Subject: [patch] unprivileged mounts update
Posted by Miklos Szeredi on Wed, 25 Apr 2007 07:45:31 GMT
View Forum Message <> Reply to Message

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"

Thanks to everyone for the comments, with special thanks to Serge
Hallyn and Eric Biederman.

For testing the new functionality provided by this patchset a simple
tool similar in syntax to mount(8) is available from:

  http://www.kernel.org/pub/linux/kernel/people/mszeredi/mmount

Signed-off-by: Miklos Szeredi <mszeredi@suse.cz>

Index: linux/fs/namespace.c
===
--- linux.orig/fs/namespace.c 2007-04-22 17:48:18.000000000 +0200
+++ linux/fs/namespace.c 2007-04-22 18:19:51.000000000 +0200
@@ -252,10 +252,12 @@ static int reserve_user_mount(void)
 static void __set_mnt_user(struct vfsmount *mnt)
 {
  BUG_ON(mnt->mnt_flags & MNT_USER);
- mnt->mnt_uid = current->uid;
+ mnt->mnt_uid = current->fsuid;
  mnt->mnt_flags |= MNT_USER;
- if (!capable(CAP_SYS_ADMIN))
- mnt->mnt_flags |= MNT_NOSUID | MNT_NODEV;
+ if (!capable(CAP_SETUID))
+ mnt->mnt_flags |= MNT_NOSUID;
+ if (!capable(CAP_MKNOD))
+ mnt->mnt_flags |= MNT_NODEV;
 }

 static void set_mnt_user(struct vfsmount *mnt)
@@ -725,10 +727,10 @@ static bool permit_umount(struct vfsmoun

```
  if (!(mnt->mnt_flags & MNT_USER))
    return false;

- if (flags & MNT_FORCE)
+ if ((flags & MNT_FORCE) && !(mnt->mnt_sb->s_type->fs_flags & FS_SAFE))
    return false;

- return mnt->mnt_uid == current->uid;
+ return mnt->mnt_uid == current->fsuid;
 }

 /*
@@ -792,13 +794,13 @@ static bool permit_mount(struct nameidat
  if (type && !(type->fs_flags & FS_SAFE))
    return false;

- if (!S_ISDIR(inode->i_mode) && !S_ISREG(inode->i_mode))
+ if (S_ISLNK(inode->i_mode))
    return false;

  if (!(nd->mnt->mnt_flags & MNT_USER))
    return false;

- if (nd->mnt->mnt_uid != current->uid)
+ if (nd->mnt->mnt_uid != current->fsuid)
    return false;

  *flags |= MS_SETUSER;
```

_____

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by Miklos Szeredi on Wed, 25 Apr 2007 15:18:12 GMT
View Forum Message <> Reply to Message

> 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.

Root should be able to set this flag on any mountpoint, _regardless_
of permissions.

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?

Thanks,
Miklos

Subject: Re: [patch] unprivileged mounts update
Posted by hpa on Wed, 25 Apr 2007 16:55:06 GMT
View Forum Message <> Reply to Message

Miklos Szeredi wrote:
>
> 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.
>
> Root should be able to set this flag on any mountpoint, _regardless_
> of permissions.
>

Right, if you're using fsuid != 0, you're not running as root (fsuid is
the equivalent to euid for the filesystem.)

I fail to see how ruid should have *any* impact on mount(2). That seems

to be a design flaw.

-hpa

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by serue on Wed, 25 Apr 2007 17:20:12 GMT
View Forum Message <> Reply to Message

Quoting H. Peter Anvin (hpa@zytor.com):
> Miklos Szeredi wrote:
> >
> > 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.
> >
> > Root should be able to set this flag on any mountpoint, _regardless_
> > of permissions.
> >
>
> Right, if you're using fsuid != 0, you're not running as root

Sure, but what I'm not clear on is why, if I've done a
prctl(PR_SET_KEEPCAPS, 1) before the setfsuid, I still lose the
CAP_FS_MASK perms. I see the special case handling in
cap_task_post_setuid(). I'm sure there was a reason for it, but
this is a piece of the capability implementation I don't understand
right now.

I would send in a patch to make it honor current->keep_capabilities,
but I have a feeling there was a good reason not to do so in the
first place.

