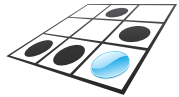
linux kernel
the history of a design flaw

Dmitry Levin

September 2019



Operating modes

64-bit mode : CS register value == 0x33

32-bit mode : CS register value == 0x23

Several methods of system call invocation

int 0x80 : Legacy 32-bit

sysenter : Fast 32-bit

syscall : 64-bit

64-bit processes can invoke both 64-bit and 32-bit system calls.

Linux API provides

- The system call number
- The value of CS register
- The value of RIP register

Legacy method of obtaining system call information

- Fetch the system call number (PTRACE_PEEKUSER ORIG_RAX)
- Fetch the value of CS register (PTRACE_PEEKUSER CS)
- **Guess the system call bitness by the value of CS register**
- Determine the system call by its number and bitness
- Fetch the system call arguments accordingly

Traditional method of obtaining system call information

- Fetch the whole set of registers (PTRACE_GETREGSET NT_PRSTATUS), the return value is decided by the value of CS register
- **Guess the system call bitness by the return value**
- Determine the system call by its number and bitness
- Interpret other registers as the system call arguments accordingly

Example based on Debian bug report #459820 submitted in 2008

```
#include <stdio.h>
#include <unistd.h>
int main() {
    setlinebuf(stdout);
    puts("------------");
    __asm__("movl $2, %eax; int $0x80");
    printf("[I am %d]\n", getpid());
    return 0;
}
```

Regular invocation: ./debbug459820

```
------------
[I am 23450]
[I am 23451]
```

Invocation under strace: strace -f -z ./debbug459820 > /dev/null

```
...
write(1, "------------\n", 13)              = 13
strace: Process 23451 attached
open(NULL, O_RDONLY|O_CREAT|O_TRUNC|O_DSYNC|O_DIRECT
|O_NOATIME|O_CLOEXEC|O_PATH|O_TMPFILE|0x1000020,
0134300) = 23451
[pid 23450] getpid()                          = 23450
[pid 23451] getpid()                          = 23451
[pid 23450] write(1, "[I am 23450]\n", 13) = 13
[pid 23451] write(1, "[I am 23451]\n", 13) = 13
[pid 23450] +++ exited with 0 +++
+++ exited with 0 +++
```

Alternative method of obtaining system call information

- Fetch the instruction pointer (PTRACE_PEEKUSER RIP)
- **Fetch the instruction (PTRACE_PEEKTEXT rip-2)**
- **Determine the system call bitness by the opcode**
- Determine the system call by its number and bitness
- Fetch the system call arguments accordingly

Drawbacks of the alternative method

- Inherent race condition between the instruction executed and the instruction fetched
- Extra ptrace system call invocation

Problem denial

- Pretend the problem does not exist
- Argue the problem has no consequences
- Claim the race is not practical
- Shrug and let those who care come up with fixes

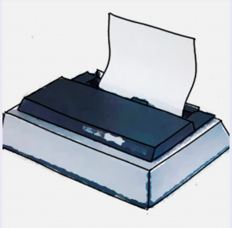
Early approaches to the problem: lively discussion in 2012

Indan Zupancic: Compat 32-bit syscall entry from 64-bit task!?

- Linus Torvalds: abuse bits #32 and #33 of eflags
- H. Peter Anvin: abuse bits [31:16] of CS
- Roland McGrath: add a new regset flavour with a pseudo-register
- Denys Vlasenko: extend NT_PRSTATUS regset
- Oleg Nesterov: add PTRACE_O_TRACESYS_VERY_GOOD
- Pedro Alves: extend PTRACE_GETEVENTMSG
- Andrew Lutomirski (2017): PTRACE_GET_SYSCALL_INFO

Process bitness does not match syscall bitness: first patches

07.11.2018: First edition



- RFC PATCH: ptrace: add PTRACE_GET_SYSCALL_INFO request

```
include/linux/ptrace.h       |  8 ++++++++
include/linux/sched.h        |  1 +
include/linux/tracehook.h    |  9 +++++---
include/uapi/linux/ptrace.h  | 23 +++++++++++++++++++++++
kernel/ptrace.c              | 53 +++++++++++++++++++++++++++++++++++++++++++++++++
kernel/signal.c              |  1 +
6 files changed, 92 insertions(+), 3 deletions(-)
```

20.11.2018 – 21.11.2018: Second edition

- v2 00/15: Prepare for PTRACE_GET_SYSCALL_INFO
  - Move EM_HEXAGON to uapi/linux/elf-em.h
  - Move EM_ARCOMPACT and EM_ARCV2 to uapi/linux/elf-em.h
  - Move EM_UNICORE to uapi/linux/elf-em.h
  - elf-em.h: add EM_NDS32
  - elf-em.h: add EM_XTENSA
  - m68k: define syscall_get_arch()
  - arc: define syscall_get_arch()
  - c6x: define syscall_get_arch()
  - h8300: define syscall_get_arch()
  - hexagon: define syscall_get_arch()
  - nds32: define syscall_get_arch()
  - nios2: define syscall_get_arch()
  - riscv: define syscall_get_arch()
  - unicore32: define syscall_get_arch()
  - **xtensa: define syscall_get_arch()**
  - syscall_get_arch: add "struct task_struct *" argument
- RFC PATCH v2: ptrace: add PTRACE_GET_SYSCALL_INFO request
- v3: powerpc/ptrace: replace ptrace_report_syscall() with a tracehook call
- **x86/ptrace: Fix documentation for tracehook_report_syscall_entry()**
- **mips: fix mips_get_syscall_arg o32 check**

