Subject: Re: [RFC PATCH 0/5] Resend -v2 - Use procfs to change a syscall behavior
Posted by Alexey Dobriyan on Thu, 10 Jul 2008 01:39:15 GMT
View Forum Message <> Reply to Message

On Wed, Jul 09, 2008 at 05:43:04PM -0700, Eric W. Biederman wrote:
> Alexey Dobriyan <adobriyan@gmail.com> writes:
>
> > On Tue, Jul 08, 2008 at 01:24:22PM +0200, Nadia.Derbey@bull.net wrote:
> >> # echo "LONG1 XX" > /proc/self/task/<my_tid>/next_syscall_data
> >
> > Same stuff.
> >
> > There is struct task_struct::did_exec , what about it?
> >
> > Also, patches are about de-serializing, how serializing from userspace looks
> > like?
> > You freezed group of processes, then what?
> >
> > How, for example, dump all VMAs correctly?
> > [prepares counter-example]
>
> Alexey userspace vs a kernel space implementation is the wrong argument.
>
> It is clearly established that the current user space interfaces are
> insufficient to do the job. So we need to implement something in the kernel.
>
> Further I have heard of no one suggesting running a single kernel on multiple
> machines. Therefore there no one seems to be doing this entirely in the kernel
> and so we need a user space component.
>
> So the question should not be user space vs. kernel space but can we build clean
> interfaces for checkpoint/restart?

> What will those interfaces be?

In case of ->did_exec the only clean interface I see is:

```
 tsk->did_exec = !!tsk_img->did_exec;
```

It would be pretty silly to wrap this one line in a system call (two actually -- one in, one out), since you're going to restore some more fields of such variety anyway (like ->pdeath_signal).

Given the diversity of kernel internal data structures and all sorts of links between them, the only system call suitable is ioctl(2), not all this zoo of system calls proposed. They are all extendable and without rules, but ioctl(2) is also without rules.

This is all said in assumption that serializing kernel-internal data for checkpoint/restart to userspace is acceptable for mainline.
I don't think it is.