Subject: Re: [PATCH 7/8] user ns: handle file sigio
Posted by serue on Wed, 24 Jan 2007 18:58:45 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > From: Serge E. Hallyn <serue@us.ibm.com>
> > Subject: [PATCH 7/8] user ns: handle file sigio
> >
> > A process in one user namespace could set a fowner and sigio on a file in a
> > shared vfsmount, ending up killing a task in another user namespace.
> >
> > Prevent this by adding a user namespace pointer to the fown_struct, and
> > enforcing that a process causing a signal to be sent be in the same
> > user namespace as the file owner.
> >
>
>
> > @@ -455,6 +460,9 @@ static const long band_table[NSIGPOLL] =
> > static inline int sigio_perm(struct task_struct *p,
> >   struct fown_struct *fown, int sig)
> > {
> > + if (fown->user_ns != init_task.nsproxy->user_ns &&
> > +   fown->user_ns != p->nsproxy->user_ns)
> > + return 0;
>
> Why is the initial user namespace being treated specially here?

Because we haven't yet agreed upon any other security model. For now,
although I know you really dislike it, the fact is that "the initial
namespace has special privileges" is our basic security model.

If you want to have a discussion about an appropriate security model,
or an infrastructure to support multiple models, I think this would be a
good time, given that several namespaces are out there. And networkns,
making its way up, also has concerns.

Three basic approaches I could see being simple to both implement and
understand/use are

1. add a set of capabilities concerning cross-ns operations, not
reassignable once they are removed. Simple to understand, very
limited.

2. maintain that any cross-ns operation is allowed if and only if
the target ns is a child of the subject ns.

3. cross-ns operations are not permitted. The only way to achieve
them is using a (as-yet unimplemented, but i'm working on it) namespace
enter feature to execute code in a child namespace.

> Especially when you start considering nested containers special treatment
> like this is semantically a real problem, to maintain.

Yup.

> If we need to I can see doing something special if the process setting
> fown has CAP_KILL

Obviously CAP_KILL is insufficient :) I assume you mean a new
CAP_XNS_CAP_KILL?

> and bypassing the security checks that way, but
> hard coding rules like that when it doesn't appear we have any
> experience to indicate we need the extra functionality looks
> premature.

Ok, in this case actually I suspect you're right and we can just ditch
the exception. But in general the security discussion is one we should
still have.

> >   return (((fown->euid == 0) ||
> >   (fown->euid == p->suid) || (fown->euid == p->uid) ||
> >   (fown->uid == p->suid) || (fown->uid == p->uid)) &&
>
> Eric

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers