New issue

Jump to bottom

Out of Bounds Write in v1.0.4 #3

Closed Halcy0nic opened this issue on Jul 22 · 10 comments

Halcy0nic commented on Jul 22

Hi!

While I was using the tool I had some fuzz tests running in the background and I think there might be an out of bounds write bug in the webp to png converter. I compiled the tool from source using the default instructions/Makefile. I can't exactly figure out from the backtrace where the out of bounds write is happening in png2webp.c, but a rough guess would be somewhere around:

png2webp/png2webp.c
Line 499 in 0c71191

```
499     if(reverse)
```

png2webp/png2webp.c
Line 504 in 0c71191

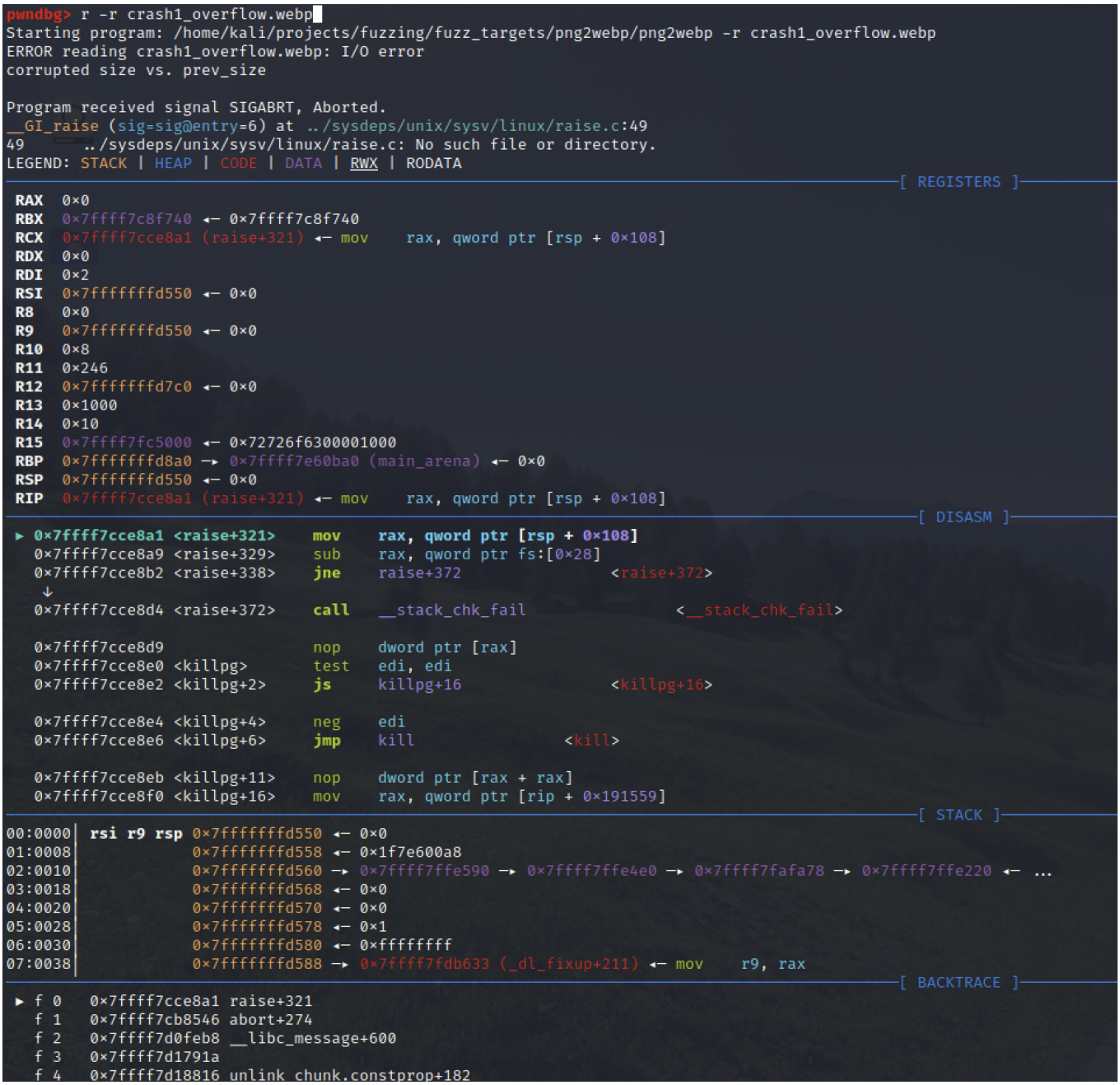
```
504     memcpy(&ext, *argv + len - 4, 4);
```

png2webp/png2webp.c
Line 505 in 0c71191

```
505     memcpy(&extmatch, (char[4]){"webp"}, 4);
```

I've attached the valgrind and gdb output below with a copy of the file used to trigger the issue:

```
└─$ valgrind ../png2webp -r ./crash1_overflow.webp
==666884== Memcheck, a memory error detector
==666884== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==666884== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==666884== Command: ../png2webp -r ./crash1_overflow.webp
==666884==
==666884== Invalid write of size 4
==666884==    at 0x407E40: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x40D4B6: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x49CF7FC: (below main) (libc-start.c:332)
==666884==  Address 0x4d852e8 is 1 bytes after a block of size 7 alloc'd
==666884==    at 0x483F7B5: malloc (vg_replace_malloc.c:381)
==666884==    by 0x407E2C: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x40D4B6: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x49CF7FC: (below main) (libc-start.c:332)
==666884==
==666884== Invalid write of size 8
==666884==    at 0x407E4B: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x40D4B6: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x49CF7FC: (below main) (libc-start.c:332)
==666884==  Address 0x4d852e0 is 0 bytes inside a block of size 7 alloc'd
==666884==    at 0x483F7B5: malloc (vg_replace_malloc.c:381)
==666884==    by 0x407E2C: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x40D4B6: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x49CF7FC: (below main) (libc-start.c:332)
==666884==
==666884== Invalid write of size 1
==666884==    at 0x4849E86: mempcpy (vg_replace_strmem.c:1668)
==666884==    by 0x4A27D7D: _IO_file_xsgetn (fileops.c:1304)
==666884==    by 0x4A1C93E: fread (iofread.c:38)
==666884==    by 0x407E66: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x40D4B6: ??? (in /home/kali/projects/fuzzing/fuzz_targets/png2webp/png2webp)
==666884==    by 0x49CF7FC: (below main) (libc-start.c:332)
==666884==  Address 0x4d85a53 is 1,827 bytes inside an unallocated block of size 2,092,208 in arena "client"
==666884==
```



Crash file

This would possibly allow an attacker to overwrite heap memory with attacker provided data.
crash.zip

landfillbaby commented on Jul 23 — Owner

What OS, architecture, and compiler were you testing on?

landfillbaby commented on Jul 23 • edited — Owner

I tested that file on a version I just compiled on Termux on my Pixel 6:

```
$ png2webp -rv crash1_overflow.webp
Decoding crash1_overflow.webp ...
FORTIFY: read: count 18446744073709549715 > SSIZE_MAX
Aborted
```

The problem seems to be that, against the C standard, certain platforms use `ssize_t` for `fread`'s parameters instead of `size_t`.
Try again using this patch, and when I'm at my PC I'll look into it further.

```
diff --git a/png2webp.c b/png2webp.c
index 42443f5..30bd4fd 100644
--- a/png2webp.c
+++ b/png2webp.c
@@ -319,6 +319,14 @@ static bool w2p(char *ip, char *op) {
   }
   size_t l = ((uint32_t)(i[4] | (i[5] << 8) | (i[6] << 16) | (i[7] << 24))) + 8;
   // ^ RIFF header size
+  if(l < 12
+#ifdef SSIZE_MAX
+    || l - 12 > SSIZE_MAX
+#endif
+  ) {
+    PF("ERROR reading %s: %s", IP, k[2]);
+    goto w2p_close;
+  }
   x = malloc(l);
   if(!x) {
     PF("ERROR reading %s: %s", IP, *k);
```

👍 1

landfillbaby commented on Jul 23 — Owner

I need to check this doesn't happen on platforms that don't define `SSIZE_MAX`, e.g. Windows.

landfillbaby commented on Jul 23 • edited — Owner

Also what fuzzer are you using? I might use it myself. I hope it's AFL

landfillbaby commented on Jul 23 — Owner

Fixed in v1.0.5 (I think).
Won't close until you answer my questions though :)
I knew there was something up with my WebP reading code, thanks!

landfillbaby commented on Jul 23 • edited Owner

@Halcy0nic feel free to add this to your trophy list lol 🏅
BTW the part you guessed is safe, it's just the file extension replacement.

😄 1

Halcy0nic commented on Jul 23 Author

Sweet! Just tested it out and it seems fixed. Thanks again!

Halcy0nic commented on Jul 23 Author

Also you are correct, the fuzzer I was using is AFL++ lol

Halcy0nic commented on Jul 23 Author

I tested on a few Linux distros (Debian, ubuntu, etc), all 64 bit

landfillbaby commented on Jul 23 Owner

Ok good thank you :) I'll close this now

 **landfillbaby** closed this as completed on Jul 23

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

---

2 participants