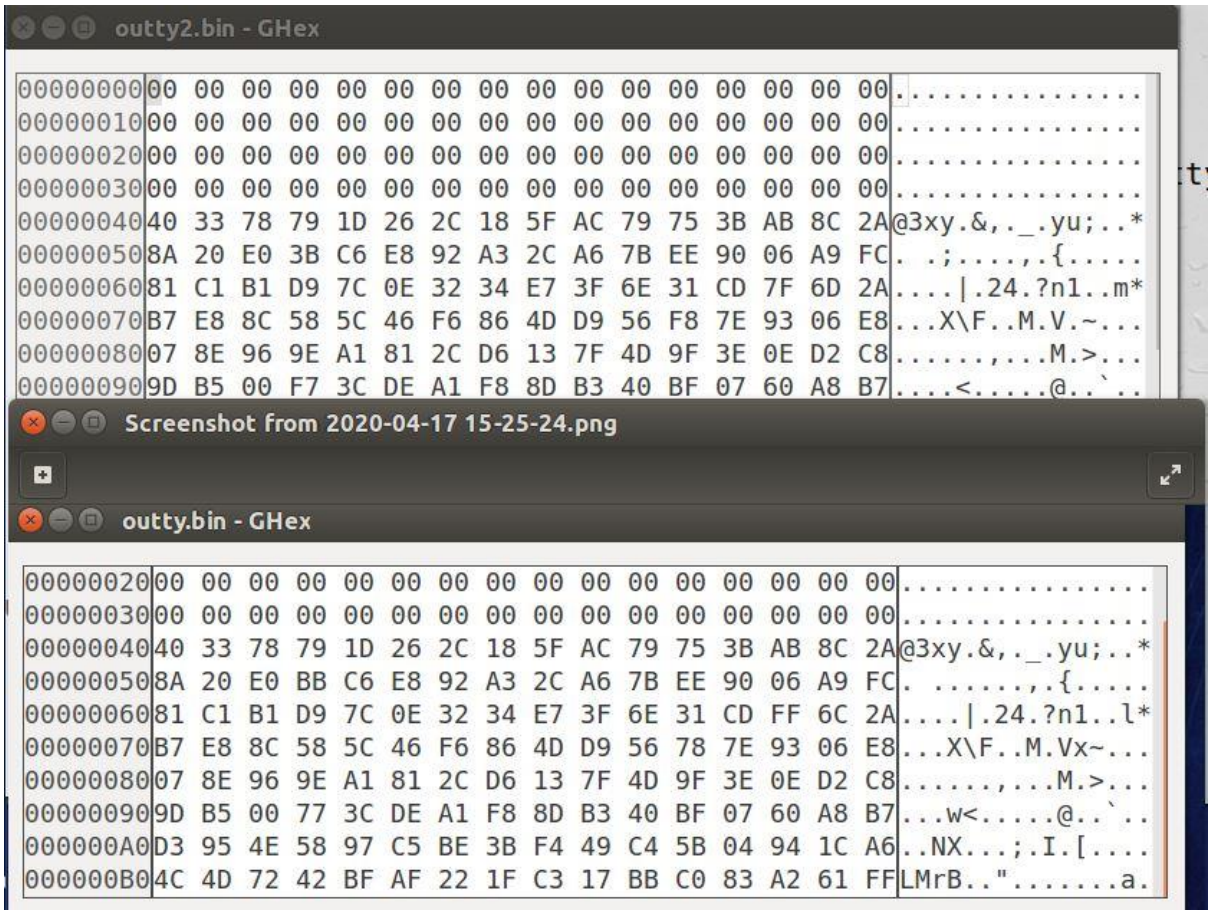
```
[04/17/20]seed@VM:~/.../lab10$ truncate -s 64 sixtyfo.txt
[04/17/20]seed@VM:~/.../lab10$ ls
out1.bin  out2.bin  prefix.txt  sixtyfo.txt
[04/17/20]seed@VM:~/.../lab10$ ls -l
total 12
-rw-rw-r-- 1 seed seed 192 Apr 17 13:18 out1.bin
-rw-rw-r-- 1 seed seed 192 Apr 17 13:18 out2.bin
-rw-rw-r-- 1 seed seed  11 Apr 17 13:17 prefix.txt
-rw-rw-r-- 1 seed seed  64 Apr 17 13:47 sixtyfo.txt
[04/17/20]seed@VM:~/.../lab10$ md5collgen -p sixtyfo.txt -o outty.bin outty2.bin

MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'outty.bin' and 'outty2.bin'
Using prefixfile: 'sixtyfo.txt'
Using initial value: ac1d1f03d08ea56eb767ab1f91773174

Generating first block: ...
Generating second block: S10.....
Running time: 2.64127 s
[04/17/20]seed@VM:~/.../lab10$ ghex outty
```

We observe when we open our hex editor that no 00 padding is required, the difference between our “hello world” file with padding.



