Security Challenges in Virtualized Environments

Joanna Rutkowska
Invisible Things Lab

Who am I? What is ITL?

- Invisible Things Lab's founder/CEO
- ITL focuses on OS security research:
  - kernel infections, advanced malware, effectiveness of OS's anti-malware mechanisms, virtualization security issues
- Right now working with Phoenix Technologies, researching security of effective thin hypervisor implementations
- ITL also does trainings and consultations:
  - "Understanding Stealth Malware" class (BH Vegas)
- Founded in April 2007; currently 2 people and growing :)

What I will be talking about?

- **Virtualization-based rootkits**
  - What is so special about them?
  - Facts & myths about virtualization rootkits
  - How real is this threat today?
- **VMs as Security Boundaries**
  - Isolation provided by VMMs?
  - VMMs vs. Microkernel-based OSes
- **Nested Virtualization**
  - What are the security implications?
  - What are the positive applications?

Virtualization based rootkits

Hardware vs. Software virtualization

S/W based (x86)

- Requires 'emulation' of guest's privileged code
  - can be implemented very efficiently: Binary Translation
- Does not allow full virtualization
  - sensitive unprivileged instructions (SxDT)
- Widely used today
  - VMWare Workstation 6

H/W virtualization

- VT-x (Intel x86/x64)
- SVM/Pacifica (AMD x64)
- Does not require guest's priv code emulation
- Should allow for full virtualization of x86/x64 guests
- Still not very popular in commercial VMMs
  - XEN3, Virtual PC 2007

Full VMMs vs. "Thin hypervisors"

Full VMMs

- Create full system abstraction and isolation for guest,
- Emulation of I/O devices
  - Disks, network cards, graphics cards, BIOS…
- Trivial to detect,
- Usage:
  - server virtualization,
  - malware analysis,
  - Development systems

"Thin hypervisors"

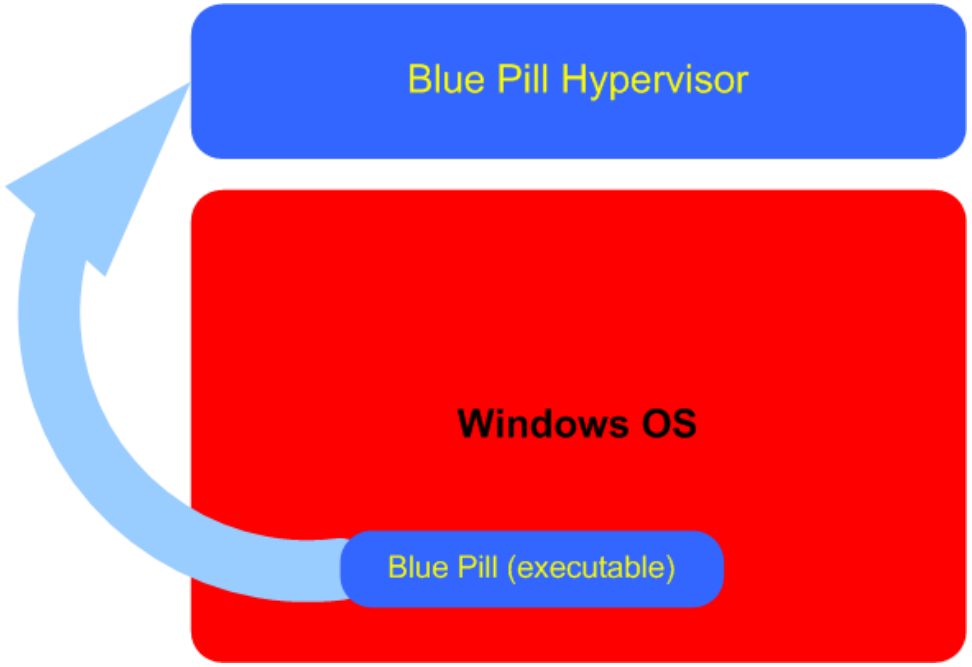
- Transparently control the target machine
- Based on hardware virtualization (SVM, VT-x)
- Isolation might not be a goal!
  - native I/O access
  - Shared address space with guest (sometimes)
- Very hard to detect
- Usage:
  - stealth malware
  - Anti-DRM

Original Blue Pill Proof-Of-Concept

- Originally developed for COSEINC by yours truly,
- Presented at Black Hat 2006 in Las Vegas,
  - Also Dino Dai Zovi presented his Vitriol, which was similar, but worked for Intel VT-x
- COSEINC owns the original Blue Pill code,
- May 2007 – we designed and wrote from scratch the New Blue Pill (NBP)
  - Alex Tereshkin wrote most of the code

Blue Pill Idea

- Exploit AMD64 SVM extensions to move the operating system into the virtual machine (do it 'on-the-fly')
- Provide thin hypervisor to control the OS
- Hypervisor is responsible for controlling "interesting" events inside guest OS



BP installs itself ON THE FLY!

- BP installs itself on the fly
- Thus, no modifications to BIOS, boot sector or system files are necessary,
- BP does not survive system reboot
  - Techniques for "restart surviving" are orthogonal to "BP technology" – e.g. BIOS infection
  - BP, like any other malware, can be made persistent, but this is out of the scope of this presentation
  - In many cases this is not needed, BTW

BP does not virtualize hardware!

- BP and New BP are thin VMMs,
- They do not virtualize I/O devices!
  - If your 3D graphics card worked before BP installation, it will still work with the same performance!
  - Bluepilled systems see the very same hardware as they saw before BP installation – h/w fingerprinting can not be used to detect BP

Memory virtualization (AKA hiding in memory)

- Original Blue Pill didn't virtualize memory!
- The assumption was that the opponent (e.g. an A/V company) doesn't have access to Blue Pill code, because BP is used in targeted attacks:
  - e.g. we generate a polymorphic version of "blue pill" malware separately for each infection
  - also – we do not publish polymorphic generator, so that it's not possible to analyze it
  - plus we make sure to encrypt the VMRUN instruction after resuming the guest.

Scanning for BP code in memory

- Without having a code sample one could find BP in memory only using heuristics, e.g.:
  - Code emulation (i.e. emulate code on each physical page and find out whether it "looks like" a hypervisor.
  - Better: whether it looks like *malicious* hypervisor?
- It's trivial to defeat those detection methods by using "classic" code obfuscation techniques…

Why bother with virtualization?

