x86-64 Exceptions: OS Perspective

Konstantin Belousov kib@FreeBSD.org

November 30, 2019, git date: 2019-11-25

Agenda

Architecture

AMD64 = x86-64 = IA32-64 = IA32e

Introductory half

- What is Exception
- Exceptions vs. Interrupts
- OS Handling of Exceptions

Advanced half

- Segments and Segments Cache
- Exceptions on return to usermode
- `POP %SS`
- VMM Exceptions: VM Exits

What is Exception

Core Loop

CPU core fetches, decodes, and executes instructions.
Speculatively and out-of-order.

Hardware Assists

Helpers used when normal execution fails. Examples:

- TLB miss: page table walker
- denormals: FPU handlers

If something went bad

If CPU has no way to execute instruction up to spec, it raises *Exception*.

What is Exception, cont.

Exception

- Non-local control transfer
- Synchronous, hw execution context is recoverable
- Typically raise the privilege mode, change hw stack
- Cannot be masked *
- One instruction can cause more than one exception

- Not exceptional, in fact very common
- Allows control software (OS, VMM) to keep control
- Syscalls are not implemented as exceptions for long time

Interrupts

Normal Interrupts

- Caused by external events
- Asynchronous
- Can be delayed (disabled, masked) *

* Non-Maskable Interrupt

- Cannot be delayed **
- ** Except inside NMI handler
- `IRETQ` reenables NMI

OS Handling of Exceptions

Out of scope

How CPU finds an exception handler: IDT, GDT, TSS, etc

Try to handle

- Page fault: page-in
- Co-processor not available: load HW context, enable FPU
- Unsupported instruction: emulate

Cannot handle

Handle == Continue faulted execution

- Send synchronous signal to the thread
- Terminate the process
- Panic the machine

Signals on Unhandled Exceptions

#PF (Page Fault)	SIGSEGV, SIGBUS
#GP (General Protection Fault), #SS (Stack Fault), #NP (Segment Not Present), #AC (Alignment)	SIGBUS
#MF (x87 floating-point error), #XM (SIMD floating point error), #DE (Divide Exception),	SIGFPE
#UD (Invalid Opcode)	SIGILL

End of Introductory Part

Questions

x86 Segmentation

Code Properties

- Code %cs
- Stack %ss
- Data %ds %es
- Segments Cache

Thread Local Storage (TLS)

- User TLS base %fs, base MSR `FSBASE`
- Kernel TLS base (Per-CPU area, PCPU) %gs, two bases: `GSBASE`, `KGSBASE`
- **SWAPGS**

SWAPGS

Kernel Entry

```
/* interrupts disabled */
if (frame->cs.rpl != 0) /* from user */
  SWAPGS;
```

NMI

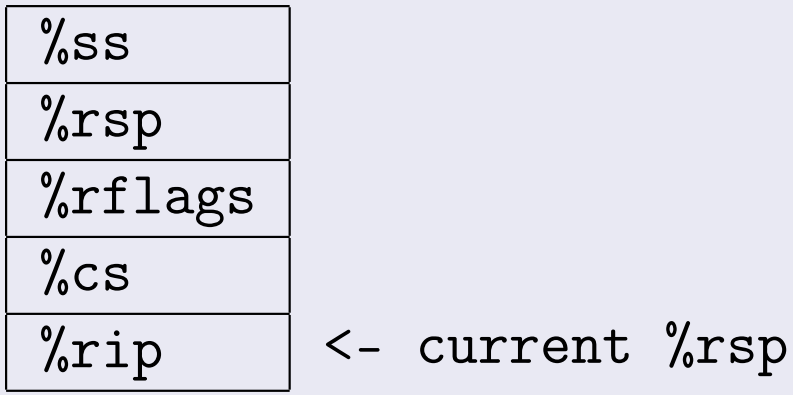
Can interrupt code above

Return to Usermode

The last kernel instructions

```
  SWAPGS
  IRETQ
...
  <Execution continues in user mode>
```

IRETQ Frame



Return to Usermode, cont.

What can go wrong ?