> (fsuid is
> the equivalent to euid for the filesystem.)

If it were really the equivalent then I could keep my capabilities :)
after changing it.

> I fail to see how ruid should have *any* impact on mount(2). That seems
> to be a design flaw.

May be, but just using fsuid at this point stops me from enabling user
mounts under /share if /share is chmod 000 (which it is).

thanks,
-serge

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by ebiederm on Wed, 25 Apr 2007 17:21:34 GMT
View Forum Message <> Reply to Message

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.

Eric

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by serue on Wed, 25 Apr 2007 17:30:09 GMT

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

Subject: Re: [patch] unprivileged mounts update
Posted by ebiederm on Wed, 25 Apr 2007 17:46:15 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Quoting H. Peter Anvin (hpa@zytor.com):
>> Miklos Szeredi wrote:
>> >
>> > 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.
>> >
>> > Root should be able to set this flag on any mountpoint, _regardless_
>> > of permissions.
>> >
>>
>> Right, if you're using fsuid != 0, you're not running as root
>
> Sure, but what I'm not clear on is why, if I've done a
> prctl(PR_SET_KEEPCAPS, 1) before the setfsuid, I still lose the
> CAP_FS_MASK perms. I see the special case handling in
> cap_task_post_setuid(). I'm sure there was a reason for it, but
> this is a piece of the capability implementation I don't understand
> right now.

So we drop CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_DAC_READ_SEARCH,
CAP_FOWNER, and CAP_FSETID

Since we are checking CAP_SETUID or CAP_SYS_ADMIN how is that
a problem?

Are there other permission checks that mount is doing that we
care about.

>> (fsuid is
>> the equivalent to euid for the filesystem.)
>
> If it were really the equivalent then I could keep my capabilities :)
> after changing it.

We drop all capabilities after we change the euid.

>> I fail to see how ruid should have *any* impact on mount(2). That seems
>> to be a design flaw.
>
> May be, but just using fsuid at this point stops me from enabling user
> mounts under /share if /share is chmod 000 (which it is).

I'm dense today. If we can't work out the details we can always use a flag.
But what is the problem with fsuid?

You are not trying to test this using a non-default security model are you?

Eric

Subject: Re: [patch] unprivileged mounts update
Posted by serge on Wed, 25 Apr 2007 17:56:09 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > Quoting H. Peter Anvin (hpa@zytor.com):
> >> Miklos Szeredi wrote:
> >> >
> >> > 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.
> >> >
> >> > Root should be able to set this flag on any mountpoint, _regardless_
> >> > of permissions.
> >> >
> >>
> >> Right, if you're using fsuid != 0, you're not running as root
> >
> > Sure, but what I'm not clear on is why, if I've done a
> > prctl(PR_SET_KEEPCAPS, 1) before the setfsuid, I still lose the
> > CAP_FS_MASK perms. I see the special case handling in
> > cap_task_post_setuid(). I'm sure there was a reason for it, but

> > this is a piece of the capability implementation I don't understand
> > right now.
>
> So we drop CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_DAC_READ_SEARCH,
> CAP_FOWNER, and CAP_FSETID
>
> Since we are checking CAP_SETUID or CAP_SYS_ADMIN how is that
> a problem?
>
> Are there other permission checks that mount is doing that we
> care about.

Not mount itself, but in looking up /share/fa/root/home/fa,
user fa doesn't have the rights to read /share, and by setting
fsuid to fa and dropping CAP_DAC_READ_SEARCH the mount action fails.

But the solution you outlined in your previous post would work around
this perfectly.

> >> (fsuid is
> >> the equivalent to euid for the filesystem.)
> >
> > If it were really the equivalent then I could keep my capabilities :)
> > after changing it.
>
> We drop all capabilities after we change the euid.

Not if we've done prctl(PR_SET_KEEPCAPS, 1)

> >> I fail to see how ruid should have *any* impact on mount(2). That seems
> >> to be a design flaw.
> >
> > May be, but just using fsuid at this point stops me from enabling user
> > mounts under /share if /share is chmod 000 (which it is).
>
> I'm dense today. If we can't work out the details we can always use a flag.
> But what is the problem with fsuid?

