Subject: Re: semantics for namespace naming
Posted by ebiederm on Thu, 14 Dec 2006 21:56:34 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Quoting Dave Hansen (haveblue@us.ibm.com):
>> On Thu, 2006-12-14 at 09:36 -0600, Serge E. Hallyn wrote:
>> > As I said, once a process is in a container, it never leaves that
>> > container. It only enters additional ones. That model fits everyone's
>> > needs, without needing some funky API.
>>
>> This makes logical sense to me. In practice this has the feel of
>> ptracing where the ptracer becomes a temporary parent of the tracee.
>>
>> The process entering a container temporarily becomes a member of that
>> container, but it doesn't completely _stop_ being a member of its
>> container. The real_parent of a process being ptraced may not be doing
>> all of the parental duties during a ptrace, but it doesn't _stop_ being
>> the real_parent.
>>
>> Maybe I'm stretching the analogy too far :)
>
> Maybe, or maybe you're showing me a kink in my reasoning. I was in
> fact thinking that entering a new container would be the one way
> to fully disengage from the old container. Meaning it would be best
> if it were forced to be done on a clone+exec. But even so, is that
> reasonable?

What I am doing today is using ptrace to force a process in
the container to fork. Then I can remote control that forked
process using ptrace to do whatever I need to do.

Because that model fundamentally keeps every process in it's own
container and never allows it to leave, nor does it allow things
from one container to cross into another container in an uncontrolled
fashion this feels to me like a very safe model.

We probably want to enhance ptrace so that it is easier to get
a remote process to execute a system call for us so without having
to jump through hoops, to find a syscall instruction etc, but I
think it is a model that makes a lot of sense.

The only nasty case I have is how do you handle a login daemon
outside of your container wanting to spawn new processes inside of your
container.

A- login daemon B - container init

+ C - login child ---> + D - process spawned by login child (child of container init)

If your login daemon (A) spawns a child process (C) that you want to place in a container you create process (D) a process of peculiar heritage, living entirely in the container. In normal situations the remote login does not terminate until D performs the desired work.

Since C has done all it needs to do ideally it would then exit. The problem is that if C exits the login daemon will normally think that the work is done and terminate the login session. Despite D holding open it's connection to the login session.

So in practice I have to keep C around doing the ptrace parent think until D exits, so I can forward the exit code back to A. Not bad but a little annoying.

Eric

_____