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

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