Subject: Re: [PATCH 2/2] pidns: Support unsharing the pid namespace.
Posted by Daniel Lezcano on Wed, 16 Feb 2011 23:47:37 GMT
View Forum Message <> Reply to Message

On 02/15/2011 08:01 PM, Oleg Nesterov wrote:
> On 02/15, Daniel Lezcano wrote:
>> - Pass both nsproxy->pid_ns and task_active_pid_ns to copy_pid_ns
>>   As they can now be different.
> But since they can be different we have to convert some users of
> current->nsproxy first? But that patch was dropped.
>
>> Unsharing of the pid namespace unlike unsharing of other namespaces
>> does not take effect immediately. Instead it affects the children
>> created with fork and clone.
> IOW, unshare(CLONE_NEWPID) implicitly affects the subsequent fork(),
> using the very subtle way.
>
> I have to admit, I can't say I like this very much. OK, if we need
> this, can't we just put something into, say, signal->flags so that
> copy_process can check and create the new namespace.
>
> Also. I remember, I already saw something like this and google found
> my questions. I didn't actually read the new version, perhaps my
> concerns were already answered...
>
> But what if the task T does unshare(CLONE_NEWPID) and then, say,
> pthread_create() ? Unless I missed something, the new thread won't
> be able to see T ?

Right. Is it really a problem ? I mean it is a weird use case where we
fall in a weird situation.
I suppose we can do the same weird combination with clone.
IMHO, the userspace is responsible of how it uses the syscalls. Until
the system is safe, everything is ok, no ?

> and, in this case the exiting sub-namespace init also kills its
> parent?

I don't think so because the zap_pid_ns_processes does not hit the
parent process when it browses the pidmap.

I tried the following program without problem:

```
#include <stdio.h>
#define _GNU_SOURCE
#include <sched.h>
#include <pthread.h>
```

```
void *routine(void *data)
{
	printf("pid %d!\n", getpid());
	return NULL;
}

int main(int argc, char *argv[])
{
	char **aux = &argv[1];
	pthread_t t;

	if (unshare(CLONE_NEWPID)) {
		perror("unshare");
		return -1;
	}

	if (pthread_create(&t, NULL, routine, NULL)) {
		perror("pthread_create");
		return -1;
	}

	if (pthread_join(t, NULL)) {
		perror("pthread_join");
		return -1;
	}

	printf("joined\n");

	return 0;
}
```

> OK, suppose it does fork() after unshare(), then another fork().
> In this case the second child lives in the same namespace with
> init created by the 1st fork, but it is not descendant ? This means
> in particular that if the new init exits, zap_pid_ns_processes()->
> do_wait() can't work.

Hmm, good question. IMO, we should prevent such case for now in the same way we added the flag 'dead', IOW adding a flag 'busy' for example.