Subject: [PATCH 5/5] Fix race between cat /proc/slab_allocators and rmmod
Posted by Alexey Dobriyan on Mon, 02 Apr 2007 14:58:00 GMT
View Forum Message <> Reply to Message

Same story as with cat /proc/*/wchan race vs rmmod race, only
/proc/slab_allocators want more info than just symbol name.

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

```
 include/linux/kallsyms.h |    6 ++++++
 include/linux/module.h   |    6 ++++++
 kernel/kallsyms.c        |   19 ++++++++++++++++++
 kernel/module.c          |   27 +++++++++++++++++++++++++++
 mm/slab.c                |   10 +++++++---
 5 files changed, 61 insertions(+), 7 deletions(-)

--- a/include/linux/kallsyms.h
+++ b/include/linux/kallsyms.h
@@ -23,6 +23,7 @@ const char *kallsyms_lookup(unsigned lon
        char **modname, char *namebuf);

 int lookup_symbol_name(unsigned long addr, char *symname);
+int lookup_symbol_attrs(unsigned long addr, unsigned long *size, unsigned long *offset, char
*modname, char *name);

 /* Replace "%s" in format with address, if found */
 extern void __print_symbol(const char *fmt, unsigned long address);
@@ -54,6 +55,11 @@ static inline int lookup_symbol_name(uns
  return -ERANGE;
 }

+static inline int lookup_symbol_attrs(unsigned long addr, unsigned long *size, unsigned long
*offset, char *modname, char *name)
+{
+ return -ERANGE;
+}
+
 /* Stupid that this does nothing, but I didn't create this mess. */
 #define __print_symbol(fmt, addr)
 #endif /*CONFIG_KALLSYMS*/
--- a/include/linux/module.h
+++ b/include/linux/module.h
@@ -457,6 +457,7 @@ const char *module_address_lookup(unsign
       unsigned long *offset,
       char **modname);
 int lookup_module_symbol_name(unsigned long addr, char *symname);
+int lookup_module_symbol_attrs(unsigned long addr, unsigned long *size, unsigned long *offset,
```

```
char *modname, char *name);

 /* For extable.c to search modules' exception tables. */
 const struct exception_table_entry *search_module_extables(unsigned long addr);
@@ -533,6 +534,11 @@ static inline int lookup_module_symbol_n
  return -ERANGE;
 }

+static inline int lookup_module_symbol_attrs(unsigned long addr, unsigned long *size, unsigned
long *offset, char *modname, char *name)
+{
+ return -ERANGE;
+}
+
 static inline int module_get_kallsym(unsigned int symnum, unsigned long *value,
       char *type, char *name,
       char *module_name, int *exported)
--- a/kernel/kallsyms.c
+++ b/kernel/kallsyms.c
@@ -286,6 +286,25 @@ int lookup_symbol_name(unsigned long add
  return lookup_module_symbol_name(addr, symname);
 }

+int lookup_symbol_attrs(unsigned long addr, unsigned long *size,
+    unsigned long *offset, char *modname, char *name)
+{
+ name[0] = '\0';
+ name[KSYM_NAME_LEN] = '\0';
+
+ if (is_ksym_addr(addr)) {
+  unsigned long pos;
+
+  pos = get_symbol_pos(addr, size, offset);
+  /* Grab name */
+  kallsyms_expand_symbol(get_symbol_offset(pos), name);
+  modname[0] = '\0';
+  return 0;
+ }
+ /* see if it's in a module */
+ return lookup_module_symbol_attrs(addr, size, offset, modname, name);
+}
+
 /* Replace "%s" in format with address, or returns -errno. */
 void __print_symbol(const char *fmt, unsigned long address)
 {
--- a/kernel/module.c
+++ b/kernel/module.c
@@ -2145,6 +2145,33 @@ out:
```

```
  return -ERANGE;
 }

+int lookup_module_symbol_attrs(unsigned long addr, unsigned long *size,
+    unsigned long *offset, char *modname, char *name)
+{
+ struct module *mod;
+
+ mutex_lock(&module_mutex);
+ list_for_each_entry(mod, &modules, list) {
+  if (within(addr, mod->module_init, mod->init_size) ||
+      within(addr, mod->module_core, mod->core_size)) {
+   const char *sym;
+
+   sym = get_ksymbol(mod, addr, size, offset);
+   if (!sym)
+    goto out;
+   if (modname)
+    strlcpy(modname, mod->name, MODULE_NAME_LEN + 1);
+   if (name)
+    strlcpy(name, sym, KSYM_NAME_LEN + 1);
+   mutex_unlock(&module_mutex);
+   return 0;
+  }
+ }
+out:
+ mutex_unlock(&module_mutex);
+ return -ERANGE;
+}
+
 int module_get_kallsym(unsigned int symnum, unsigned long *value, char *type,
    char *name, char *module_name, int *exported)
 {
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -4380,16 +4380,12 @@ static void handle_slab(unsigned long *n
 static void show_symbol(struct seq_file *m, unsigned long address)
 {
 #ifdef CONFIG_KALLSYMS
- char *modname;
- const char *name;
  unsigned long offset, size;
- char namebuf[KSYM_NAME_LEN+1];
-
- name = kallsyms_lookup(address, &size, &offset, &modname, namebuf);
+ char modname[MODULE_NAME_LEN + 1], name[KSYM_NAME_LEN + 1];

- if (name) {
```

```
+ if (lookup_symbol_attrs(address, &size, &offset, modname, name) == 0) {
   seq_printf(m, "%s+%#lx/%#lx", name, offset, size);
-  if (modname)
+   if (modname[0])
     seq_printf(m, " [%s]", modname);
   return;
  }
```

Subject: Re: [PATCH 5/5] Fix race between cat /proc/slab_allocators and rmmod
Posted by Rusty Russell on Mon, 02 Apr 2007 23:09:55 GMT
View Forum Message <> Reply to Message

On Mon, 2007-04-02 at 19:03 +0400, Alexey Dobriyan wrote:
> Same story as with cat /proc/*/wchan race vs rmmod race, only
> /proc/slab_allocators want more info than just symbol name.
>
> Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>

All these look excellent. I hope Andrew picked them up?

Thanks,
Rusty.

Subject: Re: [PATCH 5/5] Fix race between cat /proc/slab_allocators and rmmod
Posted by Andrew Morton on Mon, 02 Apr 2007 23:23:24 GMT
View Forum Message <> Reply to Message

On Tue, 03 Apr 2007 09:09:55 +1000
Rusty Russell <rusty@rustcorp.com.au> wrote:

> On Mon, 2007-04-02 at 19:03 +0400, Alexey Dobriyan wrote:
> > Same story as with cat /proc/*/wchan race vs rmmod race, only
> > /proc/slab_allocators want more info than just symbol name.
> >
> > Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>
>
> All these look excellent. I hope Andrew picked them up?
>

I wasn't cc'ed on them, but I'll push them into the hole marked "In" anyway.

Subject: Re: [PATCH 5/5] Fix race between cat /proc/slab_allocators and rmmod
Posted by Andrew Morton on Wed, 04 Apr 2007 00:08:27 GMT

On Mon, 2 Apr 2007 19:03:16 +0400
Alexey Dobriyan <adobriyan@sw.ru> wrote:

```
> +int lookup_module_symbol_attrs(unsigned long addr, unsigned long *size,
> +   unsigned long *offset, char *modname, char *name)
> +{
> + struct module *mod;
> +
> + mutex_lock(&module_mutex);
> + list_for_each_entry(mod, &modules, list) {
> +  if (within(addr, mod->module_init, mod->init_size) ||
> +      within(addr, mod->module_core, mod->core_size)) {
> +   const char *sym;
> +
> +   sym = get_ksymbol(mod, addr, size, offset);
> +   if (!sym)
> +    goto out;
> +   if (modname)
> +    strlcpy(modname, mod->name, MODULE_NAME_LEN + 1);
> +   if (name)
> +    strlcpy(name, sym, KSYM_NAME_LEN + 1);
> +   mutex_unlock(&module_mutex);
> +   return 0;
> +  }
> + }
> +out:
> + mutex_unlock(&module_mutex);
> + return -ERANGE;
> +}
> +
```

The functions lookup_symbol_name() and lookup_symbol_attrs() are quite general and are likely to be used for other applications in the future.

Could we please pick better names for them? The names you've chosen are far too general-sounding - we have a lot of different types of "symbol" in the kernel! Perhaps module_lookup_symbol_attrs() and kallsyms_lookup_symbol() and kallsyms_lookup_attrs() or something. But something prefixed with kallsyms_ and module_.

Also, as these general-use functions are exported to all of vmlinux, a little bit of documentation for them would be good.