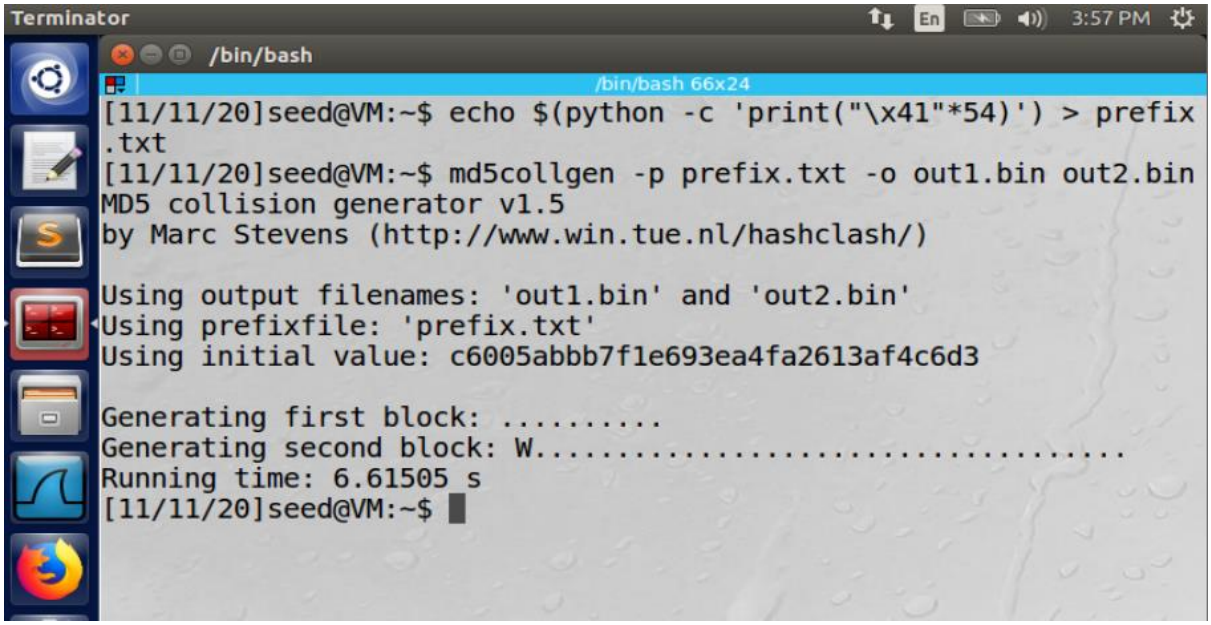
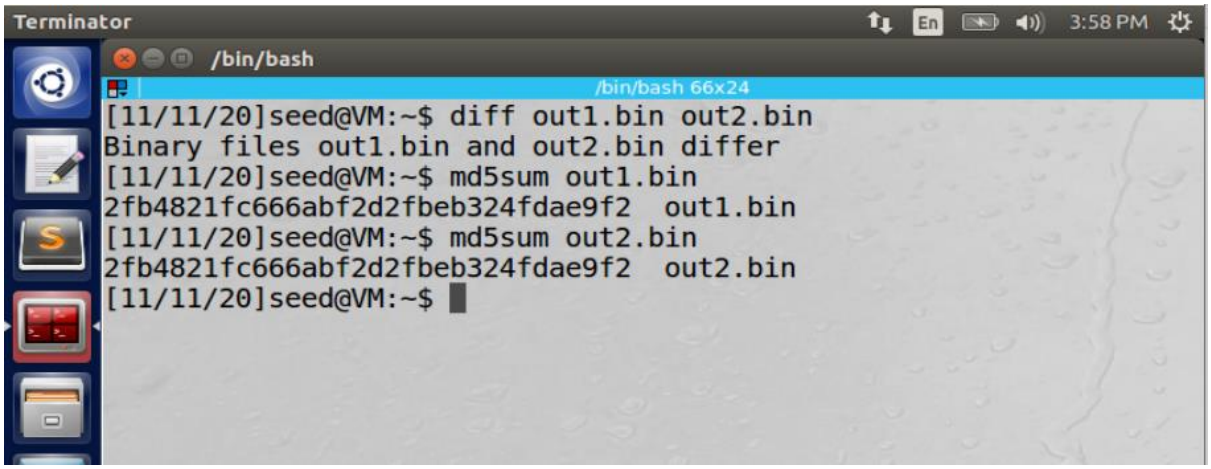
Lab: MD5 Collision Attack

