MD5 COLLISION ATTACK LAB

Jasmine Joy (500924677)

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

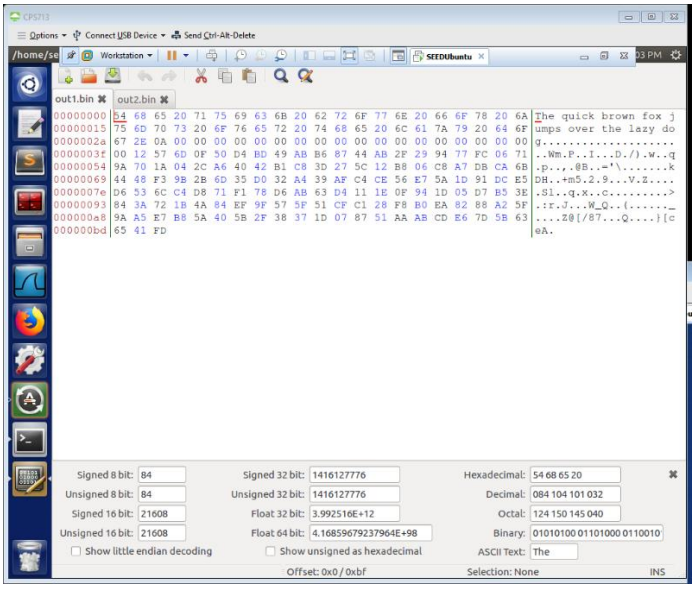
A file 'prefix' was created. Below is the implementation of `md5collgen`

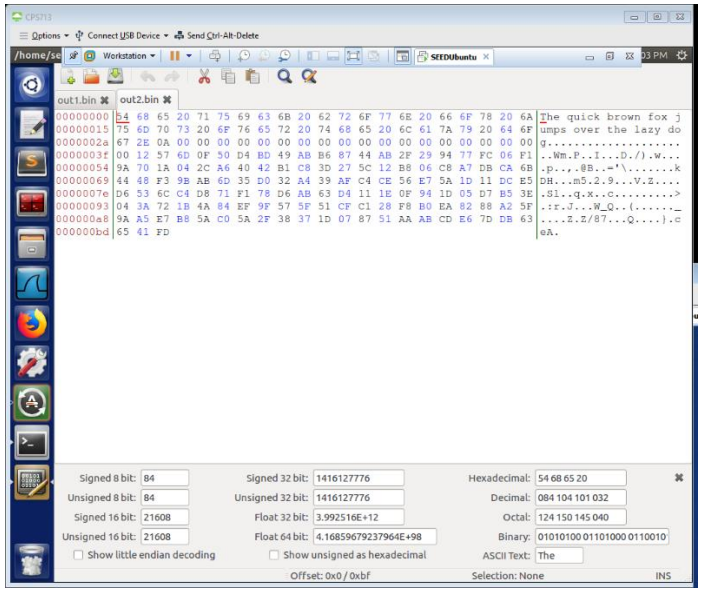
```
[03/07/20]seed@VM:~/.../Lab3$ cat prefix.txt
The quick brown fox jumps over the lazy dog.
[03/07/20]seed@VM:~/.../Lab3$ md5collgen -p prefix.txt
-o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: 9d40f4fb3ecb760166124e8c68388499

Generating first block: ................
Generating second block: S11...
Running time: 20.903 s
[03/07/20]seed@VM:~/.../Lab3$
```

Using `checksum` to determine if the output files out1.bin and out2.bin are the same.

```
[03/07/20]seed@VM:~/.../Lab3$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[03/07/20]seed@VM:~/.../Lab3$ md5sum out1.bin
dcd7205e747bb41fd21261b5d8911d34  out1.bin
[03/07/20]seed@VM:~/.../Lab3$ md5sum out2.bin
dcd7205e747bb41fd21261b5d8911d34  out2.bin
[03/07/20]seed@VM:~/.../Lab3$
```

Looking at the output files using hex editor:



