Subject: [PATCH 2/2] Merge multiple error paths in alloc_uid into one
Posted by Pavel Emelianov on Wed, 21 Nov 2007 10:53:56 GMT
View Forum Message <> Reply to Message

There are already 4 error paths in alloc_uid() that do incremental
rollbacks. I think it's time to merge them. This costs us 8 lines
of code :)

Maybe it would be better to merge this patch with the previous
one, but I remember that some time ago I sent a similar patch
(fixing the error path and cleaning it), but I was told to make
two patches in such cases.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/kernel/user.c b/kernel/user.c
index 3549c4b..cb6c6f9 100644
--- a/kernel/user.c
+++ b/kernel/user.c
@@ -328,7 +328,7 @@ void free_uid(struct user_struct *up)
 struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
 {
  struct hlist_head *hashent = uidhashentry(ns, uid);
- struct user_struct *up;
+ struct user_struct *up, *new;

  /* Make uid_hash_find() + uids_user_create() + uid_hash_insert()
   * atomic.
@@ -340,13 +340,9 @@ struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
  spin_unlock_irq(&uidhash_lock);

  if (!up) {
-  struct user_struct *new;
-
   new = kmem_cache_alloc(uid_cachep, GFP_KERNEL);
-  if (!new) {
-   uids_mutex_unlock();
-   return NULL;
-  }
+  if (!new)
+   goto out_unlock;

   new->uid = uid;
   atomic_set(&new->__count, 1);
@@ -362,28 +358,14 @@ struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
 #endif
```

```
   new->locked_shm = 0;

-  if (alloc_uid_keyring(new, current) < 0) {
-   kmem_cache_free(uid_cachep, new);
-   uids_mutex_unlock();
-   return NULL;
-  }
+  if (alloc_uid_keyring(new, current) < 0)
+   goto out_free_user;

-  if (sched_create_user(new) < 0) {
-   key_put(new->uid_keyring);
-   key_put(new->session_keyring);
-   kmem_cache_free(uid_cachep, new);
-   uids_mutex_unlock();
-   return NULL;
-  }
+  if (sched_create_user(new) < 0)
+   goto out_put_keys;

-  if (uids_user_create(new)) {
-   sched_destroy_user(new);
-   key_put(new->uid_keyring);
-   key_put(new->session_keyring);
-   kmem_cache_free(uid_cachep, new);
-   uids_mutex_unlock();
-   return NULL;
-  }
+  if (uids_user_create(new))
+   goto out_destoy_sched;

   /*
    * Before adding this, check whether we raced
@@ -411,6 +393,17 @@ struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
  uids_mutex_unlock();

  return up;
+
+out_destoy_sched:
+ sched_destroy_user(new);
+out_put_keys:
+ key_put(new->uid_keyring);
+ key_put(new->session_keyring);
+out_free_user:
+ kmem_cache_free(uid_cachep, new);
+out_unlock:
+ uids_mutex_unlock();
+ return NULL;
```

}

void switch_uid(struct user_struct *new_user)
--
1.5.3.4

Subject: Re: [PATCH 2/2] Merge multiple error paths in alloc_uid into one
Posted by Dhaval Giani on Fri, 23 Nov 2007 13:14:59 GMT
View Forum Message <> Reply to Message

On Wed, Nov 21, 2007 at 01:53:56PM +0300, Pavel Emelyanov wrote:
> There are already 4 error paths in alloc_uid() that do incremental
> rollbacks. I think it's time to merge them. This costs us 8 lines
> of code :)
>
> Maybe it would be better to merge this patch with the previous
> one, but I remember that some time ago I sent a similar patch
> (fixing the error path and cleaning it), but I was told to make
> two patches in such cases.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Looks good.

Acked-by: Dhaval Giani <dhaval@linux.vnet.ibm.com>
>
> ---
>
> diff --git a/kernel/user.c b/kernel/user.c
> index 3549c4b..cb6c6f9 100644
> --- a/kernel/user.c
> +++ b/kernel/user.c
> @@ -328,7 +328,7 @@ void free_uid(struct user_struct *up)
> struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
> {
>   struct hlist_head *hashent = uidhashentry(ns, uid);
> - struct user_struct *up;
> + struct user_struct *up, *new;
>
>   /* Make uid_hash_find() + uids_user_create() + uid_hash_insert()
>   * atomic.
> @@ -340,13 +340,9 @@ struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
>   spin_unlock_irq(&uidhash_lock);
>
>   if (!up) {
> - struct user_struct *new;
> -

```
>    new = kmem_cache_alloc(uid_cachep, GFP_KERNEL);
> -  if (!new) {
> -   uids_mutex_unlock();
> -   return NULL;
> -  }
> +  if (!new)
> +   goto out_unlock;
>
>    new->uid = uid;
>    atomic_set(&new->__count, 1);
> @@ -362,28 +358,14 @@ struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
>  #endif
>    new->locked_shm = 0;
>
> -  if (alloc_uid_keyring(new, current) < 0) {
> -   kmem_cache_free(uid_cachep, new);
> -   uids_mutex_unlock();
> -   return NULL;
> -  }
> +  if (alloc_uid_keyring(new, current) < 0)
> +   goto out_free_user;
>
> -  if (sched_create_user(new) < 0) {
> -   key_put(new->uid_keyring);
> -   key_put(new->session_keyring);
> -   kmem_cache_free(uid_cachep, new);
> -   uids_mutex_unlock();
> -   return NULL;
> -  }
> +  if (sched_create_user(new) < 0)
> +   goto out_put_keys;
>
> -  if (uids_user_create(new)) {
> -   sched_destroy_user(new);
> -   key_put(new->uid_keyring);
> -   key_put(new->session_keyring);
> -   kmem_cache_free(uid_cachep, new);
> -   uids_mutex_unlock();
> -   return NULL;
> -  }
> +  if (uids_user_create(new))
> +   goto out_destoy_sched;
>
>    /*
>     * Before adding this, check whether we raced
> @@ -411,6 +393,17 @@ struct user_struct * alloc_uid(struct user_namespace *ns, uid_t uid)
>   uids_mutex_unlock();
>
```

```
>   return up;
> +
> +out_destoy_sched:
> + sched_destroy_user(new);
> +out_put_keys:
> + key_put(new->uid_keyring);
> + key_put(new->session_keyring);
> +out_free_user:
> + kmem_cache_free(uid_cachep, new);
> +out_unlock:
> + uids_mutex_unlock();
> + return NULL;
>  }
>
>  void switch_uid(struct user_struct *new_user)
> --
> 1.5.3.4
```

--
regards,
Dhaval
