Kernel Debugging 101

Siyuan Chai

A bit about myself

- Fifth year PhD student working with Prof. Tianyin Xu
  - schai.me, siyuanc3@illinois.edu
- TA for 423 in Fall 2022, 2023, 2024
- Work on operating systems, memory systems
  - Building OS framework for emerging address translation schemes
- Interned at NVIDIA, Meta, and Google
  - Work on GPU memory, AI systems etc.
- For fun
  - Badminton, frisbee, soccer, hit the gym

Outline

- How to understand error message?
- What are the helpful kernel tools?
  - printk
  - GDB
  - Memory debugging
- Other common errors
  - Kernel version mismatch

Warmup: Version Mismatch

```
root@q:~/cs423/mp1-siyuanchai1999# insmod mp1.ko
```

```
[  216.683949][  T220] mp1: disagrees about version of symbol module_layout
```

Warmup: Version Mismatch

- Compile mp1 module on host (UTM/Vmware/WSL)
  - Run with ubuntu 24
  - Default kernel version 6.2+
- Launch cs423-q in Linux 5.15 folder
  - VM runs with 5.15 version
  - Symbol layout are different across versions
- How to fix?
  - Compile inside cs423-q VM
  - Update your Makefile to run on host
  - `make -C /path/to/your/kernel/source M=$(PWD) modules`

Kernel Error Message

```
[ 1783.059199][  T896] Kernel panic - not syncing: stack-protector: Kernel stack is corrupted in:
mp1_read+0x12a/0x130 [mp1]
[ 1783.059904][  T896] CPU: 0 PID: 896 Comm: cat Tainted: G          O      5.15.165 #2
[ 1783.060386][  T896] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.15.0-1
04/01/2014
[ 1783.060943][  T896] Call Trace:
[ 1783.061141][  T896]  <TASK>
[ 1783.061319][  T896]  dump_stack_lvl+0x34/0x48
[ 1783.061600][  T896]  panic+0xfb/0x2b9
[ 1783.061842][  T896]  ? mp1_read+0x12a/0x130 [mp1]
[ 1783.062135][  T896]  __stack_chk_fail+0x10/0x10
[ 1783.062418][  T896]  mp1_read+0x12a/0x130 [mp1]
[ 1783.062701][  T896]  proc_reg_read+0x4d/0x90
[ 1783.062972][  T896]  vfs_read+0x8d/0x190
[ 1783.063222][  T896]  ksys_read+0x5a/0xee
```

Kernel Error Message

```
[ 1783.065475][  T896] RIP: 0033:0x7fedc434a7e2
[ 1783.065770][  T896] Code: c0 e9 b2 fe ff ff 50 48 8d 3d 8a b4 0c 00 e8 a5 1d 02 00 0f 1f 44 00
00 f3 0f 1e fa 64 8b 04 25 18 00 00 00 85 c0 75 10 0f 05 <48> 3d 00 f0 ff ff 77 56 c3 0f 1f 44 00
00 48 83 ec 28 48 89 54 24
[ 1783.066948][  T896] RSP: 002b:00007ffde1040308 EFLAGS: 00000246 ORIG_RAX: 0000000000000000
[ 1783.067463][  T896] RAX: ffffffffffffffda RBX: 0000000000020000 RCX: 00007fedc434a7e2
[ 1783.067937][  T896] RDX: 0000000000020000 RSI: 00007fedc4212000 RDI: 0000000000000003
[ 1783.068398][  T896] RBP: 00007fedc4212000 R08: 00007fedc4211010 R09: 00007fedc4211010
[ 1783.068856][  T896] R10: 0000000000000022 R11: 0000000000000246 R12: 0000000000022000
[ 1783.069321][  T896] R13: 0000000000000003 R14: 0000000000020000 R15: 0000000000020000
[ 1783.069783][  T896]  </TASK>
[ 1783.070046][  T896] Kernel Offset: disabled
[ 1783.070312][  T896] ---[ end Kernel panic - not syncing: stack-protector: Kernel stack is
corrupted in: mp1_read+0x12a/0x130 [mp1] ]---
```

Locate buggy code from stack trace

- Add EXTRA_CFLAGS += -g to Makefile
  - Recompile kernel modules
- `eu-addr2line -e module.ko function:offset`
- `eu-addr2line -e vmlinux function:offset`
  - Apply to kernel functions

Connect with GDB to debug

- Kernel configs
  - Turn on CONFIG_GDB_SCRIPTS, CONFIG_KGDB
  - Turn *off* CONFIG_DEBUG_INFO_REDUCED, CONFIG_RANDOMIZE_BASE
  - Recompile kernel and make scripts_gdb
- `../qemu-script/cs423-q`
  - QEMU flag to setup gdb server at port 1234
- `(on UTM/WSL/VMware)gdb vmlinux`
  - `(inside gdb) target remote localhost:1234`

Connect with GDB to debug

- Inside QEMU VM
  - insmod mp1.ko
  - lx-symbols ../mp1-githubID/
    - GDB terminal will show loading ../mp1-githubID/mp1.ko
  - `b mp1_read # set break point`
  - `Watch *(int *)0xdeadbeef # memory watch point`
    - `Int for 4 bytes`
  - `finish # get out of current function`
  - `p VAR  # print variable`

GDB FAQ

- lx-symbols not defined
  - Missing make scripts_gdb
  - Add this line to ~/.gdbinit

  ```
  add-auto-load-safe-path
  path/to/kernel_source/scripts/gdb/vmlinux-gdb.py
  ```

- Fancy gdb support
  - https://github.com/hugsy/gef

Check memory leaking with kmemleak

- Turn on `CONFIG_DEBUG_KMEMLEAK`
- `mount -t debugfs nodev /sys/kernel/debug/`
- `cat /sys/kernel/debug/kmemleak`
- `echo scan > /sys/kernel/debug/kmemleak`

- Reference: https://docs.kernel.org/dev-tools/kmemleak.html

Quick Summary

- eu-addr2line: locate your code from back trace
- printk: quick and intuitive
  - Requires a few iterations to get the correct point
  - Requires recompiling, can be time consuming if hacking kernel code
- GDB
  - Observe pretty much all the code behavior
  - Hard to capture historical info