Subject: Re: [PATCH 2/9] security: Make capabilities relative to the user namespace.
Posted by serge on Wed, 23 Feb 2011 13:43:52 GMT
View Forum Message <> Reply to Message

Quoting David Howells (dhowells@redhat.com):
> David Howells <dhowells@redhat.com> wrote:
>
> > > int (*capable) (struct task_struct *tsk, const struct cred *cred,
> > > - int cap, int audit);
> > > + struct user_namespace *ns, int cap, int audit);
> >
> > Hmmm... A chunk of the contents of the cred struct are user-namespaced.
> > Could you add the user_namespace pointer to the cred struct and thus avoid
> > passing it as an argument to other things.
>
> Ah, no... Ignore that, I think I see that you do need it.

Thanks for reviewing, David.

> > +int cap_capable(struct task_struct *tsk, const struct cred *cred,
> > + struct user_namespace *targ_ns, int cap, int audit)
> > {
> > - return cap_raised(cred->cap_effective, cap) ? 0 : -EPERM;
> > + for (;;) {
> > + /* The creator of the user namespace has all caps. */
> > + if (targ_ns != &init_user_ns && targ_ns->creator == cred->user)
> > +   return 0;
>
> Why is that last comment so? Why should the creating namespace sport all
> possible capabilities? Do you have to have all capabilities available to you
> to be permitted create a new user namespace?

It's not the creating namespace, but the creating user, which has all caps.

So for instance, if uid 500 in init_user_ns creates a namespace, then:
  a. uid 500 in init_user_ns has all caps to the child user_ns, so it can
     kill the tasks in the userns, clean up, etc.
  b. uid X in any other child user_ns has no caps to the child user_ns.
  c. root in init_user_ns has whatever capabilities are in his pE to the
     child user_ns. Again, this is so that the admin in any user_ns can
     clean up any messes made by users in his user_ns.

One of the goals of the user namespaces it to make it safe to allow
unprivileged users to create child user namespaces in which they have
targeted privilege. Anything which happens in that child user namespace
should be:

a. constrained to resources which the user can control anyway
b. able to be cleaned up by the user
c. (especially) able to be cleaned up by the privileged user in the
parent user_ns.

> Also, would it be worth having a separate cap_ns_capable()? Wouldn't most
> calls to cap_capable() only be checking the caps granted in the current user
> namespace?

Hm. There is nsown_capable() which is targeted to current_userns(), but
that still needs to enable the caps for the privileged ancestors as
described above.

thanks,
-serge

_____