
Subject: Re: Screamm.. commit f400e198b2ed26ce55b22a1412ded0896e7516ac
Posted by [serue](#) on Thu, 29 Mar 2007 15:48:41 GMT
[View Forum Message](#) <> [Reply to Message](#)

Quoting Eric W. Biederman (ebiederm@xmission.com):

> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> >> > Where the latter is needed in, for instance, kernel/capability.c.
> >>
> >> Yes.
> >>
> >> I think more clear cut examples could be made. It isn't clear to me
> >> why we skip pid == 1 in kernel/capability.c
> >
> > Because the capset(2) manpage says:
> >
> > For capset(), pid can also be: -1, meaning
> > perform the change on all threads except the caller and
> > init(8);
>
> Which they copied from the kill(2) manpage. So they are just preserving
> the existing definition of which processes -1 applies to.
>
> The single unix/posix standard says:
>
> If pid is -1, sig shall be sent to all processes (excluding an
> unspecified set of system processes) for which the process has
> permission to send that signal.
>
> So I'm still curious why we decided not to send to pid == 1. But
> that is clearly the way things are defined to work in linux.
>
> So I guess that makes the capsetall case a good example after all.
> It is skipping pid == 1 because that is what it means. And in that
> context I suspect makes a great deal of sense to perform the skip
> by testing for pid == 1. Because that is what we really mean.

Ok, so please tear this apart as a first shot at splitting up is_init.

Note that near as I can tell, some callers of is_init() have been
changed back to pid==1 checks in Linus' tree. I'm ignoring those
for now.

-serge

From: "Serge E. Hallyn" <serue@us.ibm.com>
Subject: [PATCH] pid namespaces: define is_global_init()

is_init() is an ambiguous name for the pid==1 check. Split it into is_global_init() and is_container_init(). is_container_init() cannot yet be defined because it will need to use pid namespace features which do not yet exist in mainline. It is currently the same as is_global_init(), but the comment above describes how it should be defined.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

```

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/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     |  4 +++-
arch/xtensa/mm/fault.c     |  2 +-
drivers/char/sysrq.c       |  2 +-
include/linux/pid_namespace.h |  3 +++
include/linux/sched.h      |  7 ++++++-
kernel/capability.c        |  3 ++-
kernel/exit.c              |  2 +-
kernel/kexec.c             |  2 +-
kernel/sched.c             | 14 ++++++++
kernel/sysctl.c            |  2 +-
mm/oom_kill.c              |  4 ++--
security/commoncap.c       |  2 +-
31 files changed, 55 insertions(+), 34 deletions(-)

```

```

d0bb7398482666dbfda6e4e6efd5c773dea0d586
diff --git a/arch/alpha/mm/fault.c b/arch/alpha/mm/fault.c
index 8aa9db8..44e10aa 100644
--- a/arch/alpha/mm/fault.c
+++ b/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 (is_init(current)) {
+ if (is_global_init(current)) {
   yield();
   down_read(&mm->mmap_sem);
   goto survive;
diff --git a/arch/arm/mm/fault.c b/arch/arm/mm/fault.c
index 9fd6d2e..2b3647a 100644
--- a/arch/arm/mm/fault.c
+++ b/arch/arm/mm/fault.c
@@ -198,7 +198,7 @@ survive:
   return fault;
 }

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

/*
diff --git a/arch/arm26/mm/fault.c b/arch/arm26/mm/fault.c
index 93c0cee..1b22594 100644
--- a/arch/arm26/mm/fault.c
+++ b/arch/arm26/mm/fault.c
@@ -185,7 +185,7 @@ survive:
 }

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

/*
diff --git a/arch/i386/lib/usercopy.c b/arch/i386/lib/usercopy.c
index d22cfc9..6f0e3fc 100644
--- a/arch/i386/lib/usercopy.c
+++ b/arch/i386/lib/usercopy.c
@@ -740,7 +740,7 @@ 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;

```

```

diff --git a/arch/i386/mm/fault.c b/arch/i386/mm/fault.c
index b8c4e25..1dd8ca8 100644
--- a/arch/i386/mm/fault.c
+++ b/arch/i386/mm/fault.c
@@ -561,7 +561,7 @@ 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;
diff --git a/arch/ia64/mm/fault.c b/arch/ia64/mm/fault.c
index 59f3ab9..c0f73a1 100644
--- a/arch/ia64/mm/fault.c
+++ b/arch/ia64/mm/fault.c
@@ -280,7 +280,7 @@ ia64_do_page_fault (unsigned long address)
    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;
diff --git a/arch/m32r/mm/fault.c b/arch/m32r/mm/fault.c
index 037d58e..75dce7f 100644
--- a/arch/m32r/mm/fault.c
+++ b/arch/m32r/mm/fault.c
@@ -273,7 +273,7 @@ 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;
diff --git a/arch/m68k/mm/fault.c b/arch/m68k/mm/fault.c
index 2adbeb1..a9a08b0 100644
--- a/arch/m68k/mm/fault.c
+++ b/arch/m68k/mm/fault.c
@@ -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;
diff --git a/arch/mips/mm/fault.c b/arch/mips/mm/fault.c
index 6f90e7e..2b20e4d 100644
--- a/arch/mips/mm/fault.c
+++ b/arch/mips/mm/fault.c
@@ -175,7 +175,7 @@ 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;
diff --git a/arch/powerpc/kernel/traps.c b/arch/powerpc/kernel/traps.c
index 17724fb..939e723 100644
--- a/arch/powerpc/kernel/traps.c
+++ b/arch/powerpc/kernel/traps.c
@@ -172,7 +172,7 @@ 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_container_init(current, NULL)) {
    __sighandler_t handler;

    spin_lock_irq(&current->sigband->siglock);
diff --git a/arch/powerpc/mm/fault.c b/arch/powerpc/mm/fault.c
index 03aeb3a..691f212 100644
--- a/arch/powerpc/mm/fault.c
+++ b/arch/powerpc/mm/fault.c
@@ -386,7 +386,7 @@ 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;
diff --git a/arch/powerpc/platforms/pseries/ras.c b/arch/powerpc/platforms/pseries/ras.c
index edc0388..3f6bd53 100644
--- a/arch/powerpc/platforms/pseries/ras.c
+++ b/arch/powerpc/platforms/pseries/ras.c
@@ -333,7 +333,7 @@ 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);
diff --git a/arch/ppc/kernel/traps.c b/arch/ppc/kernel/traps.c
index 810f7aa..80a5fa4 100644
--- a/arch/ppc/kernel/traps.c
+++ b/arch/ppc/kernel/traps.c
@@ -120,7 +120,7 @@ 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_container_init(current, NULL)) {
__sighandler_t handler;

spin_lock_irq(&current->sigand->siglock);
diff --git a/arch/ppc/mm/fault.c b/arch/ppc/mm/fault.c
index 465f451..a370002 100644
--- a/arch/ppc/mm/fault.c
+++ b/arch/ppc/mm/fault.c
@@ -291,7 +291,7 @@ 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;
diff --git a/arch/s390/lib/uaccess_pt.c b/arch/s390/lib/uaccess_pt.c
index 6318167..6d7e99d 100644
--- a/arch/s390/lib/uaccess_pt.c
+++ b/arch/s390/lib/uaccess_pt.c
@@ -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;
diff --git a/arch/s390/mm/fault.c b/arch/s390/mm/fault.c

```

```

index 7462aeb..29eaa0a 100644
--- a/arch/s390/mm/fault.c
+++ b/arch/s390/mm/fault.c
@@ -424,7 +424,7 @@ 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;
diff --git a/arch/sh/mm/fault.c b/arch/sh/mm/fault.c
index fa5d7f0..7abc41f 100644
--- a/arch/sh/mm/fault.c
+++ b/arch/sh/mm/fault.c
@@ -198,7 +198,7 @@ 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;
diff --git a/arch/sh64/mm/fault.c b/arch/sh64/mm/fault.c
index 4f72ab3..6abf1f9 100644
--- a/arch/sh64/mm/fault.c
+++ b/arch/sh64/mm/fault.c
@@ -277,7 +277,7 @@ 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;
@@ -319,14 +319,14 @@ 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;
diff --git a/arch/um/kernel/trap.c b/arch/um/kernel/trap.c
index 26f15c4..bd14cb8 100644
--- a/arch/um/kernel/trap.c
+++ b/arch/um/kernel/trap.c
@@ -119,7 +119,7 @@ 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);
diff --git a/arch/x86_64/mm/fault.c b/arch/x86_64/mm/fault.c
index 6ada723..7cc9621 100644
--- a/arch/x86_64/mm/fault.c
+++ b/arch/x86_64/mm/fault.c
@@ -223,7 +223,7 @@ static int is_errata93(struct pt_regs *r

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;
@@ -557,7 +557,7 @@ no_context:
    */
out_of_memory:
    up_read(&mm->mmap_sem);
- if (is_init(current)) {
+ if (is_global_init(current)) {
    yield();
    goto again;
}
diff --git a/arch/xtensa/mm/fault.c b/arch/xtensa/mm/fault.c
index 3dc6f2f..9c39270 100644
--- a/arch/xtensa/mm/fault.c
+++ b/arch/xtensa/mm/fault.c
@@ -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;
diff --git a/drivers/char/sysrq.c b/drivers/char/sysrq.c
index 1d8c4ae..181fb72 100644
--- a/drivers/char/sysrq.c
+++ b/drivers/char/sysrq.c
@@ -251,7 +251,7 @@ 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);
    }
diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index 2833806..c7bf870 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -24,6 +24,9 @@ struct pid_namespace {

extern struct pid_namespace init_pid_ns;

+/* This is a temporary workaround until pid namespaces are hooked in correctly */
+#define task_primary_pid_ns(tsk) (&init_pid_ns)
+
static inline void get_pid_ns(struct pid_namespace *ns)
{
    kref_get(&ns->kref);
diff --git a/include/linux/sched.h b/include/linux/sched.h
index 49fe299..6db9aff 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1108,16 +1108,19 @@ 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.
+ */

```

```

-static inline int is_init(struct task_struct *tsk)
+static inline int is_global_init(struct task_struct *tsk)
{
    return tsk->pid == 1;
}

+struct pid_namespace;
+extern int is_container_init(struct task_struct *tsk, struct pid_namespace *ns);
+
extern struct pid *cad_pid;

extern void free_task(struct task_struct *tsk);
diff --git a/kernel/capability.c b/kernel/capability.c
index c8d3c77..8c8cd39 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 */
@@ -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, task_primary_pid_ns(current)))
            continue;
        found = 1;
        if (security_capset_check(target, effective, inheritable,
diff --git a/kernel/exit.c b/kernel/exit.c
index f132349..83fff76 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -218,7 +218,7 @@ 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)) {
diff --git a/kernel/kexec.c b/kernel/kexec.c
index 2a59c8a..0eb8518 100644

```

```

--- a/kernel/kexec.c
+++ b/kernel/kexec.c
@@ -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;
}
diff --git a/kernel/sched.c b/kernel/sched.c
index a4ca632..fd1f00d 100644
--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -6922,3 +6922,17 @@ void set_curr_task(int cpu, struct task_
}

#ifdef
+
+/*
+ * is_container_init:
+ * Currently same as is_global_init()
+ * Eventually:
+ * if ns is NULL, check whether tsk is a container in any pidns
+ * if ns is not NULL, check whether tsk is the initns in that
+ * namespace
+ */
+int is_container_init(struct task_struct *tsk, struct pid_namespace *ns)
+{
+ return tsk->pid == 1;
+}
+
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index 1b255df..0c334b3 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -1663,7 +1663,7 @@ 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);
}
diff --git a/mm/oom_kill.c b/mm/oom_kill.c
index 2f39169..1d06c4a 100644

```

```

--- a/mm/oom_kill.c
+++ b/mm/oom_kill.c
@@ -218,7 +218,7 @@ 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;

    /*
@@ -271,7 +271,7 @@ 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;
diff --git a/security/commoncap.c b/security/commoncap.c
index 5a5ef5c..9bdd867 100644
--- a/security/commoncap.c
+++ b/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 (!is_init(current)) {
+ if (!is_global_init(current)) {
    current->cap_permitted = new_permitted;
    current->cap_effective =
        cap_intersect (new_permitted, bprm->cap_effective);
--
1.1.6

```

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