Subject: [PATCH 6/6] Define is_global_init() and is_container_init()
Posted by Sukadev Bhattiprolu on Fri, 06 Jul 2007 05:56:43 GMT
View Forum Message <> Reply to Message

Subject: [PATCH 6/6] Define is_global_init() and is_container_init().

From: Serge E. Hallyn <serue@us.ibm.com>

is_init() is an ambiguous name for the pid==1 check. Split it into
is_global_init() and is_container_init().

A container init has it's tsk->pid == 1.

A global init also has it's tsk->pid == 1 and it's active pid namespace
is the init_pid_ns. But rather than check the active pid namespace,
compare the task structure with 'init_pid_ns.child_reaper', which is
initialized during boot to the /sbin/init process and never changes.

Changelog:

2.6.22-rc4-mm2-pidns1:
- Use 'init_pid_ns.child_reaper' to determine if a given task is the
  global init (/sbin/init) process. This would improve performance
  and remove dependence on the task_pid().

2.6.21-mm2-pidns2:

- [Sukadev Bhattiprolu] Changed is_container_init() calls in {powerpc,
  ppc,avr32}/traps.c for the _exception() call to is_global_init().
  This way, we kill only the container if the container's init has a
  bug rather than force a kernel panic.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
---
arch/alpha/mm/fault.c | 2 +-
arch/arm/mm/fault.c | 2 +-
arch/arm26/mm/fault.c | 2 +-
arch/avr32/kernel/traps.c | 2 +-
arch/avr32/mm/fault.c | 6 +++---
arch/i386/lib/usercopy.c | 2 +-
arch/i386/mm/fault.c | 2 +-
arch/ia64/mm/fault.c | 2 +-
arch/m68k/mm/fault.c | 2 +-
arch/mips/mm/fault.c | 2 +-
arch/powerpc/kernel/traps.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/lib/uaccess_pt.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               |    2 +-
 arch/xtensa/mm/fault.c               |    2 +-
 drivers/char/sysrq.c                 |    2 +-
 include/linux/sched.h                |   12 ++++++++++--
 kernel/capability.c                  |    3 ++-
 kernel/exit.c                        |    2 +-
 kernel/kexec.c                       |    2 +-
 kernel/pid.c                         |    5 +++++
 kernel/signal.c                      |    2 +-
 kernel/sysctl.c                      |    2 +-
 mm/oom_kill.c                        |    4 ++--
 security/commoncap.c                 |    3 ++-
 32 files changed, 52 insertions(+), 37 deletions(-)

Index: lx26-22-rc6-mm1/include/linux/sched.h
===================================================================
--- lx26-22-rc6-mm1.orig/include/linux/sched.h 2007-07-05 18:57:34.000000000 -0700
+++ lx26-22-rc6-mm1/include/linux/sched.h 2007-07-05 18:59:06.000000000 -0700
@@ -1248,12 +1248,20 @@ static inline int pid_alive(struct task_
 }

 /**
- * is_init - check if a task structure is init
+ * is_global_init - check if a task structure is init
  * @tsk: Task structure to be checked.
  *
  * Check if a task structure is the first user space task the kernel created.
+ *
+ * TODO: We should inline this function after some cleanups in pid_namespace.h
+ */
+extern int is_global_init(struct task_struct *tsk);
+
+/*
+ * is_container_init:
+ * check whether in the task is init in it's own pid namespace.
  */
-static inline int is_init(struct task_struct *tsk)
+static inline int is_container_init(struct task_struct *tsk)
 {
  return tsk->pid == 1;
```

```
 }
Index: lx26-22-rc6-mm1/kernel/pid.c
===================================================================
--- lx26-22-rc6-mm1.orig/kernel/pid.c 2007-07-05 18:53:48.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/pid.c 2007-07-05 18:59:06.000000000 -0700
@@ -71,6 +71,11 @@ struct pid_namespace init_pid_ns = {
  .child_reaper = &init_task
 };

+int is_global_init(struct task_struct *tsk)
+{
+ return tsk == init_pid_ns.child_reaper;
+}
+
 /*
  * Note: disable interrupts while the pidmap_lock is held as an
  * interrupt might come in and do read_lock(&tasklist_lock).
Index: lx26-22-rc6-mm1/arch/alpha/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/alpha/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/alpha/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -192,7 +192,7 @@ 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 (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/arm/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/arm/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/arm/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -197,7 +197,7 @@ survive:
   return fault;
  }

