Lab9

-By Vikramadithya Ivaturi

**Task-1**

**Output:**

First we generate a normal test file that contains our prefix.

Text

Description automatically generated

Now, we execute the md5collgen program to generate 2 files that will have the same MD5 hash value.

Text

Description automatically generated with medium confidence

Now, we have established that both the generated binary files differ.

Let’s check their MD5 hash values.

Text

Description automatically generated

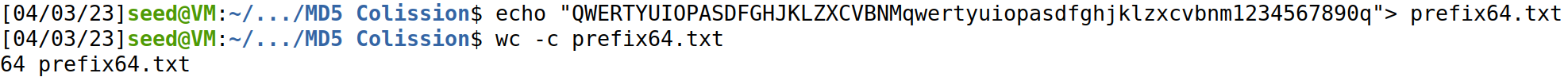
We can see in the above screenshot that both the MD5 values are the exact same values.

1. If the length of your prefix file is not multiple of 64, what is going to happen?
   1. The application is going to add padding to make it into 64 bytes.

Text

Description automatically generated

1. Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens.
   1. We create a prefix file with 64 random characters.



Now, we execute the md5collgen program to generate 2 files.

Table

Description automatically generated with medium confidence

Now we check if the files are different,

Text

Description automatically generated with low confidence

So, there is a difference.

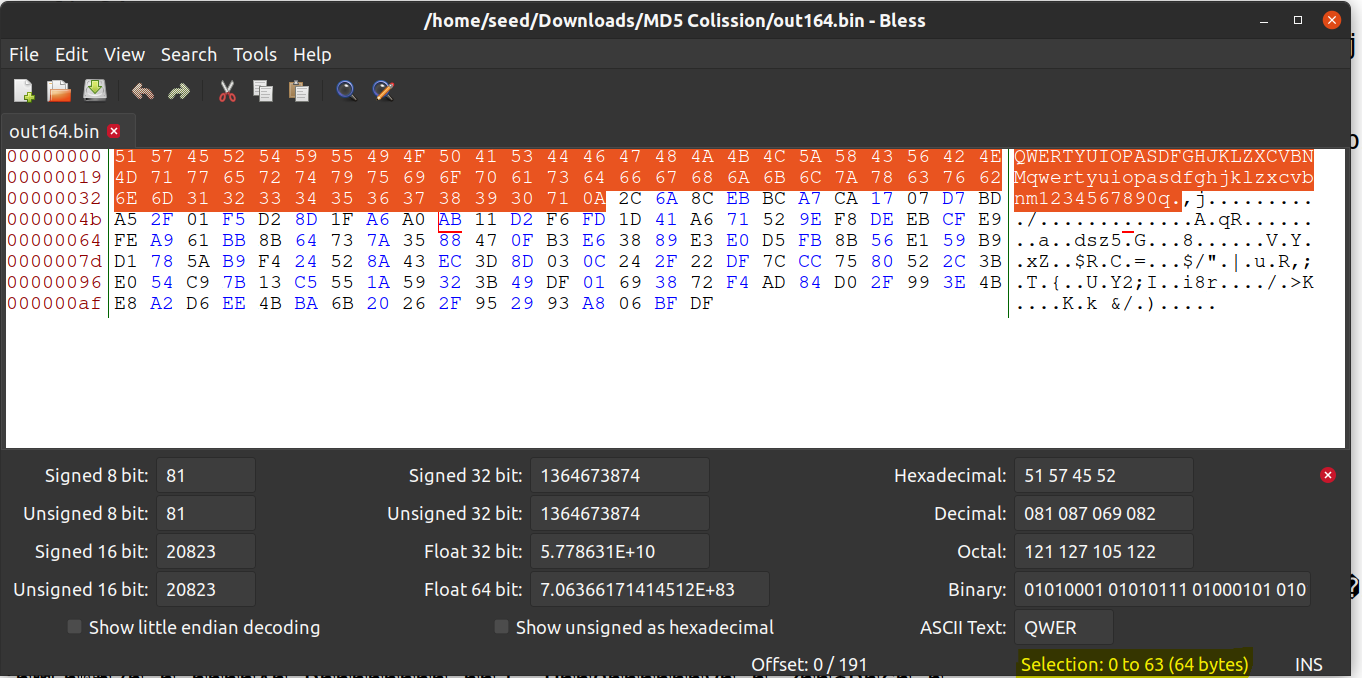
Now, we verify the MD5 sum of the 2 files.

Text

Description automatically generated with medium confidence

Both are exactly the same.

We open one of the output files in bless



So, when we create a 64 byte prefix, there will be no padding.

1. Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.
   1. We extract the hexadecimal format of the binary files, and then write them onto another file individually to check the difference between them.A screenshot of a computer

      Description automatically generated with low confidence

Running the diff command and verifying the differences.

A picture containing table

Description automatically generated

There are differences in 7 bytes between both the files.

**Task-2**

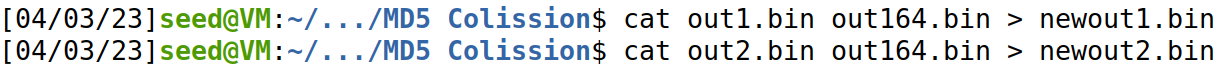
**Output:**

First we take 2 files whose MD5 sum is the same, and another file whose MD5 is different.

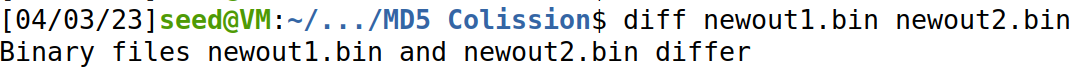
Text

Description automatically generated

Now, we concatenate ‘out1.bin’ and ‘out2.bin’ with ‘out164.bin’ individually.



Now, we verify if both the resultant files are different.



Now, we check the MD5 sums of the output files.

Text

Description automatically generated

So, the property that was stated in the question s proved.

**Task-3**

**Code:**

#include <stdio.h>

unsigned char xyz[200] = {

"LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL"

};

int main()

{

int i;

for (i=0; i<200; i++)

{

printf("%x", xyz[i]);

}

printf("\n");

