

MD5 Collision Attack Lab

I have setup my Lab in a docker container using the ubuntu 20.04 image from docker hub.

I have downloaded the `md5collgen` tool and set it up. Since I couldn't get a way to install the `bless` editor in the container, I have decided to use `xxd` to get the hex dump data from the binary file.

Lab Tasks

3.1 Generating Two Different Files with the Same MD5 Hash

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

Answer:

Prefix Size = 48 bytes

```

Generating first block: .....
.
.
.
Generating second block: S00....
Running time: 36.7736 s
root@25236d8c6337:~/Labsetup# diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
root@25236d8c6337:~/Labsetup# xxd out1.bin
00000000: 4164 6469 6e67 2063 6f6e 7465 6e74 2074 Adding content t
00000010: 6f20 7468 6973 2066 696c 6520 756e 7469 o this file until it is less op.
00000020: 6c20 6974 2069 7320 6c65 7373 206f 700a l it is less op.
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ..... ←
00000040: 0b21 ec49 2ac9 5535 b1f5 7d04 ae28 2220 .!.I*.U5..}..("
00000050: fa38 9699 c544 e3e2 401e 3f7e 4f3a a464 .8...D..@.?~0:.d
00000060: 2cfc afcd d00c eb07 8f02 d412 1128 c104 ,.....
00000070: 7f74 41d3 cbce 9ed7 c5e7 4387 df22 94d8 .tA.....C.."..
00000080: a755 2732 6545 457a 9fec bcdf 893f 0f8b .U'2eEEz.....?..
00000090: 1809 f55b eeb5 a058 c1f7 f7c4 d118 592e ...[...X.....Y.
000000a0: b96a 1ef7 2b17 af26 c5ad c1af 5416 51a8 .j...+...&....T.Q.
000000b0: c306 97e3 1876 30a8 4cdc 565e d841 1c42 .....v0.L.V^A.B
root@25236d8c6337:~/Labsetup# xxd out2.bin
00000000: 4164 6469 6e67 2063 6f6e 7465 6e74 2074 Adding content t
00000010: 6f20 7468 6973 2066 696c 6520 756e 7469 o this file until it is less op.
00000020: 6c20 6974 2069 7320 6c65 7373 206f 700a l it is less op. ←
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 ..... ←
00000040: 0b21 ec49 2ac9 5535 b1f5 7d04 ae28 2220 .!.I*.U5..}..("
00000050: fa38 9619 c544 e3e2 401e 3f7e 4f3a a464 .8...D..@.?~0:.d
00000060: 2cfc afcd d00c eb07 8f02 d412 11a8 c104 ,.....
00000070: 7f74 41d3 cbce 9ed7 c5e7 4307 df22 94d8 .tA.....C.."..
00000080: a755 2732 6545 457a 9fec bcdf 893f 0f8b .U'2eEEz.....?..
00000090: 1809 f5db eeb5 a058 c1f7 f7c4 d118 592e .....X.....Y.
000000a0: b96a 1ef7 2b17 af26 c5ad c1af 5496 50a8 .j...+...&....T.P.
000000b0: c306 97e3 1876 30a8 4cdc 56de d841 1c42 .....v0.L.V..A.B
root@25236d8c6337:~/Labsetup# S

```

Every 2.0s: stat prefix.txt

25236d8c6337: Thu Feb 20 11:18:42 20

```

File: prefix.txt
Size: 48          Blocks: 8          IO Block: 4096   regular file
Device: 0,68      Inode: 5001459      Links: 1
Access: (0644/-rw-r--r--)  Uid: (     0/    root)  Gid: (     0/    root)
Access: 2025-02-20 11:14:33.447552598 -0500
Modify: 2025-02-20 11:13:53.243826775 -0500
Change: 2025-02-20 11:13:53.243826775 -0500
Birth: 2025-02-20 11:13:53.243826775 -0500

```

Here, my prefix is less than 64 bytes(not multiple), I see the out1.bin and out2.bin files are mismatching. The `m5collgen` tool adds padding to the prefix to make it 64 bytes to ensure the input is properly formatted.

