Subject: Re: [Lxc-devel] [PATCH] pidspace: is_init()
Posted by dev on Wed, 09 Aug 2006 08:27:35 GMT
View Forum Message <> Reply to Message

Why not change comparisons of pid == 0 then as well?

One more comment. It would be nice to remove direct access to ->pid
in all code with some wrappers like get_task_pid(). Probably in next patches.
In this case, "pid" field on task_struct can be renamed to something else
and no other such comparisons will be lost occasionally.

Kirill

> Andrew,
>
> I had sent this patch as an RFC a few days ago and recieved no
> objections/comments. It is just a cleanup and does not change
> any functionality.
>
> Regards,
>
> Suka
>
> ---
>
> This is an updated version of Eric Biederman's is_init() patch.
> (http://lkml.org/lkml/2006/2/6/280). It applies cleanly to 2.6.18-rc3
> and replaces a few more instances of ->pid == 1 with is_init().
>
> Further, is_init() checks pid and thus removes dependency on Eric's
> other patches for now.
>
> Eric's original description:
>
> There are a lot of places in the kernel where we test for init
> because we give it special properties. Most significantly init
> must not die. This results in code all over the kernel test
> ->pid == 1.
>
> Introduce is_init to capture this case.
>
> With multiple pid spaces for all of the cases affected we are
> looking for only the first process on the system, not some other
> process that has pid == 1.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> Cc: Dave Hansen <haveblue@us.ibm.com>

> Cc: Serge Hallyn <serue@us.ibm.com>
> Cc: Cedric Le Goater <clg@fr.ibm.com>
> Cc: lxc-devel@lists.sourceforge.net
>
>
> arch/alpha/mm/fault.c | 2 +-
> arch/arm/mm/fault.c | 2 +-
> arch/arm26/mm/fault.c | 2 +-
> arch/i386/lib/usercopy.c | 2 +-
> arch/i386/mm/fault.c | 2 +-
> arch/ia64/mm/fault.c | 2 +-
> arch/m32r/mm/fault.c | 2 +-
> arch/m68k/mm/fault.c | 2 +-
> arch/mips/mm/fault.c | 2 +-
> arch/powerpc/mm/fault.c | 2 +-
> arch/powerpc/platforms/pseries/ras.c | 2 +-
> arch/ppc/kernel/traps.c | 2 +-
> arch/ppc/mm/fault.c | 2 +-
> arch/s390/mm/fault.c | 2 +-
> arch/sh/mm/fault.c | 2 +-
> arch/sh64/mm/fault.c | 6 ++++---
> arch/um/kernel/trap.c | 2 +-
> arch/x86_64/mm/fault.c | 4 ++--
> arch/xtensa/mm/fault.c | 2 +-
> drivers/char/sysrq.c | 2 +-
> include/linux/sched.h | 10 ++++++++++
> kernel/capability.c | 2 +-
> kernel/cpuset.c | 2 +-
> kernel/exit.c | 2 +-
> kernel/kexec.c | 2 +-
> kernel/ptrace.c | 1 +
> kernel/sysctl.c | 2 +-
> mm/oom_kill.c | 6 ++++---
> security/commoncap.c | 2 +-
> security/seclvl.c | 9 +++++++----
> 30 files changed, 48 insertions(+), 36 deletions(-)
>
> Index: linux-2.6.18-rc3/arch/alpha/mm/fault.c
> ===
> --- linux-2.6.18-rc3.orig/arch/alpha/mm/fault.c 2006-08-08 14:10:11.000000000 -0700
> +++ linux-2.6.18-rc3/arch/alpha/mm/fault.c 2006-08-08 17:23:20.000000000 -0700
> @@ -193,7 +193,7 @@ do_page_fault(unsigned long address, uns
>   /* We ran out of memory, or some other thing happened to us that
>   made us unable to handle the page fault gracefully. */
>   out_of_memory:
> - if (current->pid == 1) {
> + if (is_init(current)) {
>   yield();

>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: linux-2.6.18-rc3/arch/arm/mm/fault.c
> === =======
> --- linux-2.6.18-rc3.orig/arch/arm/mm/fault.c 2006-08-08 14:10:13.000000000 -0700
> +++ linux-2.6.18-rc3/arch/arm/mm/fault.c 2006-08-08 17:23:20.000000000 -0700
> @@ -197,7 +197,7 @@ survive:
>   return fault;
>   }
>
> - if (tsk->pid != 1)
> + if (!is_init(tsk))
>   goto out;
>
>   /*
> Index: linux-2.6.18-rc3/arch/arm26/mm/fault.c
> === =======
> --- linux-2.6.18-rc3.orig/arch/arm26/mm/fault.c 2006-08-08 14:10:00.000000000 -0700
> +++ linux-2.6.18-rc3/arch/arm26/mm/fault.c 2006-08-08 17:23:20.000000000 -0700
> @@ -185,7 +185,7 @@ survive:
>   }
>
>   fault = -3; /* out of memory */
> - if (tsk->pid != 1)
> + if (!is_init(tsk))
>   goto out;
>
>   /*
> Index: linux-2.6.18-rc3/arch/i386/lib/usercopy.c
> === =======
> --- linux-2.6.18-rc3.orig/arch/i386/lib/usercopy.c 2006-08-08 14:09:42.000000000 -0700
> +++ linux-2.6.18-rc3/arch/i386/lib/usercopy.c 2006-08-08 17:23:20.000000000 -0700
> @@ -739,7 +739,7 @@ survive:
>   retval = get_user_pages(current, current->mm,
>   (unsigned long)to, 1, 1, 0, &pg, NULL);
>
> -   if (retval == -ENOMEM && current->pid == 1) {
> +   if (retval == -ENOMEM && is_init(current)) {
>   up_read(&current->mm->mmap_sem);
>   blk_congestion_wait(WRITE, HZ/50);
>   goto survive;
> Index: linux-2.6.18-rc3/arch/i386/mm/fault.c
> === =======
> --- linux-2.6.18-rc3.orig/arch/i386/mm/fault.c 2006-08-08 14:09:42.000000000 -0700
> +++ linux-2.6.18-rc3/arch/i386/mm/fault.c 2006-08-08 17:23:20.000000000 -0700
> @@ -598,7 +598,7 @@ no_context:
>   */
> out_of_memory:

> up_read(&mm->mmap_sem);
> - if (tsk->pid == 1) {
> + if (is_init(tsk)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: linux-2.6.18-rc3/arch/ia64/mm/fault.c
> ===
> --- linux-2.6.18-rc3.orig/arch/ia64/mm/fault.c 2006-08-08 14:10:06.000000000 -0700
> +++ linux-2.6.18-rc3/arch/ia64/mm/fault.c 2006-08-08 17:23:20.000000000 -0700
> @@ -278,7 +278,7 @@ ia64_do_page_fault (unsigned long addres
>
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (current->pid == 1) {
> + if (is_init(current)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: linux-2.6.18-rc3/arch/m32r/mm/fault.c
> ===
> --- linux-2.6.18-rc3.orig/arch/m32r/mm/fault.c 2006-08-08 14:10:21.000000000 -0700
> +++ linux-2.6.18-rc3/arch/m32r/mm/fault.c 2006-08-08 17:23:20.000000000 -0700
> @@ -299,7 +299,7 @@ no_context:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (tsk->pid == 1) {
> + if (is_init(tsk)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: linux-2.6.18-rc3/arch/m68k/mm/fault.c
> ===
> --- linux-2.6.18-rc3.orig/arch/m68k/mm/fault.c 2006-08-08 14:10:02.000000000 -0700
> +++ linux-2.6.18-rc3/arch/m68k/mm/fault.c 2006-08-08 17:23:20.000000000 -0700
> @@ -181,7 +181,7 @@ good_area:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (current->pid == 1) {
> + if (is_init(current)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: linux-2.6.18-rc3/arch/mips/mm/fault.c
> ===
> --- linux-2.6.18-rc3.orig/arch/mips/mm/fault.c 2006-08-08 14:09:53.000000000 -0700

> +++ linux-2.6.18-rc3/arch/mips/mm/fault.c 2006-08-08 17:23:20.000000000 -0700
> @@ -171,7 +171,7 @@ no_context:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (tsk->pid == 1) {
> + if (is_init(tsk)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: linux-2.6.18-rc3/arch/powerpc/mm/fault.c
> ===
> --- linux-2.6.18-rc3.orig/arch/powerpc/mm/fault.c 2006-08-08 14:10:22.000000000 -0700
> +++ linux-2.6.18-rc3/arch/powerpc/mm/fault.c 2006-08-08 17:23:21.000000000 -0700
> @@ -386,7 +386,7 @@ bad_area_nosemaphore:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (current->pid == 1) {
> + if (is_init(current)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: linux-2.6.18-rc3/arch/powerpc/platforms/pseries/ras.c
> ===
> --- linux-2.6.18-rc3.orig/arch/powerpc/platforms/pseries/ras.c 2006-08-08 14:10:23.000000000
-0700
> +++ linux-2.6.18-rc3/arch/powerpc/platforms/pseries/ras.c 2006-08-08 17:23:21.000000000
-0700
> @@ -337,7 +337,7 @@ static int recover_mce(struct pt_regs *r
> err->disposition == RTAS_DISP_NOT_RECOVERED &&
> err->target == RTAS_TARGET_MEMORY &&
> err->type == RTAS_TYPE_ECC_UNCORR &&
> - !(current->pid == 0 || current->pid == 1)) {
> + !(current->pid == 0 || is_init(current))) {
> /* Kill off a user process with an ECC error */
> printk(KERN_ERR "MCE: uncorrectable ecc error for pid %d\n",
> current->pid);
> Index: linux-2.6.18-rc3/arch/ppc/kernel/traps.c
> ===
> --- linux-2.6.18-rc3.orig/arch/ppc/kernel/traps.c 2006-08-08 14:09:48.000000000 -0700
> +++ linux-2.6.18-rc3/arch/ppc/kernel/traps.c 2006-08-08 17:23:21.000000000 -0700
> @@ -119,7 +119,7 @@ void _exception(int signr, struct pt_reg
> * generate the same exception over and over again and we get
> * nowhere. Better to kill it and let the kernel panic.
> */
> - if (current->pid == 1) {
> + if (is_init(current)) {

>   __sighandler_t handler;
>
>   spin_lock_irq(&current->sighand->siglock);
> Index: linux-2.6.18-rc3/arch/ppc/mm/fault.c
> === =======
> --- linux-2.6.18-rc3.orig/arch/ppc/mm/fault.c 2006-08-08 14:09:47.000000000 -0700
> +++ linux-2.6.18-rc3/arch/ppc/mm/fault.c 2006-08-08 17:23:21.000000000 -0700
> @@ -291,7 +291,7 @@ bad_area:
>   */
> out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (current->pid == 1) {
> + if (is_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: linux-2.6.18-rc3/arch/s390/mm/fault.c
> === =======
> --- linux-2.6.18-rc3.orig/arch/s390/mm/fault.c 2006-08-08 14:10:20.000000000 -0700
> +++ linux-2.6.18-rc3/arch/s390/mm/fault.c 2006-08-08 17:23:21.000000000 -0700
> @@ -315,7 +315,7 @@ no_context:
> */
> out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (tsk->pid == 1) {
> + if (is_init(tsk)) {
>   yield();
>   goto survive;
>   }
> Index: linux-2.6.18-rc3/arch/sh/mm/fault.c
> === =======
> --- linux-2.6.18-rc3.orig/arch/sh/mm/fault.c 2006-08-08 14:09:59.000000000 -0700
> +++ linux-2.6.18-rc3/arch/sh/mm/fault.c 2006-08-08 17:23:21.000000000 -0700
> @@ -160,7 +160,7 @@ no_context:
>   */
> out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (current->pid == 1) {
> + if (is_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: linux-2.6.18-rc3/arch/sh64/mm/fault.c
> === =======
> --- linux-2.6.18-rc3.orig/arch/sh64/mm/fault.c 2006-08-08 14:10:21.000000000 -0700
> +++ linux-2.6.18-rc3/arch/sh64/mm/fault.c 2006-08-08 17:23:21.000000000 -0700
> @@ -277,7 +277,7 @@ bad_area:
>   show_regs(regs);

