bit2spr vulnerability discovery

0x1 bit2spr introduction

This bit2spr converts bitmaps in X-bitmap format to the format used by the sprite package.

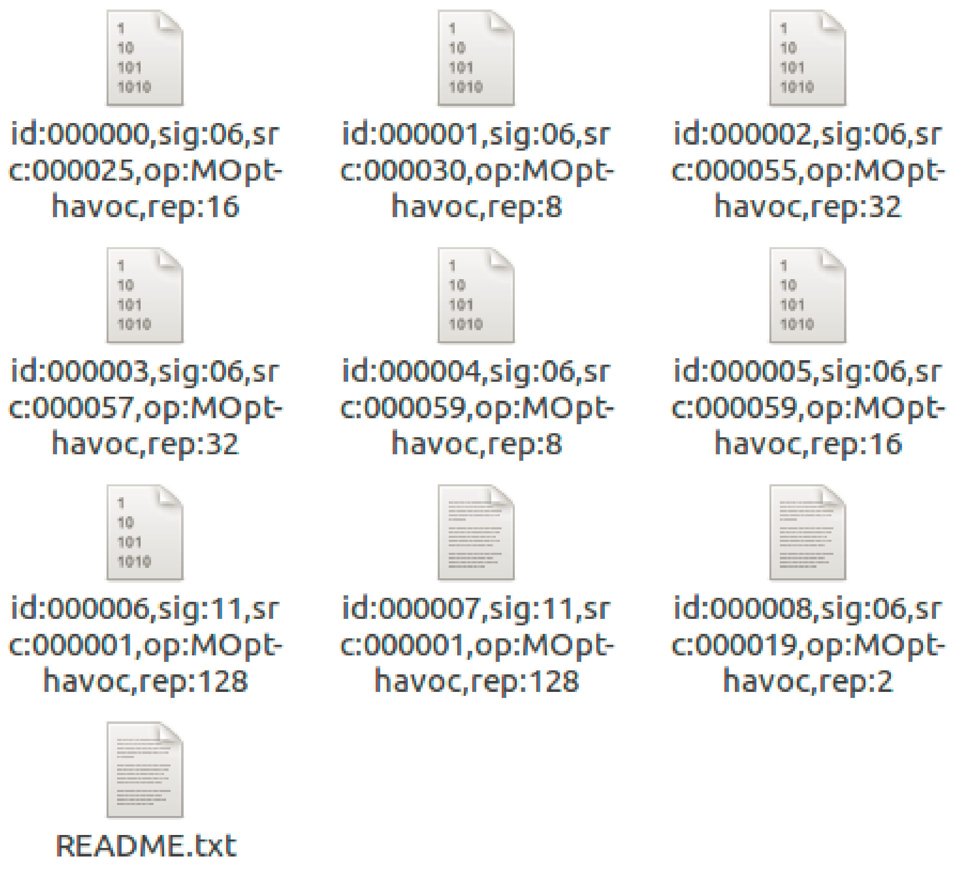
bit2spr is a ctan graphics package, widely used in texlive, ctex.

0x2 Fuzzing results

By using the modified AFL tools in 24hs, we found crashes with the cmd:

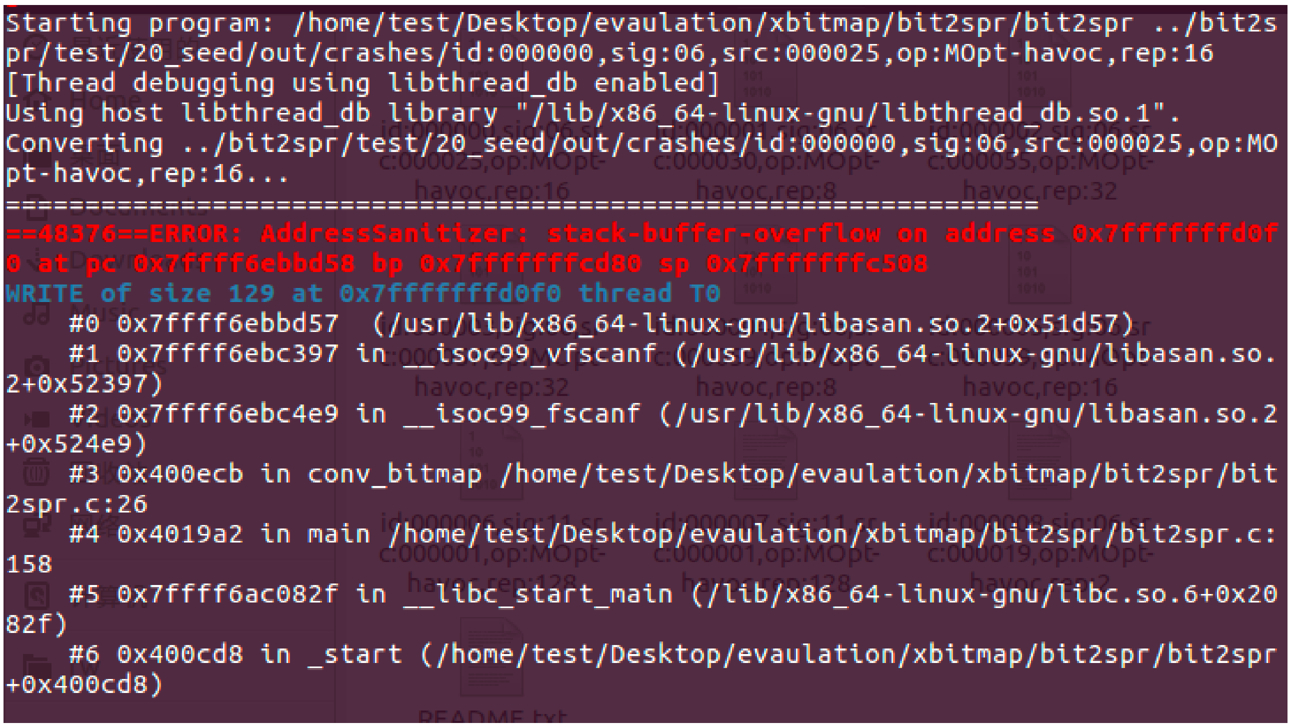
```
-i /fuzz_test/in -o /fuzz_test/out bit2spr @@ /dev/null
```

some poc can be produced in the output/crashes/

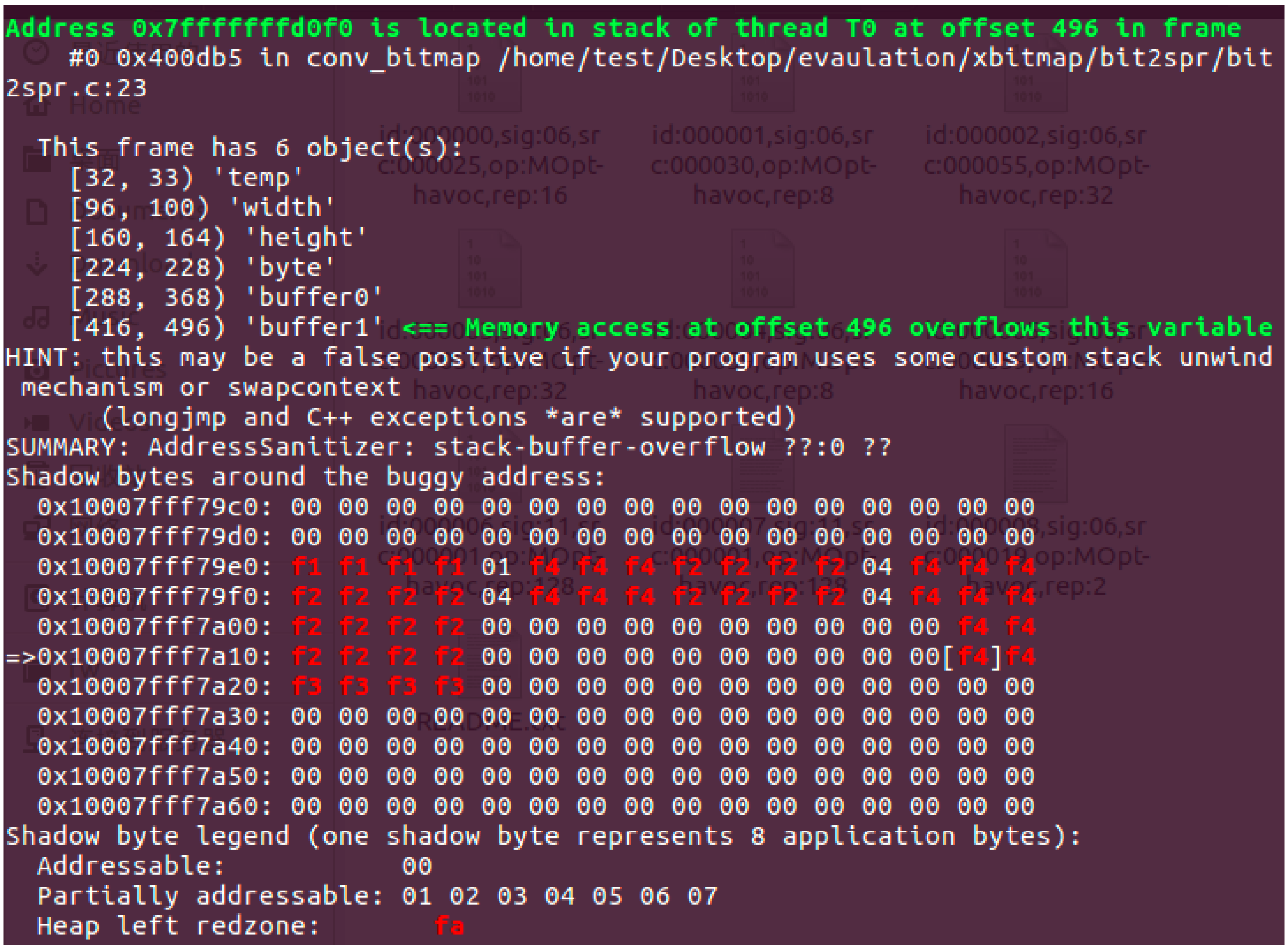


```
id:000000,sig:06,src:000025,op:MOpt-havoc,rep:16
id:000001,sig:06,src:000030,op:MOpt-havoc,rep:8
id:000002,sig:06,src:000055,op:MOpt-havoc,rep:32
id:000003,sig:06,src:000057,op:MOpt-havoc,rep:32
id:000004,sig:06,src:000059,op:MOpt-havoc,rep:8
id:000005,sig:06,src:000059,op:MOpt-havoc,rep:16
id:000006,sig:11,src:000001,op:MOpt-havoc,rep:128
id:000007,sig:11,src:000001,op:MOpt-havoc,rep:128
id:000008,sig:06,src:000019,op:MOpt-havoc,rep:2
README.txt
```

Add the address sanitizer option when compiling bit2spr using gcc, and then run the sample under the crashes folder



```
Starting program: /home/test/Desktop/evaulation/xbitmap/bit2spr/bit2spr ../bit2s
pr/test/20_seed/out/crashes/id:000000,sig:06,src:000025,op:MOpt-havoc,rep:16
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Converting ../bit2spr/test/20_seed/out/crashes/id:000000,sig:06,src:000025,op:MO
pt-havoc,rep:16...
=================================================================
==48376==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7fffffffd0f
0 at pc 0x7ffff6ebbd58 bp 0x7fffffffcd80 sp 0x7fffffffc508
WRITE of size 129 at 0x7fffffffd0f0 thread T0
    #0 0x7ffff6ebbd57  (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x51d57)
    #1 0x7ffff6ebc397 in __isoc99_vfscanf (/usr/lib/x86_64-linux-gnu/libasan.so.
2+0x52397)
    #2 0x7ffff6ebc4e9 in __isoc99_fscanf (/usr/lib/x86_64-linux-gnu/libasan.so.2
+0x524e9)
    #3 0x400ecb in conv_bitmap /home/test/Desktop/evaulation/xbitmap/bit2spr/bit
2spr.c:26
    #4 0x4019a2 in main /home/test/Desktop/evaulation/xbitmap/bit2spr/bit2spr.c:
158
    #5 0x7ffff6ac082f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x20
82f)
    #6 0x400cd8 in _start (/home/test/Desktop/evaulation/xbitmap/bit2spr/bit2spr
+0x400cd8)
```

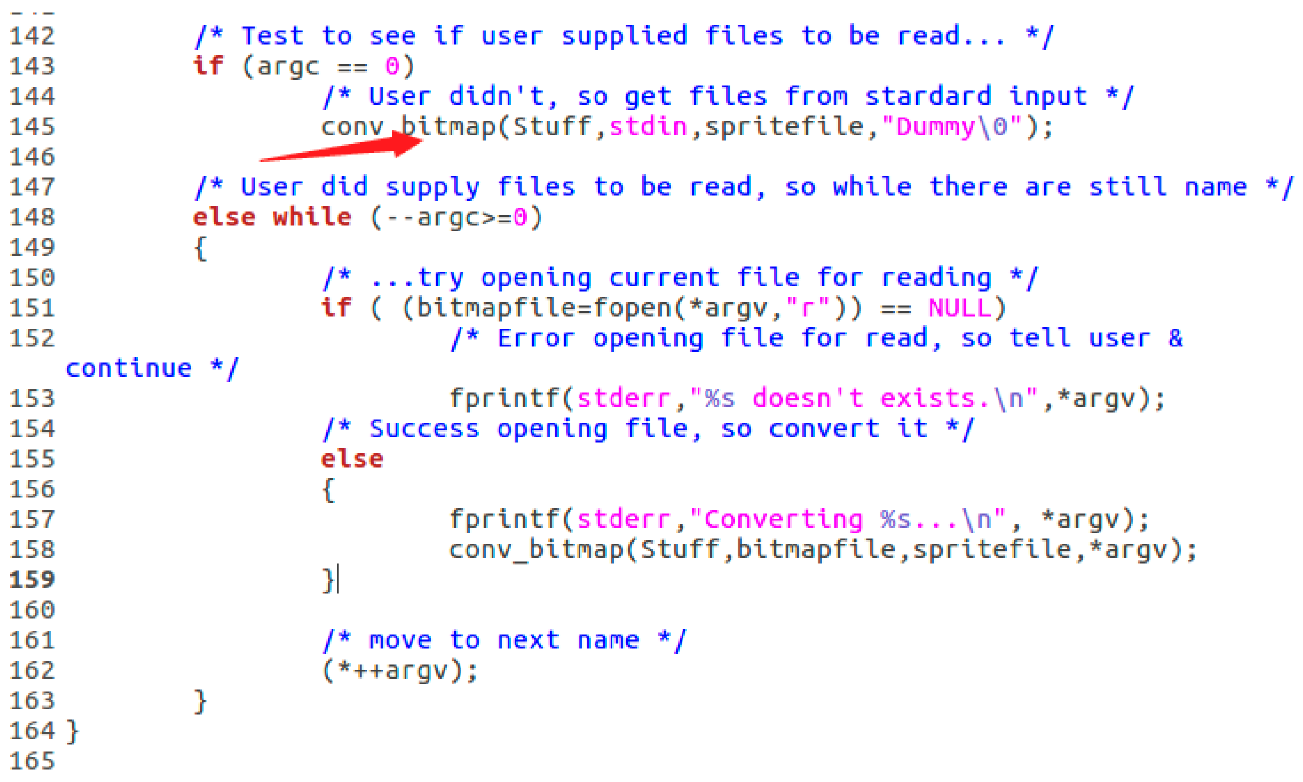


```
Address 0x7fffffffd0f0 is located in stack of thread T0 at offset 496 in frame
    #0 0x400db5 in conv_bitmap /home/test/Desktop/evaulation/xbitmap/bit2spr/bit
2spr.c:23

  This frame has 6 object(s):
    [32, 33) 'temp'
    [96, 100) 'width'
    [160, 164) 'height'
    [224, 228) 'byte'
    [288, 368) 'buffer0'
    [416, 496) 'buffer1' <== Memory access at offset 496 overflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind
 mechanism or swapcontext
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow ??:0 ??
Shadow bytes around the buggy address:
  0x10007fff79c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10007fff79d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10007fff79e0: f1 f1 f1 f1 01 f4 f4 f4 f2 f2 f2 f2 04 f4 f4 f4
  0x10007fff79f0: f2 f2 f2 f2 04 f4 f4 f4 f2 f2 f2 f2 04 f4 f4 f4
  0x10007fff7a00: f2 f2 f2 f2 00 00 00 00 00 00 00 00 00 00 f4 f4
=>0x10007fff7a10: f2 f2 f2 f2 00 00 00 00 00 00 00 00 00 00[f4]f4
  0x10007fff7a20: f3 f3 f3 f3 00 00 00 00 00 00 00 00 00 00 00 00
  0x10007fff7a30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10007fff7a40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10007fff7a50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x10007fff7a60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
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
```

0x3 Cause of vulnerability

The conv_bitmap function does not limit the size of incoming local variables, causing local variables buffer0, buffer1 stack overflow.

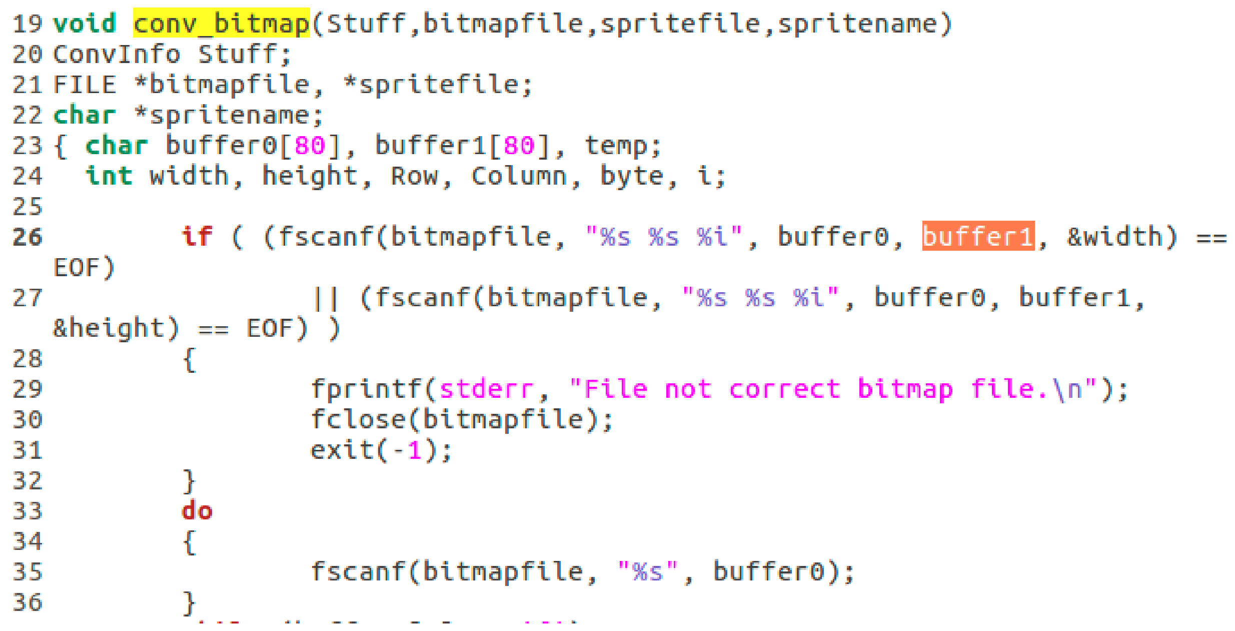


```
        /* Test to see if user supplied files to be read... */
        if (argc == 0)
                /* User didn't, so get files from stardard input */
                conv_bitmap(Stuff,stdin,spritefile,"Dummy\0");

        /* User did supply files to be read, so while there are still name */
        else while (--argc>=0)
        {
                /* ...try opening current file for reading */
                if ( (bitmapfile=fopen(*argv,"r")) == NULL)
                        /* Error opening file for read, so tell user & continue */
                        fprintf(stderr,"%s doesn't exists.\n",*argv);
                /* Success opening file, so convert it */
                else
                {
                        fprintf(stderr,"Converting %s...\n", *argv);
                        conv_bitmap(Stuff,bitmapfile,spritefile,*argv);
                }

                /* move to next name */
                (*++argv);
        }
}

```



```
void conv_bitmap(Stuff,bitmapfile,spritefile,spritename)
ConvInfo Stuff;
FILE *bitmapfile, *spritefile;
char *spritename;
{ char buffer0[80], buffer1[80], temp;
  int width, height, Row, Column, byte, i;

        if ( (fscanf(bitmapfile, "%s %s %i", buffer0, buffer1, &width) == EOF)
                || (fscanf(bitmapfile, "%s %s %i", buffer0, buffer1, &height) == EOF) )
        {
                fprintf(stderr, "File not correct bitmap file.\n");
                fclose(bitmapfile);
                exit(-1);
        }
        do
        {
                fscanf(bitmapfile, "%s", buffer0);
        }
```

0x4 Reproduce

The reproduction of the vulnerability can be performed in the poc folder, such as:

```
./bit2spr id ^% 000000, sig ^% 06, src ^% 000025, op ^% MOpt-
havoc, rep ^% 16
```