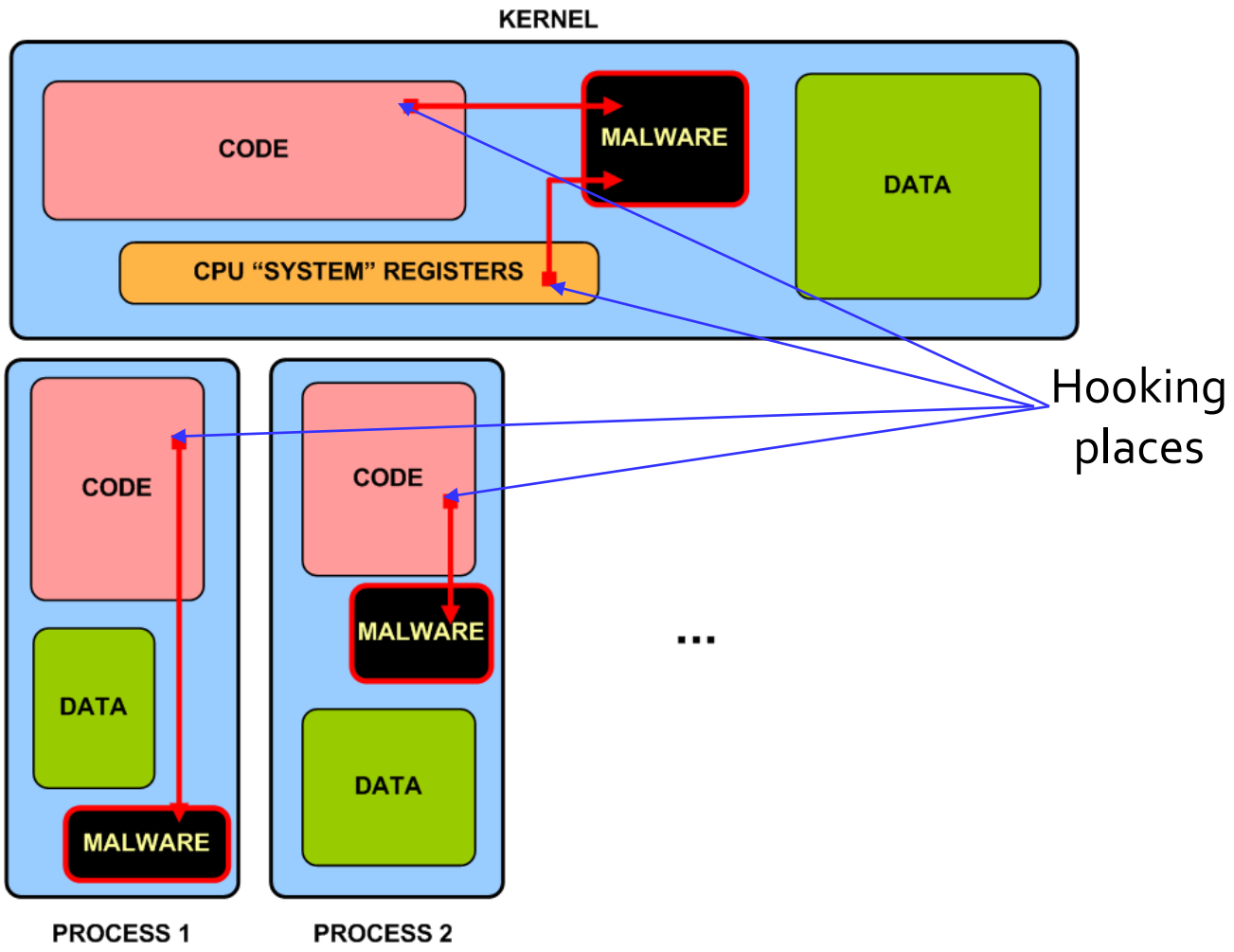
- If we could use "classic code obfuscation" to avoid detection, why bother with virtualization?
- Why not use classic kernel rootkits?

"No hooking" principle

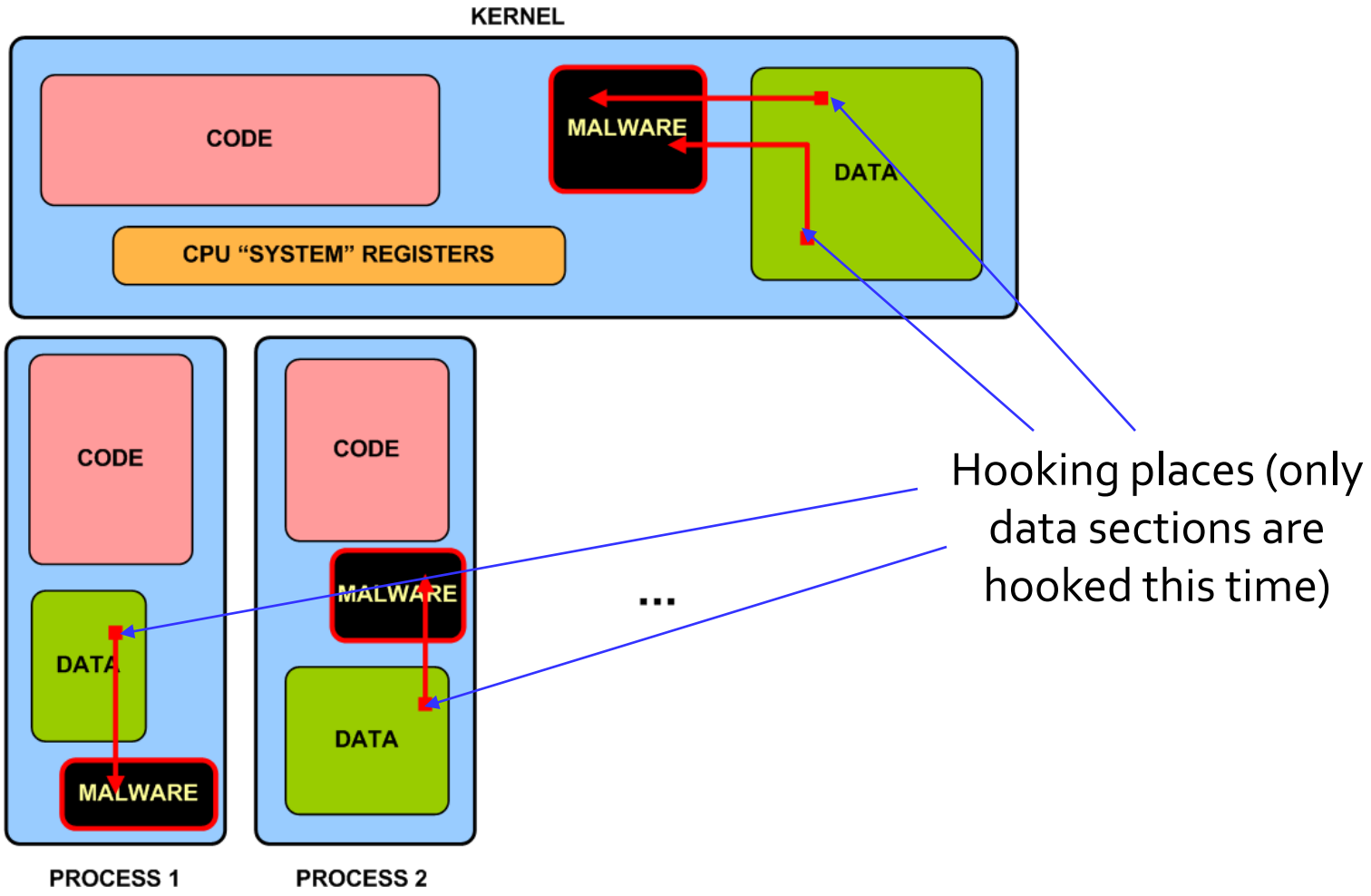
- So what so special about Blue Pill?
- That it doesn't hook even a single byte!
- Other rootkits need to hook something in the system code or at least in OS data sections…
  - thus we *can* always detect them (although this is very hard to do in a generic way)

