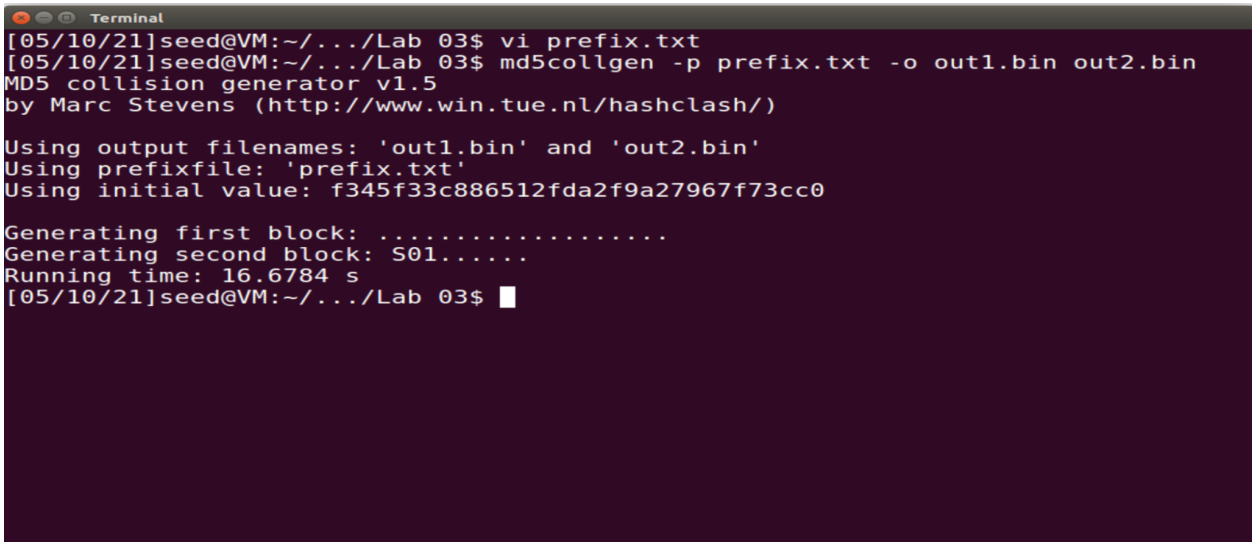
Lab 11: Hashing and Collision Attacks

Introduction:

Hashing refers to the act of passing data such as a string through a special one way function known as a cryptographic hash function. Often this is done for the sake of verifying the integrity, or maintaining the confidentiality of a file. Ideally, cryptographic hash functions should have no easily computable inverse and should produce a perfect one to one mapping of data to hash values. In other words, every input to a cryptographic hash function should produce a unique hash value as an output, and it should be computationally infeasible to derive the original data from any given hash value. Unfortunately however, this is not always the case as was seen in this lab. It is indeed possible to produce the same hash value (collision) from two different files using the deprecated MD5 cryptographic hash function. The implications of which are startling

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

Two different binary files were generated, each beginning with the same prefix.



```
Terminal
[05/10/21]seed@VM:~/.../Lab 03$ vi prefix.txt
[05/10/21]seed@VM:~/.../Lab 03$ 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: f345f33c886512fda2f9a27967f73cc0

Generating first block: ...................
Generating second block: S01......
Running time: 16.6784 s
[05/10/21]seed@VM:~/.../Lab 03$
```

Even though the files differ, the hashes are still the same.

```
Terminal
[05/10/21]seed@VM:~/.../Lab 03$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[05/10/21]seed@VM:~/.../Lab 03$ md5sum out1.bin
9be57035e705020293f61e44a7e5b616  out1.bin
[05/10/21]seed@VM:~/.../Lab 03$ md5sum out2.bin
9be57035e705020293f61e44a7e5b616  out2.bin
[05/10/21]seed@VM:~/.../Lab 03$
```

