Subject: [PATCH 11/13] Changes to show virtual ids to user
Posted by Pavel Emelianov on Thu, 24 May 2007 13:06:40 GMT
View Forum Message <> Reply to Message

This is the largest patch in the set. Make all (I hope)
the places where the pid is shown to or get from user
operate on the virtual pids.

An exception is copy_process - it was in one of the
previous patches - and the proc - this will come as a
separate patch.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
diff --git a/arch/ia64/kernel/signal.c b/arch/ia64/kernel/signal.c
index aeec818..cdb64cc 100644
--- a/arch/ia64/kernel/signal.c
+++ b/arch/ia64/kernel/signal.c
@@ -227,7 +227,7 @@ ia64_rt_sigreturn (struct sigscratch *sc
  si.si_signo = SIGSEGV;
  si.si_errno = 0;
  si.si_code = SI_KERNEL;
- si.si_pid = current->pid;
+ si.si_pid = task_pid_vnr(current);
  si.si_uid = current->uid;
  si.si_addr = sc;
  force_sig_info(SIGSEGV, &si, current);
@@ -332,7 +332,7 @@ force_sigsegv_info (int sig, void __user
  si.si_signo = SIGSEGV;
  si.si_errno = 0;
  si.si_code = SI_KERNEL;
- si.si_pid = current->pid;
+ si.si_pid = task_pid_vnr(current);
  si.si_uid = current->uid;
  si.si_addr = addr;
  force_sig_info(SIGSEGV, &si, current);
diff --git a/arch/parisc/kernel/signal.c b/arch/parisc/kernel/signal.c
index fb35ebc..2ce3806 100644
--- a/arch/parisc/kernel/signal.c
+++ b/arch/parisc/kernel/signal.c
@@ -181,7 +181,7 @@ give_sigsegv:
  si.si_signo = SIGSEGV;
  si.si_errno = 0;
  si.si_code = SI_KERNEL;
- si.si_pid = current->pid;
+ si.si_pid = task_pid_vnr(current);
```

```
  si.si_uid = current->uid;
  si.si_addr = &frame->uc;
  force_sig_info(SIGSEGV, &si, current);
diff --git a/drivers/char/tty_io.c b/drivers/char/tty_io.c
index 260a1f3..9161107 100644
--- a/drivers/char/tty_io.c
+++ b/drivers/char/tty_io.c
@@ -103,6 +103,7 @@
 #include <linux/selection.h>

 #include <linux/kmod.h>
+#include <linux/nsproxy.h>

 #undef TTY_DEBUG_HANGUP

@@ -3065,7 +3066,7 @@ static int tiocgpgrp(struct tty_struct *
    */
   if (tty == real_tty && current->signal->tty != real_tty)
    return -ENOTTY;
- return put_user(pid_nr(real_tty->pgrp), p);
+ return put_user(pid_vnr(real_tty->pgrp), p);
 }

 /**
@@ -3099,7 +3100,7 @@ static int tiocspgrp(struct tty_struct *
   if (pgrp_nr < 0)
    return -EINVAL;
   rcu_read_lock();
- pgrp = find_pid(pgrp_nr);
+ pgrp = find_vpid(pgrp_nr);
   retval = -ESRCH;
   if (!pgrp)
    goto out_unlock;
@@ -3136,7 +3137,7 @@ static int tiocgsid(struct tty_struct *t
    return -ENOTTY;
   if (!real_tty->session)
    return -ENOTTY;
- return put_user(pid_nr(real_tty->session), p);
+ return put_user(pid_vnr(real_tty->session), p);
 }

 /**
diff --git a/fs/binfmt_elf.c b/fs/binfmt_elf.c
index 7893feb..6f30d05 100644
--- a/fs/binfmt_elf.c
+++ b/fs/binfmt_elf.c
@@ -1325,10 +1325,10 @@ static void fill_prstatus(struct elf_prs
  prstatus->pr_info.si_signo = prstatus->pr_cursig = signr;
```

```
  prstatus->pr_sigpend = p->pending.signal.sig[0];
  prstatus->pr_sighold = p->blocked.sig[0];
- prstatus->pr_pid = p->pid;
- prstatus->pr_ppid = p->parent->pid;
- prstatus->pr_pgrp = task_pgrp_nr(p);
- prstatus->pr_sid = task_session_nr(p);
+ prstatus->pr_pid = task_pid_vnr(p);
+ prstatus->pr_ppid = task_pid_vnr(p->parent);
+ prstatus->pr_pgrp = task_pgrp_vnr(p);
+ prstatus->pr_sid = task_session_vnr(p);
  if (thread_group_leader(p)) {
   /*
    * This is the record for the group leader.  Add in the
@@ -1371,10 +1371,10 @@ static int fill_psinfo(struct elf_prpsin
    psinfo->pr_psargs[i] = ' ';
  psinfo->pr_psargs[len] = 0;

