**A heap buffer overflow in libufdt**

1. We found this vulnerability by afl fuzz.
2. Can you identify exploitability?

Yes.

1. Overview

Libufdt can parse the .dtb file. And it can cause a heap buffer overflow issue in fdt\_get\_string() function. It can impact Android13.

1. Root Cause Analysis
   1. Crash backtrace

#./extract\_dtb\_asan id\:000004\,sig\:11\,src\:000005\,op\:havoc\,rep\:16 /dev/null

=================================================================

==4196==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x61300000f798 at pc 0x00000046f760 bp 0x7fffffffda10 sp 0x7fffffffd1c0

READ of size 1 at 0x61300000f798 thread T0

[New process 4233]

[Thread debugging using libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

process 4233 is executing new program: /usr/lib/llvm-6.0/bin/llvm-symbolizer

Error in re-setting breakpoint 1: Function "fdt\_get\_string" not defined.

[Thread debugging using libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

#0 0x46f75f in \_\_interceptor\_memchr.part.41 (/home/ubuntu/Desktop/libufdt/fuzzextract\_dtb/extract\_dtb\_asan+0x46f75f)

#1 0x534a42 in fdt\_get\_string /home/ubuntu/Desktop/libufdt/libfdt/fdt\_ro.c:81:6

#2 0x539e32 in fdt\_string\_eq\_ /home/ubuntu/Desktop/libufdt/libfdt/fdt\_ro.c:107:18

#3 0x539e32 in fdt\_get\_property\_namelen\_ /home/ubuntu/Desktop/libufdt/libfdt/fdt\_ro.c:409

#4 0x53a62a in fdt\_getprop\_namelen /home/ubuntu/Desktop/libufdt/libfdt/fdt\_ro.c:455:9

#5 0x54f213 in find\_and\_write\_dtb /home/ubuntu/Desktop/libufdt/tests/src/extract\_dtb.c:78:19

#6 0x54f7bc in extract\_dtbs /home/ubuntu/Desktop/libufdt/tests/src/extract\_dtb.c:115:17

#7 0x54ff2b in main /home/ubuntu/Desktop/libufdt/tests/src/extract\_dtb.c:158:13

#8 0x7ffff6e22c86 in \_\_libc\_start\_main /build/glibc-CVJwZb/glibc-2.27/csu/../csu/libc-start.c:310

#9 0x419fc9 in \_start (/home/ubuntu/Desktop/libufdt/fuzzextract\_dtb/extract\_dtb\_asan+0x419fc9)

Address 0x61300000f798 is a wild pointer.

SUMMARY: AddressSanitizer: heap-buffer-overflow (/home/ubuntu/Desktop/libufdt/fuzzextract\_dtb/extract\_dtb\_asan+0x46f75f) in \_\_interceptor\_memchr.part.41

Shadow bytes around the buggy address:

0x0c267fff9ea0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

0x0c267fff9eb0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

0x0c267fff9ec0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

0x0c267fff9ed0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

0x0c267fff9ee0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

=>0x0c267fff9ef0: fa fa fa[fa]fa fa fa fa fa fa fa fa fa fa fa fa

0x0c267fff9f00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

0x0c267fff9f10: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

0x0c267fff9f20: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

0x0c267fff9f30: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

0x0c267fff9f40: 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

==4196==ABORTING

* 1. **details analsis**

Enter from the main function of extract\_dtb.c, then go into extract\_dtbs() function

|  |
| --- |
| int main(int argc, char \*\*argv) {  ...  const char \*out\_dtb\_file = argv[2];  const char \*out\_image\_file = (argc > 3) ? argv[3] : NULL;  int ret = extract\_dtbs(in\_file, out\_dtb\_file, out\_image\_file);  return ret;  } |

and finally cause heap buffer overflow in fdt\_get\_string() function.

Below is the source code of fdt\_get\_string() method.

const char \*fdt\_get\_string(const void \*fdt, int stroffset, int \*lenp)

{

...

s = (const char \*)fdt + absoffset;

n = memchr(s, '\0', len); //here is the issue

if (!n) {

/\* missing terminating NULL \*/

err = -FDT\_ERR\_TRUNCATED;

goto fail;

}

...

}

1. POC

The id\:000004\,sig\:11\,src\:000005\,op\:havoc\,rep\:16 file in the folder.