

Re: Time to remove LSM (was Re: [RESEND][RFC][PATCH 2/7] implementation of LSM hooks)

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-04/msg03147.html>

- *From:* Stephen Smalley <sds@xxxxxxxxxxxxx>
 - *Date:* Tue, 18 Apr 2006 08:59:46 -0400
-

On Tue, 2006-04-18 at 07:22 -0500, Serge E. Hallyn wrote:

Quoting Valdis.Kletnieks@xxxxxx (Valdis.Kletnieks@xxxxxx):

On Mon, 17 Apr 2006 22:26:24 BST, Alan Cox said:

(Two replies to this paragraph, addressing 2 separate issues....)

You can implement a BSD securelevel model in SELinux as far as I can see from looking at it, and do it better than the code today, so its not really a feature drop anyway just a migration away from some fossils

If we heave the LSM stuff overboard, there's one thing that **will** need addressing – what to do with kernel support of Posix-y capabilities. Currently some of the heavy lifting is done by `security/commoncap.c`.

Frankly, that's **another** thing that we need to either **fix** so it works right, or rip out of the kernel entirely. As far as I know, there's no in-tree way to make `/usr/bin/ping` be set-`CAP_NET_RAW` and have it DTRT.

As far as the capability module is concerned, ping can be setuid and can drop all other capabilities at startup, so you can approximate the above. As far as SELinux is concerned, you can bound the capabilities that ping can exercise based on its security context even if ping runs as uid 0 and is granted full capabilities by the capability module, regardless of whether ping ever drops capabilities itself. So there are in-tree ways to achieve what you want.

Re: Time to remove LSM (was Re: [RESEND][RFC][PATCH 2/7] implementation of LSM hooks)

Sigh... it's such a cool idea, and yet such a dangerously easy thing to get wrong, ie dropping the ability for a root process to drop its root privs.

If we were to drop posix caps, how would selinux change correspondingly? Would it just drop the capability class altogether, perhaps beef up the task or security class? Just wondering whether anyone had thought about this.

I doubt you'd drop capability altogether. You could incrementally enable the direct granting of capabilities based on SELinux security context by defining a new class in its policy (cap_override) that mirrors the existing capability class, and modifying SELinux to authoritatively grant the capability if it is allowed in that class for the process' security context; otherwise, you fall back to the existing combined behavior of requiring both SELinux+capability (or SELinux+dummy) to grant the capability. That is simple enough from a code perspective. You just need to be careful about the implications for userspace and policy configuration, to avoid introducing security holes in this manner.

Alternatively, we could try yet again to get support for fs caps upstream...

Given the extensible nature of the security xattr namespace, it is already possible to store the capability bits in the filesystem (just by beginning to use security.fficap, security.inhcap, ...). The code modifications to the capability module should be simple. But modifying userspace to set and preserve those attributes would take some work, work which has already been done for the SELinux attributes, and you'd have to figure out the right set of capability bits for each program, which doesn't scale very well (vs. using equivalence classes as in SELinux types). And the capability evolution logic has always been problematic, vs. explicit transition definitions as in SELinux. So fs caps doesn't seem very promising to me as a path forward.

--

Stephen Smalley
National Security Agency

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>