- psinfo->pr_pid = p->pid;
- psinfo->pr_ppid = p->parent->pid;
- psinfo->pr_pgrp = task_pgrp_nr(p);
- psinfo->pr_sid = task_session_nr(p);
+ psinfo->pr_pid = task_pid_vnr(p);
+ psinfo->pr_ppid = task_pid_vnr(p->parent);
+ psinfo->pr_pgrp = task_pgrp_vnr(p);
+ psinfo->pr_sid = task_session_vnr(p);

  i = p->state ? ffz(~p->state) + 1 : 0;
  psinfo->pr_state = i;
diff --git a/fs/binfmt_elf_fdpic.c b/fs/binfmt_elf_fdpic.c
index 9bb9ff1..87d6eaf 100644
--- a/fs/binfmt_elf_fdpic.c
+++ b/fs/binfmt_elf_fdpic.c
@@ -1332,10 +1332,10 @@ static void fill_prstatus(struct elf_prs
  prstatus->pr_info.si_signo = prstatus->pr_cursig = signr;
  prstatus->pr_sigpend = p->pending.signal.sig[0];
  prstatus->pr_sighold = p->blocked.sig[0];
- prstatus->pr_pid = p->pid;
- prstatus->pr_ppid = p->parent->pid;
- prstatus->pr_pgrp = task_pgrp_nr(p);
- prstatus->pr_sid = task_session_nr(p);
+ prstatus->pr_pid = task_pid_vnr(p);
+ prstatus->pr_ppid = task_pid_vnr(p->parent);
+ prstatus->pr_pgrp = task_pgrp_vnr(p);
+ prstatus->pr_sid = task_session_vnr(p);
  if (thread_group_leader(p)) {
   /*
    * This is the record for the group leader.  Add in the
@@ -1381,10 +1381,10 @@ static int fill_psinfo(struct elf_prpsin
```

```
    psinfo->pr_psargs[i] = ' ';
  psinfo->pr_psargs[len] = 0;

- psinfo->pr_pid = p->pid;
- psinfo->pr_ppid = p->parent->pid;
- psinfo->pr_pgrp = task_pgrp_nr(p);
- psinfo->pr_sid = task_session_nr(p);
+ psinfo->pr_pid = task_pid_vnr(p);
+ psinfo->pr_ppid = task_pid_vnr(p->parent);
+ psinfo->pr_pgrp = task_pgrp_vnr(p);
+ psinfo->pr_sid = task_session_nr(p);

  i = p->state ? ffz(~p->state) + 1 : 0;
  psinfo->pr_state = i;
diff --git a/fs/exec.c b/fs/exec.c
index 0b68588..b8a3582 100644
--- a/fs/exec.c
+++ b/fs/exec.c
@@ -713,6 +713,9 @@ static int de_thread(struct task_struct
   attach_pid(tsk, PIDTYPE_PID,  find_pid(tsk->pid));
   transfer_pid(leader, tsk, PIDTYPE_PGID);
   transfer_pid(leader, tsk, PIDTYPE_SID);
+  set_task_vpgrp(leader, task_pid_vnr(current));
+  set_task_vpid(leader, task_pid_vnr(current));
+  set_task_vtgid(current, task_pid_vnr(current));
   list_replace_rcu(&leader->tasks, &tsk->tasks);

   tsk->group_leader = tsk;
@@ -1301,7 +1304,7 @@ static int format_corename(char *corenam
    case 'p':
     pid_in_pattern = 1;
     rc = snprintf(out_ptr, out_end - out_ptr,
-          "%d", current->tgid);
+          "%d", task_tgid_vnr(current));
     if (rc > out_end - out_ptr)
      goto out;
     out_ptr += rc;
@@ -1373,7 +1376,7 @@ static int format_corename(char *corenam
  if (!ispipe && !pid_in_pattern
            && (core_uses_pid || atomic_read(&current->mm->mm_users) != 1)) {
   rc = snprintf(out_ptr, out_end - out_ptr,
-         ".%d", current->tgid);
+         ".%d", task_tgid_vnr(current));
   if (rc > out_end - out_ptr)
    goto out;
   out_ptr += rc;
diff --git a/include/net/scm.h b/include/net/scm.h
index 5637d5e..43ed2f1 100644
```

```
--- a/include/net/scm.h
+++ b/include/net/scm.h
@@ -54,7 +54,7 @@ static __inline__ int scm_send(struct so
  struct task_struct *p = current;
  scm->creds.uid = p->uid;
  scm->creds.gid = p->gid;
- scm->creds.pid = p->tgid;
+ scm->creds.pid = task_tgid_vnr(p);
  scm->fp = NULL;
  scm->seq = 0;
  unix_get_peersec_dgram(sock, scm);
diff --git a/ipc/mqueue.c b/ipc/mqueue.c
index a242c83..9480a9e 100644
--- a/ipc/mqueue.c
+++ b/ipc/mqueue.c
@@ -513,7 +513,7 @@ static void __do_notify(struct mqueue_in
    sig_i.si_errno = 0;
    sig_i.si_code = SI_MESGQ;
    sig_i.si_value = info->notify.sigev_value;
-   sig_i.si_pid = current->tgid;
+   sig_i.si_pid = task_pid_vnr(current);
    sig_i.si_uid = current->uid;

    kill_pid_info(info->notify.sigev_signo,
diff --git a/ipc/msg.c b/ipc/msg.c
index 1cdb378..75fcbf6 100644
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -615,7 +615,7 @@ static inline int pipelined_send(struct
     msr->r_msg = ERR_PTR(-E2BIG);
    } else {
     msr->r_msg = NULL;
-    msq->q_lrpid = msr->r_tsk->pid;
+    msq->q_lrpid = task_pid_vnr(msr->r_tsk);
     msq->q_rtime = get_seconds();
     wake_up_process(msr->r_tsk);
     smp_mb();
@@ -699,7 +699,7 @@ long do_msgsnd(int msqid, long mtype, vo
   }
  }

- msq->q_lspid = current->tgid;
+ msq->q_lspid = task_tgid_vnr(current);
  msq->q_stime = get_seconds();

  if (!pipelined_send(msq, msg)) {
@@ -814,7 +814,7 @@ long do_msgrcv(int msqid, long *pmtype,
    list_del(&msg->m_list);
```

```
    msq->q_qnum--;
    msq->q_rtime = get_seconds();
-   msq->q_lrpid = current->tgid;
+   msq->q_lrpid = task_tgid_vnr(current);
    msq->q_cbytes -= msg->m_ts;
    atomic_sub(msg->m_ts, &msg_bytes);
    atomic_dec(&msg_hdrs);
diff --git a/ipc/sem.c b/ipc/sem.c
index 0f96683..5b2ef9a 100644
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -797,7 +797,7 @@ static int semctl_main(struct ipc_namesp
   for (un = sma->undo; un; un = un->id_next)
    un->semadj[semnum] = 0;
   curr->semval = val;
-  curr->sempid = current->tgid;
+  curr->sempid = task_tgid_vnr(current);
   sma->sem_ctime = get_seconds();
   /* maybe some queued-up processes were waiting for this */
   update_queue(sma);
@@ -1200,7 +1200,7 @@ retry_undos:
  if (error)
   goto out_unlock_free;

- error = try_atomic_semop (sma, sops, nsops, un, current->tgid);
+ error = try_atomic_semop (sma, sops, nsops, un, task_tgid_vnr(current));
  if (error <= 0) {
   if (alter && error == 0)
    update_queue (sma);
@@ -1215,7 +1215,7 @@ retry_undos:
  queue.sops = sops;
  queue.nsops = nsops;
  queue.undo = un;
- queue.pid = current->tgid;
+ queue.pid = task_tgid_vnr(current);
  queue.id = semid;
  queue.alter = alter;
  if (alter)
@@ -1386,7 +1386,7 @@ found:
      semaphore->semval = 0;
     if (semaphore->semval > SEMVMX)
      semaphore->semval = SEMVMX;
-    semaphore->sempid = current->tgid;
+    semaphore->sempid = task_tgid_vnr(current);
    }
   }
   sma->sem_otime = get_seconds();
diff --git a/ipc/shm.c b/ipc/shm.c
```

```
index bf28d5f..ae98d4f 100644
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -170,7 +170,7 @@ static void shm_open(struct vm_area_stru
  shp = shm_lock(sfd->ns, sfd->id);
  BUG_ON(!shp);
  shp->shm_atim = get_seconds();
- shp->shm_lprid = current->tgid;
+ shp->shm_lprid = task_tgid_vnr(current);
  shp->shm_nattch++;
  shm_unlock(shp);
 }
@@ -215,7 +215,7 @@ static void shm_close(struct vm_area_str
  /* remove from the list of attaches of the shm segment */
  shp = shm_lock(ns, sfd->id);
  BUG_ON(!shp);
- shp->shm_lprid = current->tgid;
+ shp->shm_lprid = task_tgid_vnr(current);
  shp->shm_dtim = get_seconds();
  shp->shm_nattch--;
  if(shp->shm_nattch == 0 &&
@@ -390,7 +390,7 @@ static int newseg (struct ipc_namespace
  if(id == -1)
   goto no_id;

- shp->shm_cprid = current->tgid;
+ shp->shm_cprid = task_tgid_vnr(current);
  shp->shm_lprid = 0;
  shp->shm_atim = shp->shm_dtim = 0;
  shp->shm_ctim = get_seconds();
diff --git a/kernel/capability.c b/kernel/capability.c
index c8d3c77..a3d5395 100644
--- a/kernel/capability.c
+++ b/kernel/capability.c
@@ -12,6 +12,7 @@
 #include <linux/module.h>
 #include <linux/security.h>
 #include <linux/syscalls.h>
+#include <linux/pid_namespace.h>
 #include <asm/uaccess.h>

 unsigned securebits = SECUREBITS_DEFAULT; /* systemwide security settings */
@@ -67,8 +68,9 @@ asmlinkage long sys_capget(cap_user_head
      spin_lock(&task_capability_lock);
      read_lock(&tasklist_lock);

-     if (pid && pid != current->pid) {
-      target = find_task_by_pid(pid);
```

```
+     if (pid && pid != task_pid_vnr(current)) {
+      target = find_task_by_pid_ns(pid,
+         current->nsproxy->pid_ns);
       if (!target) {
            ret = -ESRCH;
            goto out;
@@ -190,7 +192,7 @@ asmlinkage long sys_capset(cap_user_head
      if (get_user(pid, &header->pid))
       return -EFAULT;

-     if (pid && pid != current->pid && !capable(CAP_SETPCAP))
+     if (pid && pid != task_pid_vnr(current) && !capable(CAP_SETPCAP))
             return -EPERM;

      if (copy_from_user(&effective, &data->effective, sizeof(effective)) ||
@@ -201,8 +203,9 @@ asmlinkage long sys_capset(cap_user_head
      spin_lock(&task_capability_lock);
      read_lock(&tasklist_lock);

-     if (pid > 0 && pid != current->pid) {
-          target = find_task_by_pid(pid);
+     if (pid > 0 && pid != task_pid_vnr(current)) {
+          target = find_task_by_pid_ns(pid,
+     current->nsproxy->pid_ns);
           if (!target) {
                ret = -ESRCH;
                goto out;
diff --git a/kernel/exit.c b/kernel/exit.c
index 43ce25b..942e01d 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -950,8 +950,8 @@ fastcall NORET_TYPE void do_exit(long co

  tsk->exit_code = code;
  proc_exit_connector(tsk);
- exit_task_namespaces(tsk);
  exit_notify(tsk);
+ exit_task_namespaces(tsk);
 #ifdef CONFIG_NUMA
  mpol_free(tsk->mempolicy);
  tsk->mempolicy = NULL;
@@ -1047,13 +1047,13 @@ static int eligible_child(pid_t pid, int
  int err;

  if (pid > 0) {
-  if (p->pid != pid)
+  if (task_pid_nr_ns(p) != pid)
     return 0;
```

```
  } else if (!pid) {
-  if (task_pgrp_nr(p) != task_pgrp_nr(current))
+  if (task_pgrp_nr_ns(p) != task_pgrp_vnr(current))
     return 0;
  } else if (pid != -1) {
-  if (task_pgrp_nr(p) != -pid)
+  if (task_pgrp_nr_ns(p) != -pid)
     return 0;
  }

@@ -1126,7 +1126,7 @@ static int wait_task_zombie(struct task_
  int status;

  if (unlikely(noreap)) {
-  pid_t pid = p->pid;
+  pid_t pid = task_pid_nr_ns(p);
    uid_t uid = p->uid;
    int exit_code = p->exit_code;
    int why, status;
@@ -1244,7 +1244,7 @@ static int wait_task_zombie(struct task_
     retval = put_user(status, &infop->si_status);
  }
  if (!retval && infop)
-  retval = put_user(p->pid, &infop->si_pid);
+  retval = put_user(task_pid_nr_ns(p), &infop->si_pid);
  if (!retval && infop)
    retval = put_user(p->uid, &infop->si_uid);
  if (retval) {
@@ -1252,7 +1252,7 @@ static int wait_task_zombie(struct task_
    p->exit_state = EXIT_ZOMBIE;
    return retval;
  }
- retval = p->pid;
+ retval = task_pid_nr_ns(p);
  if (p->real_parent != p->parent) {
    write_lock_irq(&tasklist_lock);
    /* Double-check with lock held.  */
@@ -1312,7 +1312,7 @@ static int wait_task_stopped(struct task
  read_unlock(&tasklist_lock);

  if (unlikely(noreap)) {
-  pid_t pid = p->pid;
+  pid_t pid = task_pid_nr_ns(p);
    uid_t uid = p->uid;
    int why = (p->ptrace & PT_PTRACED) ? CLD_TRAPPED : CLD_STOPPED;

@@ -1383,11 +1383,11 @@ bail_ref:
  if (!retval && infop)
```

```
   retval = put_user(exit_code, &infop->si_status);
  if (!retval && infop)
-  retval = put_user(p->pid, &infop->si_pid);
+  retval = put_user(task_pid_nr_ns(p), &infop->si_pid);
  if (!retval && infop)
   retval = put_user(p->uid, &infop->si_uid);
  if (!retval)
-  retval = p->pid;
+  retval = task_pid_nr_ns(p);
  put_task_struct(p);

