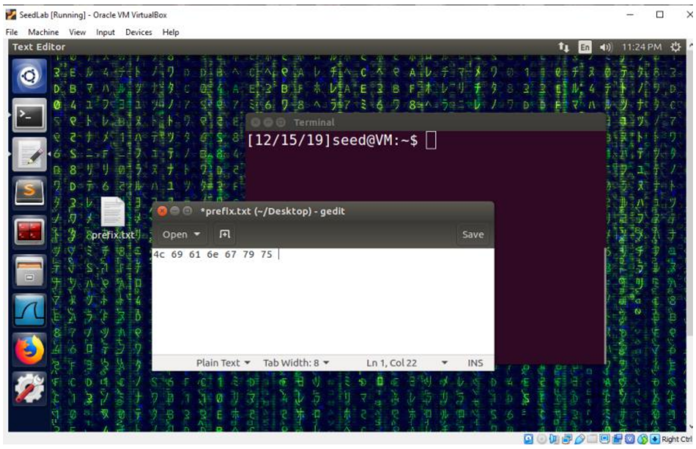
MD5 Collision Attack Lab

Liangyu W

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



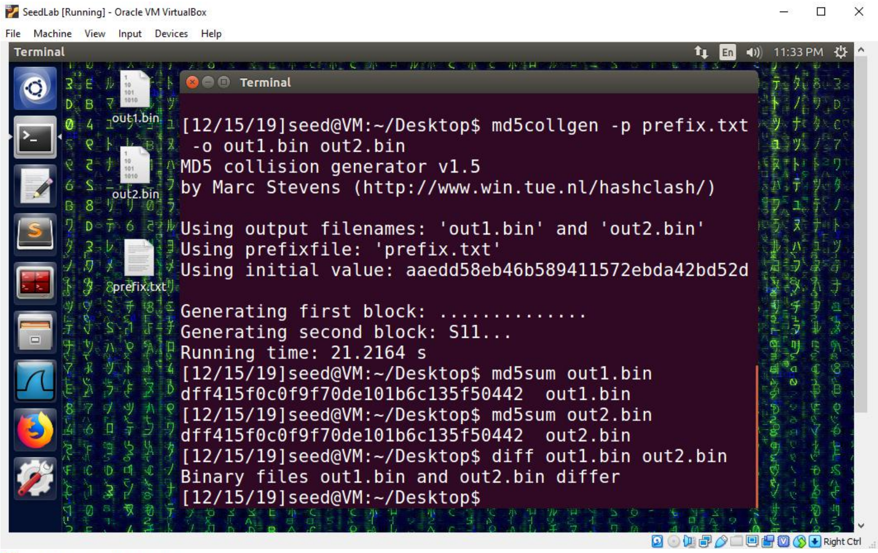
```
[12/15/19]seed@VM:~$
```

*prefix.txt (~/Desktop) - gedit

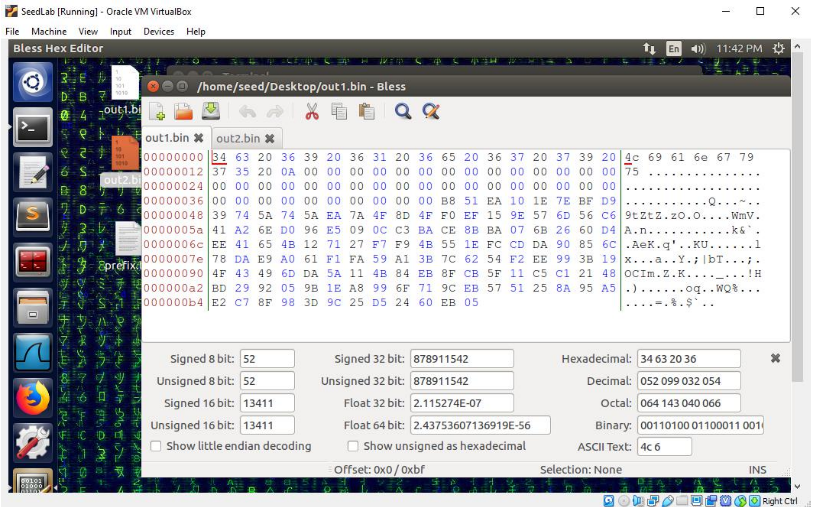
```
4c 69 61 6e 67 79 75
```

Plain Text   Tab Width: 8   Ln 1, Col 22   INS

We create a prefix file prefix.txt then run the md5collgen program to generate two files out1.bin and out2.bin with the same md5 hash.



```
[12/15/19]seed@VM:~/Desktop$ 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: aaedd58eb46b589411572ebda42bd52d

Generating first block: ..............
Generating second block: S11...
Running time: 21.2164 s
[12/15/19]seed@VM:~/Desktop$ md5sum out1.bin
dff415f0c0f9f70de101b6c135f50442  out1.bin
[12/15/19]seed@VM:~/Desktop$ md5sum out2.bin
dff415f0c0f9f70de101b6c135f50442  out2.bin
[12/15/19]seed@VM:~/Desktop$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[12/15/19]seed@VM:~/Desktop$
```



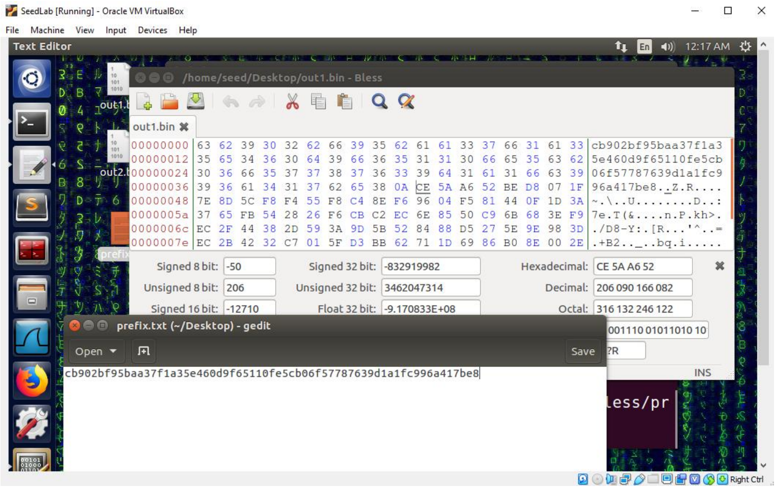
/home/seed/Desktop/out1.bin - Bless

```
00000000 34 63 20 36 39 20 36 31 20 36 65 20 36 37 20 37 39 20 4c 69 61 6e 67 79
00000012 37 35 20 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 75 ..............
00000024 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036 00 00 00 00 00 00 00 00 00 00 B8 51 EA 10 1E 7E BF D9 ...........Q...~..
00000048 39 74 5A 74 5A EA 7A 4F 8D 4F F0 EF 15 9E 57 6D 56 C6 9tZtZ.zO.O....WmV.
0000005a 41 A2 6E D0 96 E5 09 0C C3 BA CE 8B BA 07 6B 26 60 D4 A.n...........k&`.
0000006c EE 41 65 4B 12 71 27 F7 F9 4B 55 1E FC CD DA 90 85 6C .AeK.q'..KU......l
0000007e 78 DA E9 A0 61 F1 FA 59 A1 3B 7C 62 54 F2 EE 99 3B 19 x...a..Y.;|bT...;.
00000090 4F 43 49 6D DA 5A 11 4B 84 EB 8F CB 5F 11 C5 C1 21 48 OCIm.Z.K...._...!H
000000a2 BD 29 92 05 9B 1E A8 99 6F 71 9C EB 57 51 25 8A 95 A5 .)......oq..WQ%...
000000b4 E2 C7 8F 98 3D 9C 25 D5 24 60 EB 05                   ....=.%.$`..
```

Signed 8 bit: 52   Signed 32 bit: 878911542   Hexadecimal: 34 63 20 36
Unsigned 8 bit: 52   Unsigned 32 bit: 878911542   Decimal: 052 099 032 054
Signed 16 bit: 13411   Float 32 bit: 2.115274E-07   Octal: 064 143 040 066
Unsigned 16 bit: 13411   Float 64 bit: 2.43753607136919E-56   Binary: 00110100 01100011 001
□ Show little endian decoding   □ Show unsigned as hexadecimal   ASCII Text: 4c 6

Offset: 0x0 / 0xbf   Selection: None   INS

By using the Bless hex editor, we see that when content of the prefix file is not a multiple of 64 bytes, zeros are padded.   This is because md5 processes blocks of 64 bytes.


SeedLab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Text Editor
12:17 AM
/home/seed/Desktop/out1.bin - Bless
out1.bin
out1.bin
out2.bin
prefix
prefix.txt (~/Desktop) - gedit
Open
Save
INS
Right Ctrl

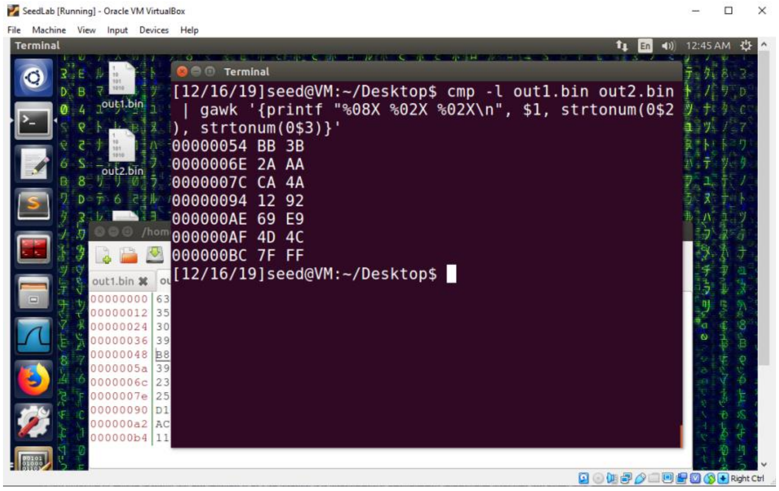
```
00000000 63 62 39 30 32 62 66 39 35 62 61 61 33 37 66 31 61 33 cb902bf95baa37f1a3
00000012 35 65 34 36 30 64 39 66 36 35 31 31 30 66 65 35 63 62 5e460d9f65110fe5cb
00000024 30 36 66 35 37 37 38 37 36 33 39 64 31 61 31 66 63 39 06f57787639d1a1fc9
00000036 39 36 61 34 31 37 62 65 38 0A CE 5A A6 52 BE D8 07 1F 96a417be8..Z.R....
00000048 7E 8D 5C F8 F4 55 F8 C4 8E F6 96 04 F5 81 44 0F 1D 3A ~.\..U........D..:
0000005a 37 65 FB 54 28 26 F6 CB C2 EC 6E 85 50 C9 6B 68 3E F9 7e.T(&....n.P.kh>.
0000006c EC 2F 44 38 2D 59 3A 9D 5B 52 84 88 D5 27 5E 9E 98 3D ./D8-Y:.[R...'^..=
0000007e EC 2B 42 32 C7 01 5F D3 BB 62 71 1D 69 86 B0 8E 00 2E .+B2.._..bq.i.....
```

