Subject: Re: [PATCH 1/4] sysfs: Remove first pass at shadow directory support
Posted by ebiederm on Sun, 22 Jul 2007 19:17:18 GMT
View Forum Message <> Reply to Message

Tejun Heo <htejun@gmail.com> writes:

> Hello,
>
> Hope my late review isn't too late.
>
> Eric W. Biederman wrote:
>> While shadow directories appear to be a good idea, the current scheme
>> of controlling their creation and destruction outside of sysfs appears
>> to be a locking and maintenance nightmare in the face of sysfs
>> directories dynamically coming and going.
>
> Yeah, shadow directory was a real PITA while restructuring sysfs. I
> tried hard to keep it working but, without test case and detailed
> documentation, I'm pretty sure I broke something.
>
> My feeling was that it was merged too early and implementation was too
> tightly coupled with other more regular paths.

It's a problem because sysfs has this tendency especially before your
cleanups to be tightly coupled.

But yes the original implementation was probably factored at the wrong
level, not giving sysfs enough responsibility. Which became apparent
when the single objects started living all over the sysfs tree.

> Anyways, glad shadow
> dirs are getting some loving care. If properly done, we should be able
> to simplify sysfs_get_dentry(), removal logic and save a pointer in
> sysfs_dirent.

Hopefully. The logic in sysfs_resolve_for_create (i.e. which shadow
dir do you intend to create something is tricky with the current locking
logic).

>> Which can now occur for
>> directories containing network devices when CONFIG_SYSFS_DEPRECATED is
>> not set.
>
> Shadow directories were always pinned - both the shadowed one and
> shadows. Well, that was the theory at least.

Yes. The difference I was referring to was:
/sys/class/net/ used to be the only directory I needed to shadow.

Now there is one net/ in each networking device and several other
little things.

>> -int sysfs_rename_dir(struct kobject *kobj, struct sysfs_dirent
> *new_parent_sd,
>> -   const char *new_name)
>> +int sysfs_rename_dir(struct kobject * kobj, const char *new_name)
>> {
>> - struct sysfs_dirent *sd = kobj->sd;
>> - struct dentry *new_parent = NULL;
>> + struct sysfs_dirent *sd;
>> + struct dentry *parent = NULL;
>>   struct dentry *old_dentry = NULL, *new_dentry = NULL;
>> + struct sysfs_dirent *parent_sd;
>>   const char *dup_name = NULL;
>>   int error;
>>
>> + if (!kobj->parent)
>> + return -EINVAL;
>
> Please don't do this. The goal is to decouple sysfs and kobj.

I have no problem moving that test back to a higher level.
My practical goal of this first patch was to remove the original
shadow dir work so I had a clean slate to start with. I got
tired and perhaps didn't do as clean a break here as I could
have.

Do you mind an incremental patch to move the kobj->parent test?

As long as we don't break git-bisect support I would really prefer
to just build on top of the patches that are there.

It has been a major hair pulling exercise to get everything to get
everything to work in the presence of other parallel changes in sysfs.
I only had to relearn the locking and organization rules 3 times. I
really don't want to repeat it but I am happy to incrementally improve
things.

>>   /* get dentries */
>> + sd = kobj->sd;
>>   old_dentry = sysfs_get_dentry(sd);
>>   if (IS_ERR(old_dentry)) {
>>   error = PTR_ERR(old_dentry);
>>   goto out_dput;
>>   }
>>

>> - new_parent = sysfs_get_dentry(new_parent_sd);
>> - if (IS_ERR(new_parent)) {
>> - error = PTR_ERR(new_parent);
>> + parent_sd = kobj->parent->sd;
>> + parent = sysfs_get_dentry(parent_sd);
>> + if (IS_ERR(parent)) {
>> + error = PTR_ERR(parent);
>>   goto out_dput;
>
> You can simply do parent = old_dentry->d_parent. parent is guaranteed
> to be there as long as we hold old_dentry. In the original code,
> new_parent needed to be looked because new_parent wasn't necessarily
> old_dentry's parent.

Yes. This part is probably the ugliest, and least well factored part
of this patchset. sysfs_rename_dir wasn't something I could completely
revert to it's pre shadow directory state as that was no longer
available. What do you think of sysfs_rename_dir after the second patch in
the patchset?

I still think I have the limitation that new_parent and old_parent may
be separate. (Although clearly not at this point in the patch series).
I seem to remember trying to minimize the flux in sysfs_rename_dir and
eventually giving up. Having found a path that seemed to be good enough.

>> @@ -965,10 +957,10 @@ int sysfs_rename_dir(struct kobject *kobj, struct
> sysfs_dirent *new_parent_sd,
>>   out_drop:
>>   d_drop(new_dentry);
>>   out_unlock:
>> - mutex_unlock(&new_parent->d_inode->i_mutex);
>> + mutex_unlock(&parent->d_inode->i_mutex);
>>   out_dput:
>>   kfree(dup_name);
>> - dput(new_parent);
>> + dput(parent);
>
> So, dput(parent) can go away too.

I actually managed to remove sysfs_get_dentry entirely from
sysfs_rename_dir and to use sysfs_addrm_start.

>> diff --git a/fs/sysfs/group.c b/fs/sysfs/group.c
>> index f318b73..4606f7c 100644
>> --- a/fs/sysfs/group.c
>> +++ b/fs/sysfs/group.c
>> @@ -13,7 +13,6 @@
>> #include <linux/dcache.h>

>> #include <linux/namei.h>
>> #include <linux/err.h>
>> -#include <linux/fs.h>
>> #include <asm/semaphore.h>
>> #include "sysfs.h"
>
> Can you explain this one a bit?

This include was added by the original shadow directory support
and with it the original shadow directory support removed the include
became unnecessary. I forget the exact details.

But so I could catch subtle things like that is why my patchset is
structured as first a removal pass and then the new stuff.

Eric

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