Subject: Re: LSM and Containers
Posted by serue on Thu, 25 Oct 2007 01:44:13 GMT
View Forum Message <> Reply to Message

Quoting Peter Dolding (oiaohm@gmail.com):
> On 10/25/07, Crispin Cowan <crispin@crispincowan.com> wrote:
> > Peter Dolding wrote:
> > > The other thing you have not though of and is critical. If LSM is the
> > > same LSM across all containers. What happens if that is breached and
> > > tripped to disable. You only want to loss one container to a breach
> > > not the whole box and dice in one hit. Its also the reason why my
> > > design does not have a direct link between controllers. No cascade
> > > threw system to take box and dice.
> > >
> > Sorry, but I totally disagree.
> >
> > If you obtain enough privilege to disable the LSM in one container, you
> > also obtain enough privilege to disable *other* LSMs that might be
> > operating in different containers. This is a limitation of the
> > Containers feature, not of LSM.
> >
> That is not a Container feature. If you have enough privilege does
> not mean you can. Root user in a Container does not mean you can play
> with other containers applications. There is a security split at the
> container edge when doing Virtual Servers what by using one LSM you
> are disregarding.
>
> Simple point if one LSM is disabled in a container it can only get the
> max rights of that Container. So cannot see the other LSM's on the
> system bellow it. Reason also why in my model its the same layout if
> there is 1 or 1000 stacked so attack cannot tell how deep they are in
> and if there is anything to be gained by digging. You have to break

You're sometimes hard to parse, but here are a few basic facts within
which to constrain our discussions:

1. LSMs are a part of the kernel. The entire kernel is in the
   same trusted computing base
2. containers all run on the same kernel
3. whether an lsm is compromised, or a tty driver, or anything
   else which is in the TCB, all containers are compromised
4. it is very explicitly NOT a goal to hide from a container
   the fact that it is in a container. So your 'cannot tell how
   deep they are' is not a goal.

If you want to be able to 'plug' lsms in per container, by all means
feel free to write a proof of concept. It is kind of a cool idea. But
be clear about what you'll gain: You allow the container admin to

contrain data access within his container in the way he chooses using
the model with which he is comfortable. It does nothing to protect one
container from another, does nothing to protect against kernel exploits,
and absolutely does nothing to protect a container from the 'host'.

Also please keep in mind that the container security framework is not
only not yet complete, it's pretty much not started. My own idea for
how to best do it are outlined in emails which are in the containers
list archive. But in terms of LSM they follow the idea Crispin
outlines, namely that the LSMs support containers themselves. And, in
a proces in a container started without CAP_NS_OVERRIDE (which does not
yet exist :) in its capability bounding set will only be able to access
files in another container as DAC user 'other' (i.e if perms are 754, it
will get read access, if 750, then none), even if it has
CAP_DAC_OVERRIDE. (unless it gets an authorization key for the owning
user in the target namespace, but *that's* probably *years* off)

-serge

_____
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 Thu, 25 Oct 2007 04:31:11 GMT
View Forum Message <> Reply to Message

> You're sometimes hard to parse, but here are a few basic facts within
> which to constrain our discussions:
>
> 1. LSMs are a part of the kernel. The entire kernel is in the
> same trusted computing base
> 2. containers all run on the same kernel
> 3. whether an lsm is compromised, or a tty driver, or anything
> else which is in the TCB, all containers are compromised
> 4. it is very explicitly NOT a goal to hide from a container
> the fact that it is in a container. So your 'cannot tell how
> deep they are' is not a goal.
You are missing what I am saying.

I am breaking the LSM in two. Kernel level comprise that is out of my hands.

If you wished in mine the controller could be user space run outside
kernel does not have to stay in the TCB. Since its not having any
direct control on the approval or rejection of access. Its only
queried when needed like when a process breaches the rules applied to
it or when processes start or changing security. So breaching it

turning it off and so on. This is to reduce the risk of someone trying to customize a LSM can cause a security flaw of hell level(Inside kernel space).