25.11.2018 – 28.11.2018: 3rd and 4th editions

- v3 0/3: ptrace: add PTRACE_GET_SYSCALL_INFO request
  - ptrace: pass type of a syscall-stop in ptrace_message
  - ptrace: add PTRACE_GET_SYSCALL_INFO request
  - ptrace: add PTRACE_EVENT_SECCOMP support to PTRACE_GET_SYSCALL_INFO
- v4 0/2: ptrace: add PTRACE_GET_SYSCALL_INFO request
  - ptrace: save the type of syscall-stop in ptrace_message
  - ptrace: add PTRACE_GET_SYSCALL_INFO request

03.12.2018 – 10.12.2018: 5th edition

- ia64: fix syscall_get_error()
- microblaze: fix syscall_set_return_value()
- nios2: fix syscall_get_error()
- sh: fix syscall_set_return_value()
- **selftests: do not macro-expand failed assertion expressions**
- **v5: powerpc/ptrace: replace ptrace_report_syscall() with a tracehook call**
- v5 00/25: ptrace: add PTRACE_GET_SYSCALL_INFO request

- alpha: define remaining syscall_get_* functions
- Move EM_ARCOMPACT and EM_ARCV2 to uapi/linux/elf-em.h
- arc: define syscall_get_arch()
- c6x: define syscall_get_arch()
- elf-em.h: add EM_CSKY
- csky: define syscall_get_arch()
- h8300: define remaining syscall_get_* functions
- Move EM_HEXAGON to uapi/linux/elf-em.h
- hexagon: define remaining syscall_get_* functions
- Move EM_NDS32 to uapi/linux/elf-em.h
- nds32: define syscall_get_arch()
- nios2: define syscall_get_arch()
- m68k: add asm/syscall.h
- mips: define syscall_get_error()
- parisc: define syscall_get_error()
- powerpc: define syscall_get_error()
- riscv: define syscall_get_arch()
- **Move EM_XTENSA to uapi/linux/elf-em.h**
- xtensa: define syscall_get_* functions
- Move EM_UNICORE to uapi/linux/elf-em.h
- unicore32: add asm/syscall.h
- syscall_get_arch: add "struct task_struct *" argument
- ptrace: add PTRACE_GET_SYSCALL_INFO request
- selftests/ptrace: add a test case for PTRACE_GET_SYSCALL_INFO

13.12.2018: 6th edition (API finalized)

- v6 00/27: ptrace: add PTRACE_GET_SYSCALL_INFO request
  - **Move EM_XTENSA to uapi/linux/elf-em.h**
  - **elf-em.h: add EM_CSKY**
  - **csky: define syscall_get_arch()**
- **powerpc/ptrace: Combine SYSCALL_EMU & SYSCALL_TRACE handling**

07.01.2019: 7th edition

- v7 00/22: ptrace: add PTRACE_GET_SYSCALL_INFO request
- Fundamentally failed to build on Alpha architecture
- Changes two different subsystems: audit and ptrace
- Changes too many architectures
- Too big to be accepted

- 06.01.2019: v5.0-rc1 tagged, merge window closed
- 09.01.2019: 00/14 Prepare syscall_get_arch for PTRACE_GET_SYSCALL_INFO
  - Move EM_ARCOMPACT and EM_ARCV2 to uapi/linux/elf-em.h
  - arc: define syscall_get_arch()
  - c6x: define syscall_get_arch()
  - h8300: define syscall_get_arch()
  - Move EM_HEXAGON to uapi/linux/elf-em.h
  - hexagon: define syscall_get_arch()
  - m68k: define syscall_get_arch()
  - Move EM_NDS32 to uapi/linux/elf-em.h
  - nds32: define syscall_get_arch()
  - nios2: define syscall_get_arch()
  - riscv: define syscall_get_arch()
  - Move EM_UNICORE to uapi/linux/elf-em.h
  - unicore32: define syscall_get_arch()
  - syscall_get_arch: add "struct task_struct *" argument
- Pinging and waiting for Acked-by
- 03.03.2019: v5.0 released, merge window opened

- 17.03.2019: v5.1-rc1 tagged, merge window closed
- 18.03.2019:
  v2 00/13: Prepare syscall_get_arch for PTRACE_GET_SYSCALL_INFO
  - Move EM_ARCOMPACT and EM_ARCV2 to uapi/linux/elf-em.h
  - arc: define syscall_get_arch()
  - c6x: define syscall_get_arch()
  - h8300: define syscall_get_arch()
  - Move EM_HEXAGON to uapi/linux/elf-em.h
  - hexagon: define syscall_get_arch()
  - m68k: define syscall_get_arch()
  - Move EM_NDS32 to uapi/linux/elf-em.h
  - nds32: define syscall_get_arch()
  - nios2: define syscall_get_arch()
  - Move EM_UNICORE to uapi/linux/elf-em.h
  - unicore32: define syscall_get_arch()
  - syscall_get_arch: add "struct task_struct *" argument
- 20.03.2019: Merged into audit/next
- 05.05.2019: v5.1 released, merge window opened
- 08.05.2019: Merged via audit tree by commit v5.2-rc1~144

- 17.03.2019: v5.1-rc1 tagged, merge window closed
- 22.03.2019: v8 0/7: ptrace: add PTRACE_GET_SYSCALL_INFO request
  - nds32: fix asm/syscall.h
  - hexagon: define syscall_get_error() and syscall_get_return_value()
  - mips: define syscall_get_error()
  - parisc: define syscall_get_error()
  - powerpc: define syscall_get_error()
  - ptrace: add PTRACE_GET_SYSCALL_INFO request
  - selftests/ptrace: add a test case for PTRACE_GET_SYSCALL_INFO
- 08.04.2019: v9 0/7: ptrace: add PTRACE_GET_SYSCALL_INFO request
- 16.04.2019: v10 0/7: ptrace: add PTRACE_GET_SYSCALL_INFO request
- 05.05.2019: v5.1 released, merge window opened
- 10.05.2019: v11 0/7: ptrace: add PTRACE_GET_SYSCALL_INFO request
- 19.05.2019: v5.2-rc1 tagged, merge window closed
- 22.05.2019: Added to -mm patch queue
- 07.07.2019: v5.2 released, merge window opened
- 17.07.2019: Merged via -mm patch queue, last commit is v5.3-rc1~65^2~22

PTRACE_GET_SYSCALL_INFO: strace >= v4.26, linux >= v5.3-rc1

```
$ git log -i -E --author=altlinux.org \
  --grep='ptrace|syscall_a|elf-em|selftests' \
  v4.20-rc2..v5.3-rc1
```

- 29 commits, 47 files changed, 703 insertions, 125 deletions
- 2 authors
- 22 Acked-by/Reviewed-by/Signed-off-by
- 07.11.2018: first RFC patch submitted
- 12.11.2018: first patch committed
- 13.12.2018: API finalized
- 17.07.2019: last patch committed
- Implements PTRACE_GET_SYSCALL_INFO on those 19 architectures that enable CONFIG_HAVE_ARCH_TRACEHOOK

Extends ptrace with PTRACE_GET_SYSCALL_INFO request

```
struct ptrace_syscall_info {
  __u8 op;                                                /* Type of system call stop */
  __aligned_u32 arch;                                /* AUDIT_ARCH_* value; see seccomp(2) */
  __u64 instruction_pointer;                               /* CPU instruction pointer */
  __u64 stack_pointer;                                           /* CPU stack pointer */
  union {
    struct {                                          /* op == PTRACE_SYSCALL_INFO_ENTRY */
      __u64 nr;                                                  /* System call number */
      __u64 args[6];                                           /* System call arguments */
    } entry;
    struct {                                           /* op == PTRACE_SYSCALL_INFO_EXIT */
      __s64 rval;                                              /* System call return value */
      __u8 is_error;        /* System call error boolean: does rval contain an error value? */
    } exit;
    struct {                                         /* op == PTRACE_SYSCALL_INFO_SECCOMP */
      __u64 nr;                                                  /* System call number */
      __u64 args[6];                                           /* System call arguments */
      __u32 ret_data;         /* SECCOMP_RET_DATA portion of SECCOMP_RET_TRACE return value */
    } seccomp;
  };
};
```

Process bitness does not match syscall bitness: no longer a problem [18/20]

Example based on Debian bug report #459820 submitted in 2008

```
#include <stdio.h>
#include <unistd.h>
int main() {
    setlinebuf(stdout);
    puts("------------");
    __asm__("movl $2, %eax; int $0x80");
    printf("[I am %d]\n", getpid());
    return 0;
}
```

Regular invocation: ./debbug459820

```
------------
[I am 23450]
[I am 23451]
```

PTRACE_GET_SYSCALL_INFO: strace >= v4.26, linux >= v5.3-rc1

Invocation under strace: strace -f -z ./debbug459820 > /dev/null

```
...
write(1, "------------\n", 13)          = 13
strace: [ Process PID=23450 runs in 32 bit mode. ]
strace: Process 23451 attached
fork()                                  = 23451
strace: [ Process PID=23450 runs in 64 bit mode. ]
[pid 23450] getpid()                    = 23450
strace: [ Process PID=23451 runs in 64 bit mode. ]
[pid 23451] getpid()                    = 23451
[pid 23450] write(1, "[I am 23450]\n", 13) = 13
[pid 23451] write(1, "[I am 23451]\n", 13) = 13
[pid 23450] +++ exited with 0 +++
+++ exited with 0 +++
```

Questions?