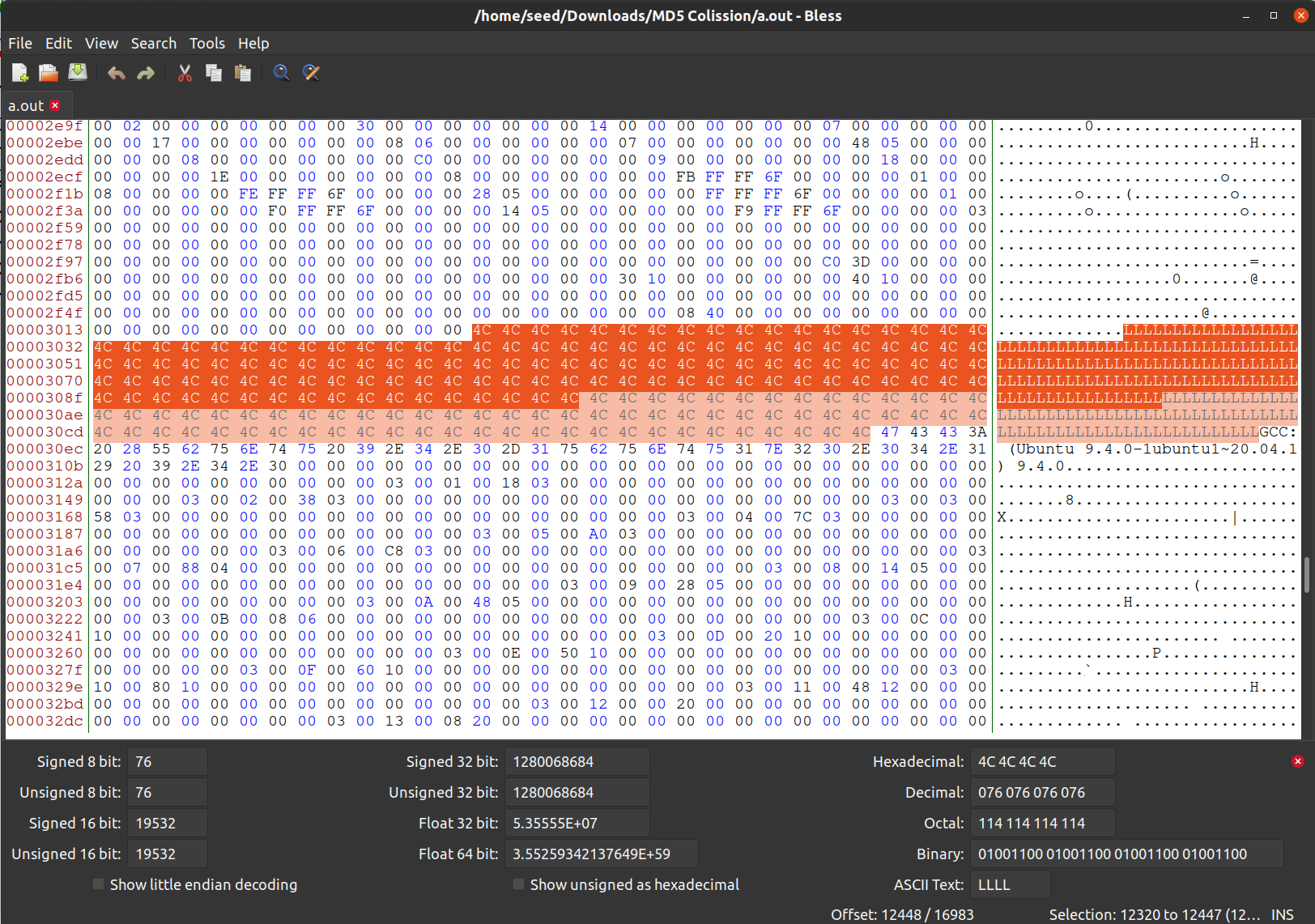
}

**Output:**

Form the above code, we have created a compiled an object code file which is named a.out (by default).Text

Description automatically generated

Now, we see this file in bless to analyze the points where the file has the character ‘L’ that we are going to consider into the 128 bit data block, everything before it is going to be considered as prefix and everything after it is going to be suffix.



We can see that the characters start from 12320 and we count till the end of 12448 which makes 128 bytes, and then assign all the data before 12320 into prefix file, and assign all data after 12448 into suffix file.



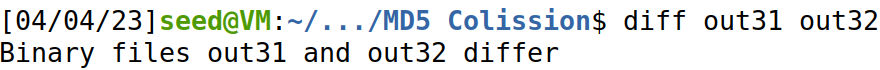


Now, generate collision files using md5collgen program.

Text

Description automatically generated with low confidence

Now, we verify if both the files differ.

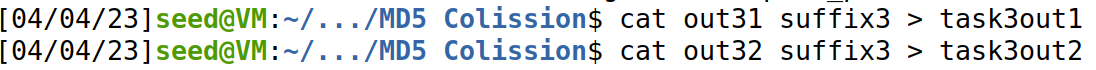


Now we check the MD5 sum of both the files,

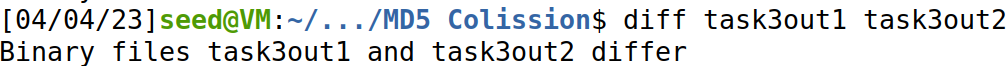
Text

Description automatically generated with medium confidence

Since both of the checksum values are the same, we proceed to concatenate the output files with the suffix file.



Now, check if both the files are different and have the same checksum value.



Text

Description automatically generated with medium confidence

A picture containing diagram

Description automatically generated

We see the files print the above outputs that are different.

**Task-4**

**Code:**

#include <stdio.h>

#define LENGTH 400

unsigned char X[LENGTH]= {

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

};

unsigned char Y[LENGTH]= {

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"

};

int main()

{

  int i = 0;

  for (i =0; i< LENGTH; i++){

    if (X[i] != Y[i]) break;

  }

  if (i==LENGTH){

     printf("%s\n", "Executing benign code... ");

  }

  else {

     printf("%s\n", "Executing malicious code... ");

  }

  return 0;

}

**Output:**

After compiling the above program into machine code, we split the ‘a.out’ file into prefix and suffix with prefix being a multiple of 64 occurring after the array content.

A screenshot of a computer

Description automatically generated with medium confidence

So, we split the file at 12352 till which everything is prefix, and we leave 128 bits for data part, and make the rest of the file into suffix.



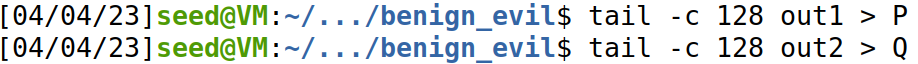


Now, we generate the collision files with md5collgen program based on the prefix.

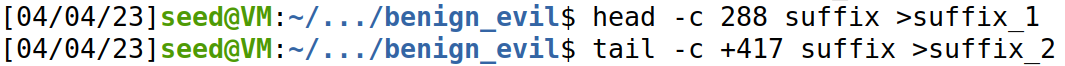
Text

Description automatically generated

Now, the last 128 bits of ‘out1’ file is P and last 128 bits of ‘out2’ is Q.



Now, based on the question, we split the suffix file into 2 leaving 128 bits in the middle for concatenating P in the middle.



Now, we do the concatenation based on the below image to form the desired files.

Diagram

Description automatically generated





Now, we check the difference between the output files, and then check the MD5 sum value of both.

Text

Description automatically generated

We can see that both the binary files differ, but they have the same MD5 value.

Now, we execute the object files and check their outputs.

In the ideal scenario, we should be getting ‘Executing benign code...’ as the output for ‘a1.out’ and ‘Executing malicious code...’ for ‘a2.out’.

Text

Description automatically generated with medium confidence

The desired outcome has been achieved.