Dynamic and Transparent Analysis of Commodity Production Systems

Aristide Fattori[1] Roberto Paleari[1] Lorenzo Martignoni[2]
Mattia Monga[1]

[1]Università degli Studi di Milano [2]University of California, Berkeley

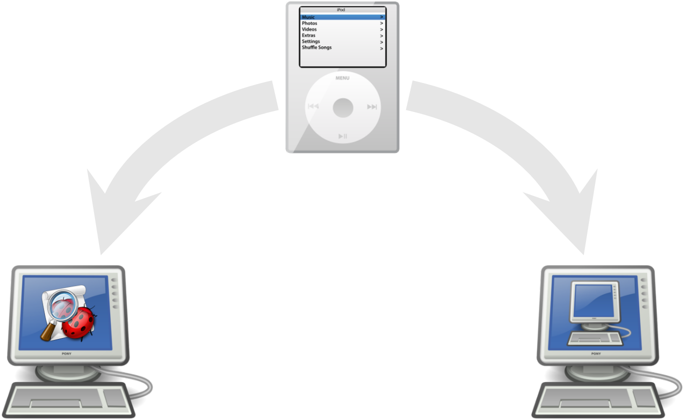
25th IEEE/ACM International Conference on Automated Software Engineering (ASE '10)

How to debug a device driver?

How to debug a device driver?

Kernel debugger

How to debug a device driver?



Kernel debugger

VMM-based debugger

How to analyze run-time properties of a system?

Properties we would like to monitor:

- Creation of new processes (or threads)
- Execution of system calls
- Execution of kernel/user functions
- Access to hardware devices
- Memory access
- . . .

Possible applications

- Profiling
- Tracing
- **Debugging**
- Dynamic instrumentation

Kernel-based solutions

- Require the installation of specific hooks in the kernel
- The analysis tool is implemented as a kernel module
- To analyze kernel-level code, these approaches leverage another kernel-level module . . .

Kernel-based solutions

- Require the installation of specific hooks in the kernel
- The analysis tool is implemented as a kernel module
- To analyze kernel-level code, these approaches leverage another kernel-level module . . .

. . . it is like a dog chasing its tail!

VMM-based solutions

- The analyzer leverages VM-introspection techniques
- The target system must be already running inside a VM!
- System-level programming inside a VM is not so easy . . .

VMM-based solutions

- The analyzer leverages VM-introspection techniques
- The target system must be already running inside a VM!
- System-level programming inside a VM is not so easy . . .

Have you ever tried to use your iPod through a VM?

Contributions

A framework to perform dynamic system-level analyses of commodity production systems

Contributions

A framework to perform dynamic system-level analyses of commodity production systems

Features

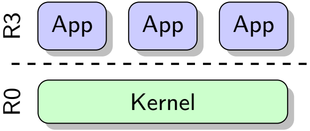
1. Does not require any native support for the analysis
   (can be used on commodity or closed-source systems)
2. Supports the analysis of running systems
   (the target must not be rebooted)
3. User- and system-level code cannot detect nor affect the analysis infrastructure
4. Guarantees isolation of the analysis tools running on its top
   (a buggy tool does not cause the target system to crash)

How?

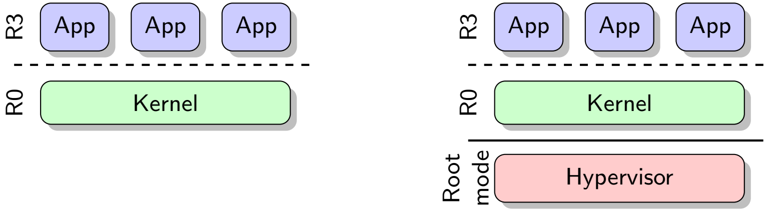
Exploit hardware support for virtualization

- A running system is migrated into a virtual machine on-the-fly
- The analysis framework runs at the hypervisor privilege level (it is more privileged than the OS and completely isolated)

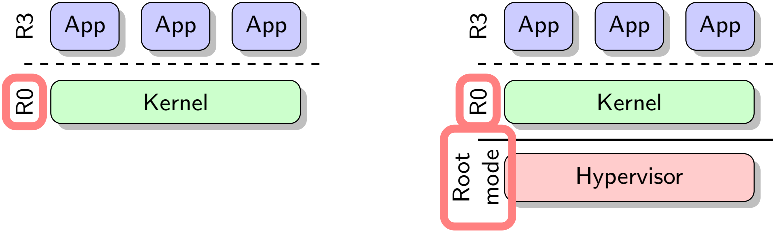
A glimpse at hardware-assisted virtualization (Intel VT-x)



A glimpse at hardware-assisted virtualization (Intel VT-x)

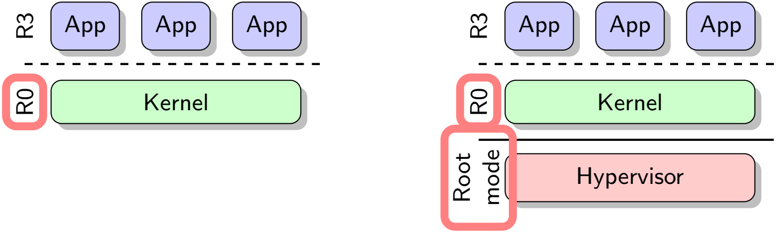


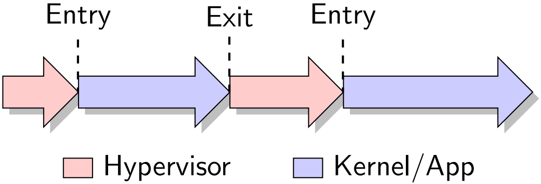
A glimpse at hardware-assisted virtualization (Intel VT-x)



- The OS needs not to be modified
- The hardware guarantees transparency & isolation
- Minimal overhead

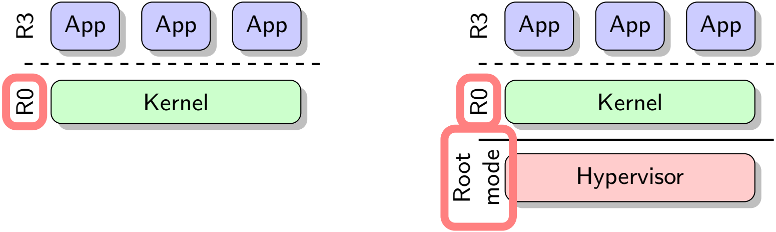
A glimpse at hardware-assisted virtualization (Intel VT-x)

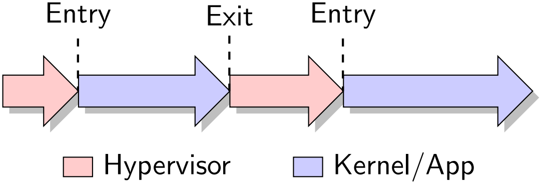




An exit/entry event causes the CPU to save the state of the guest/host inside the VMCS

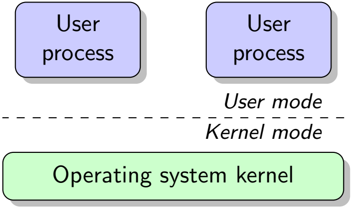
A glimpse at hardware-assisted virtualization (Intel VT-x)



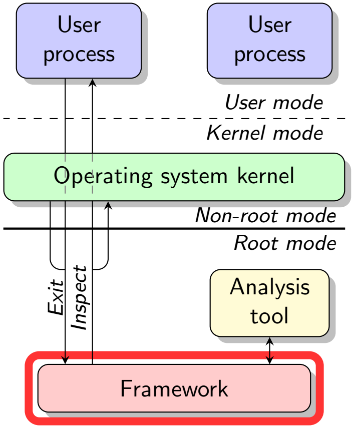


The events that trigger an exit to root mode can be configured dynamically

Overview of the framework

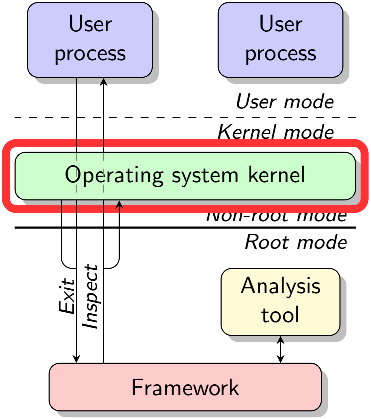


Overview of the framework



The framework is installed as the target system runs and is completely separated from the analyzed OS

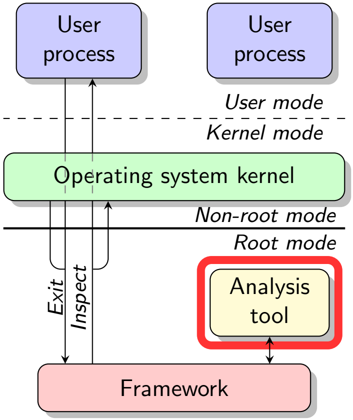
Overview of the framework



The analyzed OS needs not to be modified at all

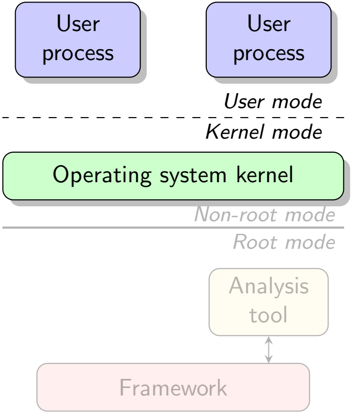
(i.e., the approach can be applied to closed-source OSes)

Overview of the framework



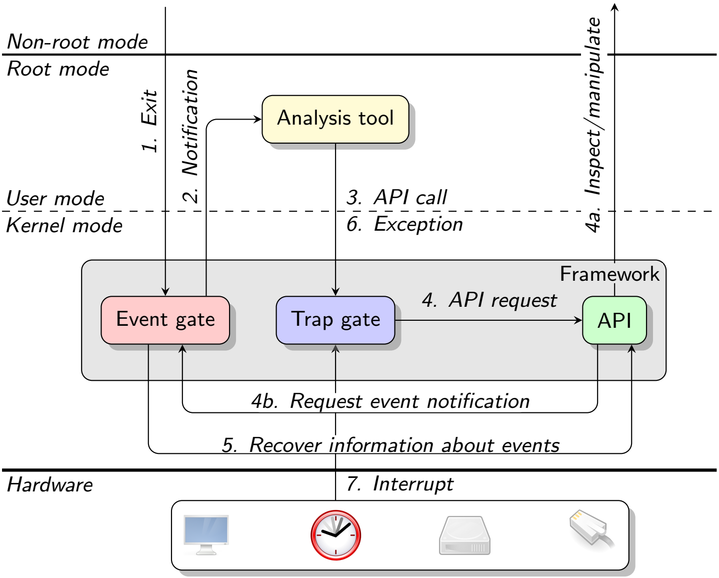
The analysis tool runs in an isolated execution environment
(a defect in the tool does not affect the stability of the OS)

Overview of the framework

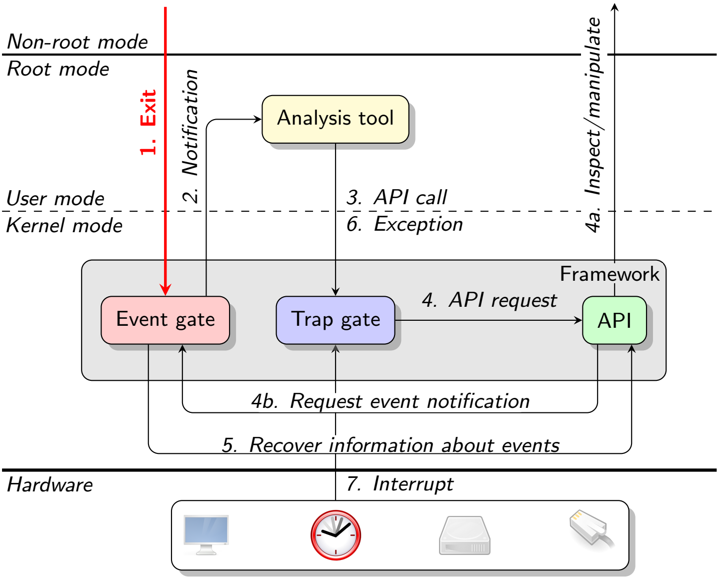


At the end of the analysis, the infrastructure can be removed on-the-fly

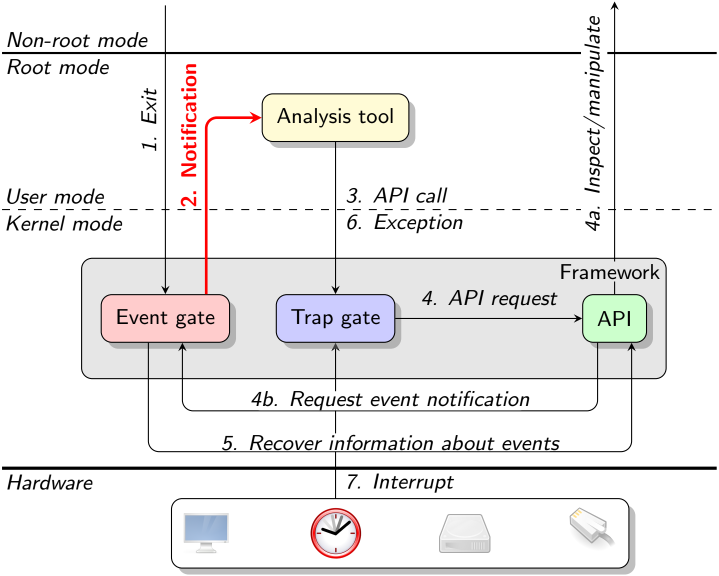
Architecture



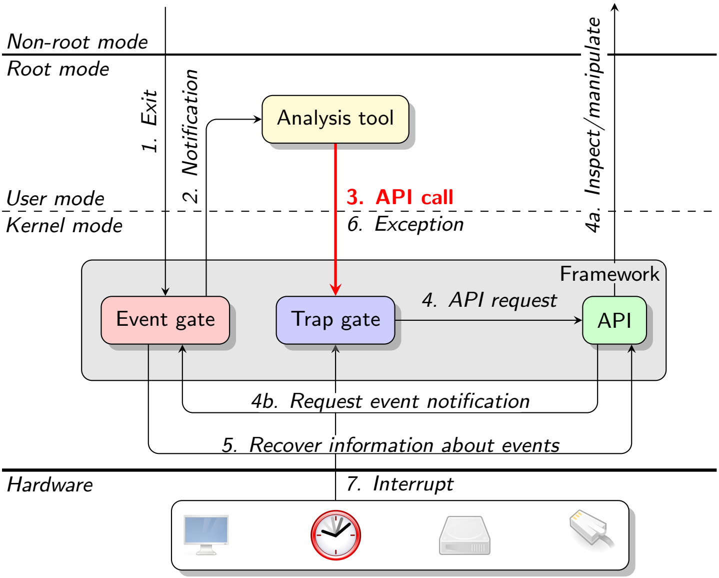
Architecture



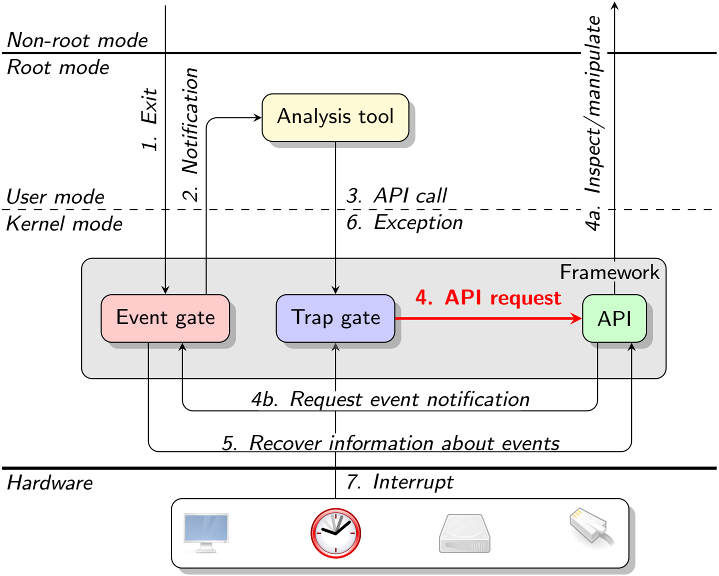
Architecture



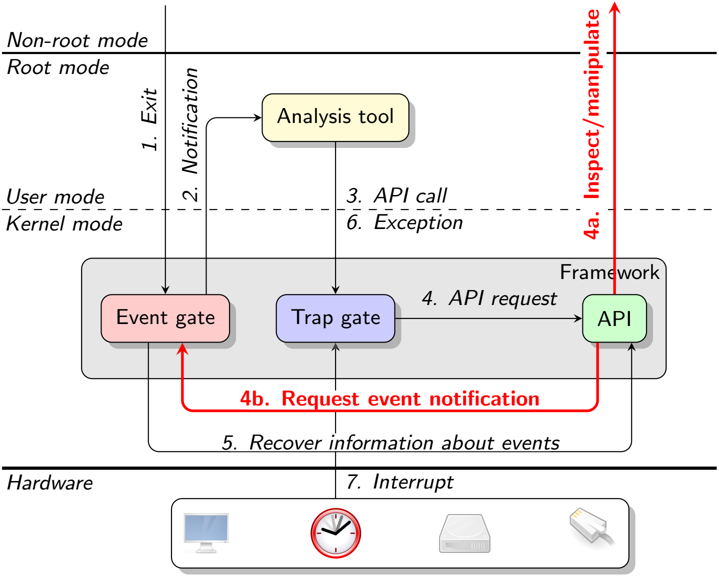
Architecture



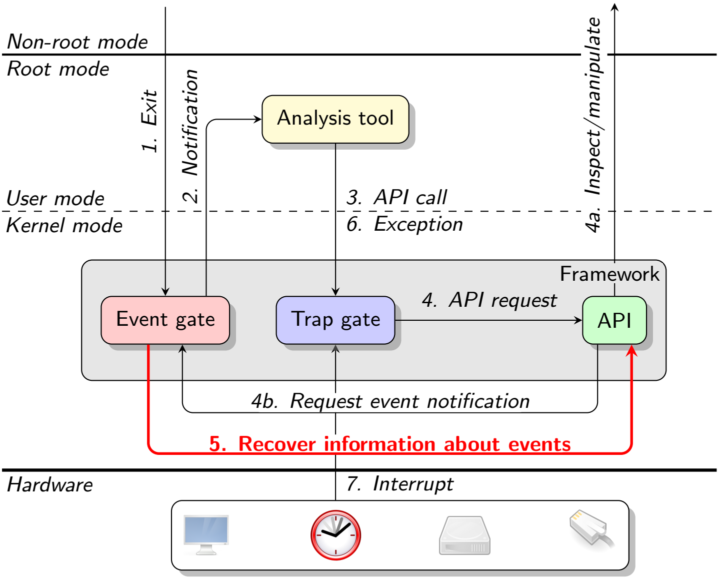
Architecture



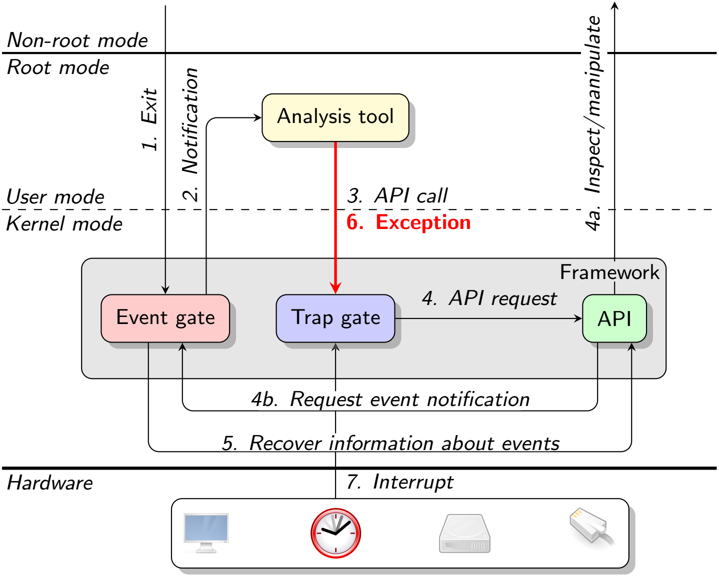
Architecture



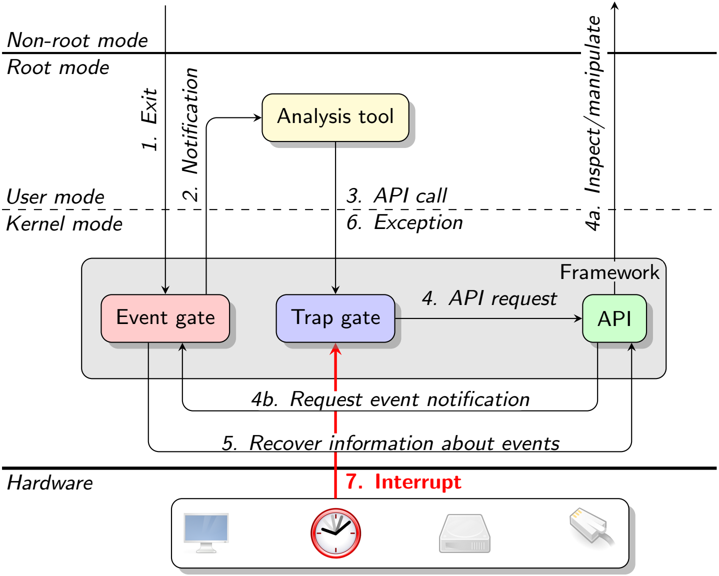
Architecture



Architecture



Architecture



Which events can be intercepted?

- Events cause exits to root mode
- All the events exit conditionally
- Conditions are expressed as boolean conditions

  $(\texttt{process\_name} = \text{“}\texttt{notepad.exe}\text{''} \wedge \texttt{syscall\_name} = \text{“}\texttt{NtReadFile}\text{''})$

Which events can be intercepted?

- Events cause exits to root mode
- All the events exit conditionally
- Conditions are expressed as boolean conditions
  (process_name = “notepad.exe″ ∧ syscall_name = “NtReadFile″)

Native events vs high-level events

- Traced directly through the hardware
- Very low-level operations (e.g., CPU exception)

- Traced through low-/high-level events
- High-level operations (e.g., Return from function)

A summary of the events

Event	Exit cause	Native exit
`ProcessSwitch`	Change of page table address	√
`Exception`	Exception	√
`Interrupt`	Interrupt	√
`BreakpointHit`	Debug or page fault except.	
`WatchpointHit`	Page fault except.	
`FunctionEntry`	Break on function entry point	
`FunctionExit`	Break on return address	
`SyscallEntry`	Break on syscall entry point	
`SyscallExit`	Break on return address	
`IOOperationPort`	Port read/write	√
`IOOperationMmap`	Watchpoint on device memory	

High-Level Events

- Two main high-level events: watchpoints and breakpoints
- Other high-level events are traced through the previous ones (e.g., `FunctionEntry`, `SyscallEntry`, . . .)

High-Level Events

- Two main high-level events: watchpoints and breakpoints
- Other high-level events are traced through the previous ones (e.g., `FunctionEntry`, `SyscallEntry`, ...)

How to set watchpoints and breakpoints from **root mode**?

High-Level Events

- Two main high-level events: watchpoints and breakpoints
- Other high-level events are traced through the previous ones (e.g., `FunctionEntry`, `SyscallEntry`, ...)

How to set watchpoints and breakpoints from **root mode**?

Watchpoints

- No native support from VT-x, few hardware watchpoints shared with the guest
- Implemented by protecting memory pages and trapping access exceptions

High-Level Events

- Two main high-level events: watchpoints and breakpoints
- Other high-level events are traced through the previous ones (e.g., `FunctionEntry`, `SyscallEntry`, . . .)

How to set watchpoints and breakpoints from **root mode**?

Breakpoints

- No native support from VT-x, few hardware breakpoints shared with the guest
- Software breakpoints are efficient, but can be detected (the byte at the breakpoint address must be modified)
- Alternatively, breakpoints can be implemented through watchpoints (transparent but not very efficient)

State inspection and manipulation

CPU registers

- Inspection & manipulation is trivial
- Guest registers are stored inside the VMCS

Memory

- Memory inspection & manipulation requires MMU virtualization
- We mimic the behavior of the hardware MMU to translate VA → PHY and map the physical page

OS-dependent interface

- OS-independent analysis can be uncomfortable
  (e.g., refer to a process by means of its PT base address)
- OS-dependent APIs can ease the analysis
  (e.g., refer to a process through its name)

OS-dependent interface

- OS-independent analysis can be uncomfortable
  (e.g., refer to a process by means of its PT base address)
- OS-dependent APIs can ease the analysis
  (e.g., refer to a process through its name)

Name	**Description**
`GetFuncAddr(`n`)`	Return the address of the function *n*
`GetFuncName(`a`)`	Return the name of the function at address *a*
`GetProcName(`p`)`	Get the name of process with page directory base address *p*
`GetProcPID(`p`)`	Get the PID of process with page directory base address *p*
`GetProcLibs(`p`)`	Enumerate DLLs loaded into process *p*
`GetProcStack(`p`)`	Get the stack base for process *p*
`GetProcHeap(`p`)`	Get the heap base for process *p*
`GetProcList()`	Enumerate processes
`GetDriverList()`	Enumerate device drivers

OS-dependent interface

- OS-independent analysis can be uncomfortable
  (e.g., refer to a process by means of its PT base address)
- OS-dependent APIs can ease the analysis
  (e.g., refer to a process through its name)

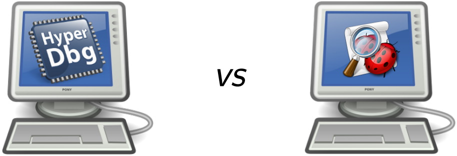
Name	**Description**
`GetFuncAddr(`n`)`	Return the address of the function n
`GetFuncName(`a`)`	Return the name of the function at address a
`GetProcName(`p`)`	Get the name of process with page directory base address p
`GetProcPID(`p`)`	Get the PID of process with page directory base address p
`GetProcLibs(`p`)`	Enumerate DLLs loaded into process p
`GetProcStack(`p`)`	Get the stack base for process p
`GetProcHeap(`p`)`	Get the heap base for process p
`GetProcList()`	Enumerate processes
`GetDriverList()`	Enumerate device drivers

Current implementation supports only
Microsoft Windows XP

HyperDbg: The key advantages

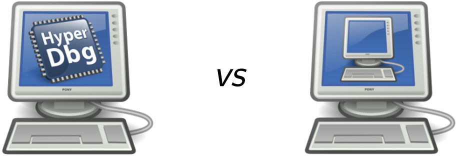
- A kernel debugger built on top of our framework
- Offers common kernel-debugging features
  (e.g., setting breakpoints and watchpoints, single-stepping, . . .)
- OS-independent and grants complete transparency to guest OS and its applications

HyperDbg: The key advantages



- Transparent to the guest OS
- (Almost) OS independent
- Fault resistant
- Debug **any** component, even critical ones
  (e.g., the scheduler, interrupt handlers, . . .)
- No need for a second machine (WinDbg)

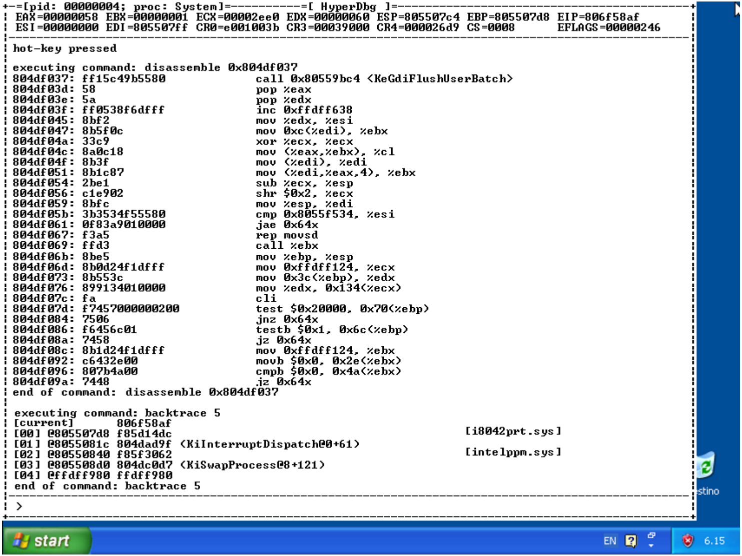
HyperDbg: The key advantages



- Installed as the system *runs*
- Direct interaction with the underlying hardware
- No need to deprivilege or modify the guest OS
- Software virtualizers are **not** so transparent. . .

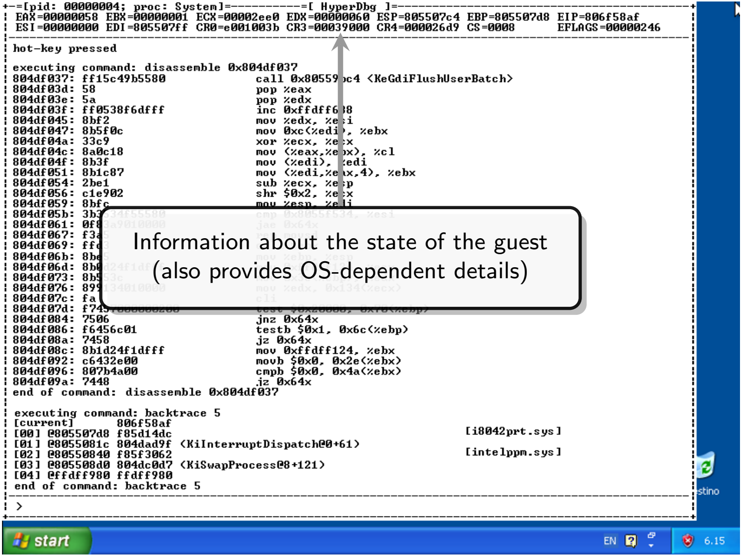
**Testing system virtual machines
(ISSTA '10)**

HyperDbg: Graphical User Interface



```
+-=[pid: 00000004; proc: System]=----------=[ HyperDbg ]=----------------------------------------+
¦ EAX=00000058 EBX=00000001 ECX=00002ee0 EDX=00000060 ESP=805507c4 EBP=805507d8 EIP=806f58af     ¦
¦ ESI=00000000 EDI=805507ff CR0=e001003b CR3=00039000 CR4=000026d9 CS=0008      EFLAGS=00000246 ¦
¦------------------------------------------------------------------------------------------------¦
¦ hot-key pressed                                                                                ¦
¦                                                                                                ¦
¦ executing command: disassemble 0x804df037                                                      ¦
¦ 804df037: ff15c49b5580                call 0x80559bc4 <KeGdiFlushUserBatch>                    ¦
¦ 804df03d: 58                          pop %eax                                                 ¦
¦ 804df03e: 5a                          pop %edx                                                 ¦
¦ 804df03f: ff0538f6dfff                inc 0xffdff638                                           ¦
¦ 804df045: 8bf2                        mov %edx, %esi                                           ¦
¦ 804df047: 8b5f0c                      mov 0xc(%edi), %ebx                                      ¦
¦ 804df04a: 33c9                        xor %ecx, %ecx                                           ¦
¦ 804df04c: 8a0c18                      mov (%eax,%ebx), %cl                                     ¦
¦ 804df04f: 8b3f                        mov (%edi), %edi                                         ¦
¦ 804df051: 8b1c87                      mov (%edi,%eax,4), %ebx                                  ¦
¦ 804df054: 2be1                        sub %ecx, %esp                                           ¦
¦ 804df056: c1e902                      shr $0x2, %ecx                                           ¦
¦ 804df059: 8bfc                        mov %esp, %edi                                           ¦
¦ 804df05b: 3b3534f55580                cmp 0x8055f534, %esi                                     ¦
¦ 804df061: 0f83a9010000                jae 0x64x                                                ¦
¦ 804df067: f3a5                        rep movsd                                                ¦
¦ 804df069: ffd3                        call %ebx                                                ¦
¦ 804df06b: 8be5                        mov %ebp, %esp                                           ¦
¦ 804df06d: 8b0d24f1dfff                mov 0xffdff124, %ecx                                     ¦
¦ 804df073: 8b553c                      mov 0x3c(%ebp), %edx                                     ¦
¦ 804df076: 899134010000                mov %edx, 0x134(%ecx)                                    ¦
¦ 804df07c: fa                          cli                                                      ¦
¦ 804df07d: f7457000000200              test $0x20000, 0x70(%ebp)                                ¦
¦ 804df084: 7506                        jnz 0x64x                                                ¦
¦ 804df086: f6456c01                    testb $0x1, 0x6c(%ebp)                                   ¦
¦ 804df08a: 7458                        jz 0x64x                                                 ¦
¦ 804df08c: 8b1d24f1dfff                mov 0xffdff124, %ebx                                     ¦
¦ 804df092: c6432e00                    movb $0x0, 0x2e(%ebx)                                    ¦
¦ 804df096: 807b4a00                    cmpb $0x0, 0x4a(%ebx)                                    ¦
¦ 804df09a: 7448                        jz 0x64x                                                 ¦
¦ end of command: disassemble 0x804df037                                                         ¦
¦                                                                                                ¦
¦ executing command: backtrace 5                                                                 ¦
¦ [current]      806f58af                                                                        ¦
¦ [00] @805507d8 f85d14dc                                   [i8042prt.sys]                       ¦
¦ [01] @8055081c 804dad9f <KiInterruptDispatch@0+61>                                             ¦
¦ [02] @80550840 f85f3062                                   [intelppm.sys]                       ¦
¦ [03] @805508d0 804dc0d7 <KiSwapProcess@8+121>                                                  ¦
¦ [04] @ffdff980 ffdff980                                                                        ¦
¦ end of command: backtrace 5                                                                    ¦
¦------------------------------------------------------------------------------------------------¦
¦ >                                                                                              ¦
+------------------------------------------------------------------------------------------------+
```

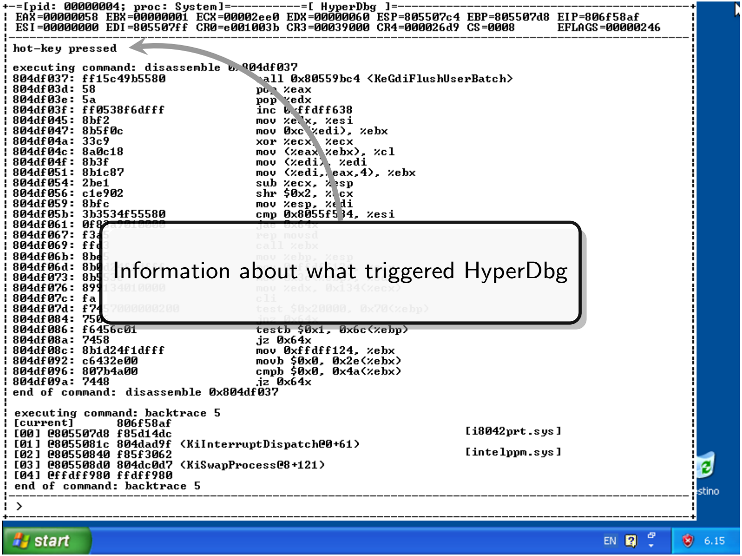
HyperDbg: Graphical User Interface



```
+-=[pid: 00000004; proc: System]=----------=[ HyperDbg ]=---------------------------------------------+
¦ EAX=00000058 EBX=00000001 ECX=00002ee0 EDX=00000060 ESP=805507c4 EBP=805507d8 EIP=806f58af        ¦
¦ ESI=00000000 EDI=805507ff CR0=e001003b CR3=00039000 CR4=000026d9 CS=0008      EFLAGS=00000246     ¦
¦-----------------------------------------------------------------------------------------------------¦
¦ hot-key pressed
¦
¦ executing command: disassemble 0x804df037
¦ 804df037: ff15c49b5580                  call 0x80559bc4 <KeGdiFlushUserBatch>
¦ 804df03d: 58                            pop %eax
¦ 804df03e: 5a                            pop %edx
¦ 804df03f: ff0538f6dfff                  inc 0xffdff638
¦ 804df045: 8bf2                          mov %edx, %esi
¦ 804df047: 8b5f0c                        mov 0xc(%edi), %ebx
¦ 804df04a: 33c9                          xor %ecx, %ecx
¦ 804df04c: 8a0c18                        mov (%eax,%ebx), %cl
¦ 804df04f: 8b3f                          mov (%edi), %edi
¦ 804df051: 8b1c87                        mov (%edi,%eax,4), %ebx
¦ 804df054: 2be1                          sub %ecx, %esp
¦ 804df056: c1e902                        shr $0x2, %ecx
¦ 804df059: 8bfc                          mov %esp, %edi
¦ 804df05b: 3b3534f55580                  cmp 0x8055f534, %esi
¦ 804df061: 0f83a9010000                  jae 0x64x
¦ 804df067: f3a5                          rep movsd
¦ 804df069: ffd3                          [illegible]
¦ 804df06b: 8be5                          mov %ebp, %esp
¦ 804df06d: 8b0d24f1dfff                  mov 0xffdff124, %ecx
¦ 804df073: 8b553c                        mov 0x3c(%ebp), %edx
¦ 804df076: 899134010000                  mov %edx, 0x134(%ecx)
¦ 804df07c: fa                            cli
¦ 804df07d: f7457000000200                test $0x20000, 0x70(%ebp)
¦ 804df084: 7506                          jnz 0x64x
¦ 804df086: f6456c01                      testb $0x1, 0x6c(%ebp)
¦ 804df08a: 7458                          jz 0x64x
¦ 804df08c: 8b1d24f1dfff                  mov 0xffdff124, %ebx
¦ 804df092: c6432e00                      movb $0x0, 0x2e(%ebx)
¦ 804df096: 807b4a00                      cmpb $0x0, 0x4a(%ebx)
¦ 804df09a: 7448                          jz 0x64x
¦ end of command: disassemble 0x804df037
¦
¦ executing command: backtrace 5
¦ [current]      806f58af
¦ [00] @805507d8 f85d14dc                                                   [i8042prt.sys]
¦ [01] @8055081c 804dad9f <KiInterruptDispatch@0+61>
¦ [02] @80550840 f85f3062                                                   [intelppm.sys]
¦ [03] @805508d0 804dc0d7 <KiSwapProcess@8+121>
¦ [04] @ffdff980 ffdff980
¦ end of command: backtrace 5
¦-----------------------------------------------------------------------------------------------------
¦ >
+-----------------------------------------------------------------------------------------------------+
```

Information about the state of the guest
(also provides OS-dependent details)

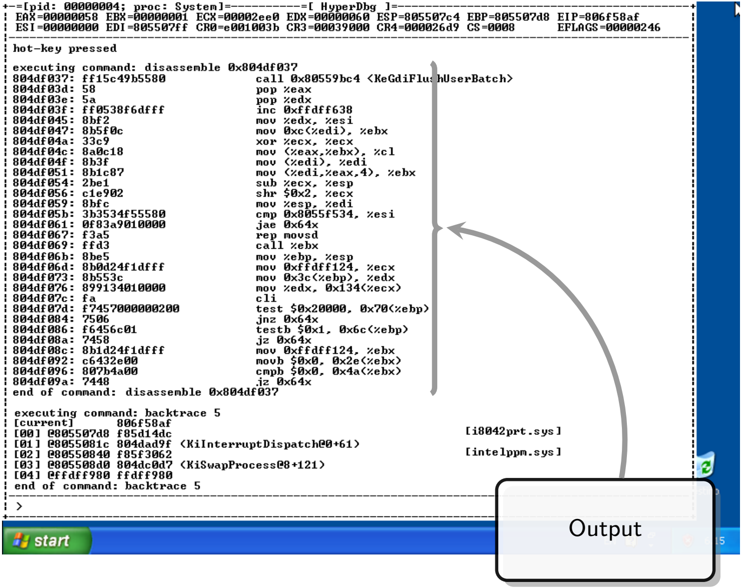
HyperDbg: Graphical User Interface



```
+-=[pid: 00000004; proc: System]=----------=[ HyperDbg ]=------------------------------------------+
¦ EAX=00000058 EBX=00000001 ECX=00002ee0 EDX=00000060 ESP=805507c4 EBP=805507d8 EIP=806f58af      ¦
¦ ESI=00000000 EDI=805507ff CR0=e001003b CR3=00039000 CR4=000026d9 CS=0008      EFLAGS=00000246    ¦
¦--------------------------------------------------------------------------------------------------¦
¦ hot-key pressed                                                                                  ¦
¦                                                                                                  ¦
¦ executing command: disassemble 0x804df037                                                        ¦
¦ 804df037: ff15c49b5580                call 0x80559bc4 <KeGdiFlushUserBatch>                      ¦
¦ 804df03d: 58                          pop %eax                                                   ¦
¦ 804df03e: 5a                          pop %edx                                                   ¦
¦ 804df03f: ff0538f6dfff                inc 0xffdff638                                             ¦
¦ 804df045: 8bf2                        mov %edx, %esi                                             ¦
¦ 804df047: 8b5f0c                      mov 0xc(%edi), %ebx                                        ¦
¦ 804df04a: 33c9                        xor %ecx, %ecx                                             ¦
¦ 804df04c: 8a0c18                      mov (%eax,%ebx), %cl                                       ¦
¦ 804df04f: 8b3f                        mov (%edi), %edi                                           ¦
¦ 804df051: 8b1c87                      mov (%edi,%eax,4), %ebx                                    ¦
¦ 804df054: 2be1                        sub %ecx, %esp                                             ¦
¦ 804df056: c1e902                      shr $0x2, %ecx                                             ¦
¦ 804df059: 8bfc                        mov %esp, %edi                                             ¦
¦ 804df05b: 3b3534f55580                cmp 0x8055f534, %esi                                       ¦
¦ 804df061: 0f83a9010000                jae 0x64x                                                  ¦
¦ 804df067: f3a5                        rep movsd                                                  ¦
¦ 804df069: ffd3                        call %ebx                                                  ¦
¦ 804df06b: 8be5                        mov %ebp, %esp                                             ¦
¦ 804df06d: 8b0d24f1dfff                mov 0xffdff124, %ecx                                       ¦
¦ 804df073: 8b5530                      mov 0x30(%ebp), %edx                                       ¦
¦ 804df076: 899134010000                mov %edx, 0x134(%ecx)                                      ¦
¦ 804df07c: fa                          cli                                                        ¦
¦ 804df07d: f7457000000200              test $0x20000, 0x70(%ebp)                                  ¦
¦ 804df084: 7506                        jnz 0x64x                                                  ¦
¦ 804df086: f6456c01                    testb $0x1, 0x6c(%ebp)                                     ¦
¦ 804df08a: 7458                        jz 0x64x                                                   ¦
¦ 804df08c: 8b1d24f1dfff                mov 0xffdff124, %ebx                                       ¦
¦ 804df092: c6432e00                    movb $0x0, 0x2e(%ebx)                                      ¦
¦ 804df096: 807b4a00                    cmpb $0x0, 0x4a(%ebx)                                      ¦
¦ 804df09a: 7448                        jz 0x64x                                                   ¦
¦ end of command: disassemble 0x804df037                                                           ¦
¦                                                                                                  ¦
¦ executing command: backtrace 5                                                                   ¦
¦ [current]     806f58af                                                                           ¦
¦ [00] @805507d8 f85d14dc                                                 [i8042prt.sys]           ¦
¦ [01] @8055081c 804dad9f <KiInterruptDispatch@0+61>                                               ¦
¦ [02] @80550840 f85f3062                                                 [intelppm.sys]           ¦
¦ [03] @805508d0 804dc0d7 <KiSwapProcess@8+121>                                                    ¦
¦ [04] @ffdff980 ffdff980                                                                          ¦
¦ end of command: backtrace 5                                                                      ¦
¦--------------------------------------------------------------------------------------------------¦
¦ >                                                                                                ¦
+--------------------------------------------------------------------------------------------------+
```

