
Subject: [PATCH 2.6.24-rc8-mm1 10/15] (RFC) IPC: new IPC_SETID command to modify an ID

Posted by [Pierre Peiffer](#) on Tue, 29 Jan 2008 16:02:39 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: Pierre Peiffer <pierre.peiffer@bull.net>

This patch adds a new IPC_SETID command to the System V IPCs set of commands, which allows to change the ID of an existing IPC.

This command can be used through the semctl/shmctl/msgctl API, with the new ID passed as the third argument for msgctl and shmctl (instead of a pointer) and through the fourth argument for semctl.

To be successful, the following rules must be respected:

- the IPC exists
- the user must be allowed to change the IPC attributes regarding the IPC permissions.
- the new ID must satisfy the ID computation rule.
- the entry (in the kernel internal table of IPCs) corresponding to the new ID must be free.

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

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

```
include/linux/ipc.h      | 9 ++++++----
ipc/compat.c             | 3 +++
ipc/msg.c                | 27 ++++++++++++++++++++++
ipc/sem.c                | 27 ++++++++++++++++++++++
ipc/shm.c                | 27 ++++++++++++++++++++++
security/selinux/hooks.c | 3 +++
6 files changed, 89 insertions(+), 7 deletions(-)
```

Index: b/include/linux/ipc.h

```
=====
--- a/include/linux/ipc.h
+++ b/include/linux/ipc.h
@@ -35,10 +35,11 @@ struct ipc_perm
 * Control commands used with semctl, msgctl and shmctl
 * see also specific commands in sem.h, msg.h and shm.h
 */
-#define IPC_RMID 0 /* remove resource */
-#define IPC_SET 1 /* set ipc_perm options */
-#define IPC_STAT 2 /* get ipc_perm options */
-#define IPC_INFO 3 /* see ipc.h */
+#define IPC_RMID 0 /* remove resource */
+#define IPC_SET 1 /* set ipc_perm options */
```

```

#define IPC_STAT 2 /* get ipc_perm options */
#define IPC_INFO 3 /* see ipcs */
#define IPC_SETID 4 /* set ipc ID */

```

```

/*

```

```

 * Version flags for semctl, msgctl, and shmctl commands

```

```

Index: b/ipc/compat.c

```

```

--- a/ipc/compat.c

```

```

+++ b/ipc/compat.c

```

```

@@ -253,6 +253,7 @@ long compat_sys_semctl(int first, int se
 switch (third & (~IPC_64)) {

```

```

 case IPC_INFO:

```

```

 case IPC_RMID:

```

```

+ case IPC_SETID:

```

```

 case SEM_INFO:

```

```

 case GETVAL:

```

```

 case GETPID:

```

```

@@ -425,6 +426,7 @@ long compat_sys_msgctl(int first, int se
 switch (second & (~IPC_64)) {

```

```

 case IPC_INFO:

```

```

 case IPC_RMID:

```

```

+ case IPC_SETID:

```

```

 case MSG_INFO:

```

```

 err = sys_msgctl(first, second, uptr);
 break;

```

```

@@ -597,6 +599,7 @@ long compat_sys_shmctl(int first, int se

```

```

 switch (second & (~IPC_64)) {

```

```

 case IPC_RMID:

```

```

+ case IPC_SETID:

```

```

 case SHM_LOCK:

```

```

 case SHM_UNLOCK:

```

```

 err = sys_shmctl(first, second, uptr);

```

```

Index: b/ipc/msg.c

```

```

--- a/ipc/msg.c

```

```

+++ b/ipc/msg.c

```

```

@@ -329,7 +329,8 @@ retry:

```

```

 msg_unlock(msg);

```

```

 up_write(&msg_ids(ns).rw_mutex);

```

```

- /* ipc_chid may return -EAGAIN in case of memory requirement */

```

```

+ /* msg_chid_nolock may return -EAGAIN if there is no more free idr

```

```

+ entry, just go and retry by filling again de idr cache */

```

```

 if (err == -EAGAIN)

```

```

 goto retry;

```

```

@@ -465,6 +466,9 @@ static int msgctl_down(struct ipc_namesp
    */
    ss_wakeup(&msq->q_senders, 0);
    break;
+ case IPC_SETID:
+ err = msg_chid_nolock(ns, msq, (int)(long)buf);
+ break;
    default:
        err = -EINVAL;
    }
@@ -475,6 +479,24 @@ out_up:
    return err;
}

+static int msgctl_setid(struct ipc_namespace *ns, int msqid, int cmd,
+ struct msqid_ds __user *buf, int version)
+{
+ int err;
+retry:
+ err = idr_pre_get(&msg_ids(ns).ipcs_idr, GFP_KERNEL);
+ if (!err)
+ return -ENOMEM;
+
+ err = msgctl_down(ns, msqid, cmd, buf, version);
+
+ /* msgctl_down may return -EAGAIN if there is no more free idr
+ entry, just go and retry by filling again de idr cache */
+ if (err == -EAGAIN)
+ goto retry;
+ return err;
+}
+
+asmlinkage long sys_msgctl(int msqid, int cmd, struct msqid_ds __user *buf)
+{
+ struct msg_queue *msq;
@@ -575,6 +597,9 @@ asmlinkage long sys_msgctl(int msqid, in
    case IPC_RMID:
        err = msgctl_down(ns, msqid, cmd, buf, version);
        return err;
+ case IPC_SETID:
+ err = msgctl_setid(ns, msqid, cmd, buf, version);
+ return err;
    default:
        return -EINVAL;
    }
Index: b/ipc/sem.c
=====
--- a/ipc/sem.c