Signed 8 bit: -50
Unsigned 8 bit: 206
Signed 16 bit: -12710
Signed 32 bit: -832919982
Unsigned 32 bit: 3462047314
Float 32 bit: -9.170833E+08
Hexadecimal: CE 5A A6 52
Decimal: 206 090 166 082
Octal: 316 132 246 122
001110 01011010 10
?R

```
cb902bf95baa37f1a35e460d9f65110fe5cb06f57787639d1a1fc996a417be8
```

```
less/pr
```

If we use a prefix of exactly 64 bytes, then the md5collgen generated files have no zero padding.

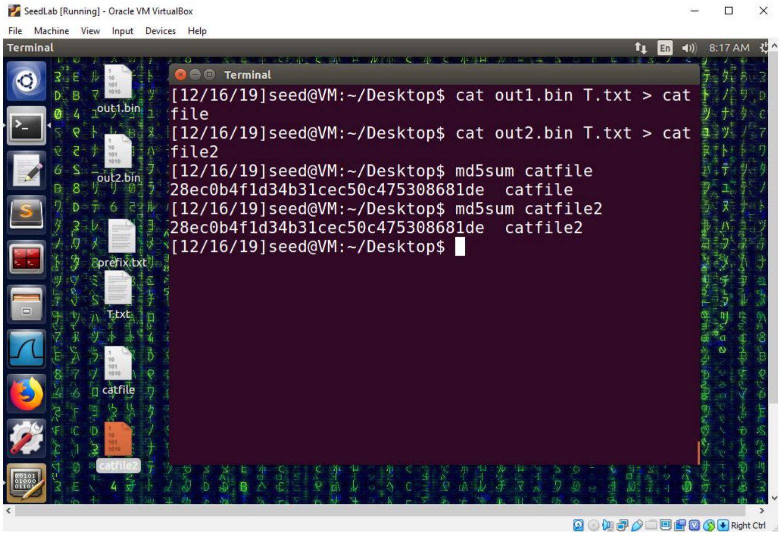

SeedLab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
12:45 AM
out1.bin
out2.bin
/hom
out1.bin
Right Ctrl

```
[12/16/19]seed@VM:~/Desktop$ cmp -l out1.bin out2.bin
 | gawk '{printf "%08X %02X %02X\n", $1, strtonum(0$2
), strtonum(0$3)}'
00000054 BB 3B
0000006E 2A AA
0000007C CA 4A
00000094 12 92
000000AE 69 E9
000000AF 4D 4C
000000BC 7F FF
[12/16/19]seed@VM:~/Desktop$
```

Using the cmp command, we can see all the differences between the two generated files.

Task 2: Understanding MD5's Property

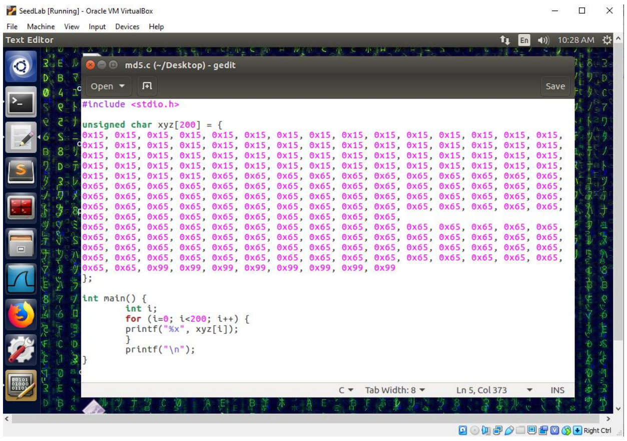
We create a text file T.txt filled with random bytes



```
[12/16/19]seed@VM:~/Desktop$ cat out1.bin T.txt > cat
file
[12/16/19]seed@VM:~/Desktop$ cat out2.bin T.txt > cat
file2
[12/16/19]seed@VM:~/Desktop$ md5sum catfile
28ec0b4f1d34b31cec50c475308681de  catfile
[12/16/19]seed@VM:~/Desktop$ md5sum catfile2
28ec0b4f1d34b31cec50c475308681de  catfile2
[12/16/19]seed@VM:~/Desktop$
```