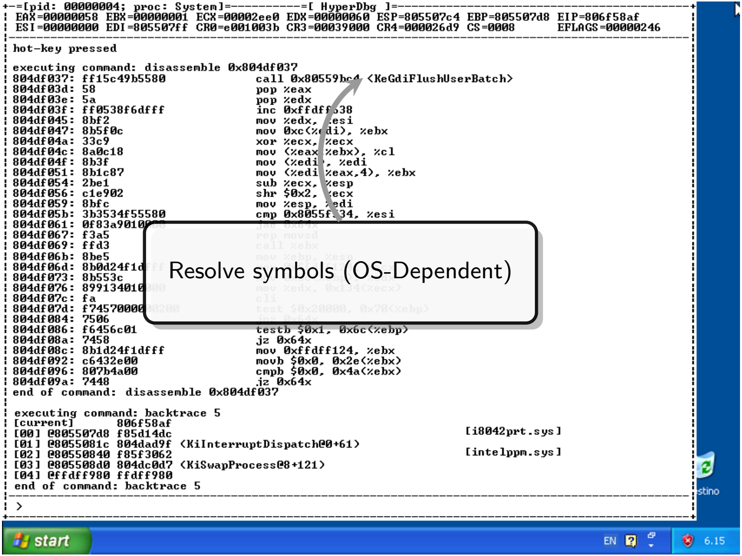
Information about what triggered HyperDbg

HyperDbg: Graphical User Interface



```
+-=[pid: 00000004; proc: System]=----------=[ HyperDbg ]=----------------------------------------+
¦ EAX=00000058 EBX=00000001 ECX=00002ee0 EDX=00000060 ESP=805507c4 EBP=805507d8 EIP=806f58af      ¦
¦ ESI=00000000 EDI=805507ff CR0=e001003b CR3=00039000 CR4=000026d9 CS=0008       EFLAGS=00000246  ¦
¦-------------------------------------------------------------------------------------------------¦
¦ hot-key pressed
¦
¦ executing command: disassemble 0x804df037
¦ 804df037: ff15c49b5580               call 0x80559bc4 <KeGdiFlushUserBatch>
¦ 804df03d: 58                         pop %eax
¦ 804df03e: 5a                         pop %edx
¦ 804df03f: ff0538f6dfff               inc 0xffdff638
¦ 804df045: 8bf2                       mov %edx, %esi
¦ 804df047: 8b5f0c                     mov 0xc(%edi), %ebx
¦ 804df04a: 33c9                       xor %ecx, %ecx
¦ 804df04c: 8a0c18                     mov (%eax,%ebx), %cl
¦ 804df04f: 8b3f                       mov (%edi), %edi
¦ 804df051: 8b1c87                     mov (%edi,%eax,4), %ebx
¦ 804df054: 2be1                       sub %ecx, %esp
¦ 804df056: c1e902                     shr $0x2, %ecx
¦ 804df059: 8bfc                       mov %esp, %edi
¦ 804df05b: 3b3534f55580               cmp 0x8055f534, %esi
¦ 804df061: 0f83a9010000               jae 0x64x
¦ 804df067: f3a5                       rep movsd
¦ 804df069: ffd3                       call %ebx
¦ 804df06b: 8be5                       mov %ebp, %esp
¦ 804df06d: 8b0d24f1dfff               mov 0xffdff124, %ecx
¦ 804df073: 8b553c                     mov 0x3c(%ebp), %edx
¦ 804df076: 899134010000               mov %edx, 0x134(%ecx)
¦ 804df07c: fa                         cli
¦ 804df07d: f7457000000200             test $0x20000, 0x70(%ebp)
¦ 804df084: 7506                       jnz 0x64x
¦ 804df086: f6456c01                   testb $0x1, 0x6c(%ebp)
¦ 804df08a: 7458                       jz 0x64x
¦ 804df08c: 8b1d24f1dfff               mov 0xffdff124, %ebx
¦ 804df092: c6432e00                   movb $0x0, 0x2e(%ebx)
¦ 804df096: 807b4a00                   cmpb $0x0, 0x4a(%ebx)
¦ 804df09a: 7448                       jz 0x64x
¦ end of command: disassemble 0x804df037
¦
¦ executing command: backtrace 5
¦ [current]      806f58af
¦ [00] @805507d8 f85d14dc                                                   [i8042prt.sys]
¦ [01] @8055081c 804dad9f <KiInterruptDispatch@0+61>
¦ [02] @80550840 f85f3062                                                   [intelppm.sys]
¦ [03] @805508d0 804dc0d7 <KiSwapProcess@8+121>
¦ [04] @ffdff980 ffdff980
¦ end of command: backtrace 5
¦-------------------------------------------------------------------------------------------------
¦ >
+-------------------------------------------------------------------------------------------------
```