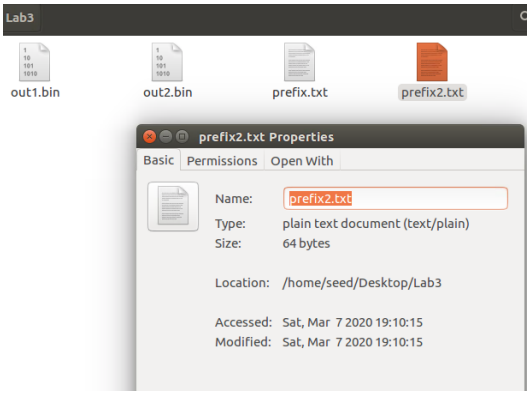


Answer 1

If the length is not a multiple of 64, then the text gets padded with 0s (as seen above in hex editor).

Answer 2

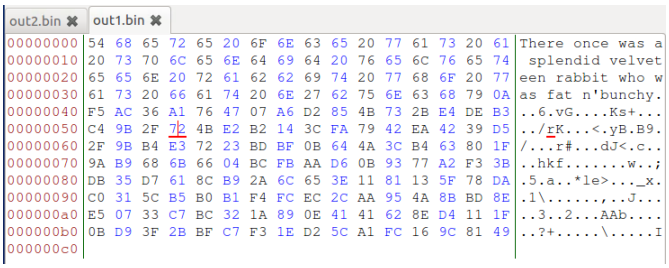
Creating a new textfile with arbitrary content (64 bytes).

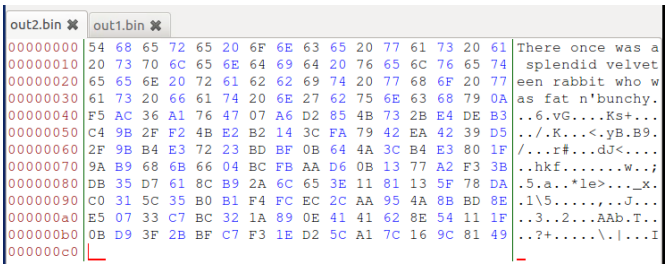


Running the collision tool again gives the following:

```
[03/07/20]seed@VM:~/.../Lab3$ cat prefix2.txt
There once was a splendid velveteen rabbit who was fat n'bunchy
[03/07/20]seed@VM:~/.../Lab3$ md5collgen -p prefix2.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: 'prefix2.txt'
Using initial value: 7b93c3bf7c78d233c90a0576f19640be

Generating first block: ........................................
Generating second block: W...
Running time: 68.6717 s
[03/07/20]seed@VM:~/.../Lab3$
```

Looking at the output files using hex editor:

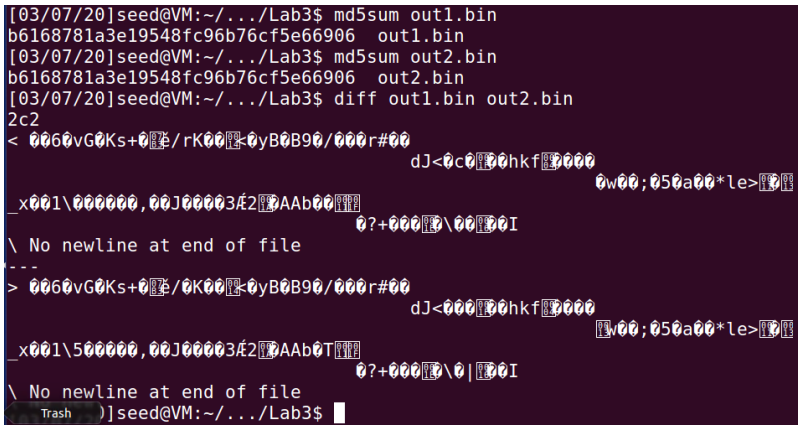




As seen above, when the file is exactly 64 bytes, then no padding with zeros is applied.

Answer 3

Not all bytes are different for the two output files (as seen in output1.bin and output2.bun above). The bytes differ at some positions (see highlighted region in red). The differences need not be always constant.