Task 1: Generating Two Different Files with the Same MD5 Hash

- Executed below command to create prefix file of size less than 64 bytes.
  *echo $(python -c 'print("\x41"*54)') > prefix.txt*
- Executed given command to generate two different files with same beginning part of prefix
  $ md5collgen -p prefix.txt -o out1.bin out2.bin



- Checking if the output files are different and the hash sums are same using given command:
  $ diff out1.bin out2.bin
  $ md5sum out1.bin
  $ md5sum out2.bin

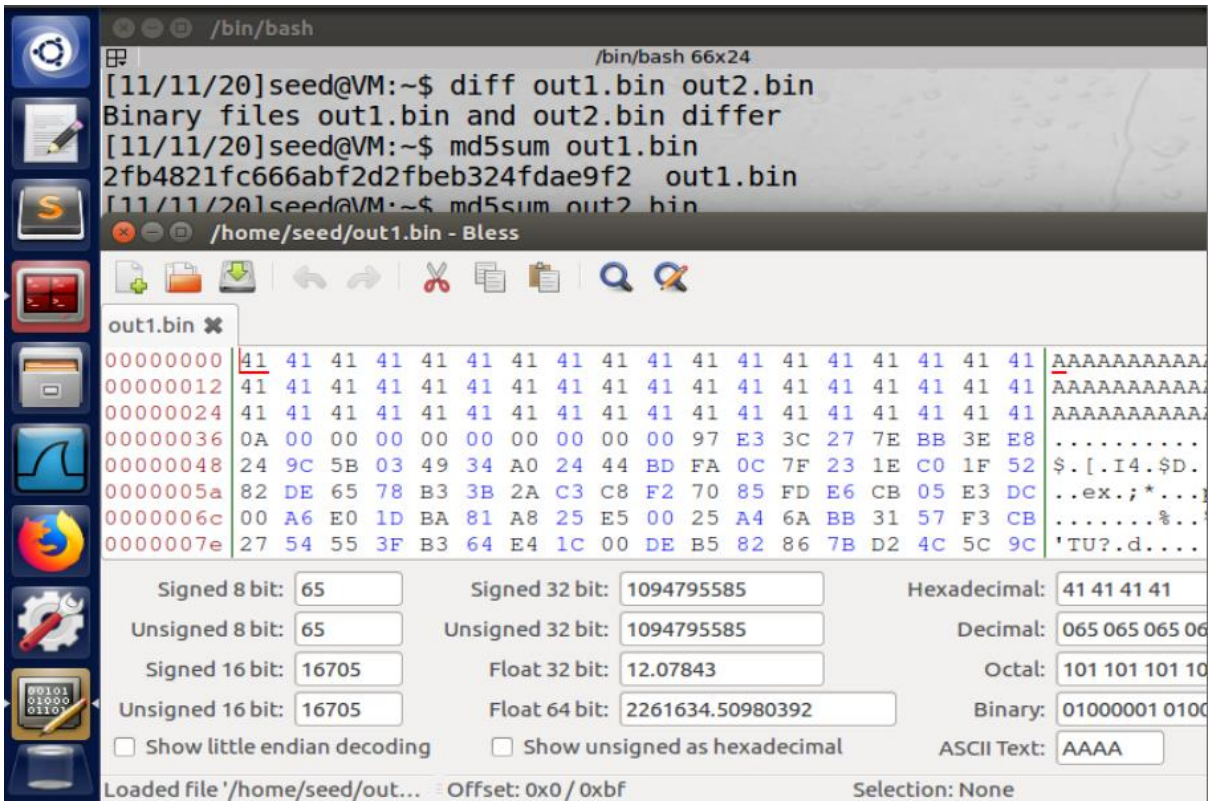


- To open binary files (out1.bin and out2.bin), installed bless on vm using below command
  sudo apt-get install bless

Question 1. If the length of your prefix file is not multiple of 64, what is going to happen?

Answer-1: If the length of the prefix file is not multiple of 64 bytes, then bytes which are multiple of 64 are padded with regular expression - (0A) (00) *.

For this experiment, I have created prefix.txt file of 55 bytes and generated two different output files out1.bin and out2.bin. Then, checked if output files are differing and hash sum (md5sum) are same for both output files. With the help of bless editor, opened binary file generated. Refer below snapshot:



Question 2. Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens.

