Subject: Re: [PATCH 1/1] capabilities: introduce per-process capability bounding set (v8)
Posted by Andrew Morgan on Tue, 20 Nov 2007 03:40:23 GMT
View Forum Message <> Reply to Message

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

Serge E. Hallyn wrote:
> Andrew, this version follows all of your suggestions. Definately nicer
> userspace interface. thanks
[...]
>
> /* Allow ioperm/iopl access */
> @@ -314,6 +314,10 @@ typedef struct kernel_cap_struct {
>
> #define CAP_SETFCAP   31
>
> +#define CAP_NUM_CAPS   32
> +
> +#define cap_valid(x) ((x) >= 0 && (x) < CAP_NUM_CAPS)
> +

Could you change the name of CAP_NUM_CAPS? There is some libcap building
code that does the following to automatically build the "cap_*" names
for libcap, and this new define above messes that up! :-(

sed -ne '/^#define[\t]CAP[_A-Z]\+[\t]\+[0-9]\+/{s/^#define \([^
\t]*\)[\t]*\([^ \t]*\)/ \{ \2, \"\1\"
\},/;y/ABCDEFGHIJKLMNOPQRSTUVWXYZ/abcdefghijklmnopqrstuvwxyz/;p;}' <
$(KERNEL_HEADERS)/linux/capability.h | fgrep -v 0x > cap_names.sed

Something like:

  #define CAP_NUM_CAPS (CAP_SETFCAP+1)

will save me some hassle. :-)

[...]

> /*
>   * Bit location of each capability (used by user-space library and kernel)
>   */
> @@ -350,6 +354,17 @@ typedef struct kernel_cap_struct {
>
> #define CAP_INIT_INH_SET   CAP_EMPTY_SET
>

Its kind of a pity to put a kernel config ifdef in a header file. Could
you put the ifdef code in the c-files that uses these definitions?

> +#ifdef CONFIG_SECURITY_FILE_CAPABILITIES

In my experience when headers define things differently based on
configuration #defines, other users of header files (apps, kernel
modules etc.), never quite know what the current define is. If we can
avoid conditional code like this in this header file, I'd be happier.

> +#ifdef CONFIG_SECURITY_FILE_CAPABILITIES

ditto.

[...]
> +extern long cap_prctl_drop(unsigned long cap);
> +#else
> +#include <linux/errno.h>
> +static inline long cap_prctl_drop(unsigned long cap)
> +{ return -EINVAL; }
> +#endif
> +
> +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.

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.

> + current->cap_effective = cap_intersect(current->cap_effective,
> + current->cap_bset);
> + current->cap_permitted = cap_intersect(current->cap_permitted,

> + current->cap_bset);
> + current->cap_inheritable = cap_intersect(current->cap_inheritable,
> + current->cap_bset);

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().

That is, CAP_SETPCAP permits the current process to raise 'any' pI
capability. I suspect that you'll want to prevent raising any bits not
masked by this:

   pI' & ~(pI | (pP & cap_bset)).

Cheers

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

iD8DBQFHQlclQheEq9QabfIRAkF4AKCHuvuL23GjPB+bLHhBP7etBGn4/gCeL74C
PIl6wm41m0dGNiGb1mKFGGU=
=7qUX
-----END PGP SIGNATURE-----