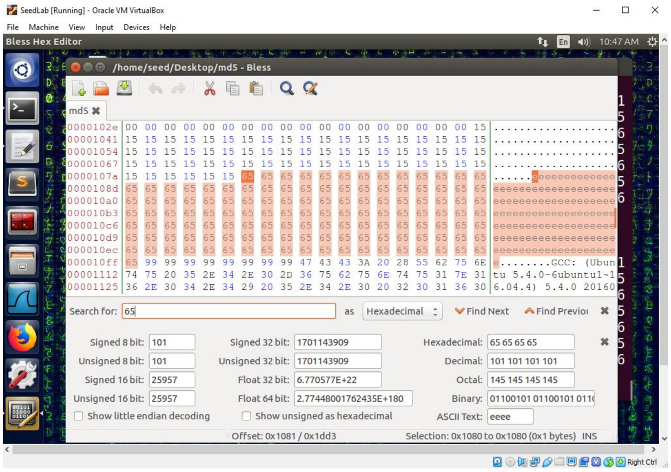
Then used the cat command to concatenate T.txt to out1.bin and out2.bin, then ran md5sum on the two generated files.   The two generated files also have identical MD5 hashes, thus confirming the common Hash function property that if M and N have the same MD5 hash, then for any input T, MD5(M||T) = MD5(N||T), where|| represents concatenation.

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



```
#include <stdio.h>

unsigned char xyz[200] = {
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99
};

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

In the xyz array, we fill prefix with 0x15, the 128-byte region with 0x65 and suffix with 0x99.



/home/seed/Desktop/md5 - Bless

```
0000102e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 15 ...................
00001041 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 ...................
00001054 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 ...................
00001067 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 ...................
0000107a 15 15 15 15 15 15 65 65 65 65 65 65 65 65 65 65 65 65 65 ......eeeeeeeeeeeee
0000108d 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 eeeeeeeeeeeeeeeeeee
000010a0 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 eeeeeeeeeeeeeeeeeee
000010b3 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 eeeeeeeeeeeeeeeeeee
000010c6 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 eeeeeeeeeeeeeeeeeee
000010d9 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 eeeeeeeeeeeeeeeeeee
000010ec 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 eeeeeeeeeeeeeeeeeee
000010ff 65 99 99 99 99 99 99 99 99 47 43 43 3A 20 28 55 62 75 6E e........GCC: (Ubun
00001112 74 75 20 35 2E 34 2E 30 2D 36 75 62 75 6E 74 75 31 7E 31 tu 5.4.0-6ubuntu1~1
00001125 36 2E 30 34 2E 34 29 20 35 2E 34 2E 30 20 32 30 31 36 30 6.04.4) 5.4.0 20160
```

Search for: 65 as Hexadecimal

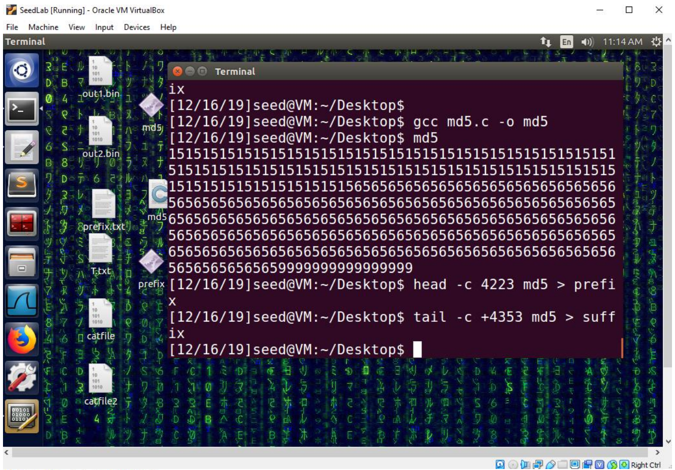
Signed 8 bit:	101	Signed 32 bit:	1701143909	Hexadecimal:	65 65 65 65
Unsigned 8 bit:	101	Unsigned 32 bit:	1701143909	Decimal:	101 101 101 101
Signed 16 bit:	25957	Float 32 bit:	6.770577E+22	Octal:	145 145 145 145
Unsigned 16 bit:	25957	Float 64 bit:	2.77448001762435E+180	Binary:	01100101 01100101 011(
□ Show little endian decoding		□ Show unsigned as hexadecimal		ASCII Text:	eeee

Offset: 0x1081 / 0x1dd3 Selection: 0x1080 to 0x1080 (0x1 bytes) INS

SeedLab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Bless Hex Editor
10:47 AM
md5
Find Next
Find Previo

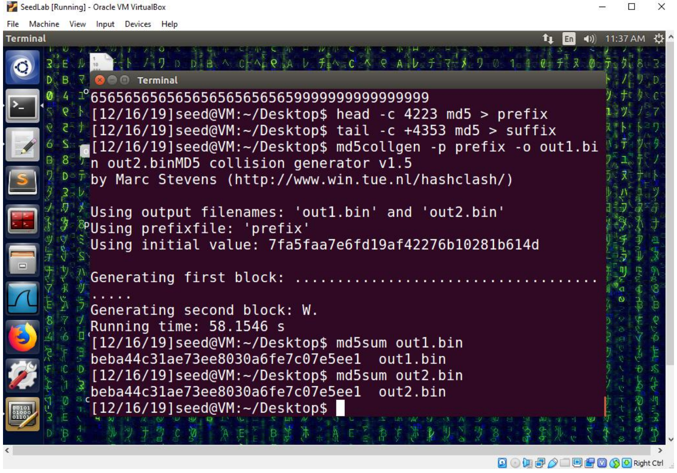
We find that the xyz array content is stored at offset 0x1040 - 0x1107 in the compiled file. The prefix of the array is stored at 0x1040 - 0x107f. The suffix of the array is stored from offset 0x1100 to the end of the file. 107f converted to decimal is 4223, 0x1100 converted to decimal is 4352.

So we use the head command to slice and copy the first part of the binary file into prefix file. Then use the tail command to slice and copy the last part of the binary file into suffix file.



