POACHER TURNED GATEKEEPER: LESSONS LEARNED FROM EIGHT YEARS OF BREAKING HYPERVISORS

Rafal Wojtczuk <rafal@bromium.com>

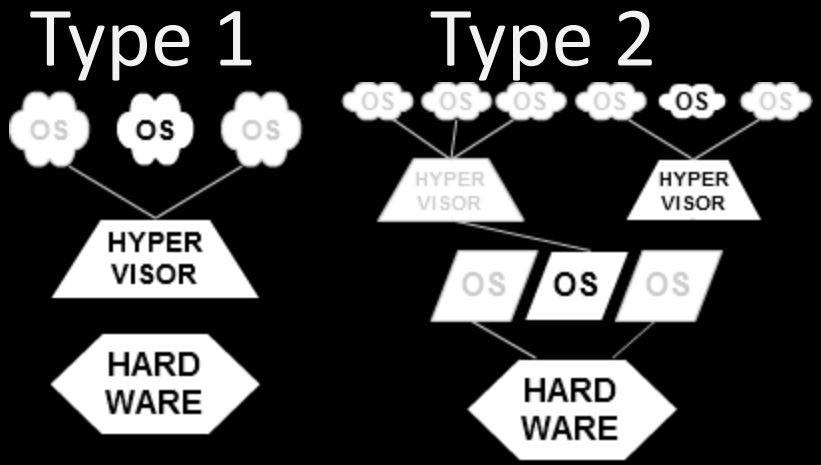
Agenda

- About the speaker
- Types of hypervisors
- Attack surface
- Examples of past and present vulnerabilities
- Mitigation techniques

Types of hypervisors

- Mainstream, popular commercial, for x86, with Windows OS VMs
  - The talk is about them
- Others
  - For embedded systems
  - Academic ones
  - Security guaranteed by formal software verification

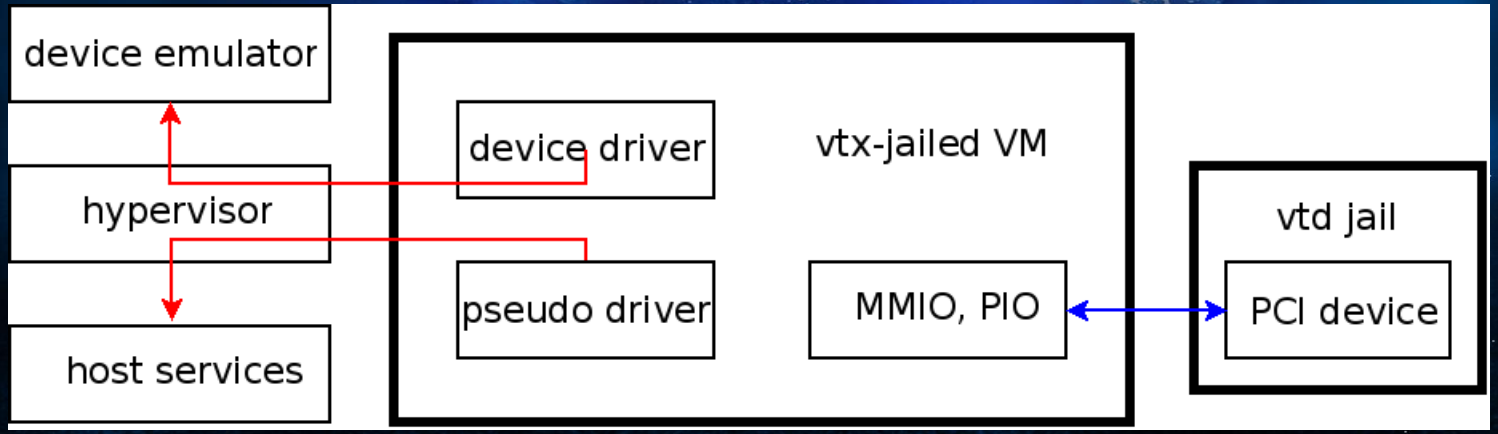
Types of hypervisors, cntd

- 

Source: http://en.wikipedia.org/wiki/Hypervisor#mediaviewer/File:Hyperviseur.png

- DeepSafe is special and different, see later

Type 1&2 attack surface



Functionality vs security

- If the goal of a virtualization system is to maximize features, the attack surface grows
- If the goal of a virtualization system is to provide security via reliable isolation, care must be taken to provide functionality in a way that does not inflate attack surface

What we compare to

- Application attack surface
  - Browsers, document editors - hopeless
- Kernel attack surface (relevant for sandbox)
  - On Windows, ca 400 syscalls, 800 win32k.sys syscalls, drivers ioctls/WDDM escapes
  - 76 CVEs for Windows kernelmode in 2013

How can we compare?

- The complexity of input is the only sensible metric – but not easy to measure quantatively
- Particularly, LOC/TCB count is close to meaningless; if you _really_ need numbers:
  - Xen-4.4.0 – ca 1.7 MLOC
  - You can strip it to 110KLOC usermode and 60KLOC ring0, still retaining useability
  - Windows7 kernel – ca 2MLOC, likely win32k.sys larger

How can we compare cntd?

- Need to rely on experience – most agree the attack surface of a well-written hypervisor is significantly smaller (see MS Drawbridge)
- One hard fact – vmexit boundary is much stronger than syscall boundary, which makes real exploitation difficult

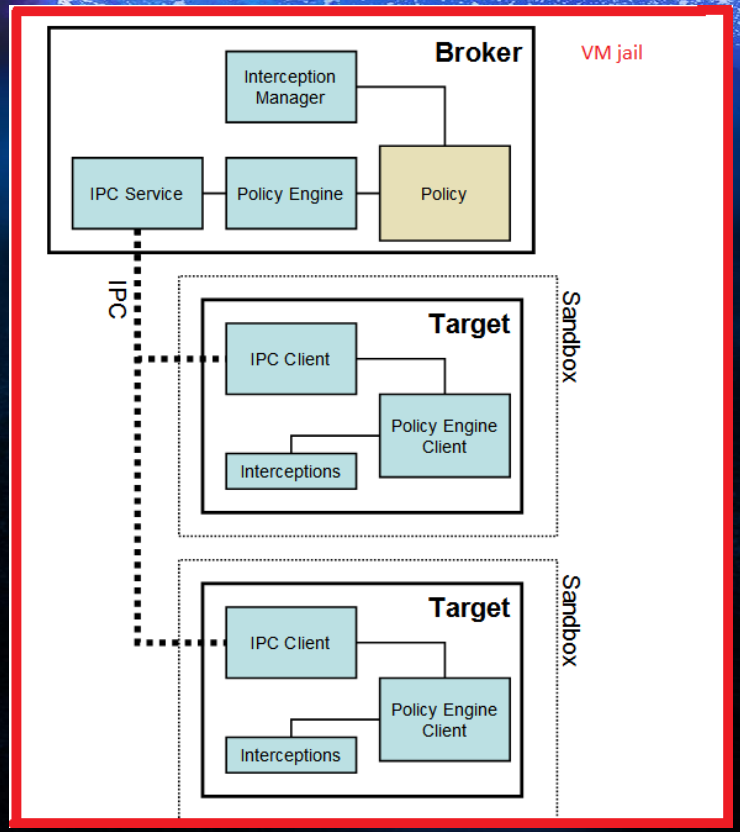
Notes on exploitability...

- ... Of memory corruption bugs
- In case of browser vulnerabilities, attacker has a lot of control over memory layout, thanks to javascript/other scripting
- In case of broker-vulnerability-based sandbox escapes, on Windows attacker knows libraries bases – no ASLR protection
- In case of kernel exploits, attacker can craft useful data structures in usermode that can be misinterpreted by the kernel, because the address space is the same (unless SMAP – but no SMAP for Windows anytime soon);
- Windows kernel hands out its memory layout for free to attacker (better on Windows8.1) [1]
- No such powerful/troublesome things against the hypervisor – usually one needs info leak + write primitive (while in the case of browser, use-after-free usually provides both instantly)
  - Cloudburst [2] is a notable, exceptional example of a reliable VM-escape memory-corruption-based exploit
  - Other exploits rely on ASLR not functional (no –fpie, non-ASLR-compatible dlls, etc)

If virtualization is another layer...

- ... And assuming that hypervisor can be attacked only after compromising the VM kernel
  - Note some products expose hypervisor services to VM unprivileged usermode
- ... And assuming there is nothing valuable in VM...
- ... And assuming hypervisor-related drivers in VM do not weaken VM kernel security...
- Then – pure gain

If virtualization is another layer...



The state of the Union

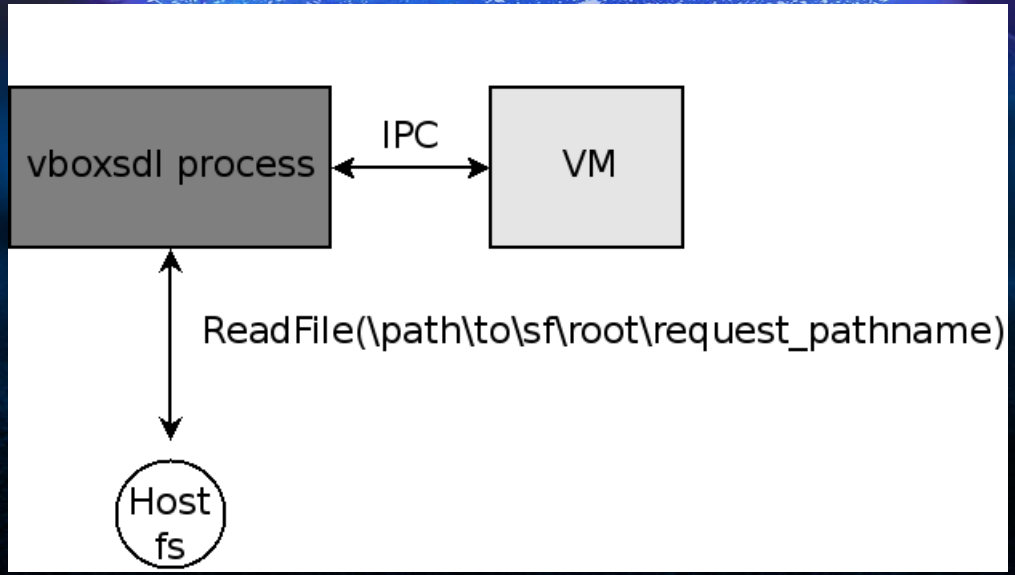
- Isolation by virtualization improves security, even with off-the-shelf products
- In order to maximize security, hypervisor-related code should be small
- Often, good design can provide functionality not sacrificing security

Case studies

New Oracle VirtualBox vulnerabilities

- 4 issues, reported by the presenter in March 2014
- Fixed in 2014 July CPU

Shared folders



Vbox sf host code is large

- Supports utf8 and unicode pathnames
  - Does not check null-termination early
- Casing corrections
- Guest can specify path delimiter; host is supposed to normalize path changing each occurence to \

S0434934

- Memory corruption in vbsfbuildfullpath()

```
397             /* Correct path delimiters */
398             if (pClient->PathDelimiter != RTPATH_DELIMITER)
399             {
400                 LogFlow(("Correct path delimiter in %ls\n", src));
401                 while (*src) // src comes from VM, not null-terminated
402                 {
403                     if (*src == pClient->PathDelimiter)
404                         *src = RTPATH_DELIMITER;
405                     src++;
406                 }
```

How to exploit for code execution

- No idea by now
- If such a vulnerability was in browser code, the usual trick would work – set up memory layout so that javascript Array object is positioned after the buffer; overwrite size field of the Array

Lesson

- Host service code should accept only narrow input – all conversions/normalization should be done in the guest (if possible).

S0434968

- Shared folders directory traversal
- Obviously, just concatenating „request_pathname” received from VM to shared folder root leads to directory traversal via „..\..\..\..\..\request_pathname” – service needs to sanitize input

S0434968, cntd

- Vbox sf sanitize algorithm:
- Split the path into components (/ or \ is the path separator)
  - Start with depth_credit=0
  - For each component do: Switch (component)
  - Case . : do nothing
  - Case ..: depth_credit-- //fail if negative
  - Default: depth_credit++;
- So „dirname\..” is ok, „dirname\..\..” Is not
- A bit untrivial? Bugs possible?

