
Subject: [PATCH] late checking of permissions during PTRACE_ATTACH
Posted by [Alexey Dobriyan](#) on Thu, 02 Aug 2007 13:08:41 GMT
[View Forum Message](#) <> [Reply to Message](#)

ptrace_attach() does permissions check after actual attaching. Given that utrace_attach is quite non-trivial operation there is no way such ordering should be allowed -- the following program should crash the box in less than second.

Something like: ./a.out \$(pidof syslogd)

```
#include <stdlib.h>
#include <sys/ptrace.h>

int main(int argc, char *argv[])
{
    int pid = atoi(argv[1]);

    while (1)
        ptrace(PTRACE_ATTACH, pid, NULL, NULL);
    return 0;
}
```

Unable to handle kernel NULL pointer dereference at 0000000000000000 RIP:
[<0000000000000000>]
PGD 17f3ed067 PUD 17ec80067 PMD 0
Oops: 0010 [1] PREEMPT SMP
CPU 1
Modules linked in: rtc
Pid: 400, comm: udevd Not tainted 2.6.23-rc1-utrace #1
RIP: 0010:[<0000000000000000>] [<0000000000000000>]
RSP: 0000:ffff81017ee4dcb0 EFLAGS: 00010202
RAX: ffffffff803cdbe0 RBX: ffff81017dfd6350 RCX: ffff81017f7a3080
RDX: 0000000000000000 RSI: ffff81017f7a3080 RDI: ffff81017dfd6350
RBP: 00000000000000021 R08: 00000000000000038 R09: ffffffff8025145e
R10: 00000000000000007 R11: 00000000000000246 R12: 0000000000000020
R13: ffff81017f7a3080 R14: ffff81017ee4def8 R15: ffff81017ecca6e0
FS: 00002b98567836d0(0000) GS:ffff81017fc017c0(0000) knlGS:0000000000000000
CS: 0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: 0000000000000000 CR3: 000000017f1e9000 CR4: 000000000000006a0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 00000000000000400
Process udevd (pid: 400, threadinfo ffff81017ee4c000, task ffff81017f7a3080)
Stack: ffffffff8025058d ffff81017ecca6f0 ffff81017f7a3080 ffff81017ecca6e0
0000000000000007 ffff81017ee4dd58 ffff81017ee4def8 ffff81017ee4de78
ffffff802506b5 ffff81017ecca700 ffff81017f7a3080 0000000000000007
Call Trace:
[<ffffff8025058d>] report_quiescent+0x36/0x13c

```
[<ffffff802506b5>] utrace_quiescent+0x22/0x215
[<ffffff80251882>] utrace_get_signal+0x513/0x576
[<ffffff802347b1>] get_signal_to_deliver+0x10c/0x3d3
[<ffffff8020abe5>] do_notify_resume+0x9c/0x728
[<ffffff803b96e0>] _spin_unlock_irqrestore+0x3d/0x69
[<ffffff80242d7f>] trace_hardirqs_on+0x116/0x13a
[<ffffff803b96ec>] _spin_unlock_irqrestore+0x49/0x69
[<ffffff803b8d35>] trace_hardirqs_on_thunk+0x35/0x37
[<ffffff80242d7f>] trace_hardirqs_on+0x116/0x13a
[<ffffff8020bca6>] retint_signal+0x46/0x90
```

Code: Bad RIP value.

RIP [<0000000000000000>]

RSP <ffff81017ee4dcb0>

CR2: 0000000000000000

Kernel panic - not syncing: Fatal exception

NOTE, NOTE, NOTE: this is exactly same backtrace as in
 "dead_engine_ops vs engine->flags" race, so it's reproducible :)

If by some miracle RHEL5 will also be patched, closes
https://bugzilla.redhat.com/bugzilla/show_bug.cgi?id=245735

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>

kernel/ptrace.c | 42 ++++++-----
 1 file changed, 21 insertions(+), 21 deletions(-)

--- a/kernel/ptrace.c

+++ b/kernel/ptrace.c

```
@@ -327,6 +327,8 @@ static int ptrace_attach(struct task_struct *task)
    goto bad;
    if (!task->mm) /* kernel threads */
    goto bad;
+ if (!ptrace_may_attach(task))
+ goto bad;

    pr_debug("%d ptrace_attach %d state %lu exit_code %x\n",
            current->pid, task->pid, task->state, task->exit_code);
@@ -346,29 +348,27 @@ static int ptrace_attach(struct task_struct *task)
    current->pid, task->pid, task->state, task->exit_code);
```

```

NO_LOCKS;
- if (ptrace_may_attach(task)) {
-   state = ptrace_setup(task, engine, current, 0,
-       capable(CAP_SYS_PTRACE));
-   if (IS_ERR(state))
-       retval = PTR_ERR(state);
-   else {
-       retval = ptrace_setup_finish(task, state);
+ state = ptrace_setup(task, engine, current, 0,
+   capable(CAP_SYS_PTRACE));
+ if (IS_ERR(state))
+   retval = PTR_ERR(state);
+ else {
+   retval = ptrace_setup_finish(task, state);

-   pr_debug("%d ptrace_attach %d after ptrace_update (%d)"
-       " %lu exit_code %x\n",
-       current->pid, task->pid, retval,
-       task->state, task->exit_code);
+ pr_debug("%d ptrace_attach %d after ptrace_update (%d)"
+   " %lu exit_code %x\n",
+   current->pid, task->pid, retval,
+   task->state, task->exit_code);

-   if (retval) {
-       /*
-        * It died before we enabled any callbacks.
-        */
-       if (retval == -EALREADY)
-           retval = -ESRCH;
-       BUG_ON(retval != -ESRCH);
-       ptrace_state_unlink(state);
-       ptrace_done(state);
-   }
+   if (retval) {
+       /*
+        * It died before we enabled any callbacks.
+        */
+       if (retval == -EALREADY)
+           retval = -ESRCH;
+       BUG_ON(retval != -ESRCH);
+       ptrace_state_unlink(state);
+       ptrace_done(state);
+   }
}
NO_LOCKS;

```
