[bug #64551] Possible null pointer dereference on the function get_buffer

12 messages

HaoxinTu <INVALID.NOREPLY@gnu.org> Sun, Aug 13, 2023 at 5:12 PM
To: HaoxinTu <haoxintu@gmail.com>, psmith@gnu.org, boris@kolpackov.net, bug-make@gnu.org

URL:
<https://savannah.gnu.org/bugs/?64551>

Summary: Possible null pointer dereference on the function get_buffer
Group: make
Submitter: haoxin
Submitted: Sun 13 Aug 2023 09:12:11 AM UTC
Severity: 3 - Normal
Item Group: Bug
Status: None
Privacy: Public
Assigned to: None
Open/Closed: Open
Discussion Lock: Any
Component Version: 4.2
Operating System: None
Fixed Release: None
Triage Status: None

_____

Follow-up Comments:

Date: Sun 13 Aug 2023 09:12:11 AM UTC By: HaoxinTu <haoxin>
Hi,

We have developed a new tool built on top of KLEE (http://klee.github.io/) to automatically test make-4.2 and found there might be a possible null pointer dereference in the function get_buffer in output.c:605. Here is the stack info when the error occurs:

Stack:
#000010267 in get_buffer (need=69) at ../make-4.2/output.c:605
#100010161 in fatal (flocp=0, len=0, fmt=93825073769664) at ../make-4.2/output.c:683
#200012801 in xrealloc (ptr=0, size=138) at ../make-4.2/misc.c:252
#300010258 in get_buffer (need=69) at ../make-4.2/output.c:602
#400010161 in fatal (flocp=0, len=0, fmt=93825073769664) at ../make-4.2/output.c:683
#500021390 in xmalloc (size=200) at ../make-4.2/misc.c:223
#600054772 in eval_makefile (filename=93825076010318, flags=0) at ../make-4.2/read.c:432
#700066966 in read_all_makefiles (makefiles=93825076779920) at ../make-4.2/read.c:218
#800062387 in __klee_posix_wrapped_main (argc=4, argv=93825076037408) at ../make-4.2/main.c:1970
#900008309 in __user_main (=7, =93825073087424, =93825073087488) at runtime/POSIX/klee_init_env.c:252
#1000001882 in __uClibc_main (=93825022291208, =7, =93825073087424, =0, =0, =0, =0) at libc/misc/internals/__uClibc_main.c:401
#1100002047 in main (=7, =93825073087424)

Here is the main code involved in the error:

```
void * xrealloc (void *ptr, unsigned int size) {
  void *result;
  result = ptr ? realloc (ptr, size) : malloc (size);
  if (result == 0)
    OUT_OF_MEM();
  return result;
}
void fatal (const floc *flocp, size_t len, ...) {
  char * p = get_buffer (len);
  ...
  die (MAKE_FAILURE);
}
static struct fmtstring {char *buffer; size_t size;} fmtbuf = {NULL, 0};
static char * get_buffer (size_t need) {
  if (need > fmtbuf.size) {
      fmtbuf.size += need * 2;
      fmtbuf.buffer = xrealloc (fmtbuf.buffer, fmtbuf.size);
  }
  fmtbuf.buffer[need-1] = '\0'; // null pointer dereference
  return fmtbuf.buffer;
}
```

The error happens due to the lack of handling of consecutive NULL pointer returns. Normally, a NULL pointer checking is conducted after allocating the memory in function `xrealloc`. If the return value from `realloc` or `malloc` equals 0, the program will be terminated using the function OUT_OF_MEM as expected. It seems ok so far. However, a memory error might happen when another `xrealloc` is called and it returns NULL again. Inside the function `fatal`, it first allocates a buffer via function `get_buffer` based on the required length len and terminates the execution via function `die` normally. In the function `get_buffer`, it first checks whether the needed size need is larger than the defined buffer size `fmtbuf.size`. If the condition is satisfied, the buffer `fmtbuf.buffer` will be updated through another `xrealloc`. The error happens if the second `xrealloc` returns 0. If this occurs, the value stored in `fmtbuf.buffer` is 0. Since the value of `fmtbuf.size` was doubled and it was defined as global scope, the condition of if-statement will be false, and the `fmtbuf.buffer` will still keep the original NULL. Once this situation occurs, the dereferencing of the pointer `fmtbuf.buffer[need-1]` will lead to the null pointer dereference.