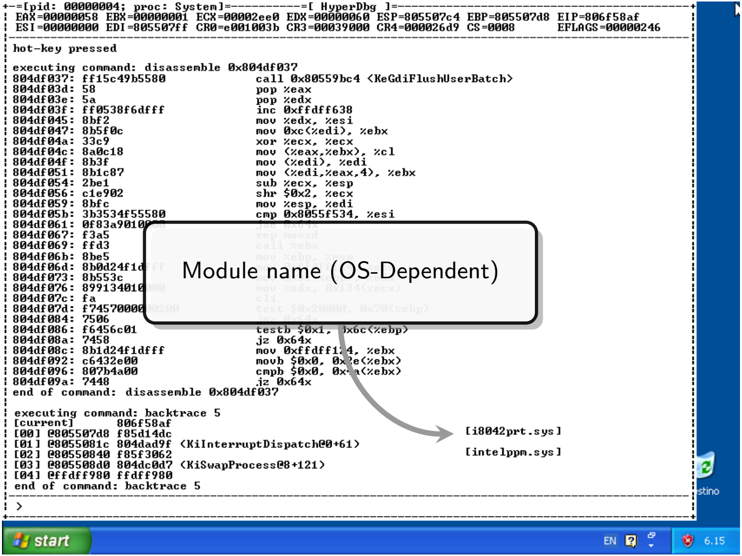
HyperDbg: Graphical User Interface



```
+-=[pid: 00000004; proc: System]=----------=[ HyperDbg ]=-----------------------------------------+
¦ EAX=00000058 EBX=00000001 ECX=00002ee0 EDX=00000060 ESP=805507c4 EBP=805507d8 EIP=806f58af      ¦
¦ ESI=00000000 EDI=805507ff CR0=e001003b CR3=00039000 CR4=000026d9 CS=0008       EFLAGS=00000246  ¦
¦-------------------------------------------------------------------------------------------------¦
¦ hot-key pressed
¦
¦ executing command: disassemble 0x804df037
¦ 804df037: ff15c49b5580                   call 0x80559bc4 <KeGdiFlushUserBatch>
¦ 804df03d: 58                             pop %eax
¦ 804df03e: 5a                             pop %edx
¦ 804df03f: ff0538f6dfff                   inc 0xffdff638
¦ 804df045: 8bf2                           mov %edx, %esi
¦ 804df047: 8b5f0c                         mov 0xc(%edi), %ebx
¦ 804df04a: 33c9                           xor %ecx, %ecx
¦ 804df04c: 8a0c18                         mov (%eax,%ebx), %cl
¦ 804df04f: 8b3f                           mov (%edi), %edi
¦ 804df051: 8b1c87                         mov (%edi,%eax,4), %ebx
¦ 804df054: 2be1                           sub %ecx, %esp
¦ 804df056: c1e902                         shr $0x2, %ecx
¦ 804df059: 8bfc                           mov %esp, %edi
¦ 804df05b: 3b3534f55580                   cmp 0x8055f534, %esi
¦ 804df061: 0f83a9010000                   jae 0x64x
¦ 804df067: f3a5                           rep movsd
¦ 804df069: ffd3                           call %ebx
¦ 804df06b: 8be5                           mov %ebp, %esp
¦ 804df06d: 8b0d24f1dfff                   mov 0xffdff124, %ecx
¦ 804df073: 8b553c                         mov 0x3c(%ebp), %edx
¦ 804df076: 899134010000                   mov %edx, 0x134(%ecx)
¦ 804df07c: fa                             cli
¦ 804df07d: f7457000000200                 test $0x20000, 0x70(%ebp)
¦ 804df084: 7506                           jnz 0x64x
¦ 804df086: f6456c01                       testb $0x1, 0x6c(%ebp)
¦ 804df08a: 7458                           jz 0x64x
¦ 804df08c: 8b1d24f1dfff                   mov 0xffdff124, %ebx
¦ 804df092: c6432e00                       movb $0x0, 0x2e(%ebx)
¦ 804df096: 807b4a00                       cmpb $0x0, 0x4a(%ebx)
¦ 804df09a: 7448                           jz 0x64x
¦ end of command: disassemble 0x804df037
¦
¦ executing command: backtrace 5
¦ [current]      806f58af
¦ [00] @805507d8 f85d14dc                                                  [i8042prt.sys]
¦ [01] @8055081c 804dad9f <KiInterruptDispatch@0+61>
¦ [02] @80550840 f85f3062                                                  [intelppm.sys]
¦ [03] @805508d0 804dc0d7 <KiSwapProcess@8+121>
¦ [04] @ffdff980 ffdff980
¦ end of command: backtrace 5
¦-------------------------------------------------------------------------------------------------¦
¦ >
+-------------------------------------------------------------------------------------------------+
```

Resolve symbols (OS-Dependent)

HyperDbg: Graphical User Interface



```
+-=[pid: 00000004; proc: System]=----------=[ HyperDbg ]=-------------------------------------------+
| EAX=00000058 EBX=00000001 ECX=00002ee0 EDX=00000060 ESP=805507c4 EBP=805507d8 EIP=806f58af       |
| ESI=00000000 EDI=805507ff CR0=e001003b CR3=00039000 CR4=000026d9 CS=0008      EFLAGS=00000246    |
|--------------------------------------------------------------------------------------------------|
| hot-key pressed
|
| executing command: disassemble 0x804df037
| 804df037: ff15c49b5580                   call 0x80559bc4 <KeGdiFlushUserBatch>
| 804df03d: 58                             pop %eax
| 804df03e: 5a                             pop %edx
| 804df03f: ff0538f6dfff                   inc 0xffdff638
| 804df045: 8bf2                           mov %edx, %esi
| 804df047: 8b5f0c                         mov 0xc(%edi), %ebx
| 804df04a: 33c9                           xor %ecx, %ecx
| 804df04c: 8a0c18                         mov (%eax,%ebx), %cl
| 804df04f: 8b3f                           mov (%edi), %edi
| 804df051: 8b1c87                         mov (%edi,%eax,4), %ebx
| 804df054: 2be1                           sub %ecx, %esp
| 804df056: c1e902                         shr $0x2, %ecx
| 804df059: 8bfc                           mov %esp, %edi
| 804df05b: 3b3534f55580                   cmp 0x8055f534, %esi
| 804df061: 0f83a9010000                   jae 0x64x
| 804df067: f3a5                           rep movsd
| 804df069: ffd3                           call %ebx
| 804df06b: 8be5                           mov %ebp, %esp
| 804df06d: 8b0d24f1dfff                   mov 0xffdff124, %ecx
| 804df073: 8b553c                         mov 0x3c(%ebp), %edx
| 804df076: 899134010000                   mov %edx, 0x134(%ecx)
| 804df07c: fa                             cli
| 804df07d: f7457000000200                 test $0x20000, 0x70(%ebp)
| 804df084: 7506                           jnz 0x64x
| 804df086: f6456c01                       testb $0x1, 0x6c(%ebp)
| 804df08a: 7458                           jz 0x64x
| 804df08c: 8b1d24f1dfff                   mov 0xffdff124, %ebx
| 804df092: c6432e00                       movb $0x0, 0x2e(%ebx)
| 804df096: 807b4a00                       cmpb $0x0, 0x4a(%ebx)
| 804df09a: 7448                           jz 0x64x
| end of command: disassemble 0x804df037
|
| executing command: backtrace 5
| [current]       806f58af
| [00] @805507d8 f85d14dc                                             [i8042prt.sys]
| [01] @8055081c 804dad9f <KiInterruptDispatch@0+61>
| [02] @80550840 f85f3062                                             [intelppm.sys]
| [03] @805508d0 804dc0d7 <KiSwapProcess@8+121>
| [04] @ffdff980 ffdff980
| end of command: backtrace 5
|--------------------------------------------------------------------------------------------------|
| >
+--------------------------------------------------------------------------------------------------+
```

Module name (OS-Dependent)

HyperDbg: Implementation

User interface

- We cannot rely on the guest OS graphic libraries
- A small VGA driver to interact with the system's video card
- The driver is neither OS nor hardware dependent

HyperDbg: Implementation

User interface

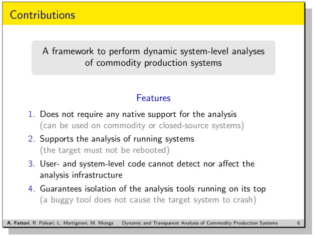
- We cannot rely on the guest OS graphic libraries
- A small VGA driver to interact with the system's video card
- The driver is neither OS nor hardware dependent

User interaction

- An user can activate HyperDbg by pressing an hot-key
- In non-root mode keystrokes are intercepted by leveraging VT-x functionalities (i.e., IOOperationPort events)
- In root mode a simple driver reads the keystrokes

In summary

In summary



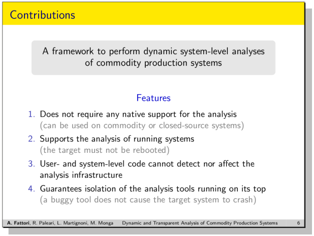
Contributions

A framework to perform dynamic system-level analyses of commodity production systems

Features

1. Does not require any native support for the analysis
   (can be used on commodity or closed-source systems)
2. Supports the analysis of running systems
   (the target must not be rebooted)
3. User- and system-level code cannot detect nor affect the analysis infrastructure
4. Guarantees isolation of the analysis tools running on its top
   (a buggy tool does not cause the target system to crash)

In summary

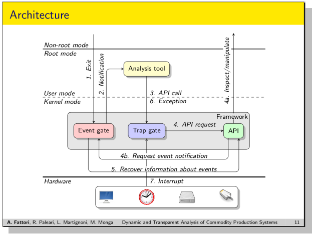


Contributions

A framework to perform dynamic system-level analyses of commodity production systems

