Subject: Re: [patch 2/8] allow unprivileged umount
Posted by Miklos Szeredi on Sun, 22 Apr 2007 07:32:34 GMT
View Forum Message <> Reply to Message

> > Does this mean, that containers will need this? Or that you don't
> > know yet?
>
> The uid namespace is something we have to handle carefully and we
> have not decided on the final design.
>
> What is clear is that all permission checks will need to become
> either (uid namspace, uid) tuple comparisons. Or struct user
> pointer comparisons. To see if we are talking about the same
> uid.
>
> So the eventual uid namespace combined with the possibility
> for rlimits if we use struct user *. See to make using a struct
> user a clear win.

OK, if we don't yet know, I'd rather leave this for later. It will be
trivial to change to user_struct if we want per-user rlimits.

> >> storing a user struct on each mount point seems sane, plus it allows
> >> per user mount rlimits which is major plus. Especially since we
> >> seem to be doing accounting only for user mounts a per user rlimit
> >> seems good.
> >
> > I'm not against per-user rlimits for mounts, but I'd rather do this
> > later...
>
> Then let's add a non-discriminate limit. Instead of a limit that
> applies only to root.

See reply to relevant patch.

> >> To get the user we should be user fs_uid as HPA suggested.
> >
> > fsuid is exclusively used for checking file permissions, which we
> > don't do here anymore. So while it could be argued, that mount() _is_
> > a filesystem operation, it is really a different sort of filesystem
> > operation than the rest.
> >
> > OTOH it wouldn't hurt to use fsuid instead of ruid...
>
> Yes. I may be confused but I'm pretty certain we want either
> the fsuid or the euid to be the mount owner. ruid just looks wrong.
> The fsuid is a special case of the effective uid. Which is who
> we should perform operations as. Unless I'm just confused.

Definitely not euid. Euid is the one which is effective, i.e. it will
basically always be zero for a privileged mount().

Ruid is the one which is returned by getuid(). If a user execs a
suid-root program, then ruid will be the id of the user, while euid
will be zero.

> >> Finally I'm pretty certain the capability we should care about in
> >> this context is CAP_SETUID. Instead of CAP_SYS_ADMIN.
> >>
> >> If we have CAP_SETUID we can become which ever user owns the mount,
> >> and the root user in a container needs this, so he can run login
> >> programs. So changing the appropriate super user checks from
> >> CAP_SYS_ADMIN to CAP_SETUID I think is the right thing todo.
> >
> > That's a flawed logic. If you want to mount as a specific user, and
> > you have CAP_SETUID, then just use set*uid() and then mount().
>
> Totally agreed for mount.
>
> > Changing the capability check for mount() would break the userspace
> > ABI.
>
> Sorry I apparently wasn't clear. CAP_SETUID should be the capability
> check for umount.

The argument applies to umount as well. For compatibility, we _need_
the CAP_SYS_ADMIN check. And if program has CAP_SETUID but not
CAP_SYS_ADMIN, it can just set the id to the mount owner before
calling umount.

Miklos

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