Subject: Re: [RFC][PATCH] Devices visibility container
Posted by Pavel Emelianov on Mon, 24 Sep 2007 14:58:10 GMT
View Forum Message <> Reply to Message

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.

> whitelist. If deemed necessary, read/write could be controlled
> by hooking security_permission() and checking whether
> file->f_path.dentry->d_inode is a device on the read or write
> whitelist.
>
> It would still be a device controller, so it can be composed with an
> ns_proxy controller, and the whitelist is modified using the
> devs_controller.whitelist file, but it registers a security_ops
> with these two hooks.
>
> I haven't implemented that yet, though, whereas you already have code :)
> As for handling blkdevs with your code, would just hooking
> fs/block_dev.c:do_open() not work? Or is that not what you are
> asking?

Well, placing a hook into needed functions is something that can
work, of course, but this is not something that community would like
to see, so I tried to integrate them deeply.

> thanks,
> -serge