S0434968, cntd

- On posix hosts (e.g. Linux), \ is NOT a path separator
- Mkdir /mnt/vboxsf/a\a\a\a\a\a\a\a\a\a
- Access /mnt/vboxsf/a\a\a\a\a\a\a\a\a\a/../../../../../../../etc/passwd

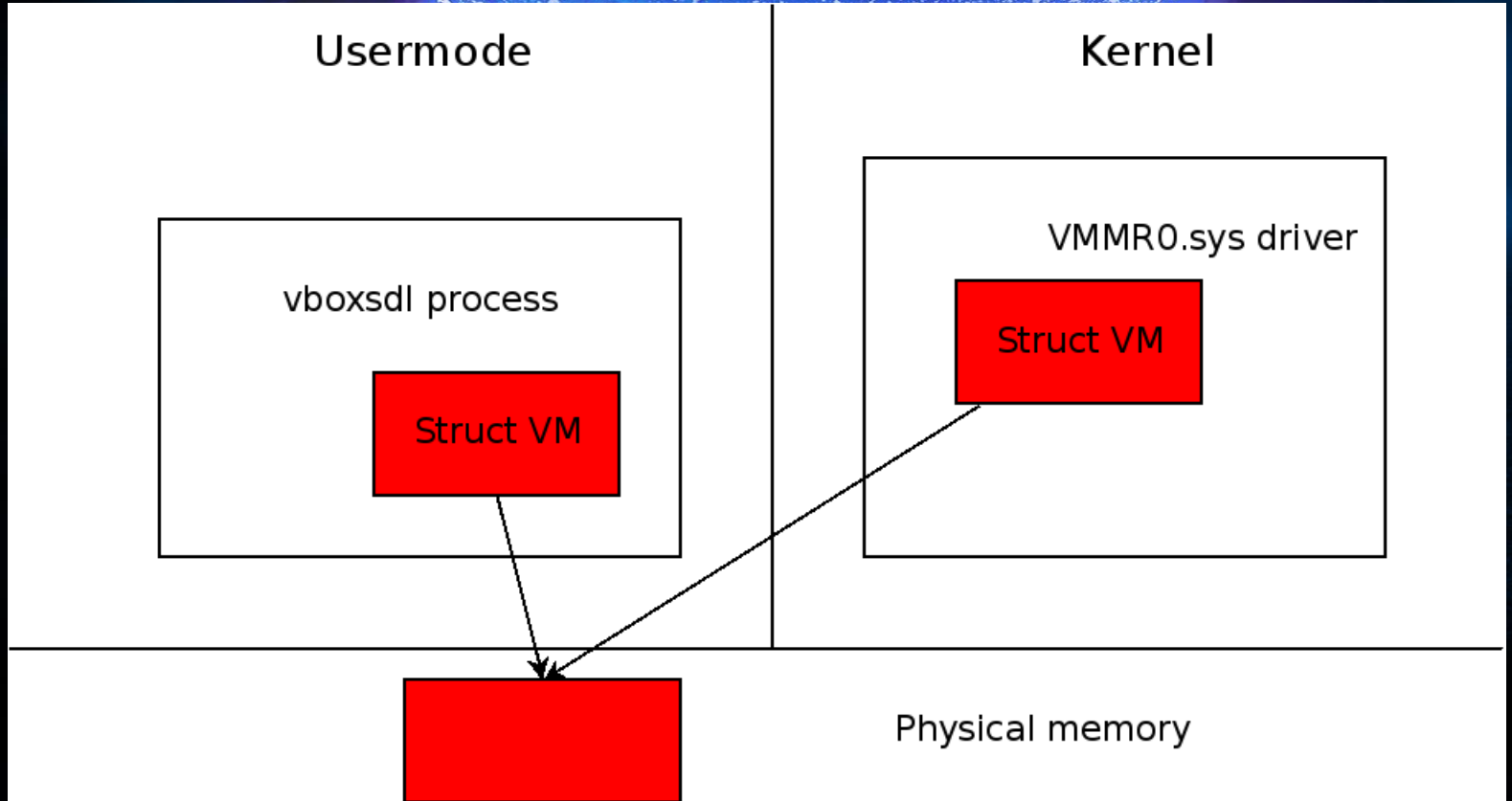
S0434968, cntd

- Lesson – same as the previous one
- Sanitization should be SIMPLE, e.g. just check for (\|/)..(\|/) In the pathname and refuse it
- Even better, on Windows prefix with \\?\
- On Linux, use chroot
- Beware - portable code can be full of surprises

S0434952

- Data leak in shared folders code
- When VM requests to read 1024 bytes from zero-length file, host returns 1024 bytes-long uninitialized buffer (plus information that 0 bytes have been read)
- Leaks contents of uninitialized malloced buffer

S0434947:Frontend to kernel escalation on the host



CVE-2007-5497

- Integer overflow in libext2fs
- Xen's Pygrub runs in [privileged] dom0, uses libext2fs to extract kernel image from VM's filesystem – bad!
- Pvgrub runs in VM, does the kernel image extraction within VM - good
- Lesson – again, offload to VM as much as possible

CVE-2011-1751

- Use-after-free in qemu/KVM (a talk at BH11)
- Triggered by emulation of PCI hotplugging, by writing to emulated chipset registers
- Any generic mitigation? E.g. can we deny all PCI config access to VMs?

