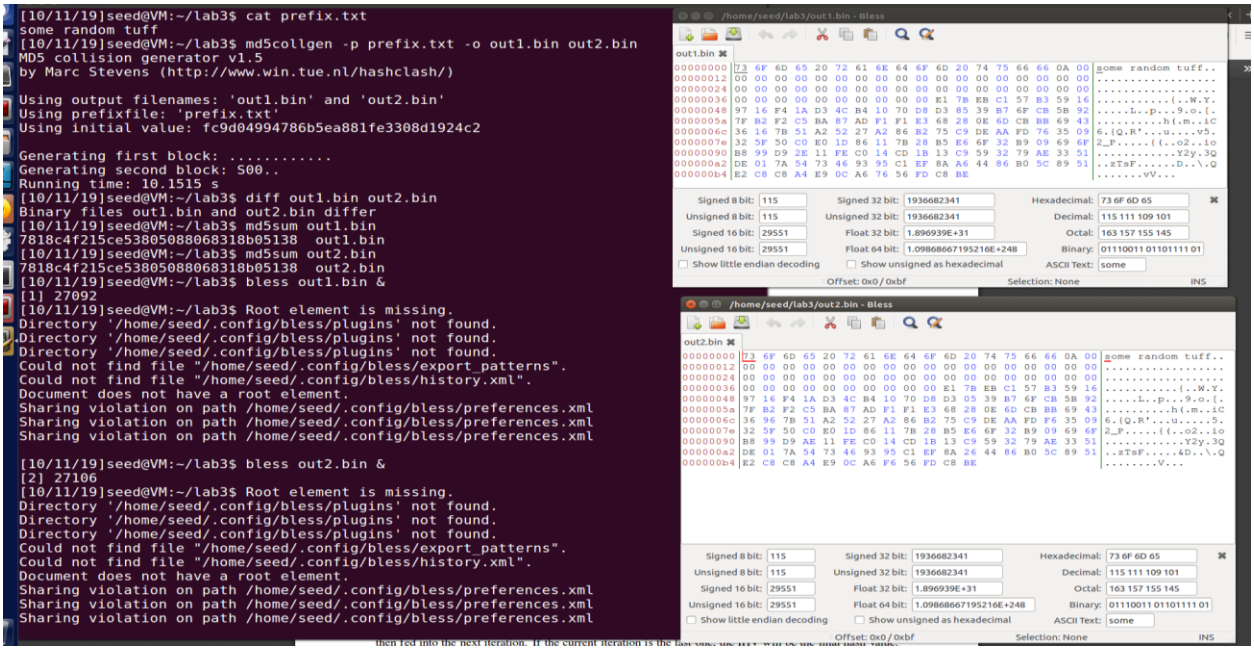
LAB3: Hash

Pramod kumar, pjk5502@psu.edu

Task 1:

Execution :

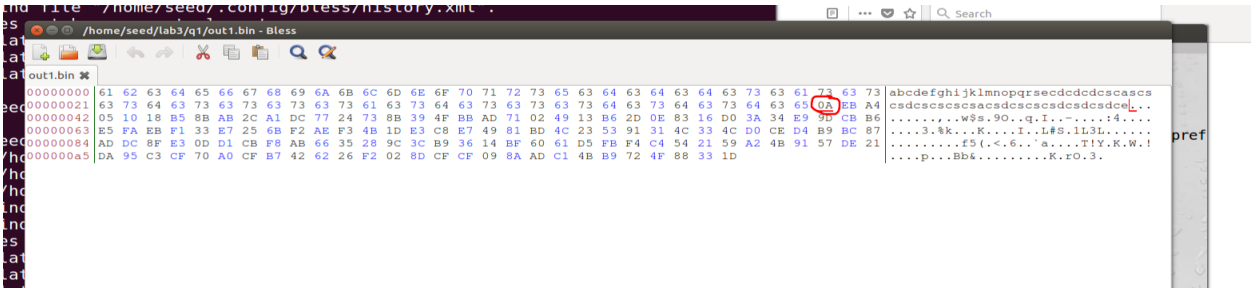


Observation

Subtask 1: when file size is less then 64 or (448 bit mod 512bit), rest of byte in are padded with one OA and followed by (OO)* .

