Subject: [PATCH] Race between cat /proc/kallsyms and rmmod
Posted by Alexey Dobriyan on Wed, 14 Mar 2007 11:05:56 GMT
View Forum Message <> Reply to Message

Iterating code of /proc/kallsyms calls module_get_kallsym() which grabs
and drops module_mutex internally and returns "struct module *",
module is removed, aforementioned "struct module *" is used in non-trivial
way.

So, grab module_mutex for entire operation like /proc/modules does.

Steps to reproduce:
 while true; do modprobe xfs; rmmod xfs; done
  vs
 while true; do cat /proc/kallsyms >/dev/null; done

[where xfs could be any module, I haven't tried]

BUG: unable to handle kernel paging request at virtual address e19f808c
 printing eip:
c01dc361
*pde = 1ff5f067
*pte = 00000000
Oops: 0000 [#1]
PREEMPT
Modules linked in:
CPU: 0
EIP: 0060:[<c01dc361>]   Not tainted VLI
EFLAGS: 00010297   (2.6.21-rc3-8b9909ded6922c33c221b105b26917780cfa497d #2)
EIP is at vsnprintf+0x2af/0x48c
eax: e19f808c   ebx: ffffffff   ecx: e19f808c   edx: fffffffe
esi: dbe7aa84   edi: dbe2bf3c   ebp: ffffffff   esp: dbe2bec4
ds: 007b   es: 007b   fs: 00d8 gs: 0033 ss: 0068
Process cat (pid: 7242, ti=dbe2b000 task=df5790b0 task.ti=dbe2b000)
Stack: e19d6fde 00000000 00000010 00000008 ffffffff 00000001 00000598 dbe7aa68
       0002f362 00000010 dbe7b000 00000000 ffffffff c034bbe0 dbe7aa68 dfd31880
       dfa31e80 00001000 c01586b0 dbe2bf2c dbe2bf2c dfd31880 dfd31880 c01289f6
Call Trace:
 [<c01586b0>] seq_printf+0x2e/0x4b
 [<c01289f6>] s_show+0x4b/0x7f
 [<c0158c6e>] seq_read+0x196/0x278
 [<c0158ad8>] seq_read+0x0/0x278
 [<c0143c35>] vfs_read+0x72/0x93
 [<c0143f1c>] sys_read+0x41/0x67
 [<c0102486>] sysenter_past_esp+0x5f/0x85
=======================
Code: 74 24 28 73 03 c6 06 20 46 4d 85 ed 7f f1 e9 b9 00 00 00 8b 0f 81 f9 ff 0f 00 00 b8 ea 45
36 c0 0f 46 c8 8b 54 24 30 89 c8 eb 06 <80> 38 00 74 07 40 4a 83 fa ff 75 f4 29 c8 89 c3 89 e8 f6

```
44 24
EIP: [<c01dc361>] vsnprintf+0x2af/0x48c SS:ESP 0068:dbe2bec4

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

 kernel/kallsyms.c |    4 ++++
 kernel/module.c   |    5 +----
 2 files changed, 5 insertions(+), 4 deletions(-)

--- a/kernel/kallsyms.c
+++ b/kernel/kallsyms.c
@@ -40,6 +40,8 @@ extern const u16 kallsyms_token_index[]

 extern const unsigned long kallsyms_markers[] __attribute__((weak));

+extern struct mutex module_mutex;
+
 static inline int is_kernel_inittext(unsigned long addr)
 {
  if (addr >= (unsigned long)_sinittext
@@ -369,6 +371,7 @@ static void *s_next(struct seq_file *m,

 static void *s_start(struct seq_file *m, loff_t *pos)
 {
+ mutex_lock(&module_mutex);
  if (!update_iter(m->private, *pos))
   return NULL;
  return m->private;
@@ -376,6 +379,7 @@ static void *s_start(struct seq_file *m,

 static void s_stop(struct seq_file *m, void *p)
 {
+ mutex_unlock(&module_mutex);
 }

 static int s_show(struct seq_file *m, void *p)
--- a/kernel/module.c
+++ b/kernel/module.c
@@ -62,7 +62,7 @@ #define INIT_OFFSET_MASK (1UL << (BITS_P
 static DEFINE_SPINLOCK(modlist_lock);

 /* List of modules, protected by module_mutex AND modlist_lock */
-static DEFINE_MUTEX(module_mutex);
+DEFINE_MUTEX(module_mutex);
 static LIST_HEAD(modules);

 static BLOCKING_NOTIFIER_HEAD(module_notify_list);
```

```
@@ -2124,19 +2124,16 @@ struct module *module_get_kallsym(unsign
 {
  struct module *mod;

- mutex_lock(&module_mutex);
  list_for_each_entry(mod, &modules, list) {
   if (symnum < mod->num_symtab) {
    *value = mod->symtab[symnum].st_value;
    *type = mod->symtab[symnum].st_info;
    strlcpy(name, mod->strtab + mod->symtab[symnum].st_name,
     namelen);
-   mutex_unlock(&module_mutex);
    return mod;
   }
   symnum -= mod->num_symtab;
  }
- mutex_unlock(&module_mutex);
  return NULL;
 }
```