Monitor and Guest Debugging

Debugging

- - Monitor debugging
  - corequery for core debugging
  - debugstub for live debugging
- - Guest debugging
  - vmss2core for guest core debugging
    - leverages guests native kernel debugger
  - debugstub for live debugging
- - HW debugging
  - not covered

Monitor core debugging example

- corequery provides very useful extensions
  - some must be run standalone
- Example: Backmapping anonymous MPNs to monitor VA's

  - Very useful for debugging corruption

  - Example: Find occurrences of 0xdeadbeef
- Could be automated - quicker but less fun

Monitor core debugging example

```
- Use 'SearchForPat' to find PTEs with the target value:
# ./corequery open vmmcore-vcpu0 \; searchforpat 0xdeadbeef 0xffffffff \;
Found Pattern 0xdeadbeef in anon page 0x18f628 at offset 0xa8c
Found Pattern 0xdeadbeef in anon page 0x1d2887 at offset 0x864

- Dump monitor page tables (MON_PAGE_TABLE:0xfffffffffc01c:0x20) and search:
(gdb) x/16384xg 0xfffffffffc01c000

# grep 18f628 pt.txt
0xfffffffffc01d920:     0x000000018f628061      0x000000011c7a8061

-Calculate mon VPN / mon VA for the PTE:
(gdb) p/x (0xfffffffffc01d920 - 0xfffffffffc01c000) / 8
$7 = 0x324
(gdb) p/x 0xfffffffffc000000 + (0x324 << 0d12)
$8 = 0xfffffffffc324000
(gdb) x/xw 0xfffffffffc324000 + 0xa8c
0xfffffffffc324a8c <FPU_PrepareX87PatchingForReplay+412>:       0xdeadbeef

-Repeat:
# grep 1d2887 pt.txt
0xfffffffffc01dcb0:     0x00000001d2887061      0x00000001eddb1061

(gdb) p/x (0xfffffffffc01dcb0 - 0xfffffffffc01c000) / 8
$9 = 0x396
(gdb) p/x 0xfffffffffc000000 + (0x396 << 0d12)
$10 = 0xfffffffffc396000
(gdb) x/xw 0xfffffffffc396000 + 0x864
0xfffffffffc396864 <HVExit+1300>:       0xdeadbeef
```

Guest core debugging

- Use vmss2core to generate guest kernel dump
  - supports Linux, Windows, MacOS, Solaris, VMM/VMK, Elf, physical and more
  - finding guest symbols is half the battle
    - without them it's hard not to get lost
- Example Solaris:
  - On Linux run vmss2core:

    # vmss2core -S t-1244b748.vmss
  - generates unix.0 (kernel) and vmcore.0 (core)
  - In a Solaris VM run mdb:

    # mdb unix.0 (kernel) and vmcore.0

Guest core debugging

```
- Dump the CPU info:
> ::cpuinfo -v
 ID ADDR             FLG NRUN BSPL PRI RNRN KRNRN SWITCH THREAD           PROC
  0 ffffffffbc304e0  1b    0    0  -1   no    no t--8740328 ffffff0009a05c40 (idle)
                        |
            RUNNING <--+
              READY
             EXISTS
             ENABLE


- Print the stacks:
> ffffff0009a05c40::findstack
stack pointer for thread ffffff0009a05c40: ffffff0009a05bc0
[ ffffff0009a05bc0 _resume_from_idle+0xf1() ]
  ffffff0009a05bf0 cpu_idle+0xbe()
  ffffff0009a05c20 idle+0x112()
  ffffff0009a05c30 thread_start+8()

- Print the log:
> ::msgbuf
MESSAGE
pcplusmp: pciexclass,060400 (pcieb) instance 3 irq 0x1b vector 0x31 ioapic 0xff intin 0xff is bound to
     cpu 0
PCI Express-device: pci15ad,7a0@18, pcieb3

- Use reference source code on OpenGrok to figure out what the guest is doing
```

Guest core debugging

- Example Linux:
- On Linux run vmss2core:

  # vmss2core -N6 VMwareRepro-d1a4aeac.vmss
- generates 'crash' compliant vmss.core0
- Find linux kernel symbols (good luck)
- Find/build/run crash