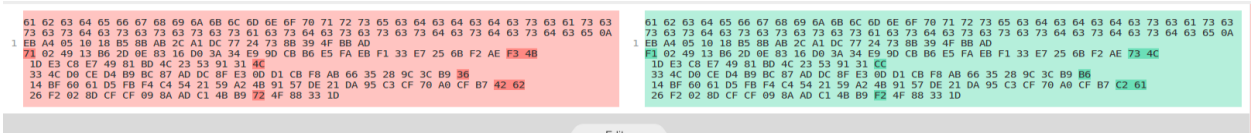
Subtask 2: Created a file with 64 byte, As we can see it doesn't have long padding but it does have 1 of padding.

```
/bin/bash 80x30
[10/11/19]seed@VM:~/.../q1$  md5collgen -p file64.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: 'file64.txt'
Using initial value: 070aceac43891f5b61d68c10576a9032

Generating first block: ..
Generating second block: S00.....................
Running time: 3.71923 s
[10/11/19]seed@VM:~/.../q1$ bless out1.bin
Root element is missing.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Document does not have a root element.
[10/11/19]seed@VM:~/.../q1$ ls -lrt
total 12
-rw-rw-r-- 1 seed seed  64 Oct 11 18:43 file64.txt
-rw-rw-r-- 1 seed seed 192 Oct 11 18:44 out2.bin
-rw-rw-r-- 1 seed seed 192 Oct 11 18:44 out1.bin
[10/11/19]seed@VM:~/.../q1$ bless out1.bin
```



Subtask 3 : those file has same hash but there is 8 bytes are different.

71,F3,4B,4C,36, 42, 62, 72



Task 2:

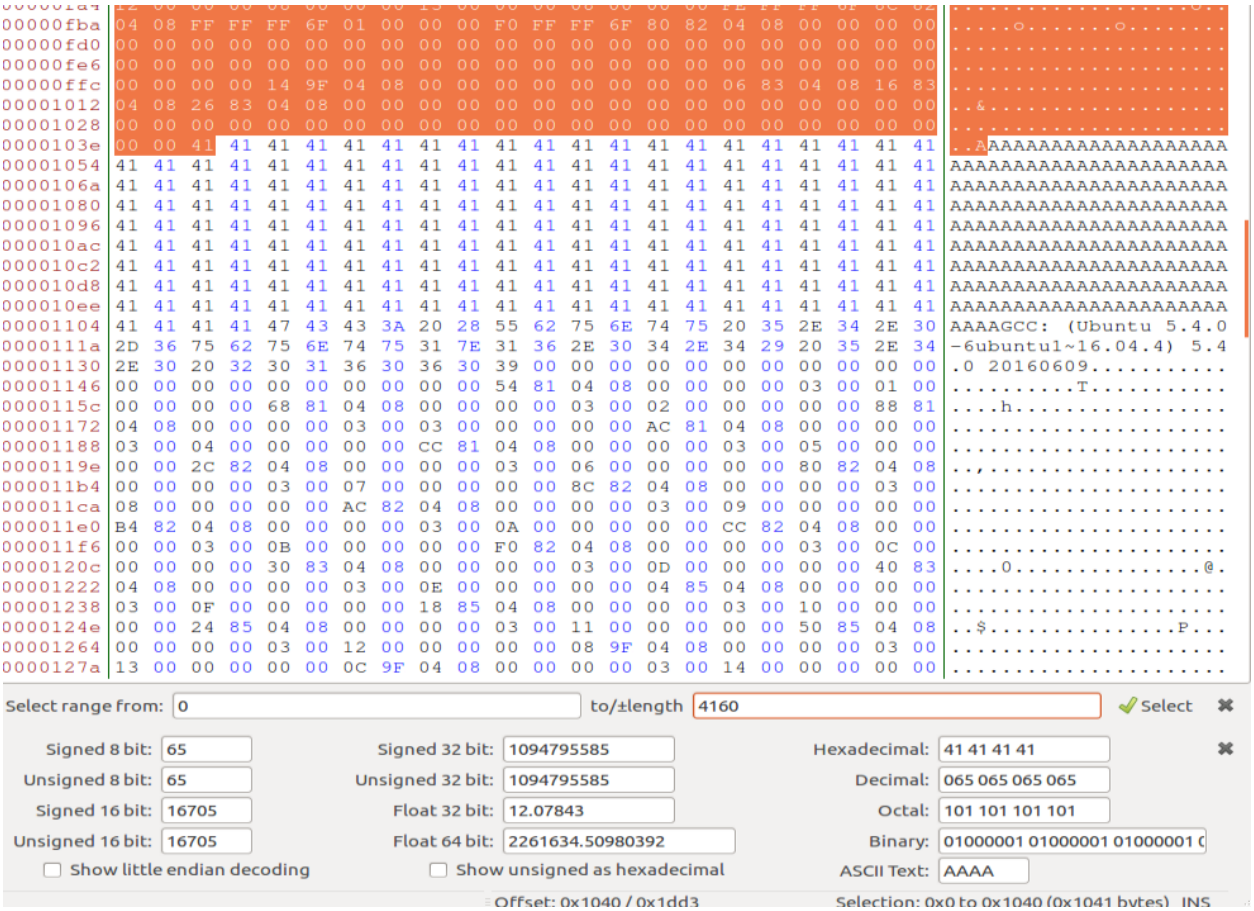
Create a string file suffix.txt. concatenate with out1.bin and out2.bin file from task1 as they have same hash function. On checking md5 of new files it proves the property

```
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ md5sum out2.bin
7818c4f215ce538050880683l8b05138  out2.bin
[10/11/19]seed@VM:~/lab3$ md5sum out1.bin
7818c4f215ce538050880683l8b05138  out1.bin
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ echo "Pramod" > suffix.txt
[10/11/19]seed@VM:~/lab3$ cat out2.bin suffix.txt > out2_concat.bin
[10/11/19]seed@VM:~/lab3$ cat out1.bin suffix.txt > out1_concat.bin
[10/11/19]seed@VM:~/lab3$ md5sum out2_concat.bin
be8c795807e945aa2629e1876c9d8ef0  out2_concat.bin
[10/11/19]seed@VM:~/lab3$ md5sum out1_concat.bin
be8c795807e945aa2629e1876c9d8ef0  out1_concat.bin
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ cat suffix.txt
Pramod
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
```

Task 3: Write a program which print 200 “A” and compile with gcc to generate executable.

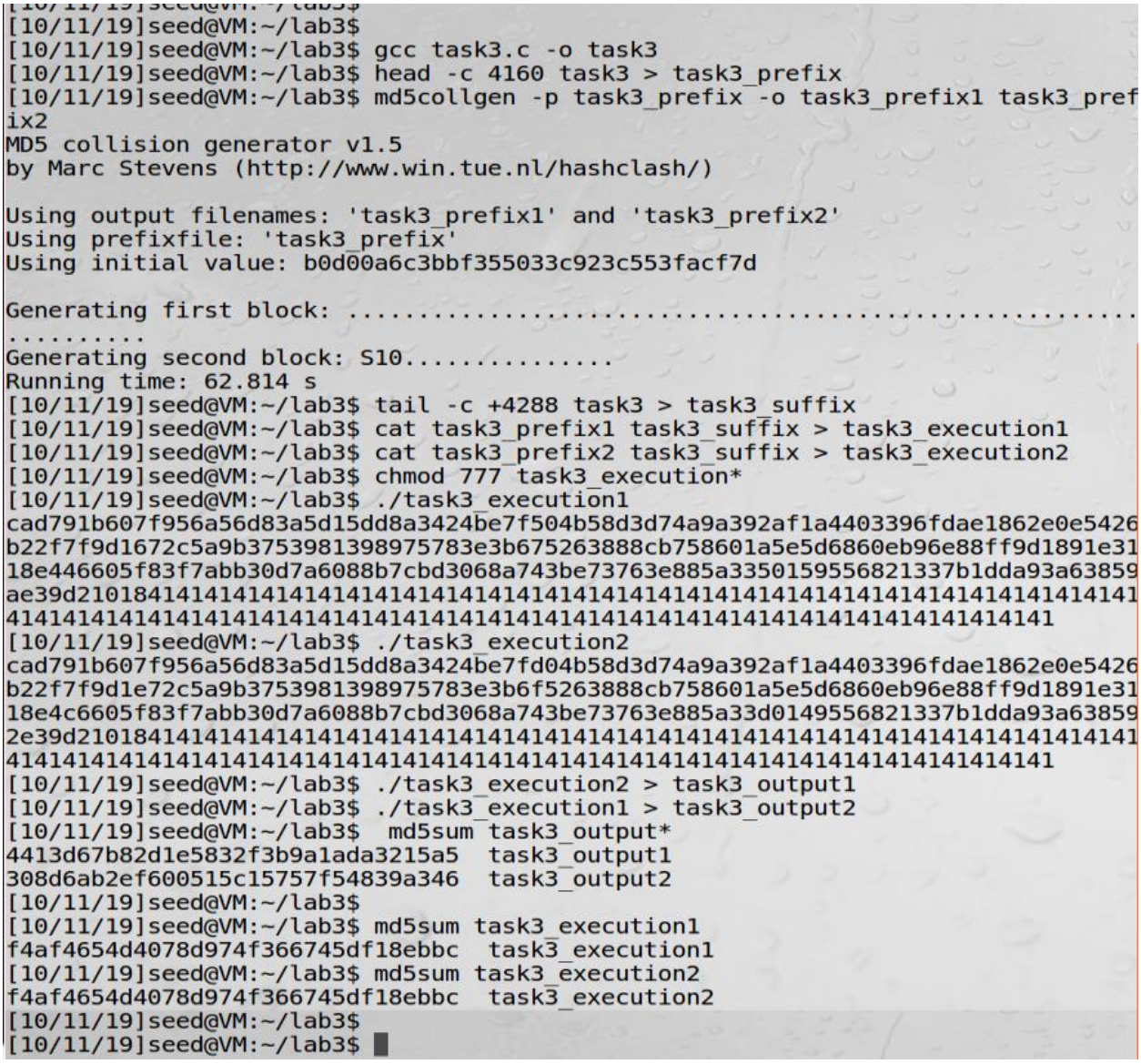
```
Terminal
#include <stdio.h>
unsigned char xyz[200] = {'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'
,'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A',
'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'
A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'
',A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'
,'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A',
'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'
A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'
',A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'
,'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'}; // populated with self
 values.
int main()
{
  int i;
  for (i=0; i<200; i++)
  {
    printf("%x", xyz[i]);
  }
  printf("\n");
  return 0;
}
~
~
~
```

