Subject: Re: Re: [patch 05/10] add "permit user mounts in new namespace" clone flag
Posted by Miklos Szeredi on Tue, 17 Apr 2007 17:44:09 GMT
View Forum Message <> Reply to Message

> I'm a bit lost about what is currently done and who advocates for what.
>
> It seems to me the MNT_ALLOWUSERMNT (or whatever :) flag should be
> propagated. In the /share rbind+chroot example, I assume the admin
> would start by doing
>
> mount --bind /share /share
> mount --make-slave /share
> mount --bind -o allow_user_mounts /share (or whatever)
> mount --make-shared /share
>
> then on login, pam does
>
> chroot /share/$USER
>
> or some sort of
>
> mount --bind /share /home/$USER/root
> chroot /home/$USER/root
>
> or whatever. In any case, the user cannot make user mounts except under
> /share, and any cloned namespaces will still allow user mounts.

I don't quite understand your method. This is how I think of it:

```
mount --make-rshared /
mkdir -p /mnt/ns/$USER
mount --rbind / /mnt/ns/$USER
mount --make-rslave /mnt/ns/$USER
mount --set-flags --recursive -oallowusermnt /mnt/ns/$USER
chroot /mnt/ns/$USER
su - $USER
```

I did actually try something equivalent (without the fancy mount
commands though), and it worked fine. The only "problem" is the
proliferation of mounts in /proc/mounts. There was a recently posted
patch in AppArmor, that at least hides unreachable mounts from
/proc/mounts, so the user wouldn't see all those. But it could still
be pretty confusing to the sysadmin.

So in that sense doing it the complicated way, by first cloning the
namespace, and then copying and sharing mounts individually which need
to be shared could relieve this somewhat.

Another point: user mounts under /proc and /sys shouldn't be allowed.
There are files there (at least in /proc) that are seemingly writable
by the user, but they are still not writable in the sense, that
"normal" files are.

Anyway, there are lots of userspace policy issues, but those don't
impact the kernel part.

As for the original question of propagating the "allowusermnt" flag, I
think it doesn't matter, as long as it's consistent and documented.

Propagating some mount flags and not propagating others is
inconsistent and confusing, so I wouldn't want that. Currently
remount doesn't propagate mount flags, that may be a bug, dunno.

Miklos

Subject: Re: Re: [patch 05/10] add "permit user mounts in new namespace" clone flag
Posted by serue on Tue, 17 Apr 2007 18:15:00 GMT
View Forum Message <> Reply to Message

Quoting Miklos Szeredi (miklos@szeredi.hu):
```
> > I'm a bit lost about what is currently done and who advocates for what.
> >
> > It seems to me the MNT_ALLOWUSERMNT (or whatever :) flag should be
> > propagated.  In the /share rbind+chroot example, I assume the admin
> > would start by doing
> >
> >  mount --bind /share /share
> >  mount --make-slave /share
> >  mount --bind -o allow_user_mounts /share (or whatever)
> >  mount --make-shared /share
> >
> > then on login, pam does
> >
> >  chroot /share/$USER
> >
> > or some sort of
> >
> >  mount --bind /share /home/$USER/root
> >  chroot /home/$USER/root
> >
> > or whatever.  In any case, the user cannot make user mounts except under
> > /share, and any cloned namespaces will still allow user mounts.
>
```

> I don't quite understand your method. This is how I think of it:
>
> mount --make-rshared /
> mkdir -p /mnt/ns/$USER
> mount --rbind / /mnt/ns/$USER
> mount --make-rslave /mnt/ns/$USER

This was my main point - that the tree in which users can mount will be
a slave of /, so that propagating the "are user mounts allowed" flag
among peers is safe and intuitive.

> mount --set-flags --recursive -oallowusermnt /mnt/ns/$USER
> chroot /mnt/ns/$USER
> su - $USER
>
> I did actually try something equivalent (without the fancy mount
> commands though), and it worked fine. The only "problem" is the
> proliferation of mounts in /proc/mounts. There was a recently posted
> patch in AppArmor, that at least hides unreachable mounts from
> /proc/mounts, so the user wouldn't see all those. But it could still
> be pretty confusing to the sysadmin.
>
> So in that sense doing it the complicated way, by first cloning the
> namespace, and then copying and sharing mounts individually which need
> to be shared could relieve this somewhat.