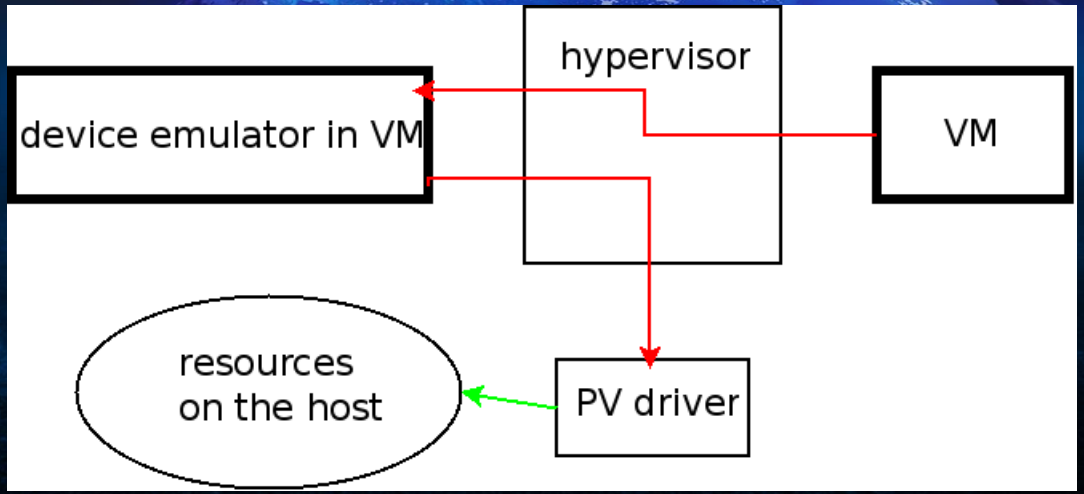
Delusional boot

- Start VM with all PCI config space access granted, let it boot (no interaction with malicious input)
- Save VM, restore VM
- Deny all PCI config space access to the restored VM; let it interact with attacker

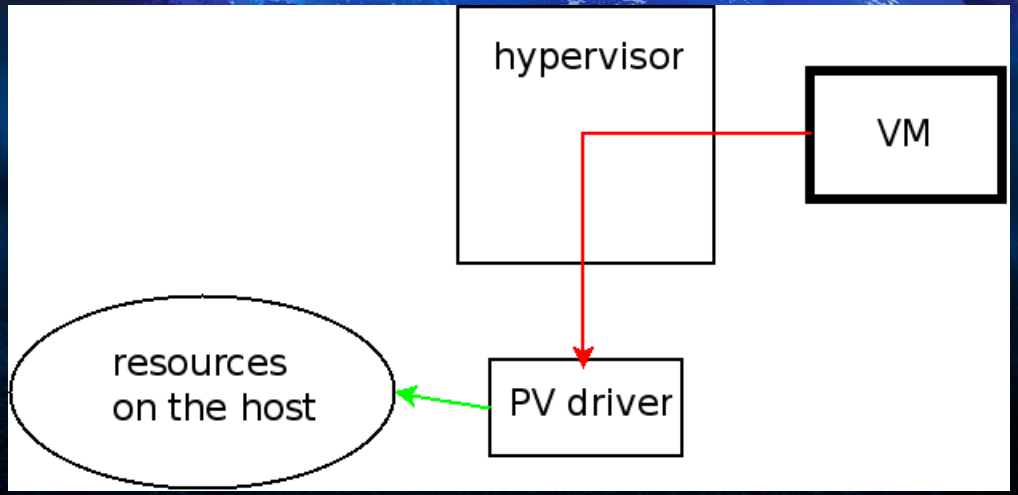
CVE-2012-0029

- Heap-based buffer overflow in the process_tx_desc function in the e1000 qemu emulation

What to do with device emulation: stub domain



What to do with device emulation: guest PV driver



CVE-2007-0069

- Windows Kernel TCP/IP/IGMPv3 and MLDv2 Vulnerability, remote code execution
- Hey, this is not a bug in virtualization software?

