Subject: Re: [PATCH 1/6] user namespace : add the framework
Posted by serue on Wed, 18 Jul 2007 14:21:51 GMT
View Forum Message <> Reply to Message

Quoting Herbert Poetzl (herbert@13thfloor.at):
> On Mon, Jul 16, 2007 at 10:08:00AM -0500, Serge E. Hallyn wrote:
> > Quoting Kirill Korotaev (dev@sw.ru):
> > > Serge E. Hallyn wrote:
> > > > Quoting Andrew Morton (akpm@linux-foundation.org):
> > > >
> > > >>On Mon, 4 Jun 2007 14:40:24 -0500 "Serge E. Hallyn" <serue@us.ibm.com> wrote:
> > > >>
> > > >>
> > > >>>Add the user namespace struct and framework
> > > >>>
> > > >>>Basically, it will allow a process to unshare its user_struct
> > > >>>table, resetting at the same time its own user_struct and all the
> > > >>>associated accounting.
> > > >>>
> > > >>>A new root user (uid == 0) is added to the user namespace upon
> > > >>>creation. Such root users have full privileges and it seems
> > > >>>that theses privileges should be controlled through some means
> > > >>>(process capabilities ?)
> > > >>
> > > >>The whole magical-uid-0-user thing in this patch seem just wrong
> > > >>to me.
> > > >>
> > > >>I'll merge it anyway, mainly because I want to merge _something_
> > > >>(why oh why do the git-tree guys leave everything to the last
> > > >>minute?) but it strikes me that there's something fundamentally
> > > >>wrong whenever the kernel starts "knowing" about the significance
> > > >>of UIDs in this fashion.
> > > >
> > > >
> > > > $(&(%
> > > >
> > > > I thought I disagreed, but now I'm pretty sure I completely agree.
> > > >
> > > > 'root_user' exists in the kernel right now, but the root_user
> > > > checks which exist (in fork.c and sys.c) shouldn't actually be
> > > > applied for root in a container, since the container may not be
> > > > trusted.
> > >
> > > This rlimit check doesn't help *untrusted* containers, so your logic
> > > is wrong here. Instead, it allows root of the container to operate
> > > in any situation.
> >
> > And I'm not sure that should be the case.

> >
> > In my view, root of a container is equivalent to a normal user on the
> > host system, just like root in a qemu process.
> >
> > > E.g. consider root user hit the limit. After that you won't be able
> > > to login/ssh to fix anything.
> >
> > That's fine in the container.
> >
> > > NOTE: container root can have no CAP_SYS_RESOURCE and CAP_SYS_ADMIN
> > > as it is in OpenVZ.
>
> > And eventually we'll want that to be the default in upstream containers.
> > But it's not the case upstream right now. Before we can do that, we
> > need an answer to per-container capabilities.
> >
> > Do you (either you specifically, or anyone at openvz) have plans to
> > address the per-container capabilities problem? Herbert? Eric?
>
> it is already addressed in Linux-VServer and OpenVZ
> Linux-VServer adds a so called 'capabilitiy mask',
> which is applied to the 'normal' capability system,
> thus a guest cannot utilize/exercise capabilities
> not included in that mask (which makes the guest
> root 'secure')

Are you special-casing some capabilities? For instance, cap_sys_admin in a container obviously shouldn't be fully granted, but some of it's abilities (setting hostname, setting enc key on loopback, etc) should be allowed.

Anyway, there unfortunately are several problems with a plain capability mask now.

First, there is the fact that we don't have just 'the host' and 'virtual hosts'. We have a set of namespaces, with usually no notion of one master namespace. So do we just mask out the masked capabilities as soon as any process does an unshare with any of the key flags? Or do we have a separate way to 'unshare capabilities'?

We could make cap-bset a per-process thing, where any process can take flags out (but not add them back in).

We could make a cap-bset container, where when creating a new container, we can take capabilities out of cap-bset.

Back to shortcomings,

Second, we wanted to implement user equivalence across namespaces. And to do that for root likely requires cross-namespace capabilities.

For instance, let's say I, user hallyn, unshare all my namespaces to create a virtual host vh.

Maybe I want user serge in vh to be equivalent to user hallyn in the real host. So I want to give it a key (host-uidns, hallyn). Now it can read all my files.

Maybe I want user hallyn in the host to have acess to all files in the vh, whether owned by root or any user. So I want to give user hallyn in the host uidns to have a key (vh-uidns, root).

That second example is nice and simple, but of course we prefer to think about capabilities. So I might just want to be able to give user hallyn in the host uid namespace a key (vh-uidns, CAP_DAC_OVERRIDE|CAP_DAC_READ_SEARCH|CAP_PTRACE).

Maybe all of this can be addressed with a posix capability container, a custom ns_capability LSM, and a uidns keyring.

thanks,
-serge

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