MD Collision Attack Lab

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

Question 1: Prefix File is not a multiple of 64.

```
[09/14/25]seed@VM:~/.../Labsetup$ echo -n "123456" > prefixlt64byt
es.txt
[09/14/25]seed@VM:~/.../Labsetup$ md5collgen -p prefixlt64bytes.tx
t -o lt64out1.bin lt64out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'lt64out1.bin' and 'lt64out2.bin'
Using prefixfile: 'prefixlt64bytes.txt'
Using initial value: a31d4b1bc6677de802ef45d1d3f46747

Generating first block: ......................
Generating second block: W......
Running time: 29.7564 s
[09/14/25]seed@VM:~/.../Labsetup$ diff lt64out1.bin lt64out2.bin
Binary files lt64out1.bin and lt64out2.bin differ
[09/14/25]seed@VM:~/.../Labsetup$ md5sum lt64out1.bin
cfaa55476fb78e0102870560f7210d7d  lt64out1.bin
[09/14/25]seed@VM:~/.../Labsetup$ md5sum lt64out2.bin
cfaa55476fb78e0102870560f7210d7d  lt64out2.bin
```

We can see that they share the same MD5 hash but the binary files are different. Let's use bless to see the differences.

lt64out1.bin

```
00000000 31 32 33 34 35 36 00 00 00 00 00 00 00 00 00 00 123456..........
00000012 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000024 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 F3 10 E4 51 CC 73 2B EA .............Q.s+.
00000048 33 90 E6 80 FB 9F 72 7B B5 7A EF 8C 26 31 75 10 15 42 3.....r{.z..&1u..B
0000005a 7C B2 90 8D 39 6F 33 B3 FA DD 5C EF 2C 9F 07 FC D9 41 |...9o3...\.,....A
0000006c D6 F1 07 50 94 A8 DA 01 C5 2F 2E 60 CB E6 06 8C 9F 72 ...P...../.`.....r
0000007e 22 B6 55 1A D9 E6 85 11 08 0F 08 90 95 1A E6 C4 D6 16 ".U..............
00000090 8A 31 DA 37 48 03 F5 3B F6 DC F8 16 CF EA FC AD 21 F8 .1.7H..;........!.
000000a2 BA 36 AC B2 EA A7 F6 57 49 BA 80 A4 1A 9D E3 54 65 EF .6.....WI......Te.
000000b4 70 C5 C3 F4 9C 9D 0C 76 B5 35 DE 12                    p......v.5..
```

lt64out2.bin

```
00000000 31 32 33 34 35 36 00 00 00 00 00 00 00 00 00 00 00 00 123456..........
00000012 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
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 F3 10 E4 51 CC 73 2B EA ..........Q.s+.
00000048 33 90 E6 80 FB 9F 72 7B B5 7A EF 0C 26 31 75 10 15 42 3.....r{.z..&1u..B
0000005a 7C B2 90 8D 39 6F 33 B3 FA DD 5C EF 2C 9F 07 FC D9 41 |...9o3...\.,....A
0000006c D6 71 08 50 94 A8 DA 01 C5 2F 2E 60 CB E6 06 0C 9F 72 .q.P...../.`.....r
0000007e 22 B6 55 1A D9 E6 85 11 08 0F 08 90 95 1A E6 C4 D6 16 ".U.............
00000090 8A 31 DA B7 48 03 F5 3B F6 DC F8 16 CF EA FC AD 21 F8 .1..H..;........!.
000000a2 BA 36 AC B2 EA A7 F6 57 49 BA 80 24 1A 9D E3 54 65 EF .6.....WI..$...Te.
000000b4 70 C5 C3 F4 9C 9D 0C F6 B5 35 DE 12                    p........5..
```

From this we can clearly see that there is a difference with the bit at offset 0x37 and 0x3A

Question 2: Exactly 64 bytes

```
[09/14/25]seed@VM:~/.../Labsetup$ echo -n "0123456789ABCDEF0123456
789ABCDEF0123456789ABCDEF0123456789ABCDEF" > prefixeq64bytes.txt
[09/14/25]seed@VM:~/.../Labsetup$ md5collgen -p prefixeq64bytes.tx
t -o eq64out1.bin eq64out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'eq64out1.bin' and 'eq64out2.bin'
Using prefixfile: 'prefixeq64bytes.txt'
Using initial value: a736bd850d690ec7c2aec660a492ff33

Generating first block: .......
Generating second block: S10.......
Running time: 13.4747 s
[09/14/25]seed@VM:~/.../Labsetup$ diff eq64out1.bin eq64out2.bin
Binary files eq64out1.bin and eq64out2.bin differ
[09/14/25]seed@VM:~/.../Labsetup$ md5sum eq64out1.bin
967fee529d2a4b2e65064377a2ee71e9  eq64out1.bin
[09/14/25]seed@VM:~/.../Labsetup$ md5sum eq64out2.bin
967fee529d2a4b2e65064377a2ee71e9  eq64out2.bin
```

From this, we can also see that they have the same MD5 hash, but they differ. Similarly, using bless

eq64out1.bin

```
00000000 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 30 31 |0123456789ABCDEF01|
00000012 32 33 34 35 36 37 38 39 41 42 43 44 45 46 30 31 32 33 |23456789ABCDEF0123|
00000024 34 35 36 37 38 39 41 42 43 44 45 46 30 31 32 33 34 35 |456789ABCDEF012345|
00000036 36 37 38 39 41 42 43 44 45 46 C5 EB 67 06 79 99 18 0F |6789ABCDEF..g.y...|
00000048 16 9F 0C 99 38 C6 57 4A 8B FA 84 35 66 EB D9 AE 4C D6 |....8.WJ...5f...L.|
0000005a 79 19 E7 07 27 A9 3A D3 FA DE 6C A6 19 13 E7 C7 F6 33 |y...'.:...l......3|
0000006c 96 B9 47 5A 84 60 EC 8B 58 AA AD E0 A8 1F DC 4A 37 EB |..GZ.`..X......J7.|
0000007e 34 3B F0 ED B0 02 42 14 D5 4D A3 CD FD 7C 9A B1 0E 3B |4;....B..M...|...;|
00000090 91 8F 2B 74 5C DB 92 CF 2D B5 20 6F 06 60 68 D6 CB 53 |..+t\...-. o.`h..S|
000000a2 26 28 A7 87 FC 55 B4 FE 01 5C 42 00 16 92 16 AF B7 40 |&(...U...\B......@|
000000b4 36 72 23 BA 4B 91 9C C2 1C 1B E0 DC                   |6r#.K.......|
```

eq64out2.bin

```
00000000 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 30 31 |0123456789ABCDEF01|
00000012 32 33 34 35 36 37 38 39 41 42 43 44 45 46 30 31 32 33 |23456789ABCDEF0123|
00000024 34 35 36 37 38 39 41 42 43 44 45 46 30 31 32 33 34 35 |456789ABCDEF012345|
00000036 36 37 38 39 41 42 43 44 45 46 C5 EB 67 06 79 99 18 0F |6789ABCDEF..g.y...|
00000048 16 9F 0C 99 38 C6 57 4A 8B FA 84 B5 66 EB D9 AE 4C D6 |....8.WJ....f...L.|
0000005a 79 19 E7 07 27 A9 3A D3 FA DE 6C A6 19 13 E7 C7 F6 33 |y...'.:...l......3|
0000006c 96 39 48 5A 84 60 EC 8B 58 AA AD E0 A8 1F DC CA 37 EB |.9HZ.`..X.......7.|
0000007e 34 3B F0 ED B0 02 42 14 D5 4D A3 CD FD 7C 9A B1 0E 3B |4;....B..M...|...;|
00000090 91 8F 2B F4 5C DB 92 CF 2D B5 20 6F 06 60 68 D6 CB 53 |..+.\...-. o.`h..S|
000000a2 26 28 A7 87 FC 55 B4 FE 01 5C 42 80 15 92 16 AF B7 40 |&(...U...\B......@|
000000b4 36 72 23 BA 4B 91 9C 42 1C 1B E0 DC                   |6r#.K..B....|
```

From this we can clearly see that there is a difference with the bit at offset 0x1A and 0x36

Question 3:

From the 2 tests that we ran, for the file with less than 64 bytes, they differed only at the bits at offset 0x37 and 0x3A and for the file with exactly 64 bytes, they differed only at the bits at offset 0x1A and 0x36

Task 2: Understanding MD5's Property

Create a new suffix file:

```
[09/14/25]seed@VM:~/.../Labsetup$ echo -n "Hello" > suffix.txt
```

Use eq64out1.bin and eq64out2.bin as we know, have the same MD5 Hash

Concatenate suffix.txt

```
[09/14/25]seed@VM:~/.../Labsetup$ cat eq64out1.bin suffix.txt > ex
tended1.bin
[09/14/25]seed@VM:~/.../Labsetup$ cat eq64out2.bin suffix.txt > ex
tended2.bin
```

Check the hashes (should be the same)

```
[09/14/25]seed@VM:~/.../Labsetup$ md5sum extended1.bin extended2.b
in
ba920387af1b3a934c9bbd84c55639b2  extended1.bin
ba920387af1b3a934c9bbd84c55639b2  extended2.bin
```

From this, we can see that they still have the same MD5 hash after concatenating a new file to them.