Answer-2: If the size of prefix file is exactly 64 bytes, then bytes which are multiple of 64 are not padded with zero.

For this experiment, I have created prefix.txt file of 64 bytes and generating two output files out1.bin and out2.bin

*echo $(python -c 'print("\x41"*63)') > prefix64.txt*

```
-rw-rw-r-- 1 seed seed      64 Nov 11 16:03 prefix64.txt
```

md5collgen -p prefix64.txt -o out64_1.bin out64_2.bin

```
Terminator                                                    En   4:04 PM
/bin/bash
/bin/bash 66x24
[11/11/20]seed@VM:~$ echo $(python -c 'print("\x41"*63)') > prefix
64.txt
[11/11/20]seed@VM:~$ md5collgen -p prefix64.txt -o out64_1.bin out
64_2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out64_1.bin' and 'out64_2.bin'
Using prefixfile: 'prefix64.txt'
Using initial value: 3a136d35359c6ae88c3b9d2b9747734b

Generating first block: ............
Generating second block: S11...............
Running time: 18.6241 s
[11/11/20]seed@VM:~$
```

Checked if output files are differing and hash sum (md5sum) are same for both output files. With the help of bless editor, opened binary file generated. Refer below snapshot:

```
/home/seed/out64_1.bin - Bless
out64_1.bin
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAA
00000012 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAA
00000024 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAA
00000036 41 41 41 41 41 41 41 41 41 0A C5 E9 F5 46 76 0F CD DD AAAAAAAAA.
00000048 B2 45 6D D1 73 BF 78 DA 43 11 16 16 1B A5 54 C3 50 F2 .Em.s.x.C.
0000005a 38 DD 45 10 28 3F AA EB FB CA F0 2D 09 22 6B 08 42 E4 8.E.(?....
0000006c D1 D7 43 40 1F 10 30 EF CF DA 4C F3 BC E8 36 C3 16 63 ..C@..0...
0000007e 4B 6E 90 A8 63 AE F3 77 B7 C3 13 8F 2A B9 19 99 95 DB Kn..c..w..

Signed 8 bit: 65          Signed 32 bit: 1094795585      Hexadecimal: 41 41 41 41
Unsigned 8 bit: 65        Unsigned 32 bit: 1094795585    Decimal: 065 065 065 06
Signed 16 bit: 16705      Float 32 bit: 12.07843         Octal: 101 101 101 10
Unsigned 16 bit: 16705    Float 64 bit: 2261634.50980392 Binary: 01000001 0100
Show little endian decoding   Show unsigned as hexadecimal   ASCII Text: AAAA
Offset: 0x0 / 0xbf                                Selection: None
```

Question 3. Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.

Answer-3: No, not all bytes generated by md5collgen are different for two output files. Byte 96, 43 are different in both output files of 64 bytes each.

out64_1.bin

```
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAA
00000012 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAA
00000024 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAA
00000036 41 41 41 41 41 41 41 41 41 0A C5 E9 F5 46 76 0F CD DD AAAAAAAAA.
00000048 B2 45 6D D1 73 BF 78 DA 43 11 16 16 1B A5 54 C3 50 F2 .Em.s.x.C.
0000005a 38 DD 45 10 28 3F AA EB FB CA F0 2D 09 22 6B 08 42 E4 8.E.(?....
0000006c D1 D7 43 40 1F 10 30 EF CF DA 4C F3 BC E8 36 C3 16 63 ..C@..0...
0000007e 4B 6E 90 A8 63 AE F3 77 B7 C3 13 8F 2A B9 19 99 95 DB Kn..c..w..
```

out64_2.bin

```
00000000 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAA
00000012 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAA
00000024 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAA
00000036 41 41 41 41 41 41 41 41 41 0A C5 E9 F5 46 76 0F CD DD AAAAAAAAA.
00000048 B2 45 6D D1 73 BF 78 DA 43 11 16 96 1B A5 54 C3 50 F2 .Em.s.x.C.
0000005a 38 DD 45 10 28 3F AA EB FB CA F0 2D 09 22 6B 08 42 E4 8.E.(?....
0000006c D1 57 44 40 1F 10 30 EF CF DA 4C F3 BC E8 36 43 16 63 .WD@..0...
0000007e 4B 6E 90 A8 63 AE F3 77 B7 C3 13 8F 2A B9 19 99 95 DB Kn..c..w..
```

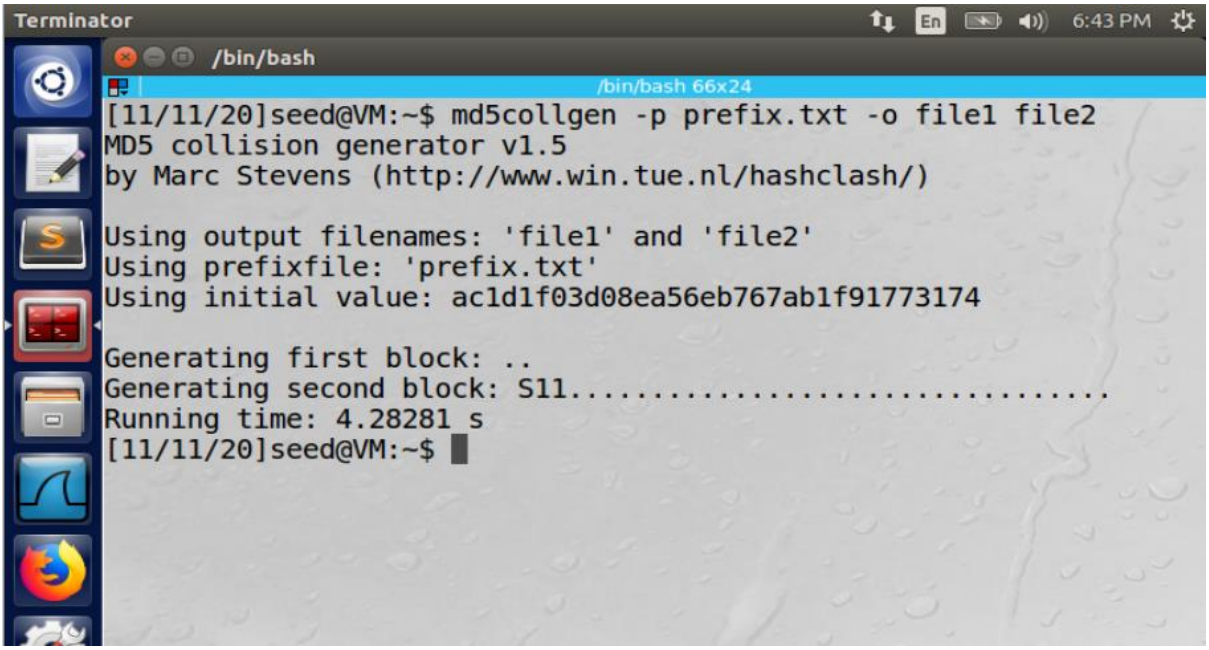
Task 2: Understanding MD5's Property

MD5 divides the data into blocks of 64 bytes and then hashes will be computed on these blocks iteratively. A compression feature that generates an intermediate hash value is the origin of MD5.

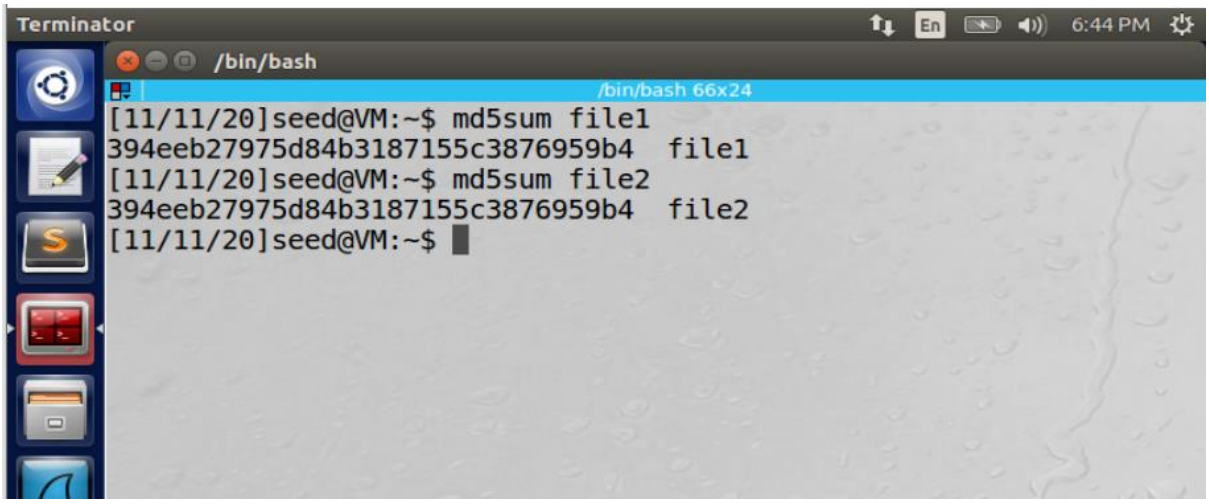
The input for the first iteration is IHV0. Deriving a property based on the function of MD5 algorithm:

