
Subject: [RFC][PATCH 2.6.22-rc5] System V IPC: new IPC_SETID command to modify an ID

Posted by [Pierre Peiffer](#) on Mon, 18 Jun 2007 14:59:23 GMT

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

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>

```
include/linux/ipc.h      |  9 +++----
ipc/msg.c                | 16 ++++++++
ipc/sem.c                | 15 ++++++++
ipc/shm.c                | 36 ++++++++
ipc/util.c               | 60 ++++++++
ipc/util.h               |  1
security/selinux/hooks.c |  3 ++
7 files changed, 136 insertions(+), 4 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 ipc.h */
```

```

#define IPC_SETID 4    /* set ipc ID */

/*
 * Version flags for semctl, msgctl, and shmctl commands
Index: b/ipc/msg.c
=====
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -491,6 +491,7 @@ asmlinkage long sys_msgctl(int msqid, in
    if (copy_msqid_from_user(&setbuf, buf, version))
        return -EFAULT;
    break;
+ case IPC_SETID:
    case IPC_RMID:
        break;
    default:
@@ -553,6 +554,21 @@ asmlinkage long sys_msgctl(int msqid, in
    msg_unlock(msq);
    break;
}
+ case IPC_SETID:
+ {
+     int nid = (int)buf;
+
+     err = ipc_mvid(&msg_ids(ns), msq->q_id,
+         nid, ns->msg_ctlmni);
+
+     if (err)
+         goto out_unlock_up;
+
+     msq->q_id = nid;
+     msq->q_ctime = get_seconds();
+     msg_unlock(msq);
+     break;
+ }
    case IPC_RMID:
        freeque(ns, msq, msqid);
        break;
Index: b/ipc/sem.c
=====
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -908,6 +908,20 @@ static int semctl_down(struct ipc_namesp
    sem_unlock(sma);
    err = 0;
    break;
+ case IPC_SETID:
+ {

```

```

+ int nid = (int)arg.val;
+ err = ipc_mvid(&sem_ids(ns), semid,
+     nid, ns->sc_semmni);
+
+ if (err)
+     goto out_unlock;
+
+ sma->sem_id = nid;
+ sma->sem_ctime = get_seconds();
+ sem_unlock(sma);
+ break;
+ }
+ default:
+     sem_unlock(sma);
+     err = -EINVAL;
@@ -950,6 +964,7 @@ asmlinkage long sys_semctl (int semid, i
+     return err;
+     case IPC_RMID:
+     case IPC_SET:
+ case IPC_SETID:
+     mutex_lock(&sem_ids(ns).mutex);
+     err = semctl_down(ns, semid, semnum, cmd, version, arg);
+     mutex_unlock(&sem_ids(ns).mutex);

```

Index: b/ipc/util.c

```

=====
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -327,6 +327,66 @@ found:
 }

/**
+ * ipc_mvid - move an IPC identifier
+ * @ids: IPC identifier set
+ * @oldid: ID of the IPC permission set to move
+ * @newid: new ID of the IPC permission set to move
+ * @size: new size limit for the id array
+ *
+ * Move an entry in the IPC arrays from the 'oldid' place to the
+ * 'newid' place. The seq number of the entry is updated to match the
+ * 'newid' value.
+ *
+ * Called with the list lock and ipc_ids.mutex held.
+ */
+
+int ipc_mvid(struct ipc_ids *ids, int oldid, int newid, int size)
+{
+ struct kern_ipc_perm *p;
+ int old_lid = oldid % SEQ_MULTIPLIER;

```

```

+ int new_lid = newid % SEQ_MULTIPLIER;
+
+ if ((new_lid >= size) ||
+     newid != (new_lid + (newid/SEQ_MULTIPLIER)*SEQ_MULTIPLIER))
+ return -ERANGE;
+
+ size = grow_ary(ids,size);
+
+ BUG_ON(old_lid >= ids->entries->size);
+
+ p = ids->entries->p[old_lid];
+
+ if (!p)
+ return -ENXIO;
+
+ /*
+
+  * but not the seq number.
+  */
+ if (new_lid != old_lid) {
+
+ if (ids->entries->p[new_lid])
+ return -EBUSY;
+
+ ids->entries->p[new_lid] = p;
+
+ ids->entries->p[old_lid] = NULL;
+
+ if (new_lid > ids->max_id)
+ ids->max_id = new_lid;
+ if (old_lid == ids->max_id) {
+ do {
+ --old_lid;
+ } while (ids->entries->p[old_lid] == NULL);
+ ids->max_id = old_lid;
+ }
+ }
+
+ p->seq = newid/SEQ_MULTIPLIER;
+ return 0;
+}
+
+/**
+ * ipc_rmid - remove an IPC identifier
+ * @ids: identifier set
+ * @id: Identifier to remove
+
+Index: b/ipc/util.h
+=====

```

```

--- a/ipc/util.h
+++ b/ipc/util.h
@@ -63,6 +63,7 @@ int ipc_findkey(struct ipc_ids* ids, key
    int ipc_addid(struct ipc_ids* ids, struct kern_ipc_perm* new, int size);

    /* must be called with both locks acquired. */
+int ipc_mvid(struct ipc_ids *ids, int oldid, int newid, int size);
    struct kern_ipc_perm* ipc_rmid(struct ipc_ids* ids, int id);

    int ipcperms (struct kern_ipc_perm *ipcp, short flg);
Index: b/security/selinux/hooks.c
=====
--- a/security/selinux/hooks.c
+++ b/security/selinux/hooks.c
@@ -4152,6 +4152,7 @@ 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:
@@ -4300,6 +4301,7 @@ 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:
@@ -4411,6 +4413,7 @@ static int selinux_sem_semctl(struct sem
    perms = SEM__DESTROY;
    break;
    case IPC_SET:
+ case IPC_SETID:
    perms = SEM__SETATTR;
    break;
    case IPC_STAT:
Index: b/ipc/shm.c
=====
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -809,6 +809,42 @@ asmlinkage long sys_shmctl (int shmid, i
    break;
}

+ case IPC_SETID:
+ {

```

```

+ int nid = (int)buf;
+ mutex_lock(&shm_ids(ns).mutex);
+ shp = shm_lock(ns, shmids);
+ err = -EINVAL;
+ if(shp == NULL)
+ goto out_up;
+ err = shm_checkid(ns, shp,shmids);
+ if(err)
+ goto out_unlock_up;
+ err = audit_ipc_obj(&(shp->shm_perm));
+ if (err)
+ goto out_unlock_up;
+
+ err = -EPERM;
+ if (current->euid != shp->shm_perm.uid &&
+     current->euid != shp->shm_perm.cuid &&
+     !capable(CAP_SYS_ADMIN))
+ goto out_unlock_up;
+
+ err = security_shm_shmctl(shp, cmd);
+ if (err)
+ goto out_unlock_up;
+
+ err = ipc_mvid(&shm_ids(ns), shp->id,
+               nid, ns->shm_ctlmni);
+
+ if (err)
+ goto out_unlock_up;
+
+ shp->id = nid;
+ shp->shm_ctim = get_seconds();
+ break;
+ }
+
+ default:
+   err = -EINVAL;
+   goto out;

```

--

Pierre Peiffer