Task 2: Understanding MD5s Property

The property explored in this task is based on the Target Collision Resistance property of Hash functions.
That is; for M and N where their hashes are the same h(M) = h(N). If a segment X was added to M and N; then the hashes would be equal h(M+x) = h(N+x)
Using the file prefix above:

```
[03/09/20]seed@VM:~/.../Lab3$ cat prefix.txt
The quick brown fox jumps over the lazy dog.
[03/09/20]seed@VM:~/.../Lab3$ md5collgen -p prefix.txt -o out1 out2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1' and 'out2'
Using prefixfile: 'prefix.txt'
Using initial value: 9d40f4fb3ecb760166124e8c68388499

Generating first block: ..............
Generating second block: S00.....
Running time: 18.3947 s
[03/09/20]seed@VM:~/.../Lab3$
```

The checksums of the original outputs of the file are, as expected, the same.

```
[03/09/20]seed@VM:~/.../Lab3$ md5sum out1 out2
ff7751768307fb0ee946a59222d03d2e  out1
ff7751768307fb0ee946a59222d03d2e  out2
[03/09/20]seed@VM:~/.../Lab3$
```

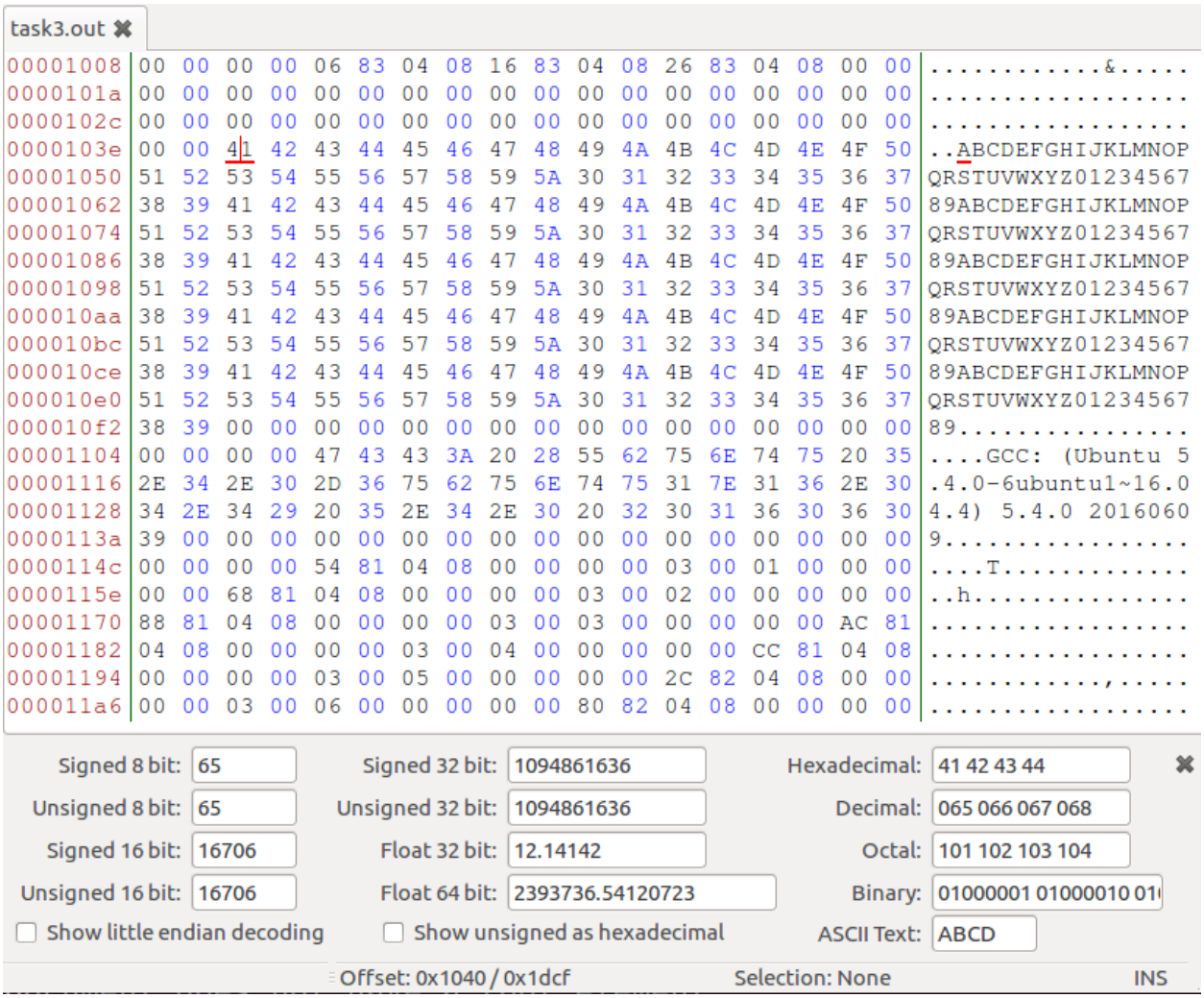