Given two inputs M and N, if MD5(M) = MD5(N), then for any input T, MD5(M || T) = MD5(N || T). Using a special suffix to any two different messages with the same MD5 hash gives two new longer messages for the original and the suffix messages by concatenating, both of which have the same MD5 hash as well. Using the cat command to concatenate the contents of files.

- Creating prefix.txt file and then executed below md5collgen command: *md5collgen -p prefix.txt -o file1 file2*

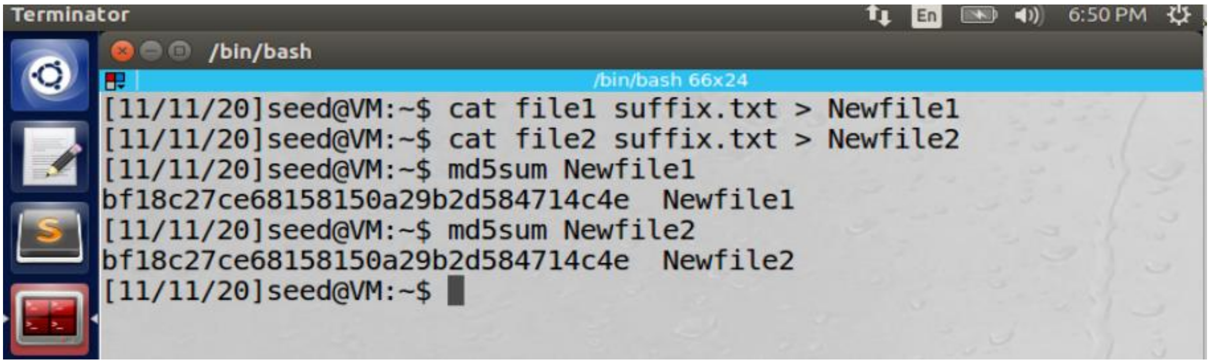


- Verifying md5 hashes generated in both file1 and file2 are same or not



- Created suffix file and executed below commands:
  $ cat file1 suffix.txt > Newfile1
  $ cat file2 suffix.txt > Newfile2

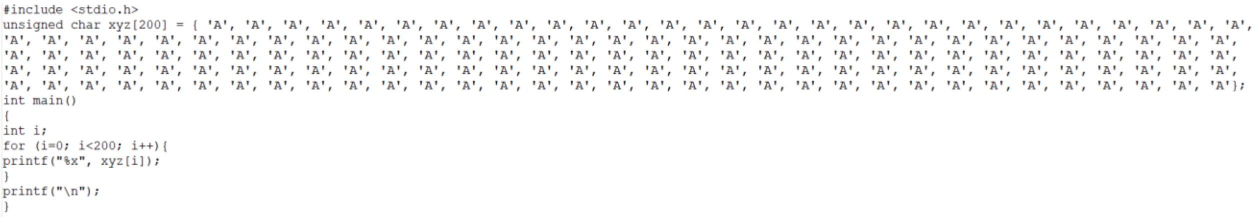
  $ md5sum Newfile1
  $ md5sum Newfile2



```
[11/11/20]seed@VM:~$ cat file1 suffix.txt > Newfile1
[11/11/20]seed@VM:~$ cat file2 suffix.txt > Newfile2
[11/11/20]seed@VM:~$ md5sum Newfile1
bf18c27ce68158150a29b2d584714c4e  Newfile1
[11/11/20]seed@VM:~$ md5sum Newfile2
bf18c27ce68158150a29b2d584714c4e  Newfile2
[11/11/20]seed@VM:~$ 
```