- %cs invalid → #np
- %ss invalid → #ss
- %rip not canonical → #gp
- %rsp not canonical → #ss

GPF, Segment, and Stack Fault handlers

- User Mode → Signal Delivery
- Kernel Mode → Kernel Bug and panic ?
- Yes, unless faulted on IRETQ to usermode
- Kernel mode, but GSBASE is User, and User page tables (KPTI)

Delaying Exceptions

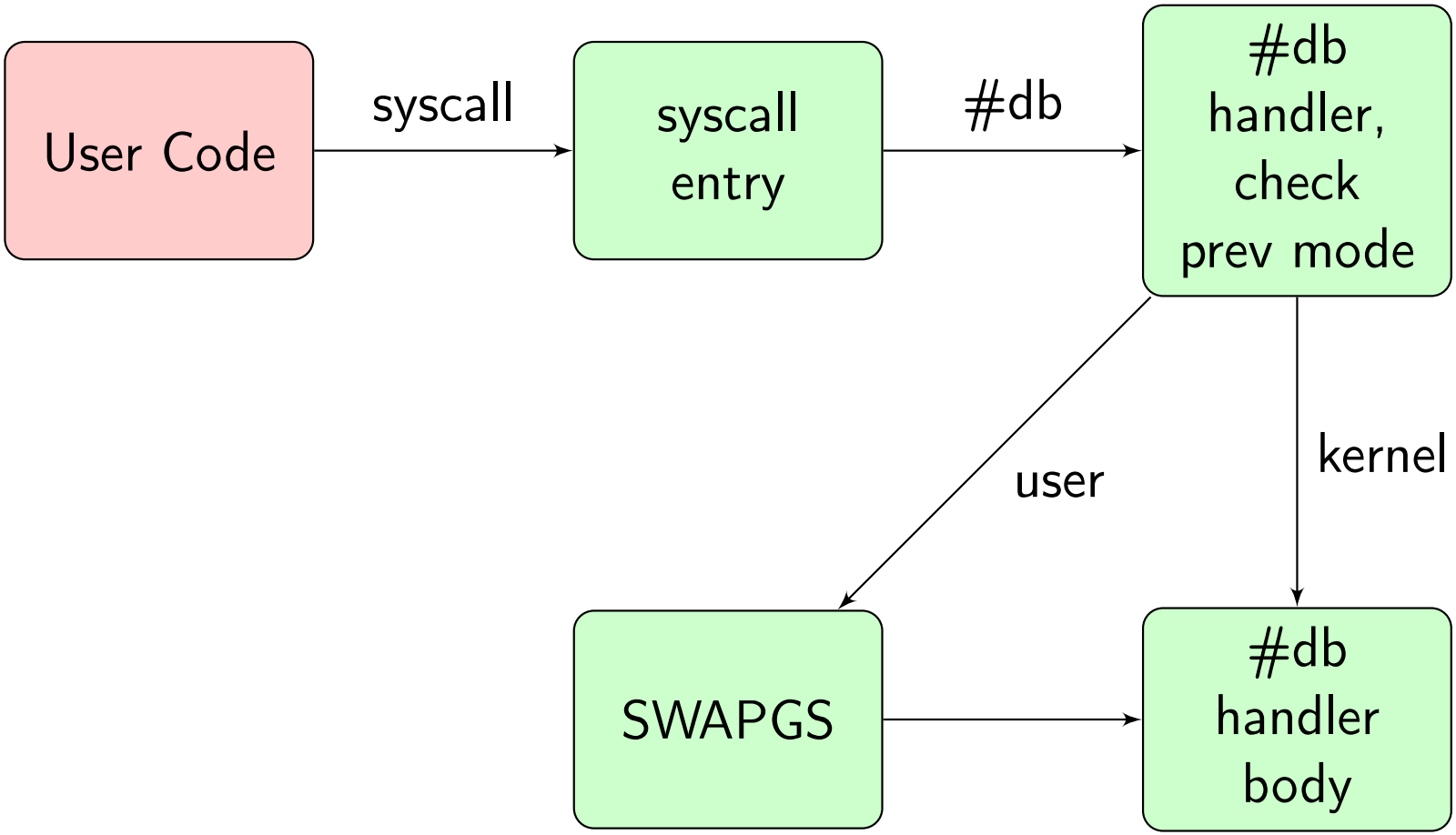
Stack reload

```
movq $STACKSEG,%ss
movq $BASE,%rsp  /* Intrs and breakpoints delayed */
next instruction /* Pending interrupts delivered  */
```

Exploit

```
movq $STACKSEG,%ss
syscall  <= Set hardware breakpoint there
            <KERNEL ENTRY>
swapgs   <= Exception occurs there
```

Delaying Exceptions: diagram



Exec modes: userspace, kernel

Delaying Exceptions

Coordinated disclosure, MS lead

CVE-2018-8897

VMX Exceptions

VM Exits

End of Second Part

Questions

References

- Intel 64 and IA-32 Architectures Software Developer Manuals, Volume 3
- AMD, AMD64 Architecture Programmer's Manual Volume 2: System Programming