Subject: Re: [PATCH 1/1] capabilities: introduce per-process capability bounding set (v8)
Posted by Andrew Morgan on Thu, 22 Nov 2007 07:10:13 GMT
View Forum Message <> Reply to Message

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Serge E. Hallyn wrote:
> I worry that what you have is just a *touch* too busy so whoever adds
> capability #32 might forget to update CAP_NUM_CAPS, but it looks like
>
> #define CAP_LAST_CAP CAP_SETFCAP
>
> #define cap_valid(x) ((x) >= 0 && (x) <= CAP_LAST_CAP)
>
> should also be ok for libcap.

FWIW libcap computes the upper limit itself in the process of consuming
all that sed'ed stuff. You do need it for the kernel, and this seems
like a fine mechanism.

>>> +long cap_prctl_drop(unsigned long cap)
>>> +{
>>> + if (!capable(CAP_SETPCAP))
>>> + return -EPERM;
>>> + if (!cap_valid(cap))
>>> + return -EINVAL;
>>> + cap_lower(current->cap_bset, cap);
>> I think the following lines are overkill. Basically, the next exec()
>> will perform the pP/pE clipping, and cap_bset should only interact with
>> fP (and not fI).
>>
>> We already have a mechanism to manipulate pI, which in turn gates fI.
>> And this same mechanism (libcap) can clip pE, pP if it is needed pre-exec().
>>
>> So, if you want to drop a capability irrevocably, you drop it in bset,
>> and separately in pI. The current process may continue to have the
>> capability, but post-exec the working process tree has lost it. For
>> things like login, this is desirable.
>
> Ok...
>
> I think this makes sense. It seems pretty subtle and complicated, and
> therefore I'm a little worried that it will be fragile against future
> code changes. Someone will think it's a good idea to slightly change
> the capset() semantics and only a year later will we realize that the
> bounding set is no longer working...

We'll have to be diligent then :-) In truth, the whole model is not
entirely unsubtle.

> So this will all have to be very well documented (and tested).
>
> (Actually I notice that capabilities(7) manpage isn't in the libcap
> sources. So an update to that is probably long overdue...)

I don't believe it ever was. So far as I can tell this file has had its
own life as part of the 'manpages' package.

>> This also makes it possible for you to allow pI to have a capability
>> otherwise banned in cap_bset which is useful with limited role accounts.
>
> Yeah... so the way you'd see this happening, I assume, is that
>
> 1. login would keep some capset in pI for user hallyn,

The pam_cap module in the libcap2 tree already does the pI part of this
via libcap (and I intend adding this prctl/cap_bset support in libcap
and that module too).

> 2. so if /bin/foo had some nonempty fI, hallyn could run
>   /bin/foo with cap_intersect(pI|fI)?

Yes. The inheritable set is precisely for supporting role-account things
like this, but unlike the superuser concept (any app can be run with
privilege), the application needs to be prepared to wield them - via its
fI bits - before the capabilities are available.

> So now the bounding set would place a restriction on what /bin/login in
> some container could leave in hallyn's pI.

To be clear, I'm saying that cap_bset will limit what a process can
'add' to the pre-existing pI set, and not what can be 'left in' there.
That is, if pI contains something not present in cap_bset, it will
survive unless some process drops it.

For completeness, without this new check:

>> You might want to replace the above three lines with a restriction
>> elsewhere on what CAP_SETPCAP can newly set in
>> commoncap.c:cap_capset_check().

it was possible to subvert the bounding set (in your container, but more
generally in any process tree) as follows:

capbound drop cap_net_raw
noraw> cp /bin/ping evil-ping
noraw> cc evil-shell.c -o evil-shell

<since you've not dropped cap_setfcap, the following will work>

noraw> sudo /sbin/setcap cap_setpcap=pe evil-shell
noraw> sudo /sbin/setcap cap_net_raw=ie evil-ping
noraw> ./evil-shell

<evil-shell uses CAP_SETPCAP to raise pI=CAP_NET_RAW>

evil> ./evil-ping another-hack-completed.evil.com

Cheers

Andrew
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.6 (GNU/Linux)

iD8DBQFHRStSQheEq9QabfIRAqo8AJ9wr+h0bTHgp7iT5fI2lYmuKKa75wCffMsT
Q7y/JN4NuImodhYvm5mvGwY=
=sTdY
-----END PGP SIGNATURE-----