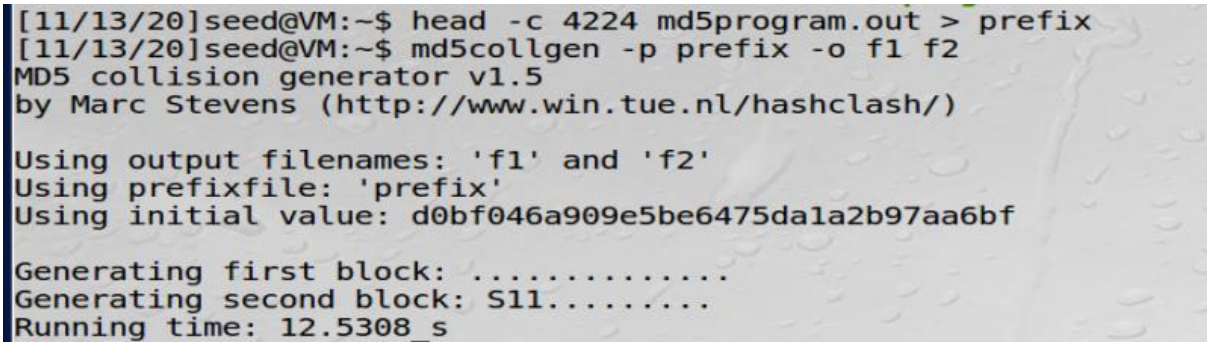
Task 3: Generating Two Executable Files with the Same MD5 Hash

- Filling the array content with A's as shown in given program (md5program.c) :



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

- Compile the program md5program.c and then execute it using below command:
  gcc md5program.c -o md5program.out
- Now dividing the executable file md5program.out into three parts of byte offset: the end part will be treated as suffix, the first part treated as prefix and in middle part, changes are required. Take byte offset as 4224 for the prefix because it is the multiple of 64.
  head -c 4224 md5program.out > prefix
- Use the md5collgen prefix file to get two files with the same hash, called f1 and f2.
  md5collgen -p prefix -o f1 f2

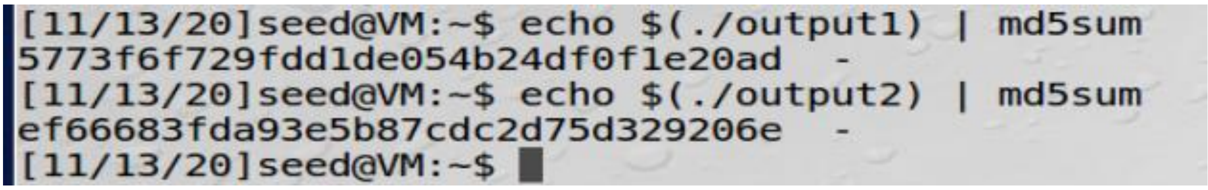


```
[11/13/20]seed@VM:~$ head -c 4224 md5program.out > prefix
[11/13/20]seed@VM:~$ md5collgen -p prefix -o f1 f2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'f1' and 'f2'
Using prefixfile: 'prefix'
Using initial value: d0bf046a909e5be6475da1a2b97aa6bf

Generating first block: ..............
Generating second block: S11.........
Running time: 12.5308 s
```

- Two files with same MD5 hash but different suffixes are generated. Appending common end using given command:
  tail -c +4353 md5program.out > suffix
- Concatenate the suffix to the two individual files f1 and f2 to create the two binary files using given command:
  cat f1 suffix > output1