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

Create executable

```
[09/14/25]seed@VM:~/.../Labsetup$ nano prog.c
[09/14/25]seed@VM:~/.../Labsetup$ gcc -o prog prog.c
```

Using bless to find the offset → 0x3020

Calculate prefix → prefix + 128 <= 0x3020 + 200 = 0x30E8 → 0x3050 (12336) to 128 bytes → 0x30CF

Create collision blocks

```
[09/14/25]seed@VM:~/.../Labsetup$ head -c 12336 prog > prefix
[09/14/25]seed@VM:~/.../Labsetup$ md5collgen -p prefix -o p.bin q.
bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'p.bin' and 'q.bin'
Using prefixfile: 'prefix'
Using initial value: b26d6dd630063b7409cc0f17e92723cb

Generating first block: ..
Generating second block: W......
Running time: 1.1939 s
[09/14/25]seed@VM:~/.../Labsetup$ tail -c +12465 prog > suffix
```

Create new executable files

```
[09/14/25]seed@VM:~/.../Labsetup$ cat p.bin suffix > prog1
[09/14/25]seed@VM:~/.../Labsetup$ cat q.bin suffix > prog2
[09/14/25]seed@VM:~/.../Labsetup$ chmod +x prog1 prog2
[09/14/25]seed@VM:~/.../Labsetup$ md5sum prog1 prog2
8f7c316612959bb420c6a4a83cbd0788  prog1
8f7c316612959bb420c6a4a83cbd0788  prog2
```

From this we can see that they share the same hash, but now lets see if they print the same array.

```
[09/14/25]seed@VM:~/.../Labsetup$ prog1
4141414141414141414141414141410000000000000000000000000000007c
4cc0093b36dccd935b55ae01ed75f578c2be563c689674109e7fde16ca129ed7ba
a6e5dc55e508ebc5c11105bcd365a88b5b30062064b396cb4258ea9e3d4ed06c69
ebb7b1c1422fdbfd21f07b7ae517199a26c6fbb82d18ed7535cb2b3dce55f83a78
a5b1c7073260096a8a48a55ce7d0d1c9b5bd33626f19e053081c25574141414141
4141414141414141414141414141414141414141414141414141414141414141
4141
[09/14/25]seed@VM:~/.../Labsetup$ prog2
4141414141414141414141414141410000000000000000000000000000007c
4cc0093b36dccd935b55ae01ed75f578c2bed63c689674109e7fde16ca129ed7ba
a6e5dc55e508ebc5c111053cd465a88b5b30062064b396cb42d8ea9e3d4ed06c69
ebb7b1c1422fdbfd21f07b7ae517199aa6c6fbb82d18ed7535cb2b3dce55f83a78
a5b1c7073260096a8ac8a45ce7d0d1c9b5bd33626f19e0d3081c25574141414141
4141414141414141414141414141414141414141414141414141414141414141
4141
```

From this we can see that they do not print the same list, we can verify with differ on the executable files

Differences:

prog1: ...578c2be563c6…
prog2: ...578c2bed63c6...

prog1: ...c11105bcd3...
prog2: ...c111053cd4...

prog1: ...8a48a55ce7d0...
prog2: ...8a45ce7d0...

```
[09/14/25]seed@VM:~/.../Labsetup$ diff prog1 prog2
Binary files prog1 and prog2 differ
```

Task 4: Making the Two Programs Behave Differently

Similarly to Task 3, after modifying this C file,

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

unsigned char X[128] = {
  /* 128 times 0x41 */
  #define A 0x41
  #include "fill128.h"
```

```
};
unsigned char Y[128] = {
  /* 128 times 0x41 */
  #include "fill128.h"
};

void benign() {
  puts("BENIGN: doing good things.");
}
void malicious() {
  puts("MALICIOUS: doing nasty things!");
}

int main(void) {
  if (memcmp(X, Y, 128) == 0) benign();
  else malicious();
  return 0;
}
```

We produce 2 separate programs in which one performs the benign operation while the other performs the malicious function every though they share the same MD5 hash

```
[09/14/25]seed@VM:~/.../Labsetup$ task4first
BENIGN: doing good things.
[09/14/25]seed@VM:~/.../Labsetup$ task4second
MALICIOUS: doing nasty things!
```

```
[09/14/25]seed@VM:~/.../Labsetup$ md5sum task4first task4second
e54b50df8248687679d64f1bfe8d7cf8  task4first
e54b50df8248687679d64f1bfe8d7cf8  task4second
```