

## Bug : Out of range ptr error in module indicates bug in slab.c

**Source:** <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-12/6222.html>

---

**From:** selvakumar nagendran (*kernelselva\_at\_yahoo.com*)

**Date:** 12/30/04

Date: Thu, 30 Dec 2004 02:54:52 -0800 (PST)

To: linux-kernel@vger.kernel.org

Hi,

I am intercepting syscalls in kernel 2.4.28. I compiled and insmod the following module which intercepts sys\_pipe system call. I am just recording the information like pid, pipe read and write end. In the exit portion of the module I am printing them. I am storing all these details in structure my\_process in a singly linked list.

While I tried to remove the module it showed the following message.

What should I do to rectify it?

Regards,

selva

Message:

```
-----  
Writing Pipe Access Details for every process..  
ProcessID^IReadEnd^I^IWriteEnd  
5062^I3^I4  
Freeing all memory allocated..  
<3>kfree: out of range ptr 5a5a5a5ah.  
kernel BUG at slab.c:1605!  
invalid operand: 0000  
CPU: 0  
EIP: 0010:[<c0132ac3>] Tainted: P  
EFLAGS: 00010086  
eax: 00000026 ebx: 01cf0ef0 ecx: ca65c000 edx:  
00000001  
esi: ffffffff edi: 5a5a5a5a ebp: 0009a5a5 esp:  
caabff4c  
ds: 0018 es: 0018 ss: 0018  
Process rmmmod (pid: 5203, stackpage=caabf000)  
Stack: c0276440 5a5a5a5a c581d790 000005f0 00000206  
5a5a5a5a ffffffff 00000000
```

Linux-Kernel: Bug : Out of range ptr error in module indicates bug in slab.c

bffe808 d07691ad 5a5a5a5a 00000004 c02a76f8  
c02a76f8 c1030020 00000207  
d0769000 c011c3aa d0769000 d0769000 ffffffff0  
c6727000 c011b71f d0769000  
Call Trace: [<d07691ad>] [<c011c3aa>] [<c011b71f>]  
[<c0108bdb>]  
Code: 0f 0b 45 06 29 5e 27 c0 e9 ee fd ff ff 8b 44 24  
04 f6 40 1d

```
-----  
Module:  
-----  
kernmalloc.c  
-----  
#include <linux/kernel.h>  
#include <linux/module.h>  
#include <linux/init.h>  
/* this one contains the system call numbers __NR...  
*/  
#include <linux/unistd.h>  
/* for struct time */  
#include <linux/time.h>  
/* for current macro */  
#include <linux/sched.h>  
/* for kmalloc */  
#include <linux/slab.h>  
/* for invalid file system identifiers */  
#define FD_INVALID -1  
struct my_process  
{  
    long pid;  
    unsigned int pipe_read_end;  
    unsigned int pipe_write_end;  
    struct my_process *next;  
};  
struct my_process *head = NULL, *tail = NULL;  
MODULE_DESCRIPTION("Intercept sys_pipe() and display  
process and pipe details");  
MODULE_AUTHOR("Selva Kumar Nagendran, (C) 2004, GPLv2  
or later");  
/* we hold the old routine address in this function  
pointer */  
static int (*sys_pipe_saved)(unsigned long *fdes);  
/* here's our own sys_pipe(). we will store  
* after calling the old routine. */  
static int my_sys_pipe(unsigned long *fdes)  
{  
    int ret;  
    struct my_process *new = NULL;  
  
    MOD_INC_USE_COUNT;  
    ret = sys_pipe_saved(fdes);  
    new = (struct my_process *) kmalloc( sizeof(struct  
my_process) , GFP_KERNEL);  
    if(new) {  
        new -> pid = (long) current -> pid;  
        if(ret) {  
            new -> pipe_read_end = FD_INVALID;  
            new -> pipe_write_end = FD_INVALID;  
        }  
    }  
}
```

## Linux-Kernel: Bug : Out of range ptr error in module indicates bug in slab.c

```
        else    {
            new -> pipe_read_end = fdes[0];
            new -> pipe_write_end = fdes[1];
        }
        new -> next = NULL;
    }
    if(head == NULL)
        head = tail = new;
    else    {
        tail -> next = new;
        tail = new;
    }

    MOD_DEC_USE_COUNT;
    return ret;
}
int __init init_pipe(void)
{
    extern long sys_call_table[];
    /* save the old routine address, indexing into the
    syscall table
    * with __NR_gettimeofday. */
    sys_pipe_saved = (int (*)(unsigned long
*)) (sys_call_table[__NR_pipe]);
    /* now put ours in - note that the seemingly
    unreasonable cast to
    * unsigned long from a function pointer is in fact
    used throughout
    * the kernel - live with it :)
    */
    sys_call_table[__NR_pipe] = (unsigned
long)my_sys_pipe;
    /* everything's OK. We'd return -EINVAL or similar if
    it wasn't */
    return 0;
}
void __exit exit_pipe(void)
{
    extern long sys_call_table[];
    struct my_process *temp = NULL;
    /* better put back the real sys call !*/
    sys_call_table[__NR_pipe] = (unsigned
long)sys_pipe_saved;
    printk("\n Writing Pipe Access Details for every
process..");
    printk("\n ProcessID      ReadEnd          WriteEnd");

    temp = head;
    while(temp != NULL)
    {
        printk("\n%d", temp -> pid);
        printk("      %d", temp -> pipe_read_end);
        printk("      %d", temp -> pipe_write_end);
        temp = temp -> next;
    }
    printk("\nFreeing all memory allocated..");
    temp = head;
    while(temp != NULL)
    {
        kfree(temp);
        temp = temp -> next;
    }
}
```

## Linux-Kernel: Bug : Out of range ptr error in module indicates bug in slab.c

```
}  
/* macros to tell module loader our init and exit  
routines */  
module_init(init_pipe);  
module_exit(exit_pipe);
```

---

Do you Yahoo!?

Yahoo! Mail - 250MB free storage. Do more. Manage less.

[http://info.mail.yahoo.com/mail\\_250](http://info.mail.yahoo.com/mail_250)

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in  
the body of a message to [majordomo@vger.kernel.org](mailto:majordomo@vger.kernel.org)

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>