Subject: Re: [PATCH 1/6] user namespace : add the framework
Posted by dev on Mon, 16 Jul 2007 14:54:28 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:
> Quoting Andrew Morton (akpm@linux-foundation.org):
>
>>On Mon, 4 Jun 2007 14:40:24 -0500 "Serge E. Hallyn" <serue@us.ibm.com> wrote:
>>
>>
>>>Add the user namespace struct and framework
>>>
>>>Basically, it will allow 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 ?)
>>
>>The whole magical-uid-0-user thing in this patch seem just wrong to
>>me.
>>
>>I'll merge it anyway, mainly because I want to merge _something_ (why oh
>>why do the git-tree guys leave everything to the last minute?) but it strikes
>>me that there's something fundamentally wrong whenever the kernel starts
>>"knowing" about the significance of UIDs in this fashion.
>
>
> $(&(%
>
> I thought I disagreed, but now I'm pretty sure I completely agree.
>
> 'root_user' exists in the kernel right now, but the root_user checks
> which exist (in fork.c and sys.c) shouldn't actually be applied for root
> in a container, since the container may not be trusted.

This rlimit check doesn't help *untrusted* containers, so your logic is wrong here.
Instead, it allows root of the container to operate in any situation.
E.g. consider root user hit the limit. After that you won't be able to login/ssh to fix anything.

NOTE: container root can have no CAP_SYS_RESOURCE and CAP_SYS_ADMIN as it is in OpenVZ.

But in general I'm not against the patch, since in OpenVZ we can replace the check
with another capability we use for VE admin - CAP_VE_SYS_ADMIN.

Kirill

> There is the root_user_keyring stuff - David, is that only special cased
> so that root's keys can be statically allocated?
>
>
>>It worries me.
>
>
> Cedric, you've probably mentioned this before, but what is wrong with
> the following patch? User namespaces still seem to work for me with
> this, but maybe I'm testing wrong. Can you give it a shot?
>
> thanks,
> -serge
>
>>From 743e4f5c15ff9ec4110adc9d06e39a3fb0541512 Mon Sep 17 00:00:00 2001
> From: Serge E. Hallyn <serue@us.ibm.com>
> Date: Mon, 16 Jul 2007 10:26:25 -0400
> Subject: [PATCH 1/1] userns: remove uid0 logic
>
> While 'root_user' is hard_coded in the kernel, as are uid 0
> checks, removing the 'root_user' from a user namespace
> does not appear to break user namespaces.
>
> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
> ---
> include/linux/user_namespace.h | 1 -
> kernel/fork.c | 2 +-
> kernel/sys.c | 2 +-
> kernel/user_namespace.c | 9 ---------
> 4 files changed, 2 insertions(+), 12 deletions(-)
>
> diff --git a/include/linux/user_namespace.h b/include/linux/user_namespace.h
> index bb32057..fcb4f30 100644
> --- a/include/linux/user_namespace.h
> +++ b/include/linux/user_namespace.h
> @@ -12,7 +12,6 @@
> struct user_namespace {
>   struct kref kref;
>   struct list_head uidhash_table[UIDHASH_SZ];
> - struct user_struct *root_user;
> };
>
> extern struct user_namespace init_user_ns;
> diff --git a/kernel/fork.c b/kernel/fork.c
> index 8b6ba70..f5c7f49 100644
> --- a/kernel/fork.c
> +++ b/kernel/fork.c
> @@ -1004,7 +1004,7 @@ static struct task_struct *copy_process(unsigned long clone_flags,

```
>   if (atomic_read(&p->user->processes) >=
>     p->signal->rlim[RLIMIT_NPROC].rlim_cur) {
>    if (!capable(CAP_SYS_ADMIN) && !capable(CAP_SYS_RESOURCE) &&
> -      p->user != current->nsproxy->user_ns->root_user)
> +      p->user != &root_user)
>     goto bad_fork_free;
>   }
>
> diff --git a/kernel/sys.c b/kernel/sys.c
> index e01b5d1..809e416 100644
> --- a/kernel/sys.c
> +++ b/kernel/sys.c
> @@ -1085,7 +1085,7 @@ static int set_user(uid_t new_ruid, int dumpclear)
>
>   if (atomic_read(&new_user->processes) >=
>       current->signal->rlim[RLIMIT_NPROC].rlim_cur &&
> -   new_user != current->nsproxy->user_ns->root_user) {
> +   new_user != &root_user) {
>    free_uid(new_user);
>    return -EAGAIN;
>   }
> diff --git a/kernel/user_namespace.c b/kernel/user_namespace.c
> index 89a27e8..df11a27 100644ldn't actually be applied for root
in a container, since the container may not be trus
> --- a/kernel/user_namespace.c
> +++ b/kernel/user_namespace.c
> @@ -14,7 +14,6 @@ struct user_namespace init_user_ns = {
>   .kref = {
>    .refcount = ATOMIC_INIT(2),
>   },
> - .root_user = &root_user,
>  };
>
>  EXPORT_SYMBOL_GPL(init_user_ns);
> @@ -41,17 +40,9 @@ static struct user_namespace *clone_user_ns(struct user_namespace
*old_ns)
>   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) {
> -  free_uid(ns->root_user);
>    kfree(ns);
>    return NULL;
>   }
```

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