True. But the kernel functionality you provide enables both ways so no
problem in either case :)

> Another point: user mounts under /proc and /sys shouldn't be allowed.
> There are files there (at least in /proc) that are seemingly writable
> by the user, but they are still not writable in the sense, that
> "normal" files are.

Good point.

> Anyway, there are lots of userspace policy issues, but those don't
> impact the kernel part.

Though it might make sense to enforce /proc and /sys not allowing user
mounts under them in the kernel.

> As for the original question of propagating the "allowusermnt" flag, I
> think it doesn't matter, as long as it's consistent and documented.
>
> Propagating some mount flags and not propagating others is
> inconsistent and confusing, so I wouldn't want that. Currently
> remount doesn't propagate mount flags, that may be a bug, dunno.

Dave, any thoughts on safety of propagating the vfsmount read-only
flags?

-serge

Subject: Re: Re: [patch 05/10] add "permit user mounts in new namespace" clone flag
Posted by Miklos Szeredi on Tue, 17 Apr 2007 18:58:57 GMT
View Forum Message <> Reply to Message

> > mount --make-rshared /
> > mkdir -p /mnt/ns/$USER
> > mount --rbind / /mnt/ns/$USER
> > mount --make-rslave /mnt/ns/$USER
>
> This was my main point - that the tree in which users can mount will be
> a slave of /, so that propagating the "are user mounts allowed" flag
> among peers is safe and intuitive.

I think it's equally intuitive if flags are not propagated. After
all, I may want to set the mount flag _only_ on this particular mount,
and not anything else.

This is something I wouln't be sure about either way without
consulting the man.

> True. But the kernel functionality you provide enables both ways so no
> problem in either case :)
>
> > Another point: user mounts under /proc and /sys shouldn't be allowed.
> > There are files there (at least in /proc) that are seemingly writable
> > by the user, but they are still not writable in the sense, that
> > "normal" files are.
>
> Good point.
>
> > Anyway, there are lots of userspace policy issues, but those don't
> > impact the kernel part.
>
> Though it might make sense to enforce /proc and /sys not allowing user
> mounts under them in the kernel.

I think not, it would just be another policy decision having to be
made in the kernel on an fs by fs basis, and getting it wrong would be
much worse than getting it wrong in userspace.

Miklos

Subject: Re: Re: [patch 05/10] add "permit user mounts in new namespace" clone flag
Posted by Ram Pai on Tue, 17 Apr 2007 19:28:31 GMT
View Forum Message <> Reply to Message

On Tue, 2007-04-17 at 19:44 +0200, Miklos Szeredi wrote:
```
> > I'm a bit lost about what is currently done and who advocates for what.
> >
> > It seems to me the MNT_ALLOWUSERMNT (or whatever :) flag should be
> > propagated.  In the /share rbind+chroot example, I assume the admin
> > would start by doing
> >
> >  mount --bind /share /share
> >  mount --make-slave /share
> >  mount --bind -o allow_user_mounts /share (or whatever)
> >  mount --make-shared /share
> >
> > then on login, pam does
> >
> >  chroot /share/$USER
> >
> > or some sort of
> >
> >  mount --bind /share /home/$USER/root
> >  chroot /home/$USER/root
> >
> > or whatever.  In any case, the user cannot make user mounts except under
> > /share, and any cloned namespaces will still allow user mounts.
>
> I don't quite understand your method.  This is how I think of it:
>
> mount --make-rshared /
> mkdir -p /mnt/ns/$USER
> mount --rbind / /mnt/ns/$USER
> mount --make-rslave /mnt/ns/$USER
> mount --set-flags --recursive -oallowusermnt /mnt/ns/$USER
> chroot /mnt/ns/$USER
> su - $USER
>
> I did actually try something equivalent (without the fancy mount
> commands though), and it worked fine.  The only "problem" is the
> proliferation of mounts in /proc/mounts.  There was a recently posted
> patch in AppArmor, that at least hides unreachable mounts from
> /proc/mounts, so the user wouldn't see all those.  But it could still
> be pretty confusing to the sysadmin.
```

unbindable mounts were designed to overcome the proliferation problem.

Your steps should be something like this:

```
mount --make-rshared /
mkdir -p /mnt/ns
mount --bind /mnt/ns /mnt/ns
mount --make-unbindable /mnt/ns
mkdir -p /mnt/ns/$USER
mount --rbind / /mnt/ns/$USER
mount --make-rslave /mnt/ns/$USER
mount --set-flags --recursive -oallowusermnt /mnt/ns/$USER
chroot /mnt/ns/$USER
su - $USER
```

try this and your proliferation problem will disappear. :-)

>
> So in that sense doing it the complicated way, by first cloning the
> namespace, and then copying and sharing mounts individually which need
> to be shared could relieve this somewhat.

the unbindable mount will just provide you permanent relief.

