Subject: [PATCH] vt: Rework the console spawning variables.
Posted by ebiederm on Sun, 10 Sep 2006 04:21:57 GMT
View Forum Message <> Reply to Message

This patch is against 2.6.18-rc6-mm1 with:
vt-update-spawnpid-to-be-a-struct-pid_t-tidy and
vt-update-spawnpid-to-be-a-struct-pid_t reverted. My previous version
of this patch did not correct the style that was already in use, and
failed to add proper SMP locking.

This is such a rare path it took me a while to figure out how to test
this after soring out the locking.

This patch does several things.
- The variables used are moved into a structure and declared in vt_kern.h
- A spinlock is added so we don't have SMP races updating the values.
- Instead of raw pid_t value a struct_pid is used to guard against
  pid wrap around issues, if the daemon to spawn a new console dies.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 drivers/char/keyboard.c | 16 ++++++++++++----
 drivers/char/vt_ioctl.c | 9 ++++++---
 include/linux/vt_kern.h | 7 +++++++
 3 files changed, 25 insertions(+), 7 deletions(-)

diff --git a/drivers/char/keyboard.c b/drivers/char/keyboard.c
index 3e90aac..99fb070 100644
--- a/drivers/char/keyboard.c
+++ b/drivers/char/keyboard.c
@@ -108,7 +108,11 @@ const int NR_TYPES = ARRAY_SIZE(max_vals
 struct kbd_struct kbd_table[MAX_NR_CONSOLES];
 static struct kbd_struct *kbd = kbd_table;

-int spawnpid, spawnsig;
+struct vt_spawn_console vt_spawn_con = {
+ .lock = SPIN_LOCK_UNLOCKED,
+ .pid = NULL,
+ .sig = 0,
+};

 /*
  * Variables exported for vt.c
@@ -578,9 +582,13 @@ static void fn_compose(struct vc_data *v

 static void fn_spawn_con(struct vc_data *vc, struct pt_regs *regs)
 {
- if (spawnpid)

```
-  if (kill_proc(spawnpid, spawnsig, 1))
-   spawnpid = 0;
+ spin_lock(&vt_spawn_con.lock);
+ if (vt_spawn_con.pid)
+  if (kill_pid(vt_spawn_con.pid, vt_spawn_con.sig, 1)) {
+   put_pid(vt_spawn_con.pid);
+   vt_spawn_con.pid = NULL;
+  }
+ spin_unlock(&vt_spawn_con.lock);
 }

 static void fn_SAK(struct vc_data *vc, struct pt_regs *regs)
diff --git a/drivers/char/vt_ioctl.c b/drivers/char/vt_ioctl.c
index a53e382..dc408af 100644
--- a/drivers/char/vt_ioctl.c
+++ b/drivers/char/vt_ioctl.c
@@ -645,13 +645,16 @@ #endif
   */
  case KDSIGACCEPT:
  {
-  extern int spawnpid, spawnsig;
   if (!perm || !capable(CAP_KILL))
    return -EPERM;
   if (!valid_signal(arg) || arg < 1 || arg == SIGKILL)
    return -EINVAL;
-  spawnpid = current->pid;
-  spawnsig = arg;
+
+  spin_lock_irq(&vt_spawn_con.lock);
+  put_pid(vt_spawn_con.pid);
+  vt_spawn_con.pid = get_pid(task_pid(current));
+  vt_spawn_con.sig = arg;
+  spin_unlock_irq(&vt_spawn_con.lock);
   return 0;
  }

diff --git a/include/linux/vt_kern.h b/include/linux/vt_kern.h
index 1009d3f..37a1a41 100644
--- a/include/linux/vt_kern.h
+++ b/include/linux/vt_kern.h
@@ -84,4 +84,11 @@ #define CON_BUF_SIZE (CONFIG_BASE_SMALL
 extern char con_buf[CON_BUF_SIZE];
 extern struct semaphore con_buf_sem;

+struct vt_spawn_console {
+ spinlock_t lock;
+ struct pid *pid;
+ int sig;
```

```
+};
+extern struct vt_spawn_console vt_spawn_con;
+
 #endif /* _VT_KERN_H */
--
1.4.2.rc3.g7e18e-dirty
```

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

Subject: Re: [PATCH] vt: Rework the console spawning variables.
Posted by Oleg Nesterov on Sun, 10 Sep 2006 14:29:42 GMT
View Forum Message <> Reply to Message

On 09/09, Eric W. Biederman wrote:
>
> This patch does several things.
> - The variables used are moved into a structure and declared in vt_kern.h
> - A spinlock is added so we don't have SMP races updating the values.
> - Instead of raw pid_t value a struct_pid is used to guard against
>   pid wrap around issues, if the daemon to spawn a new console dies.

I am not arguing against this patch, but it's a pity we can't use 'struct pid'
lockless. What dou you think about this:

```
 void delayed_free_pid(struct rcu_head *rhp)
 {
  struct pid *pid = container_of(rhp, struct pid, rcu);
  kmem_cache_free(pid_cachep, pid);
 }

 void put_pid_rcu(struct pid *pid)
 {
  if (atomic_dec_and_test(&pid->count))
   // this can happen only if delayed_put_pid()
   // was already fired, we can re-use pid->rcu
   call_rcu(&pid->rcu, delayed_free_pid);
 }
```

Now,

```
 update_pid()
 {
  // still needs some locking
  put_pid_rcu(pid);
```

```
 pid = get_pid(...);
}

 use_pid()
{
  rcu_read_lock();
  do_something(pid);
  rcu_read_unock();
}
```

Thoughts?

Oleg.

Subject: Re: [PATCH] vt: Rework the console spawning variables.
Posted by ebiederm on Sun, 10 Sep 2006 20:10:37 GMT
View Forum Message <> Reply to Message

Oleg Nesterov <oleg@tv-sign.ru> writes:

> On 09/09, Eric W. Biederman wrote:
>>
>> This patch does several things.
>> - The variables used are moved into a structure and declared in vt_kern.h
>> - A spinlock is added so we don't have SMP races updating the values.
>> - Instead of raw pid_t value a struct_pid is used to guard against
>>   pid wrap around issues, if the daemon to spawn a new console dies.
>
> I am not arguing against this patch, but it's a pity we can't use 'struct pid'
> lockless. What dou you think about this:

Actually with xchg I can use a reference counted struct pid lockless.
In the general case you have more then one variable you want to keep
in sync and you need the lock for that.

rcu is definitely not the solution in these cases as the struct pid
is stored for a long time so we need the reference count.

It might make sense to have some helper code makes that wraps
the following line so it is obvious you can do this.

put_pid(xchg(&vc->vt_pid, get_pid(task_pid(current))));

Perhaps:
```
void update_pid(struct pid **ref, struct pid *new)
{
	struct pid *old;
	get_pid(new);
	old = xchg(ref, new);
	put_pid(old);
}
```

But since I can write it as a moderately clear one liner in the
case that matters I don't much care.

Eric

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

Subject: Re: [PATCH] vt: Rework the console spawning variables.
Posted by Oleg Nesterov on Sun, 10 Sep 2006 20:33:42 GMT
View Forum Message <> Reply to Message

On 09/10, Eric W. Biederman wrote:
>
> Oleg Nesterov <oleg@tv-sign.ru> writes:
>
> > On 09/09, Eric W. Biederman wrote:
> >>
> >> This patch does several things.
> >> - The variables used are moved into a structure and declared in vt_kern.h
> >> - A spinlock is added so we don't have SMP races updating the values.
> >> - Instead of raw pid_t value a struct_pid is used to guard against
> >>   pid wrap around issues, if the daemon to spawn a new console dies.
> >
> > I am not arguing against this patch, but it's a pity we can't use 'struct pid'
> > lockless. What dou you think about this:
>
> Actually with xchg I can use a reference counted struct pid lockless.
>
> ...
>
> Perhaps:
> void update_pid(struct pid **ref, struct pid *new)
> {
>	struct pid *old;
>	get_pid(new);

> old = xchg(ref, new);
> put_pid(old);
> }

This can't work. This put_pid() can actually free the memory, while
'old' is still in use (lockless).

> rcu is definitely not the solution in these cases as the struct pid
> is stored for a long time so we need the reference count.

Surely we need the reference count, I don't understand you.
Look at put_pid_rcu().

That said,

> In the general case you have more then one variable you want to keep
> in sync and you need the lock for that.

Yes.

> But since I can write it as a moderately clear one liner in the
> case that matters I don't much care.

Ok.

Oleg.

Subject: Re: [PATCH] vt: Rework the console spawning variables.
Posted by ebiederm on Sun, 10 Sep 2006 22:56:37 GMT
View Forum Message <> Reply to Message

Oleg Nesterov <oleg@tv-sign.ru> writes:

> On 09/10, Eric W. Biederman wrote:
>>
>> Oleg Nesterov <oleg@tv-sign.ru> writes:
>>
>> > On 09/09, Eric W. Biederman wrote:
>> >>
>> >> This patch does several things.
>> >> - The variables used are moved into a structure and declared in vt_kern.h
>> >> - A spinlock is added so we don't have SMP races updating the values.

>> >> - Instead of raw pid_t value a struct_pid is used to guard against
>> >>   pid wrap around issues, if the daemon to spawn a new console dies.
>> >
>> > I am not arguing against this patch, but it's a pity we can't use 'struct
> pid'
>> > lockless. What dou you think about this:
>>
>> Actually with xchg I can use a reference counted struct pid lockless.
>>
>> ...

Ok. I think I see the where the confusion is. We were looking
at different parts of the puzzle. But I we need to resolve this
to make certain I didn't do something clever and racy.

So I have been focusing on the actual update of the struct pid
pointer, and making certain we don't get the wrong reference
count.

You have been looking at the use of the struct pid pointer,
and saying we can't free it lockless because then we would
have a chance of walking off of the pointer in the context
where we use it.

I think you have spotted a legitimate bug in the idiom I was using.

>> Perhaps:
>> void update_pid(struct pid **ref, struct pid *new)
>> {
>>   struct pid *old;
>>   get_pid(new);
>>   old = xchg(ref, new);
>>   put_pid(old);
>> }
>
> This can't work. This put_pid() can actually free the memory, while
> 'old' is still in use (lockless).

Agreed, and the above is the expanded form of my one liner.
So I was to clever and missed a case :(

At least I found the fundamental race.

>> rcu is definitely not the solution in these cases as the struct pid
>> is stored for a long time so we need the reference count.
>
> Surely we need the reference count, I don't understand you.
> Look at put_pid_rcu().

Sorry I was looking at the wrong half of the picture.

The summary is I need to rethink how I handle reset_vc when called
from interrupt context for handling do_sak.

I think the simplest method to make it race free is simply to make the
code run in process context where we can take a blocking semaphore.

As for the rest of your suggestion it would not be hard to be able to
follow a struct pid pointer in an rcu safe way, and we do in the pid
hash table. In other contexts so far I always have other variables
that need to be updated in concert, so there isn't a point in coming
up with a lockless implementation. I believe vt_pid is the only
case that I have run across where this is a problem and I have
at least preliminary patches for every place where signals are
sent.

Updating this old code is painful.

Eric

_____

Subject: Re: [PATCH] vt: Rework the console spawning variables.
Posted by Oleg Nesterov on Mon, 11 Sep 2006 01:05:34 GMT
View Forum Message <> Reply to Message

On 09/10, Eric W. Biederman wrote:
>
> Ok. I think I see the where the confusion is. We were looking
> at different parts of the puzzle. But I we need to resolve this
> to make certain I didn't do something clever and racy.

Yes, I think we misunderstood each other :)

> As for the rest of your suggestion it would not be hard to be able to
> follow a struct pid pointer in an rcu safe way, and we do in the pid
> hash table. In other contexts so far I always have other variables
> that need to be updated in concert, so there isn't a point in coming
> up with a lockless implementation. I believe vt_pid is the only
> case that I have run across where this is a problem and I have
> at least preliminary patches for every place where signals are
> sent.
>

> Updating this old code is painful.

No, no, we shouldn't change the old code, it is fine.

Just in case, to avoid any possible confusion.

put_pid(pid) has the following restrictions. The caller should ensure
that any other possible reference to this pid "owns" it (did get_pid()).

So we can add a new helper, put_pid_rcu(). It is ok if this pid is used
in parallel under rcu_read_lock() without bumping pid->count. Contrary,
the only restriction those users must not call get_pid(pid).

But yes, you are right, I don't see an immediate usage of put_pid_rcu().

Oleg.

Subject: Re: [PATCH] vt: Rework the console spawning variables.
Posted by ebiederm on Mon, 11 Sep 2006 02:40:29 GMT
View Forum Message <> Reply to Message

Oleg Nesterov <oleg@tv-sign.ru> writes:

> On 09/10, Eric W. Biederman wrote:
>>
>> Ok. I think I see the where the confusion is. We were looking
>> at different parts of the puzzle. But I we need to resolve this
>> to make certain I didn't do something clever and racy.
>
> Yes, I think we misunderstood each other :)
>
>> As for the rest of your suggestion it would not be hard to be able to
>> follow a struct pid pointer in an rcu safe way, and we do in the pid
>> hash table. In other contexts so far I always have other variables
>> that need to be updated in concert, so there isn't a point in coming
>> up with a lockless implementation. I believe vt_pid is the only
>> case that I have run across where this is a problem and I have
>> at least preliminary patches for every place where signals are
>> sent.
>>
>> Updating this old code is painful.

>
> No, no, we shouldn't change the old code, it is fine.
>
So what happens when:
cpu0: cpu1:
kill_pid(vt_pid,....) fn_SAK()->vc_reset()->put_pid(xchg(&vt_pid, NULL))

Can't kill_pid dereference vt_pid after put_pid is called?

It's a microscopic window, and requires the user to attempt a vt switch
and a sak simultaneously but I think it is there.

Eric

_____

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

Subject: Re: [PATCH] vt: Rework the console spawning variables.
Posted by Oleg Nesterov on Mon, 11 Sep 2006 02:59:40 GMT
View Forum Message <> Reply to Message

On 09/10, Eric W. Biederman wrote:
>
> Oleg Nesterov <oleg@tv-sign.ru> writes:
>
> > On 09/10, Eric W. Biederman wrote:
> >>
> >> Updating this old code is painful.
> >
> > No, no, we shouldn't change the old code, it is fine.
> >
> So what happens when:
> cpu0: cpu1:
> kill_pid(vt_pid,....) fn_SAK()->vc_reset()->put_pid(xchg(&vt_pid, NULL))
>
> Can't kill_pid dereference vt_pid after put_pid is called?

Ah, I didn't consider that patch as 'old code', sorry :)

I don't understand drivers/char/vt*, but surely put_pid(xchg()) can't work.
Again, unless we have a lock to serialize access to ->vt_pid, but in that
case we don't need xchg().

Oleg.

_____

Subject: Re: [PATCH] vt: Rework the console spawning variables.
Posted by ebiederm on Mon, 11 Sep 2006 05:01:58 GMT
View Forum Message <> Reply to Message

Oleg Nesterov <oleg@tv-sign.ru> writes:

> On 09/10, Eric W. Biederman wrote:
>>
>> Oleg Nesterov <oleg@tv-sign.ru> writes:
>>
>> > On 09/10, Eric W. Biederman wrote:
>> >>
>> >> Updating this old code is painful.
>> >
>> > No, no, we shouldn't change the old code, it is fine.
>> >
>> So what happens when:
>> cpu0: cpu1:
>> kill_pid(vt_pid,....) fn_SAK()->vc_reset()->put_pid(xchg(&vt_pid, NULL))
>>
>> Can't kill_pid dereference vt_pid after put_pid is called?
>
> Ah, I didn't consider that patch as 'old code', sorry :)

What I meant was that updating code that predates SMP support is painful.
When you said everything was ok. I was confused.

> I don't understand drivers/char/vt*, but surely put_pid(xchg()) can't work.
> Again, unless we have a lock to serialize access to ->vt_pid, but in that
> case we don't need xchg().

Ok. So we are in violent agreement then, my patch was wrong.

The xchg half works. For taking and putting a reference it is fine, you
just can't use that reference for anything safely.

So I need to come up with a new patch that gets it's locking correct,
in the fn_SAK case.

Eric