  BUG_ON(!retval);
@@ -1424,7 +1424,7 @@ static int wait_task_continued(struct ta
   p->signal->flags &= ~SIGNAL_STOP_CONTINUED;
  spin_unlock_irq(&p->sighand->siglock);

- pid = p->pid;
+ pid = task_pid_nr_ns(p);
  uid = p->uid;
  get_task_struct(p);
  read_unlock(&tasklist_lock);
@@ -1435,7 +1435,7 @@ static int wait_task_continued(struct ta
   if (!retval && stat_addr)
    retval = put_user(0xffff, stat_addr);
   if (!retval)
-   retval = p->pid;
+   retval = task_pid_nr_ns(p);
  } else {
   retval = wait_noreap_copyout(p, pid, uid,
          CLD_CONTINUED, SIGCONT,
diff --git a/kernel/fork.c b/kernel/fork.c
index d7207a1..3ab517c 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -932,7 +932,7 @@ asmlinkage long sys_set_tid_address(int
 {
  current->clear_child_tid = tidptr;

- return current->pid;
+ return task_pid_vnr(current);
 }

 static inline void rt_mutex_init_task(struct task_struct *p)
diff --git a/kernel/futex.c b/kernel/futex.c
index b7ce15c..4a2a46b 100644
--- a/kernel/futex.c
+++ b/kernel/futex.c
@@ -618,7 +618,7 @@ static int wake_futex_pi(u32 __user *uad
```

```
   * preserve the owner died bit.)
   */
  if (!(uval & FUTEX_OWNER_DIED)) {
-  newval = FUTEX_WAITERS | new_owner->pid;
+  newval = FUTEX_WAITERS | task_pid_vnr(new_owner);
   /* Keep the FUTEX_WAITER_REQUEUED flag if it was set */
   newval |= (uval & FUTEX_WAITER_REQUEUED);

@@ -1334,7 +1334,7 @@ static int fixup_pi_state_owner(u32 __us
     struct futex_hash_bucket *hb,
     struct task_struct *curr)
 {
- u32 newtid = curr->pid | FUTEX_WAITERS;
+ u32 newtid = task_pid_vnr(curr) | FUTEX_WAITERS;
  struct futex_pi_state *pi_state = q->pi_state;
  u32 uval, curval, newval;
  int ret;
@@ -1719,7 +1719,7 @@ static int futex_lock_pi(u32 __user *uad
   * (by doing a 0 -> TID atomic cmpxchg), while holding all
   * the locks. It will most likely not succeed.
   */
- newval = current->pid;
+ newval = task_pid_vnr(current);

  pagefault_disable();
  curval = futex_atomic_cmpxchg_inatomic(uaddr, 0, newval);
@@ -1729,7 +1729,7 @@ static int futex_lock_pi(u32 __user *uad
   goto uaddr_faulted;

  /* We own the lock already */
- if (unlikely((curval & FUTEX_TID_MASK) == current->pid)) {
+ if (unlikely((curval & FUTEX_TID_MASK) == task_pid_vnr(current))) {
   if (!detect && 0)
    force_sig(SIGKILL, current);
   /*
@@ -1759,7 +1759,7 @@ static int futex_lock_pi(u32 __user *uad
   */
  if ((curval & FUTEX_WAITER_REQUEUED) && !(curval & FUTEX_TID_MASK)) {
   /* set current as futex owner */
-  newval = curval | current->pid;
+  newval = curval | task_pid_vnr(current);
   lock_held = 1;
  } else
   /* Set the WAITERS flag, so the owner will know it has someone
@@ -1800,7 +1800,7 @@ static int futex_lock_pi(u32 __user *uad
    */
   if (curval & FUTEX_OWNER_DIED) {
    uval = newval;
```

```
-   newval = current->pid |
+   newval = task_pid_vnr(current) |
     FUTEX_OWNER_DIED | FUTEX_WAITERS;

    pagefault_disable();
@@ -1927,7 +1927,7 @@ retry:
  /*
   * We release only a lock we actually own:
   */
- if ((uval & FUTEX_TID_MASK) != current->pid)
+ if ((uval & FUTEX_TID_MASK) != task_pid_vnr(current))
   return -EPERM;
  /*
   * First take all the futex related locks:
@@ -1950,7 +1950,8 @@ retry_locked:
   */
  if (!(uval & FUTEX_OWNER_DIED)) {
   pagefault_disable();
-  uval = futex_atomic_cmpxchg_inatomic(uaddr, current->pid, 0);
+  uval = futex_atomic_cmpxchg_inatomic(uaddr,
+    task_pid_vnr(current), 0);
   pagefault_enable();
  }

