Subject: [PATCH 2/5] Fix race between rmmod and cat /proc/kallsyms
Posted by Alexey Dobriyan on Mon, 02 Apr 2007 14:55:42 GMT
View Forum Message <> Reply to Message

module_get_kallsym() leaks "struct module *" outside of module_mutex
which is no-no, because module can dissapear right after mutex unlock.

Copy all needed information from inside module_mutex into caller-supplied
space.

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

 include/linux/module.h | 14 +++++++-------
 kernel/kallsyms.c | 30 ++++++++++++++++++++----------
 kernel/module.c | 10 ++++++----
 3 files changed, 28 insertions(+), 26 deletions(-)

--- a/include/linux/module.h
+++ b/include/linux/module.h
@@ -370,10 +370,10 @@ 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);
+int module_get_kallsym(unsigned int symnum, unsigned long *value, char *type,
+   char *name, char *module_name, int *exported);

 /* Look for this name: can be of form module:name. */
 unsigned long module_kallsyms_lookup_name(const char *name);
@@ -527,11 +527,11 @@ static inline const char *module_address
  return NULL;
 }

-static inline struct module *module_get_kallsym(unsigned int symnum,
-      unsigned long *value,
-      char *type, char *name)
+static inline int module_get_kallsym(unsigned int symnum, unsigned long *value,
+   char *type, char *name,
+   char *module_name, int *exported)
 {
- return NULL;
+ return -ERANGE;
 }

```
 static inline unsigned long module_kallsyms_lookup_name(const char *name)
--- a/kernel/kallsyms.c
+++ b/kernel/kallsyms.c
@@ -295,25 +295,20 @@ 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);
- if (iter->owner == NULL)
+ if (module_get_kallsym(iter->pos - kallsyms_num_syms, &iter->value,
+     &iter->type, iter->name, 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 +317,7 @@ 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 +381,17 @@ 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
@@ -2120,8 +2120,8 @@ const char *module_address_lookup(unsign
  return NULL;
 }

-struct module *module_get_kallsym(unsigned int symnum, unsigned long *value,
-    char *type, char *name)
+int module_get_kallsym(unsigned int symnum, unsigned long *value, char *type,
+   char *name, char *module_name, int *exported)
 {
  struct module *mod;

@@ -2132,13 +2132,15 @@ struct module *module_get_kallsym(unsign
    *type = mod->symtab[symnum].st_info;
    strlcpy(name, mod->strtab + mod->symtab[symnum].st_name,
     KSYM_NAME_LEN + 1);
+   strlcpy(module_name, mod->name, MODULE_NAME_LEN + 1);
+   *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 2/5] Fix race between rmmod and cat /proc/kallsyms
Posted by Rusty Russell on Mon, 02 Apr 2007 23:06:58 GMT
View Forum Message <> Reply to Message

On Mon, 2007-04-02 at 19:01 +0400, Alexey Dobriyan wrote:
> +static inline int module_get_kallsym(unsigned int symnum, unsigned long *value,
> + char *type, char *name,
> + char *module_name, int *exported)
> {
> - return NULL;
> + return -ERANGE;
> }

This would normally by -ENOSYS, but since the return value is used as a
binary anyway, it doesn't matter.

Thanks,
Rusty.