Subject: [PATCH] Fix some kallsyms_lookup() vs rmmod races
Posted by Alexey Dobriyan on Thu, 15 Mar 2007 16:07:14 GMT
View Forum Message <> Reply to Message

[cc'ing folks whose proc files are affected]

kallsyms_lookup() can call module_address_lookup() which iterates over
modules list without module_mutex taken. Comment at the top of
module_address_lookup() says it's for oops resolution so races are
irrelevant, but in some cases it's reachable from regular code:

BUG: unable to handle kernel paging request at virtual address f94fb19c
 printing eip:
c0138097
*pde = 33576067
Oops: 0000 [#1]
PREEMPT
Modules linked in: ohci_hcd af_packet e1000 ehci_hcd uhci_hcd usbcore
CPU: 0
EIP: 0060:[<c0138097>]   Not tainted VLI
EFLAGS: 00010246   (2.6.21-rc3-8b9909ded6922c33c221b105b26917780cfa497d #25)
EIP is at module_address_lookup+0x43/0x24c
eax: 00000000   ebx: 00000000   ecx: f94fb080   edx: f882e704
esi: 00000000   edi: 00000000   ebp: 00000000   esp: d30b8e78
ds: 007b   es: 007b   fs: 00d8 gs: 0033 ss: 0068
Process cat (pid: 13274, ti=d30b8000 task=d2b50af0 task.ti=d30b8000)
Stack: d30b8f44 d30b8f48 00008001 00000000 c039a23c d30b8ec4 00000000 d30b8f44
       d30b8f48 c013876c d30b8f4c 00000000 cf544000 ffffffff4 d38ebb7c c01880ff
       d30b8f4c d30b8ec4 00000246 c0399600 00000001 00000001 00000000 c0399654
Call Trace:
 [<c013876c>] kallsyms_lookup+0x5e/0x7c
 [<c01880ff>] proc_pid_wchan+0x38/0x76
 [<c01404fe>] __alloc_pages+0x4f/0x2f9
 [<c01891c6>] proc_info_read+0x6f/0xa8
 [<c015bbfd>] sys_fstat64+0x1e/0x23
 [<c01592d4>] vfs_read+0x7d/0xb5
 [<c0189157>] proc_info_read+0x0/0xa8
 [<c0159621>] sys_read+0x41/0x6a
 [<c0103e72>] sysenter_past_esp+0x5f/0x99
 =======================
Code: 04 8b 51 04 0f 18 02 90 81 3d 4c 8d 39 c0 4c 8d 39 c0 74 3f 8b b9 18 01 00 00 8b 99 10
01 00 00 39 eb 77 07 8d 04 3b 39 c5 72 32 <8b> b1 1c 01 00 00 8b 81 14 01 00 00 39 c5 72 06
01 f0 39 c5 72
EIP: [<c0138097>] module_address_lookup+0x43/0x24c SS:ESP 0068:d30b8e78

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

```
 fs/proc/base.c           |    7 +++++++
 kernel/time/timer_list.c |    6 ++++++
 kernel/time/timer_stats.c |    6 ++++++
 mm/slab.c                |    3 +++
 4 files changed, 22 insertions(+)

--- a/fs/proc/base.c
+++ b/fs/proc/base.c
@@ -61,6 +61,7 @@ #include <linux/seq_file.h>
 #include <linux/namei.h>
 #include <linux/mnt_namespace.h>
 #include <linux/mm.h>
+#include <linux/module.h>
 #include <linux/smp_lock.h>
 #include <linux/rcupdate.h>
 #include <linux/kallsyms.h>
@@ -282,7 +283,13 @@ static int proc_pid_wchan(struct task_st

  wchan = get_wchan(task);

+ mutex_lock(&module_mutex);
  sym_name = kallsyms_lookup(wchan, &size, &offset, &modname, namebuf);
+ /*
+  * size, offset, modname aren't used, sym_name is copied into namebuf,
+  * so drop it now.
+  */
+ mutex_unlock(&module_mutex);
  if (sym_name)
   return sprintf(buffer, "%s", sym_name);
  return sprintf(buffer, "%lu", wchan);
--- a/kernel/time/timer_list.c
+++ b/kernel/time/timer_list.c
@@ -44,7 +44,13 @@ static void print_name_offset(struct seq
  const char *sym_name;
  char *modname;

+ mutex_lock(&module_mutex);
  sym_name = kallsyms_lookup(addr, &size, &offset, &modname, namebuf);
+ /*
+  * size, offset, modname aren't used, sym_name is copied into namebuf,
+  * so drop it now.
+  */
+ mutex_unlock(&module_mutex);
  if (sym_name)
   SEQ_printf(m, "%s", sym_name);
  else
--- a/kernel/time/timer_stats.c
+++ b/kernel/time/timer_stats.c
```

```
@@ -262,7 +262,13 @@ static void print_name_offset(struct seq
  const char *sym_name;
  char *modname;

+ mutex_lock(&module_mutex);
  sym_name = kallsyms_lookup(addr, &size, &offset, &modname, namebuf);
+ /*
+  * size, offset, modname aren't used, sym_name is copied into namebuf,
+  * so drop it now.
+  */
+ mutex_unlock(&module_mutex);
  if (sym_name)
   seq_printf(m, "%s", sym_name);
  else
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -4385,14 +4385,17 @@ #ifdef CONFIG_KALLSYMS
  unsigned long offset, size;
  char namebuf[KSYM_NAME_LEN+1];

+ mutex_lock(&module_mutex);
  name = kallsyms_lookup(address, &size, &offset, &modname, namebuf);

  if (name) {
   seq_printf(m, "%s+%#lx/%#lx", name, offset, size);
   if (modname)
    seq_printf(m, " [%s]", modname);
+  mutex_unlock(&module_mutex);
   return;
  }
+ mutex_unlock(&module_mutex);
 #endif
  seq_printf(m, "%p", (void *)address);
 }
```

Subject: Re: [PATCH] Fix some kallsyms_lookup() vs rmmod races
Posted by Paulo Marques on Thu, 15 Mar 2007 16:53:59 GMT
View Forum Message <> Reply to Message

Alexey Dobriyan wrote:
> [cc'ing folks whose proc files are affected]
>
> kallsyms_lookup() can call module_address_lookup() which iterates over
> modules list without module_mutex taken. Comment at the top of
> module_address_lookup() says it's for oops resolution so races are
> irrelevant, but in some cases it's reachable from regular code:

So maybe we should just add a new parameter to "kallsyms_lookup" to
inform it if it is safe to take a mutex or not.

Spreading module_mutex everywhere doesn't seem like the right interface
for several reasons:

  - new users of "kallsyms_lookup" might not be aware that they should
take module_mutex if it is safe

  - many times we will be taking module_mutex even when we are fetching
a kernel symbol that shouldn't require the mutex at all

  - it just creates new dependencies (hint: this patch shouldn't even
compile with current git since module_mutex is not declared in module.h,
not to mention compile when CONFIG_MODULES not set)

IMHO we should not expose module_mutex outside of module.c. That is just
wrong from an encapsulation point of view.

--
Paulo Marques - www.grupopie.com

"667: The neighbor of the beast."

Subject: Re: [PATCH] Fix some kallsyms_lookup() vs rmmod races
Posted by Alexey Dobriyan on Thu, 15 Mar 2007 18:26:53 GMT
View Forum Message <> Reply to Message

On Thu, Mar 15, 2007 at 04:53:59PM +0000, Paulo Marques wrote:
> Alexey Dobriyan wrote:
> >[cc'ing folks whose proc files are affected]
> >
> >kallsyms_lookup() can call module_address_lookup() which iterates over
> >modules list without module_mutex taken. Comment at the top of
> >module_address_lookup() says it's for oops resolution so races are
> >irrelevant, but in some cases it's reachable from regular code:
>
> So maybe we should just add a new parameter to "kallsyms_lookup" to
> inform it if it is safe to take a mutex or not.

You have to drag "mod->name" out of kallsyms_lookup(), so if you drop
module_mutex in it, you still have a bug.

We can agree on kallsyms_lookup() or whatever other low-level function
to copy everything into caller-supplied structure and not spreading
module_mutex. module's name is 64 minus a little, so stack usage should
be fine.

> Spreading module_mutex everywhere doesn't seem like the right interface
> for several reasons:
>
> - new users of "kallsyms_lookup" might not be aware that they should
> take module_mutex if it is safe

Well, yes.

> - many times we will be taking module_mutex even when we are fetching
> a kernel symbol that shouldn't require the mutex at all
>
> - it just creates new dependencies (hint: this patch shouldn't even
> compile with current git since module_mutex is not declared in module.h,

Yeah, I remembered about it only in subway :).

This patch is techically dependent on http://lkml.org/lkml/2007/3/14/128 aka
[PATCH v2] Race between cat /proc/kallsyms and rmmod

> not to mention compile when CONFIG_MODULES not set)

OK, I'll fix it.

> IMHO we should not expose module_mutex outside of module.c. That is just
> wrong from an encapsulation point of view.
