ckolivas / lrzip Public

Code | Issues 15 | Pull requests 2 | Actions | Projects | Wiki

New issue

Jump to bottom

Segmentation fault casued by use after free in multithread process from close_stream_in, stream:1870 to lzma_decompress_buf, stream:546 #165

Closed 5hadowblad3 opened this issue on Sep 4, 2020 · 3 comments

5hadowblad3 commented on Sep 4, 2020 • edited

Hi, there.

I find there is use after free issue in multithread processing in stream.c in the newest master branch `597be1f` .
The reason is that the buffer is unchecked during a multithread stream read.
Here is the detailed explanation:

```
i64 read_stream(rzip_control *control, void *ss, int streamno, uchar *p, i64 len)
{
    struct stream_info *sinfo = ss;
    struct stream *s = &sinfo->s[streamno];
    i64 ret = 0;

    while (len) {
        i64 n;

        n = MIN(s->buflen - s->bufp, len);

        if (n > 0) {
            if (unlikely(!s->buf))
                failure_return(("Stream ran out prematurely, likely corrupt archive\n"), -1);
            memcpy(p, s->buf + s->bufp, n);
            s->bufp += n;
            p += n;
            len -= n;
            ret += n;
        }

        if (len && s->bufp == s->buflen) {
            if (unlikely(fill_buffer(control, sinfo, s, streamno)))
                return -1;
            if (s->bufp == s->buflen)
                break;
        }
    }
```

allocate multiple thread for processing input

```
    st->i = s->uthread_no;
    st->control = control;
    if (unlikely(!create_pthread(control, &threads[s->uthread_no], NULL, ucompthread, st))) {
        dealloc(st);
        return -1;
    }
```

a thread is allocated in fill_buffer

```
/* close down an input stream */
int close_stream_in(rzip_control *control, void *ss)
{
    struct stream_info *sinfo = ss;
    int i;

    print_maxverbose("Closing stream at %lld, want to seek to %lld\n",
             get_readseek(control, control->fd_in),
             sinfo->initial_pos + sinfo->total_read);
    if (unlikely(read_seekto(control, sinfo, sinfo->total_read)))
        return -1;

    for (i = 0; i < sinfo->num_streams; i++)
        dealloc(sinfo->s[i].buf);

    output_thread = 0;
    dealloc(ucthread);
    dealloc(threads);
    dealloc(sinfo->s);
    dealloc(sinfo);

    return 0;
}
```

thread is freed after reading partial input

```
1507 static void *ucompthread(void *data)
1508 {
1509     stream_thread_struct *s = data;
1510     rzip_control *control = s->control;
1511     int waited = 0, ret = 0, i = s->i;
1512     struct uncomp_thread *uci;
1513
1514     dealloc(data);
1515     uci = &ucthread[i];
1516
1517     if (unlikely(setpriority(PRIO_PROCESS, 0, control->nice_val) == -1)) {
1518         print_err("Warning, unable to set thread nice value %d...Resetting to %d\n", control->nice_val, control->current_priority);
1519         setpriority(PRIO_PROCESS, 0, (control->nice_val=control->current_priority));
1520     }
1521
1522 retry:
1523     if (uci->c_type != CTYPE_NONE) {
1524         switch (uci->c_type) {
1525             case CTYPE_LZMA:
1526                 ret = lzma_decompress_buf(control, uci);
1527                 break;
1528             case CTYPE_LZO:
```

thread info is directly used without nullity check

The high-level reason is similar to issue #164, but the program behavior/path is different.

To reproduce, run

```
lrzip -t uaf-stream546.lrz
```

Since it is a problem in the multithread program, you might need to run this command multiple times to trigger.

POC (unzip first):
uaf-stream546.lrz.zip

Here is the output from the terminal:

```
Decompressing...
Bad checksum: 0x5b496f91 - expected: 0x2000210c
Segmentation fault
```

This is the trace reported by ASAN:

```
==163048==ERROR: AddressSanitizer: heap-use-after-free on address 0x62000000f0e0 at pc 0x000000440f8c bp 0x7ff7bdffdd70 sp 0x7ff7bdffdd60
Fatal error - exiting
WRITE of size 8 at 0x62000000f0e0 thread T3
    #0 0x440f8b in lzma_decompress_buf ../stream.c:546
    #1 0x440f8b in ucompthread ../stream.c:1526
    #2 0x7ff7c1d366b9 in start_thread (/lib/x86_64-linux-gnu/libpthread.so.0+0x76b9)
    #3 0x7ff7c116841c in clone (/lib/x86_64-linux-gnu/libc.so.6+0x10741c)

0x62000000f0e0 is located 96 bytes inside of 3936-byte region [0x62000000f080,0x62000000ffe0)
freed by thread T0 here:
    #0 0x7ff7c263032a in __interceptor_free (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x9832a)
    #1 0x454a42 in close_stream_in ../stream.c:1870

previously allocated by thread T0 here:
    #0 0x7ff7c26307fa in __interceptor_calloc (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x987fa)
    #1 0x44c8f0 in open_stream_in ../stream.c:1080

Thread T3 created by T0 here:
    #0 0x7ff7c25ce1e3 in pthread_create (/usr/lib/x86_64-linux-gnu/libasan.so.2+0x361e3)
    #1 0x4516f3 in create_pthread ../stream.c:133
    #2 0x4516f3 in fill_buffer ../stream.c:1699
    #3 0x4516f3 in read_stream ../stream.c:1786

SUMMARY: AddressSanitizer: heap-use-after-free ../stream.c:546 lzma_decompress_buf
Shadow bytes around the buggy address:
  0x0c407fff9dc0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c407fff9dd0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c407fff9de0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c407fff9df0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c407fff9e00: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0c407fff9e10: fd fd fd fd fd fd fd fd fd fd fd fd fd[fd]fd fd fd
  0x0c407fff9e20: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c407fff9e30: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c407fff9e40: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c407fff9e50: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c407fff9e60: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
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
  Heap right redzone:      fb
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack partial redzone:   f4
  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
==163048==ABORTING
```

5hadowblad3 changed the title ~~Segmentation fault casued by use after free in multithread process from close_stream_in, stream:1870 to lzma_decompress_buf, stream:546~~ Segmentation fault casued by use after free in multithread process from close_stream_in, stream:1870 to lzma_decompress_buf, stream:546 on Sep 4, 2020

5hadowblad3 mentioned this issue on Sep 4, 2020

Segmentation fault caused by null pointer dereference during multithread processing in ucompthread, stream.c:1523 #164

Closed

pete4abw mentioned this issue on Oct 29, 2020

Fixes to Corrupt File errors and segfaults #171

Closed

ckolivas commented on Feb 14, 2021 Owner

Fixed in git master

ckolivas closed this as completed on Feb 14, 2021

5hadowblad3 commented on Jun 9, 2021 Author

This is assigned with CVE-2021-27347.

pete4abw mentioned this issue on Jun 9, 2021

CVEs 2020-25467 and 2021-27345 and 2021-27347 pete4abw/lrzip-next#30

Closed

carnil commented on Apr 9

Fixing commit should be `be884d0`

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

3 participants