> #endif
>   }
> - if (tsk->pid == 1) {
> + if (is_init(tsk)) {
>   panic("INIT had user mode bad_area\n");
>   }
>   tsk->thread.address = address;
> @@ -319,14 +319,14 @@ no_context:
>   * us unable to handle the page fault gracefully.
>   */
> out_of_memory:
> - if (current->pid == 1) {
> + if (is_init(current)) {
>   panic("INIT out of memory\n");
>   yield();
>   goto survive;
>   }
>   printk("fault:Out of memory\n");
>   up_read(&mm->mmap_sem);
> - if (current->pid == 1) {
> + if (is_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: linux-2.6.18-rc3/arch/um/kernel/trap.c
> ===
> --- linux-2.6.18-rc3.orig/arch/um/kernel/trap.c 2006-08-08 14:09:50.000000000 -0700
> +++ linux-2.6.18-rc3/arch/um/kernel/trap.c 2006-08-08 17:23:21.000000000 -0700
> @@ -120,7 +120,7 @@ out_nosemaphore:
>   * us unable to handle the page fault gracefully.
>   */
> out_of_memory:
> - if (current->pid == 1) {
> + if (is_init(current)) {
>   up_read(&mm->mmap_sem);
>   yield();
>   down_read(&mm->mmap_sem);
> Index: linux-2.6.18-rc3/arch/x86_64/mm/fault.c
> ===
> --- linux-2.6.18-rc3.orig/arch/x86_64/mm/fault.c 2006-08-08 14:10:18.000000000 -0700
> +++ linux-2.6.18-rc3/arch/x86_64/mm/fault.c 2006-08-08 17:23:21.000000000 -0700
> @@ -250,7 +250,7 @@ static int is_errata93(struct pt_regs *r
>
> int unhandled_signal(struct task_struct *tsk, int sig)
> {
> - if (tsk->pid == 1)
> + if (is_init(tsk))
>   return 1;

```
>   if (tsk->ptrace & PT_PTRACED)
>    return 0;
> @@ -586,7 +586,7 @@ no_context:
>   */
>  out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (current->pid == 1) {
> + if (is_init(current)) {
>    yield();
>    goto again;
>   }
> Index: linux-2.6.18-rc3/arch/xtensa/mm/fault.c
>  ===================================================================
> --- linux-2.6.18-rc3.orig/arch/xtensa/mm/fault.c 2006-08-08 14:10:21.000000000 -0700
> +++ linux-2.6.18-rc3/arch/xtensa/mm/fault.c 2006-08-08 17:23:21.000000000 -0700
> @@ -144,7 +144,7 @@ bad_area:
>    */
>  out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (current->pid == 1) {
> + if (is_init(current)) {
>    yield();
>    down_read(&mm->mmap_sem);
>    goto survive;
> Index: linux-2.6.18-rc3/drivers/char/sysrq.c
>  ===================================================================
> --- linux-2.6.18-rc3.orig/drivers/char/sysrq.c 2006-08-08 14:09:12.000000000 -0700
> +++ linux-2.6.18-rc3/drivers/char/sysrq.c 2006-08-08 17:23:21.000000000 -0700
> @@ -208,7 +208,7 @@ static void send_sig_all(int sig)
>   struct task_struct *p;
>
>   for_each_process(p) {
> -  if (p->mm && p->pid != 1)
> +  if (p->mm && !is_init(p))
>     /* Not swapper, init nor kernel thread */
>     force_sig(sig, p);
>   }
> Index: linux-2.6.18-rc3/include/linux/sched.h
>  ===================================================================
> --- linux-2.6.18-rc3.orig/include/linux/sched.h 2006-08-08 14:10:27.000000000 -0700
> +++ linux-2.6.18-rc3/include/linux/sched.h 2006-08-08 17:23:21.000000000 -0700
> @@ -1017,6 +1017,16 @@ static inline int pid_alive(struct task_
>   return p->pids[PIDTYPE_PID].pid != NULL;
>  }
>
> +/**
> + * is_init - check if a task structure is the first user space
> + *      task the kernel created.
```

```
> + * @p: Task structure to be checked.
> + */
> +static inline int is_init(struct task_struct *tsk)
> +{
> + return tsk->pid == 1;
> +}
> +
>  extern void free_task(struct task_struct *tsk);
>  #define get_task_struct(tsk) do { atomic_inc(&(tsk)->usage); } while(0)
>
> Index: linux-2.6.18-rc3/kernel/capability.c
>  ===================================================================
> --- linux-2.6.18-rc3.orig/kernel/capability.c 2006-08-08 14:11:38.000000000 -0700
> +++ linux-2.6.18-rc3/kernel/capability.c 2006-08-08 17:23:21.000000000 -0700
> @@ -133,7 +133,7 @@ static inline int cap_set_all(kernel_cap
>       int found = 0;
>
>       do_each_thread(g, target) {
> -             if (target == current || target->pid == 1)
> +              if (target == current || is_init(target))
>                       continue;
>               found = 1;
>        if (security_capset_check(target, effective, inheritable,
> Index: linux-2.6.18-rc3/kernel/cpuset.c
>  ===================================================================
> --- linux-2.6.18-rc3.orig/kernel/cpuset.c 2006-08-08 14:11:39.000000000 -0700
> +++ linux-2.6.18-rc3/kernel/cpuset.c 2006-08-08 17:23:21.000000000 -0700
> @@ -240,7 +240,7 @@ static struct super_block *cpuset_sb;
>   * A cpuset can only be deleted if both its 'count' of using tasks
>   * is zero, and its list of 'children' cpusets is empty.  Since all
>   * tasks in the system use _some_ cpuset, and since there is always at
> - * least one task in the system (init, pid == 1), therefore, top_cpuset
> + * least one task in the system (init), therefore, top_cpuset
>   * always has either children cpusets and/or using tasks.  So we don't
>   * need a special hack to ensure that top_cpuset cannot be deleted.
>   *
> Index: linux-2.6.18-rc3/kernel/exit.c
>  ===================================================================
> --- linux-2.6.18-rc3.orig/kernel/exit.c 2006-08-08 14:11:39.000000000 -0700
> +++ linux-2.6.18-rc3/kernel/exit.c 2006-08-08 17:23:21.000000000 -0700
> @@ -219,7 +219,7 @@ static int will_become_orphaned_pgrp(int
>   do_each_task_pid(pgrp, PIDTYPE_PGID, p) {
>    if (p == ignored_task
>      || p->exit_state
> -    || p->real_parent->pid == 1)
> +    || is_init(p->real_parent))
>     continue;
>    if (process_group(p->real_parent) != pgrp
```