>
> Another point: user mounts under /proc and /sys shouldn't be allowed.
> There are files there (at least in /proc) that are seemingly writable
> by the user, but they are still not writable in the sense, that
> "normal" files are.
>
> Anyway, there are lots of userspace policy issues, but those don't
> impact the kernel part.
>
> As for the original question of propagating the "allowusermnt" flag, I
> think it doesn't matter, as long as it's consistent and documented.
>
> Propagating some mount flags and not propagating others is
> inconsistent and confusing, so I wouldn't want that. Currently
> remount doesn't propagate mount flags, that may be a bug,

For consistency reason, one can propagate all the flags. But
propagating only those flags that interfere with shared-subtree
semantics should suffice.

wait...Dave's read-only bind mounts infact need the ability to
selectively make some mounts readonly. In such cases propagating
the read-only flag will just step on Dave's feature. Wont' it?

RP

>
> Miklos

Subject: Re: Re: [patch 05/10] add "permit user mounts in new namespace" clone flag
Posted by Miklos Szeredi on Tue, 17 Apr 2007 19:43:52 GMT
View Forum Message <> Reply to Message

```
> > > I'm a bit lost about what is currently done and who advocates for what.
> > >
> > > It seems to me the MNT_ALLOWUSERMNT (or whatever :) flag should be
> > > propagated.  In the /share rbind+chroot example, I assume the admin
> > > would start by doing
> > >
> > >  mount --bind /share /share
> > >  mount --make-slave /share
> > >  mount --bind -o allow_user_mounts /share (or whatever)
> > >  mount --make-shared /share
> > >
> > > then on login, pam does
> > >
> > >  chroot /share/$USER
> > >
> > > or some sort of
> > >
> > >  mount --bind /share /home/$USER/root
> > >  chroot /home/$USER/root
> > >
> > > or whatever.  In any case, the user cannot make user mounts except under
> > > /share, and any cloned namespaces will still allow user mounts.
> >
> > I don't quite understand your method.  This is how I think of it:
> >
> > mount --make-rshared /
> > mkdir -p /mnt/ns/$USER
> > mount --rbind / /mnt/ns/$USER
> > mount --make-rslave /mnt/ns/$USER
> > mount --set-flags --recursive -oallowusermnt /mnt/ns/$USER
> > chroot /mnt/ns/$USER
> > su - $USER
> >
> > I did actually try something equivalent (without the fancy mount
```

> > commands though), and it worked fine. The only "problem" is the
> > proliferation of mounts in /proc/mounts. There was a recently posted
> > patch in AppArmor, that at least hides unreachable mounts from
> > /proc/mounts, so the user wouldn't see all those. But it could still
> > be pretty confusing to the sysadmin.
>
> unbindable mounts were designed to overcome the proliferation problem.
>
> Your steps should be something like this:
>
> mount --make-rshared /
> mkdir -p /mnt/ns
> mount --bind /mnt/ns /mnt/ns
> mount --make-unbindable /mnt/ns
> mkdir -p /mnt/ns/$USER
> mount --rbind / /mnt/ns/$USER
> mount --make-rslave /mnt/ns/$USER
> mount --set-flags --recursive -oallowusermnt /mnt/ns/$USER
> chroot /mnt/ns/$USER
> su - $USER
>
> try this and your proliferation problem will disappear. :-)

Right, this is needed.

My problem wasn't actually this (which would only have hit, if I tried
with more than one user), just that the number of mounts in
/proc/mounts grows linearly with the number of users.

That can't be helped in such an easy way unfortunately.

> > Propagating some mount flags and not propagating others is
> > inconsistent and confusing, so I wouldn't want that. Currently
> > remount doesn't propagate mount flags, that may be a bug,
>
> For consistency reason, one can propagate all the flags. But
> propagating only those flags that interfere with shared-subtree
> semantics should suffice.

I still don't believe not propagating "allowusermnt" interferes with
mount propagation. In my posted patches the mount (including
propagations) is allowed based on the "allowusermnt" flag on the
parent of the requested mount. The flag is _not_ checked during
propagation.

Allowing this and other flags to NOT be propagated just makes it
possible to have a set of shared mounts with asymmetric properties,
which may actually be desirable.