- 16 bytes of padding were added. The padding bytes are just Zeroes
- The padding could add unintended differences in the output files, which might affect the collision.
- This is why the generated binary files were not perfectly identical in structure.

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

Answer:

```
root@25236d8c6337:~/Labsetup# ./md5collgen -p prefix.txt -o out1.bin out2.bin
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: 10855ae3ebcd22fda7d21391e874e68b

Generating first block: .....
Generating second block: S01.....
Running time: 5.35488 s
root@25236d8c6337:~/Labsetup# xxd out1.bin
00000000: 4164 6469 6e67 2063 6f6e 7465 6e74 2074 Adding content t
00000010: 6f20 7468 6973 2066 696c 6520 756e 7469 o this file until it is 64 bytes
00000020: 6c20 6974 2069 7320 3634 2062 7974 6573 in total size..
00000030: 2069 6e20 746f 7461 6c20 7369 7a65 2e0a ...=\...m4...G
00000040: c303 ac83 3d5c 8ca0 d6d6 6d34 bb8b 0147 .*n0..5.U.B...F.
00000050: d52a 6e30 82ef 35f3 55f6 429e ad15 46cc ...|...K..;...4
00000060: d013 fbcf 7c02 ca99 4bfa c23b 03ca f034 ..o..C.....
00000070: 05ba 6fc3 1c43 a5df a6ab bc93 1d1a b300 .....Z.....
00000080: 270f febf 9863 b98e 2b51 ef64 c978 b972 '....c..+Q.d.x.r
00000090: 0efc 1f65 ffa5 2ee9 d35a 06b9 ecc6 01e6 ...e.....Z.....
000000a0: fe65 a6d6 f4cf 034d 6884 5037 a8e8 c268 .e.....Mh.P7...h
000000b0: 3223 0a6e 0b1c cbc1 7c07 ce34 57ec a0da 2#.n....|..4W...
root@25236d8c6337:~/Labsetup# xxd out2.bin
00000000: 4164 6469 6e67 2063 6f6e 7465 6e74 2074 Adding content t
00000010: 6f20 7468 6973 2066 696c 6520 756e 7469 o this file until it is 64 bytes
00000020: 6c20 6974 2069 7320 3634 2062 7974 6573 in total size..
00000030: 2069 6e20 746f 7461 6c20 7369 7a65 2e0a ...=\...m4...G
00000040: c303 ac83 3d5c 8ca0 d6d6 6d34 bb8b 0147 .*n...5.U.B...F.
00000050: d52a 6eb0 82ef 35f3 55f6 429e ad15 46cc ...|...K..;J.4
00000060: d013 fbcf 7c02 ca99 4bfa c23b 034a f134 ..o..C.....
00000070: 05ba 6fc3 1c43 a5df a6ab bc13 1d1a b300 .....Z.....
00000080: 270f febf 9863 b98e 2b51 ef64 c978 b972 '....c..+Q.d.x.r
00000090: 0efc 1fe5 ffa5 2ee9 d35a 06b9 ecc6 01e6 ...e.....Z.....
000000a0: fe65 a6d6 f4cf 034d 6884 5037 a868 c268 .e.....Mh.P7.h.h
000000b0: 3223 0a6e 0b1c cbc1 7c07 ceb4 57ec a0da 2#.n....|...W...
root@25236d8c6337:~/Labsetup#
```

Every 2.0s: stat prefix.txt

```
File: prefix.txt
Size: 64          Blocks: 8          IO Block: 4096   regular file
Device: 0,68      Inode: 5001459     Links: 1
```

Since the prefix file is now 64 bytes, the tool didn't need to add any padding. All it did is, it added P to out1.bin and Q to out2.bin.

