Subject: Re: [ckrm-tech] [PATCH 0/2] resource control file system - aka containers on top of nsproxy!

Posted by Srivatsa Vaddagiri on Mon, 12 Mar 2007 14:01:48 GMT

View Forum Message <> Reply to Message

On Fri, Mar 09, 2007 at 02:06:03PM -0800, Paul Jackson wrote:
> > if you create a 'resource container' to limit the
> > usage of a set of resources for the processes
> > belonging to this container, it would be kind of
> > defeating the purpose, if you'd allow the processes
> > to manipulate their limits, no?
>
> Wrong - this is not the only way.
>
> For instance in cpusets, -any- task in the system, regardless of what
> cpuset it is currently assigned to, might be able to manipulate -any-
> cpuset in the system.
>
> Yes -- some sufficient mechanism is required to keep tasks from
> escalating their resources or capabilities beyond an allowed point.
>
> But that mechanism might not be strictly based on position in some
> hierarchy.
>
> In the case of cpusets, it is based on the permissions on files in
> the cpuset file system (normally mounted at /dev/cpuset), versus
> the current priviledges and capabilities of the task.
>
> A root priviledged task in the smallest leaf node cpuset can manipulate
> every cpuset in the system. This is an ordinary and common occurrence.

This assumes that you can see the global vfs namespace right?

What if you are inside a container/vserver which restricts your vfs
namespace? i.e /dev/cpusets seen from one container is not same as what
is seen from another container ..Is that a unrealistic scenario? IMHO
not so. This in-fact lets vservers and containers to work with each
other. So:

```
 /dev/cpuset
  |- C1 <- Container A bound to this
  |  |- C11
  |  |- C12
  |
        |- C2 <- Container B bound to this
     |- C21
     |- C22
```

C1 and C2 are two exclusive cpusets and containers/vservers A and B are bound to C1/C2 respectively.

>From inside container/vserver A, if you were to look at /dev/cpuset, it will -appear- as if you are in the top cpuset (with just C11 and C12 child cpusets). It cannot modify C2 at all (since it has no visibility).

Similarly if you were to look at /dev/cpuset from inside B, it will list only C21/C22 with tasks in container B not being able to see C1 at all.

:)

--
Regards,
vatsa