CHIPSET BASED APPROACH TO DETECT VIRTUALIZATION MALWARE a.k.a. DeepWatch

Yuriy Bulygin

Joint work with David Samyde

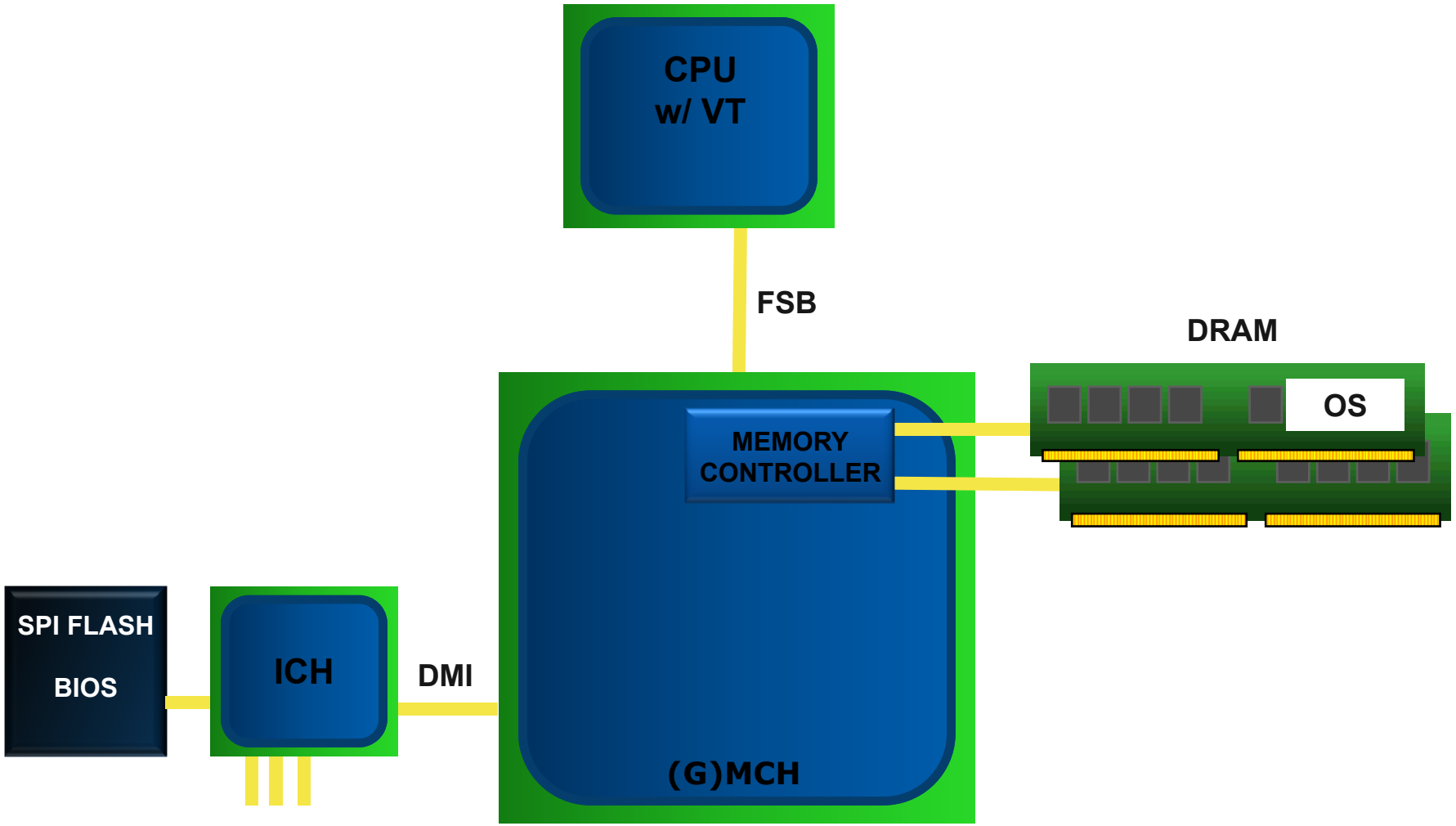
Security Center of Excellence / PSIRT @ Intel Corporation

AGENDA

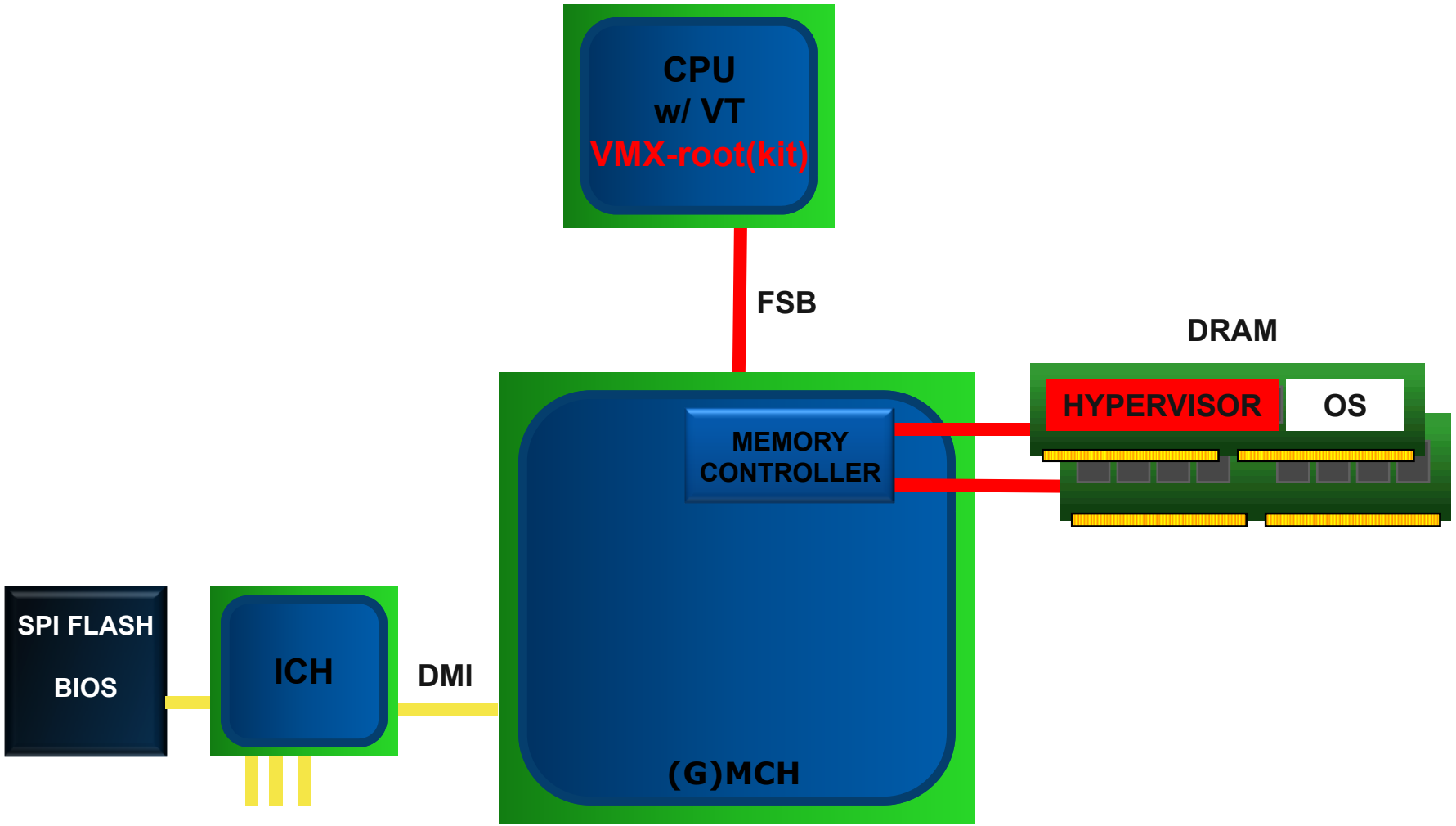
- Introduction
- Chipset based detection of virtualization malware
- DeepWatch: proof of concept detector
- Removing virtualization rootkits
- Detecting SMM rootkits
- Limitations
- Bypassing detection
- Comparing with other detection approaches
- Demo: detecting Intel VT-x based rootkit
- Conclusions

INTRODUCTION

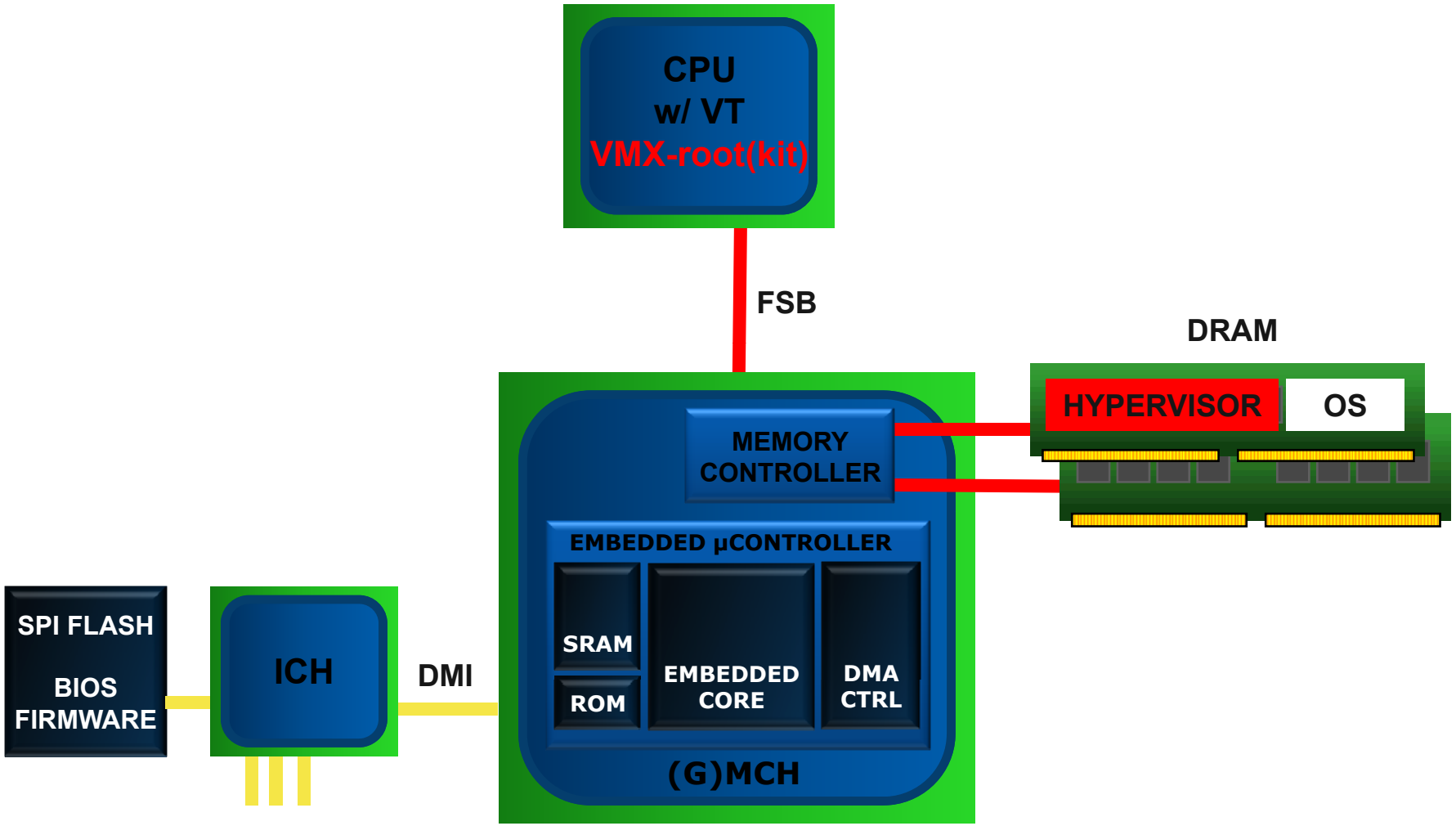
(G)MCH-BASED PLATFORM OVERVIEW



(G)HMM.. PROBLEM



EMBEDDED µCONTROLLER(S)



EMBEDDED µCONTROLLER(S): SUMMARY

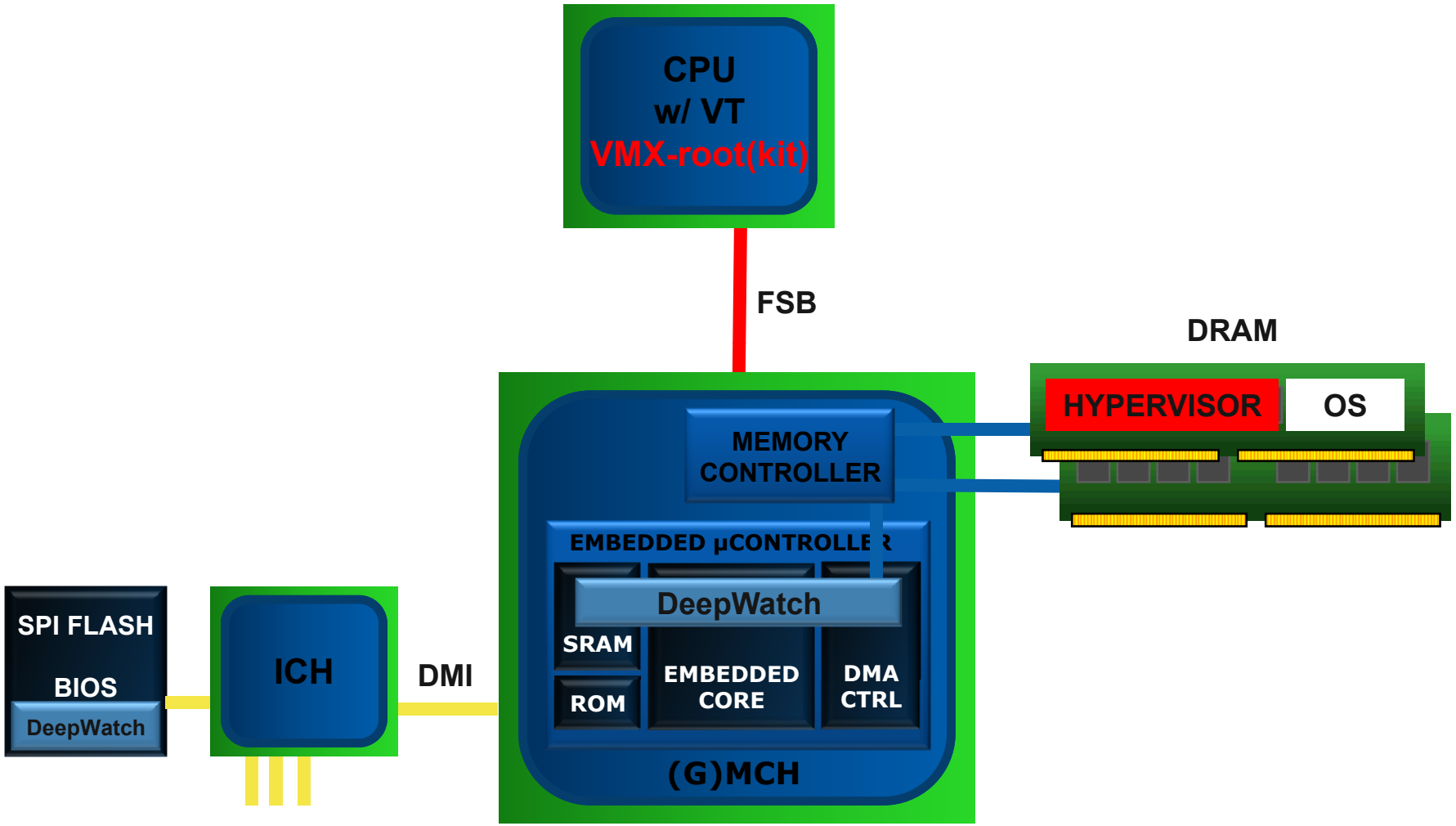
- (G)MCH a.k.a. NorthBridge has embedded microcontroller(s)
- Embedded uController in NorthBridge appears as a separate integrated device on PCI bus with one or more PCI functions
- Assigned with its B/D/F for device enumeration by BIOS
- Embedded uController may integrate different hardware engines such as various bus controllers, PIC, crypto accelerators, DMA engine(s) etc.
- DMA capable embedded uController(s) can be programmed by firmware to access system DRAM
  - host physical addresses (HPA) are used to access system DRAM

EMBEDDED µCONTROLLER(S): FIRMWARE

- Embedded uController runs firmware
  - Real-time operating system (RTOS)
  - Firmware drivers operating hardware engines
  - Firmware applications
- Embedded firmware can operate hardware engines in chipset such as crypto hardware, internal DMA ..
- Internal SRAM memory or memory stolen from DRAM is used for firmware code, stack/heap ..
- External non-volatile memory is used for storing firmware binaries: e.g. SPI Flash
- New Intel chipsets have embedded uController which executes Intel firmware digitally signed and stored in non-volatile SPI Flash memory
- Example: Intel® Active Management Technology (iAMT) [13]

