
Subject: Re: [PATCH 11/11] sysfs: user namespaces: fix bug with clone(CLONE_NEWUSER) with fairsched

Posted by [serue](#) on Tue, 10 Jun 2008 22:41:12 GMT

[View Forum Message](#) <> [Reply to Message](#)

Quoting Benjamin Thery (benjamin.thery@bull.net):

> Mark the /sys/kernel/uids directory to be tagged so that processes in
> different user namespaces can remount /sys and see their own uid
> listings.

>
> Without this patch, having CONFIG_FAIR_SCHED=y makes user namespaces
> unusable, because when you
> clone(CLONE_NEWUSER)
> it will auto-create the root userid and try to create
> /sys/kernel/uids/0. Since that already exists from the parent user
> namespace, the create fails, and the clone misleadingly ends up
> returning -ENOMEM.

>
> This patch fixes the issue by allowing each user namespace to remount
> /sys, and having /sys filter the /sys/kernel/uid/ entries by user
> namespace.

>
> Signed-off-by: Serge Hallyn <serue@us.ibm.com>
> Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>

> ---
> fs/sysfs/mount.c | 3 +++
> include/linux/sched.h | 1 +
> include/linux/sysfs.h | 2 ++
> include/linux/user_namespace.h | 1 +
> kernel/user.c | 21 ++++++++
> kernel/user_namespace.c | 13 ++++++
> 6 files changed, 40 insertions(+), 1 deletion(-)

>
> Index: linux-mm/fs/sysfs/mount.c

> =====
> --- linux-mm.orig/fs/sysfs/mount.c
> +++ linux-mm/fs/sysfs/mount.c
> @@ -81,6 +81,7 @@ static int sysfs_fill_super(struct super
> sb->s_root = root;
> sb->s_fs_info = info;
> info->tag.net_ns = hold_net(current->nsproxy->net_ns);
> + info->tag.user_ns = current->nsproxy->user_ns;
> return 0;
>
> out_err:
> @@ -100,6 +101,8 @@ static int sysfs_test_super(struct super
>
> if (task->nsproxy->net_ns != info->tag.net_ns)

```

> found = 0;
> + if (task->nsproxy->user_ns != info->tag.user_ns)
> + found = 0;
>
> return found;
> }
> Index: linux-mm/include/linux/sched.h
> =====
> --- linux-mm.orig/include/linux/sched.h
> +++ linux-mm/include/linux/sched.h
> @@ -600,6 +600,7 @@ struct user_struct {
> /* Hash table maintenance information */
> struct hlist_node uidhash_node;
> uid_t uid;
> + struct user_namespace *user_ns;
>
> #ifdef CONFIG_USER_SCHED
> struct task_group *tg;
> Index: linux-mm/include/linux/sysfs.h
> =====
> --- linux-mm.orig/include/linux/sysfs.h
> +++ linux-mm/include/linux/sysfs.h
> @@ -20,6 +20,7 @@
> struct kobject;
> struct module;
> struct net;
> +struct user_namespace;
>
> /* FIXME
> * The *owner field is no longer used, but leave around
> @@ -81,6 +82,7 @@ struct sysfs_ops {
>
> struct sysfs_tag_info {
> struct net *net_ns;
> + struct user_namespace *user_ns;
> };
>
> struct sysfs_tagged_dir_operations {
> Index: linux-mm/include/linux/user_namespace.h
> =====
> --- linux-mm.orig/include/linux/user_namespace.h
> +++ linux-mm/include/linux/user_namespace.h
> @@ -12,6 +12,7 @@
> struct user_namespace {
> struct kref kref;
> struct hlist_head uidhash_table[UIDHASH_SZ];
> + struct kset *kset;

```

Hmm, funky. I don't know where this came from - or actually, I think it survived from the second of 4 or 5 quick attempts to figure out where to do the tagging for users...

I don't think the kset should be here :)

Otherwise, looks good, thanks Benjamin.

-serge

```
> struct user_struct *root_user;
> };
>
> Index: linux-mm/kernel/user.c
> =====
> --- linux-mm.orig/kernel/user.c
> +++ linux-mm/kernel/user.c
> @@ -53,6 +53,7 @@ struct user_struct root_user = {
> .files = ATOMIC_INIT(0),
> .sigpending = ATOMIC_INIT(0),
> .locked_shm = 0,
> + .user_ns = &init_user_ns,
> #ifdef CONFIG_USER_SCHED
> .tg = &init_task_group,
> #endif
> @@ -236,6 +237,23 @@ static void uids_release(struct kobject
> return;
> }
>
> +static const void *usersns_sb_tag(struct sysfs_tag_info *info)
> +{
> + return info->user_ns;
> +}
> +
> +static const void *usersns_kobject_tag(struct kobject *kobj)
> +{
> + struct user_struct *up;
> + up = container_of(kobj, struct user_struct, kobj);
> + return up->user_ns;
> +}
> +
> +static struct sysfs_tagged_dir_operations usersns_tagged_dir_operations = {
> + .sb_tag = usersns_sb_tag,
> + .kobject_tag = usersns_kobject_tag,
> +};
> +
> static struct kobj_type uids_ktype = {
> .sysfs_ops = &kobj_sysfs_ops,
```

```

> .default_attrs = uids_attributes,
> @@ -272,6 +290,8 @@ int __init uids_sysfs_init(void)
> if (!uids_kset)
> return -ENOMEM;
>
> + sysfs_enable_tagging(&uids_kset->kobj, &userns_tagged_dir_operations);
> +
> return uids_user_create(&root_user);
> }
>
> @@ -404,6 +424,7 @@ struct user_struct *alloc_uid(struct use
> goto out_unlock;
>
> new->uid = uid;
> + new->user_ns = ns;
> atomic_set(&new->__count, 1);
>
> if (sched_create_user(new) < 0)
> Index: linux-mm/kernel/user_namespace.c
> =====
> --- linux-mm.orig/kernel/user_namespace.c
> +++ linux-mm/kernel/user_namespace.c
> @@ -22,7 +22,7 @@ static struct user_namespace *clone_user
> struct user_struct *new_user;
> int n;
>
> - ns = kmalloc(sizeof(struct user_namespace), GFP_KERNEL);
> + ns = kzalloc(sizeof(struct user_namespace), GFP_KERNEL);
> if (!ns)
> return ERR_PTR(-ENOMEM);
>
> @@ -66,11 +66,22 @@ struct user_namespace *copy_user_ns(int
> return new_ns;
> }
>
> +/* clear sysfs tag when user namespace exits */
> +static void sysfs_userns_exit(struct sysfs_tag_info *tag_info, void *data)
> +{
> + struct user_namespace *ns = (struct user_namespace *)data;
> +
> + if (tag_info->user_ns != ns)
> + return;
> + tag_info->user_ns = NULL;
> +}
> +
> void free_user_ns(struct kref *kref)
> {
> struct user_namespace *ns;

```

```
>  
> ns = container_of(kref, struct user_namespace, kref);  
> + sysfs_ns_exit(sysfs_userns_exit, ns);  
> release_uids(ns);  
> kfree(ns);  
> }  
>  
> --
```

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