Emma Stensland

Reese Pearsall

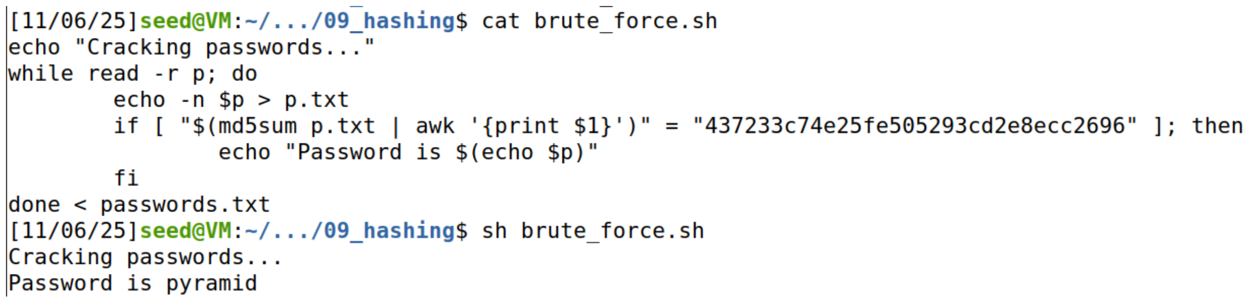
CSCI 476

October 26, 2025

Seedlabs: MD5 Collision Attack

Task 1: Brute Force Attack Against a Hashed Password

An unknown hash held a password. In order to brute force this password, a bash script was developed to loop through a file of common passwords, hash them, and compare it to the unknown hash of the password.



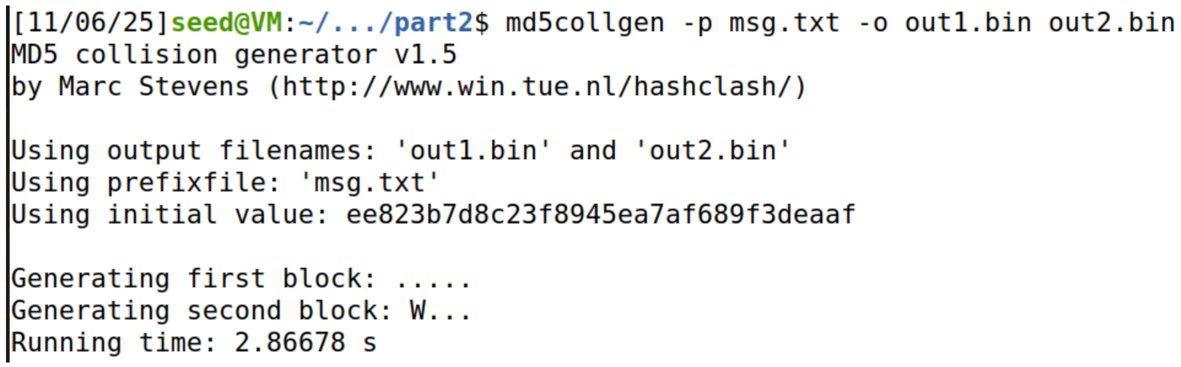
```
[11/06/25]seed@VM:~/.../09_hashing$ cat brute_force.sh
echo "Cracking passwords..."
while read -r p; do
        echo -n $p > p.txt
        if [ "$(md5sum p.txt | awk '{print $1}')" = "437233c74e25fe505293cd2e8ecc2696" ]; then
                echo "Password is $(echo $p)"
        fi
done < passwords.txt
[11/06/25]seed@VM:~/.../09_hashing$ sh brute_force.sh
Cracking passwords...
Password is pyramid
```

Observations: The password was found to be "pyramid". This did take a few seconds to brute force, and would take longer with a larger set of passwords to hash and compare against.