cat f2 suffix > output2

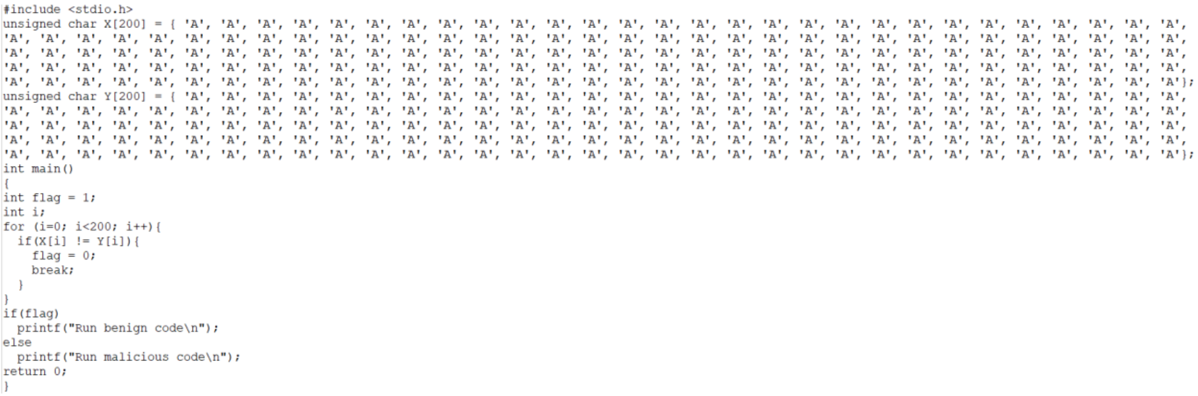
- After concatenating the files, the output files have different hash values. And can be shown using below command:
  echo $(./output1) | md5sum
  echo $(./output2) | md5sum



```
[11/13/20]seed@VM:~$ echo $(./output1) | md5sum
5773f6f729fdd1de054b24df0f1e20ad  -
[11/13/20]seed@VM:~$ echo $(./output2) | md5sum
ef66683fda93e5b87cdc2d75d329206e  -
[11/13/20]seed@VM:~$ 
```

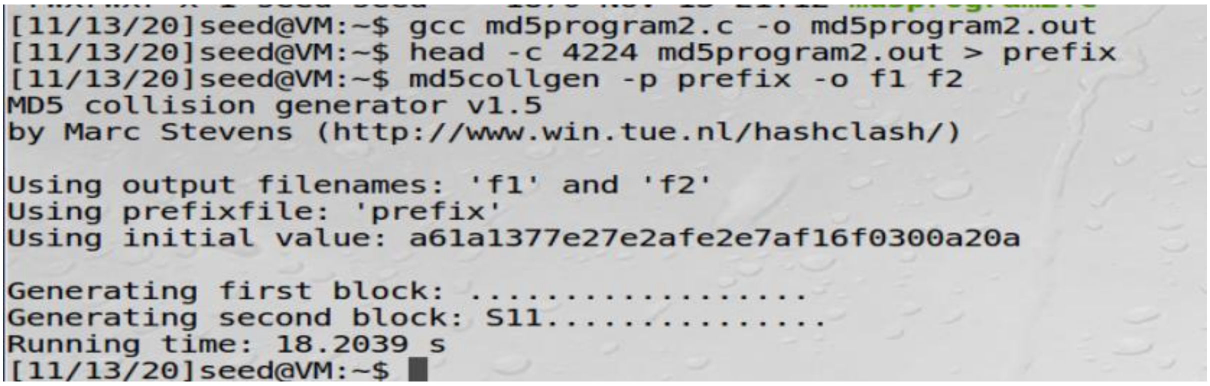
Task 4: Making the Two Programs Behave Differently

- Below program has been created which on execution provides unexpected behaviour even after it has been verified as well as checked by hashing. Two conditions are given in program- expected output and unexpected output.



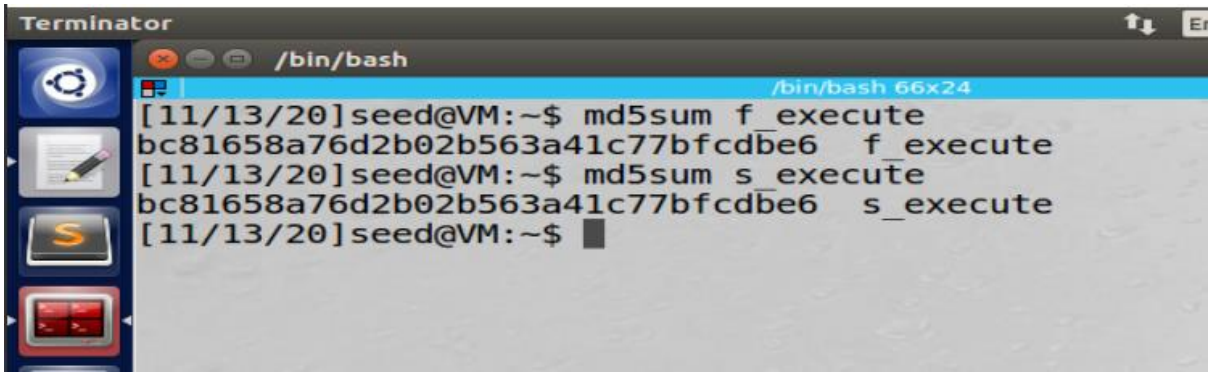
```
#include <stdio.h>
unsigned char X[200] = { 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A',
'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', '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'};
unsigned char Y[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'};
int main()
{
int flag = 1;
int i;
for (i=0; i<200; i++){
  if(X[i] != Y[i]){
    flag = 0;
    break;
  }
}
if(flag)
  printf("Run benign code\n");
else
  printf("Run malicious code\n");
return 0;
}
```

- Compile the program md5program2.c and then execute it using below command:
  gcc md5program2.c -o md5program2.out
- Take byte offset as 4224 for the prefix because it is the multiple of 64.
  head -c 4224 md5program2.out > prefix
- Use the md5collgen prefix file to get two files with the same hash, called f1 and f2.
  md5collgen -p prefix -o f1 f2

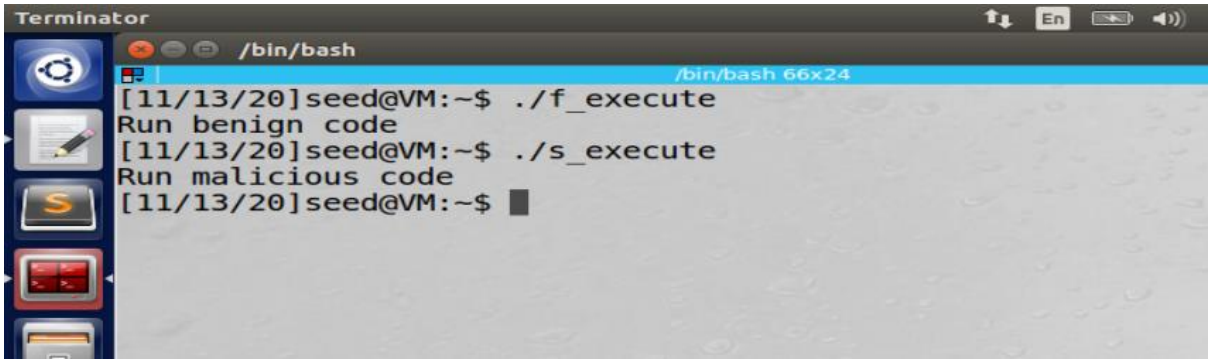


```
[11/13/20]seed@VM:~$ gcc md5program2.c -o md5program2.out
[11/13/20]seed@VM:~$ head -c 4224 md5program2.out > prefix
[11/13/20]seed@VM:~$ md5collgen -p prefix -o f1 f2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'f1' and 'f2'
Using prefixfile: 'prefix'
Using initial value: a61a1377e27e2afe2e7af16f0300a20a

Generating first block: .................
Generating second block: S11...............
Running time: 18.2039 s
[11/13/20]seed@VM:~$ 
```

- Append offset bytes at the end from md5program2.out to suffixtest using given command:
  tail -c +4353 md5program2.out > suffixtest
- Add first 8 bytes of suffixtest to those of files f1 and f2 by uing below command:
  head -c 8 suffixtest > fullarray
- Below command will create the suffix file containing all bytes after the 8th byte:
  cat f1 fullarray > out1
  cat f2 fullarray > out2
  tail -c +9 suffixtest > suffix
- With the help of below commands, added bytes between two arrays and created a file called mid.
  head -c 24 suffix > mid
- Below command will concatenate the bytes of out1 and out2 with mid to give f1mid and f2mid.
  cat out1 mid > f1mid
  cat out2 mid > f2mid
- Put the bytes after the second array in suffixtest to suffix
  tail -c +201 suffixtest > suffix
- Copy the first array from out1 to completearray
  tail -c +4161 out1 > completearray
- Now concatenate file f1mid and f2mid along with completearray and suffix to f_execute and s_execute
  cat f1mid completearray suffix > f_execute
  cat f2mid completearray suffix > s_execute
- Using f_execute and s_execute files to generate md5sum hash function
  md5sum f_execute
  md5sum s_execute



- The hashes of f_execute and s_execute are same.
- Change the mode of f_execute and s_execute to executable mode
  chmod +x f_execute
  chmod +x s_execute
- Executing f_execute and s_execute files



The hash value for the two files generated are same. Even then also, on executing these two files, one file gives benign code, and another gave malicious code as output. Hence, md5 collision vulnerability can be exploited.