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:

The 128 bytes generated by `md5collgen` are completely different for both the cases. Whether the prefix is padded to get 64 bytes or not, the P and Q are different but they do have some similarities.

here are the differences b/w `out1.bin` and `out2.bin`

With padding file:

```
Generating first block: .....  
.....  
Generating second block: S00....  
Running time: 36.7736 s  
root@25236d8c6337:~/Labsetup# diff out1.bin out2.bin  
Binary files out1.bin and out2.bin differ  
root@25236d8c6337:~/Labsetup# xxd out1.bin  
00000000: 4164 6469 6e67 2063 6f6e 7465 6e74 2074 Adding content t  
00000010: 6f20 7468 6973 2066 696c 6520 756e 7469 o this file until it is less op.  
00000020: 6c20 6974 2069 7320 6c65 7373 206f 700a  
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....  
00000040: 0b21 ec49 2ac9 5535 b1f5 7d04 ae28 2220 .!.I*.U5..}..("...  
00000050: fa38 9699 c544 e3e2 401e 3f7e 4f3a a464 .8...D..@.?~0:d  
00000060: 2fcf afcd d00c eb07 8f02 d412 1128 c104 ,.....(C  
00000070: 7f74 41d3 cbce 9ed7 c5e7 4387 df22 94d8 .tA.....C..."..  
00000080: a755 2732 6545 457a 9fec bcdf 893f 0f8b .U'2eEEz.....?..  
00000090: 1809 f55b eeb5 a058 c1f7 f7c4 d118 592e ...[...X.....Y..  
000000a0: b96a 1ef7 2b17 af26 c5ad c1af 5416 51a8 .j...+...&...T.Q..  
000000b0: c306 97e3 1876 30a8 4cdc 565e d841 1c42 .....v0.L.V^A.B  
root@25236d8c6337:~/Labsetup# xxd out2.bin  
00000000: 4164 6469 6e67 2063 6f6e 7465 6e74 2074 Adding content t  
00000010: 6f20 7468 6973 2066 696c 6520 756e 7469 o this file until it is less op.  
00000020: 6c20 6974 2069 7320 6c65 7373 206f 700a  
00000030: 0000 0000 0000 0000 0000 0000 0000 0000 .....  
00000040: 0b21 ec49 2ac9 5535 b1f5 7d04 ae28 2220 .!.I*.U5..}..("...  
00000050: fa38 9619 c544 e3e2 401e 3f7e 4f3a a464 .8...D..@.?~0:d  
00000060: 2fcf afcd d00c eb07 8f02 d412 11a8 c104 ,.....(C  
00000070: 7f74 41d3 cbce 9ed7 c5e7 4307 df22 94d8 .tA.....C..."..  
00000080: a755 2732 6545 457a 9fec bcdf 893f 0f8b .U'2eEEz.....?..  
00000090: 1809 f5db eeb5 a058 c1f7 f7c4 d118 592e ...O...X.....Y..  
000000a0: b96a 1ef7 2b17 af26 c5ad c1af 5496 50a8 .j...+...&...T.P..  
000000b0: c306 97e3 1876 30a8 4cdc 56de d841 1c42 .....v0.L.V^A.B  
root@25236d8c6337:~/Labsetup# S
```