See above.

> You are not trying to test this using a non-default security model are you?

Nope, at the moment CONFIG_SECURITY=n so I'm running with capabilities
only.

thanks,
-serge

_____

Subject: Re: [patch] unprivileged mounts update
Posted by ebiederm on Wed, 25 Apr 2007 18:41:41 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serge@hallyn.com> writes:

> Quoting Eric W. Biederman (ebiederm@xmission.com):
>>
>> Are there other permission checks that mount is doing that we
>> care about.
>
> Not mount itself, but in looking up /share/fa/root/home/fa,
> user fa doesn't have the rights to read /share, and by setting
> fsuid to fa and dropping CAP_DAC_READ_SEARCH the mount action fails.

Got it.

I'm not certain this is actually a problem it may be a feature.
But it does fly in the face of the general principle of just
getting out of roots way so things can get done.

I think we can solve your basic problem by simply doing like:
chdir(/share); mount(.); To simply avoid the permission problem.

The practical question is how much do we care.

> But the solution you outlined in your previous post would work around
> this perfectly.

If we are not using usual permissions which user do we use current->uid?
Or do we pass that user someplace?

>> > If it were really the equivalent then I could keep my capabilities :)
>> > after changing it.
>>
>> We drop all capabilities after we change the euid.
>
> Not if we've done prctl(PR_SET_KEEPCAPS, 1)

Ah cap_clear doesn't do the obvious thing.

Eric

Subject: Re: [patch] unprivileged mounts update
Posted by serge on Wed, 25 Apr 2007 18:52:44 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serge@hallyn.com> writes:
>
> > Quoting Eric W. Biederman (ebiederm@xmission.com):
> >>
> >> Are there other permission checks that mount is doing that we
> >> care about.
> >
> > Not mount itself, but in looking up /share/fa/root/home/fa,
> > user fa doesn't have the rights to read /share, and by setting
> > fsuid to fa and dropping CAP_DAC_READ_SEARCH the mount action fails.
>
> Got it.
>
> I'm not certain this is actually a problem it may be a feature.
> But it does fly in the face of the general principle of just
> getting out of roots way so things can get done.
>
> I think we can solve your basic problem by simply doing like:
> chdir(/share); mount(.); To simply avoid the permission problem.
>
> The practical question is how much do we care.
>
> > But the solution you outlined in your previous post would work around
> > this perfectly.
>
> If we are not using usual permissions which user do we use current->uid?
> Or do we pass that user someplace?

Right, I figure if the normal action is to always do
mnt->user = current->fsuid, then for the special case we
pass a uid in someplace. Of course... do we not have a
place to do that? Would it be a no-no to use 'data' for
a non-fs-specific arg?

> >> > If it were really the equivalent then I could keep my capabilities :)
> >> > after changing it.
> >>
> >> We drop all capabilities after we change the euid.

> >
> > Not if we've done prctl(PR_SET_KEEPCAPS, 1)
>
> Ah cap_clear doesn't do the obvious thing.
>
> Eric

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by Miklos Szeredi on Wed, 25 Apr 2007 19:33:43 GMT
View Forum Message <> Reply to Message

> Right, I figure if the normal action is to always do
> mnt->user = current->fsuid, then for the special case we
> pass a uid in someplace. Of course... do we not have a
> place to do that? Would it be a no-no to use 'data' for
> a non-fs-specific arg?

I guess it would be OK for bind, but not for new- and remounts, where
'data' is already used.

Maybe it's best to stay with fsuid after all, and live with having to
restore capabilities. It's not so bad after all, this seems to do the
trick:

```
cap_t cap = cap_get_proc();
setfsuid(uid);
cap_set_proc(cap);
```

Unfortunately these functions are not in libc, but in a separate
"libcap" library. Ugh.

Miklos

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by akpm on Wed, 25 Apr 2007 19:33:46 GMT
View Forum Message <> Reply to Message

On Wed, 25 Apr 2007 17:18:12 +0200 Miklos Szeredi <miklos@szeredi.hu> wrote:

> > 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.

I'll be dropping all the unprivileged-mounts stuff - it looks like it
was a bit early, and that a new patch series against 2.6.27-rc1 or thereabouts
would be best.