- if (!is_init(tsk))
+ if (!is_global_init(tsk))
   goto out;

  /*
Index: lx26-22-rc6-mm1/arch/arm26/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/arm26/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/arm26/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -185,7 +185,7 @@ survive:
```

```
  }

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

  /*
Index: lx26-22-rc6-mm1/arch/i386/lib/usercopy.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/i386/lib/usercopy.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/i386/lib/usercopy.c 2007-07-05 18:59:06.000000000 -0700
@@ -748,7 +748,7 @@ survive:
    retval = get_user_pages(current, current->mm,
      (unsigned long )to, 1, 1, 0, &pg, NULL);

-   if (retval == -ENOMEM && is_init(current)) {
+   if (retval == -ENOMEM && is_global_init(current)) {
     up_read(&current->mm->mmap_sem);
     congestion_wait(WRITE, HZ/50);
     goto survive;
Index: lx26-22-rc6-mm1/arch/i386/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/i386/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/i386/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -595,7 +595,7 @@ no_context:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(tsk)) {
+ if (is_global_init(tsk)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/ia64/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/ia64/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/ia64/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -270,7 +270,7 @@ ia64_do_page_fault (unsigned long addres

   out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/m68k/mm/fault.c
```

```
===================================================================
--- lx26-22-rc6-mm1.orig/arch/m68k/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/m68k/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -181,7 +181,7 @@ good_area:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/mips/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/mips/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/mips/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -174,7 +174,7 @@ no_context:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(tsk)) {
+ if (is_global_init(tsk)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/powerpc/kernel/traps.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/powerpc/kernel/traps.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/powerpc/kernel/traps.c 2007-07-05 18:59:06.000000000 -0700
@@ -191,7 +191,7 @@ 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 (is_init(current)) {
+ if (is_global_init(current)) {
   __sighandler_t handler;

   spin_lock_irq(&current->sighand->siglock);
Index: lx26-22-rc6-mm1/arch/powerpc/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/powerpc/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/powerpc/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -374,7 +374,7 @@ bad_area_nosemaphore:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
```

```
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/powerpc/platforms/pseries/ras.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/powerpc/platforms/pseries/ras.c 2007-07-05 18:53:42.000000000
-0700
+++ lx26-22-rc6-mm1/arch/powerpc/platforms/pseries/ras.c 2007-07-05 18:59:06.000000000
-0700
@@ -332,7 +332,7 @@ 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 || is_init(current))) {
+     !(current->pid == 0 || is_global_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: lx26-22-rc6-mm1/arch/ppc/kernel/traps.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/ppc/kernel/traps.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/ppc/kernel/traps.c 2007-07-05 18:59:06.000000000 -0700
@@ -121,7 +121,7 @@ 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 (is_init(current)) {
+ if (is_global_init(current)) {
   __sighandler_t handler;

   spin_lock_irq(&current->sighand->siglock);
Index: lx26-22-rc6-mm1/arch/ppc/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/ppc/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/ppc/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -292,7 +292,7 @@ bad_area:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/s390/lib/uaccess_pt.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/s390/lib/uaccess_pt.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/s390/lib/uaccess_pt.c 2007-07-05 18:59:06.000000000 -0700
```

```
@@ -65,7 +65,7 @@ out:

 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/s390/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/s390/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/s390/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -211,7 +211,7 @@ static int do_out_of_memory(struct pt_re
  struct mm_struct *mm = tsk->mm;

  up_read(&mm->mmap_sem);
- if (is_init(tsk)) {
+ if (is_global_init(tsk)) {
   yield();
   down_read(&mm->mmap_sem);
   return 1;
Index: lx26-22-rc6-mm1/arch/sh/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/sh/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/sh/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -191,7 +191,7 @@ no_context:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/sh64/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/sh64/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/sh64/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -276,7 +276,7 @@ bad_area:
    show_regs(regs);
 #endif
   }
-  if (is_init(tsk)) {
+  if (is_global_init(tsk)) {
    panic("INIT had user mode bad_area\n");
   }
   tsk->thread.address = address;
```

```
@@ -318,14 +318,14 @@ no_context:
  * us unable to handle the page fault gracefully.
  */
 out_of_memory:
- if (is_init(current)) {
+ if (is_global_init(current)) {
   panic("INIT out of memory\n");
   yield();
   goto survive;
  }
  printk("fault:Out of memory\n");
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/arch/um/kernel/trap.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/um/kernel/trap.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/um/kernel/trap.c 2007-07-05 18:59:06.000000000 -0700
@@ -120,7 +120,7 @@ out_nosemaphore:
  * us unable to handle the page fault gracefully.
  */
 out_of_memory:
- if (is_init(current)) {
+ if (is_global_init(current)) {
   up_read(&mm->mmap_sem);
   yield();
   down_read(&mm->mmap_sem);
Index: lx26-22-rc6-mm1/arch/x86_64/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/x86_64/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/x86_64/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -558,7 +558,7 @@ no_context:
  */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   goto again;
  }
Index: lx26-22-rc6-mm1/arch/xtensa/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/xtensa/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/xtensa/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -144,7 +144,7 @@ bad_area:
```

```
   */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
Index: lx26-22-rc6-mm1/drivers/char/sysrq.c
===================================================================
--- lx26-22-rc6-mm1.orig/drivers/char/sysrq.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/drivers/char/sysrq.c 2007-07-05 18:59:06.000000000 -0700
@@ -250,7 +250,7 @@ static void send_sig_all(int sig)
  struct task_struct *p;

  for_each_process(p) {
-  if (p->mm && !is_init(p))
+  if (p->mm && !is_global_init(p))
    /* Not swapper, init nor kernel thread */
    force_sig(sig, p);
  }
Index: lx26-22-rc6-mm1/kernel/capability.c
===================================================================
--- lx26-22-rc6-mm1.orig/kernel/capability.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/capability.c 2007-07-05 18:59:06.000000000 -0700
@@ -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 */
@@ -135,7 +136,7 @@ static inline int cap_set_all(kernel_cap
      int found = 0;

      do_each_thread(g, target) {
-             if (target == current || is_init(target))
+             if (target == current || is_container_init(target))
                     continue;
             found = 1;
       if (security_capset_check(target, effective, inheritable,
Index: lx26-22-rc6-mm1/kernel/exit.c
===================================================================
--- lx26-22-rc6-mm1.orig/kernel/exit.c 2007-07-05 18:56:53.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/exit.c 2007-07-05 18:59:06.000000000 -0700
@@ -231,7 +231,7 @@ static int will_become_orphaned_pgrp(str
  do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
```

```
   if (p == ignored_task
     || p->exit_state
-    || is_init(p->real_parent))
+    || is_global_init(p->real_parent))
     continue;
   if (task_pgrp(p->real_parent) != pgrp &&
       task_session(p->real_parent) == task_session(p)) {
Index: lx26-22-rc6-mm1/kernel/kexec.c
===================================================================
--- lx26-22-rc6-mm1.orig/kernel/kexec.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/kexec.c 2007-07-05 18:59:06.000000000 -0700
@@ -42,7 +42,7 @@ struct resource crashk_res = {

 int kexec_should_crash(struct task_struct *p)
 {
- if (in_interrupt() || !p->pid || is_init(p) || panic_on_oops)
+ if (in_interrupt() || !p->pid || is_global_init(p) || panic_on_oops)
   return 1;
  return 0;
 }
Index: lx26-22-rc6-mm1/kernel/sysctl.c
===================================================================
--- lx26-22-rc6-mm1.orig/kernel/sysctl.c 2007-07-05 18:54:29.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/sysctl.c 2007-07-05 18:59:06.000000000 -0700
@@ -1924,7 +1924,7 @@ int proc_dointvec_bset(ctl_table *table,
   return -EPERM;
  }

- op = is_init(current) ? OP_SET : OP_AND;
+ op = is_global_init(current) ? OP_SET : OP_AND;
  return do_proc_dointvec(table,write,filp,buffer,lenp,ppos,
     do_proc_dointvec_bset_conv,&op);
 }
Index: lx26-22-rc6-mm1/mm/oom_kill.c
===================================================================
--- lx26-22-rc6-mm1.orig/mm/oom_kill.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/mm/oom_kill.c 2007-07-05 18:59:06.000000000 -0700
@@ -222,7 +222,7 @@ static struct task_struct *select_bad_pr
   if (!p->mm)
    continue;
   /* skip the init task */
-  if (is_init(p))
+  if (is_global_init(p))
    continue;

   /*
@@ -275,7 +275,7 @@ static struct task_struct *select_bad_pr
  */
```

```
 static void __oom_kill_task(struct task_struct *p, int verbose)
 {
- if (is_init(p)) {
+ if (is_global_init(p)) {
   WARN_ON(1);
   printk(KERN_WARNING "tried to kill init!\n");
   return;
Index: lx26-22-rc6-mm1/security/commoncap.c
===================================================================
--- lx26-22-rc6-mm1.orig/security/commoncap.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/security/commoncap.c 2007-07-05 18:59:06.000000000 -0700
@@ -23,6 +23,7 @@
 #include <linux/xattr.h>
 #include <linux/hugetlb.h>
 #include <linux/mount.h>
+#include <linux/sched.h>

 int cap_netlink_send(struct sock *sk, struct sk_buff *skb)
 {
@@ -261,7 +262,7 @@ 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 (!is_init(current)) {
+ if (!is_global_init(current)) {
   current->cap_permitted = new_permitted;
   current->cap_effective =
       cap_intersect (new_permitted, bprm->cap_effective);
Index: lx26-22-rc6-mm1/arch/avr32/kernel/traps.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/avr32/kernel/traps.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/avr32/kernel/traps.c 2007-07-05 18:59:06.000000000 -0700
@@ -89,7 +89,7 @@ void _exception(long signr, struct pt_re
   * generate the same exception over and over again and we get
   * nowhere.  Better to kill it and let the kernel panic.
   */
- if (is_init(current)) {
+ if (is_global_init(current)) {
   __sighandler_t handler;

   spin_lock_irq(&current->sighand->siglock);
Index: lx26-22-rc6-mm1/arch/avr32/mm/fault.c
===================================================================
--- lx26-22-rc6-mm1.orig/arch/avr32/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
+++ lx26-22-rc6-mm1/arch/avr32/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
@@ -161,7 +161,7 @@ bad_area:
   if (exception_trace && printk_ratelimit())
    printk("%s%s[%d]: segfault at %08lx pc %08lx "
```

```
          "sp %08lx ecr %lu\n",
-         is_init(tsk) ? KERN_EMERG : KERN_INFO,
+         is_global_init(tsk) ? KERN_EMERG : KERN_INFO,
          tsk->comm, tsk->pid, address, regs->pc,
          regs->sp, ecr);
   _exception(SIGSEGV, regs, code, address);
@@ -210,7 +210,7 @@ no_context:
   */
 out_of_memory:
  up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
@@ -232,7 +232,7 @@ do_sigbus:
  if (exception_trace)
   printk("%s%s[%d]: bus error at %08lx pc %08lx "
          "sp %08lx ecr %lu\n",
-         is_init(tsk) ? KERN_EMERG : KERN_INFO,
+         is_global_init(tsk) ? KERN_EMERG : KERN_INFO,
          tsk->comm, tsk->pid, address, regs->pc,
          regs->sp, ecr);

Index: lx26-22-rc6-mm1/kernel/signal.c
===================================================================
--- lx26-22-rc6-mm1.orig/kernel/signal.c 2007-07-05 18:56:53.000000000 -0700
+++ lx26-22-rc6-mm1/kernel/signal.c 2007-07-05 18:59:06.000000000 -0700
@@ -257,7 +257,7 @@ flush_signal_handlers(struct task_struct