Without Padding:

```
root@25236d8c6337:~/Labsetup# xxd out1.bin
00000000: 4164 6469 6e67 2063 6f6e 7465 6e74 2074 Adding content t
00000010: 6f20 7468 6973 2066 696c 6520 756e 7469 o this file until it is 64 bytes
00000020: 6c20 6974 2069 7320 3634 2062 7974 6573 in total size..
00000030: 2069 6e20 746f 7461 6c20 7369 7a65 2e0a ...=\....m4...G
00000040: c303 ac83 3d5c 8ca0 d6d6 6d34 bb8b 0147 .*n0..5.U.B...F.
00000050: d52a 6e30 82ef 35f3 55f6 429e ad15 46cc ...|...K..;...4
00000060: d013 fbcf 7c02 ca99 4bfa c23b 03ca f034 ...e....Z....
00000070: 05ba 6fc3 1c43 a5df a6ab bc93 1d1a b300 ..o..C.....
00000080: 270f febf 9863 b98e 2b51 ef64 c978 b972 '....c..+Q.d.x.r
00000090: 0efc 1f65 ffa5 2ee9 d35a 06b9 ecc6 01e6 ...e....Z....
000000a0: fe65 a6d6 f4cf 034d 6884 5037 a8e8 c268 .e.....Mh.P7...h
000000b0: 3223 0a6e 0b1c cbc1 7c07 ce34 57ec a0da 2#.n....|...4W...
root@25236d8c6337:~/Labsetup# xxd out2.bin
00000000: 4164 6469 6e67 2063 6f6e 7465 6e74 2074 Adding content t
00000010: 6f20 7468 6973 2066 696c 6520 756e 7469 o this file until it is 64 bytes
00000020: 6c20 6974 2069 7320 3634 2062 7974 6573 in total size..
00000030: 2069 6e20 746f 7461 6c20 7369 7a65 2e0a ...=\....m4...G
00000040: c303 ac83 3d5c 8ca0 d6d6 6d34 bb8b 0147 .*n...5.U.B...F.
00000050: d52a 6eb0 82ef 35f3 55f6 429e ad15 46cc ...|...K..;J.4
00000060: d013 fbcf 7c02 ca99 4bfa c23b 034a f134 ...e....Z....
00000070: 05ba 6fc3 1c43 a5df a6ab bc13 1d1a b300 ..o..C.....
00000080: 270f febf 9863 b98e 2b51 ef64 c978 b972 '....c..+Q.d.x.r
00000090: 0efc 1fe5 ffa5 2ee9 d35a 06b9 ecc6 01e6 .....Z....
000000a0: fe65 a6d6 f4cf 034d 6884 5037 a868 c268 .e.....Mh.P7.h.h
000000b0: 3223 0a6e 0b1c cbc1 7c07 ceb4 57ec a0da 2#.n....|...W...
root@25236d8c6337:~/Labsetup#
```

I can see a pattern here. The P and Q even though they are random strings, they are almost same.

Only bytes 47, 84, 110, 124 differ from out1.bin and out2.bin.

3.2 Task 2: Understanding MD5's Property

For out1.bin and out2.bin, I ran the `md5sum` command and the below is the output:

```
root@25236d8c6337:~/Labsetup# md5sum out1.bin out2.bin
f9b824b94fc51c819a69cc91df6f7acf  out1.bin
f9b824b94fc51c819a69cc91df6f7acf  out2.bin
```

Created a new suffix file called `suffix.bin`:

```
root@25236d8c6337:~/Labsetup# echo "new suffix data to binary" > suffix.bin
root@25236d8c6337:~/Labsetup# cat suffix.bin
new suffix data to binary
root@25236d8c6337:~/Labsetup# xxd suffix.bin
00000000: 6e65 7720 7375 6666 6978 2064 6174 6120  new suffix data
00000010: 746f 2062 696e 6172 790a          to binary.
```

I have concatenated the `out1.bin` with `suffix.bin` file and the same for `out2.bin` file.

```
root@25236d8c6337:~/Labsetup# cat out1.bin suffix.bin > out1_SUFFIX.bin
root@25236d8c6337:~/Labsetup# cat out2.bin suffix.bin > out2_SUFFIX.bin
root@25236d8c6337:~/Labsetup# md5sum out1_SUFFIX.bin out2_SUFFIX.bin
720b3e90a395d5a4c90d073c69de9878  out1_SUFFIX.bin
720b3e90a395d5a4c90d073c69de9878  out2_SUFFIX.bin
root@25236d8c6337:~/Labsetup#
```

`MD5(out1.bin) = MD5(out2.bin)`

and `MD5(out1.bin I suffix.bin) = MD5(out2.bin I suffix.bin)`

where `I` represents concatenation.

