Subject: [PATCH] Fix OOPS in show_uevent()
Posted by Pavel Emelianov on Fri, 10 Aug 2007 10:13:43 GMT
View Forum Message <> Reply to Message

The platform_uevent() callback called via
  show_uevent()
  dev_uevent()
  platform_uevent()
forgot to set NULL to the last envp pointer and this caused the
show_uevent() oops while printing all the envp pointers like this:

BUG: unable to handle kernel paging request at virtual address 000280d0
...
Call Trace:
 [<c04d3d2d>] vsnprintf+0x2c7/0x48c
 [<c04d3f6d>] sprintf+0x17/0x19
 [<c052efa2>] show_uevent+0xeb/0x110
 [<c0457649>] buffered_rmqueue+0x1bf/0x1ed
 [<c04577b7>] get_page_from_freelist+0x82/0xa2
 [<c0457836>] __alloc_pages+0x5f/0x286
 [<c052eeb7>] show_uevent+0x0/0x110
 [<c052ec32>] dev_attr_show+0x15/0x1c
 [<c04aa7c3>] fill_read_buffer+0x57/0x89
 [<c04aa81d>] sysfs_read_file+0x28/0x53
 [<c0471762>] vfs_read+0x7f/0xef
 [<c04719f7>] sys_read+0x3c/0x62
 [<c0404bd6>] sysenter_past_esp+0x5f/0x99
...

The last hunk in this patch fixes this.

The other problem is that the envp passed to bus, type and platform callbacks
from dev_uevent() is the same, so the callbacks can overwrite the info, written
by the others. Did I miss something important?

This is for 2.6.23-rc2-mm1

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

--- ./drivers/base/core.c.ubug 2007-08-10 14:07:26.000000000 +0400
+++ ./drivers/base/core.c 2007-08-10 14:07:15.000000000 +0400
@@ -222,6 +222,11 @@ static int dev_uevent(struct kset *kset,
   if (retval)
    pr_debug ("%s: bus uevent() returned %d\n",
      __FUNCTION__, retval);
+

```
+  while (*envp != NULL) {
+   envp++;
+   num_envp--;
+  }
  }

  if (dev->class && dev->class->dev_uevent) {
@@ -230,6 +235,11 @@ static int dev_uevent(struct kset *kset,
   if (retval)
    pr_debug("%s: class uevent() returned %d\n",
      __FUNCTION__, retval);
+
+  while (*envp != NULL) {
+   envp++;
+   num_envp--;
+  }
  }

  if (dev->type && dev->type->uevent) {
@@ -238,6 +248,11 @@ static int dev_uevent(struct kset *kset,
   if (retval)
    pr_debug("%s: dev_type uevent() returned %d\n",
      __FUNCTION__, retval);
+
+  while (*envp != NULL) {
+   envp++;
+   num_envp--;
+  }
  }

  return retval;
--- ./drivers/base/platform.c.ubug 2007-08-10 14:07:44.000000000 +0400
+++ ./drivers/base/platform.c 2007-08-10 13:58:55.000000000 +0400
@@ -547,6 +547,7 @@ static int platform_uevent(struct device

  envp[0] = buffer;
  snprintf(buffer, buffer_size, "MODALIAS=%s", pdev->name);
+ envp[1] = NULL;
  return 0;
 }
```

Subject: Re: [PATCH] Fix OOPS in show_uevent()
Posted by Cornelia Huck on Fri, 10 Aug 2007 12:09:36 GMT
View Forum Message <> Reply to Message

On Fri, 10 Aug 2007 14:13:43 +0400,
Pavel Emelyanov <xemul@openvz.org> wrote:

> The last hunk in this patch fixes this.

Looks sane.

> The other problem is that the envp passed to bus, type and platform callbacks
> from dev_uevent() is the same, so the callbacks can overwrite the info, written
> by the others. Did I miss something important?

I don't think so.

>
> This is for 2.6.23-rc2-mm1
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
> ---
>
> --- ./drivers/base/core.c.ubug 2007-08-10 14:07:26.000000000 +0400
> +++ ./drivers/base/core.c 2007-08-10 14:07:15.000000000 +0400
> @@ -222,6 +222,11 @@ static int dev_uevent(struct kset *kset,
>   if (retval)
>   pr_debug ("%s: bus uevent() returned %d\n",
>   __FUNCTION__, retval);
> +
> + while (*envp != NULL) {
> +   envp++;
> +   num_envp--;
> + }

Hm, we may also want to adjust the remaining buffer size here, but only
the ->uevent() method has that knowledge...

>   }
>
>   if (dev->class && dev->class->dev_uevent) {
> @@ -230,6 +235,11 @@ static int dev_uevent(struct kset *kset,
>   if (retval)
>   pr_debug("%s: class uevent() returned %d\n",
>   __FUNCTION__, retval);
> +
> + while (*envp != NULL) {
> +   envp++;
> +   num_envp--;
> + }
>   }
>
>   if (dev->type && dev->type->uevent) {

> @@ -238,6 +248,11 @@ static int dev_uevent(struct kset *kset,
>   if (retval)
>   pr_debug("%s: dev_type uevent() returned %d\n",
>   __FUNCTION__, retval);
> +
> + while (*envp != NULL) {
> +   envp++;
> +   num_envp--;
> + }
>   }
>
>   return retval;
> --- ./drivers/base/platform.c.ubug 2007-08-10 14:07:44.000000000 +0400
> +++ ./drivers/base/platform.c 2007-08-10 13:58:55.000000000 +0400
> @@ -547,6 +547,7 @@ static int platform_uevent(struct device
>
>   envp[0] = buffer;
>   snprintf(buffer, buffer_size, "MODALIAS=%s", pdev->name);
> + envp[1] = NULL;
>   return 0;
> }
>

Subject: Re: [PATCH] Fix OOPS in show_uevent()
Posted by Kay Sievers on Fri, 10 Aug 2007 12:23:56 GMT
View Forum Message <> Reply to Message

On 8/10/07, Pavel Emelyanov <xemul@openvz.org> wrote:
> The platform_uevent() callback called via
>   show_uevent()
>   dev_uevent()
>   platform_uevent()
> forgot to set NULL to the last envp pointer and this caused the
> show_uevent() oops while printing all the envp pointers like this:

> The last hunk in this patch fixes this.

Looks like the right fix, yes.

> The other problem is that the envp passed to bus, type and platform callbacks
> from dev_uevent() is the same, so the callbacks can overwrite the info, written
> by the others. Did I miss something important?

Sounds like a bug, yes.

But we still don't update the remaining buffer size and the remaining
array fields which are left after the call. Shouldn't we instead just

change the:
```
  int (*dev_uevent)(struct device *dev,
                    char **envp, int num_envp,
                    char *buffer, int buffer_size);
```
to:
```
  int (*dev_uevent)(struct device *dev,
                    char **envp, int num_envp, int *cur_index,
                    char *buffer, int buffer_size, int *cur_len);
```

like we do for:
```
  int add_uevent_var(char **envp, int num_envp, int *cur_index,
                     char *buffer, int buffer_size, int *cur_len,
                     const char *format, ...)
```

and along with the change of the callers, we would update the values
properly, so the next call has the correct numbers? There are 6
classes and something like 12 buses using this method, so it shouldn't
be too much trouble.

Thanks,
Kay

Subject: Re: [PATCH] Fix OOPS in show_uevent()
Posted by Cornelia Huck on Fri, 10 Aug 2007 12:39:27 GMT
View Forum Message <> Reply to Message

On Fri, 10 Aug 2007 14:23:56 +0200,
"Kay Sievers" <kay.sievers@vrfy.org> wrote:

> But we still don't update the remaining buffer size and the remaining
> array fields which are left after the call. Shouldn't we instead just
> change the:
>   int (*dev_uevent)(struct device *dev,
>   char **envp, int num_envp,
>   char *buffer, int buffer_size);
> to:
>   int (*dev_uevent)(struct device *dev,
>   char **envp, int num_envp, int *cur_index,
>   char *buffer, int buffer_size, int *cur_len);
>
> like we do for:
>   int add_uevent_var(char **envp, int num_envp, int *cur_index,
>   char *buffer, int buffer_size, int *cur_len,
>   const char *format, ...)
>
> and along with the change of the callers, we would update the values
> properly, so the next call has the correct numbers? There are 6

> classes and something like 12 buses using this method, so it shouldn't
> be too much trouble.