> && p->real_parent->signal->session == p->signal->session) {
> Index: linux-2.6.18-rc3/kernel/kexec.c
> === =======
> --- linux-2.6.18-rc3.orig/kernel/kexec.c 2006-08-08 14:11:39.000000000 -0700
> +++ linux-2.6.18-rc3/kernel/kexec.c 2006-08-08 17:23:21.000000000 -0700
> @@ -40,7 +40,7 @@ struct resource crashk_res = {
>
> int kexec_should_crash(struct task_struct *p)
> {
> - if (in_interrupt() || !p->pid || p->pid == 1 || panic_on_oops)
> + if (in_interrupt() || !p->pid || is_init(p) || panic_on_oops)
> return 1;
> return 0;
> }
> Index: linux-2.6.18-rc3/kernel/ptrace.c
> === =======
> --- linux-2.6.18-rc3.orig/kernel/ptrace.c 2006-08-08 14:11:39.000000000 -0700
> +++ linux-2.6.18-rc3/kernel/ptrace.c 2006-08-08 17:23:21.000000000 -0700
> @@ -494,6 +494,7 @@ struct task_struct *ptrace_get_task_stru
> child = find_task_by_pid(pid);
> if (child)
> get_task_struct(child);
> +
> read_unlock(&tasklist_lock);
> if (!child)
> return ERR_PTR(-ESRCH);
> Index: linux-2.6.18-rc3/kernel/sysctl.c
> === =======
> --- linux-2.6.18-rc3.orig/kernel/sysctl.c 2006-08-08 17:22:17.000000000 -0700
> +++ linux-2.6.18-rc3/kernel/sysctl.c 2006-08-08 17:23:21.000000000 -0700
> @@ -1867,7 +1867,7 @@ int proc_dointvec_bset(ctl_table *table,
> return -EPERM;
> }
>
> - op = (current->pid == 1) ? OP_SET : OP_AND;
> + op = is_init(current) ? OP_SET : OP_AND;
> return do_proc_dointvec(table,write,filp,buffer,lenp,ppos,
> do_proc_dointvec_bset_conv,&op);
> }
> Index: linux-2.6.18-rc3/mm/oom_kill.c
> === =======
> --- linux-2.6.18-rc3.orig/mm/oom_kill.c 2006-08-08 14:11:40.000000000 -0700
> +++ linux-2.6.18-rc3/mm/oom_kill.c 2006-08-08 17:23:21.000000000 -0700
> @@ -191,8 +191,8 @@ static struct task_struct *select_bad_pr
> unsigned long points;
> int releasing;
>
> - /* skip the init task with pid == 1 */

```
> -  if (p->pid == 1)
> +  /* skip the init task */
> +  if (is_init(p))
>     continue;
>    if (p->oomkilladj == OOM_DISABLE)
>     continue;
> @@ -227,7 +227,7 @@ static struct task_struct *select_bad_pr
>   */
>  static void __oom_kill_task(struct task_struct *p, const char *message)
>  {
> - if (p->pid == 1) {
> + if (is_init(p)) {
>    WARN_ON(1);
>    printk(KERN_WARNING "tried to kill init!\n");
>    return;
> Index: linux-2.6.18-rc3/security/commoncap.c
>  ===================================================================
> --- linux-2.6.18-rc3.orig/security/commoncap.c 2006-08-08 14:11:39.000000000 -0700
> +++ linux-2.6.18-rc3/security/commoncap.c 2006-08-08 17:23:21.000000000 -0700
> @@ -169,7 +169,7 @@ void cap_bprm_apply_creds (struct linux_
>   /* For init, we want to retain the capabilities set
>    * in the init_task struct. Thus we skip the usual
>    * capability rules */
> - if (current->pid != 1) {
> + if (!is_init(current)) {
>    current->cap_permitted = new_permitted;
>    current->cap_effective =
>        cap_intersect (new_permitted, bprm->cap_effective);
> Index: linux-2.6.18-rc3/security/seclvl.c
>  ===================================================================
> --- linux-2.6.18-rc3.orig/security/seclvl.c 2006-08-08 14:11:39.000000000 -0700
> +++ linux-2.6.18-rc3/security/seclvl.c 2006-08-08 17:23:21.000000000 -0700
> @@ -287,7 +287,7 @@ static struct file_operations passwd_fil
>   */
>  static int seclvl_ptrace(struct task_struct *parent, struct task_struct *child)
>  {
> - if (seclvl >= 0 && child->pid == 1) {
> + if (seclvl >= 0 && is_init(child)) {
>    seclvl_printk(1, KERN_WARNING, "Attempt to ptrace "
>          "the init process dissallowed in "
>          "secure level %d\n", seclvl);
> @@ -305,7 +305,7 @@ static int seclvl_capable(struct task_st
>   int rc = 0;
>
>   /* init can do anything it wants */
> - if (tsk->pid == 1)
> + if (is_init(tsk))
>    return 0;
```

>
> if (seclvl > 0) {
> @@ -413,7 +413,8 @@ static void seclvl_bd_release(struct ino
> static int
> seclvl_inode_permission(struct inode *inode, int mask, struct nameidata *nd)
> {
> - if (current->pid != 1 && S_ISBLK(inode->i_mode) && (mask & MAY_WRITE)) {
> + if (!is_init(current)
> + && S_ISBLK(inode->i_mode) && (mask & MAY_WRITE)) {
>   switch (seclvl) {
>   case 2:
>   seclvl_printk(1, KERN_WARNING, "Write to block device "
> @@ -465,7 +466,7 @@ static void seclvl_file_free_security(st
> */
> static int seclvl_umount(struct vfsmount *mnt, int flags)
> {
> - if (current->pid != 1 && seclvl == 2) {
> + if (!is_init(current) && seclvl == 2) {
>   seclvl_printk(1, KERN_WARNING, "Attempt to unmount in secure "
>   "level %d\n", seclvl);
>   return -EPERM;
>
> -- -------------
> Using Tomcat but need to do more? Need to support web services, security?
> Get stuff done quickly with pre-integrated technology to make your job easier
> Download IBM WebSphere Application Server v.1.0.1 based on Apache Geronimo
> http://sel.as-us.falkag.net/sel?cmd=lnk&kid=120709&b id=263057&dat=121642
> _____
> Lxc-devel mailing list
> Lxc-devel@lists.sourceforge.net
> https://lists.sourceforge.net/lists/listinfo/lxc-devel
>