We only tested the make-4.2 version but the latest versions may have the same potential issue after we checked the code. Can you please take a look and check if this is a valid issue?

Adding a simple checking of the pointer `fmtbuf.buffer` after the if branch in the function `get_buffer` should avoid the potential issue if it is indeed an issue. Thanks.

Best,
Haoxin

Reply to this item at:

<https://savannah.gnu.org/bugs/?64551>

Paul D. Smith <INVALID.NOREPLY@gnu.org> Sun, Jan 7, 2024 at 1:49 AM
To: "Paul D. Smith" <psmith@gnu.org>, HaoxinTu <haoxintu@gmail.com>, boris@kolpackov.net, bug-make@gnu.org

Update of bug#64551 (group make):

Status:	None => Duplicate
Assigned to:	None => psmith
Open/Closed:	Open => Closed

Follow-up Comment #1:

I don't see how the "second" xrealloc() would return 0; the entire point of xrealloc is that it never returns 0.

However, I can see where the behavior of the code might lead to an infinite loop.

This issue was already addressed in GNU Make 4.3 via bug #13651
The version you're testing (4.2) was released in 2016.

It's certainly helpful to check for errors in tools like GNU Make but please check either the most recent published version or, even better, the current Git HEAD version.

Thanks!
[Quoted text hidden]

Haoxin Tu <haoxintu@gmail.com> Sat, Jan 20, 2024 at 1:53 PM
To: "Paul D. Smith" <INVALID.NOREPLY@gnu.org>
Cc: "Paul D. Smith" <psmith@gnu.org>, boris@kolpackov.net, bug-make@gnu.org

Hi,

Thank you so much for your time and reply.

We understand that the entire point of `xrealloc` is never returning 0 to client users/developers who use this function. However, the issue we reported here happens when the `xrealloc` internally handles the returned 0 from `realloc` or `malloc` functions.

In general, the key point is that the function `OUT_OF_MEM()` (invoked when the `result` gets a 0 in the implementation of `xrealloc`) does not immediately terminate the program execution, and the function `OUT_OF_MEM()` will continue to allocate buffers via `xrealloc` for printing purposes in the following and then terminate. Specifically, the continuous execution of function `OUT_OF_MEM()` calls the `xrealloc` again through the `get_buffer` function (`OUT_OF_MEM()` is a macro definition that will call the function `fatal`, which finally invocates the function `get_buffer`). As we mentioned in the initial report, once the second invocation of `xrealloc` (i.e., the one called inside `OUT_OF_MEM()`) returns zero and calls `OUT_OF_MEM()` again, a null pointer dereference is occurred in `fmtbuf.buffer[need-1] = '\0';` in the function `get_buffer`.

Please kindly check my explanation above and correct me if I am wrong. Thank you so much again and looking forward to hearing from you back again.

Best regards,
Haoxin

Paul D. Smith <INVALID.NOREPLY@gnu.org> 于2024年1月7日周日 01:49写道:
[Quoted text hidden]

Martin Dorey <Martin.Dorey@hitachivantara.com> Sat, Jan 20, 2024 at 10:43 PM
To: Haoxin Tu <haoxintu@gmail.com>, "Paul D. Smith" <INVALID.NOREPLY@gnu.org>
Cc: "Paul D. Smith" <psmith@gnu.org>, "boris@kolpackov.net" <boris@kolpackov.net>, "bug-make@gnu.org" <bug-make@gnu.org>

> ➢ once the second invocation of `xrealloc` ... returns zero

But it doesn't, it just recurses, calls get_buffer again, again likely fails and recurses still further until the stack is blown. Or it did until https://git.savannah.gnu.org/cgit/make.git/commit/?id=68be4f74fce91b76e5915449268d6b5eb687aab9.

From: bug-make-bounces+martin.dorey=hds.com@gnu.org <bug-make-bounces+martin.dorey=hds.com@gnu.org> on behalf of Haoxin Tu <haoxintu@gmail.com>
Sent: Saturday, January 20, 2024 06:17
To: Paul D. Smith <INVALID.NOREPLY@gnu.org>
Cc: Paul D. Smith <psmith@gnu.org>; boris@kolpackov.net <boris@kolpackov.net>; bug-make@gnu.org <bug-make@gnu.org>
Subject: Re: [bug #64551] Possible null pointer dereference on the function get_buffer

***** EXTERNAL EMAIL *****
[Quoted text hidden]

Haoxin Tu <haoxintu@gmail.com> Sat, Jan 20, 2024 at 11:37 PM
To: Martin Dorey <Martin.Dorey@hitachivantara.com>
Cc: "Paul D. Smith" <INVALID.NOREPLY@gnu.org>, "Paul D. Smith" <psmith@gnu.org>, "boris@kolpackov.net" <boris@kolpackov.net>, "bug-make@gnu.org" <bug-make@gnu.org>

Hi Martin,

Thanks for your speedy reply and clarifications.

But I don't understand why the second invocation of `xrealloc` can not return zero, I apologize for any imprecise information I provided in the previous emails.

From my understanding, once the second the `xrealloc` is invocated, there is a possibility that the `result` (https://github.com/mirror/make/blob/4.2/misc.c#L247) can return zero (from either `realloc` or `malloc`, just like the first invocation of `xrealloc`). Later, the `OUT_OF_MEM` function will be invocated again, and then the function `get_buffer`. In the following execution inside the function `get_buffer`, no `xrealloc` will be called as the condition `need > fmtbuf.size` (https://github.com/mirror/make/blob/4.2/output.c#L599) is false (the global static value `fmtbuf.size`, defined in https://github.com/mirror/make/blob/4.2/output.c#L593, was increased after the previous innovation of `get_buffer`), and then the execution of `fmtbuf.buffer[need-1] = '\0';` (https://github.com/mirror/make/blob/4.2/output.c#L605) leads to the null pointer dereference. Btw, I am sorry I don't see the recursive execution in this case (the link you provided seems old and I don't see such implementation in the version of make-4.2).

Please kindly let me know if I misunderstand something. Thank you so much for your time again!

Best regards,
Haoxin

Martin Dorey <Martin.Dorey@hitachivantara.com> 于2024年1月20日周六 22:43写道：
[Quoted text hidden]

Paul Smith <psmith@gnu.org> Sat, Jan 20, 2024 at 11:51 PM
Reply-To: psmith@gnu.org
To: Haoxin Tu <haoxintu@gmail.com>, Martin Dorey <Martin.Dorey@hitachivantara.com>
Cc: "bug-make@gnu.org" <bug-make@gnu.org>

On Sat, 2024-01-20 at 23:37 +0800, Haoxin Tu wrote:
> But I don't understand why the second invocation of `xrealloc` can
> not return zero, I apologize for any imprecise information I provided
> in the previous emails.

Because of what I said in my original reply:

> the entire point of xrealloc is that it never returns 0.

Look at the implementation of xrealloc():

```
  void *result = malloc (size ? size : 1);
  if (result == 0)
    OUT_OF_MEM();
  return result;
```

We know that OUT_OF_MEM() never returns. So there's no way this
function can return 0.

It will, as Martin suggests, recurse infinitely (one assumes) because
fatal() calls xrealloc() again, and malloc() will return 0, so it will
call fatal(), which calls xrealloc() again, and malloc will return 0,
so it will call fatal(), etc. etc.--this is what I meant by my
imprecise comment "infinite loop" I should have said "infinite
recursion".

As a reminder this is moot: this code has been rewritten and even the
infinite recursion problem was removed from the code back in 2017.

--
Paul D. Smith <psmith@gnu.org> Find some GNU make tips at:
https://www.gnu.org http://make.mad-scientist.net
"Please remain calm...I may be mad, but I am a professional." --Mad
Scientist

Martin Dorey <Martin.Dorey@hitachivantara.com> Sun, Jan 21, 2024 at 12:02 AM
To: "psmith@gnu.org" <psmith@gnu.org>, Haoxin Tu <haoxintu@gmail.com>
Cc: "bug-make@gnu.org" <bug-make@gnu.org>

> ➢ the link you provided seems old

Indeed, but that's because, as Paul noted, you're testing code that's even older.

> ➢ `fmtbuf.size`, defined in ..., was increased after the previous innovation of `get_buffer`), and then the execution of `fmtbuf.buffer[need-1] = '\0';`

There's a nice catch there - where, in that recursive failure, the writing of that terminator overflows a buffer that wasn't actually reallocated yet.

From: Paul Smith <psmith@gnu.org>
Sent: Saturday, January 20, 2024 07:51
To: Haoxin Tu <haoxintu@gmail.com>; Martin Dorey <Martin.Dorey@hitachivantara.com>
Cc: bug-make@gnu.org <bug-make@gnu.org>
Subject: Re: [bug #64551] Possible null pointer dereference on the function get_buffer

***** EXTERNAL EMAIL *****

On Sat, 2024-01-20 at 23:37 +0800, Haoxin Tu wrote:
> But I don't understand why the second invocation of `xrealloc` can
> not return zero, I apologize for any imprecise information I provided
> in the previous emails.

Because of what I said in my original reply:

> the entire point of xrealloc is that it never returns 0.

Look at the implementation of xrealloc():

```
  void *result = malloc (size ? size : 1);
  if (result == 0)
    OUT_OF_MEM();
  return result;
```

We know that OUT_OF_MEM() never returns. So there's no way this
function can return 0.

It will, as Martin suggests, recurse infinitely (one assumes) because
fatal() calls xrealloc() again, and malloc() will return 0, so it will

call fatal(), which calls xrealloc() again, and malloc will return 0,
so it will call fatal(), etc. etc.--this is what I meant by my
imprecise comment "infinite loop" I should have said "infinite
recursion".

As a reminder this is moot: this code has been rewritten and even the
infinite recursion problem was removed from the code back in 2017.

--
Paul D. Smith <psmith@gnu.org> Find some GNU make tips at:
https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.gnu.org%2F&data=05%7C02%7CMartin.Dorey%40hitachivantara.com%7C64b0e96f7b7a4d9cf0f308dc19cfa672%7C18791e1761594f52a8d4de814ca8284a%7C0%7C0%7C638413626848068646%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=2CgTubth7nezTPOU76hCwauMdV%2B8cz%2F9415iVpL%2FVe0%3D&reserved=0 https://nam04.safelinks.protection.outlook.com/?url=http%3A%2F%2Fmake.mad-scientist.net%2F&data=05%7C02%7CMartin.Dorey%40hitachivantara.com%7C64b0e96f7b7a4d9cf0f308dc19cfa672%7C18791e1761594f52a8d4de814ca8284a%7C0%7C0%7C638413626848074885%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=0jGEsG8MYYh26rgSn2n8dPl1eydBrKFvkzD0UqnvbuY%3D&reserved=0
[Quoted text hidden]

Haoxin Tu <haoxintu@gmail.com> Sun, Jan 21, 2024 at 12:10 AM
To: Martin Dorey <Martin.Dorey@hitachivantara.com>
Cc: "psmith@gnu.org" <psmith@gnu.org>, "bug-make@gnu.org" <bug-make@gnu.org>

Hi Paul and Martin,

->We know that OUT_OF_MEM() never returns. So there's no way this
function can return 0.

I totally agree with you all that the function `xrealloc` itself can never return 0.

->It will, as Martin suggests, recurse infinitely (one assumes) because
fatal() calls xrealloc() again, and malloc() will return 0, so it will
call fatal(), which calls xrealloc() again, and malloc will return 0,
so it will call fatal(), etc. etc.--this is what I meant by my
imprecise comment "infinite loop" I should have said "infinite
recursion".

Yeah, I also agree with your explanation here for the normal scenario, but the potential error I reported here is the situation:
fatal() calls xrealloc() again, and malloc() will return 0, so it will
call fatal(), which DOES NOT calls xrealloc() again but directly dereferences the pointer, then leading to the error.

->There's a nice catch there - where, in that recursive failure, the writing of that terminator overflows a buffer that wasn't actually reallocated yet.

Yeah, hope I made my points clear now.

Thank you all so much again.

Best regards,
Haoxin