Subject: Re: [patch] unprivileged mounts update
Posted by Miklos Szeredi on Wed, 25 Apr 2007 19:45:38 GMT
View Forum Message <> Reply to Message

> I'll be dropping all the unprivileged-mounts stuff - it looks like
> it was a bit early, and that a new patch series against 2.6.27-rc1

Yeah, I guess we can wait a few more years ;) ----------^^^

Miklos

Subject: Re: [patch] unprivileged mounts update
Posted by serue on Thu, 26 Apr 2007 14:57:33 GMT
View Forum Message <> Reply to Message

Quoting Miklos Szeredi (miklos@szeredi.hu):
> > Right, I figure if the normal action is to always do

> > mnt->user = current->fsuid, then for the special case we
> > pass a uid in someplace. Of course... do we not have a
> > place to do that? Would it be a no-no to use 'data' for
> > a non-fs-specific arg?
>
> I guess it would be OK for bind, but not for new- and remounts, where
> 'data' is already used.
>
> Maybe it's best to stay with fsuid after all, and live with having to
> restore capabilities. It's not so bad after all, this seems to do the
> trick:
>
> cap_t cap = cap_get_proc();
> setfsuid(uid);
> cap_set_proc(cap);
>
> Unfortunately these functions are not in libc, but in a separate
> "libcap" library. Ugh.

Ok, are you still planning to nix the MS_SETUSER flag, though, as Eric suggested? I think it's cleanest - always set the mnt->user field to current->fsuid, and require CAP_SYS_ADMIN if the mountpoint->mnt->user != current->fsuid.

-serge

Subject: Re: [patch] unprivileged mounts update
Posted by Miklos Szeredi on Thu, 26 Apr 2007 15:23:14 GMT
View Forum Message <> Reply to Message

> Quoting Miklos Szeredi (miklos@szeredi.hu):
> > > Right, I figure if the normal action is to always do
> > > mnt->user = current->fsuid, then for the special case we
> > > pass a uid in someplace. Of course... do we not have a
> > > place to do that? Would it be a no-no to use 'data' for
> > > a non-fs-specific arg?
> >
> > I guess it would be OK for bind, but not for new- and remounts, where
> > 'data' is already used.
> >
> > Maybe it's best to stay with fsuid after all, and live with having to
> > restore capabilities. It's not so bad after all, this seems to do the
> > trick:

> >
> > cap_t cap = cap_get_proc();
> > setfsuid(uid);
> > cap_set_proc(cap);
> >
> > Unfortunately these functions are not in libc, but in a separate
> > "libcap" library. Ugh.
>
> Ok, are you still planning to nix the MS_SETUSER flag, though, as
> Eric suggested? I think it's cleanest - always set the mnt->user
> field to current->fsuid, and require CAP_SYS_ADMIN if the
> mountpoint->mnt->user != current->fsuid.

It would be a nice cleanup, but I think it's unworkable for the
following reasons:

Up till now mount(2) and umount(2) always required CAP_SYS_ADMIN, and
we must make sure, that unless there's some explicit action by the
sysadmin, these rules are still enfoced.

For example, with just a check for mnt->mnt_uid == current->fsuid, a
fsuid=0 process could umount or submount all the "legacy" mounts even
without CAP_SYS_ADMIN.

This is a fundamental security problem, with getting rid of MS_SETUSER
and MNT_USER.

Another, rather unlikely situation is if an existing program sets
fsuid to non-zero before calling mount, hence unwantingly making that
mount owned by some user after these patches.

Also adding "user=0" to the options in /proc/mounts would be an
inteface breakage, that is probably harmless, but people wouldn't like
it. Special casing the zero uid for this case is more ugly IMO, than
the problem we are trying to solve.

If we didn't have existing systems to deal with, then of course I'd
agree with Eric's suggestion.

Miklos

Subject: Re: [patch] unprivileged mounts update

Quoting Miklos Szeredi (miklos@szeredi.hu):
> > Quoting Miklos Szeredi (miklos@szeredi.hu):
> > > > Right, I figure if the normal action is to always do
> > > > mnt->user = current->fsuid, then for the special case we
> > > > pass a uid in someplace. Of course... do we not have a
> > > > place to do that? Would it be a no-no to use 'data' for
> > > > a non-fs-specific arg?
> > >
> > > I guess it would be OK for bind, but not for new- and remounts, where
> > > 'data' is already used.
> > >
> > > Maybe it's best to stay with fsuid after all, and live with having to
> > > restore capabilities. It's not so bad after all, this seems to do the
> > > trick:
> > >
> > > cap_t cap = cap_get_proc();
> > > setfsuid(uid);
> > > cap_set_proc(cap);
> > >
> > > Unfortunately these functions are not in libc, but in a separate
> > > "libcap" library. Ugh.
> >
> > Ok, are you still planning to nix the MS_SETUSER flag, though, as
> > Eric suggested? I think it's cleanest - always set the mnt->user
> > field to current->fsuid, and require CAP_SYS_ADMIN if the
> > mountpoint->mnt->user != current->fsuid.
>
> It would be a nice cleanup, but I think it's unworkable for the
> following reasons:
>
> Up till now mount(2) and umount(2) always required CAP_SYS_ADMIN, and
> we must make sure, that unless there's some explicit action by the
> sysadmin, these rules are still enfoced.
>
> For example, with just a check for mnt->mnt_uid == current->fsuid, a
> fsuid=0 process could umount or submount all the "legacy" mounts even
> without CAP_SYS_ADMIN.
>
> This is a fundamental security problem, with getting rid of MS_SETUSER
> and MNT_USER.
>
> Another, rather unlikely situation is if an existing program sets
> fsuid to non-zero before calling mount, hence unwantingly making that
> mount owned by some user after these patches.
>
> Also adding "user=0" to the options in /proc/mounts would be an

> inteface breakage, that is probably harmless, but people wouldn't like
> it. Special casing the zero uid for this case is more ugly IMO, than
> the problem we are trying to solve.
>
> If we didn't have existing systems to deal with, then of course I'd
> agree with Eric's suggestion.
>
> Miklos

So then as far as you're concerned, the patches which were in -mm will
remain unchanged?

-serge

Subject: Re: [patch] unprivileged mounts update
Posted by Miklos Szeredi on Thu, 26 Apr 2007 16:29:40 GMT
View Forum Message <> Reply to Message

> So then as far as you're concerned, the patches which were in -mm will
> remain unchanged?

Basically yes. I've merged the update patch, which was not yet added
to -mm, did some cosmetic code changes, and updated the patch headers.

There's one open point, that I think we haven't really explored, and
that is the propagation semantics. I think you had the idea, that a
propagated mount should inherit ownership from the parent into which
it was propagated.

That sounds good if everyone agrees?

Miklos

Subject: Re: [patch] unprivileged mounts update
Posted by Jan Engelhardt on Thu, 26 Apr 2007 19:10:04 GMT
View Forum Message <> Reply to Message

On Apr 25 2007 11:21, Eric W. Biederman wrote:
>>
>> 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.

Think nfs. I access some nfs file as an unprivileged user. knfsd, by
nature, would run as euid=0, uid=0, but it needs fsuid=jengelh for
most permission logic to work as expected.

Jan
--

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by serue on Thu, 26 Apr 2007 19:42:25 GMT
View Forum Message <> Reply to Message

Quoting Miklos Szeredi (miklos@szeredi.hu):
> > So then as far as you're concerned, the patches which were in -mm will
> > remain unchanged?
>
> Basically yes. I've merged the update patch, which was not yet added
> to -mm, did some cosmetic code changes, and updated the patch headers.
>
> There's one open point, that I think we haven't really explored, and
> that is the propagation semantics. I think you had the idea, that a
> propagated mount should inherit ownership from the parent into which
> it was propagated.

Don't think that was me. I stayed out of those early discussions
because I wasn't comfortable guessing at the proper semantics yet.

But really, I, as admin, have to set up both propagation and user mounts
for a particular subtree, so why would I *not* want user mounts to be
propagated?