CHIPSET BASED DETECTION AND REMOVAL OF VIRTUALIZATION MALWARE

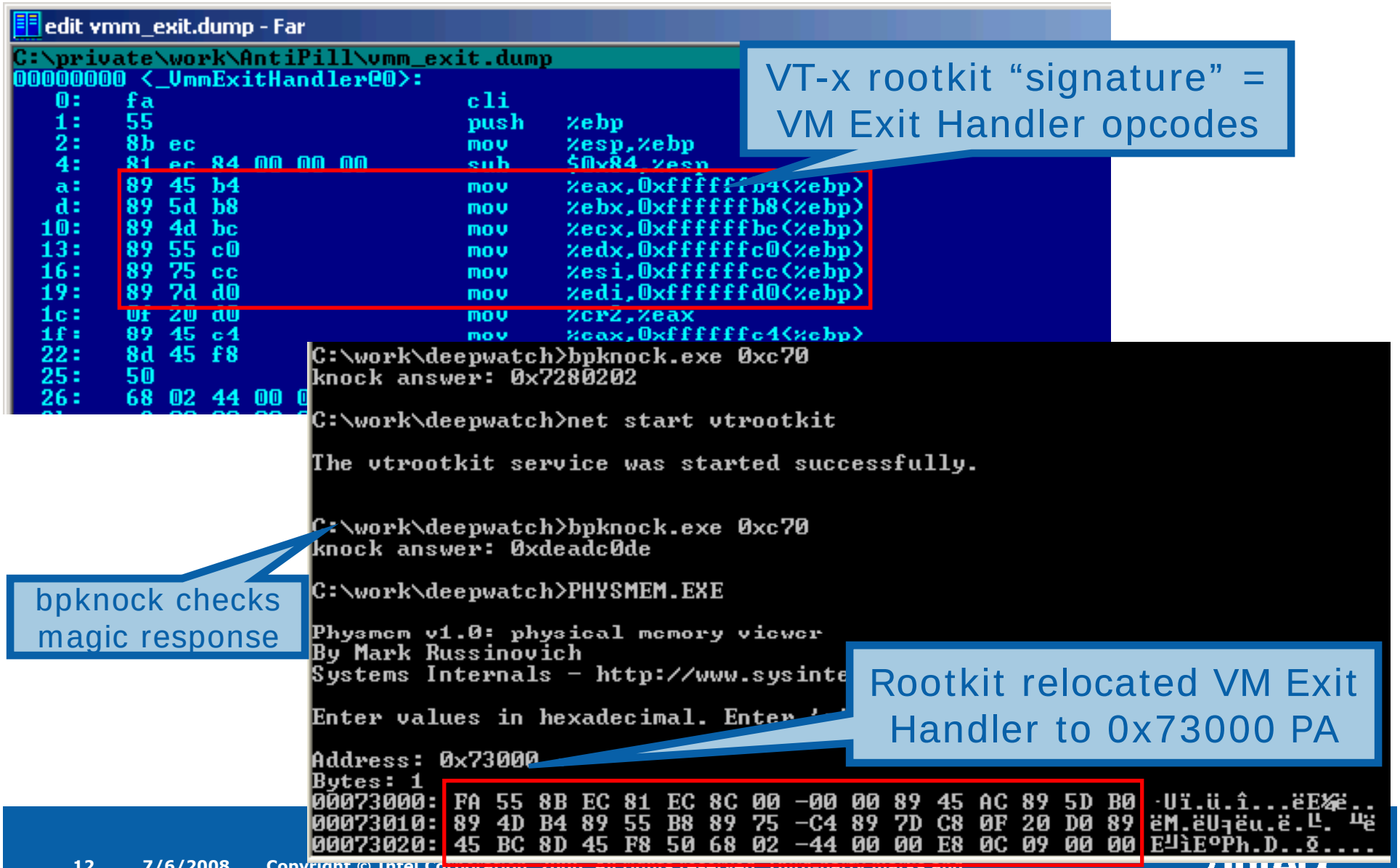
DETECTION IDEA



DETECTION SUMMARY

- Embedded firmware runs on embedded core in the chipset
- It runs “underneath” any hypervisor executing on host CPU
- uController’s internal DMA hardware can be used by embedded firmware to access system memory DRAM
- .. and scan for code/structures of **known** HW virtualization rootkits and **remove** them

VT-x ROOTKIT BRIEF



LET'S PROGRAM DMA MANUALLY

- Programming DMA hardware over JTAG port in debugger
- DMA-ing 64 bytes from system memory containing malicious VMExit handler code to internal chipset memory

```
arc> dump 0x2000000
02000000: 00000000 00000000 00000000 00000000 ¦ ................
02000010: 00000000 00000000 00000000 00000000 ¦ ................
02000020: 00000000 00000000 00000000 00000000 ¦ ................
02000030: 00000000 00000000 00000000 00000000 ¦ ................
02000040: 00000000 00000000 00000000 00000000 ¦ ................
02000050: 00000000 00000000 00000000 00000000 ¦ ................
02000060: 00000000 00000000 00000000 00000000 ¦ ................
02000070: 00000000 00000000 00000000 00000000 ¦ ................
arc> arc:dma(1,0x73000,0,0x2000000,64)
Transferred 64 B of data from Host 0x00073000
General Status = 1
02000000: fa 55 8b ec 81 ec 84 00 00 00 89 45 b4 89 5d b8 ¦ .U.........E..].
02000010: 89 4d bc 89 55 c0 89 75 cc 89 7d d0 0f 20 d0 89 ¦ .M..U..u..}.. ..
02000020: 45 c4 8d 45 f8 50 68 02 44 00 00 e8 b8 08 00 00 ¦ E..E.Ph.D.......
02000030: 8d 45 d4 50 68 1e 68 00 00 e8 aa 08 00 00 8d 45 ¦ .E.Ph.h........E
02000040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ................
02000050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ................
02000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ................
02000070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ¦ ................
arc> dump 0x2000000
02000000: ec8b55fa 0084ec81 45890000 b85d89b4 ¦ .U.........E..].
02000010: 89bc4d89 7589c055 d07d89cc 89d0200f ¦ .M..U..u..}.. ..
```

WHY IS IT INTERESTING ??

- Detects HW virtualization rootkits in system memory regions inaccessible to host OS
- Can remove virtualization rootkits from compromised system
  - E.g. replace malicious #VMEXIT handler with a good one
- Anti-virus vendors can integrate AV engines with DeepWatch in chipsets to detect and remove virtualization rootkits
- Detection can be unnoticeable by the user:
  - doesn't consume host CPU time
  - fast enough to scan entire DRAM (hundreds MBps)
  - can scan memory/SSD while host is in sleep
- Can be used to detect and remove malware rootkits in other reserved memory regions (e.g. SMRAM) inaccessible to OS and anti-viruses
- Could provide hardware based verification of Windows Patch Guard or other OS kernel-mode rootkit detectors

DeepWatch: PROOF OF CONCEPT CHIPSET BASED DETECTOR

"DeepWatch" named after Labyrinth of Reflections novel by Sergei Lukyanenko (http://en.wikipedia.org/wiki/Labyrinth_of_Reflections)

DeepWatch

- DeepWatch is implemented as a PoC in (G)MCH firmware on Intel® Q35 chipset codename “Bearlake”
- Detects Intel® VT-x based rootkits
  - Rootkit is running on Intel® Core 2 Duo CPU and virtualizes Microsoft® Windows® XP SP2
  - Rootkit relocates VM Exit handler code and VMCS structures to legacy address space (below 1MB)
- DeepWatch programs internal DMA hardware to access system memory and scan for signatures of known VT-x based rootkits
- Uses opcode sequence of VM Exit Handler of the rootkit as a signature (can implement more sophisticated detection)
- DMA transfers are done in 32/64 kB chunks

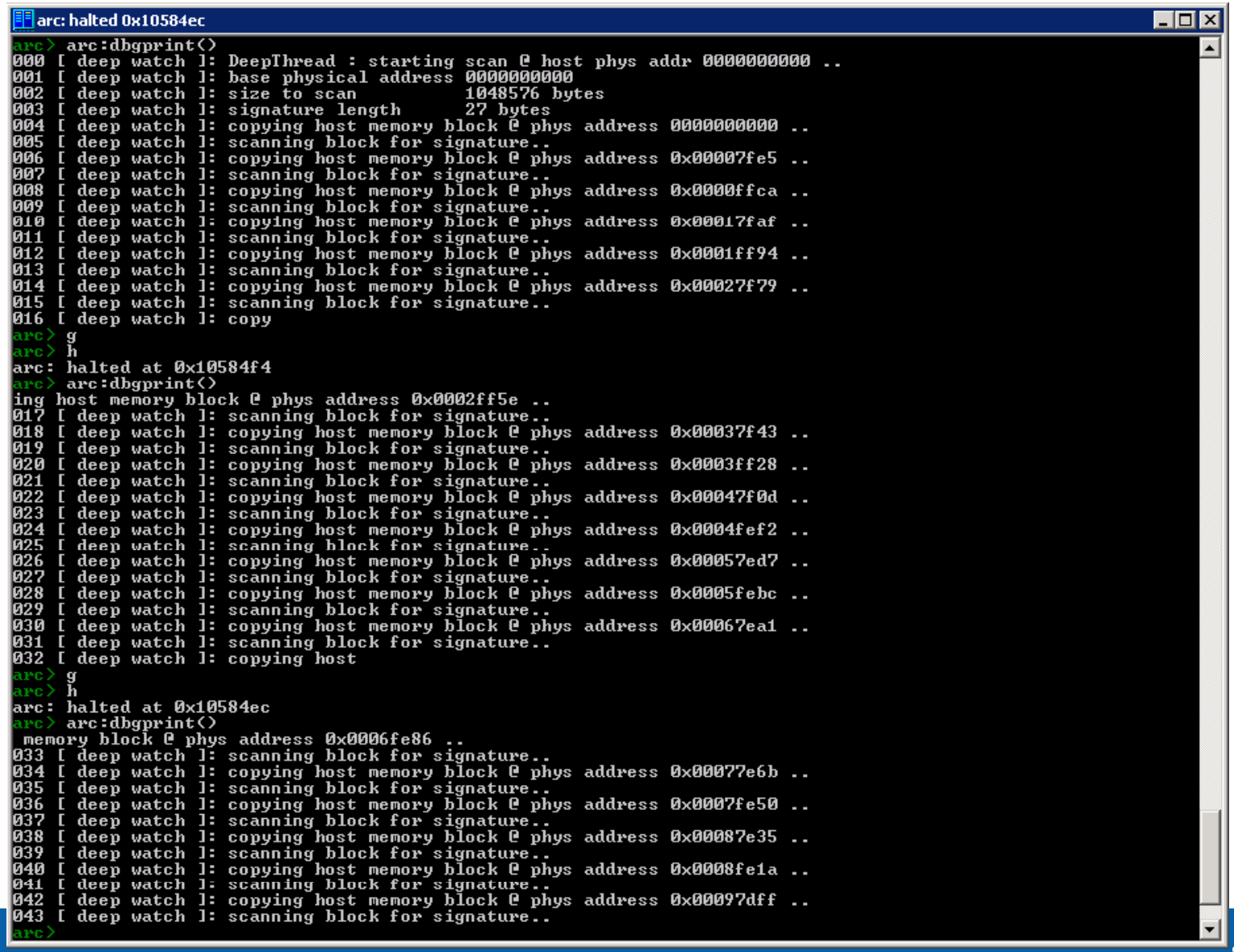
DeepWatch CONT'D

- DeepWatch scanning thread starts during BIOS POST:
  - does DMA copy of DRAM contents to internal **`g_local_buffer`**
  - sets **`g_detected_at`** to DRAM physical address of detected rootkit

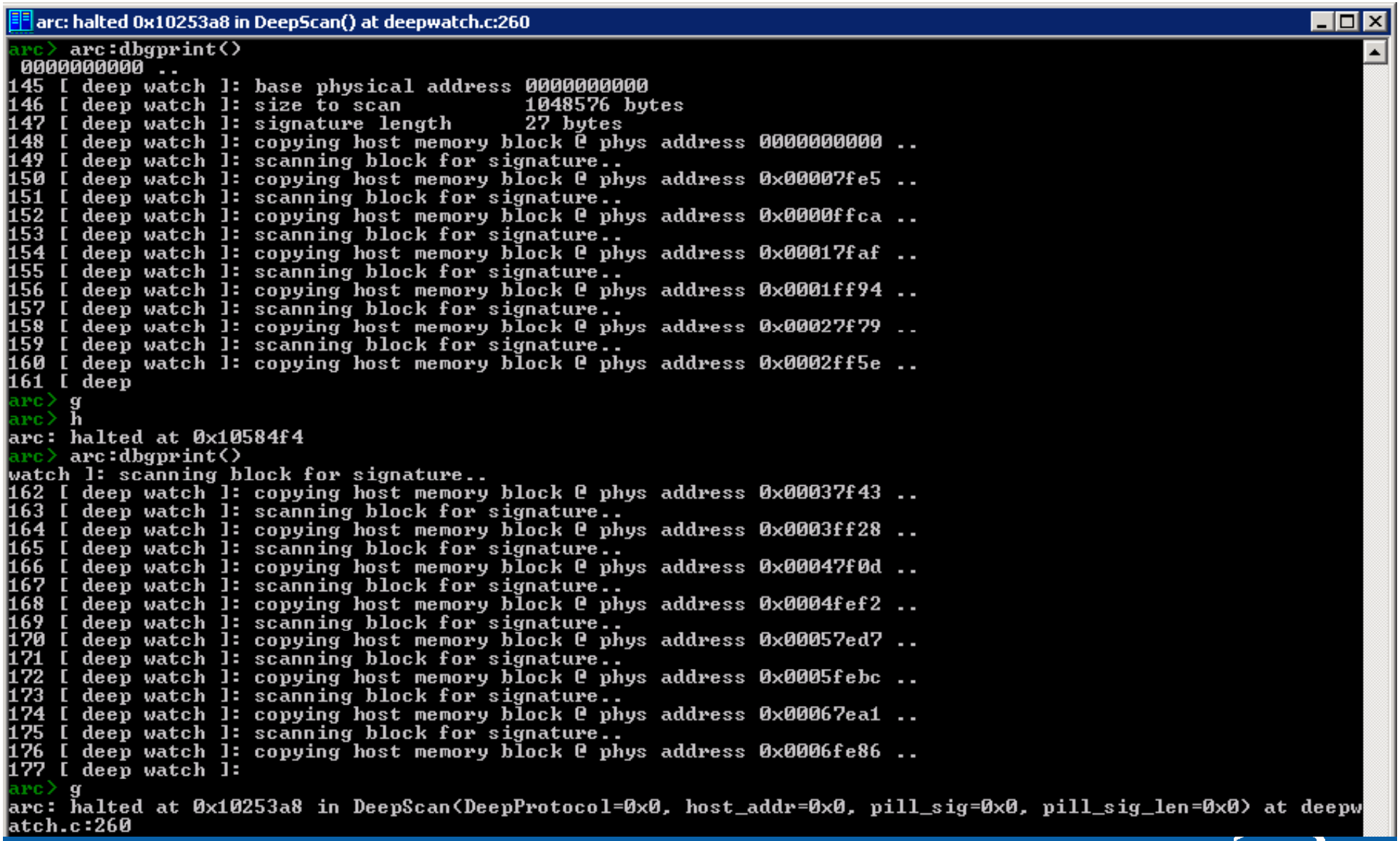
arc: halted 0x10584f4

```
arc> reset global
arc>
arc: halted at 0x1025438 in DeepThread(arg=0x0) at deepwatch.c:292
arc> v g_pill_sig
(UINT8[]) g_pill_sig = 0x1076500
arc> dump 0x1076500 4
01076500: 89b44589 4d89b85d c05589bc 89cc7589 ¦ .E..]..M..U..u..
arc> v g_detected_at
(UINT32) g_detected_at = 0x0
arc> v g_local_buffer
(UINT8[]) g_local_buffer = 0x124e3b8
arc> dump 0x124e3b8 64
0124e3b8: 00000000 00000000 00000000 00000000 ¦ ................
0124e3c8: 00000000 00000000 00000000 00000000 ¦ ................
0124e3d8: 00000000 00000000 00000000 00000000 ¦ ................
0124e3e8: 00000000 00000000 00000000 00000000 ¦ ................
arc> dump 0x125153c 16
0125153c: 00000000 00000000 00000000 00000000 ¦ ................
arc> arc:dbgprint()
```

1: CLEAN SYSTEM

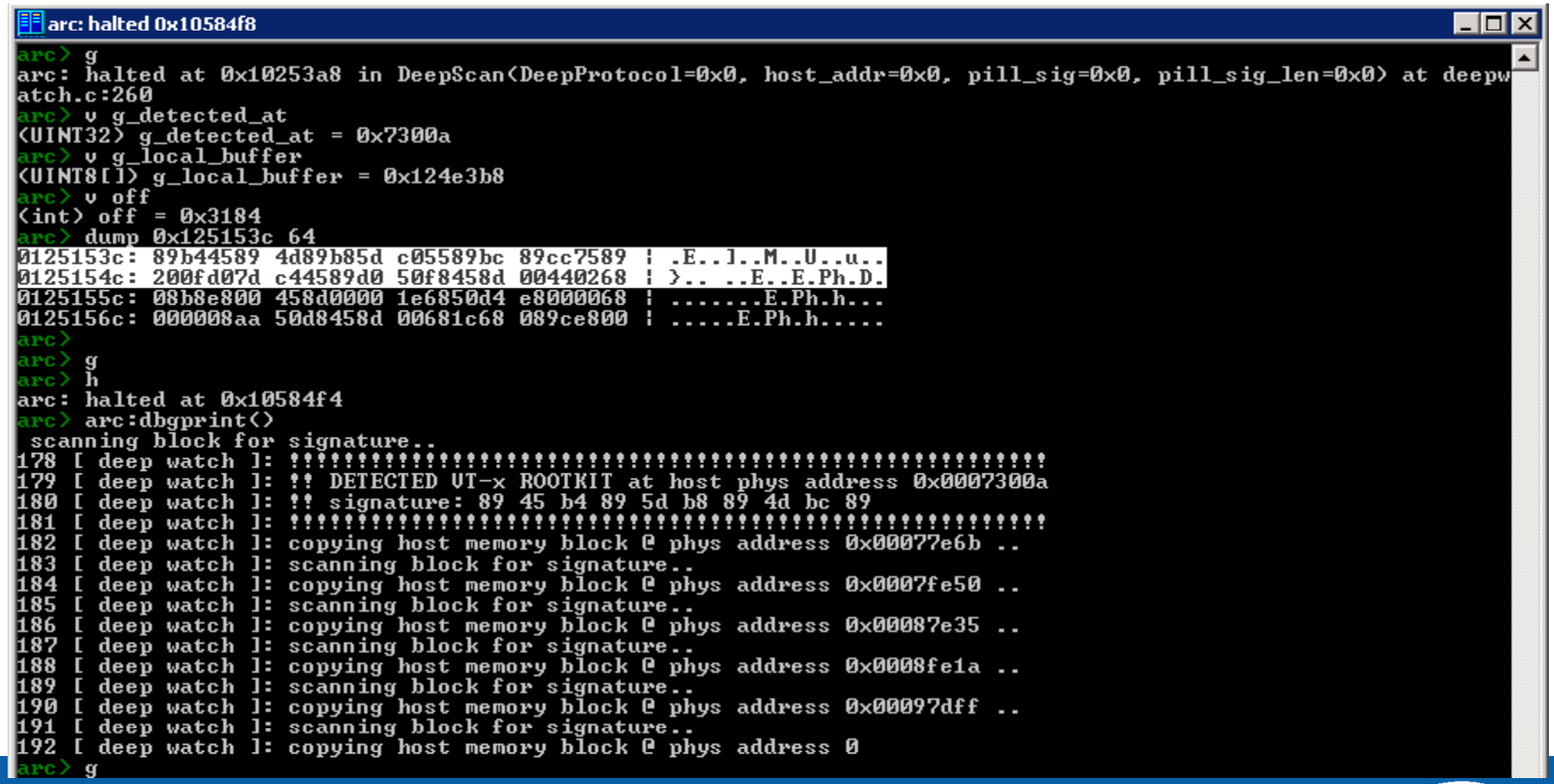


2: ROOTKITTED/VIRTUALIZED SYSTEM



2: DETECTING ROOTKIT SIGNATURE

- Malicious #VMEXIT handler detected at 73000h physical address
  - **g_detected_at** = 0x7300a
  - **g_local_buffer** contains DMA'ed VM Exit handler of the rootkit



REMOVING VIRTUALIZATION ROOTKITS

- DeepWatch not only detects malicious HW hypervisors
- It disinfects host OS from them
- Current implementation:
  - Detect malicious VM Exit Handler executed upon every #VMEXIT
  - Overwrite VM Exit Handler with harmless instructions
  - OS is still virtualized: in VMX non-root (guest) mode
  - But the rootkit does no harm
- Other options:
  - Search and modify/replace host and guest VMCS structures set up by the rootkit
  - Overwrite malicious VM Exit Handler code with VMXOFF opcodes; upon next #VMEXIT rootkit turns off VT (need to carefully restore OS context)

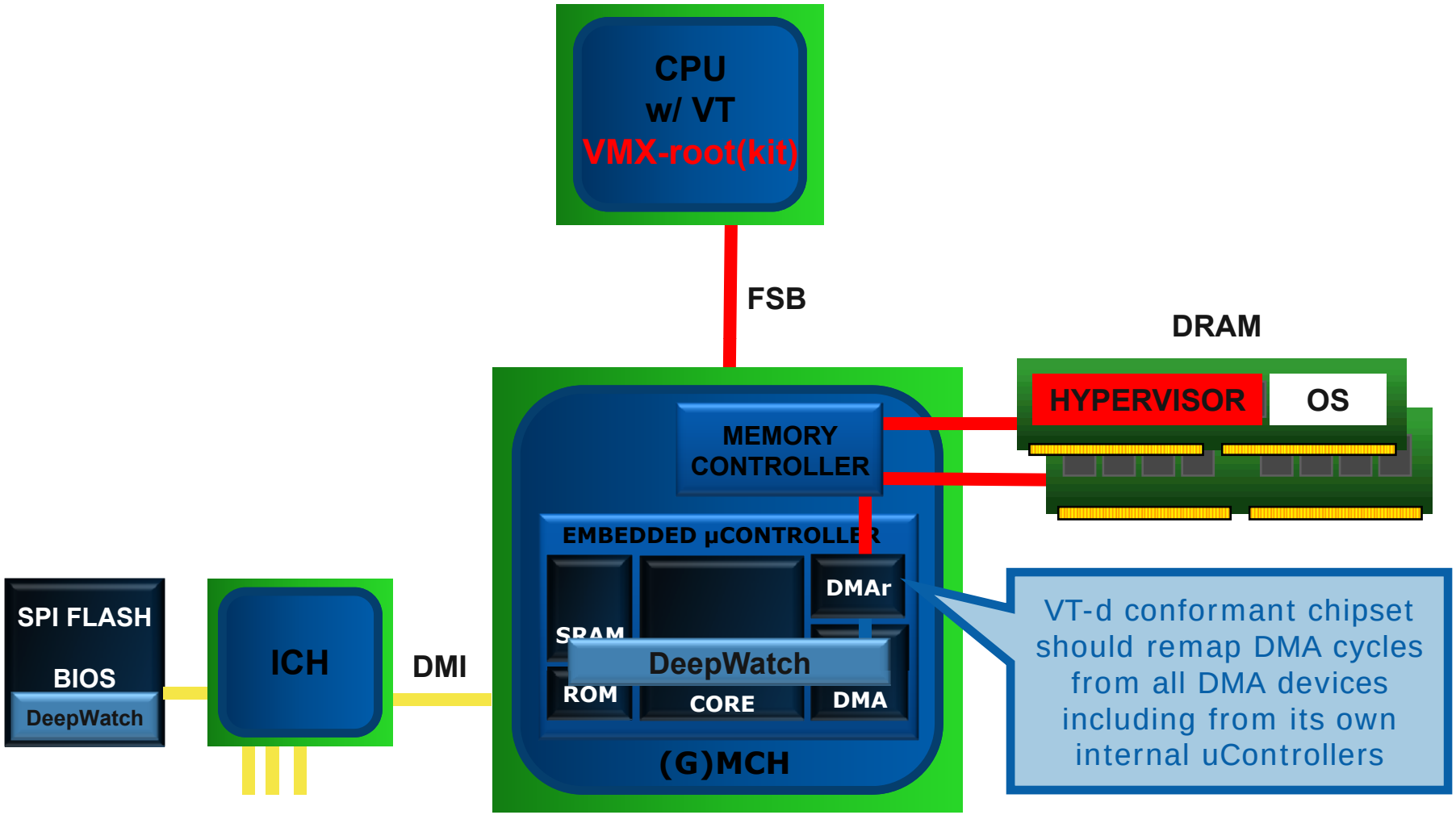
DETECTING SMM ROOTKITS

- SMM malware compromises SMM memory (SMRAM) protections to run in System Management Mode [17-19]
- Both chipset and CPU don't allow non-SMM access to SMRAM after it's locked
  - So anti-viruses cannot scan SMRAM after it's locked by malware
- DeepWatch can detect malicious code in SMRAM provided that chipset allows embedded uController to access SMRAM
  - DMA access to SMRAM by I/O devices is typically prohibited by chipset
- Periodically scan SMRAM for malicious code or/and verify integrity of SMI handlers'

LIMITATIONS

- Detection is OS-independent but chipset-specific
  - Embedded uController and internal DMA hardware are chipset-specific
  - DeepWatch can detect only VT-x based rootkits. It could detect SVM rootkits but AMD CPUs don't work well with Intel chipsets ;)
- DeepWatch is a proof of concept implementation in chipset firmware and uses simple signature matching detection
  - It's not a replacement for an anti-virus managing huge signature bases and multitude of heuristics to detect growing number of malware etc.
  - It can always use integrity checking, white-lists, heuristics or combine them

FORGOT A SMALL DETAIL: DMA REMAPPING



DMA REMAPPING

- VT-d capable chipsets have DMA remapping engine(s) virtualizing Directed I/O access [12]
- Internal DMA devices should also be a subject to DMA remapping
- Chipset has dedicated register-set for each DMA remapping engine accessible by software as MMIO range
- DMA remapping can be programmed by software to protect certain memory regions from certain DMA-capable I/O devices
  - I/O device is identifiable by its PCI Bus and Device
  - Embedded uControllers in chipset have their own PCI B/D/F
  - Create DMA remapping page tables translating DMA virtual address issued by embedded uController's B/D/F to host physical address (HPA)
  - All DMA cycles issued by embedded uController will be translated
- Additionally, certain DMA-protected memory regions (PLMR/PHMR) may be enabled to prevent DMA access
  - E.g. for initializing DMAr structures by trusted VMM

AVOIDING DETECTION BY REMAPPING DMA

To avoid detection

- rootkit can program DMA-remapping engine translating DMA transactions issued by embedded uController running DeepWatch
- relocate its code/data (VM Exit Handler, VMCS structs etc.) to physical memory protected by DMA remapping page tables or to PLMR/PHMR regions

So how to solve it

- DMA-remapping engine may not translate DMA transactions issued by DeepWatch DMA engine distinguishing them from any other DMA transactions
- Trusted software such as SMX authenticated code modules (Intel® TXT) can enable and program DMA-remapping engine for DeepWatch uController

COMPARING WITH OTHER DETECTION APPROACHES

ANOMALY BASED DETECTION

- Timing measurements of instructions causing VMEXIT
  - Local timing: RDTSC, ACPI timer, Local APIC, RTC [8, by bugcheck]
  - Remote timing: NTP [8]
  - Using another thread on SMT CPU to measure #VMEXIT latency [7]
  - Using timers of integrated devices in the chipset
- TLB profiling of instructions causing VMEXIT events
  - Measure timing of address translation due to TLB evictions by a hypervisor [5,6,8,9]
  - TLB coloring: observe if TLB VA-2-PA mappings changed due to VMEXIT [9]
  - Measure # of TLB misses due to flushing TLB's upon VMEntry/#VMEXIT
- Using µArchitectural side-channel attacks
  - RSB based side-channel: corruption of RSB state by VT Exit handler [25]
- Other: CPU errata, causing faults or exhausting resources , Last Branch Record, different CPU behavior in VMX root vs. in non-root modes

CONS

- Do not distinguish good hypervisors from bad
  - *Detected SSDT hook, it may be XXX anti-virus or a kernel rootkit. Remove ??*
- Probabilistic: need to run lots of tests to reduce probability of a false positive
- Hey, we talked about detection only !! How about removing rootkits from the system ??
- Conceptual contradiction: VMX non-root detector is less privileged than the VMX root(kit)
  - Detector shouldn't have any way to affect more VMX root(kit) by design of virtualization
  - Detectors have no legitimate way to remove VT rootkit. VT rootkit can do with detector anything it wants/can
- Agents win ;)

HARDWARE MEMORY ACQUISITION

- Uses DMA capable device to acquire physical memory dump and perform forensic analysis [15]
- A lot of information can be learned about OS or VMM from physical memory dump [20-24]
- DMA remapping (VT-d) hardware in chipsets protects VMM pages from external I/O devices. Virtualization malware can trivially avoid detection
- Method requires discrete DMA capable card. Can IT security folks scan memory of every host physically with external device ??