```
ix
[12/16/19]seed@VM:~/Desktop$
[12/16/19]seed@VM:~/Desktop$ gcc md5.c -o md5
[12/16/19]seed@VM:~/Desktop$ md5
1515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515156565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565656565659999999999999999
[12/16/19]seed@VM:~/Desktop$ head -c 4223 md5 > prefi
x
[12/16/19]seed@VM:~/Desktop$ tail -c +4353 md5 > suff
ix
[12/16/19]seed@VM:~/Desktop$ 
```

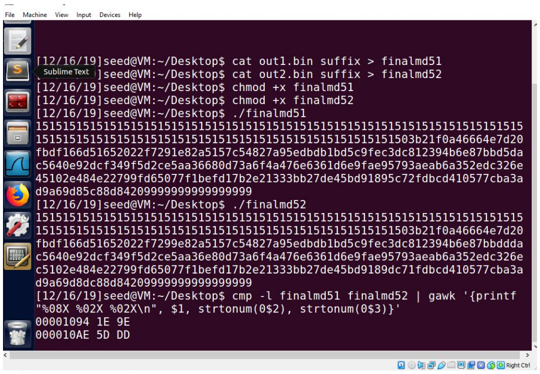
SeedLab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
11:14 AM
out1.bin
out2.bin
prefix.txt
T.txt
catfile
catfile2
md5
md5
prefix



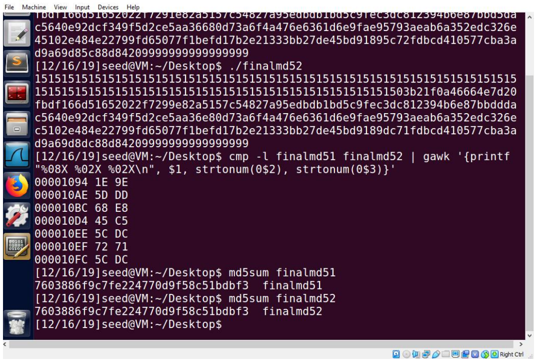
```
65656565656565656565656565659999999999999999
[12/16/19]seed@VM:~/Desktop$ head -c 4223 md5 > prefix
[12/16/19]seed@VM:~/Desktop$ tail -c +4353 md5 > suffix
[12/16/19]seed@VM:~/Desktop$ md5collgen -p prefix -o out1.bi
n out2.binMD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix'
Using initial value: 7fa5faa7e6fd19af42276b10281b614d

Generating first block: ..................................
.....
Generating second block: W.
Running time: 58.1546 s
[12/16/19]seed@VM:~/Desktop$ md5sum out1.bin
beba44c31ae73ee8030a6fe7c07e5ee1  out1.bin
[12/16/19]seed@VM:~/Desktop$ md5sum out2.bin
beba44c31ae73ee8030a6fe7c07e5ee1  out2.bin
[12/16/19]seed@VM:~/Desktop$ 
```

SeedLab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
11:37 AM

We ran md5collgen on prefix file and generated two binary files out1.bin and out2.bin with identical md5 hashes.



```
[12/16/19]seed@VM:~/Desktop$ cat out1.bin suffix > finalmd51
[12/16/19]seed@VM:~/Desktop$ cat out2.bin suffix > finalmd52
[12/16/19]seed@VM:~/Desktop$ chmod +x finalmd51
[12/16/19]seed@VM:~/Desktop$ chmod +x finalmd52
[12/16/19]seed@VM:~/Desktop$ ./finalmd51
1515151515151515151515151515151515151515151515151515151515151515151515151515
151515151515151515151515151515151515151515151515151503b21f0a46664e7d20
fbdf166d51652022f7291e82a5157c54827a95edbdb1bd5c9fec3dc812394b6e87bbd5da
c5640e92dcf349f5d2ce5aa36680d73a6f4a476e6361d6e9fae95793aeab6a352edc326e
45102e484e22799fd65077f1befd17b2e21333bb27de45bd91895c72fdbcd410577cba3a
d9a69d85c88d8420999999999999999999
[12/16/19]seed@VM:~/Desktop$ ./finalmd52
1515151515151515151515151515151515151515151515151515151515151515151515151515
151515151515151515151515151515151515151515151515151503b21f0a46664e7d20
fbdf166d51652022f7299e82a5157c54827a95edbdb1bd5c9fec3dc812394b6e87bbddda
c5640e92dcf349f5d2ce5aa36e80d73a6f4a476e6361d6e9fae95793aeab6a352edc326e
c5102e484e22799fd65077f1befd17b2e21333bb27de45bd9189dc71fdbcd410577cba3a
d9a69d8dc88d8420999999999999999999
[12/16/19]seed@VM:~/Desktop$ cmp -l finalmd51 finalmd52 | gawk '{printf
"%08X %02X %02X\n", $1, strtonum(0$2), strtonum(0$3)}'
00001094 1E 9E
000010AE 5D DD
```

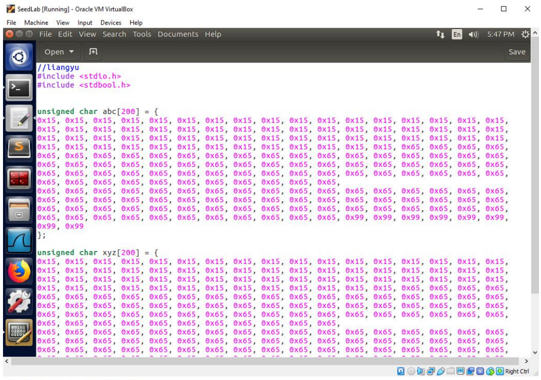