- BP is an example of type III malware…

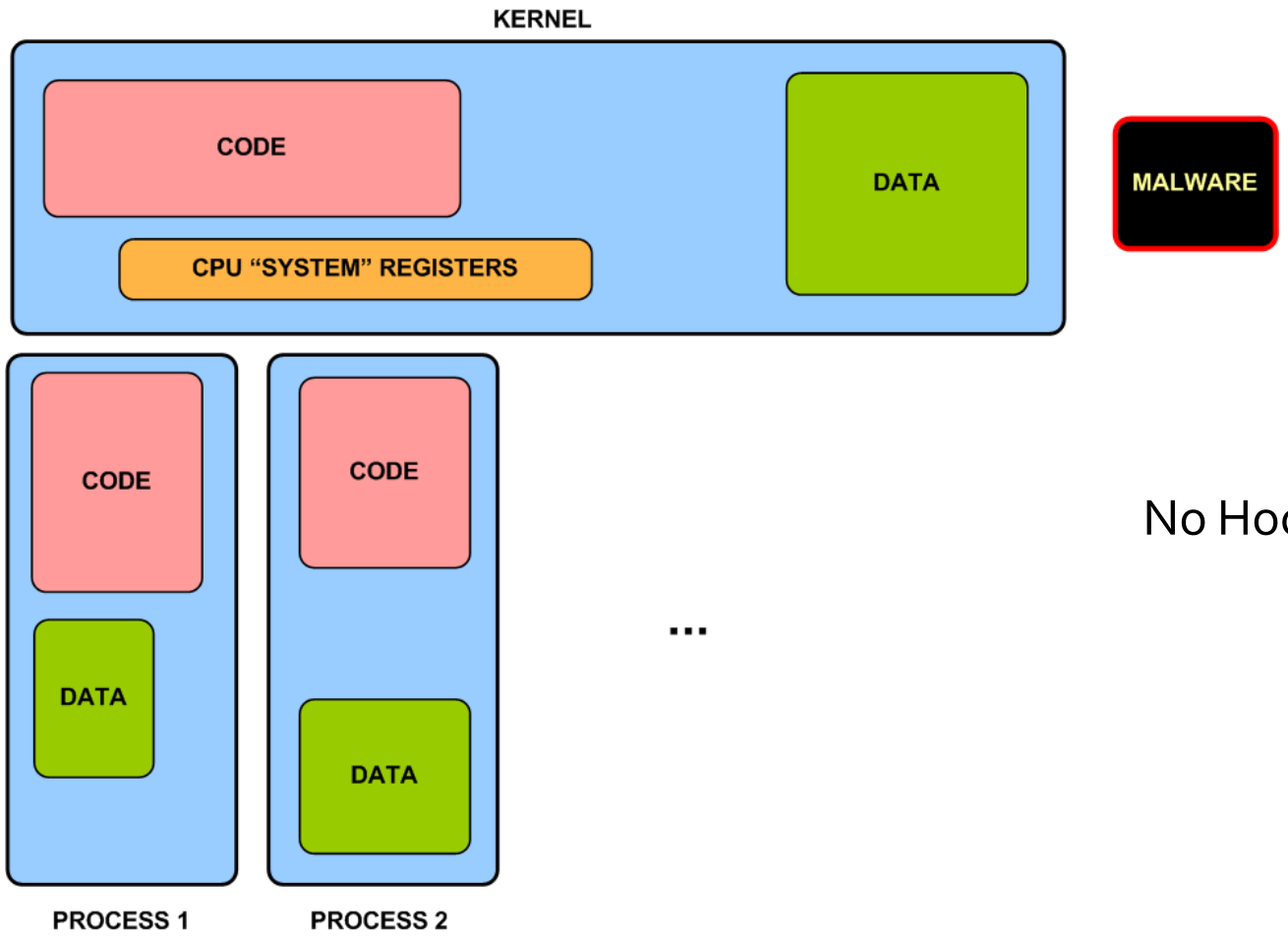
Type I Malware



Type II Malware



Type III Malware



No Hooks!

A perfect Integrity Scanner

- Imagine a *complete* kernel integrity scanner,
  - Something like Patch Guard or SVV, but *complete!*
- Such scanner would be able to detect *any* type I and type II kernel infections…
  - We also assume a reliable memory acquisition used
- In other words – the Holy Grail of rootkit hunters!
- But it still will not be able to detect Type III infections!

What about "massive attacks"?

- Now let's consider using BP for "massive attacks" (in contrast to targeted attacks)
  - For targeted attacks we don't need memory hiding!
- We could use the several strategies for hiding its code in memory…

Other strategies for hiding Blue Pill's code in memory

- Private page tables
- Shadow Page Tables (SPT)
- Nested Paging(AMD)/ ("Hardware SPT")

Memory hiding strategies

- Private page tables (private CR3)
  - Quite easily bypassable (via use custom PTEs)
  - Could be made harder to bypass (page permutations)
- Shadow Page Tables
  - popular method for memory virtualization (all current VMMs use this method)
  - SPT can be detected via performance impact
- Nested Paging (AMD)/Extended Page Tables (Intel)
  - negligible performance impact
  - requires new hardware (not available in shops now)
  - challenge: cheat about the amount of available mem

So is Blue Pill really 100% undetectable?

- Many people claimed they can detect Blue Pill…
- …however they only presented so far methods to detect *virtualization*, not the specific malware!
  - BTW, All the presented methods were based on tricks and hacks that were highly implementation specific (e.g. processor model specific)
- Wrong assumption was made:
  - If OS is executing inside (h/w) VM ➔ we detected virtualization based malware
- It's like assuming that each program that uses networking is a botnet agent!

Unexpected Virtualization detection

- But we know whether we run inside VM or not, right? So we still can detect BP in a situation when we detect an *unexpected* virtualization, right?
- No! Because we should assume that in the coming years "everything" will run inside VM!
  - Most of the servers will be virtualized
  - Desktop users will run various virtualization applications, e.g.:
    - Web Browsers in VM (for security)
    - A/V programs that run in hypervisor

VMM protect against Blue Pill!

- One might argue that if we run a VMM already then it's not possible to install virtualization based malware anymore…
- This is not true! – see the next two sections for more details on this.

Should we be afraid of virtualization based malware today? Tomorrow?

- Today we cannot effectively fight even with relatively simple kernel malware
  - Not even mentioning some more advanced kernel malware (e.g. Type II SbD malware)
  - No motivation to switch to more complex malware
- The amount of machines that support hardware virtualization (SVM or VT-x) is still relatively small

Virtual Machines as Security Boundaries

Reasons for using VMMs

- Server consolidation
  - Business argument, not related to security (although has some security implications)
- Software isolation, e.g.:
  - Trusted Computing
  - running a browser in a VM to allow "safe" browsing
  - malware/suspicious software analysis
- Development/testing
  - Not related to security

Software Isolation

- Originally software isolation was supposed to be provided by Operating Systems
  - separate address space for each process
  - user accounts & ACLs
- Can't current OSes, like Windows or Linux, provide effective isolation?

Why OSes don't provide effective software isolation?

- Bad design/wrong user habits: (XP & Vista)
  - (Almost) Everybody and everything runs as administrator on Windows – this negates all the local OS-security mechanisms!
  - UAC in Vista was announced "not a security boundary" by Microsoft at the beginning of this year!
  - Vista assumes that *every* installer/setup program should be run as Administrator!
- Implementation flaws
  - Bugs in OS core components (rare)
  - Bugs in 3rd party drivers and kernel modules (very common!)

Improving Operating Systems

- Vista shows a trend towards limiting privileges of user's programs (UAC, Protected Mode IE)
  - Even though those mechanisms are not prefect, it's a step towards the right direction
  - MacOSX Tiger also has something similar to UAC
- However…

Buggy Drivers

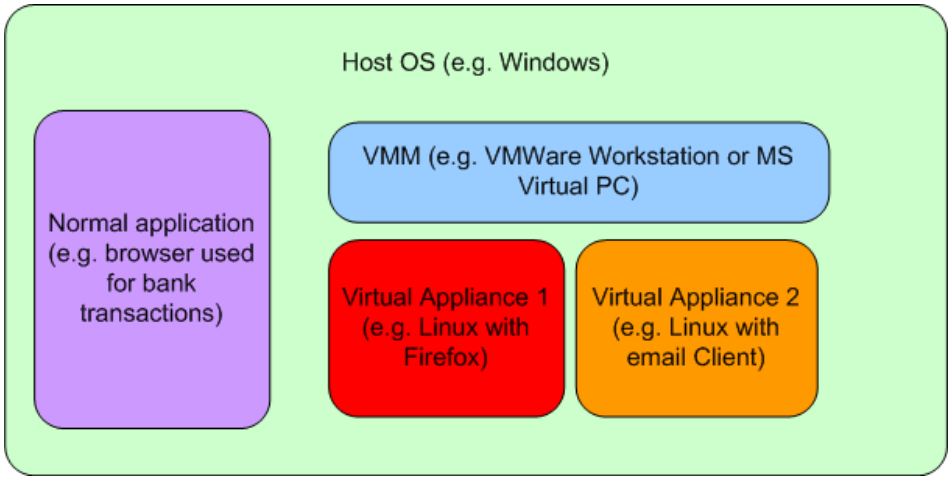
- Still there is a problem of buggy drivers
- All current OS use monolithic kernel architecture
  - Vista, Linux, even MacOSX (even though it uses Mach microkernel as a core, still all the drivers share one address space with the rest of the kernel)
- Monolithic kernel architecture has a big security implication: compromise of a single driver allows to compromise the whole OS!
  - At Black Hat Vegas in August we presented several bugs in 3rd party drivers that could be used to compromise Vista kernel, bypassing Vista kernel protection

Micorkernel based OSes

- The idea is to have a very minimal kernel that provide only very basic services (e.g. communication and scheduling)
- All other services and drivers are kept in separate address spaces
  - Thus even if one driver gets compromised, the rest of the system is still protected
- Microkernel architecture is known for years, but nothing suggests that mainstream vendors will ever adopt this model
  - The main reason is the difficulty for creating efficient drivers for microkernel based OSes

Virtualization for the rescue!

- Instead of changing the architecture of the whole OS…
  - which would e.g. require to rewrite all the drivers
- …we can use virtualization to obtain *similar* level of isolation of components that are exposed to attacks

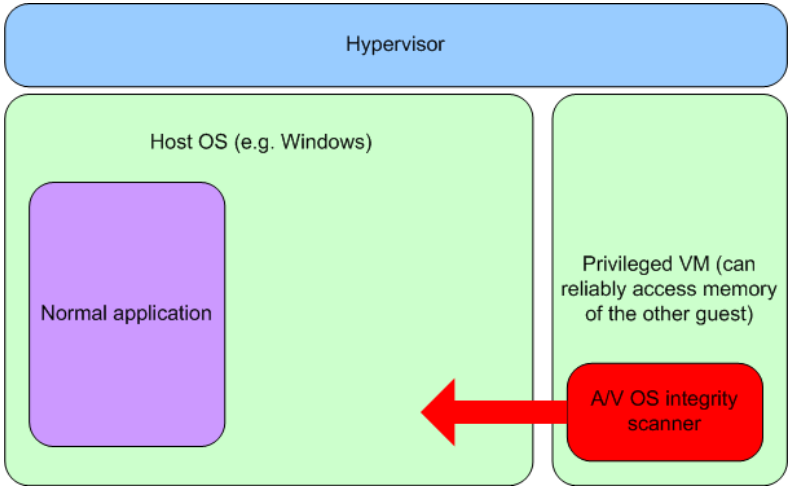


Using VMs for software isolation

- Each VM must contain a full OS
  - e.g. a virtual appliance for web browsing must contain not only Browser but also the full OS (e.g. Linux)
- There is a trend to build some OS-like services (e.g. drivers) into the VMMs which would allow for thin VMs – e.g. only the application…
  - In my opinion this is a wrong way – VMMs (hypervisors) should be kept as simple as possible
  - otherwise there would be no security benefit of using a VMM for isolation

Antivirus inside hypervisor?

- One might want to use VMMs to protect the integrity of the A/V programs
  - We should avoid building the A/V into the hypervisor -- instead it could be run in a special VM, executing in parallel:



VMM hijacking?

- Is it possible to install BP from within a VM?
- Is it possible to "escape" from the guest?
- This should not be possible!
  - At least this is what VMM-vendors would like us to believe ;)
- However…

E.g. #1: VMWare Workstation 5.5

- Reported by Tim Shelton in 2005
- CVE-2005-4459
- Description:
  - A vulnerability was identified in VMware Workstation (And others) vmnat.exe, which could be exploited by remote attackers to execute arbitrary commands. This vulnerability allows the escape from a VMware Virtual Machine into userland space and compromising the host. 'Vmnat' is unable to process specially crafted 'EPRT' and 'PORT' FTP Requests.
- Confirmed and patched by VMWare.

E.g. #2: VMWare ESX3/Workstation6

- Reported by Rafal Wojtczuk , McAfee, in September 2007
- CVE-2007-4496
- Description:
  - Vulnerability that could allow a guest operating system user with administrative privileges to cause memory corruption in a host process, and thus potentially execute arbitrary code on the host.
- Confirmed and patched by VMWare.

E.g. #3: MS Virtual Server 2005/PC 2004

- Reported by Rafal Wojtczuk, McAfee, in August 2007
- CVE-2007-0948
- Description:
  - The vulnerability is caused due to an error within certain components that communicate with the host OS and can be exploited to cause a heap-based buffer overflow.
  - Successful exploitation allows an administrative user on a guest OS to e.g. execute arbitrary code on the host OS or other guest OS's.
- Confirmed and patched by Microsoft.

E.g. #4: XEN 3

- Reported by Joris van Rantwijk in September 2007
- CVE-2007-4993
- Description:
  - When booting a guest domain, pygrub uses Python exec() statements to process untrusted data from grub.conf. By crafting a grub.conf file, the root user in a guest domain can trigger execution of arbitrary Python code in domo.
  - Reboot of the guest domain required
- Patch doesn't seem to be available
  - XEN Bugzilla says: "Fixed on 25th September by xen-unstable 15953:70bb28b."

E.g. #5: Various VMMs bugs

- A paper by Tavis Ormandy, Google:

  An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments, April 2007, CanSecWest
- Presents methodology used to find multiple bugs in several various VMMs:
  - VMWare, XEN, Bochs, MS Virtual PC, Parallels
  - mostly fuzzing-based methods used to test
    - Instruction parsing
    - I/O Device emulation
- Most of the bugs found classified as DoS

Complex VMMs → bugs?

- Complexity is the enemy of security thus VMMs should be kept as simple as possible (just like micro-kernels)
- Small VMMs/hypervisors make the code review process relatively easy
  - Sometimes we might even use the formal verification methods

Thin hypervisors

- Currently we work with Phoenix doing research on thin hypervisors
  - Phoenix works on a product called "HyperCore"
  - Phoenix is also interested in further research on Blue Pill, which is being used as a test bed for trying various ideas – e.g. nested virtualization
- Phoenix also supports The Blue Pill Project, which means that some parts of our research will be publically available (including code!)

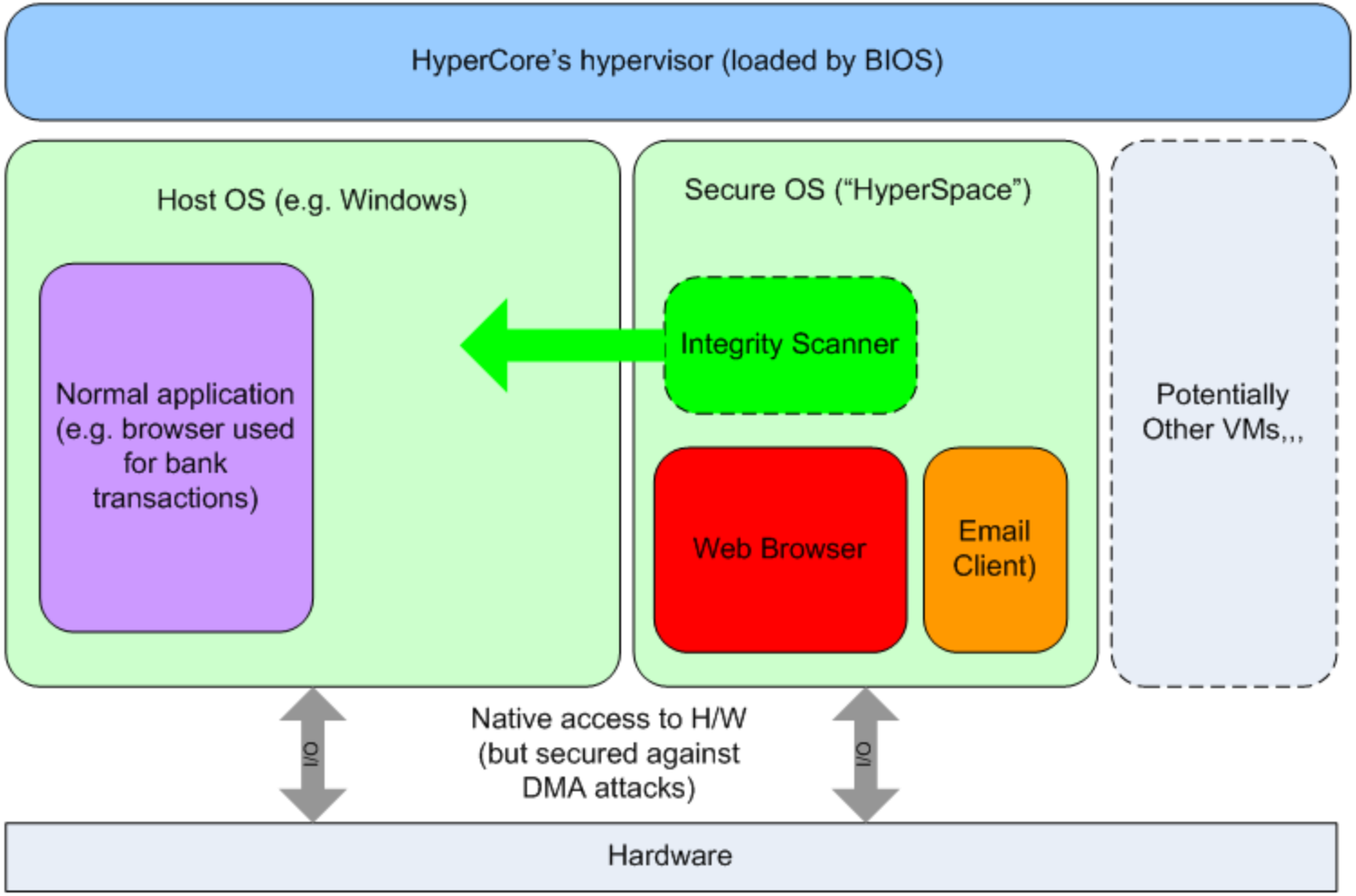
"HyperCore" a thin hypervisor by Phoenix