Open executable, As we can see in bless, 1st a is 0x1040 offset(which is multiple of 64). So we can use this as PREFIX.



Creating prefix and its md5collsion blocks

Now lets create suffix after leaving 128 and create 2 New executable and run the program



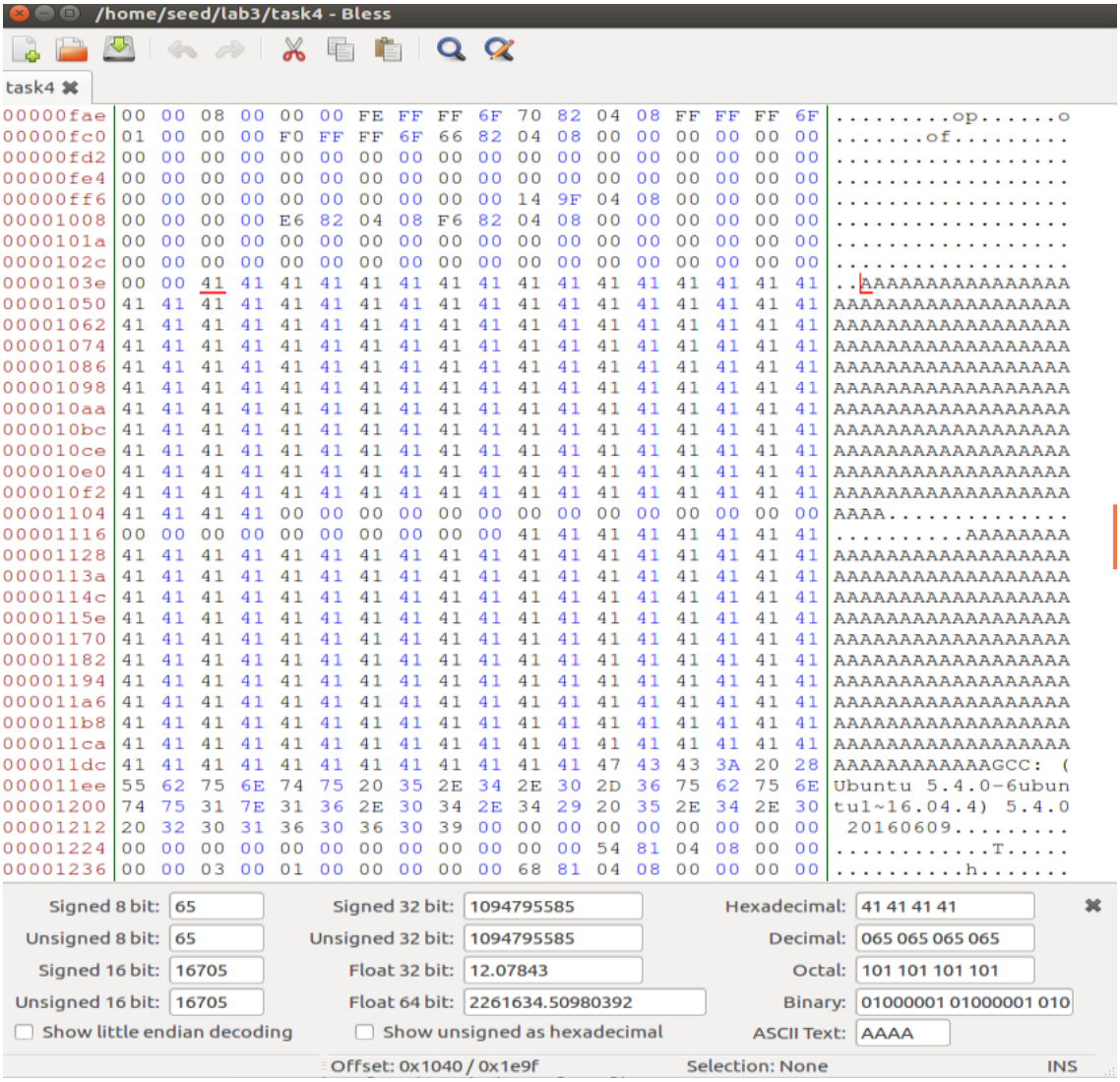
As we can see that “task3_execution1 and task3_execution2” has same md5 but output of both is different and we can validate with md5 of output of both executions.

Task4:

Create a program which has 2 array and check those array bitwise. If both are same then execute one path else other path.. according to below code, both array contain same elements

```c
#include<stdio.h>

unsigned char b[200] = {'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'};
unsigned char a[200] = {'A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A','A'};

int main()
{
        int flag = 1;
        for(int i=0;i<200;i++)
        {
                if(a[i] != b[i])
                {
                        flag = 0;
                        break;
                }
        }
        if(flag) {
                printf("benign code!!\n");
        } else {
                printf("malicious code!!!\n");
        }
        return 0;
}
~
~
~
"task4.c" 23L, 1880C written                                  4,823          All
```

Bless output to show the output of executable



Create prefix, md5 hash collision block and suffix. But this time lets take 66*64 (4224) as offset

```
[10/11/19]seed@VM:~/lab3$ head -c 4224 task4 > task4_prefix
[10/11/19]seed@VM:~/lab3$ md5collgen -p task4_prefix -o task4_prefix1 task4_prefix2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'task4_prefix1' and 'task4_prefix2'
Using prefixfile: 'task4_prefix'
Using initial value: 951e7a59b67db24ff131686ce5e39fa9

Generating first block: .
Generating second block: S01.................
Running time: 0.651811 s
[10/11/19]seed@VM:~/lab3$ md5sum task4_prefix*
4ff9b3c50cb4e81cd603b724ba696645  task4_prefix
66a95eaa480a59e1b2365b84f2d06209  task4_prefix1
66a95eaa480a59e1b2365b84f2d06209  task4_prefix2
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ tail -c +4353 task4 > task4_suffix
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
```

We have created Suffix in previous step.

Let create bad and begin programs:

Good program will have those modified 128 bytes in array A and B (which will P & P according to guide) will bad will have Q and P.

Now we need to extract 128 bytes from and copy in array B at right location in both good/bad program.

Step:

1) Get starting 8 byte, which is reaming element in array A in the program from the suffix.
2) Append these in both programs, so now both programs have full array A.
3) Remove those 8 bytes from suffix and create suffix which will be starting from end of array A to the end of program.
4) Now move get the byte between end of array A and starting of array B and append at the end of prefix in step 2. And remove those sandwich bytes from suffix
5) Now both program has code will starting of array B. and suffix has code till end.
6) Now remove 200 byte array B from suffix and create new suffix.
7) Now fetch 200 byte from program 1 ,which will have array A and we call in 128 _changed_byte array.
8) Now concatenate, code will starting of array B + 128_changed_byte_array + suffix from end of array B to end of program.
9) Check Hash of both program. Which is same.
10) Run to see both program that they have different path.

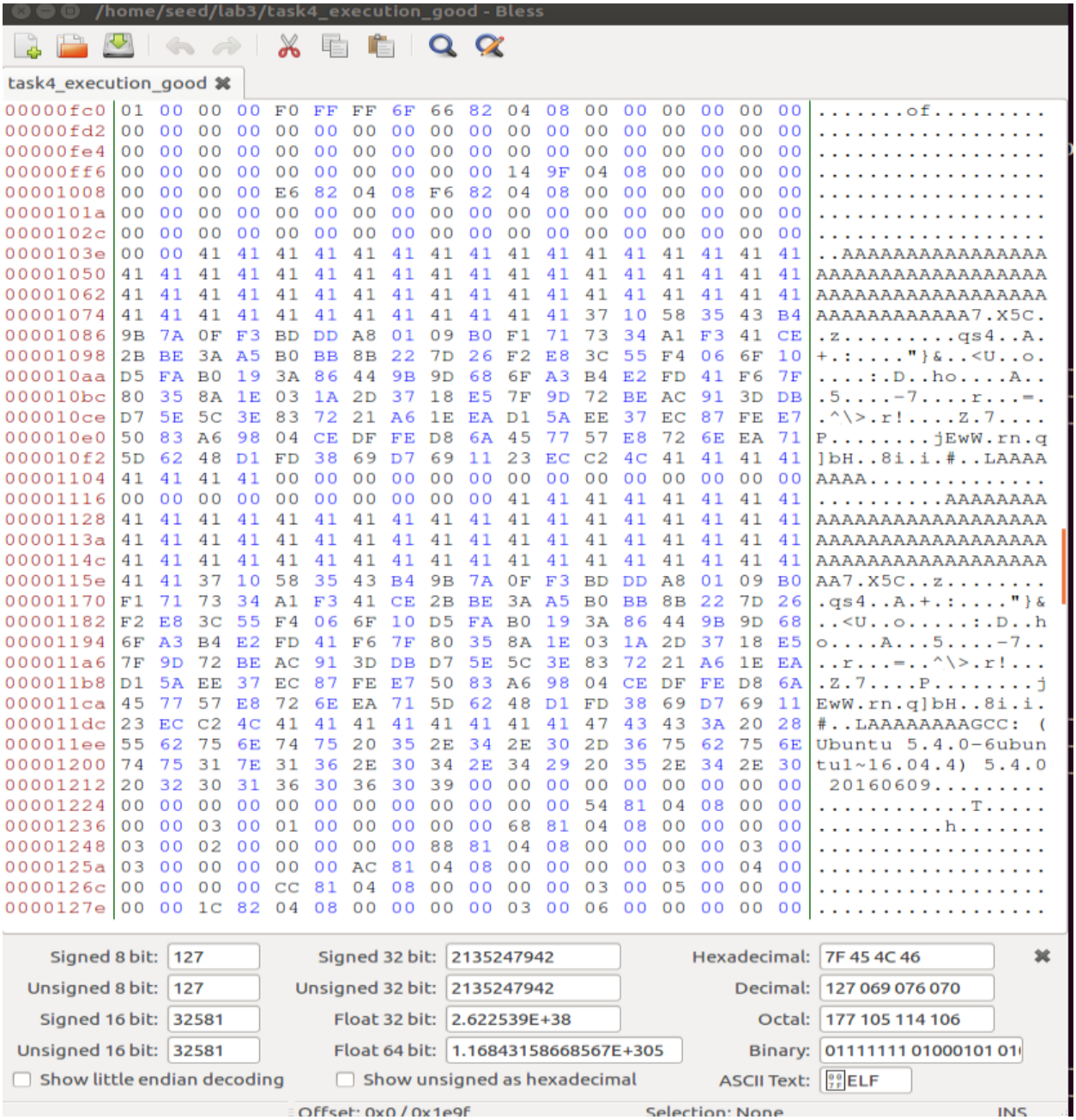
```
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ head -c 8 task4_suffix > task4_remaining_array_element
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ cat task4_prefix1 task4_remaining_array_element > task4_prefix1_till_first_array
[10/11/19]seed@VM:~/lab3$ cat task4_prefix2 task4_remaining_array_element > task4_prefix2_till_first_array
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ tail -c +9 task4_suffix > task4_suffix_trimmerd
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ tail -c +25 task4_suffix_trimmerd > task4_suffix_starting_at_second_array
[10/11/19]seed@VM:~/lab3$ head -c 24 task4_suffix_trimmerd > sandwich_between_both_array
[10/11/19]seed@VM:~/lab3$ cat task4_prefix1_till_first_array sandwich_between_both_array > task4_prefix1_till_second_array_start
[10/11/19]seed@VM:~/lab3$ cat task4_prefix2_till_first_array sandwich_between_both_array > task4_prefix2_till_second_array_start
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ tail -c +201 task4_suffix
task4_suffix
task4_suffix_starting_at_second_array
task4_suffix_trimmerd
[10/11/19]seed@VM:~/lab3$ tail -c +201 task4_suffix_starting_at_second_array > task4_suffix_after_second_array_ending
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ tail -c +4161 task4_prefix1_till_first_array > 128_changed_byte_P
[10/11/19]seed@VM:~/lab3$ cat task4_prefix1_till_second_array_start 128_changed_byte_P task4_suffix_after_second_array_ending > task4_executi
on_good
[10/11/19]seed@VM:~/lab3$ cat task4_prefix2_till_second_array_start 128_changed_byte_P task4_suffix_after_second_array_ending > task4_executi
on_bad
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$ mad5sum task4_execution_*
No command 'mad5sum' found, did you mean:
 Command 'md5sum' from package 'coreutils' (main)
mad5sum: command not found
[10/11/19]seed@VM:~/lab3$ md5sum task4_execution_*
ba0b9dd3a635968eab20c9106757f483  task4_execution_bad
ba0b9dd3a635968eab20c9106757f483  task4_execution_good
[10/11/19]seed@VM:~/lab3$ chmod 777 task4_execution_*
[10/11/19]seed@VM:~/lab3$ ./task4_execution_good
benign code!!
[10/11/19]seed@VM:~/lab3$ ./task4_execution_bad
malicious code!!!
[10/11/19]seed@VM:~/lab3$
```

To avoid confusion with “128 changed byte” name: it is the array which has 128 byte changed but whole length is 200 byte.

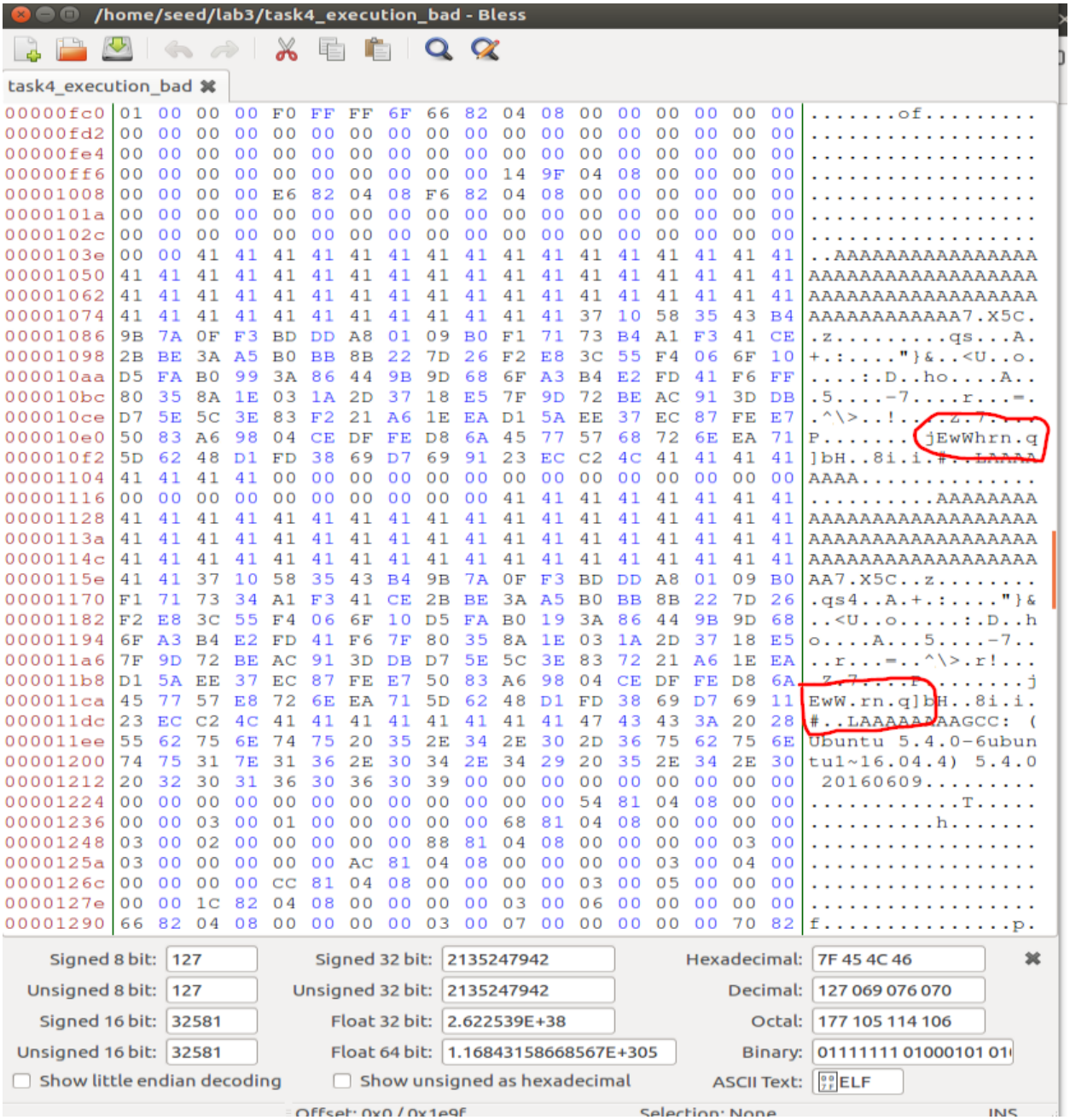
```
[10/11/19]seed@VM:~/lab3$ chmod 777 task4_execution_*
[10/11/19]seed@VM:~/lab3$ ./task4_execution_good
benign code!!
[10/11/19]seed@VM:~/lab3$ ./task4_execution_bad
malicious code!!!
[10/11/19]seed@VM:~/lab3$ ls -lrt 128_changed_byte_P
-rw-rw-r-- 1 seed seed 200 Oct 11 23:11 128_changed_byte_P
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
[10/11/19]seed@VM:~/lab3$
```

Bless of both program:

Program 1:



program 2.


/home/seed/lab3/task4_execution_bad - Bless
task4_execution_bad
00000fc0 01 00 00 00 F0 FF FF 6F 66 82 04 08 00 00 00 00 00 00 ....of.....
00000fd2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000fe4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000ff6 00 00 00 00 00 00 00 00 00 00 14 9F 04 08 00 00 00 00 ....
00001008 00 00 00 00 E6 82 04 08 F6 82 04 08 00 00 00 00 00 00 ....
0000101a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0000102c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0000103e 00 00 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 ..AAAAAAAAAAAAAAAA
00001050 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00001062 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
00001074 41 41 41 41 41 41 41 41 41 41 41 41 37 10 58 35 43 B4 AAAAAAAAAAAA7.X5C.
00001086 9B 7A 0F F3 BD DD A8 01 09 B0 F1 71 73 B4 A1 F3 41 CE .z.........qs...A.
00001098 2B BE 3A A5 B0 BB 8B 22 7D 26 F2 E8 3C 55 F4 06 6F 10 +.:....."}&..<U..o.
000010aa D5 FA B0 99 3A 86 44 9B 9D 68 6F A3 B4 E2 FD 41 F6 FF ....:.D..ho....A..
000010bc 80 35 8A 1E 03 1A 2D 37 18 E5 7F 9D 72 BE AC 91 3D DB .5....-7....r...=.
000010ce D7 5E 5C 3E 83 F2 21 A6 1E EA D1 5A EE 37 EC 87 FE E7 .^\>..!....Z.7....
000010e0 50 83 A6 98 04 CE DF FE D8 6A 45 77 57 68 72 6E EA 71 P.......jEwWhrn.q
000010f2 5D 62 48 D1 FD 38 69 D7 69 91 23 EC C2 4C 41 41 41 41]bH..8i.i.#..LAAAA
00001104 41 41 41 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AAAA..............
00001116 00 00 00 00 00 00 00 00 00 00 41 41 41 41 41 41 41 41 ....AAAAAAAA
00001128 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
0000113a 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
0000114c 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAAAA
0000115e 41 41 37 10 58 35 43 B4 9B 7A 0F F3 BD DD A8 01 09 B0 AA7.X5C..z........
00001170 F1 71 73 34 A1 F3 41 CE 2B BE 3A A5 B0 BB 8B 22 7D 26 .qs4..A.+.:...."}&
00001182 F2 E8 3C 55 F4 06 6F 10 D5 FA B0 19 3A 86 44 9B 9D 68 ..<U..o.....:.D..h
00001194 6F A3 B4 E2 FD 41 F6 7F 80 35 8A 1E 03 1A 2D 37 18 E5 o....A...5....-7..
000011a6 7F 9D 72 BE AC 91 3D DB D7 5E 5C 3E 83 72 21 A6 1E EA ..r...=..^\>.r!...
000011b8 D1 5A EE 37 EC 87 FE E7 50 83 A6 98 04 CE DF FE D8 6A .Z.7....P........j
000011ca 45 77 57 E8 72 6E EA 71 5D 62 48 D1 FD 38 69 D7 69 11 EwW.rn.q]bH..8i.i.
000011dc 23 EC C2 4C 41 41 41 41 41 41 41 41 47 43 43 3A 20 28 #..LAAAAAAAAGCC: (
000011ee 55 62 75 6E 74 75 20 35 2E 34 2E 30 2D 36 75 62 75 6E Ubuntu 5.4.0-6ubun
00001200 74 75 31 7E 31 36 2E 30 34 2E 34 29 20 35 2E 34 2E 30 tu1~16.04.4) 5.4.0
00001212 20 32 30 31 36 30 36 30 39 00 00 00 00 00 00 00 00 00 20160609.........
00001224 00 00 00 00 00 00 00 00 00 00 00 00 54 81 04 08 00 00 ....T.....
00001236 00 00 03 00 01 00 00 00 00 00 68 81 04 08 00 00 00 00 ....h.......
00001248 03 00 02 00 00 00 00 00 88 81 04 08 00 00 00 00 03 00 ....
0000125a 03 00 00 00 00 00 AC 81 04 08 00 00 00 00 03 00 04 00 ....
0000126c 00 00 00 00 CC 81 04 08 00 00 00 00 03 00 05 00 00 00 ....
0000127e 00 00 1C 82 04 08 00 00 00 00 03 00 06 00 00 00 00 00 ....
00001290 66 82 04 08 00 00 00 00 03 00 07 00 00 00 00 00 70 82 f..............p.
Signed 8 bit: 127
Unsigned 8 bit: 127
Signed 16 bit: 32581
Unsigned 16 bit: 32581
Show little endian decoding
Signed 32 bit: 2135247942
Unsigned 32 bit: 2135247942
Float 32 bit: 2.622539E+38
Float 64 bit: 1.16843158668567E+305
Show unsigned as hexadecimal
Hexadecimal: 7F 45 4C 46
Decimal: 127 069 076 070
Octal: 177 105 114 106
Binary: 01111111 01000101 01
ASCII Text: ELF
Offset: 0x0 / 0x1e9f
Selection: None
INS