Bug 25487 (CVE-2020-10029) - sinl() stack corruption from crafted input (CVE-2020-10029)

Status:	RESOLVED FIXED	**Reported:**	2020-01-31 07:57 UTC by Guido Vranken
Alias:	CVE-2020-10029	**Modified:**	2021-09-21 00:53 UTC (History)
		CC List:	4 users (show)
Product:	glibc	**See Also:**	~~4586~~
Component:	math (show other bugs)	**Host:**	
Version:	unspecified	**Target:**	
		Build:	
Importance:	P2 normal	**Last reconfirmed:**	
Target Milestone:	2.32		
Assignee:	Not yet assigned to anyone	**Flags:**	fweimer: security+
URL:			
Keywords:			
Depends on:			
Blocks:			

Attachments

Add an attachment (proposed patch, testcase, etc.)

Note
You need to log in before you can comment on or make changes to this bug.

Guido Vranken 2020-01-31 07:57:16 UTC Description

```
Initially reported privately to the security address.

#include <math.h>
#include <string.h>
int main(void)
{
    const unsigned char _v[16] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x5d};
    long double v;
    memcpy(&v, _v, sizeof(v));
    /* Return the result so that gcc doesn't optimize everything away */
    return sinl(v);
}

$ gcc poc_sinl_buffer_overflow.c -lm && ./a.out
*** stack smashing detected ***: <unknown> terminated
Aborted (core dumped)

(gdb) bt
#0  __GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:51
#1  0x00007ffff7686801 in __GI_abort () at abort.c:79
#2  0x00007ffff76cf897 in __libc_message (action=action@entry=do_abort,
fmt=fmt@entry=0x7ffff77fc988 "*** %s ***: %s terminated\n") at
../sysdeps/posix/libc_fatal.c:181
#3  0x00007ffff777acd1 in __GI___fortify_fail_abort
(need_backtrace=need_backtrace@entry=false, msg=msg@entry=0x7ffff77fc966 "stack
smashing detected") at fortify_fail.c:33
#4  0x00007ffff777ac92 in __stack_chk_fail () at stack_chk_fail.c:29
#5  0x00007ffff7abbe6a in __kernel_rem_pio2 (x=0x7fffffffda70, y=0x7fffffffda90,
e0=<optimized out>, nx=<optimized out>, prec=<optimized out>, ipio2=0x7ffff7b042a0
<two_over_pi>)
    at ../sysdeps/ieee754/dbl-64/k_rem_pio2.c:362
#6  0x0000000000000000 in ?? ()


This has been tested on x64 Linux with both the Ubuntu glibc and the
latest git glibc.

Apart from sinl, some other functions that share the same code are prone to this as
well.

You can show me the patch before you merge so I can test it for you.

Credit: ForAllSecure Mayhem
```

joseph@codesourcery.com 2020-02-04 13:48:03 UTC Comment 1

```
This is a pseudo-zero.  ~~Bug 4586~~ (relating to handling of such values in
printf) was marked INVALID, but did get fixed at some point, and I think
we now consider such values should not cause a buffer overrun, although
they need not be consistently handled like any particular valid
floating-point representation.  So we should make sure __kernel_rem_pio2
only gets called with arguments where the high significand bit is set as
it probably expects, even for pseudo-zero and pseudo-normal long double
arguments to the original function.
```

cvs-commit@gcc.gnu.org 2020-02-12 23:32:53 UTC Comment 2

```
The master branch has been updated by Joseph Myers <jsm28@sourceware.org>:

https://sourceware.org/git/gitweb.cgi?
p=glibc.git;h=9333498794cde1d5cca518badf79533a24114b6f

commit 9333498794cde1d5cca518badf79533a24114b6f
Author: Joseph Myers <joseph@codesourcery.com>
Date:   Wed Feb 12 23:31:56 2020 +0000

    Avoid ldbl-96 stack corruption from range reduction of pseudo-zero (~~bug 25487~~).

    ~~Bug 25487~~ reports stack corruption in ldbl-96 sinl on a pseudo-zero
    argument (an representation where all the significand bits, including
    the explicit high bit, are zero, but the exponent is not zero, which
    is not a valid representation for the long double type).

    Although this is not a valid long double representation, existing
    practice in this area (see ~~bug 4586~~, originally marked invalid but
    subsequently fixed) is that we still seek to avoid invalid memory
    accesses as a result, in case of programs that treat arbitrary binary
    data as long double representations, although the invalid
    representations of the ldbl-96 format do not need to be consistently
    handled the same as any particular valid representation.

    This patch makes the range reduction detect pseudo-zero and unnormal
    representations that would otherwise go to __kernel_rem_pio2, and
    returns a NaN for them instead of continuing with the range reduction
    process.  (Pseudo-zero and unnormal representations whose unbiased
    exponent is less than -1 have already been safely returned from the
    function before this point without going through the rest of range
    reduction.)  Pseudo-zero representations would previously result in
    the value passed to __kernel_rem_pio2 being all-zero, which is
    definitely unsafe; unnormal representations would previously result in
    a value passed whose high bit is zero, which might well be unsafe
    since that is not a form of input expected by __kernel_rem_pio2.
    Tested for x86_64.
```

Joseph Myers 2020-02-12 23:33:23 UTC Comment 3

```
Fixed for 2.32.
```

cvs-commit@gcc.gnu.org 2020-02-13 16:07:08 UTC Comment 4

```
The master branch has been updated by Florian Weimer <fw@sourceware.org>:

https://sourceware.org/git/gitweb.cgi?
p=glibc.git;h=c10acd40262486dac597001aecc20ad9d3bd0e4a

commit c10acd40262486dac597001aecc20ad9d3bd0e4a
Author: Florian Weimer <fweimer@redhat.com>
Date:   Thu Feb 13 17:01:15 2020 +0100

    math/test-sinl-pseudo: Use stack protector only if available

    This fixes commit 9333498794cde1d5cca518bad ("Avoid ldbl-96 stack
    corruption from range reduction of pseudo-zero (~~bug 25487~~).").
```

Andreas Schwab 2020-03-05 11:48:23 UTC Comment 5

```
__ieee754_rem_pio2l is used by cosl, sinl, sincosl, and tanl.
```

cvs-commit@gcc.gnu.org 2020-03-16 16:54:54 UTC Comment 6

```
The release/2.29/master branch has been updated by Patricia Franklin
<patsy@sourceware.org>:

https://sourceware.org/git/gitweb.cgi?
p=glibc.git;h=0474cd5de60448f31d7b872805257092faa626e4

commit 0474cd5de60448f31d7b872805257092faa626e4
Author: Joseph Myers <joseph@codesourcery.com>
Date:   Wed Feb 12 23:31:56 2020 +0000

    Avoid ldbl-96 stack corruption from range reduction of pseudo-zero (~~bug 25487~~).

    ~~Bug 25487~~ reports stack corruption in ldbl-96 sinl on a pseudo-zero
    argument (an representation where all the significand bits, including
    the explicit high bit, are zero, but the exponent is not zero, which
    is not a valid representation for the long double type).

    Although this is not a valid long double representation, existing
    practice in this area (see ~~bug 4586~~, originally marked invalid but
    subsequently fixed) is that we still seek to avoid invalid memory
    accesses as a result, in case of programs that treat arbitrary binary
    data as long double representations, although the invalid
    representations of the ldbl-96 format do not need to be consistently
    handled the same as any particular valid representation.

    This patch makes the range reduction detect pseudo-zero and unnormal
    representations that would otherwise go to __kernel_rem_pio2, and
    returns a NaN for them instead of continuing with the range reduction
    process.  (Pseudo-zero and unnormal representations whose unbiased
    exponent is less than -1 have already been safely returned from the
    function before this point without going through the rest of range
    reduction.)  Pseudo-zero representations would previously result in
    the value passed to __kernel_rem_pio2 being all-zero, which is
    definitely unsafe; unnormal representations would previously result in
    a value passed whose high bit is zero, which might well be unsafe
    since that is not a form of input expected by __kernel_rem_pio2.

    Tested for x86_64.

    (cherry picked from commit 9333498794cde1d5cca518badf79533a24114b6f)
```

cvs-commit@gcc.gnu.org 2020-03-16 16:54:59 UTC Comment 7

```
The release/2.29/master branch has been updated by Patricia Franklin
<patsy@sourceware.org>:

https://sourceware.org/git/gitweb.cgi?
p=glibc.git;h=8e5d591b101d7d8a4628522f1e5ec24b6dfa731b

commit 8e5d591b101d7d8a4628522f1e5ec24b6dfa731b
Author: Florian Weimer <fweimer@redhat.com>
Date:   Thu Feb 13 17:01:15 2020 +0100

    math/test-sinl-pseudo: Use stack protector only if available

    This fixes commit 9333498794cde1d5cca518bad ("Avoid ldbl-96 stack
    corruption from range reduction of pseudo-zero (~~bug 25487~~).").

    (cherry picked from commit c10acd40262486dac597001aecc20ad9d3bd0e4a)
```

Huzaifa Sidhpurwala 2020-08-12 04:51:09 UTC Comment 8

```
Looking at the way crash is caused via pseudo-zero numbers and after running the
poc through gdb, it seems like on systems in which glibc is not compiled with -
fstack-protector-all, (which means the vuln function is not protected by stack-
canaries), all that can be achieved is overwrite the stack and the return address
with 0's.

This can only cause a crash and jumping any other location seems very difficult to
achieve if not impossible.
```

Disconnect3d 2020-08-19 01:38:31 UTC Comment 9

```
(In reply to Guido Vranken from comment #0)
> This has been tested on x64 Linux with both the Ubuntu glibc and the
> latest git glibc.


Btw since Ubuntu 18.04 and 20.04 both seems patched as of today, if anyone wants to
play with this, it can be reproduced on e.g. gcc:9.3 docker image
(https://hub.docker.com/layers/gcc/library/gcc/9.3/images/sha256-
dd7c100e12ddbf4178f5cd524a869fa54f453d35bf1b5f287ec6b70e3230c2e4?context=explore),
or by using the following docker image
(https://hub.docker.com/layers/disconnect3d/repro-cve-2020-
10029/latest/images/sha256-
0d7cf62eee140c9a0039945f8fe2ff3c53b7670b663cb67feef57878ab92ee06?context=explore)
and command where I compiled the example in the cve-2020-10029 directory:

docker run --rm -it --cap-drop=ALL --net=none disconnect3d/repro-cve-2020-10029
/cve-2020-10029/a.out
```

cvs-commit@gcc.gnu.org 2021-09-21 00:53:32 UTC Comment 10

```
The release/2.27/master branch has been updated by Dmitry Levin
<ldv@sourceware.org>:

https://sourceware.org/git/gitweb.cgi?
p=glibc.git;h=59420258afaf73dc8fab63ce186bac792613fe08
```

```
commit 59420258afaf73dc8fab63ce186bac792613fe08
Author: Joseph Myers <joseph@codesourcery.com>
Date:   Wed Feb 12 23:31:56 2020 +0000

    Avoid ldbl-96 stack corruption from range reduction of pseudo-zero (~~bug 25487~~).

    ~~Bug 25487~~ reports stack corruption in ldbl-96 sinl on a pseudo-zero
    argument (an representation where all the significand bits, including
    the explicit high bit, are zero, but the exponent is not zero, which
    is not a valid representation for the long double type).

    Although this is not a valid long double representation, existing
    practice in this area (see ~~bug 4586~~, originally marked invalid but
    subsequently fixed) is that we still seek to avoid invalid memory
    accesses as a result, in case of programs that treat arbitrary binary
    data as long double representations, although the invalid
    representations of the ldbl-96 format do not need to be consistently
    handled the same as any particular valid representation.

    This patch makes the range reduction detect pseudo-zero and unnormal
    representations that would otherwise go to __kernel_rem_pio2, and
    returns a NaN for them instead of continuing with the range reduction
    process.  (Pseudo-zero and unnormal representations whose unbiased
    exponent is less than -1 have already been safely returned from the
    function before this point without going through the rest of range
    reduction.)  Pseudo-zero representations would previously result in
    the value passed to __kernel_rem_pio2 being all-zero, which is
    definitely unsafe; unnormal representations would previously result in
    a value passed whose high bit is zero, which might well be unsafe
    since that is not a form of input expected by __kernel_rem_pio2.

    Tested for x86_64.

    (cherry picked from commit 9333498794cde1d5cca518badf79533a24114b6f)
```