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

MyCode:

```
#include <stdio.h>

unsigned char xyz[200] = {
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
    0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
```

```

0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
};

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

```

Compile it:

```
gcc code.c
```

Check the hex output using `xxd`

```
xxd ./a.out
```

```

00002fd0: 4010 0000 0000 0000 0000 0000 0000 0000 @.....
00002fe0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00002ff0: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00003000: 0000 0000 0000 0000 0840 0000 0000 0000 .....@...
00003010: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00003020: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA
00003030: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA
00003040: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA
00003050: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA
00003060: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA
00003070: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA
00003080: 4141 4141 4242 4242 4242 4242 4242 4242 AAAABBBBBB
00003090: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB BBBB
000030a0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB BBBB
000030b0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB BBBB
000030c0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB BBBB
000030d0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB BBBB
000030e0: 4242 4242 4242 4242 4743 433a 2028 5562 BBBB BBBGCC: (Ub
000030f0: 756e 7475 2031 332e 332e 302d 3675 6275 unto 13.3.0-6ubu
00003100: 6e74 7532 7e32 342e 3034 2920 3133 2e33 ntu2~24.04) 13.3
00003110: 2e30 0000 0000 0000 0000 0000 0000 0000 .0.....
00003120: 0000 0000 0000 0000 0000 0000 0000 0000 .....
00003130: 0100 0000 0400 f1ff 0000 0000 0000 0000 .....0400 ...
00003140: 0000 0000 0000 0000 0900 0000 0100 0400 .....

```

Now, the letters A start at 3020, but I wanted to split it at half of the char array(at 3084 or 12420 in decimal) so I get all A's in the prefix.

```
head -c 12420 a.out > prefix
```

Generating the out1.bin and out2.bin

```
./md5collgen -p prefix -o out1.bin out2.bin
```

	Out1.bin	Out2.bin
	<pre>00002fd0: 4e10 0000 0000 0000 0000 0000 0000 0000 .@. 00002ff0: 0000 0000 0000 0000 0000 0000 0000 0000 . 00003000: 0000 0000 0000 0000 0000 0000 0000 0000 . 00003000: 0000 0000 0000 0000 0040 0000 0000 0000 . 00003010: 0000 0000 0000 0000 0000 0000 0000 0000 . 00003010: 0000 0000 0000 0000 0000 0000 0000 0000 . 00003020: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003030: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003040: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003050: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003060: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003070: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003080: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003090: 0000 0000 0000 0000 0000 0000 0000 0000 AAAA.....AA 00003090: 0000 0000 0000 0000 0000 0000 0000 0000 AAAA.....AA 000030a0: 0000 0000 0000 0000 0000 0000 0000 0000 . 000030b0: 0000 0000 0000 0000 0000 0000 0000 0000 . 000030c0: 3c9f f5f4 1a56 a31a 7095 fde0 d60d 464d <....V.p....FM 000030d0: c2a6 85fa 7d5b ddc3 399e 42b9 29ba 25b5 ...}[.9.B.)%. 000030e0: l1c4 bee7 38b3 c204 6702 5814 de29 692b ..8.g.X.)!+ 000030f0: e885 28c3 b9ae 2e02 d680 51e2 4d66 fcb7 ..(.....Q.M'.. 00003100: fcbe a695 9372 376c e9f3 b65c 7a7f 040c ..^Tl.....Z. 00003110: c583 eb5e 3cd4 d1cf abb5 4077 2a60 a8b7 ..<.....@7*.. 00003120: d792 f803 af05 621e fb3e 8c52 fc87 1ebf ..b.>R. 00003130: ae98 9345 895d ad47 0bbf e0b6 0ff6 ba7e ..E.]G....."</pre>	<pre>00002fd0: 0000 0000 0000 0000 0000 0000 0000 0000 .@. 00003000: 0000 0000 0000 0000 0000 0000 0000 0000 . 00003010: 0000 0000 0000 0000 0000 0000 0000 0000 . 00003020: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003030: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003040: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003050: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003060: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003070: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003080: 4141 4141 4141 4141 4141 4141 4141 4141 AAAAAAAA.....AA 00003090: 0000 0000 0000 0000 0000 0000 0000 0000 AAAA.....AA 00003090: 0000 0000 0000 0000 0000 0000 0000 0000 AAAA.....AA 000030a0: 0000 0000 0000 0000 0000 0000 0000 0000 . 000030b0: 0000 0000 0000 0000 0000 0000 0000 0000 . 000030c0: 3c9f f5f4 1a56 a31a 7095 fde0 d60d 464d <....V.p....FM 000030d0: c2a6 85fa 7d5b ddc3 399e 42b9 29ba 25b5 ...z}[.9.B.)%. 000030e0: l1c4 bee7 38b3 c204 6702 5814 dea9 692b ..8.g.X.)i+ 000030f0: e885 28c3 b9ae 2e02 d680 51e2 4d66 fcb7 ..(.....Q.M'.. 00003100: fcbe a695 9372 376c e9f3 b65c 7a7f 040c ..r7L.....Z. 00003110: c583 eb5e 3cd4 d1cf abb5 4077 2a60 a8b7 ..<.....@7*.. 00003120: d792 f803 af05 621e fb3e 8c52 fc07 1ebf ..b.>R. 00003130: ae98 9345 895d ad47 0bbf e0b6 0ff6 ba7e ..E.]G....."</pre>

Adding the remaining text to suffix (from hex 3085 or 12421)

```
tail -c +12421 a.out > suffix
```

```
root@25236d8c6337:~/Labsetup/3.3# tail -c +12421 a.out > suffix
root@25236d8c6337:~/Labsetup/3.3# xxd suffix | head
00000000: 4242 4242 4242 4242 4242 4242 4242 4242 4242 BBBBBBBBBBBBBBBBBB
00000010: 4242 4242 4242 4242 4242 4242 4242 4242 4242 BBBBBBBBBBBBBBBBBB
00000020: 4242 4242 4242 4242 4242 4242 4242 4242 4242 BBBBBBBBBBBBBBBBBB
00000030: 4242 4242 4242 4242 4242 4242 4242 4242 4242 BBBBBBBBBBBBBBBBBB
00000040: 4242 4242 4242 4242 4242 4242 4242 4242 4242 BBBBBBBBBBBBBBBBBB
00000050: 4242 4242 4242 4242 4242 4242 4242 4242 4242 BBBBBBBBBBBBBBBBBB
00000060: 4242 4242 4743 433a 2028 5562 756e 7475 BBBBBGCC: (Ubuntu
00000070: 2031 332e 332e 302d 3675 6275 6e74 7532 13.3.0-6ubuntu2
00000080: 7e32 342e 3034 2920 3133 2e33 2e30 0000 ~24.04) 13.3.0..
00000090: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
root@25236d8c6337:~/Labsetup/3.3#
```

Add the suffix to out1 and out2 to create two files using `cat` command. Also provide the executable permissions and check the output of the files and their md5sum.

The hashes match and we get the same output for both `program1` and `program2` files.

3.4 Task 4: Making the Two Programs Behave Differently

The below code is used for creating two programs that behave differently.

compile the code:

```
gcc code.c
```

```
00003010: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
00003020: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003030: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003040: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003050: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003060: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003070: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003080: 4141 4141 4242 4242 4242 4242 4242 4242 AAAABBBBBBBBBB
00003090: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
000030a0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
000030b0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
000030c0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
000030d0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
000030e0: 4242 4242 4242 4242 0000 0000 0000 0000 BBBB.....
000030f0: 0000 0000 0000 0000 0000 0000 0000 0000 ..... .
00003100: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003110: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003120: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003130: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003140: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003150: 4141 4141 4141 4141 4141 4141 4141 4141 AAAA.....AAAAAA
00003160: 4141 4141 4242 4242 4242 4242 4242 4242 AAAABBBBBBBB
00003170: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
00003180: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
00003190: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
000031a0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
000031b0: 4242 4242 4242 4242 4242 4242 4242 4242 BBBB.....BBBBB
000031c0: 4242 4242 4242 4242 4743 433a 2028 5562 BBBB.....GCC: (Ub
000031d0: 756e 7475 2031 332e 332e 302d 3675 6275 untu 13.3.0-6ubu
000031e0: 6e74 7532 7e32 342e 3034 2920 3133 2e33 ntu2~24.04) 13.3
000031f0: 2e30 0000 0000 0000 0000 0000 0000 0000 .0.....
```

We can observe the hex values for first array starting at 3020 and so I will extract the data before this to prefix.

```
head -c 12420 a.out > prefix
```

Using `md5collgen` command to create 2 binary files with the same hash

```
./md5collgen -p prefix -o P_prefix Q_prefix
```

```
root@25236d8c6337:~/Labsetup/3.4# md5sum P_prefix Q_prefix
da89403e832f806d701db7f8d62382f3  P_prefix
da89403e832f806d701db7f8d62382f3  Q_prefix
root@25236d8c6337:~/Labsetup/3.4# █
```

Now, we need to connect the prefix files to correct suffix files.

Upon observation of the prefix files, there is 188 bytes of difference b/w P_prefix and prefix file and the same for Q_prefix.

```
root@25236d8c6337:~/Labsetup/3.4# ls -l *pref*
-rw-r--r-- 1 root root 12608 Feb 21 22:11 P_prefix
-rw-r--r-- 1 root root 12608 Feb 21 22:11 Q_prefix
-rw-r--r-- 1 root root 12420 Feb 21 22:08 prefix
root@25236d8c6337:~/Labsetup/3.4# █
```

So, the suffix file is everything after the first 12421 bytes from the a.out file.

Create suffix:

```
tail -c +12609 a.out > suffix
```

To get P file, we know 188 bytes are generated extra by `md5collgen`

```
tail -c 188 P_prefix > P
tail -c 188 Q_prefix > Q
```

Checking suffix:

```
root@25236d8c6337:~/Labsetup/3.4# xxd suffix
00000000: 4242 4242 4242 4242 4242 4242 4242 4242  BBBBBBBBBBBBBBBB
00000010: 4242 4242 4242 4242 4242 4242 4242 4242  BBBBBBBBBBBBBBBB
00000020: 4242 4242 4242 4242 4242 4242 4242 4242  BBBBBBBBBBBBBBBB
00000030: 4242 4242 4242 4242 4242 4242 4242 4242  BBBBBBBBBBBBBBBB
00000040: 4242 4242 4242 4242 4242 4242 4242 4242  BBBBBBBBBBBBBBBB
00000050: 4242 4242 4242 4242 4242 4242 4242 4242  BBBBBBBBBBBBBBBB
00000060: 4242 4242 0000 0000 0000 0000 0000 0000  BBBB.....
00000070: 0000 0000 0000 0000 0000 0000 4141 4141  .....AAAA
00000080: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAA
00000090: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAA
000000a0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAA
000000b0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAA
000000c0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAA
000000d0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAA
000000e0: 4242 4242 4242 4242 4242 4242 4242 4242  BBBBBBBBBBBBBB
000000f0: 4242 4242 4242 4242 4242 4242 4242 4242  BBBBBBBBBBBBBB
00000100: 4242 4242 4242 4242 4242 4242 4242 4242  BBBBBBBBBBBBBB
00000110: 4242 4242 4242 4242 4242 4242 4242 4242  RRRRRRRRRRRRRR
```

Now, we observe second array starts at 7c(124). I'm adding extra 100 bytes to get middle of second array at e0.

```
head -c 150 suffix > middle
```

So, everything after `middle` file will be our `commonsuffix` file. Since `P` is 188 bytes and `middle` is about 174 bytes, the `commonsuffix` file would be everything in the `suffix` file after 412 (188+154) bytes.

```
tail -c 339 suffix > commonsuffix
```

At final, we will concatenate everything together:

```
cat P_prefix middle P commonsuffix > benignCode
cat Q_prefix middle Q commonsuffix > maliciousCode
```

```
root@25236d8c6337:~/Labsetup/3.4# ls -l a.out *Code
-rwxr-xr-x 1 root root 16456 Feb 21 23:53 a.out
-rw-r--r-- 1 root root 13285 Feb 21 23:55 benignCode
-rw-r--r-- 1 root root 13285 Feb 21 23:55 maliciousCode
root@25236d8c6337:~/Labsetup/3.4# █
```

Verifying MD5 Hash:

```
md5sum program1 program2
```

We have same hash.

Running the programs and checking their behavior.

Program1: