New issue

Jump to bottom

canPack@p_lx_elf.cpp:2571 BufferOverflow (both latest release version and devel version) #421

Closed Hustcw opened this issue on Nov 12, 2020 · 2 comments

Hustcw commented on Nov 12, 2020 • edited

What's the problem (or question)?

An issue was discovered in upx 3.96(**devel** branch), There is an illegal memory access in function canPack at p_lx_elf.cpp:2571.
I also check the newest release version meet the same crash, lies at p_lx_elf.cpp:2490.

What should have happened?

no illegal memory access (crash)

Do you have an idea for a solution?

check the relocation_offset and do not access the illegal memory

How can we reproduce the issue?

1. Compile the devel branch with sanitize open
   `export BUILD_TYPE_SANITIZE=1; make all`
2. Use `upx.out poc` and get crash
   download the poc here.

source

```
2566                        unsigned rel_off = get_te64(&dynseg[-1+ z_rel].d_val);
2567                        Elf64_Rela *rp = (Elf64_Rela *)&file_image[rel_off];
2568                        unsigned relsz   = get_te64(&dynseg[-1+ z_rsz].d_val);
2569                        Elf64_Rela *last = (Elf64_Rela *)(relsz + (char *)rp);
2570                        for (; rp < last; ++rp) {
2571                            unsigned r_va = get_te64(&rp->r_offset);
2572                            if (r_va == user_init_ava) { // found the Elf64_Rela
2573                                unsigned r_info = get_te64(&rp->r_info);
2574                                unsigned r_type = ELF64_R_TYPE(r_info);
2575                                if (Elf64_Ehdr::EM_AARCH64 == e_machine
2576                                &&  R_AARCH64_RELATIVE == r_type) {
2577                                    user_init_va = get_te64(&rp->r_addend);
2578                                }
```

the source code didn't check the **rel_off** so get an illegal rp

debug

```
[ SOURCE (CODE) ]
In file: /home/wanghao/upx_dev/src/p_lx_elf.cpp
   2565                        if (z_rel && z_rsz) {
   2566                            unsigned rel_off = get_te64(&dynseg[-1+ z_rel].d_val);
   2567                            Elf64_Rela *rp = (Elf64_Rela *)&file_image[rel_off];
   2568                            unsigned relsz   = get_te64(&dynseg[-1+ z_rsz].d_val);
   2569                            Elf64_Rela *last = (Elf64_Rela *)(relsz + (char *)rp);
 ► 2570                            for (; rp < last; ++rp) {
   2571                                unsigned r_va = get_te64(&rp->r_offset);
   2572                                if (r_va == user_init_ava) { // found the Elf64_Rela
   2573                                    unsigned r_info = get_te64(&rp->r_info);
   2574                                    unsigned r_type = ELF64_R_TYPE(r_info);
   2575                                    if (Elf64_Ehdr::EM_AARCH64 == e_machine
[ STACK ]
00:0000| rsp  0x7fffffffca60 —▸ 0x61b000000009 ◂— 0x0
01:0008|      0x7fffffffca68 —▸ 0x61b00001f188 —▸ 0x8f42e0 (N_BELE_RTP::le_policy) —▸ 0x6abc10 —▸ 0x55b600 (N_BELE_RTP::LEPolicy::~LEPolicy()) ◂— ..
.
02:0010|      0x7fffffffca70 —▸ 0x7fffffffd020 —▸ 0x7fffffffd4b8 ◂— 0x0
03:0018|      0x7fffffffca78 —▸ 0x61b00001f308 —▸ 0x1d00000009 ◂— 0x0
04:0020|      0x7fffffffca80 —▸ 0x61b00001f388 —▸ 0x1800000019 ◂— 0x0
05:0028|      0x7fffffffca88 —▸ 0x63000000fd40 —▸ 0x60ee10 (stub_i386_linux_elf_interp_entry+22640) ◂— and    byte ptr [rdx + 0x45], dl
06:0030|      0x7fffffffca90 —▸ 0x61b00001f310 —▸ 0x630000000400 ◂— 0x10102464c457f
07:0038|      0x7fffffffca98 —▸ 0x61b00001f478 —▸ 0x630000000440 —▸ 0x500000006 ◂— 0x0
[ BACKTRACE ]
 ► f 0          49df71 PackLinuxElf64::canPack()+5809
   f 1          5240a7
   f 2          525190
   f 3          526d1a PackMaster::getPacker(InputFile*)+42
   f 4          526e3d PackMaster::pack(OutputFile*)+45
   f 5          55bedf
   f 6          55c3af
   f 7          403dbf main+1823
   f 8    7ffff630e840 __libc_start_main+240

pwndbg> p rp
$1 = (Elf64_Rela *) 0x6300004013e8
```

bug report

```
ASAN:SIGSEGV
=================================================================
==65507==ERROR: AddressSanitizer: SEGV on unknown address 0x6300004013e8 (pc 0x00000055aff0 bp 0x0c3600003e31 sp 0x7ffc8f9b7378 T0)
    #0 0x55afef in get_le64(void const*) /home/wanghao/upx_dev/src/bele_policy.h:193
    #1 0x55afef in N_BELE_RTP::LEPolicy::get64(void const*) const /home/wanghao/upx_dev/src/bele_policy.h:194
    #2 0x49dfe2 in Packer::get_te64(void const*) const /home/wanghao/upx_dev/src/packer.h:297
    #3 0x49dfe2 in PackLinuxElf64::canPack() /home/wanghao/upx_dev/src/p_lx_elf.cpp:2571
    #4 0x5240a6 in try_pack /home/wanghao/upx_dev/src/packmast.cpp:91
    #5 0x52518f in PackMaster::visitAllPackers(Packer* (*)(Packer*, void*), InputFile*, options_t const*, void*) /home/wanghao/upx_dev/src/packmast.cpp:194
    #6 0x526d19 in PackMaster::getPacker(InputFile*) /home/wanghao/upx_dev/src/packmast.cpp:240
    #7 0x526e3c in PackMaster::pack(OutputFile*) /home/wanghao/upx_dev/src/packmast.cpp:260
    #8 0x55bede in do_one_file(char const*, char*) /home/wanghao/upx_dev/src/work.cpp:158
    #9 0x55c3ae in do_files(int, int, char**) /home/wanghao/upx_dev/src/work.cpp:271
    #10 0x403dbe in main /home/wanghao/upx_dev/src/main.cpp:1538
    #11 0x7f702527783f in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x2083f)
```

Please tell us details about your environment.

- UPX version used (both `upx 3.96(devel)` and `upx 3.96 release`):
- Host Operating System and version: ubuntu 16.04
- Host CPU architecture: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
- Target Operating System and version: ubuntu 16.04
- Target CPU architecture: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz

Hustcw changed the title ~~**canPack@p_lx_elf.cpp:2571 BufferOverflow**~~ **canPack@p_lx_elf.cpp:2571 BufferOverflow (Both latest release version and devel version)** on Nov 12, 2020

Hustcw changed the title ~~**canPack@p_lx_elf.cpp:2571 BufferOverflow (Both latest release version and devel version)**~~ **canPack@p_lx_elf.cpp:2571 BufferOverflow (both latest release version and devel version)** on Nov 12, 2020

Hustcw commented on Nov 17, 2020 — Author

Any updates to fix this bug?

jreiser added a commit that referenced this issue on Dec 11, 2020

`Check DT_REL/DT_RELA, DT_RELSZ/DT_RELASZ` ··· `3781df9`

jreiser commented on Dec 11, 2020 — Collaborator

`readelf --all` shows many many problems with the `poc`, beginning with

```
readelf: upx_crash_p_lx_elf_dev_2490: Error: Reading 72027907223978242 bytes extends past end of file for string table
readelf: upx_crash_p_lx_elf_dev_2490: Warning: Size of section 28 is larger than the entire file!
readelf: upx_crash_p_lx_elf_dev_2490: Error: no .dynamic section in the dynamic segment
readelf: upx_crash_p_lx_elf_dev_2490: Warning: Virtual address 0xddff400b38 not located in any PT_LOAD segment.
readelf: upx_crash_p_lx_elf_dev_2490: Error: Unable to determine the length of the dynamic string table
readelf: upx_crash_p_lx_elf_dev_2490: Error:  bad symbol index: f5058b48 in relocreadelf: upx_crash_p_lx_elf_dev_2490: Error: Reading 16 bytes extends past end of file for version
need aux (3)
readelf: upx_crash_p_lx_elf_dev_2490: Error: Reading 16 bytes extends past end of file for version need aux (3)
```

If you are running a fuzzer then please concentrate on fuzzing `upx -d` which is much more useful.

jreiser closed this as completed on Dec 11, 2020

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

`Check DT_REL/DT_RELA, DT_RELSZ/DT_RELASZ` ··· `13bc031`

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants