Subject: [PATCH RESEND 1/2] Race between cat /proc/kallsyms and rmmod
Posted by Alexey Dobriyan on Fri, 16 Mar 2007 11:35:12 GMT
View Forum Message <> Reply to Message

CONFIG_MODULES=n case fixed, We're still discussing if making
module_mutex global is OK, though.

[PATCH 1/2] Race between cat /proc/kallsyms and rmmod

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>

```
 include/linux/module.h |   14 +++++++++++++
 kernel/kallsyms.c      |    2 ++
 kernel/module.c        |   15 ++++++++++++----
 3 files changed, 27 insertions(+), 4 deletions(-)

--- a/include/linux/module.h
+++ b/include/linux/module.h
@@ -16,10 +16,24 @@ #include <linux/elf.h>
 #include <linux/stringify.h>
 #include <linux/kobject.h>
 #include <linux/moduleparam.h>
+#include <linux/mutex.h>
 #include <asm/local.h>

 #include <asm/module.h>

+extern struct mutex module_mutex;
+#ifdef CONFIG_MODULES
+void module_mutex_lock(void);
+void module_mutex_unlock(void);
+#else
+static inline void module_mutex_lock(void)
+{
+}
+static inline void module_mutex_unlock(void)
+{
+}
+#endif
+
 /* Not Yet Implemented */
 #define MODULE_SUPPORTED_DEVICE(name)

--- a/kernel/kallsyms.c
+++ b/kernel/kallsyms.c
@@ -369,6 +369,7 @@ static void *s_next(struct seq_file *m,

 static void *s_start(struct seq_file *m, loff_t *pos)
```

```
 {
+ module_mutex_lock();
  if (!update_iter(m->private, *pos))
   return NULL;
  return m->private;
@@ -376,6 +377,7 @@ static void *s_start(struct seq_file *m,

 static void s_stop(struct seq_file *m, void *p)
 {
+ module_mutex_unlock();
 }

 static int s_show(struct seq_file *m, void *p)
--- a/kernel/module.c
+++ b/kernel/module.c
@@ -62,9 +62,19 @@ #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);

+void module_mutex_lock(void)
+{
+ mutex_lock(&module_mutex);
+}
+
+void module_mutex_unlock(void)
+{
+ mutex_unlock(&module_mutex);
+}
+
 static BLOCKING_NOTIFIER_HEAD(module_notify_list);

 int register_module_notifier(struct notifier_block * nb)
@@ -2124,19 +2134,16 @@ 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;
 }
```

Subject: [PATCH v3] Race between cat /proc/kallsyms and rmmod
Posted by Alexey Dobriyan on Mon, 19 Mar 2007 14:25:43 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.

Steps to reproduce:

```
 modprobe/rmmod loop
 cat /proc/kallsyms >/dev/null loop
```

Copy all needed info under module_mutex.

NOTE: this patch keeps module_mutex static.

