Subject: Re: [RFC][PATCH 0/2] user namespace [try #2]
Posted by ebiederm on Thu, 31 Aug 2006 17:02:35 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> Thanks :)
>
>> My gut feel is that this is terribly incomplete, and doesn't
>> come with enough description to tell me why it could possibly be complete.
>>
>> I don't think this addresses any of my primary objections from last
>> round.
>>
>> This doesn't change the kernel to make uid comparisons as (uid_ns, uid)
>> tuples or explain why that isn't necessary. It doesn't touch keys.
>> it doesn't explain why we are not introducing possibly subtle security
> problems.
>>
>> Cedric sorry for not saying so earlier, but I thought that the incompleteness
>> was obvious.
>
> (ignoring keys and other uid actions for now)
>
> 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?

Possibly, special casing the owner like that is odd.

I would cap the permission granted with at most the other permissions.
The permissions you have described would currently give me the ability
to read /etc/shadow.

Heck I would be happy initially to say
if current->nsproxy->uid_ns != file->f_vfsmnt->uid_ns, then you get no

permissions at all. As a first pass.

> 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 keystore is simple to explain. It is global ids. So we need
a separate set of global ids, so one container does not conflict
with another.

Signals between user namespaces are also simple to explain. If I can
see your process in my ps listing, I don't have CAP_KILL, and I'm not
you. I shouldn't be able to terminate your processes. See check_kill_permission.

At this point I don't see how proper namespace can protect you in either
of the above two cases.

As long as it requires root privileges to create a new user namespace
I do agree (under the principle of allowing people to shoot themselves
in the foot) that we can require the user to setup properly.

The only other rule we need to look towards for a minimal
implementation is that you can always add more capability but
you can't take something away once someone has come to depend on it.
Just look at sys_sysctl. It only has one user and we can't remove it.

So as long as what the user namespace allows is initially very
draconian. (When in doubt say no). We can probably start with
something fairly simple.

Eric

_____