Subject: Re: [PATCH 1/2] pidns: Don't allow new pids after the namespace is dead.
Posted by Daniel Lezcano on Wed, 16 Feb 2011 23:21:12 GMT
View Forum Message <> Reply to Message

On 02/15/2011 07:30 PM, Oleg Nesterov wrote:

```
> On 02/15, Daniel Lezcano wrote:
>> In the case of unsharing or joining a pid namespace, it becomes
>> possible to attempt to allocate a pid after zap_pid_namespace has
>> killed everything in the namespace.  Close the hole for now by simply
>> not allowing any of those pid allocations to succeed.
> Daniel, please explain more. It seems, a long ago I knew the reason
> for this patch, but now I can't recall and can't understand this change.
```

The idea behind unsharing the pid namespace is the current pid is not
mapped in the newly created pid namespace and appears as the pid 0. When
it forks, the child process becomes the init pid of the new pid
namespace. When this pid namespace dies because the init pid exited, the
parent process (aka pid 0) can no longer fork because the pid namespace
is flagged dead. This is what does this patch.

The next patch allows a single process to spawn different processes in
different pid namespace. You can argue we can already do that with
clone(CLONE_NEWPID). That's true. But if we are able to unshare the pid
namespace, then the next patchset (which will come right after this one)
will allow to attach a process to a namespace and the implementation
will be very simple and consistent with attaching to any namespace.

```
>> --- a/include/linux/pid_namespace.h
>> +++ b/include/linux/pid_namespace.h
>> @@ -20,6 +20,7 @@ struct pid_namespace {
>>    struct kref kref;
>>    struct pidmap pidmap[PIDMAP_ENTRIES];
>>    int last_pid;
>> + atomic_t dead;
> Why atomic_t? It is used as a plain boolean.
>
> And I can't unde
```

I think Eric used an atomic because it is lockless with alloc_pid vs
zap_pid_ns_processes.

```
>> --- a/kernel/pid.c
>> +++ b/kernel/pid.c
>> @@ -282,6 +282,10 @@ struct pid *alloc_pid(struct pid_namespace *ns)
>>    struct pid_namespace *tmp;
>>    struct upid *upid;
>>
>> + pid = NULL;
```

>> + if (atomic_read(&ns->dead))
>> + goto out;
>> +
> So why this is needed?
>
> If we see ns->dead != 0 we are already killed by zap_pid_ns_processes()
> which sets ns->dead = 1.

The current process unshares the pid namespace.
When it forks, the child process is the pid 1. When this one exits, the
zap_pid_ns_processes is called and tag the pid namespace as dead. The
current process can no longer fork.

   -- Daniel