Subject: Re: design of user namespaces
Posted by serue on Mon, 30 Jun 2008 21:13:25 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > There are. But one key point is that the namespace ids are not
> > cryptographic keys. They don't need to be globally unique, or even 100%
> > unique on one system (though that gets too subtle).
> >
> > I was wanting to keep them tiny - or at least variably sized - so they
> > could be stored with each inode.
>
> Sure. I think they make a lot of sense. idmapd uses domain names
> for this purpose. At the moment I just don't think the are necessary.
> Like veth isn't necessary for network namespaces. Ultimately we
> will use these identifiers all of the time but it doesn't mean
> the generic code has to deal with them.
>
>
> >> In particular: Is this user allowed to use this ID?
> >
> > One way to address that is actually by having a system-wide tool with
> > CAP_SET_USERNS=fp which enforces a userid-to-unsid mapping. The host
> > sysadmin creates a table, say, 500:0 may use unsid 10-15. 500:0 (let's
> > call him hallyn) doesn't have CAP_SET_USERNS permissions himself, but
> > can run /bin/set_unsid, which runs with CAP_SET_USERNS and ensures that
> > hallyn uses only unsids 10-15.
> >
> > It's not ideal. I'd rather have some sort of fancy collision-proof
> > global persistant id system, but again I think it's important that if
> > 500:0 creates userns 1 and 400:1 creates userns 2, and 0:2 creates a
> > file, that the file be persistantly marked as belonging to
> > (500:0,400:1,0:2), distinct from another file created by
> > (500:0,400:1,1000:2). Which means these things have to be stored
> > per-inode, meaning they can't be too large.
>
> So my suggestion was something like this:
> mount -o nativemount,uidns=1 /
>
> Then the filesystem performs magic to ask if the owner of user namespace
> is allowed to use uidns 1. That magic would consult a config file like:
>
> [domains]
> local1.mydomain 1
> local2.mydomain 2
> local3.mydomain 3

>
> [users]
> bob local1.mydomain
> bob local3.mydomain
> nancy local2.mydomain
>
> Or something like that. Reporting which users are allowed to use
> which userid namespaces, and the mapping of those userid namespaces
> to something compact for storing in the filesystem.
>
> The magic could be an upcall to userspace.
> The magic could be loading the configuration file at mount time.
> The magic could be storing the config file in the filesystem
> and having special commands to access it like quotas.
>
> The very important points are that it is a remount of an existing mount
> so that we don't have to worry about corrupted filesystem attacks, and
> that authentication is performed at mount time.

Conceptually that (making corrupted fs attacks a non-issue) is
wonderful. Practically, I may be missing something: When you say
remount, it seems you must either mean a bind mount or a remount. If
remount, then that will want to change superblock flags. If the
child userns(+child mntns) does a real remount, then that will change
the flags for the parent ns as well, right?

If instead we do a bind mount we don't have that problem, but then the
fs can't be the one doing the user namespace work.

I'm probably missing something...

> I just think that once we get to the point of specifying the parameters at
> mount time. There is no need for generic kernel configuration of a
> uidns name.

thanks,
-serge