



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«Київський політехнічний інститут ім. ІГОРЯ СІКОРСЬКОГО»  
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №6**  
З дисципліни “ДПКС”

Виконав:  
студент групи ІВ-91мн  
Бояршин Ігор

Київ 2020 р.

# Завдання

У цій лабораторній роботі будуть виконані базові операції по знаходженню помилок та багів у модулі.

В якості хостової операційної системи буде використана не Ubuntu, а Arch Linux.

## Створення модуля

Створемо у файловій системі директорію stuff, де будемо зберігати необхідні файли.

Для компіляції модуля використовується таке налаштування середовища:

```
:> export PATH=/opt/gcc-arm-8.3-2019.03-x86_64-arm-eabi/bin:$PATH  
:> export CROSS_COMPILE='ccache arm-eabi-'  
:> export ARCH=arm  
:> export KDIR=/home/igorek/Stuff/linux/linux-stable
```

## Part 1

Будемо модифікувати код модуля igorek3.c, на якому зупинилися в попередній лабораторній.

Створюємо файл igorek4.c, а саме нас цікавить наступний фрагмент, в якому ви замістить друку помилкового значення параметра використовуємо BUG\_ON(), а також навмисно вводимо null pointer dereference на одній з ітерацій циклу:

```

24 static int __init igorek_init(void) {
25 {
26     unsigned int i;
27     struct my_time *ptr;
28
29     BUG_ON(times > 10);
30
31     if (times == 0 || (times <= 5 && times <= 10)) {
32         pr_warn("Inserting with %d greeting calls...\n", times);
33     }
34
35     for (i = 0; i < times; i++) {
36         if (i == 1) {
37             ptr = 0;
38         } else {
39             ptr = kmalloc(sizeof(struct my_time), GFP_KERNEL);
40         }
41         list_add(&ptr->the_list, &my_list_head);
42         ptr->start = ktime_get_ns();
43         pr_info("Hello, igorek!\n");
44         ptr->finish = ktime_get_ns();
45     }
46
47     return 0;
48 }

```

Також змінимо Kbuild та додамо пропорець -g, аби компілювати з debug symbols, що полегшує дебагінг:

```

1 CFLAGS+=-g
2 obj-m := igorek4.o

```

Тепер при завантаженні модуля зі значенням times рівному 11 маємо виконання макросу BUG\_ON:

```

/stuff # insmod igorek4.ko times=11
[ 19.226696] igorek4: loading out-of-tree module taints kernel.
[ 19.231759] ---[ cut here ]---
[ 19.232401] kernel BUG at /home/igorek/Stuff/linux/busybox/_install/stuff/igorek4.c:29!
[ 19.232369] Internal error. Oops: 0000000000000000
[ 19.232751] Modules linked in: igorek4(0+)
[ 19.233646] CPU: 0 PID: 62 Comm: insmod Tainted: G 0 4.19.114 #1
[ 19.234050] User space address: 0000000000000000 in PT-based system
[ 19.234815] PC is at igorek_init+0x18/0x1000 [igorek4]
[ 19.235277] LR is at do_one_initcall+0x54/0x208
[ 19.235467] pc : [<bf005018>] lr : [<c0302d4c>] psr: 200f0013
[ 19.235683] sp : c8bc1fdb0 ip : c8b56c80 fp : 00000000
[ 19.235861] r10: bf002040 r9 : c1604c48 r8 : 00000000
[ 19.236046] r7 : bf005000 r6 : fffffe000 r5 : c1604c48 r4 : c1783d20
[ 19.236318] r3 : bf002000 r2 : 6dc64400 r1 : 0000000b r0 : 00000000
[ 19.236671] Flags: nzCv IRQs on FIQs on Mode SVC_32 ISA ARM Segment none
[ 19.236922] Control: 10c5387d Table: 48bd806a DAC: 00000051
[ 19.237186] Process insmod (pid: 62, stack limit = 0x(ptrval))
[ 19.237421] Stack: (0xc8bc1fdb0 to 0xc8bc2000)
[ 19.237724] lda0: c1783d20 c1604c48 fffffe000 bf005000
[ 19.238075] ldc0: 00000000 c1604c48 bf002040 c0302d4c 00000000 c035c10c 00210d00 00000000
[ 19.238418] lde0: c1604c48 c8bb5000 c8bc1de4 6dc64400 00000000 e097afff ffe00000 ffffff000
[ 19.238764] le00: 8040003f c8bb52c0 dbcf0880 6dc64400 dbcf0880 c8bb5000 bf002040 6dc64400
[ 19.239095] le20: bf002040 00000002 c8b56b40 00000002 c8bb6000 c03d232c 00000001 c03d4678
[ 19.239435] le40: c8bc1f30 c8bc1f30 00000002 c8bb5fc0 00000002 c03d4694 bf00204c 00007fff
[ 19.239765] le60: bf002040 c03d1584 00000001 c03d0e98 bf002088 bf00112c bf00222c bf002170
[ 19.240112] le80: c0f089c4 c1345880 c1211518 c1211524 c121157c c1604c48 c1608e44 c8baa400
[ 19.240457] lea0: ffffff000 e0800000 c8baa400 c8bb5000 00000000 00000000 00000000 00000000
[ 19.240798] lec0: 00000000 00000000 6e72656b 00006c65 00000000 00000000 00000000 00000000
[ 19.241129] lee0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
[ 19.241449] lf00: 00000000 6dc64400 00000080 00001b30 00000000 e0979b30 0012cd80 c1604c48
[ 19.241766] lf20: 0011b1f8 fffffe000 00000051 c03d4ad8 e09682e6 e0968400 e0968000 00011b30
[ 19.242063] lf40: e09793b0 e09791e4 e0975838 00003000 00003040 00000000 00000000 00000000
[ 19.242381] lf60: 000016d4 0000002d 0000002e 00000018 00000000 00000010 00000000 6dc64400
[ 19.242710] lf80: 000f411f 0011b1f8 b6f1a950 0011b1f8 b6f1a950 0011b250 00011b30 0011b1f8 00000000
[ 19.243121] lfa0: 000f411f c0301000 0011b1f8 b6f1a950 0011b250 00011b30 0011b1f8 00000000
[ 19.243414] lfc0: 0011b1f8 b6f1a950 00011b30 00000080 00000001 be8fde80 001086c5 000f411f
[ 19.243741] lfe0: be8fdb38 be8fdb28 0003b270 b6dd41b0 600f0010 0011b250 00000000 00000000
[ 19.244490] [<bf005018>] (igorek_init [igorek4]) from [<c0302d4c>] (do_one_initcall+0x54/0x208)
[ 19.244852] [<c0302d4c>] (do_one_initcall) from [<c03d232c>] (do_init_module+0x64/0x214)
[ 19.245079] [<c03d232c>] (do_init_module) from [<c03d4694>] (load_module+0x2150/0x243c)
[ 19.245324] [<c03d4694>] (load_module) from [<c03d4ad8>] (sys_init_module+0x158/0x18c)
[ 19.245544] [<c03d4ad8>] (sys_init_module) from [<c0301000>] (ret_fast_syscall+0x0/0x54)
[ 19.245758] Exception stack(0xc8bc1fa8 to 0xc8bc1ff0)
[ 19.245911] lfa0: 0011b1f8 b6f1a950 0011b250 00011b30 0011b1f8 00000000
[ 19.246182] lfc0: 0011b1f8 b6f1a950 00011b30 00000080 00000001 be8fde80 001086c5 000f411f
[ 19.246446] lfe0: be8fdb38 be8fdb28 0003b270 b6dd41b0 600f0010 0011b250 00000000 00000000
[ 19.246811] Code: e92d47f0 e5931000 e351000a 9a000000 (e7f001f2)
[ 19.247253] ---[ end trace 56bc739582e5f46e ]---

Segmentation fault

```

Прослідкуємо ключові моменти цього повідомлення.

Використаємо утиліту objdump та знайдемо ключові моменти коду:

```
igorek > .. > linux > busybox > _install > stuff > p 1_31_stable > l_31_1 > arm-eabi-objdump -dS igorek4.ko

igorek4.ko:      file format elf32-littlearm

Disassembly of section .init.text:

00000000 <init_module>:
static int __init igorek_init(void)
{
    unsigned int i;
    struct my_time *ptr;

    BUG_ON(times > 10);
0:   e3003000      movw   r3, #0
4:   e3403000      movt   r3, #0
{
8:   e92d7000      push   {r4, r5, r6, r7, r8, r9, sl, lr}
    BUG_ON(times > 10);
c:   e59f2d00      add    r1, [r3]
10:  e351000a      cmp    r1, #10
14:  90000000      bls   1c <init_module+0x1c>
18:  e7f001f2      .word  0xe7f001f2
```

Як бачимо, значення Program Counter співпадає з рядком і операцією, яка викликала критичну ситуацію в модулі.

Тепер виконаємо аналогічні дії, аби відловити null pointer dereference. Для цього спробуємо завантажити модуль зі значенням times рівному 2, аби не спрацювала BUG\_ON(), проте спрацювала друга ітерація циклу.

```

/stuff # insmod igorek4.ko times=2
[ 15.622552] Inserting with 2 greeting calls...
[ 15.622860] netlink: ...
[ 15.62314] Unhandled fault: page domain fault (0x81b) at 0x00000010
[ 15.623582] pte= 00000000
[ 15.623730] [00000010] *pgd=48bc4835, *pte=00000000, *ppte=00000000
[ 15.624437] Internal error: : 81b [#1] SMP ARM
[ 15.624724] Modules linked in: igorek4(0+) [last unloaded: igorek4]
[ 15.625231] CPU: 0 PID: 66 Comm: insmod Tainted: G      O      4.19.114 #1
[ 15.62546] netlink: ...
[ 15.62613] PC is at igorek_init+0x8c/0x1000 [igorek4]
[ 15.626298] LR: 00000000
[ 15.626386] pc : [<bf00d08c>]    lr : [<00000017>]    psr: 600f0013
[ 15.626557] sp : c8be3db0  ip : 00000000  fp : 00000000
[ 15.626704] r10: bf00a040  r9 : bf00a004  r8 : 006000c0
[ 15.626851] r7 : c134734c  r6 : bf00a000  r5 : 00000002  r4 : 00000000
[ 15.627031] r3 : c8bc7090  r2 : 00000010  r1 : 00000003  r0 : a2b7e1c0
[ 15.627247] Flags: nZCv IRQs on FIQs on Mode SVC_32 ISA ARM Segment none
[ 15.627557] Control: 10c5387d Table: 48c2806a DAC: 00000051
[ 15.627752] Process insmod (pid: 66, stack limit = 0x(ptrval))
[ 15.627950] Stack: (0xc8be3db0 to 0xc8be4000)
[ 15.628190] 3da0:          c1783d20 c1604c48 fffffe000 bf00d000
[ 15.628578] 3dc0: 00000000 c1604c48 bf00a040 c0302d4c c0e399bc 00000000 00210d00 c0e399bc
[ 15.628899] 3de0: c1604c48 c8bb4600 00000000 c0472318 00000001 e098dff ff000000 ffffff000
[ 15.629294] 3e00: 8040003f c8bb4b00 dbcf0de0 6dc64400 dbcf0de0 c8bb4600 bf00a040 6dc64400
[ 15.629588] 3e20: bf00a040 00000002 c8bc7040 00000002 c8b569c0 c03d232c 00000001 c03d4678
[ 15.629872] 3e40: c8be3f30 c8be3f30 00000002 c8b56980 00000002 c03d4694 bf00a04c 00007fff
[ 15.630156] 3e60: bf00a040 c03d1584 00000001 c03d0e98 bf00a088 bf00912c bf00a22c bf00a170
[ 15.630436] 3e80: c0f089c4 c1345880 c1211518 c1211524 c121157c c1604c48 c1608e44 c8ba9400
[ 15.630705] 3ea0: ffffff000 e0800000 c8ba9400 c8bb4600 00000000 00000000 00000000 00000000
[ 15.630988] 3ec0: 00000000 00000000 6e72656b 00006c65 00000000 00000000 00000000 00000000
[ 15.631263] 3ee0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
[ 15.631547] 3f00: 00000000 6dc64400 00000080 00001ba4 00000000 e098cba4 0012cdf4 c1604c48
[ 15.631886] 3f20: 0011b1f8 fffffe000 00000051 c03d4ad8 e097b2ee e097b400 e097b000 00011ba4
[ 15.632232] 3f40: e098c424 e098c258 e098887c 00003000 00003040 00000000 00000000 00000000
[ 15.632538] 3f60: 000016d4 0000002d 0000002e 00000018 00000000 00000010 00000000 6dc64400
[ 15.632841] 3f80: 000f411f 0011b1f8 b6f3c950 0011b1ba4 00000080 c0301204 c8be2000 00000080
[ 15.633142] 3fa0: 000f411f c0301000 0011b1f8 b6f3c950 0011b250 0011b1ba4 0011b1f8 00000000
[ 15.633419] 3fc0: 0011b1f8 b6f3c950 0011b1ba4 00000080 00000001 bead7e90 001086c5 000f411f
[ 15.633701] 3fe0: bead7b48 bead7b38 0003b270 b6df61b0 600f0010 0011b250 00000000 00000000
[ 15.634565] [<bf00d08c>] (igorek_init [igorek4]) from [<c0302d4c>] (do_one_initcall+0x54/0x208)
[ 15.634933] [<c0302d4c>] (do_one_initcall) from [<c03d232c>] (do_init_module+0x64/0x214)
[ 15.635153] [<c03d232c>] (do_init_module) from [<c03d4694>] (load_module+0x2150/0x243c)
[ 15.635399] [<c03d4694>] (load_module) from [<c03d4ad8>] (sys_init_module+0x158/0x18c)
[ 15.635659] [<c03d4ad8>] (sys_init_module) from [<c0301000>] (ret_fast_syscall+0x0/0x54)
[ 15.636047] Exception stack(0xc8be3fa8 to 0xc8be3ff0)
[ 15.636252] 3fa0:          0011b1f8 b6f3c950 0011b250 0011b1ba4 0011b1f8 00000000
[ 15.636573] 3fc0: 0011b1f8 b6f3c950 0011b1ba4 00000080 00000001 bead7e90 001086c5 000f411f
[ 15.636857] 3fe0: bead7b48 bead7b38 0003b270 b6df61b0
[ 15.637214] Code: e2842010 e5862004 e2855001 e5832000 (e5843010)
[ 15.637651] ---[ end trace 1e286d9e38a6cf3d ]---

```

При цьому відповідне місце асемблерного коду, отриманого за допомогою objdump, має наступний вигляд:

```

static __always_inline void __write_once_size(volatile void *p, void *res, int size)
{
    switch (size) {
        case 1: *(volatile __u8 *)p = *(__u8 *)res; break;
        case 2: *(volatile __u16 *)p = *(__u16 *)res; break;
        case 4: *(volatile __u32 *)p = *(__u32 *)res; break;
    80: e5862004      str     r2, [r6, #4]
        for (i = 0; i < times; i++) {
    84: e2855001      add     r5, r5, #1
            next->prev = new;
    88: e5832004      str     r2, [r3, #4]
            new->next = next;
    8c: e5843010      str     r3, [r4, #16]
            new->prev = prev;
    90: e5849014      str     r9, [r4, #20]
            return ktime_mono_to_any(mono, TK_OFFSET_REAL);
    }
}

```

Як бачимо, на відповідному Program Counter відбувається null pointer dereference при додаванні елементу до списку.

Зauważимо, що на попередньому скріншоті видно, що саме значення адреси дорівнює 0x10, тобто 16 байт. Саме таке зміщення в структурі і має поле list\_head:

```

12 struct my_time {-
13   ▶   ktime_t start;-▶
14   ▶   ktime_t finish;-▶
15   ▶   struct list_head the_list;-▶
16 };-
```

## Part 2

Для початку перевіримо відсутність директорії /sys/kernel/debug/dynamic\_debug:

```

/stuff # ls -al /sys/kernel/debug/dynamic_debug
ls: /sys/kernel/debug/dynamic_debug: No such file or directory

```

Модифікуємо код модуля для використання pr\_debug:

```

47 static void __exit igorek_exit(void) {
48 {
49     struct list_head *pos;
50     struct list_head *temp;
51     struct my_time *entry;
52
53     pr_info(":> Removing the module...\n");
54
55     pr_debug("Before list print\n");
56     list_for_each_safe(pos, temp, &my_list_head) {
57         entry = list_entry(pos, struct my_time, the_list);
58         pr_debug("Started at %lldns and finished at %lldns, delta = %lldns.\n",
59         entry->start, entry->finish,
60         entry->finish - entry->start);
61         list_del(pos);
62         kfree(entry);
63     }
64     pr_debug("After list print\n");
65 }

```

Проте якщо завантажити на вивантажити модуль зараз, то відповідні повідомлення з'являтися не будуть.

Для того щоб вони з'явилися необхідно, по-перше, зробити `#define DEBUG` на початку файлу модуля, а по-друге встановити рівень виводу до консолі на 8, аби він захоплював рівень debug, що рівний 7:

```

/stuff # insmod igorek4.ko times=3
[ 13.083199] igorek4: loading out-of-tree module taints kernel.
[ 13.089867] Hello, igorek!
[ 13.090068] Hello, igorek!
[ 13.090188] Hello, igorek!
/stuff # rmmod igorek4
[ 16.431307] :> Removing the module...
/stuff # echo 8 > /proc/sys/kernel/printk
/stuff # cat /proc/sys/kernel/printk
8      4      1      7
/stuff # insmod igorek4.ko times=3
[ 48.562505] Hello, igorek!
[ 48.562731] Hello, igorek!
[ 48.562841] Hello, igorek!
/stuff # rmmod igorek4
[ 50.235182] :> Removing the module...
[ 50.235590] Before list print
[ 50.236113] Started at 48547361696ns and finished at 48547458000ns, delta = 96304ns.
[ 50.236424] Started at 48547249600ns and finished at 48547360928ns, delta = 111328ns.
[ 50.236760] Started at 48547013072ns and finished at 48547222256ns, delta = 209184ns.
[ 50.237056] After list print

```

Для того щоб увімкнути можливість динамічного дебагінгу потрібно перезібрати модуль зі встановленим `CONFIG_DYNAMIC_DEBUG`.

Для цього, згідно лабораторній номер 3, знаходимо файл `fragments/bbb.cfg`, та додаємо відповідне налаштування:

```

1 # --- Networking ---
2 CONFIG_BRIDGE=y
3
4 CONFIG_DYNAMIC_DEBUG=y

```

Тепер, як і в роботі 3, скомпілюємо нове ядро.

Як бачимо, тепер з'явилася папка /sys/kernel/debug/dynamic\_debug:

```

/stuff # ls -al /sys/kernel/debug/dynamic_debug/
total 0
drwxr-xr-x    2 root      root          0 Jan  1 1970 .
drwx----- 28 root      root          0 Jan  1 1970 ..
-rw-r--r--    1 root      root          0 Jan  1 1970 control

```

На наступному скріншоті показано, як змінюючи налаштування в /sys/kernel/debug/-dynamic\_debug/control можна змінювати формат, в якому будуть виводитися помідовлення pr\_debug:

```

/stuff # insmod igorek4.ko times=2
[ 370.879627] Hello, igorek!
[ 370.879826] Hello, igorek!
/stuff # cat /sys/kernel/debug/dynamic_debug/control | grep igorek4
/home/igorek/Stuff/linux/busybox/_install/stuff/igorek4.c:55 [igorek4]igorek_exit =p "Before list print\012"
/home/igorek/Stuff/linux/busybox/_install/stuff/igorek4.c:60 [igorek4]igorek_exit =p "Started at %lldns and finished at %lldns, delta = %lldns.\012"
/home/igorek/Stuff/linux/busybox/_install/stuff/igorek4.c:64 [igorek4]igorek_exit =p "After list print\012"
/stuff # echo 'file igorek4.c line 64 -p' > /sys/kernel/debug/dynamic_debug/control
/stuff # echo 'file igorek4.c line 55 =pml' > /sys/kernel/debug/dynamic_debug/control
/stuff # echo 'module igorek4 +f' > /sys/kernel/debug/dynamic_debug/control
/stuff # cat /sys/kernel/debug/dynamic_debug/control | grep igorek4
/home/igorek/Stuff/linux/busybox/_install/stuff/igorek4.c:55 [igorek4]igorek_exit =pmfl "Before list print\012"
/home/igorek/Stuff/linux/busybox/_install/stuff/igorek4.c:60 [igorek4]igorek_exit =pf "Started at %lldns and finished at %lldns, delta = %lldns.\012"
/home/igorek/Stuff/linux/busybox/_install/stuff/igorek4.c:64 [igorek4]igorek_exit =f "After list print\012"
/stuff # rmmod igorek4
[ 412.642813] :> Removing the module...
/stuff # dmesg | tail -7
[ 323.299694] igorek_exit: Started at 211143950400ns and finished at 211144114000ns, delta = 163600ns.
[ 370.879627] Hello, igorek!
[ 370.879826] Hello, igorek!
[ 412.642813] :> Removing the module...
[ 412.643090] igorek4:igorek_exit:55: Before list print
[ 412.643160] igorek_exit: Started at 3708664499360ns and finished at 370866627888ns, delta = 128528ns.
[ 412.643196] igorek_exit: Started at 370866280480ns and finished at 370866475600ns, delta = 195120ns.

```

При цьому для рядка 64 ми сказуємо зовсім не виводити повідомлення, для рядка 55 встановити вивід, ім'я модуля та номер рядка, а для всього модуля вказуємо виводити ім'я поточної функції.