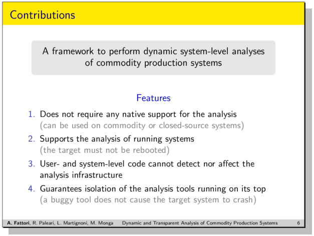
Features

1. Does not require any native support for the analysis (can be used on commodity or closed-source systems)
2. Supports the analysis of running systems (the target must not be rebooted)
3. User- and system-level code cannot detect nor affect the analysis infrastructure
4. Guarantees isolation of the analysis tools running on its top (a buggy tool does not cause the target system to crash)



Architecture

In summary

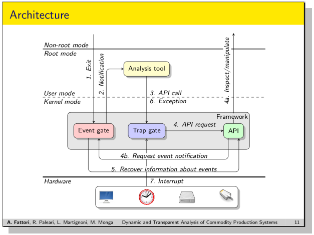


Contributions

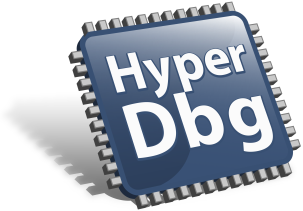
A framework to perform dynamic system-level analyses of commodity production systems

Features

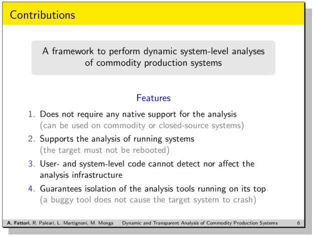
1. Does not require any native support for the analysis
   (can be used on commodity or closed-source systems)
2. Supports the analysis of running systems
   (the target must not be rebooted)
3. User- and system-level code cannot detect nor affect the analysis infrastructure
4. Guarantees isolation of the analysis tools running on its top
   (a buggy tool does not cause the target system to crash)



Architecture



In summary

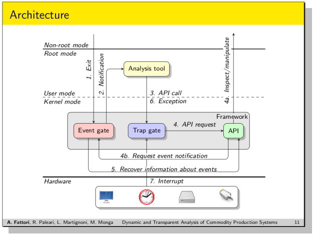


Contributions

A framework to perform dynamic system-level analyses of commodity production systems

Features

1. Does not require any native support for the analysis (can be used on commodity or closed-source systems)
2. Supports the analysis of running systems (the target must not be rebooted)
3. User- and system-level code cannot detect nor affect the analysis infrastructure
4. Guarantees isolation of the analysis tools running on its top (a buggy tool does not cause the target system to crash)



Architecture

Hyper Dbg

http://code.google.com/p/hyperdbg/

Dynamic and Transparent Analysis of Commodity Production Systems

http://code.google.com/p/hyperdbg

Thank you!
Any questions?

Aristide Fattori
aristide@security.dico.unimi.it

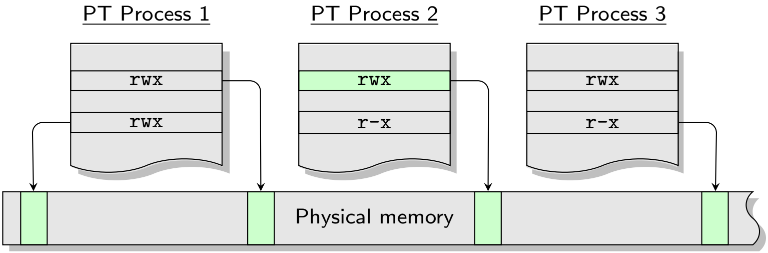
Backup slides

Watchpoints: Details

- Interrupt execution of memory access (read/write)
- Implemented by protecting memory pages and trapping access exceptions

Watchpoints: Details

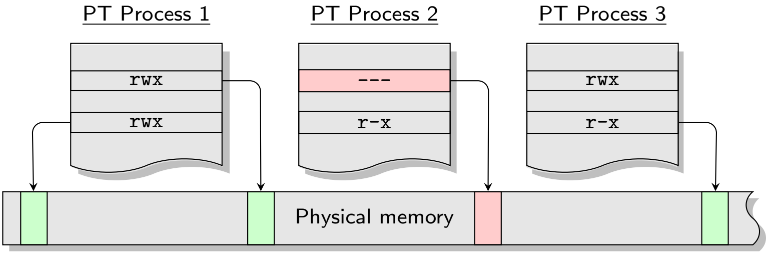
- Interrupt execution of memory access (read/write)
- Implemented by protecting memory pages and trapping access exceptions



Monitor any access to a given memory address

Watchpoints: Details

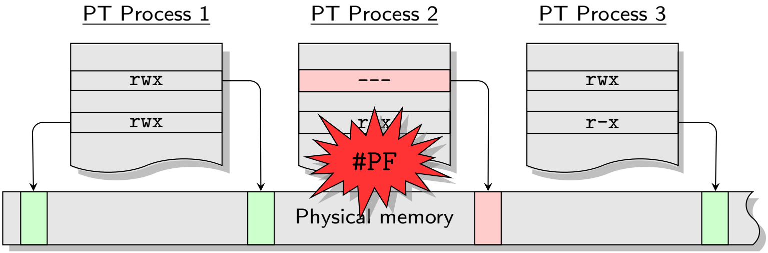
- Interrupt execution of memory access (read/write)
- Implemented by protecting memory pages and trapping access exceptions



Remove any permission from the target page

Watchpoints: Details

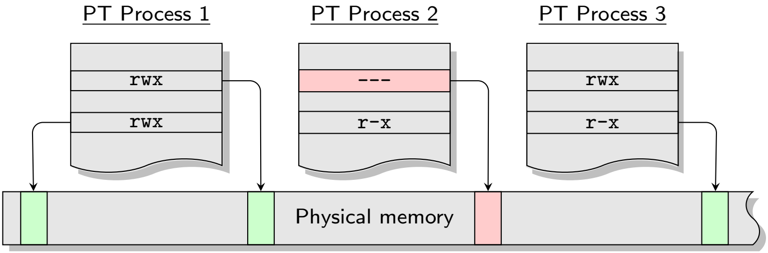
- Interrupt execution of memory access (read/write)
- Implemented by protecting memory pages and trapping access exceptions



Further accesses trigger a CPU exception

Watchpoints: Details

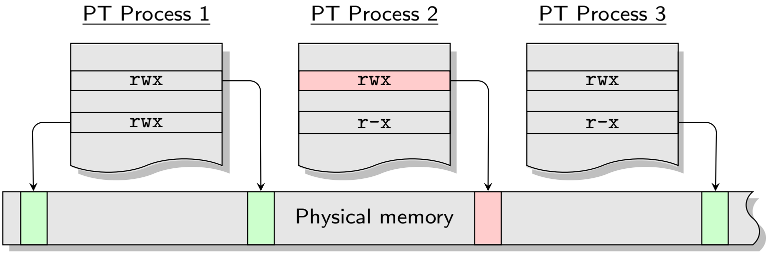
- Interrupt execution of memory access (read/write)
- Implemented by protecting memory pages and trapping access exceptions



If the faulty addr. matches a watchpoint, dispatch the event

Watchpoints: Details

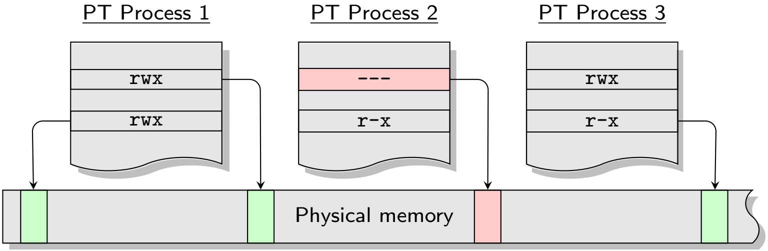
- Interrupt execution of memory access (read/write)
- Implemented by protecting memory pages and trapping access exceptions



Restore the original permissions to resume the execution

Watchpoints: Details

- Interrupt execution of memory access (read/write)
- Implemented by protecting memory pages and trapping access exceptions



To hide watchpoints we modify the entry in which the page table is mapped

(i.e.: we install a Shadow Page Table into the guest operating system with stricter permission than the original PT)

Late launching

- The target system becomes the guest of a virtual machine
- The VMCS is configured to reflect the current state of the guest
- When the framework installation is over, the control is returned to the guest
- The CPU restores the guest state from the VMCS
  (so that the guest execution is resumed just as nothing happened)