Use of Out-of-range Pointer Offset in bfabiszewski/libmobi

0

Valid Reported on Sep 11th 2021

✍️ Description

Overview

This vulnerability is of out-of-bound read, which lets attackers read memory information beyond the buffer size. Possibly, attackers can use this to do DOS (Denial of Service) attack or ALSR bypass (by reading sensitive memory address information) to all applications which use the LibMobi library.

Root Cause

The root cause is the unsafe way of decompressing user given input in mobi_decompress_huffman_internal (src/compression.c line 118). Specifically, the code_length can be much larger (the value is calculated based on user's input, in our PoC, it has the value 66) than the real size of huffcdic->mincode_table (is fixed as 33 in src/compression.h). Therefore, attacker can make the program crash (if read to an invalid memory address) or get memory sensitive information (critical object's address) to bypass ASLR.

The following code shows the vulnerable point.

```
if (!(t1 & 0x80)) {
    /* get offset from mincode, maxcode tables */
    // Here makes the code_length be much larger than 33 which is t
    while (code < huffcdic->mincode_table[code_length]) {
        code_length++;
    }
    maxcode = huffcdic->maxcode_table[code_length];
}
```

Fix Suggestion

A simple fix way is to check whether code_length 's value is greater than 33 in here . Note that this may not be a complete fix since all the similar scenario in all decompress functions should be checked. I'm willing to help you to build a complete patch!

```
if (!(t1 & 0x80)) {
    /* get offset from mincode, maxcode tables */
    while (code < huffcdic->mincode_table[code_length]) {
        // OLD CODE:
        // code_length++;
        // SIMPLE FIX, MAY NOT BE COMPLETE:
        code_length++;
        if (code_length >= (sizeof(huffcdic->mincode_table) / sizec
          return MOBI_DATA_CORRUPT;
    }
    maxcode = huffcdic->maxcode_table[code_length];
}
```

🦹 Proof of Concept

Download latest libmobi and compile it with Address Sanitizer: CFLAGS=" -fsanitize=address " CXXFLAGS=" -fsanitize=address "

Use the following command and this POC-FILE to reproduce the crash:

enable address sanitizer

export ASAN_OPTIONS=abort_on_error=1:disable_coredump=0:unmap_shadow_on_exit=1

reproduce the crash

./mobitool -cdeimsrux7 -o any-tmp-dir-path POC-FILE

You should get similar crash information as follows:

```
=================================================================
==8324==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x61a00000
READ of size 4 at 0x61a000001db0 thread T0
    #0 0x7ffff758633a in mobi_decompress_huffman_internal /src/libmobi/libm
    #1 0x7ffff75868c5 in mobi_decompress_huffman_internal /src/libmobi/libm
    #2 0x7ffff75868c5 in mobi_decompress_huffman_internal /src/libmobi/libm
    #3 0x7ffff75868c5 in mobi_decompress_huffman_internal /src/libmobi/libm
    #4 0x7ffff7586a1e in mobi_decompress_huffman /src/libmobi/libmobi-git/s
    #5 0x7ffff75ad1eb in mobi_decompress_content /src/libmobi/libmobi-git/s
    #6 0x7ffff75ad677 in mobi_dump_rawml /src/libmobi/libmobi-git/src/util.
    #7 0x5555555630e4 in dump_rawml /src/libmobi/libmobi-git/tools/mobitool
    #8 0x5555555661e3 in loadfilename /src/libmobi/libmobi-git/tools/mobitc
    #9 0x555555566e46 in main /src/libmobi/libmobi-git/tools/mobitool.c:962
    #10 0x7ffff73a90b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.
    #11 0x55555555dead in _start (/src/libmobi/libmobi-git/install/bin/mobi

0x61a000001db0 is located 0 bytes to the right of 1328-byte region [0x61a00
allocated by thread T0 here:
```

(Published)

Vulnerability Type
CWE-823: Use of Out-of-range Pointer Offset

Severity
High (7.1)

Affected Version
*

Visibility
Public

Status
Fixed

Found by
Cen Zhang

Fixed by
Cen Zhang
unranked

This report was seen 401 times.

Impact

This vulnerability is capable of overwriting memory content with user given values. For all applications using libmobi (commits c814c4aba4e090fa32805ff8ff459df6c2c61b5c in Sep 10th, 2021 or release version 0.7 (2020 Sep 10th)). Likely, attackers can use this to do DOS (Denial of Service) attack or ALSR bypass (by reading sensitive memory address information) to any application which uses the LibMobi library.

References

- compression.c L143-L145

```
#0 0x7ffff769f6e7 in calloc (/lib/x86_64-linux-gnu/libasan.so.6+0xb06e7
#1 0x7ffff758ee67 in mobi_init_huffcdic /src/libmobi/libmobi-git/src/me
#2 0x7ffff75aca61 in mobi_decompress_content /src/libmobi/libmobi-git/s
#3 0x7ffff75ad677 in mobi_dump_rawml /src/libmobi/libmobi-git/src/util.
#4 0x5555555630e4 in dump_rawml /src/libmobi/libmobi-git/tools/mobitool
#5 0x5555555661e3 in loadfilename /src/libmobi/libmobi-git/tools/mobitc
#6 0x555555566e46 in main /src/libmobi/libmobi-git/tools/mobitool.c:962
#7 0x7ffff73a90b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6

SUMMARY: AddressSanitizer: heap-buffer-overflow /src/libmobi/libmobi-git/sr
```

Cen Zhang submitted a patch a year ago

Z-Old a year ago — Admin

Hey Cen, I've emailed the maintainer for you.

We have contacted a member of the bfabiszewski/libmobi team and are waiting to hear back a year ago

Bartek Fabiszewski validated this vulnerability a year ago

Cen Zhang has been awarded the disclosure bounty ✅

The fix bounty is now up for grabs

Bartek Fabiszewski marked this as fixed with commit bec783 a year ago

Cen Zhang has been awarded the fix bounty ✅

This vulnerability will not receive a CVE ❌

compression.c#L143-L145 has been validated ✅

Jamie Slome a year ago — Admin

@bartek - the researcher has requested a CVE for this report. Before we proceed, are you happy to publish a CVE or this report?

Bartek a year ago — Maintainer

Yes, thanks!

Cen Zhang a year ago — Researcher

@admin, may you help to continue the application process, many thanks~

Jamie Slome a year ago — Admin

CVE published! 🎉

Sign in to join this conversation