The output file is padded with zeros if the prefix file's size is not a multiple of 64.

```
Terminal
[05/10/21]seed@VM:~/.../Lab 03$ hexdump -C out3.bin
00000000  57 65 20 54 68 65 20 50  65 6f 70 6c 65 2c 20 69  |We The People, i|
00000010  6e 20 6f 72 64 65 72 20  74 6f 20 66 6f 72 6d 20  |n order to form |
00000020  61 20 6d 6f 72 65 20 70  65 72 66 65 63 74 20 75  |a more perfect u|
00000030  6e 69 6f 6e 2c 20 65 73  74 61 62 6c 69 73 68 20  |nion, establish |
00000040  c1 03 bc c9 c2 74 29 58  d8 1d dc 4f e0 a5 32 b5  |.....t)X...O..2.|
00000050  c4 94 1f 48 78 18 2e 13  32 52 43 8e 4a 88 0e 35  |...Hx...2RC.J..5|
00000060  a9 d3 ae ce 4e 6d 4b 0b  67 28 68 dc 06 a2 9d 6b  |....NmK.g(h....k|
00000070  a2 16 ba 8f 1d ca a4 65  ba 06 14 5f ce 68 3f 42  |.......e..._.h?B|
00000080  dd e4 81 ba aa 1b 24 a6  15 0c ff c0 31 07 08 26  |......$.....1..&|
00000090  26 3d e3 2d 13 e6 25 26  cb 5a 06 c8 ec c6 dd ef  |&=.-..%&.Z......|
000000a0  8e 6d 3e 55 f4 18 00 93  68 52 65 d7 8a 2c e1 aa  |.m>U....hRe..,..|
000000b0  46 92 81 44 aa ae 30 1d  9c 5d 9e fc 50 c2 74 81  |F..D..0..]..P.t.|
000000c0
[05/10/21]seed@VM:~/.../Lab 03$
```

No padding is observed when the prefix is truncated to a multiple of 64 bytes

```
Terminal
[05/10/21]seed@VM:~/.../Lab 03$ hexdump -C out3.bin
00000000  57 65 20 54 68 65 20 50  65 6f 70 6c 65 2c 20 69  |We The People, i|
00000010  6e 20 6f 72 64 65 72 20  74 6f 20 66 6f 72 6d 20  |n order to form |
00000020  61 20 6d 6f 72 65 20 70  65 72 66 65 63 74 20 75  |a more perfect u|
00000030  6e 69 6f 6e 2c 20 65 73  74 61 62 6c 69 73 68 20  |nion, establish |
00000040  c1 03 bc c9 c2 74 29 58  d8 1d dc 4f e0 a5 32 b5  |.....t)X...O..2.|
00000050  c4 94 1f 48 78 18 2e 13  32 52 43 8e 4a 88 0e 35  |...Hx...2RC.J..5|
00000060  a9 d3 ae ce 4e 6d 4b 0b  67 28 68 dc 06 a2 9d 6b  |....NmK.g(h....k|
00000070  a2 16 ba 8f 1d ca a4 65  ba 06 14 5f ce 68 3f 42  |.......e..._.h?B|
00000080  dd e4 81 ba aa 1b 24 a6  15 0c ff c0 31 07 08 26  |......$.....1..&|
00000090  26 3d e3 2d 13 e6 25 26  cb 5a 06 c8 ec c6 dd ef  |&=.-...%&.Z.....|
000000a0  8e 6d 3e 55 f4 18 00 93  68 52 65 d7 8a 2c e1 aa  |.m>U....hRe..,..|
000000b0  46 92 81 44 aa ae 30 1d  9c 5d 9e fc 50 c2 74 81  |F..D..0..]..P.t.|
000000c0
[05/10/21]seed@VM:~/.../Lab 03$
```

Viewing the contents of the output files with the bless hex editor shows that they differ.

```
out1.bin   out2.bin
00000000 57 65 20 54 68 65 20 50 65 6F 70 6C 65 2C 20 69 6E 20 6F 72 64 65 72 20 74 6F 20 66 6F 72 6D 20 61 20 6D 6F 72 65 20 70 65 72 66 65 We The People, in order to form a more perfe
0000002c 63 74 20 75 6E 69 6F 6E 2C 20 65 73 74 61 62 6C 69 73 68 20 6A 75 73 74 69 63 65 2C 20 65 6E 73 75 72 65 0A 64 6F 6D 65 73 74 69 63 ct union, establish justice, ensure.domestic
00000058 20 74 72 61 6E 71 75 69 6C 69 74 79 2C 20 70 72 6F 76 69 64 65 20 66 6F 72 20 74 68 65 20 63 6F 6D 6D 6F 6E 20 64 65 66 65 6E 73 65  tranquility, provide for the common defense
00000084 2C 20 61 6E 64 20 73 65 63 75 72 65 20 74 68 65 20 62 6C 65 73 73 69 6E 67 73 0A 6F 66 20 6C 69 62 65 72 74 79 20 66 6F 72 20 6F 75 , and secure the blessings.of liberty for ou
000000b0 72 73 65 6C 76 65 73 20 61 6E 64 20 6F 75 72 20 70 6F 73 74 65 72 69 74 79 2E 20 44 6F 20 6F 72 64 61 69 6E 20 61 6E 64 20 65 73 74 rselves and our posterity. Do ordain and est
000000dc 61 62 6C 69 73 68 20 74 68 69 73 0A 43 6F 6E 73 74 69 74 75 74 69 6F 6E 20 6F 66 20 54 68 65 20 55 6E 69 74 65 64 20 53 74 61 74 65 ablish this.Constitution of The United State
00000108 73 20 6F 66 20 41 6D 65 72 69 63 61 2E 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 s of America.................................
00000134 00 00 00 00 00 00 00 00 00 00 00 00 14 D1 1B 4C 18 C1 9E 54 53 24 96 63 CE 5A B8 C9 59 38 29 32 31 B7 2E A3 26 B6 43 4D 42 B8 B6 D8 ...............L...TS$.c.Z..Y8)21...&.CMB...
00000160 EB 15 C1 DA DC 63 3D CB 67 3E C7 D4 3D 86 F7 4B D0 2C 1A 5B A9 0A 74 7D 28 CA 2B 6E 5D B8 A9 E0 5C CA 66 74 B6 D5 14 E6 01 1D 01 09 .....c=.g>..=..K.,.[..t}(.+n]...\.ft........
0000018c ED 8E ED E2 B2 7F D4 1A 2D E6 35 31 D1 5A 86 4B EC C7 FD 6B F8 24 9E 19 F6 8C D8 41 58 86 48 EF D2 AC C3 EC F9 C8 8D 12 5C EE 27 C1 ........-.51.Z.K...k.$.....AX.H.........\.'.
000001b8 C6 1C A5 29 EE B1 F4 E6                                                                                                          ...)....
```

out1.bin ✖ out2.bin ✖

```
00000000 57 65 20 54 68 65 20 50 65 6F 70 6C 65 2C 20 69 6E 20 6F 72 64 65 72 20 74 6F 20 66 6F 72 6D 20 61 20 6D 6F 72 65 20 70 65 72 66 65 We The People, in order to form a more perfe
0000002c 63 74 20 75 6E 69 6F 6E 2C 20 65 73 74 61 62 6C 69 73 68 20 6A 75 73 74 69 63 65 2C 20 65 6E 73 75 72 65 0A 64 6F 6D 65 73 74 69 63 ct union, establish justice, ensure.domestic
00000058 20 74 72 61 6E 71 75 69 6C 69 74 79 2C 20 70 72 6F 76 69 64 65 20 66 6F 72 20 74 68 65 20 63 6F 6D 6D 6F 6E 20 64 65 66 65 6E 73 65  tranquility, provide for the common defense
00000084 2C 20 61 6E 64 20 73 65 63 75 72 65 20 74 68 65 20 62 6C 65 73 73 69 6E 67 73 0A 6F 66 20 6C 69 62 65 72 74 79 20 66 6F 72 20 6F 75 , and secure the blessings.of liberty for ou
000000b0 72 73 65 6C 76 65 73 20 61 6E 64 20 6F 75 72 20 70 6F 73 74 65 72 69 74 79 2E 20 44 6F 20 6F 72 64 61 69 6E 20 61 6E 64 20 65 73 74 rselves and our posterity. Do ordain and est
000000dc 61 62 6C 69 73 68 20 74 68 69 73 0A 43 6F 6E 73 74 69 74 75 74 69 6F 6E 20 6F 66 20 54 68 65 20 55 6E 69 74 65 64 20 53 74 61 74 65 ablish this.Constitution of The United State
00000108 73 20 6F 66 20 41 6D 65 72 69 63 61 2E 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 s of America.............................
00000134 00 00 00 00 00 00 00 00 00 00 00 00 14 D1 1B 4C 18 C1 9E 54 53 24 96 63 CE 5A B8 C9 59 38 29 B2 31 B7 2E A3 26 B6 43 4D 42 B8 B6 D8 ...............L...TS$.c.Z..Y8).1...&.CMB...
00000160 EB 15 C1 DA DC 63 3D CB 67 3E C7 D4 3D 06 F8 4B D0 2C 1A 5B A9 0A 74 7D 28 CA 2B EE 5D B8 A9 E0 5C CA 66 74 B6 D5 14 E6 01 1D 01 09 .....c=.g>..=..K.,.[..t}(.+.]...\.ft........
0000018c ED 8E ED E2 B2 7F D4 9A 2D E6 35 31 D1 5A 86 4B EC C7 FD 6B F8 24 9E 19 F6 8C D8 41 58 86 48 EF D2 2C C3 EC F9 C8 8D 12 5C EE 27 C1 .........-.51.Z.K...k.$.....AX.H..,......\.'.
000001b8 C6 1C A5 A9 EE B1 F4 E6 ........
```

Task 2: Understanding MD5's Property

When two different inputs produce the same MD5 hash value, concatenating a random string to either will produce the same hash value as well. This property holds true for many other cryptographic hash functions, not just MD5.

Generating two files with the same hash value.

```
Terminal
[05/10/21]seed@VM:~/.../Lab 03$ md5collgen -p loremipsum.txt -o out5.bin out6.bi
n
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out5.bin' and 'out6.bin'
Using prefixfile: 'loremipsum.txt'
Using initial value: cdd4e629c1951f1f460435ee05f1a225

Generating first block: ....................................
Generating second block: W..............
Running time: 15.5576 s
[05/10/21]seed@VM:~/.../Lab 03$ md5sum out5.bin
6eb31c233877ec4c41b0480018733b14  out5.bin
[05/10/21]seed@VM:~/.../Lab 03$ md5sum out6.bin
6eb31c233877ec4c41b0480018733b14  out6.bin
[05/10/21]seed@VM:~/.../Lab 03$
```

Concatenating "sit" to the lorem ipsum text file does not change the hash value.

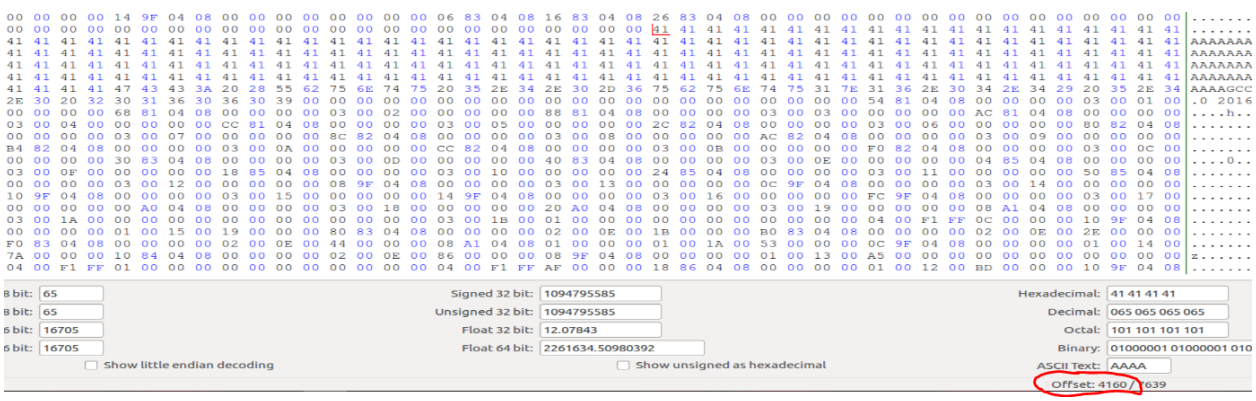
```
Terminal
[05/10/21]seed@VM:~/.../Lab 03$ echo sit >> loremipsum.txt
[05/10/21]seed@VM:~/.../Lab 03$ md5sum out5.bin out6.bin
6eb31c233877ec4c41b0480018733b14  out5.bin
6eb31c233877ec4c41b0480018733b14  out6.bin
[05/10/21]seed@VM:~/.../Lab 03$
```