```
fbdf166d51652022f7291e82a5157c54827a95edbdb1bd5c9fec3dc812394b6e87bbd5da
c5640e92dcf349f5d2ce5aa36680d73a6f4a476e6361d6e9fae95793aeab6a352edc326e
45102e484e22799fd65077f1befd17b2e21333bb27de45bd91895c72fdbcd410577cba3a
d9a69d85c88d8420999999999999999999
[12/16/19]seed@VM:~/Desktop$ ./finalmd52
1515151515151515151515151515151515151515151515151515151515151515151515151515
151515151515151515151515151515151515151515151515151503b21f0a46664e7d20
fbdf166d51652022f7299e82a5157c54827a95edbdb1bd5c9fec3dc812394b6e87bbddda
c5640e92dcf349f5d2ce5aa36e80d73a6f4a476e6361d6e9fae95793aeab6a352edc326e
c5102e484e22799fd65077f1befd17b2e21333bb27de45bd9189dc71fdbcd410577cba3a
d9a69d8dc88d8420999999999999999999
[12/16/19]seed@VM:~/Desktop$ cmp -l finalmd51 finalmd52 | gawk '{printf
"%08X %02X %02X\n", $1, strtonum(0$2), strtonum(0$3)}'
00001094 1E 9E
000010AE 5D DD
000010BC 68 E8
000010D4 45 C5
000010EE 5C DC
000010EF 72 71
000010FC 5C DC
[12/16/19]seed@VM:~/Desktop$ md5sum finalmd51
7603886f9c7fe224770d9f58c51bdbf3  finalmd51
[12/16/19]seed@VM:~/Desktop$ md5sum finalmd52
7603886f9c7fe224770d9f58c51bdbf3  finalmd52
[12/16/19]seed@VM:~/Desktop$
```

We then concatenate out1.bin and out2.bin to the suffix file, producing two executable files that have the same md5 hashes but produce different outputs.

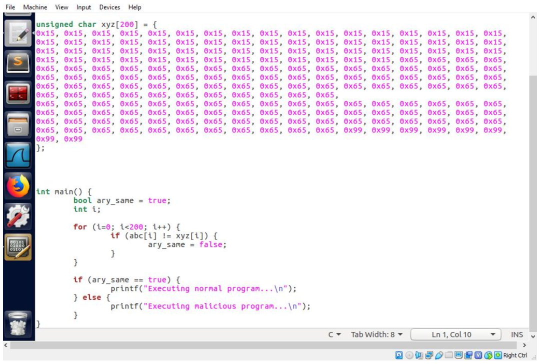
Task 4: Making the Two Programs Behave Differently

We create a C program called md5bad as follows:



SeedLab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Edit View Search Tools Documents Help
En
5:47 PM
Open
Save
Right Ctrl

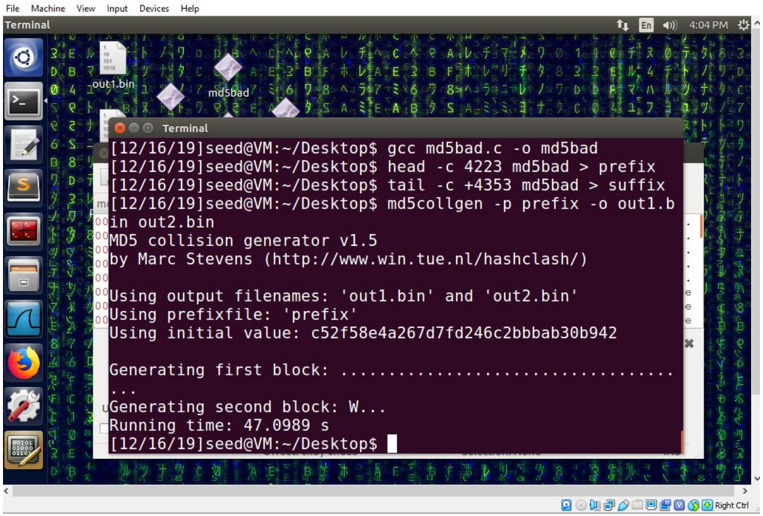
```
//liangyu
#include <stdio.h>
#include <stdbool.h>


unsigned char abc[200] = {
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99,
0x99, 0x99
};

unsigned char xyz[200] = {
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
```



```
unsigned char xyz[200] = {
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15,
0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x15, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65,
0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x65, 0x99, 0x99, 0x99, 0x99, 0x99, 0x99,
0x99, 0x99
};



int main() {
        bool ary_same = true;
        int i;

        for (i=0; i<200; i++) {
                if (abc[i] != xyz[i]) {
                        ary_same = false;
                }
        }

        if (ary_same == true) {
                printf("Executing normal program...\n");
        } else {
                printf("Executing malicious program...\n");
        }
}
```

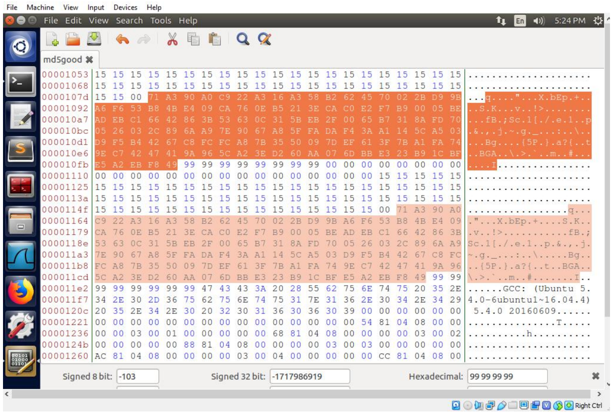
Using Bless, we can see that the two arrays' contents are stored in the compiled executable file from offset 0x1040 to 0x1107 and from 0x1120 to 0x11e7.   The prefix section of the first array is stored at offset 0x1040 to 0x107f (4223 in decimal). The suffix section of the first array is stored at offset 0x1100 (4352 in decimal).

We use the head command to slice and copy the first part of the compiled executable file into prefix file. Then use the tail command to slice and copy the last part of the file into suffix file. Then run md5collgen on prefix file to generate out1.bin and out2.bin.



```
[12/16/19]seed@VM:~/Desktop$ gcc md5bad.c -o md5bad
[12/16/19]seed@VM:~/Desktop$ head -c 4223 md5bad > prefix
[12/16/19]seed@VM:~/Desktop$ tail -c +4353 md5bad > suffix
[12/16/19]seed@VM:~/Desktop$ md5collgen -p prefix -o out1.b
in 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'
Using initial value: c52f58e4a267d7fd246c2bbbab30b942

Generating first block: .................................
...
Generating second block: W...
Running time: 47.0989 s
[12/16/19]seed@VM:~/Desktop$
```

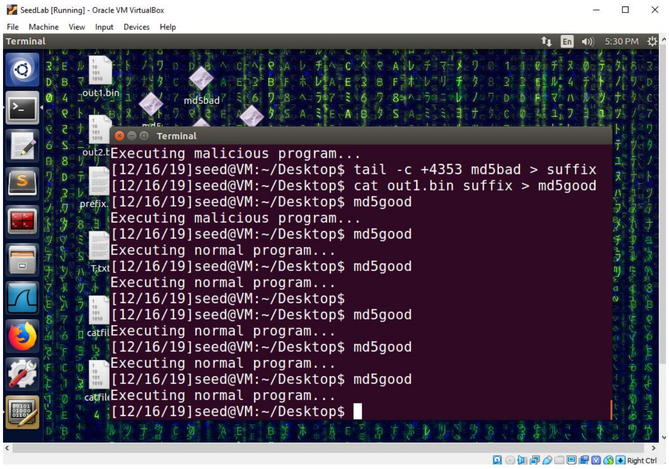
We use command “cat out1.bin suffix > md5good” to generate a file called md5good.
We copy and paste the section generated by md5collgen to the content of the second array.


File Machine View Input Devices Help
File Edit View Search Tools Help
5:24 PM
md5good
Right Ctrl

```
00001053 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 .....................
00001068 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 .....................
0000107d 15 15 00 71 A3 90 A0 C9 22 A3 16 A3 58 B2 62 45 70 02 2B D9 9B ...q...."...X.bEp.+..
00001092 A6 F6 53 B8 4B E4 09 CA 76 0E B5 21 3E CA C0 E2 F7 B9 00 05 BE ..S.K...v..!>........
000010a7 AD EB C1 66 42 86 3B 53 63 0C 31 5B EB 2F 00 65 B7 31 8A FD 70 ...fB.;Sc.1[./.e.1..p
000010bc 05 26 03 2C 89 6A A9 7E 90 67 A8 5F FA DA F4 3A A1 14 5C A5 03 .&.,.j.~.g._...:..\..
000010d1 D9 F5 B4 42 67 C8 FC FC A8 7B 35 50 09 7D EF 61 3F 7B A1 FA 74 ...Bg....{5P.}.a?{..t
000010e6 9E C7 42 47 41 9A 96 5C A2 3E D2 60 AA 07 6D BB E3 23 B9 1C BF ..BGA..\.>.`..m..#...
000010fb E5 A2 EB F8 49 99 99 99 99 99 99 99 99 00 00 00 00 00 00 00 00 ....I................
00001110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 15 15 15 15 15 .....................
00001125 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 .....................
0000113a 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 .....................
0000114f 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 00 71 A3 90 A0 .................q...
00001164 C9 22 A3 16 A3 58 B2 62 45 70 02 2B D9 9B A6 F6 53 B8 4B E4 09 ."...X.bEp.+....S.K..
00001179 CA 76 0E B5 21 3E CA C0 E2 F7 B9 00 05 BE AD EB C1 66 42 86 3B .v..!>...........fB.;
0000118e 53 63 0C 31 5B EB 2F 00 65 B7 31 8A FD 70 05 26 03 2C 89 6A A9 Sc.1[./.e.1..p.&.,.j.
000011a3 7E 90 67 A8 5F FA DA F4 3A A1 14 5C A5 03 D9 F5 B4 42 67 C8 FC ~.g._...:..\.....Bg..
000011b8 FC A8 7B 35 50 09 7D EF 61 3F 7B A1 FA 74 9E C7 42 47 41 9A 96 ..{5P.}.a?{..t..BGA..
000011cd 5C A2 3E D2 60 AA 07 6D BB E3 23 B9 1C BF E5 A2 EB F8 49 99 99 \.>.`..m..#.......I..
000011e2 99 99 99 99 99 99 47 43 43 3A 20 28 55 62 75 6E 74 75 20 35 2E ......GCC: (Ubuntu 5.
000011f7 34 2E 30 2D 36 75 62 75 6E 74 75 31 7E 31 36 2E 30 34 2E 34 29 4.0-6ubuntu1~16.04.4)
0000120c 20 35 2E 34 2E 30 20 32 30 31 36 30 36 30 39 00 00 00 00 00 00  5.4.0 20160609......
00001221 00 00 00 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 03 00 02 ..........h..........
0000124b 00 00 00 00 00 88 81 04 08 00 00 00 00 03 00 03 00 00 00 00 00 .....................
00001260 AC 81 04 08 00 00 00 00 03 00 04 00 00 00 00 00 CC 81 04 08 00 .....................
```

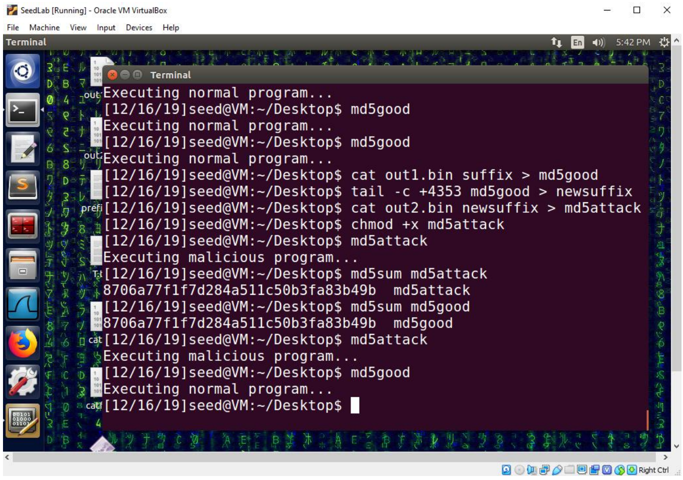
Signed 8 bit: -103 Signed 32 bit: -1717986919 Hexadecimal: 99 99 99 99

If we now run md5good program, it executes benign code.


SeedLab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
5:30 PM
out1.bin
md5bad
prefix
Right Ctrl

```
Executing malicious program...
[12/16/19]seed@VM:~/Desktop$ tail -c +4353 md5bad > suffix
[12/16/19]seed@VM:~/Desktop$ cat out1.bin suffix > md5good
[12/16/19]seed@VM:~/Desktop$ md5good
Executing malicious program...
[12/16/19]seed@VM:~/Desktop$ md5good
Executing normal program...
[12/16/19]seed@VM:~/Desktop$ md5good
Executing normal program...
[12/16/19]seed@VM:~/Desktop$
[12/16/19]seed@VM:~/Desktop$ md5good
Executing normal program...
[12/16/19]seed@VM:~/Desktop$ md5good
Executing normal program...
[12/16/19]seed@VM:~/Desktop$ md5good
Executing normal program...
[12/16/19]seed@VM:~/Desktop$ 
```

Now we use the tail command to slice the md5good program to create a new suffix. Then use the cat command to concatenate the new suffix to the out2.bin.


SeedLab [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
5:42 PM
Right Ctrl

```
Executing normal program...
[12/16/19]seed@VM:~/Desktop$ md5good
Executing normal program...
[12/16/19]seed@VM:~/Desktop$ md5good
Executing normal program...
[12/16/19]seed@VM:~/Desktop$ cat out1.bin suffix > md5good
[12/16/19]seed@VM:~/Desktop$ tail -c +4353 md5good > newsuffix
[12/16/19]seed@VM:~/Desktop$ cat out2.bin newsuffix > md5attack
[12/16/19]seed@VM:~/Desktop$ chmod +x md5attack
[12/16/19]seed@VM:~/Desktop$ md5attack
Executing malicious program...
[12/16/19]seed@VM:~/Desktop$ md5sum md5attack
8706a77f1f7d284a511c50b3fa83b49b  md5attack
[12/16/19]seed@VM:~/Desktop$ md5sum md5good
8706a77f1f7d284a511c50b3fa83b49b  md5good
[12/16/19]seed@VM:~/Desktop$ md5attack
Executing malicious program...
[12/16/19]seed@VM:~/Desktop$ md5good
Executing normal program...
[12/16/19]seed@VM:~/Desktop$ 
```

Now we have two programs that behave differently but have the exact same md5 hash.