Bug 27896 (CVE-2021-33574) - mq_notify does not handle separately allocated thread attributes (CVE-2021-33574)

Status: RESOLVED FIXED
Alias: CVE-2021-33574
Product: glibc
Component: nptl (show other bugs)
Version: 2.34
Importance: P2 normal
Target Milestone: 2.34
Assignee: Not yet assigned to anyone
URL:
Keywords:
Depends on:
Blocks:

Reported: 2021-05-21 08:54 UTC by Florian Weimer
Modified: 2021-08-02 00:51 UTC (History)
CC List: 7 users (show)
See Also:
Host:
Target:
Build:
Last reconfirmed:
Flags: siddhesh: security+

Attachments	
adapt for 2.28 (1.10 KB, application/mbox) 2021-06-17 02:00 UTC, liqingqing	Details
Add an attachment (proposed patch, testcase, etc.)	View All

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

Florian Weimer 2021-05-21 08:54:12 UTC — Description

```
mq_notify makes a shallow copy of pthread_attr_t here:

  if (notification->sigev_notify_attributes != NULL)
    {
      /* The thread attribute has to be allocated separately.  */
      data.attr = (pthread_attr_t *) malloc (sizeof (pthread_attr_t));
      if (data.attr == NULL)
        return -1;

      memcpy (data.attr, notification->sigev_notify_attributes,
              sizeof (pthread_attr_t));
    }

This introduces a potential for a use-after-free bug because the affinity mask has
been separately allocated, since before the addition of mq_notify. (A caller of
mq_notify can call pthread_attr_destroy immediately after mq_notify returns and
before the new thread is created.)

Found through code inspection. No known application impact.
```

Siddhesh Poyarekar 2021-05-31 06:42:58 UTC — Comment 1

```
Sorry I missed this in my Mitre CVE report: the only use-after-free indirection is
through the extensions member of struct pthread_attr (see
sysdeps/nptl/internaltypes.h) and it got introduced in glibc-2.32.  As a result,
only glibc-2.32 and glibc-2.33 have a use-after-free.
```

Siddhesh Poyarekar 2021-05-31 07:24:46 UTC — Comment 2

```
(In reply to Siddhesh Poyarekar from comment #1)
> Sorry I missed this in my Mitre CVE report: the only use-after-free
> indirection is through the extensions member of struct pthread_attr (see
> sysdeps/nptl/internaltypes.h) and it got introduced in glibc-2.32.  As a
> result, only glibc-2.32 and glibc-2.33 have a use-after-free.


Sorry again, this does in fact affect all versions of glibc because even though
extensions were introduced in 2.32, the cpuset before that were also an additional
indirection and hence would result in a similar use-after-free.
```

Andreas Schwab 2021-06-01 15:14:23 UTC — Comment 3

```
Fixed in 2.34.
```

Siddhesh Poyarekar 2021-06-03 05:54:38 UTC — Comment 4

```
Fixed commits:

https://sourceware.org/git/?
p=glibc.git;a=commit;h=42d3593505105O6b87101cf77202fefcbfc790cb
https://sourceware.org/git/?
p=glibc.git;a=commit;h=217b6dc298156bdb0d6aea9ea93e7e394a5ff091

A note on the security impact based on my analysis of the bug.  In
order to mount a minimal attack using this flaw, an attacker needs
many pre-requisites to be able to even crash a program using this
mq_notify bug:

1. The program call to mq_notify needs to be controlled by the attacker
2. The program must provide attributes to control creation of the
notification thread in mq_notify
3. The program must have the race condition where it may potentially
destroy the notification thread attributes before the notification
thread is created
4. The program must set CPU affinity or signal mask of the
notification thread to actually cause the use-after-free dereference

There are no known applications in distributions that have *all* these
pre-requisites and it's quite rare to have all of those conditions
together, so I reckon the attack complexity is very high.

In the worst (or theoretical) case that such an application exists, an
attacker would at best be able to control on which CPU the
notification thread runs; the signal mask, even if set is overwritten
(i.e. all signals unblocked) before the notification function is
called.  The change in scheduling should not have an impact on
confidentiality or integrity of a compliant application.
```

liqingqing 2021-06-04 03:56:21 UTC — Comment 5

```
hi all, what about the attr->stack?  I think if one thread can destroy cpuset,
means that it can also free the other memory.
```

