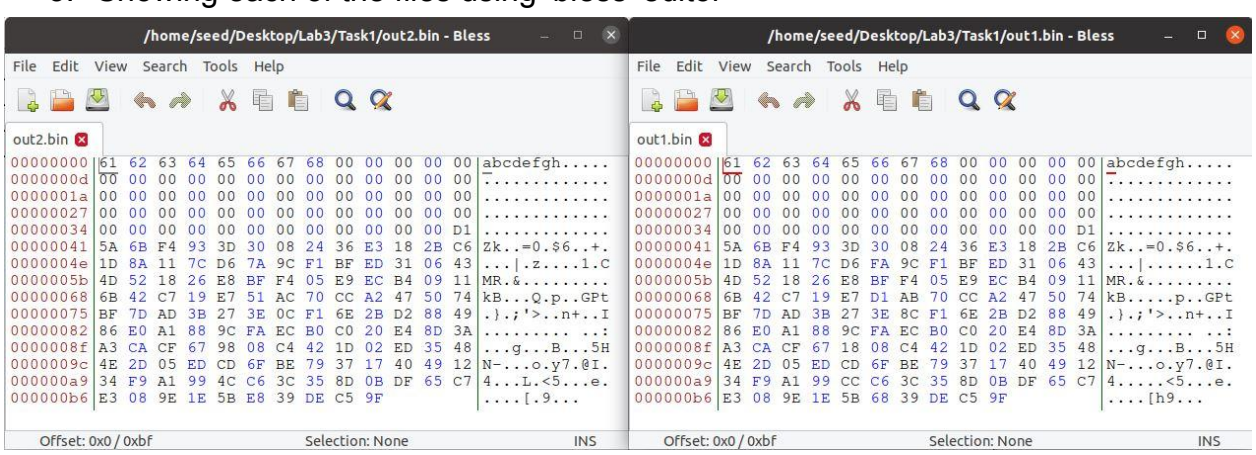
Task 1: Generating Two Different Files with the Same MD5 Hash

Initial few steps:

1. Creating files using prefix.txt whose value is “abcdefgh”.
2. Checking difference between them

```
seed@VM: ~/.../Task1
[06/22/22]seed@VM:~/.../Task1$ 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: a0b50a6e2edc375684f9575884b87364

Generating first block: .
Generating second block: W.............
Running time: 2.09926 s
[06/22/22]seed@VM:~/.../Task1$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[06/22/22]seed@VM:~/.../Task1$ md5sum out1.bin
c458c0654bb51636ea5c9e2a0404e731  out1.bin
[06/22/22]seed@VM:~/.../Task1$ md5sum out2.bin
c458c0654bb51636ea5c9e2a0404e731  out2.bin
```

3. Showing each of the files using ‘bless’ editor



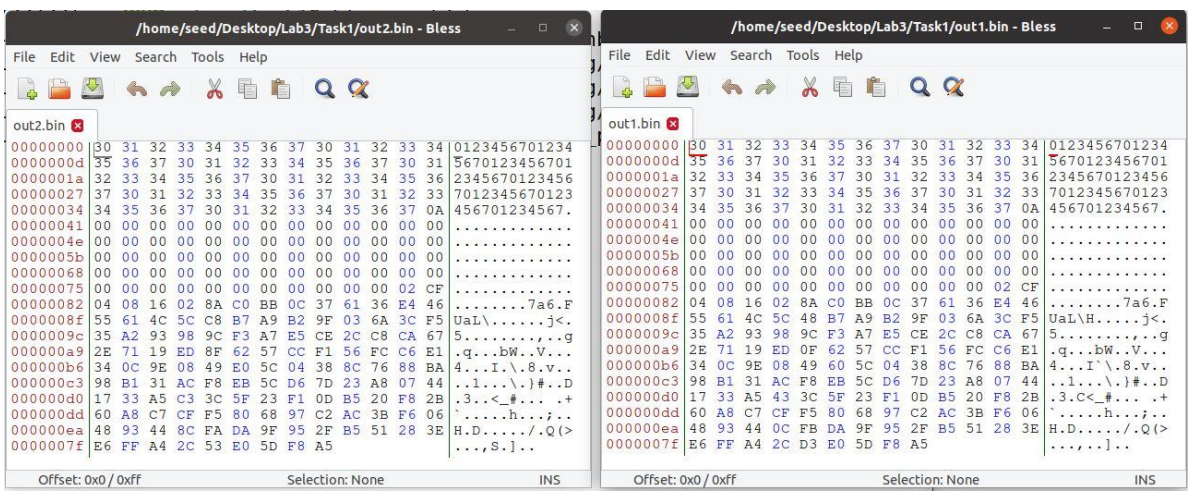
Ques1. When the prefix file is not in the multiple of 64, it will be padded with values such as '1' or '0' like the one above with only 5Bytes value.

Ques2.

Initial code :

```
[06/22/22]seed@VM:~/.../Task1$ 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: 1b6a290fd586f52bb2c2b14af5eedccd

Generating first block: ...............................
Generating second block: S10.....................
Running time: 51.2414 s
[06/22/22]seed@VM:~/.../Task1$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[06/22/22]seed@VM:~/.../Task1$ md5sum out1.bin
34f0c54f0741bab5ebd59da553dbceb8  out1.bin
[06/22/22]seed@VM:~/.../Task1$ md5sum out2.bin
34f0c54f0741bab5ebd59da553dbceb8  out2.bin
```

Bless editor :



No padding is observed amongst the two files.

Ques3.

```
01234........   01234........
.............   .............
.............   .............
.............   .............
............h   ............h
C...+.&0.....   C...+.&0.....
...m..C...MN|   ...m..C...MN|
q.C.(....L.{b   q.C.(....L.{b
.Oq..Ws3/...>   .Oq...r3/...>
[.O.Kq'......   [.O.Kq.......
.`.....UF{l.s   .`.....UF{l.s
........L.y..   ....G...L.y..
M|..=P?....gV   M|..=P?....gV
1...$....q.o.   1........q.o.
3k.S......      3k.S.j....
```

What we observe here, that prefix file and padding consist of same particular data. The only difference is of P and Q which are differences like ‘Ws’ and ‘r’

Task 2: Understanding MD5's Property

After applying the following steps:

1. Creating two files with MD5 hash
2. Checking whether they have the same Hash values.
3. Creating a new file
4. Concatenating the new file as a suffix
5. Checking whether the hash values changed

