Subject: [PATCH 2.6.24-rc8-mm1 11/15] (RFC) IPC: new IPC_SETALL command to modify all settings
Posted by Pierre Peiffer on Tue, 29 Jan 2008 16:02:40 GMT
View Forum Message <> Reply to Message

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

This patch adds a new IPC_SETALL command to the System V IPCs set of commands, which allows to change all the settings of an IPC

It works exactly the same way as the IPC_SET command, except that it additionally changes all the times and the pids values

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
Acked-by: Serge Hallyn <serue@us.ibm.com>

```
 include/linux/ipc.h      |    1 +
 ipc/compat.c             |    3 +++
 ipc/msg.c                |   15 +++++++++++++--
 ipc/sem.c                |   10 +++++++++-
 ipc/shm.c                |   13 +++++++++++++-
 ipc/util.c               |    7 +++++++-
 security/selinux/hooks.c |    3 +++
 7 files changed, 47 insertions(+), 5 deletions(-)

Index: b/include/linux/ipc.h
===================================================================
--- a/include/linux/ipc.h
+++ b/include/linux/ipc.h
@@ -40,6 +40,7 @@ struct ipc_perm
 #define IPC_STAT   2     /* get ipc_perm options */
 #define IPC_INFO   3     /* see ipcs */
 #define IPC_SETID  4     /* set ipc ID */
+#define IPC_SETALL 5     /* set all parameters */

 /*
  * Version flags for semctl, msgctl, and shmctl commands
Index: b/ipc/compat.c
===================================================================
--- a/ipc/compat.c
+++ b/ipc/compat.c
@@ -282,6 +282,7 @@ long compat_sys_semctl(int first, int se
    err = -EFAULT;
   break;

+ case IPC_SETALL:
  case IPC_SET:
```

```
   if (version == IPC_64) {
    err = get_compat_semid64_ds(&s64, compat_ptr(pad));
@@ -431,6 +432,7 @@ long compat_sys_msgctl(int first, int se
   err = sys_msgctl(first, second, uptr);
   break;

+ case IPC_SETALL:
  case IPC_SET:
   if (version == IPC_64) {
    err = get_compat_msqid64(&m64, uptr);
@@ -621,6 +623,7 @@ long compat_sys_shmctl(int first, int se
   break;


+ case IPC_SETALL:
  case IPC_SET:
   if (version == IPC_64) {
    err = get_compat_shmid64_ds(&s64, uptr);
Index: b/ipc/msg.c
===================================================================
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -426,7 +426,7 @@ static int msgctl_down(struct ipc_namesp
  struct msg_queue *msq;
  int err;

- if (cmd == IPC_SET) {
+ if (cmd == IPC_SET || cmd == IPC_SETALL) {
   if (copy_msqid_from_user(&msqid64, buf, version))
    return -EFAULT;
  }
@@ -447,6 +447,7 @@ static int msgctl_down(struct ipc_namesp
   freeque(ns, ipcp);
   goto out_up;
  case IPC_SET:
+ case IPC_SETALL:
   if (msqid64.msg_qbytes > ns->msg_ctlmnb &&
       !capable(CAP_SYS_RESOURCE)) {
    err = -EPERM;
@@ -456,7 +457,14 @@ static int msgctl_down(struct ipc_namesp
   msq->q_qbytes = msqid64.msg_qbytes;

   ipc_update_perm(&msqid64.msg_perm, ipcp);
-  msq->q_ctime = get_seconds();
+  if (cmd == IPC_SETALL) {
+   msq->q_stime = msqid64.msg_stime;
+   msq->q_rtime = msqid64.msg_rtime;
+   msq->q_ctime = msqid64.msg_ctime;
```

```
+   msq->q_lspid = msqid64.msg_lspid;
+   msq->q_lrpid = msqid64.msg_lrpid;
+  } else
+   msq->q_ctime = get_seconds();
    /* sleeping receivers might be excluded by
     * stricter permissions.
     */
@@ -507,6 +515,8 @@ asmlinkage long sys_msgctl(int msqid, in
    return -EINVAL;

   version = ipc_parse_version(&cmd);
+ if (version < 0)
+  return -EINVAL;
   ns = current->nsproxy->ipc_ns;

   switch (cmd) {
@@ -594,6 +604,7 @@ asmlinkage long sys_msgctl(int msqid, in
    return success_return;
   }
   case IPC_SET:
+ case IPC_SETALL:
   case IPC_RMID:
    err = msgctl_down(ns, msqid, cmd, buf, version);
    return err;
Index: b/ipc/sem.c
===================================================================
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -913,7 +913,7 @@ static int semctl_down(struct ipc_namesp
   struct semid64_ds semid64;
   struct kern_ipc_perm *ipcp;

- if(cmd == IPC_SET) {
+ if (cmd == IPC_SET || cmd == IPC_SETALL) {
    if (copy_semid_from_user(&semid64, arg.buf, version))
     return -EFAULT;
   }
@@ -936,6 +936,11 @@ static int semctl_down(struct ipc_namesp
    ipc_update_perm(&semid64.sem_perm, ipcp);
    sma->sem_ctime = get_seconds();
    break;
+ case IPC_SETALL:
+  ipc_update_perm(&semid64.sem_perm, ipcp);
+  sma->sem_ctime = semid64.sem_ctime;
+  sma->sem_otime = semid64.sem_otime;
+  break;
   case IPC_SETID:
    err = sem_chid_nolock(ns, sma, (int)arg.val);
```

```
   break;
@@ -978,6 +983,8 @@ asmlinkage long sys_semctl (int semid, i
   return -EINVAL;

  version = ipc_parse_version(&cmd);
+ if (version < 0)
+  return -EINVAL;
  ns = current->nsproxy->ipc_ns;

  switch(cmd) {
@@ -998,6 +1005,7 @@ asmlinkage long sys_semctl (int semid, i
   return err;
  case IPC_RMID:
  case IPC_SET:
+ case IPC_SETALL:
   err = semctl_down(ns, semid, cmd, version, arg);
   return err;
  case IPC_SETID:
Index: b/ipc/shm.c
===================================================================
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -658,7 +658,7 @@ static int shmctl_down(struct ipc_namesp
  struct shmid_kernel *shp;
  int err;

- if (cmd == IPC_SET) {
+ if (cmd == IPC_SET || cmd == IPC_SETALL) {
   if (copy_shmid_from_user(&shmid64, buf, version))
    return -EFAULT;
  }
@@ -680,6 +680,14 @@ static int shmctl_down(struct ipc_namesp
   ipc_update_perm(&shmid64.shm_perm, ipcp);
   shp->shm_ctim = get_seconds();
   break;
+ case IPC_SETALL:
+  ipc_update_perm(&shmid64.shm_perm, ipcp);
+  shp->shm_atim = shmid64.shm_atime;
+  shp->shm_dtim = shmid64.shm_dtime;
+  shp->shm_ctim = shmid64.shm_ctime;
+  shp->shm_cprid = shmid64.shm_cpid;
+  shp->shm_lprid = shmid64.shm_lpid;
+  break;
  case IPC_SETID:
   err = shm_chid_nolock(ns, shp, (int)(long)buf);
   break;
@@ -723,6 +731,8 @@ asmlinkage long sys_shmctl(int shmid, in
  }
```

```
  version = ipc_parse_version(&cmd);
+ if (version < 0)
+  return -EINVAL;
  ns = current->nsproxy->ipc_ns;

  switch (cmd) { /* replace with proc interface ? */
@@ -870,6 +880,7 @@ asmlinkage long sys_shmctl(int shmid, in
  }
  case IPC_RMID:
  case IPC_SET:
+ case IPC_SETALL:
   err = shmctl_down(ns, shmid, cmd, buf, version);
   return err;
  case IPC_SETID:
Index: b/ipc/util.c
===================================================================
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -855,7 +855,7 @@ struct kern_ipc_perm *ipcctl_pre_down(st
  if (err)
   goto out_unlock;

- if (cmd == IPC_SET) {
+ if (cmd == IPC_SET || cmd == IPC_SETALL) {
   err = audit_ipc_set_perm(extrat_perm, perm->uid,
       perm->gid, perm->mode);
   if (err)
@@ -883,6 +883,8 @@ out_up:
  * Return IPC_64 for new style IPC and IPC_OLD for old style IPC.
  * The @cmd value is turned from an encoding command and version into
  * just the command code.
+ * In case of incompatibility between the command and the style, an
+ * errcode is returned.
  */

 int ipc_parse_version (int *cmd)
@@ -891,6 +893,9 @@ int ipc_parse_version (int *cmd)
   *cmd ^= IPC_64;
   return IPC_64;
  } else {
+  /* don't support this command for old ipc */
+  if (*cmd == IPC_SETALL)
+   return -EINVAL;
   return IPC_OLD;
  }
 }
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_SETALL:
  case IPC_SETID:
   perms = MSGQ__SETATTR;
   break;
@@ -4800,6 +4801,7 @@ static int selinux_shm_shmctl(struct shm
   perms = SHM__GETATTR | SHM__ASSOCIATE;
   break;
  case IPC_SET:
+ case IPC_SETALL:
  case IPC_SETID:
   perms = SHM__SETATTR;
   break;
@@ -4912,6 +4914,7 @@ static int selinux_sem_semctl(struct sem
   perms = SEM__DESTROY;
   break;
  case IPC_SET:
+ case IPC_SETALL:
  case IPC_SETID:
   perms = SEM__SETATTR;
   break;
```

--
Pierre Peiffer