  # ./crash32 vmlinux-2.6.9-22.ELsmp vmss.core

Guest core debugging

```
- dump the log:
crash32> log
Linux version 2.6.9-22.ELsmp (bhcompile@porky.build.redhat.com) (gcc version 3.4.4 20050721 (Red Hat
     3.4.4-2)) #1 SMP Mon Sep 19 18:32:14 EDT 2005
....
Unable to handle kernel paging request at virtual address bf3ca198
 printing eip:
c0122406
*pde = a525dcbd
Oops: 0000 [#1]
SMP

Modules linked in: vmxnet3(U) vmxnet(U) ext3 jbd mptscsih mptbase sd_mod scsi_mod
CPU:    61
EIP:    0060:[<c0122406>]    Not tainted VLI
EFLAGS: 00010086   (2.6.9-22.ELsmp)
EIP is at vprintk+0xe3/0x14a
eax: f7e0808b   ebx: c040db8b   ecx: 00020000   edx: 00007c94
esi: 00000246   edi: 0000000b   ebp: 00000000   esp: f7e080f0
ds: 007b   es: 007b   ss: 0068
Process  (pid: 0, threadinfo=f7e07000 task=f7e05000)
Stack: c1b55c00 00000007 c1b47c40 c0122320 f88c99a4 f7e08110 f88c140f f88c99a4
       c1b47c40 c1b47c40 c1b55c00 f7e50d40 f88c169a c1b47c40 c1b47c40 c1b55c00
       f7e50d40 c1b51b98 f88c617e c1b51b98 c1b55c00 f7e50d40 c1b51b98 c0222080
```

Guest core debugging

```
- set the task
crash32> set 0xf7e31630
    PID: 1
COMMAND: "init"
   TASK: f7e31630  [THREAD_INFO: f7e08000]
    CPU: 11
  STATE: TASK_RUNNING (ACTIVE)

-dump the stack:
crash32> bt -S 0xf7e080f0
PID: 1      TASK: f7e31630  CPU: 11  COMMAND: "init"
 #0 [f7e080f0] schedule at c02cf361
 #1 [f7e0814c] __generic_unplug_device at c022207a
 #2 [f7e08154] elv_requeue_request at c022086b
 #3 [f7e0815c] scsi_request_fn at f88c61ac
 #4 [f7e08170] __generic_unplug_device at c022207a
 #5 [f7e08178] elv_requeue_request at c022086b
 #6 [f7e08180] scsi_request_fn at f88c61ac
 #7 [f7e08194] __generic_unplug_device at c022207a
 #8 [f7e0819c] elv_requeue_request at c022086b
 #9 [f7e081a4] scsi_request_fn at f88c61ac
#10 [f7e081b8] __generic_unplug_device at c022207a
#11 [f7e081c0] elv_requeue_request at c022086b
#12 [f7e081c8] scsi_request_fn at f88c61ac
...
#211 [f7e08f88] vfs_read at c0159c5e
#212 [f7e08fa4] sys_read at c0159e6f
#213 [f7e08fc0] system_call at c02d10c8
    EAX: 00000003  EBX: 00000003  ECX: bfff7720  EDX: 000000e4
    DS:  007b      ESI: 00000000  ES:  007b      EDI: 00000000
    SS:  007b      ESP: bfff766c  EBP: bfff7828
    CS:  0073      EIP: 0804d03d  ERR: 00000003  EFLAGS: 00000292
```

Live guest debugging

```
- Debug stub + GDB
- Add the following config settings:
   debugStub.allow=TRUE
   monitor.debugOnStartGuest32 = "1"

- Power on the VM
# vmware-vmx -qx VMwareRepro.vmx
...
Waiting for Guest 32-bit debugger to connect!
VMware Workstation is listening for debug connection on port 8832.
    target remote localhost:8832

- Run GDB with the kernel symbols
# gdb vmlinux-2.6.9-22.ELsmp

(gdb) target remote localhost:8832
Remote debugging using localhost:8832
Debug stub remote connection accepted
[New thread 1]
0xffffff0 in ?? ()
```

Live guest debugging

```
- Set breakpoints:
(gdb) b panic
Breakpoint 1 at 0xc0121adf: file kernel/panic.c, line 60.

- Breakin and see where your guest is hung:
(gdb) bt
#0  get_8254_timer_count () at include/asm/io.h:396
#1  0xc039beb4 in wait_8254_wraparound () at arch/i386/kernel/apic.c:844
#2  0xc039bf22 in calibrate_APIC_clock () at arch/i386/kernel/apic.c:961
#3  0xc039bfe0 in setup_boot_APIC_clock () at arch/i386/kernel/apic.c:1000
#4  0xc039a91e in smp_boot_cpus (max_cpus=<value optimized out>) at arch/i386/kernel/
    smpboot.c:1085
#5  0xc0100487 in init (unused=<value optimized out>) at init/main.c:699
#6  0xc01041f1 in kernel_thread_helper () at arch/i386/kernel/process.c:256

- Allows debugging any guests from gdb on Linux
```