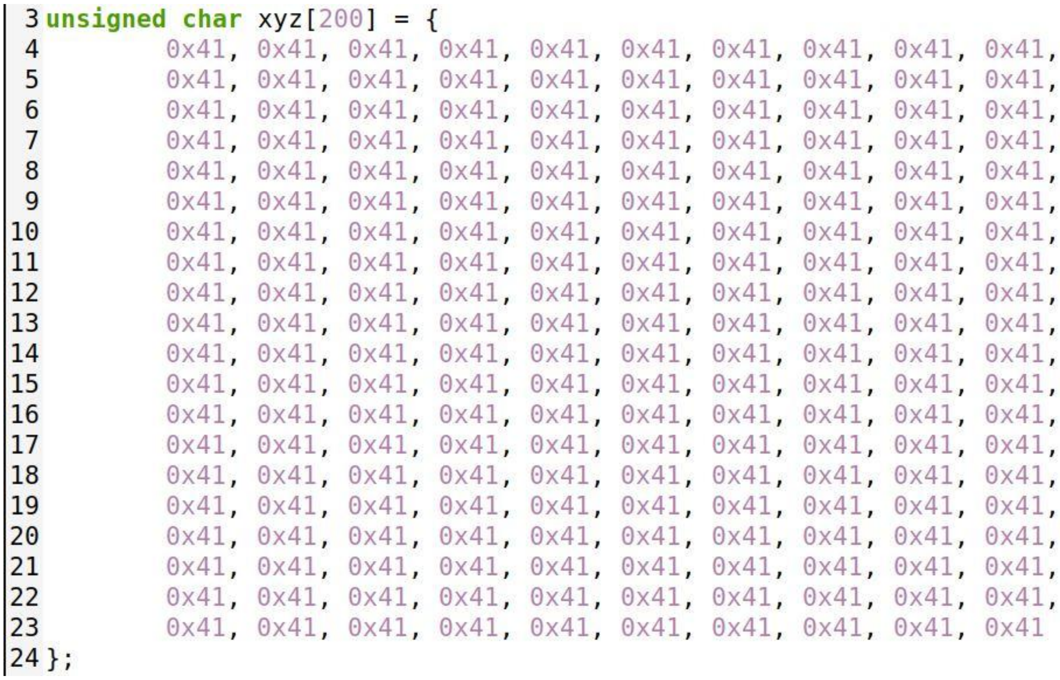
```
seed@VM: ~/.../Task2
[06/23/22]seed@VM:~/.../Task2$ echo -n "01234" > p1.txt
[06/23/22]seed@VM:~/.../Task2$ md5collgen -p p1.txt -o e1.txt e2.txt
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'e1.txt' and 'e2.txt'
Using prefixfile: 'p1.txt'
Using initial value: ed166457be6f75c4eb07fd002255f072

Generating first block: ....................
Generating second block: S11...
Running time: 29.3287 s
[06/23/22]seed@VM:~/.../Task2$ md5sum e1.txt e2.txt
8a0e7bbe37d6b4acebaeaaafb43fa307  e1.txt
8a0e7bbe37d6b4acebaeaaafb43fa307  e2.txt
[06/23/22]seed@VM:~/.../Task2$ echo -n "56" > file.txt
[06/23/22]seed@VM:~/.../Task2$ cat e1.txt file.txt > e1.txt
[06/23/22]seed@VM:~/.../Task2$ cat e2.txt file.txt > e2.txt
[06/23/22]seed@VM:~/.../Task2$ md5sum e1.txt e2.txt
9f61408e3afb633e50cdf1b20de6f466  e1.txt
9f61408e3afb633e50cdf1b20de6f466  e2.txt
```

From the above experiment we can conclude that **the hash value doesn't change after the addition of a suffix and this property holds true for MD5.**

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

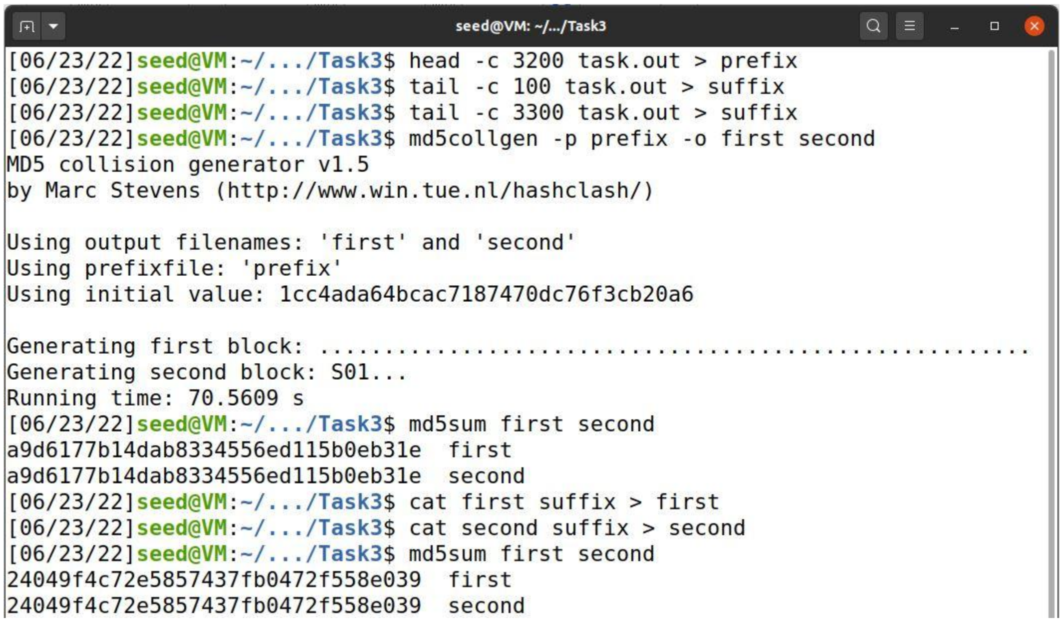
Applying the aforementioned steps:

1. Initializing the values of the array
2. Generating the output in the file 'Task.out'
3. Splitting the file 'Task.out' into 'prefix' and 'suffix'
4. Generating two hash file from prefix, 'first' and 'second'
5. Checking each of their values
6. Adding suffix in them
7. Checking their values



```
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,
        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
};
```

(step1)



```
[06/23/22]seed@VM:~/.../Task3$ head -c 3200 task.out > prefix
[06/23/22]seed@VM:~/.../Task3$ tail -c 100 task.out > suffix
[06/23/22]seed@VM:~/.../Task3$ tail -c 3300 task.out > suffix
[06/23/22]seed@VM:~/.../Task3$ md5collgen -p prefix -o first second
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'first' and 'second'
Using prefixfile: 'prefix'
Using initial value: 1cc4ada64bcac7187470dc76f3cb20a6

Generating first block: .......................................................
Generating second block: S01...
Running time: 70.5609 s
[06/23/22]seed@VM:~/.../Task3$ md5sum first second
a9d6177b14dab8334556ed115b0eb31e  first
a9d6177b14dab8334556ed115b0eb31e  second
[06/23/22]seed@VM:~/.../Task3$ cat first suffix > first
[06/23/22]seed@VM:~/.../Task3$ cat second suffix > second
[06/23/22]seed@VM:~/.../Task3$ md5sum first second
24049f4c72e5857437fb0472f558e039  first
24049f4c72e5857437fb0472f558e039  second
```

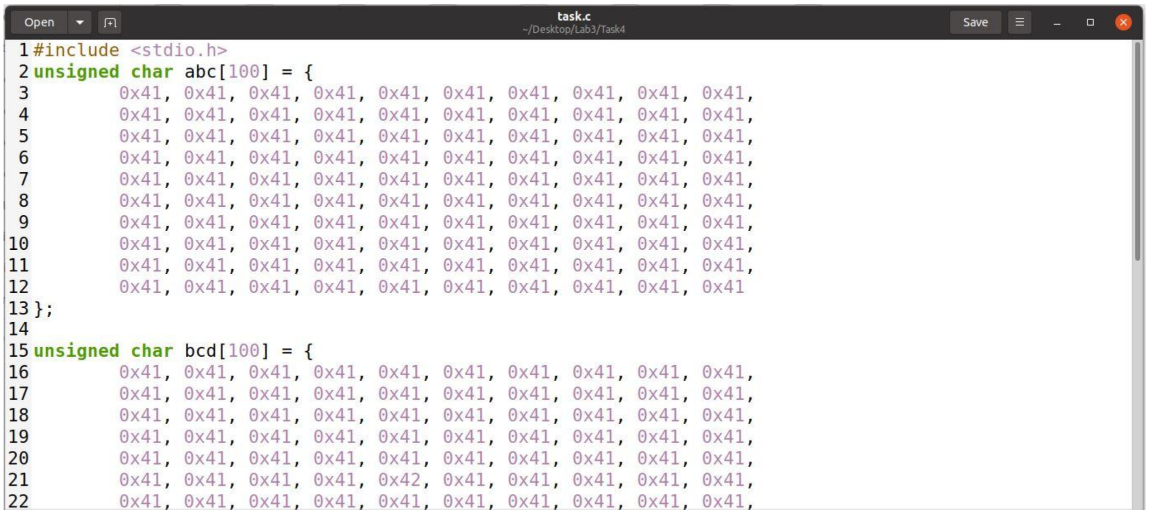
(step 2 to 7)

```
[06/23/22]seed@VM:~/.../Task3$ diff first second
Binary files first and second differ
```

Generation of two files from one executable file does lead with the **same MD5 hash values with the presence of different prefix and suffix values.**

Task 4: Making the Two Programs Behave Differently

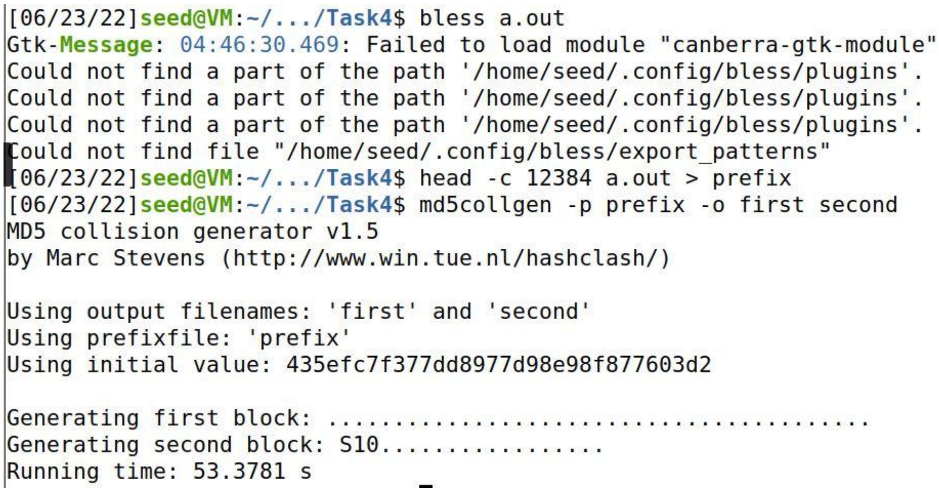
Making an executable C program:



```
#include <stdio.h>
unsigned char abc[100] = {
        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
};

unsigned char bcd[100] = {
        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, 0x42, 0x41, 0x41, 0x41, 0x41, 0x41,
        0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
```

```
int main()
{
        int i, Same =1;
        for (i=0; i<200; i++){
                if(abc[i] != bcd[i]){
                        Same = 0;
                }
        }

        if(Same){
                printf("Something malicious");
        }
        else{
                printf("All good");
                }
        printf("\n");
}
```

Making Files with same MD5 hash codes, but resulting in different outcomes.



```
[06/23/22]seed@VM:~/.../Task4$ bless a.out
Gtk-Message: 04:46:30.469: Failed to load module "canberra-gtk-module"
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find a part of the path '/home/seed/.config/bless/plugins'.
Could not find file "/home/seed/.config/bless/export_patterns"
[06/23/22]seed@VM:~/.../Task4$ head -c 12384 a.out > prefix
[06/23/22]seed@VM:~/.../Task4$ md5collgen -p prefix -o first second
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'first' and 'second'
Using prefixfile: 'prefix'
Using initial value: 435efc7f377dd8977d98e98f877603d2

Generating first block: .........................................
Generating second block: S10................
Running time: 53.3781 s
```

Combining them with different suffixes, and making them do malicious activities.

```
seed@VM: ~/.../Task4
[06/25/22]seed@VM:~/.../Task4$ tail -c +12532 a.out > badCode
[06/25/22]seed@VM:~/.../Task4$ tail -c +12533 a.out > goodCode
[06/25/22]seed@VM:~/.../Task4$ cat first >> badCode
[06/25/22]seed@VM:~/.../Task4$ cat second >> goodCode
[06/25/22]seed@VM:~/.../Task4$ chmod +x first second
[06/25/22]seed@VM:~/.../Task4$ ./first
Something malicious
[06/25/22]seed@VM:~/.../Task4$ ./second
All good
```

Even though they have the MD5 hash values, simple difference in suffix makes them behave differently and possibly even maliciously.