
Subject: [PATCH 4/5] Fix race between cat /proc/*/wchan and rmmod et al
Posted by [Alexey Dobriyan](#) on Mon, 02 Apr 2007 14:57:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

kallsyms_lookup() can go iterating over modules list unprotected which is OK for emergency situations (oops), but not OK for regular stuff like /proc/*/wchan.

Introduce lookup_symbol_name()/lookup_module_symbol_name() which copy symbol name into caller-supplied buffer or return -ERANGE. All copying is done with module_mutex held, so...

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

```
fs/proc/base.c      | 11 +++++-----
include/linux/kallsyms.h | 7 +++++++
include/linux/module.h | 6 ++++++
kernel/kallsyms.c    | 17 ++++++
kernel/module.c      | 23 ++++++
kernel/time/timer_list.c | 11 +++++-----
kernel/time/timer_stats.c | 10 +++++-----
7 files changed, 66 insertions(+), 19 deletions(-)
```

--- a/fs/proc/base.c

+++ b/fs/proc/base.c

@@ -275,16 +275,15 @@ #ifdef CONFIG_KALLSYMS

```
*/
static int proc_pid_wchan(struct task_struct *task, char *buffer)
{
- const char *sym_name;
- unsigned long wchan;
- char namebuf[KSYM_NAME_LEN+1];
+ char symname[KSYM_NAME_LEN+1];

- wchan = get_wchan(task);

- sym_name = kallsyms_lookup(wchan, NULL, NULL, NULL, namebuf);
- if (sym_name)
- return sprintf(buffer, "%s", sym_name);
- return sprintf(buffer, "%lu", wchan);
+ if (lookup_symbol_name(wchan, symname) < 0)
+ return sprintf(buffer, "%lu", wchan);
+ else
+ return sprintf(buffer, "%s", symname);
}
#endif /* CONFIG_KALLSYMS */
```

```

--- a/include/linux/kallsyms.h
+++ b/include/linux/kallsyms.h
@@ -22,6 +22,8 @@ const char *kallsyms_lookup(unsigned lon
    unsigned long *offset,
    char **modname, char *namebuf);

+int lookup_symbol_name(unsigned long addr, char *symname);
+
/* Replace "%s" in format with address, if found */
extern void __print_symbol(const char *fmt, unsigned long address);

@@ -47,6 +49,11 @@ static inline const char *kallsyms_looku
    return NULL;
}

+static inline int lookup_symbol_name(unsigned long addr, char *symname)
+{
+ 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
@@ -456,6 +456,7 @@ const char *module_address_lookup(unsigned
    unsigned long *symbolsize,
    unsigned long *offset,
    char **modname);
+int lookup_module_symbol_name(unsigned long addr, char *symname);

/* For extable.c to search modules' exception tables. */
const struct exception_table_entry *search_module_extables(unsigned long addr);
@@ -527,6 +528,11 @@ static inline const char *module_address
    return NULL;
}

+static inline int lookup_module_symbol_name(unsigned long addr, char *symname)
+{
+ 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
@@ -269,6 +269,23 @@ const char *kallsyms_lookup(unsigned lon

```

```

    return NULL;
}

+int lookup_symbol_name(unsigned long addr, char *symname)
+{
+ symname[0] = '\0';
+ symname[KSYM_NAME_LEN] = '\0';
+
+ if (is_ksym_addr(addr)) {
+  unsigned long pos;
+
+  pos = get_symbol_pos(addr, NULL, NULL);
+  /* Grab name */
+  kallsyms_expand_symbol(get_symbol_offset(pos), symname);
+  return 0;
+ }
+ /* see if it's in a module */
+ return lookup_module_symbol_name(addr, symname);
+}
+
+ /* 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
@@ -2122,6 +2122,29 @@ const char *module_address_lookup(unsigned
    return NULL;
}

+int lookup_module_symbol_name(unsigned long addr, char *symname)
+{
+ 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, NULL, NULL);
+   if (!sym)
+    goto out;
+   strcpy(symname, 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/kernel/time/timer_list.c
+++ b/kernel/time/timer_list.c
@@ -38,15 +38,12 @@ #define SEQ_printf(m, x...) \

static void print_name_offset(struct seq_file *m, void *sym)
{
- unsigned long addr = (unsigned long)sym;
- char namebuf[KSYM_NAME_LEN+1];
- const char *sym_name;
+ char symname[KSYM_NAME_LEN+1];

- sym_name = kallsyms_lookup(addr, NULL, NULL, NULL, namebuf);
- if (sym_name)
- SEQ_printf(m, "%s", sym_name);
- else
+ if (lookup_symbol_name((unsigned long)sym, symname) < 0)
SEQ_printf(m, "<%p>", sym);
+ else
+ SEQ_printf(m, "%s", symname);
}

static void
--- a/kernel/time/timer_stats.c
+++ b/kernel/time/timer_stats.c
@@ -257,14 +257,12 @@ void timer_stats_update_stats(void *time

static void print_name_offset(struct seq_file *m, unsigned long addr)
{
- char namebuf[KSYM_NAME_LEN+1];
- const char *sym_name;
+ char symname[KSYM_NAME_LEN+1];

- sym_name = kallsyms_lookup(addr, NULL, NULL, NULL, namebuf);
- if (sym_name)
- seq_printf(m, "%s", sym_name);
- else
+ if (lookup_symbol_name(addr, symname) < 0)
seq_printf(m, "<%p>", (void *)addr);
+ else
+ seq_printf(m, "%s", symname);
}

```

```
static int tstats_show(struct seq_file *m, void *v)
```
