Subject: Re: [RFC][PATCH 1/2] add user namespace [try #2]
Posted by dev on Thu, 07 Sep 2006 15:59:24 GMT
View Forum Message <> Reply to Message

comments below

> This patch adds the user namespace.
>
> Basically, it allows a process to unshare its user_struct table,
> resetting at the same time its own user_struct and all the associated
> accounting.
>
> A new root user (uid == 0) is added to the user namespace upon
> creation. Such root users have full privileges and it seems that
> theses privileges should be controlled through some means (process
> capabilities ?)
>
> Changes [try #2]
>
> - removed struct user_namespace* argument from find_user()
> - added a root_user per user namespace
>
> Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> Cc: Andrew Morton <akpm@osdl.org>
> Cc: Kirill Korotaev <dev@openvz.org>
> Cc: Eric W. Biederman <ebiederm@xmission.com>
> Cc: Herbert Poetzl <herbert@13thfloor.at>
> Cc: Serge E. Hallyn <serue@us.ibm.com>
> Cc: Dave Hansen <haveblue@us.ibm.com>
>
> ---
[... skip ...]
> +struct user_namespace init_user_ns = {
> + .kref = {
> + .refcount = ATOMIC_INIT(2),
<<<< please add some comment about why it is initally = 2

> + },
> + .root_user = &root_user,
> +};
> +
> +EXPORT_SYMBOL_GPL(init_user_ns);
>
> /*
> * The uidhash_lock is mostly taken from process context, but it is
> @@ -84,6 +93,111 @@ static inline struct user_struct *uid_ha
> return NULL;
> }

>
> +
> +#ifdef CONFIG_USER_NS
> +
> +/*
> + * Clone a new ns copying an original user ns, setting refcount to 1
> + * @old_ns: namespace to clone
> + * Return NULL on error (failure to kmalloc), new ns otherwise
> + */
> +static struct user_namespace *clone_user_ns(struct user_namespace *old_ns)
> +{
> + struct user_namespace *ns;
> +
> + ns = kmalloc(sizeof(struct user_namespace), GFP_KERNEL);
> + if (ns) {
<<<< can you remake this function please so that normal case would go without indentation, i.e.:
if (!ns) {
  error path;
}
normal path

> + int n;
> + struct user_struct *new_user;
> +
> + kref_init(&ns->kref);
> +
> + for(n = 0; n < UIDHASH_SZ; ++n)
> +   INIT_LIST_HEAD(ns->uidhash_table + n);
> +
> + /* Insert new root user. */
> + ns->root_user = alloc_uid(ns, 0);
> + if (!ns->root_user) {
> +   kfree(ns);
> +   return NULL;
> + }
> +
> + /* Reset current->user with a new one */
> + new_user = alloc_uid(ns, current->uid);
> + if (!new_user) {
> +   kfree(ns);
> +   return NULL;
> + }
> +
> + switch_uid(new_user);
<<<< I see switch_uid in this function, but you don't change nsproxy->user_ns here...
<<<< it is done much later, in sys_unshare()... This looks a bit inconsistent for me...
<<<< But I'm unsure whether it can be fixed... :/

```
> + }

> + return ns;
> +}
> +
> +/*
> + * unshare the current process' user namespace.
> + */
> +int unshare_user_ns(unsigned long unshare_flags,
> +      struct user_namespace **new_user)
> +{
> + if (unshare_flags & CLONE_NEWUSER) {
> +  if (!capable(CAP_SYS_ADMIN))
> +   return -EPERM;
<<<< such checks for CAP_SYS_ADMIN mean that we can't use copy_xxx/clone_xxx functions
directly
<<<< from OpenVZ code, since VE creation is done with dropped capabilities already.
<<<< (user level tools decide which capabilities should be granted to VE, so CAP_SYS_ADMIN
<<<<  is not normally granted :) )
<<<< Can we move capability checks into some more logical place which deals with user, e.g.
sys_unshare()?

> +
> +  *new_user = clone_user_ns(current->nsproxy->user_ns);
> +  if (!*new_user)
> +   return -ENOMEM;
> + }
> +
> + return 0;
> +}
> +
> +/*
> + * Copy task tsk's user namespace, or clone it if flags specifies
> + * CLONE_NEWUSER. In latter case, changes to the user namespace of
> + * this process won't be seen by parent, and vice versa.
> + */
> +int copy_user_ns(int flags, struct task_struct *tsk)
> +{
> + struct user_namespace *old_ns = tsk->nsproxy->user_ns;
> + struct user_namespace *new_ns;
> + int err = 0;
> +
> + if (!old_ns)
> +  return 0;
> +
> + get_user_ns(old_ns);
> +
> + if (!(flags & CLONE_NEWUSER))
```

> + return 0;
> +
> + if (!capable(CAP_SYS_ADMIN)) {
> + err = -EPERM;
> + goto out;
> + }
<<<< same as above

> +
> + new_ns = clone_user_ns(old_ns);
> + if (!new_ns) {
> + err = -ENOMEM;
> + goto out;
> + }
> + tsk->nsproxy->user_ns = new_ns;
> +
> +out:
> + put_user_ns(old_ns);
> + return err;
> +}
> +

....

Kirill

Subject: Re: [RFC][PATCH 1/2] add user namespace [try #2]
Posted by Cedric Le Goater on Mon, 11 Sep 2006 08:35:45 GMT
View Forum Message <> Reply to Message

Kirill Korotaev wrote:

[...]

>> +static struct user_namespace *clone_user_ns(struct user_namespace *old_ns)
>> +{
>> + struct user_namespace *ns;
>> +
>> + ns = kmalloc(sizeof(struct user_namespace), GFP_KERNEL);
>> + if (ns) {
> <<<< can you remake this function please so that normal case would go without indentation,
i.e.:
> if (!ns) {
>   error path;
> }
> normal path

right, the error path has been growing too much. Will do.

```
>> +  int n;
>> +  struct user_struct *new_user;
>> +
>> +  kref_init(&ns->kref);
>> +
>> +  for(n = 0; n < UIDHASH_SZ; ++n)
>> +   INIT_LIST_HEAD(ns->uidhash_table + n);
>> +
>> +  /* Insert new root user.  */
>> +  ns->root_user = alloc_uid(ns, 0);
>> +  if (!ns->root_user) {
>> +   kfree(ns);
>> +   return NULL;
>> +  }
>> +
>> +  /* Reset current->user with a new one */
>> +  new_user = alloc_uid(ns, current->uid);
>> +  if (!new_user) {
>> +   kfree(ns);
>> +   return NULL;
>> +  }
>> +
>> +  switch_uid(new_user);
> <<<< I see switch_uid in this function, but you don't change nsproxy->user_ns here...
> <<<< it is done much later, in sys_unshare()... This looks a bit inconsistent for me...
```

I would say it is safe but i agree with you that it would be better to
switch user namespace before than switching user_struct.

```
> <<<< But I'm unsure whether it can be fixed... :/
```

he, may be an argument for the clone_ns() syscall ?

```
>> + return ns;
>> +}
>> +
>> +/*
>> + * unshare the current process' user namespace.
>> + */
>> +int unshare_user_ns(unsigned long unshare_flags,
>> +      struct user_namespace **new_user)
>> +{
>> + if (unshare_flags & CLONE_NEWUSER) {
>> +  if (!capable(CAP_SYS_ADMIN))
>> +   return -EPERM;
> <<<< such checks for CAP_SYS_ADMIN mean that we can't use copy_xxx/clone_xxx functions
```

directly
> <<<< from OpenVZ code, since VE creation is done with dropped capabilities already.

OK.

> <<<< (user level tools decide which capabilities should be granted to VE, so CAP_SYS_ADMIN
> <<<< is not normally granted :))
> <<<< Can we move capability checks into some more logical place which deals with user, e.g.
sys_unshare()?

hmm, another argument for a new syscall() ?

C.

Subject: Re: [RFC][PATCH 1/2] add user namespace [try #2]
Posted by Herbert Poetzl on Tue, 12 Sep 2006 10:48:02 GMT
View Forum Message <> Reply to Message

On Mon, Sep 11, 2006 at 10:35:45AM +0200, Cedric Le Goater wrote:
> Kirill Korotaev wrote:
>
> [...]
>
> >> +static struct user_namespace *clone_user_ns(struct user_namespace *old_ns)
> >> +{
> >> + struct user_namespace *ns;
> >> +
> >> + ns = kmalloc(sizeof(struct user_namespace), GFP_KERNEL);
> >> + if (ns) {
> > <<<< can you remake this function please so that normal case would
> > go without indentation, i.e.:
> > if (!ns) {
> > error path;
> > }
> > normal path
>
> right, the error path has been growing too much. Will do.
>
> >> + int n;
> >> + struct user_struct *new_user;
> >> +
> >> + kref_init(&ns->kref);
> >> +
> >> + for(n = 0; n < UIDHASH_SZ; ++n)
> >> + INIT_LIST_HEAD(ns->uidhash_table + n);
> >> +
> >> + /* Insert new root user. */

> >> + ns->root_user = alloc_uid(ns, 0);
> >> + if (!ns->root_user) {
> >> +   kfree(ns);
> >> +   return NULL;
> >> + }
> >> +
> >> + /* Reset current->user with a new one */
> >> + new_user = alloc_uid(ns, current->uid);
> >> + if (!new_user) {
> >> +   kfree(ns);
> >> +   return NULL;
> >> + }
> >> +
> >> + switch_uid(new_user);
> > <<<< I see switch_uid in this function, but you don't change
> > nsproxy->user_ns here...
> > <<<< it is done much later, in sys_unshare()... This looks a bit
> > inconsistent for me...
>
> I would say it is safe but i agree with you that it would be better to
> switch user namespace before than switching user_struct.
>
> > <<<< But I'm unsure whether it can be fixed... :/
>
> he, may be an argument for the clone_ns() syscall ?
>
> >> + return ns;
> >> +}
> >> +
> >> +/*
> >> + * unshare the current process' user namespace.
> >> + */
> >> +int unshare_user_ns(unsigned long unshare_flags,
> >> +   struct user_namespace **new_user)
> >> +{
> >> + if (unshare_flags & CLONE_NEWUSER) {
> >> + if (!capable(CAP_SYS_ADMIN))
> >> +   return -EPERM;
> > <<<< such checks for CAP_SYS_ADMIN mean that we can't use
> > copy_xxx/clone_xxx functions directly
> > <<<< from OpenVZ code, since VE creation is done with dropped
> > capabilities already.

is there a good reason for doing so?
I mean, Linux-VServer for example drops the capabilities
at the end of initialization, right before spawning the
guest init (or running the guest's runlevel scripts)

> OK.
>
> > <<<< (user level tools decide which capabilities should be granted
> > to VE, so CAP_SYS_ADMIN
> > <<<< is not normally granted :))
> > <<<< Can we move capability checks into some more logical place
> > which deals with user, e.g. sys_unshare()?

we also do not give CAP_SYS_ADMIN by default, but I have
to admit, that we add a CAP_CONTEXT (which allows to
do guest related manipulations) to the host admin processes
which in turn controls such things like namespace or
context changes

> hmm, another argument for a new syscall() ?

well, last time I checked syscalls were pretty popular
and I do not see a good reason to complicate things
unnecessary when the new features will require new
userspace tools anyways (just my opinion) ...

best,
Herbert

> C.
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers