Subject: Re: [patch] unprivileged mounts update
Posted by serue on Wed, 25 Apr 2007 17:30:09 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Miklos Szeredi <miklos@szeredi.hu> writes:
>
> >> From: Miklos Szeredi <mszeredi@suse.cz>
> >>
> >> - refine adding "nosuid" and "nodev" flags for unprivileged mounts:
> >>   o add "nosuid", only if mounter doesn't have CAP_SETUID capability
> >>   o add "nodev", only if mounter doesn't have CAP_MKNOD capability
> >>
> >> - allow unprivileged forced unmount, but only for FS_SAFE filesystems
> >>
> >> - allow mounting over special files, but not symlinks
> >>
> >> - for mounting and umounting check "fsuid" instead of "ruid"
> >
> > Andrew, please skip this patch, for now.
> >
> > Serge found a problem with the fsuid approach: setfsuid(nonzero) will
> > remove filesystem related capabilities. So even if root is trying to
> > set the "user=UID" flag on a mount, access to the target (and in case
> > of bind, the source) is checked with user privileges.
>
> I do have a major problem with this patchset though. We still have
> the unnecessary concept of user mounts. That seems only needed now
> for the /proc/mounts output.
>
> All mounts should have an owner. Prior to the unprivileged mount work
> root owns all mounts.
>
> > Root should be able to set this flag on any mountpoint, _regardless_
> > of permissions.
>
> We don't need a flag, and thinking of it in the context of a flag
> is clearly the wrong thing. Yes if we have the proper capability we
> should be able to explicitly specify the owner of the mount
>
> > It is possible to restore filesystem capabilities after setting fsuid,
> > but the interfaces are rather horrible at all levels. mount(8) can
> > probably live with these, but I'm not sure that using "fsuid" over
> > "ruid" has enough advantages to force this.
> >
> > Why did we want to use fsuid, exactly?
>
> - Because ruid is completely the wrong thing we want mounts owned

> by whomever's permissions we are using to perform the mount.
>
>
> There are two basic cases.
> - Mounting a filesystem as who we are.
> This can use fsuid with no problems. If we are suid to root to perform
> the mount by default we want root to own the mount so that is correct.
>
> - Mounting a filesystem as another user.
> This is the tricky case rare case needed in setup. If we aren't
> jumping through to many hoops to make it work when using fsuid it
> sounds like the right thing here as well.
>
> How hard is it to set fsuid to a different value? I.e. What hoops
> does root have to jump through.
>
> Further when using fsuid we don't need an extra flag to mount.
>
> Plus things are a little more consistent with the rest of the
> linux/unix interface.
>
> Now I can see doing something like using a special flag and not using
> fsuid for the one case where we explicitly want to mount a filesystem
> as someone else. However if only user space has to special case this
> (as it does anyway) and we don't have to special case it in the
> kernel. So much the better.

Yes, what you describe (or my reading of it :) would simplify the
implementation, and solve the capability problem.

So in general, when you mount something, the mount is owned by you.

To mount something as you, either the mountpoint's mount is owned by
you, or you have some capability, maybe CAP_SYS_ADMIN.

So, before any non-root user can do a mount, root must mount an ancestor
mount in the name of that user. This would be a new mount flag, so

 mount -o user=some_user /share/$USER/home/$USER /share/$USER/home/$USER

as root. Mount does not change the fsuid, it simply passes the user=
flag into do_loopback(), which sets the mnt->user flag. And now, even
though i have /share as chmod 000, root didn't have to setfsuid so we
have the necessary caps.

(clearly, -o user requires CAP_SYS_ADMIN or something)

-serge

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