Subject: Re: LSM and Containers
Posted by Crispin Cowan on Tue, 23 Oct 2007 17:57:36 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Quoting Crispin Cowan (crispin@crispincowan.com):
>
>> It is my understanding of containers that they are intended to be a
>> *lightweight* virtualization technique, giving each container
>> effectively a private copy of identical instances of the host OS.
>>
>> If you want to rent out divergent distros, kernels, etc. then it seems
>> to me that heavier virtualization like Xen, KVM, VMware, etc. are the
>> right answer, rather than trying to force difficult kernel solutions
>> into the container and LSM features into the kernel.
>>
> For different kernels, yes, but unless you pick two distros which
> require incompatible kernel features (?) I don't see running, say,
> gentoo, fedora, and ubuntu under different containers as a problem.
>
> Perhaps the biggest reason not to do that, speaking practically, is that
> you miss out on some of the ability to share /usr, /lib, etc readonly
> among containers to save overall disk space.
>
This is why it just doesn't seem very reasonable. People who want to do
that will just use KVM or Xen. People who want the efficiency of
lightweight containers, and have enough load pressure to care, can just
have one Fedora physical box with many containers all running Fedora,
and another openSUSE physical box with many containers all running openSUSE.

I can see how you *could* manage to run different distros in different
containers, but you would have to make many compromises. No sharing of
read-only disk as Serge said. You would have to pick one kernel, as no 2
distros I know of actually run the same kernel. You would have to pick
one set of device drivers, and one LSM. By the time you deal with all
this crap, just using KVM or Xen starts to look good :-)

>> I call it "difficult" because you would have to build a great big switch
>> into the LSM interface, so that each hook is dispatched to the LSM
>> module being used by the current container. This will impose some
>> complexity and overhead, making each hook slower. Worse,the semantics
>> become painfully undefined if a syscall by one container touches an
>> object owned by a different container; which LSM gets called to mediate
>> the access?
>>
> At first my thought was this is worse than dealing with stacker.
>
> But on the other hand, perhaps introducing some sort of 'personality' to

> objects and subjects, where the personality decides which LSM is invoked
> for access, can be done more optimally than one would think. It would
> probably require strict enforcement that two "things" with different
> personalities can NOT mix, ever.
>
Yes, that's what you would have to do. But I basically don't think it is
a good idea; use full virtualization if you don't want to share kernel
features among your virtual guests.

>> What makes a *lot* more sense to me is for individual LSMs to try to
>> "containerize" themselves. This is actually the AppArmor plan: we hope
>> to eventually support having a private AppArmor policy per container.
>>
> Agreed, that had been my assumption. That, and that the configuring
> of LSM policies inside a container would simply be disabled if say
> loading a suse container under a fedora host.
>
This is why running an openSUSE container under a Fedora host (or vice
versa) seems daft to me.

>> Thus all of the containers on the physical machine will be running
>> identical kernels, and all will use AppArmor, but each one can have a
>> different AppArmor policy set, so that e.g. my private Apache process
>> instance is confined in my container different than your Apache process
>> is confined in your container.
>>
>> I see no barrier to SELinux or SMACK or TOMOYO doing the same thing. But
>> I see *big* barriers to trying to support multiple LSM modules in the
>> kernel at the same time with each container using the LSM of its choice.
>>
> It's not as clear to me how SMACK (or the MLS/MCS portion of selinux)
> would be handled.
>
That actually seems easier to me. You may not need to do anything at all
to the MLS/MCS code. There is no actual "policy" in MLS, it is just a
single policy (label dominance) and all of your "policy" is expressed
through labeling. So what you do is make the container ID be part of the
label schema, and poof! none of the containers are permitted to touch
objects belonging to any others, because they cannot dominate each other.

Crispin

--
Crispin Cowan, Ph.D. http://crispincowan.com/~crispin/
  Itanium. Vista. GPLv3. Complexity at work

_____
Containers mailing list