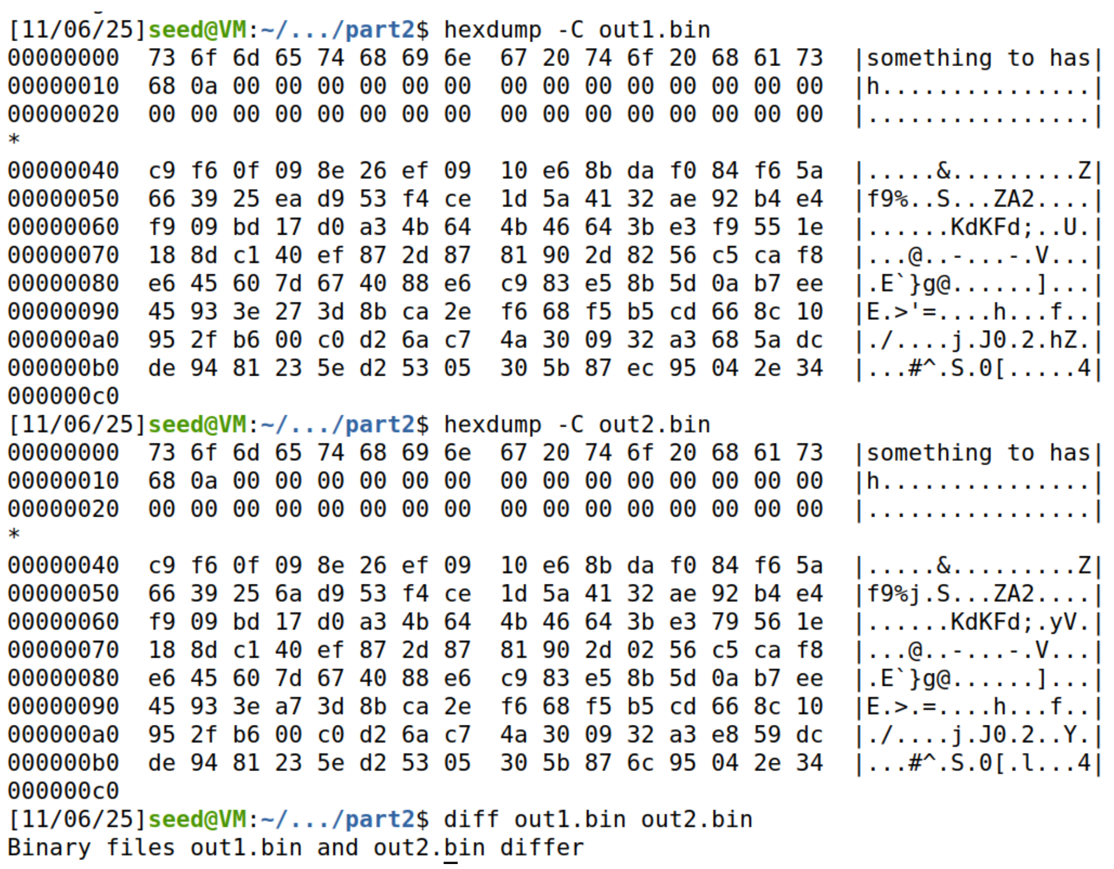
Task 2: Generating Two Different Files with the Same MD5 Hash

2.1

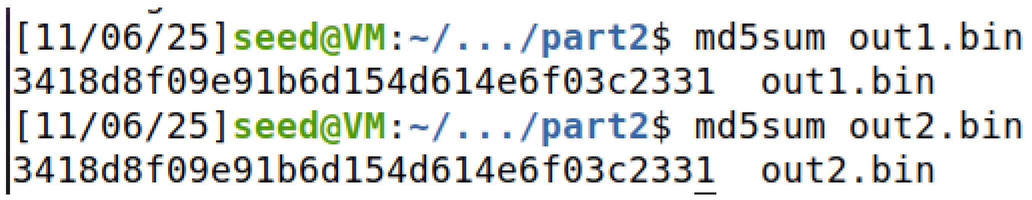
Two files with different hashes were generated using md5collgen, and the hashes were compared.



```
[11/06/25]seed@VM:~/.../part2$ md5collgen -p msg.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: 'msg.txt'
Using initial value: ee823b7d8c23f8945ea7af689f3deaaf

Generating first block: .....
Generating second block: W...
Running time: 2.86678 s
```



```
[11/06/25]seed@VM:~/.../part2$ hexdump -C out1.bin
00000000  73 6f 6d 65 74 68 69 6e  67 20 74 6f 20 68 61 73  |something to has|
00000010  68 0a 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |h...............|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000040  c9 f6 0f 09 8e 26 ef 09  10 e6 8b da f0 84 f6 5a  |.....&.........Z|
00000050  66 39 25 ea d9 53 f4 ce  1d 5a 41 32 ae 92 b4 e4  |f9%..S...ZA2....|
00000060  f9 09 bd 17 d0 a3 4b 64  4b 46 64 3b e3 f9 55 1e  |......KdKFd;..U.|
00000070  18 8d c1 40 ef 87 2d 87  81 90 2d 82 56 c5 ca f8  |...@..-...-.V...|
00000080  e6 45 60 7d 67 40 88 e6  c9 83 e5 8b 5d 0a b7 ee  |.E`}g@......]...|
00000090  45 93 3e 27 3d 8b ca 2e  f6 68 f5 b5 cd 66 8c 10  |E.>'=....h...f..|
000000a0  95 2f b6 00 c0 d2 6a c7  4a 30 09 32 a3 68 5a dc  |./....j.J0.2.hZ.|
000000b0  de 94 81 23 5e d2 53 05  30 5b 87 ec 95 04 2e 34  |...#^.S.0[.....4|
000000c0
[11/06/25]seed@VM:~/.../part2$ hexdump -C out2.bin
00000000  73 6f 6d 65 74 68 69 6e  67 20 74 6f 20 68 61 73  |something to has|
00000010  68 0a 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |h...............|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000040  c9 f6 0f 09 8e 26 ef 09  10 e6 8b da f0 84 f6 5a  |.....&.........Z|
00000050  66 39 25 6a d9 53 f4 ce  1d 5a 41 32 ae 92 b4 e4  |f9%j.S...ZA2....|
00000060  f9 09 bd 17 d0 a3 4b 64  4b 46 64 3b e3 79 56 1e  |......KdKFd;.yV.|
00000070  18 8d c1 40 ef 87 2d 87  81 90 2d 02 56 c5 ca f8  |...@..-...-.V...|
00000080  e6 45 60 7d 67 40 88 e6  c9 83 e5 8b 5d 0a b7 ee  |.E`}g@......]...|
00000090  45 93 3e a7 3d 8b ca 2e  f6 68 f5 b5 cd 66 8c 10  |E.>.=....h...f..|
000000a0  95 2f b6 00 c0 d2 6a c7  4a 30 09 32 a3 e8 59 dc  |./....j.J0.2..Y.|
000000b0  de 94 81 23 5e d2 53 05  30 5b 87 6c 95 04 2e 34  |...#^.S.0[.l...4|
000000c0
[11/06/25]seed@VM:~/.../part2$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
```



```
[11/06/25]seed@VM:~/.../part2$ md5sum out1.bin
3418d8f09e91b6d154d614e6f03c2331  out1.bin
[11/06/25]seed@VM:~/.../part2$ md5sum out2.bin
3418d8f09e91b6d154d614e6f03c2331  out2.bin
```