Miklos

Subject: Re: Re: [patch 05/10] add "permit user mounts in new namespace" clone flag
Posted by Ram Pai on Tue, 17 Apr 2007 20:25:04 GMT
View Forum Message <> Reply to Message

On Tue, 2007-04-17 at 21:43 +0200, Miklos Szeredi wrote:
```
> > > > I'm a bit lost about what is currently done and who advocates for what.
> > > >
> > > > It seems to me the MNT_ALLOWUSERMNT (or whatever :) flag should be
> > > > propagated.  In the /share rbind+chroot example, I assume the admin
> > > > would start by doing
> > > >
> > > >  mount --bind /share /share
> > > >  mount --make-slave /share
> > > >  mount --bind -o allow_user_mounts /share (or whatever)
> > > >  mount --make-shared /share
> > > >
> > > > then on login, pam does
> > > >
> > > >  chroot /share/$USER
> > > >
> > > > or some sort of
> > > >
> > > >  mount --bind /share /home/$USER/root
> > > >  chroot /home/$USER/root
> > > >
> > > > or whatever.  In any case, the user cannot make user mounts except under
> > > > /share, and any cloned namespaces will still allow user mounts.
> > >
> > > I don't quite understand your method.  This is how I think of it:
> > >
> > > mount --make-rshared /
> > > mkdir -p /mnt/ns/$USER
> > > mount --rbind / /mnt/ns/$USER
> > > mount --make-rslave /mnt/ns/$USER
> > > mount --set-flags --recursive -oallowusermnt /mnt/ns/$USER
> > > chroot /mnt/ns/$USER
> > > su - $USER
> > >
> > > I did actually try something equivalent (without the fancy mount
> > > commands though), and it worked fine.  The only "problem" is the
> > > proliferation of mounts in /proc/mounts.  There was a recently posted
> > > patch in AppArmor, that at least hides unreachable mounts from
> > > /proc/mounts, so the user wouldn't see all those.  But it could still
```

> > > be pretty confusing to the sysadmin.
> >
> > unbindable mounts were designed to overcome the proliferation problem.
> >
> > Your steps should be something like this:
> >
> > mount --make-rshared /
> > mkdir -p /mnt/ns
> > mount --bind /mnt/ns /mnt/ns
> > mount --make-unbindable /mnt/ns
> > mkdir -p /mnt/ns/$USER
> > mount --rbind / /mnt/ns/$USER
> > mount --make-rslave /mnt/ns/$USER
> > mount --set-flags --recursive -oallowusermnt /mnt/ns/$USER
> > chroot /mnt/ns/$USER
> > su - $USER
> >
> > try this and your proliferation problem will disappear. :-)
>
> Right, this is needed.
>
> My problem wasn't actually this (which would only have hit, if I tried
> with more than one user), just that the number of mounts in
> /proc/mounts grows linearly with the number of users.
>
> That can't be helped in such an easy way unfortunately.
>
> > > Propagating some mount flags and not propagating others is
> > > inconsistent and confusing, so I wouldn't want that. Currently
> > > remount doesn't propagate mount flags, that may be a bug,
> >
> > For consistency reason, one can propagate all the flags. But
> > propagating only those flags that interfere with shared-subtree
> > semantics should suffice.
>
> I still don't believe not propagating "allowusermnt" interferes with
> mount propagation. In my posted patches the mount (including
> propagations) is allowed based on the "allowusermnt" flag on the
> parent of the requested mount. The flag is _not_ checked during
> propagation.
>
> Allowing this and other flags to NOT be propagated just makes it
> possible to have a set of shared mounts with asymmetric properties,
> which may actually be desirable.

The shared mount feature was designed to ensure that the mount remained
identical at all the locations. Now designing features
to make it un-identical but still naming it shared, will break its

original purpose. Slave mounts were designed to make it asymmetric.

Whatever feature that is desired to be exploited; can that be exploited
with the current set of semantics that we have? Is there a real need to
make the mounts asymmetric but at the same time name them as shared?
Maybe I dont understand what the desired application is?

RP

>
> Miklos

Subject: Re: Re: [patch 05/10] add "permit user mounts in new namespace" clone flag
Posted by Miklos Szeredi on Wed, 18 Apr 2007 09:19:46 GMT
View Forum Message <> Reply to Message

> > Allowing this and other flags to NOT be propagated just makes it
> > possible to have a set of shared mounts with asymmetric properties,
> > which may actually be desirable.
>
> The shared mount feature was designed to ensure that the mount remained
> identical at all the locations.