Service VMs

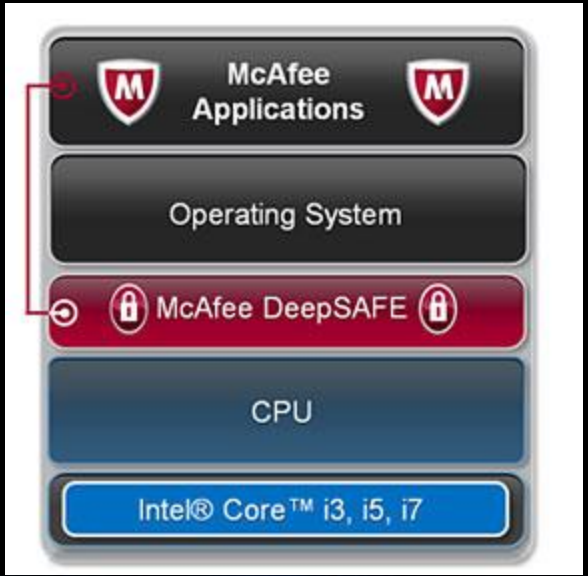
- Move some privileged code (e.g. NIC/WLAN driver, networking stack, dhcp client) to a dedicated VM
- Need to give the service VM direct access to the relevant hardware via PCI passthrough
  - QubesOS, XenClient XT: network VM by default

Host as a service VM

- Make the type 2 host a giant service VM [3]
- Need to protect VMs against the host usermode (particularly device model)
- Quite some issues – e.g. need to protect hypervisor against hardware-based attacks originating in the host; protect HID

DeepSafe architecture

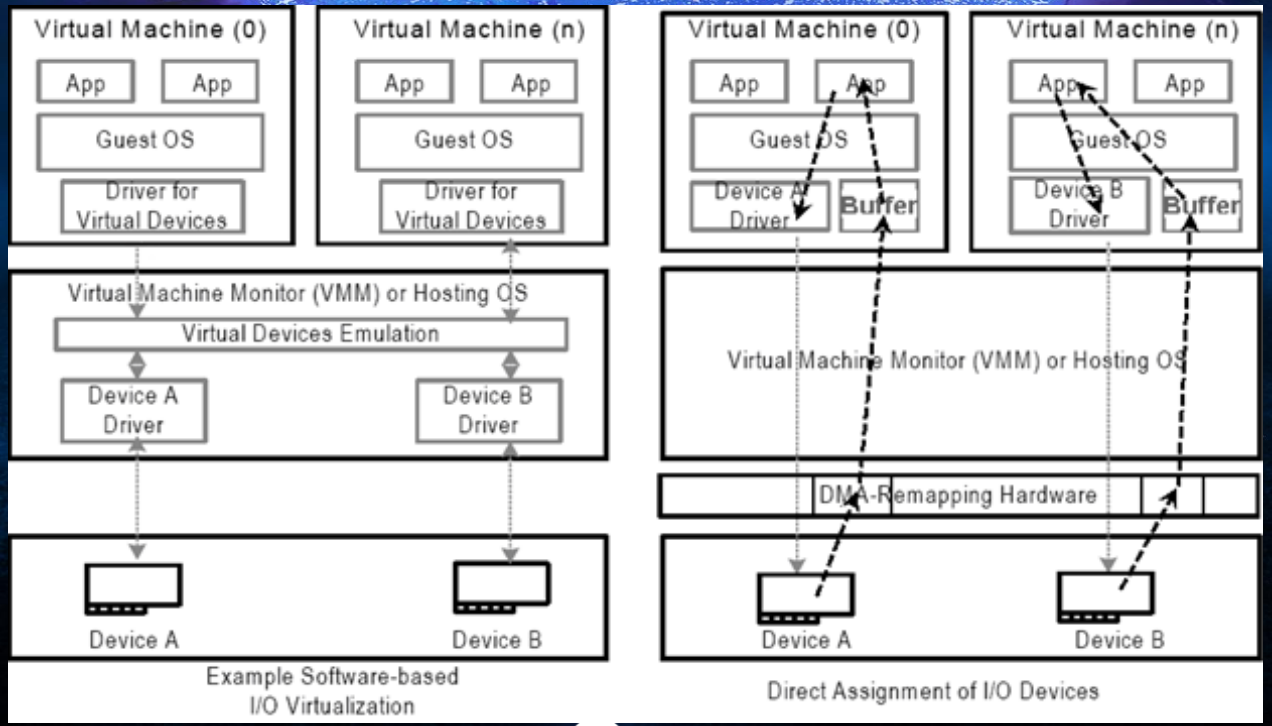
- Stress how different it is from usual type 1&2