Observations: There is a header that contains the original file, then the file contents are similar with a few bytes being different. They generated the same hash.

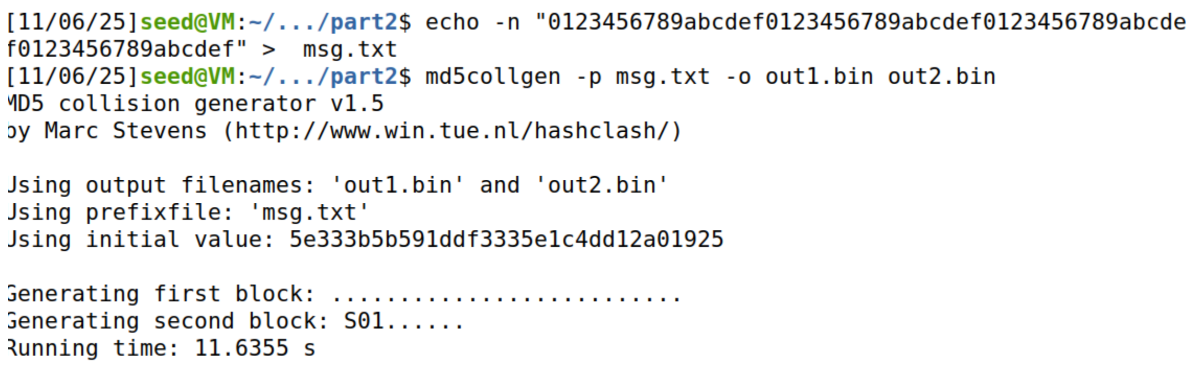
2.2

The above file was not 64 bytes long, so the hexdumps were analyzed to see how the header was modified.

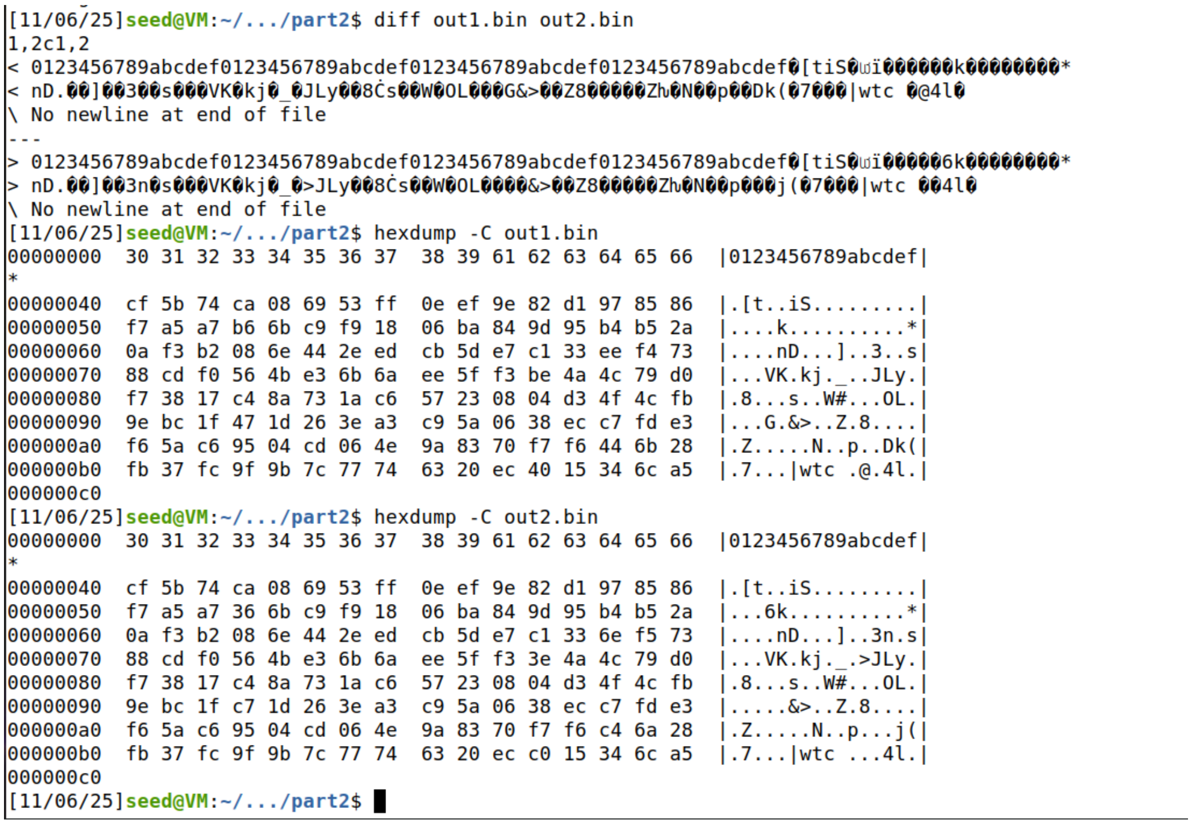
Observations: The file contained padding in order to fill in the header for output files.

2.3

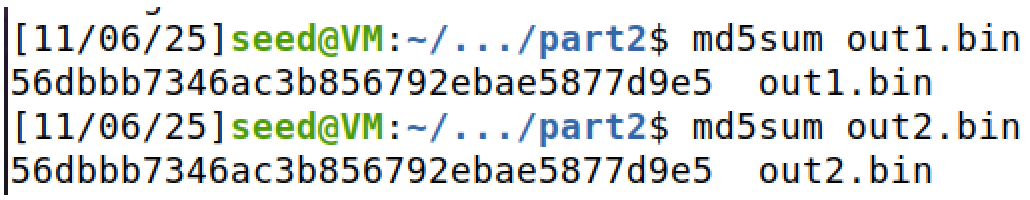
A file that was 64 bytes was then generated, and the md5collgen was run again, and out1.bin and out2.bin's hashes were compared.



```
[11/06/25]seed@VM:~/.../part2$ echo -n "0123456789abcdef0123456789abcdef0123456789abcde
f0123456789abcdef" >  msg.txt
[11/06/25]seed@VM:~/.../part2$ md5collgen -p msg.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: 'msg.txt'
Using initial value: 5e333b5b591ddf3335e1c4dd12a01925

Generating first block: ..........................
Generating second block: S01......
Running time: 11.6355 s
```



```
[11/06/25]seed@VM:~/.../part2$ diff out1.bin out2.bin
1,2c1,2
< 0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef�[tiS�ɯï������k��������*
< nD.��]��3��s���VK�kj�_�JLy��8Ċs��W�OL���G&>��Z8������Zɰ�N��p��Dk(�7���|wtc �@4l�
\ No newline at end of file
---
> 0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef�[tiS�ɯï������6k��������*
> nD.��]��3n�s���VK�kj�_�>JLy��8Ċs��W�OL����&>��Z8������Zɰ�N��p���j(�7���|wtc ��4l�
\ No newline at end of file
[11/06/25]seed@VM:~/.../part2$ hexdump -C out1.bin
00000000  30 31 32 33 34 35 36 37  38 39 61 62 63 64 65 66  |0123456789abcdef|
*
00000040  cf 5b 74 ca 08 69 53 ff  0e ef 9e 82 d1 97 85 86  |.[t..iS.........|
00000050  f7 a5 a7 b6 6b c9 f9 18  06 ba 84 9d 95 b4 b5 2a  |....k..........*|
00000060  0a f3 b2 08 6e 44 2e ed  cb 5d e7 c1 33 ee f4 73  |....nD...]..3..s|
00000070  88 cd f0 56 4b e3 6b 6a  ee 5f f3 be 4a 4c 79 d0  |...VK.kj._..JLy.|
00000080  f7 38 17 c4 8a 73 1a c6  57 23 08 04 d3 4f 4c fb  |.8...s..W#...OL.|
00000090  9e bc 1f 47 1d 26 3e a3  c9 5a 06 38 ec c7 fd e3  |...G.&>..Z.8....|
000000a0  f6 5a c6 95 04 cd 06 4e  9a 83 70 f7 f6 44 6b 28  |.Z.....N..p..Dk(|
000000b0  fb 37 fc 9f 9b 7c 77 74  63 20 ec 40 15 34 6c a5  |.7...|wtc .@.4l.|
000000c0
[11/06/25]seed@VM:~/.../part2$ hexdump -C out2.bin
00000000  30 31 32 33 34 35 36 37  38 39 61 62 63 64 65 66  |0123456789abcdef|
*
00000040  cf 5b 74 ca 08 69 53 ff  0e ef 9e 82 d1 97 85 86  |.[t..iS.........|
00000050  f7 a5 a7 36 6b c9 f9 18  06 ba 84 9d 95 b4 b5 2a  |...6k..........*|
00000060  0a f3 b2 08 6e 44 2e ed  cb 5d e7 c1 33 6e f5 73  |....nD...]..3n.s|
00000070  88 cd f0 56 4b e3 6b 6a  ee 5f f3 3e 4a 4c 79 d0  |...VK.kj._.>JLy.|
00000080  f7 38 17 c4 8a 73 1a c6  57 23 08 04 d3 4f 4c fb  |.8...s..W#...OL.|
00000090  9e bc 1f c7 1d 26 3e a3  c9 5a 06 38 ec c7 fd e3  |.....&>..Z.8....|
000000a0  f6 5a c6 95 04 cd 06 4e  9a 83 70 f7 f6 c4 6a 28  |.Z.....N..p...j(|
000000b0  fb 37 fc 9f 9b 7c 77 74  63 20 ec c0 15 34 6c a5  |.7...|wtc ...4l.|
000000c0
[11/06/25]seed@VM:~/.../part2$ █
```



```
[11/06/25]seed@VM:~/.../part2$ md5sum out1.bin
56dbbb7346ac3b856792ebae5877d9e5  out1.bin
[11/06/25]seed@VM:~/.../part2$ md5sum out2.bin
56dbbb7346ac3b856792ebae5877d9e5  out2.bin
```

Observations: Different files were generated with a header that contains no padding. They generated equivalent hashes.

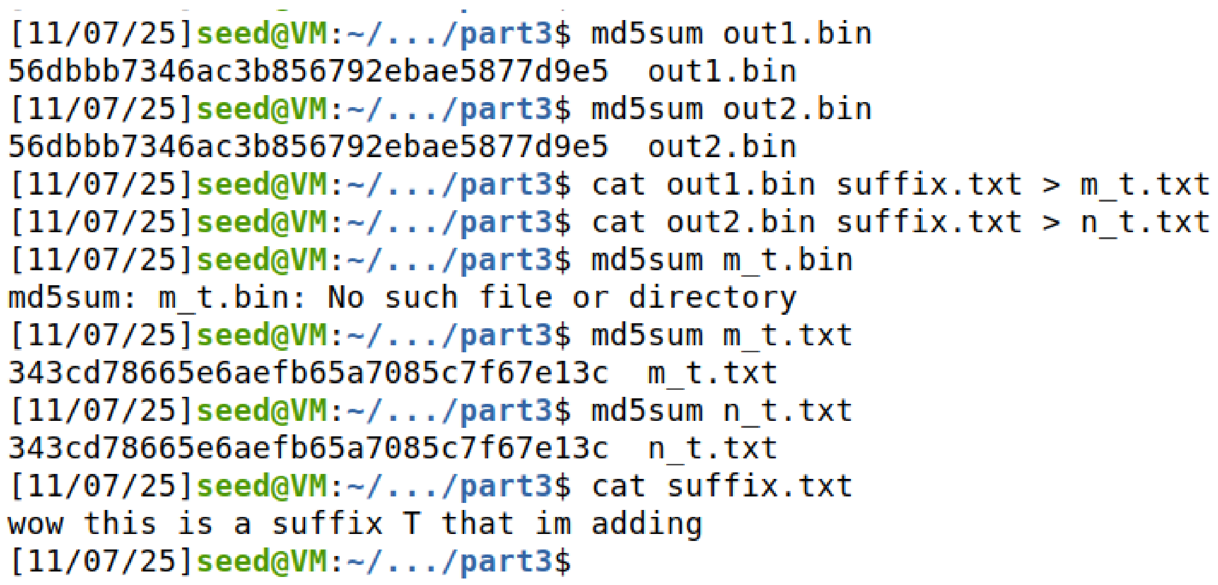
2.4

The collision generator's outputs were compared to understand where the MD5 collision occurs.

Observations: The header is equivalent, while the out.bin files appear to be almost equal, save for seven bytes in the same spots for parts 2.4 and 2.1.

Task 3: Understanding MD5's "Suffix Extension" Property

Two different files with equal hashes were generated. Then a file with a suffix was added to the end of both of these files, and the combined files were hashed again.