DEMO:
DETECTING INTEL VT-x BASED ROOTKIT

CONCLUSIONS

- Embedded µControllers in chipsets can be used to reliably **detect** and **remove** HW virtualization malware distinguishing it from legitimate VMM's
- DeepWatch can bring some benefits to traditional anti-malware solutions:
  - Can integrate capabilities of anti-virus engines with hardware capabilities of embedded uControllers in chipsets
  - Can provide hardware based verification of OS kernel anti-rootkit modules (e.g. PatchGuard)
- Can enable detection of other types of emergent malware residing in memory inaccessible to OS/anti-viruses such as SMM rootkits
- Can be used with any detection technique whether it signature scanning, integrity checking or various heuristics

FINAL REMARKS

- This work is now a joint research with Intel Corporate Technology Group, Networking Technologies Lab
- Acknowledgements:

  Dean Krekos, Sagar Dalvi, Jason Fung, Toby Kohlenberg, Howard Herbert

  All authors of research in virtualization rootkits and their detection

Thank you for your attention !!
Any questions ??

yuriy.bulygin@intel.com


intel
Leap ahead™

Intel Security Center of Excellence (SeCoE)

- Evaluates Intel products, platforms and technologies for security vulnerabilities
- Staffs Intel Product Security Incident Response Team (iPSIRT) to respond to security incidents in Intel products
- Drives Secure Development Lifecycle process across Intel products
- Doesn't do all that alone: works with all security architects, technology architects, development and validation teams

secure@intel.com
http://www.intel.com/security

REFERENCES

1. Dino A. Dai Zovi. Hardware Virtualization Rootkits. Black Hat USA 2006 http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf
2. Joanna Rutkowska, Subverting Vista Kernel For Fun And Profit, Black Hat USA 2006, SyScan 2006 http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf
3. Joanna Rutkowska, Alexander Tereshkin. IsGameOver() Anyone? Black Hat USA 2007 http://bluepillproject.org/stuff/IsGameOver.ppt
4. Nate Lawson, Peter Ferrie, Thomas Ptacek. Don't Tell Joanna The Virtualized Rootkit Is Dead. Black Hat USA 2007 https://www.blackhat.com/presentations/bh-usa-07/Ptacek_Goldsmith_and_Lawson/Presentation/bh-usa-07-ptacek_goldsmith_and_lawson.pdf
5. Peter Ferrie. Attacks on More Virtual Machine Emulators http://pferrie.tripod.com/papers/attacks2.pdf
6. Peter Ferrie. Attacks on Virtual Machines. Symantec Corporation http://www.symantec.com/avcenter/reference/Virtual_Machine_Threats.pdf
7. Edgar Barbosa, Blue Pill Detection, COSEINC Advanced Malware Labs, SyScan'07 http://rapidshare.com/files/42452008/detection.rar.html
8. Tal Garfinkel, Keith Adams, Andrew Warfield, Jason Franklin. Compatibility is Not Transparency: VMM Detection Myths and Realities. HotOS 2007 http://www.cs.cmu.edu/~jfrankli/hotos07/vmm_detection_hotos07.pdf
9. Keith Adams, Blue Pill Detection In Two Easy Steps, July 2007 http://x86vmm.blogspot.com/2007/07/bluepill-detection-in-two-easy-steps.html
10. Michael Myers, Stephen Youndt. An Introduction to Hardware-Assisted Virtual Machine (HVM) Rootkits http://crucialsecurity.com/
11. Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide (chapters 19-23) http://www.intel.com/design/processor/manuals/253669.pdf
12. Intel® Vanderpool Technology for IA-32 Processors (VT-x) http://cache-www.intel.com/cd/00/00/19/76/197666_197666.pdf
13. Intel® Virtualization Technology for Directed I/O. Architecture Specification. September 2007 http://download.intel.com/technology/computing/vptech/Intel(r)_VT_for_Direct_IO.pdf
14. Intel® Active Management Technology (iAMT) http://www.intel.com/go/iamt/

REFERENCES

15. Nick L. Petroni, Jr., Timothy Fraser, Jesus Molina, William A. Arbaugh. Copilot - a Coprocessor-based Kernel Runtime Integrity Monitor http://www.cs.umd.edu/~waa/pubs/USENIX-copilot.pdf
16. Joanna Ruthkowska. Beyond the CPU: Defeating Hardware Based RAM Acquisition. Black Hat DC 2007 http://www.blackhat.com/presentations/bh-dc-07/Rutkowska/Presentation/bh-dc-07-Rutkowska-up.pdf
17. John Heasman. Hacking Extensible Firmware Interface (EFI). Black Hat USA 2007 / DEFCON 15 https://www.blackhat.com/presentations/bh-usa-07/Heasman/Presentation/bh-usa-07-heasman.pdf
18. Loïc Duflot. Using CPU System Management Mode to Circumvent Operating System Security Functions. CanSecWest 2006
19. BSDaemon, coideloko, D0nand0n. System Management Mode Hacks http://www.phrack.org/issues.html?issue=65&id=7#article
20. Mariusz Burdach. Physical Memory Forensics. Black Hat USA 2006 http://forensic.seccure.net/pdf/mburdach_physical_memory_forensics_bh06.pdf
21. Andreas Schuster. Searching for processes and threads in Microsoft Windows memory dumps http://dfrws.org/2006/proceedings/2-Schuster.pdf
22. Tobias Klein. Process Dump Analyses: Forensical acquisition and analyses of volatile data http://www.trapkit.de/img/pdf.gif
23. int for(ensic){blog;} -- PTFinder -- KntTools and KntList http://computer.forensikblog.de/en/topics/windows/memory_analysis/
24. Windows Physical Memory Analysis http://windowsir.blogspot.com/2006/03/windows-physical-memory-analysis.html
25. Yuriy Bulygin. CPU side-channels vs. virtualization rootkits: the good, the bad, or the ugly. ToorCon Seattle 2008. http://www.c7zero.info/home.html#hyper-channel