OK, so remount not propagating mount flags is a bug then?

> Now designing features to make it un-identical but still naming it
> shared, will break its original purpose. Slave mounts were designed
> to make it asymmetric.

What if I want to modify flags in a master mount, but not the slave
mount? Would I be screwed? For example: mount is read-only in both
master and slave. I want to mark it read-write in master but not in
slave. What do I do?

> Whatever feature that is desired to be exploited; can that be exploited
> with the current set of semantics that we have? Is there a real need to
> make the mounts asymmetric but at the same time name them as shared?
> Maybe I dont understand what the desired application is?

I do think this question of propagating mount flags is totally
independent of user mounts.

As it stands, currently remount doesn't propagate mount flags, and I
don't see any compelling reasons why it should.

The patchset introduces a new mount flag "allowusermnt", but I don't see any compelling reason to propagate this flag _either_.

Please say so if you do have such a reason. As I've explained, having
this flag set differently in parts of a propagation tree does not
interfere with or break propagation in any way.

Miklos

Subject: Re: Re: [patch 05/10] add "permit user mounts in new namespace" clone flag
Posted by Ram Pai on Wed, 18 Apr 2007 18:35:19 GMT
View Forum Message <> Reply to Message

On Wed, 2007-04-18 at 11:19 +0200, Miklos Szeredi wrote:
> > > Allowing this and other flags to NOT be propagated just makes it
> > > possible to have a set of shared mounts with asymmetric properties,
> > > which may actually be desirable.
> >
> > The shared mount feature was designed to ensure that the mount remained
> > identical at all the locations.
>
> OK, so remount not propagating mount flags is a bug then?

As I said earlier, are there any flags currently that if not propagated
can lead to conflicts with the shared subtree semantics? I am not aware
of any. If you did notice a case, than according to me its a bug.

But the new proposed 'allow unpriviledged mounts' flag; if not
propagated among peers (and slaves) of a shared mount can lead to
conflicts with shared subtree semantics. Since mount in one
shared-mount; when propagated to its peer fails to mount and hence lead
to un-identical peers.

>
> > Now designing features to make it un-identical but still naming it
> > shared, will break its original purpose. Slave mounts were designed
> > to make it asymmetric.
>
> What if I want to modify flags in a master mount, but not the slave
> mount? Would I be screwed? For example: mount is read-only in both
> master and slave. I want to mark it read-write in master but not in
> slave. What do I do?

Making mounts read-only or read-write -- will that effect mount
propagation in such a way that future mounts in any one of the

peers will not be able to propagate that mount to its peers or slaves?

I don't think it will. Hence its ok to selectively mark some mounts
read-only and some mounts read-write.

However with the introduction of unpriviledged mount semantics, there
can be cases where a user has priviledges to mount at one location but
not at a different location. if these two location happen to share
a peer-relationship than I see a case of interference of read-write
flag semantics with shared subtree semantics. And hence we will end up
propagating the read-write flag too or have to craft a different
semantics that stays consistent.

>
> > Whatever feature that is desired to be exploited; can that be exploited
> > with the current set of semantics that we have? Is there a real need to
> > make the mounts asymmetric but at the same time name them as shared?
> > Maybe I dont understand what the desired application is?
>
> I do think this question of propagating mount flags is totally
> independent of user mounts.
>
> As it stands, currently remount doesn't propagate mount flags, and I
> don't see any compelling reasons why it should.
>
> The patchset introduces a new mount flag "allowusermnt", but I don't
> see any compelling reason to propagate this flag _either_.
>
> Please say so if you do have such a reason. As I've explained, having
> this flag set differently in parts of a propagation tree does not
> interfere with or break propagation in any way.

As I said earlier, I see a case where two mounts that are peers of each
other can become un-identical if we dont propagate the "allowusermnt".

As a practical example.

/tmp and /mnt are peers of each other.
/tmp has its "allowusermnt" flag set, which has not been propagated
to /mnt.

now a normal-user mounts an ext2 file system under /tmp at /tmp/1

unfortunately the mount wont appear under /mnt/1

and this breaks the shared-subtree semantics which promises: whatever is mounted under /tmp will also be visible under /mnt

and in case if you allow the mount to appear under /mnt/1, you will
break unpriviledge mounts semantics which promises: a normal user will
not be able to mount at a location that does not allow user-mounts.

RP

>
> Miklos
>
