Subject: Re: [RFC][PATCH] Devices visibility container
Posted by serue on Mon, 24 Sep 2007 15:51:39 GMT
View Forum Message <> Reply to Message

```
Quoting Pavel Emelyanov (xemul@openvz.org):
> Serge E. Hallyn wrote:
> > Quoting Pavel Emelyanov (xemul@openvz.org):
> >> Serge E. Hallyn wrote:
> >>> Quoting Pavel Emelyanov (xemul@openvz.org):
> >>>> Hi.
> >>>>
> >>>> At KS we have pointed out the need in some container, that allows
> >>>> to limit the visibility of some devices to task within it. I.e.
> >>>> allow for /dev/null, /dev/zero etc, but disable (by default) some
> >>>> IDE devices or SCSI discs and so on.
> >>>>
> >>>> Here's the beta of the container. Currently this only allows to
> >>>> hide the _character_ devices only from the living tasks. To play
> >>>> with it you just create the container like this
> >>>>
> >>>>  # mount -t container none /cont/devs -o devices
> >>>>  # mkdir /cont/devs/0
> >>>>
> >>>> it will have two specific files
> >>>>
> >>>>  # ls /cont/devs
> >>>> devices.block  devices.char  notify_on_release  releasable  release_agent  tasks
> >>>>
> >>>> then move a task into it
> >>>>
> >>>>  # /bin/echo -n $$ > /cont/devs/0/tasks
> >>>>
> >>>> after this you won't be able to read from even /dev/zero
> >>>>
> >>>>  # hexdump /dev/zero
> >>>> hexdump: /dev/zero: No such device or address
> >>>> hexdump: /dev/zero: Bad file descriptor
> >>>>
> >>>> meanwhile from another ssh session you will. You may allow access
> >>>> to /dev/zero like this
> >>>>
> >>>>  # /bin/echo -n '+1:5' > /cont/devs/0/devices.char
> >>>>
> >>>> More generally, the '+<major>:<minor>' string grants access to
> >>>> some device, and '-<major>:<minor>' disables one.
> >>>>
> >>>> The TODO list now looks like this:
> >>>> * add the block devices support :) don't know how to make it yet;
```

> >>>> * make /proc/devices show relevant info depending on who is
> >>>>   reading it. currently even if major 1 is disabled for task,
> >>>>   it will be listed in this file;
> >>>> * make it possible to enable/disable not just individual major:minor
> >>>>   pair, but something more flexible, e.g. major:* for all minors
> >>>>   for given major or major:m1-m2 for minor range, etc;
> >>>> * add the ability to restrict the read/write permissions for a
> >>>>   container. currently one may just control the visible-invisible
> >>>>   state for a device in a container, but maybe just readable or
> >>>>   just writable would be better.
> >>>>
> >>>> This patch is minimally tested, because I just want to know your
> >>>> opinion on whether it worths developing the container in such a way or not.
> >>> Hmm,
> >>>
> >>> I was thinking we would use LSM for this. Mostly it should suffice
> >>> to set up a reasonable /dev for the container to start with, and
> >>> hook security_mknod() to prevent it creating devices not on it's
> >> Are you talking about disabling of mknod() for some files? No, please
> >> no! This will break many... no - MANY tools inside such a container.
> >
> > What's going to break if I don't allow mknod /dev/hda1? Is this during
> > standard /sbin/init for a container? And what does 'break' mean? If
> > you're not allowed to use the device, why should we pretend that you
> > can create it? Isn't that more devious?
>
> Standard linux kernel allows you to create any devices you wish,
> so container must operate the same way.

And security_mknod() in standard linux kernel allows you to prevent it
being created.

> Besides, what to do if you have enables some device to it, then the
> container user creates it and after this you disable it again. In this

Well then you need to hire a new admin who can't be bribed :)

But seriously, that's where security_file_permission() would have a
check on open.

> case user will still be able to open the device and work with it :(
> With my approach we will return -EPERM during this open :)

So would security_file_permission().

> Or some better example - container owner mounts some external ext3
> partitions with plenty of deices on it. No way to disable their
> usage unless you control their open().

We can control their open(), control their read/write, and and make
their external ext3 be mounted NODEV to boot if we want.

> > A straight -EPERM on mknod just feels more warm+fuzzy to me. But if
> > things really are going to break to where you can't run a standard
> > distro in a container, then I guess we should go with your approach.
>
> If udef fails to create a statically requested device it may break.
> With broken udev no containers will work (using some latest distros).

This sounds like both the best and the worst argument all wrapped into
one :)

If udev is going to fail, and the container won't 'boot', and there is
no way around this, then maybe my approach isn't workable.

On the other hand, if udev isn't allowed to create some device, it
should just fail and move on. No excuse for it to fail the whole
system boot.

> Moreover - if you later grant access to this device udev won't try
> to re-create it again unless specially asked.

So?

Now again, mind you I'm just offering an alternative. I'm not objecting
to your patch, other than that it seems less straightforward to me.

Maybe it's best if i just code up an example. I'll be gone part of the
week, and should send out new versions of some other patches first, but
getting two approaches to this to compare and contrast can't hurt.

thanks,
-serge