Subject: Re: Re: [RFC][PATCH 0/2] user namespace [try #2]
Posted by serue on Thu, 07 Sep 2006 15:53:37 GMT
View Forum Message <> Reply to Message

Quoting Kirill Korotaev (dev@sw.ru):
> > Here's a stab at semantics for how to handle file access. Should be
> > pretty simple to implement, but i won't get a chance to implement this
> > week.
> >
> > At mount, by default the vfsmount is tagged with a uid_ns.
> > A new -o uid_ns=<pid> option instead tags the vfsmount with the uid_ns
> > belonging to pid <pid>. Since any process in a descendent pid
> > namespace should still have a valid pid in the ancestor
> > pidspaces, this should work fine.
> > At vfs_permission, if current->nsproxy->uid_ns != file->f_vfsmnt->uid_ns,
> > 1. If file is owned by root, then read permission is granted
> > 2. If file is owned by non-root, no permission is granted
> > (regardless of process uid)
> >
> > Does this sound reasonable?
> imho this in acceptable for OpenVZ as makes VE files to be inaccessiable from
> host. At least this is how I understand your idea...
> Am I correct?

Only if the host did the setup correctly. Either it could do

mount -o uid_ns=<pid> /dev/hdc1 /mnt/guest/root/5

right off the bat, or it could simply

mount -o uid_ns=<pid> --bind /mnt/guest/root/5 /mnt/guest/root/5

since after that, any access under /mnt/guest/root/5 would be looked up
with the vfsmount belonging to the guest's uid namespace.

> > I assume the list of other things we'll need to consider includes
> > signals between user namespaces
> > keystore
> > sys_setpriority and the like
> > I might argue that all of these should be sufficiently protected
> > by proper setup by userspace. Can you explain why that is not
> > the case?
> The same requirement (ability to send signals from host to VE)
> is also applicable to signals.

This property should be inherent to the use of a pid_ns. Let's say the
host is in pid_ns one, and creates a new pid_ns 2. pid_ns 2 has a

process known as (pid_ns 2, pid 22). There will be another 'struct pid' pointing to the same task_struct, calling it (pid_ns 1, pid 578).

So a process in pid_ns 1 can signal (pid_ns 2, pid 22) by sending a signal to pid 578.

A proces in pid_ns 2 has no reference to any process in pid_ns 1 (and not in pid_ns 2), therefore cannot signal those processes.

-serge