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

Inputs to the MD5 hash function may also come in the form of executable files. Similar to how two different binary files may produce the same hash value so too may executables. The following C program was used to verify this.

```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','A','A','A','A','A'
,'A','A','A','A','A','A','A','A','A','A','A','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");
}
```

Compiling with gcc and viewing the resultant out file with bless allows for the determination of the array's base address, which is then used to form the prefix for the two executables.



Using said prefix to generate two different out files. Also adding permissions for the two executables.

```
[05/10/21]seed@VM:~/.../Lab 03$ head -c 4160 program.out > prefix
[05/10/21]seed@VM:~/.../Lab 03$ md5collgen -p prefix -o program1 program2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'program1' and 'program2'
Using prefixfile: 'prefix'
Using initial value: af0310fc20c32a33c9710c228be83872

Generating first block: .......................
Generating second block: S10..........
Running time: 14.3496 s
[05/10/21]seed@VM:~/.../Lab 03$ tail -c 4288 program.out > suffix
[05/10/21]seed@VM:~/.../Lab 03$ cat suffix >> program1
[05/10/21]seed@VM:~/.../Lab 03$ cat suffix >> program2
[05/10/21]seed@VM:~/.../Lab 03$ chmod +x program1
[05/10/21]seed@VM:~/.../Lab 03$ chmod +x program2
[05/10/21]seed@VM:~/.../Lab 03$
```

Piping the output of the executables to separate files and running diff on them shows that the programs behave differently.

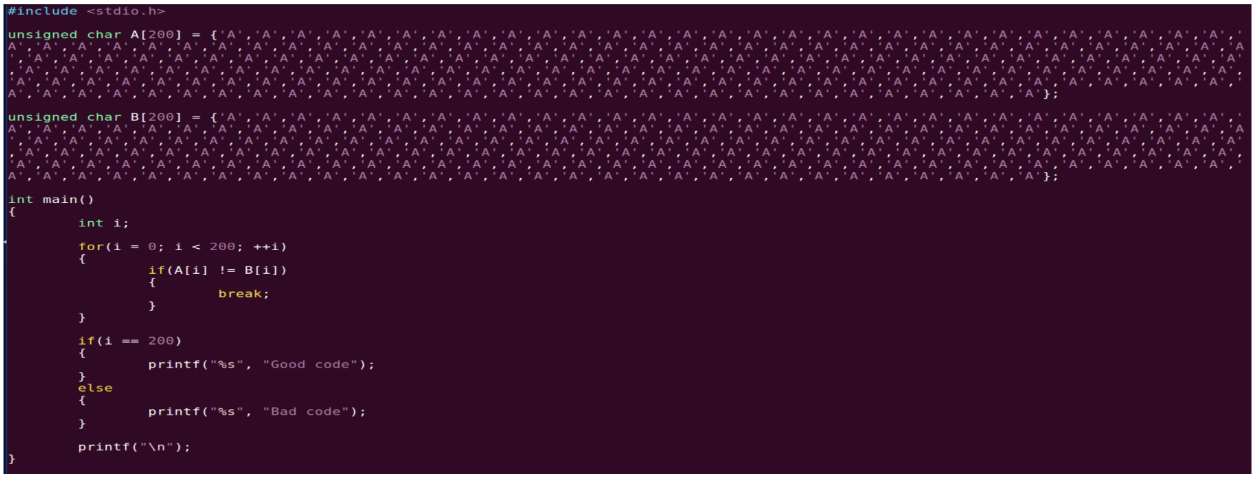
```
[05/10/21]seed@VM:~/.../Lab 03$ ./program1 > output1
[05/10/21]seed@VM:~/.../Lab 03$ ./program2 > output2
[05/10/21]seed@VM:~/.../Lab 03$ diff -q output1 output2
Files output1 and output2 differ
[05/10/21]seed@VM:~/.../Lab 03$
```

However, the hash values remain the same.

```
[05/10/21]seed@VM:~/.../Lab 03$ md5sum program1
991c1615de4a4aa32ae13bb4539c45eb  program1
[05/10/21]seed@VM:~/.../Lab 03$ md5sum program2
991c1615de4a4aa32ae13bb4539c45eb  program2
[05/10/21]seed@VM:~/.../Lab 03$
```

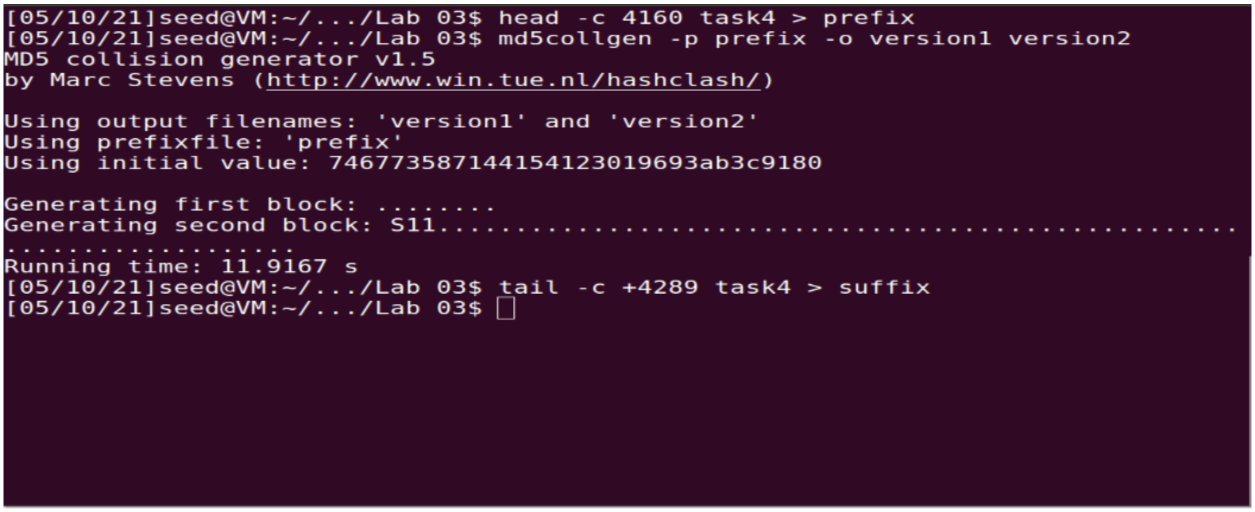
Task 4: Making the Two Programs Behave Differently

Hash collisions can be exploited to dramatically change the behavior of some programs. The following code is intended to display “Good code” to the screen if both arrays A and B are equal, and “Bad code” if they are not.



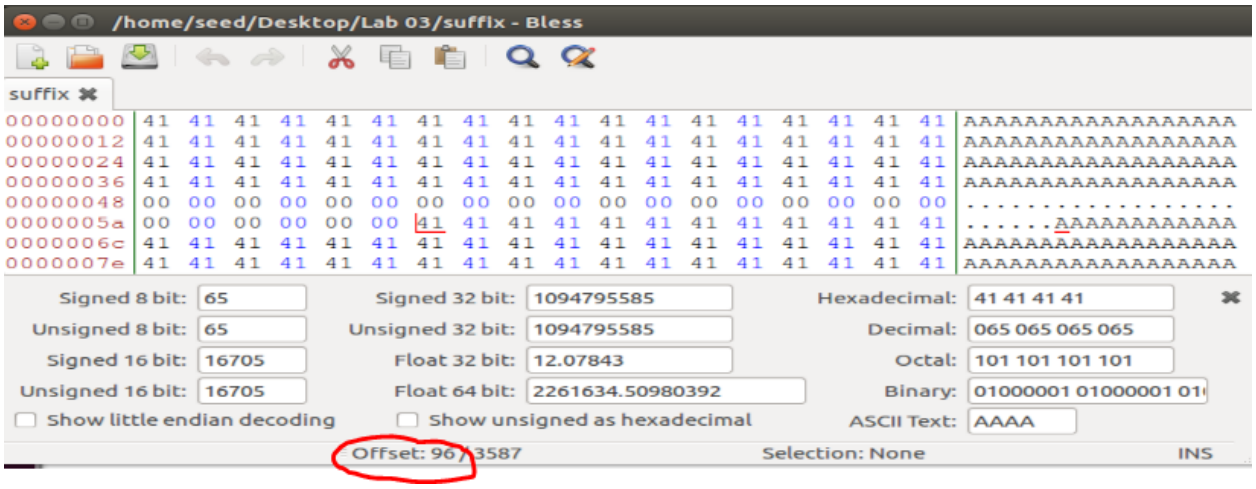
```
#include <stdio.h>

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'};

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'};

int main()
{
        int i;

        for(i = 0; i < 200; ++i)
        {
                if(A[i] != B[i])
                {
                        break;
                }
        }

        if(i == 200)
        {
                printf("%s", "Good code");
        }
        else
        {
                printf("%s", "Bad code");
        }

        printf("\n");
}
```

The prefix and suffix remain in the same byte regions as they did in task 3. Two differing executables with the same hash were then generated.



```
[05/10/21]seed@VM:~/.../Lab 03$ head -c 4160 task4 > prefix
[05/10/21]seed@VM:~/.../Lab 03$ md5collgen -p prefix -o version1 version2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'version1' and 'version2'
Using prefixfile: 'prefix'
Using initial value: 746773587144154123019693ab3c9180

Generating first block: ........
Generating second block: S11...............................................
..................
Running time: 11.9167 s
[05/10/21]seed@VM:~/.../Lab 03$ tail -c +4289 task4 > suffix
[05/10/21]seed@VM:~/.../Lab 03$ 
```

Viewing the suffix file in bless reveals the base address of the array B. This offset was then used to further subdivide the suffix into two parts.



The final 128 bytes of each executable were then used in conjunction with the two newly created suffix files to construct the two final executables. Execute permissions were given.

```
[05/10/21]seed@VM:~/.../Lab 03$ head -c 96 suffix > suffix1
[05/10/21]seed@VM:~/.../Lab 03$ tail -c +224 suffix > suffix2
[05/10/21]seed@VM:~/.../Lab 03$ tail -c 128 version1 > p
[05/10/21]seed@VM:~/.../Lab 03$ tail -c 128 version2 > q
[05/10/21]seed@VM:~/.../Lab 03$ cat prefix p suffix1 p suffix2 > task4v1
[05/10/21]seed@VM:~/.../Lab 03$ cat prefix q suffix1 p suffix2 > task4v2
[05/10/21]seed@VM:~/.../Lab 03$ chmod +x task4v1
[05/10/21]seed@VM:~/.../Lab 03$ chmod +x task4v2
[05/10/21]seed@VM:~/.../Lab 03$
```

The two programs execute completely different machine level instructions, leading to drastically different behavior.

```
[05/10/21]seed@VM:~/.../Lab 03$ ./task4v1
Good code
[05/10/21]seed@VM:~/.../Lab 03$ ./task4v2
Bad code
[05/10/21]seed@VM:~/.../Lab 03$
```

Despite producing different results, the two still share the same hash value.

```
[05/10/21]seed@VM:~/.../Lab 03$ md5sum task4v1
31d6c66ecb9a849939e9ba70119b87a5  task4v1
[05/10/21]seed@VM:~/.../Lab 03$ md5sum task4v2
31d6c66ecb9a849939e9ba70119b87a5  task4v2
[05/10/21]seed@VM:~/.../Lab 03$
```

Conclusion:

The presence of collisions in cryptographic hash functions can have potentially far reaching and devastating effects. As was seen in this lab, an attacker could use them to execute malicious code, or to fraudulently access password protected data in what is known as a pass the hash attack. It is therefore of the utmost importance to switch algorithms once one becomes compromised. Implying that cryptographic hash functions have an effective lifespan before they can no longer be used safely.