The word “random” was added into both outputs. And, as predicted, the hashes of the outputs stay the same, but a new hash was generated.

```
[03/09/20]seed@VM:~/.../Lab3$ echo random >> out1
[03/09/20]seed@VM:~/.../Lab3$ echo random >> out2
[03/09/20]seed@VM:~/.../Lab3$ cat out1
The quick brown fox jumps over the lazy dog.
x @$5@j@[illegible]S[illegible]l@h@@w@@'@91@@][illegible]q@K(kD9ÀS@Nh@@l[illegible]C@@xE[illegible]BB}}^@3B@@5@@m@@1[illegible]-@[illegible]{[illegible]p
}2[illegible]]@r @L[illegible]gG[illegible]}@@[illegible]{@[illegible]random
[03/09/20]seed@VM:~/.../Lab3$ cat out2
The quick brown fox jumps over the lazy dog.
x @$5@j@[illegible]S[illegible]l@h;@w@@'@91@@][illegible]q@K(kD9H@S@Nh@@l[illegible]C[illegible]xE[illegible]BB}}^@3B@@5@@m@@1@[illegible]-@[illegible]{[illegible]@
p}2[illegible]]@r @L@@[illegible]gG[illegible]}@@[illegible]@@[illegible]random
[03/09/20]seed@VM:~/.../Lab3$ md5sum out1 out2
2910363135d25aedeb25a592db22250b  out1
2910363135d25aedeb25a592db22250b  out2
[03/09/20]seed@VM:~/.../Lab3$
```

Task 3: Generating two executable files with the same MD5 Hash

```
 GNU nano 2.5.3                File: task3.c                          Modified
Search your computer
#include <stdio.h>

unsigned char a[200] = { 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','$
int main()
{
        int i;
       for(i = 0; i < 200; i++)
        {
                printf("%x", a[i]);
        }
        printf("\n");
}
```

```
[03/09/20]seed@VM:~/.../Lab3$ pico task3.c
[03/09/20]seed@VM:~/.../Lab3$ gcc task3.c -o task3.out
```

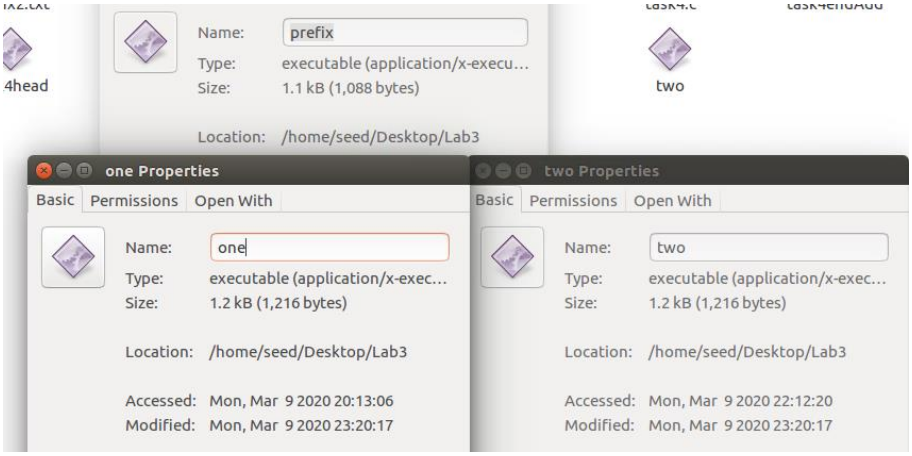


As seen above, the test starts at the 1040th position.

1040%64 = 16

The next integer that is fully divisible by 64 would be 1088.

Hence, the filesize needs to be 1088 bytes.



Dividing the binary file task3.out into two segments: prefix, and suffix.

```
[03/09/20]seed@VM:~/.../Lab3$ head -c 1088 task3.out > prefix
[03/09/20]seed@VM:~/.../Lab3$ md5collgen -p prefix -o one two
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'one' and 'two'
Using prefixfile: 'prefix'
Using initial value: f63809b1d4dc3365d4dc0f62dc1fc3ec

Generating first block: ..............................
Generating second block: S11.....................................
Running time: 39.7606 s
[03/09/20]seed@VM:~/.../Lab3$ tail -c 1217 task3.out > task3end
[03/09/20]seed@VM:~/.../Lab3$
```

Adding the tail-end segment to the output files of prefix

```
[03/09/20]seed@VM:~/.../Lab3$ cat task3end >> one
[03/09/20]seed@VM:~/.../Lab3$ cat task3end >> two
[03/09/20]seed@VM:~/.../Lab3$
```

Trying to make the outputs executable did not produce the desired result.

```
[03/09/20]seed@VM:~/.../Lab3$ chmod +x one
[03/09/20]seed@VM:~/.../Lab3$ chmod +x two
[03/09/20]seed@VM:~/.../Lab3$ ./one
Segmentation fault
[03/09/20]seed@VM:~/.../Lab3$ ./two
Segmentation fault
```

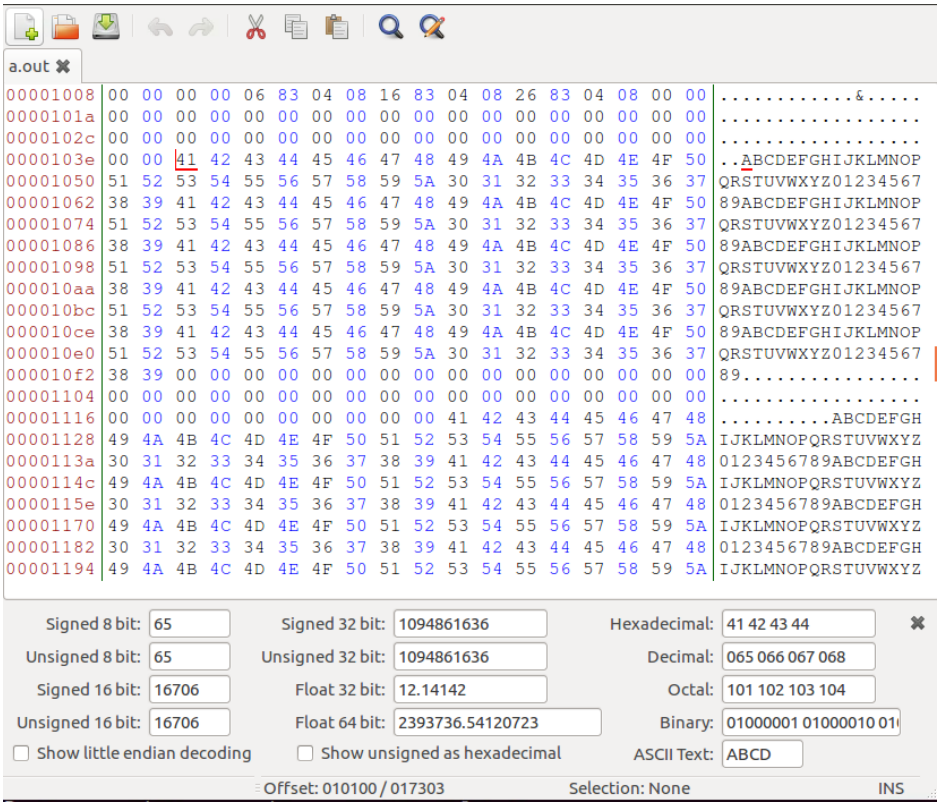
In theory, running the executable should provide. And trying to run them again should give me textfiles instead of binary. But since, I was unable to get the testfiles, did a checksum on the binaries (which provided the same hash), and a report on the difference between the binaries (they do have differences between them)

```
[03/09/20]seed@VM:~/.../Lab3$ md5sum one two
bf3703878db0ec0bdec7cfeb4e4161de  one
bf3703878db0ec0bdec7cfeb4e4161de  two
[03/09/20]seed@VM:~/.../Lab3$ diff one two
Binary files one and two differ
[03/09/20]seed@VM:~/.../Lab3$
```