So, in my own situation, I have done

```
make / rshared
mount --bind /share /share
make /share unbindable
```

```
for u in $users; do
 mount --rbind / /share/$u/root
 make /share/$u/root rslave
 make /share/$u/root rshared
 mount --bind -o user=$u /share/$u/root/home/$u /share/$u/root/home/$u
done
```

All users get chrooted into /share/$USER/root, some also get their own namespace. Clearly if a user in a new namespace does

```
mount --bind -o user=me ~/somedir ~/otherdir
```

then logs out, and logs back in, I want the ~/otherdir in the new namespace (and the one in the 'init' namespace) to also be owned by 'me'.

> That sounds good if everyone agrees?

I've shown where I think propagating the mount owner is useful. Can you detail a scenario where doing so would be bad? Then we can work toward semantics that make sense...

-serge

Subject: Re: [patch] unprivileged mounts update
Posted by Miklos Szeredi on Thu, 26 Apr 2007 19:56:26 GMT
View Forum Message <> Reply to Message

> Quoting Miklos Szeredi (miklos@szeredi.hu):
> > > So then as far as you're concerned, the patches which were in -mm will
> > > remain unchanged?
> >
> > Basically yes. I've merged the update patch, which was not yet added
> > to -mm, did some cosmetic code changes, and updated the patch headers.
> >
> > There's one open point, that I think we haven't really explored, and
> > that is the propagation semantics. I think you had the idea, that a
> > propagated mount should inherit ownership from the parent into which
> > it was propagated.
>
> Don't think that was me. I stayed out of those early discussions
> because I wasn't comfortable guessing at the proper semantics yet.

Yes, sorry, it was Eric's suggestion.

> But really, I, as admin, have to set up both propagation and user mounts
> for a particular subtree, so why would I *not* want user mounts to be
> propagated?
>
> So, in my own situation, I have done
>
> make / rshared
> mount --bind /share /share
> make /share unbindable
> for u in $users; do
>   mount --rbind / /share/$u/root
>   make /share/$u/root rslave
>   make /share/$u/root rshared
>   mount --bind -o user=$u /share/$u/root/home/$u /share/$u/root/home/$u
> done
>
> All users get chrooted into /share/$USER/root, some also get their own
> namespace. Clearly if a user in a new namespace does
>
> mount --bind -o user=me ~/somedir ~/otherdir
>
> then logs out, and logs back in, I want the ~/otherdir in the new
> namespace (and the one in the 'init' namespace) to also be owned by
> 'me'.
>
> > That sounds good if everyone agrees?
>
> I've shown where I think propagating the mount owner is useful. Can you
> detail a scenario where doing so would be bad? Then we can work toward
> semantics that make sense...

But in your example, the "propagated mount inherits ownership from parent mount" would also work, since in all namespaces the owner of the parent would necessary be "me".

The "inherits parent" semantics would work better for example in the "all nosuid" namespace, where the user is free to modify it's mount namespace.

If for example propagation is set up from the initial namespace to this user's namespace and a new mount is added to the initial namespace, it would be nice if the propagated new mount would also be owned by the user (and be "nosuid" of course).

Does the above make sense? I'm not sure I've explained clearly enough.

Miklos

_____

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by Miklos Szeredi on Thu, 26 Apr 2007 20:27:32 GMT
View Forum Message <> Reply to Message

> On Apr 25 2007 11:21, Eric W. Biederman wrote:
> >>
> >> 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.
>
> Think nfs. I access some nfs file as an unprivileged user. knfsd, by
> nature, would run as euid=0, uid=0, but it needs fsuid=jengelh for
> most permission logic to work as expected.

I don't think knfsd will ever want to call mount(2).

But yeah, I've been convinced, that using fsuid is the right thing to
do.

Miklos

_____

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by serge on Fri, 27 Apr 2007 02:10:43 GMT
View Forum Message <> Reply to Message

Quoting Miklos Szeredi (miklos@szeredi.hu):
> > Quoting Miklos Szeredi (miklos@szeredi.hu):
> > > > So then as far as you're concerned, the patches which were in -mm will
> > > > remain unchanged?
> > >
> > > Basically yes. I've merged the update patch, which was not yet added
> > > to -mm, did some cosmetic code changes, and updated the patch headers.
> > >

