Subject: [RFC PATCH 1/6] IPC/sem: RCU-protect the process semundo list
Posted by Nadia Derbey on Wed, 25 Jun 2008 13:49:11 GMT
View Forum Message <> Reply to Message

PATCH [01/06]

Today, 'current' has an exclusive access to its sem_undo_list (anchored at current->sysvsem.undo_list):

. it is created during a semop() if the SEMUNDO flag is specified for one of the semaphores.
. it can also be created during a copy_process() operation if the CLONE_SYSVSEM flag is specified (in that case the undo_list is created/ copied from 'current' into the target task but that target task is not running yet).
. it is freed during an unshare() or an exit() operation, if the caller (current) is the last task using it.

In order to allow a third party process to read a process' undo list, without a too big performance impact on the existing operations, this patch proposes to make the sem_undo_list structures rcu protected.

They can then be safely accessed by any task inside read critical section, this way:

```
struct sem_undo_list *undo_list;
int ret;
...
rcu_read_lock();
undo_list = rcu_dereference(task->sysvsem.undo_list);
if (undo_list)
        ret = atomic_inc_not_zero(&undo_list->refcnt);
rcu_read_unlock();
...
if (undo_list && ret) {
        /* section where undo_list can be used quietly */
        ...
}
...
```

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

---
include/linux/sem.h | 4 +++-
ipc/sem.c | 22 ++++++++++++++++++----
2 files changed, 21 insertions(+), 5 deletions(-)

```
Index: linux-2.6.26-rc5-mm3/include/linux/sem.h
===================================================================
--- linux-2.6.26-rc5-mm3.orig/include/linux/sem.h 2008-06-24 09:04:21.000000000 +0200
+++ linux-2.6.26-rc5-mm3/include/linux/sem.h 2008-06-24 10:01:50.000000000 +0200
@@ -112,7 +112,8 @@ struct sem_queue {
 };

 /* Each task has a list of undo requests. They are executed automatically
- * when the process exits.
+ * when the last refcnt of sem_undo_list is released (ie when the process
+ * exits in the general case).
  */
 struct sem_undo {
  struct list_head list_proc; /* per-process list: all undos from one process. */
@@ -131,6 +132,7 @@ struct sem_undo_list {
  atomic_t  refcnt;
  spinlock_t  lock;
  struct list_head list_proc;
+ struct rcu_head  rcu;
 };

 struct sysv_sem {
Index: linux-2.6.26-rc5-mm3/ipc/sem.c
===================================================================
--- linux-2.6.26-rc5-mm3.orig/ipc/sem.c 2008-06-24 09:05:03.000000000 +0200
+++ linux-2.6.26-rc5-mm3/ipc/sem.c 2008-06-24 09:37:33.000000000 +0200
@@ -939,6 +939,10 @@ static inline int get_undo_list(struct s
 {
  struct sem_undo_list *undo_list;

+ /*
+  * No need to have a rcu read critical section here: no one but
+  * current is accessing the undo_list.
+  */
  undo_list = current->sysvsem.undo_list;
  if (!undo_list) {
   undo_list = kzalloc(sizeof(*undo_list), GFP_KERNEL);
@@ -948,7 +952,7 @@ static inline int get_undo_list(struct s
   atomic_set(&undo_list->refcnt, 1);
   INIT_LIST_HEAD(&undo_list->list_proc);

-  current->sysvsem.undo_list = undo_list;
+  rcu_assign_pointer(current->sysvsem.undo_list, undo_list);
  }
  *undo_listp = undo_list;
  return 0;
@@ -1268,10 +1272,15 @@ void exit_sem(struct task_struct *tsk)
 {
```

```
  struct sem_undo_list *ulp;

- ulp = tsk->sysvsem.undo_list;
- if (!ulp)
+ rcu_read_lock();
+ ulp = rcu_dereference(tsk->sysvsem.undo_list);
+ if (!ulp) {
+  rcu_read_unlock();
   return;
- tsk->sysvsem.undo_list = NULL;
+ }
+ rcu_read_unlock();
+ rcu_assign_pointer(tsk->sysvsem.undo_list, NULL);
+ synchronize_rcu();

  if (!atomic_dec_and_test(&ulp->refcnt))
   return;
@@ -1349,6 +1358,11 @@ void exit_sem(struct task_struct *tsk)

   call_rcu(&un->rcu, free_un);
  }
+ /*
+  * No need to call synchronize_rcu() here: we come here if the refcnt
+  * is 0 and this has been done into exit_sem after synchronizing. So
+  * nobody else can be referencing to the undo_list.
+  */
  kfree(ulp);
 }


--
```

Subject: Re: [RFC PATCH 1/6] IPC/sem: RCU-protect the process semundo list
Posted by serue on Wed, 25 Jun 2008 20:33:19 GMT
View Forum Message <> Reply to Message

Quoting Nadia.Derbey@bull.net (Nadia.Derbey@bull.net):
> PATCH [01/06]
>
> Today, 'current' has an exclusive access to its sem_undo_list (anchored at
> current->sysvsem.undo_list):
> . it is created during a semop() if the SEMUNDO flag is specified for one
> of the semaphores.