Task 4: Making the Two Programs Behave Differently

Where arrays a and b are identical:

```c
#include <stdio.h>

unsigned char a[200] = { 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','$
unsigned char b[200] = { 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','$
int main()
{
        int i, variable;
        for(i = 0; i < 200; i++)
        {
                if(a[i]!=b[i])
                        {
                            variable=0;
                        }
                        else {
                            variable=1;
                        }
        }

        if(variable=1){
                printf("Safe code");
        }
        else{
                printf("Malicious code");
        }
        printf("\n");
}
```



10100%64 = 52

However, 10176%64 = 0. Therefore:

Following the same process as in Task 3:

```
[03/09/20]seed@VM:~/.../Lab3$ head -c 10176 a.out > task4head
[03/09/20]seed@VM:~/.../Lab3$ md5collgen -p prefix -o task4onef task4twof
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'task4onef' and 'task4twof'
Using prefixfile: 'prefix'
Using initial value: f63809b1d4dc3365d4dc0f62dc1fc3ec

Generating first block: ......
Generating second block: W..........
Running time: 6.93946 s
[03/09/20]seed@VM:~/.../Lab3$ md5sum tas4onef task4twof
md5sum: tas4onef: No such file or directory
d73090199bcf397401a3047f48e04210  task4twof
[03/09/20]seed@VM:~/.../Lab3$
```

```
[03/09/20]seed@VM:~/.../Lab3$ head -c 4353 a.out > endtoAdd
[03/09/20]seed@VM:~/.../Lab3$ tail -c +129 endtoAdd > task4endAdd
[03/09/20]seed@VM:~/.../Lab3$
```

The hashes are the same, but the files are different.

```
[03/09/20]seed@VM:~/.../Lab3$ md5sum bCode mCode
d73090199bcf397401a3047f48e04210  bCode
d73090199bcf397401a3047f48e04210  mCode
[03/09/20]seed@VM:~/.../Lab3$ cp task4onef bCode
[03/09/20]seed@VM:~/.../Lab3$ cp task4twof mCode
[03/09/20]seed@VM:~/.../Lab3$ diff bCode mCode
Binary files bCode and mCode differ
```

Same issue as Task 3 encountered:

```
[03/09/20]seed@VM:~/.../Lab3$ chmod +x bCode mCode
[03/09/20]seed@VM:~/.../Lab3$ ./bCode
Segmentation fault
[03/09/20]seed@VM:~/.../Lab3$ ./mCode
Segmentation fault
[03/09/20]seed@VM:~/.../Lab3$
```

I wasn’t able to figure out how to implement the code correctly to check if when benign code or malicious code is executed. But, in theory, chmod +x should’ve made the files executable.

To complete the task, I should find a way to make the two files distinct such that their hash is also different. This would make execute the ‘malicious’ portion of the code.