DeepSafe architecture, cntd

- When CPU runs a Deepsafe VM, EPT protects hypervisor memory from being accessed - good
- How about memory accesses done by PCI devices (DMA) ?

DMA attacks, VTd



Source: https://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices

Does DeepSafe use VTd?

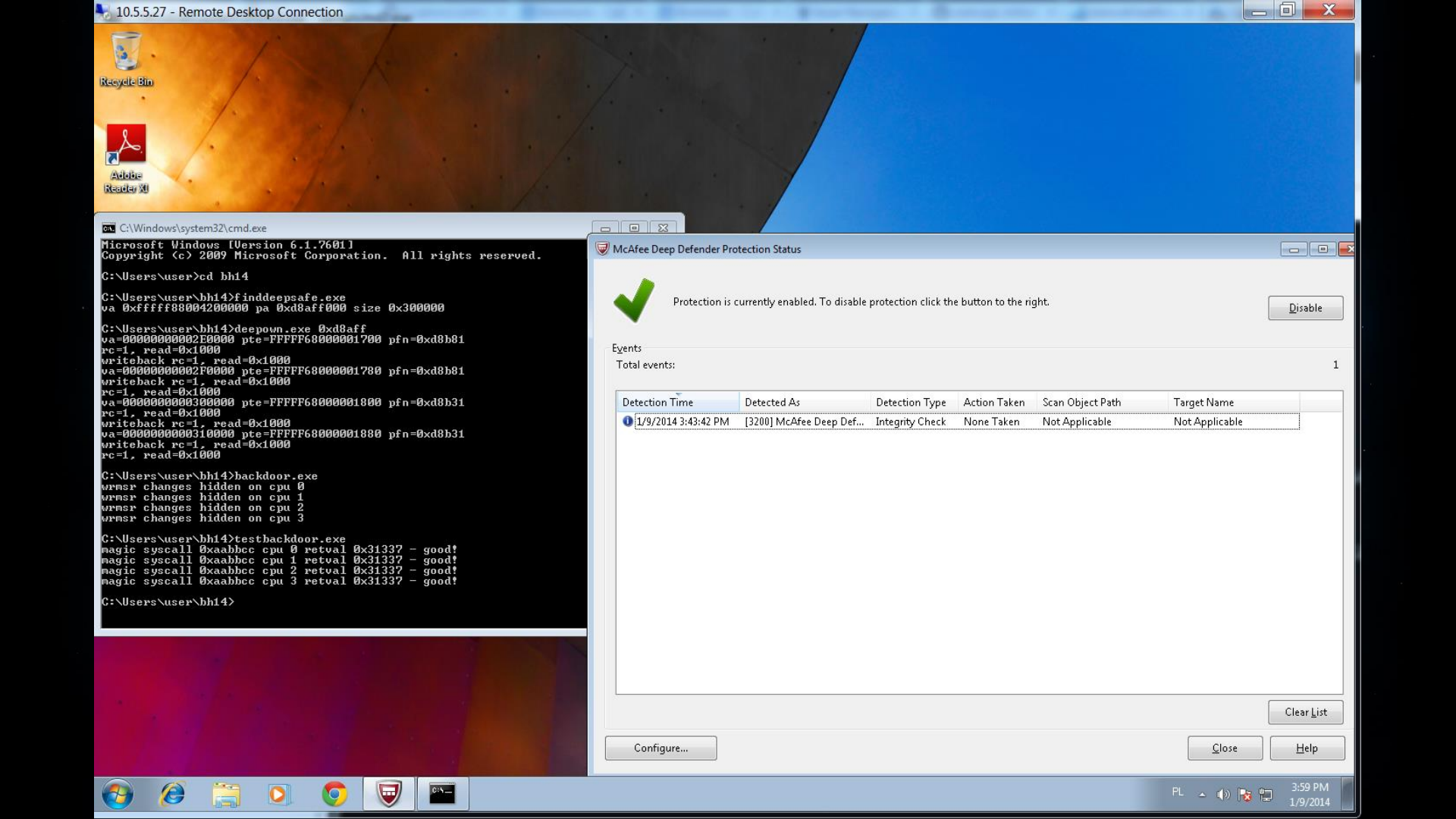
- No (tested version 1.6.0, latest available)
- Despite DMA attacks against Xen hypervisor has been demonstrated at BH2008
- Despite well-known discussions about the necessity of it [4]
- Impact – compromise of DeepSafe integrity

How to do arbitrary DMA (Windows)

- Achieve kernel privileges
- Allocate a page at virtual address V
- Change PTE of V so that it points to physical address P
- CreateFile(... FILE_FLAG_NO_BUFFERING ...)
- ReadFile/WriteFile(..., V,...) will do DMA to P
- One catch – not this straightforward with BitLocker

Is Deepsafe hypervisor hijack useful?

- We could disable it...
- ... Too much work...
- ... Why would an attacker get rid of such privileged code he/she already controls ?
- We can use it to hide some activities from OS/Patchguard, e.g. LSTAR MSR change – results in rootkit functionality


10.5.5.27 - Remote Desktop Connection
Recycle Bin
Adobe Reader XI
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\user>cd bh14
C:\Users\user\bh14>finddeepsafe.exe
va 0xfffff88004200000 pa 0xd8aff000 size 0x300000
C:\Users\user\bh14>deepown.exe 0xd8aff
va=000000000002E0000 pte=FFFFF68000001700 pfn=0xd8b81
rc=1, read=0x1000
writeback rc=1, read=0x1000
va=000000000002F0000 pte=FFFFF68000001780 pfn=0xd8b81
writeback rc=1, read=0x1000
rc=1, read=0x1000
va=0000000000300000 pte=FFFFF68000001800 pfn=0xd8b31
rc=1, read=0x1000
writeback rc=1, read=0x1000
va=0000000000310000 pte=FFFFF68000001880 pfn=0xd8b31
writeback rc=1, read=0x1000
rc=1, read=0x1000
C:\Users\user\bh14>backdoor.exe
wrmsr changes hidden on cpu 0
wrmsr changes hidden on cpu 1
wrmsr changes hidden on cpu 2
wrmsr changes hidden on cpu 3
C:\Users\user\bh14>testbackdoor.exe
magic syscall 0xaabbcc cpu 0 retval 0x31337 - good!
magic syscall 0xaabbcc cpu 1 retval 0x31337 - good!
magic syscall 0xaabbcc cpu 2 retval 0x31337 - good!
magic syscall 0xaabbcc cpu 3 retval 0x31337 - good!
C:\Users\user\bh14>
McAfee Deep Defender Protection Status
Protection is currently enabled. To disable protection click the button to the right.
Disable
Events
Total events:
1
Detection Time
Detected As
Detection Type
Action Taken
Scan Object Path
Target Name
1/9/2014 3:43:42 PM
[3200] McAfee Deep Def...
Integrity Check
None Taken
Not Applicable
Not Applicable
Clear List
Configure...
Close
Help
PL
3:59 PM
1/9/2014

There are more DeepSafe concerns

- Filter drivers in the host may provide effective backdooring capability
- Compromised host kernel can overwrite crucial usermode memory
- How secure is mfeib.sys launch, on reboot/S3 resume?
- No trusted UI domain
- Host can mess with PCI config, SMM, BIOS, PCI devices firmware

Summary

- Hypervisors have non-negligible attack surface
- Despite the above, they are still useful to isolate even less secure operation systems
- There are generic methods to reduce attack surface of a hypervisor

Questions?

Bibliography

- [1] Alex Ionescu, „KASLR Bypass Mitigations in Windows 8.1”, http://www.alex-ionescu.com/?p=82
- [2] Kostya Kortchinsky, „CLOUDBURST: A VMware Guest to Host Escape Story”, BHUSA09
- [3] Ian Pratt, „µXen”, http://www-archive.xenproject.org/xensummit/xs12na_talks/T6.html
- [4] Joanna Rutkowska, „Thoughts on DeepSafe”, http://theinvisiblethings.blogspot.co.uk/2012/01/thoughts-on-deepsafe.html