Subject: Re: - merge-sys_clone-sys_unshare-nsproxy-and-namespace.patch removed from -mm tree
Posted by Cedric Le Goater on Mon, 18 Jun 2007 12:37:25 GMT
View Forum Message <> Reply to Message

Herbert Poetzl wrote:
> On Sun, Jun 17, 2007 at 06:38:30PM +0400, Oleg Nesterov wrote:
>> On 06/16, Herbert Poetzl wrote:
>>> On Tue, May 08, 2007 at 07:45:35PM -0700, akpm@linux-foundation.org wrote:
>>>> The patch titled
>>>>   Merge sys_clone()/sys_unshare() nsproxy and namespace handling
>>>> has been removed from the -mm tree. Its filename was
>>>>   merge-sys_clone-sys_unshare-nsproxy-and-namespace.patch
>>>>
>>>> This patch was dropped because it was merged into mainline or a subsystem tree
>>>>
>>> .. [zapped] ...
>>>
>>>> + * Called from unshare. Unshare all the namespaces part of nsproxy.
>>>> + * On sucess, returns the new nsproxy and a reference to old nsproxy
>>>> + * to make sure it stays around.
>>>> + */
>>>> +int unshare_nsproxy_namespaces(unsigned long unshare_flags,
>>>> + struct nsproxy **new_nsp, struct fs_struct *new_fs)
>>>> +{
>>> this makes sys_unshare leak and nsproxy (reference)
>>>
>>> can be tested with the following command sequence:
>>>   vcmd -nu ^17 -- vcmd -nu ^17 -- sleep 10
>> I know almost nothing about this stuff, could you please explain in
>> brief what this command does ...
>
> yeah, sure, it basically calls sys_unshare() with
> bit 17 (CLONE_NEWNS) set then invokes the chained
> command, so we get a sleep which is in a separate
> namespace, unshared from a namespace != the main
> one ...
>
>> ... and how do you detect a leak?
>
>>> (and some nsproxy accounting/debugging as used in
>>> Linux-VServer)
>
> on Linux-VServer,we have accounting for those
> proxies (and several other namespace related stuff)
> because we already suspected leakage and reference
> bugs in this area some time ago ... btw, I also
> suggested to put a similar functionality in mainline

> for the time being, but it was ignored, as usual ...
>
>>> we probably want to drop the reference to the old
>>> nsproxy in sys_unshare() but I do not see a good reason
>>> to take the reference in the first place (at least not
>>> with the code in mainline 2.6.22-rc4)
>> At first glance, sys_unshare() drops the reference to
>> the old nsproxy,
>
> okay, the 'current' task has an nsproxy, and keeps
> a reference to that (let's assume it is the only
> task using this nsproxy, then the count will be 1)
>
> unshare_nsproxy_namespaces() now does get_nsproxy()
> which makes the count=2, then it creates a new
> nsproxy (which will get count=1), and returns ...
>
>>   old_nsproxy = current->nsproxy;
>>   current->nsproxy = new_nsproxy;
>>   new_nsproxy = old_nsproxy;
>
> sys_unshare, now replaces the current->nsproxy with
> the new one, which will have the correct count=1,
> and puts the old nsproxy (which has count=2), and
> thus the nsproxy will not get released, although
> it isn't referenced/used anymore ...

Herbert,

Could you give a try to the patch i've sent previously and this one
which removes an extra get_nsproxy() ? It fixes the leak for me. I've
run the ltp tests we have on namespace unsharing and i could see the
no leaks in /proc/slabinfo.

Badari,

That extra get_nsproxy() seemed a superfluous remain from the 2.6.20.
Do you see any issues with it ?

If we're all happy with these fixes, i'll send them on lkml@ for review.
They might deserve to be in 2.6.22.

Thanks,

C.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

```
---
 kernel/nsproxy.c |    7 +------
 1 file changed, 1 insertion(+), 6 deletions(-)

Index: 2.6.22-rc4-mm2/kernel/nsproxy.c
===================================================================
--- 2.6.22-rc4-mm2.orig/kernel/nsproxy.c
+++ 2.6.22-rc4-mm2/kernel/nsproxy.c
@@ -175,7 +175,6 @@ void free_nsproxy(struct nsproxy *ns)
 int unshare_nsproxy_namespaces(unsigned long unshare_flags,
 		struct nsproxy **new_nsp, struct fs_struct *new_fs)
 {
-	struct nsproxy *old_ns = current->nsproxy;
 	int err = 0;

 	if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
@@ -185,14 +184,10 @@ int unshare_nsproxy_namespaces(unsigned
 	if (!capable(CAP_SYS_ADMIN))
 		return -EPERM;

-	get_nsproxy(old_ns);
-
 	*new_nsp = create_new_namespaces(unshare_flags, current,
 				new_fs ? new_fs : current->fs);
-	if (IS_ERR(*new_nsp)) {
+	if (IS_ERR(*new_nsp))
 		err = PTR_ERR(*new_nsp);
-		put_nsproxy(old_ns);
-	}
 	return err;
 }
```