Subject: Re: [RFC PATCH 0/5] Resend - Use procfs to change a syscall behavior
Posted by serue on Tue, 08 Jul 2008 21:50:34 GMT
View Forum Message <> Reply to Message

Quoting Pavel Machek (pavel@ucw.cz):
> On Mon 2008-07-07 14:01:19, Serge E. Hallyn wrote:
> > Quoting Pavel Machek (pavel@ucw.cz):
> > > Hi!
> > >
> > > > This patchset is a part of an effort to change some syscalls behavior for
> > > > checkpoint restart.
> > > >
> > > > When restarting an object that has previously been checkpointed, its state
> > > > should be unchanged compared to the checkpointed image.
> > > > For example, a restarted process should have the same upid nr as the one it
> > > > used to have when being checkpointed; an ipc object should have the same id
> > > > as the one it had when the checkpoint occured.
> > > > Also, talking about system V ipcs, they should be restored with the same
> > > > state (e.g. in terms of pid of last operation).
> > > >
> > > > This means that several syscalls should not behave in a default mode when
> > > > they are called during a restart phase.
> > > >
> > > > One solution consists in defining a new syscall for each syscall that is
> > > > called during restart:
> > > > . sys_fork_with_id() would fork a process with a predefined id.
> > > > . sys_msgget_with_id() would create a msg queue with a predefined id
> > > > . sys_semget_with_id() would create a semaphore set with a predefined id
> > > > . etc,
> > > >
> > > > This solution requires defining a new syscall each time we need an existing
> > > > syscall to behave in a non-default way.
> > >
> > > Yes, and I believe that's better than...
> > >
> > > > An alternative to this solution consists in defining a new field in the
> > > > task structure (let's call it next_syscall_data) that, if set, would change
> > > > the behavior of next syscall to be called. The sys_fork_with_id() previously
> > > > cited can be replaced by
> > > > 1) set next_syscall_data to a target upid nr
> > > > 2) call fork().
> > >
> > > ...bloat task struct and
> > >
> > > > A new file is created in procfs: /proc/self/task/<my_tid>/next_syscall_data.
> > > > This makes it possible to avoid races between several threads belonging to
> > > > the same process.
> > >

> > > ...introducing this kind of uglyness.
> > >
> > > Actually, there were proposals for sys_indirect(), which is slightly
> > > less ugly, but IIRC we ended up with adding syscalls, too.
> >
> > Silly question...
> >
> > Oren, would you object to defining sys_fork_with_id(),
> > sys_msgget_with_id(), and sys_semget_with_id()?
> >
> > Eric, Pavel (Emelyanov), Dave, do you have preferences?
> >
> > For the cases Nadia has implemented here I'd be tempted to side with
> > Pavel Machek, but once we get to things like open() and socket(), (a)
> > the # new syscalls starts to jump, and (b) the per-syscall api starts to
> > seem a lot more cumbersome.
>
> You should not need to modify open/socket. You can already select fd
> by creatively using open/dup/close...

That's what we do right now in cryo. And if we end up patching up every API with separate syscalls, then we wouldn't create open_with_id(). But so long as the next_id were to exist, exploiting it in open is nigh on trivial and much nicer.

-serge

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers