RISC-V Linux Bug Hunting

Ben Dooks

Ben Dooks

ROLE

Senior Engineer and Fellow

TENURE

Codething since 2012

OPEN SOURCE

Linux Kernel, QEMU, OpenSBI, KiCad

EDUCATION

BSc Computer Systems and Software Engineering
University of York

Contents

Introduction

Introduction

- A tale of two bugs
- With and without a debugger

A kernel user access bug

The bug report

- **Report from syzbot**
- **Oops when running kernel stress test**

```
Unable to handle kernel access to user memory without uaccess
routines at virtual address 000000002749f0d0
Oops [#1]
Modules linked in:
CPU: 1 PID: 4875 Comm: syz-executor.0 Not tainted
5.12.0-rc2-syzkaller-00467-g0d7588ab9ef9 #0
Hardware name: riscv-virtio,qemu (DT)
epc : schedule_tail+0x72/0xb2 kernel/sched/core.c:4264
 ra : task_pid_vnr include/linux/sched.h:1421 [inline]
 ra : schedule_tail+0x70/0xb2 kernel/sched/core.c:4264
```

An OOPS

```
Unable to handle kernel access to user memory without uaccess routines at
virtual address 000000002749f0d0
Oops [#1]
Modules linked in:
CPU: 1 PID: 4875 Comm: syz-executor.0 Not tainted
5.12.0-rc2-syzkaller-00467-g0d7588ab9ef9 #0
Hardware name: riscv-virtio,qemu (DT)
epc : schedule_tail+0x72/0xb2 kernel/sched/core.c:4264
 ra : task_pid_vnr include/linux/sched.h:1421 [inline]
 ra : schedule_tail+0x70/0xb2 kernel/sched/core.c:4264
epc : ffffffe00008c8b0 ra : ffffffe00008c8ae sp : ffffffe025d17ec
 gp : ffffffe005d25378 tp : ffffffe00f0d0000 t0 : 0000000000000000
 …
 t5 : ffffffc4043cafba t6 : 0000000000040000
 status: 0000000000000120 badaddr: 000000002749f0d0 cause: 000000000000000f
Call Trace:
[<ffffffe00008c8b0>] schedule_tail+0x72/0xb2 kernel/sched/core.c:4264
[<ffffffe000005570>] ret_from_exception+0x0/0x14
```

The basics

- The kernel tried to access user memory
  - `Unable to handle kernel access to user memory without uaccess`
- RISC-V has a user memory protection
  - Kernel has copy functions for user memory

    Example: copy_to_user, put_user, get_user
  - CPU flag in STATUS CSR called SUM
  - SUM=1 then S mode can access U pages

Starting to investigate

- The syzbot report includes kernel git-ref and config
  - So we can build the same binary
  - Make a test system of our own
- Next steps
  - Retry on our own system
  - Look at the code
  - Read the assembly output

Disassembly of code

- **Use gnu disassembly via objdump**
  - **Use kconfig to add debug info to allow source to object display**
- **Highlighted user-access area**
- **Only one call to put_user**
- **The code:**

```
if (current->set_child_tid)
    put_user(
        task_pid_vnr(current),
        current->set_child_tid));
```

```
$ riscv64-linux-gnu-objdump -D
kernel/sched/core.o

<schedule_tail>:
...
    lui     a5,0xf0000
    srli    a5,a5,0x1a
    bltu    a5,s1,eac8 <.L3667>
    lui     s2,0x40
    csrs    sstatus,s2
    li      a2,0
    li      a1,0
    mv      a0,tp
    auipc   ra,0x0
    jalr    ra # eab8 <.LBB16792+0x6>
    li      a5,0
    sw      a0,0(s1)
    csrc    sstatus,s2
    auipc   ra,0x0
    jalr    ra # eac8 <.L3667>
```

A closer look and analysis

- Load s2 to 0x40 (bit for SUM flag)
- The csrs sstatus sets SUM flag
- The sw a0, 0(s1) stores the value
- The csrc clears the SUM flag

- But what is the auipc and jalr pair?
- Call to `__task_pid_nr_ns`
  - For `task_pid_vnr(current)`

```
<schedule_tail>:
...
    lui     s2,0x40
    csrs    sstatus,s2
    li      a2,0
    li      a1,0
    mv      a0,tp
    auipc   ra,0x0
    jalr    ra # eab8
    li      a5,0
    sw      a0,0(s1)
    csrc    sstatus,s2
```

Likely causes

- The `current->set_child_tid`?
- The call to `task_pid_vnr(current)`
  - Ends up in kernel/pid.c calling __task_pid_nr_ns()
  - No sign of SUM changes
- Add a check after the task_pid_vnr call
  - And SUM occasionally gets unset
  - Bet it is a task switch issue
  - Side quest: check for SUM flag leak

Fix(es)

- **Store SUM over task/thread switch**
- **Reorder macro to evaluate first**

```
#define __put_user(x, ptr)                                  \
 ({                                                         \
    __typeof__(*(ptr)) __user *__gu_ptr = (ptr);            \
+   __typeof__(*__gu_ptr) __val = (x);                      \
    long __pu_err = 0;                                      \
                                                            \
    __chk_user_ptr(__gu_ptr);                               \
                                                            \
    __enable_user_access();                                 \
-   __put_user_nocheck(x, __gu_ptr, __pu_err);              \
+   __put_user_nocheck(__val, __gu_ptr, __pu_err);          \
    __disable_user_access();                                \
```

Bonus side-bug

- Debugging big endian RISC-V startup
- Init code failed with stack frame outside memory
- Worked back to instruction endian issue
- Byte swapped pause interpreted as stack move

User illegal instruction

The bug report

- Build Freedesktop SDK (fdsdk) on RISC-V runners
  - https://gitlab.com/freedesktop-sdk/freedesktop-sdk/-/issues/1771
- Build fails with SIGILL

```
     24 ev_epoll1_linux.cc:125] grpc epoll fd: 260
     24 server_builder.cc:392] Synchronous server. Num CQs: 1, Min pollers: 1, Max Pollers: 2,
CQ timeout (msec): 10000
 /usr/bin/bash: line 207:    24 Illegal instruction     (core dumped) buildbox-casd --verbose
CAS
```

128

First look

- **Run job under gdb**
- **Backtrace shows bug in libabsl (abseil)**
- **Instruction dump shows `rdcycle` as bad instruction**

```
Thread 5 "default-executo" received signal SIGILL, Illegal instruction.
[Switching to Thread 0x3ff54398e0 (LWP 90477)]
0x0000003ff73f8394 in absl::debian3::base_internal::UnscaledCycleClock::Now() () from
/lib/riscv64-linux-gnu/libabsl_base.so.20220623
...
0x3ff73f8394 <_ZN4absl7debian313base_internal18UnscaledCycleClock3NowEv>:    rdcycle a0
```

128

LMSTFY

- Searching found this is a known issue...
- The `rdcycle` is disabled for user from Linux 6.6
  - Security reasons!
- Why libabsl is using this to read time?
  - We have gettimeofday() and clock_gettime()
  - And efficient access via vdso
- Newer libabsl has removed this

Fixing (the easy way)

- Replace libabsl
- Use sbi-pmu driver sysfs to enable rdcycle

So that's fixed then

- Replaced libabsl
- Then another crash.

Decoding code from Kernel dmesg

- **Kernel output includes code**
  - **But as raw hex**
  - **Current code in ()**

```
[1859270.379682] Code: 8593 08a5 1517
000c 0513 e3a5 7097 ffae 80e7 b5a0
(2573) c000
```

…

```
        .text
.hword 0x8593
.hword 0x08a5
.hword 0x1517
.hword 0x000c
.hword 0x0513
.hword 0xe3a5
```

…

Decoding code from hex

- **Convert hex to gnu-as format**
- **Use gnu-as to assemble**
- **Then objdump to view**

```
…
# cat > tmp.s
# riscv64-unknown-elf-as -o tmp.o tmp.S
# riscv64-unknown-elf-objdump -D tmp.o
Disassembly of section .text:
0000000000000000 <.text>:
   0:   08a58593                addi    a1,a1,138
   4:   000c1517                auipc   a0,0xc1
   8:   e3a50513                addi    a0,a0,-454
   c:   ffae7097                auipc   ra,0xffae7
  10:   b5a080e7                jalr    -1190(ra)
  14:   c0002573                rdcycle a0
```

And more?

- Nope, this is the same bug.
- GRPC includes own libabsl....

Fixing (the hard way)

- Add rdcycle check and emulation into trap code
  - Into `do_trap_insn_illegal(struct pt_regs *regs)`
  - Use rdtime and then return to next instruction
- Issues
  - Nothing in arch/riscv has done this before
  - It also seems a bit of a hack

RISC-V Instruction format

Instructions are 16 or 32bit words. Each subset of bits specify the instruction and what the source and destination data is:

31 27	26 25	24 20	19 15	14 12	11 7	6 0	
funct7		rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]			rs1	funct3	rd	opcode	I-type
imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode	B-type
imm[31:12]					rd	opcode	U-type
imm[20|10:1|11|19:12]					rd	opcode	J-type

RISC-V Instruction format (2)

- CSR access part of SYSTEM group (0x7C)
- The **rdcycle** must be CSRRS (CSR read then set)

RV32/RV64 Zicsr Standard Extension					
csr	rs1	001	rd	1110011	CSRRW
csr	rs1	010	rd	1110011	CSRRS
csr	rs1	011	rd	1110011	CSRRC
csr	uimm	101	rd	1110011	CSRRWI
csr	uimm	110	rd	1110011	CSRRSI
csr	uimm	111	rd	1110011	CSRRCI

Handling illegal instruction

- Instruction already fetched
- Check for system instruction
- Decode bitfields
  - Helpers in header files
- For rdtime, bitfields
  - The rs == 0 and f3 == 2
  - Then csr checked
- Fixup code
  - Reads the CSR_TIME
  - Sets the rd (dest) register
  - Moves EPC (PC)
  - Returns true for success

```
bool riscv_try_csr_fixup_user(struct pt_regs *regs,
u32 insn)
…
if (riscv_insn_is_system(insn)) {
    u32 csr = RVG_EXTRACT_SYSTEM_CSR(insn);
    u32 rd = RV_EXTRACT_RD_REG(insn);
    u32 rs = RV_EXTRACT_RS1_REG(insn);
    u32 funct3 = RV_EXTRACT_FUNCT3(insn);
…
if (rs == 0 && funct3 == 2 && csr == CSR_CYCLE) {
    u64 val = csr_read(CSR_TIME);
    regs_set_register(regs, rd, val);
    regs->epc += 4;
    return true;
```

What was missing

- No `regs_set_register(regs, rd, val)`
  - Had to add this as the set for `regs_get_register()` call
- No `RV_EXTRACT_FUNCT3()`
  - Had to add that
- Updated exception code to read instruction
  - Might be passed 0, meaning read value at regs→epc
  - Previously the only user was the vector code for lazy state saves

The result

- Posted CSR fixup code to list
- No-one liked it
- Got dropped

Conclusion, resources and references

Conclusion

- May not be as easy as using a debugger
- Kernel also has useful logging systems
  - Such as trace, probes, tracepoint
- This is probably “old school”

References and resources

- Freedesktop SDK issue
  - https://gitlab.com/freedesktop-sdk/freedesktop-sdk/-/issues/1771
- Kernel code
  - https://gitlab.com/CodethinkLabs/linux-kernel/-/commits/bjdooks%2Frdcycle5c
- Test code for rdcycle
  - https://gitlab.com/CodethinkLabs/riscv_rdcycle_test
- GRPC issue
  - https://github.com/grpc/grpc/issues/37791
- Blog post on uaccess bug
  - https://www.codethink.co.uk/articles/2021/RISC-V-user-space-access-oops/

References and resources

- Mailing list discussion
  - https://lists.infradead.org/pipermail/linux-riscv/2021-March/005116.html
  - https://groups.google.com/g/syzkaller-bugs/c/G8qPRr9ievU/m/mySzCBVdAgAJ
- Discussion on SR_SUM saving
  - https://lore.kernel.org/linux-riscv/20210318151010.100966-1-ben.dooks@codethink.co.uk/

Thank You.

Codethink Ltd.

3rd Floor Dale House,
35 Dale Street,
MANCHESTER,
M1 2HF,
United Kingdom