- Very thin hypervisor
- Use latest hardware mechanisms (e.g. NP/EMT) instead of software based virtualization (e.g. SPT)
  - Goal: reduce complexity of the VMM
- Direct I/O access for guests
  - but protect e.g. against DMA attacks
  - No device emulation!
- Initially 2 guest OS:
  - "Normal" Windows OS (e.g. Vista)
  - Custom small-footprint OS

Phoenix's HyperCore goals

- Usability:
  - The "other" OS will have some features that would be attractive for a user (This is beyond the scope of this presentation)
- Security:
  - The "other" OS will be protected from the "Windows OS". This OS will be small and secure (hardened), users will not be installing any 3rd party software
  - A user might want to use it to do banking transactions or other sensitive operations
  - We might run an A/V scanner that would check the integrity of the other OS
    - think: rootkit detector that is not prone to implementation-specific attacks!

HyperCore at a glance



Nested Virtualization

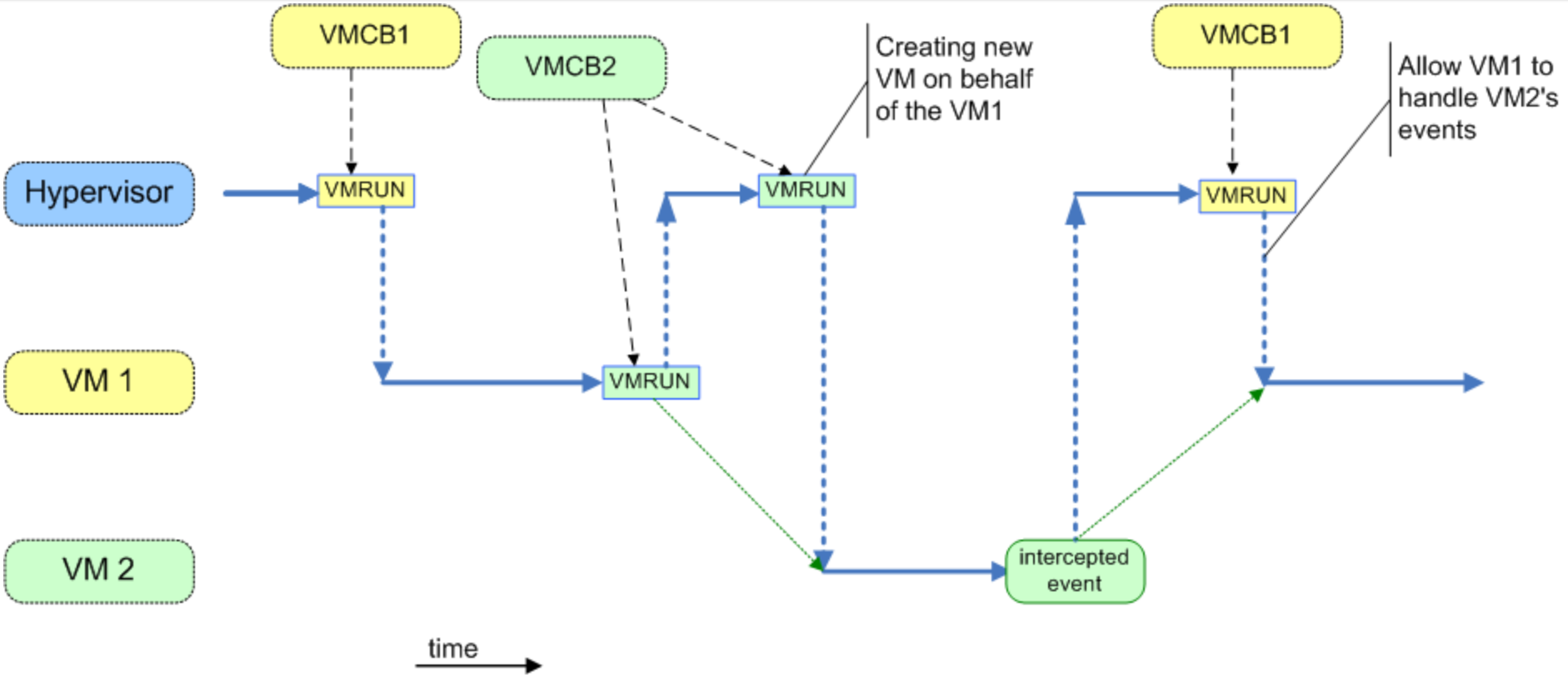
Supporting Nested VMMs

- If Blue Pill didn't support creation of nested VMMs,
- … then it would be trivial to detect it by trying to create a test virtual machine…
- Our New Blue Pill supports nested hypervisors
- In other words you can install a hypervisor as a Blue Pill's guest!
  - Think: Blue Pill inside Blue Pill :)

Supporting Nested VMMs

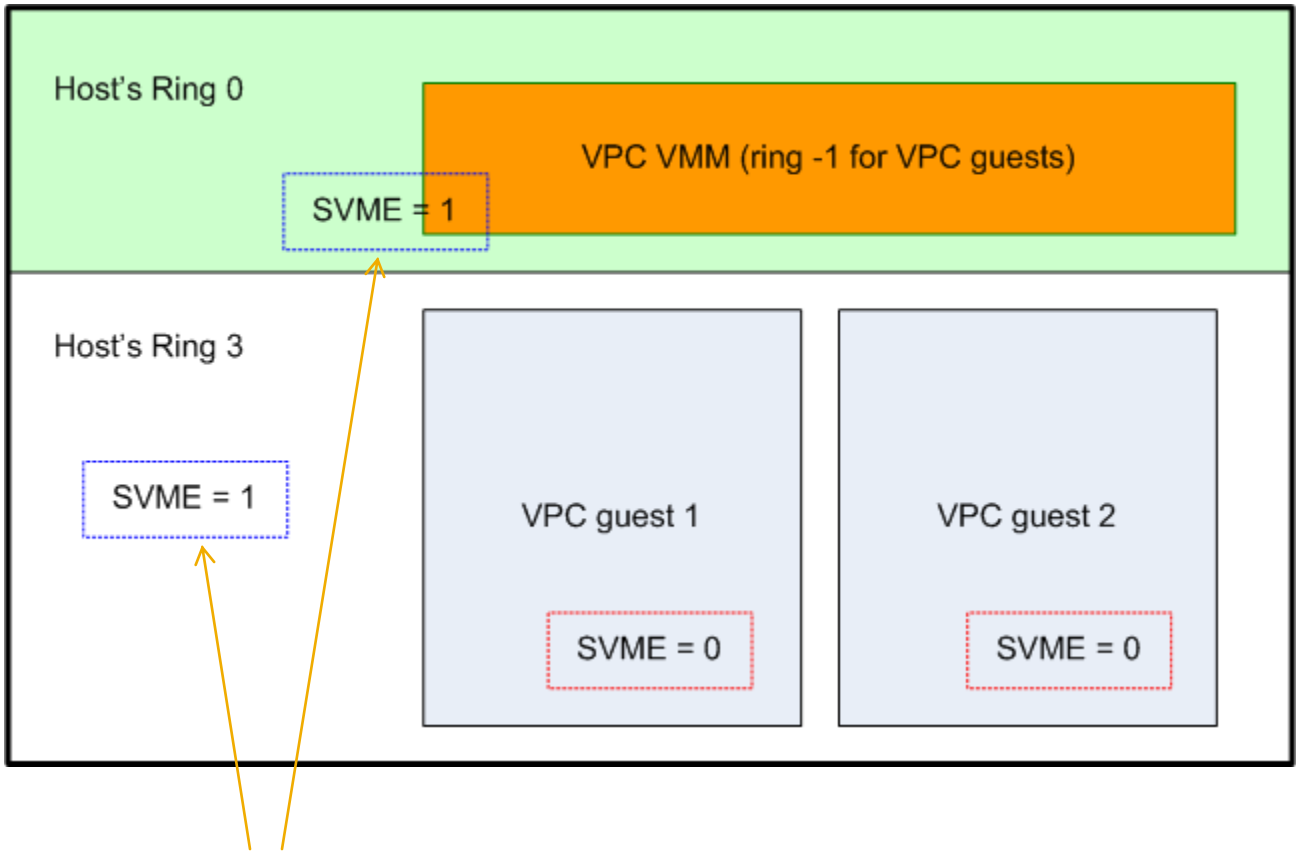
- Also, nested virtualization could be used by BP to install itself on top of other, already existing, VMMs!
- Can we implement support for nested virtualization?

Supporting nested VMMs – idea



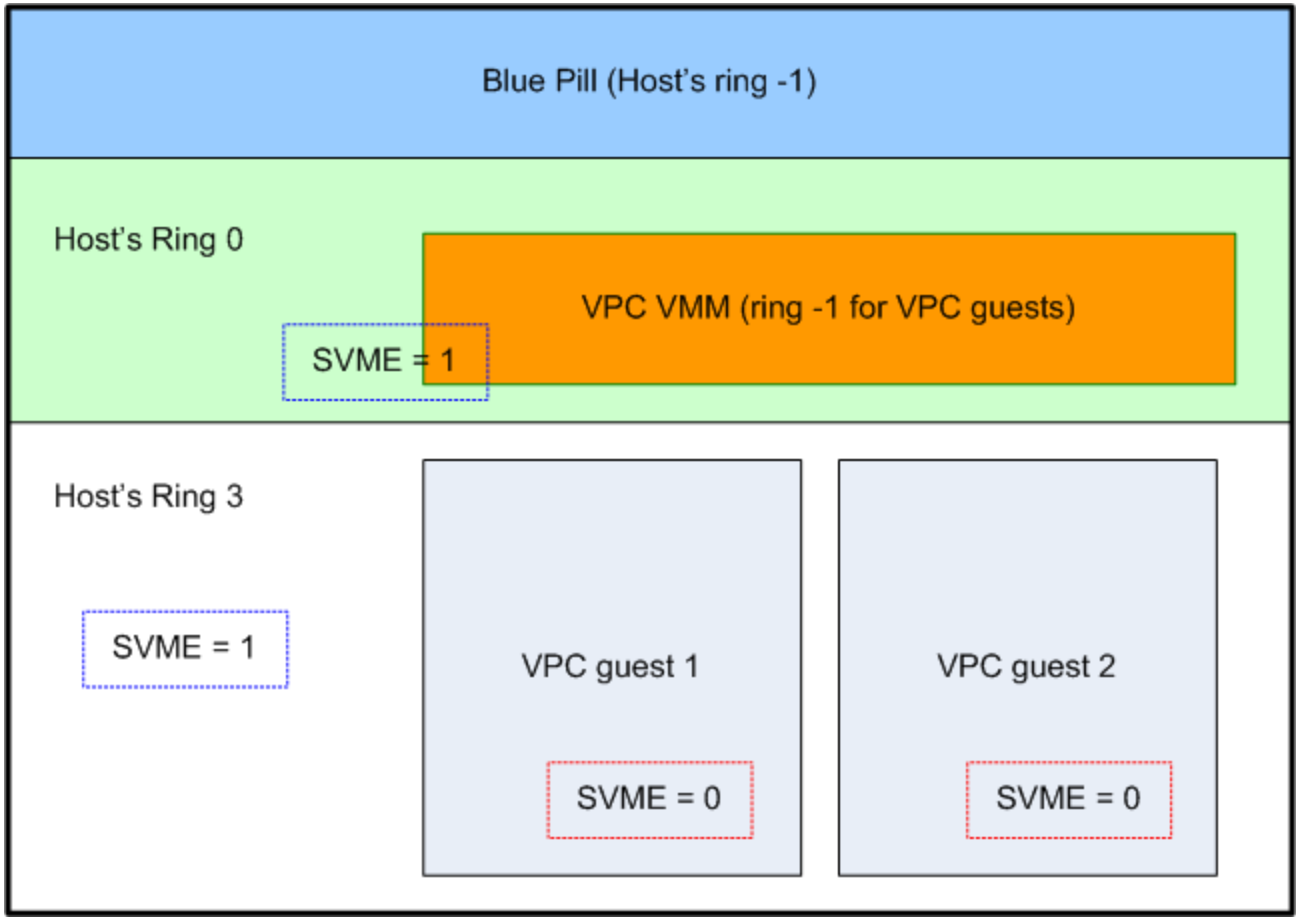
source: J. Rutkowska, Black Hat USA 2006, © COSEINC

Virtual PC 2007/ Server 2005 R2



Note that EFER.SVME=1 is set globally for the host

Bluepilling Virtual PC/Server



Windows Virtual Server 2005 R2

- When VS 2005 R2 is installed, SVME is always set! :)
- This means that we can install Blue Pill and do not care about intercepting EFER accesses anymore!
- All the detection methods discussed before (that focus on generic VMM detection), do not work now!
  - Even if we build "virtualization detector" into VPC hypervisor!
  - This is because one doesn't need to intercept anything besides VMRUN instruction on SVM
    - On Intel we need to intercept CPUID, on AMD we don't!

