Subject: Re: LSM and Containers (was: LSM conversion to static interface)
Posted by serue on Tue, 23 Oct 2007 13:32:10 GMT
View Forum Message <> Reply to Message

Quoting Crispin Cowan (crispin@crispincowan.com):
> Peter, I may be mistaken, but I think you are talking about an entirely
> different issue than the LSM static interface issue, so I've changed the
> subject.
>
> Peter Dolding wrote:
> > You are all focusing on vendors. I am thinking server farm or people
> > running many different distros side by side using containers.
> This right here is a challenging goal.
>
> It completely surprises me that anyone would consider trying to run
> different distros in different containers.

It seems reasonable to me.

> It would especially surprise
> me if one tried to run different kernels in different containers.

That's not just unreasonable, it's impossible :)

> 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.

> 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.

> 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.

> 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.

-serge

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
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: LSM and Containers
Posted by Peter Dolding on Wed, 24 Oct 2007 00:07:50 GMT
View Forum Message <> Reply to Message

Crispin Cowan wrote:
> 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 :-)
>
Sorry the reason for doing it is 1 set of driver. In particular
Opengl. KVM, Xen, Lguest... All virtual machines cannot handle it
effectively so you have one very high comprise. Containers can.

Same with other devices some devices don't take well at all being
wrapped up in KVM, Xen and other systems. Running the same kernel is
not a issue to me.

Most distributions I have run with standard from kernel.org with there
LSM. Nothing else added to kernel. So the different kernel is null
and void. If LSM are rebuild we will see distro kernels with support
for a broad range of distro support. You may have a distro that allows
like the top 10 server Distributions to be run under it. Of course you

are not going to be using any stock kernel from those Distros if they don't support running other distros. I would not bring it up if I had not run OpenSuse Fedora and Debian and other distros of the same kernel before. How simple in initrd load the LSM for the Distro that was it. There is no technical problem at the kernel for doing it. Also part of the reason why I despise static LSM's becoming only option cross distro performance testing of a kernel to see if there is a difference will be made harder since I will have to build the kernel more than once. Now when I get to containers you are now saying I cannot do the same thing.

I know current LSM model does not fit well for this. Because its a giant hooking solution. This solution is not suitable to exist well with containers and massively limiting on security improvements to the over all os that can be done..

The LSM model needs ripping in two to work great with containers and along the way allow applications and users to take care of there own problems.

Enforcement modules and Controllers.

Controllers work on threads and processors created to allocate security limitations. LSM level controller can grant higher security access to threads and processors than what they all ready have to the level of the LSM. User/program level controllers can only remove permissions to do things this is most likely better as a feature of the Enforcement modules. LSM level controllers have to be attached to kernel or a security containers. Since a container could all ready have limits from another LSMs put on it could only allocate permissions inside those limts.

Enforcement is exactly that. Enforcement modules would be like posix file caps and so on. An option has to be choose here if Enforcement modules have to report to LSM level controller on secuirty change request to lower and expand or only to expand or not at all. Not at all being default because it could be a new feature the controller does not know how to handle or want to handle. Most likely this would be controller option on what if any information it wants out of the Enforcement modules. Default operation of all enforcement modules is like posix file caps is lower only. No option to expand what has all ready been given. With a LSM controller asking to be given notice of change of a Enforcement section it can allow raising or forbid lowering. Applications developers like GUI Filemanagers wine... Can depend on the Enforcement parts always being there.

This is a overall system wide upgrade of security that does not fail completely just because you disable the contoller. Because if all applications are doing there own internal thread based and process id

based enforcements what can be far tighter than what any controller can do since they know what the application should be doing where.

I hear different people asking for a LSM to be build into kernel. This is not a requirement. The enforcement features of LSM's are. Each one of the enforcement features is a upgrade to the overall security of the operating system no matter the LSM loaded or even if LSM is loaded.

Basically just like containers take responsibility for parts individually when put into once piece become a virtual server. The same needs to be done with LSM's. The controller is the final thing that joins the parts up. Not coming with the parts in one huge blob as LSM do now. The huge blob problem is why they don't want to work right containers. Over head is small with Controllers. From Enforcement module to Controller is always the same depth. When Controller goes to change permissions its checked against the list its allowed. At this point does not matter if you are 1 deep or 1000 deep. Since there is no need to go any deeper or run other Controllers to get information. Just the first Controller the one you might statically build in gets told it has every permission to do whatever it sees fit maybe. The first Controller at build time could have its rights limited. Just like selinux strict mode for all. As you can see 100 percent the same path level no matter what. Only thing that changes in what controller you are in.

As bad as it sounds to some people doing this will lower the security difference between selinux apparmor or other LSM's . This forces user and security friendly solutions out of LSM makers. Yes true completion to produce the best and leave users as least exposed as able. Not my LSM is better than yours arguments with complexity of comparing. It will be simpler to compare LSM's ok what enforcement modules does it control. Does that cover the area I need. Then look at how it controls them and choose.

Note LSM controllers could be Security Containers. Just one Container is set default off the start line.

Peter Dolding