Subject: [PATCH] Fix user struct leakage with locked IPC shem segment
Posted by Pavel Emelianov on Mon, 16 Jul 2007 12:24:12 GMT
View Forum Message <> Reply to Message

When user locks an ipc shmem segmant with SHM_LOCK ctl and the
segment is already locked the shmem_lock() function returns 0.
After this the subsequent code leaks the existing user struct:

```
== ipc/shm.c: sys_shmctl() ==
     ...
     err = shmem_lock(shp->shm_file, 1, user);
     if (!err) {
          shp->shm_perm.mode |= SHM_LOCKED;
          shp->mlock_user = user;
     }
     ...
==
```

Other results of this are:
1. the new shp->mlock_user is not get-ed and will point to freed
   memory when the task dies.
2. the RLIMIT_MEMLOCK is screwed on both user structs.

The exploit looks like this:

```
==
    id = shmget(...);
    setresuid(uid, 0, 0);
    shmctl(id, SHM_LOCK, NULL);
    setresuid(uid + 1, 0, 0);
    shmctl(id, SHM_LOCK, NULL);
==
```

My solution is to return 0 to the userspace and do not change the
segment's user.

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

```
--- ./ipc/shm.c.shlfix 2007-07-06 10:58:57.000000000 +0400
+++ ./ipc/shm.c 2007-07-16 16:12:34.000000000 +0400
@@ -715,7 +715,7 @@ asmlinkage long sys_shmctl (int shmid, i
    struct user_struct * user = current->user;
    if (!is_file_hugepages(shp->shm_file)) {
     err = shmem_lock(shp->shm_file, 1, user);
-    if (!err) {
+    if (!err && !(shp->shm_perm.mode & SHM_LOCKED)){
```

```
 shp->shm_perm.mode |= SHM_LOCKED;
 shp->mlock_user = user;
}
```

Subject: Re: [PATCH] Fix user struct leakage with locked IPC shem segment
Posted by Andrew Morton on Mon, 16 Jul 2007 22:17:38 GMT
View Forum Message <> Reply to Message

On Mon, 16 Jul 2007 16:24:12 +0400
Pavel Emelianov <xemul@openvz.org> wrote:

> When user locks an ipc shmem segmant with SHM_LOCK ctl and the
> segment is already locked the shmem_lock() function returns 0.
> After this the subsequent code leaks the existing user struct:

I'm curious. For the past few months, people@openvz.org have discovered
(and fixed) an ongoing stream of obscure but serious and quite
long-standing bugs.

How are you discovering these bugs?

> == ipc/shm.c: sys_shmctl() ==
>   ...
>   err = shmem_lock(shp->shm_file, 1, user);
>   if (!err) {
>   shp->shm_perm.mode |= SHM_LOCKED;
>   shp->mlock_user = user;
>   }
>   ...
> ==
>
> Other results of this are:
> 1. the new shp->mlock_user is not get-ed and will point to freed
>   memory when the task dies.

That sounds fairly serious - can this lead to memory corruption and crashes?

> 2. the RLIMIT_MEMLOCK is screwed on both user structs.
>
> The exploit looks like this:
>
> ==
>   id = shmget(...);
>   setresuid(uid, 0, 0);
>   shmctl(id, SHM_LOCK, NULL);
>   setresuid(uid + 1, 0, 0);
>   shmctl(id, SHM_LOCK, NULL);

> ==
>
> My solution is to return 0 to the userspace and do not change the
> segment's user.
>
> Signed-off-by: Pavel Emelianov <xemul@openvz.org>
>
> ---
>
> --- ./ipc/shm.c.shlfix 2007-07-06 10:58:57.000000000 +0400
> +++ ./ipc/shm.c 2007-07-16 16:12:34.000000000 +0400
> @@ -715,7 +715,7 @@ asmlinkage long sys_shmctl (int shmid, i
> struct user_struct * user = current->user;
> if (!is_file_hugepages(shp->shm_file)) {
> err = shmem_lock(shp->shm_file, 1, user);
> - if (!err) {
> + if (!err && !(shp->shm_perm.mode & SHM_LOCKED)){
> shp->shm_perm.mode |= SHM_LOCKED;
> shp->mlock_user = user;
> }

