Subject: [PATCH 2.6.24-rc8-mm1 08/15] IPC: consolidate all xxxctl_down() functions
Posted by Pierre Peiffer on Tue, 29 Jan 2008 16:02:37 GMT
View Forum Message <> Reply to Message

semctl_down(), msgctl_down() and shmctl_down() are used to handle the same
set of commands for each kind of IPC. They all start to do the same job (they
retrieve the ipc and do some permission checks) before handling the commands
on their own.

This patch proposes to consolidate this by moving these same pieces of code
into one common function called ipcctl_pre_down().
It simplifies a little these xxxctl_down() functions and increases a little
the maintainability.

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
Acked-by: Serge Hallyn <serue@us.ibm.com>
---
 ipc/msg.c | 48 +++++----------------------------------
 ipc/sem.c | 42 ++++-------------------------------
 ipc/shm.c | 42 ++++-------------------------------
 ipc/util.c | 51 +++
 ipc/util.h | 2 ++
 5 files changed, 66 insertions(+), 119 deletions(-)

Index: b/ipc/sem.c
===
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -142,21 +142,6 @@ void __init sem_init (void)
 }

 /*
- * This routine is called in the paths where the rw_mutex is held to protect
- * access to the idr tree.
- */
-static inline struct sem_array *sem_lock_check_down(struct ipc_namespace *ns,
-      int id)
-{
- struct kern_ipc_perm *ipcp = ipc_lock_check_down(&sem_ids(ns), id);
-
- if (IS_ERR(ipcp))
- return (struct sem_array *)ipcp;
-
- return container_of(ipcp, struct sem_array, sem_perm);
-}
-
-/*

```
  * sem_lock_(check_) routines are called in the paths where the rw_mutex
  * is not held.
  */
@@ -880,31 +865,12 @@ static int semctl_down(struct ipc_namesp
   if (copy_semid_from_user(&semid64, arg.buf, version))
    return -EFAULT;
  }
- down_write(&sem_ids(ns).rw_mutex);
- sma = sem_lock_check_down(ns, semid);
- if (IS_ERR(sma)) {
-  err = PTR_ERR(sma);
-  goto out_up;
- }
-
- ipcp = &sma->sem_perm;

- err = audit_ipc_obj(ipcp);
- if (err)
-  goto out_unlock;
+ ipcp = ipcctl_pre_down(&sem_ids(ns), semid, cmd, &semid64.sem_perm, 0);
+ if (IS_ERR(ipcp))
+  return PTR_ERR(ipcp);

- if (cmd == IPC_SET) {
-  err = audit_ipc_set_perm(0, semid64.sem_perm.uid,
-      semid64.sem_perm.gid,
-      semid64.sem_perm.mode);
-  if (err)
-   goto out_unlock;
- }
- if (current->euid != ipcp->cuid &&
-     current->euid != ipcp->uid && !capable(CAP_SYS_ADMIN)) {
-      err=-EPERM;
-  goto out_unlock;
- }
+ sma = container_of(ipcp, struct sem_array, sem_perm);

  err = security_sem_semctl(sma, cmd);
  if (err)
Index: b/ipc/util.c
===================================================================
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -774,6 +774,57 @@ void ipc_update_perm(struct ipc64_perm *
   | (in->mode & S_IRWXUGO);
 }

+/**
```

```
+ * ipcctl_pre_down - retrieve an ipc and check permissions for some IPC_XXX cmd
+ * @ids:  the table of ids where to look for the ipc
+ * @id:   the id of the ipc to retrieve
+ * @cmd:  the cmd to check
+ * @perm: the permission to set
+ * @extra_perm: one extra permission parameter used by msq
+ *
+ * This function does some common audit and permissions check for some IPC_XXX
+ * cmd and is called from semctl_down, shmctl_down and msgctl_down.
+ * It must be called without any lock held and
+ *  - retrieves the ipc with the given id in the given table.
+ *  - performs some audit and permission check, depending on the given cmd
+ *  - returns the ipc with both ipc and rw_mutex locks held in case of success
+ *    or an err-code without any lock held otherwise.
+ */
+struct kern_ipc_perm *ipcctl_pre_down(struct ipc_ids *ids, int id, int cmd,
+          struct ipc64_perm *perm, int extrat_perm)
+{
+ struct kern_ipc_perm *ipcp;
+ int err;
+
+ down_write(&ids->rw_mutex);
+ ipcp = ipc_lock_check_down(ids, id);
+ if (IS_ERR(ipcp)) {
+  err = PTR_ERR(ipcp);
+  goto out_up;
+ }
+
+ err = audit_ipc_obj(ipcp);
+ if (err)
+  goto out_unlock;
+
+ if (cmd == IPC_SET) {
+  err = audit_ipc_set_perm(extrat_perm, perm->uid,
+      perm->gid, perm->mode);
+  if (err)
+   goto out_unlock;
+ }
+ if (current->euid == ipcp->cuid ||
+     current->euid == ipcp->uid || capable(CAP_SYS_ADMIN))
+  return ipcp;
+
+ err = -EPERM;
+out_unlock:
+ ipc_unlock(ipcp);
+out_up:
+ up_write(&ids->rw_mutex);
+ return ERR_PTR(err);
```

```
+}
+
 #ifdef __ARCH_WANT_IPC_PARSE_VERSION


Index: b/ipc/util.h
===================================================================
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -113,6 +113,8 @@ struct kern_ipc_perm *ipc_lock(struct ip
 void kernel_to_ipc64_perm(struct kern_ipc_perm *in, struct ipc64_perm *out);
 void ipc64_perm_to_ipc_perm(struct ipc64_perm *in, struct ipc_perm *out);
 void ipc_update_perm(struct ipc64_perm *in, struct kern_ipc_perm *out);
+struct kern_ipc_perm *ipcctl_pre_down(struct ipc_ids *ids, int id, int cmd,
+          struct ipc64_perm *perm, int extrat_perm);

 #if defined(__ia64__) || defined(__x86_64__) || defined(__hppa__) || defined(__XTENSA__)
   /* On IA-64, we always use the "64-bit version" of the IPC structures.  */
Index: b/ipc/msg.c
===================================================================
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -104,21 +104,6 @@ void __init msg_init(void)
 }

 /*
- * This routine is called in the paths where the rw_mutex is held to protect
- * access to the idr tree.
- */
-static inline struct msg_queue *msg_lock_check_down(struct ipc_namespace *ns,
-      int id)
-{
- struct kern_ipc_perm *ipcp = ipc_lock_check_down(&msg_ids(ns), id);
-
- if (IS_ERR(ipcp))
-  return (struct msg_queue *)ipcp;
-
- return container_of(ipcp, struct msg_queue, q_perm);
-}
-
-/*
  * msg_lock_(check_) routines are called in the paths where the rw_mutex
  * is not held.
  */
@@ -400,35 +385,12 @@ static int msgctl_down(struct ipc_namesp
    return -EFAULT;
  }
```

```
- down_write(&msg_ids(ns).rw_mutex);
- msq = msg_lock_check_down(ns, msqid);
- if (IS_ERR(msq)) {
-  err = PTR_ERR(msq);
-  goto out_up;
- }
-
- ipcp = &msq->q_perm;
-
- err = audit_ipc_obj(ipcp);
- if (err)
-  goto out_unlock;
-
- if (cmd == IPC_SET) {
-  err = audit_ipc_set_perm(msqid64.msg_qbytes,
-      msqid64.msg_perm.uid,
-      msqid64.msg_perm.gid,
-      msqid64.msg_perm.mode);
-  if (err)
-   goto out_unlock;
- }
+ ipcp = ipcctl_pre_down(&msg_ids(ns), msqid, cmd,
+          &msqid64.msg_perm, msqid64.msg_qbytes);
+ if (IS_ERR(ipcp))
+  return PTR_ERR(ipcp);

- if (current->euid != ipcp->cuid &&
-     current->euid != ipcp->uid &&
-     !capable(CAP_SYS_ADMIN)) {
-  /* We _could_ check for CAP_CHOWN above, but we don't */
-  err = -EPERM;
-  goto out_unlock;
- }
+ msq = container_of(ipcp, struct msg_queue, q_perm);

  err = security_msg_queue_msgctl(msq, cmd);
  if (err)
Index: b/ipc/shm.c
===================================================================
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -127,18 +127,6 @@ static inline struct shmid_kernel *shm_l
  return container_of(ipcp, struct shmid_kernel, shm_perm);
 }

-static inline struct shmid_kernel *shm_lock_check_down(
-      struct ipc_namespace *ns,
-      int id)
```

```
-{
- struct kern_ipc_perm *ipcp = ipc_lock_check_down(&shm_ids(ns), id);
-
- if (IS_ERR(ipcp))
-  return (struct shmid_kernel *)ipcp;
-
- return container_of(ipcp, struct shmid_kernel, shm_perm);
-}
-
 /*
  * shm_lock_(check_) routines are called in the paths where the rw_mutex
  * is not held.
@@ -629,33 +617,11 @@ static int shmctl_down(struct ipc_namesp
    return -EFAULT;
  }

- down_write(&shm_ids(ns).rw_mutex);
- shp = shm_lock_check_down(ns, shmid);
- if (IS_ERR(shp)) {
-  err = PTR_ERR(shp);
-  goto out_up;
- }
-
- ipcp = &shp->shm_perm;
-
- err = audit_ipc_obj(ipcp);
- if (err)
-  goto out_unlock;
-
- if (cmd == IPC_SET) {
-  err = audit_ipc_set_perm(0, shmid64.shm_perm.uid,
-      shmid64.shm_perm.gid,
-      shmid64.shm_perm.mode);
-  if (err)
-   goto out_unlock;
- }
+ ipcp = ipcctl_pre_down(&shm_ids(ns), shmid, cmd, &shmid64.shm_perm, 0);
+ if (IS_ERR(ipcp))
+  return PTR_ERR(ipcp);

- if (current->euid != ipcp->uid &&
-     current->euid != ipcp->cuid &&
-     !capable(CAP_SYS_ADMIN)) {
-  err = -EPERM;
-  goto out_unlock;
- }
+ shp = container_of(ipcp, struct shmid_kernel, shm_perm);
```

```
  err = security_shm_shmctl(shp, cmd);
  if (err)
```

--
Pierre Peiffer

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