> . it can also be created during a copy_process() operation if the
>   CLONE_SYSVSEM flag is specified (in that case the undo_list is created/
>   copied from 'current' into the target task but that target task is not
>   running yet).
> . it is freed during an unshare() or an exit() operation, if the caller
>   (current) is the last task using it.
>
>
> In order to allow a third party process to read a process' undo list, without
> a too big performance impact on the existing operations, this patch proposes
> to make the sem_undo_list structures rcu protected.
>
> They can then be safely accessed by any task inside read critical section,
> this way:
>
>   struct sem_undo_list *undo_list;
>   int ret;
>   ...
>   rcu_read_lock();
>   undo_list = rcu_dereference(task->sysvsem.undo_list);
>   if (undo_list)
>   ret = atomic_inc_not_zero(&undo_list->refcnt);
>   rcu_read_unlock();
>   ...
>   if (undo_list && ret) {
>   /* section where undo_list can be used quietly */
>   ...
>   }
>   ...
>
> Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
> Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

Acked-by: Serge Hallyn <serue@us.ibm.com>

thanks,
-serge

>
> ---
> include/linux/sem.h | 4 +++-
> ipc/sem.c | 22 ++++++++++++++++++----
> 2 files changed, 21 insertions(+), 5 deletions(-)
>
> Index: linux-2.6.26-rc5-mm3/include/linux/sem.h
> ===
> --- linux-2.6.26-rc5-mm3.orig/include/linux/sem.h 2008-06-24 09:04:21.000000000 +0200
> +++ linux-2.6.26-rc5-mm3/include/linux/sem.h 2008-06-24 10:01:50.000000000 +0200

```
> @@ -112,7 +112,8 @@ struct sem_queue {
>  };
>
>  /* Each task has a list of undo requests. They are executed automatically
> - * when the process exits.
> + * when the last refcnt of sem_undo_list is released (ie when the process
> + * exits in the general case).
>   */
>  struct sem_undo {
>   struct list_head list_proc; /* per-process list: all undos from one process. */
> @@ -131,6 +132,7 @@ struct sem_undo_list {
>   atomic_t  refcnt;
>   spinlock_t  lock;
>   struct list_head list_proc;
> + struct rcu_head  rcu;
>  };
>
>  struct sysv_sem {
> Index: linux-2.6.26-rc5-mm3/ipc/sem.c
> ===================================================================
> --- linux-2.6.26-rc5-mm3.orig/ipc/sem.c 2008-06-24 09:05:03.000000000 +0200
> +++ linux-2.6.26-rc5-mm3/ipc/sem.c 2008-06-24 09:37:33.000000000 +0200
> @@ -939,6 +939,10 @@ static inline int get_undo_list(struct s
>  {
>   struct sem_undo_list *undo_list;
>
> + /*
> +  * No need to have a rcu read critical section here: no one but
> +  * current is accessing the undo_list.
> +  */
>   undo_list = current->sysvsem.undo_list;
>   if (!undo_list) {
>    undo_list = kzalloc(sizeof(*undo_list), GFP_KERNEL);
> @@ -948,7 +952,7 @@ static inline int get_undo_list(struct s
>    atomic_set(&undo_list->refcnt, 1);
>    INIT_LIST_HEAD(&undo_list->list_proc);
>
> -  current->sysvsem.undo_list = undo_list;
> +  rcu_assign_pointer(current->sysvsem.undo_list, undo_list);
>   }
>   *undo_listp = undo_list;
>   return 0;
> @@ -1268,10 +1272,15 @@ void exit_sem(struct task_struct *tsk)
>  {
>   struct sem_undo_list *ulp;
>
> - ulp = tsk->sysvsem.undo_list;
> - if (!ulp)
```

```
> + rcu_read_lock();
> + ulp = rcu_dereference(tsk->sysvsem.undo_list);
> + if (!ulp) {
> +  rcu_read_unlock();
>    return;
> - tsk->sysvsem.undo_list = NULL;
> + }
> + rcu_read_unlock();
> + rcu_assign_pointer(tsk->sysvsem.undo_list, NULL);
> + synchronize_rcu();
>
>   if (!atomic_dec_and_test(&ulp->refcnt))
>    return;
> @@ -1349,6 +1358,11 @@ void exit_sem(struct task_struct *tsk)
>
>    call_rcu(&un->rcu, free_un);
>   }
> + /*
> +  * No need to call synchronize_rcu() here: we come here if the refcnt
> +  * is 0 and this has been done into exit_sem after synchronizing. So
> +  * nobody else can be referencing to the undo_list.
> +  */
>   kfree(ulp);
>  }
>
>
> --
```