Sounds like a sensible approach. We may want the remaining non-users to
add_uevent_var() at the same time, I guess.

Subject: Re: [PATCH] Fix OOPS in show_uevent()
Posted by Pavel Emelianov on Fri, 10 Aug 2007 13:21:51 GMT
View Forum Message <> Reply to Message

Cornelia Huck wrote:
> On Fri, 10 Aug 2007 14:23:56 +0200,
> "Kay Sievers" <kay.sievers@vrfy.org> wrote:
>
>> But we still don't update the remaining buffer size and the remaining
>> array fields which are left after the call. Shouldn't we instead just
>> change the:
>> int (*dev_uevent)(struct device *dev,
>> char **envp, int num_envp,
>> char *buffer, int buffer_size);
>> to:
>> int (*dev_uevent)(struct device *dev,
>> char **envp, int num_envp, int *cur_index,
>> char *buffer, int buffer_size, int *cur_len);
>>
>> like we do for:
>> int add_uevent_var(char **envp, int num_envp, int *cur_index,
>> char *buffer, int buffer_size, int *cur_len,
>> const char *format, ...)
>>
>> and along with the change of the callers, we would update the values
>> properly, so the next call has the correct numbers? There are 6
>> classes and something like 12 buses using this method, so it shouldn't
>> be too much trouble.

isn't it better to change
  int (*dev_uevent)(struct device *dev,
  char **envp, int num_envp,
  char *buffer, int buffer_size);
to
  int (*dev_uevent)(struct device *dev,
  char **envp, int num_envp,
  char **buffer);
and alter the buffer pointer inside?

> Sounds like a sensible approach. We may want the remaining non-users to

> add_uevent_var() at the same time, I guess.
>

Thanks,
Pavel

Subject: Re: [PATCH] Fix OOPS in show_uevent()
Posted by Cornelia Huck on Fri, 10 Aug 2007 13:37:28 GMT
View Forum Message <> Reply to Message

On Fri, 10 Aug 2007 17:21:51 +0400,
Pavel Emelyanov <xemul@openvz.org> wrote:

> Cornelia Huck wrote:
> > On Fri, 10 Aug 2007 14:23:56 +0200,
> > "Kay Sievers" <kay.sievers@vrfy.org> wrote:
> >
> >> But we still don't update the remaining buffer size and the remaining
> >> array fields which are left after the call. Shouldn't we instead just
> >> change the:
> >>   int (*dev_uevent)(struct device *dev,
> >>   char **envp, int num_envp,
> >>   char *buffer, int buffer_size);
> >> to:
> >>   int (*dev_uevent)(struct device *dev,
> >>   char **envp, int num_envp, int *cur_index,
> >>   char *buffer, int buffer_size, int *cur_len);
> >>
> >> like we do for:
> >>   int add_uevent_var(char **envp, int num_envp, int *cur_index,
> >>   char *buffer, int buffer_size, int *cur_len,
> >>   const char *format, ...)
> >>
> >> and along with the change of the callers, we would update the values
> >> properly, so the next call has the correct numbers? There are 6
> >> classes and something like 12 buses using this method, so it shouldn't
> >> be too much trouble.
> >
> > isn't it better to change
> >   int (*dev_uevent)(struct device *dev,
> >   char **envp, int num_envp,
> >   char *buffer, int buffer_size);
> > to
> >   int (*dev_uevent)(struct device *dev,
> >   char **envp, int num_envp,
> >   char **buffer);
> > and alter the buffer pointer inside?

But the function wants to know the buffer_size, doesn't it?
(And the caller can make the adjustments easily; it saves duplicated code.)