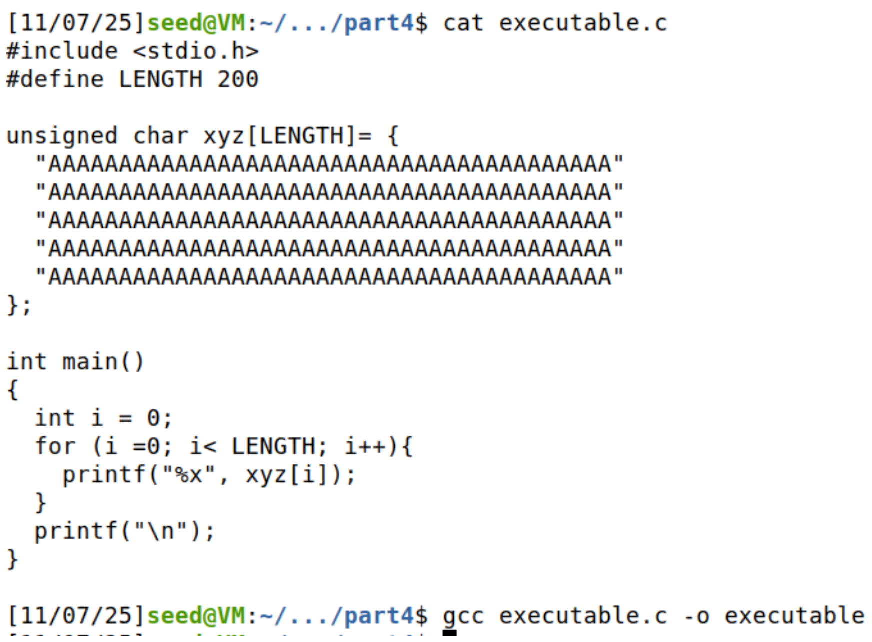


```
[11/07/25]seed@VM:~/.../part3$ md5sum out1.bin
56dbbb7346ac3b856792ebae5877d9e5  out1.bin
[11/07/25]seed@VM:~/.../part3$ md5sum out2.bin
56dbbb7346ac3b856792ebae5877d9e5  out2.bin
[11/07/25]seed@VM:~/.../part3$ cat out1.bin suffix.txt > m_t.txt
[11/07/25]seed@VM:~/.../part3$ cat out2.bin suffix.txt > n_t.txt
[11/07/25]seed@VM:~/.../part3$ md5sum m_t.bin
md5sum: m_t.bin: No such file or directory
[11/07/25]seed@VM:~/.../part3$ md5sum m_t.txt
343cd78665e6aefb65a7085c7f67e13c  m_t.txt
[11/07/25]seed@VM:~/.../part3$ md5sum n_t.txt
343cd78665e6aefb65a7085c7f67e13c  n_t.txt
[11/07/25]seed@VM:~/.../part3$ cat suffix.txt
wow this is a suffix T that im adding
[11/07/25]seed@VM:~/.../part3$
```

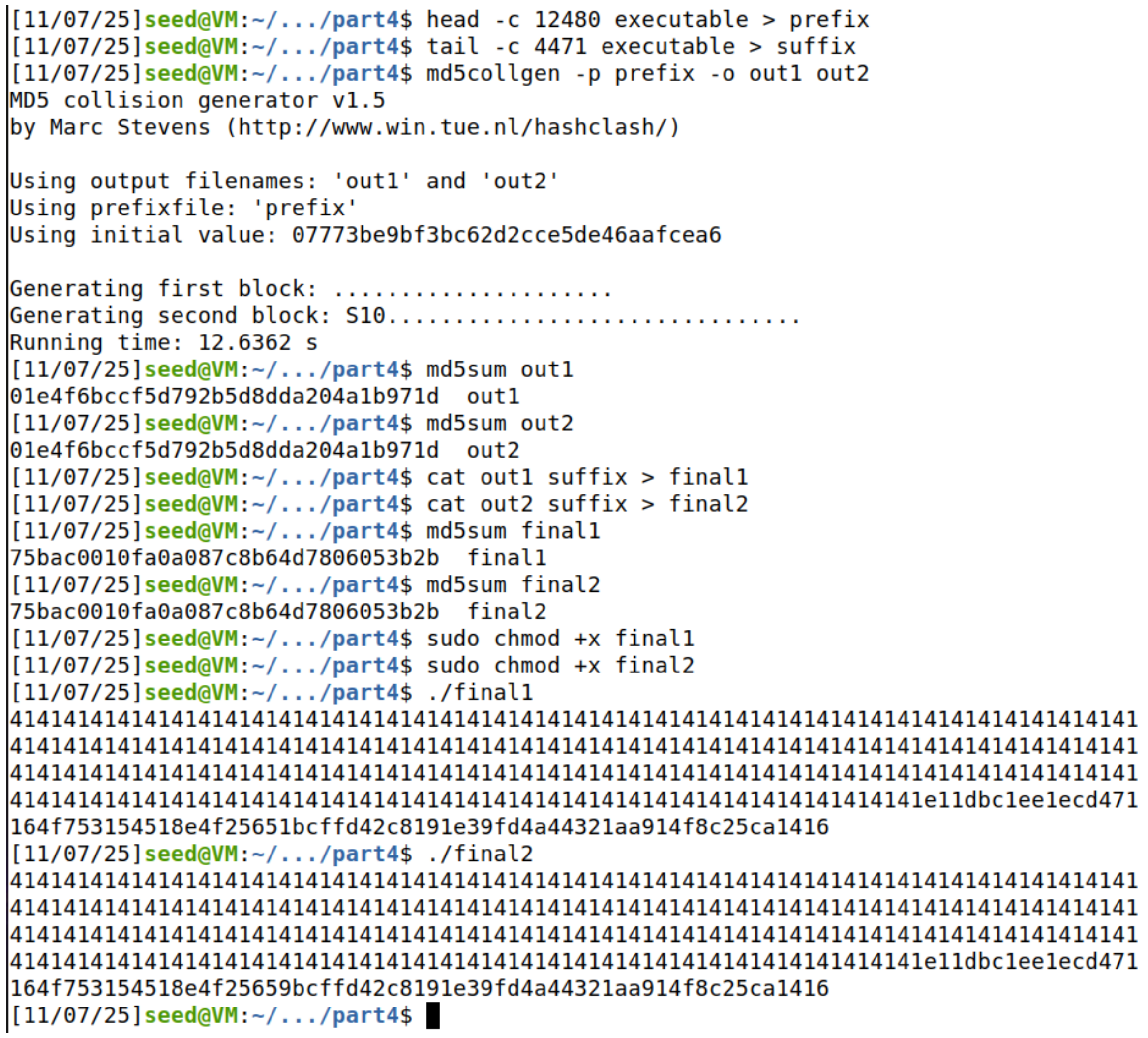
Observations: When a suffix T was added to the end of out1.bin and out2.bin, two different files with equal hashes, the resulting hash of the files with the added suffix was equal.

Task 4: Generating Two Executable Files with the Same MD5 Hash

A C program with searchable values was created and compiled. Then, bless was used to get a prefix divisible by 64, and a suffix of the compiled part of the program excluding some of the A values. Then md5collgen was used to create different executables with the same prefix. Lastly, different suffixes were appended.



```
[11/07/25]seed@VM:~/.../part4$ cat executable.c
#include <stdio.h>
#define LENGTH 200

unsigned char xyz[LENGTH]= {
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
};

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

[11/07/25]seed@VM:~/.../part4$ gcc executable.c -o executable
```

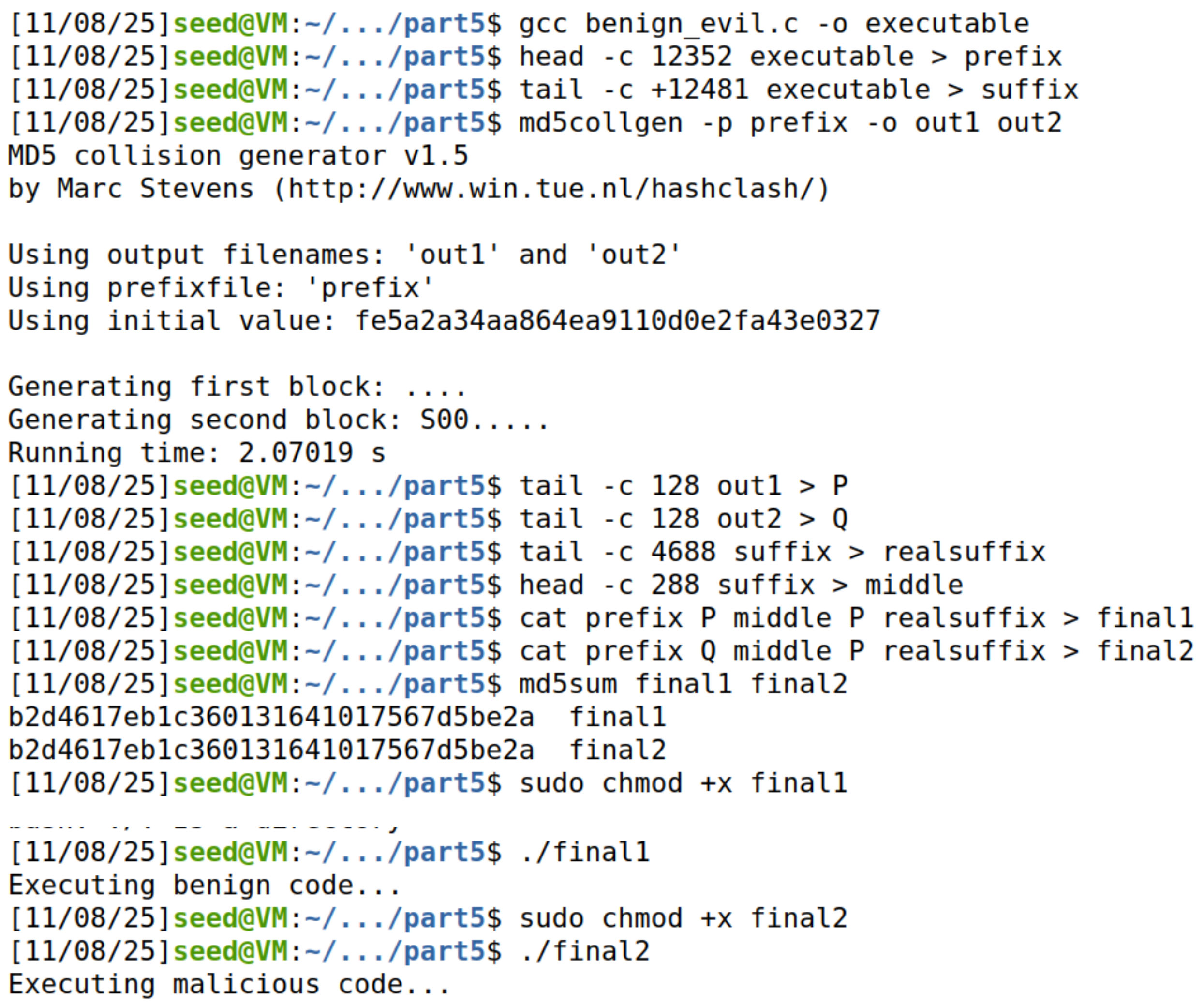


```
[11/07/25]seed@VM:~/.../part4$ head -c 12480 executable > prefix
[11/07/25]seed@VM:~/.../part4$ tail -c 4471 executable > suffix
[11/07/25]seed@VM:~/.../part4$ md5collgen -p prefix -o out1 out2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1' and 'out2'
Using prefixfile: 'prefix'
Using initial value: 07773be9bf3bc62d2cce5de46aafcea6

Generating first block: .....................
Generating second block: S10..............................
Running time: 12.6362 s
[11/07/25]seed@VM:~/.../part4$ md5sum out1
01e4f6bccf5d792b5d8dda204a1b971d  out1
[11/07/25]seed@VM:~/.../part4$ md5sum out2
01e4f6bccf5d792b5d8dda204a1b971d  out2
[11/07/25]seed@VM:~/.../part4$ cat out1 suffix > final1
[11/07/25]seed@VM:~/.../part4$ cat out2 suffix > final2
[11/07/25]seed@VM:~/.../part4$ md5sum final1
75bac0010fa0a087c8b64d7806053b2b  final1
[11/07/25]seed@VM:~/.../part4$ md5sum final2
75bac0010fa0a087c8b64d7806053b2b  final2
[11/07/25]seed@VM:~/.../part4$ sudo chmod +x final1
[11/07/25]seed@VM:~/.../part4$ sudo chmod +x final2
[11/07/25]seed@VM:~/.../part4$ ./final1
414141414141414141414141414141414141414141414141414141414141414141414141414141414141
414141414141414141414141414141414141414141414141414141414141414141414141414141414141
414141414141414141414141414141414141414141414141414141414141414141414141414141414141
41414141414141414141414141414141414141414141414141414141414141414141e11dbc1ee1ecd471
164f753154518e4f25651bcffd42c8191e39fd4a44321aa914f8c25ca1416
[11/07/25]seed@VM:~/.../part4$ ./final2
414141414141414141414141414141414141414141414141414141414141414141414141414141414141
414141414141414141414141414141414141414141414141414141414141414141414141414141414141
414141414141414141414141414141414141414141414141414141414141414141414141414141414141
41414141414141414141414141414141414141414141414141414141414141414141e11dbc1ee1ecd471
164f753154518e4f25659bcffd42c8191e39fd4a44321aa914f8c25ca1416
[11/07/25]seed@VM:~/.../part4$ 
```

Observations: The output was successfully hashed, and the 128 bytes were printed with the array.

Task 5: Making the Programs Behave Differently

An executable was parsed so that the prefix was divisible by 64 bytes and then a collision was generated with 128 bytes inside the first array. Then, the suffix of this hash was altered to insert one of the generated 128 bytes into both of the arrays at the same position as the first array. Since the original executable compares two equivalent arrays to output the different results, now the collision allows for one array to be equal and the other unequal, while maintaining equal hashes.



```
[11/08/25]seed@VM:~/.../part5$ gcc benign_evil.c -o executable
[11/08/25]seed@VM:~/.../part5$ head -c 12352 executable > prefix
[11/08/25]seed@VM:~/.../part5$ tail -c +12481 executable > suffix
[11/08/25]seed@VM:~/.../part5$ md5collgen -p prefix -o out1 out2
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1' and 'out2'
Using prefixfile: 'prefix'
Using initial value: fe5a2a34aa864ea9110d0e2fa43e0327

Generating first block: ....
Generating second block: S00.....
Running time: 2.07019 s
[11/08/25]seed@VM:~/.../part5$ tail -c 128 out1 > P
[11/08/25]seed@VM:~/.../part5$ tail -c 128 out2 > Q
[11/08/25]seed@VM:~/.../part5$ tail -c 4688 suffix > realsuffix
[11/08/25]seed@VM:~/.../part5$ head -c 288 suffix > middle
[11/08/25]seed@VM:~/.../part5$ cat prefix P middle P realsuffix > final1
[11/08/25]seed@VM:~/.../part5$ cat prefix Q middle P realsuffix > final2
[11/08/25]seed@VM:~/.../part5$ md5sum final1 final2
b2d4617eb1c360131641017567d5be2a  final1
b2d4617eb1c360131641017567d5be2a  final2
[11/08/25]seed@VM:~/.../part5$ sudo chmod +x final1
[11/08/25]seed@VM:~/.../part5$ ./final1
Executing benign code...
[11/08/25]seed@VM:~/.../part5$ sudo chmod +x final2
[11/08/25]seed@VM:~/.../part5$ ./final2
Executing malicious code...
```

Observations: The final executables generated the same hash and created different outputs in a successful collision attack.