> > > There's one open point, that I think we haven't really explored, and
> > > that is the propagation semantics. I think you had the idea, that a
> > > propagated mount should inherit ownership from the parent into which
> > > it was propagated.
> >
> > Don't think that was me. I stayed out of those early discussions
> > because I wasn't comfortable guessing at the proper semantics yet.
>
> Yes, sorry, it was Eric's suggestion.
>
> > But really, I, as admin, have to set up both propagation and user mounts
> > for a particular subtree, so why would I *not* want user mounts to be
> > propagated?
> >
> > So, in my own situation, I have done
> >
> > make / rshared
> > mount --bind /share /share
> > make /share unbindable
> > for u in $users; do
> >   mount --rbind / /share/$u/root
> >   make /share/$u/root rslave
> >   make /share/$u/root rshared
> >   mount --bind -o user=$u /share/$u/root/home/$u /share/$u/root/home/$u
> > done
> >
> > All users get chrooted into /share/$USER/root, some also get their own
> > namespace. Clearly if a user in a new namespace does
> >
> > mount --bind -o user=me ~/somedir ~/otherdir
> >
> > then logs out, and logs back in, I want the ~/otherdir in the new
> > namespace (and the one in the 'init' namespace) to also be owned by
> > 'me'.
> >
> > > That sounds good if everyone agrees?
> >
> > I've shown where I think propagating the mount owner is useful. Can you
> > detail a scenario where doing so would be bad? Then we can work toward
> > semantics that make sense...
>
> But in your example, the "propagated mount inherits ownership from
> parent mount" would also work, since in all namespaces the owner of
> the parent would necessary be "me".

true.

> The "inherits parent" semantics would work better for example in the

> "all nosuid" namespace, where the user is free to modify it's mount
> namespace.
>
> If for example propagation is set up from the initial namespace to
> this user's namespace and a new mount is added to the initial
> namespace, it would be nice if the propagated new mount would also be
> owned by the user (and be "nosuid" of course).

ok, so in the example i gave, this would be the admin in the
initial namespace mounting something under /home/$USER/, which
gets propagated to slave /share/$USER/root/home/$USER, where
we would want a different mount owner.

> Does the above make sense? I'm not sure I've explained clearly
> enough.

I think I see. Sounds like inherit from parent does the right thing
all around, at least in cases we've thought of so far.

thanks,
-serge

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by ebiederm on Fri, 27 Apr 2007 04:10:31 GMT
View Forum Message <> Reply to Message

Miklos Szeredi <miklos@szeredi.hu> writes:

>> On Apr 25 2007 11:21, Eric W. Biederman wrote:
>> >>
>> >> 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.
>>
>> Think nfs. I access some nfs file as an unprivileged user. knfsd, by
>> nature, would run as euid=0, uid=0, but it needs fsuid=jengelh for
>> most permission logic to work as expected.
>
> I don't think knfsd will ever want to call mount(2).
>
> But yeah, I've been convinced, that using fsuid is the right thing to
> do.

Actually knfsd does call mount when it crosses a mount point on the nfs
server it generates an equivalent mount point in linux. At least I think
that is the what it is doing. It is very similar to our mount propagation
path.

However as a special case I don't think the permission checking is likely
to bite us there. It is worth double checking once we have the other details
ironed out.

Eric

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

---

Subject: Re: [patch] unprivileged mounts update
Posted by Jan Engelhardt on Fri, 27 Apr 2007 07:01:44 GMT
View Forum Message <> Reply to Message

On Apr 26 2007 22:27, Miklos Szeredi wrote:
>> On Apr 25 2007 11:21, Eric W. Biederman wrote:
>> >>
>> >> 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.
>>
>> Think nfs. I access some nfs file as an unprivileged user. knfsd, by
>> nature, would run as euid=0, uid=0, but it needs fsuid=jengelh for
>> most permission logic to work as expected.
>
>I don't think knfsd will ever want to call mount(2).

I was actually out at something different...

```
        /* Make sure a caller can chown. */
        if ((ia_valid & ATTR_UID) &&
            (current->fsuid != inode->i_uid ||
             attr->ia_uid != inode->i_uid) && !capable(CAP_CHOWN))
                goto error;
```

for example. Using current->[e]uid would not make sense here.

>But yeah, I've been convinced, that using fsuid is the right thing to
>do.

Jan
--

_____

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