```

```

+++ b/ipc/sem.c
@@ -608,7 +608,8 @@ retry:
    sem_unlock(sma);
    up_write(&sem_ids(ns).rw_mutex);

- /* ipc_chid may return -EAGAIN in case of memory requirement */
+ /* sem_chid_nolock may return -EAGAIN if there is no more free idr
+  entry, just go and retry by filling again de idr cache */
    if (err == -EAGAIN)
        goto retry;

@@ -935,6 +936,9 @@ static int semctl_down(struct ipc_namesp
    ipc_update_perm(&semid64.sem_perm, ipcp);
    sma->sem_ctime = get_seconds();
    break;
+ case IPC_SETID:
+ err = sem_chid_nolock(ns, sma, (int)arg.val);
+ break;
    default:
        err = -EINVAL;
    }
@@ -946,6 +950,24 @@ out_up:
    return err;
}

+static int semctl_setid(struct ipc_namespace *ns, int semid,
+ int cmd, int version, union semun arg)
+{
+ int err;
+retry:
+ err = idr_pre_get(&sem_ids(ns).ipcs_idr, GFP_KERNEL);
+ if (!err)
+ return -ENOMEM;
+
+ err = semctl_down(ns, semid, cmd, version, arg);
+
+ /* semctl_down may return -EAGAIN if there is no more free idr
+  entry, just go and retry by filling again de idr cache */
+ if (err == -EAGAIN)
+ goto retry;
+ return err;
+}
+
+asmlinkage long sys_semctl (int semid, int semnum, int cmd, union semun arg)
+{
+ int err = -EINVAL;
@@ -978,6 +1000,9 @@ asmlinkage long sys_semctl (int semid, i
    case IPC_SET:

```

```

    err = semctl_down(ns, semid, cmd, version, arg);
    return err;
+ case IPC_SETID:
+ err = semctl_setid(ns, semid, cmd, version, arg);
+ return err;
    default:
        return -EINVAL;
    }

```

Index: b/ipc/shm.c

=====

```

--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -202,7 +202,8 @@ retry:
    shm_unlock(shp);
    up_write(&shm_ids(ns).rw_mutex);

- /* ipc_chid may return -EAGAIN in case of memory requirement */
+ /* shm_chid_nolock may return -EAGAIN if there is no more free idr
+  entry, just go and retry by filling again de idr cache */
    if (err == -EAGAIN)
        goto retry;

@@ -679,6 +680,9 @@ static int shmctl_down(struct ipc_namesp
    ipc_update_perm(&shm64.shm_perm, ipc);
    shp->shm_ctim = get_seconds();
    break;
+ case IPC_SETID:
+ err = shm_chid_nolock(ns, shp, (int)(long)buf);
+ break;
    default:
        err = -EINVAL;
    }
@@ -689,6 +693,24 @@ out_up:
    return err;
}

```

```

+static int shmctl_setid(struct ipc_namespace *ns, int shm64, int cmd,
+ struct shm64_ds __user *buf, int version)
+{
+ int err;
+retry:
+ err = idr_pre_get(&shm_ids(ns).ipcs_idr, GFP_KERNEL);
+ if (!err)
+ return -ENOMEM;
+
+ err = shmctl_down(ns, shm64, cmd, buf, version);
+
+ /* shmctl_down may return -EAGAIN if there is no more free idr

```

```

+ entry, just go and retry by filling again de idr cache */
+ if (err == -EAGAIN)
+ goto retry;
+ return err;
+}
+
asmlinkage long sys_shmctl(int shmid, int cmd, struct shmctl_ds __user *buf)
{
    struct shmctl_kernel *shp;
@@ -850,6 +872,9 @@ asmlinkage long sys_shmctl(int shmid, in
    case IPC_SET:
        err = shmctl_down(ns, shmid, cmd, buf, version);
        return err;
+ case IPC_SETID:
+ err = shmctl_setid(ns, shmid, cmd, buf, version);
+ return err;
    default:
        return -EINVAL;
}

```

Index: b/security/selinux/hooks.c

```

=====
--- a/security/selinux/hooks.c
+++ b/security/selinux/hooks.c
@@ -4651,6 +4651,7 @@ static int selinux_msg_queue_msgctl(stru
    perms = MSGQ__GETATTR | MSGQ__ASSOCIATE;
    break;
    case IPC_SET:
+ case IPC_SETID:
        perms = MSGQ__SETATTR;
        break;
    case IPC_RMID:
@@ -4799,6 +4800,7 @@ static int selinux_shm_shmctl(struct shm
    perms = SHM__GETATTR | SHM__ASSOCIATE;
    break;
    case IPC_SET:
+ case IPC_SETID:
        perms = SHM__SETATTR;
        break;
    case SHM_LOCK:
@@ -4910,6 +4912,7 @@ static int selinux_sem_semctl(struct sem
    perms = SEM__DESTROY;
    break;
    case IPC_SET:
+ case IPC_SETID:
        perms = SEM__SETATTR;
        break;
    case IPC_STAT:

```

--

Pierre Peiffer

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