Task 1D

Here we want to know if the two files we created from task 1C are any different. Lets take a look.



Upon inspecting we see that these two do not have any difference between the two files. We suspect this may cause a collision error, but we shall see.

Task 2: Understanding MD5's "Suffix Extension" Property

Here we are tasked with trying to show that even if you create and hash files you can change the contents of what's in the files by concatenating something to it and still get the same hashes for the two binary output files. To better show this explanation we first create two files

```
[04/17/20]seed@VM:~/.../lab10$ md5collgen -p file1.txt -o fileout1.bin fileout2.
bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'fileout1.bin' and 'fileout2.bin'
Using prefixfile: 'file1.txt'
Using initial value: 013f372f7d1b6e628953e4115b2b2f74

Generating first block: .....
Generating second block: S11.....
...
Running time: 7.61282 s
```

Then we run the MD5sum to make sure the files are equal to each other, which we see they are.

```
[04/17/20]seed@VM:~/.../lab10$ md5sum fileout1.bin fileout2.bin
f9360464b20fa94ca366178106423c45  fileout1.bin
f9360464b20fa94ca366178106423c45  fileout2.bin
```

Next we concatenate the word “hi” into each file to change them and see what happens.

```
[04/17/20]seed@VM:~/.../lab10$ echo -n "hi" >> fileout1.bin
[04/17/20]seed@VM:~/.../lab10$ echo -n "hi" >> fileout2.bin
[04/17/20]seed@VM:~/.../lab10$ md5sum fileout1.bin fileout2.bin
adbaa126fe92b4339daebc0f5230fac3  fileout1.bin
adbaa126fe92b4339daebc0f5230fac3  fileout2.bin
```

We can see that the hashes have changed, but they indeed remain the same. Proving what we set out to discover.

Task 3: Generating two executables with the same MD5 hash

This task we are setting out to create two different programs that the contents are slightly different, but the hash values are the same. We copy the code provided by Travis and Seed labs, program.c, and begin the tasks.

First we compile the code as prog, and then grab the head of the program.

```
[04/18/20]seed@VM:~/.../lab10$ vi program.c
[04/18/20]seed@VM:~/.../lab10$ gcc -o prog program.c
[04/18/20]seed@VM:~/.../lab10$ head -c 4224 pr
prefix.txt  prog        program.c
[04/18/20]seed@VM:~/.../lab10$ head -c 4224 prog > prefix
```

Next, we want to take the head of the file and save is as the prefix and run MD5collgen on the program and create prog1 and prog2 from it.

```
[04/18/20]seed@VM:~/.../lab10$ md5collgen -p prefix -o agenbgen
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Error: exactly two output filenames should be specified.
[04/18/20]seed@VM:~/.../lab10$ gcc -o prog program.c
[04/18/20]seed@VM:~/.../lab10$ head -c 4224 prog > prefix
[04/18/20]seed@VM:~/.../lab10$ md5collgen -p prefix -o pre1out pre2out
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'pre1out' and 'pre2out'
Using prefixfile: 'prefix'
Using initial value: 266177d958c5082472a68345a839abfa

Generating first block: ...............
Generating second block: S10........
```

Lets take a look at the hashes with Bless. We note that the files are the same MD5 hash, but with different suffix's.

```
&.....................
......................
..AAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA
AAA........_=.V.P...E
9.......?...,.#....O.
DO...!n.....Z...9...\
.pL...1.6....cE.U.Dg.
.CXp.! ....&`........
^.7..[@ .R....Z...6.|
.sJ..
```

```
00 |....................
41 |....................A
41 |AAAAAAAAAAAAAAAAAAAA
41 |AAAAAAAAAAAAAAAAAAAA
41 |AAAAAAAAAAAAAAAAAAAA
BE |AAAAAA........_=.V.
14 |P...E9.......?...,.
8A |#....O.DO...!......
98 |Z...9.....pL...1.6.
AB |...cE.U.Dg..C.p.! .
5B |...&`........^.7..[
A4 |@..R....Z...6.|TsJ.
   |.
```

ecimal: 7F 45 4C 46

Now we want to grab the common ending and concatenate them with the two different files.

```
[04/18/20]seed@VM:~/.../lab10$ tail -c 4353 prog > commonend
[04/18/20]seed@VM:~/.../lab10$ cat commonend >> pre1out
[04/18/20]seed@VM:~/.../lab10$ cat commonend >> pre2out
[04/18/20]seed@VM:~/.../lab10$
```

Next we run the programs, and check if they are indeed different will complete this task.

```
[04/18/20]seed@VM:~/.../lab10$ chmod +x pre1out
[04/18/20]seed@VM:~/.../lab10$ chmod +x pre2out
[04/18/20]seed@VM:~/.../lab10$ ls -l
```

```
[04/18/20]seed@VM:~/.../lab10$ ./pre1out
414141414141414141414141414141414141414141414141414141414141414141414141414141414141
414141414141414141414141414141414141414141414141414116abcd84a8a6d3c35f3dbc56be50adf3
fb45398b818d9d2e1eb63fead2b72c1423bbedf0d24fd4444fe0d8f2216ed3adaee78a5aa29f4399
480e25c4704cfad8f831a83698fcbabd6345b655c9446793a7435870a02120abb510e72660a8b6d7
d3fa3af835e1e373905b4020e52b9b85e85aa9e0aa36b57cd4734aa4100000454e44
[04/18/20]seed@VM:~/.../lab10$ ./pre2out
414141414141414141414141414141414141414141414141414141414141414141414141414141414141
414141414141414141414141414141414141414141414141414116abcd84a8a6d3c35f3dbc56be50adf3
fb4539b818d9d2e1eb63fead2b72c1423bbedf0d24fd4444fe0d8f221eed3adaee78a5aa29f43994
80e2dc4704cfad8f831a83698fcbabd6345b655c9446793a743d870a02120abb510e72660a8b6d7d
3fa3af835e1e373905b40a0d52b9b85e85aa9e0aa36b57c54734aa4100000454e44
```

```
[04/18/20]seed@VM:~/.../lab10$ ./pre1out > 1out
[04/18/20]seed@VM:~/.../lab10$ ./pre2out > 2out
[04/18/20]seed@VM:~/.../lab10$ diff -q 1out 2out
Files 1out and 2out differ
```

Task 4: Making the two programs behave differently

The idea of this portion of the lab is to create a file that has two separate programs with the same hash, one that will run a “safe program” and another that inserts malicious code.

We begin by modifying our program.c code from earlier.

~/Documents/lab10/program.c • - Sublime Text (UNREGISTERED)

non.py | plaintext.txt | xor.py | program.c | [04/05/20]seed@VM:~/.../lab09$ openssl enc -aes-12

```c
#include <stdio.h>
unsigned char a[300] = { 'H','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A'
unsigned char b[300] = { 'H','A','A','A','A','A','A','A','A','A','H','A','A','A','A','A','A','A','A'
int main()
{
        int i;
        int isSame=1;
        for(i = 0; i < 200; i++)
        {
                if(a[i]!=b[i])
                        isSame=0;
        }
        if(isSame)
                printf("No pirates here, smooth sailing matey");
        else
                printf("Arghhh give me the booty");
        printf("\n");
}
```

Then we want to compile and save the program

Next we check out the contents of the binary with bless, we see the first array starts at 0X1040 which, same as before, is located at 4160. So we will use similar commands as task 3 for cutting the head, and MD5 hash.

```
[04/18/20]seed@VM:~/.../lab10$ head -c 4224 newprog >> pre2
[04/18/20]seed@VM:~/.../lab10$ md4collgen -p newprog -o newout1 newout2
md4collgen: command not found
[04/18/20]seed@VM:~/.../lab10$ md5collgen -p newprog -o newout1 newout2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'newout1' and 'newout2'
Using prefixfile: 'newprog'
Using initial value: 6a9b0cb76855a37a6d2be1964f287d90

Generating first block: ....
```

Next, we want to create two new files that we can grab the middle section where “HAAA****” begins. We use the head function again. And we do the same for the offset for the middle and the end. The next thing we do is concatenate them all together and check MD5sum to make sure they are the same. After verifying the are the same we chmod the programs to make them executable and run them and get the following results.

```
[04/18/20]seed@VM:~/.../lab10$ head -c +4353 newprog > newend1
[04/18/20]seed@VM:~/.../lab10$ head -c +4353 newprog > newend2
[04/18/20]seed@VM:~/.../lab10$ tail -c +321 newend1 > commonend1
[04/18/20]seed@VM:~/.../lab10$ tail -c +321 newend2 > commonend2
Terminator seed@VM:~/.../lab10$ cp newout1 goodguy
[04/18/20]seed@VM:~/.../lab10$ cp newout2 badguy
[04/18/20]seed@VM:~/.../lab10$ cat newend1 >> goodguy
[04/18/20]seed@VM:~/.../lab10$ cat newend2 >> badguy
[04/18/20]seed@VM:~/.../lab10$ cat newend1 >> goodguy
[04/18/20]seed@VM:~/.../lab10$ cat newend2 >> badguy
[04/18/20]seed@VM:~/.../lab10$ cat commonend1 >> goodguy
[04/18/20]seed@VM:~/.../lab10$ cat commonend2 >> badguy
[04/18/20]seed@VM:~/.../lab10$ md5sum goodguy badguy
52837f447c618aa68427111d0af03750  goodguy
52837f447c618aa68427111d0af03750  badguy
[04/18/20]seed@VM:~/.../lab10$ chmod +x goodguy badguy
[04/18/20]seed@VM:~/.../lab10$ ./goodguy
No pirates here, smooth sailing matey
[04/18/20]seed@VM:~/.../lab10$ ./badguy
```