Subject: [RFC][PATCH 2/2] System V IPC: new IPC_SETID command to modify an ID

Posted by Pierre Peiffer on Fri, 28 Sep 2007 14:58:47 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>

```
 include/linux/ipc.h      |    9 ++++----
 ipc/msg.c                |   31 ++++++++++++++++++-------
 ipc/sem.c                |   31 ++++++++++++++++++-------
 ipc/shm.c                |   55 ++++++++++++++++++++++++++++++++++++++--------
 security/selinux/hooks.c |    3 +++
 5 files changed, 100 insertions(+), 29 deletions(-)

diff --git a/include/linux/ipc.h b/include/linux/ipc.h
index 3fd3ddd..f1edef5 100644
--- 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 ipcs */
+#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
diff --git a/ipc/msg.c b/ipc/msg.c
index d9d4093..9671156 100644
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -81,6 +81,8 @@ static struct ipc_ids init_msg_ids;
 #define msg_buildid(ns, id, seq) \
  ipc_buildid(&msg_ids(ns), id, seq)

+static long msg_chid_nolock(struct ipc_namespace *ns, struct msg_queue *msq,
+       int newid);
 static void freeque (struct ipc_namespace *ns, struct msg_queue *msq, int id);
 static int newque (struct ipc_namespace *ns, key_t key, int msgflg);
 #ifdef CONFIG_PROC_FS
@@ -382,6 +384,21 @@ copy_msqid_from_user(struct msq_setbuf *out, void __user *buf, int
version)
  }
 }

+static long msg_chid_nolock(struct ipc_namespace *ns, struct msg_queue *msq,
+       int newid)
+{
+ long err;
+ err = ipc_mvid(&msg_ids(ns), msq->q_id,
+         newid, ns->msg_ctlmni);
+
+ if (err)
+  return err;
+
+ msq->q_id = newid;
+ msq->q_ctime = get_seconds();
+ return 0;
+}
+
 long msg_mvid(struct ipc_namespace *ns, int id, int newid)
 {
  long err;
@@ -398,14 +415,7 @@ long msg_mvid(struct ipc_namespace *ns, int id, int newid)
  if (err)
   goto out_unlock_up;

- err = ipc_mvid(&msg_ids(ns), id,
-         newid, ns->msg_ctlmni);
-
- if (err)
```

```
-  goto out_unlock_up;
-
- msq->q_id = newid;
- msq->q_ctime = get_seconds();
+ err = msg_chid_nolock(ns, msq, newid);

 out_unlock_up:
  msg_unlock(msq);
@@ -521,6 +531,7 @@ asmlinkage long sys_msgctl(int msqid, int cmd, struct msqid_ds __user
*buf)
   if (copy_msqid_from_user(&setbuf, buf, version))
    return -EFAULT;
   break;
+ case IPC_SETID:
  case IPC_RMID:
   break;
  default:
@@ -583,6 +594,10 @@ asmlinkage long sys_msgctl(int msqid, int cmd, struct msqid_ds __user
*buf)
   msg_unlock(msq);
   break;
  }
+ case IPC_SETID:
+  err = msg_chid_nolock(ns, msq, (int)buf);
+  msg_unlock(msq);
+  break;
  case IPC_RMID:
   freeque(ns, msq, msqid);
   break;
diff --git a/ipc/sem.c b/ipc/sem.c
index 606f2e9..b78b433 100644
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -98,6 +98,8 @@

 static struct ipc_ids init_sem_ids;

+static long sem_chid_nolock(struct ipc_namespace *ns, struct sem_array *sma,
+       int newid);
 static int newary(struct ipc_namespace *, key_t, int, int);
 static void freeary(struct ipc_namespace *ns, struct sem_array *sma, int id);
 #ifdef CONFIG_PROC_FS
@@ -906,6 +908,10 @@ static int semctl_down(struct ipc_namespace *ns, int semid, int
semnum,
   sem_unlock(sma);
   err = 0;
   break;
+ case IPC_SETID:
```

```
+  err = sem_chid_nolock(ns, sma, (int)arg.val);
+  sem_unlock(sma);
+  break;
   default:
    sem_unlock(sma);
    err = -EINVAL;
@@ -918,6 +924,21 @@ out_unlock:
   return err;
  }

+static long sem_chid_nolock(struct ipc_namespace *ns, struct sem_array *sma,
+       int newid)
+{
+ long err;
+ err = ipc_mvid(&sem_ids(ns), sma->sem_id,
+         newid, ns->sc_semmni);
+
+ if (err)
+  return err;
+
+ sma->sem_id = newid;
+ sma->sem_ctime = get_seconds();
+ return 0;
+}
+
  long sem_mvid(struct ipc_namespace *ns, int id, int newid)
  {
   long err;
@@ -934,14 +955,7 @@ long sem_mvid(struct ipc_namespace *ns, int id, int newid)
   if (err)
    goto out_unlock_up;

- err = ipc_mvid(&sem_ids(ns), id,
-         newid, ns->sc_semmni);
-
- if (err)
-  goto out_unlock_up;
-
- sma->sem_id = newid;
- sma->sem_ctime = get_seconds();
+ err = sem_chid_nolock(ns, sma, newid);

  out_unlock_up:
   sem_unlock(sma);
@@ -980,6 +994,7 @@ asmlinkage long sys_semctl (int semid, int semnum, int cmd, union
semun arg)
    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);
diff --git a/ipc/shm.c b/ipc/shm.c
index 5f4bca6..7d4b27d 100644
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -70,6 +70,8 @@ static struct ipc_ids init_shm_ids;

 static int newseg (struct ipc_namespace *ns, key_t key,
   int shmflg, size_t size);
+static long shm_chid_nolock(struct ipc_namespace *ns, struct shmid_kernel *shp,
+       int newid);
 static void shm_open(struct vm_area_struct *vma);
 static void shm_close(struct vm_area_struct *vma);
 static void shm_destroy (struct ipc_namespace *ns, struct shmid_kernel *shp);
@@ -156,6 +158,21 @@ static inline int shm_addid(struct ipc_namespace *ns, struct
shmid_kernel *shp)
  return ipc_addid(&shm_ids(ns), &shp->shm_perm, ns->shm_ctlmni);
 }

+static long shm_chid_nolock(struct ipc_namespace *ns, struct shmid_kernel *shp,
+       int newid)
+{
+ long err;
+ err = ipc_mvid(&shm_ids(ns), shp->id,
+         newid, ns->shm_ctlmni);
+
+ if (err)
+  return err;
+
+ shp->id = newid;
+ shp->shm_ctim = get_seconds();
+ return 0;
+}
+
 long shm_mvid(struct ipc_namespace *ns, int id, int newid)
 {
  long err;
@@ -172,14 +189,7 @@ long shm_mvid(struct ipc_namespace *ns, int id, int newid)
  if (err)
   goto out_unlock_up;

- err = ipc_mvid(&shm_ids(ns), id,
-         newid, ns->shm_ctlmni);
-
```

```
- if (err)
-  goto out_unlock_up;
-
- shp->id = newid;
- shp->shm_ctim = get_seconds();
+ err = shm_chid_nolock(ns, shp, newid);

 out_unlock_up:
  shm_unlock(shp);
@@ -839,12 +849,39 @@ asmlinkage long sys_shmctl (int shmid, int cmd, struct shmid_ds
__user *buf)
   break;
  }

+ case IPC_SETID:
+ {
+  mutex_lock(&shm_ids(ns).mutex);
+  shp = shm_lock(ns, shmid);
+  err = -EINVAL;
+  if (shp == NULL)
+   goto out_up;
+  err = shm_checkid(ns, shp, shmid);
+  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 = shm_chid_nolock(ns, shp, (int)buf);
+  break;
+ }
+
  default:
   err = -EINVAL;
   goto out;
  }

- err = 0;
```

```
 out_unlock_up:
  shm_unlock(shp);
 out_up:
diff --git a/security/selinux/hooks.c b/security/selinux/hooks.c
index 0753b20..b4a2899 100644
--- a/security/selinux/hooks.c
+++ b/security/selinux/hooks.c
@@ -4163,6 +4163,7 @@ static int selinux_msg_queue_msgctl(struct msg_queue *msq, int cmd)
    perms = MSGQ__GETATTR | MSGQ__ASSOCIATE;
    break;
   case IPC_SET:
+ case IPC_SETID:
    perms = MSGQ__SETATTR;
    break;
   case IPC_RMID:
@@ -4311,6 +4312,7 @@ static int selinux_shm_shmctl(struct shmid_kernel *shp, int cmd)
    perms = SHM__GETATTR | SHM__ASSOCIATE;
    break;
   case IPC_SET:
+ case IPC_SETID:
    perms = SHM__SETATTR;
    break;
   case SHM_LOCK:
@@ -4422,6 +4424,7 @@ static int selinux_sem_semctl(struct sem_array *sma, int cmd)
    perms = SEM__DESTROY;
    break;
   case IPC_SET:
+ case IPC_SETID:
    perms = SEM__SETATTR;
    break;
   case IPC_STAT:
```

--
Pierre

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