 int unhandled_signal(struct task_struct *tsk, int sig)
 {
- if (is_init(tsk))
+ if (is_global_init(tsk))
   return 1;
  if (tsk->ptrace & PT_PTRACED)
   return 0;
```


Subject: Re: [PATCH 6/6] Define is_global_init() and is_container_init()
Posted by Pavel Emelianov on Fri, 13 Jul 2007 05:24:21 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:

> Subject: [PATCH 6/6] Define is_global_init() and is_container_init().
>
> From: Serge E. Hallyn <serue@us.ibm.com>
>
>
> is_init() is an ambiguous name for the pid==1 check. Split it into
> is_global_init() and is_container_init().
>
> A container init has it's tsk->pid == 1.
>
> A global init also has it's tsk->pid == 1 and it's active pid namespace
> is the init_pid_ns. But rather than check the active pid namespace,
> compare the task structure with 'init_pid_ns.child_reaper', which is
> initialized during boot to the /sbin/init process and never changes.
>
> Changelog:
>
> 2.6.22-rc4-mm2-pidns1:
> - Use 'init_pid_ns.child_reaper' to determine if a given task is the
>   global init (/sbin/init) process. This would improve performance
>   and remove dependence on the task_pid().
>
> 2.6.21-mm2-pidns2:
>
> - [Sukadev Bhattiprolu] Changed is_container_init() calls in {powerpc,
>   ppc,avr32}/traps.c for the _exception() call to is_global_init().
>   This way, we kill only the container if the container's init has a
>   bug rather than force a kernel panic.
>
> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

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

> ---
> arch/alpha/mm/fault.c | 2 +-
> arch/arm/mm/fault.c | 2 +-
> arch/arm26/mm/fault.c | 2 +-
> arch/avr32/kernel/traps.c | 2 +-
> arch/avr32/mm/fault.c | 6 ++++---
> arch/i386/lib/usercopy.c | 2 +-
> arch/i386/mm/fault.c | 2 +-
> arch/ia64/mm/fault.c | 2 +-
> arch/m68k/mm/fault.c | 2 +-
> arch/mips/mm/fault.c | 2 +-
> arch/powerpc/kernel/traps.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/lib/uaccess_pt.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 | 2 +-
> arch/xtensa/mm/fault.c | 2 +-
> drivers/char/sysrq.c | 2 +-
> include/linux/sched.h | 12 ++++++++++--
> kernel/capability.c | 3 ++-
> kernel/exit.c | 2 +-
> kernel/kexec.c | 2 +-
> kernel/pid.c | 5 +++++
> kernel/signal.c | 2 +-
> kernel/sysctl.c | 2 +-
> mm/oom_kill.c | 4 ++--
> security/commoncap.c | 3 ++-
> 32 files changed, 52 insertions(+), 37 deletions(-)
>
> Index: lx26-22-rc6-mm1/include/linux/sched.h
> ===
> --- lx26-22-rc6-mm1.orig/include/linux/sched.h 2007-07-05 18:57:34.000000000 -0700
> +++ lx26-22-rc6-mm1/include/linux/sched.h 2007-07-05 18:59:06.000000000 -0700
> @@ -1248,12 +1248,20 @@ static inline int pid_alive(struct task_
> }
>
> /**
> - * is_init - check if a task structure is init
> + * is_global_init - check if a task structure is init
> * @tsk: Task structure to be checked.
> *
> * Check if a task structure is the first user space task the kernel created.
> + *
> + * TODO: We should inline this function after some cleanups in pid_namespace.h
> */
> +extern int is_global_init(struct task_struct *tsk);
> +
> +/*
> + * is_container_init:
> + * check whether in the task is init in it's own pid namespace.
> */
> -static inline int is_init(struct task_struct *tsk)
> +static inline int is_container_init(struct task_struct *tsk)
> {
> return tsk->pid == 1;
> }

```
> Index: lx26-22-rc6-mm1/kernel/pid.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/kernel/pid.c 2007-07-05 18:53:48.000000000 -0700
> +++ lx26-22-rc6-mm1/kernel/pid.c 2007-07-05 18:59:06.000000000 -0700
> @@ -71,6 +71,11 @@ struct pid_namespace init_pid_ns = {
>   .child_reaper = &init_task
>  };
>
> +int is_global_init(struct task_struct *tsk)
> +{
> + return tsk == init_pid_ns.child_reaper;
> +}
> +
>  /*
>   * Note: disable interrupts while the pidmap_lock is held as an
>   * interrupt might come in and do read_lock(&tasklist_lock).
> Index: lx26-22-rc6-mm1/arch/alpha/mm/fault.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/arch/alpha/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/alpha/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -192,7 +192,7 @@ 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 (is_init(current)) {
> + if (is_global_init(current)) {
>    yield();
>    down_read(&mm->mmap_sem);
>    goto survive;
> Index: lx26-22-rc6-mm1/arch/arm/mm/fault.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/arch/arm/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/arm/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -197,7 +197,7 @@ survive:
>    return fault;
>   }
>
> - if (!is_init(tsk))
> + if (!is_global_init(tsk))
>    goto out;
>
>   /*
> Index: lx26-22-rc6-mm1/arch/arm26/mm/fault.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/arch/arm26/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/arm26/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -185,7 +185,7 @@ survive:
>   }
```

>
> fault = -3; /* out of memory */
> - if (!is_init(tsk))
> + if (!is_global_init(tsk))
> goto out;
>
> /*
> Index: lx26-22-rc6-mm1/arch/i386/lib/usercopy.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/i386/lib/usercopy.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/i386/lib/usercopy.c 2007-07-05 18:59:06.000000000 -0700
> @@ -748,7 +748,7 @@ survive:
> retval = get_user_pages(current, current->mm,
> (unsigned long)to, 1, 1, 0, &pg, NULL);
>
> - if (retval == -ENOMEM && is_init(current)) {
> + if (retval == -ENOMEM && is_global_init(current)) {
> up_read(&current->mm->mmap_sem);
> congestion_wait(WRITE, HZ/50);
> goto survive;
> Index: lx26-22-rc6-mm1/arch/i386/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/i386/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/i386/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -595,7 +595,7 @@ no_context:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(tsk)) {
> + if (is_global_init(tsk)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: lx26-22-rc6-mm1/arch/ia64/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/ia64/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/ia64/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -270,7 +270,7 @@ ia64_do_page_fault (unsigned long addres
>
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: lx26-22-rc6-mm1/arch/m68k/mm/fault.c
> ===

> --- lx26-22-rc6-mm1.orig/arch/m68k/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/m68k/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -181,7 +181,7 @@ good_area:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: lx26-22-rc6-mm1/arch/mips/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/mips/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/mips/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -174,7 +174,7 @@ no_context:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(tsk)) {
> + if (is_global_init(tsk)) {
> yield();
> down_read(&mm->mmap_sem);
> goto survive;
> Index: lx26-22-rc6-mm1/arch/powerpc/kernel/traps.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/powerpc/kernel/traps.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/powerpc/kernel/traps.c 2007-07-05 18:59:06.000000000 -0700
> @@ -191,7 +191,7 @@ 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 (is_init(current)) {
> + if (is_global_init(current)) {
> __sighandler_t handler;
>
> spin_lock_irq(&current->sighand->siglock);
> Index: lx26-22-rc6-mm1/arch/powerpc/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/powerpc/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/powerpc/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -374,7 +374,7 @@ bad_area_nosemaphore:
> */
> out_of_memory:
> up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
> yield();

>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: lx26-22-rc6-mm1/arch/powerpc/platforms/pseries/ras.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/powerpc/platforms/pseries/ras.c 2007-07-05 18:53:42.000000000
-0700
> +++ lx26-22-rc6-mm1/arch/powerpc/platforms/pseries/ras.c 2007-07-05 18:59:06.000000000
-0700
> @@ -332,7 +332,7 @@ 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 || is_init(current))) {
> +   !(current->pid == 0 || is_global_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: lx26-22-rc6-mm1/arch/ppc/kernel/traps.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/ppc/kernel/traps.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/ppc/kernel/traps.c 2007-07-05 18:59:06.000000000 -0700
> @@ -121,7 +121,7 @@ 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 (is_init(current)) {
> + if (is_global_init(current)) {
>   __sighandler_t handler;
>
>   spin_lock_irq(&current->sighand->siglock);
> Index: lx26-22-rc6-mm1/arch/ppc/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/ppc/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/ppc/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -292,7 +292,7 @@ bad_area:
>   */
> out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: lx26-22-rc6-mm1/arch/s390/lib/uaccess_pt.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/s390/lib/uaccess_pt.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/s390/lib/uaccess_pt.c 2007-07-05 18:59:06.000000000 -0700
> @@ -65,7 +65,7 @@ out:

>
> out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: lx26-22-rc6-mm1/arch/s390/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/s390/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/s390/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -211,7 +211,7 @@ static int do_out_of_memory(struct pt_re
>   struct mm_struct *mm = tsk->mm;
>
>   up_read(&mm->mmap_sem);
> - if (is_init(tsk)) {
> + if (is_global_init(tsk)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   return 1;
> Index: lx26-22-rc6-mm1/arch/sh/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/sh/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/sh/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -191,7 +191,7 @@ no_context:
>   */
> out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: lx26-22-rc6-mm1/arch/sh64/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/sh64/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/sh64/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -276,7 +276,7 @@ bad_area:
>   show_regs(regs);
> #endif
>   }
> - if (is_init(tsk)) {
> + if (is_global_init(tsk)) {
>   panic("INIT had user mode bad_area\n");
>   }
>   tsk->thread.address = address;
> @@ -318,14 +318,14 @@ no_context:

>   * us unable to handle the page fault gracefully.
>   */
> out_of_memory:
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   panic("INIT out of memory\n");
>   yield();
>   goto survive;
>   }
>   printk("fault:Out of memory\n");
>   up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   yield();
>   down_read(&mm->mmap_sem);
>   goto survive;
> Index: lx26-22-rc6-mm1/arch/um/kernel/trap.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/um/kernel/trap.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/um/kernel/trap.c 2007-07-05 18:59:06.000000000 -0700
> @@ -120,7 +120,7 @@ out_nosemaphore:
>   * us unable to handle the page fault gracefully.
>   */
> out_of_memory:
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   up_read(&mm->mmap_sem);
>   yield();
>   down_read(&mm->mmap_sem);
> Index: lx26-22-rc6-mm1/arch/x86_64/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/x86_64/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/x86_64/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -558,7 +558,7 @@ no_context:
>   */
> out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>   yield();
>   goto again;
>   }
> Index: lx26-22-rc6-mm1/arch/xtensa/mm/fault.c
> ===
> --- lx26-22-rc6-mm1.orig/arch/xtensa/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/xtensa/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -144,7 +144,7 @@ bad_area:
>   */

```
>  out_of_memory:
>   up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>    yield();
>    down_read(&mm->mmap_sem);
>    goto survive;
> Index: lx26-22-rc6-mm1/drivers/char/sysrq.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/drivers/char/sysrq.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/drivers/char/sysrq.c 2007-07-05 18:59:06.000000000 -0700
> @@ -250,7 +250,7 @@ static void send_sig_all(int sig)
>   struct task_struct *p;
>
>   for_each_process(p) {
> -  if (p->mm && !is_init(p))
> +  if (p->mm && !is_global_init(p))
>     /* Not swapper, init nor kernel thread */
>     force_sig(sig, p);
>   }
> Index: lx26-22-rc6-mm1/kernel/capability.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/kernel/capability.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/kernel/capability.c 2007-07-05 18:59:06.000000000 -0700
> @@ -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 */
> @@ -135,7 +136,7 @@ static inline int cap_set_all(kernel_cap
>       int found = 0;
>
>       do_each_thread(g, target) {
> -             if (target == current || is_init(target))
> +              if (target == current || is_container_init(target))
>                       continue;
>               found = 1;
>        if (security_capset_check(target, effective, inheritable,
> Index: lx26-22-rc6-mm1/kernel/exit.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/kernel/exit.c 2007-07-05 18:56:53.000000000 -0700
> +++ lx26-22-rc6-mm1/kernel/exit.c 2007-07-05 18:59:06.000000000 -0700
> @@ -231,7 +231,7 @@ static int will_become_orphaned_pgrp(str
>   do_each_pid_task(pgrp, PIDTYPE_PGID, p) {
>    if (p == ignored_task
```

```
>      || p->exit_state
> -    || is_init(p->real_parent))
> +    || is_global_init(p->real_parent))
>      continue;
>    if (task_pgrp(p->real_parent) != pgrp &&
>        task_session(p->real_parent) == task_session(p)) {
> Index: lx26-22-rc6-mm1/kernel/kexec.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/kernel/kexec.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/kernel/kexec.c 2007-07-05 18:59:06.000000000 -0700
> @@ -42,7 +42,7 @@ struct resource crashk_res = {
>
>  int kexec_should_crash(struct task_struct *p)
>  {
> - if (in_interrupt() || !p->pid || is_init(p) || panic_on_oops)
> + if (in_interrupt() || !p->pid || is_global_init(p) || panic_on_oops)
>    return 1;
>   return 0;
>  }
> Index: lx26-22-rc6-mm1/kernel/sysctl.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/kernel/sysctl.c 2007-07-05 18:54:29.000000000 -0700
> +++ lx26-22-rc6-mm1/kernel/sysctl.c 2007-07-05 18:59:06.000000000 -0700
> @@ -1924,7 +1924,7 @@ int proc_dointvec_bset(ctl_table *table,
>    return -EPERM;
>   }
>
> - op = is_init(current) ? OP_SET : OP_AND;
> + op = is_global_init(current) ? OP_SET : OP_AND;
>   return do_proc_dointvec(table,write,filp,buffer,lenp,ppos,
>      do_proc_dointvec_bset_conv,&op);
>  }
> Index: lx26-22-rc6-mm1/mm/oom_kill.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/mm/oom_kill.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/mm/oom_kill.c 2007-07-05 18:59:06.000000000 -0700
> @@ -222,7 +222,7 @@ static struct task_struct *select_bad_pr
>    if (!p->mm)
>     continue;
>    /* skip the init task */
> -  if (is_init(p))
> +  if (is_global_init(p))
>     continue;
>
>    /*
> @@ -275,7 +275,7 @@ static struct task_struct *select_bad_pr
>   */
>  static void __oom_kill_task(struct task_struct *p, int verbose)
```

```
>  {
> - if (is_init(p)) {
> + if (is_global_init(p)) {
>    WARN_ON(1);
>    printk(KERN_WARNING "tried to kill init!\n");
>    return;
> Index: lx26-22-rc6-mm1/security/commoncap.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/security/commoncap.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/security/commoncap.c 2007-07-05 18:59:06.000000000 -0700
> @@ -23,6 +23,7 @@
>  #include <linux/xattr.h>
>  #include <linux/hugetlb.h>
>  #include <linux/mount.h>
> +#include <linux/sched.h>
>
>  int cap_netlink_send(struct sock *sk, struct sk_buff *skb)
>  {
> @@ -261,7 +262,7 @@ 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 (!is_init(current)) {
> + if (!is_global_init(current)) {
>    current->cap_permitted = new_permitted;
>    current->cap_effective =
>        cap_intersect (new_permitted, bprm->cap_effective);
> Index: lx26-22-rc6-mm1/arch/avr32/kernel/traps.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/arch/avr32/kernel/traps.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/avr32/kernel/traps.c 2007-07-05 18:59:06.000000000 -0700
> @@ -89,7 +89,7 @@ void _exception(long signr, struct pt_re
>    * generate the same exception over and over again and we get
>    * nowhere.  Better to kill it and let the kernel panic.
>    */
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>    __sighandler_t handler;
>
>    spin_lock_irq(&current->sighand->siglock);
> Index: lx26-22-rc6-mm1/arch/avr32/mm/fault.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/arch/avr32/mm/fault.c 2007-07-05 18:53:42.000000000 -0700
> +++ lx26-22-rc6-mm1/arch/avr32/mm/fault.c 2007-07-05 18:59:06.000000000 -0700
> @@ -161,7 +161,7 @@ bad_area:
>    if (exception_trace && printk_ratelimit())
>     printk("%s%s[%d]: segfault at %08lx pc %08lx "
>            "sp %08lx ecr %lu\n",
```

```
> -          is_init(tsk) ? KERN_EMERG : KERN_INFO,
> +          is_global_init(tsk) ? KERN_EMERG : KERN_INFO,
>            tsk->comm, tsk->pid, address, regs->pc,
>            regs->sp, ecr);
>    _exception(SIGSEGV, regs, code, address);
> @@ -210,7 +210,7 @@ no_context:
>     */
>  out_of_memory:
>    up_read(&mm->mmap_sem);
> - if (is_init(current)) {
> + if (is_global_init(current)) {
>     yield();
>     down_read(&mm->mmap_sem);
>     goto survive;
> @@ -232,7 +232,7 @@ do_sigbus:
>    if (exception_trace)
>     printk("%s%s[%d]: bus error at %08lx pc %08lx "
>            "sp %08lx ecr %lu\n",
> -          is_init(tsk) ? KERN_EMERG : KERN_INFO,
> +          is_global_init(tsk) ? KERN_EMERG : KERN_INFO,
>            tsk->comm, tsk->pid, address, regs->pc,
>            regs->sp, ecr);
>
> Index: lx26-22-rc6-mm1/kernel/signal.c
> ===================================================================
> --- lx26-22-rc6-mm1.orig/kernel/signal.c 2007-07-05 18:56:53.000000000 -0700
> +++ lx26-22-rc6-mm1/kernel/signal.c 2007-07-05 18:59:06.000000000 -0700
> @@ -257,7 +257,7 @@ flush_signal_handlers(struct task_struct
>
>  int unhandled_signal(struct task_struct *tsk, int sig)
>  {
> - if (is_init(tsk))
> + if (is_global_init(tsk))
>     return 1;
>    if (tsk->ptrace & PT_PTRACED)
>     return 0;
> _______________________________________________
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
>

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