Siddhesh Poyarekar 2021-06-04 04:19:41 UTC — Comment 6

The thread stack memory is not freed with pthread_attr_destroy; it continues to be reachable and valid since the thread would need it. Just like with regular threads, it is the responsibility of the application to ensure that the stack memory remains reachable and valid for the duration of the notification thread.

manojh3012 2021-06-16 17:43:27 UTC Comment 7

The suggested patch doesn't work for glibc 2.28 since `__pthread_attr_copy` is not available in that version. Any suggestions/fixes?

Florian Weimer 2021-06-16 18:18:43 UTC Comment 8

(In reply to manojh3012 from comment #7)
> The suggested patch doesn't work for glibc 2.28 since `__pthread_attr_copy`
> is not available in that version. Any suggestions/fixes?

You need to backport this additional commit:

commit 331c6e8a184167dd21a9f0b3fc165aeefea6eeca
Author: Florian Weimer <fweimer@redhat.com>
Date: Tue May 19 12:32:39 2020 +0200

    nptl: Add __pthread_attr_copy for copying pthread_attr_t objects

It should be self-contained.

manojh3012 2021-06-17 01:26:12 UTC Comment 9

Thanks but that commit uses __pthread_attr_setaffinity_np which is not available in 2.28 as well. I have to backport other commits looks like to brin in __pthread_attr_setaffinity_np. Any pointers on what is needed for that?

Also, is it possible to create a patch suitable for older glibc versions that don't have __pthread_attr_copy()?

liqingqing 2021-06-17 02:00:54 UTC Comment 10

Created attachment 13497 [details]
adapt for 2.28

hi, all, how about this two patches? ther are all from glibc upstream, and I had do some modification for the second patch.

```
diff --git a/sysdeps/unix/sysv/linux/mq_notify.c
b/sysdeps/unix/sysv/linux/mq_notify.c
index c4091169..76963567 100644
--- a/sysdeps/unix/sysv/linux/mq_notify.c
+++ b/sysdeps/unix/sysv/linux/mq_notify.c
@@ -260,7 +260,34 @@ mq_notify (mqd_t mqdes, const struct sigevent *notification)
       if (data.attr == NULL)
        return -1;

-      __pthread_attr_copy (data.attr, notification->sigev_notify_attributes);
+      memcpy (data.attr, notification->sigev_notify_attributes,
+            sizeof (pthread_attr_t));
+
+      struct pthread_attr *source =
+         (struct pthread_attr *) (notification->sigev_notify_attributes);
+      struct pthread_attr *target = (struct pthread_attr *) (data.attr);
+      cpu_set_t *newp;
+      cpu_set_t *cpuset = source->cpuset;
+      size_t cpusetsize = source->cpusetsize;
+
+      /* alloc a new memory for cpuset to avoid use after free */
+      if (cpuset != NULL && cpusetsize > 0)
+       {
+         newp = (cpu_set_t *) malloc (cpusetsize);
+         if (newp == NULL)
+           {
+             free(data.attr);
+             return -1;
+           }
+
+         memcpy (newp, cpuset, cpusetsize);
+         target->cpuset = newp;
+       }
+      else
+       {
+         target->cpuset = NULL;
+         target->cpusetsize = 0;
+       }
     }

   /* Construct the new request.  */
@@ -273,7 +300,7 @@ mq_notify (mqd_t mqdes, const struct sigevent *notification)
   int retval = INLINE_SYSCALL (mq_notify, 2, mqdes, &se);

   /* If it failed, free the allocated memory.  */
-  if (__glibc_unlikely (retval != 0))
+  if (retval != 0 && data.attr != NULL)
     {
       pthread_attr_destroy (data.attr);
       free (data.attr);
--
```

detailes:
https://gitee.com/src-openeuler/glibc/blob/openEuler-20.03-LTS-SP2/backport-CVE-2021-33574-0002-Fix-mq_notify-bug-27896.patch

liqingqing 2021-06-17 02:02:55 UTC Comment 11

(In reply to liqingqing from comment #10)
> Created attachment 13497 [details]
> adapt for 2.28
>
> hi, all, how about this two patches? ther are all from glibc upstream, and
> I had do some modification for the second patch.
>
>
> diff --git a/sysdeps/unix/sysv/linux/mq_notify.c
> b/sysdeps/unix/sysv/linux/mq_notify.c
> index c4091169..76963567 100644
> --- a/sysdeps/unix/sysv/linux/mq_notify.c
> +++ b/sysdeps/unix/sysv/linux/mq_notify.c
> @@ -260,7 +260,34 @@ mq_notify (mqd_t mqdes, const struct sigevent
> *notification)
>   if (data.attr == NULL)
>   return -1;
>
> -   __pthread_attr_copy (data.attr,
> notification->sigev_notify_attributes);
> +   memcpy (data.attr, notification->sigev_notify_attributes,
> +   sizeof (pthread_attr_t));
> +
> +   struct pthread_attr *source =
> +   (struct pthread_attr *) (notification->sigev_notify_attributes);
> +   struct pthread_attr *target = (struct pthread_attr *) (data.attr);

```
> +      cpu_set_t *newp;
> +      cpu_set_t *cpuset = source->cpuset;
> +      size_t cpusetsize = source->cpusetsize;
> +
> +      /* alloc a new memory for cpuset to avoid use after free */
> +      if (cpuset != NULL && cpusetsize > 0)
> +     {
> +       newp = (cpu_set_t *) malloc (cpusetsize);
> +       if (newp == NULL)
> +         {
> +           free(data.attr);
> +           return -1;
> +         }
> +
> +       memcpy (newp, cpuset, cpusetsize);
> +       target->cpuset = newp;
> +     }
> +      else
> +     {
> +       target->cpuset = NULL;
> +       target->cpusetsize = 0;
> +     }
>       }
>
>     /* Construct the new request.  */
> @@ -273,7 +300,7 @@ mq_notify (mqd_t mqdes, const struct sigevent
> *notification)
>     int retval = INLINE_SYSCALL (mq_notify, 2, mqdes, &se);
>
>     /* If it failed, free the allocated memory.  */
> -  if (__glibc_unlikely (retval != 0))
> +  if (retval != 0 && data.attr != NULL)
>       {
>         pthread_attr_destroy (data.attr);
>         free (data.attr);
> --
>
>
>
>
> detailes:
> https://gitee.com/src-openeuler/glibc/blob/openEuler-20.03-LTS-SP2/backport-
> CVE-2021-33574-0002-Fix-mq_notify-bug-27896.patch


the first one is commit: 42d3593505105O6b87101cf77202fefcbfc790cb :
https://gitee.com/src-openeuler/glibc/blob/openEuler-20.03-LTS-SP2/backport-
> CVE-2021-33574-0001-Fix-mq_notify-bug-27896.patch
```

Liming Liu 2021-08-01 15:54:44 UTC Comment 12

when will the 2.28 in buster be patched? thanks.

Carlos O'Donell 2021-08-02 00:51:52 UTC Comment 13

(In reply to Liming Liu from comment #12)
> when will the 2.28 in buster be patched? thanks.

This is the upstream glibc bug tracker. Please consider reporting your issue with the Debian glibc team. It looks like this has already been reported here: https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=989147, I would follow up there with Debian.