Subject: [PATCH] usb: Fixup usb so it uses struct pid
Posted by ebiederm on Sun, 10 Sep 2006 04:42:10 GMT
View Forum Message <> Reply to Message

The problem by remember a user space process by it's pid it is
possible that the process will exit, pid wrap around will occur and a
different process will appear in it's place. Holding a reference
to a struct pid avoid that problem, and paves the way
for implementing a pid namespace.

Also since usb is the only user of kill_proc_info_as_uid
rename kill_proc_info_as_uid to kill_pid_info_as_uid
and have the new version take a struct pid.

This patch is against 2.6.18-rc6-mm1.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 drivers/usb/core/devio.c | 10 ++++++----
 drivers/usb/core/inode.c | 2 +-
 drivers/usb/core/usb.h | 2 +-
 include/linux/sched.h | 2 +-
 kernel/signal.c | 8 ++++----
 5 files changed, 13 insertions(+), 11 deletions(-)

diff --git a/drivers/usb/core/devio.c b/drivers/usb/core/devio.c
index 3ef2778..93087c7 100644
--- a/drivers/usb/core/devio.c
+++ b/drivers/usb/core/devio.c
@@ -65,7 +65,7 @@ DEFINE_MUTEX(usbfs_mutex);
 struct async {
  struct list_head asynclist;
  struct dev_state *ps;
- pid_t pid;
+ struct pid *pid;
  uid_t uid, euid;
  unsigned int signr;
  unsigned int ifnum;
@@ -225,6 +225,7 @@ static struct async *alloc_async(unsigne

 static void free_async(struct async *as)
 {
+ put_pid(as->pid);
  kfree(as->urb->transfer_buffer);
  kfree(as->urb->setup_packet);
  usb_free_urb(as->urb);
@@ -317,7 +318,7 @@ static void async_completed(struct urb *
   sinfo.si_errno = as->urb->status;

```
   sinfo.si_code = SI_ASYNCIO;
   sinfo.si_addr = as->userurb;
-  kill_proc_info_as_uid(as->signr, &sinfo, as->pid, as->uid,
+  kill_pid_info_as_uid(as->signr, &sinfo, as->pid, as->uid,
          as->euid, as->secid);
  }
  snoop(&urb->dev->dev, "urb complete\n");
@@ -572,7 +573,7 @@ static int usbdev_open(struct inode *ino
  INIT_LIST_HEAD(&ps->async_completed);
  init_waitqueue_head(&ps->wait);
  ps->discsignr = 0;
- ps->disc_pid = current->pid;
+ ps->disc_pid = get_pid(task_pid(current));
  ps->disc_uid = current->uid;
  ps->disc_euid = current->euid;
  ps->disccontext = NULL;
@@ -610,6 +611,7 @@ static int usbdev_release(struct inode *
  usb_autosuspend_device(dev, 1);
  usb_unlock_device(dev);
  usb_put_dev(dev);
+ put_pid(ps->disc_pid);
  kfree(ps);
  return 0;
 }
@@ -1062,7 +1064,7 @@ static int proc_do_submiturb(struct dev_
   as->userbuffer = NULL;
  as->signr = uurb->signr;
  as->ifnum = ifnum;
- as->pid = current->pid;
+ as->pid = get_pid(task_pid(current));
  as->uid = current->uid;
  as->euid = current->euid;
  security_task_getsecid(current, &as->secid);
diff --git a/drivers/usb/core/inode.c b/drivers/usb/core/inode.c
index 7c77c2d..b5d6a79 100644
--- a/drivers/usb/core/inode.c
+++ b/drivers/usb/core/inode.c
@@ -699,7 +699,7 @@ static void usbfs_remove_device(struct u
    sinfo.si_errno = EPIPE;
    sinfo.si_code = SI_ASYNCIO;
    sinfo.si_addr = ds->disccontext;
-   kill_proc_info_as_uid(ds->discsignr, &sinfo, ds->disc_pid, ds->disc_uid, ds->disc_euid,
ds->secid);
+   kill_pid_info_as_uid(ds->discsignr, &sinfo, ds->disc_pid, ds->disc_uid, ds->disc_euid,
ds->secid);
   }
  }
 }
```

```
diff --git a/drivers/usb/core/usb.h b/drivers/usb/core/usb.h
index e8bc2e4..e324164 100644
--- a/drivers/usb/core/usb.h
+++ b/drivers/usb/core/usb.h
@@ -122,7 +122,7 @@ struct dev_state {
  struct list_head async_completed;
  wait_queue_head_t wait;     /* wake up if a request completed */
  unsigned int discsignr;
- pid_t disc_pid;
+ struct pid *disc_pid;
  uid_t disc_uid, disc_euid;
  void __user *disccontext;
  unsigned long ifclaimed;
diff --git a/include/linux/sched.h b/include/linux/sched.h
index f11b874..36031ef 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1303,12 +1303,12 @@ extern int force_sig_info(int, struct si
 extern int __kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp);
 extern int kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp);
 extern int kill_pid_info(int sig, struct siginfo *info, struct pid *pid);
+extern int kill_pid_info_as_uid(int, struct siginfo *, struct pid *, uid_t, uid_t, u32);
 extern int kill_pgrp(struct pid *pid, int sig, int priv);
 extern int kill_pid(struct pid *pid, int sig, int priv);
 extern int __kill_pg_info(int sig, struct siginfo *info, pid_t pgrp);
 extern int kill_pg_info(int, struct siginfo *, pid_t);
 extern int kill_proc_info(int, struct siginfo *, pid_t);
-extern int kill_proc_info_as_uid(int, struct siginfo *, pid_t, uid_t, uid_t, u32);
 extern void do_notify_parent(struct task_struct *, int);
 extern void force_sig(int, struct task_struct *);
 extern void force_sig_specific(int, struct task_struct *);
diff --git a/kernel/signal.c b/kernel/signal.c
index e04fd9e..0175b24 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -1260,8 +1260,8 @@ kill_proc_info(int sig, struct siginfo *
  return error;
 }

-/* like kill_proc_info(), but doesn't use uid/euid of "current" */
-int kill_proc_info_as_uid(int sig, struct siginfo *info, pid_t pid,
+/* like kill_pid_info(), but doesn't use uid/euid of "current" */
+int kill_pid_info_as_uid(int sig, struct siginfo *info, struct pid *pid,
         uid_t uid, uid_t euid, u32 secid)
 {
  int ret = -EINVAL;
@@ -1271,7 +1271,7 @@ int kill_proc_info_as_uid(int sig, struc
   return ret;
```

```
  read_lock(&tasklist_lock);
- p = find_task_by_pid(pid);
+ p = pid_task(pid, PIDTYPE_PID);
  if (!p) {
   ret = -ESRCH;
   goto out_unlock;
@@ -1295,7 +1295,7 @@ out_unlock:
  read_unlock(&tasklist_lock);
  return ret;
 }
-EXPORT_SYMBOL_GPL(kill_proc_info_as_uid);
+EXPORT_SYMBOL_GPL(kill_pid_info_as_uid);

 /*
  * kill_something_info() interprets pid in interesting ways just like kill(2).
--
1.4.2.rc3.g7e18e-dirty
```