@@ -1960,7 +1961,7 @@ retry_locked:
   * Rare case: we managed to release the lock atomically,
   * no need to wake anyone else up:
   */
- if (unlikely(uval == current->pid))
+ if (unlikely(uval == task_pid_vnr(current)))
   goto out_unlock;

  /*
@@ -2228,7 +2229,7 @@ retry:
  if (get_user(uval, uaddr))
   return -1;

- if ((uval & FUTEX_TID_MASK) == curr->pid) {
+ if ((uval & FUTEX_TID_MASK) == task_pid_vnr(curr)) {
   /*
    * Ok, this dying thread is truly holding a futex
    * of interest. Set the OWNER_DIED bit atomically
diff --git a/kernel/ptrace.c b/kernel/ptrace.c
index b1d11f1..11557c6 100644
--- a/kernel/ptrace.c
+++ b/kernel/ptrace.c
@@ -19,6 +19,7 @@
 #include <linux/security.h>
```

```
 #include <linux/signal.h>
 #include <linux/audit.h>
+#include <linux/pid_namespace.h>

 #include <asm/pgtable.h>
 #include <asm/uaccess.h>
@@ -439,7 +440,8 @@ struct task_struct *ptrace_get_task_stru
   return ERR_PTR(-EPERM);

  read_lock(&tasklist_lock);
- child = find_task_by_pid(pid);
+ child = find_task_by_pid_ns(pid,
+   current->nsproxy->pid_ns);
  if (child)
   get_task_struct(child);

diff --git a/kernel/sched.c b/kernel/sched.c
index 4784a8d..0c0a955 100644
--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -54,6 +54,7 @@
 #include <linux/kprobes.h>
 #include <linux/delayacct.h>
 #include <linux/reciprocal_div.h>
+#include <linux/pid_namespace.h>

 #include <asm/tlb.h>
 #include <asm/unistd.h>
@@ -1888,7 +1889,7 @@ asmlinkage void schedule_tail(struct tas
  preempt_enable();
 #endif
  if (current->set_child_tid)
-  put_user(current->pid, current->set_child_tid);
+  put_user(task_pid_vnr(current), current->set_child_tid);
 }

 /*
diff --git a/kernel/signal.c b/kernel/signal.c
index 75c5d77..0949043 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -663,7 +663,7 @@ static int send_signal(int sig, struct s
    q->info.si_signo = sig;
    q->info.si_errno = 0;
    q->info.si_code = SI_USER;
-   q->info.si_pid = current->pid;
+   q->info.si_pid = task_pid_vnr(current);
    q->info.si_uid = current->uid;
```

```
    break;
   case (unsigned long) SEND_SIG_PRIV:
@@ -1231,9 +1231,9 @@ static int kill_something_info(int sig,
   read_unlock(&tasklist_lock);
   ret = count ? retval : -ESRCH;
  } else if (pid < 0) {
-  ret = kill_pgrp_info(sig, info, find_pid(-pid));
+  ret = kill_pgrp_info(sig, info, find_vpid(-pid));
  } else {
-  ret = kill_pid_info(sig, info, find_pid(pid));
+  ret = kill_pid_info(sig, info, find_vpid(pid));
  }
  rcu_read_unlock();
  return ret;
@@ -1515,7 +1515,11 @@ void do_notify_parent(struct task_struct

  info.si_signo = sig;
  info.si_errno = 0;
- info.si_pid = tsk->pid;
+ /*
+  * we are under tasklist_lock here so our parent is tied to
+  * us and cannot exit and release its namespace.
+  */
+ info.si_pid = __task_pid_nr_ns(tsk, tsk->parent->nsproxy->pid_ns);
  info.si_uid = tsk->uid;

  /* FIXME: find out whether or not this is supposed to be c*time. */
@@ -1580,7 +1584,11 @@ static void do_notify_parent_cldstop(str

  info.si_signo = SIGCHLD;
  info.si_errno = 0;
- info.si_pid = tsk->pid;
+ /*
+  * we are under tasklist_lock here so our parent is tied to
+  * us and cannot exit and release its namespace.
+  */
+ info.si_pid = __task_pid_nr_ns(tsk, tsk->parent->nsproxy->pid_ns);
  info.si_uid = tsk->uid;

  /* FIXME: find out whether or not this is supposed to be c*time. */
@@ -1709,7 +1717,7 @@ void ptrace_notify(int exit_code)
  memset(&info, 0, sizeof info);
  info.si_signo = SIGTRAP;
  info.si_code = exit_code;
- info.si_pid = current->pid;
+ info.si_pid = task_pid_vnr(current);
  info.si_uid = current->uid;
```

```
  /* Let the debugger run.  */
@@ -1879,7 +1887,7 @@ relock:
     info->si_signo = signr;
     info->si_errno = 0;
     info->si_code = SI_USER;
-    info->si_pid = current->parent->pid;
+    info->si_pid = task_pid_vnr(current->parent);
     info->si_uid = current->parent->uid;
    }

@@ -2268,7 +2276,7 @@ sys_kill(int pid, int sig)
  info.si_signo = sig;
  info.si_errno = 0;
  info.si_code = SI_USER;
- info.si_pid = current->tgid;
+ info.si_pid = task_tgid_vnr(current);
  info.si_uid = current->uid;

  return kill_something_info(sig, &info, pid);
@@ -2284,12 +2292,12 @@ static int do_tkill(int tgid, int pid, i
  info.si_signo = sig;
  info.si_errno = 0;
  info.si_code = SI_TKILL;
- info.si_pid = current->tgid;
+ info.si_pid = task_tgid_vnr(current);
  info.si_uid = current->uid;

  read_lock(&tasklist_lock);
  p = find_task_by_pid(pid);
- if (p && (tgid <= 0 || p->tgid == tgid)) {
+ if (p && (tgid <= 0 || task_tgid_vnr(p) == tgid)) {
   error = check_kill_permission(sig, &info, p);
   /*
    * The null signal is a permissions and process existence
diff --git a/kernel/sys.c b/kernel/sys.c
index 8aefd5e..890ff1c 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -674,7 +674,8 @@ asmlinkage long sys_setpriority(int whic
  switch (which) {
   case PRIO_PROCESS:
    if (who)
-    p = find_task_by_pid(who);
+    p = find_task_by_pid_ns(who,
+      current->nsproxy->pid_ns);
    else
     p = current;
    if (p)
```

```
@@ -731,7 +732,8 @@ asmlinkage long sys_getpriority(int whic
  switch (which) {
   case PRIO_PROCESS:
    if (who)
-    p = find_task_by_pid(who);
+    p = find_task_by_pid_ns(who,
+      current->nsproxy->pid_ns);
    else
     p = current;
    if (p) {
@@ -1436,7 +1438,7 @@ asmlinkage long sys_setpgid(pid_t pid, p
  int err = -EINVAL;

  if (!pid)
-  pid = group_leader->pid;
+  pid = task_pid_vnr(group_leader);
  if (!pgid)
   pgid = pid;
  if (pgid < 0)
@@ -1448,7 +1450,7 @@ asmlinkage long sys_setpgid(pid_t pid, p
  write_lock_irq(&tasklist_lock);

  err = -ESRCH;
- p = find_task_by_pid(pid);
+ p = find_task_by_pid_ns(pid, current->nsproxy->pid_ns);
  if (!p)
   goto out;

@@ -1475,7 +1477,8 @@ asmlinkage long sys_setpgid(pid_t pid, p

  if (pgid != pid) {
   struct task_struct *g =
-   find_task_by_pid_type(PIDTYPE_PGID, pgid);
+   find_task_by_pid_type_ns(PIDTYPE_PGID, pgid,
+     current->nsproxy->pid_ns);

   if (!g || task_session(g) != task_session(group_leader))
    goto out;
@@ -1486,9 +1489,13 @@ asmlinkage long sys_setpgid(pid_t pid, p
   goto out;

  if (task_pgrp_nr(p) != pgid) {
+  struct pid *pid;
+
   detach_pid(p, PIDTYPE_PGID);
-  p->signal->pgrp = pgid;
-  attach_pid(p, PIDTYPE_PGID, find_pid(pgid));
+  pid = find_vpid(pgid);
```

```
+  attach_pid(p, PIDTYPE_PGID, pid);
+  p->signal->pgrp = pid_nr(pid);
+  set_task_vpgrp(p, pid_vnr(pid));
  }

  err = 0;
@@ -1501,19 +1508,20 @@ out:
 asmlinkage long sys_getpgid(pid_t pid)
 {
  if (!pid)
-  return task_pgrp_nr(current);
+  return task_pgrp_vnr(current);
  else {
   int retval;
   struct task_struct *p;

   read_lock(&tasklist_lock);
-  p = find_task_by_pid(pid);
+  p = find_task_by_pid_ns(pid,
+    current->nsproxy->pid_ns);

   retval = -ESRCH;
   if (p) {
    retval = security_task_getpgid(p);
    if (!retval)
-    retval = task_pgrp_nr(p);
+    retval = task_pgrp_vnr(p);
   }
   read_unlock(&tasklist_lock);
   return retval;
@@ -1525,7 +1533,7 @@ asmlinkage long sys_getpgid(pid_t pid)
 asmlinkage long sys_getpgrp(void)
 {
  /* SMP - assuming writes are word atomic this is fine */
- return task_pgrp_nr(current);
+ return task_pgrp_vnr(current);
 }

 #endif
@@ -1533,19 +1541,20 @@ asmlinkage long sys_getpgrp(void)
 asmlinkage long sys_getsid(pid_t pid)
 {
  if (!pid)
-  return task_session_nr(current);
+  return task_session_vnr(current);
  else {
   int retval;
   struct task_struct *p;
```

```
   read_lock(&tasklist_lock);
-  p = find_task_by_pid(pid);
+  p = find_task_by_pid_ns(pid,
+    current->nsproxy->pid_ns);

   retval = -ESRCH;
   if (p) {
    retval = security_task_getsid(p);
    if (!retval)
-    retval = task_session_nr(p);
+    retval = task_session_vnr(p);
   }
   read_unlock(&tasklist_lock);
   return retval;
@@ -1577,12 +1586,14 @@ asmlinkage long sys_setsid(void)

  group_leader->signal->leader = 1;
  __set_special_pids(session, session);
+ set_task_vsession(current, task_pid_vnr(group_leader));
+ set_task_vpgrp(current, task_pid_vnr(group_leader));

  spin_lock(&group_leader->sighand->siglock);
  group_leader->signal->tty = NULL;
  spin_unlock(&group_leader->sighand->siglock);

- err = task_pgrp_nr(group_leader);
+ err = task_pgrp_vnr(group_leader);
 out:
  write_unlock_irq(&tasklist_lock);
  return err;
diff --git a/kernel/timer.c b/kernel/timer.c
index 00a38b8..a60eed1 100644
--- a/kernel/timer.c
+++ b/kernel/timer.c
@@ -36,6 +36,7 @@
 #include <linux/delay.h>
 #include <linux/tick.h>
 #include <linux/kallsyms.h>
+#include <linux/pid_namespace.h>

 #include <asm/uaccess.h>
 #include <asm/unistd.h>
@@ -931,7 +932,7 @@ asmlinkage unsigned long sys_alarm(unsig
  */
 asmlinkage long sys_getpid(void)
 {
- return current->tgid;
```

```
+ return task_tgid_vnr(current);
 }

 /*
@@ -945,7 +946,7 @@ asmlinkage long sys_getppid(void)
  int pid;

  rcu_read_lock();
- pid = rcu_dereference(current->real_parent)->tgid;
+ pid = task_ppid_nr_ns(current, current->nsproxy->pid_ns);
  rcu_read_unlock();

  return pid;
@@ -1077,7 +1078,7 @@ EXPORT_SYMBOL(schedule_timeout_uninterru
 /* Thread ID - the internal kernel "pid" */
 asmlinkage long sys_gettid(void)
 {
- return current->pid;
+ return task_pid_vnr(current);
 }

 /**
diff --git a/net/core/scm.c b/net/core/scm.c
index 292ad8d..63c6ea9 100644
--- a/net/core/scm.c
+++ b/net/core/scm.c
@@ -42,7 +42,7 @@

 static __inline__ int scm_check_creds(struct ucred *creds)
 {
- if ((creds->pid == current->tgid || capable(CAP_SYS_ADMIN)) &&
+ if ((creds->pid == task_tgid_vnr(current) || capable(CAP_SYS_ADMIN)) &&
      ((creds->uid == current->uid || creds->uid == current->euid ||
        creds->uid == current->suid) || capable(CAP_SETUID)) &&
      ((creds->gid == current->gid || creds->gid == current->egid ||
diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c
index fc12ba5..261c344 100644
--- a/net/unix/af_unix.c
+++ b/net/unix/af_unix.c
@@ -456,7 +456,7 @@ static int unix_listen(struct socket *so
  sk->sk_max_ack_backlog = backlog;
  sk->sk_state  = TCP_LISTEN;
  /* set credentials so connect can copy them */
- sk->sk_peercred.pid = current->tgid;
+ sk->sk_peercred.pid = task_tgid_vnr(current);
  sk->sk_peercred.uid = current->euid;
  sk->sk_peercred.gid = current->egid;
  err = 0;
```

```
@@ -1069,7 +1069,7 @@ restart:
  unix_peer(newsk) = sk;
  newsk->sk_state  = TCP_ESTABLISHED;
  newsk->sk_type  = sk->sk_type;
- newsk->sk_peercred.pid = current->tgid;
+ newsk->sk_peercred.pid = task_tgid_vnr(current);
  newsk->sk_peercred.uid = current->euid;
  newsk->sk_peercred.gid = current->egid;
  newu = unix_sk(newsk);
@@ -1133,7 +1133,7 @@ static int unix_socketpair(struct socket
  sock_hold(skb);
  unix_peer(ska)=skb;
  unix_peer(skb)=ska;
- ska->sk_peercred.pid = skb->sk_peercred.pid = current->tgid;
+ ska->sk_peercred.pid = skb->sk_peercred.pid = task_tgid_vnr(current);
  ska->sk_peercred.uid = skb->sk_peercred.uid = current->euid;
  ska->sk_peercred.gid = skb->sk_peercred.gid = current->egid;
```