Heap buffer overflow in acc_ua_get_be32() #391

Closed **giantbranch** opened this issue on Jul 23, 2020 · 1 comment

giantbranch commented on Jul 23, 2020 • edited

Author: giantbranch of NSFOCUS Security Team

What's the problem (or question)?

A heap buffer overflow read in the latest commit of the devel branch

ASAN reports:

```
==21053==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x6160000014bb at pc 0x000000755961 bp 0x7ffe38d0c080 sp 0x7ffe38d0c078
READ of size 1 at 0x6160000014bb thread T0
    #0 0x755960 in acc_ua_get_be32(void const*) /src/upx-multi/src/./miniacc.h:6099:12
    #1 0x755960 in get_be32(void const*) /src/upx-multi/src/./bele.h:78:12
    #2 0x755960 in N_BELE_RTP::BEPolicy::get32(void const*) const /src/upx-multi/src/./bele_policy.h:115:18
    #3 0x58a254 in Packer::get_te32(void const*) const /src/upx-multi/src/./packer.h:296:65
    #4 0x58a254 in PackLinuxElf32::invert_pt_dynamic(N_Elf::Dyn<N_Elf::ElfITypes<LE16, LE32, LE32, LE32, LE32> > const*) /src/upx-multi/src/p_lx_elf.cpp:1601:32
    #5 0x588959 in PackLinuxElf32::PackLinuxElf32help1(InputFile*) /src/upx-multi/src/p_lx_elf.cpp:305:13
    #6 0x5d70d4 in PackLinuxElf32Be::PackLinuxElf32Be(InputFile*) /src/upx-multi/src/./p_lx_elf.h:385:9
    #7 0x5d70d4 in PackLinuxElf32armBe::PackLinuxElf32armBe(InputFile*) /src/upx-multi/src/p_lx_elf.cpp:4969:58
    #8 0x6e50c0 in PackMaster::visitAllPackers(Packer* (*)(Packer*, void*), InputFile*, options_t const*, void*) /src/upx-multi/src/packmast.cpp:196:9
    #9 0x6e8ff1 in PackMaster::getUnpacker(InputFile*) /src/upx-multi/src/packmast.cpp:248:18
    #10 0x6e8ff1 in PackMaster::unpack(OutputFile*) /src/upx-multi/src/packmast.cpp:266:9
    #11 0x75826b in do_one_file(char const*, char*) /src/upx-multi/src/work.cpp:160:12
    #12 0x7597c2 in do_files(int, int, char**) /src/upx-multi/src/work.cpp:271:13
    #13 0x555aed in main /src/upx-multi/src/main.cpp:1538:5
    #14 0x7fcbbeced83f in __libc_start_main /build/glibc-e6zv40/glibc-2.23/csu/../csu/libc-start.c:291
    #15 0x41ce98 in _start (/out/upx-multi/upx-multi+0x41ce98)

0x6160000014bb is located 4 bytes to the right of 567-byte region [0x616000001280,0x6160000014b7)
allocated by thread T0 here:
    #0 0x49519d in malloc (/out/upx-multi/upx-multi+0x49519d)
    #1 0x569797 in MemBuffer::alloc(unsigned long long) /src/upx-multi/src/mem.cpp:194:42

SUMMARY: AddressSanitizer: heap-buffer-overflow /src/upx-multi/src/./miniacc.h:6099:12 in acc_ua_get_be32(void const*)
Shadow bytes around the buggy address:
  0x0c2c7fff8240: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff8250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2c7fff8260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2c7fff8270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c2c7fff8280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c2c7fff8290: 00 00 00 00 00 00 07[fa]fa fa fa fa fa fa fa fa
  0x0c2c7fff82a0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff82b0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff82c0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff82d0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c2c7fff82e0: 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
  Shadow gap:              cc
==21053==ABORTING
```

What should have happened?

Check if the file is normal, exit if abnormal

Do you have an idea for a solution?

Add more checks

How can we reproduce the issue?

upx.out -d <poc_filename>

poc:
poc-heap-buffer-overflow-2.tar.gz

Please tell us details about your environment.

- UPX version used (`upx --version`):

```
upx 4.0.0-git-87b73e5cfdc1+
UCL data compression library 1.03
zlib data compression library 1.2.8
LZMA SDK version 4.43
Copyright (C) 1996-2020 Markus Franz Xaver Johannes Oberhumer
Copyright (C) 1996-2020 Laszlo Molnar
Copyright (C) 2000-2020 John F. Reiser
Copyright (C) 2002-2020 Jens Medoch
Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler
Copyright (C) 1999-2006 Igor Pavlov
UPX comes with ABSOLUTELY NO WARRANTY; for details t
```

- Host Operating System and version: Ubuntu 16.04.2 LTS
- Host CPU architecture: x86_64
- Target Operating System and version: same as Host
- Target CPU architecture: same as Host

jreiser added a commit that referenced this issue on Jul 23, 2020

`Defend against bad PT_DYNAMIC` ··· ✕ `8764fdc`

jreiser mentioned this issue on Jul 23, 2020

Heap buffer overflow in PackLinuxElf32::invert_pt_dynamic #392

Closed

jreiser commented on Jul 23, 2020 Collaborator

Fixed on `devel` branch by above commit.

giantbranch closed this as completed on Jul 27, 2020

markus-oberhumer pushed a commit that referenced this issue on Aug 17

`Defend against bad PT_DYNAMIC` ··· `4e2fdb4`

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants