Subject: [Lxc-devel] + pidspace-is_init.patch added to -mm tree
Posted by Andrew Morton on Wed, 09 Aug 2006 07:15:20 GMT
View Forum Message <> Reply to Message

The patch titled

    pidspace: is_init()

has been added to the -mm tree. Its filename is

    pidspace-is_init.patch

See http://www.zip.com.au/~akpm/linux/patches/stuff/added-to-mm.txt to find
out what to do about this

Subject: pidspace: is_init()
From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

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>
Acked-by: Paul Mackerras <paulus@samba.org>
Signed-off-by: Andrew Morton <akpm@osdl.org>

```
 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                      |    2 +-
 security/commoncap.c               |    2 +-
 29 files changed, 41 insertions(+), 30 deletions(-)

diff -puN arch/alpha/mm/fault.c~pidspace-is_init arch/alpha/mm/fault.c
--- a/arch/alpha/mm/fault.c~pidspace-is_init
+++ a/arch/alpha/mm/fault.c
@@ -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;
diff -puN arch/arm26/mm/fault.c~pidspace-is_init arch/arm26/mm/fault.c
--- a/arch/arm26/mm/fault.c~pidspace-is_init
+++ a/arch/arm26/mm/fault.c
@@ -185,7 +185,7 @@ survive:
```

```
  }

  fault = -3; /* out of memory */
- if (tsk->pid != 1)
+ if (!is_init(tsk))
    goto out;

  /*
diff -puN arch/arm/mm/fault.c~pidspace-is_init arch/arm/mm/fault.c
--- a/arch/arm/mm/fault.c~pidspace-is_init
+++ a/arch/arm/mm/fault.c
@@ -197,7 +197,7 @@ survive:
    return fault;
  }

- if (tsk->pid != 1)
+ if (!is_init(tsk))
    goto out;

  /*
diff -puN arch/i386/lib/usercopy.c~pidspace-is_init arch/i386/lib/usercopy.c
--- a/arch/i386/lib/usercopy.c~pidspace-is_init
+++ a/arch/i386/lib/usercopy.c
@@ -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;
diff -puN arch/i386/mm/fault.c~pidspace-is_init arch/i386/mm/fault.c
--- a/arch/i386/mm/fault.c~pidspace-is_init
+++ a/arch/i386/mm/fault.c
@@ -596,7 +596,7 @@ 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;
diff -puN arch/ia64/mm/fault.c~pidspace-is_init arch/ia64/mm/fault.c
--- a/arch/ia64/mm/fault.c~pidspace-is_init
+++ a/arch/ia64/mm/fault.c
@@ -280,7 +280,7 @@ 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;
diff -puN arch/m32r/mm/fault.c~pidspace-is_init arch/m32r/mm/fault.c
--- a/arch/m32r/mm/fault.c~pidspace-is_init
+++ a/arch/m32r/mm/fault.c
@@ -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;
diff -puN arch/m68k/mm/fault.c~pidspace-is_init arch/m68k/mm/fault.c
--- a/arch/m68k/mm/fault.c~pidspace-is_init
+++ a/arch/m68k/mm/fault.c
@@ -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;
diff -puN arch/mips/mm/fault.c~pidspace-is_init arch/mips/mm/fault.c
--- a/arch/mips/mm/fault.c~pidspace-is_init
+++ a/arch/mips/mm/fault.c
@@ -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;
diff -puN arch/powerpc/mm/fault.c~pidspace-is_init arch/powerpc/mm/fault.c
--- a/arch/powerpc/mm/fault.c~pidspace-is_init
+++ a/arch/powerpc/mm/fault.c
@@ -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;
diff -puN arch/powerpc/platforms/pseries/ras.c~pidspace-is_init
arch/powerpc/platforms/pseries/ras.c
--- a/arch/powerpc/platforms/pseries/ras.c~pidspace-is_init
+++ a/arch/powerpc/platforms/pseries/ras.c
@@ -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);
diff -puN arch/ppc/kernel/traps.c~pidspace-is_init arch/ppc/kernel/traps.c
--- a/arch/ppc/kernel/traps.c~pidspace-is_init
+++ a/arch/ppc/kernel/traps.c
@@ -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);
diff -puN arch/ppc/mm/fault.c~pidspace-is_init arch/ppc/mm/fault.c
--- a/arch/ppc/mm/fault.c~pidspace-is_init
+++ a/arch/ppc/mm/fault.c
@@ -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;
diff -puN arch/s390/mm/fault.c~pidspace-is_init arch/s390/mm/fault.c
--- a/arch/s390/mm/fault.c~pidspace-is_init
+++ a/arch/s390/mm/fault.c
```

```
@@ -353,7 +353,7 @@ no_context:
 */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (tsk->pid == 1) {
+ if (is_init(tsk)) {
   yield();
   goto survive;
  }
diff -puN arch/sh64/mm/fault.c~pidspace-is_init arch/sh64/mm/fault.c
--- a/arch/sh64/mm/fault.c~pidspace-is_init
+++ a/arch/sh64/mm/fault.c
@@ -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;
diff -puN arch/sh/mm/fault.c~pidspace-is_init arch/sh/mm/fault.c
--- a/arch/sh/mm/fault.c~pidspace-is_init
+++ a/arch/sh/mm/fault.c
@@ -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;
diff -puN arch/um/kernel/trap.c~pidspace-is_init arch/um/kernel/trap.c
--- a/arch/um/kernel/trap.c~pidspace-is_init
+++ a/arch/um/kernel/trap.c
@@ -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);
diff -puN arch/x86_64/mm/fault.c~pidspace-is_init arch/x86_64/mm/fault.c
--- a/arch/x86_64/mm/fault.c~pidspace-is_init
+++ a/arch/x86_64/mm/fault.c
@@ -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;
@@ -591,7 +591,7 @@ no_context:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (current->pid == 1) {
+ if (is_init(current)) {
   yield();
   goto again;
  }
diff -puN arch/xtensa/mm/fault.c~pidspace-is_init arch/xtensa/mm/fault.c
--- a/arch/xtensa/mm/fault.c~pidspace-is_init
+++ a/arch/xtensa/mm/fault.c
@@ -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;
diff -puN drivers/char/sysrq.c~pidspace-is_init drivers/char/sysrq.c
```

```
--- a/drivers/char/sysrq.c~pidspace-is_init
+++ a/drivers/char/sysrq.c
@@ -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);
  }
diff -puN include/linux/sched.h~pidspace-is_init include/linux/sched.h
--- a/include/linux/sched.h~pidspace-is_init
+++ a/include/linux/sched.h
@@ -1029,6 +1029,16 @@ 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)

diff -puN kernel/capability.c~pidspace-is_init kernel/capability.c
--- a/kernel/capability.c~pidspace-is_init
+++ a/kernel/capability.c
@@ -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,
diff -puN kernel/cpuset.c~pidspace-is_init kernel/cpuset.c
--- a/kernel/cpuset.c~pidspace-is_init
+++ a/kernel/cpuset.c
@@ -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.
  *
diff -puN kernel/exit.c~pidspace-is_init kernel/exit.c
--- a/kernel/exit.c~pidspace-is_init
+++ a/kernel/exit.c
@@ -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) {
diff -puN kernel/kexec.c~pidspace-is_init kernel/kexec.c
--- a/kernel/kexec.c~pidspace-is_init
+++ a/kernel/kexec.c
@@ -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;
 }
diff -puN kernel/ptrace.c~pidspace-is_init kernel/ptrace.c
--- a/kernel/ptrace.c~pidspace-is_init
+++ a/kernel/ptrace.c
@@ -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);
diff -puN kernel/sysctl.c~pidspace-is_init kernel/sysctl.c
--- a/kernel/sysctl.c~pidspace-is_init
+++ a/kernel/sysctl.c
@@ -1912,7 +1912,7 @@ 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);
 }
diff -puN mm/oom_kill.c~pidspace-is_init mm/oom_kill.c
--- a/mm/oom_kill.c~pidspace-is_init
+++ a/mm/oom_kill.c
@@ -256,7 +256,7 @@ 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;
diff -puN security/commoncap.c~pidspace-is_init security/commoncap.c
--- a/security/commoncap.c~pidspace-is_init
+++ a/security/commoncap.c
@@ -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);
diff -puN /dev/null /dev/null
_
```

Patches currently in -mm which might be from sukadev@us.ibm.com are

kthread-airoc.patch
kthread-airoc-race-fix.patch
kthread-drivers-base-firmware_classc.patch
pidspace-is_init.patch

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&bid=263057&dat=121642
_____
Lxc-devel mailing list