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [PATCH] usb: Fixup usb so it uses struct pid
Posted by Pete Zaitcev on Sun, 10 Sep 2006 18:12:49 GMT
View Forum Message <> Reply to Message

On Sat, 09 Sep 2006 22:42:10 -0600, ebiederm@xmission.com (Eric W. Biederman) wrote:

> The problem by remember a user space process by it's pid it is
> possible that the process will exit, pid wrap around will occur and a
> different process will appear in it's place.

... which is completely all right in this case. We used to have an
implementation which tried to hold onto the task_struct and that sucked.
It is only possible for the task to disappear without notifying devio
under very special conditions only, which involve forking with parent
exiting. In other words, even a buggy application won't trigger this
without deliberately trying. And when it happens, uid checks make sure
that other users are not affected.

> Holding a reference
> to a struct pid avoid that problem, and paves the way
> for implementing a pid namespace.

That may be useful.

The patch itself seems straightforward if we can trust your struct
pid thingies. If OpenVZ people approve, I don't mind.

-- Pete

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [PATCH] usb: Fixup usb so it uses struct pid
Posted by ebiederm on Sun, 10 Sep 2006 20:04:06 GMT
View Forum Message <> Reply to Message

Pete Zaitcev <zaitcev@redhat.com> writes:

> On Sat, 09 Sep 2006 22:42:10 -0600, ebiederm@xmission.com (Eric W. Biederman)
> wrote:
>
>> The problem by remember a user space process by it's pid it is
>> possible that the process will exit, pid wrap around will occur and a
>> different process will appear in it's place.
>
> ... which is completely all right in this case. We used to have an
> implementation which tried to hold onto the task_struct and that sucked.
> It is only possible for the task to disappear without notifying devio
> under very special conditions only, which involve forking with parent
> exiting. In other words, even a buggy application won't trigger this
> without deliberately trying. And when it happens, uid checks make sure
> that other users are not affected.

Right. I looked to see how hard it was in the usb case, but since
you are in the open and release case I can see it being hard. I think
this case can also be triggered by file descriptor passing, as that
is another subtle way to dup a file descriptor.

The uid checks keep the current situation from being a security
hole but it is still possible to confuse user space, although you
should be able to do that much more simply by just sending the signal
yourself :)

>> Holding a reference
>> to a struct pid avoid that problem, and paves the way
>> for implementing a pid namespace.
>
> That may be useful.
>

> The patch itself seems straightforward if we can trust your struct
> pid thingies. If OpenVZ people approve, I don't mind.

So far I haven't seen any complaints on that score. None from
the mainstream kernel folks the vserver guys or the OpenVz guys.
struct pid itself is in 2.6.18, performing this same function for
proc, but not all of the helper functions have made it beyond -mm
yet. Most of the rest should make it into 2.6.19.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [PATCH] usb: Fixup usb so it uses struct pid
Posted by Greg KH on Sun, 10 Sep 2006 21:37:46 GMT
View Forum Message <> Reply to Message

On Sat, Sep 09, 2006 at 10:42:10PM -0600, Eric W. Biederman wrote:
>
> The problem by remember a user space process by it's pid it is
> possible that the process will exit, pid wrap around will occur and a
> different process will appear in it's place. Holding a reference
> to a struct pid avoid that problem, and paves the way
> for implementing a pid namespace.
>
> Also since usb is the only user of kill_proc_info_as_uid
> rename kill_proc_info_as_uid to kill_pid_info_as_uid
> and have the new version take a struct pid.
>
> This patch is against 2.6.18-rc6-mm1.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Looks good to me.

Do you want me to take this in my tree, or will you be going through
Andrew, like your other, related pid stuff? If through Andrew, please
feel free to add:
Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

thanks,

greg k-h

_____
Containers mailing list

Subject: Re: [PATCH] usb: Fixup usb so it uses struct pid
Posted by ebiederm on Sun, 10 Sep 2006 23:58:33 GMT
View Forum Message <> Reply to Message

Greg KH <greg@kroah.com> writes:

> On Sat, Sep 09, 2006 at 10:42:10PM -0600, Eric W. Biederman wrote:
>>
>> The problem by remember a user space process by it's pid it is
>> possible that the process will exit, pid wrap around will occur and a
>> different process will appear in it's place. Holding a reference
>> to a struct pid avoid that problem, and paves the way
>> for implementing a pid namespace.
>>
>> Also since usb is the only user of kill_proc_info_as_uid
>> rename kill_proc_info_as_uid to kill_pid_info_as_uid
>> and have the new version take a struct pid.
>>
>> This patch is against 2.6.18-rc6-mm1.
>>
>> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
>
> Looks good to me.
>
> Do you want me to take this in my tree, or will you be going through
> Andrew, like your other, related pid stuff? If through Andrew, please
> feel free to add:
> Signed-off-by: Greg Kroah-Hartman <gregkh@suse.de>

Sure. I think going through Andrew makes sense. As I probably have
a helper function or two in Andrews tree that hasn't hit wider distribution.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [PATCH] usb: Fixup usb so it uses struct pid
Posted by Herbert Poetzl on Mon, 11 Sep 2006 14:02:50 GMT
View Forum Message <> Reply to Message

On Sun, Sep 10, 2006 at 11:12:49AM -0700, Pete Zaitcev wrote:
> On Sat, 09 Sep 2006 22:42:10 -0600, ebiederm@xmission.com (Eric W. Biederman) wrote:
>
> > The problem by remember a user space process by it's pid it is
> > possible that the process will exit, pid wrap around will occur and a
> > different process will appear in it's place.
>
> ... which is completely all right in this case. We used to have an
> implementation which tried to hold onto the task_struct and that sucked.
> It is only possible for the task to disappear without notifying devio
> under very special conditions only, which involve forking with parent
> exiting. In other words, even a buggy application won't trigger this
> without deliberately trying. And when it happens, uid checks make sure
> that other users are not affected.
>
> > Holding a reference
> > to a struct pid avoid that problem, and paves the way
> > for implementing a pid namespace.
>
> That may be useful.
>
> The patch itself seems straightforward if we can trust your struct
> pid thingies. If OpenVZ people approve, I don't mind.

perfectly fine from my side

best,
Herbert

> -- Pete
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers