

[PATCH] bogus sigaltstack calls by rt_sigreturn

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-05/4559.html>

From: Roland McGrath (roland_at_redhat.com)

Date: 05/23/04

Date: Sat, 22 May 2004 19:01:59 -0700

To: Linus Torvalds <torvalds@osdl.org>, Andrew Morton <akpm@osdl.org>

There is a longstanding bug in the `rt_sigreturn` system call.
This exists in both 2.4 and 2.6, and for almost every platform.

I am referring to this code in `sys_rt_sigreturn` (`arch/i386/kernel/signal.c`):

```
if (__copy_from_user(&st, &frame->uc.uc_stack, sizeof(st)))
    goto badframe;
/* It is more difficult to avoid calling this function than to
   call it and ignore errors. */
/*
   * THIS CANNOT WORK! "&st" is a kernel address, and "do_sigaltstack()"
   * takes a user address (and verifies that it is a user address). End
   * result: it does exactly _nothing_.
   */
do_sigaltstack(&st, NULL, regs->esp);
```

As the comment says, this is bogus. On vanilla i386 kernels, this is just harmlessly stupid—`do_sigaltstack` always does nothing and returns `-EFAULT`. However this code actually bites users on kernels using Ingo Molnar's 4G/4G address space layout changes. There some kernel stack address might very well be a lovely and readable user address as well. When that happens, we make a `sigaltstack` call with some random buffer, and then the fun begins.

To my knowledge, this has produced trouble in the real world only for 4G i386 kernels (RHEL and Fedora "hugemem" kernels) on machines that actually have several GB of physical memory (and in programs that are actually using `sigaltstack` and handling a lot of signals). However, the same clearly broken code has been blindly copied to most other architecture ports, and off hand I don't know the address space details of any other well enough to know if real kernel stack addresses and real user addresses are in fact disjoint as they are on i386 when not using the nonstandard 4GB address space layout.

The obvious intent of the call being there in the first place is to permit a signal handler to diddle its `ucontext_t.uc_stack` before returning, and have this effect a `sigaltstack` call on the signal handler return. This is not only an optimization vs doing the extra system call, but makes it

Linux-Kernel: [PATCH] bogus sigaltstack calls by rt_sigreturn

possible to make a sigaltstack change when that handler itself was running on the signal stack. AFAICT this has never actually worked before, so certainly noone depends on it. But the code certainly suggests that someone intended at one time for that to be the behavior. Thus I am inclined to fix it so it works in that way, though it has not done so before. It would also be reasonable enough to simply rip out the bogus call and not have this functionality.

From the current state of code in both 2.4 and 2.6, there is no fathoming how this broken code came about. It's actually much simpler to just make it work! I can only presume that at some point in the past the sigaltstack implementation functions were different such that this made sense. Of the few ports I've looked at briefly, only the ppc/pc64 porters (go paulus!) actually tried to understand what the i386 code was doing and implemented it correctly rather than just carefully transliterating the bug.

The patch below fixes only the i386 and x86_64 versions. The x86_64 patches I have not actually tested. I think each and every arch (except ppc and ppc64) need to make the corresponding fixes as well. Note that there is a function to fix for each native arch, and then one for each emulation flavor. The details differ minutely for getting the calls right in each emulation flavor, but I think that most or all of the arch's with biarch/emulation support have similar enough code that each emulation flavor's fix will look very much like the arch/x86_64/ia32/ia32_signal.c patch here.

Thanks,
Roland

Index: linux-2.6/arch/i386/kernel/signal.c

```
=====
RCS file: /home/roland/redhat/bkcvcs/linux-2.5/arch/i386/kernel/signal.c,v
retrieving revision 1.37
diff -b -p -u -r1.37 signal.c
--- linux-2.6/arch/i386/kernel/signal.c 12 Apr 2004 20:32:02 -0000 1.37
+++ linux-2.6/arch/i386/kernel/signal.c 22 May 2004 22:20:58 -0000
@@ -231,7 +231,6 @@ asmlinkage int sys_rt_sigreturn(unsigned
     struct pt_regs *regs = (struct pt_regs *) &__unused;
     struct rt_sigframe __user *frame = (struct rt_sigframe __user *) (regs->esp - 4);
     sigset_t set;
- stack_t st;
     int eax;

     if (verify_area(VERIFY_READ, frame, sizeof(*frame)))
@@ -248,16 +247,8 @@ asmlinkage int sys_rt_sigreturn(unsigned
     if (restore_sigcontext(regs, &frame->uc.uc_mcontext, &eax))
         goto badframe;

- if (__copy_from_user(&st, &frame->uc.uc_stack, sizeof(st)))
+ if (do_sigaltstack(&frame->uc.uc_stack, NULL, regs->esp) == -EFAULT)
     goto badframe;
```

[PATCH] bogus sigaltstack calls by rt_sigreturn

Linux-Kernel: [PATCH] bogus sigaltstack calls by rt_sigreturn

```
- /* It is more difficult to avoid calling this function than to
- call it and ignore errors. */
- /*
- * THIS CANNOT WORK! "&st" is a kernel address, and "do_sigaltstack()"
- * takes a user address (and verifies that it is a user address). End
- * result: it does exactly _nothing_.
- */
- do_sigaltstack(&st, NULL, regs->esp);

return eax;
```

Index: linux-2.6/arch/x86_64/kernel/signal.c

```
=====
RCS file: /home/roland/redhat/bkcvcs/linux-2.5/arch/x86_64/kernel/signal.c,v
retrieving revision 1.20
diff -b -p -u -r1.20 signal.c
--- linux-2.6/arch/x86_64/kernel/signal.c 30 Dec 2003 05:50:32 -0000 1.20
+++ linux-2.6/arch/x86_64/kernel/signal.c 22 May 2004 22:22:05 -0000
@@ -138,7 +138,6 @@ asmlinkage long sys_rt_sigreturn(struct
 {
     struct rt_sigframe *frame = (struct rt_sigframe *) (regs.rsp - 8);
     sigset_t set;
- stack_t st;
     long eax;

     if (verify_area(VERIFY_READ, frame, sizeof(*frame))) {
@@ -162,12 +161,8 @@ asmlinkage long sys_rt_sigreturn(struct
     printk("%d sigreturn rip:%lx rsp:%lx frame:%p rax:%lx\n", current->pid, regs.rip, regs.rsp, frame, eax);
 #endif

- if (__copy_from_user(&st, &frame->uc.uc_stack, sizeof(st))) {
+ if (do_sigaltstack(&frame->uc.uc_stack, NULL, regs.rsp) == -EFAULT)
     goto badframe;
- }
- /* It is more difficult to avoid calling this function than to
- call it and ignore errors. */
- do_sigaltstack(&st, NULL, regs.rsp);

return eax;
```

Index: linux-2.6/arch/x86_64/ia32/ia32_signal.c

```
=====
RCS file: /home/roland/redhat/bkcvcs/linux-2.5/arch/x86_64/ia32/ia32_signal.c,v
retrieving revision 1.23
diff -b -p -u -r1.23 ia32_signal.c
--- linux-2.6/arch/x86_64/ia32/ia32_signal.c 10 May 2004 21:04:25 -0000 1.23
+++ linux-2.6/arch/x86_64/ia32/ia32_signal.c 22 May 2004 22:49:13 -0000
@@ -323,16 +323,8 @@ asmlinkage long sys32_rt_sigreturn(struc
     if (ia32_restore_sigcontext(&regs, &frame->uc.uc_mcontext, &eax))
         goto badframe;
```

Linux-Kernel: [PATCH] bogus sigaltstack calls by rt_sigreturn

```
- if (__copy_from_user(&st, &frame->uc.uc_stack, sizeof(st)))
+ if (sys32_sigaltstack(&frame->uc.uc_stack, NULL, regs) == -EFAULT)
    goto badframe;
- /* It is more difficult to avoid calling this function than to
- call it and ignore errors. */
- {
- mm_segment_t oldds = get_fs();
- set_fs(KERNEL_DS);
- do_sigaltstack(&st, NULL, regs.rsp);
- set_fs(oldds);
- }

    return eax;
```

-
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>