Subject: Re: Re: [PATCH] Fix user struct leakage with locked IPC shem segment
Posted by dev on Tue, 17 Jul 2007 09:06:05 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Mon, 16 Jul 2007 16:24:12 +0400
> Pavel Emelianov <xemul@openvz.org> wrote:
>
>
>>When user locks an ipc shmem segmant with SHM_LOCK ctl and the
>>segment is already locked the shmem_lock() function returns 0.
>>After this the subsequent code leaks the existing user struct:
>
>
> I'm curious. For the past few months, people@openvz.org have discovered
> (and fixed) an ongoing stream of obscure but serious and quite
> long-standing bugs.

thanks a lot :@)

> How are you discovering these bugs?

Not sure what to answer :) Just trying to do our best.

This bug was thought over by Pavel for about 3 month after a single
uid leak in container was detected by beancounters' kernel memory accounting...

>>== ipc/shm.c: sys_shmctl() ==
>>   ...
>>   err = shmem_lock(shp->shm_file, 1, user);
>>   if (!err) {
>>   shp->shm_perm.mode |= SHM_LOCKED;
>>   shp->mlock_user = user;
>>   }
>>   ...
>>==
>>
>>Other results of this are:
>>1. the new shp->mlock_user is not get-ed and will point to freed
>>   memory when the task dies.
>
>
> That sounds fairly serious - can this lead to memory corruption and crashes?

Yes it can. According to Pavel when the shmem segment is destroyed it
puts the mlock_user pointer, which can already be stalled.

Kirill

Subject: Re: Re: [PATCH] Fix user struct leakage with locked IPC shem segment
Posted by Andrew Morton on Tue, 17 Jul 2007 09:15:05 GMT
View Forum Message <> Reply to Message

On Tue, 17 Jul 2007 13:07:55 +0400 Kirill Korotaev <dev@sw.ru> wrote:

> Andrew Morton wrote:
> > On Mon, 16 Jul 2007 16:24:12 +0400
> > Pavel Emelianov <xemul@openvz.org> wrote:
> >
> >
> > >When user locks an ipc shmem segmant with SHM_LOCK ctl and the
> > >segment is already locked the shmem_lock() function returns 0.
> > >After this the subsequent code leaks the existing user struct:
> >
> >
> > I'm curious. For the past few months, people@openvz.org have discovered
> > (and fixed) an ongoing stream of obscure but serious and quite
> > long-standing bugs.
>
> thanks a lot :@)
>
> > How are you discovering these bugs?
>

> Not sure what to answer :) Just trying to do our best.

hm, OK, I was visualising some mysterious Russian bugfinding machine or
something.

Don't stop ;)

> This bug was thought over by Pavel for about 3 month after a single
> uid leak in container was detected by beancounters' kernel memory accounting...
>
> >>== ipc/shm.c: sys_shmctl() ==
> >>   ...
> >>   err = shmem_lock(shp->shm_file, 1, user);
> >>   if (!err) {
> >>   shp->shm_perm.mode |= SHM_LOCKED;
> >>   shp->mlock_user = user;
> >>   }
> >>   ...
> >>==
> >>
> >>Other results of this are:
> >>1. the new shp->mlock_user is not get-ed and will point to freed
> >>   memory when the task dies.
> >
> >
> > That sounds fairly serious - can this lead to memory corruption and crashes?
>
> Yes it can. According to Pavel when the shmem segment is destroyed it
> puts the mlock_user pointer, which can already be stalled.

OK, thanks, I'll feed a copy in stable@kernel.org's direction.
