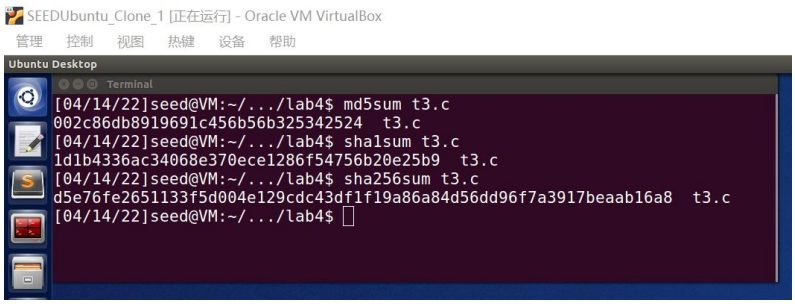
Lab 4

58119125 JiangZhuoyang

1. Task 1: Calculate MD5, SHA-1 and SHA-256 hash values of the file t3.c

Solution:



-Q1: What is the MD5 hash value of the file t3.c?

Answer:

002c86db8919691c456b56b325342524

-Q2: What is the SHA-1 hash value of the file t3.c?

Answer:

1d1b4336ac34068e370ece1286f54756b20e25b9

-Q3: What is the SHA-256 hash value of the file t3.c?

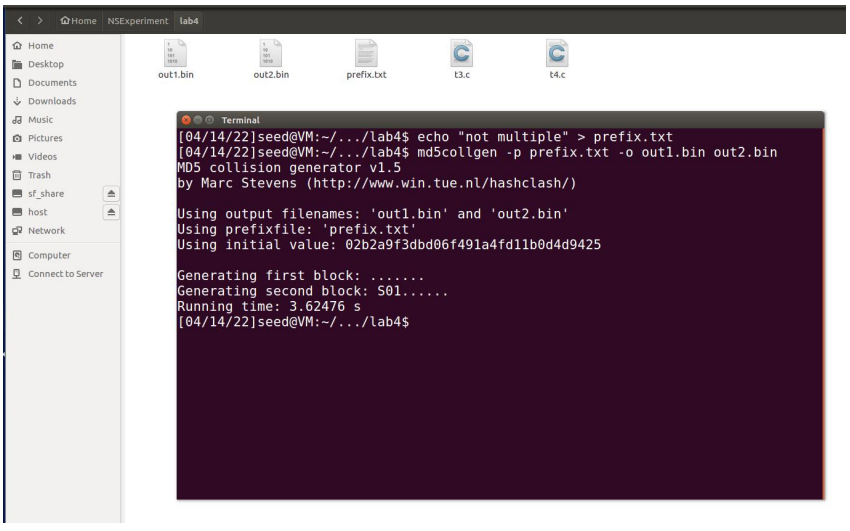
Answer:

d5e76fe2651133f5d004e129cdc43df1f19a86a84d56dd96f7a3917beaab16a8

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

Solution:

Step1: Create a 'prefix.txt' which is not multiple of 64 and Generate 'out1.bin' and 'out2.bin'



Step2: Compare them to find that they are differ. But their md5-hash values are the same.

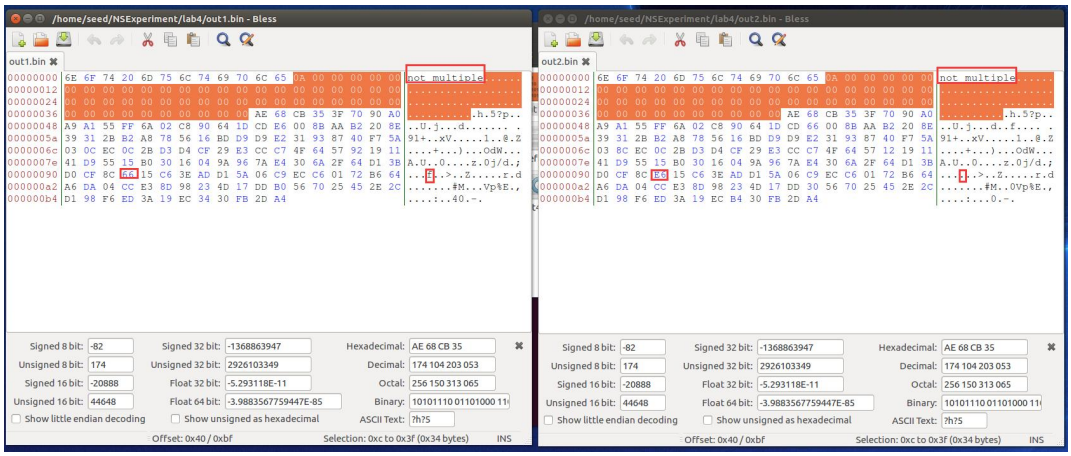

```
Terminal
[04/14/22]seed@VM:~/.../lab4$ echo "not multiple" > prefix.txt
[04/14/22]seed@VM:~/.../lab4$ md5collgen -p prefix.txt -o out1.bin out2.bin
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'out1.bin' and 'out2.bin'
Using prefixfile: 'prefix.txt'
Using initial value: 02b2a9f3dbd06f491a4fd11b0d4d9425

Generating first block: .......
Generating second block: S01......
Running time: 3.62476 s
[04/14/22]seed@VM:~/.../lab4$ diff out1.bin out2.bin
Binary files out1.bin and out2.bin differ
[04/14/22]seed@VM:~/.../lab4$ md5sum out1.bin
5df12a58546350dd94f19f9b08b424c0  out1.bin
[04/14/22]seed@VM:~/.../lab4$ md5sum out2.bin
5df12a58546350dd94f19f9b08b424c0  out2.bin
[04/14/22]seed@VM:~/.../lab4$
```

Step3: Compare them in the Bless Hex Editor

-Q4. If the length of your prefix file is not multiple of 64, what is going to happen?

Answer:

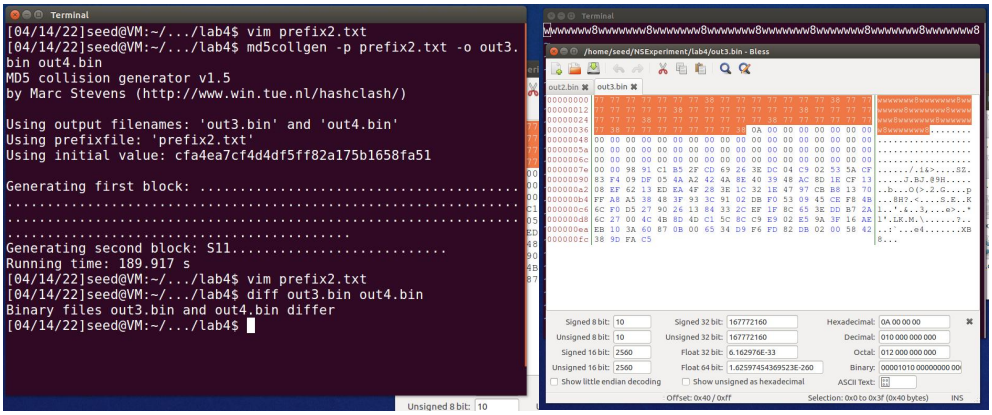
If the prefix file length is not a multiple of 64, Md5collgn will automatically fill in 0 until the prefix length meets the multiple of 64.



-Q5. Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens.

Answer:

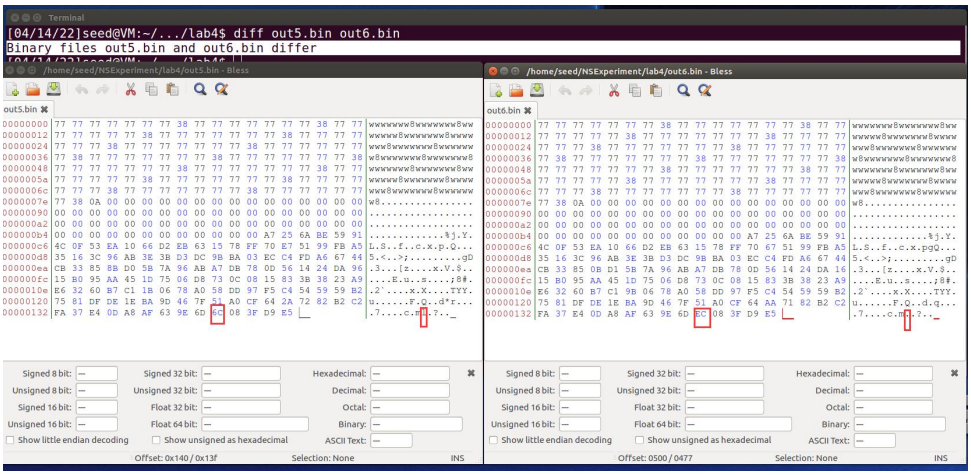
If the prefix file length is exactly of 64, Md5collgn will not fill 0, and 64 bytes is the prefix.



-Q6. Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.

Answer:

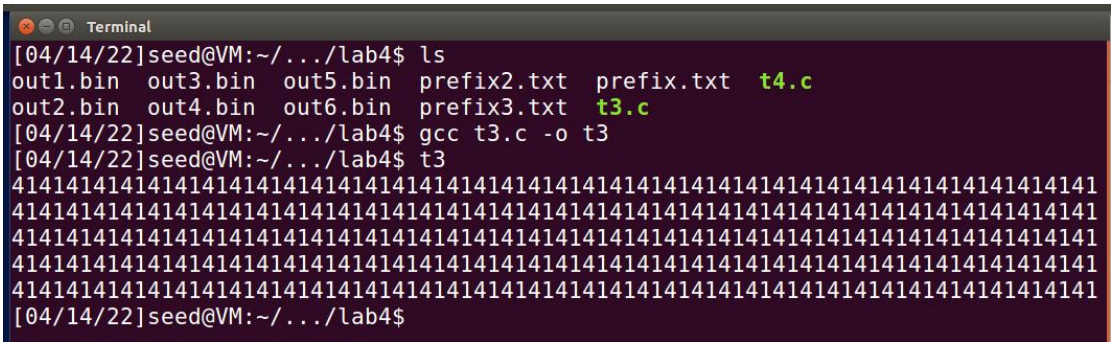
The data of the two output files are different, but not completely different.



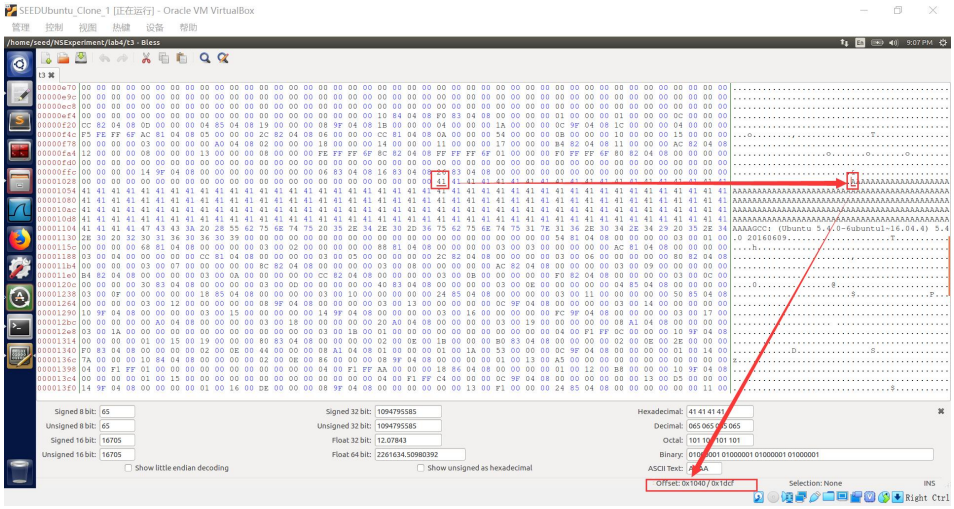
3. Task 3: Generating Two Executable Files with the Same MD5 Hash

Solution:

Step1: Compile file t3.c and generate the executable file t3.

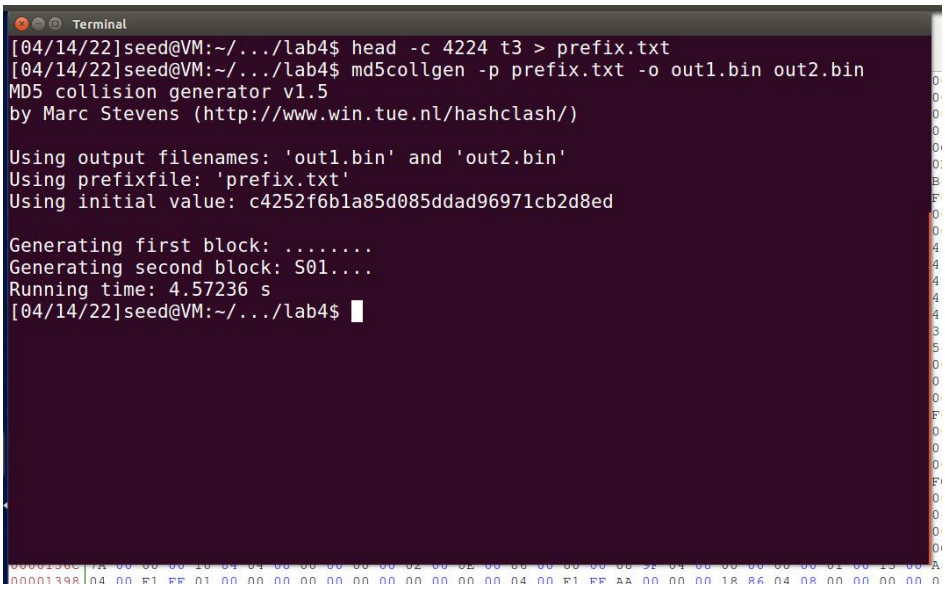


Step2: Use bless to view the contents of array a in the binary file. We find that the starting position of the array is 0x00001040. There are 0x1040 bytes in front of it, which is already a multiple of 0x40. We can use the first 0x1080 bytes (also a multiple of 0x40) as a prefix;

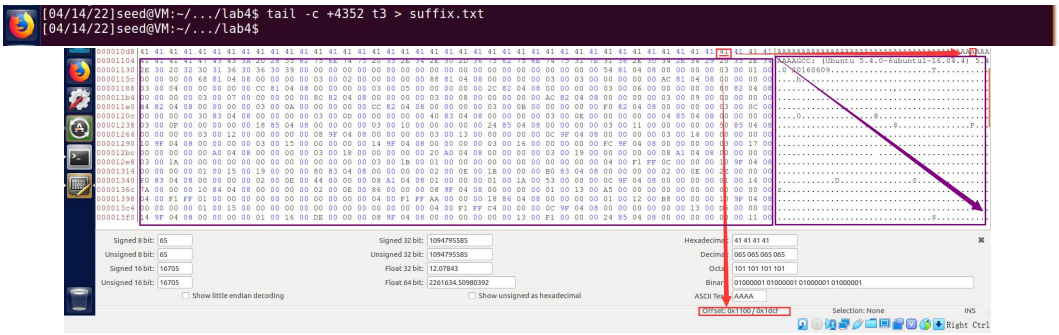


Step3: Use the head command to take the first 0x1080 (4224) bytes of the executable t3 as the prefix, and then use md5collgen to generate two files with the same hash value:

out1.bin and out2.bin



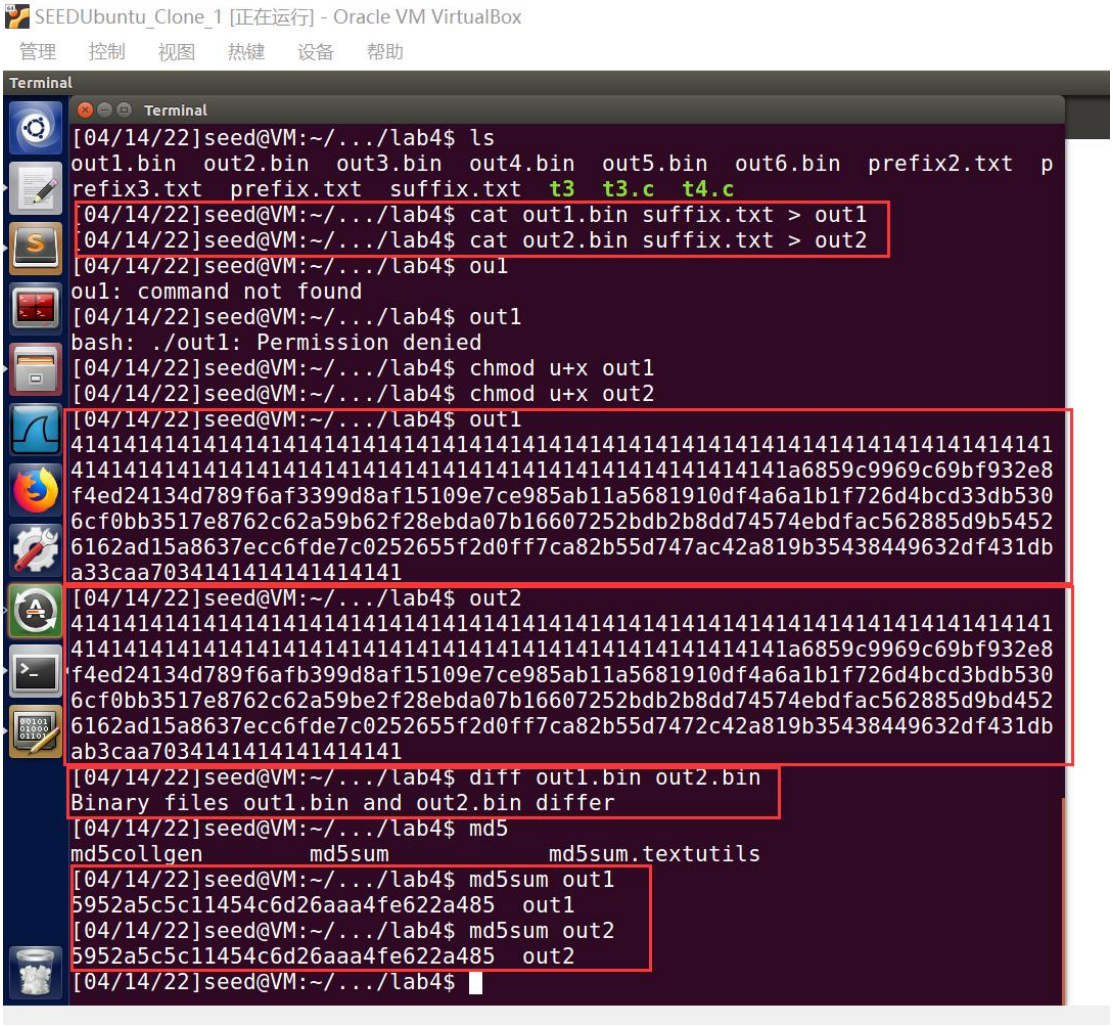
Step4: Leave the 128-byte region (0x0080) of the array, and use the tail command to take the rest of the executable file (from 0x1100 to the end) as the suffix file;



Step5: Add the suffix file to the generated out1 Bin and out2 After bin, we can get two executable files out1 and out2. Run out1 and out2 and calculate the hash value.

We can find that their output results are different, indicating that these are two different programs.

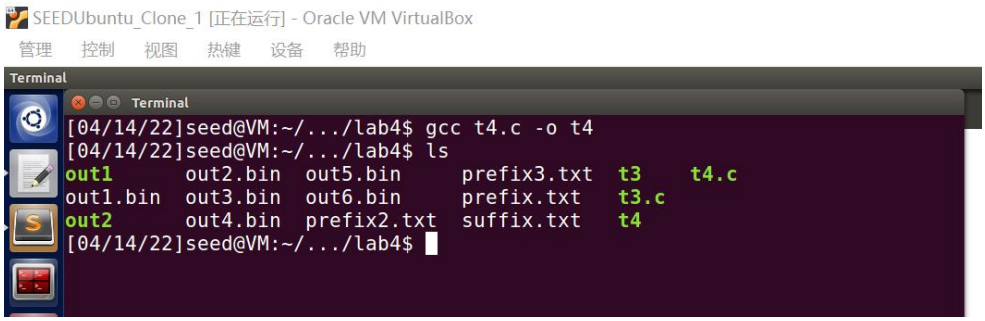
The MD5 hash values of the two programs are the same. We successfully constructed two executable files with the same hash value but different output.



Task 4: Making the Two Programs Behave Differently

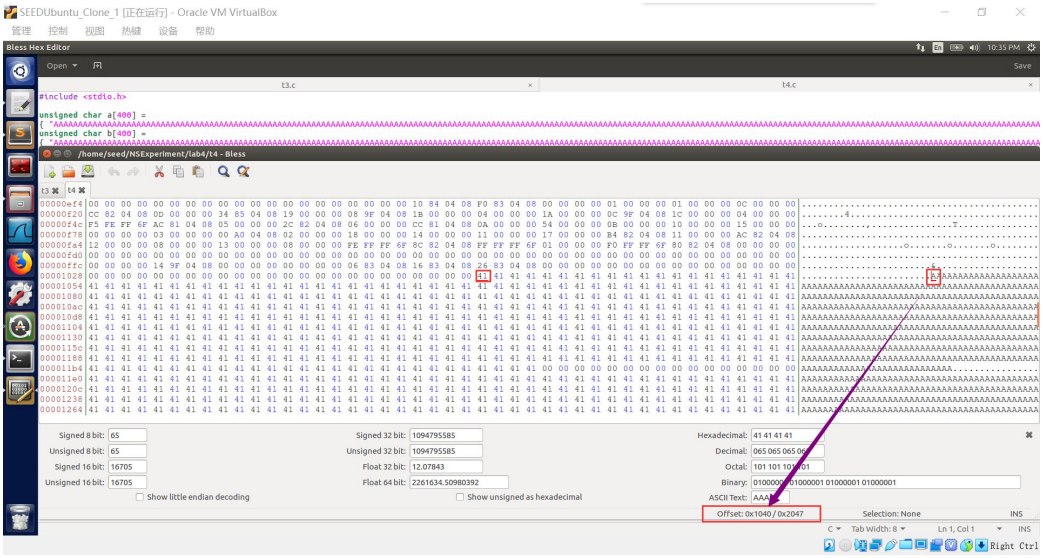
Solution:

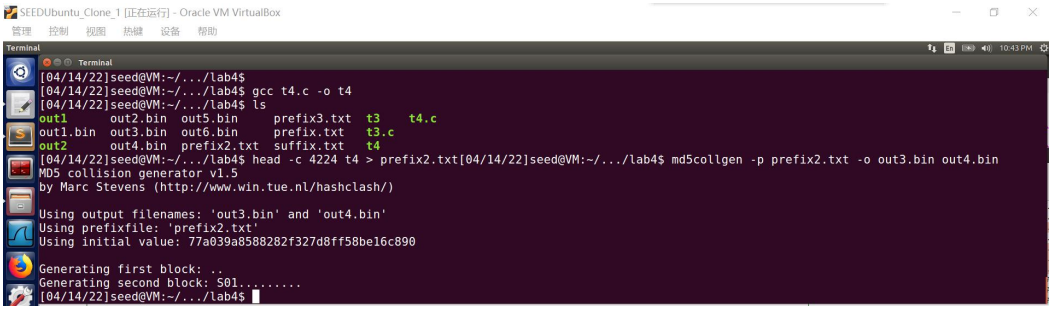
Step1: Compile file t4.c and generate the executable file t4



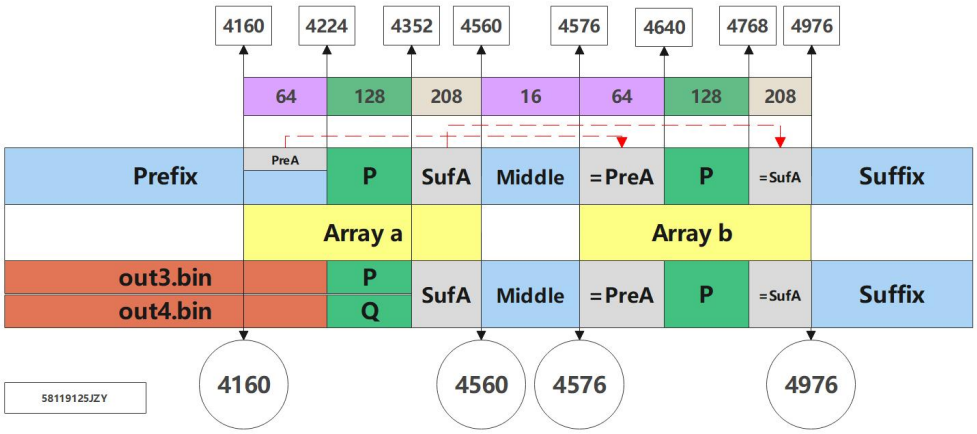
Step2: In bless we can find that the array a is begin at 0x1040(4160). So use the head command to take the first 0x1080(4224) bytes of the executable t4 as the prefix, and then use md5collgen to generate two files with the same hash value:

Out3.bin and out4.bin





Step3: Use the head and tail commands to intercept binary files into the slices below:




```
[04/14/22]seed@VM:~/.../lab4$ head -c 4560 t4 > tmp
[04/14/22]seed@VM:~/.../lab4$ tail -c +4354 tmp > sufA
[04/14/22]seed@VM:~/.../lab4$ tail -c +4561 t4 > temp2
[04/14/22]seed@VM:~/.../lab4$ head -c 16 temp2 > middle
[04/14/22]seed@VM:~/.../lab4$ tail -c +4161 out3.bin > preAP
[04/14/22]seed@VM:~/.../lab4$ tail -c +4977 t4 >suffix
```

Step4: Splicing with cat command:

```
[04/15/22]seed@VM:~/.../lab4$ cat out3.bin sufA middle preAP sufA suffix >out3
[04/15/22]seed@VM:~/.../lab4$ cat out4.bin sufA middle preAP sufA suffix >out4
[04/15/22]seed@VM:~/.../lab4$
```

Step5: Run out3 and out4 and calculate hash values. Success.

```
[04/15/22]seed@VM:~/.../lab4$ du -b out3 out4 t4
8264    out3
8264    out4
8264    t4
[04/15/22]seed@VM:~/.../lab4$ out3
This would run the safe code and display the intended behaviour
[04/15/22]seed@VM:~/.../lab4$ out4
This is where malicious code would be run
[04/15/22]seed@VM:~/.../lab4$ md5sum out3 out4
b204310963d373c3f17ecef5b19babeb  out3
b204310963d373c3f17ecef5b19babeb  out4
```