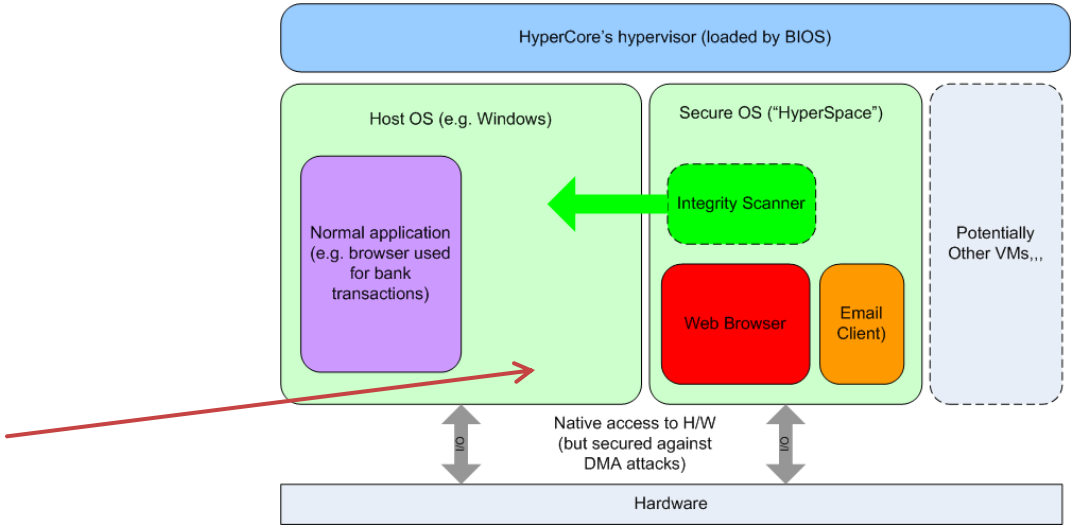
Security Implications

- Imagine a future BP that would be able to run any 3rd party hypervisor as its own guest
  - Currently we can run other BP as a guest, but still have problems with e.g. Virtual PC 2007
- We would be able to install it on top of other, existing hypervisors
  - using bugs similar to those that were presented in the previous chapter)
- Any "detection" method based on detecting virtualization will be useless by definition
  - Unless we decided to build "virtualization detectors" into each commercial hypervisor (but not always – see previous slide)
  - This however is a very unwise decision, as we should try to minimize the footprint of a hypervisor

Nested Virtualization: Positive Side

- Why would anybody be interested in nested virtualization? (besides malware authors)?
- Consider e.g. the HyperCore product:

Imagine a user would like to use some product that uses hardware virtualization? E.g. Virtual PC?

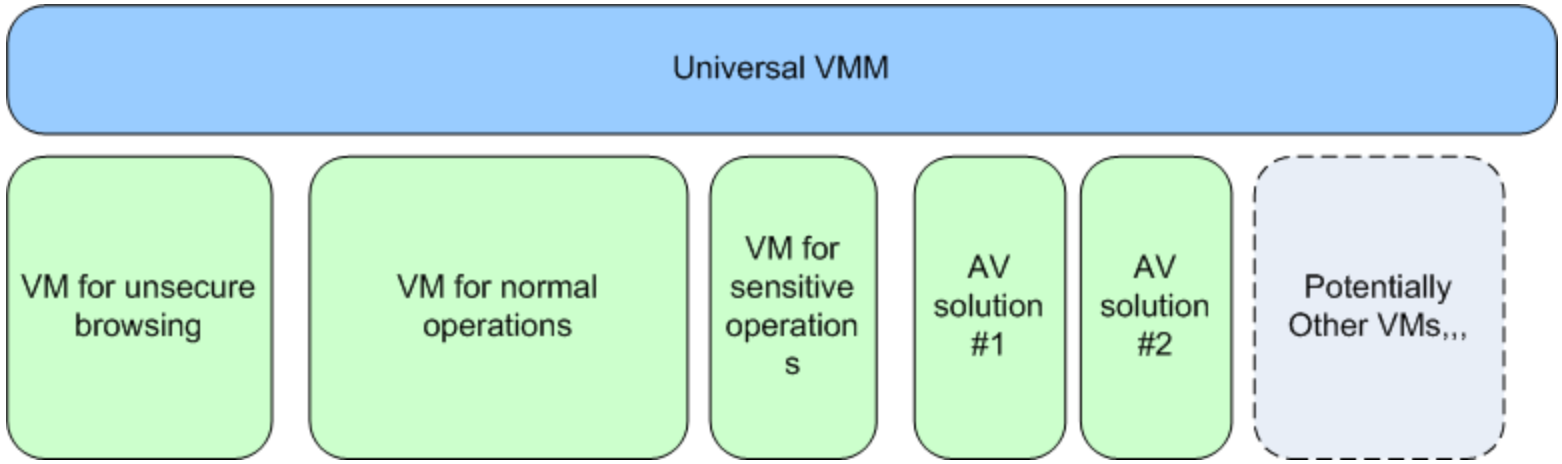


Nested Virtualization Applications

- Imagine an A/V solution that uses a VMM to create another VM where the A/V module is located (like presented before)
- If the A/V's VMMs didn't support nested virtualization, then the user will not be able to use any other virtualization solution – e.g. Virtual PC

Elegant solution?

- Obviously a more elegant solution would be to always have only one VMM in the system and to make sure that it supports all the possible Virtualization based products…



Problems with Universal VMM

- One standardized interface (VM-VMM and VM-VM) needed
- Standardized set of services provided by a VMM needed
- VMM from one provider (MS?) or many different from various vendors?
  - Who will verify whether they work with each other?
- The interface and services will become more and more complicated → VMM will get complex → difficult to verify → bugs
  - We will return to the point where we're right now, but this time with a conclusion that VMMs can't provide effective isolation

Summary

Main thesis

- Today we can't effectively prevent nor detect virtualization based rootkits
  - Several presented methods allowed only for detection of *virtualization* but not for detection of virtualization based malware
- Current VMMs do not offer perfect isolation
  - Many bugs have been found in all popular VMMs that allow to escape from VM!
  - More research needed on VMM security
- Nested virtualization is an exciting subject for research – it has both negative and positive implications

Main message

- Virtualization is a great technology but we need more research to make sure it's secure itself and also to effectively exploit all the benefits it offers to make our systems more secure!

Refferences

- J. Rutkowska, A. Tereshkin: *IsGameOver()?*, Black Hat USA, August 2007
- T. Ormandy, *An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments*, CanSecWest, April 2007
- http://bluepillproject.org

Thank you!

joanna at invisiblethingslab.com
http://invisiblethingslab.com