

Lab 8 report

309552026

鄭偉丞

Crash:

一共出現了兩個問題，將出現 crash 的 input 餵給 ASAN 插樁好的程式後，兩者出現的 error message 相同。

```
weicheng@ubuntu:~/st/st_lab8/Lab_8$ ./bmpgrayscale ../out_1/default/crashes/id:000000,sig:06,src:000005,time:264746,op:int16,pos:12,val:+0 /dev/null
==32397==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x602000000011 at pc 0x00000049bdfc bp 0x7fffffffdeb0 sp 0x7fffffff660
WRITE of size 1 at 0x602000000011 thread T0
#0 0x49bdfb in __interceptor_fread.part.47 (/home/weicheng/st/st_lab8/Lab_8/bmpgrayscale+0x49bdfb)
#1 0x514013 in bmp_transform /home/weicheng/st/st_lab8/Lab_8/bmp_lib.c:158:9
#2 0x51291e in main /home/weicheng/st/st_lab8/Lab_8/bmpgrayscale.c:18:9
#3 0x7ffff7c55564 in __libc_start_main csu/../csu/libc-start.c:332:16
#4 0x41a3dd in _start (/home/weicheng/st/st_lab8/Lab_8/bmpgrayscale+0x41a3dd)

0x602000000011 is located 0 bytes to the right of 1-byte region [0x602000000010,0x602000000011)
allocated by thread T0 here:
#0 0x4da290 in malloc (/home/weicheng/st/st_lab8/Lab_8/bmpgrayscale+0x4da290)
#1 0x513dc4 in bmp_transform /home/weicheng/st/st_lab8/Lab_8/bmp_lib.c:132:17
#2 0x51291e in main /home/weicheng/st/st_lab8/Lab_8/bmpgrayscale.c:18:9
#3 0x7ffff7c55564 in __libc_start_main csu/../csu/libc-start.c:332:16

SUMMARY: AddressSanitizer: heap-buffer-overflow (/home/weicheng/st/st_lab8/Lab_8/bmpgrayscale+0x49bdfb) in __interceptor_fread.part.47
Shadow bytes around the buggy address:
 0x0c047fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0c047fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0c047fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0c047fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x0c047fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c047fff8000: fa fa 01 fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c047fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone: fa
Freed heap region: fd
Stack left redzone: f1
Stack mid redzone: f2
Stack right redzone: f3
Stack after return: f5
Stack use after scope: f8
Global redzone: f9
Global init order: f6
Poisoned by user: f7
Container overflow: fc
Array cookie: ac
Intra object redzone: bb
ASAN internal: fe
Left alloca redzone: ca
Right alloca redzone: cb
==32397==ABORTING
```

可以看出是 line 132 和 line 158 有問題。

poc0:

使用 gdb 分析之後發現這兩個 crash,其中 poc0 會有 SIGABRT, 這個 signal 主要是由於多次 free 同一 memory, 或者沒有 malloc 任何空間, 卻用了 free。print image_size 後發現, poc0 的 image_size 為 0, 因此 malloc 沒有分配空間給他。

poc1:

poc1 的 image_size 不為 0, 也沒有 SIGABRT 出現, crash 的原因是 image_size 的計算方式是高*寬*3, 透過 fuzz 改動過的 input file 會讓實際 image_size 遠大於計算出來的值, 使得分配的記憶體不夠用, 所以 line 158 在 read 的時候, 還沒讀到 eof 就 crash 了。

Fix:

解決方法是在計算完 image_size 後檢查大小，若 ≤ 0 ，則為不合法輸入，這樣就可以解決 poc0。

```
131 // Fix
132 if (image_size <= 0){
133     printf("Wrong image_size.\n");
134     return 1;
135 }
136 // Fix end
```

poc1 則是在 read 的時候，檢查是否有讀超過分配的空間，超過則停止讀取檔案。

```
160 while (1)
161 {
162     // Fix
163     if (feof(bmpfile) || idx >= image_size)
164         break;
```