With Intel's and Amd new memory table splitting yes the controller could be put away from the main kernel in time with the rest of the container only data. Yes less bits trusted. This includes when sending the container between machines the controller with it current state could be sent with it. If the interface to controller is standard of course. Ok apparmor and selinux will cope with your system but some of the more complex state based need to transfer state as well this should be in the controller bit of a LSM and not mixed up with the state of the overall machine either.

Over all this is making it simpler to do advanced container things like sending between servers.

The permission enforcement parts stay the same no matter the controller in use. Breaching the enforcement parts still remain a problem. Allows running many controllers unlike LSM's where they can fight and on way around that cleanly. Just like LSM two controllers controlling the same space would still fight so guess how many each container take 1. Since a different controller has to be give its own zone. So people cannot complain about not being able to run there preferred controller if they want a different one they have to give it a different zone. Yes this is getting on top of a lot of long term problems of people wanting a or b LSM. Stuff it have both and be happy.

Its also removing the single overall kill switch. And replacing them with a per container kill switches. So someone does what will normally turn X controller off and it does. Yet still the outside security is applied to everything run in the container. Security is lowed but not off.

The controllers don't need to keep track of how deep they are. They just don't need to know. From there point of view the world ends from there startup security settings. Only time the controller would have to be give notice is if those outside security settings were changed so it could update the programs under it to the way it wanted. Even that the security settings if they were lower would have just been brute forced applied. Basically its have you done containers running containers is you model Serge E. Hallyn. This avoids have to much around with flags for the next container in. Only thing in mine are you doing is changing the path back to the controller from the security enforcing modules. Enforcing modules process not doing permissions were told to allow I report that to x secuirty container that reports it to contoller. Note the security container only hands

out what its allowed nothing more the controller is not trusted. No processing to work out what security files it should be using since the controller only knows the files it should be using.

Yes there is overhead in my system. But were able the system is avoiding traveling in a perfectly functioning system controller could be only being bothered when new processors were being created the rest of the time the enforcement modules just do there job as they were configured.

Traveling down a tree of stacked LSM is going to cause massive lag. Ie knowing that something is below you. Not knowing reduces travel and time to resolve. And allows security to be controlled over the complete container from one spot. The problem with running many LSM again having to tweek them all to get it locked down on container. Ie enabling some disabling others so simple to loss track and have big problems. No problem here the secuirty model is applied to the container and if you want to lock its outside down you just do and the controller inside has to cope. Yes it will be possible to take too much away from a security container. Ok this security container should not have access to that device slam not a question. With running many LSM the question what one does that container own to so I do turn that device off and not have another LSM turn it back on.

Besides having selinux and apparmor installed side by side is going to lag the system due to overlapping hooks. This what I am avoiding. The enforcement modules are only hooked in once no reason at all to overlap them. This overlapping is what makes running many LSM problematic.

The outside security on a container will stay on even if the host LSM is turned off in the design I am look at. It was set when the container started and would have to be directly updated. The outside security defined to a container would be everything selinux could do to a program and then some. Just like selinux opt in.

Over lapping I am handling differently in this design. The outside security set on the container covers overlaps. Far more flex able option. Using directory filtering and other options are on the table as well as far more complex cap options. My design depends on the common protective modules. Not the bit that says this is Mac or Role based or State or so on.

Basically apparmor and selinux would be sitting on the same engine that everyone can use to build solid controllers. If you want a new enforcement feature you would add it for everyone. Since altering the controller would be toothless only can tweak what already exists since you can never be sure its even in kernel space no direct tweaking no

hooking no bad stuff with controllers.

There is a security advantage from the design change even for people not using containers or LSM. Everything in system can apply LSM style restrictions lower than what it has to what it wants to run even if a controller is present or not. The controller is to provide the security model in use. With the enforcement system you could almost do any security model with user space code. The difference being not catching process starts to apply security to them. This is useful for programs running untrusted content web browers that java program running from web does not need all the access I am cut it back. Application level security something LSM overlook and it not wise to use different interfaces for it.

Yes these are major changes in design. Yes they will force clear designs and more sharing so everyone can take advantage of advancements. As well should kill off lots of problems of poor quality LSM's and LSM limitations. Basically my design and LSM cannot live side by side effectively.

Peter Dolding

_____