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

```
 include/linux/module.h |   13 +++++++++----
 kernel/kallsyms.c      |   32 ++++++++++++++++++++++++++----------
 kernel/module.c        |   12 ++++++++----
 3 files changed, 34 insertions(+), 23 deletions(-)

--- a/include/linux/module.h
+++ b/include/linux/module.h
@@ -370,10 +370,12 @@ struct module *module_text_address(unsig
 struct module *__module_text_address(unsigned long addr);
 int is_module_address(unsigned long addr);

-/* Returns module and fills in value, defined and namebuf, or NULL if
+/* Returns 0 and fills in value, defined and namebuf, or -ERANGE if
    symnum out of range. */
-struct module *module_get_kallsym(unsigned int symnum, unsigned long *value,
-    char *type, char *name, size_t namelen);
+int module_get_kallsym(unsigned int symnum, unsigned long *value, char *type,
+   char *name, size_t namelen,
```

```
+   char *module_name, size_t module_namelen,
+   int *exported);

 /* Look for this name: can be of form module:name. */
 unsigned long module_kallsyms_lookup_name(const char *name);
@@ -530,7 +532,10 @@ static inline const char *module_address
 static inline struct module *module_get_kallsym(unsigned int symnum,
        unsigned long *value,
        char *type, char *name,
-       size_t namelen)
+       size_t namelen,
+       char *module_name,
+       size_t module_namelen,
+       int *exported)
 {
  return NULL;
 }
--- a/kernel/kallsyms.c
+++ b/kernel/kallsyms.c
@@ -295,25 +295,22 @@ void __print_symbol(const char *fmt, uns
 struct kallsym_iter
 {
  loff_t pos;
- struct module *owner;
  unsigned long value;
  unsigned int nameoff; /* If iterating in core kernel symbols */
  char type;
  char name[KSYM_NAME_LEN+1];
+ char module_name[MODULE_NAME_LEN + 1];
+ int exported;
 };

 static int get_ksymbol_mod(struct kallsym_iter *iter)
 {
- iter->owner = module_get_kallsym(iter->pos - kallsyms_num_syms,
-       &iter->value, &iter->type,
-       iter->name, sizeof(iter->name));
- if (iter->owner == NULL)
+ if (module_get_kallsym(iter->pos - kallsyms_num_syms,
+       &iter->value, &iter->type,
+       iter->name, sizeof(iter->name),
+       iter->module_name, sizeof(iter->module_name),
+       &iter->exported) < 0)
   return 0;
-
- /* Label it "global" if it is exported, "local" if not exported. */
- iter->type = is_exported(iter->name, iter->owner)
-  ? toupper(iter->type) : tolower(iter->type);
```

```
-
  return 1;
 }

@@ -322,7 +319,7 @@ static unsigned long get_ksymbol_core(st
 {
  unsigned off = iter->nameoff;

- iter->owner = NULL;
+ iter->module_name[0] = '\0';
  iter->value = kallsyms_addresses[iter->pos];

  iter->type = kallsyms_get_symbol_type(off);
@@ -386,12 +383,17 @@ static int s_show(struct seq_file *m, vo
  if (!iter->name[0])
   return 0;

- if (iter->owner)
+ if (iter->module_name[0]) {
+  char type;
+
+  /* Label it "global" if it is exported,
+   * "local" if not exported. */
+  type = iter->exported ? toupper(iter->type) :
+     tolower(iter->type);
   seq_printf(m, "%0*lx %c %s\t[%s]\n",
       (int)(2*sizeof(void*)),
-      iter->value, iter->type, iter->name,
-      module_name(iter->owner));
- else
+      iter->value, type, iter->name, iter->module_name);
+ } else
   seq_printf(m, "%0*lx %c %s\n",
       (int)(2*sizeof(void*)),
       iter->value, iter->type, iter->name);
--- a/kernel/module.c
+++ b/kernel/module.c
@@ -2119,8 +2119,10 @@ const char *module_address_lookup(unsign
  return NULL;
 }

-struct module *module_get_kallsym(unsigned int symnum, unsigned long *value,
-    char *type, char *name, size_t namelen)
+int module_get_kallsym(unsigned int symnum, unsigned long *value, char *type,
+   char *name, size_t namelen,
+   char *module_name, size_t module_namelen,
+   int *exported)
 {
```

```
  struct module *mod;

@@ -2131,13 +2133,15 @@ struct module *module_get_kallsym(unsign
     *type = mod->symtab[symnum].st_info;
     strlcpy(name, mod->strtab + mod->symtab[symnum].st_name,
      namelen);
+    strlcpy(module_name, mod->name, module_namelen);
+    *exported = is_exported(name, mod);
     mutex_unlock(&module_mutex);
-    return mod;
+    return 0;
    }
    symnum -= mod->num_symtab;
   }
   mutex_unlock(&module_mutex);
- return NULL;
+ return -ERANGE;
  }

  static unsigned long mod_find_symname(struct module *mod, const char *name)
```

Subject: Re: [PATCH v3] Race between cat /proc/kallsyms and rmmod
Posted by Paulo Marques on Mon, 19 Mar 2007 15:20:43 GMT
View Forum Message <> Reply to Message

Alexey Dobriyan wrote:
> 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.
>
> Steps to reproduce:
>
> modprobe/rmmod loop
> cat /proc/kallsyms >/dev/null loop
>
> Copy all needed info under module_mutex.
>
> NOTE: this patch keeps module_mutex static.

Yes, this patch fixes the "cat /proc/kallsyms" race without changing any
"external" interfaces, so I think it should go into mainline in any case.

Acked-by: Paulo Marques <pmarques@grupopie.com>

--
Paulo Marques - www.grupopie.com

"All I ask is a chance to prove that money can't make me happy."

Subject: Re: [PATCH v3] Race between cat /proc/kallsyms and rmmod
Posted by Rusty Russell on Mon, 19 Mar 2007 23:35:11 GMT
View Forum Message <> Reply to Message

On Mon, 2007-03-19 at 17:33 +0300, Alexey Dobriyan wrote:
> 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.

Hi Alexey,

 I like the patch, but I wonder if passing the lengths for the symbol
and module name is overkill? There are KSYM_NAME_LEN and
MODULE_NAME_LEN constants, after all, and we could just have the callers
use buffers of sufficient size. Simplifies the code and reduces
confusion if someone passes a small buffer and it gets surprisingly
truncated...

Thanks!
Rusty.