Guest debugging with replay

- - very powerful tool to make guest failure reproduce deterministically
  - similar to PSMI and restart/replay for RTL debug at Intel
- - itrace and dtrace make guest debugging a mechanical process
- - limited to UP failures with no HMWWU
- Set a wachfault (optional) and start recording
  - debug.watchfault.num=14
  - debug.watchfault.addr=0

Replay guest debugging

```
# vmrun beginrecording VMwareRepro.vmx rec1
StateLogger::STATE LOGGING ENABLED (interponly 0 interpbt 0)
StateLogger::LOG checksum
StateLogger::USING BOUNCE BUFFERS
Starting replayCheck
StateLogger::Continue sync while logging or replaying 27656
...

# vmrun endrecording VMwareRepro.vmx rec1

> Find the fault at address 0x160015 start itracing a few branches before this:

(5262472099) seq   23381963 DEBUG               (  2) len     224 brCnt  734656148 rip
    0xc0105ca0 rcx         0xf7e07cd8 D___V___:  DBG Fault# data 0xe000000:160015 lvl 2
    kind 0

(gdb) x/i 0xc0105ca0
0xc0105ca0 <show_trace+17>:     mov    (%esi),%ebx
```

Replay guest debugging

```
> Setup the following in your config settings and create a replay.cmd file
replay.replayCmdFile=/root/replay.cmd

/root/replay.cmd:
734650000 itrace 1
734656200 itrace 0

> Start replaying:

# vmrun reverttosnapshot VMwareRepro.vmx rec5

# vmware-vmx -qx VMwareRepro.vmx &
[root@eng-rhel5-64 bug_666768]# PowerOn
NUMA: automatic VM sizing request ignored
VUsb powered on, but no USB controllers.
Failed to load OpenGL utility (GLU) library.
StateLogger::STATE REPLAYING ENABLED (interponly 1 interpbt 0)
StateLogger::LOG checksum
StateLogger::USING BOUNCE BUFFERS
Restoring VMM
SL: Resume (rip c0112696 holdoff 0)
```

Replay guest debugging

```
> Examine itrace output and plan your next move - itrace or dtrace – to find out how XBX got 0x160015
StateLogger::
StateLogger::brCnt=734656145
StateLogger::eflags=0x2 CPL=0 codeSz=4 stkSz=4
StateLogger::XAX=0xc02dae86 XBX=0x160015 XCX=0xf7e07cd8 XDX=0xc02dae86 XDI=0x160015 XSI=0xf7e08000 XBP=0x68
     XSP=0xf7e07cd8
StateLogger::R8=0x0 R9=0x0 R10=0x0 R11=0x0 R12=0x0 R13=0x0 R14=0x0 R15=0x0
StateLogger::ES=0x7b CS=0x60 SS=0x68 DS=0x7b FS=0x0 GS=0x0
StateLogger::cr0=0x8005003b cr2=0x6dba0ff0 cr3=0x37f8f2e0 cr4=0x6f0 cr8=0x0
StateLogger::
StateLogger::0x0060:0xc0105c96 | 89de                  MOV   %esi,%ebx
MEMRD: LA0 c0105c98 Reg 1 PageNum 0x105 pgOff c98 size=1 CODE MEMRD: LA0 c0105c99 Reg 1 PageNum 0x105 pgOff c99
     size=1 CODE
MEMRD: LA0 c0105c9a Reg 1 PageNum 0x105 pgOff c9a size=4 CODE
...
...
...
StateLogger::
StateLogger::brCnt=734656148
StateLogger::eflags=0x97 CPL=0 codeSz=4 stkSz=4
StateLogger::XAX=0x160ffd XBX=0x160015 XCX=0xf7e07cd8 XDX=0xc02dae86 XDI=0x160000 XSI=0x160015 XBP=0x68
     XSP=0xf7e07cd8
StateLogger::R8=0x0 R9=0x0 R10=0x0 R11=0x0 R12=0x0 R13=0x0 R14=0x0 R15=0x0
StateLogger::ES=0x7b CS=0x60 SS=0x68 DS=0x7b FS=0x0 GS=0x0
StateLogger::cr0=0x8005003b cr2=0x6dba0ff0 cr3=0x37f8f2e0 cr4=0x6f0 cr8=0x0
StateLogger::
StateLogger::0x0060:0xc0105ca0 | 8b1e                  MOV   %ebx,(%esi)


> Iterative process - challenge is doing it quickly


> itrace runs in the interpreter so monitor bugs may be converted to divergences.
```

Debugging

- Tracking structures:
  - Page tracker/PMIO ring useful for debugging memory corruption
    - Hosted only *
  - Ring buffer in the monitor useful for recent VM execution history and instrumentation
  - CCFs in BT mode provide record of recently executed guest code
- Questions/discussion ???